Install Guide

This guide shows you how to install and deploy Autonomous Identity for intelligent entitlements management in production environments. For hardware and software requirements, see the Release notes.

ForgeRock® Autonomous Identity is an entitlements 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. 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. Autonomous Identity eliminates the manual re-certification of entitlements and provides a centralized, transparent, and contextual view of all access points within your company.

Quick Start

Architecture in Brief

Learn about the Autonomous Identity architecture.

Deployment Architectures

Learn about the different deployment architectures.

Install a Single-Node Deployment

Install a single-node Autonomous Identity installation.

Install a Single-Node Air-Gapped

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

Install a Multi-Node

Install a multi-node Autonomous Identity installation.

Install a Multi-Node Air-Gapped

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

Upgrade Autonomous Identity Upgrade to the latest version of Autonomous Identity.	**Appendix: Ports** Learn about the Autonomous Identity ports.	**Appendix: vars.yml** Learn about the main deployment configuration file.

For a description of the Autonomous Identity UI console, see the Autonomous Identity Users Guide.

Features

Autonomous Identity provides the following features:

- **Broad Support for Major Identity Governance and Administration (IGA) Providers**. 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**. Autonomous Identity deploys using a microservices architecture, either on-prem, cloud, or hybrid-cloud environments. Autonomous Identity's architecture supports scalable reads and writes for efficient processing.

- **Powerful UI dashboard**. 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**. Autonomous Identity's analytics engine is capable of processing millions of access points. 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**. Autonomous Identity lets administrators discover and extend the schema.

- **UI-Driven Data Ingestion and Mappings**. 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**. Autonomous Identity supports both Apache Cassandra and MongoDB databases. Both are highly distributed databases with wide usage throughout the industry.

- **Improved Search Support**. Autonomous Identity now incorporates Open Distro for Elasticsearch, a distributed, open-source search engine based on Lucene, to improve database search results and performance.

Architecture in Brief

Autonomous Identity's flexible architecture can deploy in any number of ways: single-node or multi-node configurations across on-prem, cloud, hybrid, or multi-cloud environments. The Autonomous Identity architecture has a simple three-layer conceptual model:

- **Application Layer**. 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:

  - **Autonomous Identity UI**. Autonomous Identity supports a dynamic UI that displays the entitlements, confidence scores, and recommendations.

  - **Autonomous Identity API**. Autonomous Identity provides an API that can access endpoints using REST. This allows easy scripting and programming for your system.

  - **Self-Service Tool**. The self-service tool lets users reset their Autonomous Identity passwords.

  - **Backend Repository**. The backend repository stores Autonomous Identity user information. To interface with the backend repository, you can use the `phpldapadmin` tool to enter and manage users.

  - **Configuration Service**. Autonomous Identity supports a configuration service that allows you to set parameters for your system and processes.

  - **Nginx**. Nginx is a popular HTTP server and reverse proxy for routing HTTPS traffic.

  - **Hashicorp Consul**. Consul is a third-party system for service discovery and configuration.

  - **Apache Livy**. Autonomous Identity supports Apache Livy to provide a RESTful interface to Apache Spark.

  - **Java API Service**. Autonomous Identity supports the Java API Service for RESTful interface to the Cassandra or MongoDB database.

- **Data Layer**. 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 REST interface to the databases.

Autonomous Identity also implements Open Distro for Elasticsearch and Kibana to improve search performance for its entitlement data. Elastic Persistent Search supports scalable writes and reads.

- **Analytics and Administration Layer**. 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. Autonomous Identity runs the analytics jobs directly from the Spark master over Apache Livy REST interface.

Figure 1: A Simple Conceptual Image of the Autonomous Identity Architecture
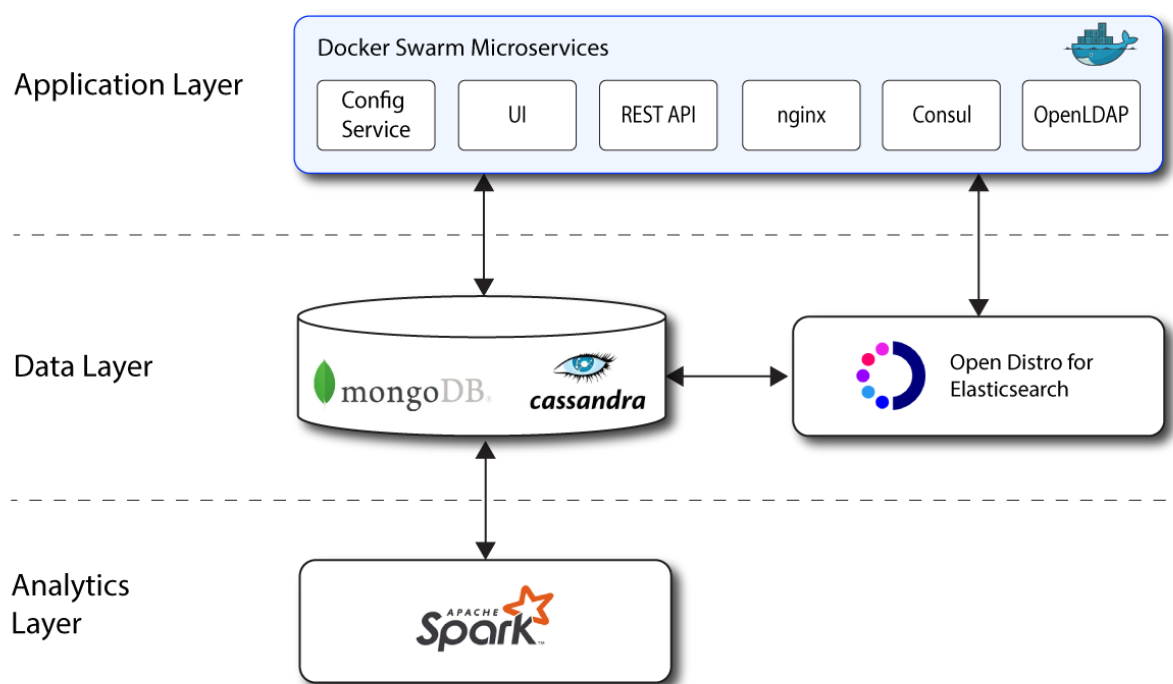
Deployment Architectures

To simplify your deployments, ForgeRock provides a deployer script to install Autonomous Identity on a target node. The deployer pulls in images from the ForgeRock Google Cloud Repository (gcr.io) and uses it to deploy the the microservices, analytics machine, and database for Autonomous Identity on a target machine. The target machine only requires the base operating system, CentOS 7 or later.

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

- **Single-Node Target Deployment**. Deploy 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, see Install a Single-Node Deployment.

Figure 2: A single-node target deployment.
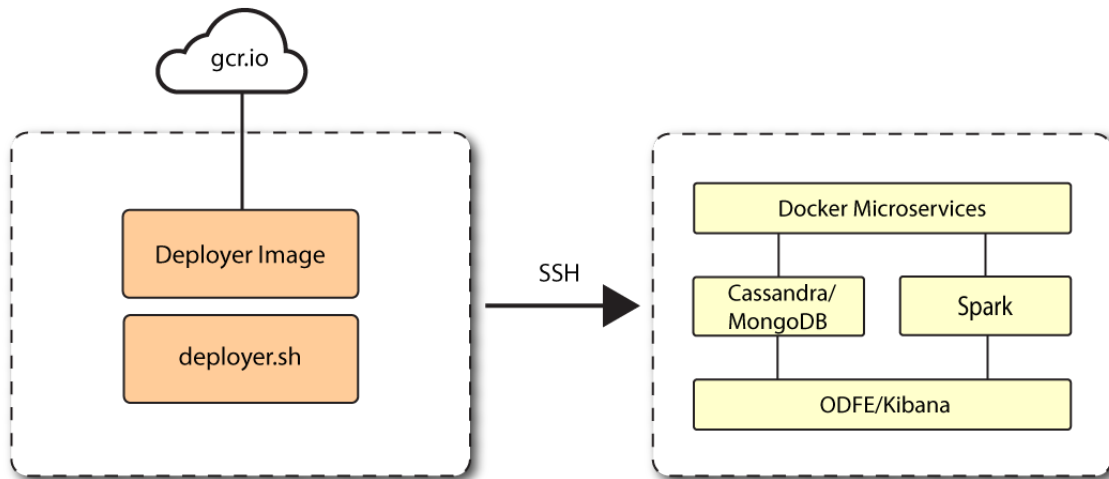
- **Single-Node Air-Gapped Target Deployment**. Deploy Autonomous Identity on a single-node target machine that resides in an air-gapped environment. In an air-gapped environment, the target machine is placed in an enhanced security environment where external Internet access is not available. You transfer the deployer and image to the target machine using media, such as a USB stick or portable drive. Then, run the deployment within the air-gapped environment. For installation instruction, see Install a Single-Node Air-Gapped.

Figure 3: An air-gapped environment.
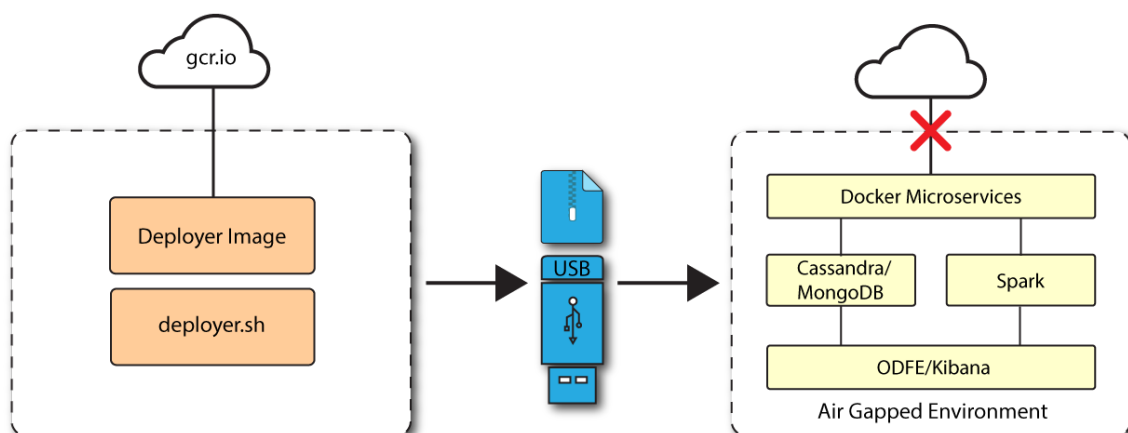
- **Multi-Node Deployment**. Deploy Autonomous Identity on multi-node deployment to distribute the process load on the servers. For installation instruction, see Install a Multi-Node.

Figure 4: A multi-node target environment.
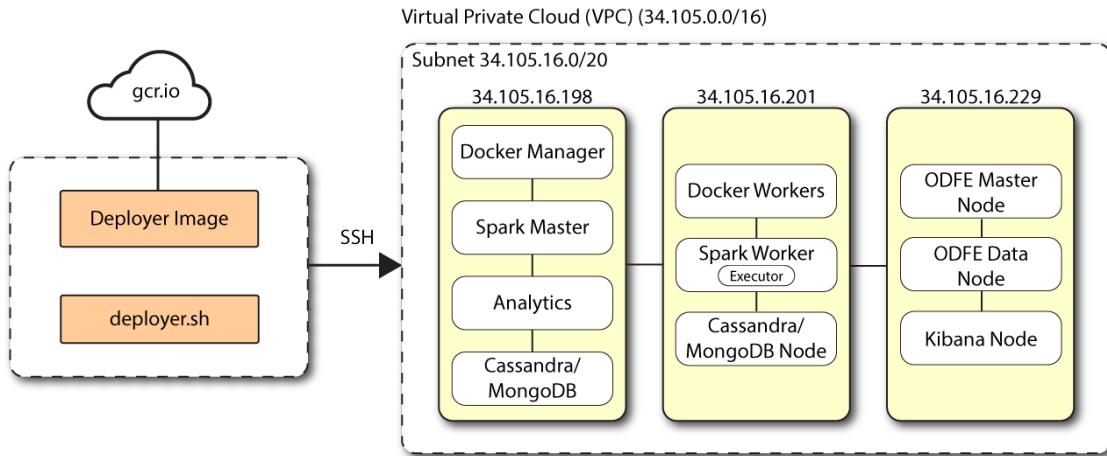
- **Multi-Node Air-Gapped Deployment**. Deploy Autonomous Identity a multi-node configuration in an air-gapped network. The multinode network has no external Internet connection. For installation instruction, see Install a Multi-Node Air-Gapped.

