FORGEROCK®

Deployment Planning Guide

/ Autonomous Identity 2020.6.4

Latest update: 2020.6.4

Copyright © 2020 ForgeRock AS.

Abstract

This guide is targeted to deployer, administrators, and system architects who must plan a production deployment of the Autonomous Identity system.

Table of Contents

Overview

Use this guide to plan your production Autonomous Identity deployment.

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.

> **Important**
>
> This guide is for deployers, technical consultants, and administrators who are familiar with Autonomous Identity and are responsible for architecting a production deployment.

Quick Start

☑	▦	🧊
Topology Planning	Deployment Checklist	Glossary
Learn about topology sizing considerations.	Learn about the Autonomous Identity deployment checklist.	Review the Autonomous Identity glossary terms

For installation instructions, see the Installation Guide.

For component versions, see the Release Notes.

FORGEROCK

Chapter 1
Architecture in Brief

The Autonomous Identity architecture has a simple three-layer conceptual model:

Figure 1: A Conceptual Image of the Autonomous Identity Architecture
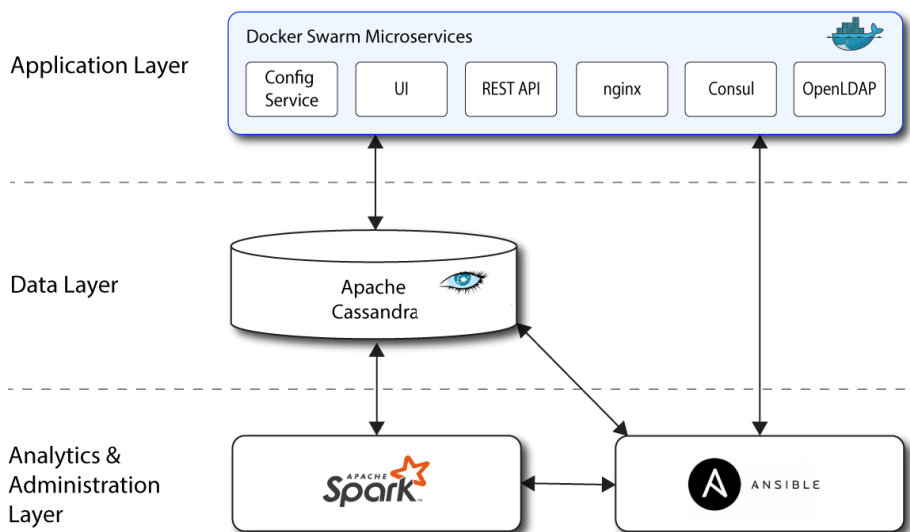
- **Application Layer**. Autonomous Identity implements a flexible Docker Swarm microservices architecture, where multiple applications run together in containers. The microservices component provides flexible configuration and end-user interaction to the deployment. The microservices components are the following:

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

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

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

FORGEROCK

- **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.

- **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.

- **Data Layer**. Autonomous Identity uses a Apache Cassandra NoSQL database 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.

- **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 also uses a deployer wrapper script to launch Ansible playbooks for easy and quick deployment of the components.

Autonomous Identity's flexible architecture can be deployed in any number of ways: single-node or multi-node configurations across on-prem, cloud, hybrid, or multi-cloud environments. For example, we typically configure a two-server deployment with the minimum hardware and software requirements for a pilot deployment.

Figure 2: An Image of a Two-Server Autonomous Identity Pilot Architecture

FORGEROCK
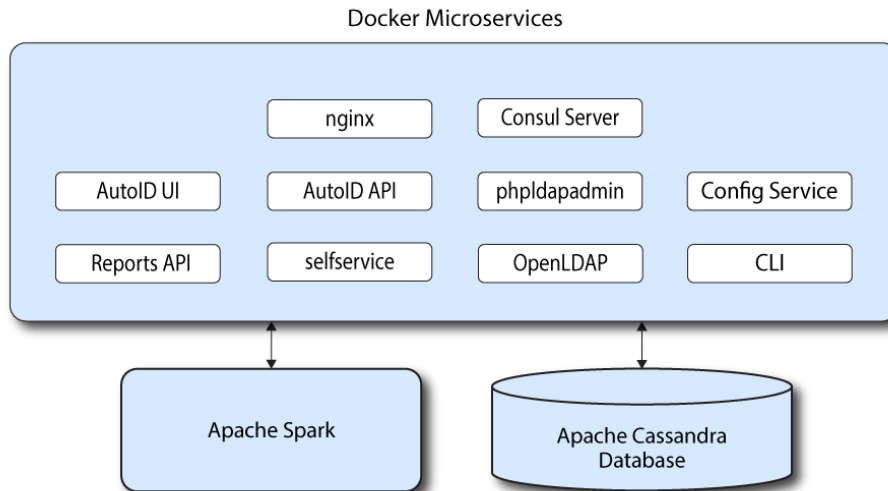
Chapter 2
Topology Planning

Based on existing production deployments, we have determined a suggested number of servers based on total entitlement assignments. These suggested number of servers are guidelines for your particular deployment requirements. Unique client requirements may require customization, which may differ from the listed number of servers.

