

AutoID

July 7, 2025



AUTOID

Version: 2022.11.8

Copyright

All product technical documentation is
Ping Identity Corporation
1001 17th Street, Suite 100
Denver, CO 80202
U.S.A.

Refer to <https://docs.pingidentity.com> for the most current product documentation.

Trademark

Ping Identity, the Ping Identity logo, PingAccess, PingFederate, PingID, PingDirectory, PingDataGovernance, PingIntelligence, and PingOne are registered trademarks of Ping Identity Corporation ("Ping Identity"). All other trademarks or registered trademarks are the property of their respective owners.

Disclaimer

The information provided in Ping Identity product documentation is provided "as is" without warranty of any kind. Ping Identity disclaims all warranties, either express or implied, including the warranties of merchantability and fitness for a particular purpose. In no event shall Ping Identity or its suppliers be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages, even if Ping Identity or its suppliers have been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of liability for consequential or incidental damages so the foregoing limitation may not apply.

Table of Contents

Release notes	4
What's New	7
Before You Install	9
Changelog.	11
Doc Updates	15
Security Advisories.	17
Release Levels	17
Get Support.	19
Deployment planning	20
Architecture in brief	22
Security controls overview	24
Topology planning	25
Checklist.	28
Installation	30
Deployment architectures	33
Install a single node	35
Install a single node air-gapped	49
Install a multinode deployment	58
Install a multinode air-gapped	71
Upgrade.	80
Deployment ports	101
Appendix: vars.yml	102
Administrator tasks	105
Self service	109
Manage identities	111
Data preparation.	116
Deployment tasks	117
Entity definitions.	152
Data sources	153
Attribute mappings	161
Analytics settings	162
Analytics pipeline	168
Admin tasks.	184
Server maintenance	189
Roles.	199

Troubleshooting	210
Users tasks	213
Features.	216
PingOne Autonomous Identity user types	216
The PingOne Autonomous Identity UI.	218
Supervisor tasks	229
App owner tasks	233
Entitlement owner tasks	237
Autonomous Identity API	241
About the API.	247
Obtain the API key	248
API service	254
Authentication	255
SSO.	259
Config	261
Report	271
Companyview	272
Userdetails	278
Singleview with app	294
Access control	296
Applications.	302
Entitlements	308
Assignments	316
Rules.	320
Filters	326
Roles.	328
Ingest	336
Jobs	346
Datasink.	353

Release notes

PingOne® Autonomous Identity is an entitlements and roles analytics system that lets you fully manage your company's access to your data.

These release notes are written for anyone using the PingOne Autonomous Identity 2022.11.8 release. Read these notes before you install PingOne Autonomous Identity software, especially for production deployments.



What's New

Discover new features.



Before You Install

Check prerequisites.



Changelog

Check key fixes, known issues, deprecated, and removed items.



Documentation

Track doc changes.



Interface Stability

Learn about the Release Levels, upgrades, and stability levels.



Getting Support

Get support and training.

What's new in PingOne Autonomous Identity

PingOne Autonomous Identity 2022.11.8 is the latest patch release containing a collection of bug and security fixes released as part of our commitment to our customers.

For general information on ForgeRock's maintenance and patch releases, see [Maintenance and Patch availability policy](#).

You can deploy PingOne Autonomous Identity 2022.11.8 as an initial deployment or upgrade it from an existing 2022.11.x deployment.

PingOne Autonomous Identity 2022.11.8

- **Security and bug fixes.** PingOne Autonomous Identity introduces security and bug fixes. For specific information on the fixes, contact ForgeRock.

PingOne Autonomous Identity 2022.11.7

- **Security and bug fixes.** PingOne Autonomous Identity introduces security and bug fixes. For specific information on the fixes, contact ForgeRock.
- **Opensearch 1.3.13.** PingOne Autonomous Identity now requires Opensearch 1.3.13.
 - For all new deployments from 2022.11.x to the latest version 2022.11.7 using `deployer-pro`, update Opensearch from version 1.3.9 to 1.3.13 prior to running your upgrade. For more information, refer to [Opensearch 1.3.13](#).
 - For deployments upgraded from 2022.8.x using the `deployer` installer, PingOne Autonomous Identity upgrades to version Opensearch 1.3.13 automatically.

PingOne Autonomous Identity 2022.11.6

PingOne Autonomous Identity introduces updated container images.

PingOne Autonomous Identity 2022.11.5

- **Security and bug fixes.** PingOne Autonomous Identity introduces security and bug fixes. For specific information on the fixes, contact ForgeRock.

- **Upgraded components.** PingOne Autonomous Identity requires the following third-party software dependency:
 - Python 3.10.9

PingOne Autonomous Identity 2022.11.4

- **Security and bug fixes.** PingOne Autonomous Identity introduces security and bug fixes. For specific information on the fixes, contact ForgeRock.

PingOne Autonomous Identity 2022.11.3

- **New property to use MongoDB with LDAP.** PingOne Autonomous Identity has a new `vars.yml` property, `mongo_ldap=false`, which when set to `true`, lets PingOne Autonomous Identity authenticate with MongoDB, configured with LDAP.
- **New assignments endpoint.** PingOne Autonomous Identity now provides an endpoint to support the extraction of assignments. Refer to [Assignments](#).
- ForgeRock discovered a regression in 2022.11.3. Refer to [Known issues in 2022.11.3](#).

PingOne Autonomous Identity 2022.11.2

- **Security and bug fixes.** PingOne Autonomous Identity introduces security and bug fixes. For specific information on the fixes, contact ForgeRock.

PingOne Autonomous Identity 2022.11.1

- **Security and bug fixes.** PingOne Autonomous Identity introduces security and bug fixes. For specific information on the fixes, contact ForgeRock.

PingOne Autonomous Identity 2022.11.0

- **Upgraded deployer script.** PingOne Autonomous Identity introduces a new deployer script, *Deployer Pro*. The Deployer Pro script downloads and installs PingOne Autonomous Identity within your environment. However, customers must now install the third-party software dependencies required for PingOne Autonomous Identity prior to running Deployer Pro *on new deployments only*. The deployer pro lets customers install and configure those dependencies best suited for their network environment as well as their scale, performance, high availability (HA), and disaster recovery (DR) requirements.

Note

Customers with existing 2021.8.7 deployments can upgrade their PingOne Autonomous Identity systems to 2022.11, while maintaining their existing third-party software components used in their 2021.8.7 deployments.

- **Upgraded components.** PingOne Autonomous Identity requires the following third-party software dependencies:
 - Opensearch and Opensearch Dashboards 1.3.6
 - Apache Cassandra 4
 - Apache MongoDB 4.4

- Apache Spark 3.3
 - Apache Livy with log4j2 support
 - Python 3.8
 - OpenJDK 11
- **Internal Security Fixes.** ForgeRock has made a number of important security fixes and updates.

Before you start

PingOne Autonomous Identity server software requires the following hardware, storage, and operating system requirements to run in your production environment. PingOne Autonomous Identity's flexible architecture runs in a variety of network environments: on-prem, cloud, multi-cloud, and hybrid.



Important

All production systems differ in many ways. Please discuss with your ForgeRock Professional Services, installers, or partner representatives about your environment specifics.

ForgeRock Google Cloud registry key

You deploy PingOne Autonomous Identity using a Docker image that pulls other dependent images from the ForgeRock Google Cloud Registry repository and installs the components on a target node.

For specific instructions on obtaining the registry key, refer to [How To Configure Service Credentials \(Push Auth, Docker\) in Backstage](#).

Hardware and memory requirements

PingOne Autonomous Identity has a number of components that include application, data, and analytics servers, which are all included in the Docker images. The minimum hardware and memory requirements for a single-node target and a separate deployer machine are as follows:

Hardware and memory requirements

Vendor	Versions
Deployer Node	32 GB RAM, 8 CPU
Analytics (Target) Node	64 GB RAM, 16 CPU

Storage requirements

PingOne Autonomous Identity has a number of components that include application, data, and analytics servers, which are included in the Docker images. The minimum storage requirements for a single-node deployment are as follows:

PingOne Autonomous Identity requires the following minimum storage requirements:

Storage requirements

Type	Size
Data Storage	500 GB (minimum), 1 TB (production)

Operating systems requirements

PingOne Autonomous Identity is supported on the following operating system:

Operating System Requirements

Vendor	Versions ^[1]
CentOS Stream	8.0
Redhat Enterprise Linux	8.0

Cloud services requirements

PingOne Autonomous Identity has been successfully deployed on the following cloud services:

Cloud Services Requirements

Vendor	Versions
Google Cloud Platform (GCP)	Latest
Amazon Web Services (AWS) standard Elastic File System (EFS) shared drive	Latest

Java requirements

PingOne Autonomous Identity software supports the following Java version:

Java requirements

Vendor	Versions
OpenJDK	11.0.16

Third-party software

PingOne Autonomous Identity uses the following third-party software in the deployment.

IMPORTANT:

If your existing deployment uses the deployer-pro installer (2022.11.0 and later), you can upgrade these third-party dependencies to these versions.

If your existing deployment uses the deployer installer (pre-2022.11.0 or earlier), you do not need to pre-install or upgrade these components in your environment. The PingOne Autonomous Identity deployer installs or upgrades these dependencies.

Third-party software

Component	Version	Usage
Python	3.10.9	Deployer and Deployer Pro scripts
Docker CE	20.10.17	Container cluster management
Apache Cassandra	4.0.8	Database for all PingOne Autonomous Identity services
MongoDB	4.4.19	Database for all PingOne Autonomous Identity services. If using MongoDB with LDAP, set the <code>mongo_ldap</code> property to <code>true</code> in the <code>vars.yml</code> file.
Apache Spark	3.3.2 with Hadoop 3	Cluster to run PingOne Autonomous Identity analytics
Apache Livy	Updated to work with Spark 3.3.2	REST interface to Spark master to run PingOne Autonomous Identity analytics
Opensearch/Opensearch Dashboards	1.3.13	Distributed, open source search engine and visualization tool for all data types.

Supported browsers

PingOne Autonomous Identity supports the following browsers:

Supported browsers

Vendor	Versions
Google Chrome	version 85.0.4183.121 and higher
Mozilla Firefox	version 86.0.1 and higher

1. For PingOne Autonomous Identity 2022.8.x systems that use CentOS/Redhat Enterprise Linux 7.0 and upgrade to PingOne Autonomous Identity 2022.11.x, PingOne Autonomous Identity continues to run on CentOS/Redhat Enterprise Linux 7.0. For new PingOne Autonomous Identity 2022.11.x installations, use CentOS/Redhat Enterprise Linux 8.0.

Changelog

ForgeRock continuously provides updates to PingOne Autonomous Identity to introduce new features, fix known bugs and address security issues.

Key fixes

2022.11.8

This release contains a collection of security and bug fixes.

2022.11.7

This release contains a collection of security and bug fixes. Additionally, PingOne Autonomous Identity requires Opensearch 1.3.13 in this release.

2022.11.6

This release contains the latest container images.

2022.11.5

This release contains a collection of security and bug fixes.

2022.11.4

This release contains a collection of security and bug fixes.

2022.11.3

The following bugs were fixed in this release as well as other security fixes:

- [AUTOID-3174](#): Need an assignments API
- [AUTOID-3362](#): Allow customer to change timeout for API container when run Opensearch query

2022.11.2

The following bugs were fixed in this release:

- [AUTOID-3329](#): Misspelled http header for kibana conf
- [AUTOID-3331](#): Elasticsearch keystore and truststore password

2022.11.1

This release contains a collection of important security fixes.

2022.11.0

The following bugs were fixed in this release as well as other security fixes:

- [AUTOID-2766](#): Analytics results show inconsistent results

- [AUTOID-2864](#): Not able to delete data sources in AutoID
- [AUTOID-2894](#): Support for updating all certificates in AutoID
- [AUTOID-3130](#): Upgrade Spark to 3.3
- [AUTOID-3135](#): Upgrade Open Distro to Opensearch
- [AUTOID-3145](#): Upgrade Python to 3.8
- [AUTOID-3160](#): Upgrade OpenJDK to 11

Known Issues

2022.11.8

There are no known issues in this release.

2022.11.5

There are no known issues in this release.

2022.11.4

There are no known issues in this release.

2022.11.3

• Discovered regression

PingOne Autonomous Identity 2022.11.3 was originally released on 04-11-2023.

We discovered a regression where Apache Livy has log4j1 binaries included with the deployer. If you installed 2022.11.3 *before* 04/13/2023, run the steps below to upgrade log4j1 to log4j2.

If you installed 2022.11.3 *after* 04/13/2023, the binaries are updated, and you do not need to upgrade log4j1 binaries.

Update log4j1 to log4j2

1. Stop the Apache Livy server:

```
~/livy/bin/livy-server stop
```

2. Back up your old log4j and related jar files:

```
cd ~/livy/jars
mv log4j-1.2.16.jar ~/log4j-1.2.16.jar.bkp
mv slf4j-log4j12-1.6.1.jar ~/slf4j-log4j12-1.6.1.jar.bkp
mv slf4j-reload4j-1.7.36.jar ~/slf4j-reload4j-1.7.36.jar.bkp
mv slf4j-api-1.7.25.jar ~/slf4j-api-1.7.25.jar.bkp
```

3. Replace with log4j2 jar and its bridge jars:

```
cd ~/livy/jars
wget https://repo1.maven.org/maven2/org/apache/logging/log4j/log4j-1.2-api/2.18.0/log4j-1.2-api-2.18.0.jar
wget https://repo1.maven.org/maven2/org/apache/logging/log4j/log4j-core/2.18.0/log4j-core-2.18.0.jar
wget https://repo1.maven.org/maven2/org/apache/logging/log4j/log4j-slf4j-impl/2.18.0/log4j-slf4j-impl-2.18.0.jar
wget https://repo1.maven.org/maven2/org/apache/logging/log4j/log4j-api/2.18.0/log4j-api-2.18.0.jar
wget https://repo1.maven.org/maven2/org/slf4j/slf4j-api/1.7.36/slf4j-api-1.7.36.jar
```

4. Under the `conf` folder, create a `log4j2.properties` file:

```
cd ~/livy/conf
vi log4j2.properties
```

5. In your `log4j2.properties` file, adjust the log level and related configuration suited for your requirements:

```
status = info
name= RollingFileLogConfigDemo
# Log files location
property.basePath = ./logs
# RollingFileAppender name, pattern, path and rollover policy
appender.rolling.type = RollingFile
appender.rolling.name = fileLogger
appender.rolling.fileName= ${basePath}/autoid.log
appender.rolling.filePattern= ${basePath}/autoid_%d{yyyyMMdd}.log.gz
appender.rolling.layout.type = PatternLayout
appender.rolling.layout.pattern = %d{yyyy-MM-dd HH:mm:ss.SSS} %level [%t] [%l] - %msg%n
appender.rolling.policies.type = Policies
# RollingFileAppender rotation policy
appender.rolling.policies.size.type = SizeBasedTriggeringPolicy
appender.rolling.policies.size.size = 10MB
appender.rolling.policies.time.type = TimeBasedTriggeringPolicy
appender.rolling.policies.time.interval = 1
appender.rolling.policies.time.modulate = true
appender.rolling.strategy.type = DefaultRolloverStrategy
appender.rolling.strategy.delete.type = Delete
appender.rolling.strategy.delete.basePath = ${basePath}
appender.rolling.strategy.delete.maxDepth = 10
appender.rolling.strategy.delete.ifLastModified.type = IfLastModified
# Delete all files older than 30 days
appender.rolling.strategy.delete.ifLastModified.age = 30d
# Configure root logger
rootLogger.level = info
rootLogger.appenderRef.rolling.ref = fileLogger
log4j1.compatibility = true
```

6. Restart Apache Livy:

```
cd ~/livy/  
./bin/livy-server start
```

7. Check that Apache Livy is up and running. You can access a log on an analytics jobs. Specific PingOne Autonomous Identity logs are at `~/livy/logs/autoid.log`.

2022.11.2

There are no known issues in this release.

2022.11.1

There are no known issues in this release.

2022.11.0

There is a known issue with RHEL8/CentOS Stream 8 when Docker swarm overlay network configuration breaks when the outside network maximum transmission unit (mtu) is smaller than the default value. The `mtu` is the maximum size of the packet that can be transmitted from a network interface.

Refer to <https://github.com/moby/libnetwork/issues/2661> and <https://github.com/moby/moby/pull/43197>.

When deploying a multinode configuration on RHEL 8/CentOS Stream 8, run the following steps:

1. Check mtu for docker0 and eth0 using `ifconfig | grep mtu`.
2. Set the docker0 mtu value to be equal to `eth0` using `sudo ifconfig eth0 mtu 1500`. Make sure to set the command on all nodes and also after each virtual machine reboot.

Deprecated

2022.11.0–2022.11.8

No functionality has been deprecated in these releases.

Removed

2022.11.0–2022.11.8

No functionality has been removed in these releases.

Documentation updates

The following table tracks changes to the documentation following the release of PingOne Autonomous Identity 2022.11.8:

Documentation Change Log

Date	Description
2023-12-12	Initial release of PingOne Autonomous Identity 2022.11.8.
2023-10-23	Initial release of PingOne Autonomous Identity 2022.11.7. <ul style="list-style-type: none"> Updated the Opensearch version to 1.3.13. Refer to Third-party software.
2023-09-11	Added a step to install the Python wheel file after upgrade with deployer pro. Refer to Upgrade from 2022.11.x to 2022.11.6 (Non Air-Gap) using deployer pro or Upgrade from 2022.11.x to 2022.11.6 Air-Gapped using deployer pro .
2023-09-05	Initial release of PingOne Autonomous Identity 2022.11.6. <ul style="list-style-type: none"> Conflated the fixes, known issues, deprecated, and removed sections into one changelog file. Refer to Changelog.
2023-08-15	Added a line that you need to update your third-party software packages to the supported versions prior to upgrading PingOne Autonomous Identity. Refer to Upgrade from Autonomous Identity 2022.11.x to 2022.11.5 using deployer pro .
2023-07-20	Initial release of PingOne Autonomous Identity 2022.11.5. <ul style="list-style-type: none"> Updated the Python version. Refer to Third-party software. Moved the Ports section from the release notes to the installation section.
2023-06-05	Initial release of PingOne Autonomous Identity 2022.11.4.
2023-04-13	Added a known issue. Refer to Known issues in 2022.11.3 .
2023-04-12	Added a section on updating the domain and namespace in existing deployments. Refer to Customize the Domain and Namespace (New deployments) .
2023-04-11	<ul style="list-style-type: none"> Initial release of PingOne Autonomous Identity 2022.11.3. Added a section to change the default timeout (30 ms) for the API's Elasticsearch client request timeout. Refer to Change the API's Elasticsearch client request timeout. Added the Assignments endpoint to the API. Refer to Assignments.
2023-02-24	Initial release of PingOne Autonomous Identity 2022.11.2.
2023-01-09	Initial release of PingOne Autonomous Identity 2022.11.1.

Date	Description
2022-12-08	<ul style="list-style-type: none">Removed the refresh-company-view command as it is no longer required in the Analytics pipeline process.Updated the steps to install the Python egg file. Refer to Install PingOne Autonomous Identity.Removed sections pertaining to third-party component backup-restore and import-export data in the Server Maintenance.
2022-11-28	Added a section to change the MongoDB password post-deployment. Refer to Change the MongoDB password post-deployment .
2022-11-15	Initial release of PingOne Autonomous Identity 2022.11.0.

Security advisories

ForgeRock issues security advisories in collaboration with our customers and the open source community to address any security vulnerabilities transparently and rapidly. ForgeRock's security advisory policy governs the process on how security issues are submitted, received, and evaluated as well as the timeline for the issuance of security advisories and patches.

Release levels and interface stability

ForgeRock defines Major, Minor, and Patch product release levels. The release level is reflected in the version number. The release level tells you what sort of compatibility changes to expect.



Important

PingOne Autonomous Identity uses a different version numbering system from other ForgeRock products. The version number use the following format: **Major.Minor.Patch**, where *Major* is the year of the release, *Minor* is the month of the release, *Patch* is the number beginning with 0, and increases for each patch release. Thus, for this release of PingOne Autonomous Identity, the version number is **2022.11.8**.

Release Level Definitions

Release Label	Version Numbers	Characteristics
Major	Version: x[.0.0]	<ul style="list-style-type: none">Bring major new features, minor features, and bug fixesCan include changes even to Stable interfacesMajor indicates the year of the release, for example, 2021

Release Label	Version Numbers	Characteristics
Minor	Version: x.y[.0]	<ul style="list-style-type: none"> • Bring minor features, and bug fixes • Can include backwards-compatible changes to Stable interfaces in the same Major release, and incompatible changes to Evolving interfaces • Minor indicates the month of the release, for example, 8 for August
Patch	Version: x.y.z	<ul style="list-style-type: none"> • Bring bug fixes • Are intended to be fully compatible with previous versions from the same Minor release • Patch starts with 0 and increases for each bug fix release

Upgrade and Patching

ForgeRock plans to introduce quarterly upgrades and patches for PingOne Autonomous Identity as a service to our customers. PingOne Autonomous Identity's architecture supports seamless rolling upgrades to simplify the process.

The following are some general points about upgrades and patches:

- Upgrades and patches are implemented using a simple swap of the underlying container. The operation is zero down-time as long as the cluster has a redundant instance of the microservice.
- Patching does not require schema changes.

PingOne Autonomous Identity schema changes are additive and backward-compatible. This means that during a zero-downtime upgrade, older versions of the container can still write to the new version of the schema. Also, newer versions of the container may alter the tables in a way that preserves the semantics of the previous columns.

- If an upgrade requires a downgrade due to some issue, the downgrade will not restore the previous schema.

More information about upgrading, refer to [Upgrade PingOne Autonomous Identity](#).

ForgeRock Product Stability Labels

ForgeRock products support many features, protocols, APIs, GUIs, and command-line interfaces. Some of these are standard and very stable. Others offer new functionality that is continuing to evolve.

ForgeRock acknowledges that you invest in these features and interfaces, and therefore must know when and how ForgeRock expects them to change. For that reason, ForgeRock defines stability labels and uses these definitions in ForgeRock products.

ForgeRock Stability Label Definitions

Stability Label	Definition
Stable	This documented feature or interface is expected to undergo backwards-compatible changes only for major releases. Changes may be announced at least one minor release before they take effect.
Evolving	<p>This documented feature or interface is continuing to evolve and so is expected to change, potentially in backwards-incompatible ways even in a minor release. Changes are documented at the time of product release.</p> <p>While new protocols and APIs are still in the process of standardization, they are Evolving. This applies for example to recent Internet-Draft implementations, and also to newly developed functionality.</p>
Legacy	<p>This feature or interface has been replaced with an improved version, and is no longer receiving development effort from ForgeRock.</p> <p>You should migrate to the newer version, however the existing functionality will remain. Legacy features or interfaces will be marked as <i>Deprecated</i> if they are scheduled to be removed from the product.</p>
Deprecated	This feature or interface is deprecated and likely to be removed in a future release. For previously stable features or interfaces, the change was likely announced in a previous release. Deprecated features or interfaces will be removed from ForgeRock products.
Removed	This feature or interface was deprecated in a previous release and has now been removed from the product.
Technology Preview	<p>Technology previews provide access to new features that are considered as new technology that is not yet supported. Technology preview features may be functionally incomplete and the function as implemented is subject to change without notice. DO NOT DEPLOY A TECHNOLOGY PREVIEW INTO A PRODUCTION ENVIRONMENT.</p> <p>Customers are encouraged to test drive the technology preview features in a non-production environment and are welcome to make comments and suggestions about the features in the associated forums.</p> <p>ForgeRock does not guarantee that a technology preview feature will be present in future releases, the final complete version of the feature is liable to change between preview and the final version. Once a technology preview moves into the completed version, said feature will become part of the ForgeRock platform. Technology previews are provided on an “AS-IS” basis for evaluation purposes only and ForgeRock accepts no liability or obligations for the use thereof.</p>
Internal/Undocumented	Internal and undocumented features or interfaces can change without notice. If you depend on one of these features or interfaces, contact ForgeRock support or email info@forgerock.com to discuss your needs.

Getting support

ForgeRock provides support services, professional services, training through ForgeRock University, and partner services to assist you in setting up and maintaining your deployments. For a general overview of these services, refer to <https://www.forgerock.com>.

ForgeRock has staff members around the globe who support our international customers and partners. For details on ForgeRock's support offering, including support plans and service level agreements (SLAs), visit <https://www.forgerock.com/support>.

ForgeRock publishes comprehensive documentation online:

- The ForgeRock [Knowledge Base](#) offers a large and increasing number of up-to-date, practical articles that help you deploy and manage ForgeRock software.

While many articles are visible to community members, ForgeRock customers have access to much more, including advanced information for customers using ForgeRock software in a mission-critical capacity.

- ForgeRock product documentation, such as this document, aims to be technically accurate and complete with respect to the software documented. It is visible to everyone and covers all product features and examples of how to use them.

Deployment planning

Use this chapter to plan your PingOne Autonomous Identity deployment.



Important

This chapter is for deployers, technical consultants, and administrators who are familiar with PingOne Autonomous Identity and are responsible for architecting a production deployment.



Architecture in brief

Learn about the PingOne Autonomous Identity architecture.



Security controls

Learn about the PingOne Autonomous Identity security controls.



Topology planning

Review topology sizing considerations.



Deployment checklist

Use the checklist.

For installation instructions, refer to the [PingOne Autonomous Identity installation guide](#).

For component versions, refer to the [PingOne Autonomous Identity Release notes](#).

Architecture in brief

PingOne Autonomous Identity has a powerful and flexible architecture that lets you deploy PingOne Autonomous Identity in any number of ways: single-node or multi-node configurations across on-prem, cloud, hybrid, or multi-cloud environments. The PingOne Autonomous Identity architecture has a simple three-layer conceptual model as follows:

- **Application Layer.** PingOne Autonomous Identity implements a flexible Docker Swarm microservices architecture, where multiple applications run together in containers. The microservices component provides flexible configuration and end-user interaction to the deployment. The microservices components are the following:
 - **PingOne Autonomous Identity UI.** PingOne Autonomous Identity supports a dynamic UI that displays the entitlements, confidence scores, and recommendations.
 - **PingOne Autonomous Identity API.** PingOne Autonomous Identity provides an API that can access endpoints using REST. This allows easy scripting and programming for your system.
 - **Backend Repository.** The backend repository stores PingOne Autonomous Identity user information.
 - **Nginx.** Nginx is a popular HTTP server and reverse proxy for routing HTTPS traffic.
 - **Apache Livy.** PingOne Autonomous Identity supports Apache Livy to provide a RESTful interface to Apache Spark.
 - **Java API Service.** PingOne Autonomous Identity supports a private Java API Service (JAS) for a RESTful interface to the Cassandra or MongoDB database.
- **Data Layer.** PingOne Autonomous Identity supports Apache Cassandra NoSQL and MongoDB databases to serve predictions, confidence scores, and prediction data to the end user. Apache Cassandra is a distributed and linearly scalable database with no single point of failure. MongoDB is a schema-free, distributed database that uses JSON-like documents as data objects. Java API Service (JAS) provides a RESTful interface to the databases.

PingOne Autonomous Identity also implements Opensearch and Opensearch Dashboards to improve search performance for its entitlement data. Opensearch supports scalable writes and reads. Opensearch Dashboards provides a useful visualization tool for your Opensearch backend.
- **Analytics and Administration Layer.** PingOne Autonomous Identity uses a multi-source Apache Spark analytics engine to generate the predictions and confidence scores. Apache Spark is a distributed, cluster-computing framework for AI machine learning for large datasets. PingOne Autonomous Identity runs the analytics jobs directly from the Spark main over Apache Livy REST interface.

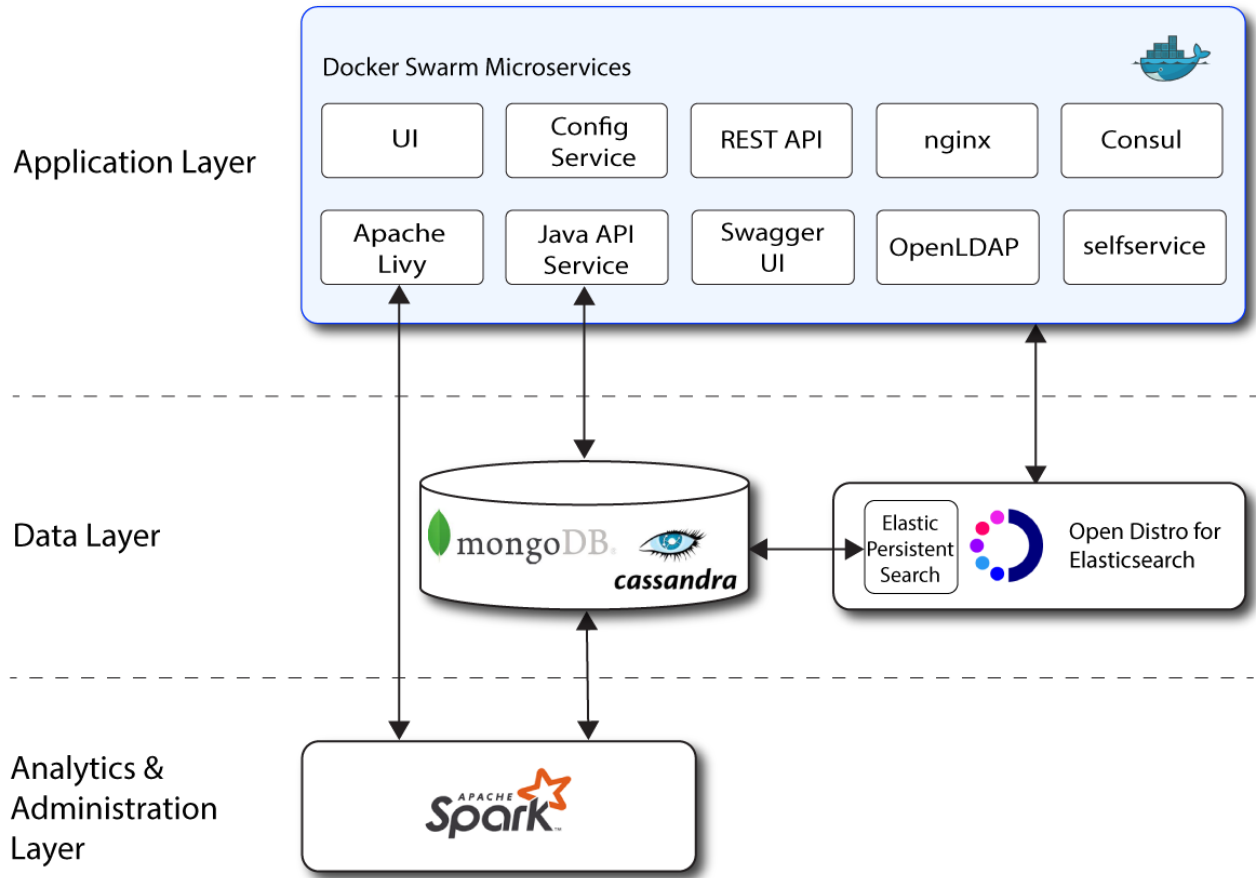


Figure 1. A Simple Conceptual Image of the PingOne Autonomous Identity Architecture

Security controls overview

PingOne Autonomous Identity uses a number of security protocols as summarized below.

Security Controls Summary

Security	Description
Encryption Protocol	TLSv1.2
Encryption: External Data in Transit	All data in transit from PingOne Autonomous Identity to the outside world is encrypted. SSL certificates must be configured with the load balancer. By default, PingOne Autonomous Identity configures self-signed certificates used by Nginx. Customers can also use their own certificates during deployment.

Encryption: Internal Data in Transit	<p>Within the PingOne Autonomous Identity secure server network, most data in transit between the PingOne Autonomous Identity services is encrypted, but not all. The exception is any non-encrypted communication between PingOne Autonomous Identity servers. You can protect this communication via network firewalls.</p> <p>It is also recommended to disable access on network and firewall ports for services like Spark and Livy that are meant for internal access only. The rest of the services are SSL/TLS-protected including all Nginx-protected services, MongoDB, Cassandra, and Opensearch nodes.</p>
Encryption: Data at Rest	<p>MongoDB is not encrypted natively in PingOne Autonomous Identity, but can be encrypted via third-party disk encryption or using the MongoDB enterprise version. If encryption at rest is required, please confirm with the MongoDB vendors how this is handled in existing MongoDB clusters.</p> <p>Likewise, Cassandra is not natively encrypted, but can be supported through its enterprise versions.</p>
Authentication	<p>PingOne Autonomous Identity uses various authentication methods within its systems, such as the following:</p> <ul style="list-style-type: none"> • Local Authentication. User credentials (user/groups) are stored in Opensearch. Users can log in with a username and password. This is mostly used for development or QA scenarios. • OpenID Connect. PingOne Autonomous Identity can use Single Sign-On (SSO) by integrating SSO providers like Azure AD and ForgeRock® Access Management (AM). <p>The API service and Java API Service (JAS) are protected by authentication handlers that support token-based access. JAS also supports certificate-based authentication, which is only used by internal services that require elevated access.</p>

Topology planning

Based on existing production deployments, we have determined a suggested number of servers and settings based on the numbers of identities, entitlements, assignments, and applications. These suggested number of servers and settings are general guidelines for your particular deployment requirements. Each deployment is unique, and requires review prior to implementation.

For a description of possible production deployments, refer to [Deployment Architecture](#) in the [PingOne Autonomous Identity Installation Guide](#).

Data sizing

ForgeRock has determined general categories of dataset sizes based on a company's total number of identities, entitlements, assignments, and applications.

A key determining factor for sizing is the number of applications. If a company has identities, entitlements, and assignments in the Medium range, but if applications are close to 150, then the deployment could be sized for large datasets.

Data Set Ranges

	Small	Medium	Large	Extra Large
Total Identities	<10K	10K-50K	50K-100K	100K-1M
Total Entitlements	<10K	10K-50K	50K-100K	100K+
Total Assignments	<1M	1M-6M	6M-15M	15M+
Total Applications	<50	50-100	100-150	150+

Suggested number of servers

Based on dataset sizing, the following chart shows the number of servers for each deployment. These numbers were derived from existing customer deployments and internal testing setups.



Important

These numbers are not hard-and-fast rules, but are only presented as starting points for deployment planning purposes. Each deployment is unique and requires proper review prior to implementation.

Suggested Number of Servers

	Small	Medium	Large	Extra Large
Deployer	1 ^[1]	1	1	1
Docker	1	2 (manager; worker)	2 (manager; worker)	Custom ^[2]
Database	1	2 (2 seeds)	3 (3 seeds)	Custom ^[2]
Analytics	1	3 (master; 2 workers)	5 (master; 4 workers)	Custom ^[2]
Opensearch	1	2 (master; worker)	3 (master; 2 workers)	Custom ^[2]
Opensearch Dashboards	1	1	1	1

[1] This figure assumes that you have a separate deployer machine from the target machine for single-node deployments. You can also run the deployer on the target machine for a single-node deployment. For multi-node deployments, we recommend running the deployer on a dedicated low-spec box.

[2] For extra-large deployments, server requirements will need to be specifically determined.

Suggested analytics settings

Analytics settings require proper sizing for optimal machine-learning performance.

The following chart shows the analytics settings that are for each deployment size. The numbers were derived from customer deployments and internal testing setups.



Important

These numbers are not hard-and-fast rules, but are only presented as starting points for deployment planning purposes. Each deployment is unique and requires proper review prior to implementation.

Suggested Analytics Settings

	Small	Medium	Large	Extra Large
Driver Memory (GB)	2	10	50	Custom ^[1]
Driver Cores	3	3	12	Custom ^[1]
Executor Memory (GB)	3	3-6	12	Custom ^[1]
Executor Cores	6	6	6	Custom ^[1]
Elastic Heap Size ^[2]	2	4-8	8	Custom ^[1]

[1] For extra-large deployments, server requirements will need to be specifically customized.

[2] Set in the `vars.yml` file.

Production technical recommendations

PingOne Autonomous Identity 2022.11.8 has the following technical specifications for production deployments:

Production Technical Specifications

	Deployer	Database	Database	Analytics	Opensearch
Installed Components	Docker	Cassandra	MongoDB	Spark (Spark Master)/Apache Livy	Opensearch
OS	CentOS	CentOS	CentOS	CentOS	CentOS
Number of Servers	Refer to Suggested number of servers	Refer to Suggested number of servers	Refer to Suggested number of servers	Refer to Suggested number of servers	Refer to Suggested number of servers
RAM (GB)	4-32	32	32	64-128	64

CPUs	2-4	8	8	16	16
Non-OS Disk Space (GB)[1]	32	1000	1000	1000	1000
NFS Shared Mount	N/A	N/A	N/A	1 TB NFS mount shared across all Docker Swarm nodes (if more than 1 node is provisioned) at location separate from the non-OS disk space requirement. For example, <code>/data</code> or <code>shared</code> .	N/A
Networking	nginx: 443 Docker Manager: 2377 (TCP) Docker Swarm: 7946, 4789 (UDP) 7946, 2049 (TCP)	Client Protocol Port: 9042 Cassandra Nodes: 7000	Client Protocol Port: 27017 MongoDB Nodes: 30994	Spark Master: 7077 Spark Workers: Randomly assigned ports	Opensearch: 9300 Opensearch (REST): 9200 Opensearch Dashboards: 5601
Licensing	N/A using Docker CE free version	N/A	N/A	N/A	N/A
Software Version	Docker: 20.10.17	Cassandra: 4.0.6	MongoDB: 4.4	Spark: 3.3.2 Apache Livy: 0.8.0-incubating	Opensearch/ Opensearch Dashboards 1.3.13
Component Reference	Refer to below.[2]	Refer to below.[3]	Refer to below.[4]	Refer to below.[5]	Refer to below.[6]

[1] At root directory "/"

[2] <https://docs.docker.com/ee/ucp/admin/install/system-requirements/>

[3] <https://docs.datastax.com/en/dse-planning/doc/planning/planningHardware.html>

[4] <http://cassandra.apache.org/doc/latest/operating/hardware.html>

[4] <http://www.mongodb.com>

[5] <https://spark.apache.org/docs/latest/security.html#configuring-ports-for-network-security>

[6] <https://Opensearch.org/>

Deployment checklist

Use the following checklist to ensure key considerations are covered for your 2022.11.8 deployment:

Deployment Checklist

Check	Requirement	Details
Access		
<input type="checkbox"/>	Remote Access	The PingOne Autonomous Identity Team is a global team. To support the needs of client teams, remote access to all servers is required for deployment and support of product.
<input type="checkbox"/>	Service Account	The service account must have the ability to run passwordless sudo commands. The deployer will not without this ability.
<input type="checkbox"/>	File Transfer Process	The PingOne Autonomous Identity Team require access to a file transfer process, which lets specified packages be transferred from the vendor to the client infrastructure.
Service Account		
<input type="checkbox"/>	Service Account Group	The service account group must be the same as the service account name. For example, if the service account name is <code>srv-autoid</code> , that user must be in the group <code>srv-autoid</code> .
<input type="checkbox"/>	PingOne Autonomous Identity Team Access	PingOne Autonomous Identity team members must be able to switch to this user after logging in to the servers.
<input type="checkbox"/>	SSH Ability	The service account must be able to passwordless SSH between all PingOne Autonomous Identity servers; preferred method is RSA SSH key authentication.
<input type="checkbox"/>	Default Shell	The default shell of the service account must be Bash.
<input type="checkbox"/>	Directory Ownership	Ownership of the following directories must be given to the Service Account. <ul style="list-style-type: none"> • /data or applicable name of the shared mount (Docker and Spark servers) • /opt/autoid (all servers) • /tmp (R, W, E required + NOEXEC flag must not be present)
<input type="checkbox"/>	Docker Commands	The service account must have permissions to run Docker commands. Note that Docker should NOT need to be installed as a prerequisite; this will be installed by deployment team.
Networking/Internet		

<input type="checkbox"/>	Access to the Internet	If available, the front-end servers download the required Docker images from the official PingOne Autonomous Identity image repository.
<input type="checkbox"/>	SSL Certificates	<p>If SSL is being implemented, SSL certificates are required for the UI, Cassandra or MongoDB nodes, and Spark nodes. These certificates can be generated using one of the following four options:</p> <ul style="list-style-type: none"> • Self-signed certificates for all 3 components • Valid certificate for the UI and self-signed certificates for Cassandra, MongoDB, and Spark nodes (self-signed certs only used in server-server traffic) • Valid and separate certificates for the UI, Cassandra, MongoDB, and Spark • *.domainname.com certificate (wildcard)
<input type="checkbox"/>	Ports Open (Internal)	All internal ports specified in the Networking section of the Environment Specifications need to be opened for the specified servers.
<input type="checkbox"/>	Ports Open (external browser)	<p>The following ports must be accessible from a web browser within the client network:</p> <ul style="list-style-type: none"> • 443 (Front-end) <p>For a list of PingOne Autonomous Identity ports, refer to Autonomous Identity Ports.</p>
Required Packages		
<input type="checkbox"/>	Dependencies	<p>The following packages must be installed on specified servers as prerequisites:</p> <ul style="list-style-type: none"> • Analytics Servers: <ul style="list-style-type: none"> ◦ OpenJDK version "11.0.16" ◦ Python 3.10.9 with symlinks to Python 3 (sudo ln -s /usr/bin/python3.10 /usr/bin/python3)
Other		
<input type="checkbox"/>	Infrastructure Support POC	A point-of-contact (POC) with sufficient access to the infrastructure is required. The POC can support in case of infrastructure blockers arise (e.g., proxy, account access, or port issues).
<input type="checkbox"/>	SELinux	SELinux must be disabled on the Docker boxes. The package "container-selinux" must be present (this can be done as part of the root scripts described in the "Root Access" category).

Installation

This chapter shows you how to install and deploy PingOne Autonomous Identity for intelligent entitlements management in production environments. For hardware and software requirements, refer to the [Release notes](#).



Deployment architectures

Learn about the different deployment architectures.



Install single node

Install a single-node PingOne Autonomous Identity installation.



Install single node air-gap

Install a single-node air-gapped PingOne Autonomous Identity installation.



Install multi-node deployment

Install a multi-node PingOne Autonomous Identity installation for evaluation.



Install multi-node air-gapped

Install a multi-node PingOne Autonomous Identity air-gapped installation.



Upgrade

Upgrade to the latest version.



Appendix: Ports

Learn about the PingOne Autonomous Identity ports.



Appendix: vars.yml

Learn about the main deployment configuration file.

For a description of the PingOne Autonomous Identity UI console, refer to the [PingOne Autonomous Identity Users Guide](#).

Deployment architectures

To simplify your deployments, ForgeRock provides a deployer script to install PingOne Autonomous Identity on a target node. The deployer pulls in images from the ForgeRock Google Cloud Repository and uses it to deploy the microservices and analytics for PingOne Autonomous Identity on a target machine. The target machine only requires the base operating system.



Important

If you are upgrading PingOne Autonomous Identity on a RHEL 7/CentOS 7, the upgrade to 2022.11 uses RHEL 7/CentOS 7 only. For new and clean installations, PingOne Autonomous Identity requires RHEL 8 or CentOS Stream 8 only.

PingOne Autonomous Identity 2022.11.0 introduced a new deployer, Deployer Pro, that pulls in the base code from the ForgeRock Google Cloud repository. Customers must now pre-install the third-party software dependencies prior to running deployer pro. For more information, refer to [Install a single node deployment](#).

There are four basic deployments, all of them similar, but in slightly different configurations:

- **Single-Node Target Deployment.** Deploy PingOne Autonomous Identity on a single Internet-connected target machine. The deployer script lets you deploy the system from a local laptop or machine or from the target machine itself. The target machine can be on on-prem or on a cloud service, such as Google Cloud Platform (GCP), Amazon Web Services (AWS), Microsoft Azure or others. For installation instructions, refer to [Install a Single-Node Deployment](#).

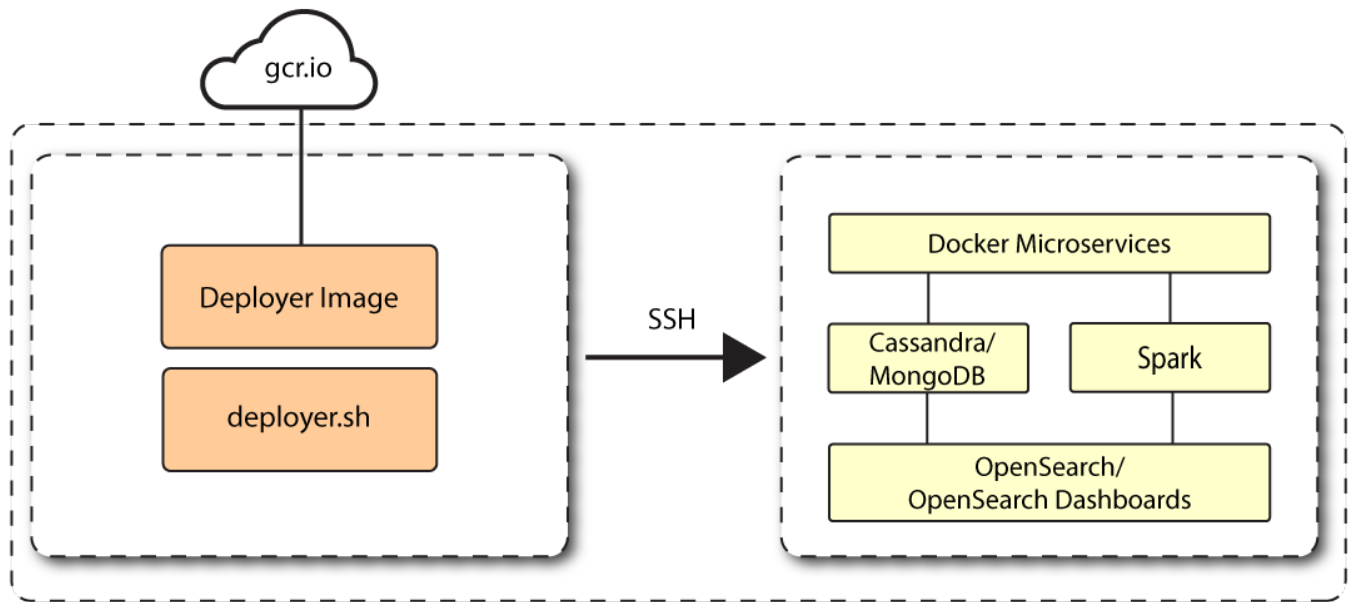


Figure 1. A single-node target deployment.

- **Single-Node Air-Gapped Target Deployment.** Deploy PingOne Autonomous Identity on a single-node target machine that resides in an air-gapped deployment. In an air-gapped deployment, the target machine is placed in an enhanced security environment where external Internet access is not available. You transfer the deployer and image to the target machine using media, such as a portable drive. Then, run the deployment within the air-gapped environment. For installation instruction, refer to [Install a Single-Node Air-Gapped](#).

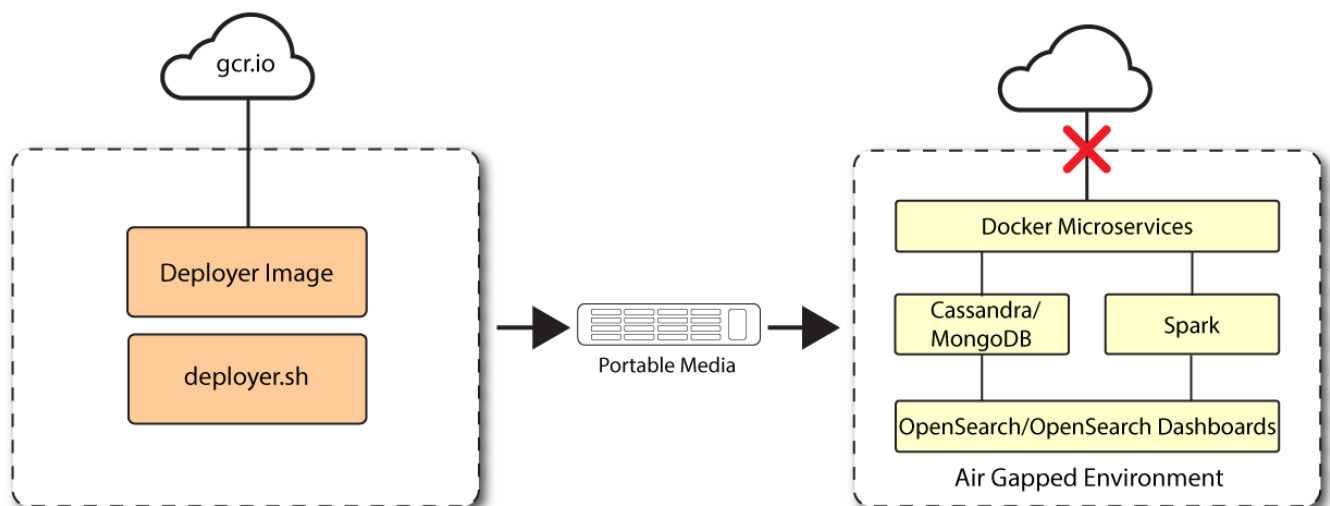


Figure 2. An air-gapped deployment.

- **Multi-Node Deployment.** Deploy PingOne Autonomous Identity on multi-node deployment to distribute the process load on the servers. For installation instruction, refer to [Install a Multi-Node Deployment](#).

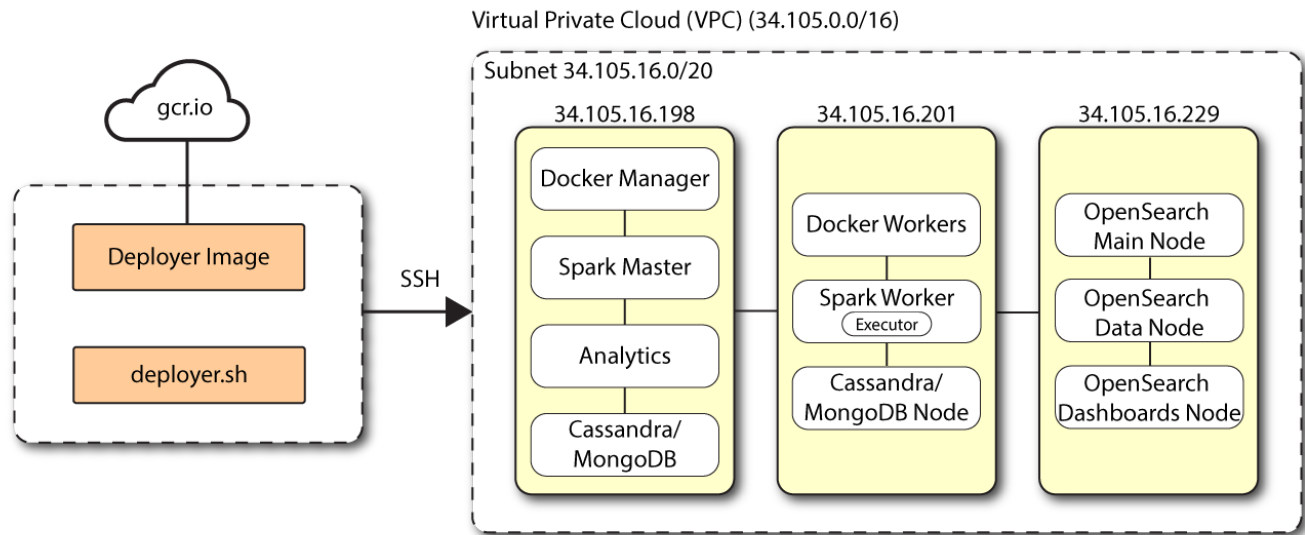


Figure 3. A multi-node target deployment.

- **Multi-Node Air-Gapped Deployment.** Deploy PingOne Autonomous Identity a multi-node configuration in an air-gapped network. The multinode network has no external Internet connection. For installation instruction, refer to [Install a Multi-Node Air-Gapped Deployment](#).

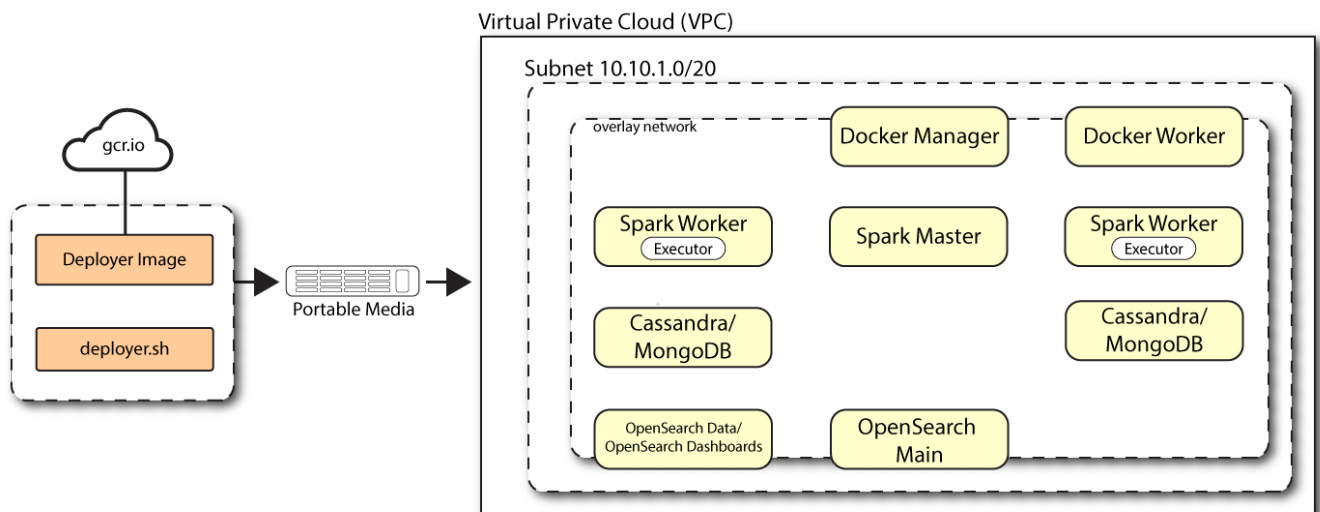


Figure 4. A multi-node air-gapped target deployment.

Install a single node deployment

This section presents instructions on deploying PingOne Autonomous Identity in a single-target machine with Internet connectivity.

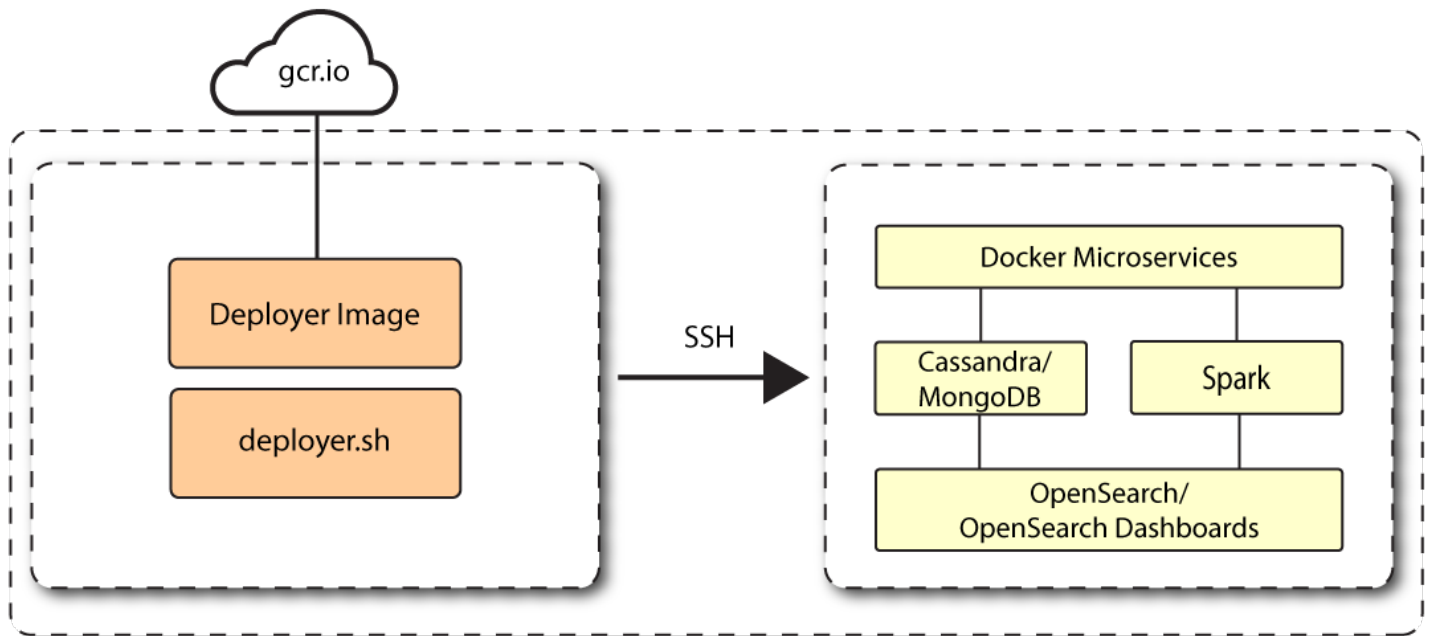


Figure 1. A single-node target deployment.

PingOne Autonomous Identity 2022.11.0 introduced a new installation script, *deployer pro* (**Deployer** for **Production**), letting customers manage their third-party software dependencies in their particular PingOne Autonomous Identity environments. PingOne Autonomous Identity 2022.11.8 continues to use this deployer script. For background information about the deployer, refer to [About the new deployer pro script](#).

Note

The procedures presented in this section are generalized examples to help you get acquainted with PingOne Autonomous Identity. Consult with your ForgeRock Professional Services or technical partner for specific assistance to install PingOne Autonomous Identity within your particular environment.

Summary of the installation steps

To set up the 2022.11.8 deployment, run the following steps:

- [Prerequisites](#)
- [Install third-party components](#)
- [Set up SSH on the deployer](#)
- [Install PingOne Autonomous Identity](#)

Prerequisites

For new and clean deployments, the following are prerequisites:

- **Operating System.** The target machine requires Red Hat Linux 8/CentOS Stream 8. The deployer machine can use any operating system as long as Docker is installed. For this chapter, we use CentOS Stream 8 as its base operating system.



Important

If you are upgrading PingOne Autonomous Identity on a RHEL 7/CentOS 7, the upgrade to 2022.11 uses RHEL 7/CentOS 7 only. For new and clean installations, PingOne Autonomous Identity requires RHEL 8 or CentOS Stream 8 only.

- **Memory Requirements.** Make sure you have enough free disk space on the deployer machine before running the `deployer.sh` commands. We recommend at least 500GB.
- **Default Shell.** The default shell for the `autoid` user must be `bash`.
- **Deployment Requirements.** PingOne Autonomous Identity provides a Docker image that creates a `deployer.sh` script. The script downloads additional images necessary for the installation. To download the deployment images, you must first obtain a registry key to log into the ForgeRock Google Cloud Registry. The registry key is only available to ForgeRock PingOne Autonomous Identity customers. For specific instructions on obtaining the registry key, refer to [How To Configure Service Credentials \(Push Auth, Docker\) in Backstage](#).
- **Database Requirements.** Decide which database you are using: Apache Cassandra or MongoDB.
- **IPv4 Forwarding.** Many high security environments run their CentOS-based systems with IPv4 forwarding disabled. However, Docker Swarm does not work with a disabled IPv4 forward setting. In such environments, make sure to enable IPv4 forwarding in the file `/etc/sysctl.conf`:

```
net.ipv4.ip_forward=1
```

Install third-party components

First, set up your GCP virtual machine and install the third-party package dependencies required for the PingOne Autonomous Identity deployment:

Install third-party packages:

1. Create a GCP Red Hat Enterprise Linux (RHEL) 8 or CentOS Stream 8 virtual machine: n2-standard-4 (4 vCPU and 16GB memory). Refer to <https://www.centos.org/centos-stream/>.
2. Create an `autoid` user with the proper privileges to run the installation. For example:

```
sudo adduser autoid
sudo passwd autoid
echo "autoid ALL=(ALL) NOPASSWD:ALL" | sudo tee /etc/sudoers.d/autoid
sudo usermod -aG wheel autoid
su - autoid
```

3. Install the following packages needed in the PingOne Autonomous Identity deployment:

- **Java 11.** For example, `sudo dnf install java-11-openjdk-devel`.
- **wget.** For example, `sudo dnf install wget`.
- **unzip.** For example, `sudo dnf install unzip`.

- **elinks**. For example, `sudo yum install -y elinks`.

4. Install Python 3.10.9.

1. Refer to <https://docs.python.org/release/3.10.9/>.
2. Make sure no other Python versions are installed on the machine. Remove those versions. For example:

```
sudo rm -rf /usr/bin/python3
sudo rm -rf /usr/bin/python3.6
sudo rm -rf /usr/bin/python3m
sudo rm -rf /usr/bin/pip3
sudo rm -rf /usr/bin/easy_install-3
sudo rm -rf /usr/bin/easy_install-3.6
```

3. Create symlinks for python3:

```
sudo ln -s /usr/bin/python 3.10 /usr/bin/python3
sudo ln -s /usr/bin/easy_install-3.10 /usr/bin/easy_install-3
sudo ln -s /usr/bin/pip3.10 /usr/bin/pip3
```

5. Install Cassandra 4.0.6. Refer to https://cassandra.apache.org/doc/latest/cassandra/getting_started/index.html. (For MongoDB installations, follow the instructions in [Download MongoDB](#).)

1. Log in to the Cassandra shell. For example:

```
cassandra/bin/cqlsh <$ipaddress> -u cassandra -p cassandra
```

2. Create the Cassandra roles for PingOne Autonomous Identity. Refer to <https://cassandra.apache.org/doc/latest/cassandra/cql/security.html>. For example:

```
cassandra/bin/cqlsh <$ipaddress> -u cassandra -p cassandra -e "CREATE ROLE zoran_dba WITH
PASSWORD = 'password' AND SUPERUSER = true AND LOGIN = true;"
cassandra/bin/cqlsh <$ipaddress> -u cassandra -p cassandra -e "CREATE ROLE zoranuser WITH
PASSWORD = 'password' AND LOGIN = true;"
cassandra/bin/cqlsh <$ipaddress> -u zoran_dba -p 'password' -e "ALTER ROLE cassandra WITH
PASSWORD='randompassword123' AND SUPERUSER=false AND LOGIN = false;"
cassandra/bin/cqlsh <$ipaddress> -u zoran_dba -p 'password' -e "ALTER KEYSPACE 'system_auth'
WITH REPLICATION = {'class' : 'NetworkTopologyStrategy', 'datacenter1' : 1};"
```

3. Configure security for Cassandra. Refer to <https://cassandra.apache.org/doc/latest/cassandra/operating/security.html>.

1. Install MongoDB 4.4. Follow the instructions in <https://www.mongodb.com/docs/v4.4/tutorial/install-mongodb-on-red-hat/>.

1. Create a MongoDB user with username `mongoadmin` with admin privileges. Follow the instructions in <https://www.mongodb.com/docs/v4.4/core/security-users/>.

For example:

```
db.createUser({ user: "mongoadmin",pwd: "~@C~0>@%^()-_+=|<Y*$rH&&/m#g{?-o!z/1}2??3=!*&",
roles: [ { role: "userAdminAnyDatabase", db: "admin" }, "readWriteAnyDatabase" ]})
```

2. Set up SSL, refer to <https://www.mongodb.com/docs/v4.4/tutorial/configure-ssl/#procedures—using-net.ssl-settings>. For example, the MongoDB configuration file (`/etc/mongod.conf`) would include a section similar to the following:

```
net:
  tls:
    mode: requireTLS
    certificateKeyFile: /etc/ssl/mongodb.pem
    CAFile: /etc/ssl/rootCA.pem
```

IMPORTANT

Make sure that the CN entry in the `mongodb.pem` certificate is the IP address/hostname of the `mongodb` instance. You need to add this same CN value to the `hosts` file during the PingOne Autonomous Identity deployment.

3. Restart the daemon and MongoDB.

```
sudo systemctl daemon-reload
sudo systemctl restart mongod
```

2. Install Apache Spark 3.3.2. Refer to <https://spark.apache.org/downloads.html>.

Note

The official release of Apache Livy does not support Apache Spark 3.3.1 or 3.3.2. ForgeRock has re-compiled and packaged Apache Livy to work with Apache Spark 3.3.1 hadoop 3 and Apache Spark 3.3.2 hadoop 3. Use the zip file located at `autoid-config/apache-livy/apache-livy-0.8.0-incubating-SNAPSHOT-bin.zip` to install Apache Livy on the Spark-Livy machine.

1. Configure the `SPARK_HOME` in your `bashrc` file. For example:

```
SPARK_HOME=/opt/spark/spark-3.3.2-bin-hadoop3
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
```

2. Configure authentication on Spark, refer to <https://spark.apache.org/docs/latest/security.html#authentication>. For example:

```
spark.authenticate          true
spark.authenticate.secret   <your-secret>
```

3. Enable and start the Spark main and secondary servers:

```
sudo chown -R $USER:USER $SPARK_HOME
```

4. Spark 3.3.1 and 3.3.2 no longer uses log4j1 and has upgraded to log4j2. Copy or move the log4j template file to the `log4j2.properties` file. For example:

```
mv /opt/spark/spark-3.3.2-bin-hadoop3/conf/log4j.properties.template /opt/spark/spark-3.3.2-bin-hadoop3/conf/log4j2.properties
```

Note

You will install Apache Livy in a later step. Refer to [Install Apache Livy](#).

3. Install Opensearch 1.3.13 and Opensearch Dashboards 1.3.13. For more information, refer to [Opensearch 1.3.13](#).
1. Configure Opensearch Dashboards using the `/Opensearch-dashboards/config/Opensearch_dashboards.yml` file. Refer to <https://Opensearch.org/docs/1.3/dashboards/install/index/>.
2. Configure TLS/SSL security:
 - Follow the instructions in <https://Opensearch.org/docs/latest/security-plugin/configuration/tls/>.
 - Follow the instructions in <https://Opensearch.org/docs/2.0/security-plugin/configuration/generate-certificates/>.

IMPORTANT

Make sure that the CN entry in the `esnode.pem` certificate is the IP address/hostname of the Opensearch instance. You need to add this same CN value to the `hosts` file during the PingOne Autonomous Identity deployment.

4. Set up Docker using the procedures in <https://docs.docker.com/engine/install/centos/>.
 - For post-installation Docker steps, follow the instructions in <https://docs.docker.com/engine/install/linux-postinstall/>.

Important

Do not use `/opt/autoid` as Docker root as the directory is overwritten during the PingOne Autonomous Identity installation and will result in a recursive error.

Set up SSH on the deployer

This section shows how to set up SSH keys for the `autoid` user to the target machine. This is a critical step and necessary for a successful deployment.

1. On the deployer machine, change to the SSH directory.

```
cd ~/.ssh
```

2. Run `ssh-keygen` to generate a 2048-bit RSA keypair for the `autoid` user, and then click **Enter**. Use the default settings, and do not enter a passphrase for your private key.

```
ssh-keygen -t rsa -C "autoid"
```

The public and private rsa key pair is stored in `home-directory/.ssh/id_rsa` and `home-directory/.ssh/id_rsa.pub`.

3. Copy the SSH key to the `autoid-config` directory.

```
cp id_rsa ~/autoid-config
```

4. Change the privileges and owner to the file.

```
chmod 400 ~/autoid-config/id_rsa
```

5. Copy your public SSH key, `id_rsa.pub`, to the target machine's `~/.ssh/authorized_keys` folder. If your target system does not have an `~/.ssh/authorized_keys`, create it using `sudo mkdir -p ~/.ssh`, then `sudo touch ~/.ssh/authorized_keys`.

This example uses `ssh-copy-id` to copy the public key to the target machine, which may or may not be available on your operating system. You can also manually copy-n-paste the public key to your `~/.ssh/authorized_keys` on the target machine.

```
ssh-copy-id -i id_rsa.pub autoid@<Target IP Address>
```



Note

The `ssh-copy-id` command requires that you have public key authentication enabled on the target server. You can enable it by editing the `/etc/ssh/sshd_config` file on the target machine. For example: `sudo vi /etc/ssh/sshd_config`, set **PubkeyAuthentication yes**, and save the file. Next, restart `sshd`: `sudo systemctl restart sshd`.

6. On the deployer machine, test your SSH connection to the target machine. This is a critical step. Make sure the connection works before proceeding with the installation.

```
ssh -i ~/.ssh/id_rsa autoid@<Target IP Address>
```

```
Last login: Tue Dec 14 14:06:06 2020
```

7. While SSH'ing into the target node, set the privileges on your `~/.ssh` and `~/.ssh/authorized_keys`.

```
chmod 700 ~/.ssh && chmod 600 ~/.ssh/authorized_keys
```

8. If you successfully accessed the remote server and changed the privileges on the `~/.ssh`, enter `exit` to end your SSH session.

Install PingOne Autonomous Identity

Make sure you have the following prerequisites:

- IP address of machines running Opensearch, MongoDB, or Cassandra.
- The PingOne Autonomous Identity user should have permission to write to `/opt/autoid` on all machines
- To download the deployment images for the install, you still need your registry key to log into the ForgeRock Google Cloud Registry to download the artifacts.
- Make sure you have the proper Opensearch certificates with the exact names for both pem and JKS files copied to `~/autoid-config/certs/elastic`:
 - `esnode.pem`
 - `esnode-key.pem`
 - `root-ca.pem`
 - `elastic-client-keystore.jks`
 - `elastic-server-truststore.jks`
- Make sure you have the proper MongoDB certificates with exact names for both pem and JKS files copied to `~/autoid-config/certs/mongo`:
 - `mongo-client-keystore.jks`
 - `mongo-server-truststore.jks`
 - `mongodb.pem`
 - `rootCA.pem`
- Make sure you have the proper Cassandra certificates with exact names for both pem and JKS files copied to `~/autoid-config/certs/cassandra`:
 - `Zoran-cassandra-client-cer.pem`
 - `Zoran-cassandra-client-keystore.jks`
 - `Zoran-cassandra-server-cer.pem`
 - `zoran-cassandra-server-keystore.jks`
 - `Zoran-cassandra-client-key.pem`
 - `Zoran-cassandra-client-truststore.jks`
 - `Zoran-cassandra-server-key.pem`
 - `Zoran-cassandra-server-truststore.jks`

Install PingOne Autonomous Identity:

1. Create the `autoid-config` directory.

```
mkdir autoid-config
```

2. Change to the directory.

```
cd autoid-config
```

3. Log in to the ForgeRock Google Cloud Registry using the registry key. The registry key is only available to ForgeRock PingOne Autonomous Identity customers. For specific instructions on obtaining the registry key, refer to [How To Configure Service Credentials \(Push Auth, Docker\) in Backstage](#).

```
docker login -u _json_key -p "$(cat autoid_registry_key.json)" https://gcr.io/forgerock-autoid
```

The following output is displayed:

```
Login Succeeded
```

4. Run the create-template command to generate the `deployer.sh` script wrapper and configuration files. Note that the command sets the configuration directory on the target node to `/config`. The `--user` parameter eliminates the need to use `sudo` while editing the hosts file and other configuration files.

```
docker run --user=$(id -u) -v ~/autoid-config:/config -it gcr.io/forgerock-autoid/deployer-pro:2022.11.8 create-template
```

5. Create a certificate directory for elastic.

```
mkdir -p autoid-config/certs/elastic
```

6. Copy the Opensearch certificates and JKS files to `autoid-config/certs/elastic`.

7. Create a certificate directory for MongoDB.

```
mkdir -p autoid-config/certs/mongo
```

8. Copy the MongoDB certificates and JKS files to `autoid-config/certs/mongo`.

9. Create a certificate directory for Cassandra.

```
mkdir -p autoid-config/certs/cassandra
```

10. Copy the Cassandra certificates and JKS files to `autoid-config/certs/cassandra`.

11. Update the `hosts` file with the IP addresses of the machines. The `hosts` file must include the IP addresses for Docker nodes, Spark main/livy, and the MongoDB master. While the deployer pro does not install or configure the MongoDB main server, the entry is required to run the MongoDB CLI to seed the PingOne Autonomous Identity schema.

```
[docker-managers]

[docker-workers]

[docker:children]
docker-managers
docker-workers

[spark-master-livy]

[cassandra-seeds]
#For replica sets, add the IPs of all Cassandra nodes

[mongo_master]
# Add the MongoDB main node in the cluster deployment
# For example: 10.142.15.248 mongodb_master=True

[odfe-master-node]
# Add only the main node in the cluster deployment

[kibana-node]
# Please add only the master node in cluster deployment
```

12. Update the `vars.yml` file:

1. Set `db_driver_type` to `mongo` or `cassandra`.
2. Set `elastic_host`, `elastic_port`, and `elastic_user` properties.
3. Set `kibana_host`.
4. Set the Apache livy install directory.
5. Ensure the `elastic_user`, `elastic_port`, and `mongo_part` are correctly configured.
6. Update the `vault.yml` passwords for elastic and mongo to reflect your installation.
7. Set the `mongo_ldap` variable to `true` if you want PingOne Autonomous Identity to authenticate with Mongo DB, configured as LDAP.

Note

The `mongo_ldap` variable only appears in fresh installs of 2022.11.0 and its upgrades (2022.11.1+). If you upgraded from a 2021.8.7 deployment, the variable is not available in your upgraded 2022.11.x deployment.

8. If you are using Cassandra, set the Cassandra-related parameters in the `vars.yml` file. Default values are:

```
cassandra:
  enable_ssl: "true"
  contact_points: 10.142.15.248 # comma separated values in case of replication set
  port: 9042
  username: zoran_dba
  cassandra_keystore_password: "Acc#1234"
  cassandra_truststore_password: "Acc#1234"
  ssl_client_key_file: "zoran-cassandra-client-key.pem"
  ssl_client_cert_file: "zoran-cassandra-client-cer.pem"
  ssl_ca_file: "zoran-cassandra-server-cer.pem"
  server_truststore_jks: "zoran-cassandra-server-truststore.jks"
  client_truststore_jks: "zoran-cassandra-client-truststore.jks"
  client_keystore_jks: "zoran-cassandra-client-keystore.jks"
```

13. Download images:

```
./deployer.sh download-images
```

1. Install Apache Livy.

- The official release of Apache Livy does not support Apache Spark 3.3.1 or 3.3.2. ForgeRock has re-compiled and packaged Apache Livy to work with Apache Spark 3.3.1 hadoop 3 and Apache Spark 3.3.2 hadoop 3. Use the zip file located at `autoid-config/apache-livy/apache-livy-0.8.0-incubating-SNAPSHOT-bin.zip` to install Apache Livy on the Spark-Livy machine.
- For Livy configuration, refer to <https://livy.apache.org/get-started/>.

2. On the Spark-Livy machine, run the following commands to install the python package dependencies:

1. Change to the `/opt/autoid` directory:

```
cd /opt/autoid
```

2. Create a `requirements.txt` file with the following content:

```
six==1.11
certifi==2019.11.28
python-dateutil==2.8.1
jsonschema==3.2.0
cassandra-driver
numpy==1.22.0
pyarrow==6.0.1
wrapt==1.11.0
PyYAML==6.0
requests==2.31.0
urllib3==1.26.18
pymongo
pandas==1.3.5
tabulate
openpyxl
wheel
cython
```

3. Install the requirements file:

```
pip3 install -r requirements.txt
```

3. Make sure that the `/opt/autoid` directory exists and that it is both readable and writable.

4. Run the deployer script:

```
./deployer.sh run
```

5. On the Spark-Livy machine, run the following commands to install the Python egg file:

1. Install the egg file:

```
cd /opt/autoid/eggs
pip3.10 install autoid_analytics-2021.3-py3-none-any.whl
```

2. Source the `.bashrc` file:

```
source ~/.bashrc
```

3. Restart Spark and Livy.

```
./spark/sbin/stop-all.sh
./livy/bin/livy-server stop

./spark/sbin/start-all.sh
./livy/bin/livy-server start
```

Resolve Hostname

After installing PingOne Autonomous Identity, set up the hostname resolution for your deployment.

Resolve the hostname:

1. Configure your DNS servers to access PingOne Autonomous Identity dashboard on the target node. The following domain names must resolve to the IP address of the target node: `<target-environment>-ui.<domain-name>`.
2. If DNS cannot resolve target node hostname, edit it locally on the machine that you want to access PingOne Autonomous Identity using a browser. Open a text editor and add an entry in the `/etc/hosts` (Linux/Unix) file or `C:\Windows\System32\drivers\etc\hosts` (Windows) for the self-service and UI services for each managed target node.

```
<Target IP Address> <target-environment>-ui.<domain-name>
```

For example:

```
34.70.190.144 autoid-ui.forgerock.com
```

3. If you set up a custom domain name and target environment, add the entries in `/etc/hosts`. For example:

```
34.70.190.144 myid-ui.abc.com
```

For more information on customizing your domain name, refer to [Customize Domains](#).

Access the Dashboard

Access the PingOne Autonomous Identity console UI:

1. Open a browser. If you set up your own url, use it for your login.

```
https://autoid-ui.forgerock.com/
```

2. Log in as a test user.

```
test user: bob.rodgers@forgerock.com  
password: <password>
```

Check Apache Cassandra

Check Cassandra:

1. Make sure Cassandra is running in cluster mode. For example

```
/opt/autoid/apache-cassandra-3.11.2/bin/nodetool status
```

Check MongoDB

Check MongoDB:

1. Make sure MongoDB is running. For example:

```
mongo --tls \
--host <Host IP> \
--tlsCAFile /opt/autoid/mongo/certs/rootCA.pem \
--tlsAllowInvalidCertificates \
--tlsCertificateKeyFile /opt/autoid/mongo/certs/mongodb.pem
```

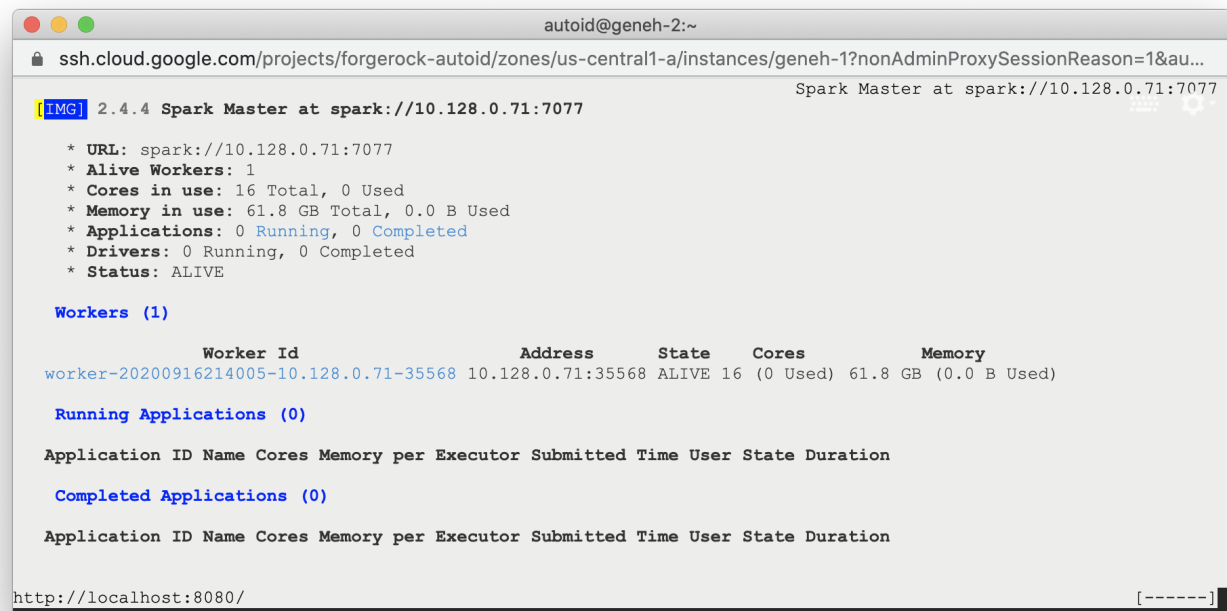
Check Apache Spark

Check Spark:

1. SSH to the target node and open Spark dashboard using the bundled text-mode web browser

```
elinks http://localhost:8080
```

Spark Master status should display as ALIVE and worker(s) with State ALIVE.



```
autoid@geneh-2:~
ssh.cloud.google.com/projects/forgerock-autoid/zones/us-central1-a/instances/geneh-1?nonAdminProxySessionReason=1&au...
[IMG] 2.4.4 Spark Master at spark://10.128.0.71:7077 Spark Master at spark://10.128.0.71:7077
* URL: spark://10.128.0.71:7077
* Alive Workers: 1
* Cores in use: 16 Total, 0 Used
* Memory in use: 61.8 GB Total, 0.0 B Used
* Applications: 0 Running, 0 Completed
* Drivers: 0 Running, 0 Completed
* Status: ALIVE

Workers (1)
Worker Id Address State Cores Memory
worker-20200916214005-10.128.0.71-35568 10.128.0.71:35568 ALIVE 16 (0 Used) 61.8 GB (0.0 B Used)

Running Applications (0)
Application ID Name Cores Memory per Executor Submitted Time User State Duration

Completed Applications (0)
Application ID Name Cores Memory per Executor Submitted Time User State Duration

http://localhost:8080/ [-----]
```

Start the Analytics

If the previous installation steps all succeeded, you must now prepare your data's entity definitions, data sources, and attribute mappings prior to running your analytics jobs. These steps are required and are critical for a successful analytics process.

For more information, refer to [Set Entity Definitions](#).

Install a single node air-gapped deployment

This section presents instructions on deploying PingOne Autonomous Identity in a single-node target machine that has no Internet connectivity. This type of configuration, called an *air-gap* or *offline* deployment, provides enhanced security by isolating itself from outside Internet or network access.

The air-gap installation is similar to that of the single-node target deployment with Internet connectivity, except that the image and deployer script must be saved on a portable drive and copied to the air-gapped target machine.

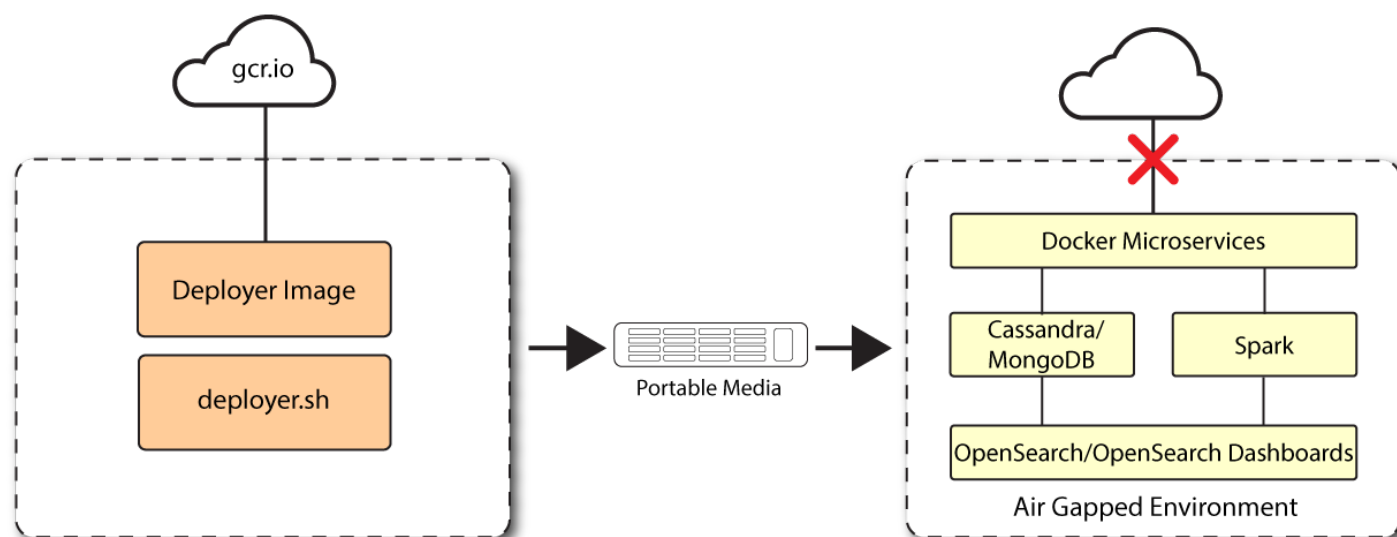


Figure 1. A single-node air-gapped target deployment.