Figure 5: A multi-node air-gapped target environment.
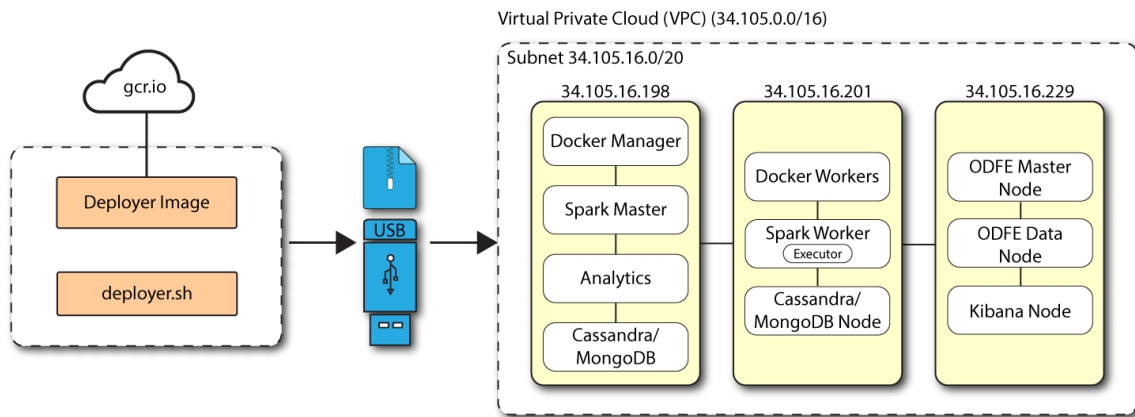
Install a Single Node Deployment

This chapter presents instructions on deploying Autonomous Identity in a single-target machine that has Internet connectivity. ForgeRock provides a deployer script that pulls a Docker container image from ForgeRock's Google Cloud Registry (gcr.io) repository. The image contains the microservices, analytics, and backend databases needed for the system.

This installation assumes that you set up the deployer script on a separate machine from the target. This lets you launch a build from a laptop or local server.

Figure 6: A single-node target deployment.

Prerequisites

Let's deploy Autonomous Identity on a single-node target on CentOS 7. The following are prerequisites:

- **Operating System**. The target machine requires CentOS 7. The deployer machine can use any operating system as long as Docker is installed. For this guide, we use CentOS 7 as its base operating system.

- **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**. 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 (gcr.io). The registry key is only available to ForgeRock Autonomous Identity customers. For specific instructions on obtaining the registry key, see 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
```

Set Up the Target Machine

Autonomous Identity is configured on a target machine. Make sure you have sufficient storage for your particular deployment. For more information on sizing considerations, see Deployment Planning Guide.

1. The install assumes that you have CentOS 7 as your operating system. Check your CentOS 7 version.

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

2. Set the user for the target machine to a username of your choice. For example, `autoid`.

```
$ sudo adduser autoid
```

3. Set the password for the user you created in the previous step.

```
$ sudo passwd autoid
```

4. Configure the user for passwordless sudo.

```
$ echo "autoid    ALL=(ALL)  NOPASSWD:ALL" | sudo tee
/etc/sudoers.d/autoid
```

5. Add administrator privileges to the user.

```
$ sudo usermod -aG wheel autoid
```

6. Change to the user account.

```
$ su - autoid
```

7. 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 -y install yum-utils
```

Set Up the Deployer Machine

Set up another machine as a deployer node. You can use any OS-based machine for the deployer as long as it has Docker installed. For this example, we use CentOS 7.

1. The install assumes that you have CentOS 7 as your operating system. Check your CentOS 7 version.

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

2. Set the user for the target machine to a username of your choice. For example, `autoid`.

```
# sudo adduser autoid
```

3. Set the password for the user you created in the previous step.

```
$ sudo passwd autoid
```

4. Configure the user for passwordless sudo.

```
$ echo "autoid    ALL=(ALL)  NOPASSWD:ALL" | sudo tee
/etc/sudoers.d/autoid
```

5. Add administrator privileges to the user.

```
$ sudo usermod -aG wheel autoid
```

6. Change to the user account.

```
$ su - autoid
```

7. 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 -y install yum-utils
```

8. Create the installation directory. Note that you can use any install directory for your system as long as your run the `deployer.sh` script from there. Also, the disk volume where you have the install directory must have at least 8GB free space for the installation.

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

Install Docker on the Deployer Machine

Install Docker on the deployer machine. We run commands from this machine to install Autonomous Identity on the target machine. In this example, we use CentOS 7.

1. On the target machine, set up the Docker-CE repository.

```
$ sudo yum-config-manager \
    --add-repo
https://download.docker.com/linux/centos/docker-ce.repo
```

2. Install the latest version of the Docker CE, the command-line interface, and containerd.io, a containerized website.

```
$ sudo yum -y install  docker-ce docker-ce-cli
containerd.io
```

3. Enable Docker to start at boot.

```
$ sudo systemctl enable docker
```

4. Start Docker.

```
$ sudo systemctl start docker
```

5. Check that Docker is running.

```
$ systemctl status docker
```

6. Add the user to the Docker group.

```
$ sudo usermod -aG docker ${USER}
```

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. 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` .

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` file.

> **NOTE**
>
> If your target system does not have an `/authorized_keys` directory, create it using **mkdir -p ~/.ssh/authorized_keys**.

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

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 Mar 23 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 Autonomous Identity

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

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

2. Log in to the ForgeRock Google Cloud Registry (gcr.io) using the registry key. The registry key is only available to ForgeRock 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
```

3. 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:2021.3.0 create-
template


…
d6c7c6f3303e: Pull complete
Digest:
sha256:15225be65417f8bfb111adea37c83eb5e0d87140ed498bfb624
a358f43fbbf
Status: Downloaded newer image for gcr.io/forgerock-
autoid/autoid/dev-
compact/deployer@sha256:15225be65417f8bfb111a
dea37c83eb5e0d87140ed498bfb624a358f43fbbf
Config template is copied to host machine directory mapped
to /config
```

4. Make the script executable.

```
$ chmod +x deployer.sh
```

5. To see the list of commands, enter `deployer.sh`.

```
$ ./deployer.sh

Usage: deployer <command>
```

```
Commands:
   create-template
   download-images
   import-deployer
   encrypt-vault
   decrypt-vault
   run
   create-tar
   install-docker
   install-dbutils
   upgrade
```

Configure Autonomous Identity

The `create-template` command from the previous section creates a number of configuration files, required for the deployment: `ansible.cfg`, `vars.yml`, `hosts`, and `vault.yml`.

> **NOTE**
>
> If you are running a deployment for evaluation, you can minimally set the private IP address mapping in the `vars.yml` file in step 2, edit the `hosts` file in step 3, jump to step 6 to download the images, and then run the deployer in step 7.

1. On the deployer machine, open a text editor and edit the `~/autoid-config/ansible.cfg` to set up the remote user and SSH private key file location on the target node. Make sure that the `remote_user` exists on the target node and that the deployer machine can ssh to the target node as the user specified in the `id_rsa` file. In most cases, you can use the default values.

   ```
   [defaults]
   host_key_checking = False
   remote_user = autoid
   private_key_file = id_rsa
   ```

2. On the deployer machine, open a text editor and edit the `~/autoid-config/vars.yml` file to configure specific settings for your deployment:

   - **Domain and Target Environment**. Set the domain name and target environment specific to your deployment 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

Autonomous Identity URL will be: `https://autoid-ui.forgerock.com` .
For this example, we use the default values.

```
domain_name: forgerock.com
target_environment: autoid
```

If you change the domain name and target environment, you need to also
change the certificates to reflect the new changes. For more information,
see Customize the Domain and Namespace.

- **Analytics Data Directory and Analytics Configuration Direction**. Although
  rarely necessary for a single node deployment, you can change the
  analytics and analytics configuration mount directories by editing the
  properties in the `~/autoid-config/vars.yml` file.

```
analytics_data_dir: /data
analytics_conf_dif: /data/conf
```

- **Dark Theme Mode**. Optional. By default, the Autonomous Identity UI
  displays its pages with a light background. You can set a dark theme mode
  by setting the `enable_dark_theme` property to `true` .

- **Database Type**. By default, Apache Cassandra is set as the default database
  for Autonomous Identity. For MongoDB, set the `db_driver_type:` to
  `mongo` .

```
db_driver_type: mongo
```

- **Private IP Address Mapping**. If your external and internal IP addresses are
  different, for example, when deploying the target host in a cloud, define a
  mapping between the external IP address and the private IP address in the
  `~/autoid-config/vars.yml` file.

  If your external and internal IP addresses are the same, you can skip this
  step.

  On the deployer node, add the `private_ip_address_mapping` property in
  the `~/autoid-config/vars.yml` file. You can look up the private IP on the
  cloud console, or run **sudo ifconfig** on the target host. Make sure the
  values are within double quotes. The key should not be in double quotes
  and should have two spaces preceding the IP address.

```
private_ip_address_mapping:
   external_ip:  "internal_ip"
```

For example:

```
private_ip_address_mapping:
  34.70.190.144: "10.128.0.71"
```

- ○ **Authentication Option**. Autonomous Identity provides a single sign-on (SSO) feature that you can configure with an OIDC identity provider. For more information, see Set Up SSO.

- ○ **Access Log**. By default, the access log is enabled. If you want to disable the access log, set the `access_log_enabled` variable to "false".

- ○ **JWT Expiry and Secret File**. Optional. By default, the session JWT is set at 30 minutes. To change this value, set the `jwt_expiry` property to a different value.

```
jwt_expiry: "30 minutes"
jwt_secret_file: "{{install path}}"/jwt/secret.txt"
jwt_audience: "http://my.service"
oidc_jwks_url: "na"
```

- ○ **Elasticsearch Heap Size**. Optional. The default heap size for Elasticsearch is 1GB, which may be small for production. For production deployments, uncomment the option and specify `2G` or `3G`.

```
#elastic_heap_size: 1g   # sets the heap size
(1g|2g|3g) for the Elastic Servers
```

- ○ **Java API Service**. Optional. Set the Java API Service (JAS) properties for the deployment: authentication, maximum memory, directory for attribute mappings data source entities:

```
jas:
  auth_enabled: true
  max_memory: 2048M
  mapping_entity_type: /common/mappings
  datasource_entity_type: /common/datasources
```

3. Open a text editor and enter the target host's public IP addresses in the `~/autoid-config/hosts` file. Make sure the target machine's external IP address is accessible from the deployer machine. NOTE: `[notebook]` is not used in Autonomous Identity.

▼ *Click to See a Host File for Cassandra Deployments*

If you configured Cassandra as your database, the `~/autoid-config/hosts` file is as follows for single-node target deployments:

```
[docker-managers]
34.70.190.144

[docker-workers]
34.70.190.144

[docker:children]
docker-managers
docker-workers

[cassandra-seeds]
34.70.190.144

[cassandra-workers]
34.70.190.144

[spark-master]
34.70.190.144

[spark-workers]
34.70.190.144

[mongo_master]

[mongo_replicas]

[mongo:children]
mongo_replicas
mongo_master

# ELastic Nodes
[odfe-master-node]
34.70.190.144

[odfe-data-nodes]
34.70.190.144

[kibana-node]
34.70.190.144

[notebook]
#ip#
```

▼ *Click to See a Host File for MongoDB Deployments*

If you configured MongoDB as your database, the `~/autoid-config/hosts` file is as follows for single-node target deployments:

```
[docker-managers]
34.70.190.144

[docker-workers]
34.70.190.144

[docker:children]
docker-managers
docker-workers

[cassandra-seeds]

[cassandra-workers]

[spark-master]
34.70.190.144

[spark-workers]
34.70.190.144

[mongo_master]
34.70.190.144   mongodb_master=True

[mongo_replicas]
34.70.190.144

[mongo:children]
mongo_replicas
mongo_master

# ELastic Nodes
[odfe-master-node]
34.70.190.144

[odfe-data-nodes]
34.70.190.144

[kibana-node]
34.70.190.144

[notebook]
#ip#
```

4. Open a text editor and set the Autonomous Identity passwords for the configuration service, LDAP backend, and Cassandra database. The vault passwords file is located at `~/autoid-config/vault.yml`.

> **WARNING**
>
> Do not include special characters & or $ in `vault.yml` passwords as it will result in a failed deployer process.

```
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
```

5. Encrypt the vault file that stores the Autonomous Identity passwords, located at `~/autoid-config/vault.yml` . The encrypted passwords will be saved to `/config/.autoid_vault_password` . The `/config/` mount is internal to the deployer container.

```
$ ./deployer.sh encrypt-vault
```

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

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

Make sure you have no unreachable or failed processes before proceeding to the next step.