For a description of possible production deployments, see "*Deployment Architectures*" in the *Installation Guide* in the Installation Guide.

Suggested Number of Servers

Data Set Ranges

	Small	Medium	Large	Custom
Total Assignments	<1M	1-5M	5-15M	15M+
Suggested # of Servers				
Application[a]	Discuss with Autonomous Identity Team (dependent on HA requirements)			
Database	2	2	3	Custom[b]
Analytics	1	2	3	Custom[b]
Admin[c]	1	1	1	1

[a]Docker Swarm is designed to be highly available and may require 5 or more nodes. For a production deployment, the specific requirements can be discussed with the Autonomous Identity Team. Docker Swarm requirements are not correlated to data set size, but to front-end user requirements (for example, the number of users and frequency of use).

[b]For environments with more than 15 million assignments, server requirements will need to be specifically customized.

[c]The Ansible deployment server is seldomly used, except for initial deployment or potentially during future maintenance). If necessary, Ansible could be deployed on one of the front-end servers to eliminate the need for a standalone deployment server.

Production Technical Specifications

Autonomous Identity 2020.6.0 has the following technical specifications for production deployments:

Production Technical Specifications

	Application	Database	Analytics	Admin

Installed Components	Docker	Cassandra	Spark/Docker (Spark Master)	Docker
OS	CentOS	CentOS	CentOS	CentOS
# of Servers	See "Suggested Number of Servers"	See "Suggested Number of Servers"	See "Suggested Number of Servers"	1
RAM (GB)	32	32	64	32
CPUs	8	8	16	8
Non-OS Disk Space (GB)[a]	1000	1000	1000	1000
NFS Shared Mount	1 TB NFS mount shared across all Docker Swarm nodes (if more than 1 node is provisioned) at location separate from the non-OS disk space requirement. For example, /shared.	N/A	1 TB NFS mount shared across all Docker Swarm nodes (if more than 1 node is provisioned) at location separate from the non-OS disk space requirement. For example, /shared.	N/A
Networking	nginx: 443 Docker Manager: 2377 (TCP) Docker Swarm: 7946 & 4789 (UDP) + 7946 & 2049 (TCP)	Client Protocol Port: 9042 Cassandra Nodes: 7000	Spark Master: 7077 Spark Worders: Randomly assigned ports	N/A
Licensing	N/A using Docker CE free version	N/A	N/A	N/A
Software Version	Docker: 19.03.8	Cassandra: 3.11.2	Spark: 2.4.4 Docker: 19.03.8	Docker: 19.03.8
Component Reference	See below.[b]	See below. [c]	See below. [d]	See below. [e]

[a]At root directory "/"

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

[c]http://cassandra.apache.org/doc/latest/operating/hardware.html

https://docs.datastax.com/en/dse-planning/doc/planning/planningHardware.html

[d]https://spark.apache.org/docs/2.2.0/hardware-provisioning.html

https://spark.apache.org/docs/latest/security.html#configuring-ports-for-network-security

[e]https://docs.ansible.com/ansible/latest/index.html

Chapter 3
Deployment Checklist

Use the following checklist to ensure key considerations are covered for your 2020.6.4 deployment:

Deployment Checklist

Check ☑	Requirement	Details
Access		
☐	Remote Access	The Autonomous Identity Team is a global team. To support the needs of client teams, remote access to all servers is required for deployment and support of product.
☐	Root Access	Root access is required to run required package installations (YUM), perform Docker installation, Docker Swarm-based installation applicable boxes, and potential troubleshooting. Please discuss with delivery team if this requirement is a concern. If so, submit a specified contact to run admin tasks.
☐	Service Account	The Autonomous Identity Team should have access to a single service account user (e.g., "autoid"), which will be used throughout as the primary owner of Autonomous Identity specific directories. Specific requirements regarding the service account are specified in this section.
☐	File Transfer Process	The Autonomous Identity Team require access to a file transfer process, which lets specified packages be transferred from the vendor to the client infrastructure.
Service Account		
☐	Autonomous Identity Team Access	Autonomous Identity team members must be able to switch to this user after logging in to the servers
☐	SSH Ability	The service account must be able to passwordless SSH between all Autonomous Identity servers; preferred method is RSA SSH key authentication.
☐	Default Shell	The default shell of the service account must be Bash.
☐	Directory Ownership	Ownership of the following directories must be given to the Service Account. • /data (all servers) • /opt/autoid (all servers) • /shared (if applicable - Docker & Spark servers

		• /tmp/zoran (R, W, E required at least)
☐	Docker Commands	The service account must have permissions to run Docker commands. Note that Docker should NOT need to be installed as a prerequisite; this will be installed by deployment team.

