FORGEROCK®

Installation Guide

/ Autonomous Identity 7.1

Latest update: 2020.6.3

,
, ,

Abstract

Guide to installing, and uninstalling ForgeRock® Autonomous Identity software. This software provides the entitlement analytics for your Identity and Access Management (IAM) systems.

Table of Contents

Overview

This guide shows you how to install and deploy Autonomous Identity for intelligent entitlements and role 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 Target 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 airgap installation.

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

Chapter 1 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 scales linearly as the load increases.
- **Powerful UI dashboard**. Autonomous Identity displays your company's entitlements graphically on its UI console. You can immediately investigate those entitlement outliers that could potentially be a security risk. The UI also lets you quickly identify the 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.
- **Automated Workflows**. Autonomous Identity reduces the burden on managers who must manually approve new entitlements, for example assigning access for new hires, by auto-approving high confidence, low-risk access requests and automate the re-certification of entitlements. Predictive recommendations lends itself well to automation, which saves time and cost.
- **Powerful Analytics Engine**. Autonomous Identity's analytics engine is capable of processing millions of access points within a short period of time. 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.
- **Powerful Explainable AI Algorithms**. The Analytics Engine provides transparent and explainable results that lets business users get insight into why the end-user has the access they have, or what access they should have.

Chapter 2
Architecture in Brief

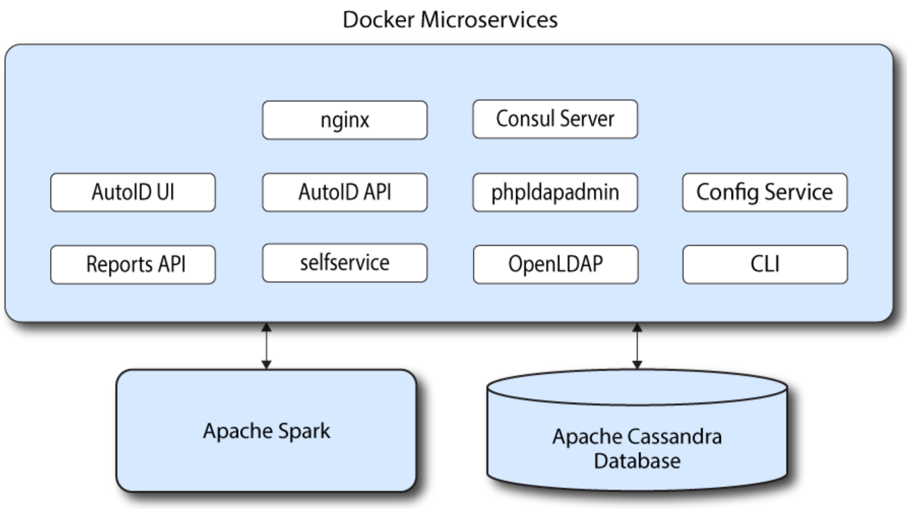
The Autonomous Identity architecture has a simple three-layer conceptual model:

- **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.
  - **Command-Line Interface**. Autonomous Identity supports a command-line interface for easy scripting and automation.
  - **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.
- **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.

Figure 1: A Conceptual Image of the Autonomous Identity Architecture

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. For example, you can configure a two-server deployment with the minimum hardware and software requirements as shown below. Note that the example architecture is typically used for pilot deployments.

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



Chapter 3
Deployment Architectures

To simplify your deployments, ForgeRock provides a deployer script to install Autonomous Identity on a target node. The deployer pulls in an image 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.

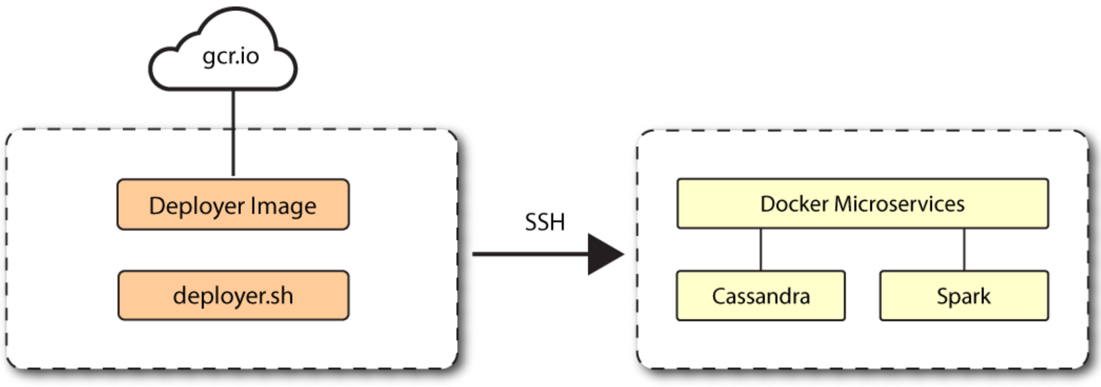
There are five basic deployments, all of them similar, but in slightly different configurations:

- **Lightweight Single-Node Target Deployment**. Deploy Autonomous Identity on a single target machine without the analytics pipeline. This alleviates any direct data load on your target machine, which can be memory intensive. You can deploy this configuration on a 2-core 8 GB virtual machine on a cloud service, such as Google Cloud Platform (GCP), Amazon Web Services (AWS), or others. This configuration is only for evaluation purposes and is outlined in Getting Started.