Installation steps for an airgap deployment

The general procedure for an air-gap deployment is practically identical to that of a single node non-airgapped, except that you must prepare a tar file and copy the files to an air-gap machine.

- [Set up the nodes](#)
- [Set up the third-party software dependencies](#)
- [Set up SSH on the deployer](#)
- [Prepare the tar file](#)
- [Install PingOne Autonomous Identity](#)

Set up the nodes

Set up each node as presented in [Install a single node deployment](#).

Make sure you have sufficient storage for your particular deployment. For more information on sizing considerations, refer to [Deployment Planning Guide](#).

Set up the third-party software dependencies

Download and unpack the third-party software dependencies in [Install third-party components](#).

Set up SSH on the deployer

While SSH is not necessary to connect the deployer to the target node as the machines are isolated from one another. You still need SSH on the deployer so that it can communicate with itself.

1. On the deployer machine, run `ssh-keygen` to generate an RSA keypair, and then click **Enter**. You can use the default filename. Enter a password for protecting your private key.

```
ssh-keygen -t rsa -C "autoid"
```

The public and private rsa key pair is stored in `home-directory/.ssh/id_rsa` and `home-directory/.ssh/id_rsa.pub`.

2. Copy the SSH key to the `~/autoid-config` directory.

```
cp ~/.ssh/id_rsa ~/autoid-config
```

3. Change the privileges to the file.

```
chmod 400 ~/autoid-config/id_rsa
```

Prepare the tar file

Run the following steps on an Internet-connected host machine:

1. On the deployer machine, change to the installation directory.

```
cd ~/autoid-config/
```

2. Log in to the ForgeRock Google Cloud Registry using the registry key. The registry key is only available to ForgeRock PingOne Autonomous Identity customers. For specific instructions on obtaining the registry key, refer to [How To Configure Service Credentials \(Push Auth, Docker\) in Backstage](#).

```
docker login -u _json_key -p "$(cat autoid_registry_key.json)" https://gcr.io/forgerock-autoid
```

The following output is displayed:

```
Login Succeeded
```

3. Run the `create-template` command to generate the `deployer.sh` script wrapper. The command sets the configuration directory on the target node to `/config`. Note that the `--user` parameter eliminates the need to use `sudo` while editing the hosts file and other configuration files.

```
docker run --user=$(id -u) -v ~/autoid-config:/config -it gcr.io/forgerock-autoid/deployer-pro:2022.11.8 create-template
```

4. Open the `~/autoid-config/vars.yml` file, set the `offline_mode` property to `true`, and then save the file.

```
offline_mode: true
```

5. Download the Docker images. This step downloads software dependencies needed for the deployment and places them in the `autoid-packages` directory.

```
./deployer.sh download-images
```

6. Create a tar file containing all of the PingOne Autonomous Identity binaries.

```
tar czf autoid-packages.tgz deployer.sh autoid-packages/*
```

7. Copy the `autoid-packages.tgz`, `deployer.sh`, and SSH key (`id_rsa`) to a portable hard drive.

Install on the air-gap target

Before you begin, make sure you have CentOS Stream 8 and Docker installed on your air-gapped target machine.

1. Create the `~/autoid-config` directory if you haven't already.

```
mkdir ~/autoid-config
```

2. Copy the `autoid-package.tgz` tar file from the portable storage device.
3. Unpack the tar file.

```
tar xf autoid-packages.tgz -C ~/autoid-config
```

4. On the air-gap host node, copy the SSH key to the `~/autoid-config` directory.
5. Change the privileges to the file.

```
chmod 400 ~/autoid-config/id_rsa
```

6. Change to the configuration directory.

```
cd ~/autoid-config
```

7. Import the deployer image.

```
./deployer.sh import-deployer
```

The following output is displayed:

```
...
db631c8b06ee: Loading layer [=====→]    2.56kB/2.56kB
2d62082e3327: Loading layer [=====→]    753.2kB/753.2kB
Loaded image: gcr.io/forgerock-autoid/deployer:2022.11.8
```

8. Create the configuration template using the `create-template` command. This command creates the configuration files: `ansible.cfg`, `vars.yml`, `vault.yml` and `hosts`.

```
./deployer.sh create-template
```

The following output is displayed:

```
Config template is copied to host machine directory mapped to /config
```

Install PingOne Autonomous Identity

Make sure you have the following prerequisites:

- IP address of machines running Opensearch, MongoDB, or Cassandra.
- The PingOne Autonomous Identity user should have permission to write to `/opt/autoid` on all machines
- To download the deployment images for the install, you still need your registry key to log into the ForgeRock Google Cloud Registry to download the artifacts.
- Make sure you have the proper Opensearch certificates with the exact names for both pem and JKS files copied to `~/autoid-config/certs/elastic`:
 - `esnode.pem`
 - `esnode-key.pem`
 - `root-ca.pem`

- elastic-client-keystore.jks
- elastic-server-truststore.jks
- Make sure you have the proper MongoDB certificates with exact names for both pem and JKS files copied to `~/autoid-config/certs/mongo`:
 - mongo-client-keystore.jks
 - mongo-server-truststore.jks
 - mongod.pem
 - rootCA.pem
- Make sure you have the proper Cassandra certificates with exact names for both pem and JKS files copied to `~/autoid-config/certs/cassandra`:
 - Zoran-cassandra-client-cer.pem
 - Zoran-cassandra-client-keystore.jks
 - Zoran-cassandra-server-cer.pem
 - zoran-cassandra-server-keystore.jks
 - Zoran-cassandra-client-key.pem
 - Zoran-cassandra-client-truststore.jks
 - Zoran-cassandra-server-key.pem
 - Zoran-cassandra-server-truststore.jks

Install PingOne Autonomous Identity:

1. Create a certificate directory for elastic.

```
mkdir -p autoid-config/certs/elastic
```

2. Copy the Opensearch certificates and JKS files to `autoid-config/certs/elastic`.

3. Create a certificate directory for MongoDB.

```
mkdir -p autoid-config/certs/mongo
```

4. Copy the MongoDB certificates and JKS files to `autoid-config/certs/mongo`.

5. Create a certificate directory for Cassandra.

```
mkdir -p autoid-config/certs/cassandra
```

6. Copy the Cassandra certificates and JKS files to `autoid-config/certs/cassandra`.

7. Update the `hosts` file with the IP addresses of the machines. The `hosts` file must include the IP addresses for Docker nodes, Spark main/livy, and the MongoDB master. While the deployer pro does not install or configure the MongoDB main server, the entry is required to run the MongoDB CLI to seed the PingOne Autonomous Identity schema.

```
[docker-managers]

[docker-workers]

[docker:children]
docker-managers
docker-workers

[spark-master-livy]

[cassandra-seeds]
#For replica sets, add the IPs of all Cassandra nodes

[mongo_master]
# Add the MongoDB main node in the cluster deployment
# For example: 10.142.15.248 mongodb_master=True

[odfe-master-node]
# Add only the main node in the cluster deployment
```

8. Update the `vars.yml` file:

1. Set `offline_mode` to `true`.
2. Set `db_driver_type` to `mongo` or `cassandra`.
3. Set `elastic_host`, `elastic_port`, and `elastic_user` properties.
4. Set `kibana_host`.
5. Set the Apache livy install directory.
6. Ensure the `elastic_user`, `elastic_port`, and `mongo_part` are correctly configured.
7. Update the `vault.yml` passwords for elastic and mongo to reflect your installation.
8. Set the `mongo_ldap` variable to `true` if you want PingOne Autonomous Identity to authenticate with Mongo DB, configured as LDAP.

Note

The `mongo_ldap` variable only appears in fresh installs of 2022.11.0 and its upgrades (2022.11.1+). If you upgraded from a 2021.8.7 deployment, the variable is not available in your upgraded 2022.11.x deployment.

9. If you are using Cassandra, set the Cassandra-related parameters in the `vars.yml` file. Default values are:

```
cassandra:
  enable_ssl: "true"
  contact_points: 10.142.15.248 # comma separated values in case of replication set
  port: 9042
  username: zoran_dba
  cassandra_keystore_password: "Acc#1234"
  cassandra_truststore_password: "Acc#1234"
  ssl_client_key_file: "zoran-cassandra-client-key.pem"
  ssl_client_cert_file: "zoran-cassandra-client-cer.pem"
  ssl_ca_file: "zoran-cassandra-server-cer.pem"
  server_truststore_jks: "zoran-cassandra-server-truststore.jks"
  client_truststore_jks: "zoran-cassandra-client-truststore.jks"
  client_keystore_jks: "zoran-cassandra-client-keystore.jks"
```

9. Install Apache Livy.

- The official release of Apache Livy does not support Apache Spark 3.3.1 or 3.3.2. ForgeRock has re-compiled and packaged Apache Livy to work with Apache Spark 3.3.1 hadoop 3 and Apache Spark 3.3.2 hadoop 3. Use the zip file located at `autoid-config/apache-livy/apache-livy-0.8.0-incubating-SNAPSHOT-bin.zip` to install Apache Livy on the Spark-Livy machine.
- For Livy configuration, refer to <https://livy.apache.org/get-started/>.

10. On the Spark-Livy machine, run the following commands to install the python package dependencies:

1. Change to the `/opt/autoid` directory:

```
cd /opt/autoid
```

2. Create a `requirements.txt` file with the following content:

```
six==1.11
certifi==2019.11.28
python-dateutil==2.8.1
jsonschema==3.2.0
cassandra-driver
numpy==1.22.0
pyarrow==6.0.1
wrapt==1.11.0
PyYAML==6.0
requests==2.31.0
urllib3==1.26.18
pymongo
pandas==1.3.5
tabulate
openpyxl
wheel
cython
```

3. Install the requirements file:

```
pip3 install -r requirements.txt
```

11. Make sure that the `/opt/autoid` directory exists and that it is both readable and writable.

12. Run the deployer script:

```
./deployer.sh run
```

13. On the Spark-Livy machine, run the following commands to install the Python egg file:

1. Install the egg file:

```
cd /opt/autoid/eggs
pip3.10 install autoid_analytics-2021.3-py3-none-any.whl
```

2. Source the `.bashrc` file:

```
source ~/.bashrc
```

3. Restart Spark and Livy.

```
./spark/sbin/stop-all.sh
./livy/bin/livy-server stop

./spark/sbin/start-all.sh
./livy/bin/livy-server start
```

Resolve Hostname

After installing PingOne Autonomous Identity, set up the hostname resolution for your deployment.

Resolve the hostname:

1. Configure your DNS servers to access PingOne Autonomous Identity dashboard on the target node. The following domain names must resolve to the IP address of the target node: `<target-environment>-ui.<domain-name>`.
2. If DNS cannot resolve target node hostname, edit it locally on the machine that you want to access PingOne Autonomous Identity using a browser. Open a text editor and add an entry in the `/etc/hosts` (Linux/Unix) file or `C:\Windows\System32\drivers\etc\hosts` (Windows) for the self-service and UI services for each managed target node.

```
<Target IP Address> <target-environment>-ui.<domain-name>
```

For example:

```
34.70.190.144 autoid-ui.forgerock.com
```

3. If you set up a custom domain name and target environment, add the entries in `/etc/hosts` . For example:

```
34.70.190.144 myid-ui.abc.com
```

For more information on customizing your domain name, refer to [Customize Domains](#).

Access the Dashboard

Access the PingOne Autonomous Identity console UI:

1. Open a browser. If you set up your own url, use it for your login.

```
https://autoid-ui.forgerock.com/
```

2. Log in as a test user.

```
test user: bob.rodgers@forgerock.com
password: <password>
```

Check Apache Cassandra

Check Cassandra:

1. Make sure Cassandra is running in cluster mode. For example

```
/opt/autoid/apache-cassandra-3.11.2/bin/nodetool status
```

Check MongoDB

Check MongoDB:

1. Make sure MongoDB is running. For example:

```
mongo --tls \
--host <Host IP> \
--tlsCAFile /opt/autoid/mongo/certs/rootCA.pem \
--tlsAllowInvalidCertificates \
--tlsCertificateKeyFile /opt/autoid/mongo/certs/mongodb.pem
```

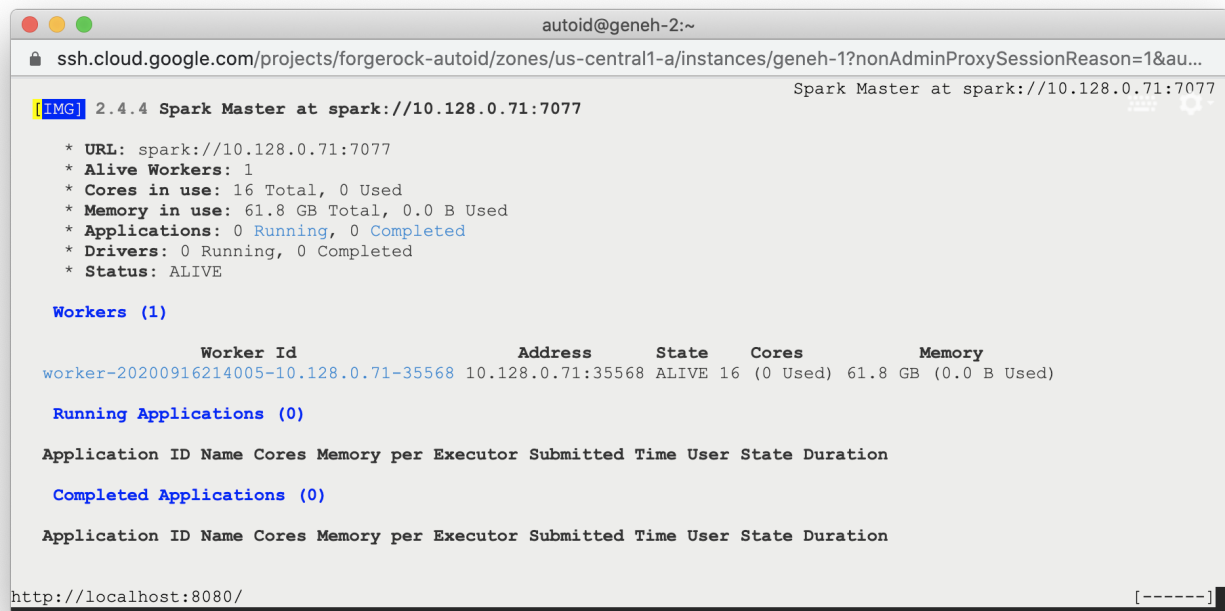
Check Apache Spark

Check Spark:

1. SSH to the target node and open Spark dashboard using the bundled text-mode web browser

```
elinks http://localhost:8080
```

Spark Master status should display as ALIVE and worker(s) with State ALIVE.



```

autoid@geneh-2:~
ssh.cloud.google.com/projects/forgerock-autoid/zones/us-central1-a/instances/geneh-1?nonAdminProxySessionReason=1&au...
Spark Master at spark://10.128.0.71:7077

2.4.4 Spark Master at spark://10.128.0.71:7077
* URL: spark://10.128.0.71:7077
* Alive Workers: 1
* Cores in use: 16 Total, 0 Used
* Memory in use: 61.8 GB Total, 0.0 B Used
* Applications: 0 Running, 0 Completed
* Drivers: 0 Running, 0 Completed
* Status: ALIVE

Workers (1)
Worker Id Address State Cores Memory
worker-20200916214005-10.128.0.71-35568 10.128.0.71:35568 ALIVE 16 (0 Used) 61.8 GB (0.0 B Used)

Running Applications (0)
Application ID Name Cores Memory per Executor Submitted Time User State Duration

Completed Applications (0)
Application ID Name Cores Memory per Executor Submitted Time User State Duration

http://localhost:8080/ [-----]

```

Start the Analytics

If the previous installation steps all succeeded, you must now prepare your data's entity definitions, data sources, and attribute mappings prior to running your analytics jobs. These steps are required and are critical for a successful analytics process.

For more information, refer to [Set Entity Definitions](#).

Install a multi-node deployment

This section presents instructions on deploying PingOne Autonomous Identity in a multi-node deployment. Multi-node deployments are configured in production environments, providing performant throughput by distributing the processing load across servers and supporting failover redundancy.

Like single-node deployment, ForgeRock provides a Deployer Pro script to pull a Docker image from ForgeRock's Google Cloud Registry repository with the microservices and analytics needed for the system. The deployer also uses the node IP addresses specified in your `hosts` file to set up an overlay network and your nodes.

 **Note**

The procedures are similar to multinode deployments using older PingOne Autonomous Identity release, except that you must install and configure the dependent software packages (for example, Apache Cassandra/MongoDB, Apache Spark and Livy, Opensearch and Opensearch Dashboards, and Docker) prior to running PingOne Autonomous Identity.

Summary of the installation steps

To set up the 2022.11.8 deployment, run the following steps:

- [Prerequisites](#)
- [Set up the nodes](#)
- [Install third-party components](#)
- [Set up SSH on the deployer](#)
- [Set up a shared data folder](#)
- [Install PingOne Autonomous Identity](#)
- [Set the Cassandra replication factor](#)

Prerequisites

Deploy PingOne Autonomous Identity on a multi-node target on Redhat Linux Enterprise 8 or CentOS Stream 8. The following are prerequisites:

- **Operating System.** The target machine requires Redhat Linux Enterprise 8 or CentOS Stream 8. The deployer machine can use any operating system as long as Docker is installed. For this chapter, we use Redhat Linux Enterprise 8 as its base operating system.



Important

If you are upgrading PingOne Autonomous Identity on a RHEL 7/CentOS 7, the upgrade to 2022.11 uses RHEL 7/CentOS 7 only. For new and clean installations, PingOne Autonomous Identity requires RHEL 8 or CentOS Stream 8 only.

- **Default Shell.** The default shell for the `autoid` user must be `bash`.
- **Subnet Requirements.** We recommend deploying your multi-node machines within the same subnet. Ports must be open for the installation to succeed. Each instance should be able to communicate to the other instances.



Important

If any hosts used for the Docker cluster (docker-managers, docker-workers) have an IP address in the range of 10.0.x.x, they will conflict with the Swarm network. As a result, the services in the cluster will not connect to the Cassandra database or Elasticsearch backend.

The Docker cluster hosts must be in a subnet that provides IP addresses 10.10.1.x or higher.

- **Deployment Requirements.** PingOne Autonomous Identity provides a `deployer.sh` script that downloads and installs the necessary Docker images. To download the deployment images, you must first obtain a registry key to log into the ForgeRock Google Cloud Registry. The registry key is only available to ForgeRock PingOne Autonomous Identity customers. For specific instructions on obtaining the registry key, refer to [How To Configure Service Credentials \(Push Auth, Docker\) in Backstage](#).
- **Filesystem Requirements.** PingOne Autonomous Identity requires a shared filesystem accessible from the Spark main, Spark worker, analytics hosts, and application layer. The shared filesystem should be mounted at the same mount directory on all of those hosts. If the mount directory for the shared filesystem is different from the default, `/data`, update the `/autoid-config/vars.yml` file to point to the correct directories:

```
analytics_data_dir: /data
analytics_conf_dir: /data/conf
```

- **Architecture Requirements.** Make sure that the Spark main is on a separate node from the Spark workers.
- **Database Requirements.** Decide which database you are using: Apache Cassandra or MongoDB. The configuration procedure is slightly different for each database.
- **Deployment Best-Practice.** The example combines the Opensearch data and Opensearch Dashboards nodes. For best performance in production, dedicate a separate node to Opensearch, data nodes, and Opensearch Dashboards.
- **IPv4 Forwarding.** Many high-security environments run their CentOS-based systems with IPv4 forwarding disabled. However, Docker Swarm does not work with a disabled IPv4 forward setting. In such environments, make sure to enable IPv4 forwarding in the file `/etc/sysctl.conf`:

```
net.ipv4.ip_forward=1
```

Note

We recommend that your deployer team have someone with Cassandra expertise. This guide is not sufficient to troubleshoot any issues that may arise.

Set up the nodes

Set up three virtual machines.

1. Create a Redhat Linux Enterprise 8 or CentOS Stream 8 virtual machine: N2 4 core and 16 GB. Verify your operating system.

```
sudo cat /etc/centos-release
```

Important

For multinode deployments, there is a known issue with RHEL 8/CentOS Stream 8 and overlay network configurations. Refer to [Known Issues in 2022.11.0](#).

2. Set the user for the target node to `autoid`. In this example, create user `autoid`:

```
sudo adduser autoid
sudo passwd autoid
echo "autoid ALL=(ALL) NOPASSWD:ALL" | sudo tee /etc/sudoers.d/autoid
sudo usermod -aG wheel autoid
su - autoid
```

3. Optional. Install yum-utils package on the deployer machine. yum-utils is a utilities manager for the Yum RPM package repository. The repository compresses software packages for Linux distributions.

```
sudo yum install -y yum-utils
```

4. Install the following packages needed in the PingOne Autonomous Identity deployment:

- **Java 11.** For example, `sudo dnf install java-11-openjdk-devel`.
- **wget.** For example, `sudo dnf install wget`.
- **unzip.** For example, `sudo dnf install unzip`.
- **elinks.** For example, `sudo yum install -y elinks`.
- **Python 3.10.9.** Refer to <https://docs.python.org/release/3.10.9/>.

5. Repeat this procedure for the other nodes.

Install third-party components

Set up a machine with the required third-party software dependencies. Refer to: [Install third-party components](#).

Set up SSH on the deployer

1. On the deployer machine, change to the `~/.ssh` directory.

```
cd ~/.ssh
```

2. Run `ssh-keygen` to generate an RSA keypair, and then click **Enter**. You can use the default filename.



Important

Do not add a key passphrase as it results in a build error.

```
ssh-keygen -t rsa -C "autoid"
```

The public and private rsa key pair is stored in `home-directory/.ssh/id_rsa` and `home-directory/.ssh/id_rsa.pub`.

3. Copy the SSH key to the `autoid-config` directory.

```
cp id_rsa ~/autoid-config
```

4. Change the privileges to the file.

```
chmod 400 ~/autoid-config/id_rsa
```

5. Copy your public SSH key, `id_rsa.pub`, to each of your nodes.



Note

If your target system does not have an `~/.ssh/authorized_keys`, create it using `sudo mkdir -p ~/.ssh`, then `sudo touch ~/.ssh/authorized_keys`.

For this example, copy the SSH key to each node:

```
ssh-copy-id -i id_rsa.pub autoid@<Node IP Address>
```

6. On the deployer machine, test your SSH connection to each target machine. This is a critical step. Make sure the connection works before proceeding with the installation.

For example, SSH to first node:

```
ssh -i id_rsa autoid@<Node 1 IP Address>
```

```
Last login: Sat Oct 3 03:02:40 2020
```

7. If you can successfully SSH to each machine, set the privileges on your `~/.ssh` and `~/.ssh/authorized_keys`.

```
chmod 700 ~/.ssh && chmod 600 ~/.ssh/authorized_keys
```

8. Enter Exit to end your SSH session.

9. Repeat steps 5–8 again for each node.

Set up a shared data folder

The Docker main and worker nodes plus the analytics main and worker nodes require a shared data directory, typically, `/data`. There are numerous ways to set up a shared directory, the following procedure is just one example and sets up an NFS server on the analytics master.

1. On the Analytics Spark Main node, install `nfs-utils`. This step may require that you run the install with root privileges, such as `sudo` or equivalent.

```
sudo yum install -y nfs-utils
```

2. Create the `/data` directory.

```
mkdir -p /data
```

3. Change the permissions on the `/data` directory.

```
chmod -R 755 /data
chown nfsnobody:nfsnobody /data
```

4. Start the services and enable them to start at boot.

```
systemctl enable rpcbind
systemctl enable nfs-server
systemctl enable nfs-lock
systemctl enable nfs-idmap

systemctl start rpcbind
systemctl start nfs-server
systemctl start nfs-lock
systemctl start nfs-idmap
```

5. Define the sharing points in the `/etc/exports` file.

```
vi /etc/exports

/data <Remote IP Address 1>(rw,sync,no_root_squash,no_all_squash)
/data <Remote IP Address 2>(rw,sync,no_root_squash,no_all_squash)
```

If you change the domain name and target environment, you need to also change the certificates to reflect the new changes. For more information, refer to [Customize Domains](#).

6. Start the NFS service.

```
systemctl restart nfs-server
```

7. Add the NFS service to the `firewall-cmd` public zone service:

```
firewall-cmd --permanent --zone=public --add-service=nfs
firewall-cmd --permanent --zone=public --add-service=mountd
firewall-cmd --permanent --zone=public --add-service=rpc-bind
firewall-cmd --reload
```

8. On each spark worker node, run the following:

1. Install `nfs-utils`:

```
yum install -y nfs-utils
```

2. Create the NFS directory mount points:

```
mkdir -p /data
```

3. Mount the NFS shared directory:

```
mount -t nfs <NFS Server IP>:/data /data
```

4. Test the new shared directory by creating a small text file. On an analytics worker node, run the following, and then check for the presence of the test file on the other servers:

```
cd /data  
touch test
```

Install PingOne Autonomous Identity

Make sure you have the following prerequisites:

- IP address of machines running Opensearch, MongoDB, or Cassandra.
- The PingOne Autonomous Identity user should have permission to write to `/opt/autoid` on all machines
- To download the deployment images for the install, you still need your registry key to log into the ForgeRock Google Cloud Registry to download the artifacts.
- Make sure you have the proper Opensearch certificates with the exact names for both pem and JKS files copied to `~/autoid-config/certs/elastic`:
 - esnode.pem
 - esnode-key.pem
 - root-ca.pem
 - elastic-client-keystore.jks
 - elastic-server-truststore.jks
- Make sure you have the proper MongoDB certificates with exact names for both pem and JKS files copied to `~/autoid-config/certs/mongo`:
 - mongo-client-keystore.jks
 - mongo-server-truststore.jks
 - mongod.pem
 - rootCA.pem

- Make sure you have the proper Cassandra certificates with exact names for both pem and JKS files copied to `~/autoid-config/certs/cassandra`:
 - `Zoran-cassandra-client-cer.pem`
 - `Zoran-cassandra-client-keystore.jks`
 - `Zoran-cassandra-server-cer.pem`
 - `zoran-cassandra-server-keystore.jks`
 - `Zoran-cassandra-client-key.pem`
 - `Zoran-cassandra-client-truststore.jks`
 - `Zoran-cassandra-server-key.pem`
 - `Zoran-cassandra-server-truststore.jks`

Install PingOne Autonomous Identity:

1. Create the `autoid-config` directory.

```
mkdir autoid-config
```

2. Change to the directory.

```
cd autoid-config
```

3. Log in to the ForgeRock Google Cloud Registry using the registry key. The registry key is only available to ForgeRock PingOne Autonomous Identity customers. For specific instructions on obtaining the registry key, refer to [How To Configure Service Credentials \(Push Auth, Docker\) in Backstage](#).

```
docker login -u _json_key -p "$(cat autoid_registry_key.json)" https://gcr.io/forgerock-autoid
```

The following output is displayed:

```
Login Succeeded
```

4. Run the create-template command to generate the `deployer.sh` script wrapper and configuration files. Note that the command sets the configuration directory on the target node to `/config`. The `--user` parameter eliminates the need to use `sudo` while editing the hosts file and other configuration files.

```
docker run --user=$(id -u) -v ~/autoid-config:/config -it gcr.io/forgerock-autoid/deployer-pro:2022.11.8 create-template
```

5. Create a certificate directory for elastic.

```
mkdir -p autoid-config/certs/elastic
```

6. Copy the Opensearch certificates and JKS files to `autoid-config/certs/elastic`.

7. Create a certificate directory for MongoDB.

```
mkdir -p autoid-config/certs/mongo
```

8. Copy the MongoDB certificates and JKS files to `autoid-config/certs/mongo`.

9. Create a certificate directory for Cassandra.

```
mkdir -p autoid-config/certs/cassandra
```

10. Copy the Cassandra certificates and JKS files to `autoid-config/certs/cassandra`.

11. Update the `hosts` file with the IP addresses of the machines. The `hosts` file must include the IP addresses for Docker nodes, Spark main/livy, and the MongoDB master. While the deployer pro does not install or configure the MongoDB main server, the entry is required to run the MongoDB CLI to seed the PingOne Autonomous Identity schema.

```
[docker-managers]

[docker-workers]

[docker:children]
docker-managers
docker-workers

[spark-master-livy]

[cassandra-seeds]
#For replica sets, add the IPs of all Cassandra nodes

[mongo_master]
# Add the MongoDB main node in the cluster deployment
# For example: 10.142.15.248 mongodb_master=True

[odfe-master-node]
# Add only the main node in the cluster deployment
```

12. Update the `vars.yml` file:

1. Set `db_driver_type` to `mongo` or `cassandra`.
2. Set `elastic_host`, `elastic_port`, and `elastic_user` properties.
3. Set `kibana_host`.
4. Set the Apache livy install directory.

5. Ensure the `elastic_user`, `elastic_port`, and `mongo_part` are correctly configured.
6. Update the `vault.yml` passwords for elastic and mongo to reflect your installation.
7. Set the `mongo_ldap` variable to `true` if you want PingOne Autonomous Identity to authenticate with Mongo DB, configured as LDAP.

Note

The `mongo_ldap` variable only appears in fresh installs of 2022.11.0 and its upgrades (2022.11.1+). If you upgraded from a 2021.8.7 deployment, the variable is not available in your upgraded 2022.11.x deployment.

8. If you are using Cassandra, set the Cassandra-related parameters in the `vars.yml` file. Default values are:

```
cassandra:
  enable_ssl: "true"
  contact_points: 10.142.15.248 # comma separated values in case of replication set
  port: 9042
  username: zoran_dba
  cassandra_keystore_password: "Acc#1234"
  cassandra_truststore_password: "Acc#1234"
  ssl_client_key_file: "zoran-cassandra-client-key.pem"
  ssl_client_cert_file: "zoran-cassandra-client-cer.pem"
  ssl_ca_file: "zoran-cassandra-server-cer.pem"
  server_truststore_jks: "zoran-cassandra-server-truststore.jks"
  client_truststore_jks: "zoran-cassandra-client-truststore.jks"
  client_keystore_jks: "zoran-cassandra-client-keystore.jks"
```

13. Download images:

```
./deployer.sh download-images
```

14. Install Apache Livy.

- The official release of Apache Livy does not support Apache Spark 3.3.1 or 3.3.2. ForgeRock has re-compiled and packaged Apache Livy to work with Apache Spark 3.3.1 hadoop 3 and Apache Spark 3.3.2 hadoop 3. Use the zip file located at `autoid-config/apache-livy/apache-livy-0.8.0-incubating-SNAPSHOT-bin.zip` to install Apache Livy on the Spark-Livy machine.
- For Livy configuration, refer to <https://livy.apache.org/get-started/>.

15. On the Spark-Livy machine, run the following commands to install the python package dependencies:

1. Change to the `/opt/autoid` directory:

```
cd /opt/autoid
```

2. Create a `requirements.txt` file with the following content:

```
six==1.11
certifi==2019.11.28
python-dateutil==2.8.1
jsonschema==3.2.0
cassandra-driver
numpy==1.22.0
pyarrow==6.0.1
wrapt==1.11.0
PyYAML==6.0
requests==2.31.0
urllib3==1.26.18
pymongo
pandas==1.3.5
tabulate
openpyxl
wheel
cython
```

3. Install the requirements file:

```
pip3 install -r requirements.txt
```

16. Make sure that the `/opt/autoid` directory exists and that it is both readable and writable.

17. Run the deployer script:

```
./deployer.sh run
```

18. On the Spark-Livy machine, run the following commands to install the Python egg file:

1. Install the egg file:

```
cd /opt/autoid/eggs
pip3.10 install autoid_analytics-2021.3-py3-none-any.whl
```

2. Source the `.bashrc` file:

```
source ~/.bashrc
```

3. Restart Spark and Livy.

```
./spark/sbin/stop-all.sh
./livy/bin/livy-server stop

./spark/sbin/start-all.sh
./livy/bin/livy-server start
```

Set the Cassandra replication factor

Once Cassandra has been deployed, you need to set the replication factor to match the number of nodes on your system. This ensures that each record is stored in each of the nodes. In the event one node is lost, the remaining node can continue to serve content even if the cluster itself is running with reduced redundancy.

You can define replication on a per keyspace-basis as follows:

1. Start the Cassandra shell, `cqlsh`, and define the `autoid` keyspace. Change the replication factor to match the number of seed nodes. The default admin user for Cassandra is `zoran_dba`.

```
bin/cqlsh -u zoran_dba

zoran_dba@cqlsh> desc keyspace autoid;
CREATE KEYSPACE autoid WITH replication = {'class':'SimpleStrategy','replication_factor':'2'} AND
durable_writes=true;

CREATE TABLE autoid.user_access_decisions_history(
  user text,
  entitlement text,
  date_created timestamp,
  ...
```

2. Restart Cassandra on this node.
3. Repeat these steps on the other Cassandra seed node(s).

Resolve Hostname

After installing PingOne Autonomous Identity, set up the hostname resolution for your deployment.

1. Configure your DNS servers to access PingOne Autonomous Identity dashboard on the target node. The following domain names must resolve to the IP address of the target node:

```
<target-environment>-ui.<domain-name>
```

2. If DNS cannot resolve target node hostname, edit it locally on the machine that you want to access PingOne Autonomous Identity using a browser.

Open a text editor and add an entry in the `/etc/hosts` (Linux/Unix) file or `C:\Windows\System32\drivers\etc\hosts` (Windows) for the target node.

For multi-node, use the Docker Manager node as your target.

```
<Docker Mgr Node Public IP Address> <target-environment>-ui.<domain-name>
```

For example:

```
<IP Address> autoid-ui.forgerock.com
```

3. If you set up a custom domain name and target environment, add the entries in `/etc/hosts` . For example:

```
<IP Address> myid-ui.abc.com
```

For more information on customizing your domain name, see [Customize Domains](#).

Access the Dashboard

Access the PingOne Autonomous Identity console UI:

1. Open a browser. If you set up your own url, use it for your login.

```
https://autoid-ui.forgerock.com/
```

2. Log in as a test user.

```
test user: bob.rodgers@forgerock.com  
password: <password>
```

Check Apache Cassandra

Check Cassandra:

1. Make sure Cassandra is running in cluster mode. For example

```
/opt/autoid/apache-cassandra-3.11.2/bin/nodetool status
```

Check MongoDB

Check MongoDB:

1. Make sure MongoDB is running. For example:

```
mongo --tls \  
--host <Host IP> \  
--tlsCAFile /opt/autoid/mongo/certs/rootCA.pem \  
--tlsAllowInvalidCertificates \  
--tlsCertificateKeyFile /opt/autoid/mongo/certs/mongodb.pem
```

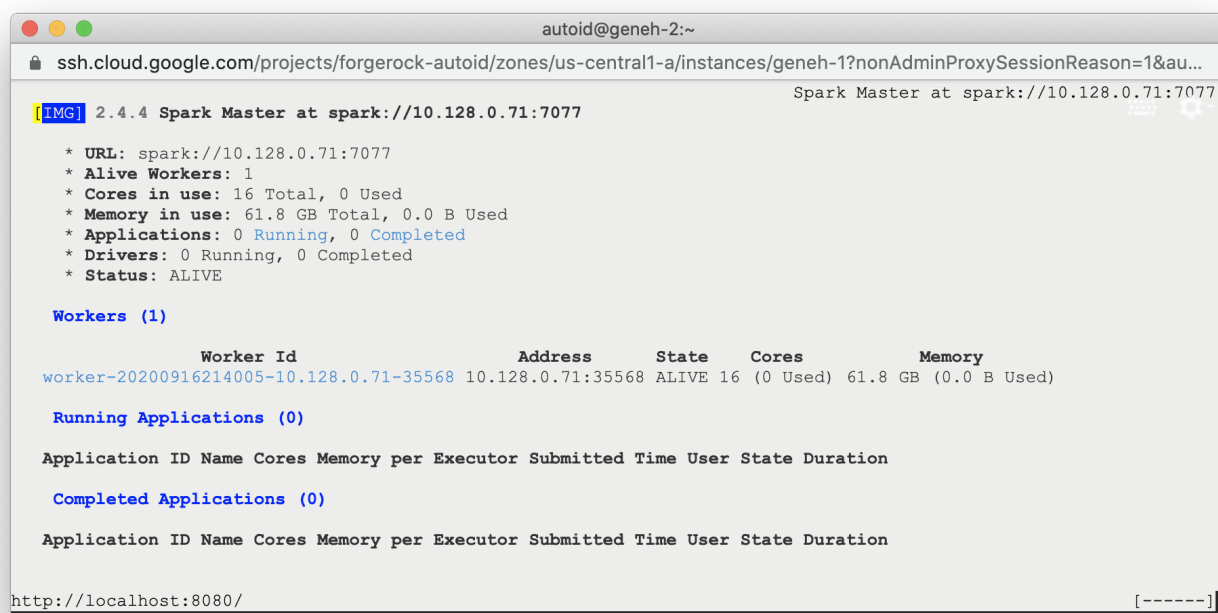
Check Apache Spark

Check Spark:

1. SSH to the target node and open Spark dashboard using the bundled text-mode web browser

```
elinks http://localhost:8080
```

Spark Master status should display as ALIVE and worker(s) with State ALIVE.



```

autoid@geneh-2:~
ssh.cloud.google.com/projects/forgerock-autoid/zones/us-central1-a/instances/geneh-1?nonAdminProxySessionReason=1&au...
Spark Master at spark://10.128.0.71:7077

2.4.4 Spark Master at spark://10.128.0.71:7077
* URL: spark://10.128.0.71:7077
* Alive Workers: 1
* Cores in use: 16 Total, 0 Used
* Memory in use: 61.8 GB Total, 0.0 B Used
* Applications: 0 Running, 0 Completed
* Drivers: 0 Running, 0 Completed
* Status: ALIVE

Workers (1)
Worker Id Address State Cores Memory
worker-20200916214005-10.128.0.71-35568 10.128.0.71:35568 ALIVE 16 (0 Used) 61.8 GB (0.0 B Used)

Running Applications (0)
Application ID Name Cores Memory per Executor Submitted Time User State Duration

Completed Applications (0)
Application ID Name Cores Memory per Executor Submitted Time User State Duration

http://localhost:8080/

```

Start the Analytics

If the previous installation steps all succeeded, you must now prepare your data's entity definitions, data sources, and attribute mappings prior to running your analytics jobs. These step are required and are critical for a successful analytics process.

For more information, refer to [Set Entity Definitions](#).

Install a multi-node air-gapped deployment

This chapter presents instructions on deploying PingOne Autonomous Identity in a multi-node air-gapped or offline target machine with no external Internet connectivity. ForgeRock provides a deployer script that pulls a Docker image from ForgeRock's Google Cloud Registry repository. The image contains the microservices, analytics, and backend databases needed for the system.

The air-gap installation is similar to that of the multi-node deployment, except that the image and deployer script must be stored on a portable drive and copied to the air-gapped target environment.

The deployment depends on how the network is configured. You could have a Docker cluster with multiple Spark nodes and Cassandra or MongoDB nodes. The key is to determine the IP addresses of each node.

Summary of the installation steps

To set up the 2022.11.8 deployment, run the following steps:

- [Prerequisites](#)
- [Set up the nodes](#)
- [Set up SSH on the deployer](#)
- [Prepare the tar file](#)
- [Install third-party components](#)
- [Install PingOne Autonomous Identity air-gapped](#)
- [Set the replication factor](#)

Prerequisites

Deploy PingOne Autonomous Identity on a multi-node air-gapped target on Redhat Linux Enterprise 8 or CentOS Stream 8. The following are prerequisites:

- **Operating System.** The target machine requires Redhat Linux Enterprise 8 or CentOS Stream 8. The deployer machine can use any operating system as long as Docker is installed. For this chapter, we use Redhat Linux Enterprise 8 as its base operating system.



Important

If you are upgrading PingOne Autonomous Identity on a RHEL 7/CentOS 7, the upgrade to 2022.11 uses RHEL 7/CentOS 7 only. For new and clean installations, PingOne Autonomous Identity requires RHEL 8 or CentOS Stream 8 only.

- **Default Shell.** The default shell for the `autoid` user must be `bash`.
- **Subnet Requirements.** We recommend deploying your multi-node machines within the same subnet. Ports must be open for the installation to succeed. Each instance should be able to communicate to the other instances.



Important

If any hosts used for the Docker cluster (docker-managers, docker-workers) have an IP address in the range of 10.0.x.x, they will conflict with the Swarm network. As a result, the services in the cluster will not connect to the Cassandra database or Elasticsearch backend.

The Docker cluster hosts must be in a subnet that provides IP addresses 10.10.1.x or higher.

- **Deployment Requirements.** PingOne Autonomous Identity provides a `deployer.sh` script that downloads and installs the necessary Docker images. To download the deployment images, you must first obtain a registry key to log into the ForgeRock Google Cloud Registry. The registry key is only available to ForgeRock PingOne Autonomous Identity customers. For specific instructions on obtaining the registry key, refer to [How To Configure Service Credentials \(Push Auth, Docker\) in Backstage](#).
- **Filesystem Requirements.** PingOne Autonomous Identity requires a shared filesystem accessible from the Spark main, Spark worker, analytics hosts, and application layer. The shared filesystem should be mounted at the same mount directory on all of those hosts. If the mount directory for the shared filesystem is different from the default, `/data`, update the `/autoid-config/vars.yml` file to point to the correct directories:

```
analytics_data_dir: /data
analytics_conf_dir: /data/conf
```

- **Architecture Requirements.** Make sure that the Spark main is on a separate node from the Spark workers.
- **Database Requirements.** Decide which database you are using: Apache Cassandra or MongoDB. The configuration procedure is slightly different for each database.
- **Deployment Best-Practice.** The example combines the Opensearch data and Opensearch Dashboards nodes. For best performance in production, dedicate a separate node to Opensearch, data nodes, and Opensearch Dashboards.
- **IPv4 Forwarding.** Many high-security environments run their CentOS-based systems with IPv4 forwarding disabled. However, Docker Swarm does not work with a disabled IPv4 forward setting. In such environments, make sure to enable IPv4 forwarding in the file `/etc/sysctl.conf`:

```
net.ipv4.ip_forward=1
```

Note

We recommend that your deployer team have someone with Cassandra expertise. This guide is not sufficient to troubleshoot any issues that may arise.

Set up the nodes

Set up each node as presented in [Set Up the Nodes](#).

Make sure you have sufficient storage for your particular deployment. For more information on sizing considerations, refer to [Deployment Planning Guide](#).

Important

For multinode deployments, there is a known issue with RHEL 8/CentOS Stream 8 and overlay network configurations. Refer to [Known Issues in 2022.11.0](#).

Install third-party components

Set up a machine with the required third-party software dependencies. Refer to: [Install third-party components](#).

Set up SSH on the deployer

1. On the deployer machine, run `ssh-keygen` to generate an RSA keypair, and then click **Enter**. You can use the default filename. Enter a password for protecting your private key.

```
ssh-keygen -t rsa -C "autoid"
```

The public and private rsa key pair is stored in `home-directory/.ssh/id_rsa` and `home-directory/.ssh/id_rsa.pub`.

2. Copy the SSH key to the `autoid-config` directory.

```
cp ~/.ssh/id_rsa ~/autoid-config
```

3. Change the privileges to the file.

```
chmod 400 ~/autoid-config/id_rsa
```

Prepare the tar file

Run the following steps on an Internet-connected host machine:

1. On the deployer machine, change to the installation directory.

```
cd ~/autoid-config/
```

2. Log in to the ForgeRock Google Cloud Registry using the registry key. The registry key is only available to ForgeRock PingOne Autonomous Identity customers. For specific instructions on obtaining the registry key, refer to [How To Configure Service Credentials \(Push Auth, Docker\) in Backstage](#).

```
docker login -u _json_key -p "$(cat autoid_registry_key.json)" https://gcr.io/forgerock-autoid
```

The following output is displayed:

```
Login Succeeded
```

3. Run the `create-template` command to generate the `deployer.sh` script wrapper. Note that the command sets the configuration directory on the target node to `/config`. Note that the `--user` parameter eliminates the need to use `sudo` while editing the hosts file and other configuration files.

```
docker run --user=$(id -u) -v ~/autoid-config:/config -it gcr.io/forgerock-autoid/deployer-pro:2022.11.8 create-template
```

4. Open the `~/autoid-config/vars.yml` file, set the `offline_mode` property to `true`, and then save the file.

```
offline_mode: true
```

5. Download the Docker images. This step downloads software dependencies needed for the deployment and places them in the `autoid-packages` directory.

```
sudo ./deployer.sh download-images
```

6. Create a tar file containing all of the PingOne Autonomous Identity binaries.

```
tar czf autoid-packages.tgz deployer.sh autoid-packages/*
```

7. Copy the `autoid-packages.tgz` to a portable hard drive.

Install PingOne Autonomous Identity air-gapped

Make sure you have the following prerequisites:

- IP address of machines running Opensearch, MongoDB, or Cassandra.
- The PingOne Autonomous Identity user should have permission to write to `/opt/autoid` on all machines
- To download the deployment images for the install, you still need your registry key to log into the ForgeRock Google Cloud Registry to download the artifacts.
- Make sure you have the proper Opensearch certificates with the exact names for both pem and JKS files copied to `~/autoid-config/certs/elastic`:
 - `esnode.pem`
 - `esnode-key.pem`
 - `root-ca.pem`
 - `elastic-client-keystore.jks`
 - `elastic-server-truststore.jks`
- Make sure you have the proper MongoDB certificates with exact names for both pem and JKS files copied to `~/autoid-config/certs/mongo`:
 - `mongo-client-keystore.jks`
 - `mongo-server-truststore.jks`
 - `mongodb.pem`
 - `rootCA.pem`
- Make sure you have the proper Cassandra certificates with exact names for both pem and JKS files copied to `~/autoid-config/certs/cassandra`:
 - `Zoran-cassandra-client-cer.pem`

- Zoran-cassandra-client-keystore.jks
- Zoran-cassandra-server-cer.pem
- zoran-cassandra-server-keystore.jks
- Zoran-cassandra-client-key.pem
- Zoran-cassandra-client-truststore.jks
- Zoran-cassandra-server-key.pem
- Zoran-cassandra-server-truststore.jks

Install PingOne Autonomous Identity:

1. Change to the directory.

```
cd autoid-config
```

2. Run the create-template command to generate the `deployer.sh` script wrapper and configuration files. Note that the command sets the configuration directory on the target node to `/config`. The `--user` parameter eliminates the need to use `sudo` while editing the hosts file and other configuration files.

```
docker run --user=$(id -u) -v ~/autoid-config:/config -it gcr.io/forgerock-autoid/deployer-pro:2022.11.3 create-template
```

3. Create a certificate directory for elastic.

```
mkdir -p autoid-config/certs/elastic
```

4. Copy the Opensearch certificates and JKS files to `autoid-config/certs/elastic`.

5. Create a certificate directory for MongoDB.

```
mkdir -p autoid-config/certs/mongo
```

6. Copy the MongoDB certificates and JKS files to `autoid-config/certs/mongo`.

7. Create a certificate directory for Cassandra.

```
mkdir -p autoid-config/certs/cassandra
```

8. Copy the Cassandra certificates and JKS files to `autoid-config/certs/cassandra`.

9. Update the `hosts` file with the IP addresses of the machines. The `hosts` file must include the IP addresses for Docker nodes, Spark main/livy, and the MongoDB master. While the deployer pro does not install or configure the MongoDB main server, the entry is required to run the MongoDB CLI to seed the PingOne Autonomous Identity schema.

```
[docker-managers]

[docker-workers]

[docker:children]
docker-managers
docker-workers

[spark-master-livy]

[cassandra-seeds]
#For replica sets, add the IPs of all Cassandra nodes

[mongo_master]
# Add the MongoDB main node in the cluster deployment
# For example: 10.142.15.248 mongodb_master=True

[odfe-master-node]
# Add only the main node in the cluster deployment
```

10. Update the `vars.yml` file:

1. Set `offline_mode` to `true`.
2. Set `db_driver_type` to `mongo` or `cassandra`.
3. Set `elastic_host`, `elastic_port`, and `elastic_user` properties.
4. Set `kibana_host`.
5. Set the Apache livy install directory.
6. Ensure the `elastic_user`, `elastic_port`, and `mongo_part` are correctly configured.
7. Update the `vault.yml` passwords for elastic and mongo to reflect your installation.
8. Set the `mongo_ldap` variable to `true` if you want PingOne Autonomous Identity to authenticate with Mongo DB, configured as LDAP.

 **Note**

The `mongo_ldap` variable only appears in fresh installs of 2022.11.0 and its upgrades (2022.11.1+). If you upgraded from a 2021.8.7 deployment, the variable is not available in your upgraded 2022.11.x deployment.

9. If you are using Cassandra, set the Cassandra-related parameters in the `vars.yml` file. Default values are:

```
cassandra:
  enable_ssl: "true"
  contact_points: 10.142.15.248 # comma separated values in case of replication set
  port: 9042
  username: zoran_dba
  cassandra_keystore_password: "Acc#1234"
  cassandra_truststore_password: "Acc#1234"
  ssl_client_key_file: "zoran-cassandra-client-key.pem"
  ssl_client_cert_file: "zoran-cassandra-client-cer.pem"
  ssl_ca_file: "zoran-cassandra-server-cer.pem"
  server_truststore_jks: "zoran-cassandra-server-truststore.jks"
  client_truststore_jks: "zoran-cassandra-client-truststore.jks"
  client_keystore_jks: "zoran-cassandra-client-keystore.jks"
```

11. Install Apache Livy.

- The official release of Apache Livy does not support Apache Spark 3.3.1 or 3.3.2. ForgeRock has re-compiled and packaged Apache Livy to work with Apache Spark 3.3.1 hadoop 3 and Apache Spark 3.3.2 hadoop 3. Use the zip file located at [autoid-config/apache-livy/apache-livy-0.8.0-incubating-SNAPSHOT-bin.zip](#) to install Apache Livy on the Spark-Livy machine.
- For Livy configuration, refer to <https://livy.apache.org/get-started/>.

12. On the Spark-Livy machine, run the following commands to install the python package dependencies:

1. Change to the `/opt/autoid` directory:

```
cd /opt/autoid
```

2. Create a `requirements.txt` file with the following content:

```
six==1.11
certifi==2019.11.28
python-dateutil==2.8.1
jsonschema==3.2.0
cassandra-driver
numpy==1.19.5
pyarrow==0.16.0
wrapt==1.11.0
PyYAML==5.4
requests==2.31.0
urllib3==1.26.18
pymongo
pandas==1.0.5
tabulate
openpyxl
```

3. Install the requirements file:

```
pip3 install -r requirements.txt
```

13. Make sure that the `/opt/autoid` directory exists and that it is both readable and writable.

14. Run the deployer script:

```
./deployer.sh run
```

15. On the Spark-Livy machine, run the following commands to install the Python egg file:

1. Install the egg file:

```
cd /opt/autoid/eggs  
pip3.10 install autoid_analytics-2021.3-py3-none-any.whl
```

2. Source the `.bashrc` file:

```
source ~/.bashrc
```

3. Restart Spark and Livy.

```
./spark/sbin/stop-all.sh  
./livy/bin/livy-server stop  
  
./spark/sbin/start-all.sh  
./livy/bin/livy-server start
```

Set the replication factor

Once Cassandra has been deployed, you need to set the replication factor to match the number of nodes on your system. This ensures that each record is stored in each of the nodes. In the event one node is lost, the remaining node can continue to serve content even if the cluster itself is running with reduced redundancy.

Refer to [Set the Replication Factor for Non-Airgap](#).

Resolve Hostname

After installing PingOne Autonomous Identity, set up the hostname resolution for your deployment.

1. Configure your DNS servers to access PingOne Autonomous Identity dashboard on the target node. The following domain names must resolve to the IP address of the target node:

```
<target-environment>-ui.<domain-name>
```

2. If DNS cannot resolve target node hostname, edit it locally on the machine that you want to access PingOne Autonomous Identity using a browser.

Open a text editor and add an entry in the `/etc/hosts` (Linux/Unix) file or `C:\Windows\System32\drivers\etc\hosts` (Windows) for the target node.

For multi-node, use the Docker Manager node as your target.

```
<Docker Mgr Node Public IP Address> <target-environment>-ui.<domain-name>
```

For example:

```
<IP Address> autoid-ui.forgerock.com
```

3. If you set up a custom domain name and target environment, add the entries in `/etc/hosts` . For example:


```
<IP Address> myid-ui.abc.com
```

For more information on customizing your domain name, see [Customize Domains](#).

Access the Dashboard

Access the PingOne Autonomous Identity console UI:

1. Open a browser. If you set up your own url, use it for your login.

```
https://autoid-ui.forgerock.com/ 
```

2. Log in as a test user.

```
test user: bob.rodgers@forgerock.com  
password: <password>
```

Start the Analytics

If the previous installation steps all succeeded, you must now prepare your data's entity definitions, data sources, and attribute mappings prior to running your analytics jobs. These step are required and are critical for a successful analytics process.

For more information, refer to [Set Entity Definitions](#).

Upgrade PingOne Autonomous Identity

PingOne Autonomous Identity provides an upgrade command to update your core software to the latest version while migrating your data.

Upgrade Considerations

- **Database Systems are the Same.** If your current database is Apache Cassandra, you cannot upgrade to a MongoDB-based system. You will need to run a clean installation with the new version.
- **Host IPs should be the Same.** Host IP addresses must be the same for existing components. You must update the `~/autoid-config/hosts` file by adding the IP addresses for the Elasticsearch entries. Refer to the instructions below.
- **Registry Key Required.** To download the deployment images for the upgrade, you still need your registry key to log into the ForgeRock Google Cloud Registry. Copy your registry key from your previous build to your new upgrade.

Important

Make sure to test the upgrade on a staging or QA server before running it in production.

Upgrade Paths

The upgrade assumes the following upgrade paths depends on your current deployment version. The preferred upgrade path is to the latest patch release.

Important

Clean installations of PingOne Autonomous Identity 2022.11.x (2022.11.0–2022.11.7) to 2022.11.8 use the new `deployer pro` script. Upgrades from version 2021.8.7 to 2022.11.x to 2022.11.8 use the older `deployer` script. The upgrade procedures differ slightly between the `deployer pro` and `deployer` versions, primarily in certificates directory creation (`deployer` versions) and using the proper image name during the `create-template` command (`deployer pro` and `deployer` versions).

The following chart summarizes these upgrade paths:

Table 1: Upgrade Paths

Version	Upgrade To	Refer to
2022.11.x (deployer-pro)	2022.11.8 (deployer-pro)	<ul style="list-style-type: none">• Upgrade from PingOne Autonomous Identity 2022.11.x to 2022.11.8 using deployer pro
2022.11.x Air-Gapped (deployer-pro)	2022.11.8 Air-Gapped (deployer-pro)	<ul style="list-style-type: none">• Upgrade from PingOne Autonomous Identity 2022.11.x to 2022.11.8 Air-Gapped using deployer pro
2022.11.0 (deployer)	2022.11.8 (deployer)	<ul style="list-style-type: none">• Upgrade from PingOne Autonomous Identity 2022.11.x to 2022.11.8 using the deployer

Version	Upgrade To	Refer to
2022.11.0 Air-Gapped (deployer)	2022.11.8 Air Gapped (deployer)	<ul style="list-style-type: none"> • Upgrade from PingOne Autonomous Identity 2022.11.x to 2022.11.8 Air-Gapped using the deployer

Upgrade from PingOne Autonomous Identity 2022.11.x to 2022.11.8 using deployer pro

The following instructions are for upgrading from PingOne Autonomous Identity version 2022.11.0–2022.11.7 to the latest version **2022.11.8** in non air-gapped deployments using the **deployer pro**.



Important

The following steps assume you ran a fresh install of PingOne Autonomous Identity 2022.11.x, which uses deployer pro. Make sure you have upgraded your [third-party software](#) packages to the supported versions prior to upgrade.

Upgrade from 2022.11.x to 2022.11.8 (Non Air-Gap) using deployer pro:

1. Start on the target server, and back up your `/data/conf` configuration file. The upgrade overwrites this file when updating, so you must restore this file after running the upgrade.

```
sudo mv /data/conf ~/backup-data-conf-2022.11.x
```

2. Next, if you changed any analytic settings on your deployment, make note of your configuration, so that you can replicate those settings on the upgraded server. Log in to PingOne Autonomous Identity, navigate to **Administration > Analytic Settings**, and record your settings.
3. On the deployer machine, back up the 2022.11.x `~/autoid-config` directory or move it to another location.

```
mv ~/autoid-config ~/backup-2022.11.x
```

4. Create a new `~/autoid-config` directory.

```
mkdir ~/autoid-config
```

5. Copy your `autoid_registry_key.json`, `ansible.cfg`, and `vault.yml` files from your backup directory to `~/autoid-config`. If your `vault.yml` file is encrypted, copy the `.autoid_vault_password` file to `~/autoid-config`.
6. Set up your certificate directories for Opensearch, MongoDB, or Cassandra for the deployer:

1. Create a certificate directory Opensearch:

```
mkdir -p autoid-config/certs/elastic
```

2. Copy the Opensearch certificates and JKS files to `autoid-config/certs/elastic`.

3. Create a certificate directory for MongoDB (if you use MongoDB):

```
mkdir -p autoid-config/certs/mongo
```

4. Copy the MongoDB certificates and JKS files to `autoid-config/certs/mongo`.

5. Create a certificate directory for Cassandra (if you use Cassandra):

```
mkdir -p autoid-config/certs/cassandra
```

6. Copy the Cassandra certificates and JKS files to `autoid-config/certs/cassandra`.

7. Copy your original SSH key into the new directory.

```
cp ~/.ssh/id_rsa ~/autoid-config
```

8. Change the permission on the SSH key.

```
chmod 400 ~/autoid-config/id_rsa
```

9. Check if you can successfully SSH to the target server.

```
ssh autoid@<Target-IP-Address>

Last login: Mon Dec 04 12:20:18 2023
```

10. On the deployer node, change to the `~/autoid-config` directory.

```
cd ~/autoid-config
```

11. Log in to the ForgeRock Google Cloud Registry using the registry key. The registry key is only available to ForgeRock PingOne Autonomous Identity customers. For specific instructions on obtaining the registry key, see [How To Configure Service Credentials \(Push Auth, Docker\) in Backstage](#).

```
docker login -u _json_key -p "$(cat autoid_registry_key.json)" https://gcr.io/forgerock-autoid
```

You should see:

```
Login Succeeded
```

12. Run the `create-template` command to generate the `deployer.sh` script wrapper and configuration files. Note that the command sets the configuration directory on the target node to `/config`. The `--user` parameter eliminates the need to use `sudo` while editing the hosts file and other configuration files.

```
docker run --user=$(id -u) -v ~/autoid-config:/config \
-it gcr.io/forgerock-autoid/deployer-pro:2022.11.8 create-template
```

13. Configure your upgraded system by editing the `~/autoid-config/vars.yml`, `~/autoid-config/hosts`, and `~/autoid-config/vault.yml` files on the deployer machine.



Important

You must keep your configuration settings consistent from one system to another.

14. Stop the stack.



Note

If you are upgrading a multi-node deployment, run this command on the Docker Manager node.

```
docker stack rm configuration-service consul-server consul-client nginx jas swagger-ui ui api
notebook
```

You should see:

```
Removing service configuration-service_configuration-service
Removing service consul-server_consul-server
Removing service consul-client_consul-client
Removing service nginx_nginx
Removing service jas_jasnode
Removing service swagger-ui_swagger-ui
Removing service ui_zoran-ui
Removing service api_zoran-api
Nothing found in stack: notebook
```

15. Prune old Docker images before running the upgrade command:

1. Get all of the Docker images:

```
docker images
```

2. Identify the images that are PingOne Autonomous Identity-related. They start with the URL of the ForgeRock Google Cloud Registry (ForgeRock GCR). For example:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
<ForgeRock GCR>/ci/develop/deployer	650879186	075481cea4c2	2 hours ago	823MB
<ForgeRock GCR>/ci/develop/offline-packages	650879186	e1a90f389ccc	2 hours ago	3.03GB
<ForgeRock GCR>/ci/develop/zoran-ui	650879186	bd303a28b5df	2 hours ago	35.3MB
<ForgeRock GCR>/ci/develop/zoran-api	650879186	114d1aca5b0a	2 hours ago	421MB
<ForgeRock GCR>/ci/develop/nginx	650879186	43b410661269	2 hours ago	16.7MB
<ForgeRock GCR>/ci/develop/jas	650879186	2821e5c365d8	2 hours ago	491MB

3. Remove the old images using the `docker rmi` command. For example:

```
docker rmi -f <image ID>

Example:
docker rmi -f 075481cea4c2
```

4. Repeat the previous command to remove all of the PingOne Autonomous Identity-related Docker images.

16. For multinode deployments, run the following on the Docker Worker node:

```
docker swarm leave
```

17. Enter `exit` to end your SSH session.

18. From the deployer, restart Docker command:

```
sudo systemctl restart docker
```

19. Download the images. This step downloads software dependencies needed for the deployment and places them in the `autoid-packages` directory. Make sure you are in the `~/autoid-config` directory.

```
./deployer.sh download-images
```

20. On the Spark-Livy machine, run the following commands to install the python package dependencies:

1. Change to the `/opt/autoid` directory:

```
cd /opt/autoid
```

2. Create a `requirements.txt` file with the following content:

```
six==1.11
certifi==2019.11.28
python-dateutil==2.8.1
jsonschema==3.2.0
cassandra-driver
numpy==1.22.0
pyarrow==6.0.1
wrapt==1.11.0
PyYAML==6.0
requests==2.31.0
urllib3==1.26.18
pymongo
pandas==1.3.5
tabulate
openpyxl
wheel
cython
```

3. Install the requirements file:

```
pip3 install -r requirements.txt
```

21. Run the upgrade:

```
./deployer.sh upgrade
```

22. On the Spark-Livy machine, run the following commands to install the Python wheel distribution:

1. Install the wheel file:

```
cd /opt/autoid/eggs
pip3.10 install autoid_analytics-2021.3-py3-none-any.whl
```

2. Source the `.bashrc` file:

```
source ~/.bashrc
```

3. Restart Spark and Livy.

```
./spark/sbin/stop-all.sh
./livy/bin/livy-server stop

./spark/sbin/start-all.sh
./livy/bin/livy-server start
```

23. SSH to the target server.

24. On the target server, restore your `/data/conf` configuration data file from your previous installation.

```
sudo mv ~/backup-data-conf-2022.11.x /data/conf
```

25. Re-apply your analytics settings to your upgraded server if you made changes on your previous PingOne Autonomous Identity machine. Log in to PingOne Autonomous Identity, navigate to **Administration > Analytics Settings**, and edit your changes.
26. Log out, and then log back in to PingOne Autonomous Identity.

You have successfully upgraded your PingOne Autonomous Identity server to 2022.11.8.

Upgrade from PingOne Autonomous Identity 2022.11.x to 2022.11.8 Air-Gapped using deployer pro

The following instructions are for upgrading from PingOne Autonomous Identity version 2022.11.0–2022.11.7 on air-gapped deployments using the **deployer pro**.



Important

The following steps assume you ran a fresh install of PingOne Autonomous Identity 2022.11.x, which uses **deployer pro**. Make sure you have upgraded your [third-party software](#) packages to the supported versions prior to upgrade.

Upgrade from 2022.11.x to 2022.11.8 Air-Gapped using deployer pro:

1. Start on the target server, and back up your `/data/conf` configuration file. The upgrade overwrites this file when updating, so you must restore this file after running the upgrade.

```
sudo mv /data/conf ~/backup-data-conf-2022.11.x
```

2. Next, if you changed any analytic settings on your deployment, make note of your configuration, so that you can replicate those settings on the upgraded server. Log in to PingOne Autonomous Identity, navigate to **Administration > Analytic Settings**, and record your settings.
3. On the deployer machine, back up the 2022.11.x `~/autoid-config` directory or move it to another location.

```
mv ~/autoid-config ~/backup-2022.11.x
```

4. Create a new `~/autoid-config` directory.

```
mkdir ~/autoid-config
```

5. Copy your `autoid_registry_key.json`, `ansible.cfg`, and `vault.yml` files from your backup directory to `~/autoid-config`. If your `vault.yml` file is encrypted, copy the `.autoid_vault_password` file to `~/autoid-config`.
6. Set up your certificate directories for Opensearch, MongoDB, or Cassandra for the deployer:

1. Create a certificate directory Opensearch:

```
mkdir -p autoid-config/certs/elastic
```

2. Copy the Opensearch certificates and JKS files to `autoid-config/certs/elastic`.

3. Create a certificate directory for MongoDB (if you use MongoDB):

```
mkdir -p autoid-config/certs/mongo
```

4. Copy the MongoDB certificates and JKS files to `autoid-config/certs/mongo`.

5. Create a certificate directory for Cassandra (if you use Cassandra):

```
mkdir -p autoid-config/certs/cassandra
```

6. Copy the Cassandra certificates and JKS files to `autoid-config/certs/cassandra`.

7. Copy your original SSH key into the new directory.

```
cp ~/.ssh/id_rsa ~/autoid-config
```

8. Change the permission on the SSH key.

```
chmod 400 ~/autoid-config/id_rsa
```

9. On the deployer node, change to the `~/autoid-config` directory.

```
cd ~/autoid-config
```

10. Log in to the ForgeRock Google Cloud Registry using the registry key. The registry key is only available to ForgeRock PingOne Autonomous Identity customers. For specific instructions on obtaining the registry key, see [How To Configure Service Credentials \(Push Auth, Docker\) in Backstage](#).

```
docker login -u _json_key -p "$(cat autoid_registry_key.json)" https://gcr.io/forgerock-autoid
```

You should see:

```
Login Succeeded
```

11. Run the `create-template` command to generate the `deployer.sh` script wrapper and configuration files. Note that the command sets the configuration directory on the target node to `/config`. The `--user` parameter eliminates the need to use `sudo` while editing the hosts file and other configuration files.

```
docker run --user=$(id -u) -v ~/autoid-config:/config \
-it gcr.io/forgerock-autoid/deployer-pro:2022.11.8 create-template
```

12. Configure your upgraded system by editing the `~/autoid-config/vars.yml`, `~/autoid-config/hosts`, and `~/autoid-config/vault.yml` files on the deployer machine.



Important

You must keep your configuration settings consistent from one system to another.

13. Download the images. This step downloads software dependencies needed for the deployment and places them in the `autoid-packages` directory. Make sure you are in the `~/autoid-config` directory.

```
./deployer.sh download-images
```

14. On the Spark-Livy machine, run the following commands to install the python package dependencies:

1. Change to the `/opt/autoid` directory:

```
cd /opt/autoid
```

2. Create a `requirements.txt` file with the following content:

```
six==1.11
certifi==2019.11.28
python-dateutil==2.8.1
jsonschema==3.2.0
cassandra-driver
numpy==1.22.0
pyarrow==6.0.1
wrapt==1.11.0
PyYAML==6.0
requests==2.31.0
urllib3==1.26.18
pymongo
pandas==1.3.5
tabulate
openpyxl
wheel
cython
```

3. Install the requirements file:

```
pip3 install -r requirements.txt
```

15. Stop the stack.

Note

If you are upgrading a multi-node deployment, run this command on the Docker Manager node.

```
docker stack rm configuration-service consul-server consul-client nginx jas swagger-ui ui api
notebook
```

You should see:

```
Removing service configuration-service_configuration-service
Removing service consul-server_consul-server
Removing service consul-client_consul-client
Removing service nginx_nginx
Removing service jas_jasnode
Removing service swagger-ui_swagger-ui
Removing service ui_zoran-ui
Removing service api_zoran-api
Nothing found in stack: notebook
```

16. Prune old Docker images before running the upgrade command:

1. Get all of the Docker images:

```
docker images
```

2. Identify the images that are PingOne Autonomous Identity-related. They start with the URL of the ForgeRock Google Cloud Registry (ForgeRock GCR). For example:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
<ForgeRock GCR>/ci/develop/deployer	650879186	075481cea4c2	2 hours ago	823MB
<ForgeRock GCR>/ci/develop/offline-packages	650879186	e1a90f389ccc	2 hours ago	3.03GB
<ForgeRock GCR>/ci/develop/zoran-ui	650879186	bd303a28b5df	2 hours ago	35.3MB
<ForgeRock GCR>/ci/develop/zoran-api	650879186	114d1aca5b0a	2 hours ago	421MB
<ForgeRock GCR>/ci/develop/nginx	650879186	43b410661269	2 hours ago	16.7MB
<ForgeRock GCR>/ci/develop/jas	650879186	2821e5c365d8	2 hours ago	491MB

3. Remove the old images using the **docker rmi** command. For example:

```
docker rmi -f <image ID>
```

Example:

```
docker rmi -f 075481cea4c2
```

17. For multinode deployments, run the following on the Docker Worker node:

```
docker swarm leave
```

18. From the deployer, restart Docker:

```
sudo systemctl restart docker
```

19. Create a tar file containing all of the Autonomous Identity binaries.

```
tar czf autoid-packages.tgz deployer.sh autoid-packages/*
```

20. Copy the `autoid-packages.tgz`, `deployer.sh`, and SSH key (`id_rsa`) to a portable hard drive.

21. On the air-gapped target machine, backup your previous `~/autoid-config` directory, and then create a new `~/autoid-config` directory.

```
mkdir ~/autoid-config
```

22. Copy the `autoid-package.tgz` tar file, `deployer.sh`, and SSH key from the portable storage device to the `/autoid-config` folder.

23. Unpack the tar file.

```
tar xf autoid-packages.tgz -C ~/autoid-config
```

24. Set up your certificate directories for Opensearch, MongoDB, or Cassandra for the deployer:

1. Create a certificate directory Opensearch:

```
mkdir -p autoid-config/certs/elastic
```

2. Copy the Opensearch certificates and JKS files to `autoid-config/certs/elastic`.

3. Create a certificate directory for MongoDB (if you use MongoDB):

```
mkdir -p autoid-config/certs/mongo
```

4. Copy the MongoDB certificates and JKS files to `autoid-config/certs/mongo`.

5. Create a certificate directory for Cassandra (if you use Cassandra):

```
mkdir -p autoid-config/certs/cassandra
```

6. Copy the Cassandra certificates and JKS files to `autoid-config/certs/cassandra`.

25. Copy the SSH key to the `~/autoid-config` directory.

26. Change the privileges to the file.

```
chmod 400 ~/autoid-config/id_rsa
```

27. Change to the configuration directory.

```
cd ~/autoid-config
```

28. Import the deployer image.

```
./deployer.sh import-deployer
```

You should see:

```
...
db631c8b06ee: Loading layer [=====>]    2.56kB/2.56kB
2d62082e3327: Loading layer [=====>]    753.2kB/753.2kB
Loaded image: <ForgeRock Google cloud registry URL>/deployer:2022.11.8
```

29. Create the configuration template using the `create-template` command. This command creates the configuration files: `ansible.cfg`, `vars.yml`, `vault.yml` and `hosts`.

```
./deployer.sh create-template
```

You should see:

```
Config template is copied to host machine directory mapped to /config
```

30. Configure your upgraded system by editing the `~/autoid-config/vars.yml`, `~/autoid-config/hosts`, and `~/autoid-config/vault.yml` files on the deployer machine.



Important

You must keep your configuration settings consistent from one system to another.

31. Run the upgrade:

```
./deployer.sh upgrade
```

32. On the Spark-Livy machine, run the following commands to install the Python wheel distribution:

1. Install the wheel file:

```
cd /opt/autoid/eggs
pip3.10 install autoid_analytics-2021.3-py3-none-any.whl
```

2. Source the `.bashrc` file:

```
source ~/.bashrc
```

3. Restart Spark and Livy.

```
./spark/sbin/stop-all.sh
./livy/bin/livy-server stop

./spark/sbin/start-all.sh
./livy/bin/livy-server start
```

33. SSH to the target server.

34. On the target server, restore your `/data/conf` configuration data file from your previous installation.

```
sudo mv ~/backup-data-conf-2022.11.x /data/conf
```

35. Re-apply your analytics settings to your upgraded server if you made changes on your previous PingOne Autonomous Identity machine. Log in to PingOne Autonomous Identity, navigate to **Administration > Analytics Settings**, and edit your changes.

36. Log out, and then log back in to PingOne Autonomous Identity.

You have successfully upgraded your PingOne Autonomous Identity server to 2022.11.8.

Upgrade from PingOne Autonomous Identity 2022.11.x to 2022.11.8 using the deployer

The following instructions are for upgrading from PingOne Autonomous Identity version 2022.11.0–2022.11.7 to the latest version **2022.11.8** in non air-gapped deployments using the **deployer**.

Important

If you upgraded from any PingOne Autonomous Identity version 2021.8.7 or earlier to version 2022.11.x, then you are using the deployer.

Upgrade from 2022.11.x to 2022.11.8 (Non Air-Gap) using deployer:

1. Start on the target server, and back up your `/data/conf` configuration file. The upgrade overwrites this file when updating, so you must restore this file after running the upgrade.

```
sudo mv /data/conf ~/backup-data-conf-2022.11.x
```

2. Next, if you changed any analytic settings on your deployment, make note of your configuration, so that you can replicate those settings on the upgraded server. Log in to PingOne Autonomous Identity, navigate to **Administration > Analytic Settings**, and record your settings.
3. On the deployer machine, back up the 2022.11.x `~/autoid-config` directory or move it to another location.

```
mv ~/autoid-config ~/backup-2022.11.x
```

4. Create a new `~/autoid-config` directory.

```
mkdir ~/autoid-config
```

5. Copy your `autoid_registry_key.json` from your backup directory to `~/autoid-config`.
6. Copy your original SSH key into the new directory.

```
cp ~/.ssh/id_rsa ~/autoid-config
```

7. Change the permission on the SSH key.

```
chmod 400 ~/autoid-config/id_rsa
```

8. Check if you can successfully SSH to the target server.

```
ssh autoid@<Target-IP-Address>

Last login: Mon Dec 04 12:20:18 2023
```

9. On the deployer node, change to the `~/autoid-config` directory.

```
cd ~/autoid-config
```

10. Log in to the ForgeRock Google Cloud Registry using the registry key. The registry key is only available to ForgeRock PingOne Autonomous Identity customers. For specific instructions on obtaining the registry key, see [How To Configure Service Credentials \(Push Auth, Docker\) in Backstage](#).

```
docker login -u _json_key -p "$(cat autoid_registry_key.json)" https://gcr.io/forgerock-autoid
```

You should see:

```
Login Succeeded
```

11. Run the `create-template` command to generate the `deployer.sh` script wrapper and configuration files. Note that the command sets the configuration directory on the target node to `/config`. The `--user` parameter eliminates the need to use `sudo` while editing the hosts file and other configuration files.

```
docker run --user=$(id -u) -v ~/autoid-config:/config \
-it gcr.io/forgerock-autoid/deployer:2022.11.8 create-template
```

12. Configure your upgraded system by editing the `~/autoid-config/vars.yml`, `~/autoid-config/hosts`, and `~/autoid-config/vault.yml` files on the deployer machine.



Important

You must keep your configuration settings consistent from one system to another.

13. Stop the stack.



Note

If you are upgrading a multi-node deployment, run this command on the Docker Manager node.

```
docker stack rm configuration-service consul-server consul-client nginx jas swagger-ui ui api
notebook
```

You should see:

```
Removing service configuration-service_configuration-service
Removing service consul-server_consul-server
Removing service consul-client_consul-client
Removing service nginx_nginx
Removing service jas_jasnode
Removing service swagger-ui_swagger-ui
Removing service ui_zoran-ui
Removing service api_zoran-api
Nothing found in stack: notebook
```

14. Prune old Docker images before running the upgrade command:

1. Get all of the Docker images:

```
docker images
```

2. Identify the images that are PingOne Autonomous Identity-related. They start with the URL of the ForgeRock Google cloud registry (ForgeRock GCR). For example:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
<ForgeRock GCR>/ci/develop/deployer	650879186	075481cea4c2	2 hours ago	823MB
<ForgeRock GCR>/ci/develop/offline-packages	650879186	e1a90f389ccc	2 hours ago	3.03GB
<ForgeRock GCR>/ci/develop/zoran-ui	650879186	bd303a28b5df	2 hours ago	35.3MB
<ForgeRock GCR>/ci/develop/zoran-api	650879186	114d1aca5b0a	2 hours ago	421MB
<ForgeRock GCR>/ci/develop/nginx	650879186	43b410661269	2 hours ago	16.7MB
<ForgeRock GCR>/ci/develop/jas	650879186	2821e5c365d8	2 hours ago	491MB

3. Remove the old images using the `docker rmi` command. For example:

```
docker rmi -f <image ID>

Example:
docker rmi -f 075481cea4c2
```

4. Repeat the previous command to remove all of the PingOne Autonomous Identity-related Docker images.

15. For multinode deployments, run the following on the Docker Worker node:

```
docker swarm leave
```

16. Enter `exit` to end your SSH session.

17. From the deployer, restart Docker command:

```
sudo systemctl restart docker
```

18. Download the images. This step downloads software dependencies needed for the deployment and places them in the `autoid-packages` directory. Make sure you are in the `/autoid-config` directory.

```
./deployer.sh download-images
```

19. Run the upgrade:

```
./deployer.sh upgrade
```

20. SSH to the target server.

21. On the target server, restore your `/data/conf` configuration data file from your previous installation.

```
sudo mv ~/backup-data-conf-2022.11.x /data/conf
```

22. Re-apply your analytics settings to your upgraded server if you made changes on your previous PingOne Autonomous Identity machine. Log in to PingOne Autonomous Identity, navigate to **Administration > Analytics Settings**, and edit your changes.

23. Log out, and then log back in to PingOne Autonomous Identity.

You have successfully upgraded your PingOne Autonomous Identity server to 2022.11.8.

Upgrade from PingOne Autonomous Identity 2022.11.x to 2022.11.8 Air-Gapped using the deployer

The following instructions are for upgrading from PingOne Autonomous Identity version 2022.11.0–2022.11.7 to the latest version **2022.11.8** on air-gapped deployments using the **deployer**.

Upgrade from 2022.11.x to 2022.11.8 Air-Gapped using deployer:

1. Start on the target server, and back up your `/data/conf` configuration file. The upgrade overwrites this file when updating, so you must restore this file after running the upgrade.

```
sudo mv /data/conf ~/backup-data-conf-2022.11.x
```

2. Next, if you changed any analytic settings on your deployment, make note of your configuration, so that you can replicate those settings on the upgraded server. Log in to PingOne Autonomous Identity, navigate to **Administration > Analytic Settings**, and record your settings.

3. On the deployer machine, back up the 2022.11.x `~/autoid-config` directory or move it to another location.

```
mv ~/autoid-config ~/backup-2022.11.x
```

4. Create a new `~/autoid-config` directory.

```
mkdir ~/autoid-config
```

5. Copy your `autoid_registry_key.json` from your backup directory to `~/autoid-config`.

6. Copy your original SSH key into the new directory.

```
cp ~/.ssh/id_rsa ~/autoid-config
```

7. Change the permission on the SSH key.

```
chmod 400 ~/autoid-config/id_rsa
```

8. On the deployer node, change to the `~/autoid-config` directory.

```
cd ~/autoid-config
```

9. Log in to the ForgeRock Google Cloud Registry using the registry key. The registry key is only available to ForgeRock PingOne Autonomous Identity customers. For specific instructions on obtaining the registry key, see [How To Configure Service Credentials \(Push Auth, Docker\) in Backstage](#).

```
docker login -u _json_key -p "$(cat autoid_registry_key.json)" https://gcr.io/forgerock-autoid
```

You should see:

```
Login Succeeded
```

10. Run the `create-template` command to generate the `deployer.sh` script wrapper and configuration files. Note that the command sets the configuration directory on the target node to `/config`. The `--user` parameter eliminates the need to use `sudo` while editing the hosts file and other configuration files.

```
docker run --user=$(id -u) -v ~/autoid-config:/config \
-it gcr.io/forgerock-autoid/deployer:2022.11.8 create-template
```

11. Configure your upgraded system by editing the `~/autoid-config/vars.yml`, `~/autoid-config/hosts`, and `~/autoid-config/vault.yml` files on the deployer machine.



Important

You must keep your configuration settings consistent from one system to another.

12. Download the images. This step downloads software dependencies needed for the deployment and places them in the `autoid-packages` directory. Make sure you are in the `~/autoid-config` directory.

```
./deployer.sh download-images
```

13. Stop the stack.



Note

If you are upgrading a multi-node deployment, run this command on the Docker Manager node.

```
docker stack rm configuration-service consul-server consul-client nginx jas swagger-ui ui api
notebook
```

You should see:

```
Removing service configuration-service_configuration-service
Removing service consul-server_consul-server
Removing service consul-client_consul-client
Removing service nginx_nginx
Removing service jas_jasnode
Removing service swagger-ui_swagger-ui
Removing service ui_zoran-ui
Removing service api_zoran-api
Nothing found in stack: notebook
```

14. Prune old Docker images before running the upgrade command:

1. Get all of the Docker images:

```
docker images
```

2. Identify the images that are PingOne Autonomous Identity-related. They start with the URL of the ForgeRock Google Cloud Registry (ForgeRock GCR). For example:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
<ForgeRock GCR>/ci/develop/deployer	650879186	075481cea4c2	2 hours ago	823MB
<ForgeRock GCR>/ci/develop/offline-packages	650879186	e1a90f389ccc	2 hours ago	3.03GB
<ForgeRock GCR>/ci/develop/zoran-ui	650879186	bd303a28b5df	2 hours ago	35.3MB
<ForgeRock GCR>/ci/develop/zoran-api	650879186	114d1aca5b0a	2 hours ago	421MB
<ForgeRock GCR>/ci/develop/nginx	650879186	43b410661269	2 hours ago	16.7MB
<ForgeRock GCR>/ci/develop/jas	650879186	2821e5c365d8	2 hours ago	491MB

3. Remove the old images using the `docker rmi` command. For example:

```
docker rmi -f <image ID>
```

Example:

```
docker rmi -f 075481cea4c2
```

15. For multinode deployments, run the following on the Docker Worker node:

```
docker swarm leave
```

16. From the deployer, restart Docker:

```
sudo systemctl restart docker
```

17. Create a tar file containing all of the Autonomous Identity binaries.

```
tar czf autoid-packages.tgz deployer.sh autoid-packages/*
```

18. Copy the `autoid-packages.tgz`, `deployer.sh`, and SSH key (`id_rsa`) to a portable hard drive.

19. On the air-gapped target machine, backup your previous `~/autoid-config` directory, and then create a new `~/autoid-config` directory.

```
mkdir ~/autoid-config
```

20. Copy the `autoid-package.tgz` tar file, `deployer.sh`, and SSH key from the portable storage device to the `/autoid-config` folder.

21. Unpack the tar file.

```
tar xf autoid-packages.tgz -C ~/autoid-config
```

22. Copy the SSH key to the `~/autoid-config` directory.

23. Change the privileges to the file.

```
chmod 400 ~/autoid-config/id_rsa
```

24. Change to the configuration directory.

```
cd ~/autoid-config
```

25. Import the deployer image.

```
./deployer.sh import-deployer
```

You should see:

```
...
db631c8b06ee: Loading layer [=====>] 2.56kB/2.56kB
2d62082e3327: Loading layer [=====>] 753.2kB/753.2kB
Loaded image: https://gcr.io/forgerock-autoid/deployer:2022.11.8
```

26. Create the configuration template using the `create-template` command. This command creates the configuration files: `ansible.cfg`, `vars.yml`, `vault.yml` and `hosts`.

```
./deployer.sh create-template
```

You should see:

```
Config template is copied to host machine directory mapped to /config
```

27. Configure your upgraded system by editing the `~/autoid-config/vars.yml`, `~/autoid-config/hosts`, and `~/autoid-config/vault.yml` files on the deployer machine.



Important

You must keep your configuration settings consistent from one system to another.

28. Run the upgrade:

```
./deployer.sh upgrade
```

29. On the target server, restore your `/data/conf` configuration data file from your previous installation.

```
sudo mv ~/backup-data-conf-2022.11.x /data/conf
```

30. Re-apply your analytics settings to your upgraded server if you made changes on your previous PingOne Autonomous Identity machine. Log in to PingOne Autonomous Identity, navigate to **Administration > Analytics Settings**, and edit your changes.