Networking/Internet

☐	Access to the Internet	If available, the front-end servers downloads the required Docker images from the official Autonomous Identity image repository.
☐	SSL Certificates	If SSL is being implemented, SSL certificates are required for the UI, Cassandra nodes, and Spark nodes. These certificates can be generated using one of the following four options: • Self-signed certificates for all 3 components • Valid certificate for the UI and self-signed certificates for Cassandra and Spark nodes (self-signed certs only used in server-server traffic) • Valid and separate certificates for the UI, Cassandra, and Spark • *.domainname.com certificate (wildcard)
☐	Ports Open (Internal)	All internal ports specified in the Networking section of the Environment Specifications need to be opened for the specified servers.
☐	Ports Open (external browser)	The following ports must be accessible from a web browser within the client network: • 443 (Front-end) • 8080 (Spark) • 8081 (Spark)

Required Packages

☐	Dependencies	The following packages must be installed on specified servers as prerequisites: • All Servers: Python 2.7 or Python 3 (3.5+) • Cassandra Servers: java-1.8.0-openjdk-devel.x86_64 • Analytics Servers: java-1.8.0-openjdk-devel.x86_64

Other

☐	Infrastructure Support POC	A point-of-contact (POC) with sufficient access to the infrastructure is required. The POC can support in case of infrastructure blockers arise (e.g., proxy, account access, or port issues).
☐	SELinux	SELinux must be disabled on the Docker boxes. The package "container-selinux" must be present (this can be done as part of the root scripts described in the "Root Access" category).
☐	Components Not Pre-installed	The following software must NOT be pre-installed on the box: • Docker

<table>
<tr><td></td><td></td><td>

- Cassandra

- Spark

If any do come pre-installed, discuss the details with the Delivery Team ahead of time.

</td></tr>
</table>

Glossary

anomaly report	A report that identifies potential anomalous assignments.
as-is predictions	A process where confidence scores are assigned to the entitlements that users have.
confidence score	A score from a scale from 0 to 100% that indicates the strength of correlation between an assigned entitlement and a user's data profile.
data audit	A pre-analytics process that audits the seven data files to ensure data validity with the client.
data ingestion	A pre-analytics process that pushes the seven .csv files into the Cassandra database. This allows the entire training process to be performed from the database.
data sparsity	A reference to data that has null values. Autonomous Identity requires dense, high quality data with very few null values in the user attributes to get accurate analysis scores.
data validation	A pre-analytics process that tests the data to ensure that the content is correct and complete prior to the training process.
driving factor	An association rule that is a key factor in a high entitlement confidence score. Any rule that exceeds a confidence threshold level (e.g., 75%) is considered a driving factor.
entitlement	An entitlement is a specialized type of `assignment`. A user or device with an entitlement gets access rights to specified resources.
insight report	A report that provides metrics on the rules and predictions generated in the analytics run.

recommendation	A process run after the as-is predictions that assigns confidence scores to all entitlements and recommends entitlements that users do not currently have. If the confidence score meets a threshold, set by the `conf_thresh` property in the configuration file, the entitlement will be recommended to the user in the UI console.
resource	An external system, database, directory server, or other source of identity data to be managed and audited by an identity management system.
REST	Representational State Transfer. A software architecture style for exposing resources, using the technologies and protocols of the World Wide Web. REST describes how distributed data objects, or resources, can be defined and addressed.
stemming	A process that occurs after training that removes similar association rules that exist in a parent-child relationship. If the child meets three criteria, then it will be removed by the system. The criteria are: 1) the child must match the parent; 2) the child (e.g., [San Jose, Finance]) is a superset of the parent rule. (e.g., [Finance]); 3) the child and parent's confidence scores are within a +/- range of each other. The range is set in the configuration file.
training	A multi-step process that generates the association rules with confidence scores for each entitlement. First, Autonomous Identity models the frequent itemsets that appear in the user attributes for each user. Next, Autonomous Identity merges the user attributes with the entitlements that were assigned to the user. It then applies association rules to model the sets of user attributes that result in an entitlement access and calculates confidence scores, based on their frequency of appearances in the dataset.