```
PLAY RECAP
*****************************************************
```

```
   *******************
   localhost   : ok=20  changed=12  unreachable=0  failed=0
   skipped=8  rescued=0  ignored=0
```

7. Run the deployment. The command installs the packages, and starts the microservices and the analytics service. Make sure you have no failed processes before proceeding to the next step.

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

Make sure you have no unreachable or failed processes before proceeding to the next step.

```
PLAY RECAP
**********************************************************
************************
34.70.190.144   : ok=643  changed=319  unreachable=0
failed=0  skipped=79  rescued=0  ignored=1
localhost        : ok=34   changed=14   unreachable=0
failed=0  skipped=4   rescued=0  ignored=0
```

Resolve Hostname

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

Resolve the hostname:

1. Configure your DNS servers to access Autonomous Identity dashboard and self-service applications on the target node. The following domain names must resolve to the IP address of the target node: `<target-environment>-ui.<domain-name>` and `<target-environment>-selfservice.<domain-name>`.

2. If DNS cannot resolve target node hostname, edit it locally on the machine that you want to access Autonomous Identity using a browser. Open a text editor and add an entry in the `/etc/hosts` file 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  myid-selfservice.abc.com
    etc.
```

For more information on customizing your domain name, see <u>Customize
Domains</u>.

Access the Dashboard

Access the 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: Welcome123
```

Check Apache Cassandra

Check Cassandra:

1. On the target node, check the status of Apache Cassandra.

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

2. An example output is as follows:

```
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
++--++  Address        Load        Tokens        Owns (effective)
```

```
Host ID                              Rack
UN  34.70.190.144 1.33 MiB   256          100.0%
a10a91a4-96e83dd-85a2-4f90d19224d9  rack1
++--++
```

Check MongoDB

Check the status of MongoDB:

1. On the target node, check the status of MongoDB.

   ```
   $ 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
   ```

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

   ▼ *Click to See an Example of the Spark Dashboard*

```
                                                        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
  [IMG] 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, see Set Entity Definitions.

Install a Single Node Air-Gapped Deployment

This chapter presents instructions on deploying 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 media, such as USB drive or drive, and copied to the air-gapped target machine.

Figure 7: A single-node air-gapped target deployment.
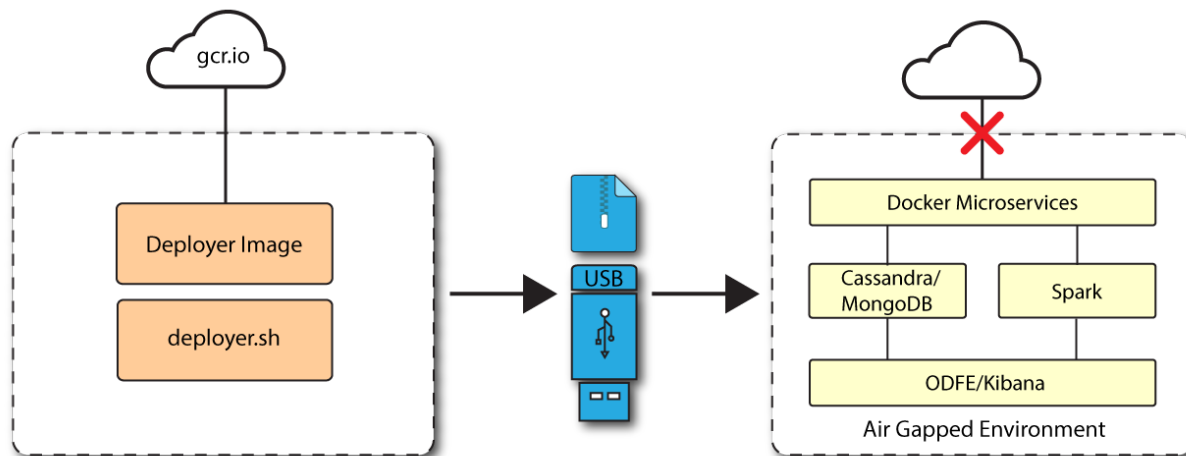
Prerequisites

Let's deploy Autonomous Identity on a single-node air-gapped target on CentOS 7. The following are prerequisites:

- **Operating System**. The target machine requires CentOS 7. The deployer machine can use any operating system as long as Docker is installed. For this guide, we use CentOS 7 as its base operating system.

- **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**. 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 (gcr.io). The registry key is only available to ForgeRock Autonomous Identity customers. For specific instructions on obtaining the registry key, see 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
```

Set Up the Deployer Machine

Set up the deployer on an Internet-connect machine.

1. The install assumes that you have CentOS 7 as your operating system. Check your CentOS 7 version.

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

2. Set the user for the target machine to a username of your choice. For example, `autoid`.

   ```
   $ sudo adduser autoid
   ```

3. Set the password for the user you created in the previous step.

   ```
   $ sudo passwd autoid
   ```

4. Configure the user for passwordless sudo.

   ```
   $ echo "autoid    ALL=(ALL)  NOPASSWD:ALL" | sudo tee /etc/sudoers.d/autoid
   ```

5. Add administrator privileges to the user.

   ```
   $ sudo usermod -aG wheel autoid
   ```

6. Change to the user account.

   ```
   $ su - autoid
   ```

7. 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 -y install yum-utils
   ```

8. Create the installation directory. Note that you can use any install directory for your system as long as your run the **deployer.sh** script from there. Also, the disk volume where you have the install directory must have at least 8GB free space for the installation.

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

Install Docker on the Deployer Machine

1. On the target machine, set up the Docker-CE repository.

   ```
   $ sudo yum-config-manager \
         --add-repo
   https://download.docker.com/linux/centos/docker-ce.repo
   ```

2. Install the latest version of the Docker CE, the command-line interface, and containerd.io, a containerized website.

   ```
   $ sudo yum -y install  docker-ce docker-ce-cli
   containerd.io
   ```

3. Enable Docker to start at boot.

   ```
   $ sudo systemctl enable docker
   ```

4. Start Docker.

   ```
   $ sudo systemctl start docker
   ```

5. Check that Docker is running.

   ```
   $ systemctl status docker
   ```

6. Add the user to the Docker group.

   ```
   $ sudo usermod -aG docker ${USER}
   ```

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 (gcr.io) using the registry key. The registry key is only available to ForgeRock 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
```

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:2021.3.0 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 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 USB drive or portable hard drive.

Install from the Air-Gap Target

Before you begin, make sure you have CentOS 7 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. Install Docker.

```
$ sudo ./deployer.sh install-docker
```

8. Log out and back in.

9. Change to the configuration directory.

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

10. 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: gcr.io/forgerock-autoid/deployer:2021.3.0
```

11. 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
```

Configure Autonomous Identity Air-Gapped

The **create-template** command from the previous section creates a number of
configuration files, required for the deployment: `ansible.cfg`, `vars.yml`, `hosts`, and
`vault.yml`.

> **NOTE**

If you are running a deployment for evaluation, you can minimally set the private IP address mapping in the `vars.yml` file in step 2, edit the `hosts` file in step 3, and jump to step 6 run the deployer.

For air-gapped deployments, you must set the `offline_mode` property to `true` in the `~/autoid-config/vars.yml` file in step 2 below. This is a new change in 2021.3.0 from prior releases.

1. Open a text editor and edit the `~/autoid-config/ansible.cfg` to set up the target machine user and SSH private key file location on the target node. Make sure that the remote_user exists on the target node and that the deployer machine can ssh to the target node as the user specified in the `id_rsa` file.

   ```
   [defaults]
   host_key_checking = False
   remote_user = autoid
   private_key_file = id_rsa
   ```

2. Open a text editor and edit the `~/autoid-config/vars.yml` file to configure specific settings for your deployment:

   - **Domain and Target Environment**. Set the domain name and target environment specific to your deployment 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 Autonomous Identity URL will be: `https://autoid-ui.forgerock.com`. For this example, we use the default values.

     ```
     domain_name: forgerock.com
     target_environment: autoid
     ```

     If you change the domain name and target environment, you need to also change the certificates to reflect the new changes. For more information, see Customize the Domain and Namespace.

   - **Analytics Data Directory and Analytics Configuration Direction**. Although rarely necessary for a single node deployment, you can change the analytics and analytics configuration mount directories by editing the properties in the `~/autoid-config/vars.yml` file.

     ```
     analytics_data_dir: /data
     analytics_conf_dif: /data/conf
     ```

- **Dark Theme Mode**. Optional. By default, the Autonomous Identity UI displays its pages with a light background. You can set a dark theme mode by setting the `enable_dark_theme` property to `true`.

- **Offline Mode**. Set the `offline_mode` to `true` for air-gapped deployments.

  ```
  offline_mode: true
  ```

- **Database Type**. By default, Apache Cassandra is set as the default database for Autonomous Identity. For MongoDB, set the `db_driver_type:` to `mongo`.

  ```
  db_driver_type: mongo
  ```

- **Private IP Address Mapping**. An air-gap deployment has no external IP addresses, but you may still need to define a mapping in the `~/autoid-config/vars.yml` file, if your internal IP address differs from an external IP, say in a virtual air-gapped configuration.

  If your external and internal IP addresses are the same, you can skip this step.

  Add the `private_ip_address_mapping` property in the `~/autoid-config/vars.yml` file. You can look up the private IP on the cloud console, or run **sudo ifconfig** on the target host. Make sure the values are within double quotes. The key should not be in double quotes and should have two spaces preceding the IP address.

  ```
  private_ip_address_mapping:
    external_ip:  "internal_ip"
  ```

  For example:

  ```
  private_ip_address_mapping:
    34.70.190.144:  "10.128.0.71"
  ```

- **Authentication Option**. Autonomous Identity provides a single sign-on (SSO) feature that you can configure with an OIDC identity provider. For more information, see Set Up SSO.

- **Access Log**. By default, the access log is enabled. If you want to disable the access log, set the `access_log_enabled` variable to "false".

- **JWT Expiry and Secret File**. Optional. By default, the session JWT is set at 30 minutes. To change this value, set the `jwt_expiry` property to a different value.

```
jwt_expiry: "30 minutes"
jwt_secret_file: "{{install path}}"/jwt/secret.txt"
jwt_audience: "http://my.service"
oidc_jwks_url: "na"
```

- **Elasticsearch Heap Size**. Optional. The default heap size for Elasticsearch is 1GB, which may be small for production. For production deployments, uncomment the option and specify `2G` or `3G`.

```
#elastic_heap_size: 1g   # sets the heap size
(1g|2g|3g) for the Elastic Servers
```

- **Java API Service**. Optional. Set the Java API Service (JAS) properties for the deployment: authentication, maximum memory, directory for attribute mappings data source entities:

```
jas:
   auth_enabled: true
   max_memory: 2048M
   mapping_entity_type: /common/mappings
   datasource_entity_type: /common/datasources
```

3. Open a text editor and enter the target host's private IP addresses in the `~/autoid-config/hosts` file. The following is an example of the `~/autoid-config/hosts` file: NOTE: `[notebook]` is not used in Autonomous Identity.

   ▼ *Click to See a Host File for Cassandra Deployments*

   If you configured Cassandra as your database, the `~/autoid-config/hosts` file is as follows for single-node air-gapped target deployment:

```
[docker-managers]
10.128.0.34

[docker-workers]
10.128.0.34

[docker:children]
docker-managers
docker-workers

[cassandra-seeds]
10.128.0.34

[cassandra-workers]
```

```
10.128.0.34

[spark-master]
10.128.0.34

[spark-workers]
10.128.0.34

[mongo_master]
#ip# mongodb_master=True

[mongo_replicas]

[mongo:children]
mongo_replicas
mongo_master

# ELastic Nodes
[odfe-master-node]
10.128.0.34

[odfe-data-nodes]
10.128.0.34

[kibana-node]
10.128.0.34

[notebook]
#ip#
```

▼ *Click to See a Host File for MongoDB Deployments*

If you configured MongoDB as your database, the `~/autoid-config/hosts` file
is as follows for single-node air-gapped target deployment:

```
[docker-managers]
10.128.0.34

[docker-workers]
10.128.0.34

[docker:children]
docker-managers
docker-workers
```

```
[cassandra-seeds]

[cassandra-workers]

[spark-master]
10.128.0.34

[spark-workers]
10.128.0.34

[mongo_master]
10.128.0.34   mongodb_master=True

[mongo_replicas]
10.128.0.34

[mongo:children]
mongo_replicas
mongo_master

# ELastic Nodes
[odfe-master-node]
10.128.0.34

[odfe-data-nodes]
10.128.0.34

[kibana-node]
10.128.0.34

[notebook]
#ip#
```

4. Set the Autonomous Identity passwords, located at ~/autoid-config/vault.yml.

> **WARNING**
>
> Do not include special characters & or $ in vault.yml passwords as it will result in a failed deployer process.