31. Log out, and then log back in to PingOne Autonomous Identity.

You have successfully upgraded your PingOne Autonomous Identity server to 2022.11.8.

Appendix A: PingOne Autonomous Identity ports

The PingOne Autonomous Identity deployment uses the following ports. The Docker deployer machine opens the ports in the firewall on the target node. If you are using cloud virtual machines, you need to open these ports on the virtual cloud network.

PingOne Autonomous Identity uses the following ports:

PingOne Autonomous Identity ports

Port	Protocol	Machine	Source	Description
2377	TCP	Docker managers	Docker managers and nodes	Communication between the nodes of a Docker swarm cluster
7946	TCP/UDP	Docker managers and workers	Docker managers and workers	Communication among nodes for container network discovery
4789	UDP	Docker managers and workers	Docker managers and workers	Overlay network traffic
7001	TCP	Cassandra	Cassandra nodes	Internode communication
9042	TCP	Cassandra	Cassandra nodes, Docker managers and nodes	CQL native transport
27017	TCP	MongoDB	MongoDB nodes, Docker managers and nodes	Default ports for mongod and mongos instances
9200	TCP	Open Distro for Elasticsearch	Docker managers and nodes	Elasticsearch REST API endpoint
7077	TCP	Spark master	Spark workers	Spark master internode communication port
40040-40045	TCP	Spark Master	Spark Workers	Spark driver ports for Spark workers to callback
443	TCP	Docker managers	User's browsers/API clients	Port to access the dashboard and API

Port	Protocol	Machine	Source	Description
10081	TCP	Docker managers	User's browsers/API clients	Port for the JAS service.

Appendix B: vars.yml

PingOne Autonomous Identity has a configuration file where you can set the analytics data and configuration directories, private IP address mapping, LDAP/SSO options, and session duration during installation. The file is created when running the `create-template` command during the installation and is located in the `/autoid-config` directory.

The file is as follows:

```

ai_product: auto-id                # Product name
domain_name: forgerock.com         # Default domain name
target_environment: autoid         # Default namespace
analytics_data_dir: /data         # Default data directory
analytics_conf_dir: /data/conf    # Default config directory for analytics

# set to true for air-gap installation
offline_mode: false

# choose the DB Type : cassandra| mongo
db_driver_type: cassandra

# Needed only if private and public IP address of
# target nodes are different. If cloud VMs the private
# is different than the IP address (public ip) used for
# SSH. Private IP addresses are used by various services
# to reach other services in the cluster
# Example:
# private_ip_address_mapping:
#   35.223.33.21: "10.128.0.5"
#   108.59.83.132: "10.128.0.37"
#   ...
private_ip_address_mapping:        # private and external IP mapping
#private_ip_address_mapping-ip-addresses#

api:
  authentication_option: "Local"    # Values: "Local", "SSO", "LocalAndSSO"
  access_log_enabled: true         # Enable access logs
  jwt_expiry: "30 minutes"         # Default session duration
  jwt_secret_file: "{ install_path }/jwt/secret.txt" # Location of JWT secret file
  jwt_audience: "http://my.service"
  oidc_jwks_url: "na"
  local_auth_mode_password: Welcome123
  session_secret: "q0civ3L33W"

# set the following API parameters when                # SSO and LdapAndSSO properties
# authentication_option is SSO or LdapAndSSO
# oidc_issuer:
# oidc_auth_url
# oidc_token_url:
# oidc_user_info_url:
# oidc_callback_url:
# oidc_jwks_url:
# oidc_client_scope:
# oidc_groups_attribute:
# oidc_uid_attribute:
# oidc_client_id:
# oidc_client_secret:
# admin_object_id:
# entitlement_owner_object_id:
# executive_object_id:
# supervisor_object_id:
# user_object_id:
# application_owner_object_id:

```

```

# role_owner_object_id:
# role_engineer_object_id:
# oidc_end_session_endpoint:
# oidc_logout_redirect_url:

# mongo config starts
# uncomment below for mongo with replication enabled. Not needed for
# single node deployments
# mongodb_replication_replset: mongors

# custom key
# password for inter-process authentication
#
# please regenerate this file on production environment with command 'openssl rand -base64 741'
#mongodb_keyfile_content: |
# 8pYcxvCqoe89kcp33KuTtKVf5MoHGEFjTnudrq5BosvWRoIxLowmdjrmUpVfAivh
# CHjqM6w0zVBytAxH1lW+7teMYe6eDn2S/O/1YlRRiW57bWU3zjliW3VdguJar5i9
# Z+1a8lI+0S9pWynbv9+Ao0aXFjSJYVxAm/w7DJbVRGcPhsPmExiSBDw8szfQ8PAU
# 2hwRl7nqPZZMMR+uQThg/zV9r0zHJmkqZts04UJSilG9euLCYrzW2hdoPuCrEDhu
# Vsi5+nwAgYR9dP2oWkmGN1dwRe0ixSIM2UzFgpaXZaMOG6VztmFr1VXh8oFDRGM0
# cGrFHcnGF7oUGfWnI2Cekngk64dHA2qD7WxXPbQ/svn9EfTY5aPw5lXzKA87Ds8p
# KHVfUYvmA6wVsxb/riGLwc+XZ1b6M9ggHn1XSpsnYRjF6UzfRcRR2WycXLZELaqu
# iKxLKB5FYqMBH7Sqq3qBCtE53vZ7T1nefq5RFzmykviYP63Uhu/A2EQatrMnaFP1
# TTG5CaPjob45CBSyMrheYRWKqxdWN93BTgiTW7p0U6RB0/OCUbsVX6IG3I9N8Uqt
# l8Kc+7aOmtUqFkwo8w30prIOjStMroKxNsuK9KTUiPu2cj7gwYQ574vV3hQvQPAR
# hhb9ohKr0zoPQt31iITj0FDKJzPepeuzqeq8F51HB56RZKpXdRTfy8G6Oa0T68cV5
# vP106T/okFKrl41FQ3CyYN5eRHyrTK99zTytrjoP2EbtIZ18z+bg/angRHYNzbGk
# lc3jpiGzs1ZWHd0nx0mHCMhU4usEcFbV6F10xzlwrsEhHkeiununlCsNHatiDgZp
# ZWLnP/mXKV992/Jhu0Z577DHlh+3JIYx0PceB9yzACJ8MMARHF7QpBkhtuGMGZpF
# T+c73exupZFxItXs1Bnhe3djgE3MKKyYvxNUIbcTJoe7nhVMrw0/7lBSpVLvC4p3
# wR700U0LdaGGQpslGtiE56SengoP

# mongo config ends

elastic_heap_size: 1g # sets the heap size (1g|2g|3g) for the Elastic Servers

jas:
  auth_enabled: true
  auth_type: 'jwt'
  signature_key_id: 'service1-hmac'
  signature_algorithm: 'hmac-sha256'
  max_memory: 4096M
  mapping_entity_type: /common/mappings
  datasource_entity_type: /common/datasources

mongo_port: 27017 # Port where Mongo is running
mongo_ldap: false # Specify if Mongo is authenticated against an LDAP

elastic_host: 10.128.0.28 # IP Address of master node where Opensearch is running
elastic_port: 9200 # Port of master node where Opensearch is running
elastic_user: elasticadmin # Opensearch username

kibana_host: 10.128.0.28 # IP Address of node where Opensearch Dashboard is running

apache_livy:

```

```
dest_dir: /home/ansible/livy # Folder where livy is installed. AutoID copies analytics files to this
directory.

cassandra: # Cassandra Nodes details.
  enable_ssl: "true" # Set if SSL is enabled.
  contact_points: # Comma seperated list of ip
addresses - first ip is master#
  port: 9042 # Port where cassandra node is
running
  username: zoranuser # User created for AutoID to seed
Schema
  cassandra_keystore_password: "Acc#1234" # Keystore Password
  cassandra_truststore_password: "Acc#1234" # Truststore Password
  ssl_client_key_file: "zoran-cassandra-client-key.pem" # Cassandra Client Key File
  ssl_client_cert_file: "zoran-cassandra-client-cer.pem" # Cassandra Client Cert File
  ssl_ca_file: "zoran-cassandra-server-cer.pem" # Cassandra Server Root CA File
  server_truststore_jks: "zoran-cassandra-server-truststore.jks" # Server Truststore file for
services to connect
  client_truststore_jks: "zoran-cassandra-client-truststore.jks" # Client Truststore file for
services to connect
  client_keystore_jks: "zoran-cassandra-client-keystore.jks" # Client Keystore file for
services to use
```

Administrator tasks

This chapter is written for administrators who must manage and maintain PingOne Autonomous Identity.

PingOne® Autonomous Identity is an entitlements and roles analytics system that lets you fully manage your company's access to your data.

An entitlement refers to the rights or privileges assigned to a user or thing for access to specific resources. A company can have millions of entitlements without a clear picture of what they are, what they do, and who they are assigned to. PingOne Autonomous Identity solves this problem by using advanced artificial intelligence (AI) and automation technology to determine the full entitlements landscape for your company. The system also detects potential risks arising from incorrect or over-provisioned entitlements that lead to policy violations. PingOne Autonomous Identity eliminates the manual re-certification of entitlements and provides a centralized, transparent, and contextual view of all access points within your company.



Self service

Run self-service tasks.



Manage identities

Add, edit, or remove user identities.



Prepare data

Prepare your data for ingestion.



Deployment tasks

Run deployment-related tasks.

**Set entity definitions**

Set your attribute entity definitions.

**Set data sources**

Set your data sources.

**Set attribute mappings**

Set attribute mappings.

**Set analytics settings**

Set analytic settings.

**Run analytics**

Run the analytics pipeline.

**Admin tasks**

Run admin tasks.



Server maintenance

Run server maintenance-related tasks.



Roles management

Manage your roles.

For installation instructions, refer to the [PingOne Autonomous Identity Installation Guide](#).

For a description of the PingOne Autonomous Identity UI console, refer to the [PingOne Autonomous Identity Users Guide](#).

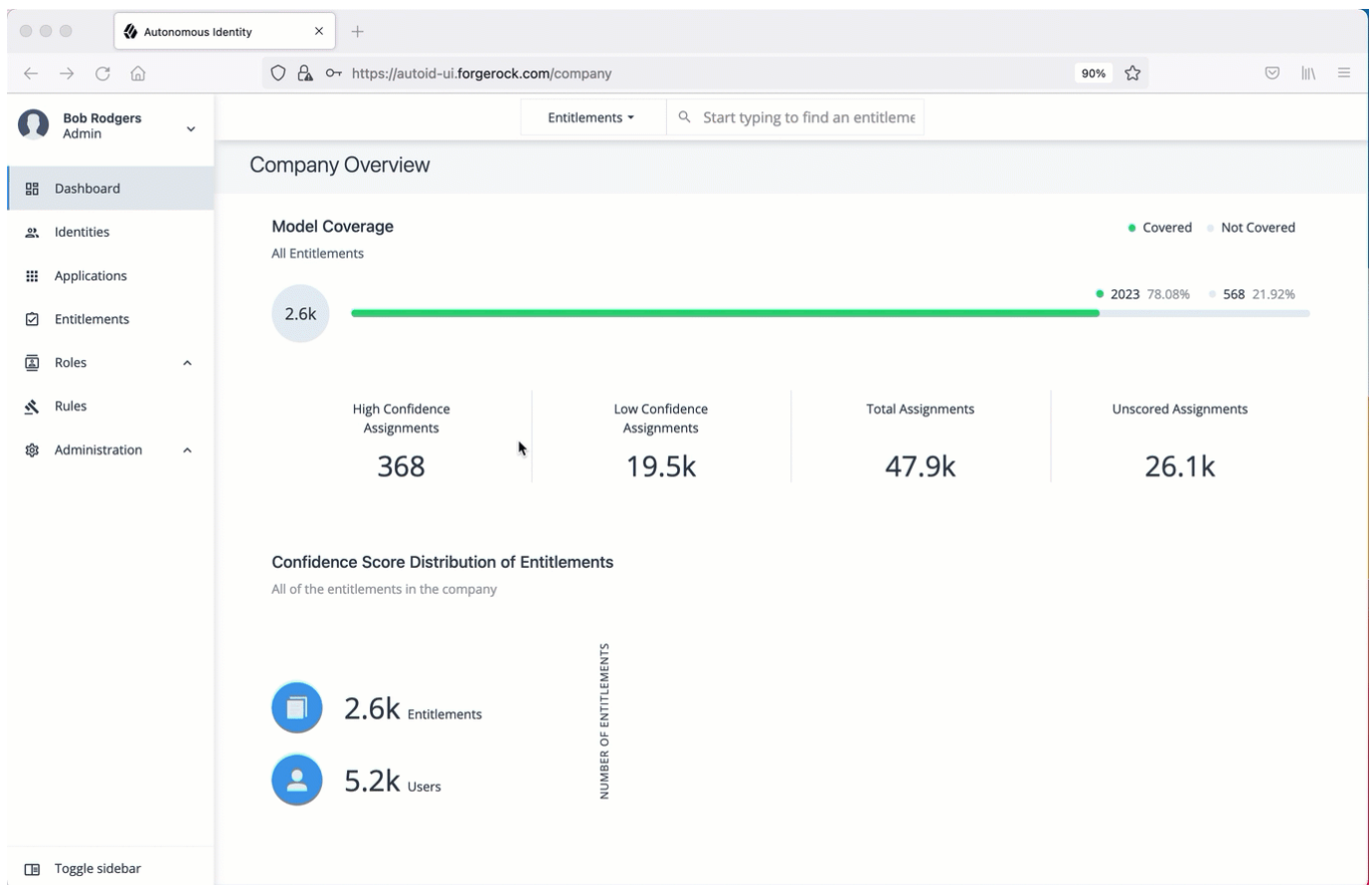
Self service

PingOne Autonomous Identity provides a self service UI page for administrators to change their profile and password information.

The page also lets administrators create time-based API keys for users to access the PingOne Autonomous Identity system. For more information, refer to [Generate an API key](#).

Reset your password

1. On the PingOne Autonomous Identity UI, click the admin drop-down on the top-left of the page.
2. Click **Self Service**.
3. On the Profile page, enter and re-enter a new password, and then click **Save**.



Update your profile

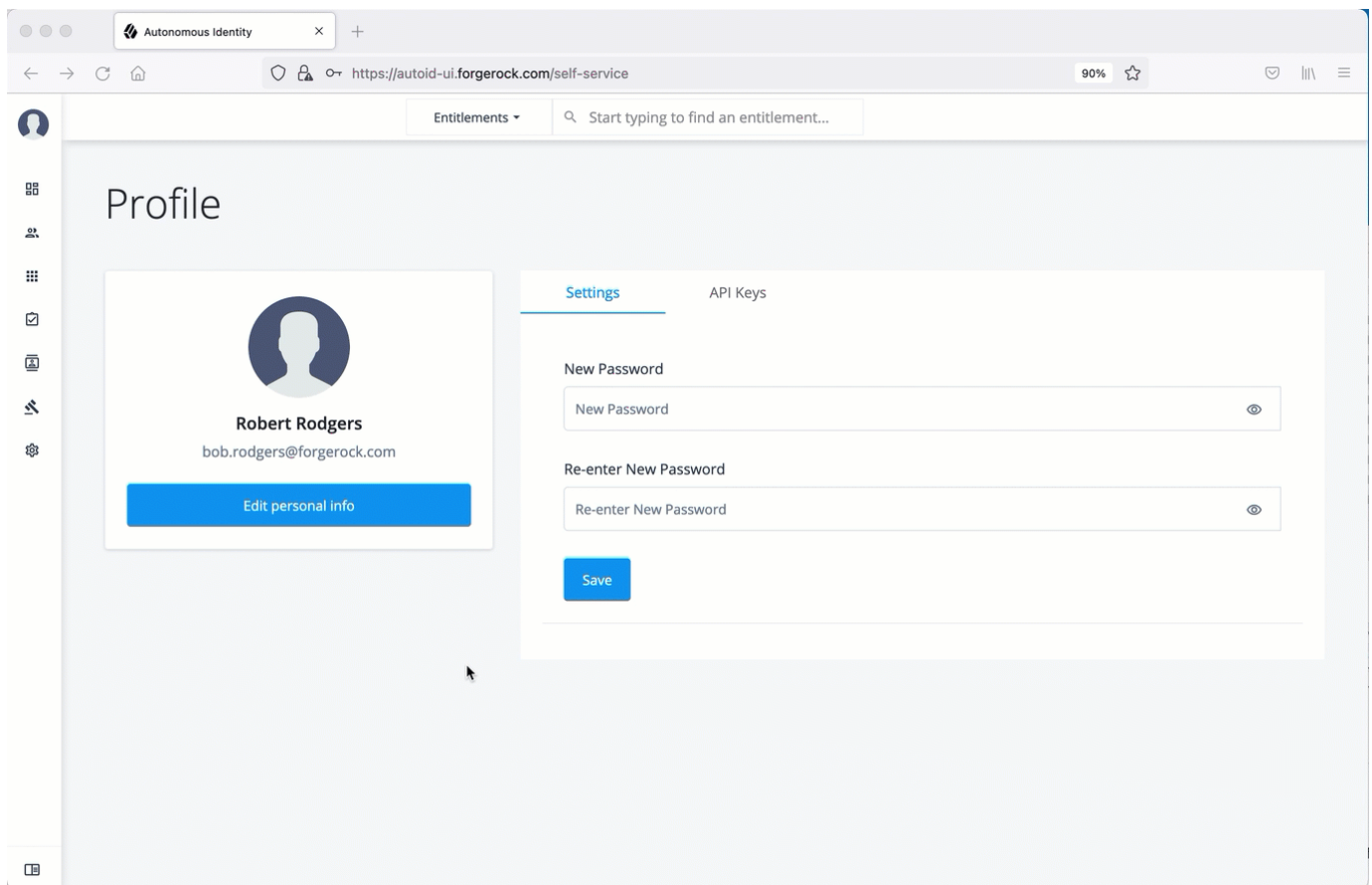
1. On the PingOne Autonomous Identity UI, click the admin drop-down on the top-left of the page.
2. Click **Self Service**.
3. On the Profile page, click **Edit personal info** to update your profile details:



Note

You cannot change your email address or group ID as these are used to identify each user.

1. Update the display name.
2. Update your distinguished name (DN).
3. Update your uid.
4. Click **Save** to apply your changes.

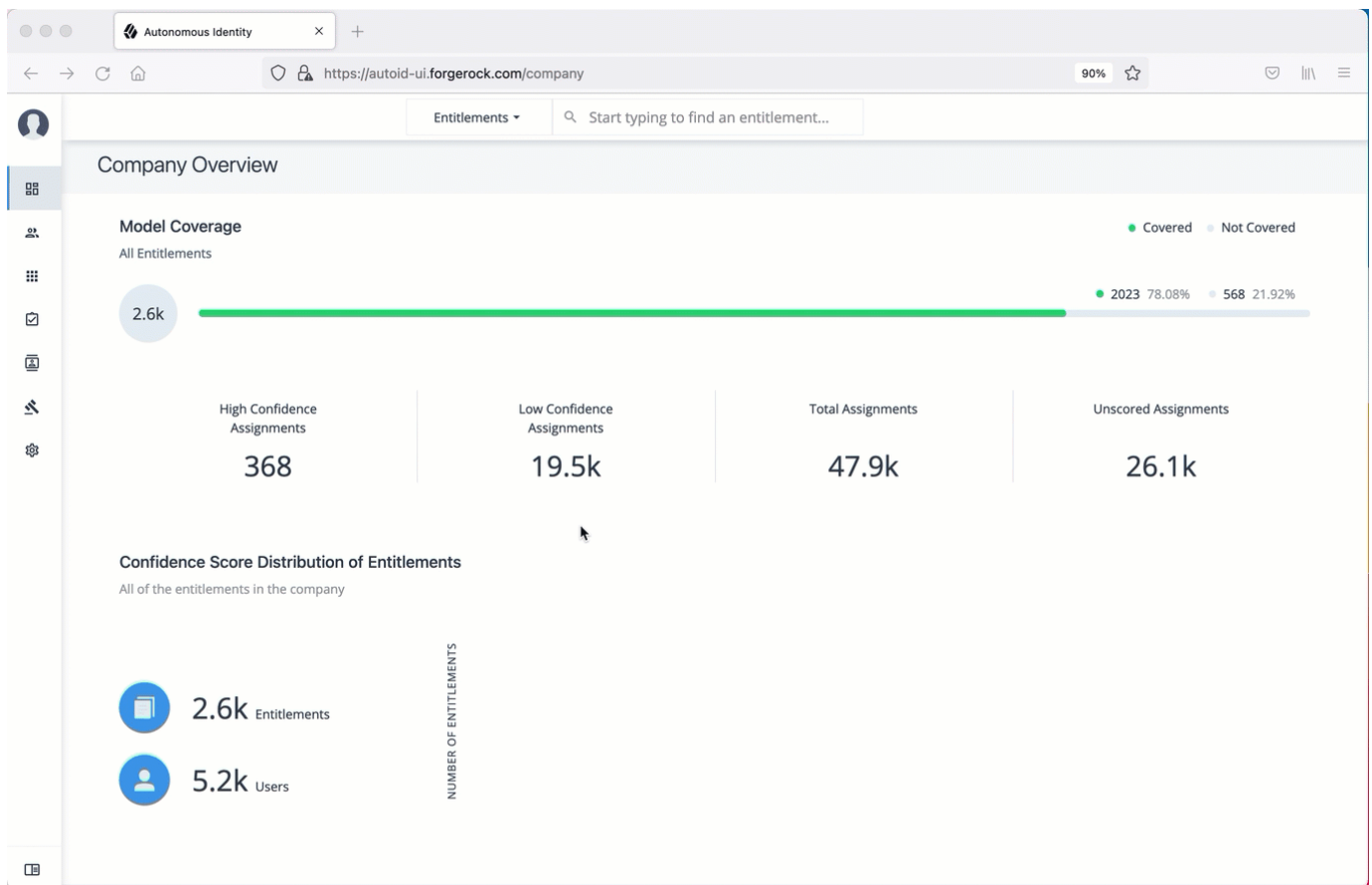


Manage identities

The Manage Identities page lets administrators add or edit, assign roles, and deactivate users to PingOne Autonomous Identity.

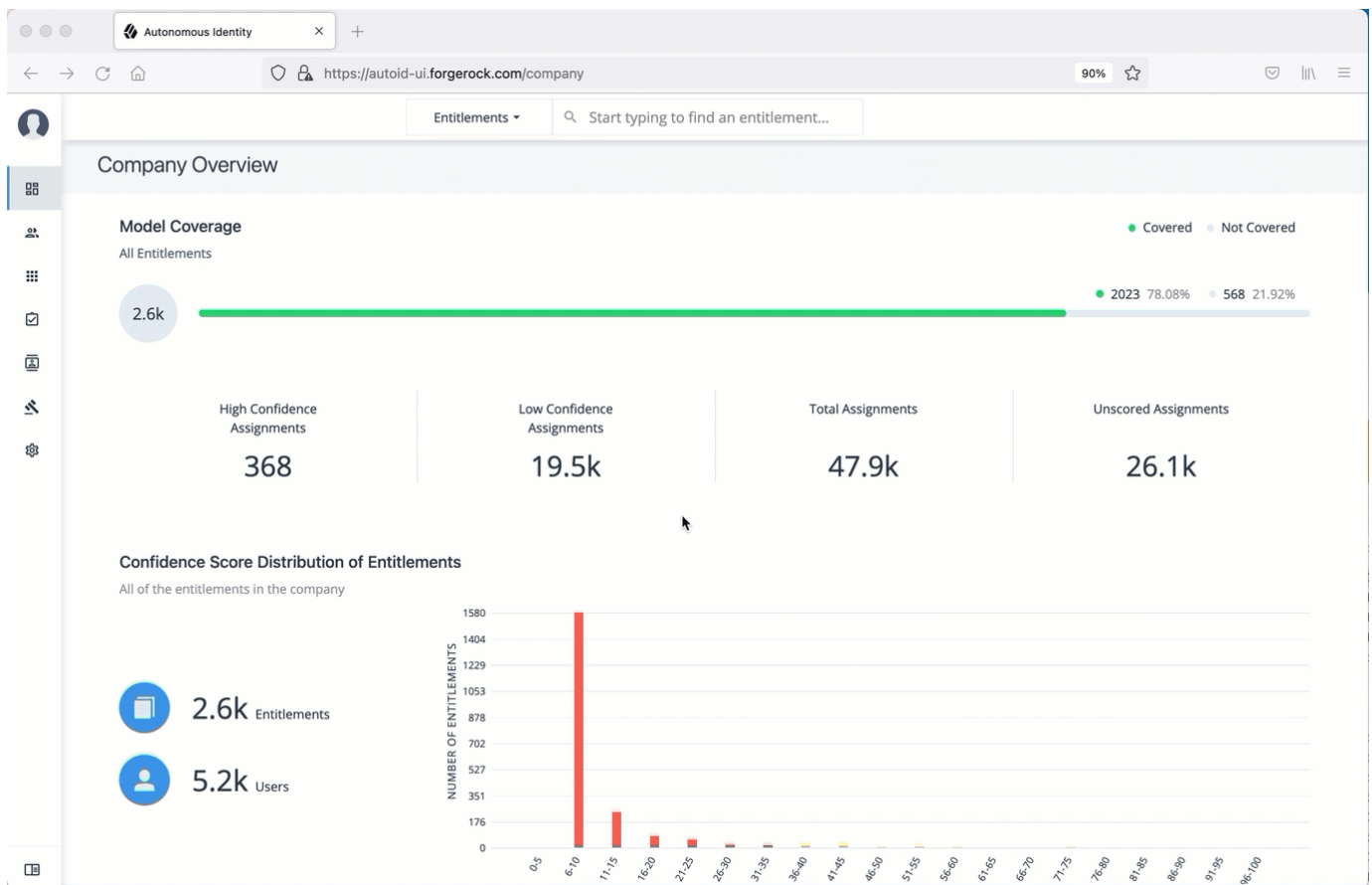
View the default roles

1. On the PingOne Autonomous Identity UI, click the administration icon on the navigation menu, and then click **Manage**.
2. On the Manage Identities page, click **Roles**.
3. Select a specific role, and then click **Edit** to view its details.
4. Click through the Details and Permissions to view its details. You cannot change the permissions in these roles.
5. Click **Role Members** to access the members associated with this role. If you want to add a user to this Role group, click **New Role Member** and enter the user's name. You can enter multiple users. When finished, click **Save**.



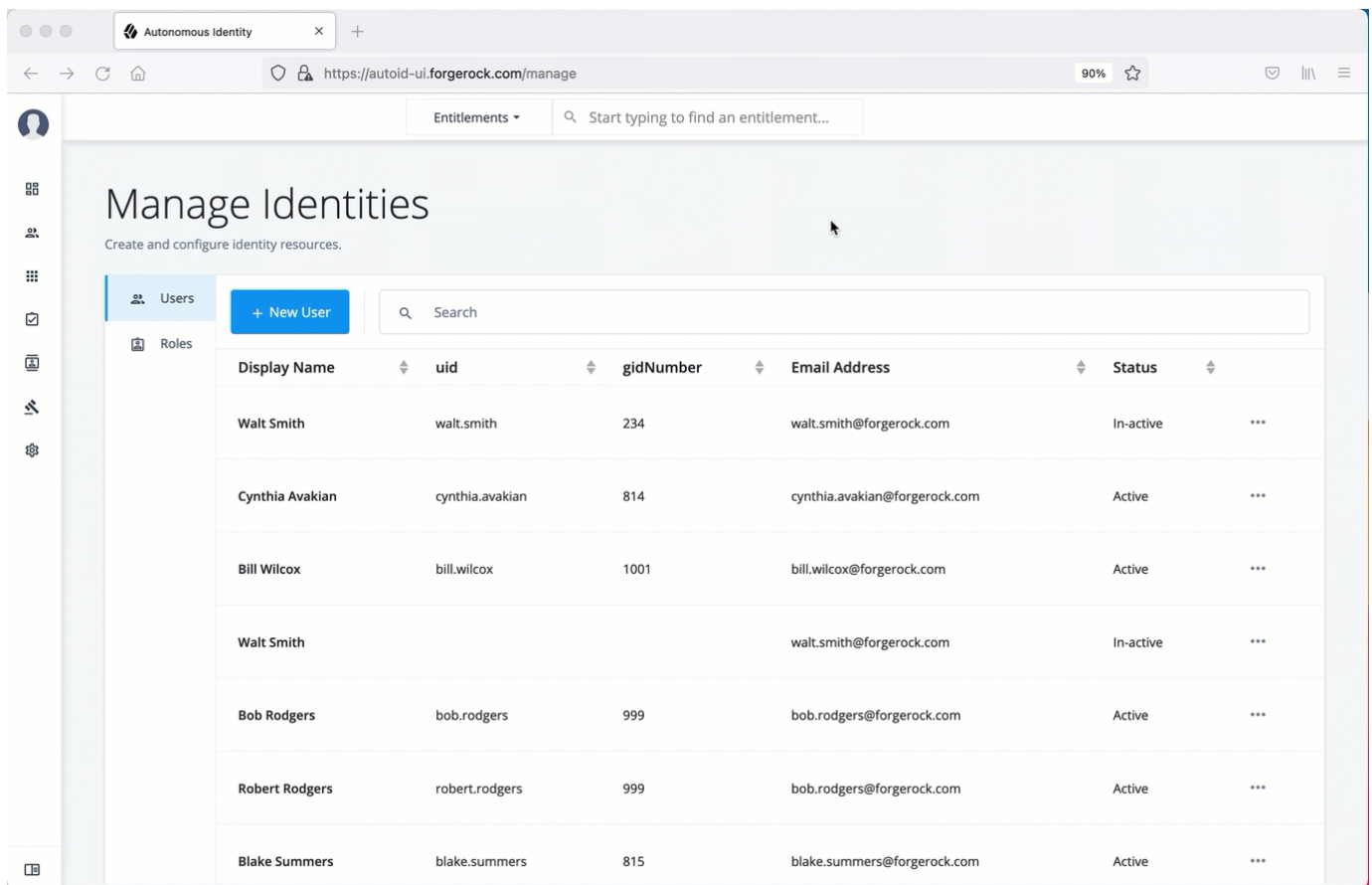
Create a new user

1. On the PingOne Autonomous Identity UI, click the administration icon on the navigation menu, and then click **Manage**.
2. On the Manage Identities page, click **New User**.
3. Enter the Display Name, Email Address, DN, Gid Number, Uid, and Password for the user.
4. Click **Save**.
5. Click **Authorization Roles**, and then click **New Authorization Roles**. This step is important to assign the proper role to the user.
6. Select a role to assign the user, and then click **Save**.



Reset a user's password

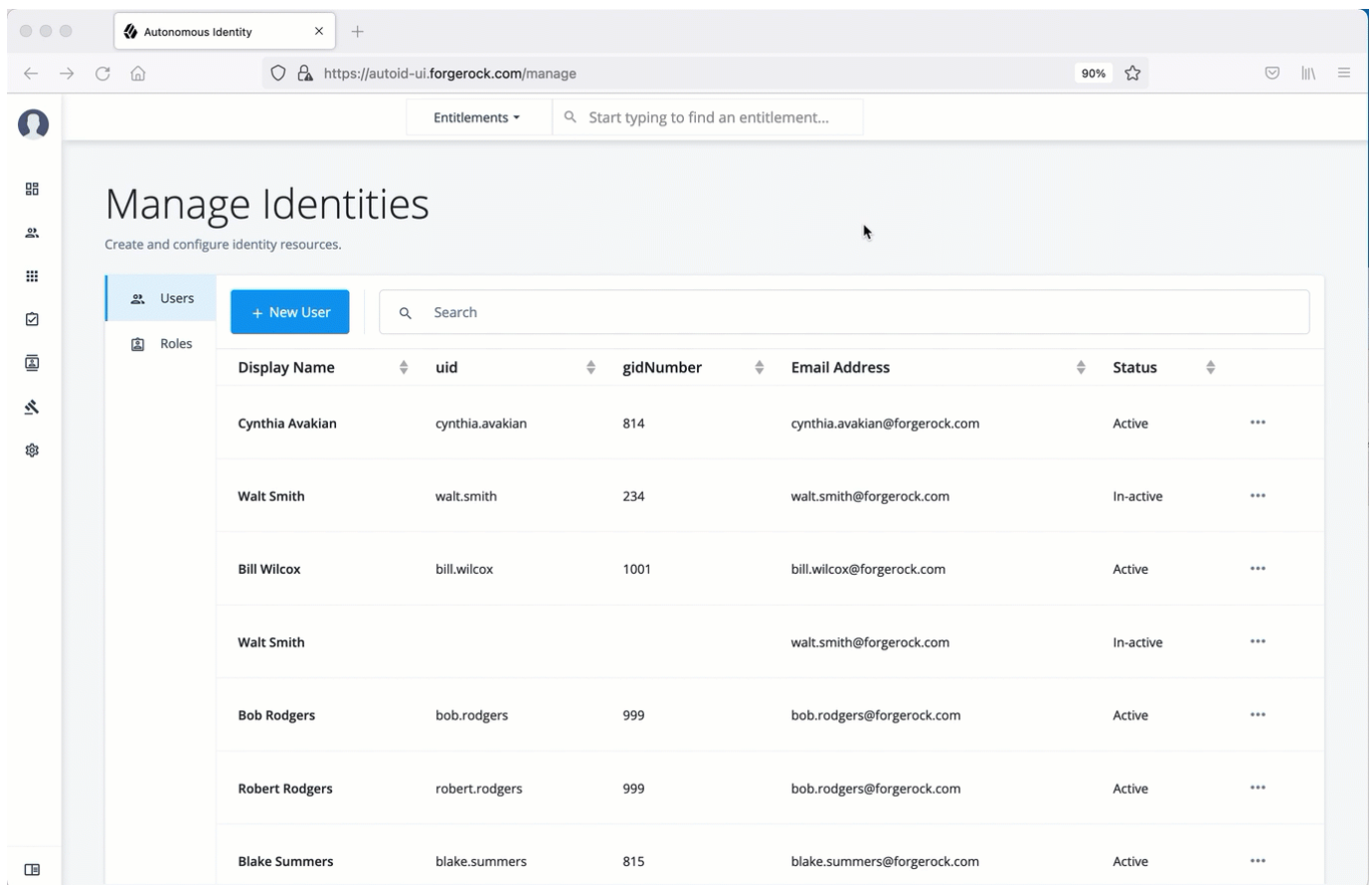
1. On the PingOne Autonomous Identity UI, click the administration icon on the navigation menu, and then click **Manage**.
2. On the Manage Identities page, search for a user.
3. For a specific user, click **Edit**.
4. Click **Reset Password**, enter a temporary password, and then click **Save**.



Add a role to an existing user

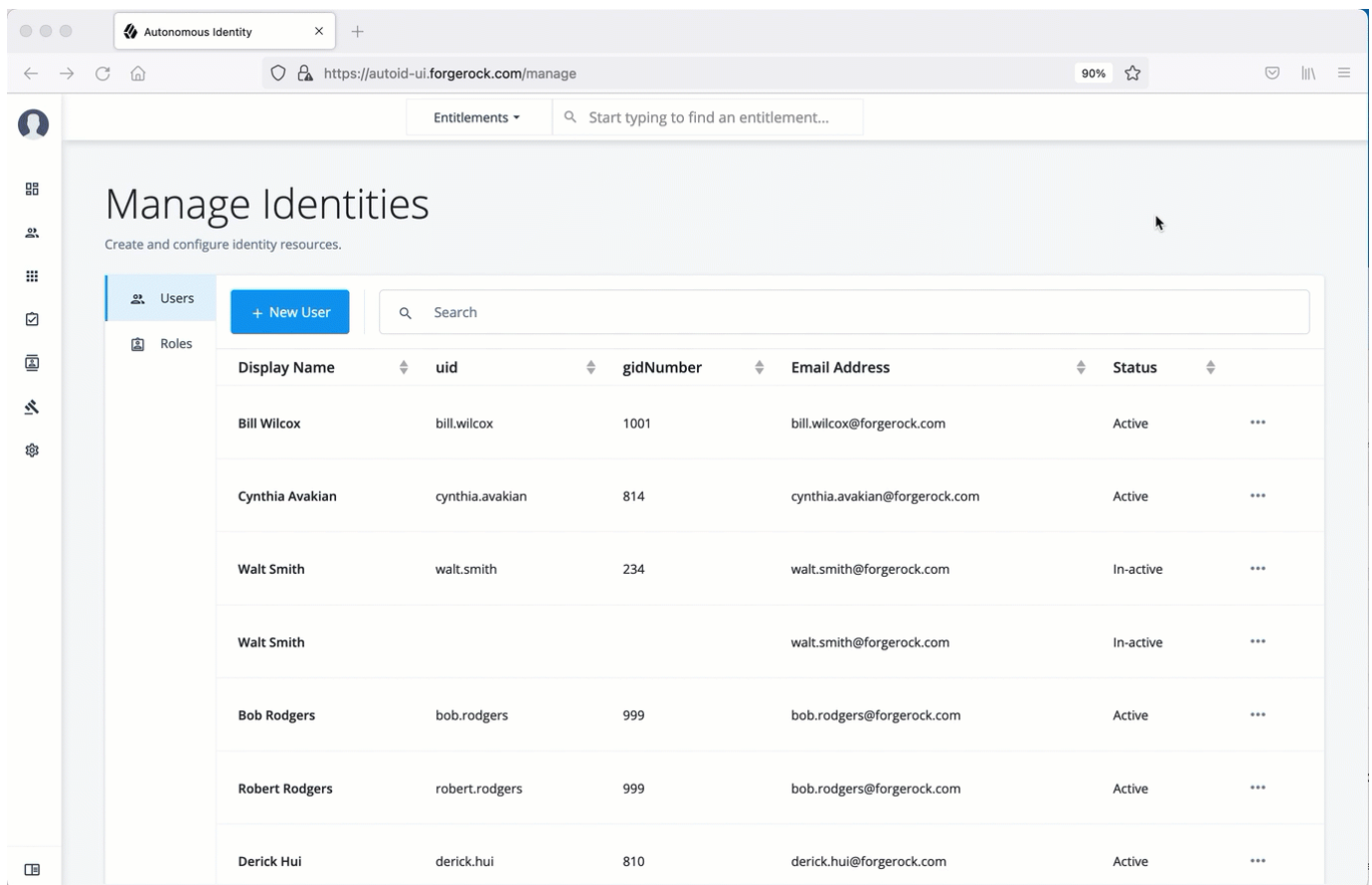
Often administrators need to assign roles to existing members. There are two ways to do this: from the user's detail page and through the role's Role Members page (refer to [View the default roles](#)).

1. On the PingOne Autonomous Identity UI, click the administration icon on the navigation menu, and then click **Manage**.
2. On the Manage Identities page, search for a user.
3. For a specific user, click **Edit**.
4. Click **Authorization Roles**, and then click **New Authorization Roles**.
5. Select one or more roles to add, and then click **Save**.



Deactivate an existing user

1. On the PingOne Autonomous Identity UI, click the administration icon on the navigation menu, and then click **Manage**.
2. On the Manage Identities page, search for a user.
3. For a specific user, click **Deactivate**. The user's status changes to "In-active".



Prepare data

PingOne Autonomous Identity administrators and deployers must set up additional tasks prior to your installment.

The following are some deployments tasks that may occur:

Data preparation

Once you have deployed PingOne Autonomous Identity, you can prepare your dataset into a format that meets the schema.

The initial step is to obtain the data as agreed upon between ForgeRock and your company. The files contain a subset of user attributes from the HR database and entitlement metadata required for the analysis. Only the attributes necessary for analysis are used.

There are a number of steps that must be carried out before your production entitlement data is input into PingOne Autonomous Identity. The summary of these steps are outlined below:

Data collection

Typically, the raw client data is not in a form that meets the PingOne Autonomous Identity schema. For example, a unique user identifier can have multiple names, such as `user_id`, `account_id`, `user_key`, or `key`. Similarly, entitlement columns can have several names, such as `access_point`, `privilege_name`, or `entitlement`.

To get the correct format, here are some general rules:

- Submit the raw client data in `.csv` file format. The data can be in a single file or multiple files. Data includes application attributes, entitlement assignments, entitlements descriptions, and identities data.
- Duplicate values should be removed.
- Add optional columns for additional training attributes, for example, `MANAGERS_MANAGER` and `MANAGER_FLAG`. You can add these additional attributes to the schema using the PingOne Autonomous Identity UI. For more information, refer to [Set Entity Definitions](#).
- Make a note of those attributes that differ from the PingOne Autonomous Identity schema, which is presented below. This is crucial for setting up your attribute mappings. For more information, refer to [Set Attribute Mappings](#).

CSV files and schema

The required attributes for the schema are as follows:

CSV Files Schema

Files	Schema
applications.csv	<p>This file depends on the attributes that the client wants to include. Here are some required columns:</p> <ul style="list-style-type: none">• app_id. Specifies the applications's unique ID.• app_name. Specifies the applications's name.• app_owner_id. Specifies the ID of the application's owner.
assignments.csv	<ul style="list-style-type: none">• user_id. Specifies the unique user ID to which the entitlement is assigned.• ent_id. Specifies the entitlements's unique ID.
entitlements.csv	<ul style="list-style-type: none">• ent_id. Specifies the entitlements's unique ID.• ent_name. Specifies the entitlement name.• ent_owner_id. Specifies the entitlement's owner.• app_id. Specifies the applications's unique ID.
identities.csv	<ul style="list-style-type: none">• usr_id. Specifies the user's unique ID.• user_name. Specifies a human readable username. For example, <code>John Smith</code>.• usr_manager_id. Specifies the user's manager ID.

Deployment tasks

PingOne Autonomous Identity administrators and deployers must set up additional tasks during installment.

The following are some deployments tasks that may occur:

Customize the domain and namespace

By default, the PingOne Autonomous Identity URL and domain for the UI console is set to `autoid-ui.forgerock.com`, and the URL and domain for the self-service feature is `autoid-selfservice.forgerock.com`.

Note

These instructions are for new deployments. To change the domain and certificates in existing deployments, refer to [Customize domain and namespace \(existing deployments\)](#).

Customize domain and namespace (new deployments)

1. Customize the domain name and target environment by editing the `/autoid-config/vars.xml` file. By default, the domain name is set to `forgerock.com` and the target environment is set to `autoid`. The default PingOne Autonomous Identity URL will be: <https://autoid-ui.forgerock.com>. For example, set the domain name to the following:

```
domain_name: example.com
target_environment: autoid
```

2. If you set up your domain name and target environment in the previous step, you need to change the certificates to reflect the changes. PingOne Autonomous Identity generates self-signed certificates for its default configuration. You must generate new certificates as follows:

1. Generate the private key (that is, `server.key`).

```
openssl genrsa -out server.key 2048
```

2. Generate the certificate signing request using your key. When prompted enter attributes sent with your certificate request:

```
openssl req -new -key server.key -out server.csr

Country Name (2 letter code) [XX]:US
State or Province Name (full name) {}:Texas
Locality Name (eg, city) [Default City]:Austin
Organization Name (eg, company) [Default Company Ltd]:Ping
Organizational Unit Name (eg, section) []:Eng
Common Name (eg, your name or your server's hostname) []:autoid-ui.example.com
Email Address []:

A challenge password []:
An optional company name []:
```

3. Generate the self-signed certificate.

```
openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
```

4. Copy the certificate to the `/autoid-config/certs` directory. Make sure to use the following filename: `nginx-jas-wildcard.pem`.

```
openssl x509 -in server.crt -out nginx-jas-wildcard.pem
cp nginx-jas-wildcard.pem ~/autoid-config/certs
```

5. Copy the key to the `/autoid-config/certs` directory. Make sure to use the following filename: `nginx-jas.key`, depending on where your `~/autoid-config/certs/` resides.

```
cp -i ~/.ssh/server.key /autoid-config/certs/nginx-jas.key

or

scp -i ~/.ssh/server.key autoid@remotehost:/autoid-config/certs/nginx-jas.key
```

6. Run the PingOne Autonomous Identity deployer. Make sure that there are no errors after running the `./deployer.sh run` command.

```
./deployer.sh run
```

1. Make the domain changes on your DNS server or update your `/etc/hosts` (Linux/Unix) file or `C:\Windows\System32\drivers\etc\hosts` (Windows) locally on your machine.

Customize domain and namespace (existing deployments)

1. Modify the server name values with your updated domain name in the following files under `/opt/autoid/mounts/nginx/conf.d`:
 - `api.conf`
 - `ui.conf`
 - `kibana.conf`
 - `jas.conf`
2. Copy the SSL certificate file and corresponding SSL certificate key to the `/opt/autoid/mounts/nginx/cert` directory. The `/opt/autoid/mounts/nginx/cert` directory is mounted under `/etc/nginx/cert` in the container.
3. Modify `ssl_certificate` and `ssl_certificate_key` in `/opt/autoid/mounts/nginx/nginx.conf` with the correct filenames. Only the name of the files need to be updated, the path stays the same.

 **Note**

When using self-signed certificates, you need to import the new certificates in the JAS keystore and truststore at: `/opt/autoid/certs/jas/jas-client-keystore.jks` and `/opt/autoid/certs/jas/jas-server-truststore.jks`:

```
keytool -importcert -keystore jas-client-keystore.jks -alias myalias -file /opt/autoid/mounts/nginx/cert/cert.crt -noprompt -keypass mypass -storepass mypass
```

```
keytool -importcert -keystore jas-server-truststore.jks -alias myalias -file /opt/autoid/mounts/nginx/cert/cert.crt -noprompt -keypass mypass -storepass mypass
```

4. Restart the nginx container:

```
docker stack rm nginx
docker stack deploy -c /opt/autoid/res/nginx/docker-compose.yml nginx
```

5. Update the new domain name in your hosts file where an entry exists for `JAS`. For example, the default JAS url is `autoid-jas.forgerock.com`. Change the JAS URL with your domain name.

6. Update the `JAS_URL` environment variable on all nodes by updating and sourcing your `.bashrc` file.

7. Restart Spark and Livy.

Configuring your filters

The filters on the Applications pages let you focus your searches based on entitlement and user attributes. In most cases, the default filters should suffice for most environments. However, if you need to customize the filters, you can do so by accessing Searchable attribute under entity definitions. For information, refer to [Set Entity Definitions](#).

The default filters for an entitlement are the following:

- Risk Level
- Criticality

The default filters for an user attributes are the following:

- User Department Name
- Line of Business Subgroup
- City
- Jobcode Name
- User Employee Type
- Chief Yes No
- Manager Name
- Line of Business

- Cost Center

Change the Vault Passwords

PingOne Autonomous Identity uses the ansible vault to store passwords in encrypted files, rather than in plaintext. PingOne Autonomous Identity stores the vault file at `/autoid-config/vault.yml` and saves the encrypted passwords to `/config/.autoid_vault_password`. The `/config/` mount is internal to the deployer container. The default encryption algorithm used is AES256.

By default, the `/autoid-config/vault.yml` file uses the following parameters:

```
configuration_service_vault:
  basic_auth_password: Welcome123

openldap_vault:
  openldap_password: Welcome123

cassandra_vault:
  cassandra_password: Welcome123
  cassandra_admin_password: Welcome123

mongo_vault:
  mongo_admin_password: Welcome123
  mongo_root_password: Welcome123

elastic_vault:
  elastic_admin_password: Welcome123
  elasticsearch_password: Welcome123
```

Assume that the vault file is encrypted during the installation. To edit the file:

Edit the Vault file

1. Change to the `/autoid-config/` directory.

```
$ cd ~/autoid-config/
```

2. First, decrypt the vault file.

```
$ ./deployer.sh decrypt-vault
```

3. Open a text editor and edit the `vault.yml` file.

4. Encrypt the file again.

```
$ ./deployer.sh encrypt-vault
```

Set Up single sign-on (SSO)

PingOne Autonomous Identity supports single sign-on (SSO) using OpenID Connect (OIDC) JWT tokens. SSO lets you log in once and access multiple applications without the need to re-authenticate yourself. You can use any third-party identity provider (IdP) to connect to PingOne Autonomous Identity.

There are two scenarios for SSO configuration:

- **Set up SSO for initial deployments.** In this example, we use ForgeRock Access Management (AM) as an OpenID Connect (OIDC) IdP for PingOne Autonomous Identity during the original installation of PingOne Autonomous Identity. Refer to [Set up SSO in initial deployments](#).
- **Set up SSO for existing deployments.** For procedures to set up SSO in an existing PingOne Autonomous Identity deployment, see [Set up SSO in existing deployments](#).

Note

If you set up SSO-only, be aware that the following services are not deployed with this setting:

- Self Service
- Manage Identities

If you want to use these services and SSO, set up the authentication as "LocalAndSSO" in the `vars.yml` file. Otherwise, for SSO-only, you must use the user services provided by your SSO provider.

Set up SSO in initial deployments

The following procedure requires a running instance of ForgeRock AM. For more information, refer to [PingOne Access Management Authentication and Single Sign-On Guide](#).

1. First, set up your hostnames locally in `/etc/hosts` (Linux/Unix) file or `C:\Windows\System32\drivers\etc\hosts` (Windows):

```
35.189.75.99 autoid-ui.forgerock.com autoid-selfservice.forgerock.com
35.246.65.234 openam.example.com
```

2. Open a browser and point to <http://openam.example.com:8080/openam>. Log in with username: `amadmin`, password: `cangetinam`.
3. In AM, select Realm > Identities > Groups tab, and add the following groups:
 - AutoidAdmin
 - AutoidEntitlementOwner
 - AutoidExecutive
 - AutoidSupervisor
 - AutoidUser
 - AutoidAppOwner
 - AutoidRoleOwner

- AutoidRoleEngineer

Note

The group names above are arbitrary and are defined in the `/autoid-config/vars.yml` file. Ensure that the groups you create in AM match the values in the `vars.yml` file.

4. Add the `demo` user to each group.
5. Go back to the main AM Admin UI page. Click **Configure OAuth Provider**.
6. Click **Configure OpenID Connect**, and then **Create**.
7. Select desired Realm > Go to Applications > OAuth 2.0, and then click **Add Client**. Enter the following properties, specific to your deployment:

```
Client ID:      <autoid>
Client secret:  <password>
Redirection URIs: https://<autoid-ui>.<domain>/api/sso/finish
Scope(s):      openid profile
```

For example:

```
Client ID:      autoid
Client secret:  Welcome123
Redirection URIs: https://autoid-ui.forgerock.com/api/sso/finish
Scope(s):      openid profile
```

8. On the New Client page, go to the Advanced tab, and enable **Implied Consent**. Next, change the **Token Endpoint Authentication Method** to `client_secret_post`.
9. Edit the OIDC claims script to return `roles (groups)`, so that AM can match the PingOne Autonomous Identity groups. Additionally, add the groups as a claim in the script:

```
"groups": { claim, identity -> [ "groups" : identity.getMemberships(IdType.GROUP).collect { group -> group.name } ] }
```

In the `utils.setScopeClaimsMap` block, add:

```
groups: ['groups']
```


Note

For more information about the OIDC claims script, refer to the [ForgeRock Knowledge Base](#).

The `id_token` returns the content that includes the group names.

```
{
  "at_hash": "QJRGiQgr1c1s0E4Q8BNyyg",
  "sub": "demo",
  "auditTrackingId": "59b6524d-8971-46da-9102-704694cae9bc-48738",
  "iss": "http://openam.example.com:8080/openam/oauth2",
  "tokenName": "id_token",
  "groups": [
    "AutoIdAdmin",
    "AutoIdSupervisor",
    "AutoIdUser",
    "AutoIdExecutive",
    "AutoIdEntitlementOwner",
    "AutoIdAppOwner",
    "AutoIdRoleOwner",
    "AutoIdRoleEngineer"
  ],
  "given_name": "demo",
  "aud": "autoid",
  "c_hash": "SoLsfC3zjGq9xF5mJG_C9w",
  "acr": "0",
  "org.forgerock.openidconnect.ops": "B15A_wXm581f08INtYHHcwSQtJI",
  "s_hash": "b0htX8F73IMjSPeVAqxyTQ",
  "azp": "autoid",
  "auth_time": 1592390726,
  "name": "demo",
  "realm": "/",
  "exp": 1592394729,
  "tokenType": "JWTToken",
  "family_name": "demo",
  "iat": 1592391129,
  "email": "demo@example.com"
}
```

Note

For more information on how to retrieve the `id_token` for observation, refer to [OpenID Connect 1.0 Endpoints](#) .

You have successfully configured AM as an OIDC provider.

- Next, we set up PingOne Autonomous Identity. Change to the PingOne Autonomous Identity install directory on the deployer machine.

```
cd ~/autoid-config/
```

- Open a text editor, and set the SSO parameters in the `/autoid-config/vars.yml` file. Make sure to change `LDAP` to `SSO`.

```

authentication_option: "SSO"

oidc_issuer: "http://openam.example.com:8080/openam/oauth2"
oidc_auth_url: "http://openam.example.com:8080/openam/oauth2/authorize"
oidc_token_url: "http://openam.example.com:8080/openam/oauth2/access_token"
oidc_user_info_url: "http://openam.example.com:8080/openam/oauth2/userinfo"
oidc_jwks_url: "http://openam.example.com:8080/openam/oauth2/connect/jwk_uri"
oidc_callback_url: "https://autoid-ui.forgerock.com/api/sso/finish"
oidc_client_scope: 'openid profile'
oidc_groups_attribute: groups
oidc_uid_attribute: sub
oidc_client_id: autoid
oidc_client_secret: Welcome1
admin_object_id: AutoIdAdmin
entitlement_owner_object_id: AutoIdEntitlementOwner
executive_object_id: AutoIdExecutive
supervisor_object_id: AutoIdSupervisor
user_object_id: AutoIdUser
application_owner_object_id: AutoIdAppOwner
role_owner_object_id: AutoIdRoleOwner
role_engineer_object_id: AutoIdRoleEngineer
oidc_end_session_endpoint: "http://openam.example.com:8080/openam/oauth2/logout"
oidc_logout_redirect_url: "http://openam.example.com:8088/openam/logout"

```

12. On the target machine, edit the `/etc/hosts` file or your DNS server, and add an entry for `openam.example.com`.

```
35.134.60.234 openam.example.com
```

13. On the deployer machine, run `deployer.sh` to push the new configuration.

```
$ deployer.sh run
```

14. Test the connection now. Access <https://autoid-ui/forgerock.com>. The redirect should occur with the following:

```

http://openam.example.com:8080/openam/XUI/?
realm=%2F&goto=http%3A%2F%2Fopenam.example.com%3A8080%2Fopenam%2Foauth2%2Fauthorize%3Fresponse_type%3Dcode%26c

```

Set up SSO in existing deployments

1. First, update the permissions configuration object as follows:

1. Obtain an PingOne Autonomous Identity admin level JWT bearer token. You can obtain it using curl and the PingOne Autonomous Identity login endpoint with administrator credentials. Use your admin username and password:

```
curl -X POST \
https://autoid-ui.forgerock.com/api/authentication/login \
-k \
-H 'Content-Type: application/json' \
-d '{
  "username": "bob.rodgers@forgerock.com",
  "password": "Welcome123"
}'
```

The response is:

[illegible]

2. Use curl and the bearer token from the previous step to obtain the PingOne Autonomous Identity JAS tenant ID:

```
curl -k -L -X GET 'https://autoid-ui.forgerock.com/jas/tenants' \
-H 'Authorization: Bearer <token_value>'
```

The response is:

```
[
  {
    "id": "31092f95-3eed-418e-8ffb-f1b707bc9372",
    "name": "autonomous-iam",
    "description": "System Tenancy",
    "created": "2023-03-02T20:15:30.166Z"
  }
]
```

3. To open the current permissions object, run the following curl command with the bearer token and tenant ID from the previous steps:

```
curl -k -L -X POST 'https://autoid-ui.forgerock.com/jas/entity/search/common/config' \
-H 'X-TENANT-ID: <tenant_id>' \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer <token_value>' \
-d '{
  "query": {
    "query": {
      "bool": {
        "must": {
          "match": {
            "name": "PermissionsConf"
          }
        }
      }
    }
  }
}
```

An example response is as follows:

 **Note**

You can find the permissions value under the hits object > hits array > _source > value.

```
{
  "took": 1,
  "timed_out": false,
  "_shards": {
    "total": 3,
    "successful": 3,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 1,
      "relation": "eq"
    },
    "max_score": 0.9808291,
    "hits": [
      {
        "_index": "autonomous-iam_common_config_latest",
        "_type": "_doc",
        "_id":
"f72a58dd8bf5a38205c2d4c9eeafe85ebbaa1c3a2670b45c57f0219022b90ea6fc50ebf88e720c98410600e427528f0fe702b55",
        "_score": 0.9808291,
        "_source": {
          "name": "PermissionsConf",
          "value": {
            "permissions": {
              "Zoran Admin": {
                "title": "Admin",
                "can": "*"
              }
            }
          }
        }
      }
    ]
  }
}
```

4. Edit the Permissions object in the template by replacing the "###Zoran_...Token###" fields with the SSO group ID. For example, the Permissions object would appear as follows before the change:

```
"###Zoran_Admin_Token###":
{
  "title": "Admin",
  "can": "*"
},
```

For SSO only setup, the following is used:

Note

f5bd09ca-096c-4a6e-b06d-65decc22cb09 is an example group ID for an organization's administrators.

```
"f5bd09ca-096c-4a6e-b06d-65decc22cb09":
{
  "title": "Admin",
  "can": "*"
},
```

For SSO and local setup, use the following:

```
"###Zoran_Admin_Token###":
{
  "title": "Admin",
  "can": "*"
},
"f5bd09ca-096c-4a6e-b06d-65decc22cb09":
{
  "title": "Admin",
  "can": "*"
},
```

5. Update the Permissions object in JAS with the edited JSON file:

```
curl -k -L -X PATCH 'https://autoid-ui.forgerock.com/jas/entity/upsert/common/config' \
-H 'X-TENANT-ID: <tenant_id>' \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer <token_value>' \
-d @<path/to/SSO.json>
```

A successful response is:

```
{
  "indexName": "autonomous-iam_common_config_latest",
  "indices":
  {
    "latest": "autonomous-iam_common_config_latest",
    "log": "autonomous-iam_common_config"
  }
}
```

6. Depending on how you want to configure SSO, use one of the following templates:

Note

The ContextID is an arbitrary UUID that can be any UUID. It is used just to track this transaction.

```

{
  "branch": "actual",
  "contextId": "ecba1baa-66d1-4548-8c74-6012bea9b838",
  "indexingRequired": true,
  "tags": {},
  "indexInSync": true,
  "entityData": [
    {
      "name": "PermissionsConf",
      "value":
      {
        "permissions":
        {
          "Zoran Admin":
          {
            "title": "Admin",
            "can": "*"
          },
          "###Zoran_Admin-Token###":
          {
            "title": "Admin",
            "can": "*"
          },
          "Zoran Role Engineer":
          {
            "title": "Role Engineer",
            "can": [
              "SHOW__ROLE_PAGE",
              "SEARCH__ALL_ROLES",
              "CREATE__ROLE",
              "UPDATE__ROLE",
              "DELETE__ROLE",
              "SHOW__ENTITLEMENT",
              "SHOW__USER",
              "SHOW__CERTIFICATIONS",
              "STATS_ALL__USERS",
              "SEARCH_ALL__USERS",
              "SEARCH_ALL__ENTITLEMENTS",
              "SEARCH__ROLE_USERS",
              "SEARCH__ROLE_ENTITLEMENTS",
              "SEARCH__ROLE_JUSTIFICATIONS",
              "SHOW_JUSTIFICATIONS",
              "SHOW_ROLE_METADATA",
              "SHOW_ROLE_ATTRIBUTES",
              "WORKFLOW__REQUESTS",
              "WORKFLOW__TASKS",
              "WORKFLOW__TASK_APPROVE"
            ]
          },
          "###Zoran_Role_Engineer-Token###":
          {
            "title": "Role Engineer",
            "can": [
              "SHOW__ROLE_PAGE",

```

```

        "SEARCH__ALL_ROLES",
        "CREATE__ROLE",
        "UPDATE__ROLE",
        "DELETE__ROLE",
        "SHOW__ENTITLEMENT",
        "SHOW__USER",
        "SHOW__CERTIFICATIONS",
        "STATS_ALL__USERS",
        "SEARCH_ALL__USERS",
        "SEARCH_ALL__ENTITLEMENTS",
        "SEARCH__ROLE_USERS",
        "SEARCH__ROLE_ENTITLEMENTS",
        "SEARCH__ROLE_JUSTIFICATIONS",
        "SHOW_JUSTIFICATIONS",
        "SHOW_ROLE_METADATA",
        "SHOW_ROLE_ATTRIBUTES",
        "WORKFLOW__REQUESTS",
        "WORKFLOW__TASKS",
        "WORKFLOW__TASK_APPROVE"
    ]
},
"Zoran Role Owner":
{
    "title": "Role Owner",
    "can": [
        "SHOW__ROLE_PAGE",
        "SEARCH__ROLES",
        "CREATE__ROLE",
        "UPDATE__ROLE",
        "DELETE__ROLE",
        "SHOW__ENTITLEMENT",
        "SHOW__USER",
        "SHOW__CERTIFICATIONS",
        "STATS__USERS",
        "SEARCH_ALL__USERS",
        "SEARCH_ALL__ENTITLEMENTS",
        "SEARCH__ROLES",
        "SEARCH__ROLE_USERS",
        "SEARCH__ROLE_ENTITLEMENTS",
        "SEARCH__ROLE_JUSTIFICATIONS",
        "SHOW_JUSTIFICATIONS",
        "SHOW_ROLE_METADATA",
        "SHOW_ROLE_ATTRIBUTES",
        "WORKFLOW__REQUESTS",
        "WORKFLOW__TASKS",
        "WORKFLOW__TASK_APPROVE"
    ]
},
"###Zoran_Role_Owner_Token###":
{
    "title": "Role Owner",
    "can": [
        "SHOW__ROLE_PAGE",
        "SEARCH__ROLES",
        "CREATE__ROLE",

```

```

        "UPDATE__ROLE",
        "DELETE__ROLE",
        "SHOW__ENTITLEMENT",
        "SHOW__USER",
        "SHOW__CERTIFICATIONS",
        "STATS__USERS",
        "SEARCH_ALL__USERS",
        "SEARCH_ALL__ENTITLEMENTS",
        "SEARCH__ROLES",
        "SEARCH__ROLE_USERS",
        "SEARCH__ROLE_ENTITLEMENTS",
        "SEARCH__ROLE_JUSTIFICATIONS",
        "SHOW_JUSTIFICATIONS",
        "SHOW_ROLE_METADATA",
        "SHOW_ROLE_ATTRIBUTES",
        "WORKFLOW__REQUESTS",
        "WORKFLOW__TASKS",
        "WORKFLOW__TASK_APPROVE"
    ]
},
"Zoran Role Auditor":
{
    "title": "Role Auditor",
    "can": [
        "SEARCH__ALL_ROLES",
        "STATS_ALL__USERS",
        "SEARCH_ALL__USERS",
        "SEARCH_ALL__ENTITLEMENTS",
        "SEARCH__ROLE_USERS",
        "SEARCH__ROLE_ENTITLEMENTS",
        "SEARCH__ROLE_JUSTIFICATIONS",
        "SHOW_JUSTIFICATIONS",
        "SHOW_ROLE_METADATA",
        "SHOW_ROLE_ATTRIBUTES",
        "WORKFLOW__REQUESTS",
        "WORKFLOW__TASKS"
    ]
},
"###Zoran_Role_Auditor_Token###":
{
    "title": "Role Auditor",
    "can": [
        "SEARCH__ALL_ROLES",
        "STATS_ALL__USERS",
        "SEARCH_ALL__USERS",
        "SEARCH_ALL__ENTITLEMENTS",
        "SEARCH__ROLE_USERS",
        "SEARCH__ROLE_ENTITLEMENTS",
        "SEARCH__ROLE_JUSTIFICATIONS",
        "SHOW_JUSTIFICATIONS",
        "SHOW_ROLE_METADATA",
        "SHOW_ROLE_ATTRIBUTES",
        "WORKFLOW__REQUESTS",
        "WORKFLOW__TASKS"
    ]
}

```

```

    },
    "Zoran Application Owner":
    {
        "title": "Application Owner",
        "can": [
            "SHOW__APPLICATION_PAGE",
            "SEARCH__USER",
            "SEARCH__ENTITLEMENTS_BY_APP_OWNER",
            "SHOW_OVERVIEW_PAGE",
            "SHOW__ENTITLEMENT",
            "SHOW__ENTITLEMENT_USERS",
            "SHOW__APP_OWNER_FILTER_OPTIONS",
            "SHOW__ENTT_OWNER_UNSCORED_ENTITLEMENTS",
            "SHOW__ENTT_OWNER_PAGE",
            "SHOW__ENTT_OWNER_USER_PAGE",
            "SHOW__ENTT_OWNER_ENT_PAGE",
            "SHOW__USER_ENTITLEMENTS",
            "SHOW__RULES_BY_APP_OWNER",
            "REVOKE__CERTIFY_ACCESS",
            "SHOW__USER",
            "SHOW__CERTIFICATIONS"
        ]
    },
    "###Zoran_Application_Owner_Token###":
    {
        "title": "Application Owner",
        "can": [
            "SHOW__APPLICATION_PAGE",
            "SEARCH__USER",
            "SEARCH__ENTITLEMENTS_BY_APP_OWNER",
            "SHOW_OVERVIEW_PAGE",
            "SHOW__ENTITLEMENT",
            "SHOW__ENTITLEMENT_USERS",
            "SHOW__APP_OWNER_FILTER_OPTIONS",
            "SHOW__ENTT_OWNER_UNSCORED_ENTITLEMENTS",
            "SHOW__ENTT_OWNER_PAGE",
            "SHOW__ENTT_OWNER_USER_PAGE",
            "SHOW__ENTT_OWNER_ENT_PAGE",
            "SHOW__USER_ENTITLEMENTS",
            "SHOW__RULES_BY_APP_OWNER",
            "REVOKE__CERTIFY_ACCESS",
            "SHOW__USER",
            "SHOW__CERTIFICATIONS"
        ]
    },
    "Zoran Entitlement Owner":
    {
        "title": "Entitlement Owner",
        "can": [
            "SEARCH__ENTITLEMENTS_BY_ENTT_OWNER",
            "SHOW_OVERVIEW_PAGE",
            "SHOW__ENTITLEMENT",
            "SHOW__ENTITLEMENT_USERS",
            "SHOW__ENTT_OWNER_FILTER_OPTIONS",
            "SHOW__ENTT_OWNER_UNSCORED_ENTITLEMENTS",

```

```

        "SHOW__ENTT_OWNER_PAGE",
        "SHOW__ENTT_OWNER_USER_PAGE",
        "SHOW__ENTT_OWNER_ENT_PAGE",
        "SHOW__USER_ENTITLEMENTS",
        "SHOW__RULES_BY_ENTT_OWNER",
        "REVOKE__CERTIFY_ACCESS",
        "SHOW__USER",
        "SHOW__CERTIFICATIONS",
        "LOOKUP_USER",
        "SEARCH__ROLE_USERS",
        "SEARCH__ROLE_ENTITLEMENTS",
        "SEARCH__ROLE_JUSTIFICATIONS",
        "SHOW_ROLE_METADATA",
        "SHOW_ROLE_ATTRIBUTES",
        "WORKFLOW__REQUESTS",
        "WORKFLOW__TASKS",
        "WORKFLOW__TASK_APPROVE"
    ]
},
"###Zoran_Entitlement_Owner_Token###":
{
    "title": "Entitlement Owner",
    "can": [
        "SEARCH__ENTITLEMENTS_BY_ENTT_OWNER",
        "SHOW_OVERVIEW_PAGE",
        "SHOW__ENTITLEMENT",
        "SHOW__ENTITLEMENT_USERS",
        "SHOW__ENTT_OWNER_FILTER_OPTIONS",
        "SHOW__ENTT_OWNER_UNSCORED_ENTITLEMENTS",
        "SHOW__ENTT_OWNER_PAGE",
        "SHOW__ENTT_OWNER_USER_PAGE",
        "SHOW__ENTT_OWNER_ENT_PAGE",
        "SHOW__USER_ENTITLEMENTS",
        "SHOW__RULES_BY_ENTT_OWNER",
        "REVOKE__CERTIFY_ACCESS",
        "SHOW__USER",
        "SHOW__CERTIFICATIONS",
        "LOOKUP_USER",
        "SEARCH__ROLE_USERS",
        "SEARCH__ROLE_ENTITLEMENTS",
        "SEARCH__ROLE_JUSTIFICATIONS",
        "SHOW_ROLE_METADATA",
        "SHOW_ROLE_ATTRIBUTES",
        "WORKFLOW__REQUESTS",
        "WORKFLOW__TASKS",
        "WORKFLOW__TASK_APPROVE"
    ]
},
"Zoran Executive":
{
    "title": "Executive",
    "can": [
        "SEARCH__USER",
        "SHOW__ASSIGNMENTS_STATS",
        "SHOW__COMPANY_PAGE",

```

```

        "SHOW__COMPANY_ENTITLEMENTS_DATA",
        "SHOW__CRITICAL_ENTITLEMENTS",
        "SHOW__ENTITLEMENT_AVG_GROUPS",
        "SHOW__USER_ENTITLEMENTS"
    ]
},
"###Zoran_Executive_Token###":
{
    "title": "Executive",
    "can": [
        "SEARCH__USER",
        "SHOW__ASSIGNMENTS_STATS",
        "SHOW__COMPANY_PAGE",
        "SHOW__COMPANY_ENTITLEMENTS_DATA",
        "SHOW__CRITICAL_ENTITLEMENTS",
        "SHOW__ENTITLEMENT_AVG_GROUPS",
        "SHOW__USER_ENTITLEMENTS"
    ]
},
"Zoran Supervisor":
{
    "title": "Supervisor",
    "can": [
        "SEARCH__USER",
        "SHOW__OVERVIEW_PAGE",
        "SHOW__SUPERVISOR_FILTER_OPTIONS",
        "SHOW__SUPERVISOR_PAGE",
        "SHOW__SUPERVISOR_ENTITLEMENT_USERS",
        "SHOW__SUPERVISOR_USER_ENTITLEMENTS",
        "SHOW__SUPERVISOR_UNSCORED_ENTITLEMENTS",
        "SEARCH__SUPERVISOR_USER_ENTITLEMENTS",
        "REVOKE__CERTIFY_ACCESS",
        "SHOW__ENTITLEMENT",
        "SHOW__USER",
        "SHOW__CERTIFICATIONS"
    ]
},
"###Zoran_Supervisor_Token###":
{
    "title": "Supervisor",
    "can": [
        "SEARCH__USER",
        "SHOW__OVERVIEW_PAGE",
        "SHOW__SUPERVISOR_FILTER_OPTIONS",
        "SHOW__SUPERVISOR_PAGE",
        "SHOW__SUPERVISOR_ENTITLEMENT_USERS",
        "SHOW__SUPERVISOR_USER_ENTITLEMENTS",
        "SHOW__SUPERVISOR_UNSCORED_ENTITLEMENTS",
        "SEARCH__SUPERVISOR_USER_ENTITLEMENTS",
        "REVOKE__CERTIFY_ACCESS",
        "SHOW__ENTITLEMENT",
        "SHOW__USER",
        "SHOW__CERTIFICATIONS"
    ]
},

```

```

    "Zoran User":
    {
      "title": "User",
      "can": [
        "SHOW__ENTITLEMENT",
        "SHOW__USER",
        "SHOW__CERTIFICATIONS"
      ]
    },
    "###Zoran_User_Token###":
    {
      "title": "User",
      "can": [
        "SHOW__ENTITLEMENT",
        "SHOW__USER",
        "SHOW__CERTIFICATIONS"
      ]
    },
    "Zoran Service Connector":
    {
      "title": "Service Connector",
      "can": [
        "SERVICE_CONNECTOR",
        "SHOW__API_KEY_MGMT_PAGE",
        "SHOW__ENTITLEMENT",
        "SHOW__USER",
        "SHOW__CERTIFICATIONS",
        "SHOW__RULES",
        "REVOKE__CERTIFY_ACCESS"
      ]
    },
    "###Zoran_Service_Connector_Token###":
    {
      "title": "Service Connector",
      "can": [
        "SERVICE_CONNECTOR",
        "SHOW__API_KEY_MGMT_PAGE",
        "SHOW__ENTITLEMENT",
        "SHOW__USER",
        "SHOW__CERTIFICATIONS",
        "SHOW__RULES",
        "REVOKE__CERTIFY_ACCESS"
      ]
    }
  }
}
]
}

```

```

{
  "branch": "actual",
  "contextId": "ecba1baa-66d1-4548-8c74-6012bea9b838",
  "indexingRequired": true,
  "tags": {},
  "indexInSync": true,
  "entityData": [
    {
      "name": "PermissionsConf",
      "value": {
        "permissions": {
          {
            "###Zoran_Admin_Token###": {
              "title": "Admin",
              "can": "*"
            },
            "###Zoran_Role_Engineer_Token###": {
              "title": "Role Engineer",
              "can": [
                "SHOW__ROLE_PAGE",
                "SEARCH__ALL_ROLES",
                "CREATE__ROLE",
                "UPDATE__ROLE",
                "DELETE__ROLE",
                "SHOW__ENTITLEMENT",
                "SHOW__USER",
                "SHOW__CERTIFICATIONS",
                "STATS_ALL__USERS",
                "SEARCH_ALL__USERS",
                "SEARCH_ALL__ENTITLEMENTS",
                "SEARCH__ROLE_USERS",
                "SEARCH__ROLE_ENTITLEMENTS",
                "SEARCH__ROLE_JUSTIFICATIONS",
                "SHOW_JUSTIFICATIONS",
                "SHOW_ROLE_METADATA",
                "SHOW_ROLE_ATTRIBUTES",
                "WORKFLOW__REQUESTS",
                "WORKFLOW__TASKS",
                "WORKFLOW__TASK_APPROVE"
              ]
            },
            "###Zoran_Role_Owner_Token###": {
              "title": "Role Owner",
              "can": [
                "SHOW__ROLE_PAGE",
                "SEARCH__ROLES",
                "CREATE__ROLE",
                "UPDATE__ROLE",
                "DELETE__ROLE",
                "SHOW__ENTITLEMENT",

```

```

        "SHOW__USER",
        "SHOW__CERTIFICATIONS",
        "STATS__USERS",
        "SEARCH_ALL__USERS",
        "SEARCH_ALL__ENTITLEMENTS",
        "SEARCH__ROLES",
        "SEARCH__ROLE_USERS",
        "SEARCH__ROLE_ENTITLEMENTS",
        "SEARCH__ROLE_JUSTIFICATIONS",
        "SHOW_JUSTIFICATIONS",
        "SHOW_ROLE_METADATA",
        "SHOW_ROLE_ATTRIBUTES",
        "WORKFLOW__REQUESTS",
        "WORKFLOW__TASKS",
        "WORKFLOW__TASK_APPROVE"
    ]
},
"###Zoran_Role_Auditor_Token###":
{
    "title": "Role Auditor",
    "can": [
        "SEARCH__ALL_ROLES",
        "STATS_ALL__USERS",
        "SEARCH_ALL__USERS",
        "SEARCH_ALL__ENTITLEMENTS",
        "SEARCH__ROLE_USERS",
        "SEARCH__ROLE_ENTITLEMENTS",
        "SEARCH__ROLE_JUSTIFICATIONS",
        "SHOW_JUSTIFICATIONS",
        "SHOW_ROLE_METADATA",
        "SHOW_ROLE_ATTRIBUTES",
        "WORKFLOW__REQUESTS",
        "WORKFLOW__TASKS"
    ]
},
"###Zoran_Application_Owner_Token###":
{
    "title": "Application Owner",
    "can": [
        "SHOW__APPLICATION_PAGE",
        "SEARCH__USER",
        "SEARCH__ENTITLEMENTS_BY_APP_OWNER",
        "SHOW_OVERVIEW_PAGE",
        "SHOW__ENTITLEMENT",
        "SHOW__ENTITLEMENT_USERS",
        "SHOW__APP_OWNER_FILTER_OPTIONS",
        "SHOW__ENTT_OWNER_UNSCORED_ENTITLEMENTS",
        "SHOW__ENTT_OWNER_PAGE",
        "SHOW__ENTT_OWNER_USER_PAGE",
        "SHOW__ENTT_OWNER_ENT_PAGE",
        "SHOW__USER_ENTITLEMENTS",
        "SHOW__RULES_BY_APP_OWNER",
        "REVOKE__CERTIFY_ACCESS",
        "SHOW__USER",
        "SHOW__CERTIFICATIONS"
    ]
}

```

```

    ]
  },
  "###Zoran_Entitlement_Owner_Token###":
  {
    "title": "Entitlement Owner",
    "can": [
      "SEARCH__ENTITLEMENTS_BY_ENTT_OWNER",
      "SHOW_OVERVIEW_PAGE",
      "SHOW__ENTITLEMENT",
      "SHOW__ENTITLEMENT_USERS",
      "SHOW__ENTT_OWNER_FILTER_OPTIONS",
      "SHOW__ENTT_OWNER_UNSCORED_ENTITLEMENTS",
      "SHOW__ENTT_OWNER_PAGE",
      "SHOW__ENTT_OWNER_USER_PAGE",
      "SHOW__ENTT_OWNER_ENT_PAGE",
      "SHOW__USER_ENTITLEMENTS",
      "SHOW__RULES_BY_ENTT_OWNER",
      "REVOKE__CERTIFY_ACCESS",
      "SHOW__USER",
      "SHOW__CERTIFICATIONS",
      "LOOKUP_USER",
      "SEARCH__ROLE_USERS",
      "SEARCH__ROLE_ENTITLEMENTS",
      "SEARCH__ROLE_JUSTIFICATIONS",
      "SHOW_ROLE_METADATA",
      "SHOW_ROLE_ATTRIBUTES",
      "WORKFLOW__REQUESTS",
      "WORKFLOW__TASKS",
      "WORKFLOW__TASK_APPROVE"
    ]
  },
  "###Zoran_Executive_Token###":
  {
    "title": "Executive",
    "can": [
      "SEARCH__USER",
      "SHOW__ASSIGNMENTS_STATS",
      "SHOW__COMPANY_PAGE",
      "SHOW__COMPANY_ENTITLEMENTS_DATA",
      "SHOW__CRITICAL_ENTITLEMENTS",
      "SHOW__ENTITLEMENT_AVG_GROUPS",
      "SHOW__USER_ENTITLEMENTS"
    ]
  },
  "###Zoran_Supervisor_Token###":
  {
    "title": "Supervisor",
    "can": [
      "SEARCH__USER",
      "SHOW_OVERVIEW_PAGE",
      "SHOW__SUPERVISOR_FILTER_OPTIONS",
      "SHOW__SUPERVISOR_PAGE",
      "SHOW__SUPERVISOR_ENTITLEMENT_USERS",
      "SHOW__SUPERVISOR_USER_ENTITLEMENTS",
      "SHOW__SUPERVISOR_UNSCORED_ENTITLEMENTS",

```

```
        "SEARCH__SUPERVISOR_USER_ENTITLEMENTS",
        "REVOKE__CERTIFY_ACCESS",
        "SHOW__ENTITLEMENT",
        "SHOW__USER",
        "SHOW__CERTIFICATIONS"
    ]
},
"###Zoran_User_Token###":
{
    "title": "User",
    "can": [
        "SHOW__ENTITLEMENT",
        "SHOW__USER",
        "SHOW__CERTIFICATIONS"
    ]
},
"###Zoran_Service_Connector_Token###":
{
    "title": "Service Connector",
    "can": [
        "SERVICE_CONNECTOR",
        "SHOW__API_KEY_MGMT_PAGE",
        "SHOW__ENTITLEMENT",
        "SHOW__USER",
        "SHOW__CERTIFICATIONS",
        "SHOW__RULES",
        "REVOKE__CERTIFY_ACCESS"
    ]
}
}
```

7. Verify your changes by opening the permissions object as shown in step 1c.

2. Next, update the JAS container environment variables:

1. On the instance where Docker is running, create a backup of the `/opt/autoid/res/jas/docker-compose.yml` file, and edit the variables in the environment section. For example, change the following variables:

From:

- OIDC_ENABLED=False
- GROUPS_ATTRIBUTE=_groups
- OIDC_JWKS_URL=na

To:

```
- OIDC_ENABLED=True
- GROUPS_ATTRIBUTE=groups
- OIDC_JWKS_URL= <Same value as in the zoran-api. Refer to step 3 below>
```

Note

The GROUPS_ATTRIBUTE variable must match the OIDC_GROUPS_ATTRIBUTE variable used in the docker-compose.yml file.

2. Remove the running JAS container and re-deploy:

```
docker stack rm jas
docker stack deploy --with-registry-auth --compose-file /opt/autoid/res/jas/docker-compose.yml
jas
```

1. Next, update the **zoran-api** container environment variables:

1. On the instance where Docker is running, create a backup of the `/opt/autoid/res/api/docker-compose.yml` file, and edit the following variables in the file replacing the `\$\{...\}` placeholders:

```
- OIDC_ISSUER=${OIDC_ISSUER}
- OIDC_AUTH_URL=${OIDC_AUTH_URL}
- OIDC_TOKEN_URL=${OIDC_TOKEN_URL}
- OIDC_USER_INFO_URL=${OIDC_USER_INFO_URL}
- OIDC_CLIENT_ID=${OIDC_CLIENT_ID}
- OIDC_CLIENT_SECRET=${OIDC_CLIENT_SECRET}
- OIDC_CALLBACK_URL=${OIDC_CALLBACK_URL}
- OIDC_JWKS_URL=${OIDC_JWKS_URL}
- OIDC_CLIENT_SCOPE=${OIDC_CLIENT_SCOPE}
- OIDC_GROUPS_ATTRIBUTE=${OIDC_GROUPS_ATTRIBUTE}
- OIDC_UID_ATTRIBUTE=${OIDC_UID_ATTRIBUTE}
- OIDC_END_SESSION_ENDPOINT=${OIDC_END_SESSION_ENDPOINT}
- OIDC_LOGOUT_REDIRECT_URL=${OIDC_LOGOUT_REDIRECT_URL}
```

For example, PingOne Autonomous Identity displays something similar below (the example uses Azure links and object IDs):

```

version: 3
services:
  zoran-api:
    image: "gcr.io/forgerock-autoid/zoran-api:2021.8.2"
    networks:
      - swarm_network
    volumes:
      - /opt/autoid/mounts/api/cert:/opt/app/cert/
      - /opt/autoid/mounts/api/logs:/opt/app/logs/
      - /opt/autoid/mounts/jwt:/opt/jwt/
      - /opt/autoid/mounts/jas-tls:/jas-tls/
      - /opt/autoid/mounts/signature:/opt/signature
    dns:
      - 8.8.8.8
      - 8.8.4.4
      - 169.254.169.254
    environment:
      - CONSUL_SERVER_URL=consul-client
      - LOCAL_AUTH_MODE=True
      - LOCAL_AUTH_MODE_PASSWORD=Welcome123
      - LDAP_URL=${LDAP_URL}
      - LDAP_BINDDN=${LDAP_BINDDN}
      - LDAP_BINDCREDENTIALS=${openldap_pw}
      - LDAP_SEARCHBASE=${LDAP_BASE_DN}
      - LDAP_GROUPSEARCHBASE=${LDAP_GROUPSEARCHBASE}
      - LDAP_DOMAIN=${LDAP_DOMAIN}
      - JWT_EXPIRY=30 minutes
      - JWT_SECRET_FILE=${JWT_SECRET_FILE}
      - JWT_KEY_ALGORITHM=RS512
      - JWT_KEY_ID=${JWT_KEY_ID}
      - JWT_PRIVATE_KEY_PASSPHRASE=${JWT_PRIVATE_KEY_PASSPHRASE}
      - JWT_PRIVATE_KEY_FILE=/opt/jwt/jwtprivate.pem
      - JWT_PUBLIC_KEY_FILE=/opt/jwt/jwtpublic.pem
      - LOG_DIR=/opt/app/logs
      - ACCESS_LOG_ENABLED=True
      - NODE_ENV=production
      - OIDC_ISSUER=https://login.microsoftonline.com/f7e/v2.0
      - OIDC_AUTH_URL=https://login.microsoftonline.com/f7e/v2.0/authorize
      - OIDC_TOKEN_URL=https://login.microsoftonline.com/f7e/v2.0/token
      - OIDC_USER_INFO_URL=https://graph.microsoft.com/oidc/userinfo
      - OIDC_CLIENT_ID=
      - OIDC_CLIENT_SECRET=
      - OIDC_CALLBACK_URL=https://autoid-ui.forgerock.com/api/sso/finish
      - OIDC_JWKS_URL=https://login.microsoftonline.com/f7e/discovery/v2.0/keys
      - OIDC_CLIENT_SCOPE=openid profile
      - OIDC_GROUPS_ATTRIBUTE=groups
      - OIDC_UID_ATTRIBUTE=oid
      - OIDC_END_SESSION_ENDPOINT=https://login.microsoftonline.com/f7e/oidc/logout
      - OIDC_LOGOUT_REDIRECT_URL=https://login.microsoftonline.com/f7e/logout
      - ELASTICSEARCH_USER_PASSWORD=Welcome123
      - ELASTICSEARCH_URL=https://10.154.0.11:9200
      - ELASTICSEARCH_ASSIGNMENT_INDEX=entitlement-assignment
      - ELASTICSEARCH_SSL_ENABLED=True
      - ELASTICSEARCH_USE_TRUSTED_KEYSSTORE=True
      - ELASTICSEARCH_SSL_KEY_FILE=/opt/app/cert/esnode-key.pem
      - ELASTICSEARCH_SSL_CERT_FILE=/opt/app/cert/esnode.pem
      - ELASTICSEARCH_SSL_CA_FILE=/opt/app/cert/root-ca.pem
      - JAS_URL=https://jasnode:10001

```



Important

If you are configuring SSO only for the login mode, set `LOCAL_AUTH_MODE` to `false` (for example, `LOCAL_AUTH_MODE=false`). If you keep `LOCAL_AUTH_MODE` to `true`, PingOne Autonomous Identity defaults to `LocalAndSSO`, which uses the OIDC and email options for login.