  Figure 3: A lightweight single-node target deployment.

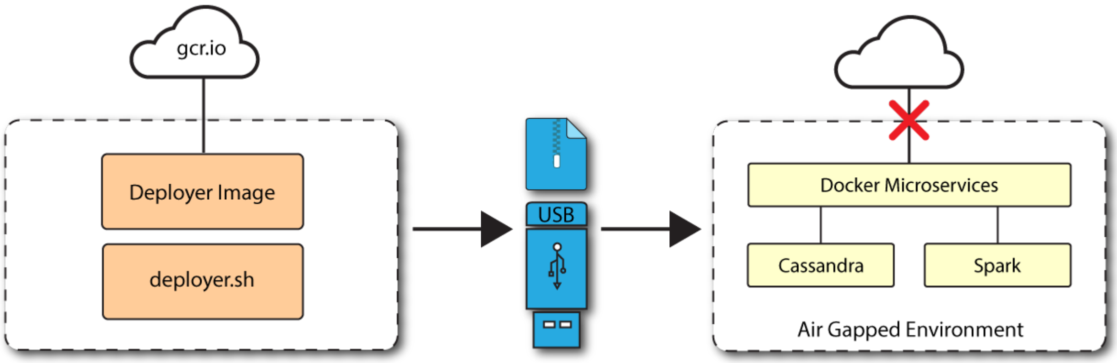
- **Single-Node Target Deployment**. Deploy Autonomous Identity on a single Internet-connected target machine. The deployer script lets you deploy the system from a local laptop or machine or from the target machine itself. The target machine can be on on-prem or on a cloud service, such as Google Cloud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

  Figure 4: A single-node target deployment.



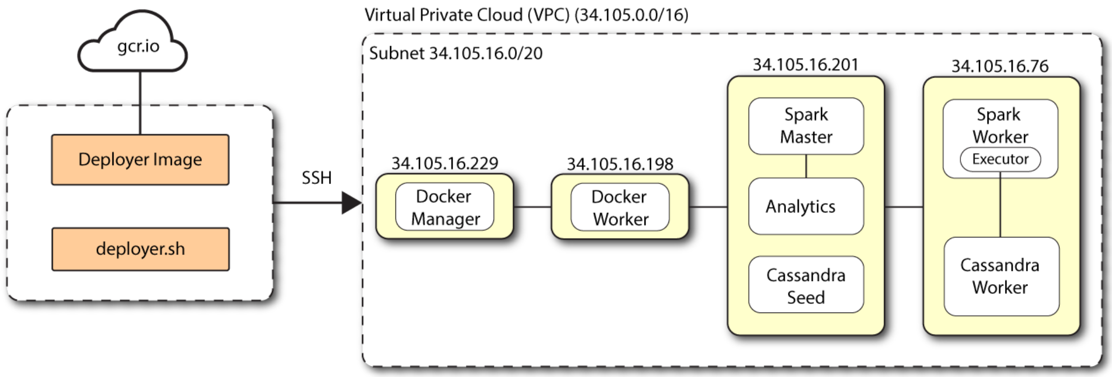
- **Single-Node Air-Gapped Target Deployment**. Deploy Autonomous Identity on a single-node target machine that resides in an air-gapped environment. In an air-gapped environment, the target machine is placed in an enhanced security environment where external Internet access is not available. You transfer 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-Gap Target*".

  Figure 5: An air-gapped environment.