```
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
```

5. Encrypt the vault file that stores the Autonomous Identity passwords, located at
   `~/autoid-config/vault.yml` . The encrypted passwords will be saved to
   `/config/.autoid_vault_password` . The `/config/` mount is internal to the
   deployer container.

   ```
   $ ./deployer.sh encrypt-vault
   ```

6. Run the deployment.

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

Resolve Hostname

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

Resolve the hostname:

1. Configure your DNS servers to access Autonomous Identity dashboard and self-
   service applications on the target node. The following domain names must
   resolve to the IP address of the target node: `<target-environment>-ui.`
   `<domain-name>` and `<target-environment>-selfservice.<domain-name>` .

2. If DNS cannot resolve target node hostname, edit it locally on the machine that
   you want to access Autonomous Identity using a browser. Open a text editor
   and add an entry in the `/etc/hosts` file 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  myid-selfservice.abc.com
etc.
```

For more information on customizing your domain name, see Customize
Domains.

Access the Dashboard

Access the 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: Welcome123
```

Check Apache Cassandra

Check Cassandra:

1. On the target node, check the status of Apache Cassandra.

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

2. An example output is as follows:

```
Datacenter: datacenter1
=======================
Status=Up/Down
```

```
|/ State=Normal/Leaving/Joining/Moving
++--++  Address         Load         Tokens         Owns (effective)
Host ID                              Rack
UN  34.70.190.144 1.33 MiB    256          100.0%
a10a91a4-96e83dd-85a2-4f90d19224d9   rack1
++--++
```

Check MongoDB

Check the status of MongoDB:

1. On the target node, check the status of MongoDB.

   ```
   $ 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
   ```

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

   ▼ *Click to See an Example of the Spark Dashboard*

```
                                                          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
  [IMG] 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, see Set Entity Definitions.

Install a Multi-Node Deployment

This chapter presents instructions on deploying Autonomous Identity in a multi-node target deployment that has Internet connectivity. ForgeRock provides a deployer script that pulls a Docker container image from ForgeRock's Google Cloud Registry (gcr.io) repository. The image contains the microservices, analytics, and backend databases needed for the system.

This installation assumes that you set up the deployer on a separate machine from the target.

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, which the deployer uses to set up the overlay network for your multinode system.

Figure 8: A multi-node deployment.
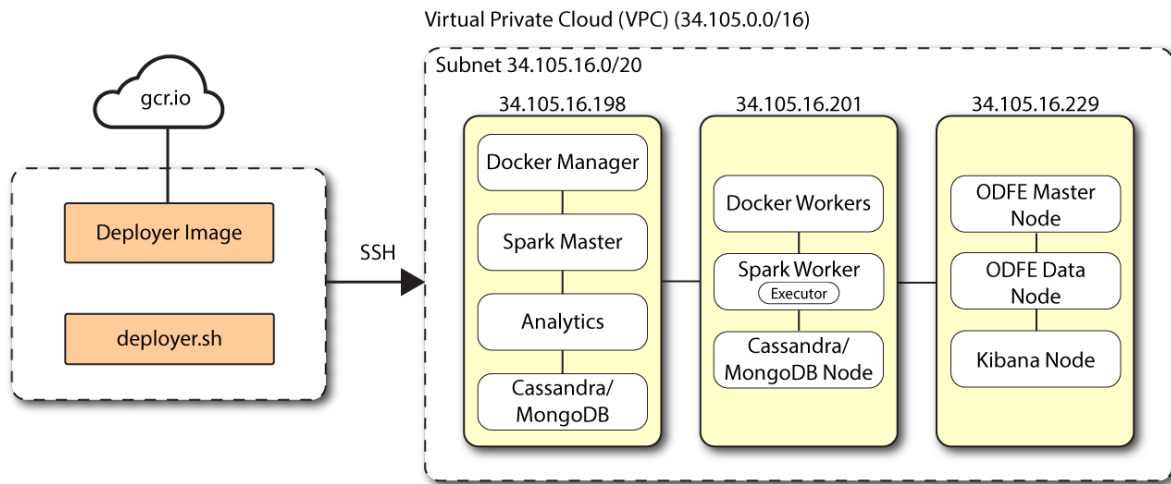
Prerequisites

Let's deploy Autonomous Identity on a multi-node target on CentOS 7. The following are prerequisites:

- **Operating System**. The target machine requires CentOS 7. The deployer machine can use any operating system as long as Docker is installed. For this guide, we use CentOS 7 as its base operating system.

- **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.

- **Subnet Requirements**. We recommend deploying your multinode instances 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**. Autonomous Identity provides a Docker image that creates a `deployer.sh` script that downloads and installs the images necessary. To download the deployment images, you must first obtain a registry key to log into the ForgeRock Google Cloud Registry (gcr.io). The registry key is only available to ForgeRock Autonomous Identity customers. For specific instructions on obtaining the registry key, see How To Configure Service Credentials (Push Auth, Docker) in Backstage.

- **Filesystem Requirements**. Autonomous Identity requires a shared filesystem accessible from the Spark master, 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_dif: /data/conf
  ```

- **Architecture Requirements**. Make sure that the analytics server is on the same node as the Spark master.

- **Database Requirements**. Decide which database you are using: Apache Cassandra or MongoDB. The configuration procedure is slightly different for each database.

- **Deployment Best-Practice**. For best performance, dedicate a separate node to Elasticsearch, data nodes, and Kibana.

- **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
  ```

Set Up the Target Nodes

Make sure you have sufficient storage for your particular deployment. For more information on sizing considerations, see Deployment Planning Guide.

For each target node, run the following commands.

1. The install assumes that you have CentOS 7 as your operating system. Check your CentOS 7 version.

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

2. Set the user for the target machine to a username of your choice. For example, `autoid`.

   ```
   $ sudo adduser autoid
   ```

3. Set the password for the user you created in the previous step.

```
$ sudo passwd autoid
```

4. Configure the user for passwordless sudo.

```
$ echo "autoid  ALL=(ALL)  NOPASSWD:ALL" | sudo tee
/etc/sudoers.d/autoid
```

5. Add administrator privileges to the user.

```
$ sudo usermod -aG wheel autoid
```

6. Change to the user account.

```
$ su - autoid
```

7. Set up a shared directory on your target servers. The method depends on your machine configuration. Make sure the deployer can write to the shared directory.

Set Up the Deployer Machine

Set up another machine as a deployer node. You can use any OS-based machine for the deployer as long as it has Docker installed. For this example, we use CentOS 7.

1. The install assumes that you have CentOS 7 as your operating system. Check your CentOS 7 version.

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

2. Set the user for the target machine to a username of your choice. For example, `autoid`.

```
$ sudo adduser autoid
```

3. Set the password for the user you created in the previous step.

```
$ sudo passwd autoid
```

4. Configure the user for passwordless sudo.

```
$ echo "autoid  ALL=(ALL)  NOPASSWD:ALL" | sudo tee
/etc/sudoers.d/autoid
```

5. Add administrator privileges to the user.

```
$ sudo usermod -aG wheel autoid
```

6. Change to the user account.

```
$ su - autoid
```

7. 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
```

8. Create the installation directory. Note that you can use any install directory for your system as long as your run the `deployer.sh` script from there. Also, the disk volume where you have the install directory must have at least 8GB free space for the installation.

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

Install Docker on the Deployer Machine

Install Docker on the deployer machine. We run commands from this machine to install Autonomous Identity on the target machine. In this example, we use CentOS 7.

1. On the target machine, set up the Docker-CE repository.

```
$ sudo yum-config-manager \
      --add-repo
https://download.docker.com/linux/centos/docker-ce.repo
```

2. Install the latest version of the Docker CE, the command-line interface, and containerd.io, a containerized website.

```
$ sudo yum install -y docker-ce docker-ce-cli
containerd.io
```

3. Enable Docker to start at boot.

```
$ sudo systemctl enable docker
```

4. Start Docker.

```
$ sudo systemctl start docker
```

5. Check that Docker is running.

```
$ systemctl status docker
```

6. Add the user to the Docker group.

```
$ sudo usermod -aG docker ${USER}
```

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. 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` .

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 target machine's `~/.ssh/authorized_keys` file.

> **NOTE**
>
> If your target system does not have an `/authorized_keys` directory, create it using `mkdir -p ~/.ssh/authorized_keys`.

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

```
$ ssh-copy-id -i id_rsa.pub autoid@34.105.15.198
```

```
$ ssh-copy-id -i id_rsa.pub autoid@34.105.15.201
```

```
$ ssh-copy-id -i id_rsa.pub autoid@34.105.15.229
```

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.

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

   - SSH to first node:

     ```
     $ ssh autoid@34.105.15.198

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

     Set the privileges.

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

     Enter Exit to end your SSH session.

   - SSH to the second node:

     ```
     $ ssh autoid@34.105.15.201

     Last login: Sat Oct  3 03:06:40 2020
     ```

     Set the privileges.

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

     Enter Exit to end your SSH session.

   - SSH to the third node:

     ```
     $ ssh autoid@34.105.15.229
     ```

```
Last login: Sat Oct  3 03:10:40 2020
```

Set the privileges.

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

Enter Exit to end your SSH session.

Install Autonomous Identity

Before you begin, make sure you have CentOS 7 installed on your target machine.

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

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

2. Log in to the ForgeRock Google Cloud Registry (gcr.io) using the registry key. The registry key is only available to ForgeRock 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
   ```

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:2021.3.0 create-
   template
   ```

4. Make the script executable.

   ```
   $ chmod +x deployer.sh
   ```

5. To see the list of commands, enter `deployer.sh` .

```
$ ./deployer.sh

Usage: deployer <command>

Commands:
   create-template
   download-images
   import-deployer
   encrypt-vault
   decrypt-vault
   run
   create-tar
   install-docker
   install-dbutils
   upgrade
```

Configure Autonomous Identity

The **create-template** command from the previous section creates a number of configuration files, required for the deployment: `ansible.cfg`, `vars.yml`, `hosts`, and `vault.yml`.

> NOTE
>
> If you are running a deployment for evaluation, you can minimally set the private IP address mapping in the `vars.yml` file in step 2, edit the `hosts` file in step 3, jump to step 6 to download the images, and then run the deployer in step 7.

1. The **create-template** commands creates a number of configuration files, including ~/autoid-config/ansible.cfg . Open a text editor and edit the `ansible.cfg` to set up the remote user and SSH private key file location on the target node. Make sure that the remote_user exists on the target node and that the deployer machine can ssh to the target node as the user specified in the `id_rsa` file.

```
[defaults]
host_key_checking = False
remote_user = autoid
private_key_file = id_rsa
```

2. On the deployer machine, open a text editor and edit the `~/autoid-config/vars.yml` file to configure specific settings for your deployment:

- **Domain and Target Environment**. Set the domain name and target environment specific to your deployment 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 Autonomous Identity URL will be: `https://autoid-ui.forgerock.com`. For this example, we use the default values.

  ```
  domain_name: forgerock.com
  target_environment: autoid
  ```

  If you change the domain name and target environment, you need to also change the certificates to reflect the new changes. For more information, see Customize Domains.

- **Analytics Data Directory and Analytics Configuration Direction**. For a multi-node Spark deployment, Autonomous Identity requires a shared filesystem accessible from Spark Master, Spark Worker(s), and Analytics hosts. The shared filesystem should be mounted at same mount directory on all of the above hosts. If the mount directory for shared filesystem is different than `/data`, update the following properties in the `vars.yaml` file to point to the correct location:

  ```
  analytics_data_dir: /data
  analytics_conf_dif: /data/conf
  ```

- **Dark Theme Mode**. Optional. By default, the Autonomous Identity UI displays its pages with a light background. You can set a dark theme mode by setting the `enable_dark_theme` property to `true`.

- **Database Type**. By default, Apache Cassandra is set as the default database for Autonomous Identity. For MongoDB, set the `db_driver_type:` to `mongo`.

  ```
  db_driver_type: mongo
  ```

- **Private IP Address Mapping**. Define a mapping between the external IP and private IP addresses. This occurs when your target host is in a cloud, so that your external and internal IP addresses are different.

  For each target node, add the `private_ip_address_mapping` property in the `~/autoid-config/vars.yml` file. You can look up the private IP on the cloud console, or run **sudo ifconfig** on the target host. Make sure the values are within double quotes. The key should not be in double quotes and should have two spaces preceding the IP address.