2. Remove the running `zoran-api` Docker container and re-deploy:

```

docker stack rm api
docker stack deploy --with-registry-auth --compose-file /opt/autoid/res/api/docker-compose.yml
api

```

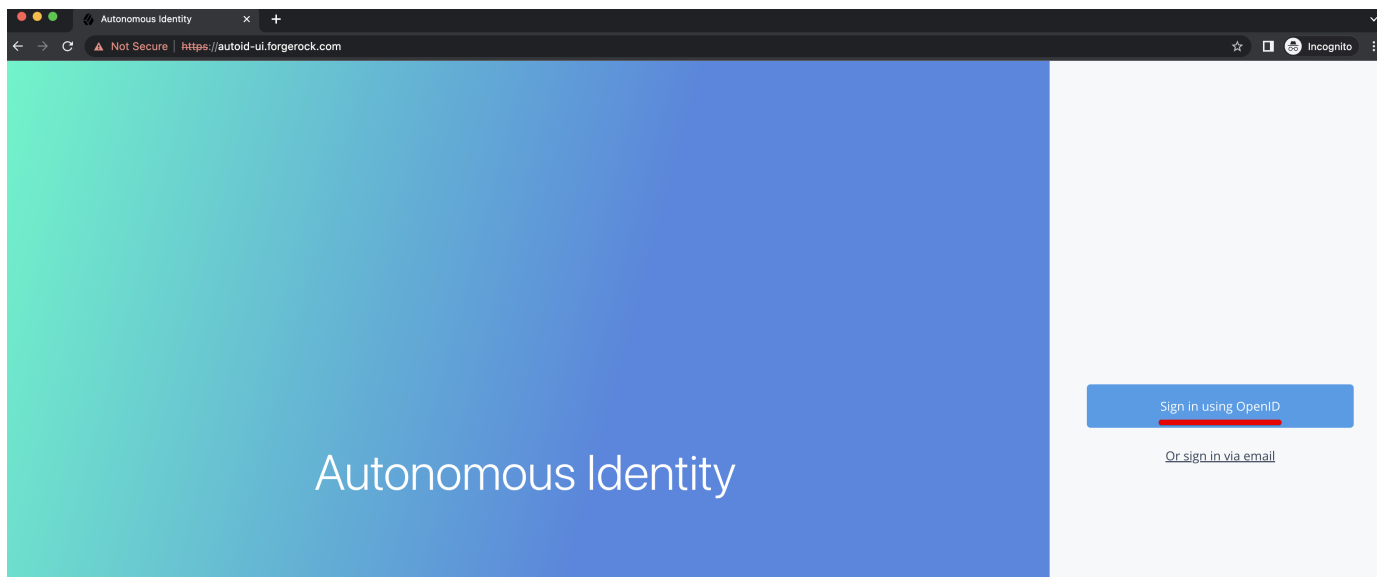
3. Restart the UI and Nginx Docker containers:

```

docker service update --force ui_zoran-ui
docker service update --force nginx_nginx

```

1. Open the PingOne Autonomous Identity UI to verify the SSO login.



Setting the session duration

By default, the session duration is set to 30 minutes. You can change this value at installation by setting the `JWT_EXPIRY` property in the `/autoid-config/vars.yml` file.

If you did not set the value at installation, you can make the change after installation by setting the `JWT_EXPIRY` property using the API service.

1. Log in to the Docker manager node.
2. Verify the `JWT_EXPIRY` property.

```
$ docker inspect api_zoran-api
```

3. Go to the API folder.

```
$ cd /opt/autoid/res/api
```

4. Edit the `docker-compose.yml` file and update the `JWT_EXPIRY` property. The `JWT_EXPIRY` property is set to minutes.
5. Redeploy the Docker stack API.

```
$ docker stack deploy --with-registry-auth --compose-file docker-compose.yml api
```

If the command returns any errors, such as "image could not be accessed by the registry," then try the following command:

```
$ docker stack deploy --with-registry-auth --resolve-image changed \  
  --compose-file /opt/autoid/res/api/docker-compose.yml api
```

6. Verify the new `JWT_EXPIRY` property.

```
$ docker inspect api_zoran-api
```

7. Log in to the Docker worker node.

8. Stop the worker node.

```
$ docker stop [container ID]
```

The Docker manager node re-initiates the worker node. Repeat this step on any other worker node.

Custom certificates

By default, PingOne Autonomous Identity uses self-signed certificates in its services. You can replace these self-signed certificates with a certificate issued by a Certificate Authority (CA). This section provides instructions on how to replace your self-signed certificates and also update your existing certificates when expired.

- [Update certificates on Cassandra](#)
- [Update certificates on MongoDB](#)
- [Update certificates on JAS](#)
- [Update the certificates on NGINX](#)
- [Update certificates on Opensearch](#)

Pre-requisites

The following items were used to test the custom certificate procedures:

- **Private key file.** The procedure uses a private key file `privkey.pem`.
- **Full trust chain.** The procedure also uses a full trust certificate chain, `fullchain.pem`.
- **Keystore password.** The procedure was tested using the keystore password is `Acc#1234`.
- **NGINX certificate.** The NGINX certificate must support subject alternative name (SAN) for the following domains:
 - `autoid-ui.<domain-name>`
 - `autoid-jas.<domain-name>`
 - `autoid-configuration-service.<domain-name>`
 - `autoid-kibana.<domain-name>`
 - `autoid-api.<domain-name>`
- **Domain name.** The domain name used in the procedure below is `certupdate.autoid.me`. Change the value in various places to the domain name applicable to your deployment.

- **PingOne Autonomous Identity version.** The procedures were tested on PingOne Autonomous Identity versions 2021.8.5 and 2021.8.6.

Update certificates on Cassandra

1. Create the Java keystore and truststore files for the server using `keytool`. The commands generate two JKS files: `cassandra-keystore.jks` and `cassandra-truststore.jks`. You need these files for configuring Cassandra and the Java API Service (JAS).

```
openssl pkcs12 -export -in fullchain.pem -inkey privkey.pem -out server.p12 -name cassandranode

keytool -importkeystore -deststorepass Acc#1234 -destkeypass Acc#1234 -destkeystore cassandra-keystore.jks -srckeystore server.p12 -srcstoretype PKCS12 -srcstorepass Acc#1234 -alias cassandranode

keytool -importcert -keystore cassandra-truststore.jks -alias rootCa -file fullchain.pem -noprompt -keypass Acc#1234 -storepass Acc#1234
```

2. Create the client certificate. The client certificate is used by external clients, such as JAS and `cqlsh` to authenticate to Cassandra. In the following example, use the same client certificate for the Cassandra nodes to authenticate with each other.

Note

You can create a different certificate, if desired, using similar steps.

```
# Create client.conf with following contents
[ req ]
distinguished_name = CA_DN
prompt              = no
default_bits        = 2048

[ CA_DN ]
C   = cc
O   = eng
OU  = cass
CN  = CA_CN

# Create client key and CSR
openssl req -newkey rsa:2048 -nodes -keyout client.key -out signing_request.csr -config client.conf

# Create client certificate
openssl x509 -req -CA fullchain.pem -CAkey privkey.pem -in signing_request.csr -out client.crt -days 3650 -CAcreateserial

# Import client cert into a Java keystore for JAS
openssl pkcs12 -export -in client.crt -inkey client.key -out client.p12 -name jas

keytool -importkeystore -deststorepass Acc#1234 -destkeypass Acc#1234 -destkeystore client-keystore.jks -srckeystore client.p12 -srcstoretype PKCS12 -srcstorepass Acc#1234 -alias jas
```

3. View the files that are needed in later steps:

```
$ ls -l .  
  
cassandra-keystore.jks  
cassandra-truststore.jks  
client.conf  
client.crt  
client.key  
client-keystore.jks  
client.p12  
fullchain.pem  
fullchain.srl  
privkey.pem  
server.p12  
signing_request.csr
```

4. Copy the following files to the `/opt/autoid/apache-cassandra-3.11.2/conf/certs` directory on each Cassandra node:

- **cassandra-keystore.jks**
- **cassandra-truststore.jks**
- **client-keystore.jks**

5. Copy the following files to the `<autoid-user-home-dir>/cassandra` directory on each Cassandra node:

- **client.key**
- **client.crt**
- **fullchain.pem**

6. Make the following edits in the `/opt/autoid/apache-cassandra-3.11.2/conf/cassandra.yaml` file on each Cassandra node:

1. Change the keystore and truststore paths in the `server_encryption_options` and `client_encryption_options` sections:

```
keystore: /opt/autoid/apache-cassandra-3.11.2/conf/certs/client-keystore.jks  
truststore: /opt/autoid/apache-cassandra-3.11.2/conf/certs/cassandra-truststore.jks
```

7. Update the `<autoid-user-home-dir>/cassandra/cqlshrc` file with the following edits:

```
[authentication]
username = zoranuser
password = <password>
[connection]
hostname = <ip address>
factory = cqlshlib.ssl.ssl_transport_factory

[ssl]
certfile = <autoid-user-home-dir>/cassandra/fullchain.pem
validate = false
version = SSLv23
# Next 2 lines must be provided when require_client_auth = true in the cassandra.yaml file
userkey = <autoid-user-home-dir>/cassandra/client_key.key
usercert = <autoid-user-home-dir>/cassandra/client.crt
```

8. Restart Cassandra.

```
ps auxf | grep cassandra
kill <pid>
cd /opt/autoid/apache-cassandra-3.11.2/bin
nohup ./cassandra >/opt/autoid/apache-cassandra-3.11.2/cassandra.out 2>&1 &
```

9. Make sure that Cassandra is running normally using `cqlsh`. Use your server's IP. :

```
$ cqlsh --ssl
Connected to Zoran Cluster at <server-ip>:9042.
[cqlsh 5.0.1 | Cassandra 3.11.2 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
zoranuser@cqlsh> describe keyspaces;

autonomous_iam  system_auth  system_distributed  autoid_analytics
autoid          system       system_traces      autoid_base
system_schema  autoid_meta  autoid_staging

zoranuser@cqlsh>
```

Update certificates on MongoDB

1. Create the keystore and truststore using `keytool`.

```
openssl pkcs12 -export -in fullchain.pem -inkey privkey.pem -out server.p12 -name mongonode

keytool -importkeystore -deststorepass Acc#1234 -destkeypass Acc#1234 -destkeystore mongo-client-keystore.jks -srckeystore server.p12 -srcstoretype PKCS12 -srcstorepass Acc#1234 -alias mongonode

keytool -importcert -keystore mongo-server-truststore.jks -alias rootCa -file fullchain.pem -noprompt -keypass Acc#1234 -storepass Acc#1234
```

2. Create a new mongodb.pem file.

```
cat fullchain.pem privkey.pem > mongodb.pem
```

3. Download the `isg root x1` root certificate from Lets Encrypt at <https://letsencrypt.org/certs/isgrootx1.pem>, and save it as `rootCA.pem`.
4. Back up the MongoDB certificates.

```
cd /opt/autoid/mongo/certs/
mkdir backup
mv mongodb.pem backup/
mv rootCA.pem backup/
mv mongo-*.jks backup
```

5. Copy the new certificates to the mongodb installation.

```
cp mongodb.pem /opt/autoid/mongo/certs/
cp rootCA.pem /opt/autoid/mongo/certs/
```

6. Restart Mongo DB.

```
sudo systemctl stop mongo-autoid
sudo systemctl start mongo-autoid
```

7. Check for logs for errors in `/opt/autoid/mongo/mongo-autoid.log`. The log may show connection errors until JAS has been updated and restarted.
8. Add the hostname to the hosts file. For example, we are using: `autoid-mongo.certupdate.autoid.me`.
9. Check the MongoDB connection from the command line.

```
mongo --tls --host autoid-mongo.certupdate.autoid.me --tlsCAFile /opt/autoid/mongo/certs/rootCA.pem
--tlsCertificateKeyFile /opt/autoid/mongo/certs/mongodb.pem --username mongoadmin
```

10. Back up and copy the new keystore and truststore to JAS.

```
cd /opt/autoid/mounts/jas
mkdir backup
mv mongo-*.jks backup

cp mongo-server-truststore.jks /opt/autoid/mounts/jas
cp mongo-client-keystore.jks /opt/autoid/mounts/jas
```

11. Update the JAS configuration. On each Docker manager and worker node, copy the following files to the `/opt/autoid/mount/jas` directory.
 - `mongo-client-keystore.jks`

- mongo-server-truststore.jks

Note

The certificates must exist on all Docker nodes (all managers and worker nodes).

12. On each Docker *manager* node, update `/opt/autoid/res/jas/docker-compose.yml` file and set the Mongo keystore, truststore, and host, and add the `extra_hosts` line as follows:

```
MONGO_KEYSTORE_PATH=/db-certs/mongo-client-keystore.jks
MONGO_TRUSTSTORE_PATH=/db-certs/mongo-server-truststore.jks
MONGO_HOST=autoid-mongo.certupdate.autoid.me
```

```
extra_hosts:
- "autoid-mongo.certupdate.autoid.me:<ip of mongodb host>"
```

13. Restart JAS.

```
docker stack rm jas nginx
docker stack deploy -c /opt/autoid/res/jas/docker-compose.yml jas
docker stack deploy -c /opt/autoid/res/nginx/docker-compose.yml nginx
```

14. Check JAS logs for errors.

```
docker service logs -f jas_jasnode
```

Update certificates on JAS

1. On each Docker manager and worker node, copy the following keystore and truststore files to `/opt/autoid/mounts/jas` directory:

- client-keystore.jks
- cassandra-truststore.jks

2. On each Docker *manager* node, update `/opt/autoid/res/jas/docker-compose.yml` with the correct keystore and truststore paths:

```
CASSANDRA_KEYSTORE_PATH=/db-certs/client-keystore.jks
CASSANDRA_TRUSTSTORE_PATH=/db-certs/cassandra-truststore.jks
```

3. Redeploy the JAS server.

```
docker stack rm jas
docker stack deploy jas -c /opt/autoid/res/jas/docker-compose.yml
```

4. Make sure JAS has no errors.

```
docker service logs -f jas_jasnode
```

Update the certificates on NGINX

1. Copy the following files to `/opt/autoid/mounts/nginx/cert/` :
 - `privkey.pem`
 - `fullchain.pem`
2. In the `/opt/autoid/mounts/nginx/nginx.conf` file, update the `ssl_certificate` and `ssl_certificate_key` properties as follows:

```
#SSL Settings
ssl_certificate      /etc/nginx/cert/fullchain.pem;
ssl_certificate_key  /etc/nginx/cert/privkey.pem;
```

3. Make sure that the domain names in the configuration files (`/opt/autoid/mounts/nginx/conf.d`) matches the domain names used for certificate generation.
4. Restart the Docker container.

```
docker stack rm nginx
docker stack deploy -c /opt/autoid/res/nginx/docker-compose.yml nginx
```

Update certificates on Opensearch

1. Create a keystore and truststore using `keystore` .

```
openssl pkcs12 -export -in fullchain.pem -inkey privkey.pem -out server.p12 -name esnodekey

keytool -importkeystore -deststorepass Acc#1234 -destkeypass Acc#1234 -destkeystore elastic-client-keystore.jks -srckeystore server.p12 -srcstoretype PKCS12 -srcstorepass Acc#1234 -alias esnodekey

keytool -importcert -keystore elastic-server-truststore.jks -alias rootCa -file fullchain.pem -noprompt -keypass Acc#1234 -storepass Acc#1234
```

2. Create backups.

```
cd /opt/autoid/certs/elastic
mkdir backup
mv *.jks backup
```

3. Copy the new jks files, `fullchain.pem`, `privkey.pem`, and `chain.pem` to `/opt/autoid/certs/elastic` .
4. Also, copy the `fullchain.pem`, `privkey.pem`, and `chain.pem` certificates to `/opt/autoid/elastic/Opensearch-1.3.13/config/`.

5. Update the following attributes in the `/opt/autoid/elastic/Opensearch-1.3.13/config/elasticsearch.yml` file:

```
Opensearch_security.ssl.transport.pemcert_filepath: fullchain.pem
Opensearch_security.ssl.transport.pemkey_filepath: privkey.pem
Opensearch_security.ssl.transport.pemtrustedcas_filepath: chain.pem

Opensearch_security.ssl.http.pemcert_filepath: fullchain.pem
Opensearch_security.ssl.http.pemkey_filepath: privkey.pem
Opensearch_security.ssl.http.pemtrustedcas_filepath: chain.pem

Opensearch_security.nodes_dn:
  - CN=elastic.certupdate.autoid.me
```

6. Restart Opensearch on all Opensearch nodes:

```
sudo systemctl restart elastic
```

7. Check `/opt/autoid/elastic/logs/elasticcluster.log` for errors. The file shows any certificate error until all nodes have been restarted.

8. In the `/opt/autoid/res/jas/docker-compose.yml` file, add the following:

```
extra_hosts:
  - "elastic.certupdate.autoid.me:<ip of ES host>"

update ES_HOST env var:
ES_HOST=elastic.certupdate.autoid.me
```

9. Restart the JAS container:

```
docker stack rm jas
docker stack rm nginx
docker stack deploy -c /opt/autoid/res/jas/docker-compose.yml jas
docker stack deploy -c /opt/autoid/res/nginx/docker-compose.yml nginx
```

10. Test the connection from the JAS container to Opensearch:

```
docker container exec -it <jas container id> sh
apk add curl
curl -v --cacert /elastic-certs/fullchain.pem -u elasticadmin https://elastic.certupdate.autoid.me:9200
```

11. Update the configuration in the JAS service using curl:

1. First log in using `curl`.

```
curl -X POST https://autoid-ui.certupdate.autoid.me:443/api/authentication/login -k -H
'Content-Type: application/json' -H 'X-TENANT_ID: <tenant_id >' -d '{
  "username": "bob.rodgers@forgerock.com",
  "password": "Welcome123"
}'
```

2. Pull in the current configuration using `curl`.

```
curl -k -H "Content-Type: application/json" -H 'X-TENANT-ID: <tenant_id>' -H 'Authorization:
Bearer <bearer_token>' --request GET https://jasnode:10081/config/analytics_env_config
```

3. Modify value for `elasticsearch` to `"host": elastic.certupdate.autoid.me``.

4. Push the updated configuration:

```
curl -k -H "Content-Type: application/json" -H 'X-TENANT-ID: <tenant_id>' -H 'Authorization:
Bearer <bearer_token>' --request PUT https://jasnode:10081/config/analytics_env_config -d
'<updated json config>'
```

12. Update the environment variable in your `.bashrc` on all Opensearch nodes and Spark nodes:

```
ES_HOST=elastic.certupdate.autoid.me
```

Set entity definitions

Before you run analytics, you can add new attributes to the schema using the PingOne Autonomous Identity UI. Only administrators have access to this functionality.

1. Open a browser. If you set up your own url, use it for your login.

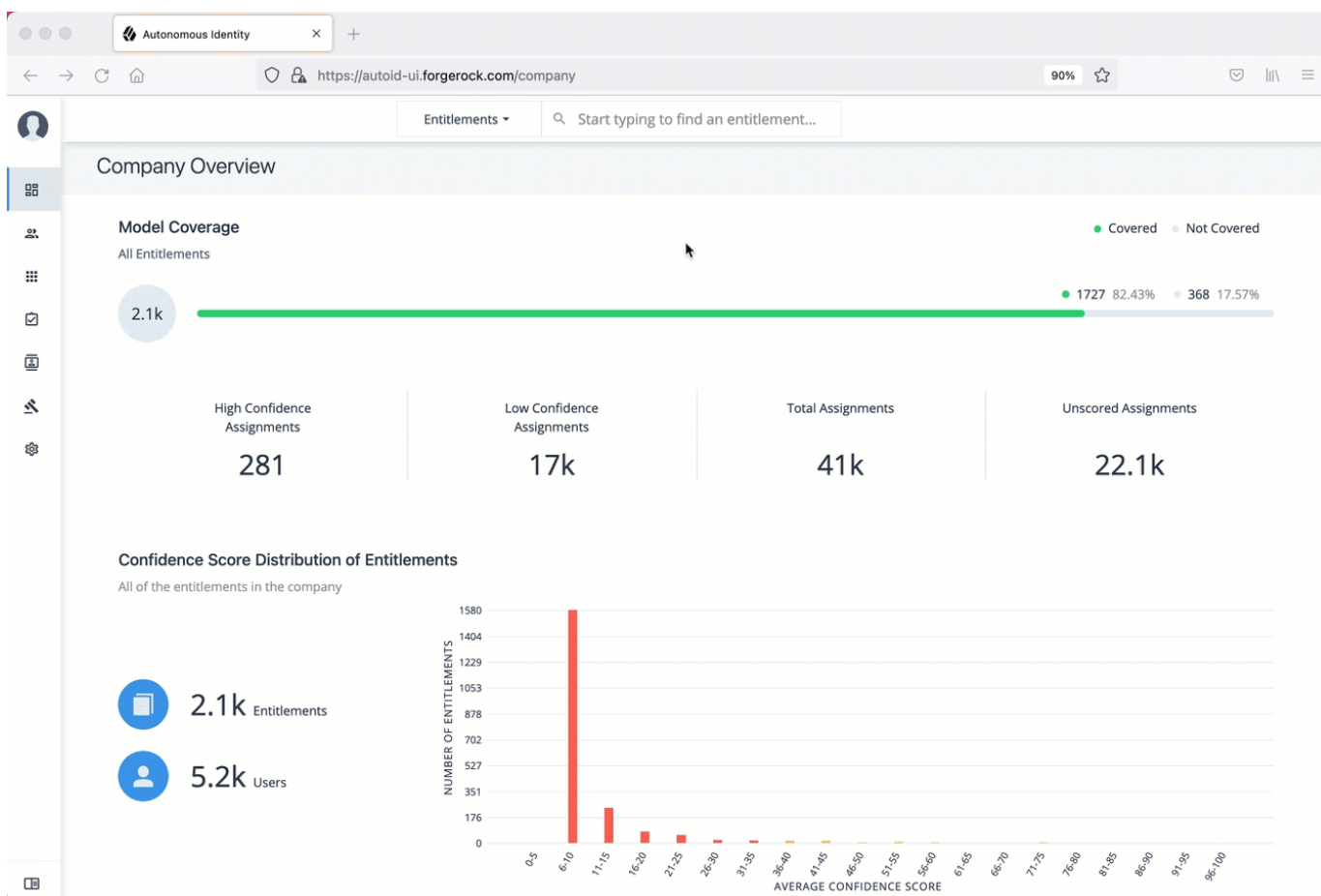
```
$ https://autoid-ui.forgerock.com/
```

2. Log in as an admin user, specific to your system. For example:

```
test user: bob.rodgers@forgerock.com
password: Welcome123
```

3. Click the Administration icon, indicated by the sprocket. Then, click **Entity Definitions**.
4. On the Entity Definitions page, click **Applications** to add any new attributes to the schema.
 1. Click the **Add attribute** button.
 2. In the Attribute Name field, enter the name of the new attribute. For example, `app_owner_id`.

3. In the Display Name field, enter a human-readable name of the attribute.
 4. Select the attribute type. The options are: **Text** , **Boolean** , **Integer** , **Float** , **Date** , and **Number** .
 5. Click **Searchable** if you want the attribute to be indexed and available in your filters.
 6. Click **Save**.
 7. Click **Save** again to apply your attribute.
 8. If you need to edit the attribute, click the **Edit** icon. If you need to remove the attribute, click the **Remove** icon. Note that attributes marked as **Required** cannot be removed.
5. Click **Assignments**, and repeat the previous steps.
 6. Click **Entitlements**, and repeat the previous steps.
 7. Click **Identities**, and repeat the previous steps.



1. Next, you must set the data sources. Refer to [Set Data Sources](#).

Set data sources

After defining any new attributes, you must set your data sources, so that PingOne Autonomous Identity can import and ingest your data. PingOne Autonomous Identity supports three types of data source files:

- **Comma-separated values (CSV).** A comma-separated values (CSV) file is a text file that uses a comma delimiter to separate each field value. Each line of text represents a record, consisting of one or more fields of data.
- **Java Database Connectivity (JDBC).** Java Database Connectivity (JDBC) is a Java API that connects to and executes queries on databases, like Oracle, MySQL, PostgreSQL, and MSSQL.
- **Generic.** Generic data sources are those data types from vendors that have neither CSV nor JDBC-based formats, such as JSON, or others.

Data source sync types

PingOne Autonomous Identity supports partial or incremental data ingestion for faster and efficient data uploads. The four types are `full`, `incremental`, `enrichment`, and `delete`, and are summarized below:

Table: Summary of Data Sync Types

Sync Type	Data Source	In AutoID	Result
Full	The records from the entity represents the full set of all records that you intend to ingest. For example: <ul style="list-style-type: none">• 0, "amy.user"• 1, "bob.user"	An existing table may have the following: <ul style="list-style-type: none">• 2, "walt.user"• 3, "kelly.user"	After the ingest job runs, all existing records are fully replaced: <ul style="list-style-type: none">• 0, "amy.user"• 1, "bob.user"
Incremental	The records from the entity represents the records that you want to add to AutoID. For example: <ul style="list-style-type: none">• 2, "walt.user"• 3, "kelly.user"	An existing table may have the following: <ul style="list-style-type: none">• 0, "amy.user"• 1, "bob.user"	After the ingest job runs, the records in the data source are added to the existing records: <ul style="list-style-type: none">• 0, "amy.user"• 1, "bob.user"• 2, "walt.user"• 3, "kelly.user"

Sync Type	Data Source	In AutoID	Result
Enrichment	<p>The records from the entity represents changes to existing data, such as adding a department attribute. No new objects are added, but here you want to edit or "patch" in new attributes to existing records:</p> <ul style="list-style-type: none"> • 0, "finance" • 1, "finance" • 2, "finance" 	<p>An existing table may have the following:</p> <ul style="list-style-type: none"> • 0, "amy.user" • 1, "bob.user" • 2, "walt.user" • 3, "kelly.user" 	<p>After the ingest job runs, the attributes in the data source is added to the existing records. If attributes exist, they get updated. If attributes do not exist, they do not get updated, but you can add also attributes using mappings:</p> <ul style="list-style-type: none"> • 0, "amy.user", "finance" • 1, "bob.user", "finance" • 2, "walt.user", "finance" • 3, "kelly.user"
Delete	<p>The records from the entity represent records to be deleted, identified by the primary key:</p> <ul style="list-style-type: none"> • 3, "kelly.user" 	<p>An existing table may have the following:</p> <ul style="list-style-type: none"> • 0, "amy.user", "finance" • 1, "bob.user", "finance" • 2, "walt.user", "finance" • 3, "kelly.user" 	<p>After the ingest job completes, the records with the primary key are deleted:</p> <ul style="list-style-type: none"> • 0, "amy.user", "finance" • 1, "bob.user", "finance" • 2, "walt.user", "finance"

CSV data sources

The following are general tips for setting up your comma-separated-values (CSV) files:

- Make sure you have access to your CSV files: `applications.csv`, `assignments.csv`, `entitlements.csv`, and `identities.csv`.
- You can review the [Data Preparation](#) chapter for more tips on setting up your files.

Set Up a CSV Data Source:

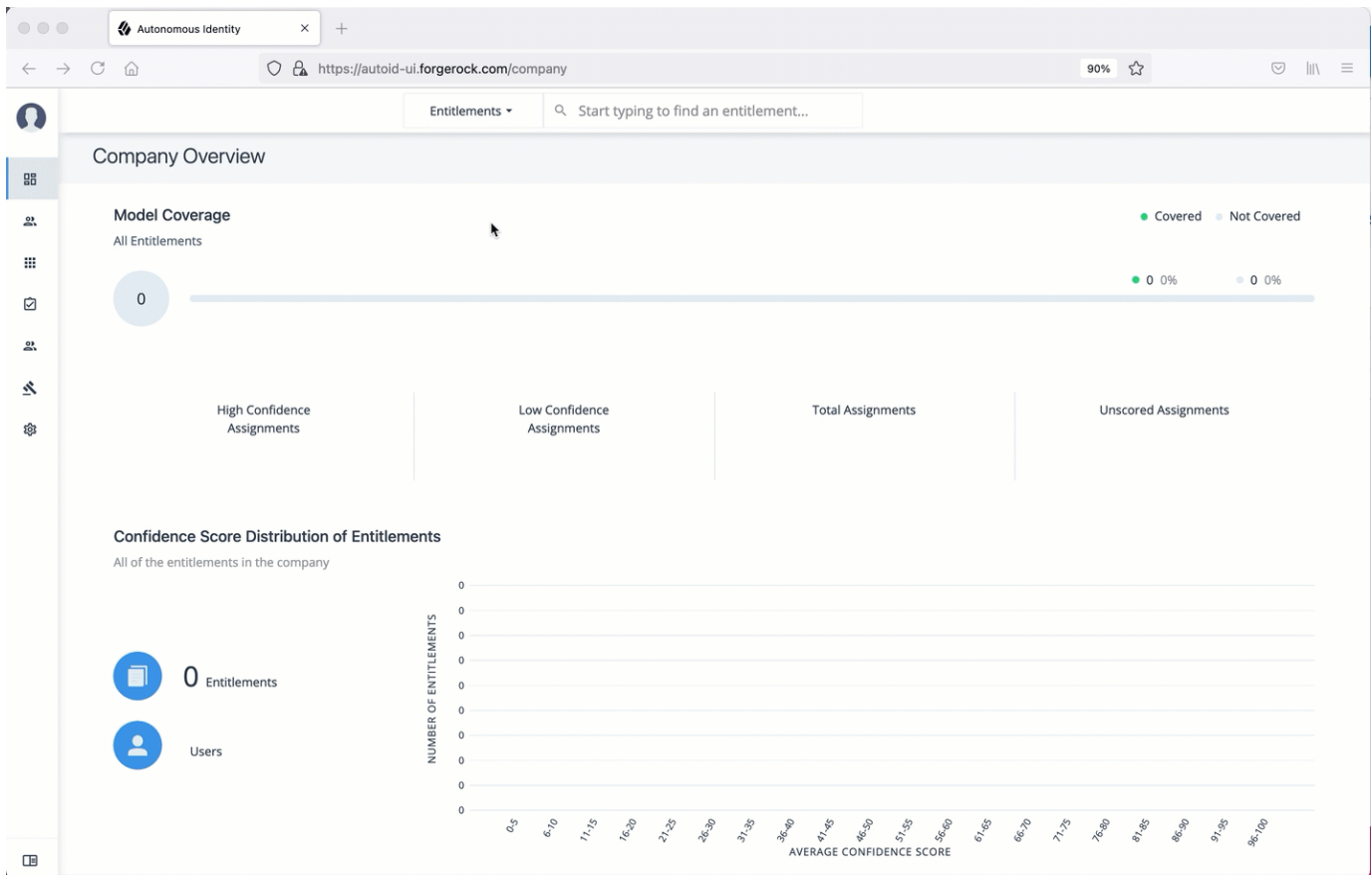
1. Log in to the PingOne Autonomous Identity UI as an administrator.
2. On the PingOne Autonomous Identity UI, click the **Administration > Data Sources > Add data source > CSV > Next**.
3. In the CSV Details dialog box, enter a human-readable name for your CSV file.
4. Select the Sync Type. The options are as follows:
 - **Full**. Runs a full replacement of data if any.
 - **Incremental**. Adds new records to existing data.

- **Enrichment.** Adds new attributes to existing data records.
- **Delete.** Delete any existing data objects.

5. Click **Add Object**, and then select the data source file.

1. Click **Applications**, enter the path to the `application.csv` file. For example, `/data/input/applications.csv`.
2. Click **Assignments**, enter the path to the `assignments.csv` file. For example, `/data/input/assignments.csv`.
3. Click **Entitlements**, enter the path to the `entitlements.csv` file. For example, `/data/input/entitlements.csv`.
4. Click **Identities**, enter the path to the `identities.csv` file. For example, `/data/input/identities.csv`.

6. Click **Save**.



7. Repeat the previous steps to add more CSV data source files if needed.

8. Next, you must set the attribute mappings. This is a critical step to ensure a successful analytics run. Refer to [Set Attribute Mappings](#).

JDBC Data Sources

The following are general tips for setting up your JDBC data sources (Oracle, MySQL, PostgreSQL, and MSSQL):

- When configuring your JDBC database, make sure you have properly "whitelisted" the IP addresses that can access the server. For example, you should include your local `autoid` instance and other remote systems, such as a local laptop.

- Make sure you have configured your database tables on your system: `applications`, `assignments`, `entitlements`, and `identities`.
- Make sure to make note of the IP address of your database server.

The following procedure assumes that you have set up PingOne Autonomous Identity with connectivity to a database:

Set Up a JDBC Data Source:

1. Log in to the PingOne Autonomous Identity UI as an administrator.
2. On the PingOne Autonomous Identity UI, click the **Administration** icon > **Data Sources** > **Add data source** > **JDBC** > **Next**.
3. In the JDBC Details dialog box, enter a human-readable name for your JDBC files.
4. Select the Sync Type. The options are as follows:
 - **Full**. Runs a full replacement of data if any.
 - **Incremental**. Adds new records to existing data.
 - **Enrichment**. Adds new attributes to existing data records.
 - **Delete**. Delete any existing data objects.
5. For Connection Settings, enter the following:
 1. **Database Username**. Enter a user name for the database user that connects to the data source.
 2. **Database Password**. Enter a password for the database user.
 3. **Database Driver**. Select the database driver. Options are:
 - `Oracle`
 - `Mysql`
 - `Postgresql`
 - `Mssqlserver`
 4. **Database Connect String**. Enter the database connection URI to the data source. For example, `jdbc:<Database Type>://<Database IP Address>/<Database Acct Name>`, where:
 - `jdbc` is the SQL driver type
 - `<Database Type>` is the database management system type. Options are: `oracle`, `mysql`, `postgresql`, or `sqlserver`.
 - `<Database IP Address>` is the database IP address
 - `<Database Acct Name>` is the database account name created in the database instance.

For example: * Oracle: `jdbc:oracle://35.180.130.161/autoid` * MySQL: `jdbc:mysql://35.180.130.161/autoid` * PostgreSQL: `jdbc:postgresql://35.180.130.161/autoid` * MSSQL: `jdbc:sqlserver://35.180.130.161;database=autoid`

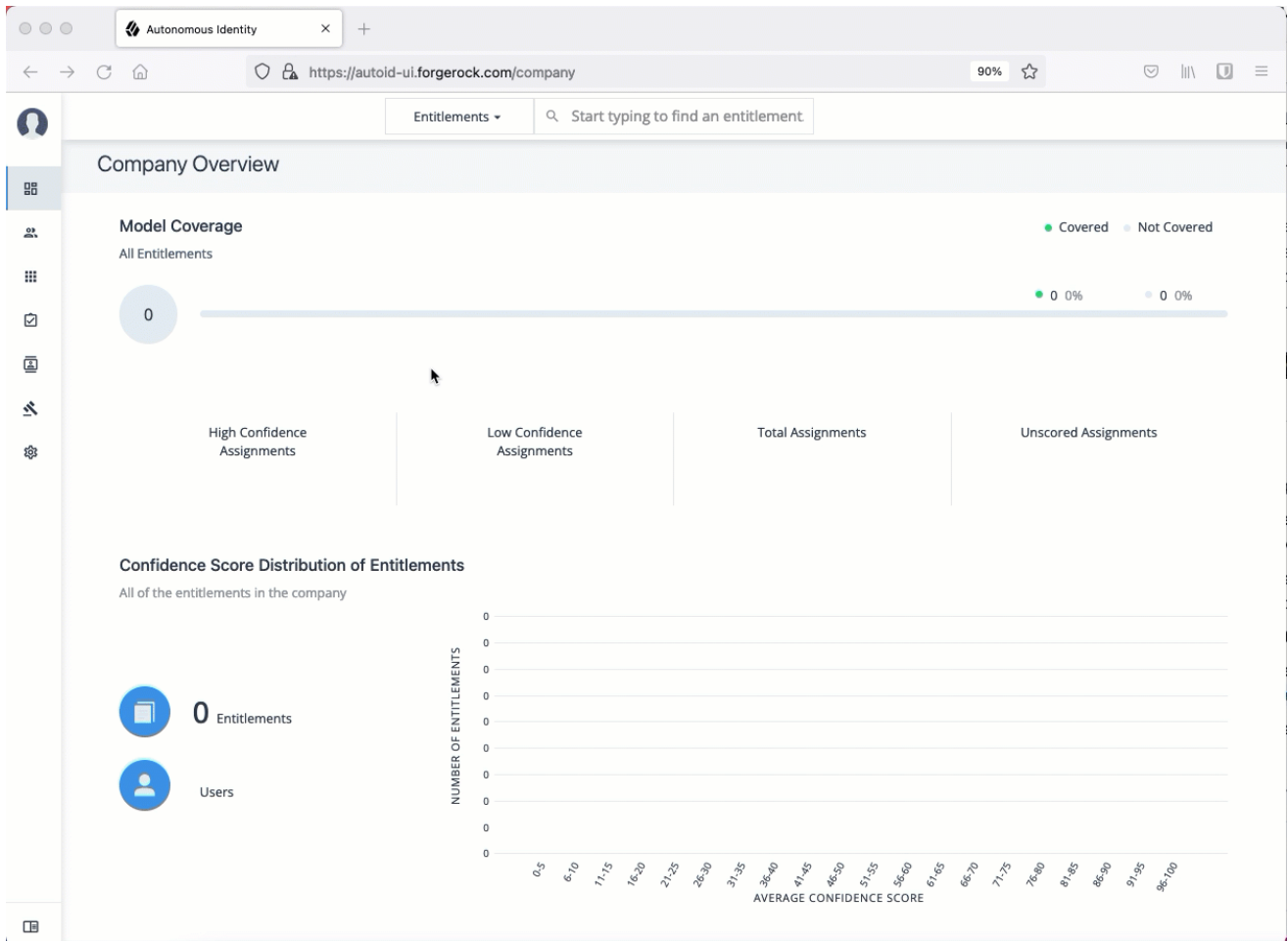
Note

There are other properties that you can use for each JDBC connection URI. Refer to the respective documentation for more information.

1. Click **Add Object**, and then select the data source file:

1. Click **Applications**, enter the path to the **APPLICATIONS** table. For example, using PostgreSQL, `SELECT * FROM public.applications`, where `public` is the PostgreSQL schema. Make sure to use your company's database schema.
2. Click **Assignments**, enter the path to the **ASSIGNMENTS** table. For example, `SELECT * FROM public.assignments`.
3. Click **Entitlements**, enter the path to the **ENTITLEMENTS** table. For example, `SELECT * FROM public.entitlements`.
4. Click **Identities**, enter the path to the **IDENTITIES** table. For example, `SELECT * FROM public.identities`.

2. Click **Save**.



3. If you are having connection issues, check the Java API Service (JAS) logs to verify the connection failure:

```
$ docker service logs -f jas_jasnode
```

The following entry can appear, which possibly indicates the whitelist was not properly set on the database server:

```
jas_jasnode.1.5gauc33o1nnn@autonomous-base-dev | java.lang.RuntimeException:  
org.postgresql.util.PSQLException: The connection attempt failed.  
. . .  
jas_jasnode.1.5gauc33o1nnn@autonomous-base-dev | Caused by: org.postgresql.util.PSQLException:  
The connection attempt failed.  
. . .  
Caused by: java.net.SocketTimeoutException: connect timed out
```

4. Next, you must set the attribute mappings. This is a critical step to ensure a successful analytics run. Refer to [Set Attribute Mappings](#).

Generic Data Sources

The following are general tips for setting up your generic data sources:

- Make sure you have configured data source files: `applications`, `assignments`, `entitlements`, and `identities`.
- Make sure you have the metadata (e.g., URL, prefix) required to access your generic data source files.

Set Up a Generic Data Source:

1. Log in to the PingOne Autonomous Identity UI as an administrator.
2. On the PingOne Autonomous Identity UI, click the **Administration** icon > **Data Sources** > **Add data source** > **Generic** > **Next**.
3. In the Generic Details dialog box, enter a human-readable name for your generic files.
4. Select the Sync Type. The options are as follows:
 - **Full**. Runs a full replacement of data if any.
 - **Incremental**. Adds new records to existing data.
 - **Enrichment**. Adds new attributes to existing data records.
 - **Delete**. Delete any existing data objects.
5. For Connection Settings, enter the settings to connect to your database server. For example:

```
{  
  "username": "admin",  
  "password": "Password123",  
  "connectURL": "http://identity.generic.com"  
}
```

6. Click **Add Object**, and then select the data source file:

1. Click **Applications**, enter the metadata for applications file. For example:

```
{
  "appMetaUrl": "http://identity.generic.com?q=applications&appName=Ac*",
  "prefix": "autoid"
}
```

2. Click **Assignments**, enter the metadata for the assignments file. For example:

```
{
  "appMetaUrl": "http://identity.generic.com?q=assignments&userId=*",
  "prefix": "autoid"
}
```

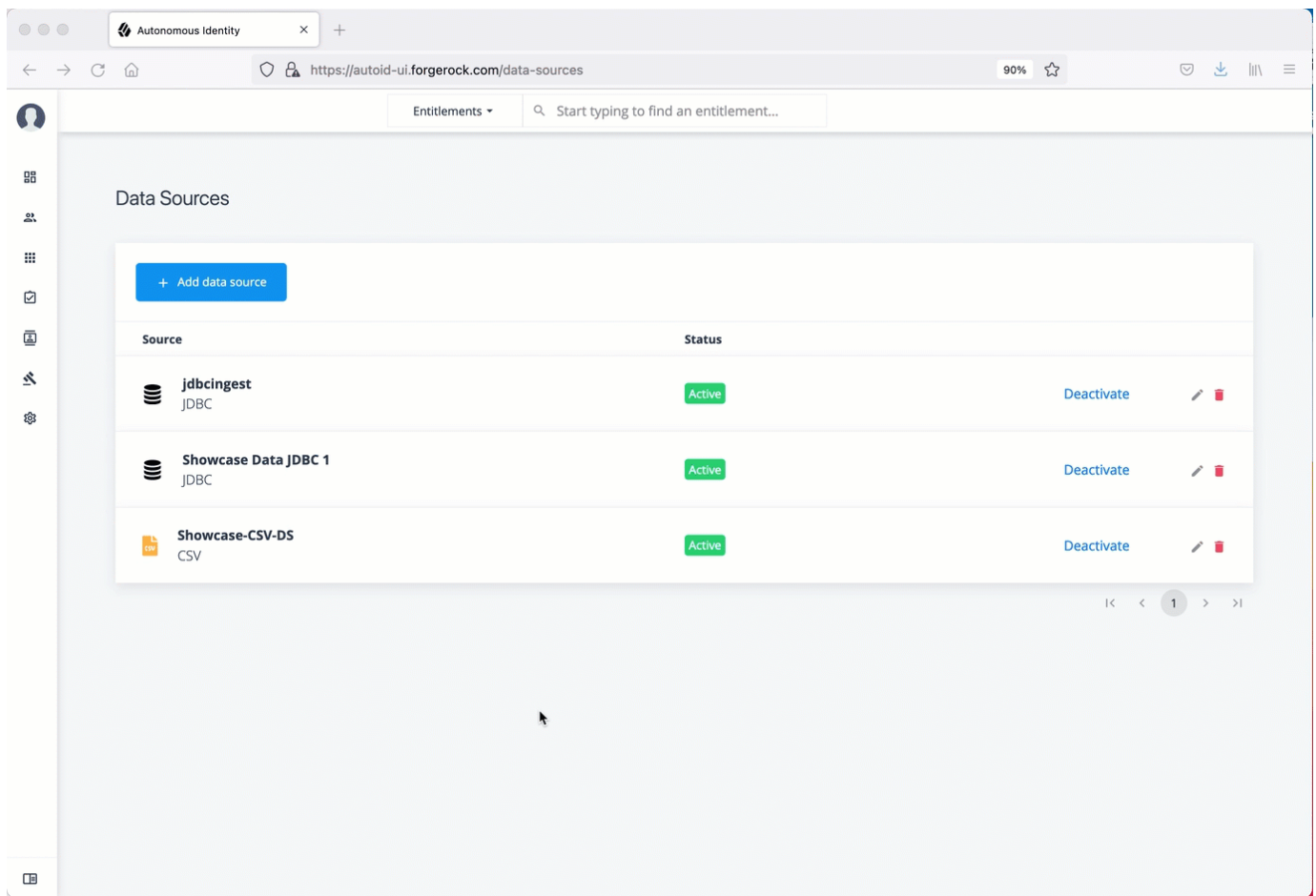
3. Click **Entitlements**, enter the metadata for the entitlements file. For example:

```
{
  "appMetaUrl": "http://identity.generic.com?q=entitlements&appId=*",
  "prefix": "autoid"
}
```

4. Click **Identities**, enter the metadata for the identities file. For example:

```
{
  "appMetaUrl": "http://identity.generic.com?q=identities&userId=*",
  "prefix": "autoid"
}
```

7. Click **Save**.



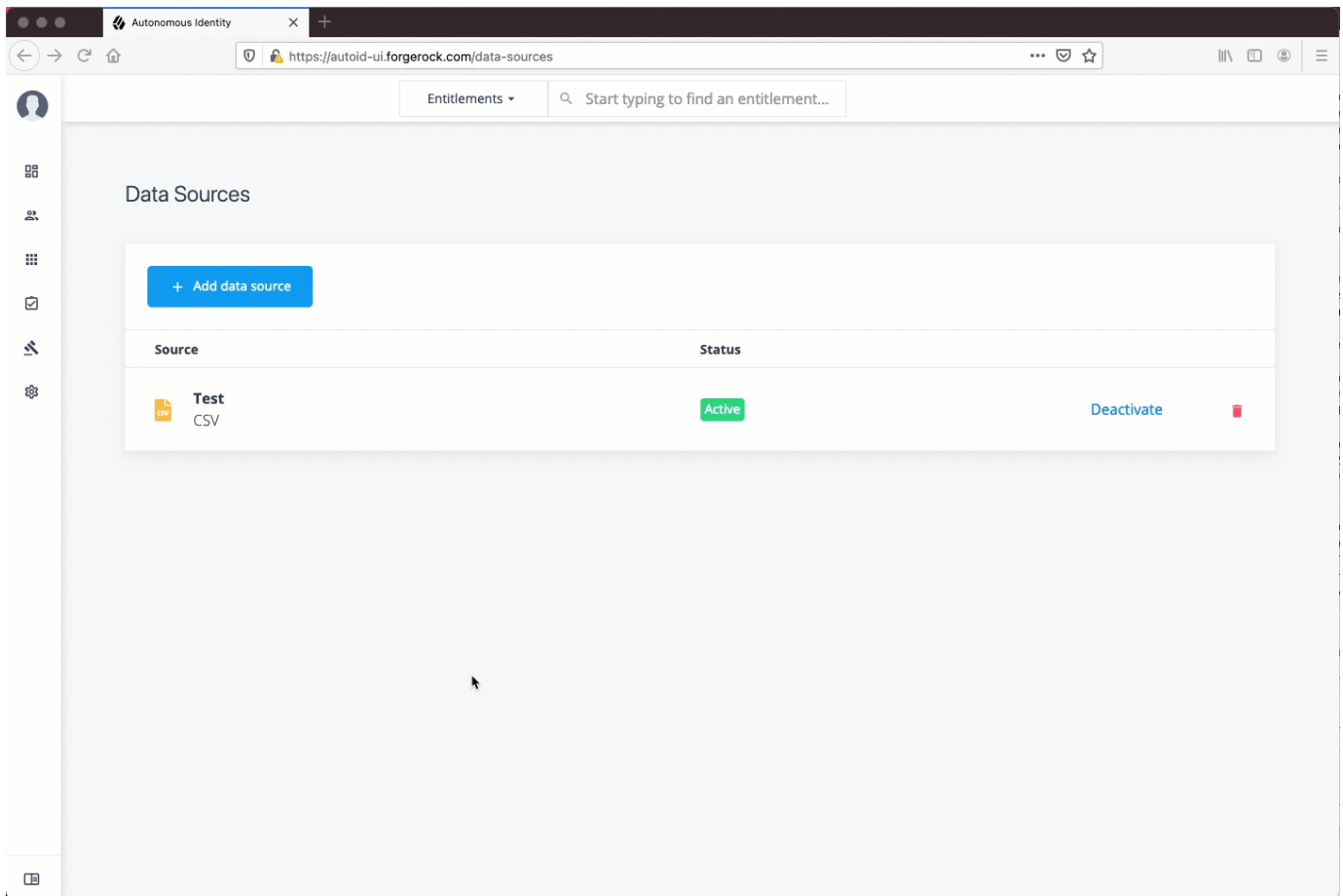
8. Repeat the previous steps to add more JDBC data source files if necessary.
9. Next, you must set the attribute mappings. This is a critical step to ensure a successful analytics run. Refer to [Set Attribute Mappings](#).

Set attribute mappings

After setting your data sources for your CSV files, you must map any attributes specific to each of your data files to the PingOne Autonomous Identity schema.

1. On the PingOne Autonomous Identity UI, click the **Administration** icon > **Data Sources**.
2. Click the specific data source file to map.
3. Click **Applications** to set up its attribute mappings.
 1. Click **Discover Schema** to view the current attributes in the schema, and then click **Save**.
 2. Click **Edit mapping** to set up attribute mappings. On the Choose an attribute menu, select the corresponding attribute to map to the required attributes. Repeat for each attribute.
 3. Click **Save**.
4. Click **Assignments** and repeat the previous steps.

5. Click **Entitlements** and repeat the previous steps.
6. Click **Identities** and repeat the previous steps.
7. Repeat the procedures for each data source file that you want to map.



1. Optional. Next, adjust the analytics thresholds. Refer to [Set Analytic Thresholds](#).

Set analytics thresholds

The PingOne Autonomous Identity UI now supports the configuration of the analytics threshold values to calculate confidence scores, predications, and recommendations.

Warning

In general, there is little reason to change the default threshold values. If you do edit these values, be aware that incorrect threshold values can negatively affect your analytics results.

There are six types of threshold settings that administrators can edit:

- **Confidence Score Thresholds.** *Confidence score thresholds* lets you define High, Medium, and Low confidence score ranges. PingOne Autonomous Identity computes a confidence score for each access assignment based on its machine learning algorithm. The properties are:

Table 1: Confidence Score Thresholds

Settings	Default	Description
Confidence Score Thresholds	<ul style="list-style-type: none"> High: 0.75 or 75% Medium: 0.35 or 35% 	<ul style="list-style-type: none"> Confidence scores from 75 to 100 are set to <i>High</i>. Confidence scores from 35 to 74 are set to <i>Medium</i>, and scores from 0 to 34 are set to <i>Low</i>.

- **Automation Score Threshold.** *Automation score threshold* is a UI setting determining if an approval button and checkbox appears before a justification rule on the Entitlement Details and Rules pages.

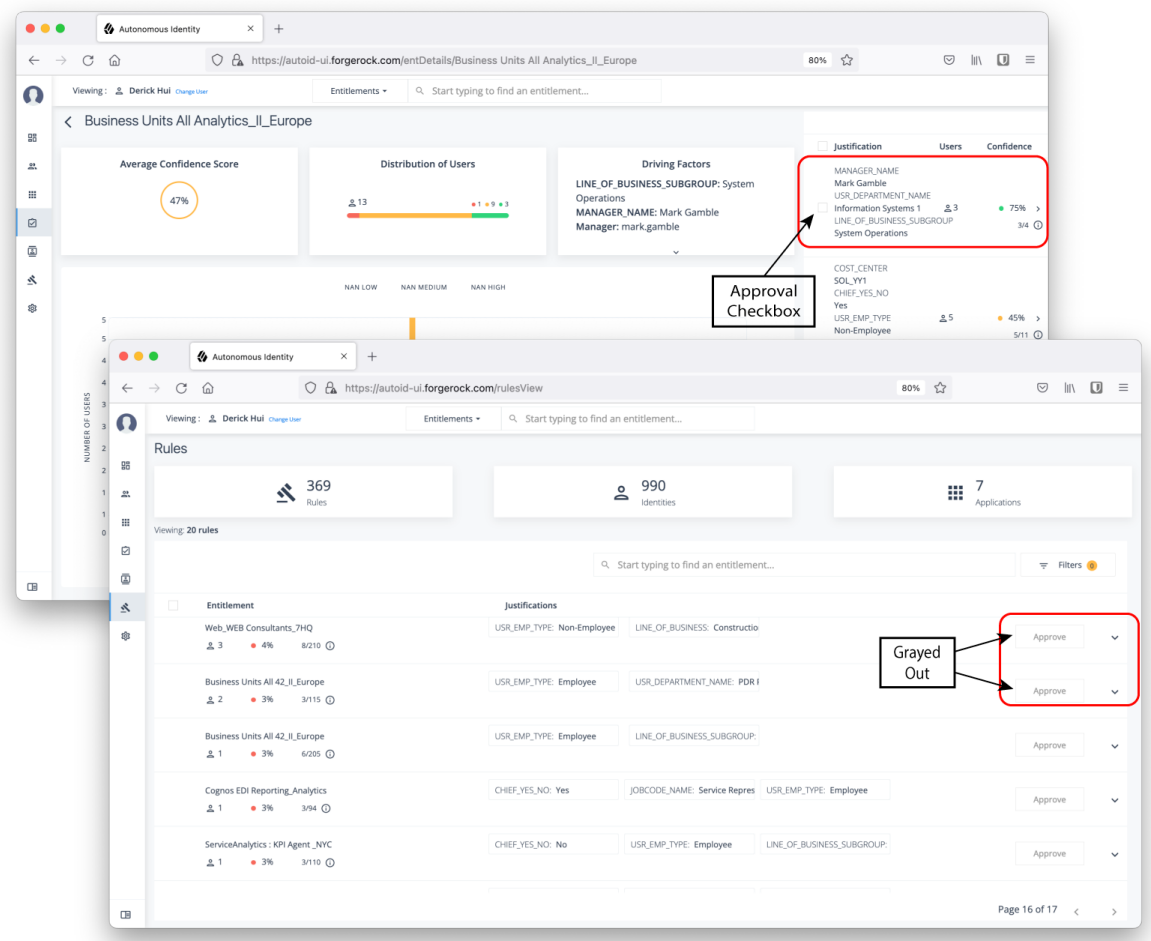


Table 2: Automation Score Threshold

Settings	Default	Description
Automation Score Threshold	0.5 or 50%	Specifies if any confidence score less than 50% will not give the user the option to approve the justification or rule.

- **Role Discovery Settings.** *Role discovery settings* determine the key factors for roles for inclusion in the role mining process. Roles are a collection of entitlements and their associated justifications and access patterns. This collection is produced from the output training rules.

Table 3: Role Discovery

Settings	Default	Description
Confidence Threshold	0.75 or 75%	Specifies the minimum rule confidence required for inclusion in the role mining process.
Entitlements Threshold	1	Specifies the minimum number of entitlements a role may contain. The PingOne Autonomous Identity role mining process does not produce candidate roles <i>below</i> this threshold value. For example, if the threshold is 2, there are no roles that contain only one entitlement.
Minimum Role Membership	30	Specifies the granularity of the role through its membership. For example, the default is 30, which means that no role produced can have fewer than 30 members.
Remove Redundant Access Patterns	Enabled	<p>Specifies a pruning process that removes redundant patterns when more general patterns can be retained.</p> <p>For example, if a user is an <i>Employee</i> AND in the department, <i>Finance</i>, they receive the <i>Excel</i> access entitlement. However, there may be a more general rule that provisions <i>Excel</i> access by simply being an <i>Employee</i>.</p> <ul style="list-style-type: none"> ◦ [ENT_Excel DEPT_Finance, EMP_TYPE_Employee] ◦ [ENT_Excel EMP_TYPE_Employee] <p>As a result, PingOne Autonomous Identity removes the first pattern and retains the latter more general rule.</p>

- **Training Settings.** Administrators can set the thresholds for the AI/ML training process, specifically the stemming process and general training properties.

Stemming: During the training process, PingOne Autonomous Identity generates rules by searching the data for if-then patterns that have a parent-child relationship in their composition. These if-then patterns are also known as *antecedent-consequent* relationships, which means *rule-entitlement* for PingOne Autonomous Identity.

Stemming is a process to remove any redundant final association rules output. For a rule to be stemmed, it must match the following criteria:

1. Rule B consequent must match Rule A consequent.
2. Rule B antecedent must be a superset of Rule A antecedent.
3. Rule B confidence score must be within a given range +/- (*offset*) of Rule A confidence score.

For example, the Payroll Report entitlement has two rules, each that involves the Finance department. All Dublin employees in the Finance department also get the entitlement. Stemming prunes the second rule (B) and retains the more general first rule (A).

ID	Consequent (ENT)	Antecedent (Rule)	Confidence Score
A	Payroll report	[Finance]	90%
B	Payroll report	[Dublin,Finance]	89%

Table 4: Training Settings

Type	Settings	Default	Description
Stemming Predictions			Determines stemming, or pruning, properties.
	Stemming Enabled	Enabled	Specifies if stemming occurs or not. Do not disable this feature.
	Stemming Offset	0.02	Specifies the confidence range (plus or minus) of one rule to another rule.
	Stemming Feature Size	3	Specifies the "up-to" maximum antecedent/justification size that may be the size priority of stemming. Because we want to retain the smallest rules possible, we start by prioritizing rules with an antecedent size of 1. We can then increase the antecedent size, iteratively, until we reach the Stemming Feature Size setting.
Batch Size		15000	Specifies the number of samples viewed is indicated by the batch size. An <i>epoch</i> is defined as the number of passes for a model to iterate through the entire dataset once and update its learning algorithm. To process the entire epoch, the model views a few samples at a time in batches.
Base Minimum Group		2	Specifies the minimum support value used in training. It is referred as <i>Base</i> , because it is used to find the minimum support value for the initial chunk of training.
Minimum Confidence		0.02	Specifies the lowest acceptable confidence score to be included in the entitlement-rule combination.
Number of Partitions		200	Specifies the number of partitions in a Spark configuration. A partition is a smaller chunk of a large dataset. Spark can run one concurrent task in a single partition. As a rule of thumb, the more partitions you have, the more work can be distributed among Spark worker nodes. In this case, small chunks of data are processed by each worker. In the case of fewer partitions, PingOne Autonomous Identity can process larger chunks of data.

- **Predictions Settings.** Administrators can set the thresholds for the recommendation and as-is prediction processes.

Table 5: Predictions Settings

Type	Settings	Default	Description
Recommendation Settings			Properties for setting the recommendation predictions settings.
	Threshold	0.75	Specifies the confidence score threshold to be considered for recommendation.
	Batch Size	1000	Specifies the number of rules and user entitlements processed at one time.
	Minimum Frequency	0	Specifies the minimum frequency for a rule to appear for consideration as a recommendation. During the first training stage, PingOne Autonomous Identity models the frequent itemsets that appear in the HR attributes-only of each user. Only rules that appear a minimum of N times are considered. The value of N is the <i>Minimum Frequency</i> .
As-Is Prediction Settings			Properties for setting the as-is predictions settings.
	Batch Size	15000	Specifies the number of rules and user entitlements processed at one time.
	Confidence Threshold	0	Specifies the confidence score threshold to be considered for an as-is prediction.
Minimum Rule Length		1	Specifies the minimum justification size for rules to be considered in predictions. You only would increase this property if you don't want a single rule overriding more specific or granular rules when determining access. For example, if the minimum rule length is 2, PingOne Autonomous Identity only uses the rule <i>DEPT_Finance, JOB_TITLE_Account_II</i> . However, if the default is kept at 1, the second and shorter rule can include a broader number of entitlement assignments. <ul style="list-style-type: none"> ◦ [ENT_AccountingSoftware DEPT_Finance, JOB_TITLE_Account_II] ◦ [ENT_AccountingSoftware EMP_TYPE_Employee]
Maximum Rule Length		10	Specifies the maximum justification size for rules to be considered in predictions. This property is a guardrail to keep rules that contain extremely large or complex justifications out of the prediction set.

Type	Settings	Default	Description
	Prediction Confidence Window	0.05	Specifies the range of acceptable values for a prediction confidence score. Rules with confidence scores outside the prediction confidence window range are filtered out. A confidence window is determined from the values set in the configuration file: max=maxConf, min=maxConf - pred_conf_window.

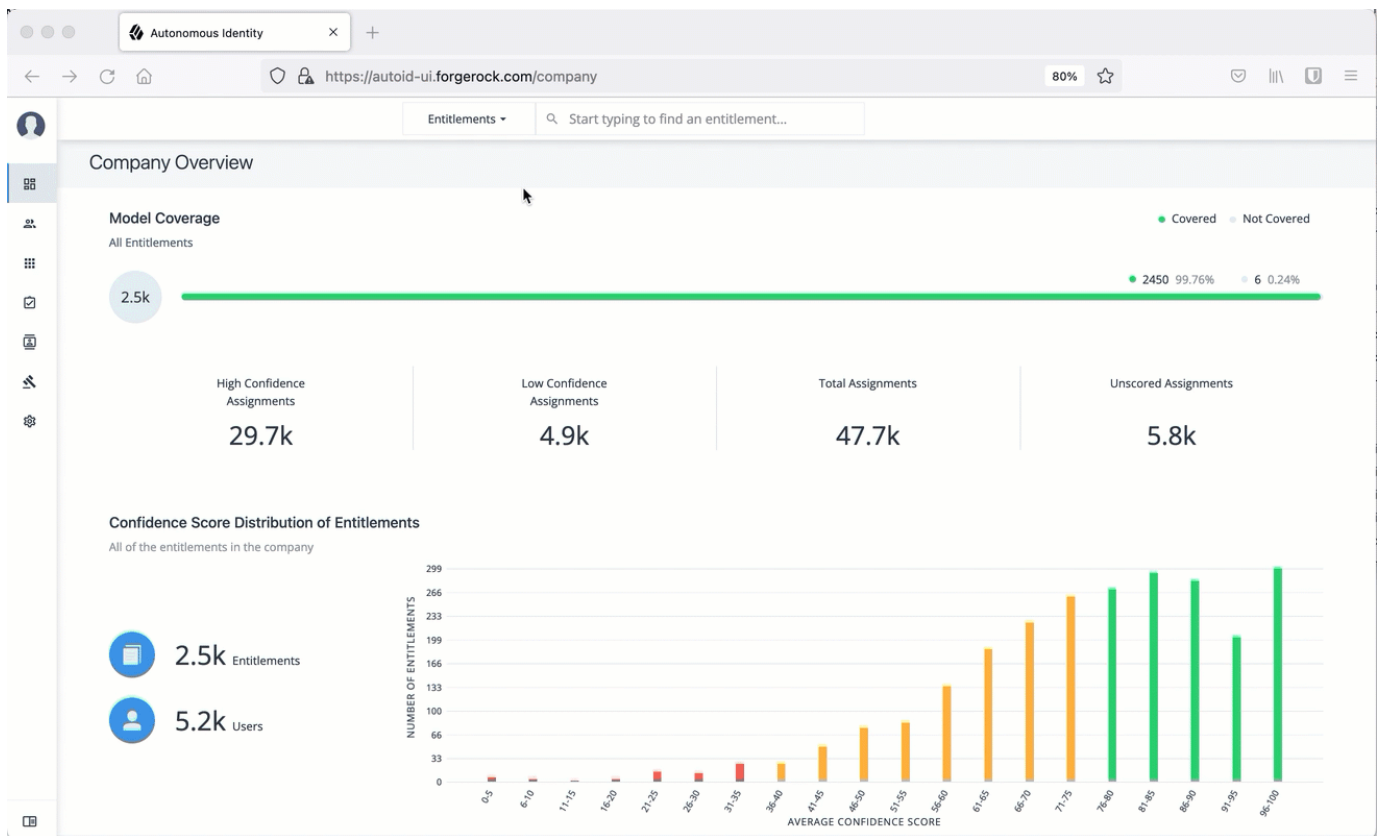
- **Analytics Spark Job Config.** Administrators can adjust the Apache Spark job configuration if needed.

Table 6: Analytics Spark Job Configuration

Settings	Default	Description
Driver Memory	2G	Specifies the amount of memory for the driver process.
Driver Cores	3	Specifies the number of cores to use for the driver process in cluster mode.
Executor Memory	3G	Specifies the amount of memory to use per executor process.
Executor Cores	6	Specifies the number of executor cores per worker node.

Configure Analytic Settings

1. Log in to the PingOne Autonomous Identity UI as an administrator.
2. On the PingOne Autonomous Identity UI, click **Administration**.
3. Click **Analytics Settings**.
4. Under Confidence Score Thresholds, click **Edit** next to the High threshold value, and then enter a new value. Click **Save**. Repeat for the Medium threshold value.
5. Under Automation Score Threshold, click **Edit** next to a threshold value, and then enter a new value.
6. Under Role Discovery Setting, click **Edit** next to a threshold value, and then enter a new value.
7. Under Training Settings, click **Edit** next to a threshold value, and then enter a new value.
8. Under Prediction Settings, click **Edit** next to a threshold value, and then enter a new value.
9. Under Analytics Spark Job Config, click **Edit** next to a threshold value, and then enter a new value.
10. Click **Save**.



1. Next, you can run the analytics. Refer [Run Analytics](#).

Run analytics

The Analytics pipeline is the heart of PingOne Autonomous Identity. The pipeline analyzes, calculates, and determines the association rules, confidence scores, predictions, and recommendations for assigning entitlements and roles to the users.

The analytics pipeline is an intensive processing operation that can take time depending on your dataset and configuration. To ensure an accurate analysis, the data needs to be as complete as possible with little or no null values. Once you have prepared the data, you must run a series of analytics jobs to ensure an accurate rendering of the entitlements and confidence scores.

Pre-analytics tasks

Before running the analytics, you must run the following pre-analytics steps to set up your datasets and schema using the PingOne Autonomous Identity UI:

- Add attributes to the schema. For more information, refer to [Set Entity Definitions](#).
- Define your datasources. PingOne Autonomous Identity supports different file types for ingestion: CSV, JDBC, and generic. You can enter more than one data source file, specifying the dataset location on your target machine. For more information, refer to [Set Data Sources](#).
- Define attribute mappings between your data and the schema. For more information, refer to [Set Attribute Mappings](#).
- Configure your analytics threshold values. For more information, refer to [Set Analytics Thresholds](#).

About the analytics process

Once you have finished the pre-analytics steps, you can start the analytics. The general analytics process is outlined as follows:

- **Ingest.** The ingestion job pulls in data into the system. You can ingest CSV, JDBC, and generic JSON files depending on your system.
- **Training.** The training job creates the association rules for each user-assigned entitlement. This is a somewhat intensive operation as the analytics generates a million or more association rules. Once the association rules have been determined, they are applied to user-assigned entitlements.
- **Role Mining.** The role mining job analyzes all existing entitlements and analyzes candidate configurations for new roles.
- **Predict As-Is.** The predict as-is job determines the current confidence scores for all assigned entitlements.
- **Predict Recommendation.** The predict recommendations job looks at all users who do not have a specific entitlement, but are good candidates to receive the entitlement based on their user attribute data.
- **Publish.** The publish run publishes the data to the backend Cassandra or MongoDB databases.
- **Create Assignment Index.** The create-assignment-index creates the PingOne Autonomous Identity index.
- **Run Reports.** You can run the create-assignment-index-report (report on index creation), anomaly (report on anomalous entitlement assignments), insight (summary of the analytics jobs), and audit (report on change of data).

Autonomous Identity Analytics Pipeline

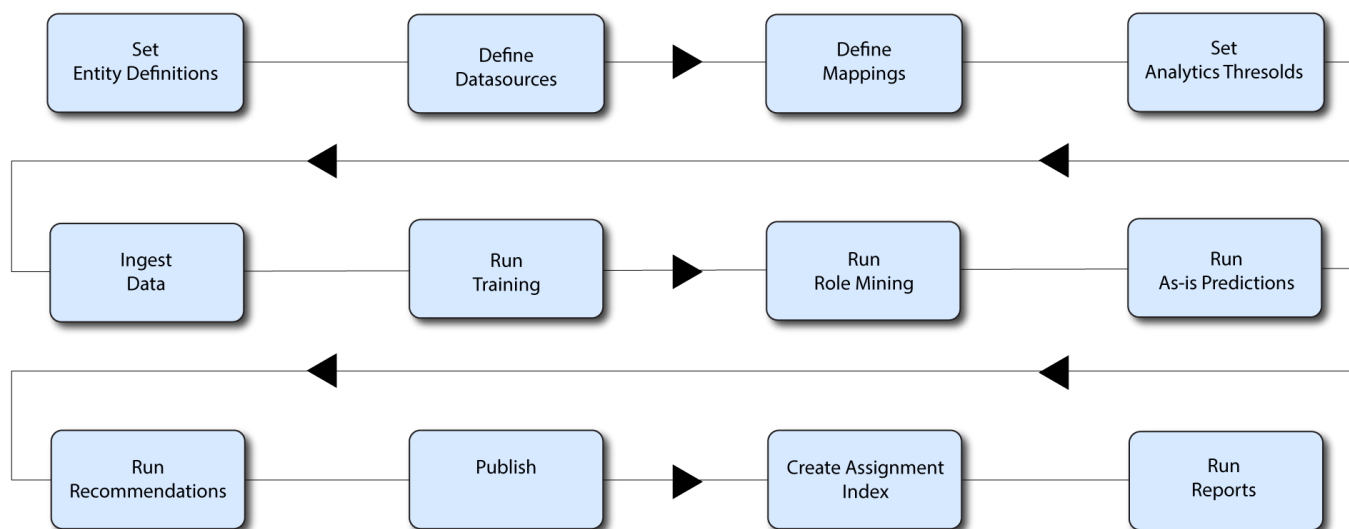


Figure 1. Autonomous Identity Analytics Pipeline Jobs



Note

The analytics pipeline requires that DNS properly resolve the hostname before its start. Make sure to set it on your DNS server or locally in your `/etc/hosts` file.

The following sections present the steps to run the analytics pipeline using the Jobs UI.

Note

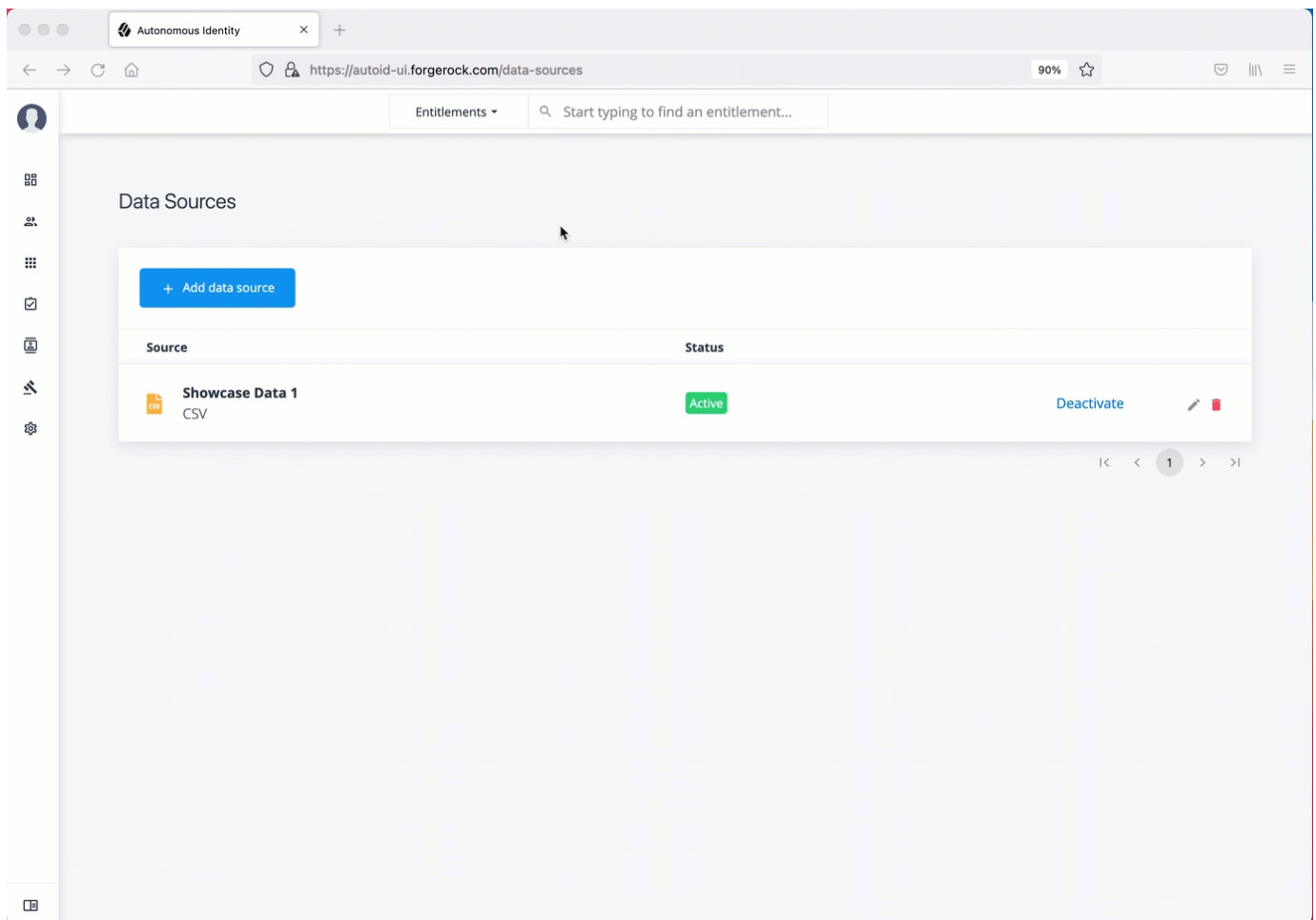
You can continue to use the command-line to run each step of the analytics pipeline. For instructions, refer to [Run analytics on the command Line](#).

Ingest the data files

At this point, you should have set your data sources and configured your attribute mappings. You can now run the initial analytics job to import the data into the Cassandra or MongoDB database.

Run ingest using the UI:

1. On the PingOne Autonomous Identity UI, click the Administration link, and then click **Jobs**.
2. On the Jobs page, click **New Job**. PingOne Autonomous Identity displays a job schedule with each job in the analytics pipeline.
3. Click **Ingest**, and then click **Next**.
4. On the New Ingest Job box, enter the name of the job, and then select the data source file.
5. Click **Advanced** and adjust any of the Spark properties, if necessary:
 - Driver Memory (GB)
 - Driver Cores
 - Executor Memory (GB)
 - Executor Cores
6. Click **Save** to continue.
7. Click one of the following commands:
 1. If you need to edit any of the job settings, click **Edit**.
 2. If you want to remove the job from your Jobs page, click **Delete job**.
8. Click **Run Now** to start the ingestion run.
9. Next monitor the state of the job by clicking **Logs**, or click **Refresh** to update the Jobs page.
10. When the job completes, the change in the status appears.



Run training

After you have ingested the data into PingOne Autonomous Identity, start the training run.

Training involves two steps:

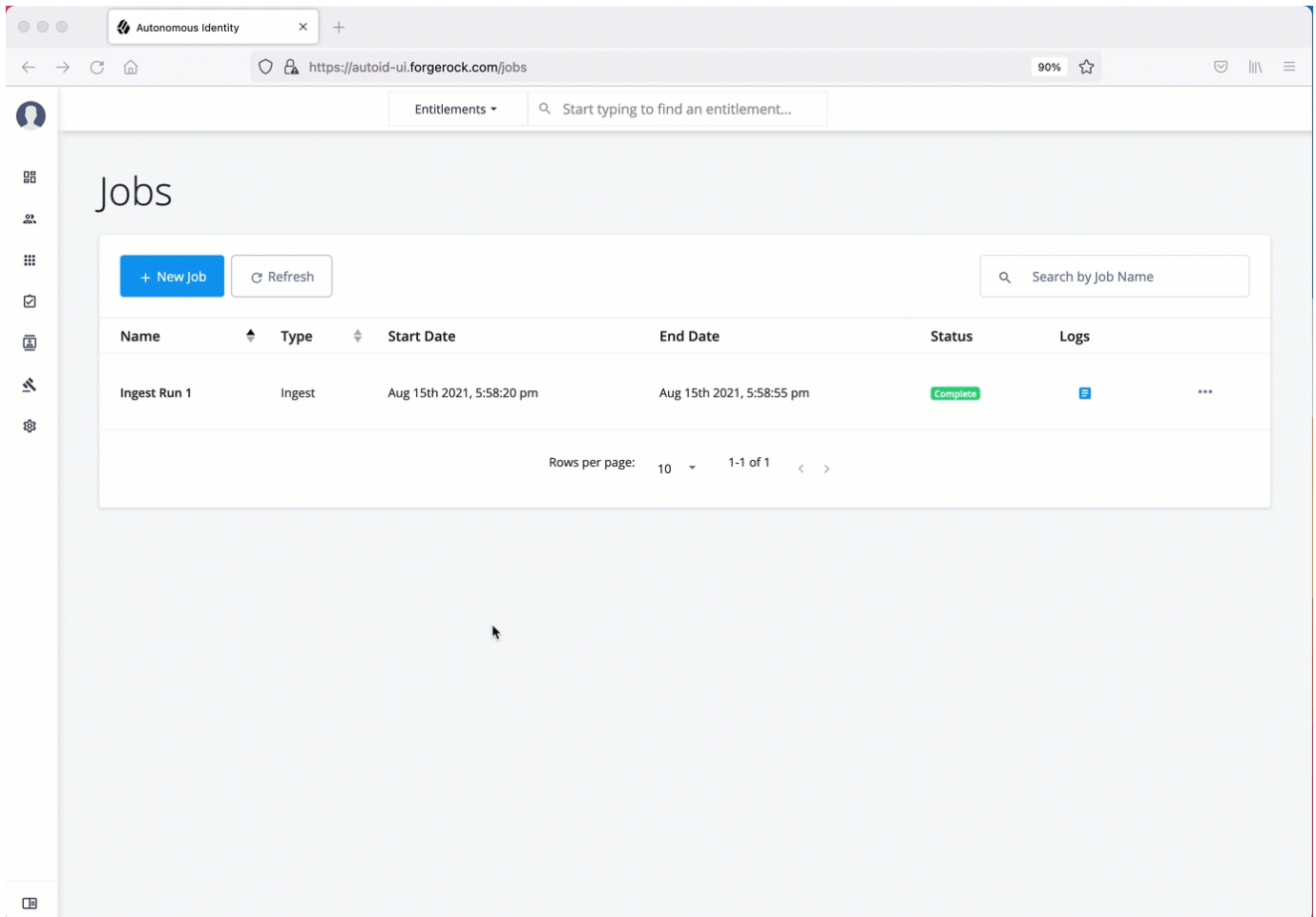
- PingOne Autonomous Identity starts an initial machine learning run where it analyzes the data and produces association rules, which are relationships discovered within your large set of data. In a typical deployment, you can have several million generated rules. The training process can take time depending on the size of your data set.
- Each of these rules are mapped from the user attributes to the entitlements and assigned a confidence score.

The initial training run may take time as it goes through the analysis process. Once it completes, it saves the results directly to the database.

Run training using the UI:

1. On the PingOne Autonomous Identity UI, click the Administration link, and then click **Jobs**.
2. On the Jobs page, click **New Job**. PingOne Autonomous Identity displays a job schedule with each job in the analytics pipeline.
3. Click **Training**, and then click **Next**.
4. On the New Training Job box, enter the name of the job.

5. Click **Advanced** and adjust any of the Spark properties, if necessary.
6. Click **Save** to continue.
7. Click **Run Now**.
8. Next monitor the state of the job by clicking **Logs**, or click **Refresh** to update the Jobs page.
9. When the job completes, the change in the status is displayed.



Run role mining

After you have run training, you can now run the role mining job.

Note

If you want to update your role mining data after an initial analytics job, you can minimally run the ingest, train, and mine analytics jobs. However, we recommend re-running the full analytics pipeline, so that other pages can pick up changes to your access landscape.

Run role mining using the UI:

1. On the PingOne Autonomous Identity UI, click the Administration link, and then click **Jobs**.

2. On the Jobs page, click **New Job**. PingOne Autonomous Identity displays a job schedule with each job in the analytics pipeline.
3. Click **Role Mining**, and then click **Next**.
4. On the New Role Mining Job box, enter the name of the job.
5. Click **Advanced** and adjust any of the Spark properties, if necessary.
6. Click **Save** to continue.
7. Click **Run Now**.
8. Next monitor the state of the job by clicking **Logs**, or click **Refresh** to update the Jobs page.
9. When the job completes, the change in the status appears.

The screenshot displays the 'Jobs' page in the PingOne Autonomous Identity interface. The page features a sidebar with navigation icons and a main content area. At the top of the main area, there is a search bar for entitlements and a search bar for job names. Below these, the 'Jobs' section contains a '+ New Job' button and a 'Refresh' button. A table lists the jobs, with columns for Name, Type, Start Date, End Date, Status, and Logs. The table shows two jobs: 'Ingest Run 1' and 'Training Run 1', both of which are 'Complete'. The page also includes a pagination control at the bottom of the table.

Name	Type	Start Date	End Date	Status	Logs
Ingest Run 1	Ingest	Aug 15th 2021, 5:58:20 pm	Aug 15th 2021, 5:58:55 pm	Complete	...
Training Run 1	Training	Aug 15th 2021, 6:03:31 pm	Aug 15th 2021, 6:05:01 pm	Complete	...

Run as-is predictions

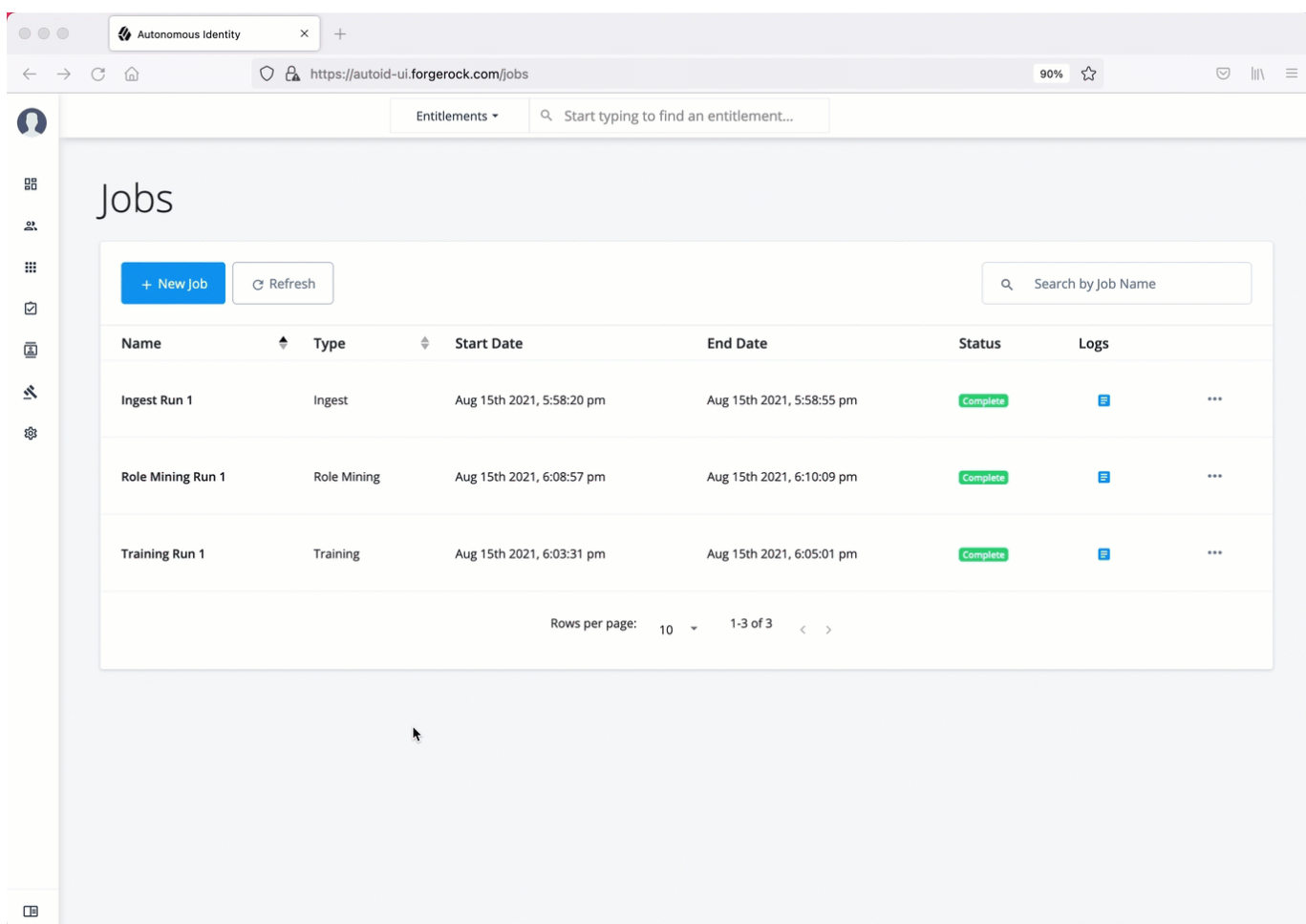
After your initial training run, the association rules are saved to disk. The next phase is to use these rules as a basis for the predictions module.

The predictions module is comprised of two different processes:

- **as-is.** During the As-Is Prediction process, confidence scores are assigned to the entitlements that users currently have. The as-is process maps the highest confidence score to the highest **freqUnion** rule for each user-entitlement access. These rules will then be displayed in the UI and saved directly to the database.
- **Recommendations.** Refer to [Run recommendations](#).

Run predict as-is using the UI:

1. On the PingOne Autonomous Identity UI, click the Administration link, and then click **Jobs**.
2. On the Jobs page, click **New Job**. PingOne Autonomous Identity displays a job schedule with each job in the analytics pipeline.
3. Click **Predict-As-Is**, and then click **Next**.
4. On the New Predict-As-Is Job box, enter the name of the job.
5. Click **Advanced** and adjust any of the Spark properties, if necessary.
6. Click **Save** to continue.
7. Click **Run Now**.
8. Next monitor the state of the job by clicking Logs, or click Refresh to update the Jobs page.
9. When the job completes, the change in the status is displayed.



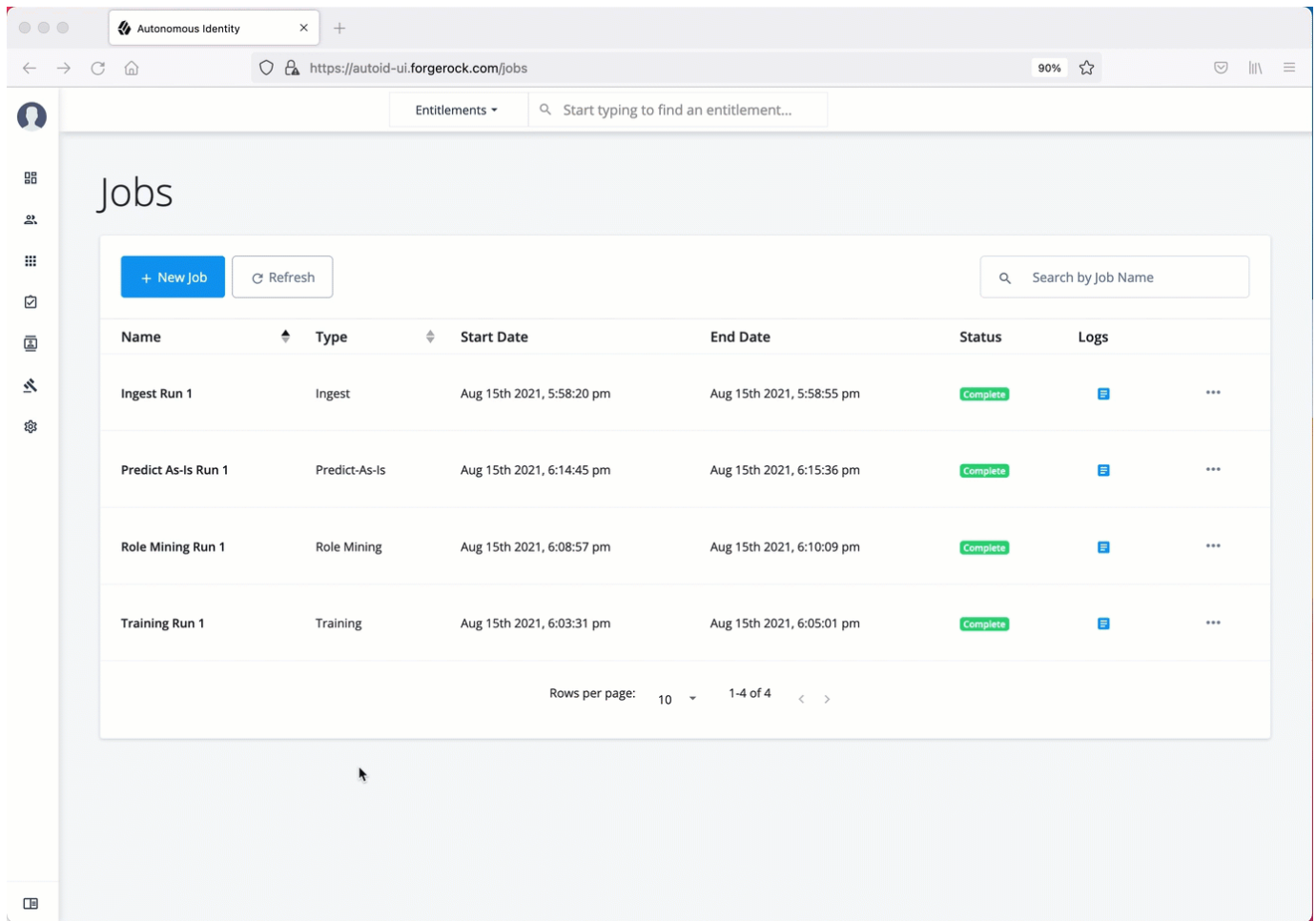
Run recommendations

During the second phase of the predictions process, the recommendations process analyzes each employee who may not have a particular entitlement and predicts the access rights that they should have according to their high confidence score justifications. These rules will then be displayed in the UI and saved directly to the database.

Run predict-recommendation using the UI:

1. On the PingOne Autonomous Identity UI, click the Administration link, and then click **Jobs**.
2. On the Jobs page, click **New Job**. PingOne Autonomous Identity displays a job schedule with each job in the analytics pipeline.
3. Click **Predict-Recommendation**, and then click **Next**.
4. On the New Predict-Recommendation Job box, enter the name of the job.
5. Click **Advanced** and adjust any of the Spark properties, if necessary.
6. Click **Save** to continue.
7. Click **Run Now**.
8. Next monitor the state of the job by clicking **Logs**, or click **Refresh** to update the Jobs page.

9. When the job completes, the change in the status appears.



Publish the analytics data

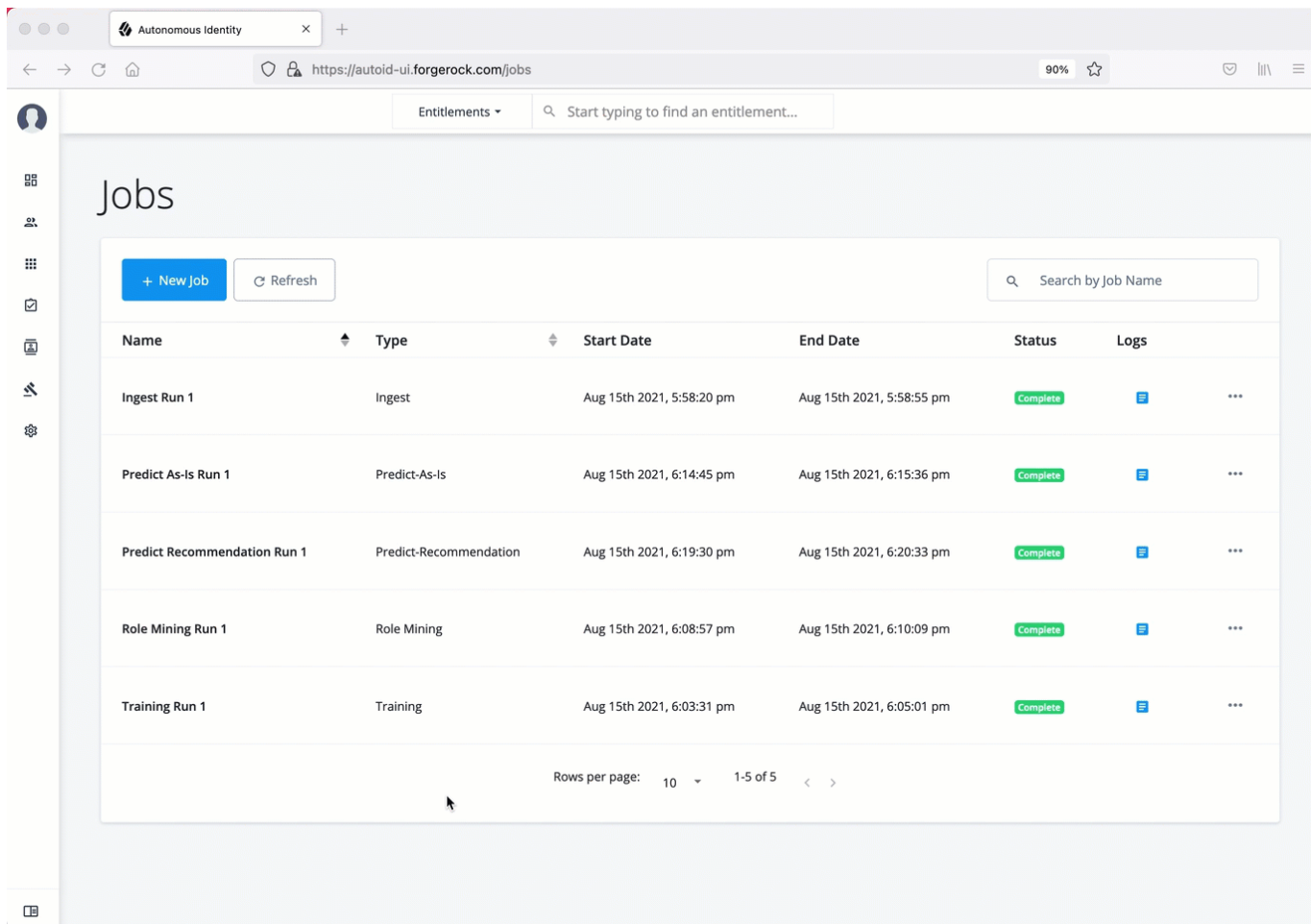
Populate the output of the training, predictions, and recommendation runs to a large table with all assignments and justifications for each assignment. The table data is then pushed to the Cassandra or MongoDB backend.

Run publish using the UI:

1. On the PingOne Autonomous Identity UI, click the Administration link, and then click **Jobs**.
2. On the Jobs page, click **New Job**. PingOne Autonomous Identity displays a job schedule with each job in the analytics pipeline.
3. Click **Publish**, and then click **Next**.
4. On the New Publish Job box, enter the name of the job.
5. Click **Advanced** and adjust any of the Spark properties, if necessary.
6. Click **Save** to continue.
7. Click one of the following commands:
8. Click **Run Now**.

9. Next monitor the state of the job by clicking **Logs**, or click **Refresh** to update the Jobs page.

10. When the job completes, the change in the status appears.



Create assignment index

Next, run the `create-assignment-index` job. This command creates a master index by joining together all database tables. The combined index becomes a source index for the APIs.

Run create-assignment-index using the UI:

1. On the PingOne Autonomous Identity UI, click the Administration link, and then click **Jobs**.
2. On the Jobs page, click **New Job**. PingOne Autonomous Identity displays a job schedule with each job in the analytics pipeline.
3. Click **Create Assignment Index**, and then click **Next**.
4. On the New Create Assignment Index Job box, enter the name of the job.
5. Click **Advanced** and adjust any of the Spark properties, if necessary.
6. Click **Save** to continue.
7. Click **Run Now**.

8. Next monitor the state of the job by clicking Logs, or click **Refresh** to update the Jobs page.

9. When the job completes, the change in the status appears.

The screenshot displays the 'Jobs' page in the PingOne Autonomous Identity interface. The page features a sidebar with navigation icons, a top bar with 'Entitlements' and a search bar, and a main content area with a 'Jobs' table. The table has columns for Name, Type, Start Date, End Date, Status, and Logs. All jobs listed are 'Complete'. The jobs are: Ingest Run 1, Predict As-Is Run 1, Predict Recommendation Run 1, Publish Run 1, Role Mining Run 1, and Training Run 1. The page includes a sidebar with navigation icons, a top bar with 'Entitlements' and a search bar, and a bottom bar with pagination controls showing '1-6 of 6' rows.

Name	Type	Start Date	End Date	Status	Logs
Ingest Run 1	Ingest	Aug 15th 2021, 5:58:20 pm	Aug 15th 2021, 5:58:55 pm	Complete	...
Predict As-Is Run 1	Predict-As-Is	Aug 15th 2021, 6:14:45 pm	Aug 15th 2021, 6:15:36 pm	Complete	...
Predict Recommendation Run 1	Predict-Recommendation	Aug 15th 2021, 6:19:30 pm	Aug 15th 2021, 6:20:33 pm	Complete	...
Publish Run 1	Publish	Aug 15th 2021, 6:24:56 pm	Aug 15th 2021, 6:25:37 pm	Complete	...
Role Mining Run 1	Role Mining	Aug 15th 2021, 6:08:57 pm	Aug 15th 2021, 6:10:09 pm	Complete	...
Training Run 1	Training	Aug 15th 2021, 6:03:31 pm	Aug 15th 2021, 6:05:01 pm	Complete	...

Note

The create-assignment-index-report is an export of the assignment index to a csv file. This allows users to create custom reports from the master table.

Run anomaly report

PingOne Autonomous Identity provides a report on any anomalous entitlement *assignments* that have a low confidence score but are for entitlements that have a high average confidence score. The report's purpose is to identify true anomalies rather than poorly managed entitlements.

The report generates the following points:

- Identifies potential anomalous assignments.
- Identifies the number of users who fall below a low confidence score threshold. For example, if 100 people all have low confidence score assignments to the same entitlement, then it is likely not an anomaly. The entitlement is either missing data or the assignment is poorly managed.

Run the anomaly report using the UI:

1. On the PingOne Autonomous Identity UI, click the Administration link, and then click **Jobs**.
2. On the Jobs page, click **New Job**. PingOne Autonomous Identity displays a job schedule with each job in the analytics pipeline.
3. Click **Anomaly**, and then click **Next**.
4. On the New Anomaly Job box, enter the name of the job.
5. Click **Advanced** and adjust any of the Spark properties, if necessary.
6. Click **Save** to continue.
7. Click **Run Now** to start the ingestion run.
8. Next monitor the state of the job by clicking **Logs**, or click **Refresh** to update the Jobs page.
9. When the job completes, the change in the status appears.
10. Access the anomaly report. The report is available at `/data/output/reports/anomaly_report/<report-id>.csv`.

Run insight report

Next, run an insight report on the generated rules and predictions that were generated during the training and predictions runs. The analytics command generates `insight_report.txt` and `insight_report.xlsx` and writes them to the `/data/input/spark_runs/reports` directory.

The report provides the following insights:

- Total number of assignments received, scored, and unscored.
- Total number of valid assignments received.
- Total number of invalid assignments received.
- Total number of assignments received, scored, and unscored.
- Number of entitlements received, scored, and unscored.
- Number of assignments scored greater than 80% and less than 5%.
- Distribution of assignment confidence scores.
- List of the high volume, high average confidence entitlements.
- List of the high volume, low average confidence entitlements.
- Top 25 users with more than 10 entitlements.
- Top 25 users with more than 10 entitlements and confidence scores greater than 80%.
- Top 25 users with more than 10 entitlements and confidence scores less than 5%.
- Breakdown of all applications and confidence scores of their assignments.
- Supervisors with most employees and confidence scores of their assignments.

- Top 50 role owners by number of assignments.
- List of the "Golden Rules," high confidence justifications that apply to a large volume of people.

Run the insight report using the UI:

1. On the PingOne Autonomous Identity UI, click the Administration link, and then click **Jobs**.
2. On the Jobs page, click **New Job**. PingOne Autonomous Identity displays a job schedule with each job in the analytics pipeline.
3. Click **Insight**, and then click **Next**.
4. On the New Insight Job box, enter the name of the job.
5. Click **Advanced** and adjust any of the Spark properties, if necessary.
6. Click **Save** to continue.
7. Click **Run Now**.
8. Next monitor the state of the job by clicking Logs, or click Refresh to update the Jobs page.
9. When the job completes, the change in the status appears.
10. Access the insight report. The report is available at `/data/output/reports/insight_report.xlsx`.

Run analytics on the command Line

PingOne Autonomous Identity supports the ability to run the pipeline from the command-line. Before you run the pipeline commands, you must run the pre-analytic tasks as defined in [Pre-Analytics Tasks](#), and then define the jobs on the Jobs UI.



Important

The analytics pipeline CLI commands will be deprecated in a future release. We recommend using the Jobs UI to run the analytics jobs.

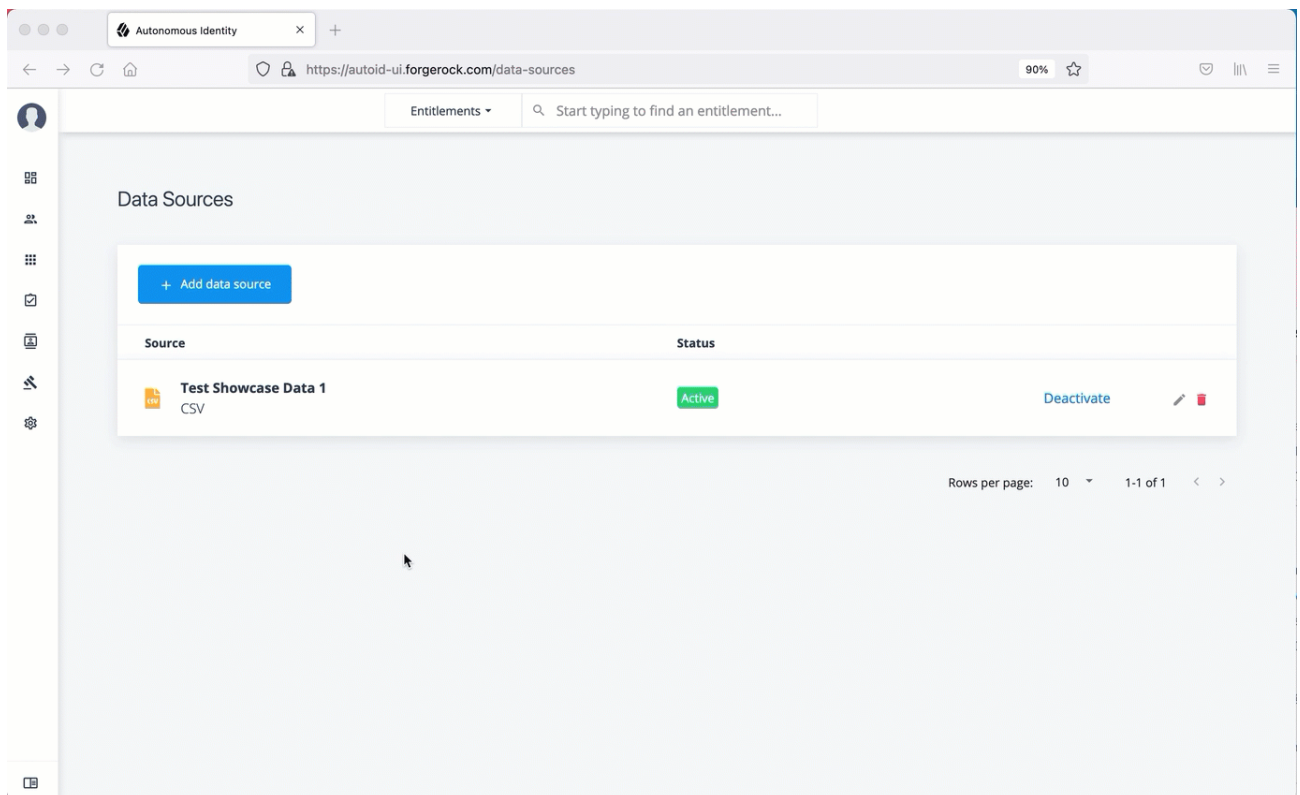
1. Make sure to run the pre-analytics tasks, such as adding attributes to the schema, define your datasources, set up your attribute mappings, and adjusting your analytics threshold values, if necessary:
 - Add attributes to the schema. For more information, refer to [Set Entity Definitions](#).
 - Define your datasources. PingOne Autonomous Identity supports different file types for ingestion: CSV, JDBC, and generic. You can enter more than one data source file, specifying the dataset location on your target machine. For more information, refer to [Set Data Sources](#).
 - Define attribute mappings between your data and the schema. For more information, refer to [Set Attribute Mappings](#).
 - Configure your analytics threshold values. For more information, refer to [Set Analytics Thresholds](#).
2. Define your job definitions on the UI for each of the following:



Important

You can only define your job definitions on the PingOne Autonomous Identity UI. There is no command-line equivalent to define the jobs.

- Ingest
- Train
- Role Mine
- Predict As-Is
- Predict Recommendation
- Publish
- Create Assignment Index
- Anomaly Report (Optional)
- Insight Report (Optional)
- Audit Report (Optional)



3. In a terminal window, SSH to the target server.

4. Change to the `analytics` directory.

```
$ cd /opt/autoid/apache-livy/analytics
```

5. Run each of the following jobs to completion, and then submit the next job.

1. Run the ingest job.

```
$ analytics run <ingest-job-definition-name>
```

For example:

```
$ analytics run ingestShowcaseData
```

2. When the ingest job completes, you can run a status command to confirm its completion:

```
$ analytics status ingestShowcaseData
```

```
2021-09-20 23:18:55 INFO  AnalyticsJobsClient:104 - → checking analytic job status for --→
ingestShowcaseData
2021-09-20 23:18:55 INFO  ServiceConfigParser:54 - Building JAS config
2021-09-20 23:18:55 INFO  JASHelper:49 - → Building new SSL context for JAS REST Client using
trust store
2021-09-20 23:18:55 INFO  SSLUtils:36 - ---→ KeyStore path :
2021-09-20 23:18:55 INFO  SSLUtils:44 - ---→ Truststore path : /opt/autoid/certs/jas/jas-
server-truststore.jks
2021-09-20 23:18:55 INFO  ServiceConfigParser:54 - Building JAS config
Job Status result
*****
Job Status for ingestShowcaseData --→ COMPLETED
*****
```

3. Run the training job.

```
$ analytics run <training-job-definition-name>
```

For example:

```
$ analytics run trainShowcaseData
```

4. Run the role mining job.

```
$ analytics run <role-mining-job-definition-name>
```

For example:

```
$ analytics run roleMining
```

5. Run the predict-as-is job.

```
$ analytics run <predict-asis-job-definition-name>
```

For example:

```
$ analytics run predictAsIs
```

6. Run the predict-recommendation job.

```
$ analytics run <predict-recommendation-job-definition-name>
```

For example:

```
$ analytics run predictRecommendation
```

7. Run the publish job.

```
$ analytics run <publish-job-definition-name>
```

For example:

```
$ analytics run publishShowcaseData
```

8. Run the create assignment index job.

```
$ analytics run <create-assignment-index-definition-name>
```

For example:

```
$ analytics run createAssignmentIndex
```

9. Optional. Run the anomaly report job.

```
$ analytics run <anomaly-report-definition-name>
```

For example:

```
$ analytics run anomalyReport
```

10. Optional. Run the insight report job.

```
$ analytics run <insight-report-definition-name>
```

For example:

```
$ analytics run insightReport
```

11. Optional. Run the audit report job.

```
$ analytics run <audit-report-definition-name>
```

For example:

```
$ analytics run auditReport
```

6. Click the PingOne Autonomous Identity UI Dashboard. The page reloads with your data.

Admin user tasks

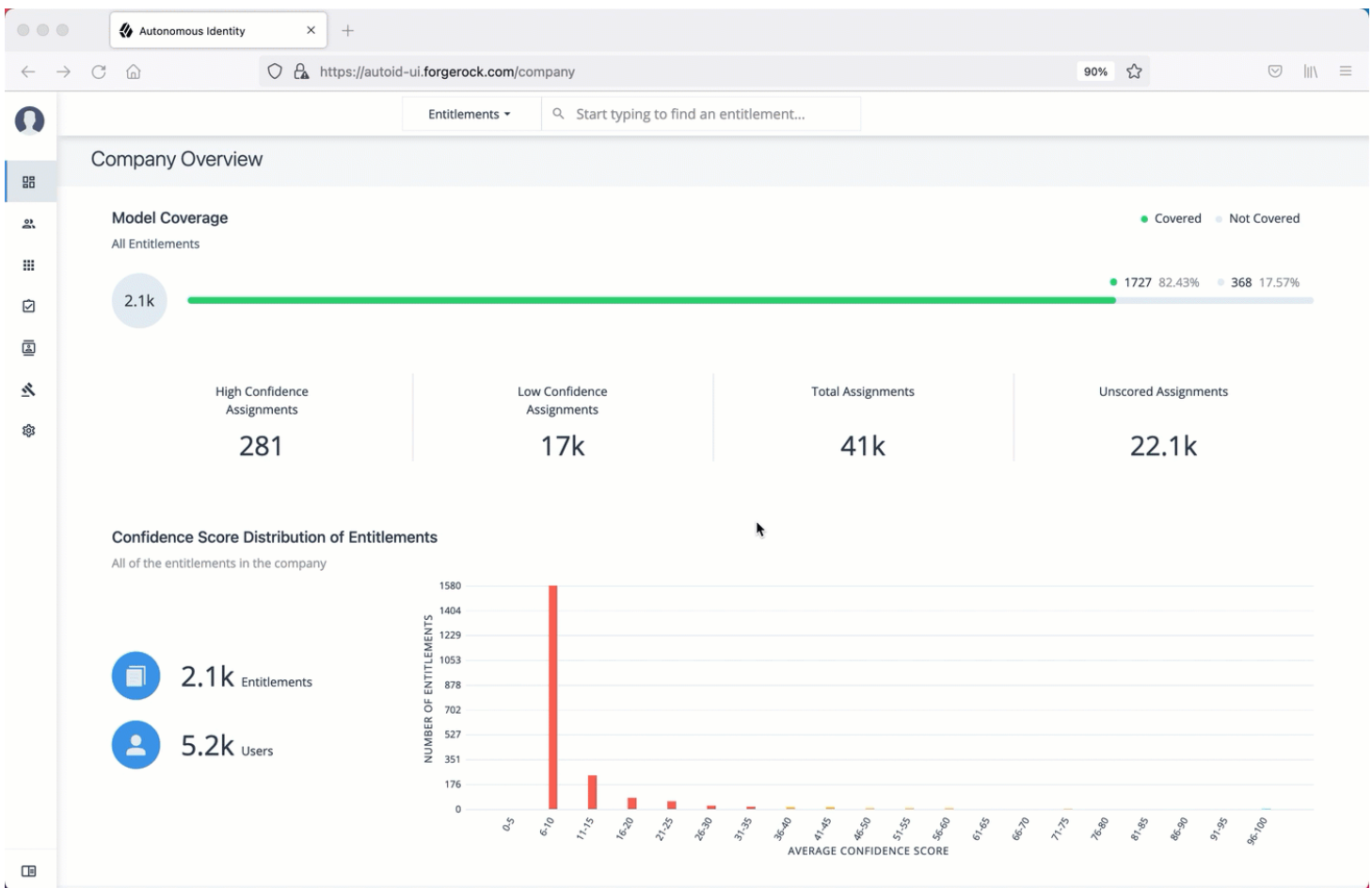
The Admin user functionality is similar to that of a system administration *superuser*. Admin users have the access rights to company-wide entitlement data on the PingOne Autonomous Identity console. Admin users can approve or revoke a user's entitlement.

Investigate Most Critical Entitlements

One important task that an administrator must perform is to examine all critical entitlements. Critical entitlements are assigned entitlements that have are highly-assigned but have a low confidence score associated with it. The PingOne Autonomous Identity console provides a means to examine these entitlements.

Follow these steps to evaluate the most critical entitlements list:

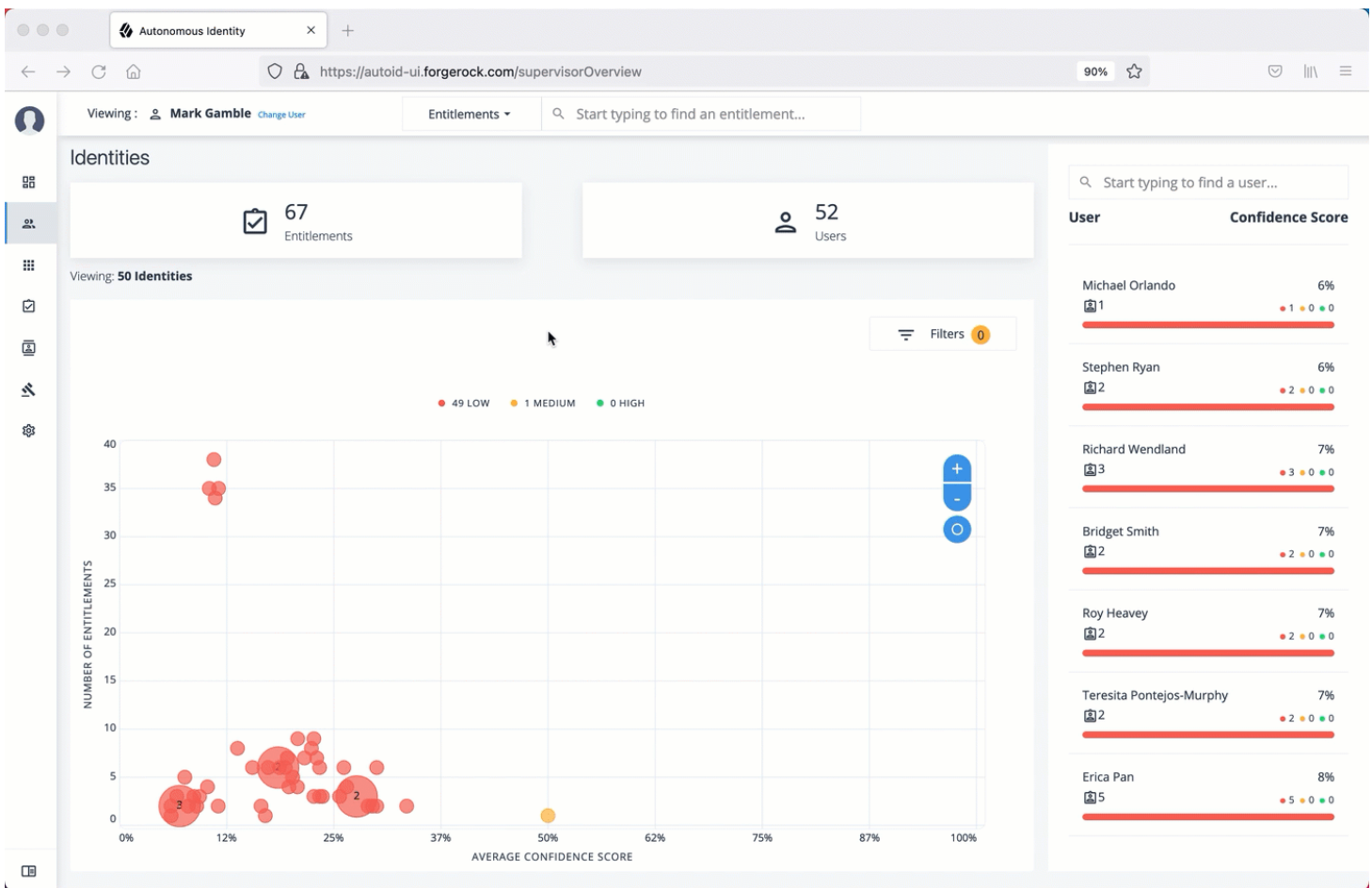
1. On the Dashboard, scroll down to the Most Critical Entitlements section. This section displays the entitlements that have low confidence scores and a high number of employees who have this entitlement.
2. Click an entitlement to view its details.
3. On the Entitlements detail page, review the key metrics.
4. Click the right arrow in one of the category ranges to view the users, and then click one of the users in the list.
5. On the User's Entitlements page, scroll down to review the Confidence Score Comparison table to display the differences between the user's attribute and the driving factor attributes.
6. Click Employees associated with this entitlement to review other uses who have this entitlement.
7. Click **Actions**, and then click **Approve** or **Revoke** for this entitlement. You can also bulk approve more than one entitlement. You can only revoke one entitlement at a time.



Approve or revoke access an entitlement for a user

Follow these steps to investigate a confidence score and approve or revoke access an entitlement assigned to a specific user:

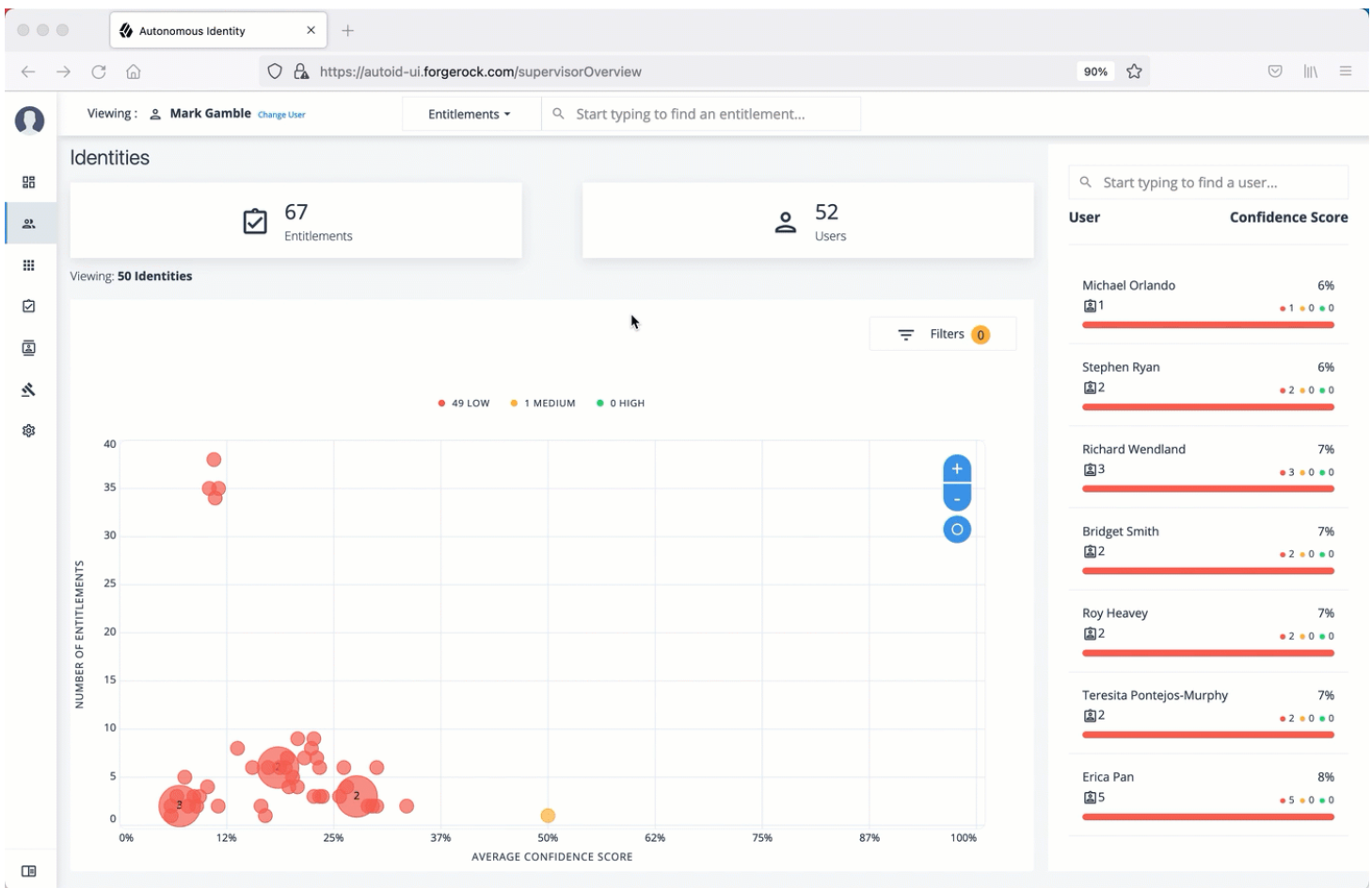
1. On PingOne Autonomous Identity console, click **Identities**, and enter a name of a supervisor. The only way to access a user's entitlements is through the Most Critical Entitlements section or the Identities page.
2. On the Identities page, click a circle, and then click the user in the list on the right.
3. On the User Entitlement page, click a confidence circle on the graph to highlight the entitlement below.
4. For the selected entitlement, click the down arrow on the right to view the Driving Factor Comparison.
5. Click Employees associated with this entitlement to view the justifications for those users with high confidence scores with this entitlement.
6. Click **Actions**, and then click **Approve Access** or **Revoke access**. If you have more than one entitlement that you want to approve, select them all and do a bulk Approval. You can only do one Revoke Access at a time.



Check not-scored users

Follow these steps to check Not Scored entitlements. Not-scored indicates that it does not have a justification associated with the entitlement:

1. On PingOne Autonomous Identity console, click **Identities**, and enter a name of a supervisor. The only way to access a user's entitlements is through the Most Critical Entitlements section or the Identities page.
2. On the Identities page, click a circle, and then click the user in the list on the right.
3. On the User Entitlement page, click **Not Scored**.
4. On the Not Scored Entitlements page, click the down arrow to view the driving factors comparison.
5. Click Employees associated with this entitlement to view the justifications for those users with high confidence scores with this entitlement.
6. Click **Actions**, and then click **Approve Access** or **Revoke access**. At a later date, you can re-click the **Approve** or **Revoke** button to cancel the operation.



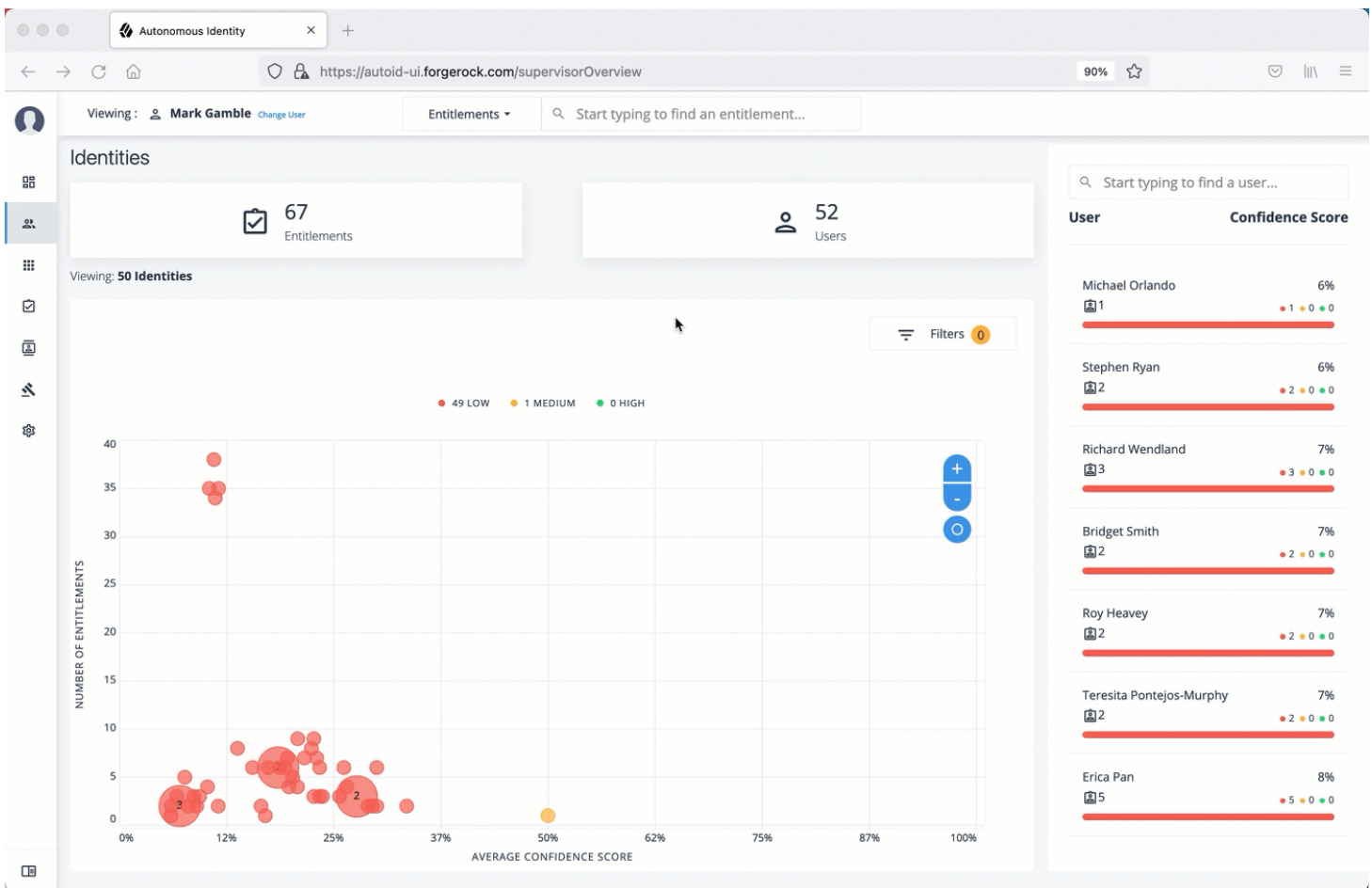
Apply filters

Follow these steps to apply filters to your confidence score graphs on the Identities and Entitlements pages:

Note

The Filters for the Identities and Entitlements are similar. The filters for the Applications and Rules pages offer different options to filter your searches.

1. On the Identities or Entitlements page, view the average confidence score graph.
2. On the right, click **Filters**.
3. Under filters, do one or all of the following:
 - Click **Remove High Scores from Average** or enable any filter in the Application Filters section.
 - Under Applications, click one or more applications to display the identities or entitlements associated with the selected application.
 - Click **Add Filters** to further display only those identities or entitlements based on a user attribute, such as `city`. When ready, click **Apply Filters**.
4. Click **Clear Filters** to remove your filters.



Changing the API's elasticsearch client request timeout

The following steps outline how to change the PingOne Autonomous Identity API's Elasticsearch client timeout to override the default of 30 seconds.

1. Open the `/opt/autoid/res/api/docker-compose.yml` file, and edit the `ELASTICSEARCH_CLIENT_TIMEOUT` variable as necessary (time in milliseconds):

```
environment:
  ...
  - ELASTICSEARCH_CLIENT_TIMEOUT=30000
```

For example:

```
environment:
  ...
  - ELASTICSEARCH_CLIENT_TIMEOUT=60000
```

2. Remove the currently running `zoran-api` container, and redeploy the `zoran-api` Docker image:

```
docker stack rm api
docker stack deploy --with-registry-auth --compose-file /opt/autoid/res/api/docker-compose.yml api
```

3. Restart the `zoran-api` and `nginx` containers:

```
docker service update --force ui_zoran-ui && docker service update --force nginx_nginx
```

4. Verify that PingOne Autonomous Identity is running by opening the UI in a web browser.

Server maintenance

PingOne Autonomous Identity administrators must conduct various tasks to maintain the service for their users.

The following are basic server maintenance tasks that may occur:

Stopping and starting

You can run the following command to stop or start PingOne Autonomous Identity components:

Docker

Stop Docker

- Stop docker. This will shutdown all of the containers.

```
$ sudo systemctl stop docker
```

Restart Docker

1. To restart docker, first set the docker to start on boot using the `enable` command:

```
$ sudo systemctl enable docker
```

2. To start docker, run the `start` command:

```
$ sudo systemctl start docker
```

3. After restarting Docker, restart the JAS service to ensure the service can write to its logs:

```
$ docker service update --force jas_jasnode
```

Cassandra

Stop Cassandra

1. On the deployer node, SSH to the target node.
2. Check Cassandra status.

```
Datacenter: datacenter1
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving - Address      Load          Tokens        Owns (effective)
Host ID                               Rack
UN 10.128.0.38  1.17 MiB    256           100.0%         d134e7f6-408e-43e5-bf8a-7adff055637a
rack1
```

3. To stop Cassandra, find the process ID and run the kill command.

```
$ pgrep -u autoid -f cassandra | xargs kill -9
```

4. Check the status again.

```
nodetool: Failed to connect to '127.0.0.1:7199' - ConnectException: 'Connection refused (Connection refused)'.
```

Restart Cassandra

1. On the deployer node, SSH to the target node.
2. Restart Cassandra. When the `No gossip backlog; proceeding` message is displayed, hit `Enter` to continue.

```
$ cassandra

...
INFO [main] 2020-11-10 17:22:49,306 Gossiper.java:1670 - Waiting for gossip to settle...
INFO [main] 2020-11-10 17:22:57,307 Gossiper.java:1701 - No gossip backlog; proceeding
```

3. Check the status of Cassandra. Make sure that it is in `UN` status ("Up" and "Normal").

```
$ nodetool status
```

MongoDB

Shut Down MongoDB

1. Check the status of the MongoDB

```
$ ps -ef | grep mongod
```

2. Connect to the Mongo shell.

```
$ mongo --tls --tlsCAFile /opt/autoid/mongo/certs/rootCA.pem --tlsCertificateKeyFile /opt/autoid/mongo/certs/mongod.pem  
--tlsAllowInvalidHostnames --host <ip-address>
```

```
MongoDB shell version v4.2.9  
connecting to: mongod://<ip-address>:27017/?compressors=disabled&gssapiServiceName=mongod  
2020-10-08T18:46:23.285+0000 W NETWORK [js] The server certificate does not match the hostname.  
Hostname: <ip-address> does not match CN: mongonode  
Implicit session: session { "id" : UUID("22c0123-30e3-4dc9-9d16-5ec310e1ew7b") }  
MongoDB server version: 4.2.9
```

3. Switch the admin table.

```
> use admin  
  
switched to db admin
```

4. Authenticate using the password set in `vault.yml` file.

```
> db.auth("root", "Welcome123")  
  
1
```

5. Start the shutdown process.

```
> db.shutdownServer()  
  
2020-10-08T18:47:06.396+0000 I NETWORK [js] DBClientConnection failed to receive message from <ip-  
address>:27017 - SocketException: short read  
server should be down...  
2020-10-08T18:47:06.399+0000 I NETWORK [js] trying reconnect to <ip-address>:27017 failed  
2020-10-08T18:47:06.399+0000 I NETWORK [js] reconnect <ip-address>:27017 failed
```

6. Exit the mongo shell.

```
$ quit()  
or <Ctrl-C>
```

7. Check the status of the MongoDB

```
$ ps -ef | grep mongod

no instance of mongod found
```

Restart MongoDB

1. Re-start the MongoDB service.

```
$ /usr/bin/mongod --config /opt/autoid/mongo/mongo.conf

about to fork child process, waiting until server is ready for connections.
forked process: 31227
child process started successfully, parent exiting
```

2. Check the status of the MongoDB

```
$ ps -ef | grep mongod

autoid      9245      1   0 18:48 ?          00:00:45 /usr/bin/mongod --config /opt/autoid/mongo/
mongo.conf
autoid      22003    6037   0 21:12 pts/1    00:00:00 grep --color=auto mongod
```

== Apache Spark

1. On the deployer node, SSH to the target node.
2. Check Spark status. Make sure that it is up-and-running.

```
$ elinks http://localhost:8080
```

3. Stop the Spark Master and workers.

```
$ /opt/autoid/spark/spark-2.4.4-bin-hadoop2.7/sbin/stop-all.sh

localhost: stopping org.apache.spark.deploy.worker.Worker
stopping org.apache.spark.deploy.master.Master
```

4. Check the Spark status again. You should see: `Unable to retrieve http://localhost:8080: Connection refused.`

Restart Spark

1. On the deployer node, SSH to the target node.
2. Start the Spark Master and workers. Enter the user password on the target node when prompted.

```
$ /opt/autoid/spark/spark-2.4.4-bin-hadoop2.7/sbin/start-all.sh
```

```
starting org.apache.spark.deploy.master.Master, logging to /opt/autoid/spark/spark-2.4.4-bin-hadoop2.7/logs/spark-autoid-org.apache.spark.deploy.master.Master-1.out
autoid-2 password:
localhost: starting org.apache.spark.deploy.worker.Worker, logging to /opt/autoid/spark/spark-2.4.4-bin-hadoop2.7/logs/spark-autoid-org.apache.spark.deploy.worker.Worker-1.out
```

3. Check the Spark status again. Make sure that it is up-and-running.

Apache Livy

Apache Livy lets you manage Spark Clusters context management using a simple REST interface.

Stop and Start Livy

1. Stop Livy.

```
$ /opt/autoid/apache-livy/apache-livy-080-incubating-SNAPSHOT-bin/bin/livy-server stop
```

2. Start Livy.

```
$ /opt/autoid/apache-livy/apache-livy-080-incubating-SNAPSHOT-bin/bin/livy-server start
```

Accessing log files

PingOne Autonomous Identity provides different log files to monitor or troubleshoot your system.

Getting Docker container information

1. On the target node, get system wide information about the Docker deployment. The information shows the number of containers running, paused, and stopped containers as well as other information about the deployment.

```
$ docker info
```

2. If you want to get debug information, use the `-D` option. The option specifies that all docker commands will output additional debug information.

```
$ docker -D info
```

3. Get information on all of your containers on your system.

```
$ docker ps -a
```

4. Get information on the docker images on your system.

```
$ docker images
```

5. Get docker service information on your system.

```
$ docker service ls
```

6. Get docker the logs for a service.

```
$ docker service logs <service-name>
```

For example, to access the nginx service logs:

```
$ docker service logs nginx_nginx
```

Other useful arguments:

- `--details` . Show extra details.
- `--follow, -f` . Follow log output. The command will stream new output from STDOUT and STDERR.
- `--no-trunc` . Do not truncate output.
- `--tail {n|all}` . Show the number of lines from the end of log files, where `n` is the number of lines or `all` for all lines.
- `--timestamps, -t` . Show timestamps.

Getting Cassandra logs

The Apache Cassandra output log is kicked off at startup. PingOne Autonomous Identity pipes the output to a log file in the directory, `/opt/autoid/` .

1. On the target node, get the log file for the Cassandra install.

```
$ cat /opt/autoid/cassandra/installcassandra.log
```

2. Get startup information. Cassandra writes to `cassandra.out` at startup.

```
$ cat /opt/autoid/cassandra.out
```

3. Get the general Cassandra log file.

```
$ cat /opt/autoid/apache-cassandra-3.11.2/logs/system.log
```

By default, the log level is set to **INFO**. You can change the log level by editing the `/opt/autoid/apache-cassandra-3.11.2/conf/logback.xml` file. After any edits, the change will take effect immediately. No restart is necessary. The log levels from most to least verbose are as follows:

- **TRACE**
- **DEBUG**
- **INFO**
- **WARN**
- **ERROR**
- **FATAL**

4. Get the JVM garbage collector logs.

```
$ cat /opt/autoid/apache-cassandra-3.11.2/logs/gc.log.<number>.current
```

For example:

```
$ cat /opt/autoid/apache-cassandra-3.11.2/logs/gc.log.0.current
```

The output is configured in the `/opt/autoid/apache-cassandra-3.11.2/conf/cassandra-env.sh` file. Add the following JVM properties to enable them:

- `JVM_OPTS="$JVM_OPTS -XX:+PrintGCDetails"`
- `JVM_OPTS="$JVM_OPTS -XX:+PrintGCDateStamps"`
- `JVM_OPTS="$JVM_OPTS -XX:+PrintHeapAtGC"`
- `JVM_OPTS="$JVM_OPTS -XX:+PrintGCApplicationStoppedTime"`

5. Get the debug log.

```
$ cat /opt/autoid/apache-cassandra-3.11.2/logs/debug.log
```

Other useful Cassandra monitoring tools and files

Apache Cassandra has other useful monitoring tools that you can use to observe or diagnose and issue. To access the complete list of options, refer to the Apache Cassandra documentation.

1. View statistics for a cluster, such as IP address, load, number of tokens,

```
$ /opt/autoid/apache-cassandra-3.11.2/bin/nodetool status
```

2. View statistics for a node, such as uptime, load, key cache hit, rate, and other information.

```
$ /opt/autoid/apache-cassandra-3.11.2/bin/nodetool info
```

3. View the Cassandra configuration file to determine how properties are pre-set.

```
$ cat /opt/autoid/apache-cassandra-3.11.2/conf/cassandra.yaml
```

Apache Spark logs

Apache Spark provides several ways to monitor the server after an analytics run.

1. To get an overall status of the Spark server, point your browser to <http://<spark-master-ip>:8080>.
2. Print the logging message sent to the output file during an analytics run.

```
$ cat /opt/autoid/spark/spark-2.4.4-bin-hadoop2.7/logs/<file-name>
```

For example:

```
$ cat /opt/autoid/spark/spark-2.4.4-bin-hadoop2.7/logs/spark-  
org.apache.spark.deploy.master.Master-1-autonomous-id-test.out
```

3. Print the data logs that were written during an analytics run.

```
$ cat /data/log/files/<filename>
```

For example:

```
$ cat /data/log/files/f6c0870e-5782-441e-b145-b0e662f05f79.log
```

Updating IDP Certificates

When an IDP provider changes their certificates, you can update these certificates in PingOne Autonomous Identity.

1. SSH to the target machine.
2. View the current certificate settings in the `docker-compose.yml` file, and locate the `NODE_EXTRA_CA_CERTS` property

```
$ vi /opt/autoid/res/api/docker-compose.yml  
  
NODE_EXTRA_CA_CERTS=/opt/app/cert/<customer ID>-sso.pem
```

3. Access your IDP URL, and export the `.cer` file from the browser.
4. Convert the `.cer` file into a `.pem` file using the following command:

```
$ openssl x509 -in certificatename.cer -outform PEM -out certificatename.pem
```

5. Open the `docker-compose.yml` file and update the `NODE_EXTRA_CA_CERTS` property.

```
$ vi /opt/autoid/res/api/docker-compose.yml

NODE_EXTRA_CA_CERTS=/opt/app/cert/<new-cert>.pem
```

6. Restart the PingOne Autonomous Identity services.

```
$ docker stack rm api
$ docker stack deploy --with-registry-auth --compose-file
  /opt/autoid/res/api/docker-compose.yml api
$ docker service update --force ui_zoran-ui
$ docker service update --force nginx_nginx
```

Changing the Cassandra `zoran_dba` and `zoran_user` passwords

During deployment, PingOne Autonomous Identity creates two *user* accounts to interact with the Cassandra database: `zoran_dba` and `zoran_user`. The `zoran_dba` is an administrator or superuser account used by PingOne Autonomous Identity to set up the Cassandra database. The `zoran_user` is a non-admin account used to log in to the Cassandra command-line interface, `cqlsh`.

You can change the passwords after deploying PingOne Autonomous Identity using `cqlsh`.

Change the Cassandra `zoran_dba` and `zoran_user` passwords

1. Access `cqlsh`.

```
$ cqlsh -u zoran_dba -p admin_password
Connected to Zoran Cluster at <server-ip>:9042.
[cqlsh 5.0.1 | Cassandra 3.11.2 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
zoran_dba@cqlsh>
```

2. In `cqlsh`, change the `zoran_dba` password:

```
zoran_dba@cqlsh>ALTER USER zoran_dba WITH PASSWORD 'new_admin_password';

zoran_dba@cqlsh>exit
```

3. Use a text editor and change the environment variables in the `/opt/autoid/res/jas/docker-compose.yml` file:

```
- CASSANDRA_DB_PASSWORD=new_admin_password
```

4. Remove the running container and redeploy it:

```
$ docker stack rm jas
$ **docker stack deploy --with-registry-auth --compose-file /opt/autoid/res/jas/docker-compose.yml
jas
```

5. Update the `zoran_user` password:

```
zoran_dba@cqlsh>ALTER USER zoran_user WITH PASSWORD 'new_user_password';
zoran_dba@cqlsh>exit
```

Change MongoDB password post-deployment

You can update the MongoDB password by running the following steps on a running instance of MongoDB.

Note

Update the various parameters for host IP and current root password as pertains to your environment. Also, the `--tlsAllowInvalidHostnames` parameter is necessary if you are using self-signed certificates.

1. Open the MongoDB shell. Use your host IP and root password:

```
mongo admin --host 10.10.10.10 --tls \
--tlsCertificateKeyFile /opt/autoid/certs/mongo/mongodb.pem \
--tlsCAFile /opt/autoid/certs/mongo/rootCA.pem \
--tlsAllowInvalidHostnames \
--username root \
--password 'current_root_password'
```

2. On the MongoDB shell, run the `changeUserPassword` command:

```
db.changeUserPassword("mongoadmin", "new_password")
```

3. Update the password as an environment variable in the JAS service. Update the following variable in the `/opt/autoid/res/jas/docker-compose.yml` file:

```
- MONGO_ROOT_PASSWORD=new_password
```

4. Delete the currently running JAS container and redeploy:

```
docker stack rm jas

docker stack deploy \
--with-registry-auth \
--compose-file /opt/autoid/res/jas/docker-compose.yml jas
```

5. Check that there are no stack errors in the container logs. The logs should show successful connections to MongoDB:

```
2022-11-21 19:07:40, 257 INFO c.m.d.l.SLF4JLogger
[cluster-ClusterId{value='637bcc764cb8670d06c2feb8',description='null'}-10.10.10.:27017]
Opened connection [connectionId{localValue:2, serverValue:30}] to 10.10.10.10.:27017

2022-11-21 19:07:40, 257 INFO c.m.d.l.SLF4JLogger
[cluster-rtt-ClusterId{value='637bcc764cb8670d06c2feb8',description='null'}-10.10.10.:27017]
Opened connection [connectionId{localValue:1, serverValue:31}] to 10.10.10.10.:27017

2022-11-21 19:07:40, 257 INFO c.m.d.l.SLF4JLogger
[cluster-rtt-ClusterId{value='637bcc764cb8670d06c2feb8',description='null'}-10.10.10.:27017]
Monitor thread successfully connected to server with description
ServerDescription{address=10.10.10.10:27017, type=STANDALONE, State=CONNECTED,
ok=true, minWireVersion=0, maxWireVersion=9, maxDocumentSize=16777216,
logicalSessionTimeoutMinutes=30, roundTripTimeNanos=221098137}

2022-11-21 19:07:45, 383 INFO c.m.d.l.SLF4JLogger [main]
Opened connection [connectionId{localValue:3, serverValue:32}] to 10.10.10.10.:27017
```

Roles management tasks

PingOne Autonomous Identity supports a powerful role analysis and management system that examines all roles and their assigned entitlements within your access landscape. The system uses machine learning rules and analytics thresholds to determine the confidence scores and driving factors for each role.

The central hub of the roles management system is the *Roles Workshop*. The Roles Workshop lets authorized users review, edit, and test new or existing roles before publishing them to production.

In a typical scenario, an administrator runs a role mining job as part of the analytics pipeline. During a role mining analytics run, PingOne Autonomous Identity discovers *candidates* for any new roles and displays them in the Roles Workshop with confidence scores and driving factors. Authorized users can review these roles, make edits to entitlements, and re-run the role mining analytics until the correct mix of entitlements meets your threshold objectives for given rules.

A month or two later, the administrator can re-run the role mining job to pick up changes in the entitlements landscape. PingOne Autonomous Identity re-analyzes each role and recommends updates to existing roles, such as the indication of stale data, or changes in the confidence scores. Based on these recommendations, the authorized user can revoke any active roles, make new configuration changes to a draft, and publish these draft roles to production.

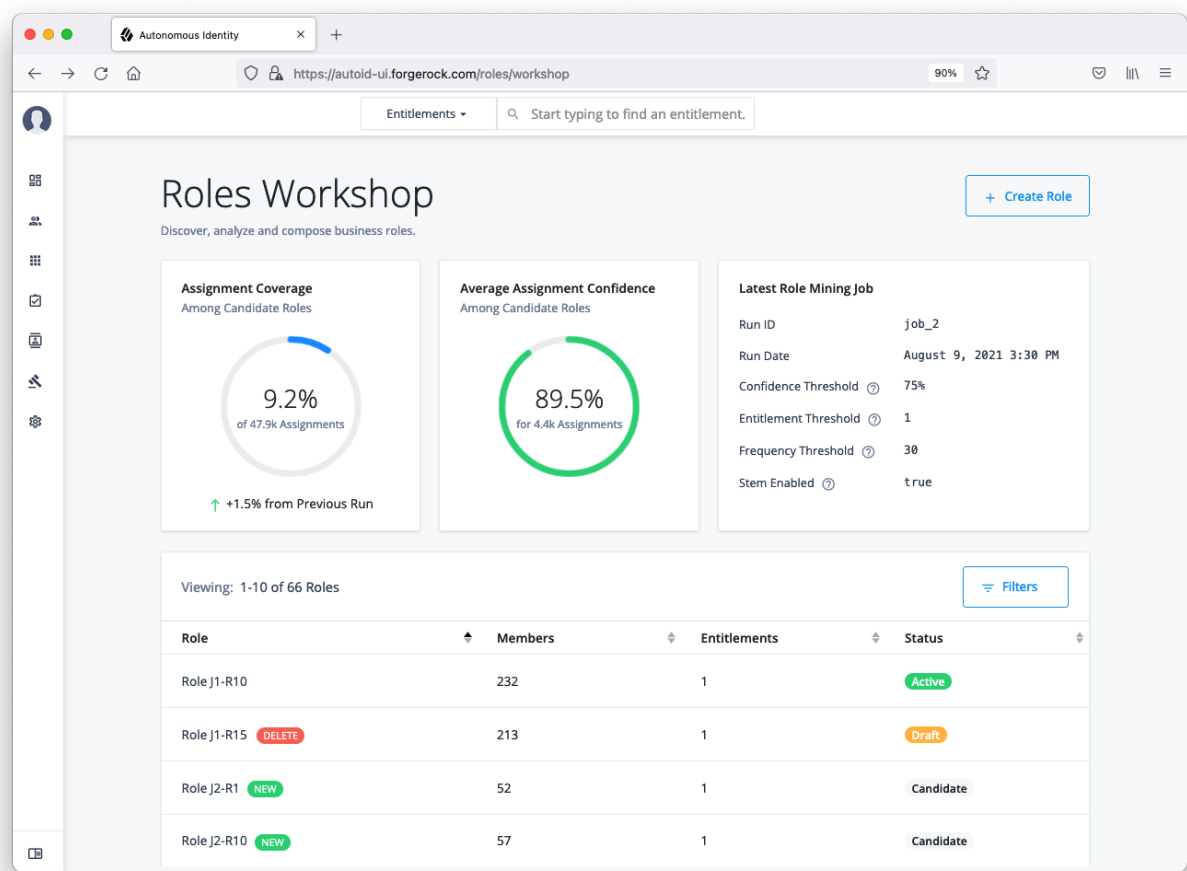


Figure 1. Roles workshop

Roles user types

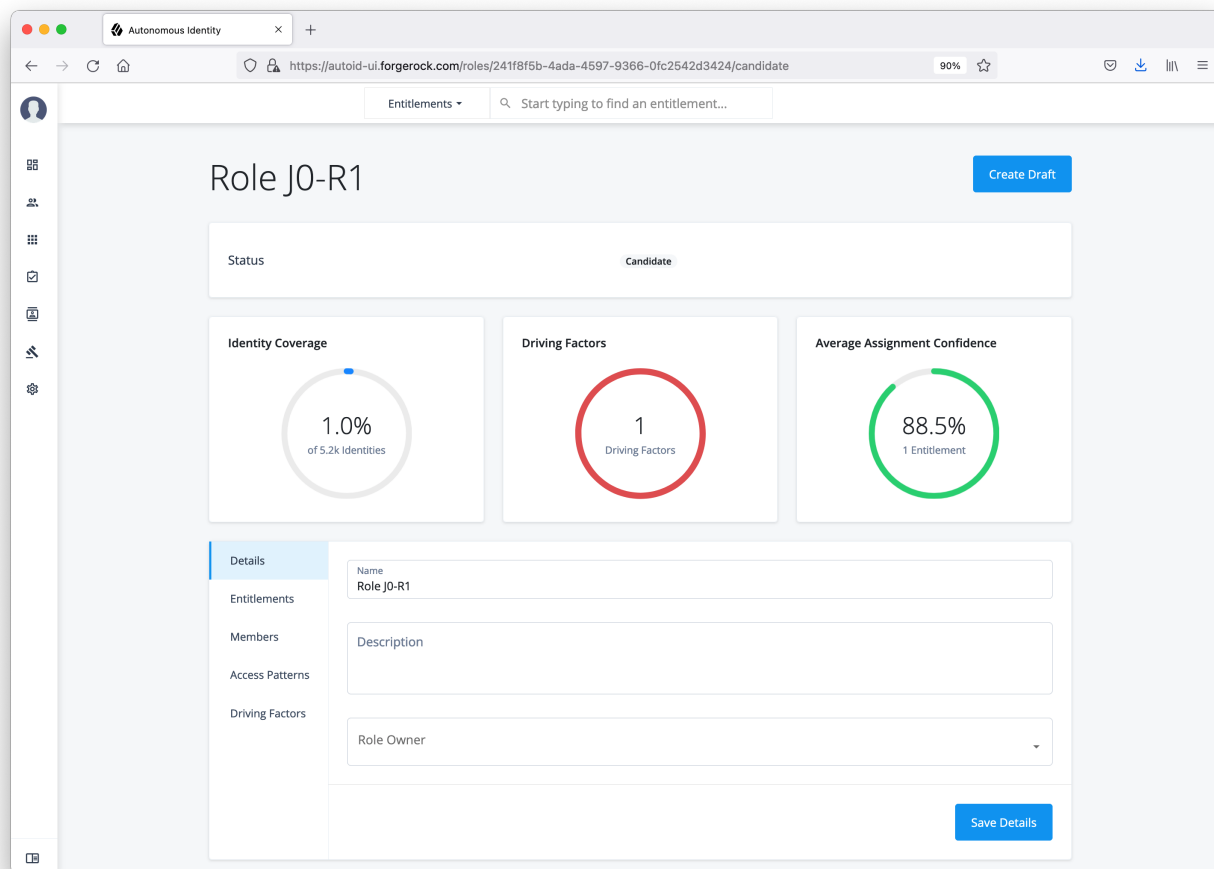
PingOne Autonomous Identity supports two types of user role types to manage roles with the system. You can assign these roles using the Manage Identities function.

User Type	Description
Role Engineer	A user who has the ability to view, edit, delete, and export all roles. Role engineers can create drafts from mined candidates and assign role owners to the draft. They can also create custom roles for further evaluation and testing. PingOne Autonomous Identity administrators automatically are assigned this role.
Role Owner	A user who has the ability to view, edit, delete, and export active and draft roles assigned to them.

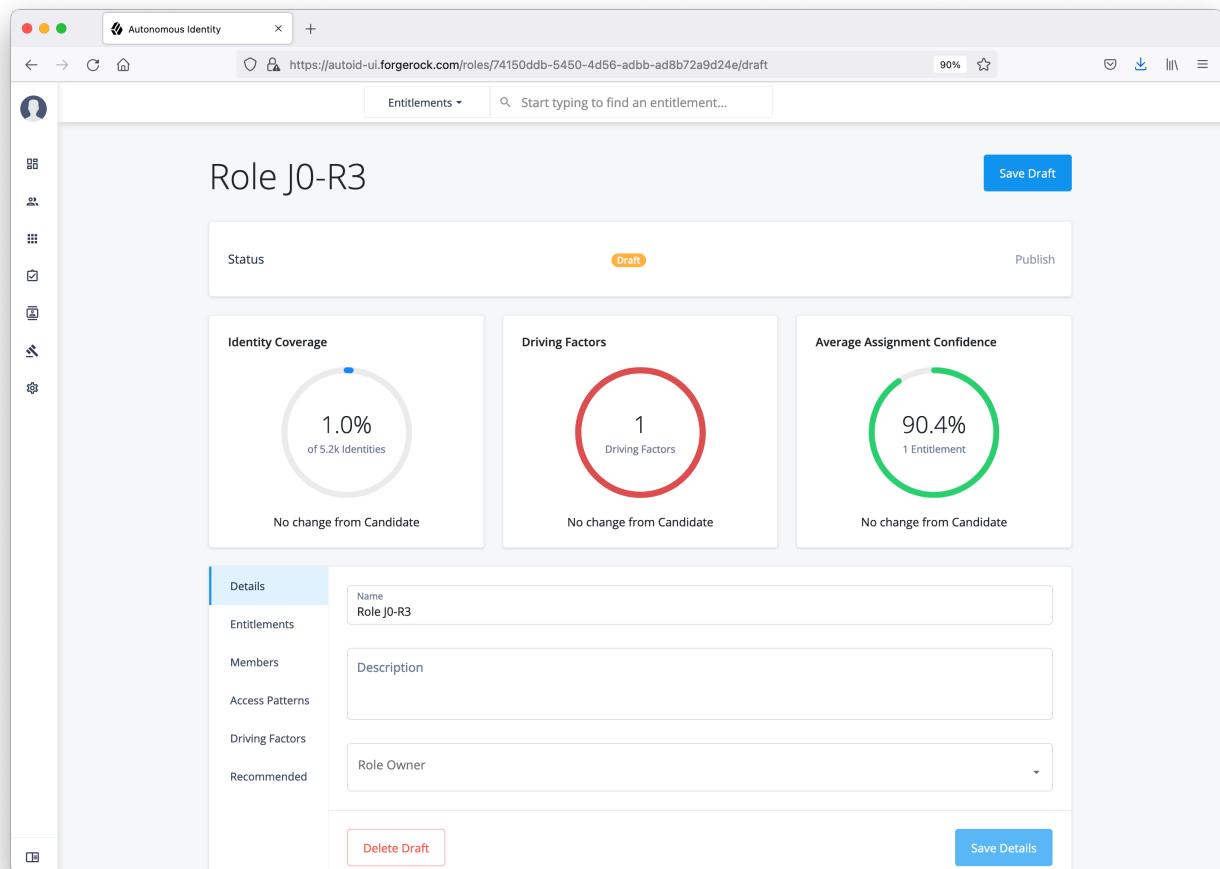
Roles workflow

The Roles Workshop displays roles in three different states: candidate, draft, and active.

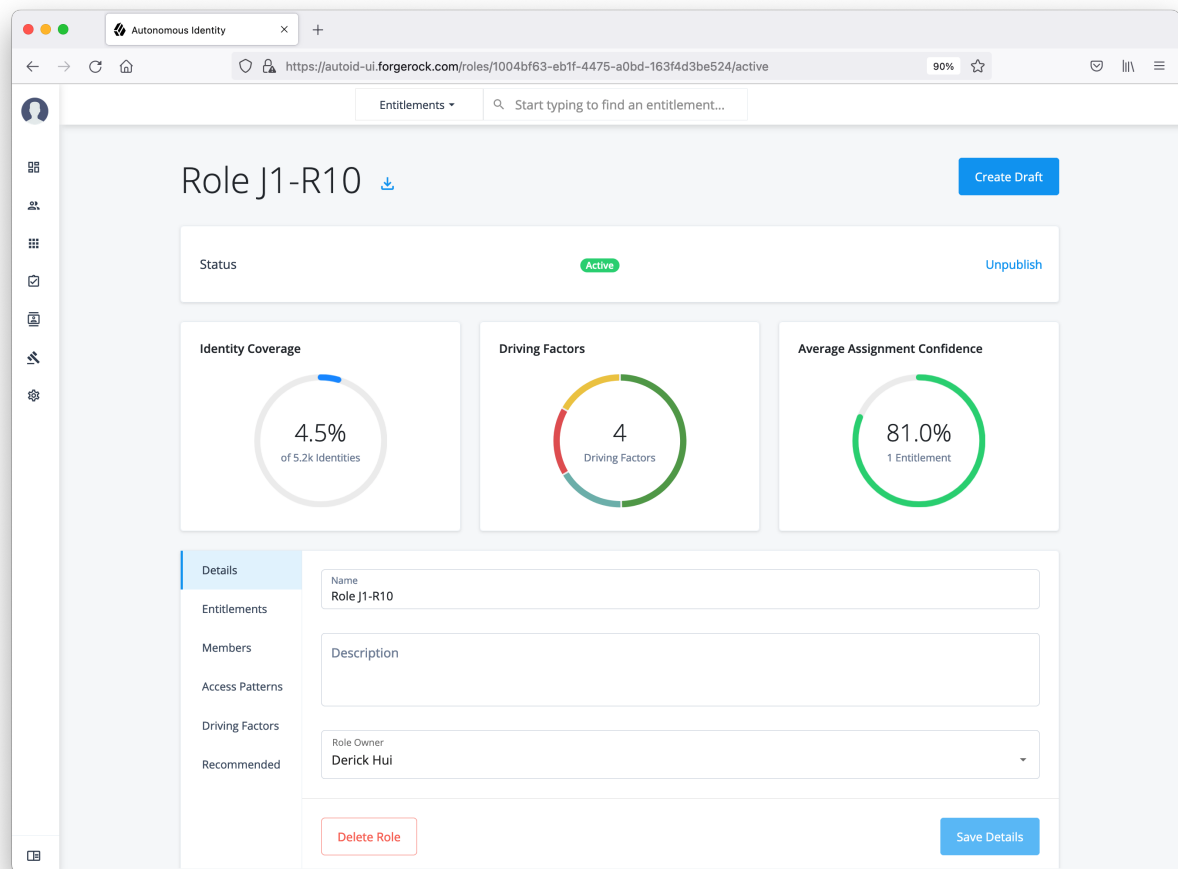
- **Candidate.** A *candidate* is a template role that is discovered through the latest role mining analytics job. After each role mining job, all newly *mined* roles are marked as new and as a *candidate*. Role engineers can review a candidate, assign a role owner to it, and approve the role as a *draft*. You cannot edit or remove a candidate role as is, but must create a draft from a candidate to change its details. Candidate roles are retained in the system for later adjustments and for the creation of additional new roles until the next role mining job, where all candidates are deleted and a new candidate pool is rebuilt.



- **Draft.** A *draft* is a role that requires review and approval by an authorized approver to become active. Role engineers can re-run a role mining job to pick up the latest changes in the access landscape. The Roles Workshop displays the latest confidence scores, driving factors, and a *Recommended* section that shows a suggested course of action for the role. Also, when you create a custom role, PingOne Autonomous Identity saves the role in draft status. You can edit the draft, make another custom role from it, publish the role for production, or delete the draft.



- **Active.** Once a draft has been approved, the role is *active* for production use. The role appears with an *Active* status and also appears on the Roles Catalog page. The Recommended section presents suggested updates for each role analyzed in the latest mining job. You can create a draft from this active role or unpublish it back to *draft* status.



Role-mined vs custom roles

You can originate roles in two different ways: role-mined and custom.

Role-mined roles are discovered through PingOne Autonomous Identity's machine learning process. The result of the role mining run is a generated list of candidates that a role engineer can edit and review on the Roles Workshop page.

Custom roles are created through different workflows:

- **From Scratch.** You can create a totally new role on the Roles Workshop using the Create Role function.
- **From Existing.** You can create a custom role from an existing draft or active role, which can occur in the following scenarios:
 - When you run a new role-mining job and an existing candidate role is deleted.
 - When you have a draft or an active role that is associated with a deleted candidate, and the recommendation is to delete the draft/active role as role mining determines that it is stale or no longer relevant.

Note

Custom roles do not have recommendations. Recommendations are based on the difference between a mined role and its candidate.

Roles workshop tasks

The following procedures presents the typical Roles Workshop tasks:

Create a custom role

PingOne Autonomous Identity lets authorized users create new custom role drafts on the Roles Workshop.

1. On the Roles Workshop, click **Create Role**. PingOne Autonomous Identity creates a random label for the role at the top of the page.
2. Click **Details**. Enter a name, description, and select the Role Owner from the list of authorized users on the drop-down menu.
3. Click **Entitlements**. On the page, click **Add Entitlements**. You can search by entitlement name or application.
4. Click **Access Patterns**. On the page, click **Add Access Pattern**. Select a User Attribute, and enter an associated value.
5. Review your entries. When ready, click **Save Draft**.

The role will be saved in the Roles Workshop as a draft. An authorized user must review and approve the role to activate it in production.

The screenshot displays the PingOne Roles Workshop interface. At the top, there's a navigation bar with a search bar and a 'Create Role' button. The main content area is divided into three sections: 'Assignment Coverage' (0.8% for 47.9k Assignments), 'Average Assignment Confidence' (88.4% for 368 Assignments), and 'Latest Role Mining Job' (Run ID: job_0, Run Date: August 12, 2021 1:07 PM, Confidence Threshold: 75%, Entitlement Threshold: 1, Frequency Threshold: 30, Stem Enabled: true). Below these sections is a table titled 'Viewing: 8 Roles' with columns for Role, Members, Entitlements, and Status. The table lists five roles (Role J0-R1 to Role J0-R5) with 52 members each, 1 entitlement, and a 'Candidate' status. A 'Filters' button is located to the right of the table.

Role	Members	Entitlements	Status
Role J0-R1 NEW	52	1	Candidate
Role J0-R2 NEW	52	1	Candidate
Role J0-R3 NEW	52	1	Candidate
Role J0-R4 NEW	52	1	Candidate
Role J0-R5 NEW	52	1	Candidate

Warning

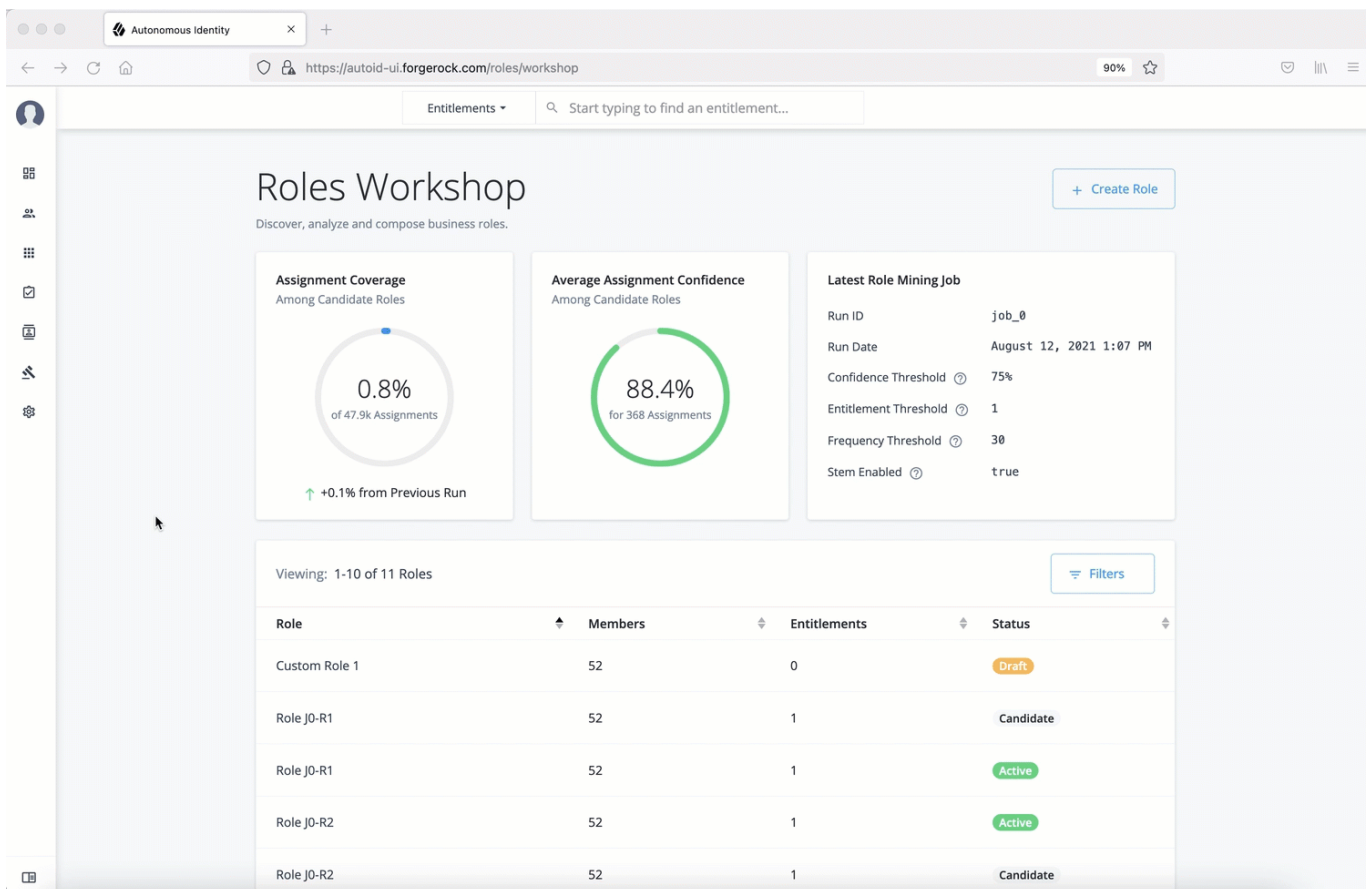
Do not create a custom role before running an initial analytics pipeline job. Doing so can result in the role mining job failing.

Search roles using the filter

Many companies have a large number of roles within their system. The Roles Workshop provides a useful filter to locate specific roles.

Search Roles using the filter:

1. On the Roles Workshop, click **Filters**.
2. Enter any of the following data:
 1. **Name**. Enter a role name.
 2. **Status**. Select **Active**, **Draft**, or **Candidate**.
 3. **Application**. Enter an application with which the role is associated.
 4. **Role Owner**. Enter a role owner.
 5. **Origin**. Select **Custom** or **Role Mining**.
3. Click **Done**.



Roles Workshop

Discover, analyze and compose business roles.