```
private_ip_address_mapping:
   external_ip:  "internal_ip"
```

For example:

```
private_ip_address_mapping:
   34.105.16.198:  "10.128.0.51"
   34.105.16.201:  "10.128.0.54"
   34.105.16.229:  "10.128.0.71"
```

- **Authentication Option**. Autonomous Identity provides a single sign-on (SSO) feature that you can configure with an OIDC identity provider. For more information, see <u>Set Up SSO</u>.

- **Access Log**. By default, the access log is enabled. If you want to disable the access log, set the `access_log_enabled` variable to "false".

- **JWT Expiry and Secret File**. Optional. By default, the session JWT is set at 30 minutes. To change this value, set the `jwt_expiry` property to a different value.

```
jwt_expiry: "30 minutes"
jwt_secret_file: "{{install path}}"/jwt/secret.txt"
jwt_audience: "http://my.service"
oidc_jwks_url: "na"
```

- **MongoDB Configuration**. For MongoDB clusters, enable replication by uncommenting the `mongodb_replication_replset` property.

```
# uncomment below for mongo with replication enabled.
Not needed for single node deployments
mongodb_replication_replset: mongors
```

Also, enable a custom key for inter-machine authentication in the clustered nodes.

```
# custom key
# password for inter-process authentication
# please regenerate this file on production environment
with
# command 'openssl rand -base64 741'
mongodb_keyfile_content: |

8pYcxvCqoe89kcp33KuTtKVf5MoHGEFjTnudrq5BosvWRoIxLowmdjr
mUpVfAivh
```

CHjqM6w0zVBytAxH1lW+7teMYe6eDn2S/O/1YlRRiW57bWU3zjliW3V
dguJar5i

Z+1a8lI+0S9pWynbv9+Ao0aXFjSJYVxAm/w7DJbVRGcPhsPmExiSBDw
8szfQ8PAU

2hwRl7nqPZZMMR+uQThg/zV9rOzHJmkqZtsO4UJSilG9euLCYrzW2hd
oPuCrEDhu

Vsi5+nwAgYR9dP2oWkmGN1dwRe0ixSIM2UzFgpaXZaMOG6VztmFrlVX
h8oFDRGM0

cGrFHcnGF7oUGfWnI2Cekngk64dHA2qD7WxXPbQ/svn9EfTY5aPw5lX
zKA87Ds8p

KHVFUYvmA6wVsxb/riGLwc+XZlb6M9gqHn1XSpsnYRjF6UzfRcRR2Wy
CxLZELaqu

iKxLKB5FYqMBH7Sqg3qBCtE53vZ7T1nefq5RFzmykviYP63Uhu/A2EQ
atrMnaFPl

TTG5CaPjob45CBSyMrheYRWKqxdWN93BTgiTW7p0U6RB0/OCUbsVX6I
G3I9N8Uqt

l8Kc+7aOmtUqFkwo8w30prIOjStMrokxNsuK9KTUiPu2cj7gwYQ574v
V3hQvQPAr

hhb9ohKr0zoPQt31iTj0FDkJzPepeuzqeq8F51HB56RZKpXdRTfY8G6
OaOT68cV5

vP1O6T/okFKrl41FQ3CyYN5eRHyRTK99zTytrjoP2EbtIZ18z+bg/an
gRHYNzbgk

lc3jpiGzs1ZWHD0nxOmHCMhU4usEcFbV6FlOxzlwrsEhHkeiununlCs
NHatiDgzp

ZWLnP/mXKV992/Jhu0Z577DHlh+3JIYx0PceB9yzACJ8MNARHF7QpBk
htuGMGZpF

T+c73exupZFxItXs1Bnhe3djgE3MKKyYvxNUIbcTJoe7nhVMrwO/7lB
SpVLvC4p3
  wR700U0LDaGGQpslGtiE56SemgoP

On production deployments, you can regenerate this file by running the following command:

```
$ openssl rand -base64 741
```

- Elasticsearch Heap Size. Optional. The default heap size for Elasticsearch is 1GB, which may be small for production. For production deployments, uncomment the option and specify `2G` or `3G`.

```
#elastic_heap_size: 1g    # sets the heap size
(1g|2g|3g) for the Elastic Servers
```

- Java API Service. Optional. Set the Java API Service (JAS) properties for the deployment: authentication, maximum memory, directory for attribute mappings data source entities:

```
jas:
  auth_enabled: true
  max_memory: 2048M
  mapping_entity_type: /common/mappings
  datasource_entity_type: /common/datasources
```

3. Open a text editor and enter the public IP addresses of the target machines in the `~/autoid-config/hosts` file. Make sure the target host IP addresses are accessible from the deployer machine. The following is an example of the `~/autoid-config/hosts` file. NOTE: `[notebook]` is not used in Autonomous Identity.

▼ *Click to See a Host File for a Multi-Node Cassandra Deployment*

If you configured Cassandra as your database, the `~/autoid-config/hosts` file is as follows for multi-node target deployments:

```
[docker-managers]
34.105.16.198

[docker-workers]
34.105.16.201

[docker:children]
docker-managers
docker-workers

[cassandra-seeds]
```

```
34.105.16.198

[cassandra-workers]
34.105.16.201

[spark-master]
34.105.16.198

[spark-workers]
34.105.16.201

[mongo_master]

[mongo_replicas]

[mongo:children]
mongo_replicas
mongo_master

# ELastic Nodes
[odfe-master-node]
34.105.16.229

[odfe-data-nodes]
34.105.16.229

[kibana-node]
34.105.16.229

[notebook]
#ip#
```

▼ *Click to See a Host File for a Multi-Node MongoDB Deployment*

If you configured MongoDB as your database, the `~/autoid-config/hosts` file is as follows for multi-node target deployments:

```
[docker-managers]
34.105.16.198

[docker-workers]
34.105.16.201

[docker:children]
docker-managers
```

```
docker-workers

[cassandra-seeds]

[cassandra-workers]

[spark-master]
34.105.16.198

[spark-workers]
34.105.16.201

[mongo_master]
34.105.16.198  mongodb_master=True

[mongo_replicas]
34.105.16.201

[mongo:children]
mongo_replicas
mongo_master

# ELastic Nodes
[odfe-master-node]
34.105.16.229

[odfe-data-nodes]
34.105.16.229

[kibana-node]
34.105.16.229

[notebook]
#ip#
```

4. Open a text editor and set the Autonomous Identity passwords for the configuration service, LDAP backend, and Cassandra database. The vault passwords file is located at `~/autoid-config/vault.yml`.

> **WARNING**
>
> Do not include special characters & or $ in `vault.yml` passwords as it will result in a failed deployer process.

```
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
```

5. Encrypt the vault file that stores the Autonomous Identity passwords, located at `~/autoid-config/vault.yml` . The encrypted passwords will be saved to `/config/.autoid_vault_password` . The `/config/` mount is internal to the deployer container.

```
$ ./deployer.sh encrypt-vault
```

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

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

7. Run the deployment.

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

Resolve Hostname

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

Resolve the hostname:

1. Configure your DNS servers to access Autonomous Identity dashboard and self-service applications on the target node. The following domain names must resolve to the IP address of the target node: `<target-environment>-ui.<domain-name>` and `<target-environment>-selfservice.<domain-name>` .

2. If DNS cannot resolve target node hostname, edit it locally on the machine that you want to access Autonomous Identity using a browser. Open a text editor and add an entry in the `/etc/hosts` file 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  myid-selfservice.abc.com
etc.
```

For more information on customizing your domain name, see Customize Domains.

Access the Dashboard

Access the 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: Welcome123
```

Check Apache Cassandra

Check Cassandra:

1. On the target node, check the status of Apache Cassandra.

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

2. An example output is as follows:

```
Datacenter: datacenter1
========================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
++--++  Address        Load        Tokens        Owns (effective)
Host ID                              Rack
UN  34.70.190.144 1.33 MiB    256            100.0%
a10a91a4-96e83dd-85a2-4f90d19224d9   rack1
++--++
```

Check MongoDB

Check the status of MongoDB:

1. On the target node, check the status of MongoDB.

   ```
   $ 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
   ```

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

   ▼ *Click to See an Example of the Spark Dashboard*

```
                                          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
  [IMG] 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, see Set Entity Definitions.

Install a Multi-Node Air-Gapped Deployment

This chapter presents instructions on deploying Autonomous Identity in a multi-node air-gapped or offline target machine that has no external Internet connectivity. ForgeRock provides a deployer script that pulls a Docker container image from ForgeRock's Google Cloud Registry (gcr.io) 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 with Internet connectivity, except that the image and deployer script must be stored on a portable media, such as USB drive or 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.

Figure 9: A multi-node air-gap deployment.
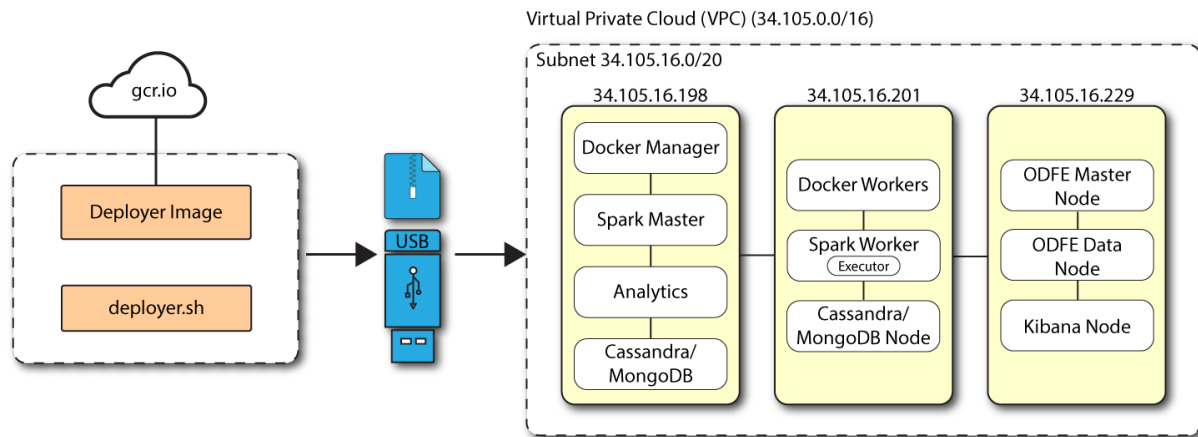
Prerequisites

Let's deploy Autonomous Identity on a single-node target on CentOS 7. The following are prerequisites:

- **Operating System**. The target machine requires CentOS 7. The deployer machine can use any operating system as long as Docker is installed. For this guide, we use CentOS 7 as its base operating system.

- **Memory Requirements**. Make sure you have enough free disk space on the deployer machine before running the `deployer.sh` commands. We recommend at least a 500GB.

- **Default Shell**. The default shell for the `autoid` user must be bash.

- **Subnet Requirements**. We recommend deploying your multinode instances 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/16, they will conflict with the Swarm network. As a result, the services in the cluster will 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**. 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 (gcr.io). The registry key is only available to ForgeRock Autonomous Identity customers. For specific instructions on obtaining the registry key, see How To Configure Service Credentials (Push Auth, Docker) in Backstage.

- **Filesystem Requirements**. Autonomous Identity requires a shared filesystem accessible from the Spark master, 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_dif: /data/conf
  ```

- **Architecture Requirements**. Make sure that the analytics server is on the same node as the Spark master.

- **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
  ```

Set Up the Target Nodes

Make sure you have sufficient storage for your particular deployment. For more information on sizing considerations, see <u>Deployment Planning Guide</u>.

For each target node, run the following commands.

1. The install assumes that you have CentOS 7 as your operating system. Check your CentOS 7 version.

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

2. Set the user for the target machine to a username of your choice. For example, `autoid`.

   ```
   $ sudo adduser autoid
   ```

3. Set the password for the user you created in the previous step.

   ```
   $ sudo passwd autoid
   ```

4. Configure the user for passwordless sudo.

```
$ echo "autoid  ALL=(ALL)  NOPASSWD:ALL" | sudo tee
/etc/sudoers.d/autoid
```

5. Add administrator privileges to the user.

```
$ sudo usermod -aG wheel autoid
```

6. Change to the user account.

```
$ su - autoid
```

7. Set up a shared directory on your target servers. The method depends on your machine configuration. Make sure the deployer can write to the shared directory.

Set Up the Deployer Machine

Set up the deployer on an Internet-connect machine.