+ Create Role

Assignment Coverage Among Candidate Roles

0.8%
of 47.9k Assignments

↑ +0.1% from Previous Run

Average Assignment Confidence Among Candidate Roles

88.4%
for 368 Assignments

Latest Role Mining Job

Run ID	job_0
Run Date	August 12, 2021 1:07 PM
Confidence Threshold	75%
Entitlement Threshold	1
Frequency Threshold	30
Stem Enabled	true

Viewing: 1-10 of 11 Roles

Filters

Role	Members	Entitlements	Status
Custom Role 1	52	0	Draft
Role J0-R1	52	1	Candidate
Role J0-R1	52	1	Active
Role J0-R2	52	1	Active
Role J0-R2	52	1	Candidate

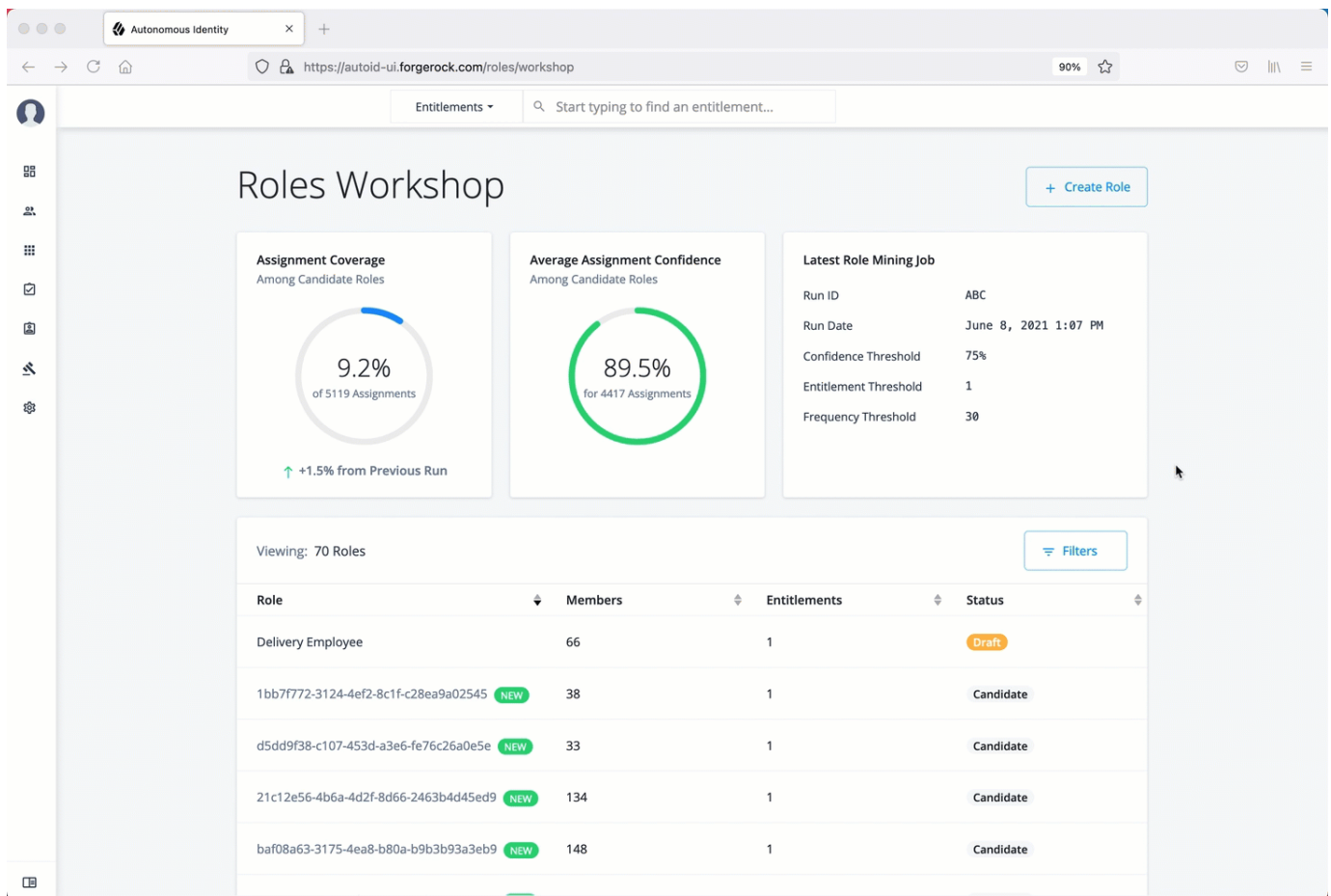
Create a new draft

PingOne Autonomous Identity lets authorized users review role-mined candidates on the Roles Workshop. If the role engineer and role owner approves the candidate, the role goes to draft state.

Create a New Draft:

1. On the Roles Workshop, review the list of candidates. Click a candidate role.
2. Change the role name, and then add a description.
3. Add an authorized user as a role owner. All drafts or active roles must have a role owner assigned to it.

Note: The role owner must have the Role Owner role assigned to them. Administrators can add them on the Manage Identities page, or make a request to your administrator.
4. Click **Entitlements** to review the entitlement name, application if assigned, and the average confidence score.
5. Click **Members** to review the members or users associated with this role.
6. Click **Access Patterns** to review the access pattern for this role.
7. Click **Driving Factors** to review the list of attributes, driving factors, frequency, and percentage of members with the role.
8. Click **Recommended** to review any suggested action on the role. You must run the role mining job several times to pick up new changes in your access landscape. Initial role mining jobs will not have recommendations other than Create Draft.
9. Click **Create Draft** when all entries are accepted.

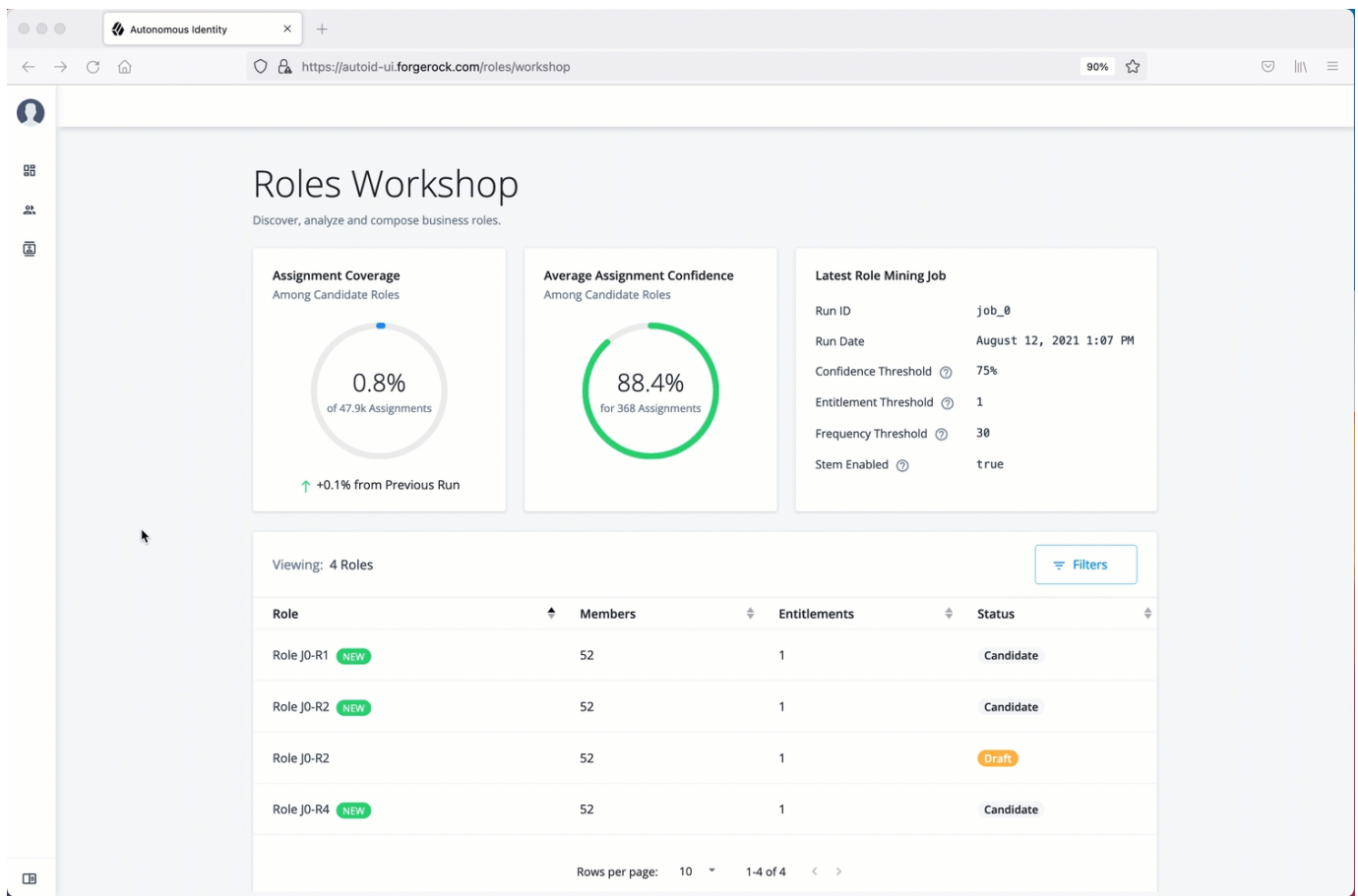


Publish a role

PingOne Autonomous Identity lets role engineers and role owners approve a draft and push it into production. The Roles Workshop displays the role in an *active* state.

Publish a Draft:

1. On the Roles Workshop, review the list of candidates. Click a draft role.
2. Review the role details.
3. Click **Publish**.

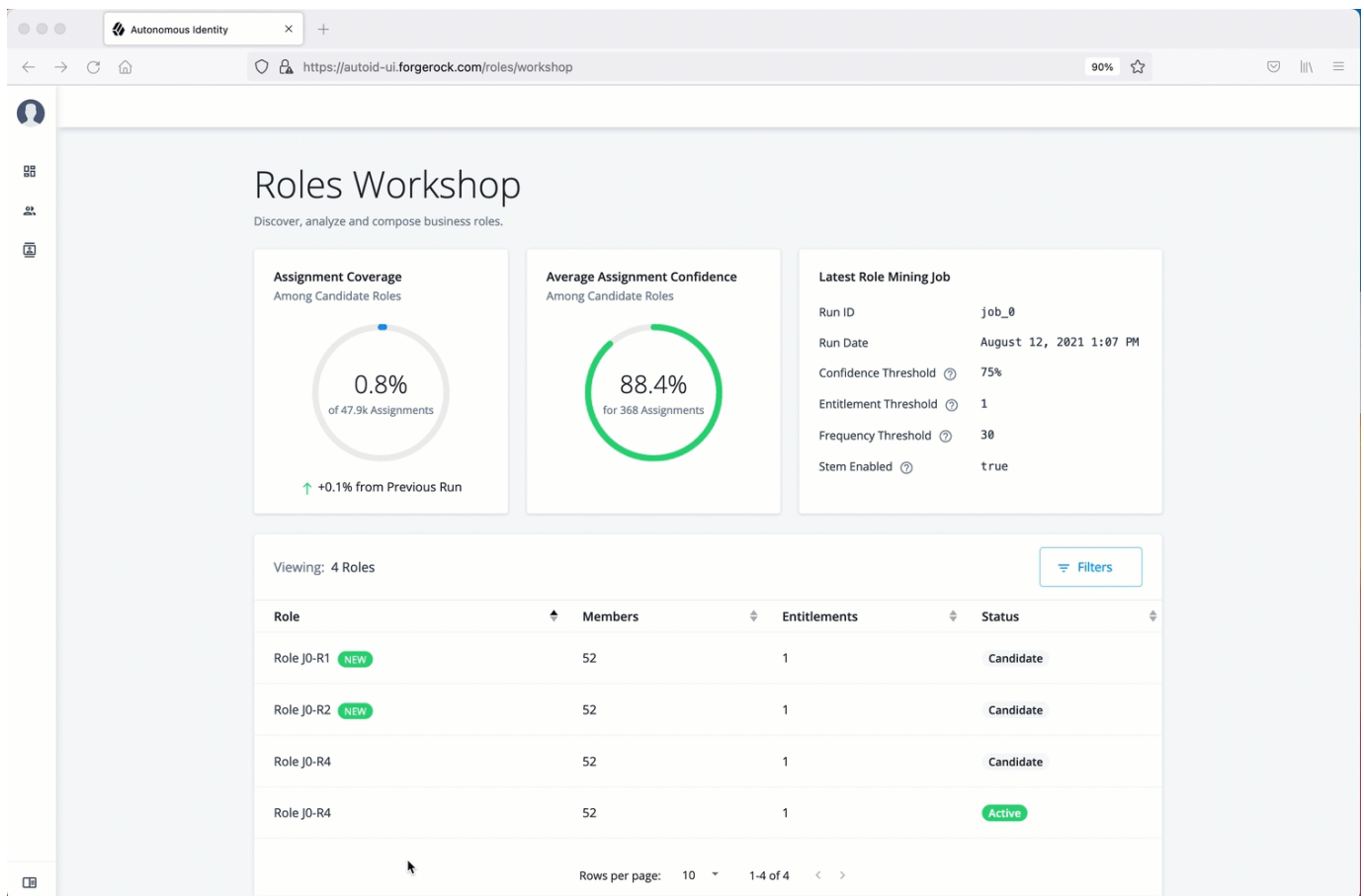


Delete a role

PingOne Autonomous Identity lets role engineers and role owners delete a draft or an active role.

Delete a Role:

1. On the Roles Workshop, review the list of candidates. Click a draft or active role.
2. Review the role details.
3. Click **Delete Draft**.



Roles catalog tasks

The Roles Catalog lists all active roles within your system. Role engineers can export any role as a json file, edit, unpublish, and delete the roles if necessary. Role owners can carry out the same tasks on only roles assigned to them.

Export a role

Export a Role to JSON:

1. On the UI Dashboard, click **Roles**, and then click **Roles Catalog**.
2. Click the role(s) to export.
3. Click **Export** Selected. PingOne Autonomous Identity sends a JSON file to your local drive.

Filter roles in the catalog

Search the Catalog using the filter:

1. On the UI Dashboard, click **Roles**, and then click **Roles Catalog**.
2. Click **Filters** to view specific roles in your catalog.
3. On the Filters menu, enter and select from the following drop-down lists:
 1. **Name**. Enter a role name.

2. **Status.** Select `Active`, `Draft`, or `Candidate`.
 3. **Application.** Enter an application with which the role is associated.
 4. **Role Owner.** Enter a role owner.
 5. **Origin.** Select `Custom` or `Role Mining`.
4. Click **Done**.

Troubleshooting

The following section provides information to help you troubleshoot PingOne Autonomous Identity. The topics are:

- [Where to access the logs](#)
- [How to change the Docker root folder](#)
- [Tune Cassandra for large data](#)

Note

More troubleshooting tips will be added in the future.

Where to access the logs

PingOne Autonomous Identity captures information in its log files to troubleshoot any problem.

Deployment logs

When running the ansible playbook during the deployment, logs print to your screen (STDOUT). You can access additional information is available through the `-v` or `--verbose`. For more information, try `-vvx`. To enable connection debugging, try `-vvvv`.

The Cassandra install log file (`installcassandra.log`) is located at `/data/opt/autoid/cassandra`.

Front-end logs

You can view any output logs of the running services on Docker using the following commands:

```
docker service logs <SERVICE NAME> --follow
docker service ps <SERVICE NAME> --no-trunc
```

Cassandra logs

You can view any output logs of the Cassandra database, which is kicked-off at startup. PingOne Autonomous Identity pipes the output message to a log file in the standard installation folder.

Cassandra Log Locations

Log	Location
Standard Cassandra log	/data/opt/autoid/cassandra.out
Backup Log	/data/opt/autoid/cassandra/cassandra-backup/cassandra-backup.log
Other Logs	/data/opt/autoid/apache-cassandra-<version>/logs

Analytic logs

You can specify the path for analytic logs in the configuration file.

Note

You can change the file path to any path, but it must always be within the same base path.

For example, you can determine the analytic files in the `/data/other/logs`.

Spark UI logs

If the Spark UI is not available on port 8080 of the Spark master server, then do the following:

- Check the Spark start-up logs. Check if the status of Spark UI port 8080 is not the default port, or if there is another service using the port.
- If the UI is not accessible, run some `curl` commands to check the core and memory in the cluster.

```
curl -s https://<ip-address>:8080 | grep -A 2 'Memory in use'
curl -s https://<ip-address>:8080 | grep -A 2 'Cores in use'
```

Note

For more information, Refer to [Spark REST API](#).

How to change the Docker root folder

Docker stores its images in the root `/var` folder. Customers who mount `/var` with low storage can run out of disk space quickly.

How to change the Docker root folder?

1. Stop the Docker service:

```
sudo systemctl stop docker.service
sudo systemctl stop docker.socket
```

2. Edit the Docker service file, and add a `-g` option to the file to redirect the root folder to another location:

```
sudo vi /usr/lib/systemd/system/docker.service
```

```
ExecStart=/usr/bin/dockerd -g /opt/autoid/docker -H fd:// --containerd=/run/containerd/
containerd.sock
```

3. Make a new folder for the Docker root if needed:

```
sudo mkdir -p /opt/autoid/docker
```

4. Copy all of the content from the old Docker root folder to the new Docker root folder:

```
sudo rsync -aqxP /var/lib/docker/* /new/path/docker/.
```

5. Reload the system daemon:

```
sudo systemctl daemon-reload
```

6. Start the docker service:

```
sudo systemctl start docker
```

7. Make sure Docker is running with right arguments. The output should show Docker is running with right parameters set:

```
ps aux | grep -i docker | grep -v grep
```

Tune Cassandra for large data

You can tune Cassandra for large data sets or when Cassandra times out during analytics.

1. Navigate to Cassandra Folder:

```
cd /opt/autoid/apache-cassandra-3.11.2/conf
```

2. Edit the `jvm.options`, and change the Java heap size and the size of the heap size for young generation as follows:

```
-Xms10G
-Xmx10G
-Xmn2800M
```

3. Edit the `cassandra.yaml` file, and change the files as follows:

```
vi cassandra.yml
```

```
key_cache_size_in_mb: 1000
key_cache_save_period: 34400
max_mutation_size_in_kb: 65536
commitlog_segment_size_in_mb: 128
read_request_timeout_in_ms: 200000
write_request_timeout_in_ms: 200000
request_timeout_in_ms: 200000
counter_write_request_timeout_in_ms: 200000
cas_contention_timeout_in_ms: 50000
truncate_request_timeout_in_ms: 600000
slow_query_log_timeout_in_ms: 50000
concurrent_writes: 256
commitlog_compression:
- class_name: LZ4Compressor
```

4. After saving the file, restart the Cassandra and Docker jobs:

1. First, find the Cassandra job:

```
ps -ef | grep cassandra. // find the PID
```

2. Kill the Cassandra PID.

```
kill -9 PID
```

3. Make sure no Cassandra process is running:

```
ps -ef|grep cassandra
```

4. Restart Cassandra:

```
cd /opt/autoid/apache-cassandra-3.11.2/bin
nohup cassandra > /opt/autoid/apache-cassandra-3.11.2/cassandra.out 2>&1 &
```

5. Make sure to check if the following information message or similar is present:

```
INFO [main] 2022-01-24 23:38:26,207 Gossiper.java:1701 - No gossip backlog; proceeding
```

6. Restart Docker:

```
sudo systemctl docker restart
```

Users tasks

This chapter provides background information to understand how to read the PingOne Autonomous Identity UI, confidence scores, and different page views for non-administrators.



Features

Learn about the PingOne Autonomous Identity features.



User types

Learn about PingOne Autonomous Identity user types.



The UI

See an overview of PingOne Autonomous Identity's UI.



Supervisor tasks

Learn about the Employee Overview page and Supervisor tasks.



Application owner tasks

Learn about the Applications page and application owner tasks.



Entitlement owner tasks

Learn about the Entitlement Owner page and tasks.

Features

PingOne Autonomous Identity provides the following features:

- **Broad Support for Major Identity Governance and Administration (IGA) Providers.** PingOne Autonomous Identity supports a wide variety of Identity as a Service (IDaaS) and Identity Management (IDM) data including but not limited to comma-separated values (CSV), Lightweight Directory Access Protocol (LDAP), human resources (HR), database, and IGA solutions.
- **Highly-Scalable Architecture.** PingOne Autonomous Identity deploys using a microservices architecture, either on-prem, cloud, or hybrid-cloud environments. PingOne Autonomous Identity's architecture supports scalable reads and writes for efficient processing.
- **Powerful UI dashboard.** PingOne Autonomous Identity displays your company's entitlements graphically on its UI console. You can immediately investigate those entitlement outliers as possible security risks. The UI also lets you quickly identify those entitlements that are good candidates for automated low-risk approvals or re-certifications. Users can also view a trend-line indicating how well they are managing their entitlements. The UI also provides an application-centric view and a single-page rules view for a different look at your entitlements.
- **Powerful Analytics Engine.** PingOne Autonomous Identity's analytics engine is capable of processing millions of access points within a short period of time. PingOne Autonomous Identity lets you configure the machine learning process and prune less productive rules. Customers can run analyses, predictions, and recommendations frequently to improve the machine learning process.
- **UI-Driven Schema Extension.** PingOne Autonomous Identity lets administrators discover and extend the schema, and set up attribute mappings using the UI.
- **UI-Driven Data Ingestion and Mappings.** PingOne Autonomous Identity provides improved data ingestion tools to define multiple csv input files needed for analysis and their attribute mappings to the schema using the UI.
- **Broad Database Support.** PingOne Autonomous Identity supports both Apache Cassandra and MongoDB databases. Both are highly distributed databases with wide usage throughout the industry.
- **Improved Search Support.** PingOne Autonomous Identity now incorporates Open Distro for Elasticsearch, a distributed, open-source search engine based on Lucene, to improve database search results and performance.

PingOne Autonomous Identity User Types

PingOne Autonomous Identity recognizes six different user types, or personas, within its system. Each user type has access to certain pages on the PingOne Autonomous Identity console.

- **Admin.** An *Admin* user is similar to the notion of a system administration *superuser* within PingOne Autonomous Identity. Admins have access to every PingOne Autonomous Identity page view within the console. The Admin user can view the list of critical entitlements, approve or revoke access, and run other tasks.
- **Executive.** An *Executive* user is a senior manager within a company. Executives have access to the PingOne Autonomous Identity company overview page, critical entitlements, employee page, user entitlements page, but cannot approve or revoke access, or certify entitlements to users.
- **Supervisor.** A *Supervisor* user is one who has responsibility of other users or things and grants access to resources for these users. Supervisors can only see the entitlements of those users who report to them. They cannot view the entitlement assignments of users who report to another supervisor. Supervisors can certify entitlements assigned to users, entitlements to unscored users, and approve or revoke access.
- **Application Owner.** An *application owner* is any person or thing that owns an application and every entitlement within that application. A single entitlement can have an entitlement owner and an application owner. The application owner can have the permissions to approve, auto-certify entitlement assignments, and approve or revoke rule justifications.
- **Entitlement Owner.** An *Entitlement Owner* is one who has the ability to grant access to entitlements that they manage to other users. Entitlement owners can only view the entitlements that they have created. Entitlement owners can certify the entitlements that they manage, users to these entitlements, and approve or revoke access to these entitlements.
- **Role Engineer.** A *Role Engineer*
- **Role Owner.** A *Role Owner*
- **Role Auditor.** A *Role Auditor*
- **User.** A *user* is any person or thing that has access to a resource. General users cannot access the system.

Table: Summary of PingOne Autonomous Identity Users and Accessible Views

User Type/ View	Dashboard	Identities	Applications	Entitlements	Roles	Rules
Admin	✓	✓	✓	✓	✓	✓
Executive	✓					
Supervisor		✓		✓	🔍[1]	
Application Owner			✓	✓	🔍[1]	✓
Entitlement Owner				✓	🔍[1]	✓

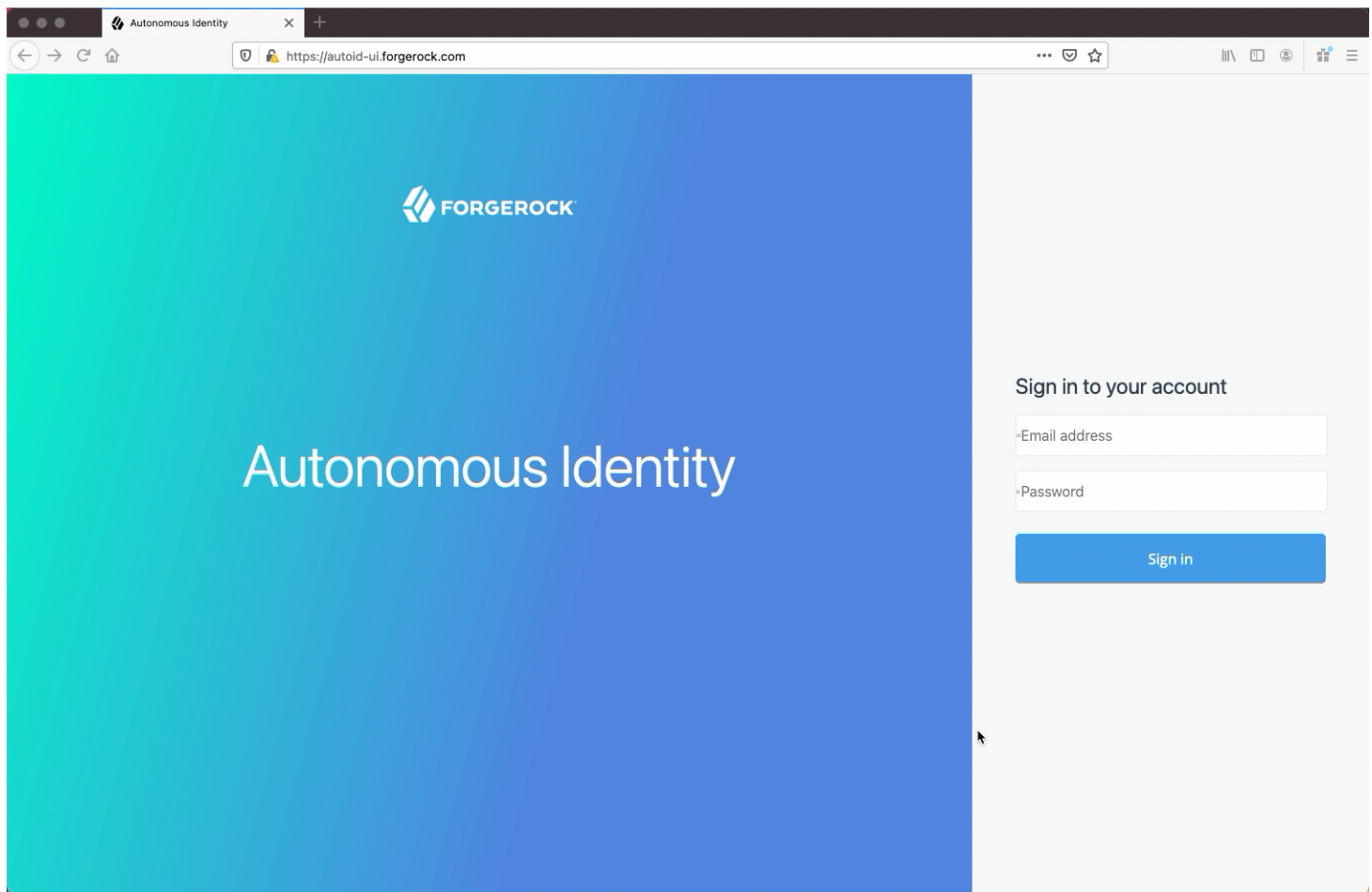
[1] If assigned a Roles user type: Roles Engineer, Roles Owner, or Roles Auditor

The PingOne Autonomous Identity UI

PingOne Autonomous Identity provides a powerful UI dashboard, displaying all of your entitlements, attributes, and confidence scores across your company. The UI provides different filtered levels of information depending on the user's access rights.

Dashboard

The Dashboard, also known as the **Company Overview**, provides a complete summary of your organizations's entitlements, confidences scores, and entitlement assignments. Only admin users and executives can view this page. The page also shows the trend lines of your confidence score history over time plus a list of the Most Critical Entitlements.



The Dashboard is partitioned into several modules as you scroll down:

- **Model Coverage.** Displays data on model coverage and confidence scoring of the assigned entitlements. The section summarizes the total number of entitlements processed by PingOne Autonomous Identity, and the number and percentage of those entitlements that were covered and not covered by the system. The section also displays a summary of entitlement *assignments*, specifically the number of High Confidence Assignments (90% and above), Low Confidence Assignments (20% and below), total assignments, and number of unscored entitlements. "Unscored" indicates that PingOne Autonomous Identity could not learn any patterns for a specific entitlement to properly assign a confidence score to it.

- **Confidence Score Distribution of Entitlements.** Displays a histogram of the distribution of confidence scores across your entitlements landscape. The chart provides a good summary of the current state of your entitlements landscape. In general, you want to set up your high confidence-scoring entitlements as candidates for automated approval and certification. You also want to move a good percentage of your middle level confidence scores to high confidence entitlements.
- **User Type.** Displays a summary of users versus non-users covered by the system.
- **Most Critical Entitlements.** Displays the list of the most critical entitlements with the low average confidence scores and the number of employees assigned with the entitlement. You can drill down to view each entitlement, where you can approve or remove access to the entitlement for that user.
- **All Entitlements Distribution.** Displays the number of one-to-one matching and the highly assigned entitlements to distinct users.
 - **One-to-one matching** indicates the number of entitlement assigned to one user only.
 - **Highly Assigned** indicates the number of entitlements assigned to a number of distinct users listed below. These highly-assigned entitlements are good candidates for automated access approval or certification using policies or roles.
 - **Graph of All Entitlements Distribution** displays a chart of the number of entitlements versus the number of users. The number range on the left (e.g., 0-5) indicate the number of entitlements assigned. The number on the right indicates the actual number of users. Thus, in the image below, there are 207 users who have 0-5 assigned entitlements. In the second row from the bottom, there are 979 users who have between 5-10 assigned entitlements. In the third row from the bottom, there are 1451 users who have between 10-100 assigned entitlements. In the fourth row from the bottom, there are 33 users who have between 100-1000 assigned entitlements. In the fifth row from the bottom, there are 0 users who have between 1k-10k assigned entitlements. In the sixth row from the bottom, there are 0 users who have between 10k-100k assigned entitlements. In the seventh row from the bottom, there are 0 users who have between 100k-1000k assigned entitlements.

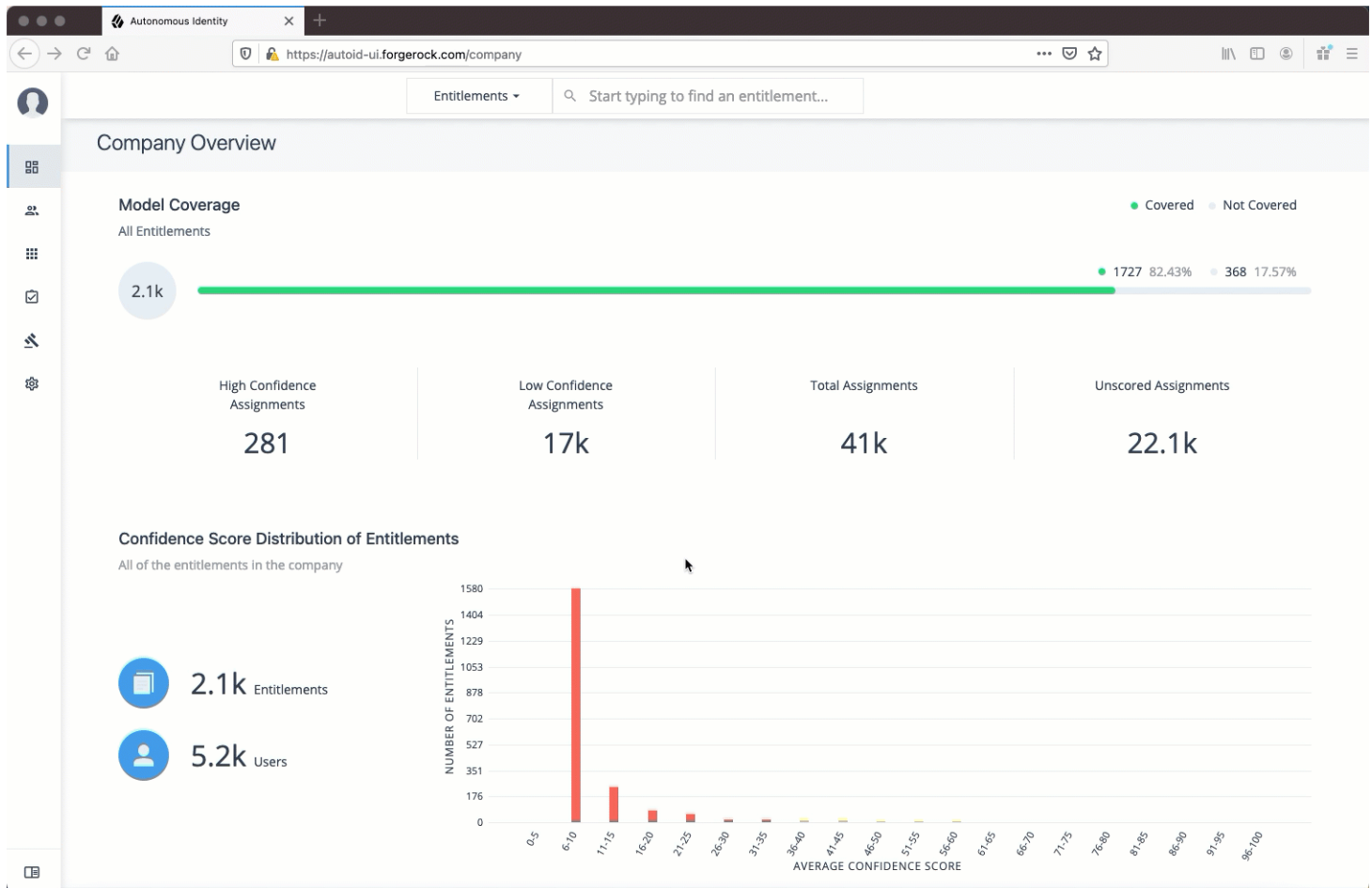


Figure 1. An Example of the All Entitlements Distribution Graph

- **History of Assignment Confidence Scores.** Displays a history of assigned confidence scores (high, medium, and low) over the past year (in months) versus the number of assignments. The time range on the x-axis shows the month and year for a specific range of assignments. This graph shows the confidence score trends over time and indicates how well you are managing your entitlements. In general, you want rising high and mid confidence scores and decreasing low score trends.

Identities

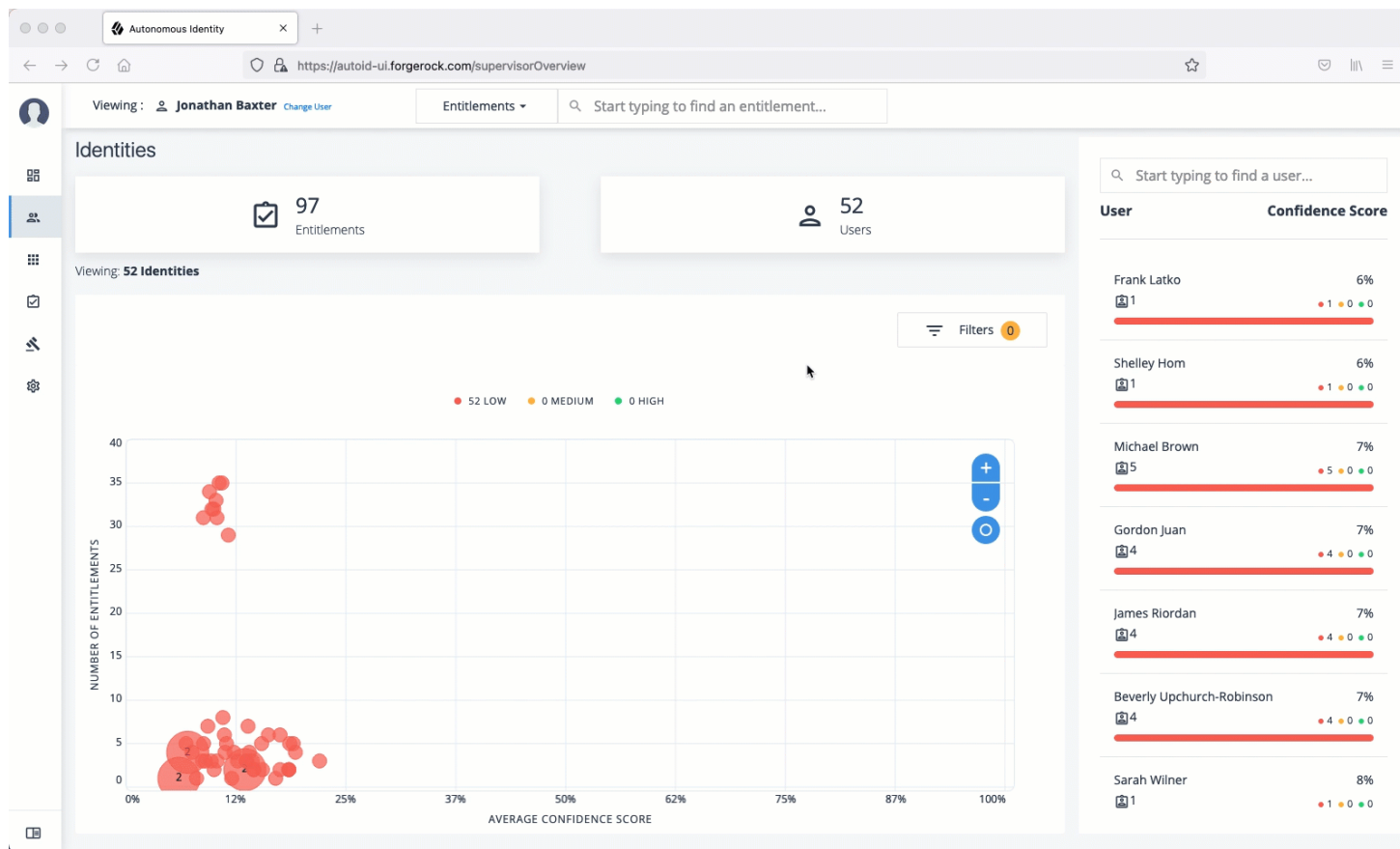
The Identities page displays a supervisor-based view of all users reporting to a specific supervisor and their entitlements. Admin users can see all supervisors and their users, while supervisors can only view their direct reports.



The Identities page is partitioned into several modules:

- **Total Number of Entitlements.** Displays the total number of entitlements assigned to users who report to the supervisor.
- **Total Number of Users.** Displays the total number of users that are assigned the entitlements.
- **Graph of Average Confidence Scores.** Displays a chart of the average confidence scores versus the number of entitlements. You can hover over each circle to see the user's name, average confidence score, and number of entitlements assigned. If you double-click a circle, you can select the user in the list on the right.
- **Filters.** Enable any of the application filters to display only those entitlements for a specific application. Enable the **Remove High Scores from Averages** filter to view only the mid and low confidence scores.
- **List of Users.** Displays a full list of users who have the assigned entitlements and their confidence scores. You can drill down and see each user's entitlements details by clicking on the user's name. To search for a specific user in the list, enter their name in the Search box above.

From the Identities page, you can view the User Details page by clicking a name in the right-hand menu, or by clicking the drop-down search at the top, select User, and enter the user's name.



The User Detail is partitioned into several areas that display the following:

- **Not Scored.** Click the button to see any entitlements that were not scored by the system. Click **Approve** to approve or **Revoke** the entitlement for the user.
- **Recommended.** Click the button to see any entitlements that were not assigned to the user but are good candidates to assign to the user.
- **Range of Confidence Scores.** Displays the low, medium, and high confidence scores for the assigned entitlements to the user. Click a circle to highlight the entitlement in the list below the graph.
- **Display Filter.** Displays a filter that matches any features set in the **Assignments** entity definition. This filter is also present on the Not Scored and Recommended pages.
- **Entitlements.** Displays the list of user's assigned entitlements, the application, and confidence score associated with the entitlement. Admins and supervisors can approve or revoke one or more entitlements for the user.

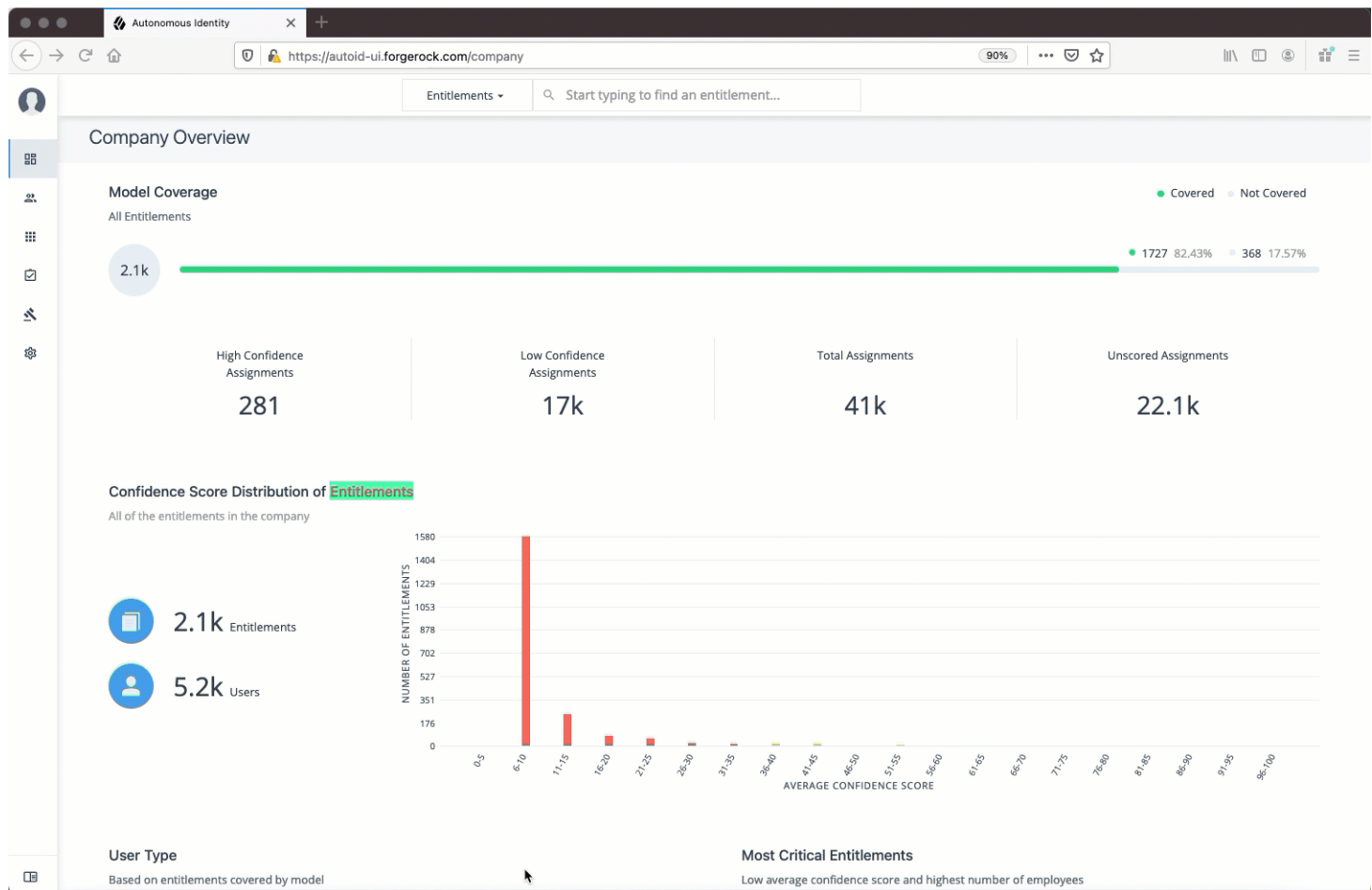
Click the down arrow to review entitlement details that helps you run the following:

- **Justifications.** Displays the attributes that lead to the confidence score.
- **Driving Factor Comparison.** Displays a comparison of attributes and the driving factors that lead to a high confidence scored compared to the user's attribute values.

- **Employees associated with the entitlement.** Displays the users, justifications, and confidence scores of users who also have the recommended entitlement.
- **User Detail.** On the right, the page displays user's attributes as ingested from the company's HR database.

Applications

The Applications page provides an app-centric view for application owners and admin users to view the entitlements and assignments for an application. Admin users must enter the application owner to view the entitlement information on an application.

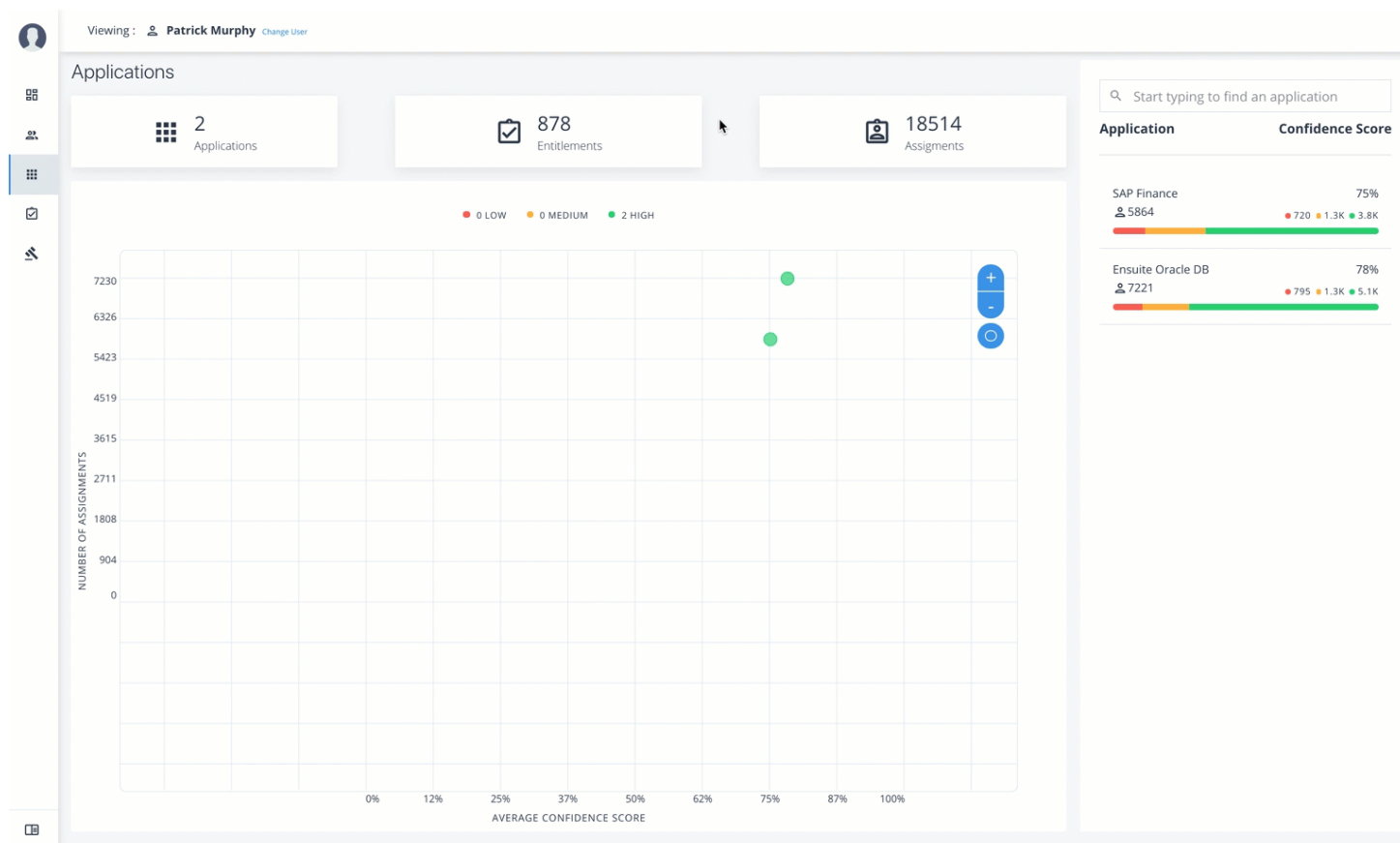


The Applications page is partitioned into several modules:

- **Total Number of Applications.** Displays the total number of applications for the application owner.
- **Total Number of Entitlements.** Displays the total number of entitlements that are associated with the applications.
- **Total Number of Assignments.** Displays the total number of entitlement assignments that are associated with the applications.
- **Graph of Average Confidence Scores.** Displays the Average Confidence Scores versus the Number of Assignments. You can hover over each circle to see the application's name, average confidence score, and number of users assigned to the application. If you double-click a circle, you can highlight an application on the right-hand list the list.

- **List of Application and Confidence Scores.** Displays the list of applications and confidence scores. If you click an application, you can drill down to the Application Details page to see more information. To search through your list, enter an application name to access it.

Application Details page is partitioned into several modules:



- **Total Number of Entitlements.** Displays the total number of entitlements associated with the application.
- **Total Number of Users.** Displays the total number of users who have access to the application.
- **Total Number of Rules.** Displays the total number of rules that are associated with the application.
- **Filters.** Displays options to filter the data based on entitlement attributes and user attributes.

The Application filters let you filter the viewable entitlements based on the following attributes:

- **Owner.** Filters the entitlements based on entitlement owner.
- **Risk Level.** Filters the entitlements based on risk level: low, medium, and high.
- **Criticality.** Filters the entitlements based on criticality of the entitlement: Essential or Non-Essential.

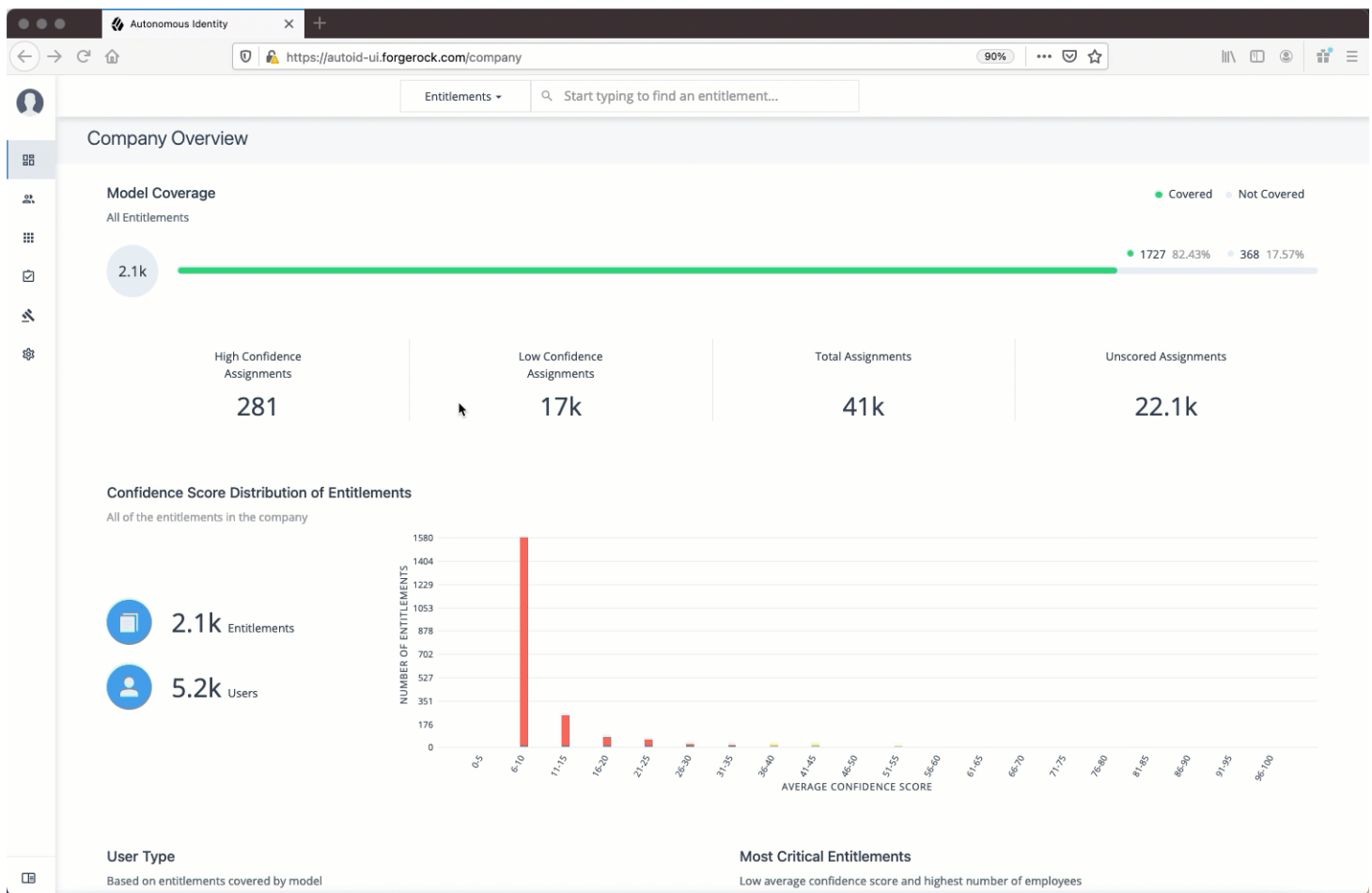
You can also filter based on driving factor attributes:

- **Manager Name.** Filters based on the manager's name. The menu displays the managers associated with the users who are assigned entitlements for the application.
- **Chief.** Filters based on if the user is a manager or not.

- **User Department Name.** Filters based on the department name. The menu displays the departments associated with the users who are assigned entitlements for the application.
- **Line of Business Subgroup.** Filters based on the Line of Business Subgroup. The menu displays the subgroups associated with the users who are assigned entitlements for the application.
- **Line of Business.** Filters based on the Line of Business. The menu displays the line of businesses associated with the users who are assigned entitlements for the application.
- **Cost Center.** Filters based on the cost center. The menu displays the cost centers associated with the users who are assigned entitlements for the application.
- **Job Code Name.** Filters based on the job code name. The menu displays the job code names associated with the users who are assigned entitlements for the application.
- **City** Filters based on the city. The menu displays the cities associated with the users who are assigned entitlements for the application.
- **User Employee Type.** Filters on user employee type, either **Employee** and **Non-Employee**.
- **Graph of Average Confidence Scores.** Displays the average confidence scores versus the number of users. You have the option to view bubbles or list view. You can hover over each circle to see the application's name to highlight it on the right-hand list. If you click list view, you can see the entitlement, user, confidence scores and an option to re-certify the entitlement for the user to access the application.
- **List of Entitlements and Confidence Scores.** Displays the list of entitlements and confidence scores for the application. If you click an entitlement, you can drill down to the Entitlement Detail page to see more information. To search for a specific entitlement, enter its name in the Search box.

Entitlements

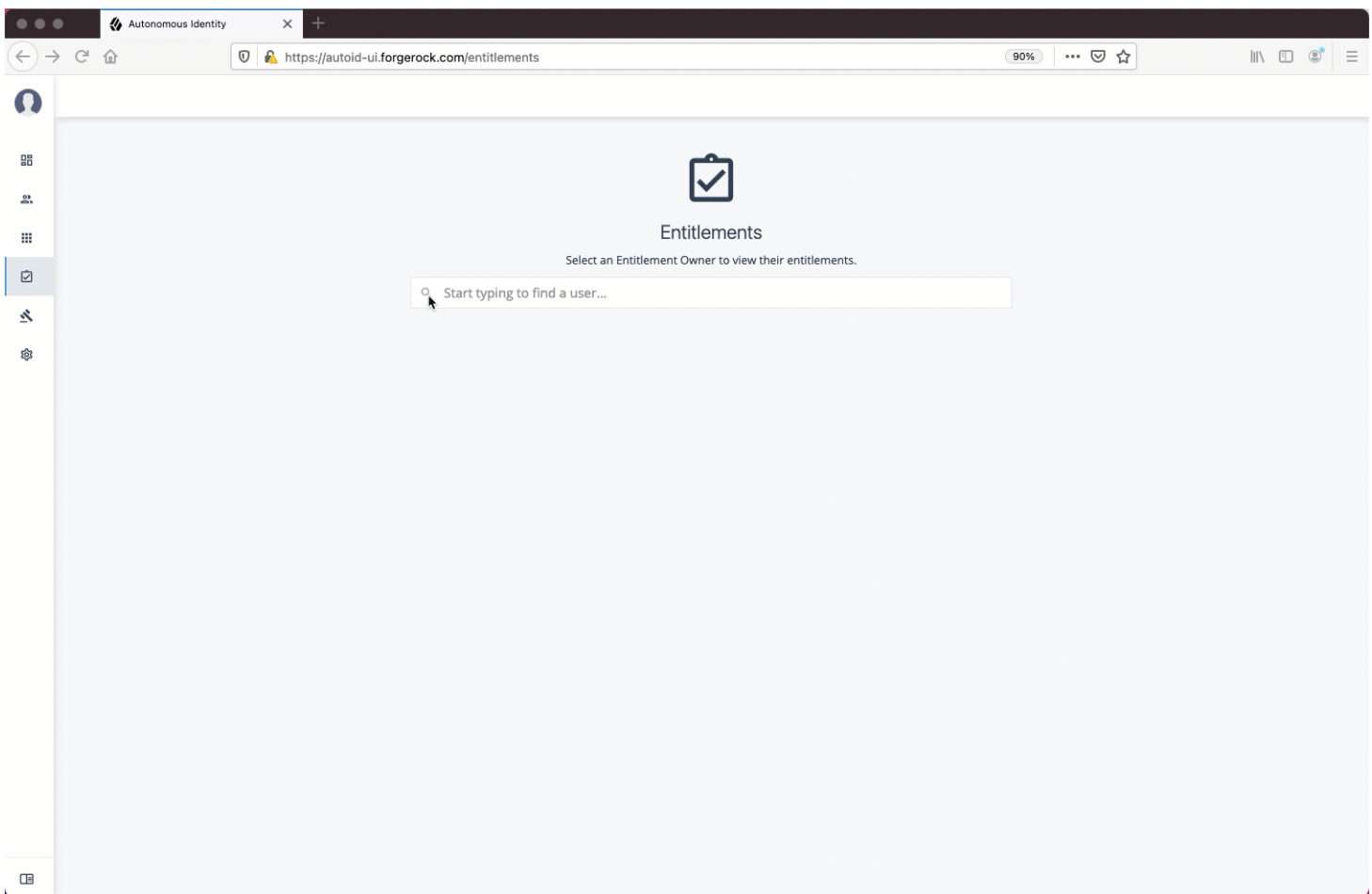
The Entitlements page provides an entitlement-centric view of an owner's entitlements. Entitlement owners cannot see the entitlements of other owners. Admin users can access this page and must enter an entitlement owner to view a specific entitlement.



The Entitlements page is partitioned into several modules as you scroll down:

- **Total Number of Entitlements.** Displays the total number of entitlements that the entitlement owner has responsibility for.
- **Total Number of Users.** Displays the total number of users that are assigned to the entitlements.
- **Graph of Average Confidence Scores.** Displays the Average Confidence Scores versus the Number of Users. You can hover over each circle to see the entitlement's name, average confidence score, and number of users with the assigned entitlement. If you double-click a circle, you can see the entitlement on the list on the right.
- **Filters.** Enable the **Remove High Scores from Averages** filter to view only the mid and low confidence scores. You can also filter based on one or more applications. Click **Add Filters** to further filter based on a user attribute, such as **city**.
- **List of Entitlements.** Displays a full list of entitlements and its average confidence score. You can drill down to see the Entitlement Details page by clicking on the entitlement's name. To search for an entitlement, enter it in the Search box.

When you drill down to view a specific entitlement, the Entitlement Details page is displayed with the following sections:

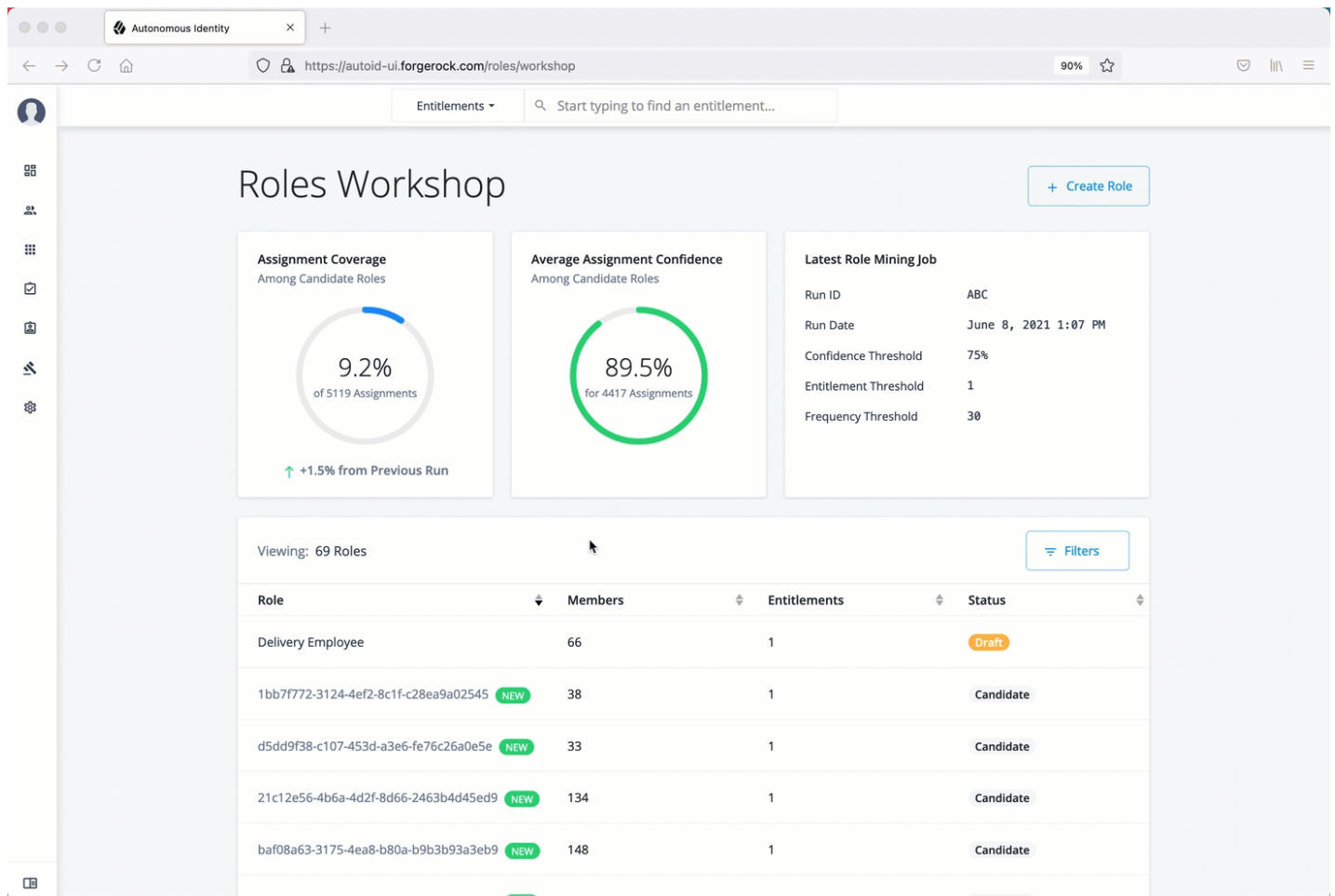


- **Average Confidence Score.** Displays the average confidence score for the entitlement.
- **Distribution of Users.** Displays the total number of users with the entitlements and the breakdown of low, medium, and high confidence scores.
- **Driving Factors.** Displays the driving factors, the attributes that lead to the confidence score. You can click the down arrow to see more information.
- **Graph of Average Confidence Score.** Displays a graph of the average confidence score versus the number of users with the confidence score. You can click one of the bars to highlight the justifications on the right-hand list.
- **List of Justifications.** Displays a list of justifications with the number of users and average confidence scores. You can click the right arrow to see the users with this entitlement and justifications. The checkbox next to each justification set lets you approve it. If you click a user's name, you can drill down to see the User Entitlements Detail page, which provides more detailed information from the user perspective.

Roles

PingOne Autonomous Identity introduces new UI pages, Roles Workshop and Roles Catalog pages for authorized users to work on new and current roles assignments.

The Roles Workshop provides a UI where authorized users can view, create, or edit any roles based on PingOne Autonomous Identity's latest role mining job. The page provides a **what-if** scenario with confidence scores based on the addition or removal of justifications. The Roles Workshop also displays a list of recommended roles that were discovered in the most recent role mining job.

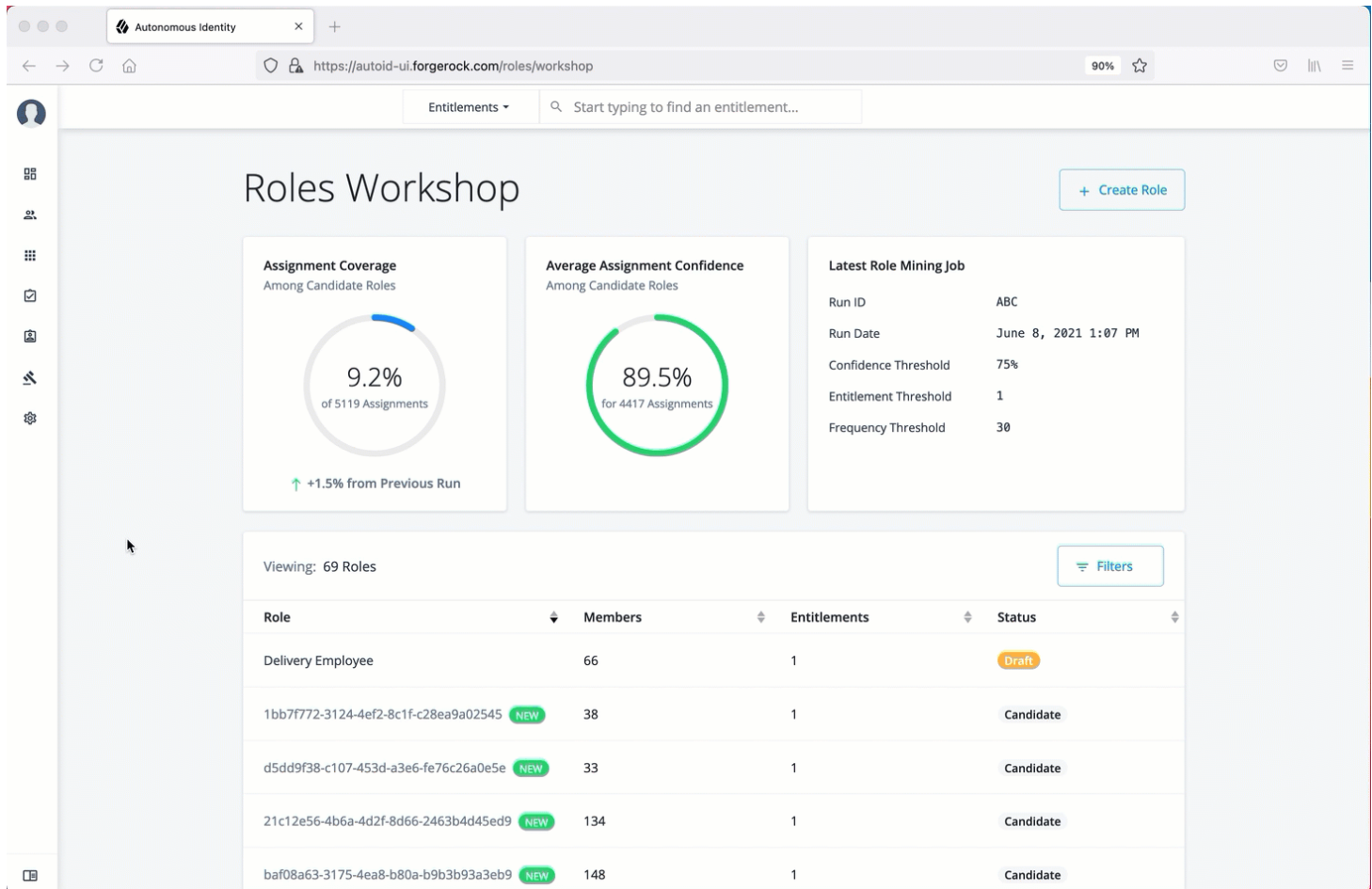


The Roles Workshop page is partitioned into several modules as you scroll down:

- **Assignment Coverage.** Displays data on average coverage percentage of the candidate roles to the total number of roles assigned. The model also displays any increase or decrease from the previous mining run.
- **Average Assignment Confidence.** Displays the average assignment confidence score to total assignment among the candidate roles.
- **Latest Role Mining Job.** Displays a timestamp for the latest role mining job along with confidence, entitlement, and frequency thresholds.
- **Table of Candidate Roles.** Displays a table of candidate roles discovered in the latest role mining job. The table displays the role, number of members associated with the role, number of entitlements in the role, and the status. The status displays the different workflow stages for the role.
 - **Candidate.** Indicates the recommended role discovered in the role mining run. Role engineers or role owners can review the candidate and approve the role to a draft status.
 - **Draft.** Indicates that the recommended role is in a draft state and requires review by an approver.

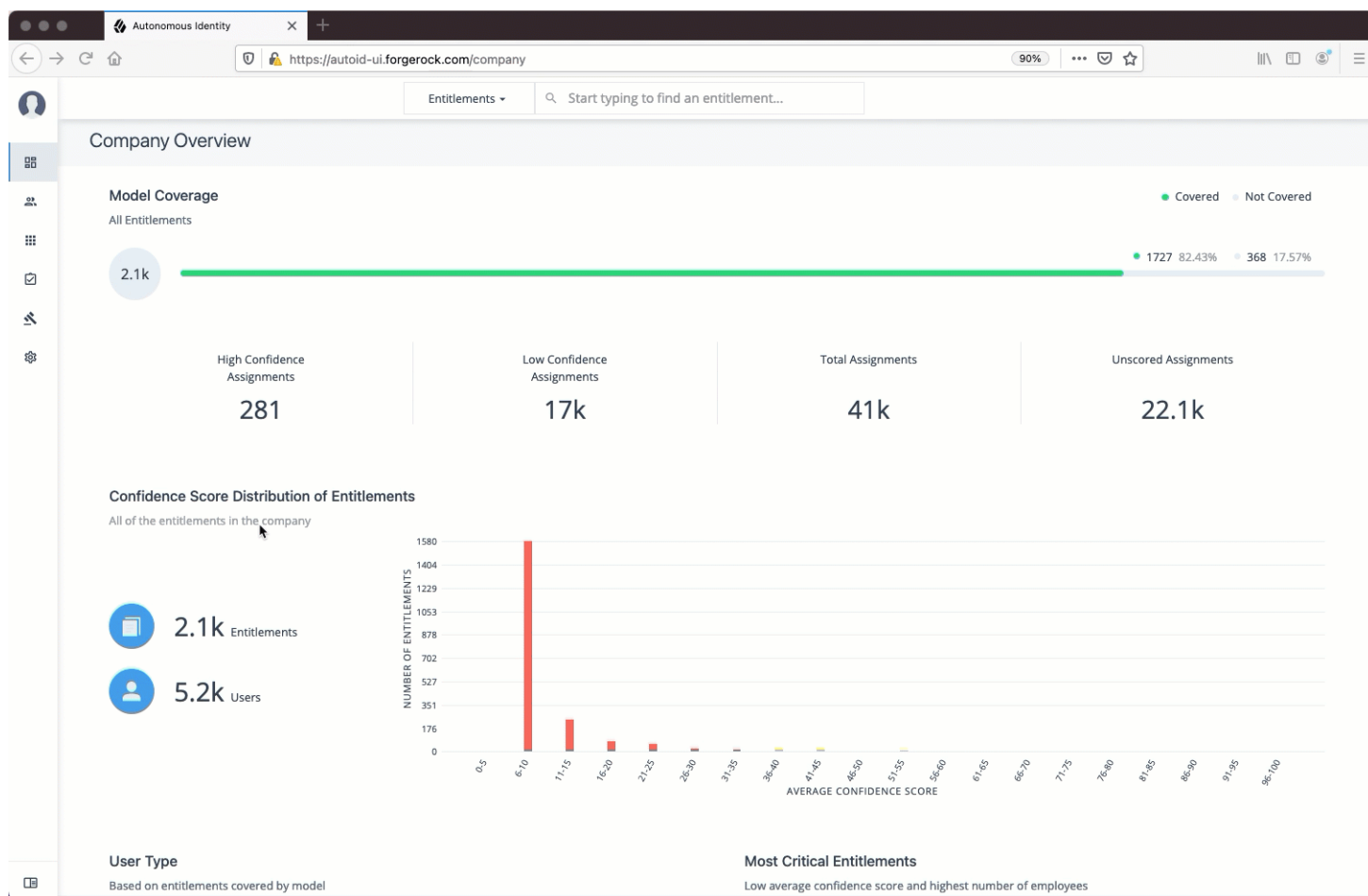
- **Active.** Indicates that the role is ready for production use.

The Roles Catalog page displays a history of your roles include the role name, number of members, number of entitlements, and current status. You can filter your list of roles based on name, application, role owner, or origin.



Rules

The Rules page displays a rules-centric view of the entitlements for application and entitlement owners. Admin users must search for an application or entitlement owner to view a rule.



The Rules page is partitioned into several modules as you scroll down:

- **Total Number of Rules.** Displays the total number of rules that the entitlement owner has responsibility for.
- **Total Number of Identities.** Displays the total number of identities that are assigned the entitlements.
- **Total Number of Applications.** Displays the total number of applications that are associated with the rules.
- **Filters.** Click **Filters** to view a segment of the total list. You can hide already reviewed auto-certified or auto-approved rules, low, medium, and high confidence scores, and by applications.
- **List of Entitlements and Justifications.** Displays the list of entitlements and their justifications. Entitlements with more than three justification attributes displays a **Show more** link. Application and entitlement owners can approve the entitlement based on the information displayed.

Click the down arrow on the right to view the user attributes, driving factors, and values for a specific identity. If more than one users exists for that rule, you can change the user under the Identity drop-down list. The icons on the right indicate if a justification is appropriate for the entitlement or not. You can drill down to see the user's details by clicking the View <user identity>.

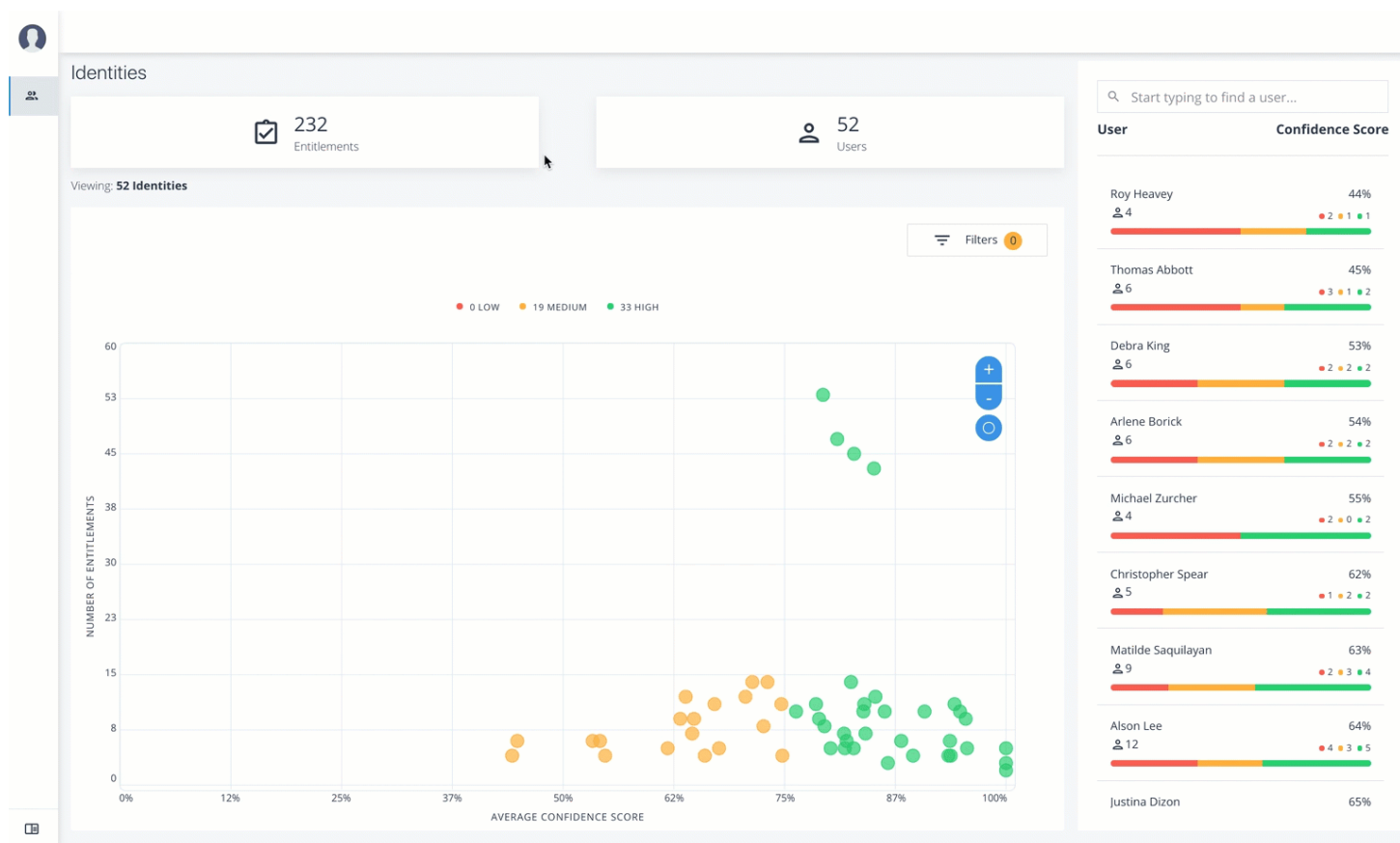
Supervisor Tasks

A Supervisor user is one who has responsibility of other users and grants or revoke access to resources for these users. A supervisor has access to the Employee Overview, User Detail, and User Entitlement Detail pages. Supervisors can only view their reports' information and cannot view the data of other supervisor's users.

Check Not Scored Users

Follow these steps to check Not Scored entitlements. *Not scored* indicates that there are no justifications associated with the entitlement:

1. Log in to the PingOne Autonomous Identity console.
2. On the Identities page, click a circle, and then click the user in the list on the right.
3. On the User Entitlement page, click Not Scored.
4. On the Not Scored Entitlements page, click the down arrow to view the driving factors comparison table.
5. Click Employees associated with this entitlement to view the justifications for those users with this entitlement.
6. Click Actions, and then click **Approve Access** or **Revoke access**. At a later date, you can re-click the Approve or Revoke button to cancel the operation.

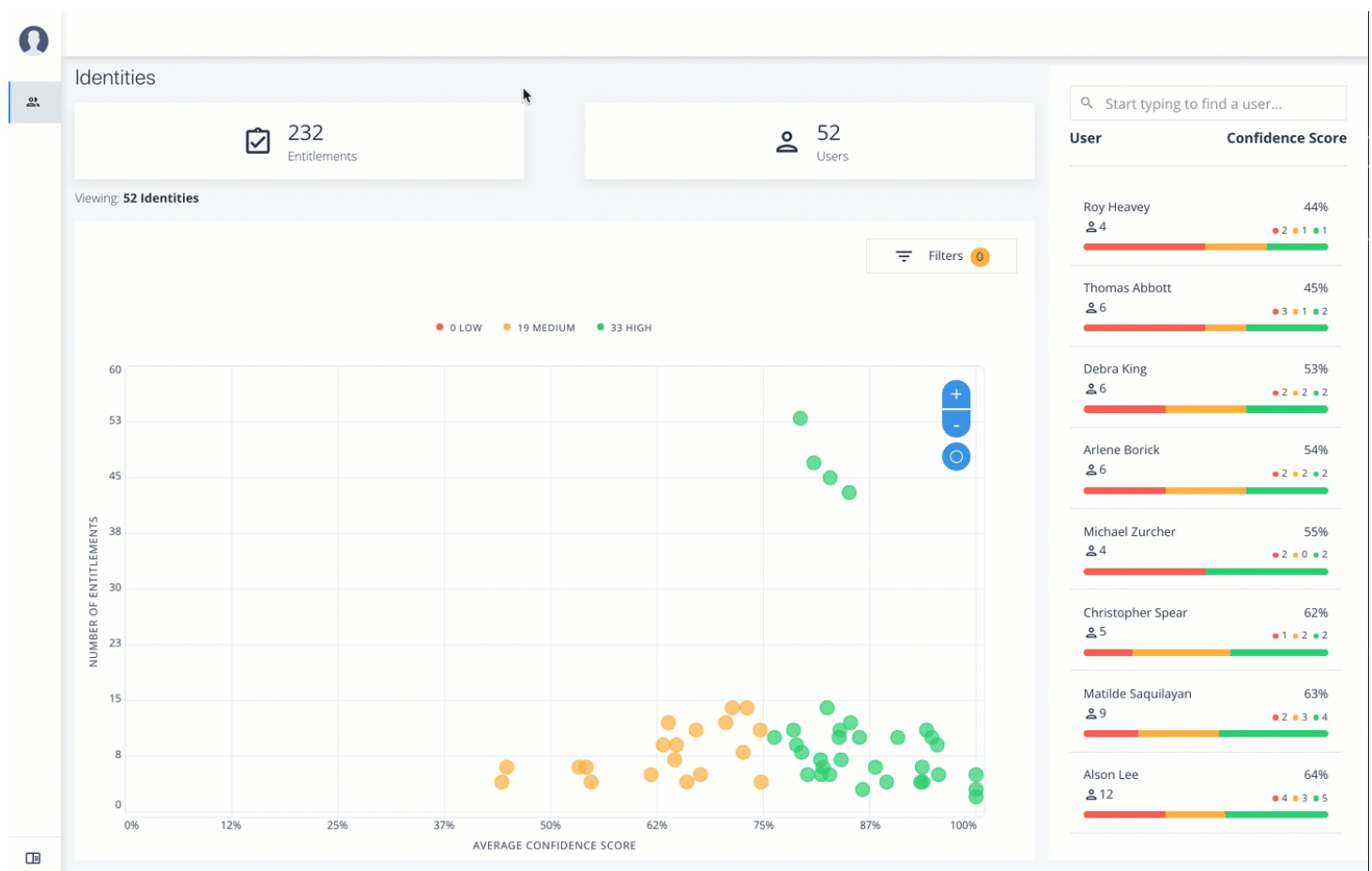


View Recommended Entitlements

Follow these steps to check Recommended entitlements.

The analytics engine determines if any entitlement, not currently assigned to a user, should be assigned to the user based on their attributes. PingOne Autonomous Identity generates a list of these *recommended* entitlements.

1. Log in to the PingOne Autonomous Identity console.
2. On the Identities page, click a circle, and then click the user in the list on the right.
3. On the User Entitlement page, click Recommended.
4. Review the recommended entitlement that PingOne Autonomous Identity determined was a good candidate for assignment to the user. Note that this page has no actions available since the entitlement is not assigned to the user. The page only presents information on the recommended entitlement.
5. Click the down arrow to view more information. View the Justifications that lead to the confidence score. Review the Driving Factor Comparison table. Click Employees associated with this entitlement to compare users with this entitlement.

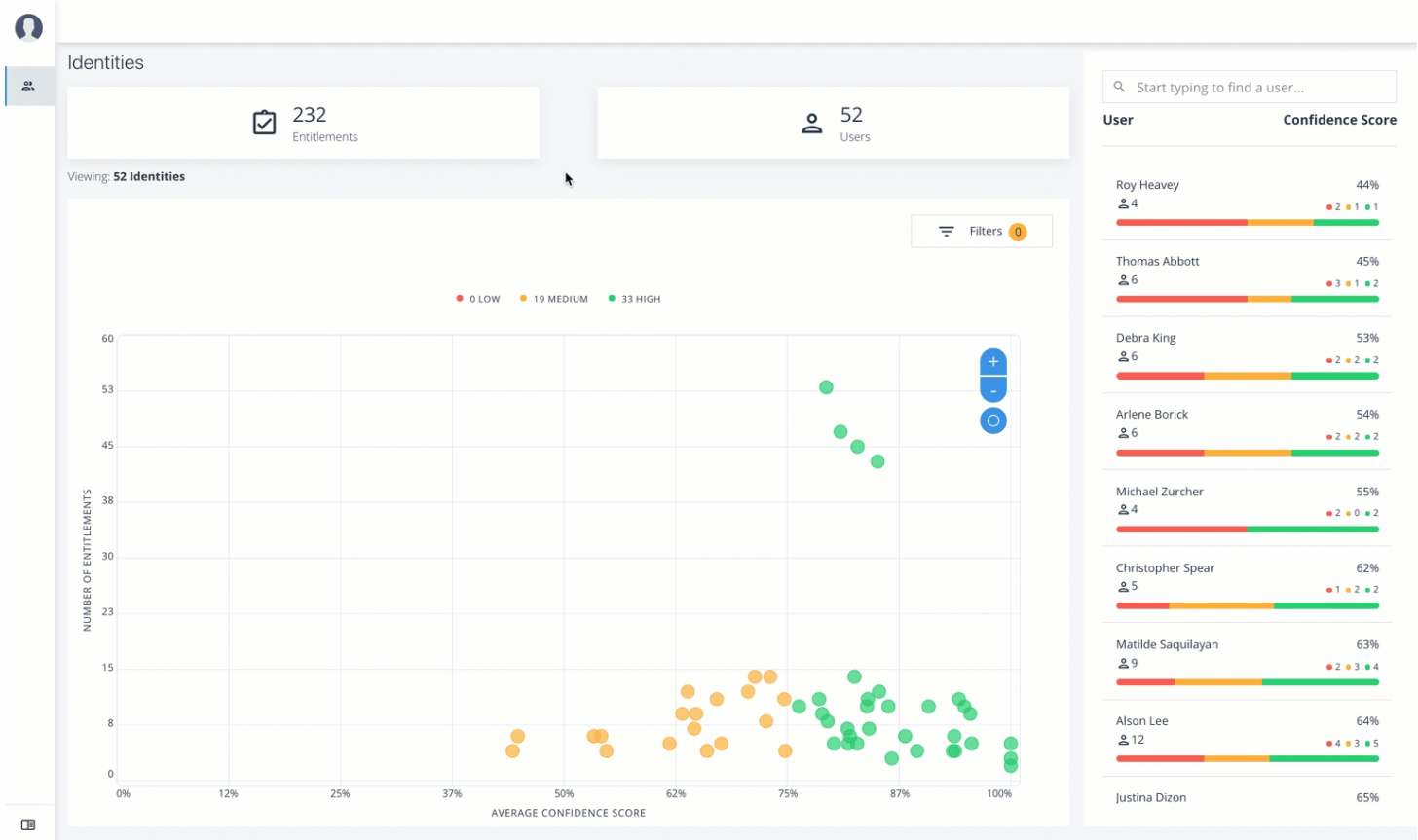


Approve or Revoke Access

Follow these steps to investigate a confidence score and approve or revoke access an entitlement assigned to a specific user:

1. Log in to the PingOne Autonomous Identity console.

- On the Identities page, click a circle, and then click the user in the list on the right.
- On the User Entitlement page, click a confidence circle on the graph to highlight the entitlement below.
- For the selected entitlement, click the down arrow on the right to view the Driving Factor Comparison.
- Click Employees associated with this entitlement to view the justifications for those users with this entitlement.
- Click Actions, and then click **Approve Access** or **Revoke access**.

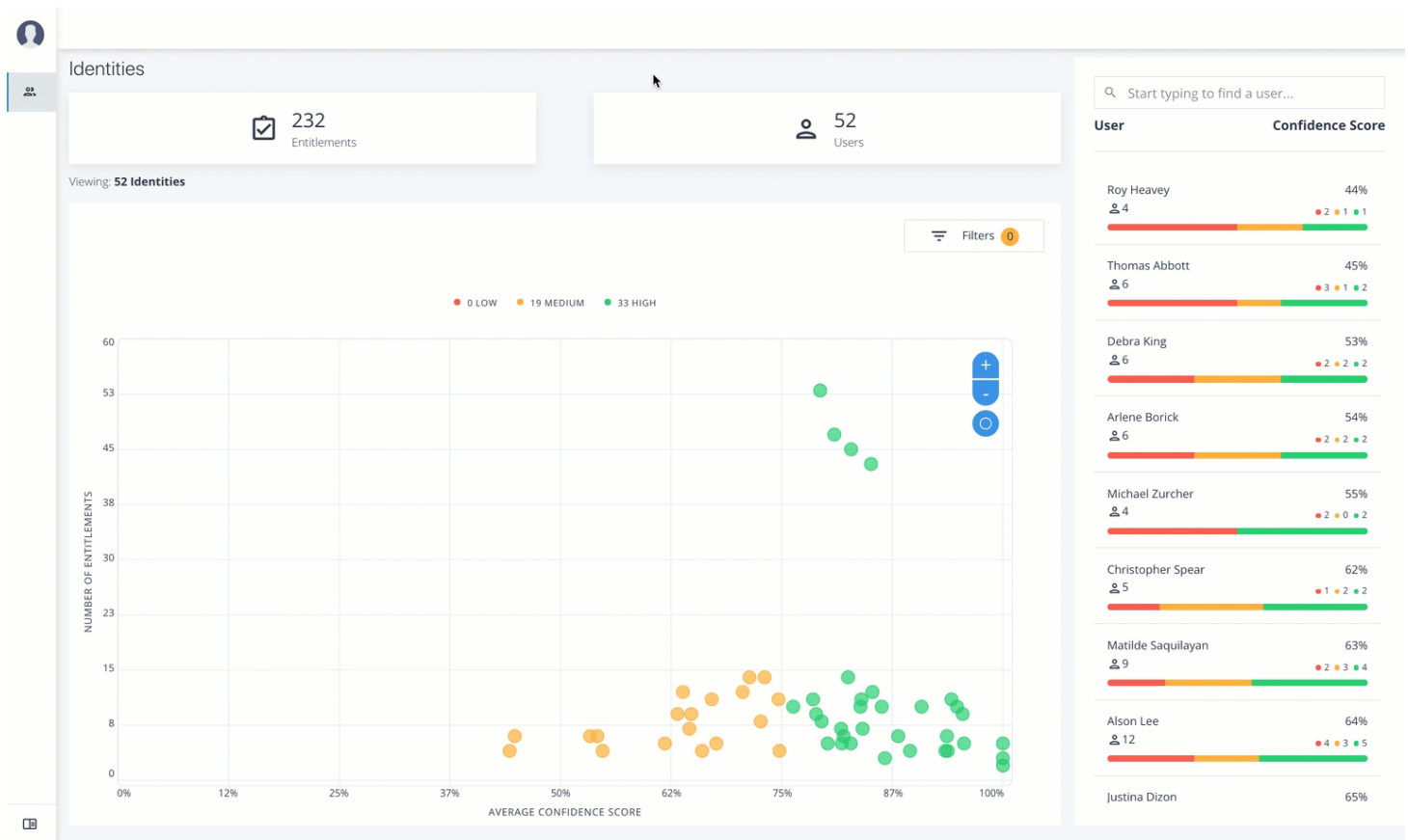


Apply Filters

Follow these steps to apply filters to your confidence score graphs on the Identities page:

- On the Identities page, view the average confidence score graph.
- On the right, click Filters.
- Under filters, do one or all of the following:
 - Click **Remove High Scores from Average** or enable any filter in the Application Filters section.
 - Under Applications, click one or more applications to see the identities or entitlements associated with the selected application.
 - Click Add Filters to further see only those identities or entitlements based on a user attribute, such as **city**. When ready, click Apply Filters.

4. Click Clear Filters to remove your filters.



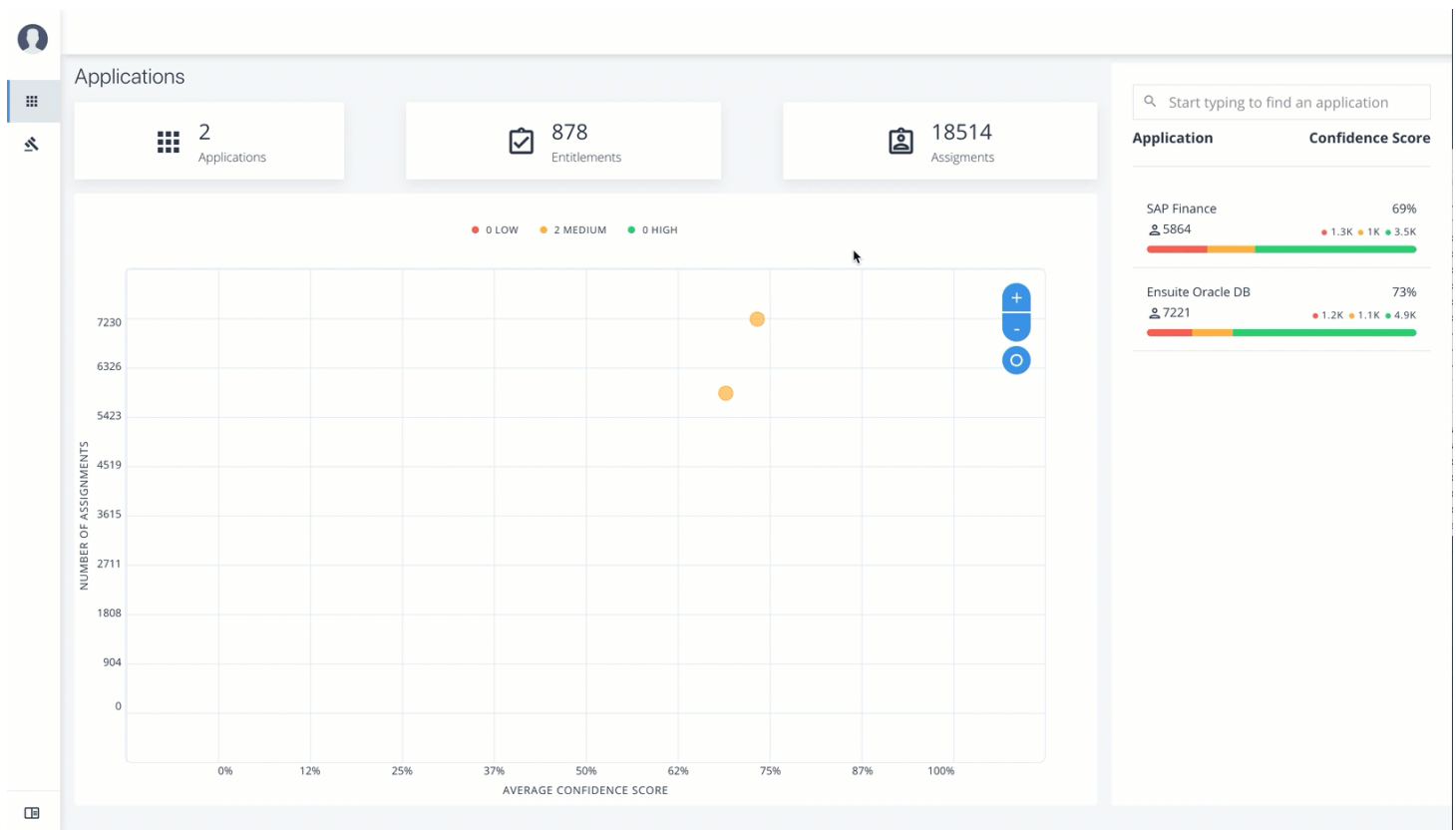
Application Owner Tasks

The Applications lets an application owner view their applications and all associated entitlements.

View Applications

Follow these steps to view applications:

1. As an Application Owner, log in to the PingOne Autonomous Identity console.
2. On the Applications page, click a circle in the graph or an application in the Applications list on the right.
3. On the Applications Detail page, review the information on the page: the number of entitlements associated with the application, the number of users, the number of rules, and a graph of the average confidence score versus number of users.
4. To view the list of entitlements for the application ordered by confidence score, click the list icon on the top left. From there, click Re-certify to approve the entitlement assignment for the application.



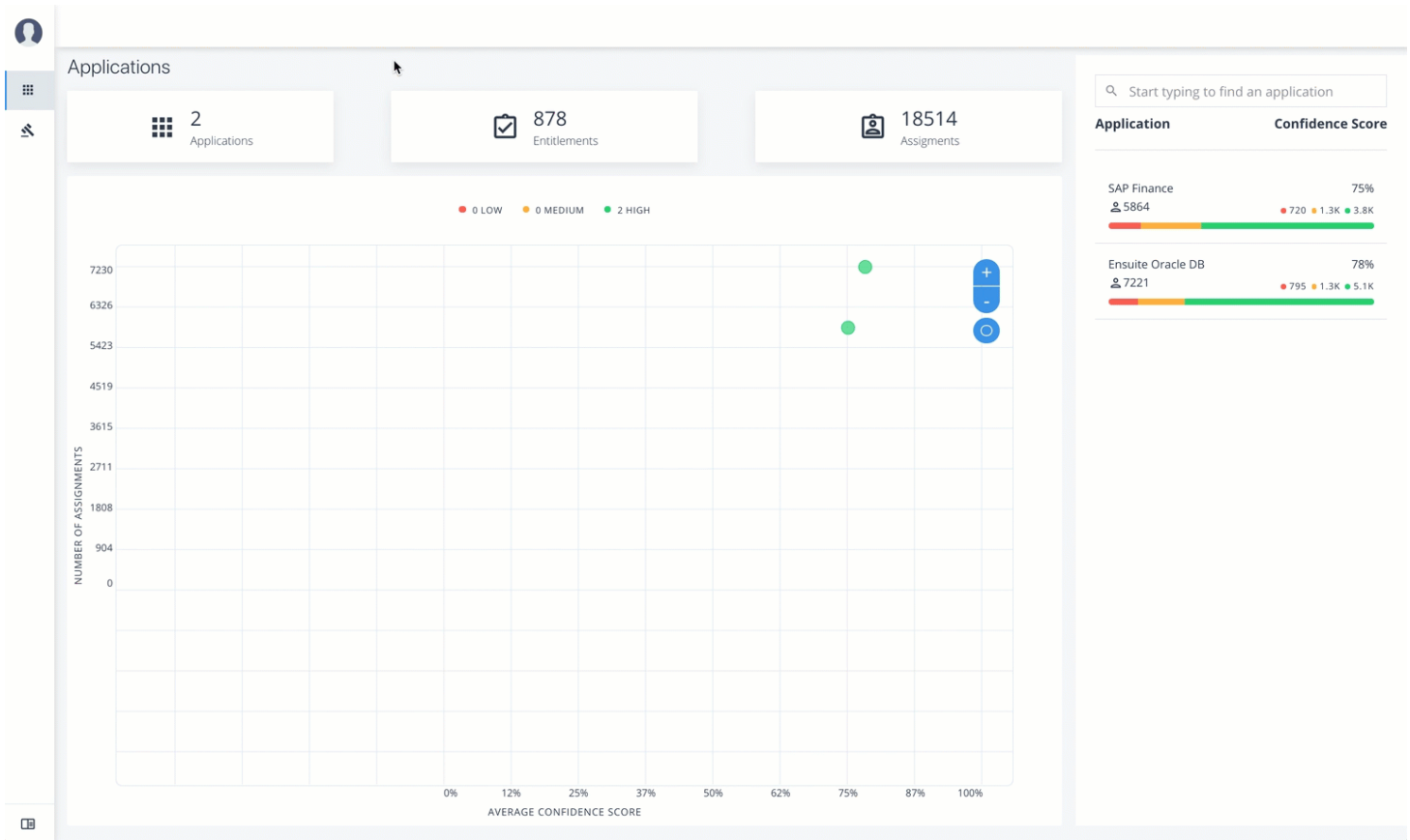
Apply Filters

Follow these steps to apply filters to your confidence score graphs:

1. Log in to the PingOne Autonomous Identity console.
2. On the Applications page, click **Filters**.
3. Under Entitlement Attributes, do one or all of the following:
 - Click Owner to filter on the entitlement owner. You can make more than one selection.
 - Click Risk Level to filter on low, high, and middle risk entitlements. You can make more than one selection.
 - Click Criticality to filter on Essential or Non-Essential entitlements.
4. Under User Attributes, do one or all of the following:
 - Click Manager to filter on a manager. You can make more than one selection.
 - Click Chief to filter if the entitlement is manager or not.
 - Click Department to filter on a specific department. You can make more than one selection.
 - Click LOB Sub Group to filter on a line of business subgroup. You can make more than one selection.
 - Click LOB to filter on the line of business for the division. You can make more than one selection.
 - Click Cost Center to filter on a cost center. You can make more than one selection.

- Click Job Code Name to filter on a job code. You can make more than one selection.
- Click City to filter on the city of the operations. You can make more than one selection.
- Click Employee Type to filter Employee or Non-Employee.

5. Click Apply Filters to see the results on the graph. You can cancel your filters by click the **clear filters** link.

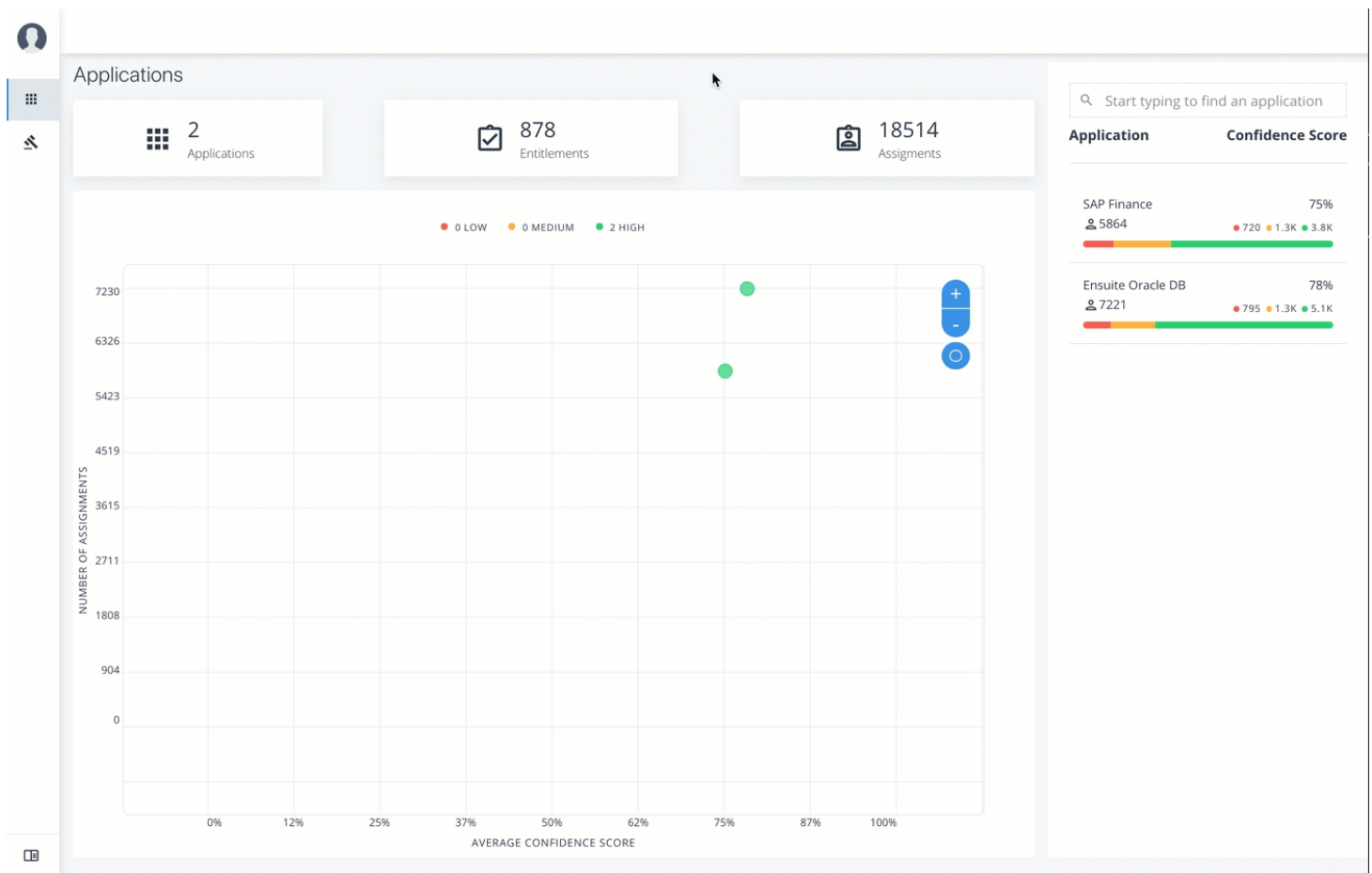


Re-certify Entitlement Assignments

Follow these steps to re-certify an entitlement assignment:

1. Log in to the PingOne Autonomous Identity console as an Application Owner.
2. On the Applications page, select an application to view by clicking a circle in the graph or the application on the right-hand menu.
3. Click list view.
4. Click Re-Certify, and then click Re-Certify again to confirm the assignment.

You can also select all or multiple entitlements for bulk re-certify.



Approve Rule Justifications

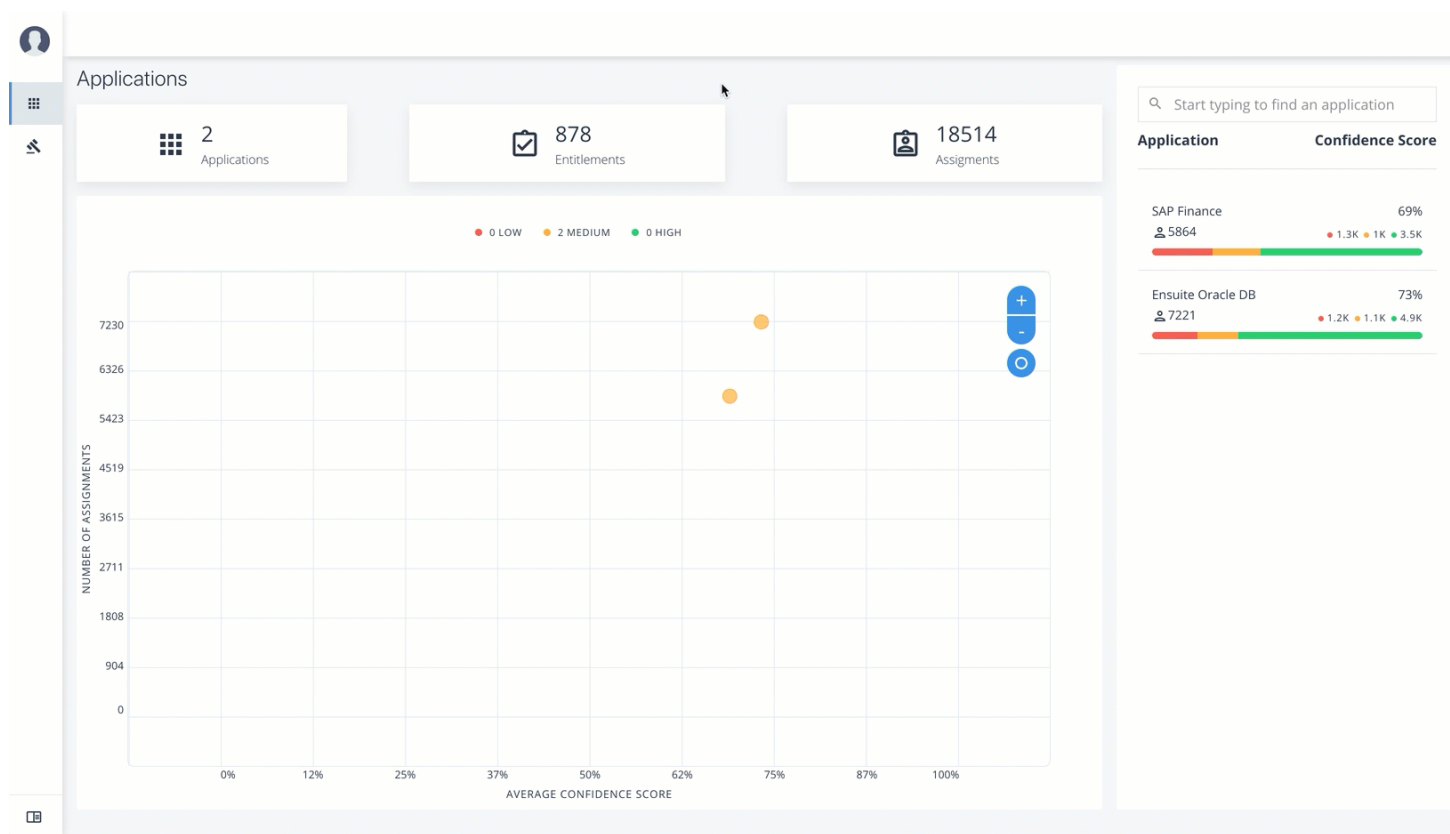
Follow these steps to apply rule justifications for an entitlement:

1. Log in to the PingOne Autonomous Identity console.
2. Click Rules.
3. On the Rules page, select an entitlement to view, and then click the down arrow to see the driving factors for the entitlement.
4. Under Identity, change to see another user's attributes and driving factors. If you want to see the user's entitlements page, click View <User>.
5. After researching the entitlement, click Approve. Click Auto Certify or Auto Request or both, and enter a reason for the approval. Click Submit Approval when ready.

You can also select all or multiple entitlements to do a bulk approve. PingOne Autonomous Identity only allows a single revoke action at a time.

Note

Auto Certify indicates that any user who has this justification is automatically approved for this entitlement. *Auto Request* indicates that anyone who matches these set of criteria and may not already have access, automatically gets provisioned for this entitlement.



Entitlement Owner Tasks

An *Entitlement Owner* is one who has responsibility for a given access to a resource, but may not be a supervisor. Entitlement owners can only carry out tasks on those entitlements they are responsible for.

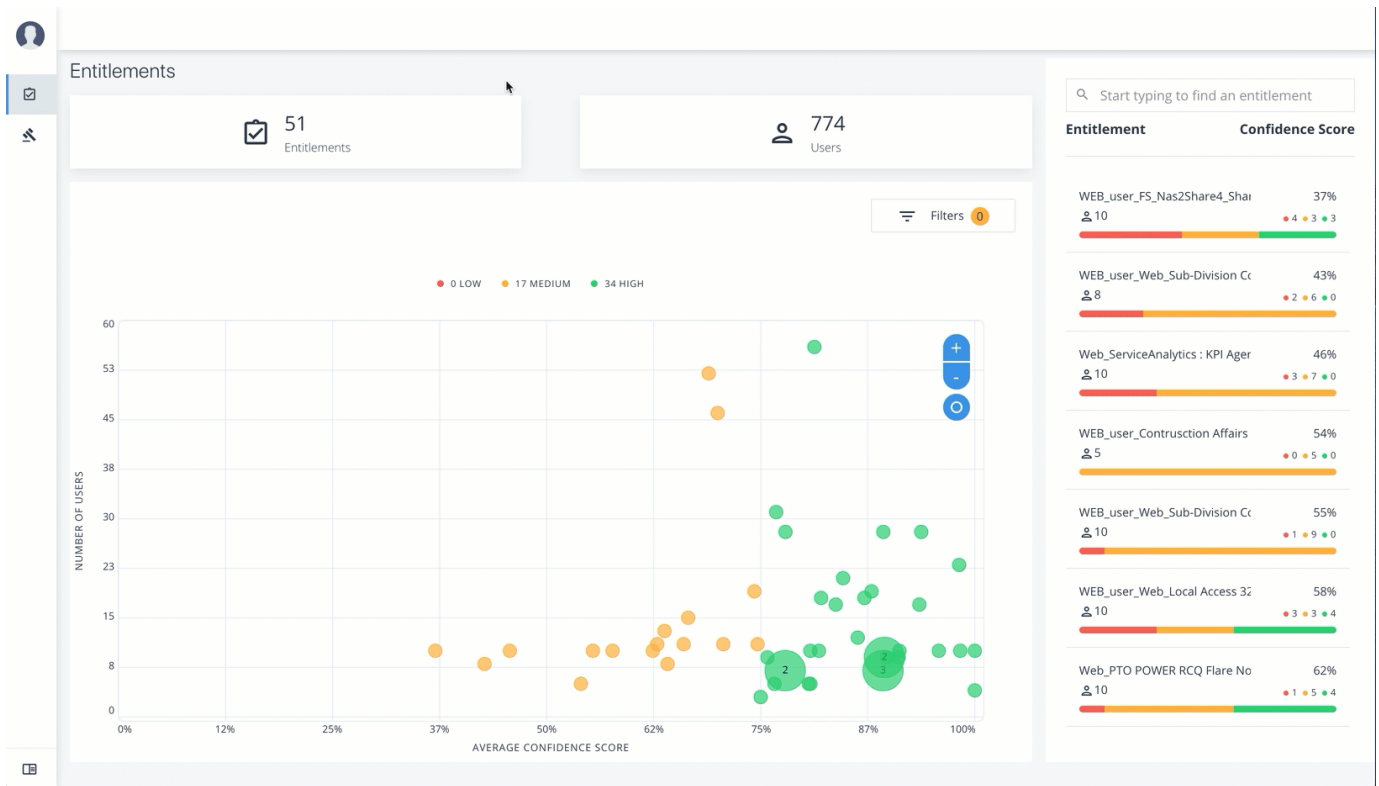
Auto-Certify and Auto-Request an Entitlement

Follow these steps to auto-certify and auto-request an entitlement:

1. Log in to the PingOne Autonomous Identity console as an Entitlement Owner.
2. On the graph, click a circle or click an entitlement in the right-hand list.
3. Review the details of the entitlement, especially the Driving Factors list.
4. Click the right arrow to view the users associated with the entitlement and confidence score. You can click a user to drill down to the Users Entitlements page.
5. Click the checkbox, and then Approve Justification to allow automated certifications and/or requests. Enter a reason for the approval and then click Submit Approval. You can cancel this auto certify or auto request transaction at any time.

Note

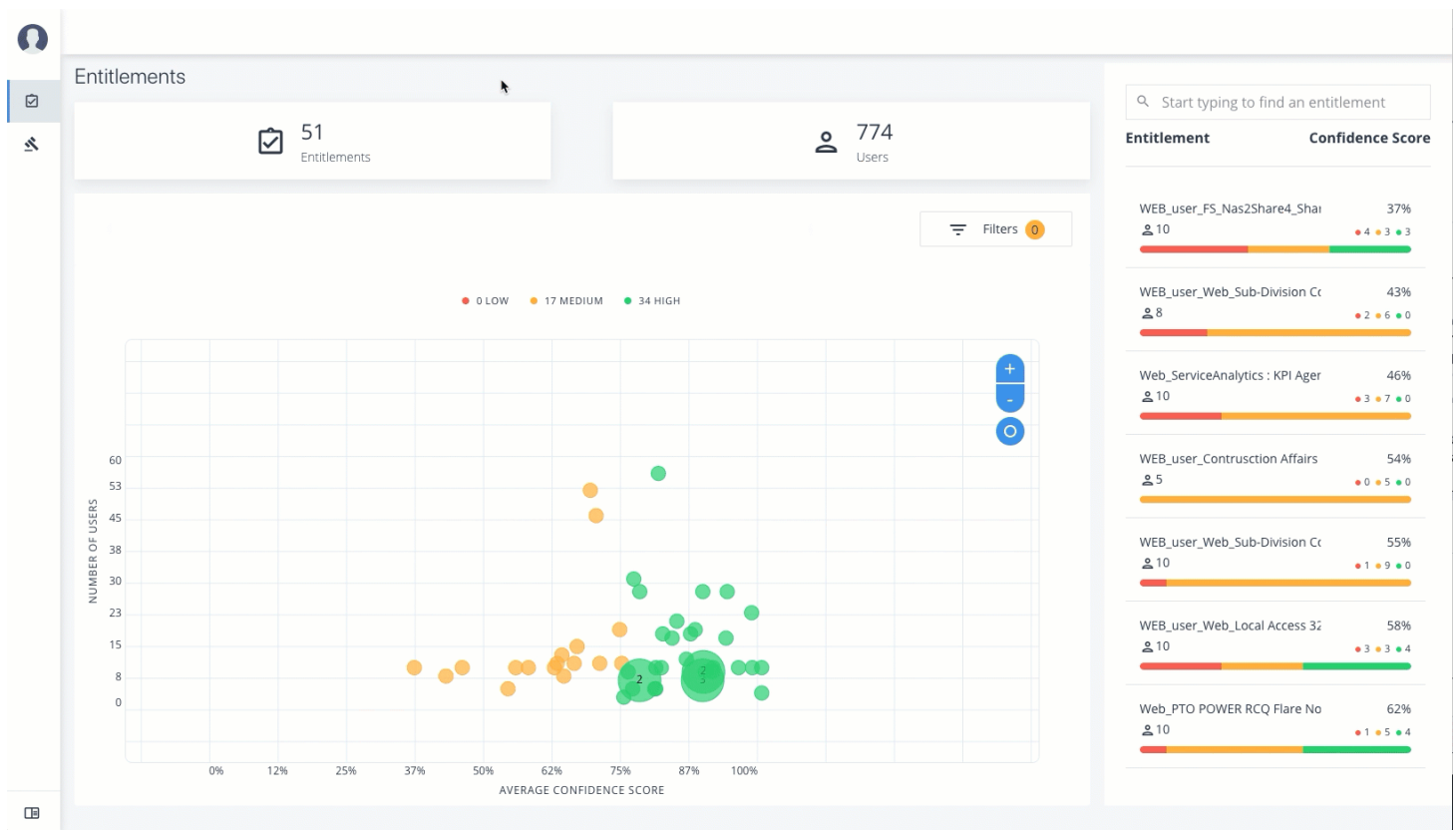
Auto Certify indicates that any user who has this justification is automatically approved for this entitlement.
Auto Request indicates that anyone who matches these set of criteria and may not already have access, automatically gets provisioned for this entitlement.



Apply Filters

Follow these steps to apply filters to your confidence score graphs:

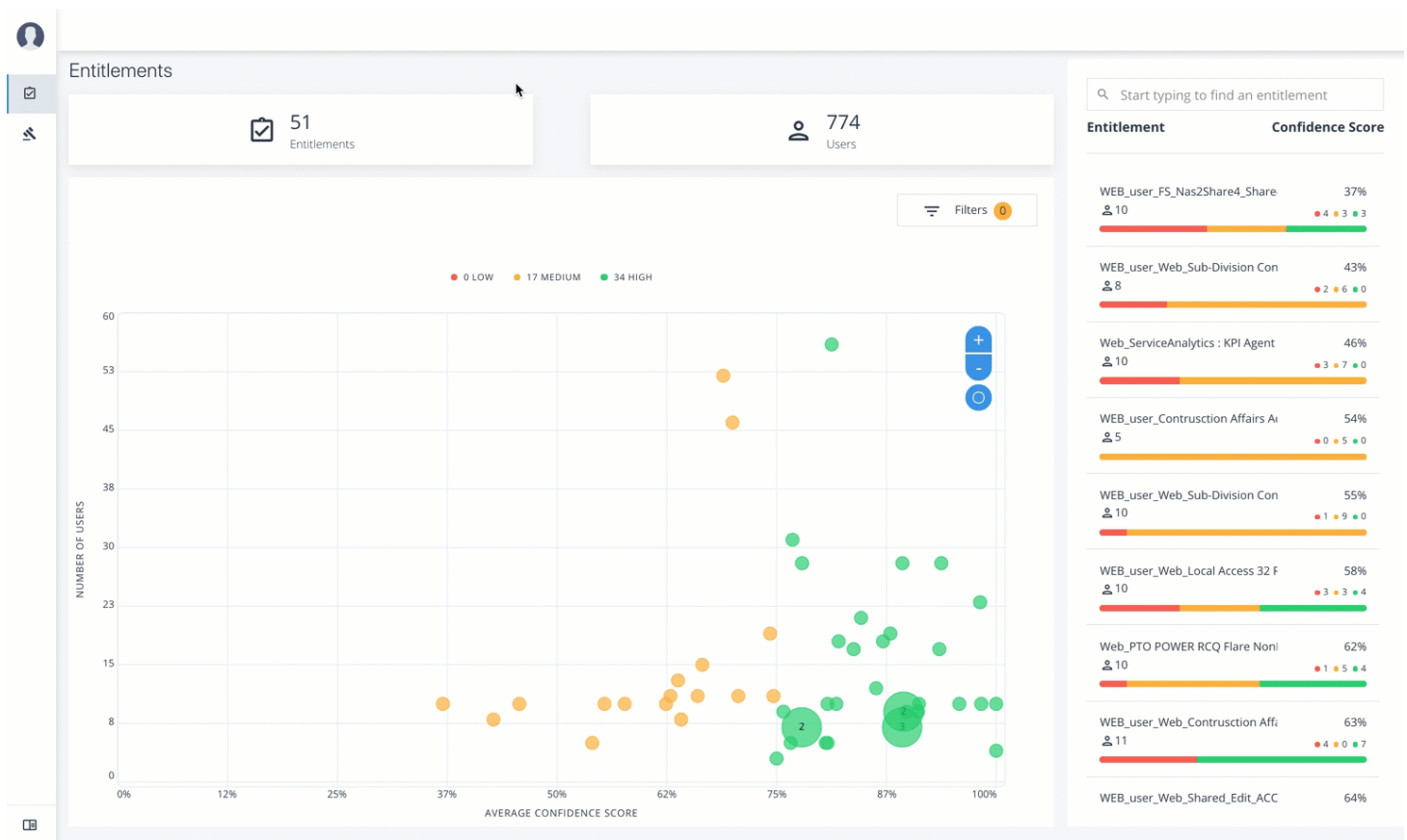
1. On the Entitlements page, view the average confidence score graph.
2. On the right, click **Filters**.
3. Do one or all of the following:
 - Click Remove high scores from Averages.
 - Click an application to filter the results.
 - Click Add Filters to further filter on a user attribute.



Approve or Revoke Access to an Assigned Entitlement

Follow these steps to investigate a confidence score and approve or revoke access to an entitlement assigned to a specific user:

1. Log in to the PingOne Autonomous Identity console.
2. On the Entitlements page, click an entitlement to investigate on the list on the right. You can also type a specific entitlement in the Search box.
3. Click the down arrow under Driving Factor to review the key attributes that leads to the average confidence score.
4. Under Justification, click the right arrow to review the users who have the assigned attribute. Click a user to drill down to the User Entitlements page.
5. On the User Entitlements page, click one or more entitlements, and then click Actions to approve or revoke the entitlement or group of entitlements. You can select more than one entitlement for a bulk approve, or you can only revoke one entitlement at a time.



Approve Rule Justifications

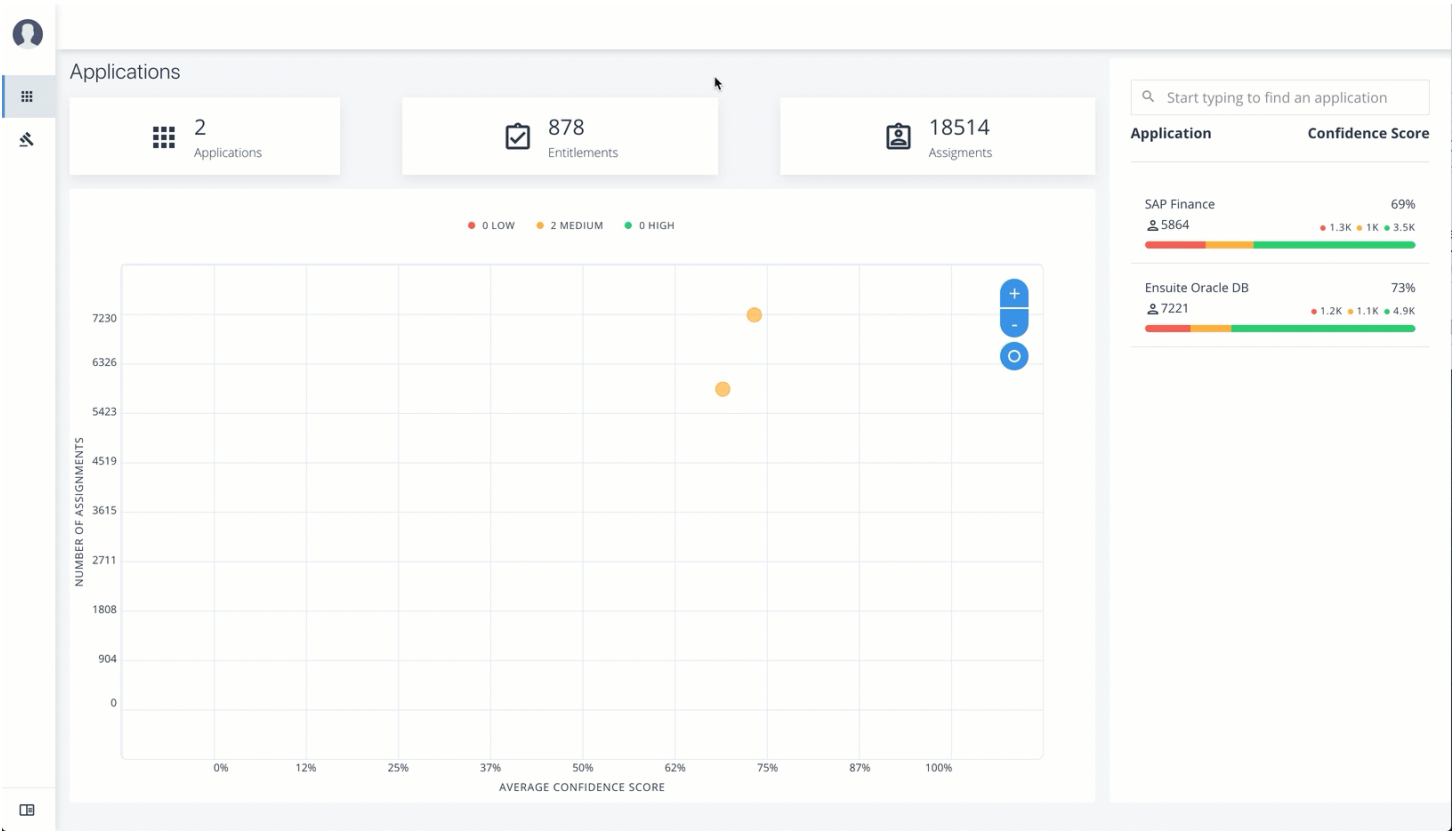
Follow these steps to apply rule justifications for an entitlement:

1. Log in to the PingOne Autonomous Identity console.
2. Click Rules.
3. On the Rules page, select an entitlement to view, and then click the down arrow to see the driving factors for the entitlement.
4. Under Identity, change to see another user's attributes and driving factors. If you want to see the user's entitlements page, click View <User>.
5. After researching the entitlement, click Approve. Click Auto Certify or Auto Request or both, and enter a reason for the approval. Click Submit Approval when ready.

You can also select all or multiple entitlements to do a bulk approve. PingOne Autonomous Identity only allows a single revoke action at a time.

Note

Auto Certify indicates that any user who has this justification is automatically approved for this entitlement. *Auto Request* indicates that anyone who matches these set of criteria and may not already have access, automatically gets provisioned for this entitlement.



Autonomous Identity API

This chapter is targeted to developers who want to access PingOne Autonomous Identity using the REST Application Programming Interface (API).

PingOne® Autonomous Identity is an entitlements and roles analytics system that lets you fully manage your company's access to your data.

An entitlement refers to the rights or privileges assigned to a user or thing for access to specific resources. A company can have millions of entitlements without a clear picture of what they are, what they do, and who they are assigned to. PingOne Autonomous Identity solves this problem by using advanced artificial intelligence (AI) and automation technology to determine the full entitlements landscape for your company. The system also detects potential risks arising from incorrect or over-provisioned entitlements that lead to policy violations. PingOne Autonomous Identity eliminates the manual re-certification of entitlements and provides a centralized, transparent, and contextual view of all access points within your company.



About the API

Learn about the PingOne Autonomous Identity API.



Obtain the API key

Learn how to get an API key.



API service

Access the API Service endpoints.



Authentication

Access the Authentication endpoints.



SSO

Access the SSO endpoints.



Config

Access the Config endpoints.



Report

Access the Report API.



Company view

Access the Company View API.



User details

Access the user details endpoints.



Access control

Access the Access Control endpoints.



Applications

Access the applications endpoints.



Entitlements

Access the entitlements endpoints.



Assignments

Access the assignments endpoint.



Rules

Access the rules endpoints.



Filters

Access the filters endpoints.



Roles

Access the roles endpoints.



Ingest

Access the ingest endpoints.



Jobs

Access the jobs endpoints.

About the PingOne Autonomous Identity API

PingOne Autonomous Identity provides a RESTful application programming interface (API) that lets you use HTTP request methods (GET, PUT, and POST) to interact with the system and its components. The API lets a developer make requests to send or receive data to an PingOne Autonomous Identity endpoint, a point where the API communicates with the system. The data that is sent or returned is in JavaScript Object Notation (JSON) format.



Important

With the release of version 2021.8.4, PingOne Autonomous Identity no longer provides a Swagger client that you can access on the console. The Swagger UI was removed to tighten security within PingOne Autonomous Identity. However, you can download the PingOne Autonomous Identity API and import it into Postman.



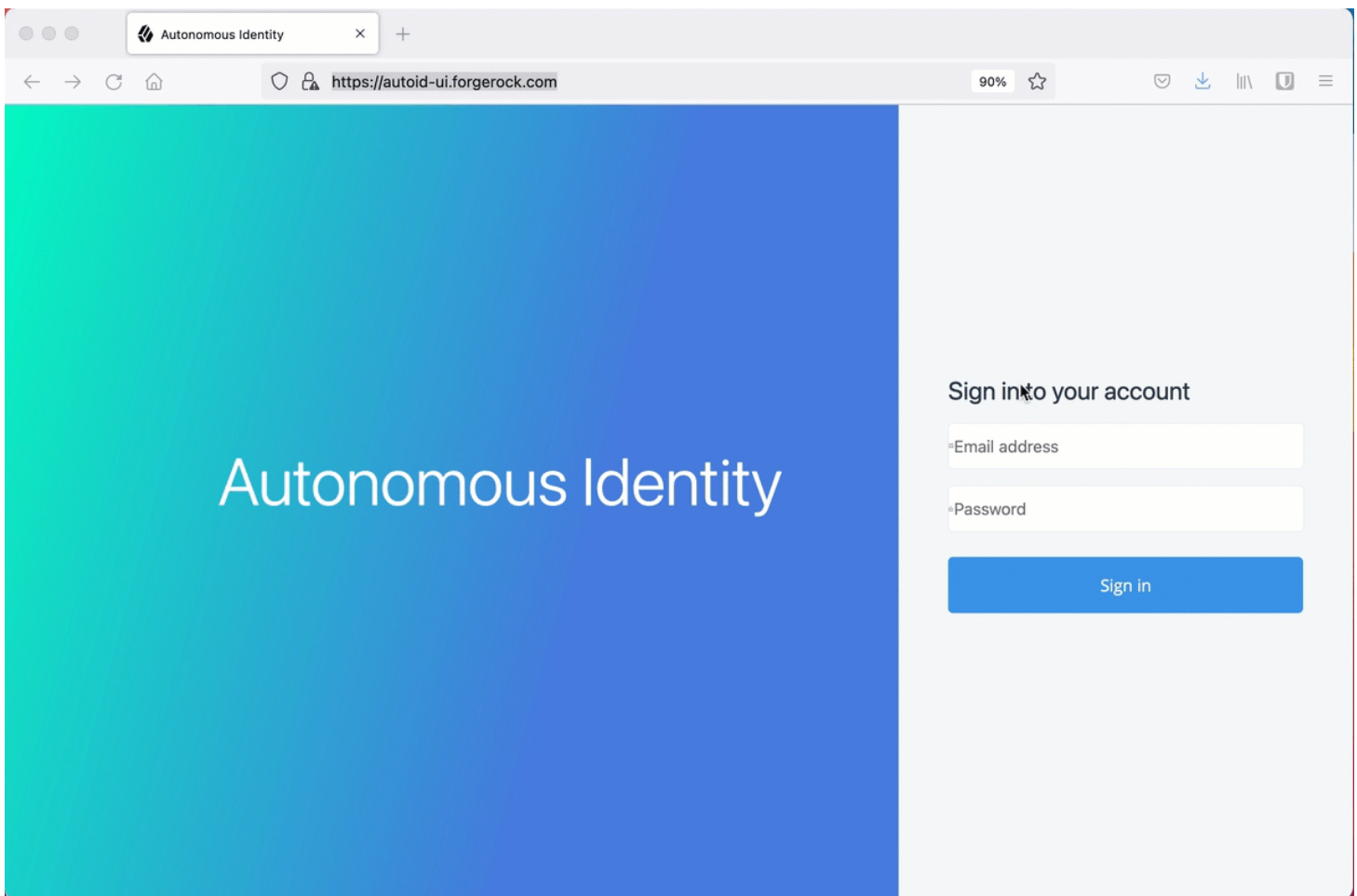
Warning

You cannot import the PingOne Autonomous Identity API into Swagger as there is an existing CORS issue that breaks functionality.

Using Postman

Download the PingOne Autonomous Identity API

1. On an upgraded PingOne Autonomous Identity instance, open a browser, and log in using your account at <https://autoid-ui.forgerock.com/>.
2. Point your browser to <https://autoid-ui.forgerock.com/api/swagger>. An Opening Swagger dialog appears.
3. Save the file as `api.yml` or `conf.yml` to your local server or laptop.
4. Open Postman, and click **Import**. The file is imported into Postman.
5. Click **Zoran-API-Service**.



You now can access the PingOne Autonomous Identity API in Postman.

Generate an API key

PingOne Autonomous Identity gives an administrator the ability to generate API keys for those who want to access certain endpoints using REST. Administrators can create an API from the Self-Service page of the PingOne Autonomous Identity UI.

Note

As of this release, only the Ingest endpoints use the API key.

Important

To use an API token, you need both the tenant ID that the API key belongs to, and the API token itself.

Obtain the tenant ID

In PingOne Autonomous Identity 2021.8.0 and later, the tenant ID is set as an environment variable that you can easily access.

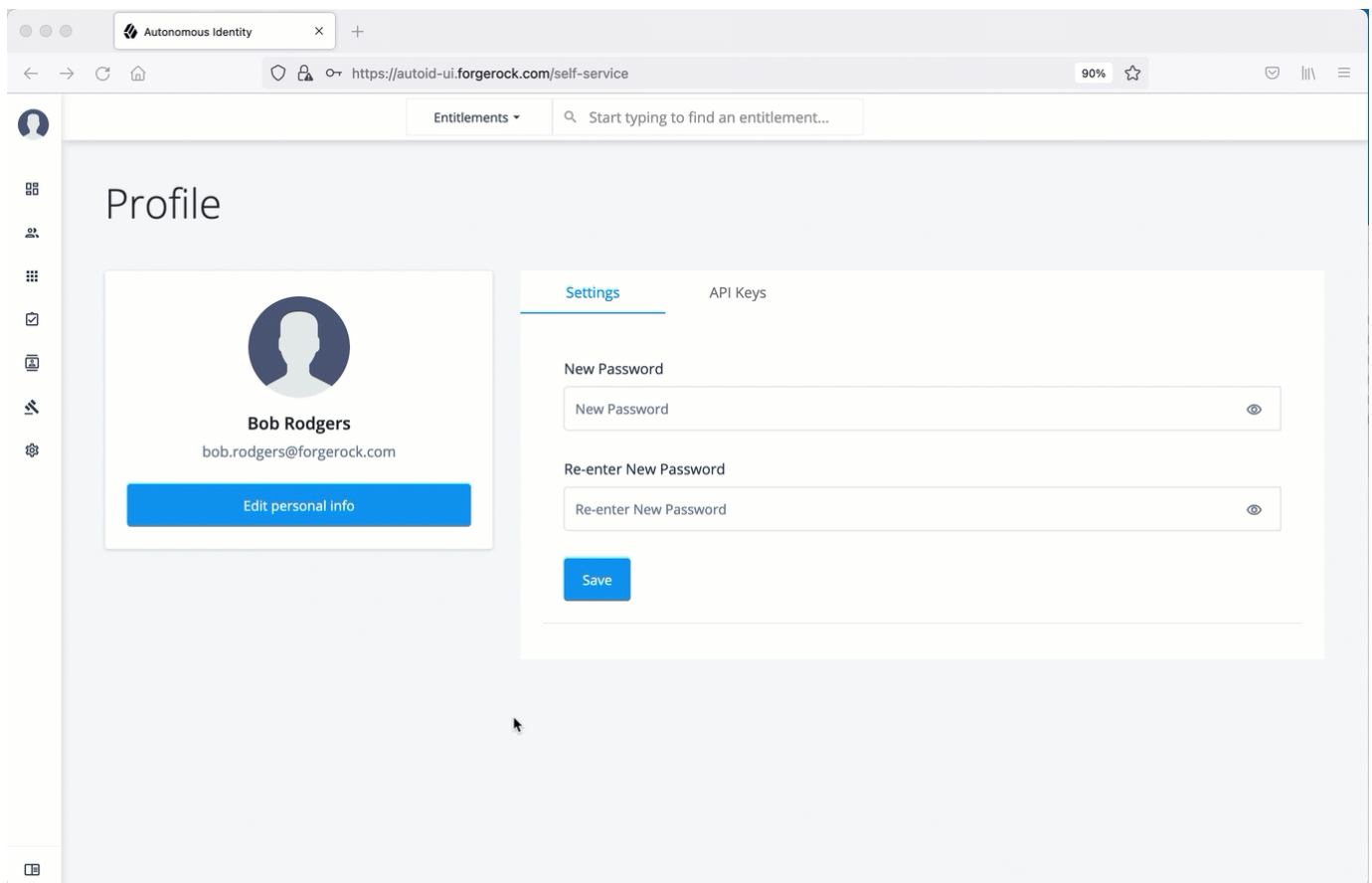
- On the target node, get the tenant ID.

```
$ env | grep TENANT_ID
TENANT_ID=8700f5cb-eaca-461e-8c2e-245a25f2399d
```

Create an API key using the UI

Administrators can create API keys on the Self-Service page of the PingOne Autonomous Identity UI.

1. On the PingOne Autonomous Identity UI, click the admin drop-down on the top-left of the page.
2. Click **Self Service**.
3. Click the **API Keys** tab.
4. Click **Generate API Key**.
5. Set the name, description, and expiration date for API key, and then click **Create**.
6. Make sure to make a copy of the key in the box as it cannot be retrieved once the dialog box is closed. The new API key appears in the list of keys on the API Keys page.

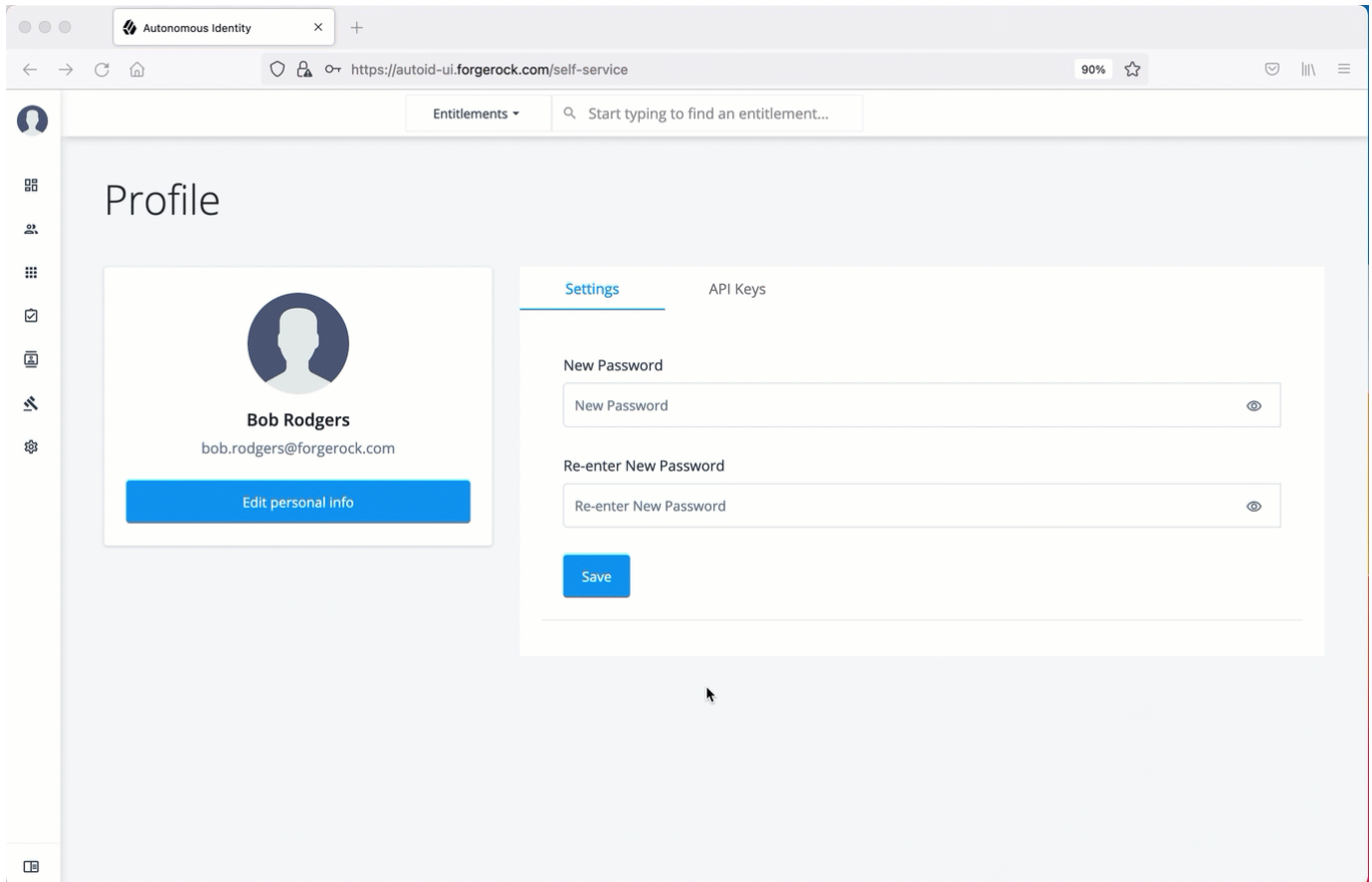


Deactivate API keys using the UI

Administrators can revoke or delete API keys. Use the following procedure to revoke an API key.

1. On the PingOne Autonomous Identity UI, click the admin drop-down on the top-left of the page.

2. Click **Self Service**.
3. Click the **API Keys** tab.
4. In the Search field, enter the API key.
5. In the list of API keys, click the three dots, and select **Revoke**. This action deletes the API key for use.



Create an API key using curl

Administrators can create API keys on the command line using curl commands.

1. Open a terminal, and create an authentication bearer token for an admin user:

```
curl -k -X POST \
  https://autoid-ui.forgerock.com/api/authentication/login \
  -H 'Content-Type: application/json' \
  -d '{
    "username": "bob.rogers@forgerock.com",
    "password": "Welcome123"
  }'
```

The response is:

```
{
  "user": {
    "dn": "cn=bob.rodgers@forgerock.com,ou=People,dc=zoran,dc=com",
    "controls": [],
    "displayName": "Bob Rodgers",
    "gidNumber": "999",
    "uid": "bob.rodgers",
    "_groups": [
      "Zoran User",
      "Zoran Admin"
    ]
  },
  "token": "token_value"
}
```

2. Set the **TOKEN** environment variable:

```
export TOKEN=token_value
```

3. Generate a new API key:

```
curl -k -X POST \
  https://autoid-ui.forgerock.com/api/admin/createApiToken \
  -H "Authorization: Bearer $TOKEN" \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "Ingest Key",
    "description": "API key for ingestion endpoints",
    "expiration": "2022-01-02"
  }'
```

The response is:

```
{
  "token": "19412ace-1d99-44b2-88e0-16136fc5c77a"
}
```

API key examples

The following curl example illustrates how to use the API key to get a datasource ID for an ingestion job:

1. Obtain an API key from an administrator. See [Create API Keys](#).
2. Obtain the tenant ID using the environment variable.

```
$ env | grep TENANT_ID
TENANT_ID=8700f5cb-eaca-461e-8c2e-245a25f2399d
```

3. Query PingOne Autonomous Identity's Java API Service (JAS) to obtain a data source ID using the API Key (for example, '1b7789f0-6c2f-4afa-a84b-a65a28f5c1a1'):

```
curl 'https://autoid-ui.forgerock.com/jas/datasource/search' \
-H 'authority: autoid-ui.forgerock.com' \
-H 'sec-ch-ua: "Chromium";v="92", " Not A;Brand";v="99", "Google Chrome";v="92"' \
-H 'accept: application/json, text/plain, /' \
-H 'x-tenant-id: 8700f5cb-eaca-461e-8c2e-245a25f2399d' \
-H 'authorization: 1b7789f0-6c2f-4afa-a84b-a65a28f5c1a1' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.159 Safari/537.36' \
-H 'content-type: application/json' \
-H 'origin: https://autoid-ui.forgerock.com' \
-H 'sec-fetch-site: same-origin' \
-H 'sec-fetch-mode: cors' \
-H 'sec-fetch-dest: empty' \
-H 'referrer: https://autoid-ui.forgerock.com/data-sources' \
-H 'accept-language: en-US,en;q=0.9' \
--data-raw '{
  "query": {
    "sort": [
      {
        "datasource_id.keyword": {
          "order": "desc"
        }
      }
    ],
    "size": 10,
    "track_total_hits": true,
    "query": {
      "match_all": {}
    }
  }
}' \
--compressed \
--insecure
```

The response includes the datasource ID:

```

{
  "took": 8,
  "timed_out": false,
  "_shards": {
    "total": 3,
    "successful": 3,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 1,
      "relation": "eq"
    },
    "max_score": null,
    "hits": [
      {
        "_index": "autonomous-iam_common_datasources_latest",
        "_type": "_doc",
        "_id":
"259b80c7693e92c4c29bd64deac4cd99826d427027645c9413afdb3f083b891d8d34cefaebd5fcf098c066dc1a4da2879d8732d59bfd2",
        "_score": null,
        "_source": {
          "datasource_id": "2d7a6a76-469c-4035-b312-fb1daf104e98",
          "name": "Showcase-CSV-DS",
          "sync_type": "full",
          "icon": "apps",
          "isActive": true,
          "entityTypes": {
            "/autoid/system/datasources/2d7a6a76-469c-4035-b312-fb1daf104e98/applications": {
              "uri": {
                "file": "file:/data/input/applications.csv"
              }
            },
            "/autoid/system/datasources/2d7a6a76-469c-4035-b312-fb1daf104e98/assignments": {
              "uri": {
                "file": "file:/data/input/assignments.csv"
              }
            },
            "/autoid/system/datasources/2d7a6a76-469c-4035-b312-fb1daf104e98/entitlements": {
              "uri": {
                "file": "file:/data/input/entitlements.csv"
              }
            },
            "/autoid/system/datasources/2d7a6a76-469c-4035-b312-fb1daf104e98/identities": {
              "uri": {
                "file": "file:/data/input/identities.csv"
              }
            }
          },
          "connectionSettings": {
            "csv": {}
          },
          "metadata": {

```

```

        "contextId": "scripts",
        "entityType": "/common/datasources",
        "primaryKey": "2d7a6a76-469c-4035-b312-fb1daf104e98",
        "entityPath": "/common/datasources/2d7a6a76-469c-4035-b312-fb1daf104e98",
        "entityDefinition": "datasources",
        "namespace": "/common",
        "branch": "actual",
        "created": "2021-08-25T03:53:33.634Z",
        "tenantId": "autonomous-iam"
    }
},
"sort": [
    "2d7a6a76-469c-4035-b312-fb1daf104e98"
]
}
]
}
}

```

4. Make sure your client that accesses the JAS configuration has something similar to the following:

```

public static final String CONFIG_JAS_API_TOKEN = "JAS_API_KEY";
public static final String CONFIG_JAS_TENANT_ID = "TENANT_ID";

```

API service

The following are PingOne Autonomous Identity API Service endpoints:

GET /health-check

GET /health-check

Check that the PingOne Autonomous Identity API service is running. Get uptime statistics. [All]

Endpoint

/health-check

Headers

Content-Type **application/json**

Body

Example Request

```
curl --request GET "<instance-IP>/health-check" \  
--header "Content-Type: application/json"
```

Example Response

```
{  
  "status": "zorán-api: OK",  
  "uptime": 5412.465875997,  
  "uptimeFormatted": "1:30:12"  
}
```

GET /version

GET /version

Get the version number of this service. [All]

Endpoint

/version

Headers

Content-Type	application/json
---------------------	-------------------------

Body

Example Request

```
curl --request GET "https://autoid-api.forgerock.com/version" \  
--header "Content-Type: application/json"
```

Example Response

```
{  
  "version": "1.0",  
}
```

Authentication

The following are PingOne Autonomous Identity authentication endpoints:

POST /api/authentication/login

POST /api/authentication/login

Log in to the system. The endpoint accepts the `username` and `password` in the body of the request. The token provided has an expiry date that can be obtained by decoding the returned JWT and using the `exp` data inside the token. [All]

Endpoint

```
/api/authentication/login
```

Headers

Content-Type	application/json
--------------	------------------

Body

```
{
  "username": "admin@test.com",
  "password": "test"
}
```

Example Request

```
curl --location --request POST 'https://autoid-api.forgerock.com/api/authentication/login' \
--header 'Content-Type: application/json' \
--data-raw '{
  "username": "admin@test.com",
  "password": "test"
}'
```

Example Response

```
{
  "user": {
    "dn": "cn=test.user@test.com,dc=example,dc=org",
    "controls": [],
    "gidNumber": "7777",
    "uid": "test.user",
    "displayName": "Test User",
    "_groups": [
      "Admin"
    ]
  },
  "token": "123456"
}
```

GET /api/authentication/verify

GET /api/authentication/verify

Verify the authenticity of a bearer token.

Endpoint

```
/api/authentication/verify
```

Authorization

Token	<token>
-------	---------

Headers

Content-Type	application/json
--------------	------------------

Body

```
..
```

Example Request

```
curl --location --request GET 'https://autoid-api.forgerock.com/api/authentication/verify' \
--header 'Content-Type: application/json'
```

Example Response

```
{
  "user": {
    "controls": [],
    "displayName": "Bob Rodgers",
    "email": "bob.rodgers@forgerock.com",
    "dn": "cn=bob.rodgers@forgerock.com,ou=People,dc=zoran,dc=com",
    "gidNumber": "999",
    "uid": "bob.rodgers",
    "_groups": [
      "Zoran Admin"
    ],
    "iat": 1628893019,
    "exp": 1628936219,
    "aud": "http://my.service",
    "sub": "6711197"
  }
}
```

POST /api/authentication/renewToken

POST /api/authentication/renewToken

Renew a token for the system. The endpoint accepts the JWT in the header `Authorization: Bearer JWT`. The expiry time of the token is reset and return in the new token. [All]

Endpoint

/api/authentication/renewToken

Authorization

Token **<token>**

Headers

Content-Type **application/json**

Body

..

Example Request

```
curl --location --request POST 'https://autoid-api.forgerock.com/api/authentication/renewToken' \
--header 'Content-Type: application/json' \
--data-raw ''
```

Example Response

```
{
  "user": {
    "dn": "cn=test.user@test.com,dc=example,dc=org",
    "controls": [],
    "gidNumber": "7777",
    "uid": "test.user",
    "displayName": "Test User",
    "_groups": [
      "Admin"
    ]
  },
  "token": "123456"
}
```

GET /api/authentication/actions

GET /api/authentication/actions

Retrieve the permitted actions of the currently authenticated user. [All]

Endpoint

```
/api/authentication/action
```

Headers

Content-Type	application/json
---------------------	-------------------------

Example Request

```
curl --location --request GET 'https://autoid-api.forgerock.com/api/authentication/actions' \
--header 'Content-Type: application/json'
```

Example Response

```
{
  "userActions": [
    "*"
  ],
  "roleTitle": "Unknown",
  "homepage": "company"
}
```

SSO

The following are PingOne Autonomous Identity SSO endpoints:

GET /api/sso/start

GET /api/sso/start

Endpoint to initiate SSO. [All]

Endpoint

```
/api/sso/start
```

Authorization

Token <token>

Headers

Content-Type /

Example Request

```
curl --request GET "https://autoid-api.forgerock.com/api/sso/start" \
--header "Content-Type: /"
```

GET /api/sso/finish

GET /api/sso/finish

Endpoint to finish SSO. [All]

Endpoint

/api/sso/finish

Authorization

Token <token>

Headers

Content-Type /

Example Request

```
curl --request GET "https://autoid-api.forgerock.com/api/sso/finish" \
--header "Content-Type: /"
```

GET /api/slo/logout

GET /api/slo/logout

Endpoint to log out of SSO.

Endpoint

```
/api/slo/logout
```

Authorization

```
Token      <token>
```

Headers

```
Content-Type  application/json
```

Example Request

```
curl --request GET "https://autoid-api.forgerock.com/api/slo/logout" \
--header "Content-Type: application/json"
```

Config

The following are PingOne Autonomous Identity config endpoint:

GET /api/config

GET /api/config

Get the configuration. This endpoint is mainly used by the PingOne Autonomous Identity UI microservice to get values stored in Consul. [All]

Endpoint

```
/api/config
```

Headers

```
Content-Type  application/json
```

Example Request

```
curl --request GET "https://autoid-api.forgerock.com/api/config" \
--header "Content-Type: application/json"
```

Example Response

```
{
  "thresholds": {
    "top": 1.01,
    "high": 0.75,
    "medium": 0.35,
    "low": 0,
    "autoAccess": 0.5
  },
  "volumeThresholds": {
    "high": 90,
    "low": 20
  },
  "mostAssignedStats": {
    "count": 100
  },
  "highVolumeStats": {
    "high": {
      "minScore": 0.9,
      "minUsersCount": 100
    },
    "low": {
      "maxScore": 0.2,
      "minUsersCount": 100
    }
  },
  "authorizers": {
    "ldap": true,
    "oidc": false
  }
}
```

GET /api/admin/reloadUIConfig

GET /api/admin/reloadUIConfig

Reload justification and filterable attributes configuration from JAS. [User, Supervisor, Ent Owner, App Owner, Admin]

Endpoint

/api/admin/reloadUIConfig

Headers

Content-Type /

Example Request

```
curl --request GET "https://autoid-api.forgerock.com/api/admin/reloadUIConfig" \
-H "accept: /"
```

POST /api/admin/updateUser

POST /api/admin/updateUser

Update credentials for a user.

Endpoint

/api/admin/updateUser

Authorization

<Bearer Token JWT-value>

Body

```
{
  email: "john.doe@forgerock.com",
  password: "password",
  groups: ["Zoran Supervisor", "Zoran Role Engineer"]
}
```

Example Request

```
curl --location --request POST 'https://autoid-api.forgerock.com/api/admin/updateUser' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer <token>' \
--data-raw '{
  "email": "john.doe@forgerock.com",
  "password": "password",
  "groups": ["Zoran Supervisor", "Zoran Role Engineer"]
}'
```

Example Response

```
{ message: 'success' }
```

POST /api/admin/updateSelf

POST /api/admin/updateSelf

Update credentials for a user.

Endpoint

```
/api/admin/updateSelf
```

Authorization

```
<Bearer Token JWT-value>
```

Body

```
{
  email: "john.doe@forgerock.com",
  password: "password",
  groups: ["Zoran Supervisor", "Zoran Role Engineer"]
}
```

Example Request

```
curl --location --request POST 'https://autoid-api.forgerock.com/api/admin/updateSelf' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer <token>' \
--data-raw '{
  "email": "john.doe@forgerock.com",
  "password": "password",
  "groups": ["Zoran Supervisor", "Zoran Role Engineer"]
}'
```

Example Response

```
{ message: 'success' }
```

POST /api/admin/disableUser

POST /api/admin/disableUser

Disable user account.

Endpoint

```
/api/admin/disableuser
```

Authorization

```
<Bearer Token JWT-value>
```

Body

```
{
  email: "john.doe@forgerock.com"
}
```

Example Request

```
curl --location --request POST 'https://autoid-api.forgerock.com/api/admin/disableUser' \
--header 'Content-type: application/json' \
--header 'Authorization: Bearer <token>' \
--data-raw '{
  "email": "john.doe@forgerock.com"
}'
```

Example Response

```
{ message: 'success' }
```

POST /api/admin/enableUser

POST /api/admin/enableUser

Enable a user account.

Endpoint

```
/api/admin/enableUser
```

Authorization

```
<Bearer Token JWT-value>
```

Body

```
{
  email: "john.doe@forgerock.com"
}
```

Example Request

```
curl --location --request POST "https://autoid-api.forgerock.com/api/admin/enableUser" \
--header "Content-Type: application/json" \
--header "Authorization: Bearer <token>" \
--data-raw '{
  "email": "john.doe@forgerock.com"
}'
```

Example Response

```
{ message: 'success' }
```

POST /api/admin/createUser

POST /api/admin/createUser

Create credentials for a user

Endpoint

/api/admin/createUser

Authorization

<Bearer Token JWT-value>

Body

```
{
  dn: "cn=john.doe@forgerock.com,ou=People,dc=zoran,dc=com",gidNumber: "321",
  email: "john.doe@forgerock.com",
  password: "password",
  controls: [],
  displayName: "John Doe",
  uid: "john.doe",
  groups: ["Zoran Admin"]
}
```

Example Request

```
curl --request POST "https://autoid-api.forgerock.com/api/admin/createUser" \
--header "Content-Type: application/json" \
--header "Authorization: Bearer <token>" \
--data-raw '{
  dn: "cn=john.doe@forgerock.com,ou=People,dc=zoran,dc=com",gidNumber: "321",
  email: "john.doe@forgerock.com",
  password: "password",
  controls: [],
  displayName: "John Doe",
  uid: "john.doe",
  groups: ["Zoran Admin"]
}'
```

Example Response

```
{ message: 'success' }
```

POST /api/admin/createApiToken

POST /api/admin/createApiToken

Create API credentials (token) for a user.

Endpoint

/api/admin/createApiToken

Authorization

<Bearer Token JWT-value>

Body

```
{
  name: "John Doe",
  description: "description",
  expiration: "2021-08-12T12:00:00.000Z"
}
```

Example Request

```
curl --request POST "https://autoid-api.forgerock.com/api/admin/createApiToken" \
--header "Content-Type: application/json" \
--header "Authorization: Bearer <token>" \
--data-raw '{
  name: "John Doe",
  description: "description",
  expiration: "2021-08-12T12:00:00.000Z"
}'
```

Example Response

```
{ token: uuid }
```

POST /api/admin/revokeApiToken

POST /api/admin/revokeApiToken

Revoke API credentials for a user.

Endpoint

/api/admin/revokeApiToken

Authorization

<Bearer Token JWT-value>

Body

```
{
  token: "11b57769-d436-4357-bc1c-0e0b9c6a49b6"
}
```

Example Request

```
curl --location --request POST "https://autoid-api.forgerock.com/api/admin/revokeApiToken" \
--header "Content-Type: application/json" \
--header "Authorization: Bearer <token>" \
--data-raw '{
  token: "11b57769-d436-4357-bc1c-0e0b9c6a49b6"
}'
```

Example Response

```
{ message: 'User token disabled' }
```

GET /api/admin/verifyApiToken/{token}

GET /api/admin/verifyApiToken/{token}

Verify that an API token is valid

Endpoint

```
/api/admin/verifyApiToken
```

Authorization

```
<Bearer Token JWT-value>
```

Example Request

```
curl --request GET "https://autoid-api.forgerock.com/api/admin/verifyApiToken/da0b5228-1e11-4278-ad1c-f0938fccdf82" \
--header "Content-Type: application/json" \
--header "Authorization: Bearer <token>"
```

Example Response

```
{
  "name": "John Doe",
  "description": "description",
  "expiration": "2021-08-17T12:00:00.000Z",
  "is_valid": true,
  "creator": "bob.rodgers@forgerock.com",
  "modifier": "bob.rodgers@forgerock.com"
}
```

GET /api/admin/getTokens

GET /api/admin/getTokens

Get a list of tokens.

Endpoint

```
/api/admin/getTokens
```

Authorization

<Bearer Token JWT-value>

Body

```
{
    maxResults: 6,
    offset: 5
}
```

Example Request

```
curl --request GET "https://autoid-api.forgerock.com/api/admin/getTokens" \
--header "Content-Type: application/json" \
--header "Authorization: Bearer <token>" \
--data-raw '{
    maxResults: 6,
    offset: 5
}'
```

Example Response

```
[{
  "token": "da0b5228-1e11-4278-ad1c-f0938fccdf82",
  "name": "John Doe",
  "description": "description",
  "expiration": "2021-08-17T12:00:00.000Z",
  "is_valid": true,
  "creator": "bob.rodgers@forgerock.com",
  "modifier": "bob.rodgers@forgerock.com"
}]
```

GET /api/admin/getUsers

GET /api/admin/getUsers

Get a list of users.

Endpoint

/api/admin/getUsers

Authorization

<Bearer Token JWT-value>

Body

```
{
  maxResults: 6,
  offset: 5,
  sortBy: uid | displayName | gidNumber
}
```

Example Request

```
curl --location --request GET "https://autoid-api.forgerock.com/api/admin/getUsers" \
--header "Content-Type: application/json" \
--header "Authorization: Bearer <token>" \
--data-raw '{
  maxResults: 6,
  offset: 5,
  sortBy: uid
}'
```

Example Response

```
[{
  "controls": [],
  "displayName": "David Elliott",
  "email": "david.elliott@forgerock.com",
  "dn": "cn=david.elliott@forgerock.com,ou=People,dc=zoran,dc=com",
  "gidNumber": "809",
  "uid": "david.elliott",
  "_groups": [
    "Zoran Entitlement Owner"
  ],
  "enabled": true
}]
```

Report

PingOne Autonomous Identity captures information in its log files that are useful when troubleshooting problems. You can access the reports using REST calls to the Report API endpoint.

POST /api/report

POST /api/report

Get reporting data. [All]

Endpoint

/api/report

Authorization

<Bearer Token JWT-value>

Headers

Content-Type **application/json**

Params

fields

Body

```
{
  "fields": [
    "id",
    "type",
    "batch_id",
    "original",
    "update"
  ],
  "reportType": "EventBasedCertification"
}
```

Example Request

```
curl --request POST "https://autoid-api.forgerock.com/api/report" \
--header "Content-Type: application/json" \
--header 'Authorization: Bearer <token>' \
--data-raw '{
  "fields": [
    "id",
    "type",
    "batch_id",
    "original",
    "update"
  ],
  "reportType": "EventBasedCertification"
}'
```

Company View

The following are PingOne Autonomous Identity company view endpoints:

GET /api/companyview

GET /api/companyview

Get the data for company overview dashboard data. [Executive, Admin]

Endpoint

```
/api/companyview
```

Authorization

```
<Bearer Token JWT-value>
```

Example Request

```
curl --request GET "https://autoid-api.forgerock.com/api/companyview" \
--header "Content-Type: application/json"
```

Example Response

```

{
  "companyView": {
    "employeeTypes": [
      {
        "type": "Employee",
        "high": 723,
        "low": 27,
        "medium": 1796,
        "null_conf": 0,
        "total": 2546
      },
      {
        "type": "Non-Employee",
        "high": 867,
        "low": 14,
        "medium": 1768,
        "null_conf": 0,
        "total": 2649
      }
    ],
    "employees_wo_manager": 0,
    "employees_w_manager": 5200,
    "entitlements_without_roleowners": 0,
    "entitlements_with_roleowners": 2456,
    "total_employees": 5200,
    "coverage": {
      "total": 2456,
      "covered": 2456,
      "not_covered": 0
    },
    "entitlementsDistribution": {
      "no_users": 0,
      "one_user": 0,
      "zero_to_five_users": 1,
      "five_to_ten_users": 1064,
      "ten_to_hundred_users": 1549,
      "hundred_to_onek_user": 35,
      "onek_to_tenk_users": 0,
      "tenk_users": 0,
      "hundredk_users": 0
    }
  }
}

```

GET /api/companyview/allEntitlementsAvgGroups

GET /api/companyview/allEntitlementsAvgGroups

Get the average confidence score list for the company view chart. [Executive, Admin]

Endpoint

```
/api/companyview/allEntitlementAvgGroups
```

Authorization

```
<Bearer Token JWT-value>
```

Example Request

```
curl --request GET "https://autoid-api.forgerock.com/api/companyview/allEntitlementAvgGroups" \
--header "Content-Type: application/json"
```

Example Response

```
{
  "entitlementList": [
    {
      "start": 0,
      "end": 0.05,
      "entitlementCount": 2
    },
    {
      "start": 0.06,
      "end": 0.1,
      "entitlementCount": 14
    }
  ]
}
```

GET /api/companyview/mostCriticalEntitlements

GET /api/companyview/mostCriticalEntitlements

Get the most critical entitlements list. [Executive, Admin]

Endpoint

```
/api/companyview/mostCriticalEntitlements
```

Authorization

```
<Bearer Token JWT-value>
```

Example Request

```
curl --request GET "https://autoid-api.forgerock.com/api/companyview/mostCriticalEntitlements" \
--header "Content-Type: application/json"
```

Example Response

```
[
  {
    "org": "organization",
    "entt_id": "ent1",
    "avg_conf_score": 0.04,
    "entt_name": "Ent 1",
    "high": 0,
    "low": 1,
    "medium": 0,
    "seq": 0,
    "total_employees": 6
  },
  {
    "org": "organization",
    "entt_id": "ent2",
    "avg_conf_score": 0.04571,
    "entt_name": "Ent 2",
    "high": 0,
    "low": 1,
    "medium": 0,
    "seq": 1,
    "total_employees": 7
  }
]
```

GET /api/companyview/assignmentStats

GET /api/companyview/assignmentStats

Get the total assignments, low/high confidence, high volume and low/high confidence, most assigned [Executive, Admin]

Endpoint

```
/api/companyview/assignmentsStats
```

Authorization

```
<Bearer Token JWT-value>
```

Params

```
assignmentLimit 1
highVolumeHighMinScore 0.9
highVolumenHighMinUsersCount 100
highVolumenLowMaxScore 0.2
highVolumeLowMinUsersCount 100
```

Example Request

```
curl --request GET "https://autoid-api.forgerock.com/api/companyview/assignmentsStats?
assignmentsLimit=5" \
--header "Content-Type: application/json"
```

Example Response

```
{
  "total": 47670,
  "high": 13145,
  "low": 4992,
  "unscored": 4986,
  "mostAssigned": [
    {
      "count": 344,
      "entitlement": "ent1"
    }
  ],
  "mostAssignedCount": 35,
  "highVolume": {
    "high": 23,
    "low": 17
  }
}
```

GET /api/companyview/assignmentHistConfSummary/{year}/{month}

GET /api/companyview/assignmentHistConfSummary/{year}/{month}

Get the number of high, medium, and low confidence assignments for the past 12-month period ending in a given year and month. [Executive, admin]

Endpoint

```
/api/companyview/assignmentsHistConfSummary/2020/01
```

Authorization

```
<Bearer Token JWT-value>
```

Example Request

```
curl --request GET "https://autoid-api.forgerock.com/api/companyview/assignmentsHistConfSummary/2020/1" \
--header "Content-Type: application/json"
```

Example Response

```
[
  {
    "year": 0,
    "month": 0,
    "highConf": 0,
    "medConf": 0,
    "lowConf": 0,
    "total": 0
  }
]
```

User Details

The following are PingOne Autonomous Identity user details endpoints:

POST /api/userDetails

POST /api/userDetails

Obtain the employee details for Identities views. [User, Supervisor, Ent Owner, App Owner, Admin] NOTE: This is a POST due to the endpoint receiving a JSON body query.