1. The install assumes that you have CentOS 7 as your operating system. Check your CentOS 7 version.

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

2. Set the user for the target machine to a username of your choice. For example, autoid.

```
$ sudo adduser autoid
```

3. Set the password for the user you created in the previous step.

```
$ sudo passwd autoid
```

4. Configure the user for passwordless sudo.

```
$ echo "autoid  ALL=(ALL)  NOPASSWD:ALL" | sudo tee
/etc/sudoers.d/autoid
```

5. Add administrator privileges to the user.

```
$ sudo usermod -aG wheel autoid
```

6. Change to the user account.

```
$ su - autoid
```

7. 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
```

8. Create the installation directory. Note that you can use any install directory for your system as long as your run the **deployer.sh** script from there. Also, the disk volume where you have the install directory must have at least 8GB free space for the installation.

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

Install Docker on the Deployer Machine

1. On the target machine, set up the Docker-CE repository.

```
$ sudo yum-config-manager \
     --add-repo
https://download.docker.com/linux/centos/docker-ce.repo
```

2. Install the latest version of the Docker CE, the command-line interface, and containerd.io, a containerized website.

```
$ sudo yum install -y docker-ce docker-ce-cli
containerd.io
```

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-connect host machine:

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

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

2. Log in to the ForgeRock Google Cloud Registry (gcr.io) using the registry key. The registry key is only available to ForgeRock Autonomous Identity customers. For specific instructions on obtaining the registry key, see <u>How To Configure Service Credentials (Push Auth, Docker) in Backstage</u>.

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

   You should see:

   ```
   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:2021.3.0 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 Autonomous Identity binaries.

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

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

Install from the Air-Gap Target

Before you begin, make sure you have CentOS 7 installed on your air-gapped target
machine.

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

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

2. Unpack the tar file.

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

3. On the air-gap host node, copy the SSH key to the ~/autoid-config directory.

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

4. Change the privileges to the file.

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

5. Change to the configuration directory.

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

6. Install Docker.

```
$ sudo ./deployer.sh install-docker
```

7. Log out and back in.

8. Change to the configuration directory.

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

9. Import the deployer image.

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

10. Create the configuration template using he **create-template** command. This command creates a configuration file, `ansible.cfg` .

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

11. Make the script executable.

```
$ chmod +x deployer.sh
```

12. To see the list of commands, enter `deployer.sh` .

```
$ ./deployer.sh

Usage: deployer <command>

Commands:
   create-template
   download-images
   import-deployer
   encrypt-vault
   decrypt-vault
   run
   create-tar
   install-docker
   install-dbutils
   upgrade
```

Configure Autonomous Identity

The `create-template` command from the previous section creates a number of configuration files, required for the deployment: `ansible.cfg`, `vars.yml`, `hosts`, and `vault.yml`.

> **NOTE**
>
> If you are running a deployment for evaluation, you can minimally set the private IP address mapping in the `vars.yml` file in step 2, edit the `hosts` file in step 3, jump to step 6 to download the images and then run the deployer in step 7.

> **IMPORTANT**
>
> For air-gapped deployments, you must set the `offline_mode` property to `true` in the `~/autoid-config/vars.yml` file in step 2 below. This is a new change in 2021.3.0 from prior releases.

1. Open a text editor and edit the `~/autoid-config/ansible.cfg` to set up the remote user and SSH private key file location on the target node. Make sure that the remote_user exists on the target node and that the deployer machine can ssh to the target node as the user specified in the `id_rsa` file.

   ```
   [defaults]
   host_key_checking = False
   remote_user = autoid
   private_key_file = id_rsa
   ```

2. On the deployer machine, open a text editor and edit the `~/autoid-config/vars.yml` file to configure specific settings for your deployment:

   - **Domain and Target Environment**. Set the domain name and target environment specific to your deployment 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 Autonomous Identity URL will be: `https://autoid-ui.forgerock.com`. For this example, we use the default values.

     ```
     domain_name: forgerock.com
     target_environment: autoid
     ```

     If you change the domain name and target environment, you need to also change the certificates to reflect the new changes. For more information, see Customize Domains.

- **Analytics Data Directory and Analytics Configuration Direction**. For a multi-node Spark deployment, Autonomous Identity requires a shared filesystem accessible from Spark Master, Spark Worker(s), and Analytics hosts. The shared filesystem should be mounted at same mount directory on all of the above hosts. If the mount directory for shared filesystem is different than `/data`, update the following properties in the `vars.yaml` file to point to the correct location:

  ```
  analytics_data_dir: /data
  analytics_conf_dif: /data/conf
  ```

- **Dark Theme Mode**. Optional. By default, the Autonomous Identity UI displays its pages with a light background. You can set a dark theme mode by setting the `enable_dark_theme` property to `true`.

- **Offline Mode**. Set the `offline_mode` to `true` for air-gapped deployments.

  ```
  offline_mode: true
  ```

- **Database Type**. By default, Apache Cassandra is set as the default database for Autonomous Identity. For MongoDB, set the `db_driver_type:` to `mongo`.

  ```
  db_driver_type: mongo
  ```

- **Private IP Address Mapping**. An air-gap deployment has no external IP addresses, but you may still need to define a mapping if your internal IP address differs from an external IP, say in a virtual air-gapped configuration.

  If the IP addresses are the same, you can skip this step.

  On the target machine, add the `private_ip_address_mapping` property in the `/inventory/vars.yml` file. Make sure the values are within double quotes. The key should not be in double quotes and should have two spaces preceding the IP address.

  ```
  private_ip_address_mapping:
    external_ip:  "internal_ip"
  ```

  For example:

  ```
  private_ip_address_mapping:
    34.105.16.198:  "10.128.0.51"
  ```

```
34.105.16.201:  "10.128.0.54"
34.105.16.229:  "10.128.0.71"
```

- **Authentication Option**. Autonomous Identity provides a single sign-on (SSO) feature that you can configure with an OIDC identity provider.

- **Access Log**. By default, the access log is enabled. If you want to disable the access log, set the `access_log_enabled` variable to "false".

- **JWT Expiry and Secret File**. Optional. By default, the session JWT is set at 30 minutes. To change this value, set the `jwt_expiry` property to a different value.

```
jwt_expiry: "30 minutes"
jwt_secret_file: "{{install path}}"/jwt/secret.txt"
jwt_audience: "http://my.service"
oidc_jwks_url: "na"
```

- **MongoDB Configuration**. For MongoDB clusters, enable replication by uncommenting the `mongodb_replication_replset` property.

```
# uncomment below for mongo with replication enabled.
Not needed for single node deployments
mongodb_replication_replset: mongors
```

Also, enable a custom key for inter-machine authentication in the clustered nodes.

```
# custom key
# password for inter-process authentication
# please regenerate this file on production environment
with
# command 'openssl rand -base64 741'
mongodb_keyfile_content: |
```

```
8pYcxvCqoe89kcp33KuTtKVf5MoHGEFjTnudrq5BosvWRoIxLowmdjr
mUpVfAivh

CHjqM6w0zVBytAxH1lW+7teMYe6eDn2S/O/1YlRRiW57bWU3zjliW3V
dguJar5i

Z+1a8lI+0S9pWynbv9+Ao0aXFjSJYVxAm/w7DJbVRGcPhsPmExiSBDw
8szfQ8PAU

2hwRl7nqPZZMMR+uQThg/zV9rOzHJmkqZtsO4UJSilG9euLCYrzW2hd
oPuCrEDhu
```

```
Vsi5+nwAgYR9dP2oWkmGN1dwRe0ixSIM2UzFgpaXZaMOG6VztmFrlVX
h8oFDRGM0

cGrFHcnGF7oUGfWnI2Cekngk64dHA2qD7WxXPbQ/svn9EfTY5aPw5lX
zKA87Ds8p

KHVFUYvmA6wVsxb/riGLwc+XZlb6M9gqHn1XSpsnYRjF6UzfRcRR2Wy
CxLZELaqu

iKxLKB5FYqMBH7Sqg3qBCtE53vZ7T1nefq5RFzmykviYP63Uhu/A2EQ
atrMnaFPl

TTG5CaPjob45CBSyMrheYRWKqxdWN93BTgiTW7p0U6RB0/OCUbsVX6I
G3I9N8Uqt

l8Kc+7aOmtUqFkwo8w30prIOjStMrokxNsuK9KTUiPu2cj7gwYQ574v
V3hQvQPAr

hhb9ohKr0zoPQt31iTj0FDkJzPepeuzqeq8F51HB56RZKpXdRTfY8G6
OaOT68cV5

vP1O6T/okFKrl41FQ3CyYN5eRHyRTK99zTytrjoP2EbtIZ18z+bg/an
gRHYNzbgk

lc3jpiGzs1ZWHD0nxOmHCMhU4usEcFbV6FlOxzlwrsEhHkeiununlCs
NHatiDgzp

ZWLnP/mXKV992/Jhu0Z577DHlh+3JIYx0PceB9yzACJ8MNARHF7QpBk
htuGMGZpF

T+c73exupZFxItXs1Bnhe3djgE3MKKyYvxNUIbcTJoe7nhVMrwO/7lB
SpVLvC4p3
   wR700U0LDaGGQpslGtiE56SemgoP
```

On production deployments, you can regenerate this file by running the
following command:

```
$ openssl rand -base64 741
```

- **Elasticsearch Heap Size**. Optional. The default heap size for Elasticsearch is
  1GB, which may be small for production. For production deployments,
  uncomment the option and specify `2G` or `3G`.

```
#elastic_heap_size: 1g   # sets the heap size
(1g|2g|3g) for the Elastic Servers
```

  ○ **Java API Service**. Optional. Set the Java API Service (JAS) properties for the
    deployment: authentication, maximum memory, directory for attribute
    mappings data source entities:

```
jas:
    auth_enabled: true
    max_memory: 2048M
    mapping_entity_type: /common/mappings
    datasource_entity_type: /common/datasources
```

3. Open a text editor and enter the public IP addresses of the target machines in
   the ~/autoid-config/hosts file. Make sure the target host IP addresses are
   accessible from the deployer machine. The following is an example of the
   ~/autoid-config/hosts file:

   ▼ *Click to See a Host File for Cassandra Deployments*

   If you configured Cassandra as your database, the ~/autoid-config/hosts file
   is as follows for single-node target deployments:

```
[docker-managers]
34.105.16.198

[docker-workers]
34.105.16.201

[docker:children]
docker-managers
docker-workers

[cassandra-seeds]
34.105.16.198

[cassandra-workers]
34.105.16.201

[spark-master]
34.105.16.198

[spark-workers]
34.105.16.201
```

```
[mongo_master]

[mongo_replicas]

[mongo:children]
mongo_replicas
mongo_master

# ELastic Nodes
[odfe-master-node]
34.105.16.229

[odfe-data-nodes]
34.105.16.229

[kibana-node]
34.105.16.229

[notebook]
#ip#
```

▼ *Click to See a Host File for MongoDB Deployments*

If you configured MongoDB as your database, the `~/autoid-config/hosts` file
is as follows for single-node target deployments:

```
[docker-managers]
34.105.16.198

[docker-workers]
34.105.16.201

[docker:children]
docker-managers
docker-workers

[cassandra-seeds]

[cassandra-workers]

[spark-master]
34.105.16.198

[spark-workers]
34.105.16.201
```

```
[mongo_master]
34.105.16.198   mongodb_master=True

[mongo_replicas]
34.105.16.201

[mongo:children]
mongo_replicas
mongo_master

# ELastic Nodes
[odfe-master-node]
34.105.16.229

[odfe-data-nodes]
34.105.16.229

[kibana-node]
34.105.16.229

[notebook]
#ip#
```

4. Set the Autonomous Identity passwords, located at `~/autoid-config/vault.yml`.

> **WARNING**
>
> Do not include special characters & or $ in `vault.yml` passwords as it will result in a failed deployer process.

```
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
```

5. Encrypt the vault file that stores the Autonomous Identity passwords, located at `~/autoid-config/vault.yml` . The encrypted passwords will be saved to `/config/.autoid_vault_password` . The `/config/` mount is internal to the deployer container.

```
$ ./deployer.sh encrypt-vault
```

6. Run the deployment.

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

Access the Dashboard

Access the 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: Welcome123
```

Check Apache Cassandra

Check Cassandra:

1. On the target node, check the status of Apache Cassandra.

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

2. An example output is as follows:

```
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
++--++  Address        Load       Tokens      Owns (effective)
Host ID                           Rack
UN  34.70.190.144 1.33 MiB   256         100.0%
a10a91a4-96e83dd-85a2-4f90d19224d9  rack1
++--++
```

Check MongoDB

Check the status of MongoDB:

1. On the target node, check the status of MongoDB.

   ```
   $ 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
   ```

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

   ▼ *Click to See an Example of the Spark Dashboard*

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, see Set Entity Definitions.

Upgrading Autonomous Identity

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

The upgrade assumes the following:

- **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. See the instructions below.

- **Registry Key Required**. To download the deployment images for the upgrade, you still need a registry key to log into the ForgeRock Google Cloud Registry (gcr.io). The registry key is only available to ForgeRock Autonomous Identity customers. For specific instructions on obtaining the registry key, see How To Configure Service Credentials (Push Auth, Docker) in Backstage.

Upgrade to Autonomous Identity 2021.3.1

The following instruction is for upgrade from Autonomous Identity **2021.3.0** to version **2021.3.0**.

> Upgrade to version 2021.3.0:
>
> 1. On the deployer machine, back up the 2021.3.0 `~/autoid-config` directory or move it to another location.
>
>    ```
>    $ mv ~/autoid-config ~/backup-2021.3
>    ```
>
> 2. Create a new `~/autoid-config` directory.
>
>    ```
>    $ mkdir ~/autoid-config
>    ```
>
> 3. Copy your `autoid_registry_key.json` from your backup directory to `~/autoid-config`.
>
> 4. Remove your `known_files`.
>
>    ```
>    $ rm ~/.ssh/known_hosts
>    ```
>
> 5. Copy your original SSH key into the new directory.
>
>    ```
>    $ cp ~/.ssh/id_rsa ~/autoid-config
>    ```
>
> 6. Change the permission on the SSH key.
>
>    ```
>    $ chmod 400 ~/autoid-config/id_rsa
>    ```
>
> 7. Check if you can successfully SSH to the target server.
>
>    ```
>    $ ssh -i ~/autoid-config/id_rsa autoid@<Target-IP-Address>
>
>    Last login: Tue May 23 18:19:14 2021
>    ```
>
> 8. Stop the stack.
>
>    ```
>    $ docker stack rm configuration-service consul-server
>    consul-client nginx jas openldap selfservice swagger-ui ui
>    api
>    ```
>
>   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 openldap_openldap
Removing service openldap_phpldapadmin
Removing service selfservice_selfservice
Removing service swagger-ui_swagger-ui
Removing service ui_zoran-ui
Removing service api_zoran-api
```

9. Remove the contents of the consul data for Docker Manager and Swarm:

```
$ sudo rm -r /opt/autoid/mounts/consul-data/*
```

10. Remove the analytics container of the analytics node:

```
$ docker rm -f analytics
```

11. Enter **exit** to end your SSH session.

12. Repeat the restart Docker command:

```
$ sudo systemctl restart docker
```

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

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

14. Log in to the ForgeRock Google Cloud Registry (gcr.io) using the registry key. The registry key is only available to ForgeRock 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
```

15. 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:2021.3.0 create-
template


…
d6c7c6f3303e: Pull complete
Digest:
sha256:15225be65417f8bfb111adea37c83eb5e0d87140ed498bfb624
a358f43fbbf
Status: Downloaded newer image for gcr.io/forgerock-
autoid/autoid/dev-
compact/deployer@sha256:15225be65417f8bfb111a
dea37c83eb5e0d87140ed498bfb624a358f43fbbf
Config template is copied to host machine directory mapped
to /config
```

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

    The key here is to keep your configuration settings consistent from one system to another.

17. 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
```

    You should see:

```
PLAY RECAP *
localhost              : ok=19    changed=17    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

18. SSH to the target node.

19. Stop the Spark master and workers.

```
$ /opt/autoid/spark/spark-2.4.4-bin-hadoop2.7/sbin/stop-
all.sh
```

You should see:

```
localhost: stopping org.apache.spark.deploy.worker.Worker
stopping org.apache.spark.deploy.master.Master
```

20. Exit your SSH session.

21. Run the upgrade.

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

You should see:

```
PLAY RECAP *
<Target-IP-Addr>   : ok=407   changed=177   unreachable=0
failed=0    skipped=134  rescued=0    ignored=0
localhost          : ok=65    changed=22    unreachable=0
failed=0    skipped=7    rescued=0    ignored=0
```

22. Add a reference to Autonomous Identity JAS to your `/etc/hosts` or DNS server.

```
<Public IP Address> autoid-ui.forgerock.com autoid-
selfservice.forgerock.com autoid-jas.forgerock.com
```

23. At this point, you must import the entity definitions for assignments into Autonomous Identity, which will require several steps. You can use Postman or curl to access and post the definitions. For example, use Postman in the following steps:

   a. Open Postman, enter a request to get the latest index value: `https://autoid-jas.forgerock.com/entityDefinitions/autoid/base/assignments?latest=true`. Enter the following headers:

   ```
   Content-type      application/json

   X-TENANT-ID       <tenant ID>

   Authorization     <Bearer>
   ```

   To get the Tenant ID and Authorization token:

      i. Open the Autonomous Identity UI and log in as an administrator.

ii. On the Autonomous Identity Dashboard, right-click the browser page and open the DevTools by clicking `Inspect`.

iii. Click the Network tab and reload the page.

iv. Click `tenants`, and then `Response`. You will see the tenant ID.

v. Click `6`, and then click `Headers`. Scroll down to get the Authorization Bearer token.
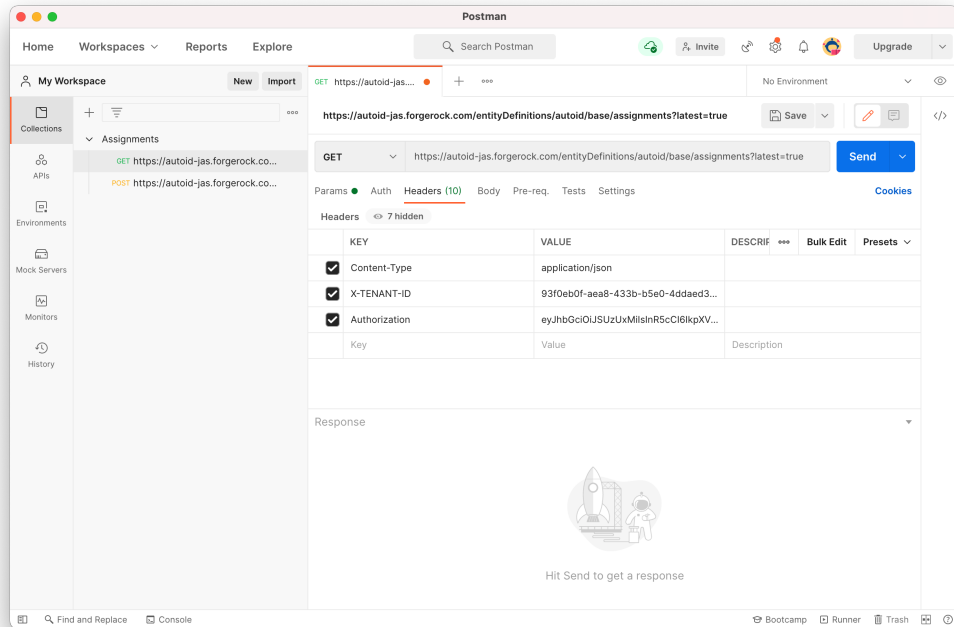
▼ *See it in action.*
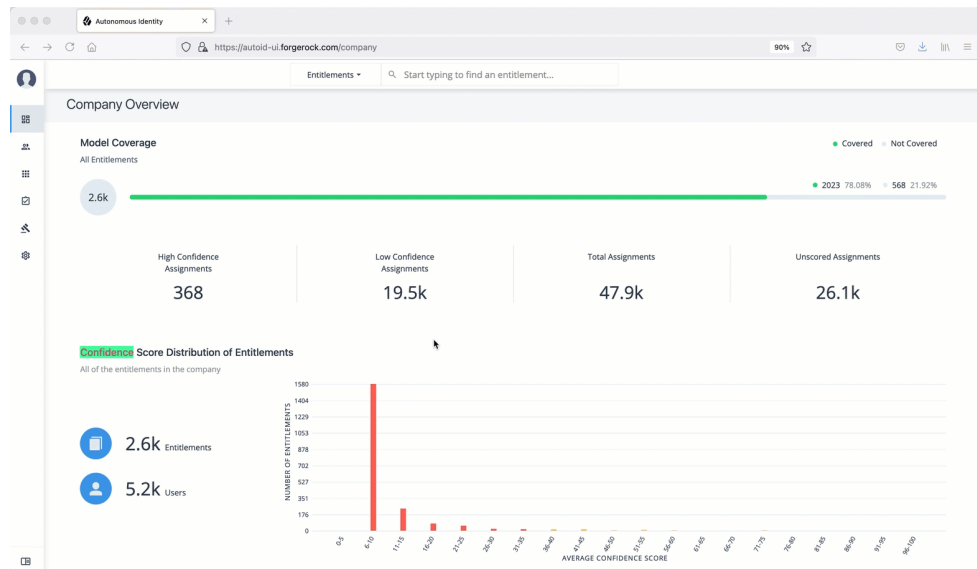
> **NOTE**
>
> If you get a 502 Bad Gateway nginx error:
>
> 1. SSH to the target server.
>
> 2. Restart nginx:
>
> ```
> $ docker service scale nginx_nginx=0
> $ docker service scale nginx_nginx=1
> ```

b. On Postman, enter the Tenant ID and Authorization Bearer token, and then click Send to get the assignments entity definition in the response.

▼ *See an image.*

c. Next, on Postman, enter another request to POST the assignments entity definition to Autonomous Identity. Use the following URL: `https://autoid-jas.forgerock.com/entityDefinitions/autoid/base/3`. Use the same headers as the previous GET example. In the Body, paste in the Assignments entity definition, and then click Send.

▼ *See an image.*
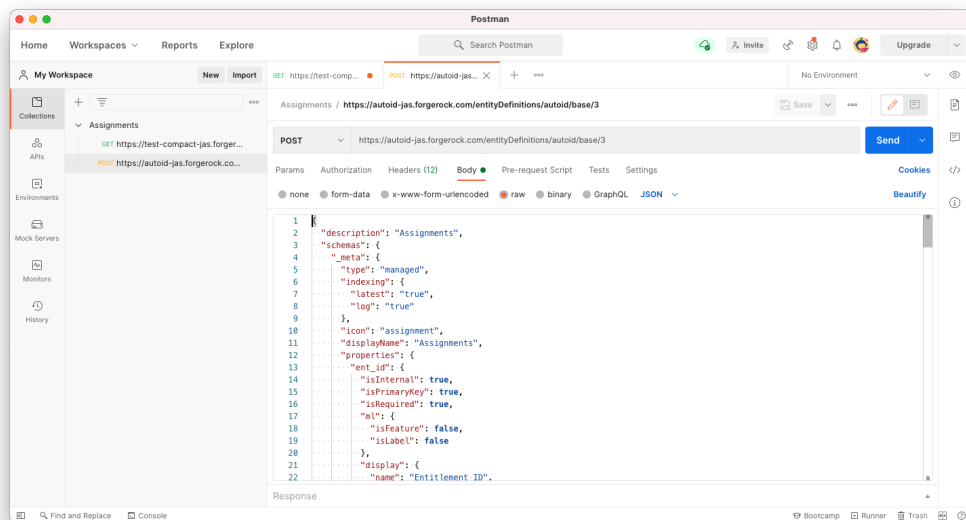
24. Go back to the Autonomous Identity UI, click Administration > Entity Definitions. You should see the `Assignments` entity definitions.

25. Redo the attribute mappings. See Set Attribute Mappings.

26. Then, rerun the analytics pipeline.

```
$ analytics ingest
$ analytics train
$ analytics predict-as-is
$ analytics predict-recommendation
$ analytics publish
$ analytics create-assignment-index
```

You have successfully upgraded your Autonomous Identity server to 2021.3.0.

Upgrade to Autonomous Identity 2021.3.0

The following instruction is for upgrade from Autonomous Identity **2021.10.x** to version **2021.3.0**.

Upgrade to version 2021.3.0:

1. On the deployer machine, back up the 2020.10.x ~/autoid-config directory or move it to another location.

   ```
   $ mv ~/autoid-config ~/backup-2020.10
   ```

2. Create a new ~/autoid-config directory.

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

3. Remove your known_files .

   ```
   $ rm ~/.ssh/known_hosts
   ```

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

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

5. Change the permission on the SSH key.

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

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

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

   Last login: Tue Mar 23 18:19:14 2020
   ```

7. Stop the stack.