Endpoint

/api/userDetails

Authorization

<Bearer Token JWT-value>

Headers

Content-Type **application/json**

Body

```
{
  "employeeId": "john.doe",
  "sortDir": "asc, desc",
  "lastEntitlementId": "Web_NAS_Share_Case Management_7HQ",
  "lastRecommendedEnttId": "string"
}
```

Example Request

```
curl --request POST "https://autoid-api.forgerock.com/api/userDetails" \
--header "Content-Type: application/json" \
--header "Authorization: Bearer <token>" \
--data-raw '{
  "employeeId": "john.doe",
  "sortDir": "asc, desc",
  "lastEntitlementId": "Web_NAS_Share_Case Management_7HQ",
  "lastRecommendedEnttId": "string"
}'
```

Example Response

```

{
  "recommendedEntt": {
    "predictions": [
      {
        "usr_key": "john.doe",
        "ent": "ent1",
        "conf": "0.88",
        "freq": "10.0",
        "frequnion": "9",
        "rule": [
          {
            "title": "Chief",
            "value": "Yes"
          },
          {
            "title": "Employee Type",
            "value": "Employee"
          }
        ]
      },
      {
        "entt": {
          "entitlement": "Ent 1",
          "app_id": "app1",
          "role": "role.owner",
          "app_name": "App 1",
          "entitlement_name": "Ent 1",
          "high_risk": "High",
          "roleOwnerDisplayName": "Role Owner",
          "requestorCanAccess": false
        }
      },
      {
        "usr_key": "john.doe",
        "ent": "ent2",
        "conf": "1.00",
        "freq": "4.0",
        "frequnion": "4",
        "rule": [
          {
            "title": "Job Code Name",
            "value": "Business Representative"
          },
          {
            "title": "Line of Business",
            "value": "Portfolio Management"
          },
          {
            "title": "Department",
            "value": "South"
          },
          {
            "title": "Employee Type",
            "value": "Employee"
          }
        ]
      }
    ]
  }
}

```

```

      "entt": {
        "entitlement": "ent2",
        "app_id": "app1",
        "role": "role.owner",
        "app_name": "App 1",
        "entitlement_name": "Ent 2",
        "high_risk": "High",
        "roleOwnerDisplayName": "Role Owner",
        "requestorCanAccess": true
      }
    ],
    "entitlementsCount": 14,
    "entitlementsRemainingCount": 9,
    "lastEntitlementId": "ent2"
  },
  "userEntt": [
    {
      "user": "john.doe",
      "entitlement": "ent3",
      "app_id": "app1",
      "app_name": "App 1",
      "entitlement_name": "Ent 3",
      "freq": "10.0",
      "frequion": "9",
      "high_risk": "High",
      "justification": [
        {
          "title": "Chief",
          "value": "Yes"
        },
        {
          "title": "Employee Type",
          "value": "Employee"
        }
      ]
    },
    {
      "score": 0.88,
      "user_name": "John Doe",
      "lastAccessed": "2020-01-01 00:00:00",
      "requestorCanAccess": false,
      "rawJustification": [
        "CHIEF_YES_NO_Yes",
        "USR_EMP_TYPE_Employee"
      ]
    }
  ],
  {
    "user": "john.doe",
    "entitlement": "ent4",
    "app_id": "app1",
    "app_name": "App 1",
    "entitlement_name": "Ent 4",
    "freq": "4.0",
    "frequion": "4",
    "high_risk": "High",
    "justification": [

```

```

    {
      "title": "Job Code Name",
      "value": "Business Representative"
    },
    {
      "title": "Line of Business",
      "value": "Portfolio Management"
    },
    {
      "title": "Department",
      "value": " South"
    },
    {
      "title": "Employee Type",
      "value": "Employee"
    }
  ],
  "score": 1,
  "user_name": "John Doe",
  "lastAccessed": "2020-01-01 00:00:00",
  "requestorCanAccess": false,
  "rawJustification": [
    "JOB_CODE_NAME_Business Representative",
    "LINE_OF_BUSINESS_Portfolio Management",
    "USR_DEPARTMENT_NAME_Customer Operations_ South",
    "USR_EMP_TYPE_Employee"
  ]
}
],
"user": {
  "displayName": "John Doe",
  "hrData": [
    {
      "title": "Job Code Name",
      "id": "JOB_CODE_NAME",
      "value": "Business Representative"
    },
    {
      "title": "Line of Business",
      "id": "LINE_OF_BUSINESS",
      "value": "Portfolio Management"
    },
    {
      "title": "Department",
      "id": "DEPARTMENT",
      "value": " South"
    },
    {
      "title": "Employee Type",
      "id": "EMPLOYEE_TYPE",
      "value": "Employee"
    }
  ]
}
]
```

```
  },
  "entitlementsCount": 2,
  "entitlementsRemainingCount": 9,
  "lastEntitlementId": "ent4"
}
```

GET /api/userDetails/decisions

GET /api/userDetails/decisions

Get the current entitlement decisions for the user [Supervisor, Ent Owner, App Owner, Admin].

Endpoint

/api/userDetails/decisions

Authorization

<Bearer Token JWT-value> OR <API-KEY>

Query Parameters

Parameter	Type	Description
user	string	User ID (required)
filter	object	Filter to add (single property shown below)

Filter Query Object Properties

Parameter	Type	Description
datasinkStatus	string	Datasink status filter ('ack' or 'nack')
timestampThresholds	timestampThresholds object	Timestamp threshold object (available properties below)

timestampThresholds Object Properties

Parameter	Type	Description
gt	string	Greater than timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ). Cannot be present with gte .
gte	string	Greater than or equal timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ). Cannot be present with gt .

Parameter	Type	Description
lt	string	Less than timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ). Cannot be present with lte .
lte	string	Less than or equal timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ). Cannot be present with le .

Example Request (DatasinkStatus Filter)

```
curl -k -X GET \
'datasinkStatus=ack' \
-H 'Authorization: Bearer <token value>' \ <or> -H 'X-API-KEY: <api key value>' \
-H 'Content-Type: application/json'
```

Example Response (DatasinkStatus Filter)

```
{
  "decisions": [
    {
      "user": "john.doe",
      "entitlement": "ent_1",
      "is_certified": false,
      "is_revoked": false,
      "is_processed": false,
      "is_archived": false,
      "author": "jane.smith",
      "author_name": "Jane Smith",
      "author_type": "Zoran Admin",
      "reason": null,
      "last_updated": "2022-01-11T19:48:17.195Z",
      "datasink_status": "ack",
      "usr_name": "John Doe",
      "ent_name": "Entitlement 1",
      "app_id": "Gateway",
      "app_name": "Gateway",
      "usr_manager_id": "john.smith",
      "conf": 0.75,
      "freq": 4,
      "freqUnion": 3
    }
  ]
}
```

POST /api/userDetails/decisions

POST /api/userDetails/decisions

Update entitlement decisions for users. [Supervisor, Ent Owner, App Owner, Admin]

Endpoint

/api/userDetails/decisions

Authorization

<Bearer Token JWT-value>

Request Body Parameters

Parameter	Type	Description
assignments	array of assignment objects	List of assignments affected by the decision (available properties listed below)(required)
is_certified	boolean	Certification decision
is_revoked	boolean	Revoke decision
is_requested	boolean	Decision is processed
reason	string	Reason for decision
datasink_status	string	Datasink status ('ack' or 'nack')

Assignments Object Properties

Parameter	Type	Description
user	string	User ID (required)
entitlements	string array	List of entitlement ID's (required)

Body

```
{
  "assignments": [
    {
      "user": "string",
      "entitlements": [
        "string"
      ]
    }
  ],
  "is_certified": true,
  "is_revoked": true,
  "is_requested": true,
  "is_processed": true,
  "reason": "string",
  "datasink_status": "nack"
}
```

Example Request

```
curl -k -X POST \
  "https://autoid-api.forgerock.com/api/userDetails/decisions" \
  -H 'Authorization: Bearer <token-value>' \
  -H 'accept: /' -H 'Content-Type: application/json' \
  --data-raw '{
    "assignments": [
      {
        "user": "john.doe",
        "entitlements": [
          "ABC",
          "DEFFF"
        ]
      }
    ],
    "is_certified": true,
    "is_revoked": false,
    "is_requested": false,
    "is_processed": false,
    "reason": "string",
    "datasink_status": "nack"
  }'
```

Example Response

```
{
  "status": 200
}
```

POST /api/userDetails/hrData

POST /api/userDetails/hrData

Get a user's HR data. [User, Supervisor, Ent Owner, App Owner, Admin]

Endpoint

```
/api/userDetails/hrData
```

Authorization

```
<Bearer Token JWT-value>
```

Headers

```
Content-Type      application/json
```

Body

```
{
  "employeeId": "john.doe"
}
```

Example Request

```
curl --request POST "https://autoid-api.forgerock.com/api/userDetails/hrData" \
--header "Content-Type: application/json" \
--header "Authorization: Bearer <token>" \
--data-raw '{
  "employeeId": "john.doe"
}'
```

Example Response

```
{
  "user": [
    {
      "id": "USER_NAME",
      "title": "User Name",
      "value": "john.doe"
    },
    {
      "id": "CHIEF",
      "title": "Chief",
      "value": "Yes"
    },
    {
      "id": "CITY",
      "title": "City",
      "value": "Toledo"
    },
    {
      "id": "USER_DISPLAY_NAME",
      "title": "User Display Name",
      "value": "John Doe"
    },
    {
      "id": "EMPLOYEE_TYPE",
      "title": "Employee Type",
      "value": "Employee"
    },
    {
      "id": "MANAGER",
      "title": "Manager",
      "value": "the.manager"
    }
  ],
  "displayName": "John Doe"
}
```

POST /api/userDetails/search

POST /api/userDetails/search

Search for users by name and with applied filters. [Executive, Supervisor, App Owner, Admin]

Endpoint

/api/userDetails/search

Authorization

<Bearer Token JWT-value>

Headers

Content-Type	application/json
---------------------	-------------------------

Body

```
{
  "username": "john.doe"
}
```

Example Request

```
curl --request POST "https://autoid-api.forgerock.com/api/userDetails/search" \
--header "Content-Type: application/json"
--data-raw '{
  "username": "john.doe"
}'
```

Example Response

```
{
  "values": [
    {
      "user": "john.doe",
      "isapplicationowner": "false",
      "isentitlementowner": "false",
      "issupervisor": "false",
      "userdisplayname": "John Doe"
    }
  ]
}
```

POST /api/userDetails/ent/autoprovision***POST /api/userDetails/ent/autoprovision***

Get user's entitlements for autoprovisioning. [Admin]

Endpoint

```
/api/userDetails/ent/autoprovision
```

Authorization

```
<Bearer Token JWT-value>
```

Headers

Content-Type **application/json**

Body

```
{
  "user": "john.doe"
}
```

Example Request

```
curl --request POST "https://autoid-api.forgerock.com/api/userDetails/ent/autoprovision" \
--header "Content-Type: application/json" \
--header 'Authorization: Bearer <token>' \
--data-raw '{
    "user": "john.doe"
}'
```

Example Response

```
{
  "usr_id": "string",
  "usr_name": "string",
  "ents": [
    {
      "ent_id": "string",
      "ent_name": "string",
      "ent_attribute": "string",
      "ent_risk_level": "string",
      "score": 0,
      "freq": 0,
      "frequion": 0,
      "justification": [
        {
          "title": "string",
          "value": "string"
        }
      ],
      "app_id": "string",
      "app_name": "string"
    }
  ],
  "cursor": "string"
}
```

No links

POST /api/userDetails/autoAction

POST /api/userDetails/autoAction

Get the list of entitlements for a user or list of users for an entitlement to provision, revoke, or certify. [Admin]

Endpoint

```
/api/userDetails/autoAction
```

Authorization

```
<Bearer Token JWT-value>
```

Headers

```
Content-Type      application/json
```

Body

```
{
  "action": "addAccess",
  "usrId": "john.doe",
  "entId": "entitlement_1",
  "thresholds": {
    "gte": 0,
    "gt": 0,
    "lte": 0,
    "lt": 0
  },
  "cursor": "string"
}
```

Example Request

```
curl --request POST "https://autoid-api.forgerock.com/api/userDetails/autoAction" \
--header "Content-Type: application/json" \
--header "Authorization: Bearer <token>" \
--data-raw '{
  "action": "addAccess",
  "usrId": "john.doe",
  "entId": "entitlement_1",
  "thresholds": {
    "gte": 0,
    "gt": 0,
    "lte": 0,
    "lt": 0
  },
  "cursor": "string"
}'
```

POST /api/userDetails/drivingFactor

POST /api/userDetails/drivingFactor

Get the driving factor data. [User, Supervisor, Ent Owner, App Owner, Admin]

Endpoint

/api/userDetails/drivingFactor

Authorization

<Bearer Token JWT-value>

Headers

Content-Type	application/json
---------------------	-------------------------

Params

```
{
  "entitlement": "entitlement1"
}
```

Example Request

```
curl --request POST "https://autoid-api.forgerock.com/api/userDetails/drivingFactor" \
--header "Content-Type: application/json" \
--header "Authorization: Bearer <token>" \
--data-raw '{
  "entitlement": "entitlement1"
}'
```

Example Request

```
[
  {
    "ent": "ent1",
    "attribute": {
      "title": "Chief",
      "value": "No"
    },
    "count": 3,
    "rawAttribute": "CHIEF_YES_NO_No"
  },
  {
    "ent": "ent1",
    "attribute": {
      "title": "City",
      "value": "Tacoma"
    },
    "count": 5,
    "rawAttribute": "CITY_Tacoma"
  }
]
```

POST /api/userDetails/distinct

POST /api/userDetails/distinct

Get a list of all users.

Endpoint

```
/api/userDetails/distinct
```

Authorization

```
<Bearer Token JWT-value>
```

Headers

Content-Type	application/json
---------------------	-------------------------

Params

pageSize number (optional)	Specify the number of assignments to return per page
cursor (optional)	String (Indicator on where to start a 2+ page list)
<searchable-attribute>	Any searchable attribute specified in the Identities Entity Definitions
page	

Body

```
{
  "pageSize": 2,
  "cursor": "eyJ1c3JfaWQiOiJtYXJ5LmRvbm92YW4ifQ==",
  "cost_center": "OP"
}
```

Example Request

```
curl --request POST "https://autoid-api.forgerock.com/api/userDetails/distinct" \
--header "Content-Type: application/json" \
--header "Authorization: Bearer <token>"
--data-raw {
  "pageSize": 2,
  "cursor": "eyJ1c3JfaWQiOiJtYXJ5LmRvbm92YW4ifQ==",
  "cost_center": "OP"
}
```

Example Response

```
[
  {
    "usr_id": "john.doe",
    "usr_name": "John Doe",
    "usr_manager_id": "jane.smith",
    "cost_center": "OP_TT4"
  }
]
```

Single view with application



Important

This endpoint has been deprecated in this release and will be removed in a future release.



Note

This endpoint has been updated in this release to accept only string values for all fields.

The following is an PingOne Autonomous Identity single view with applications endpoint:

POST employees

POST employees

Endpoint

```
/api/singleViewWithApp/employees
```

Authorization

```
<Bearer Token JWT-value>
```

Body

```
{
  "employeeId": "elizabeth.saiz",
  "includeLastAccessed": "true",
  "pageSize": "5"
}
```

Example Request

```
curl --location --request POST '/api/singleViewWithApp/employees' \
--header 'Content-Type: application/json' \
--data-raw '{
  "employeeId": "elizabeth.saiz",
  "includeLastAccessed": "true",
  "pageSize": "5"
}'
```

Example Response

```

{
  "high": 0,
  "medium": 1,
  "low": 1,
  "avg_score": 0.25,
  "app_name": "",
  "app_id": "",
  "entitlement_name": "",
  "high_risk": null,
  "userEntt": [
    {
      "user": "elizabeth.saiz",
      "entitlement": "192aed21-a7d1-40c3-87a3-9dfa4a3d21f5",
      "app_id": "null",
      "app_name": "test3",
      "entitlement_name": "null",
      "freq": null,
      "frequion": null,
      "high_risk": "null",
      "justification": [],
      "score": 0.1,
      "user_name": "alpha"
    },
    {
      "user": "elizabeth.saiz",
      "entitlement": "36bad416-d42c-47c2-991e-623aa3833028",
      "app_id": "null",
      "app_name": "test6",
      "entitlement_name": "null",
      "freq": null,
      "frequion": null,
      "high_risk": "null",
      "justification": [],
      "score": 0.4,
      "user_name": "vce"
    }
  ],
  "user": "elizabeth.saiz",
  "entitlementsCount": 14,
  "entitlementsRemainingCount": 10,
  "lastEntitlementId": "36bad416-d42c-47c2-991e-623aa3833028"
}

```

Access Control

The following are PingOne Autonomous Identity access control endpoints:

GET /api/userDetails/decisions

GET /api/userDetails/decisions

Get the current entitlement decisions for the user. [Supervisor, Ent Owner, App Owner, Admin]

Endpoint

/api/userDetails/decisions

Authorization

<Bearer Token JWT-value> OR <API-KEY>

Param

user=john.doe

Query Parameters

Parameter	Type	Description
user	string	User ID (required)
filter	object	Filter to add (single property shown below)

Filter Query Object Properties

Parameter	Type	Description
datasinkStatus	string	Datasink status filter ('ack' or 'nack')
timestampThresholds		Timestamp threshold object

timestampThresholds Object Properties

Parameter	Type	Description
gt	string	Greater than timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ). Cannot be present with gte .
gte	string	Greater than or equal timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ). Cannot be present with gt .

Parameter	Type	Description
lt	string	Less than timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ). Cannot be present with lte .
lte	string	Less than or equal timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ). Cannot be present with le .

Example Request (DatasinkStatus Filter)

```
curl -k -X GET \
  'datasinkStatus=ack' \
  -H 'Authorization: Bearer <token value>' \ <or> -H 'X-API-KEY: <api key value>' \
  -H 'Content-Type: application/json'
```

Example Response (DatasinkStatus Filter)

```
{
  "decisions": [
    {
      "user": "john.doe",
      "entitlement": "ent_1",
      "is_certified": false,
      "is_revoked": false,
      "is_processed": false,
      "is_archived": false,
      "author": "jane.smith",
      "author_name": "Jane Smith",
      "author_type": "Zoran Admin",
      "reason": null,
      "last_updated": "2022-01-11T19:48:17.195Z",
      "datasink_status": "ack",
      "usr_name": "John Doe",
      "ent_name": "Entitlement 1",
      "app_id": "Gateway",
      "app_name": "Gateway",
      "usr_manager_id": "john.smith",
      "conf": 0.75,
      "freq": 4,
      "freqUnion": 3
    }
  ]
}
```

POST /api/userDetails/decisions

POST /api/userDetails/decisions

Update entitlement decisions for users. [Supervisor, Ent Owner, App Owner, Admin]

Endpoint

/api/userDetails/decisions

Authorization

<Bearer Token JWT-value> OR <API-KEY>

Request Body Parameters

Parameter	Type	Description
assignments	array of assignment objects	List of assignments affected by the decision (available properties listed below)(required)
is_certified	boolean	Certification decision
is_revoked	boolean	Revoke decision
is_requested	boolean	Decision is processed
reason	string	Reason for decision
datasink_status	string	Datasink status ('ack' or 'nack')

Assignments Object Properties

Parameter	Type	Description
user	string	User ID (required)
entitlements	string array	List of entitlement ID's (required)

Body

```
{
  "assignments": [
    {
      "user": "string",
      "entitlements": [
        "string"
      ]
    }
  ],
  "is_certified": true,
  "is_revoked": true,
  "is_requested": true,
  "is_processed": true,
  "reason": "string",
  "datasink_status": "nack"
}
```

Example Request

```
curl --request POST "https://autoid-api.forgerock.com/api/userDetails/decisions" \
-H "accept: /" -H "Content-Type: application/json" \
--data-raw '{
  "assignments": [
    {
      "user": "string",
      "entitlements": [
        "string"
      ]
    }
  ],
  "is_certified": true,
  "is_revoked": true,
  "is_requested": true,
  "is_processed": true,
  "reason": "string",
  "datasink_status": "nack"
}'
```

Example Response

```
{
  "status": 200
}
```

POST /api/rules/decision

POST /api/rules/decision

Update rule decisions. [Supervisor, Ent Owner, App Owner, Admin]

Endpoint

/api/rules/decision

Authorization

<Bearer Token JWT-value>

Request Body Parameters

Parameter	Type	Description
rules	array of rules objects	List of rules affected by the decision (available properties listed below)(required)
is_autocertify	boolean	Auto-Certification decision (required)
is_autorequest	boolean	Auto-Request decision (required)
autocertify_reason	boolean	Auto-Certification reason (required)
autorequest_reason	boolean	Auto-Request reason (required)
datasink_status	string	Datasink status ('ack' or 'nack')

Rule Object Properties

Parameter	Type	Description
entitlement	string	Entitlement ID (required)
justification	string array	List of raw justifications (required)

Body

```
{
  "rules": [
    {
      "entitlement": "string",
      "justification": [
        "string"
      ]
    }
  ],
  "is_autocertify": true,
  "is_autorequest": true,
  "autocertify_reason": "string",
  "autorequest_reason": "string"
}
```

Example Request

```
curl -k -X POST \
  "https://autoid-api.forgerock.com/api/rules/decision" \
  -H 'Authorization: Bearer <token-value>' \
  -H "accept: /" -H "Content-Type: application/json" \
  --data-raw '{
    "rules": [
      {
        "entitlement": "Ent_1",
        "justification": [
          "0C_CHIEF_YES_NO_Yes",
          "0C_JOBCODE_NAME_Service Representative II",
          "0C_MANAGER_NAME_John_Doe",
          "0C_USR_EMP_TYPE_Non-Employee"
        ]
      }
    ],
    "is_autocertify": true,
    "is_autorequest": false,
    "autocertify_reason": "Goodbye, world.",
    "autorequest_reason": "Hello, world."
  }'
```

Example Response

```
Status 204: No Content
```

Applications

The following are PingOne Autonomous Identity applications view endpoints:

GET /api/applications

GET /api/applications

Get a list of applications and stats for an Application Owner. [App Owner, Admin]

Endpoint

```
/api/applications
```

Authorization

```
<Bearer Token JWT-value>
```

Params

```
ownerId (optional)  derick.hui  
cursor (optional)   string (Indicator on where to start a 2+ page list)
```

Example Request

```
curl --request GET "https://autoid-api.forgerock.com/api/applications?ownerId=derick.hui" \  
--header "Content-Type: application/json"
```

Example Response

```
{  
  "cursor": "string",  
  "total_applications": 0,  
  "total_entitlements": 0,  
  "total_assignments": 0,  
  "applications": [  
    {  
      "app_id": "string",  
      "app_name": "string",  
      "high": 0,  
      "medium": 0,  
      "low": 0,  
      "avg": 0  
    }  
  ]  
}
```

POST /api/applications/{appId}

POST /api/applications/{appId}

Get a list of entitlements and stats for a selected application. [App Owner, Admin]

Endpoint

```
/api/applications/{appId}
```

Authorization

```
<Bearer Token JWT-value>
```

Params

```
appId (required)    app_1
cursor (optional)   string (Indicator on where to start a 2+ page list)
```

Body

```
{
  "filters": [
    {
      "type": "user",
      "attribute": "city",
      "value": ["Seattle", "Denver"]
    },
    {
      "type": "user",
      "attribute": "line_of_business",
      "value": ["Distribution Operations"]
    }
  ]
}
```

Example Request

```
curl --request POST "https://autoid-api.forgerock.com/api/applications/app_1" \
--header "Content-Type: application/json" \
--header "Authorization: Bearer <token>" \
--data-raw '{
  "filters": [
    {
      "type": "user",
      "attribute": "city",
      "value": ["Seattle", "Denver"]
    },
    {
      "type": "user",
      "attribute": "line_of_business",
      "value": ["Distribution Operations"]
    }
  ]
}'
```

Example Response

```
{
  "cursor": "string",
  "total_entitlements": 0,
  "total_users": 0,
  "total_rules": 0,
  "entitlements": [
    {
      "ent": "string",
      "ent_name": "string",
      "high": 0,
      "medium": 0,
      "low": 0,
      "avg": 0
    }
  ]
}
```

POST /api/applications/{appId}/assignments

POST /api/applications/{appId}/assignments

Get filterable user-entitlement assignment and decision data for a specific application. [App Owner, Admin]

Endpoint

```
api/applications/{appId}/assignments
```

Authorization

<Bearer Token JWT-value>

Params

appId (required)	app_1
user	string
cursor (optional)	string (Indicator on where to start a 2+ page list)
sortBy	string
sortDir	string

Body

```
{
  "filters": [
    {
      "type": "user",
      "attribute": "city",
      "value": [
        "Seattle",
        "Denver"
      ]
    },
    {
      "type": "user",
      "attribute": "line_of_business",
      "value": [
        "Distribution Operations"
      ]
    }
  ]
}
```

Example Request

```
curl --request POST "https://autoid-api.forgerock.com/api/applications/app_1/assignments" \
--header "Content-Type: application/json" \
--header "Authorization: Bearer <token>" \
--data-raw '{
  "filters": [
    {
      "type": "user",
      "attribute": "city",
      "value": [
        "Seattle",
        "Denver"
      ]
    },
    {
      "type": "user",
      "attribute": "line_of_business",
      "value": [
        "Distribution Operations"
      ]
    }
  ]
}'
```

Example Response

```
{
  "cursor": "string",
  "total_users": 0,
  "total_entitlements": 0,
  "total_assignments": 0,
  "assignments": [
    {
      "ent": "string",
      "ent_name": "string",
      "confidence": 0,
      "user_id": "string",
      "user_name": "string",
      "isCertified": true,
      "dateCertified": "2021-04-14T19:10:39.178Z",
      "isRevoked": true,
      "dateRevoked": "2021-04-14T19:10:39.178Z",
      "isRequested": true,
      "dateRequested": "2021-04-14T19:10:39.178Z",
      "isProcessed": true,
      "approvalAuthor": {
        "id": "string",
        "name": "string"
      }
    }
  ]
}
```

GET /api/applications/search

GET /api/applications/search

Search all applications. [App Owner, Admin]

Endpoint

```
/api/applications/search
```

Authorization

```
<Bearer Token JWT-value>
```

Params

```
by          appOwner or enttOwner
user        user ID
q           Search query string
```

Example Request

```
curl --request GET "https://autoid-api.forgerock.com/api/applications/search" \
--header "Content-Type: application/json"
```

Example Response

```
{
  "values": [
    {
      "app_id": "string",
      "app_name": "string"
    }
  ]
}
```

Entitlements

The following are PingOne Autonomous Identity filtering by entitlements endpoints:

GET /api/entitlements/search

GET /api/entitlements/search

Search for entitlements by name and with applied filters. [Ent Owner, App Owner, Admin]

Endpoint

```
/api/entitlements/search?q=QueryString
```

Authorization

```
<Bearer Token JWT-value>
```

Params

```
by      appOwner or enttOwner
user    user ID
q       Search query string (required)
appId   Application ID to use as a filter
```

Example Request

```
curl --location --request GET 'https://autoid-api.forgerock.com/api/entitlements/search?
by=enttOwner&user=john.doe&q=WEB&appId=Salesforce' \
--header 'Content-Type: application/json'
```

Example Response

```
{
  "values": [
    {
      "id": "string",
      "app_id": "string",
      "app_name": "string",
      "entt_name": "string"
    }
  ]
}
```

POST /api/entitlements/stats

POST /api/entitlements/stats

Get data for entitlements view. [Supervisor, Ent Owner, Admin]

Endpoint

```
/api/entitlements/stats?by=supervisor/entitlementOwner/admin
```

Authorization

```
<Bearer Token JWT-value>
```

Params

```
by      supervisor, roleOwner
```

Body

```
{
  "ownerId": "timothy.slack",
  "isHighRiskOnly": true,
  "isMediumLowRiskOnly": false,
  "isUserEntitlementsIncluded": true,
  "filters": [{
    "type": "app_id",
    "group": "criticality",
    "value": "Essential"
  }]
}
```

Example Request

```
curl --location --request POST 'https://autoid-api.forgerock.com/api/entitlements/stats?by=supervisor' \
--header 'content-type: application/json' \
--data-raw '{
  "ownerId": "timothy.slack",
  "isHighRiskOnly": true,
  "isMediumLowRiskOnly": false,
  "isUserEntitlementsIncluded": true,
  "filters": [{
    "type": "app_id",
    "group": "criticality",
    "value": "Essential"
  }]
}'
```

Example Response

```
{
  "total_entitlements": 0,
  "total_subordinates": 0,
  "unscoredEntitlements": 0,
  "scoredEntitlements": 0,
  "usersWithNoEntitlement": 0,
  "usersWithNoScoredEntitlement": 0,
  "distinct_apps": [
    {
      "app_id": "string",
      "app_name": "string",
      "low": 0,
      "medium": 0,
      "high": 0
    }
  ],
  "users": [
    {
      "user": "string",
      "user_name": "string",
      "high": 0,
      "medium": 0,
      "low": 0,
      "avg": "string"
    }
  ],
  "entitlements": [
    {
      "entitlement": "string",
      "entitlement_name": "string",
      "app_id": "string",
      "high_risk": "string",
      "high": 0,
      "medium": 0,
      "low": 0,
      "avg": "string"
    }
  ]
}
```

GET /api/entitlements/id/{id}

GET /api/entitlements/id/{id}

Get entitlement details. [User, Supervisor, Ent Owner, App Owner, Admin]

Endpoint

/api/entitlements/id/{id+}

Authorization

<Bearer Token JWT-value>

Params

by entitlement ID

Example Request

```
curl --request GET "https://autoid-api.forgerock.com/api/entitlements/id/1234" \  
--header "Content-Type: application/json"
```

Example Response

```

{
  "entitlement_name": "string",
  "scores": {
    "avg": 0,
    "high": 0,
    "medium": 0,
    "low": 0
  },
  "drivingFactors": [
    {
      "attribute": {
        "id": "string",
        "title": "string",
        "value": "string"
      },
      "count": 0
    }
  ],
  "userScores": [
    {
      "score": 0,
      "count": 0
    }
  ],
  "users": [
    {
      "user": "string",
      "user_name": "string",
      "app_id": "string",
      "freq": 0,
      "frequion": 0,
      "justification": [
        {
          "title": "string",
          "value": "string"
        }
      ],
      "rawJustification": [
        "string"
      ],
      "score": 0
    }
  ]
}

```

GET /api/entitlements/unscored

GET /api/entitlements/unscored

Get unscored entitlements and users for a given Supervisor or Entitlement Owner ID. [Supervisor, Ent Owner, Admin]

Endpoint

```
/api/entitlements/unscored
```

Authorization

```
<Bearer Token JWT-value>
```

Params

```
by      supervisor, entitlement owner
user    supervisor or entitlement owner user ID
```

Example Request

```
curl --request GET "https://autoid-api.forgerock.com/api/entitlements/unscored?
by=supervisor&user=1234" \
--header "Content-Type: application/json"
```

GET /api/entitlements/distinct

GET /api/entitlements/distinct

Get a list of all entitlements.

Endpoint

```
/api/entitlements/distinct
```

Authorization

```
<Bearer Token JWT-value>
```

Example Request

```
curl --location --request GET 'https://autoid-api.forgerock.com/api/entitlements/distinct' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer <token>'
```

Example Response

```
[
  {
    "ent_id": "AccessType : XMLP_ADMIN",
    "ent_name": "AccessType : XMLP_ADMIN",
    "ent_owner_id": "julie.yee",
    "app_id": "Salesforce",
    "ent_criticality": "Non-Essential",
    "ent_risk_level": "Medium"
  }
]
```

GET /api/entitlements/recommendations

GET /api/entitlements/recommendations

Get a list of entitlement recommendations for a given set of user attributes.

Endpoint

/api/entitlements/recommendations

Authorization

<Bearer Token JWT-value>

Body

```
{
  "confidenceThreshold": 0.1,
  "maxResults": 1000,
  "offset": 200,
  "userAttributes": [
    "0E_USR_MANAGER_ID_gregory.suhr",
    "13_USR_DEPARTMENT_NAME_Facilities Area A",
    "0C_CHIEF_YES_NO_No",
    "0C_MANAGER_NAME_Gregory Suhr",
    "0C_USR_EMP_TYPE_Employee",
    "13_USR_DEPARTMENT_NAME_Wireless Operations"
  ]
}
```

Example Request

```
curl --request GET "https://autoid-api.forgerock.com/api/entitlements/recommendations" \
--header "Content-Type: application/json" \
--header "Authorization: Bearer <token>" \
--data-raw '{
  "confidenceThreshold": 0.1,
  "maxResults": 1000,
  "offset": 200,
  "userAttributes": [
    "0E_USR_MANAGER_ID_gregory.suhr",
    "13_USR_DEPARTMENT_NAME_Facilities Area A",
    "0C_CHIEF_YES_NO_No",
    "0C_MANAGER_NAME_Gregory Suhr",
    "0C_USR_EMP_TYPE_Employee",
    "13_USR_DEPARTMENT_NAME_Wireless Operations"
  ]
}'
```

Example Response

```
[
  {
    "attributes": [
      "0C_CHIEF_YES_NO_No",
      "0E_USR_MANAGER_ID_gregory.suhr"
    ],
    "entitlement": "06_ENT_ID_WEB_user_WEB RCQ Flare NonIT Distribution_II",
    "confidence": 0.14,
    "frequency": 22
  },
  {
    "attributes": [
      "0C_MANAGER_NAME_Gregory Suhr",
      "13_USR_DEPARTMENT_NAME_Facilities Area A"
    ],
    "entitlement": "06_ENT_ID_Web_tildeNon-security plus",
    "confidence": 0.14,
    "frequency": 28
  },
]
```

Assignments

The following endpoint has been added to support the extraction of assignments. New APIs introduced in this release are marked with ★.

POST /api/assignments

★ POST /api/assignments

Get a list of all assignments or filtered assignments when the following optional filtering is sent as a JSON request body.

The endpoint requires a valid API key passed in a `X-API-KEY` header for authorization (refer to [Generate an API key](#)) or an authorized admin-level bearer token.

Endpoint

+

`/api/assignments`

Headers

+

Content-Type `application/json`

Body Parameters

Parameter	Type	Description
cursor	string	Cursor to send for the next page of assignments (the response returns null if there are no more pages)
pageSize	integer	Number of assignments to return per page
ent_id	string	Entitlement ID to filter by
usr_id	string	User ID to filter by
app_id	string	Application ID to filter by

Example Request

+

```
curl -k -X POST \
https://autoid-ui.forgerock.com/api/assignments \
-H 'X-API-KEY: <api key value>' \
-H 'Content-Type: application/json' \
-d '{
  "pageSize": 20,
  "cursor": "WyJhYXJvbi5maXNjaGVyIiwVVCX3VzZXJfU2hhcmVkaXRfQURNSU5fsUkiXQ=="
}'
```

Example Response

+

```

{
  "cursor":
  "WyJhYXJvbi5mb2x0eiIsIldlYl9TZXJ2aWNlQW5hbHl0aWNzIDogS1BJIEFnZW50IGFuZCBQb3dlciBBY2Nlc3NfSUKiXQ==",
  "assignments": [
    {
      "ent_id": "WEB_user_Web_Local Access 32 All_II_Europe",
      "usr_id": "aaron.fischer",
      "score": null,
      "justification": []
    },
    {
      "ent_id": "WEB_user_Web_Shared_Edit_ADMIN_Europe",
      "usr_id": "aaron.fischer",
      "score": null,
      "justification": []
    },
    {
      "ent_id": "WEB_user_Web_WEB RCQ Flare NonIT Distribution_Europe",
      "usr_id": "aaron.fischer",
      "score": 0.98,
      "justification": [
        {
          "id": "CHIEF_YES_NO",
          "title": "chief_yes_no",
          "value": "Yes"
        },
        {
          "id": "USR_DEPARTMENT_NAME",
          "title": "USR_DEPARTMENT_NAME",
          "value": "Information Systems 1"
        }
      ]
    },
    {
      "ent_id": "Web_Flare NonIT Distribution",
      "usr_id": "aaron.fischer",
      "score": 0.6,
      "justification": [
        {
          "id": "CITY",
          "title": "city",
          "value": "Jacksonville"
        },
        {
          "id": "USR_EMP_TYPE",
          "title": "usr_emp_type",
          "value": "Employee"
        },
        {
          "id": "LINE_OF_BUSINESS",
          "title": "line_of_business",
          "value": "Strategy and Policy"
        }
      ]
    }
  ]
}

```

```
}  
]  
}  
]  
}
```

Rules

The following are PingOne Autonomous Identity rules endpoints:

GET /api/rules/info

GET /api/rules/info

List information and statistics regarding available rules. [Ent Owner, App Owner, Admin]

Endpoint

```
/api/rules/info
```

Authorization

```
<Bearer Token JWT-value>
```

Params

```
by      enttowner, appOwner  
user    patrick.murphy
```

Example Request

```
curl --request GET "https://autoid-api.forgerock.com/api/rules/info?by=appOwner&user=patrick.murphy"  
\br/>--header "Content-Type: application/json"
```

Example Response

```
{
  "countRules": 0,
  "countAssignments": 0,
  "countApplications": 0,
  "applications": [
    {
      "app_id": "string",
      "app_name": "string",
      "countAssignments": 0,
      "low": 0,
      "medium": 0,
      "high": 0
    }
  ]
}
```

GET /api/rules

GET /api/rules

List the available rules for a user [Ent Owner, App Owner, Admin]. You can filter by time period by using greater than and/or less than date-times.

Endpoint

/api/rules/

Authorization

<Bearer Token JWT-value> OR <API-KEY>

Query Parameters

Parameter	Type	Description
by	string	User type to apply when searching rules: <code>enttOwner</code> , <code>appOwner</code>
user	string	User ID when applying <i>by</i> filters (required if using "by")
cursor	string	Position to start a 2+ page list.
pageSize	number	Number of records per page to retrieve.
filter	filter object	Additional filters to apply (refer to properties below).

Filter Query Object Properties

Parameter	Type	Description
lowConfidence	boolean	Low confidence scores only.
medConfidence	boolean	Medium confidence scores only.
highConfidence	boolean	High confidence scores only.
app_id	string array	Array of application IDs.
autoCertify	boolean	Auto-certified only.
autoRequest	boolean	Auto-requested only.
entitlement	string	Entitlement ID
datasinkStatus	string	Datasink status filter ('ack' or 'nack')
timestampThresholds	object	Timestamp threshold object (available properties below)

timestampThresholds Object Properties

Parameter	Type	Description
gt	string	Greater than timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ). Cannot be present with gte .
gte	string	Greater than or equal timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ). Cannot be present with gt .
lt	string	Less than timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ). Cannot be present with lte .
lte	string	Less than or equal timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ). Cannot be present with le .

Example Request (Datasink status filter)

```
curl -k -X GET \
'datasinkStatus=ack' \
-H 'Authorization: Bearer <token value>' <OR> -H 'X-API-KEY: <api-key-value>' \
-H 'Content-Type: application/json'
```

Example Response (Datasink status filter)

```
{
  "cursor": null,
  "totalRulesCount": 1,
  "rules": [
    {
      "entitlement": {
        "ent_id": "Cognos EDI Reporting",
        "ent_owner_id": "david.elliott",
        "ent_criticality": "Essential",
        "ent_risk_level": "High",
        "ent_name": "Cognos EDI Reporting",
        "app_id": "SAP"
      },
      "app": {
        "app_criticality": "Essential",
        "app_name": "SAP",
        "app_owner_id": "derick.hui",
        "app_risk_level": "High",
        "app_id": "SAP"
      },
      "justification": [
        {
          "id": "CHIEF_YES_NO",
          "title": "Chief?",
          "value": "No"
        },
        {
          "id": "MANAGER_NAME",
          "title": "Manager Name",
          "value": "Patrick Gardner"
        },
        {
          "id": "USR_EMP_TYPE",
          "title": "Employee Type",
          "value": "Employee"
        },
        {
          "id": "USR_DEPARTMENT_NAME",
          "title": "User department Name",
          "value": "General Office"
        }
      ],
      "rawJustification": [
        "0C_CHIEF_YES_NO_No",
        "0C_MANAGER_NAME_Patrick Gardner",
        "0C_USR_EMP_TYPE_Employee",
        "13_USR_DEPARTMENT_NAME_General Office"
      ],
      "assignees": [
        {
          "id": "gordon.choy",
          "name": "Gordon Choy"
        }
      ]
    }
  ]
}
```

```

    },
    {
      "id": "jennifer.kanenaga",
      "name": "Jennifer Kanenaga"
    },
    {
      "id": "lawrence.nicholls",
      "name": "Lawrence Nicholls"
    },
    {
      "id": "roel.dilag",
      "name": "Roel Dilag"
    },
    {
      "id": "salvatore.taormina",
      "name": "Salvatore Taormina"
    }
  ],
  "confidence": 1,
  "countUnassigned": 0,
  "countAssigned": 5,
  "isAutoCertify": true,
  "autoCertifyDate": "2021-10-01T19:01:31.567Z",
  "isAutoRequest": true,
  "autoRequestDate": "2021-10-01T19:01:31.567Z",
  "approvalAuthor":
  {
    "id": "bob.rodgers",
    "name": "Bob Rodgers"
  },
  "requestApprovalReason": "Test",
  "certifyApprovalReason": "Test",
  "datasink_status": "ack"
}
]
}

```

POST /api/rules/decision

POST /api/rules/decision

Update rule decisions. [Supervisor, Ent Owner, App Owner, Admin]

Endpoint

/api/rules/decision

Authorization

<Bearer Token JWT-value>

Request Body Parameters

Parameter	Type	Description
rules	array of rules objects	List of rules affected by the decision (available properties listed below)(required)
is_autocertify	boolean	Auto-Certification decision (required)
is_autorequest	boolean	Auto-Request decision (required)
autocertify_reason	boolean	Auto-Certification reason (required)
autorequest_reason	boolean	Auto-Request reason (required)
datasink_status	string	Datasink status ('ack' or 'nack')

Rule Object Properties

Parameter	Type	Description
entitlement	string	Entitlement ID (required)
justification	string array	List of raw justifications (required)

Body

```
{
  "rules": [
    {
      "entitlement": "string",
      "justification": [
        "string"
      ]
    }
  ],
  "is_autocertify": true,
  "is_autorequest": true,
  "autocertify_reason": "string",
  "autorequest_reason": "string"
}
```

Example Request

```
curl -k -X POST \
'https://autoid-ui.forgerock.com/api/rules/decisions' \
-H 'Authorization: Bearer <token value>' \
-H 'Content-Type: application/json' \
-d '{
  "rules": [
    {
      "entitlement": "Ent_1",
      "justification": [
        "0C_CHIEF_YES_NO_Yes",
        "0C_JOBCODE_NAME_Service Representative II",
        "0C_MANAGER_NAME_John_Doe",
        "0C_USR_EMP_TYPE_Non-Employee"
      ]
    }
  ],
  "is_autocertify": true,
  "is_autorequest": false,
  "autocertify_reason": "Goodbye, world.",
  "autorequest_reason": "Hello, world."
}'
```

Example Response

```
Status 204: No Content
```

Filters

The following are PingOne Autonomous Identity Filters endpoints:

GET /api/filters/owner

GET /api/filters/owner

Get filterable attributes and values. [Supervisor, Ent Owner, Admin]

Endpoint

```
/api/filters/owner?by=supervisor&user=albert.pardini
```

Authorization

```
<Bearer Token JWT-value>
```

Query Parameters

```
by      supervisor, enttOwner
user    albert.pardini
```

Example Request

```
curl --request GET "https://autoid-api.forgerock.com/api/filters/owner?
by=supervisor&user=albert.pardini" \
--header "Content-Type: application/json"
```

Example Response

```
{
  "items": [
    {
      "title": "string",
      "field": "string",
      "filters": {
        "field": "string",
        "title": "string",
        "options": [
          {
            "text": "string",
            "value": "string",
            "count": 0
          }
        ]
      }
    }
  ]
}
```

GET /api/filters/app

GET /api/filters/app

Get filterable attributes and values. [App Owner, Admin]

Endpoint

```
/api/filters/app
```

Authorization

```
<Bearer Token JWT-value>
```

Query Parameters

```
id: application ID
```

Example Request

```
curl --request GET "https://autoid-api.forgerock.com/api/filters/app?id=app_1" \
--header "Content-Type: application/json"
```

Example Response

```
{
  "items": [
    {
      "title": "string",
      "field": "string",
      "filters": {
        "field": "string",
        "title": "string",
        "options": [
          {
            "text": "string",
            "value": "string",
            "count": 0
          }
        ]
      }
    }
  ]
}
```

Roles

The following are PingOne Autonomous Identity filtering by roles endpoints:

POST /api/roles

POST /api/roles

Create draft roles and make updates to roles.

Endpoint

```
/api/roles
```

Authorization

<Bearer Token JWT-value> OR <API-KEY>

Request Body Parameters

Parameter	Type	Description
action	string	Action to perform ('create', 'save', 'publish', 'unpublish') (required)
updateAllMetadata	boolean	Update metadata for all related roles regardless of status
role	role object	Role object (properties below) (required)

Role Object Properties

Parameter	Type	Description
role_id	string	Role ID in uuid format (required)
status	string	Status of role ('draft', 'candidate', or 'active') (required)
custom_role	boolean	Role is a custom role
member_count	number	Number of users the roles applies to
assignment_count	number	Number of assignments the role applies to
entitlements	string array	List of entitlement IDs that are part of the role
justifications	string array	List of raw justifications
datasink_status	string	Datasink status ('ack' or 'nack')
role_metadata	role metadata object	Role metadata (properties below)
entitlements_metadata	array of entitlement metadata objects	List of entitlement metadata for each entitlement (object properties below)

Role Metadata Object Properties

Parameter	Type	Description
role_name	string	Display name of role

Parameter	Type	Description
description	string	Role description
role_owner_id	string	Role owner ID
role_owner_display_name	string	Role owner display name

Entitlement Metadata Object Properties

Parameter	Type	Description
ent_id	string	Entitlement ID (required)
ent_name	string	Entitlement name (required)
application	application object	Application metadata (object properties below)
entitlement_owner	owner object	Entitlement owner data (object properties below)

Application Object Properties

Parameter	Type	Description
app_id	string	Application ID
app_name	string	Application name
application_owner	owner object	Application owner data (object properties below)

Owner Object Properties

Parameter	Type	Description
usr_id	string	User ID (required)
usr_name	string	User name (required)
usr_manager_id	string	User manager ID (required)

Example Request

```
curl --location --request POST 'https://autoid-api.forgerock.com/api/roles' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer <token>' \
--data-raw '{
  "action": "save",
  "updateAllMetadata": false,
  "role": {
    "role_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "status": "draft",
    "custom_role": false,
    "member_count": 0,
    "assignment_count": 0,
    "entitlements": [
      "string"
    ],
    "entitlements_metadata": [
      {
        "ent_id": "string",
        "ent_name": "string",
        "application": {
          "app_id": "string",
          "app_name": "string",
          "application_owner": {
            "usr_id": "string",
            "usr_name": "string",
            "usr_manager_id": "string"
          }
        }
      },
      "entitlement_owner": {
        "usr_id": "string",
        "usr_name": "string",
        "usr_manager_id": "string"
      }
    ],
    "justifications": [
      "string"
    ],
    "role_metadata": {
      "role_name": "string",
      "description": "string",
      "role_owner_display_name": "string",
      "role_owner_id": "string"
    }
  }
}'
```

Example Response

204 (No Content)

POST /api/roles/delete

POST /api/roles/delete

Delete roles.

Endpoint

```
/api/roles/delete
```

Authorization

```
<Bearer Token JWT-value>
```

Body

```
{
  "role_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "status": "draft" | "active"
}
```

Example Request

```
curl --location --request POST 'https://autoid-api.forgerock.com/api/admin/updateSelf' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer <token>' \
--data-raw '{
  "role_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "status": "draft" | "active"
}'
```

Example Response

```
204 (No content)
```

POST /api/roles/export

POST /api/roles/export

Export role data to json.

Endpoint

```
/api/roles/export
```

Authorization

<Bearer Token JWT-value> OR <API-KEY>

Query Parameters

Parameter	Type	Description
usrId	string	Roles that apply for a particular user ID
entId	string	Roles that apply for a particular entitlement ID
status	string	Status of role ('draft', 'candidate', or 'active')
role_name	string	Role name
description	string	Role description
role_owner_id	string	Role owner ID
role_owner_display_name	string	Role owner name
datasinkStatus	string	Datasink status filter ('ack', 'nack')
timestampThresholds	object	Timestamp threshold object (available properties below)

timestampThresholds Object Properties

Parameter	Type	Description
gt	string	Greater than timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ). Cannot be present with gte .
gte	string	Greater than or equal timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ). Cannot be present with gt .
lt	string	Less than timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ). Cannot be present with lte .
lte	string	Less than or equal timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ). Cannot be present with le .

Body

```
{
  "usrId": "john.doe",
  "datasinkStatus": "nack"
}
```

Example Request (Datasink Filter)

```
curl -k -X POST \
'https://autoid-ui.forgerock.com/api/roles/export' \
--header 'Content-type: application/json' \
--header 'Authorization: Bearer <token>' <OR> -H 'X-API-KEY: <api-key-value>' \
--data-raw '{
  "usrId": "john.doe",
  "datasinkStatus": "nack"
}'
```

Example Response (Datasink Filter)

```

{
  "roles": [
    {
      "temp_role_name": "Role J0-R21",
      "normalized_role_name": "role j0-r21",
      "member_count": 1,
      "assignment_count": 1,
      "entitlement_count": 1,
      "role_id": "4aaf81db-2f8c-42b4-b954-1018a71743de",
      "status": "candidate",
      "entitlements": [
        "Ent_1"
      ],
      "entitlements_metadata": [
        {
          "ent_criticality": "Essential",
          "ent_id": " Ent_1",
          "ent_name": " Ent_1",
          "ent_risk_level": "Low",
          "application": {
            "app_criticality": "Essential",
            "app_id": "Active Directory",
            "app_name": "Active Directory",
            "app_risk_level": "High",
            "application_owner": {
              "chief_yes_no": "Yes",
              "city": "Kansas City",
              "cost_center": "CON_SD9",
              "department": "Facilities Area A",
              "is_active": "Y",
              "job_description": "Facilities Area A",
              "jobcode_name": "Operating Clerk",
              "line_of_business": "Transmission Operations",
              "line_of_business_subgroup": "Real Estate",
              "manager_name": "Thomas Shawyer",
              "usr_department_name": "Facilities Area A",
              "usr_display_name": "Derick Hui",
              "usr_emp_type": "Non-Employee",
              "usr_id": "derick.hui",
              "usr_manager_id": "thomas.shawyer",
              "usr_name": "Derick Hui"
            }
          }
        },
        {
          "ent_criticality": "Essential",
          "ent_id": " Ent_2",
          "ent_name": " Ent_2",
          "ent_risk_level": "Low",
          "application": {
            "app_criticality": "Essential",
            "app_id": "Active Directory",
            "app_name": "Active Directory",
            "app_risk_level": "High",
            "application_owner": {
              "chief_yes_no": "Yes",
              "city": "Kansas City",
              "cost_center": "CON_SD9",
              "department": "Facilities Area A",
              "is_active": "Y",
              "job_description": "Facilities Area A",
              "jobcode_name": "Operating Clerk",
              "line_of_business": "Transmission Operations",
              "line_of_business_subgroup": "Real Estate",
              "manager_name": "Thomas Shawyer",
              "usr_department_name": "Facilities Area A",
              "usr_display_name": "Derick Hui",
              "usr_emp_type": "Non-Employee",
              "usr_id": "derick.hui",
              "usr_manager_id": "thomas.shawyer",
              "usr_name": "Derick Hui"
            }
          }
        }
      ],
      "entitlement_owner": {
        "chief_yes_no": "No",
        "city": "Saint Paul",
        "cost_center": "OP_TT4",
        "department": "InfoSYS Power Gen",
        "is_active": "Y",
        "job_description": "InfoSYS Power Gen",
        "jobcode_name": "Lineman",

```

```

        "line_of_business": "Ethics and Compliance",
        "line_of_business_subgroup": "System Operations",
        "manager_name": "James Bosch",
        "usr_department_name": "InfoSYS Power Gen",
        "usr_display_name": "Carolyn Latanafrancia",
        "usr_emp_type": "Non-Employee",
        "usr_id": "carolyn.latanafrancia",
        "usr_manager_id": "james.bosch",
        "usr_name": "Carolyn Latanafrancia"
    }
}
],
"justifications": [
    "0B_COST_CENTER_SOL_ER2 19_LINE_OF_BUSINESS_SUBGROUP_Energy%20Solutions"
],
"users": [
    {
        "usr_id": "aaron.lozada",
        "usr_display_name": "Aaron Lozada",
        "attributes": [
            "13_USR_DEPARTMENT_NAME_Operations%20SUP",
            "0F_JOB_DESCRIPTION_Operations_%20SUP",
            "0C_JOB_CODE_NAME_Apprentice",
            "0C_MANAGER_NAME_Gary%20Amelio",
            "09_IS_ACTIVE_Y",
            "10_LINE_OF_BUSINESS_Distribution%20Operations",
            "10_USR_DISPLAY_NAME_Aaron%20Lozada",
            "0B_COST_CENTER_SOL_ER2",
            "08_USR_NAME_Aaron%20Lozada",
            "0C_CHIEF_YES_NO_No",
            "0C_USR_EMP_TYPE_Employee",
            "19_LINE_OF_BUSINESS_SUBGROUP_Energy%20Solutions",
            "04_CITY_Kansas%20City"
        ]
    }
]
}

```

Ingest

The following endpoints support the ingestion of base entities, such as applications, entitlements, identities, assignments, data sources, and mappings. New APIs introduced in this release are marked with ★.

To access these endpoints, you need a valid API key in the *X-API-KEY* header for authorization. To obtain an API key, see [Generate an API key](#).

POST /api/ingest/applications

POST /api/ingest/applications

Create, update, upsert, or delete application entities.

Endpoint

```
/api/ingest/applications
```

Authorization

```
<API Key-value>
```

Body Parameters

Parameter	Type	Description
action	string	Action to perform: create, update, upsert, delete (required)
apps	array	Array of application objects (properties below) (required)

Base Application Object Properties:

Parameter	Type	Description
app_id	string	Application ID (required)
app_name	string	Application Name
app_owner_id	string	Application owner user ID

Example Request

```
curl -k -X POST \
https://autoid-ui.forgerock.com/api/ingest/applications \
-H 'X-API-KEY: <api key value>' \
-H 'Content-Type: application/json' \
-d '{
  "action": "create",
  "apps": [
    {
      "app_id": "app1",
      "app_name": "Test App",
      "app_owner_id": "bob.rodgers"
    }
  ]
}'
```

Example Response

```
{
  "message": "ok"
}
```

POST /api/ingest/entitlements

POST /api/ingest/entitlements

Create, update, upsert, or delete entitlement entities.

Endpoint

```
/api/ingest/entitlements
```

Authorization

```
<API Key-value>
```

Body

Body Parameters

Parameter	Type	Description
action	string	Action to perform: create, update, upsert, delete (required)

Parameter	Type	Description
entitlements	array	Array of entitlements objects (properties below) (required)

Base Entitlements Object Properties

Parameter	Type	Description
ent_id	string	Entitlement ID (required)
ent_name	string	Entitlement Name
ent_owner_id	string	Entitlement owner user ID
app_id	string	Application ID

Example Request

```
curl -k -X POST \
https://autoid-ui.forgerock.com/api/ingest/entitlements \
-H 'X-API-KEY: <api key value>' \
-H 'Content-Type: application/json' \
-d '{
  "action": "upsert",
  "entitlements": [
    {
      "ent_id": "ent1",
      "ent_name": "Test Ent",
      "ent_owner_id": "bob.rodgers",
      "app_id": "app1"
    }
  ]
}'
```

Example Response

```
{
  "message": "ok"
}
```

POST /api/ingest/identities

POST /api/ingest/identities

Create, update, upsert, or delete identity entities.

Endpoint

/api/ingest/identities

Authorization

<API Key-value>

Body Parameters

Parameter	Type	Description
action	string	Action to perform: create, update, upsert, delete (required)
entitlements	array	Array of identities objects (properties below) (required)

Base Entitlements Object Properties

Parameter	Type	Description
usr_id	string	User ID (required)
usr_name	string	User name
usr_manager_id	string	User’s manager ID

Example Request

```
curl -k -X POST \
https://autoid-ui.forgerock.com/api/ingest/identities \
-H 'X-API-KEY: <api key value>' \
-H 'Content-Type: application/json' \
-d '{
  "action": "upsert",
  "identities": [
    {
      "usr_id": "john.doe",
      "usr_name": "John Doe",
      "usr_manager_id": "bob.rogers"
    },
    {
      "usr_id": "jane.smith",
      "usr_name": "Jane Smith",
      "usr_manager_id": "bob.rogers"
    }
  ]
}
```

Example Response

```
{
  "message": "ok"
}
```

POST /api/ingest/assignments

POST /api/ingest/assignments

Create, update, upsert, or delete assignment entities.

Endpoint

/api/ingest/assignments

Authorization

<API Key-value>

Body Parameters

Parameter	Type	Description
action	string	Action to perform: create, update, upsert, delete (required)
entitlements	array	Array of assignment objects (properties below) (required)

Base Entitlements Object Properties

Parameter	Type	Description
ent_id	string	Entitlementd ID (required)
usr_id	string	User ID (required)

Example Request

```
curl -k -X POST \
https://autoid-ui.forgerock.com/api/ingest/assignments \
-H 'X-API-KEY: <api key value>' \
-H 'Content-Type: application/json' \
-d '{
  "action": "upsert",
  "assignments": [
    {
      "usr_id": "john.doe",
      "ent_id": "ent1"
    },
    {
      "usr_id": "jane.smith",
      "ent_id": "ent1"
    }
  ]
}'
```

Example Response

```
{
  "message": "ok"
}
```

POST /api/ingest/datasources

POST /api/ingest/datasources

Get data sources. Optional filtering can be applied as a JSON request body outlined below:

Endpoint

```
/api/ingest/datasources
```

Authorization

```
<API Key-value>
```

Body Parameters

Parameter	Type	Description
datasourceId	string	Data source ID
name	string	Data source name

Parameter	Type	Description
isActive	boolean	Data source activated
connectionTypes	string array	List of connection types to filter: jdbc, csv, generic
entityTypes	string array	List of entity types

Example Request

```
curl -k -X GET \  
https://autoid-ui.forgerock.com/api/ingest/datasources \br/>-H 'X-API-KEY: <api key value>' \  
-H 'Content-Type: application/json' \  
-d '{  
  "connectionTypes": ["csv"],  
  "isActive": true  
}
```

Example Response

```
[
  {
    "datasource_id": "fdbfb998-7b3e-4ddc-9e4a-a4c46cace49e",
    "name": "Test data",
    "sync_type": "full",
    "icon": "apps",
    "isActive": true,
    "entityTypes": {
      "/autoid/system/datasources/fdbfb998-7b3e-4ddc-9e4a-a4c46cace49e/applications": {
        "uri": {
          "file": "file:/data/input/applications.csv"
        }
      },
      "/autoid/system/datasources/fdbfb998-7b3e-4ddc-9e4a-a4c46cace49e/assignments": {
        "uri": {
          "file": "file:/data/input/assignments.csv"
        }
      },
      "/autoid/system/datasources/fdbfb998-7b3e-4ddc-9e4a-a4c46cace49e/entitlements": {
        "uri": {
          "file": "file:/data/input/entitlements.csv"
        }
      },
      "/autoid/system/datasources/fdbfb998-7b3e-4ddc-9e4a-a4c46cace49e/identities": {
        "uri": {
          "file": "file:/data/input/identities.csv"
        }
      }
    },
    "connectionSettings": {
      "csv": {}
    }
  }
]
```

POST /api/ingest/mappings

POST /api/ingest/mappings

Get mappings. Optional filtering can be applied as a JSON request body outlined below:

Endpoint

/api/ingest/mappings

Authorization

<API Key-value>

Body Parameters

Parameter	Type	Description
mappingId	string	Mapping ID
sourceEntity	string	Mapping source entity
targetEntity	string	Mapping target entity
sourceProperties	string array	List of source properties to filter on
targetProperties	string array	List of target properties to filter on

Example Request

```
curl -k -X GET \
https://autoid-ui.forgerock.com/api/ingest/mappings \
-H 'X-API-KEY: <api key value>' \
-H 'Content-Type: application/json' \
-d '{
  "targetProperties": ["app_id", "app_name"]
}'
```

Example Response

```
[
  {
    "mapping_id": "fb6896e5-8d0a-4bd7-b10d-5608c9a953a1",
    "source_entity": "/autoid/system/datasources/0474f92c-d530-43cc-a012-29fb6c8b3b8b/applications",
    "target_entity": "/autoid/base/applications",
    "properties": [
      {
        "source": "APP_ID",
        "target": "app_id",
        "apply": true
      },
      {
        "source": "APP_NAME",
        "target": "app_name",
        "apply": true
      },
      {
        "source": "APP_OWNER_ID",
        "target": "app_owner_id",
        "apply": true
      }
    ]
  }
]
```

Jobs

You can define, run, and get the status of each job using REST API endpoints. When using REST calls, the available job types are the following:

- ingest
- train
- mine
- predict-as-is
- recommend
- load
- create-assignment-index
- create-assignment-index-report
- anomaly
- insight
- audit

The following are PingOne Autonomous Identity jobs endpoints:

POST /api/job_definition

POST /api/job_definition

Set up a job definition.

Endpoint

```
/api/job_definition
```

Authorization

```
Bearer <Token JWT-value> or  
<API Key-value>
```

Body

```
{
  "branch": "actual",
  "contextId": "40c20f01-a9d8-4284-b290-c8b6ccdb8b77",
  "entityData": [
    {
      "job_name": "ShowCaseCSVAnomaly",
      "job_type": "anomaly",
      "job_parameters": {
        "driverMemory": "2g",
        "driverCores": 3,
        "executorMemory": "3G",
        "executorCores": 6
      }
    }
  ],
  "indexingRequired": true,
  "tags": {},
  "indexInSync": true
}
```

**Note**

contextId is a unique identifier string. It can be anything you define.

Example Request

```

curl 'https://autoid-ui.forgerock.com/jas/entity/persist/autoid/api/job_definition' \
-H 'authority: autoid-ui.forgerock.com' \
-H 'sec-ch-ua: "Chromium";v="92", " Not A;Brand";v="99", "Google Chrome";v="92"' \
-H 'accept: application/json, text/plain, /' \
-H 'x-tenant-id: 8700f5cb-eaca-461e-8c2e-245a25f2399d' \
-H 'authorization: 1b7789f0-6c2f-4afa-a84b-a65a28f5c1a1' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/92.0.4515.131 Safari/537.36' \
-H 'content-type: application/json' \
-H 'origin: https://autoid-ui.forgerock.com' \
-H 'sec-fetch-site: same-origin' \
-H 'sec-fetch-mode: cors' \
-H 'sec-fetch-dest: empty' \
-H 'referrer: https://autoid-ui.forgerock.com/jobs' \
-H 'accept-language: en-US,en;q=0.9' \
--data-raw '{
  "branch": "actual",
  "contextId": "40c20f01-a9d8-4284-b290-c8b6ccdb8b77",
  "entityData": [
    {
      "job_name": "ShowCaseCSVAnomaly",
      "job_type": "anomaly",
      "job_parameters": {
        "driverMemory": "2g",
        "driverCores": 3,
        "executorMemory": "3G",
        "executorCores": 6
      }
    }
  ],
  "indexingRequired": true,
  "tags": {},
  "indexInSync": true
}' \
--compressed \
--insecure
}
}'

```

The Job definition for data ingestion requires a `datasourceId`, which you can query. Refer to [\[get-datasource-id\]](#).

Example Request (ingest)

```

curl 'https://autoid-ui.forgerock.com/jas/entity/persist/autoid/api/job_definition' \
-H 'authority: autoid-ui.forgerock.com' \
-H 'sec-ch-ua: "Chromium";v="92", " Not A;Brand";v="99", "Google Chrome";v="92"' \
-H 'accept: application/json, text/plain, /' \
-H 'x-tenant-id: 8700f5cb-eaca-461e-8c2e-245a25f2399d' \
-H 'authorization: 1b7789f0-6c2f-4afa-a84b-a65a28f5c1a1' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/92.0.4515.131 Safari/537.36' \
-H 'content-type: application/json' \
-H 'origin: https://autoid-ui.forgerock.com' \
-H 'sec-fetch-site: same-origin' \
-H 'sec-fetch-mode: cors' \
-H 'sec-fetch-dest: empty' \
-H 'referer: https://autoid-ui.forgerock.com/jobs' \
-H 'accept-language: en-US,en;q=0.9' \
--data-raw '{
  "branch": "actual",
  "contextId": "ba9cefff-1e06-4cc3-b7d6-d15e2126351c",
  "entityData": [
    {
      "job_name": "ShowCaseCSVIngest",
      "job_type": "ingest",
      "job_parameters": {
        "driverMemory": "2g",
        "driverCores": 3,
        "executorMemory": "3G",
        "executorCores": 6,
        "datasourceId": "2d7a6a76-469c-4035-b312-fb1daf104e98"
      }
    }
  ],
  "indexingRequired": true,
  "tags": {},
  "indexInSync": true
}' \
--compressed \
--insecure

```

POST /jas/job/run

POST /jas/job/run

Run the job definition.

Endpoint

```
/jas/job/run
```

Authorization

Bearer <Token JWT-value> or
<API Key-value>

Body

```
{
  "jobType":"ingest",
  "jobDefinitionName":"ShowCaseCSVIngest"
}
```

Example Request

```
curl 'https://autoid-ui.forgerock.com/jas/job/run' \
-H 'authority: autoid-ui.forgerock.com' \
-H 'sec-ch-ua: "Chromium";v="92", " Not A;Brand";v="99", "Google Chrome";v="92"' \
-H 'accept: application/json, text/plain, /' \
-H 'x-tenant-id: 8700f5cb-eaca-461e-8c2e-245a25f2399d' \
-H 'authorization: 1b7789f0-6c2f-4afa-a84b-a65a28f5c1a1' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36' \
-H 'content-type: application/json' \
-H 'origin: https://autoid-ui.forgerock.com' \
-H 'sec-fetch-site: same-origin' \
-H 'sec-fetch-mode: cors' \
-H 'sec-fetch-dest: empty' \
-H 'referrer: https://autoid-ui.forgerock.com/jobs' \
-H 'accept-language: en-US,en;q=0.9' \
--data-raw '{
  "jobType":"ingest",
  "jobDefinitionName":"ShowCaseCSVIngest"
}' \
--compressed \
--insecure
```

GET /jas/job/status

GET /jas/job/status

Obtain the job's status.

Endpoint

/jas/job/status

Authorization

Bearer <Token JWT-value> or
<API Key-value>

Body

```
{
  "jobType": "anomaly",
  "jobDefinitionName": "ShowCaseCSVAnomaly"
}
```

Example Request

```
curl 'https://autoid-ui.forgerock.com/jas/job/status' \
-H 'authority: autoid-ui.forgerock.com' \
-H 'sec-ch-ua: "Chromium";v="92", " Not A;Brand";v="99", "Google Chrome";v="92"' \
-H 'accept: application/json, text/plain, /' \
-H 'x-tenant-id: 8700f5cb-eaca-461e-8c2e-245a25f2399d' \
-H 'authorization: 1b7789f0-6c2f-4afa-a84b-a65a28f5c1a1' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36' \
-H 'content-type: application/json' \
-H 'origin: https://autoid-ui.forgerock.com' \
-H 'sec-fetch-site: same-origin' \
-H 'sec-fetch-mode: cors' \
-H 'sec-fetch-dest: empty' \
-H 'referrer: https://autoid-ui.forgerock.com/jobs' \
-H 'accept-language: en-US,en;q=0.9' \
--data-raw '{
  "jobType": "anomaly",
  "jobDefinitionName": "ShowCaseCSVAnomaly"
}' \
--compressed \
--insecure
```

GET /jas/job/search

GET /jas/job/search

Search for a job definition.

Endpoint

/jas/job/search

Authorization

Bearer <Token JWT-value> or
<API Key-value>

Body

```
{
  "query": {
    "sort": [
      {
        "job_name.keyword": {
          "order": "asc"
        }
      },
      {
        "metadata.primaryKey.keyword": {
          "order": "desc"
        }
      }
    ],
    "size": 10,
    "track_total_hits": true,
    "query": {
      "match_all": {}
    }
  }
}
```

Example Request

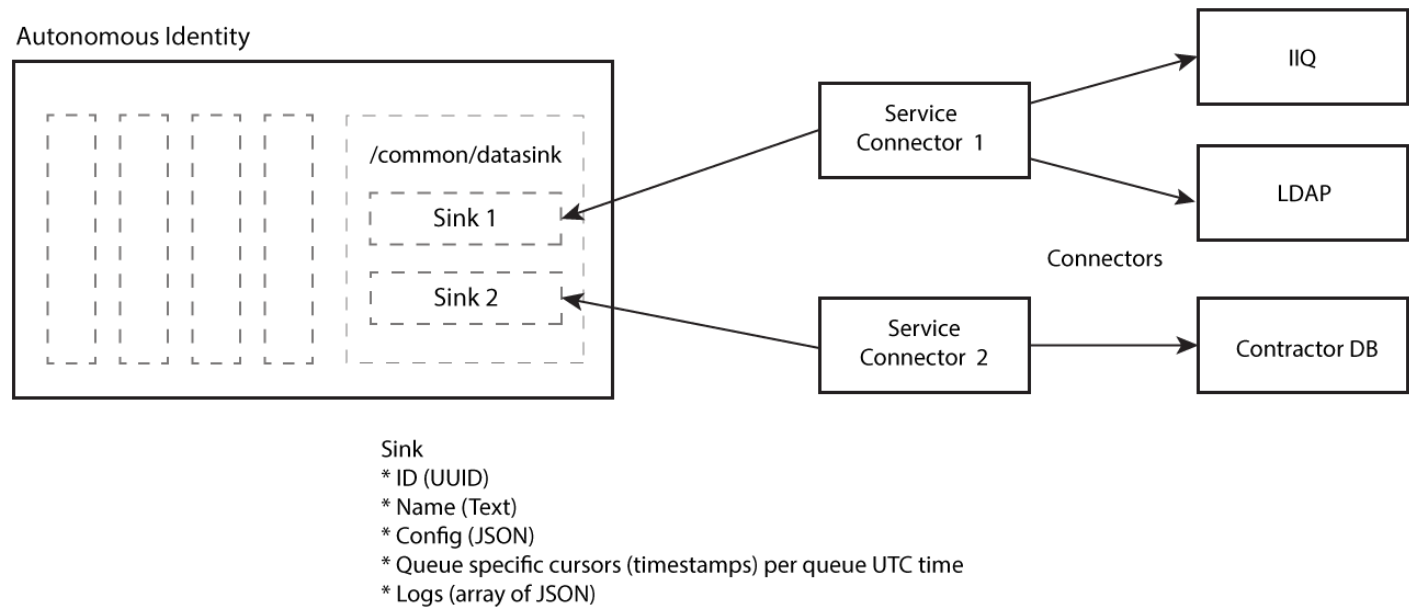
```

curl 'https://autoid-ui.forgerock.com/jas/entity/search/autoid/api/job_definition' \
-H 'authority: autoid-ui.forgerock.com' \
-H 'sec-ch-ua: "Chromium";v="92", " Not A;Brand";v="99", "Google Chrome";v="92"' \
-H 'accept: application/json, text/plain, /' \
-H 'x-tenant-id: 8700f5cb-eaca-461e-8c2e-245a25f2399d' \
-H 'authorization: 1b7789f0-6c2f-4afa-a84b-a65a28f5c1a1' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/92.0.4515.131 Safari/537.36' \
-H 'content-type: application/json' \
-H 'origin: https://autoid-ui.forgerock.com' \
-H 'sec-fetch-site: same-origin' \
-H 'sec-fetch-mode: cors' \
-H 'sec-fetch-dest: empty' \
-H 'referer: https://autoid-ui.forgerock.com/jobs' \
-H 'accept-language: en-US,en;q=0.9' \
--data-raw '{
    "query": {
      "sort": [
        {
          "job_name.keyword": {
            "order": "asc"
          }
        },
        {
          "metadata.primaryKey.keyword": {
            "order": "desc"
          }
        }
      ],
      "size": 10,
      "track_total_hits": true,
      "query": {
        "match_all": {}
      }
    }
  }' \
--compressed \
--insecure

```

Data Sink

Data sink is the downstream consumer of data within PingOne Autonomous Identity for service connectors and is accessible through API endpoints.



The following are PingOne Autonomous Identity datasink endpoints:

POST /api/datasink/create

POST /api/datasink/create

Create a new /common/datasink entity.

Endpoint

```
/api/datasink/create
```

Authorization

```
<Bearer Token JWT-value>
```

JSON Body Parameters

Parameter	Type	Description
id	string	Data sink ID (format: UUID) (required)
name	string	Data sink name (required)
config	object	Free form client-defined connection object
autoCertifyTimestamp	string	Last auto-certification operation timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ)

Parameter	Type	Description
autoRequestTimestamp	string	Last auto-request operation timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ)
certifyTimestamp	string	Last certification operation timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ)
revokeTimestamp	string	Last revoke operation timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ)
rolePublishTimestamp	string	Last role publish timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ)
logs	object	Free form client-defined log object

Example Request

```
curl -k -X POST https://autoid-ui.forgerock.com/api/datasink/create' \
-H 'Authorization: Bearer <token value>' \
-H 'Content-Type: application/json' \
-d '{
  "id": "eaa19702-3806-4ee7-9466-91f0968699d9",
  "name": "Test",
  "config": { "something": 1234 }
}'
```

Example Response (Success)

```
{
  "message": "ok"
}
```

POST /api/datasink/update

POST /api/datasink/update

Update a /common/datasink entity.

Endpoint

```
/api/datasink/update
```

Authorization

```
<Bearer Token JWT-value>
```

JSON Body Parameters

Parameter	Type	Description
id	string	Data sink ID (format: UUID) (required)
name	string	Data sink name (required)
config	object	Free form client-defined connection object
autoCertifyTimestamp	string	Last auto-certification operation timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ)
autoRequestTimestamp	string	Last auto-request operation timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ)
certifyTimestamp	string	Last certification operation timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ)
revokeTimestamp	string	Last revoke operation timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ)
rolePublishTimestamp	string	Last role publish timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ)
logs	object	Free form client-defined log object

Example Request

```
curl -k -X POST https://autoid-ui.forgerock.com/api/datasink/update' \
-H 'Authorization: Bearer <token value>' \
-H 'Content-Type: application/json' \
-d '{
  "id": "eaa19702-3806-4ee7-9466-91f0968699d9",
  "name": "Test3",
  "config": { "something": 5678 }
}'
```

Example Response (Success)

```
{
  "message": ok
}
```

POST /api/datasink/delete

POST /api/datasink/delete

Remove a /common/datasink entity.

Endpoint

```
/api/datasink/delete
```

Authorization

```
<Bearer Token JWT-value>
```

JSON Body Parameters

Parameter	Type	Description
id	string	Data sink ID (format: UUID) (required)

Example Request

```
curl -k -X POST https://autoid-ui.forgerock.com/api/datasink/delete' \
-H 'Authorization: Bearer <token value>' \
-H 'Content-Type: application/json' \
-d '{
  "id": "eaa19702-3806-4ee7-9466-91f0968699d9"
}'
```

Example Response (Success)

```
{
  "message": ok
}
```

POST /api/datasink/update/status/assignments

POST /api/datasink/update/status/assignments

Update data sink status for assignment decisions. Decisions are stored in the /autoid/api/user_access_decisions JAS entity and *entitlement-assignment* Elasticsearch index.

Endpoint

/api/datasink/update/status/assignments

Authorization

<API Key>

JSON Body Parameters

Parameter	Type	Description
assignments	array	Array of assignment objects (properties below) (required)

Base Assignment Object Properties

Parameter	Type	Description
user	string	User ID (required)
entitlements	string array	Array of entitlement IDs (required)
datasink_status	string	Updated data sink status (required)

Example Request

```
curl -k -X POST \
https://autoid-ui.forgerock.com/api/datasink/update/status/assignments \
-H 'X-API-KEY: <api key value>' \
-H 'Content-Type: application/json' \
-d '{
  "assignments": [
    {
      "user": "john.doe",
      "entitlements": [
        "entitlement_1"
      ],
      "datasink_status": "ack"
    },
    {
      "user": "jane.smith",
      "entitlements": [
        "entitlement_1",
        "entitlement_2",
      ],
      "datasink_status": "nack"
    }
  ]
}'
```

Example Response (Success)

```
{
  "message" : ok
}
```

POST /api/datasink/update/status/rules

POST /api/datasink/update/status/rules

Update data sink status for rule decisions. Decisions are stored in the /autoid/api/rule_access_decisions JAS entity and entitlement-assignment Elasticsearch index.

Endpoint

```
/api/datasink/update/status/rules
```

Authorization

```
<API Key>
```

JSON Body Parameters

Parameter	Type	Description
rules	array	Array of rule objects (properties below) (required)

Base Rules Object Properties

Parameter	Type	Description
entitlement	string	Entitlement ID (required)
justification	string array	Array of justifications (required)
datasink_status	string	Updated data sink status (required)

Example Request

```
curl -k -X POST \
https://autoid-ui.forgerock.com/api/datasink/update/status/rules \
-H 'X-API-KEY: <api key value>' \
-H 'Content-Type: application/json' \
-d '{
  "rules": [
    {
      "entitlement": "entitlement_1",
      "justification": [
        "0B_COST_CENTER_OP_TS5",
        "0C_USR_EMP_TYPE_Non-Employee",
        "10_LINE_OF_BUSINESS_Health and Safety",
        "13_USR_DEPARTMENT_NAME_Testing"
      ],
      "datasink_status": "nack"
    }
  ]
}'
```

Example Response (Success)

```
{
  "message": ok
}
```

POST /api/datasink/update/status/roles

POST /api/datasink/update/status/roles

Update data sink status for exported roles.

Endpoint

```
/api/datasink/update/status/roles
```

Authorization

```
<API Key>
```

JSON Body Parameters

Parameter	Type	Description
roles	array	Array of role objects (properties below) (required)

Base Roles Object Properties

Parameter	Type	Description
role_id	string	Role ID (required)
status	string	Role status (draft, candidate, active) (required)
datasink_status	string	Updated data sink status (required)

Example Request

```
curl -k -X POST \
/https://autoid-ui.forgerock.com/api/datasink/update/status/roles \
-H 'X-API-KEY: <api key value>' \
-H 'Content-Type: application/json' \
-d '{
  "roles": [
    {
      "role_id": "027d9a1d-9a2f-488a-8ab2-adf404e0aecb",
      "status": "draft",
      "datasink_status": "nack"
    }
  ]
}'
```

Example Response (Success)

```
{
  "message": ok
}
```

POST /api/datasink/query

POST /api/datasink/query

Query data sink entities. Optional filtering can be applied as a JSON request body outlined below.

Endpoint

```
/api/datasink/query
```

Authorization

```
<API Key>
```

JSON Body Parameters

Parameter	Type	Description
id	string	Data sink ID (format: UUID)
name	string	Data sink name
autoCertifyTimestamp	object	Last auto-certify operation timestamp filter object (available properties below)
autoRequestTimestamp	object	Last auto-request operation timestamp filter object (available properties below)
certifyTimestamp	object	Last certify operation timestamp filter object (available properties below)
revokeTimestamp	object	Last revoke operation timestamp filter object (available properties below)
rolePublishTimestamp	object	Last role publish timestamp filter object (available properties below)

timestampThresholds Object Properties

Parameter	Type	Description
gt	string	Greater than timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ). Cannot be present with gte .
gte	string	Greater than or equal timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ). Cannot be present with gt .
lt	string	Less than timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ). Cannot be present with lte .
lte	string	Less than or equal timestamp (format: yyyy-mm-ddThh:mm:ss.SSSZ). Cannot be present with le .

Example Request

```
curl -k -X POST \
https://autoid-ui.forgerock.com/api/datasink/query \
-H 'X-API-KEY: <api key value>' \
-H 'Content-Type: application/json' \
-d '{
  "name": "IIQ",
  "certifyTimestamp": {
    "gt": "2021-11-19T10:01:19.937Z",
    "lte": "2021-11-20T10:01:19.937Z"
  }
}'
```

Example Response (Success)

```
[
  {
    "id": "b72c15b7-7dcb-44ac-b1d3-162565e360b4",
    "name": "IIQ",
    "certifyTimestamp": "2021-11-19T11:01:19.937Z",
    "auto_request_timestamp": "2021-10-01T10:01:19.937Z"
  },
  {
    "id": "9501810e-1480-4f41-80d4-bc97154fddeb",
    "name": "IIQ",
    "certifyTimestamp": "2021-11-20T09:01:19.937Z",
    "auto_request_timestamp": "2021-10-01T10:01:19.937Z"
  }
]
```

POST /api/datasink/update/timestamp

POST /api/datasink/update/timestamp

Update timestamps for a data sink entity.

Endpoint

/api/datasink/update/timestamp

Authorization

<API Key>

JSON Body Parameters

Parameter	Type	Description
id	string	Data sink ID (format: UUID)
autoCertifyTimestamp	object	Last auto-certify operation timestamp filter object (format: yyyy-mm-ddThh:mm:ss.SSSZ)
autoRequestTimestamp	object	Last auto-request operation timestamp filter object (format: yyyy-mm-ddThh:mm:ss.SSSZ)
certifyTimestamp	object	Last certify operation timestamp filter object (format: yyyy-mm-ddThh:mm:ss.SSSZ)
revokeTimestamp	object	Last revoke operation timestamp filter object (format: yyyy-mm-ddThh:mm:ss.SSSZ)
rolePublishTimestamp	object	Last role publish timestamp filter object (format: yyyy-mm-ddThh:mm:ss.SSSZ)

Example Request

```
curl -k -X POST \
https://autoid-ui.forgerock.com/api/datasink/update/timestamp \
-H 'X-API-KEY: <api key value>' \
-H 'Content-Type: application/json' \
-d '{
  "id": "87e341c0-c1aa-4b0e-9ae5-1384bb6de8fc",
  "certifyTimestamp": "2021-11-19T10:01:19.937Z",
  "revokeTimestamp": "2021-11-19T10:01:19.937Z"
}'
```

Example Response (Success)

```
{
  "message": "ok"
}
```

POST /api/datasink/query/logs

POST /api/datasink/query/logs

Query data sink logs. Optional filtering can be applied as a JSON request body outlined below.

Endpoint

/api/datasink/query/logs

Authorization

<API Key>

JSON Body Parameters

Parameter	Type	Description
id	string	Data sink ID (format: UUID)
name	string	Data sink name

Example Request

```
curl -k -X POST \
https://autoid-ui.forgerock.com/api/datasink/query/logs \
-H 'X-API-KEY: <api key value>' \
-H 'Content-Type: application/json' \
-d '{
  "name": "IIQ"
}'
```

Example Response (Success)

```
[
  {
    "id": "5f8c48c5-8f70-43a0-a9a6-61d1b017dac7",
    "name": "IIQ",
    "certify_timestamp": "2021-10-01T10:01:19.937Z",
    "revokeTimestamp": "2021-10-23T10:01:19.937Z",
    "logs": [
      {
        "message": "log 1"
      },
      {
        "message": "log 2"
      }
    ]
  },
  {
    "id": "9c68c658-2d7d-487c-a0ce-6d9cdcc7eaf7",
    "name": "IIQ",
    "certify_timestamp": "2021-10-01T10:01:19.937Z",
    "revokeTimestamp": "2021-10-01T10:01:19.937Z",
    "logs": [
      {
        "message": "log 1"
      },
      {
        "message": "log 2"
      }
    ]
  }
]
```

POST /api/datasink/update/logs

POST /api/datasink/update/logs

Update data sink logs.

Endpoint

/api/datasink/update/logs

Authorization

<API Key>

JSON Body Parameters

Parameter	Type	Description
id	string	Data sink ID (format: UUID) (required)
logs	object	JSON logs object (required)

Example Request

```
curl -k -X POST \
https://autoid-ui.forgerock.com/api/datasink/update/logs \
-H 'X-API-KEY: <api key value>' \
-H 'Content-Type: application/json' \
-d '{
  "id": "87e341c0-c1aa-4b0e-9ae5-1384bb6de8fc",
  "logs": {
    "logs": [
      {
        "created": "2021-11-19T09:01:19.937Z",
        "type": "INFO",
        "message": "transaction started"
      },
      {
        "created": "2021-11-19T10:01:19.937Z",
        "type": "INFO",
        "message": "transaction successful"
      }
    ]
  }
}'
```

Example Response (Success)

```
{
  "message": "ok"
}
```