```
$ docker stack rm configuration-service consul-server
consul-client nginx openldap selfservice swagger-ui ui api
```

You should see:

```
Removing service configuration-service_configuration-
service
Removing service consul-server_consul-server
Removing service nginx_nginx
Removing service openldap_openldap
Removing service openldap_phpldapadmin
Removing service selfservice_selfservice
Removing service swagger-ui_swagger-ui
Removing service ui_zoran-ui
Removing service ui_zoran-api
Removing service consul-client_consul-client
```

8. Remove the contents of the consul data:

```
$ sudo rm -r /opt/autoid/mounts/consul-data/*
```

9. Enter **exit** to end your SSH session.

10. From the deployer, restart Docker:

```
$ sudo systemctl restart docker
```

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

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

12. Log in to the ForgeRock Google Cloud Registry (gcr.io) using the registry key. The registry key is only available to ForgeRock 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
```

13. 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:2021.3.0 create-
template


…
d6c7c6f3303e: Pull complete
Digest:
sha256:15225be65417f8bfb111adea37c83eb5e0d87140ed498bfb624
a358f43fbbf
Status: Downloaded newer image for gcr.io/forgerock-
autoid/autoid/dev-
compact/deployer@sha256:15225be65417f8bfb111a
dea37c83eb5e0d87140ed498bfb624a358f43fbbf
Config template is copied to host machine directory mapped
to /config
```

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

The key here is to keep your configuration settings consistent from one system to another. For example, if your hosts file from your 2020.10.x system is as follows:

▼ *See an example hosts file from a 2020.10.x Deployment*

```
[docker-managers]
34.70.190.144

[docker-workers]
34.70.190.144

[docker:children]
docker-managers
docker-workers

[cassandra-seeds]
34.70.190.144
```

```
[cassandra-workers]
34.70.190.144

[spark-master]
34.70.190.144

[spark-workers]
34.70.190.144

[analytics]
34.70.190.144

[mongo_master]
#ip#   mongodb_master=True

[mongo_replicas]
#ip-1#
##ip-2#
##...

[mongo:children]
mongo_replicas
mongo_master

# ELastic Nodes
[odfe-master-node]
34.70.190.144

[odfe-data-nodes]
34.70.190.144

[kibana-node]
34.70.190.144
```

An example 2021.3.0 hosts file would be as follows (single-node example):

▼ *See an example hosts file from a 2021.3.0 Deployment*

```
[docker-managers]
34.70.190.144

[docker-workers]
34.70.190.144
```

```
[docker:children]
docker-managers
docker-workers

[cassandra-seeds]
34.70.190.144

[cassandra-workers]
34.70.190.144

[spark-master]
34.70.190.144

[spark-workers]
34.70.190.144

[mongo_master]
#ip#   mongodb_master=True

[mongo_replicas]
#ip-1#
##ip-2#
##...

[mongo:children]
mongo_replicas
mongo_master

# ELastic Nodes
[odfe-master-node]
34.70.190.144

[odfe-data-nodes]
34.70.190.144

[kibana-node]
34.70.190.144

[noteboook]
#ip#
```

15. 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
```

You should see:

```
PLAY RECAP *
localhost              : ok=24    changed=17    unreachable=0
failed=0     skipped=0     rescued=0     ignored=0
```

16. SSH to the target node.

17. Stop Apache Spark so that the deployer can upgrade the version to 3.0.1.

   ○ Stop the Spark master and workers.

   ```
   $ /opt/autoid/spark/spark-2.4.4-bin-
   hadoop2.7/sbin/stop-all.sh
   ```

   You should see:

   ```
   localhost: stopping
   org.apache.spark.deploy.worker.Worker
   stopping org.apache.spark.deploy.master.Master
   ```

18. Exit your SSH session.

19. On the deployer node, run the upgrade.

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

   You should see:

   ```
   PLAY RECAP **
   <Target-IP-Addr>    : ok=313   changed=153   unreachable=0
   failed=0     skipped=15   rescued=0     ignored=0
   localhost              : ok=11    changed=5     unreachable=0
   failed=0     skipped=6     rescued=0     ignored=0
   ```

20. On the target node, take a backup of the `/data/conf` directory. This directory holds the configuration files used in 2020.10.x.

   ```
   $ cp -r /data/conf <backup_directory>
   ```

21. Change to the upgrade directory:

   ```
   $ cd /opt/autoid/apache-livy/analytics-artifacts/upgrade
   ```

22. Edit the `upgrade.yml` file. Assuming a Cassandra database (MongoDB will have analogous properties), add the following values in the `cassandra` section:

- Add the IP addresses of the Cassandra machines where the 2020.10.x data is stored next to the `hosts` propert.

- Add the password for the Cassandra account.

- Add the keystore password for `spark.cassandra.connection.ssl.keyStore.password`.

- And the truststore passwords for `spark.cassandra.connection.ssl.trustStore.password`.

```
upgrade:
  old_base:         'autoid_base'
  old_analytics:    'autoid_analytics'
  old_ui:           'autoid'
  steps:            ['base', 'analytics', 'ui']
  ui_steps:         ['ui_tables',
'ui_tables_with_batch_history',
'ui_tables_with_date_history',
'ui_tables_with_history_only']
  tracker_file:     /tmp/upgrade_tracker_file.yaml
  batch_restart:    False
spark:
  logging_level: FATAL
  config:
    spark.scheduler.mode: FAIR
    spark.executor.memory: 10G
    spark.driver.memory: 20G
    spark.driver.maxResultSize: 5G
cassandra:
  hosts: []
  port: 9042
  username: zoran_dba
  password:
  ssl:
    enabled: true
    python:
      keyfile: /opt/autoid/certs/zoran-cassandra-
client-key.pem
      certfile: /opt/autoid/certs/zoran-cassandra-
client-cer.pem
        spark:
          spark.cassandra.connection.ssl.enabled: true
```

```
spark.cassandra.connection.ssl.clientAuth.enabled: true
        spark.cassandra.connection.ssl.keyStore.password:

spark.cassandra.connection.ssl.trustStore.password:
        spark.cassandra.connection.ssl.keyStore.path:
/opt/autoid/certs/zoran-cassandra-client-keystore.jks
        spark.cassandra.connection.ssl.trustStore.path:
/opt/autoid/certs/zoran-cassandra-server-truststore.jks
eps:
    tenant:        autonomous-iam
    batch_size:    10000
```

23. Change to the analytics directory.

```
$ cd /opt/autoid/apache-livy/analytics
```

24. Upgrade the analytics:

```
$ analytics upgrade
```

25. Open the 2020.10.1 `/data/conf/analytics_init_config.yml` file. You will need to get some properties from that file later.

26. Log in to the 2021.3.0 Autonomous Identity as an Admin account. Navigate to `Administration > Entity Definitions`.

27. Open the Identities definition, this page contains the definition for the User attribute entity used by Autonomous Identity.

28. Open the 2020.10.1 `/data/conf/analytics_init_config.yml` file and take the list of properties stored as `ui_config > user_column_descriptions`.

29. On the 2021.3.0 UI, click the **Add attribute** button, and the add each attribute to the list. Note: You will see that `usr_id` (replaces `usr_key`), `usr_name`, and `usr_manager` (replaces `usr_manager_key`) are already present.

30. The **Add attribute** requires five pieces of information that you must add. You will add attributes listed under `ui_config > user_column_descriptions`:

    a. Attribute Name. This field corresponds to the value before the colon in your `user_column_descriptions` list. For example, if you have `'JOBCODE_NAME':'Job Code Name'`, use `'JOBCODE_NAME'`.

    b. Display Name. This field corresponds to the value after the colon in your `user_column_descriptions`. For example, if you have `'JOBCODE_NAME':'Job Code Name'`, use `'JOBCODE_NAME'`.

    c. Dropdown. The dropdown shows the data type. Note that only attributes marked as 'Text' can be used in machine learning.

d. Use in machine learning. This indicates whether this attribute should be used in training. If the attribute you are adding appears in `assoc_rules >` `features_filter` from the `/data/conf/analytics_init_config.yml` file, then click this box.

e. Searchable. This indicates whether this attribute can be used as a filter in the UI. if the attribute appears in `ui_config > usr_filtering_columns`, then click this box.

f. Once you have added all the attributes, click **Save** at the top of the `Identities` table.

31. Repeat steps 27-28 for the Applications entities. In this case, you will add the attributes listed under `ui_config > app_filter_columns`:

a. Attribute Name. This field corresponds to the value before the colon in your `app_filter_columns` list. For example, if you have `'APP_CRITICALITY':'Application Criticality'`, use `'APP_CRITICALITY'`.

b. Display Name. This field corresponds to the value after the colon in your `app_filter_columns`. For example, if you have `'APP_CRITICALITY':'Application Criticality'`, use `'APP_CRITICALITY'`.

c. Dropdown. The dropdown shows the data type.

d. Searchable. This indicates whether you want the attribute to be filterable in the UI.

32. Repeat steps 27-28 for the Entitlements entities. In this case, you will add the attributes listed under `ui_config > ent_filter_columns`:

a. Attribute Name. This field corresponds to the value before the colon in your `ent_filter_columns` list. For example, if you have `'ENT_RISK_LEVEL':'Entitlement Risk'`, use `'ENT_RISK_LEVEL'`.

b. Display Name. This field corresponds to the value after the colon in your `ent_filter_columns`. For example, if you have `'ENT_RISK_LEVEL':'Entitlement Risk'`, use `'ENT_RISK_LEVEL'`.

c. Dropdown. The dropdown shows the data type.

d. Searchable. This indicates whether you want the attribute to be filterable in the UI.

33. Navigate to `Administration > Analytics Settings`. Compare the values in the 2020.10.1 `analytics_config.yml` and modify if required. Make sure to save your settings. The mappings are shown below:

Autonomous Identity Threshold Mappings

2020.10.1 analytics_config.yml	2021.3 UI Administration > Analytics Settings

etl > med_conf	Confidence Score Thresholds > Medium
etl > high_conf	Confidence Score Thresholds > High
prediction > recommend	Recommendation Threshold > Threshold

34. Run `create-assignment-index` to generate a new index using the migrated data:

```
$ analytics create-assignment-index
```

You have successfully upgraded your Autonomous Identity server to 2021.3.0.

Appendix A: Appendix A: Autonomous Identity Ports

The 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.

To see the available Autonomous Identity ports, see Autonomous Identity Ports.

Appendix B: vars.yml

Autonomous Identity has a configuration file where you can set the analytics data and configuration directories, UI dark theme mode, 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
enable_dark_theme: false                # Set true for dark UI
theme mode
```

```
# 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-addesses#

api:
  authentication_option: "Ldap"                    # Values:
"Ldap", "SSO", "LdapAndSSO"
  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"

# 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:
#   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/zV9rOzHJmkqZtsO4UJSilG9euLCYrzW2hdoPuCrEDhu
#
Vsi5+nwAgYR9dP2oWkmGN1dwRe0ixSIM2UzFgpaXZaMOG6VztmFrlVXh8oFDRGM0
#
cGrFHcnGF7oUGfWnI2Cekngk64dHA2qD7WxXPbQ/svn9EfTY5aPw5lXzKA87Ds8p
#
KHVFUYvmA6wVsxb/riGLwc+XZlb6M9gqHn1XSpsnYRjF6UzfRcRR2WyCxLZELaqu
#
iKxLKB5FYqMBH7Sqg3qBCtE53vZ7T1nefq5RFzmykviYP63Uhu/A2EQatrMnaFPl
#
TTG5CaPjob45CBSyMrheYRWKqxdWN93BTgiTW7p0U6RB0/OCUbsVX6IG3I9N8Uqt
#
l8Kc+7aOmtUqFkwo8w30prIOjStMrokxNsuK9KTUiPu2cj7gwYQ574vV3hQvQPAr
```

```
#  hhb9ohKr0zoPQt31iTj0FDkJzPepeuzqeq8F51HB56RZKpXdRTfY8G6OaOT68cV5
#  vP1O6T/okFKrl41FQ3CyYN5eRHyRTK99zTytrjoP2EbtIZ18z+bg/angRHYNzbgk
#  lc3jpiGzs1ZWHD0nxOmHCMhU4usEcFbV6FlOxzlwrsEhHkeiununlCsNHatiDgzp
#  ZWLnP/mXKV992/Jhu0Z577DHlh+3JIYx0PceB9yzACJ8MNARHF7QpBkhtuGMGZpF
#  T+c73exupZFxItXs1Bnhe3djgE3MKKyYvxNUIbcTJoe7nhVMrwO/7lBSpVLvC4p3
#   wR700U0LDaGGQpslGtiE56SemgoP

# mongo config ends

elastic_heap_size: 1g    # sets the heap size (1g|2g|3g) for the
Elastic Servers

jas:
  auth_enabled: true
  max_memory: 2048M
  mapping_entity_type: /common/mappings
  datasource_entity_type: /common/datasources
```

Was this helpful? 👍 👎