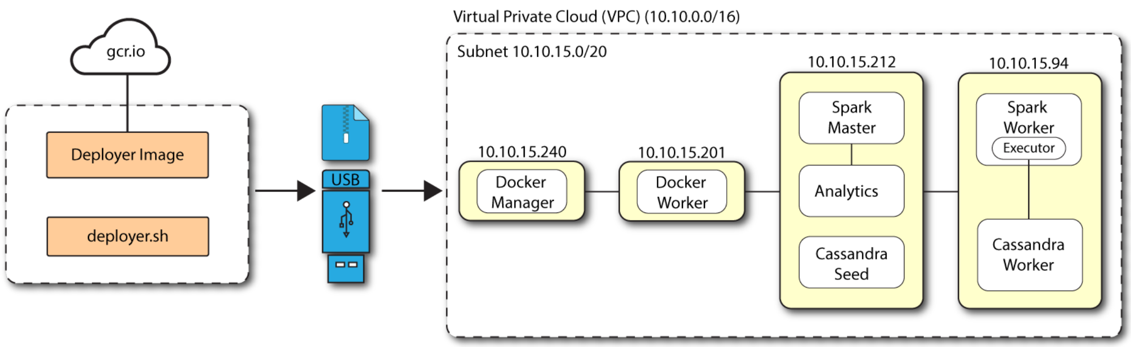
- **Multi-Node Deployment**. Deploy Autonomous Identity on multi-node deployment to distribute the process load on the servers. For installation instruction, see "*Install a Multi-Node Deployment*"

  Figure 6: A multi-node target environment.



- **Multi-Node Air-Gapped Deployment**. Deploy Autonomous Identity a multi-node configuration in an air-gap network. The multinode network has no external Internet connection. For installation instruction, see "*Install a Multi-Node Air-Gapped Deployment*".

Figure 7: A multi-node air-gapped target environment.



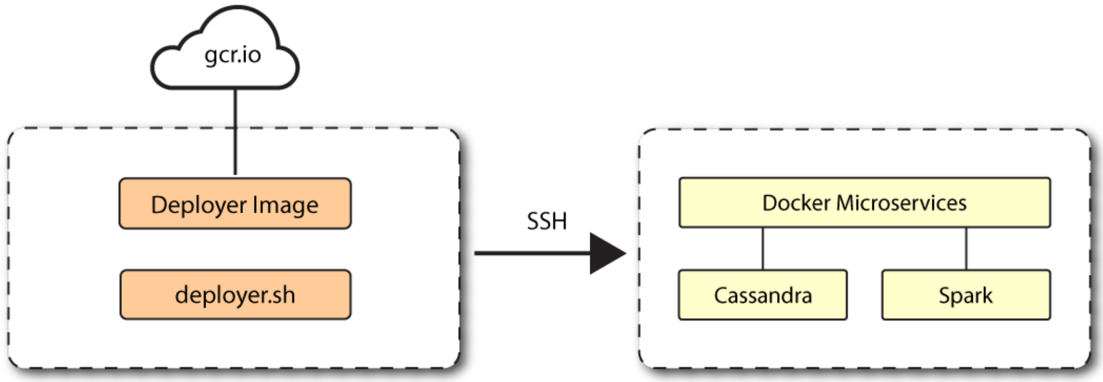
Chapter 4

Install a Single Node Target

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 8: A single-node target deployment.



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

- **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 40GB/partition with 14GB used and 27GB free after running the commands.
- **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.

Set Up the Target 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
   ```

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, 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 and owner to the file.

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

4. Copy your public SSH key, `id_rsa.pub`, to the target machine's `~/.ssh/authorized_keys` file. If your system does not have an `/authorized_keys` directory, create it using **mkdir -p ~/.ssh/authorized_keys**.

5. On the target machine, set the privileges on your `~/.ssh` and `~/.ssh/authorized_keys`.

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

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 <username>@34.70.190.144
   Last login: Sun Jun 14 23:23:36 2020 from 74.125.45.78
   ```

7. If you successfully accessed the remote server, 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
   ```

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:2020.6.2
    create-template
       ...
   d6c7c6f3303e: Pull complete
   Digest: sha256:15225be65417f8bfb111adea37c83eb5e0d87140ed498bfb624a358f43fb48bf
   Status: Downloaded newer image for gcr.io/forgerock-autoid/autoid/dev-compact/
   deployer@sha256:15225be65417f8bfb111a
   dea37c83eb5e0d87140ed498bfb624a358f43fb48bf
   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
   ```

6. The **create-template** command creates a number of configuration files, including 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
   ```

7. 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. The following is an example of the ~/autoid-config/hosts file for a single-node target 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
```

8. 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:
     34.72.28.214:  "10.128.0.52"
   ```

9. Edit other properties in the `~/autoid-config/vars.yml` file, specific to your deployment, such as the following:

   - **Domain name**. For information, see "*Customize the Domain and Namespace*" in the *Admin Guide*.

   - **UI Theme**. Autonomous Identity provides a dark theme mode for its UI. Set the `enable_dark_theme` to `true` to enable it.

- **Session Duration**. The default session duration is set to 30 minutes. You can alter this value by editing the `jwt_expiry` property to a time value in minutes of your choice.

- **SSO**. Autonomous Identity provides a single sign-on (SSO) feature that you can configure with an OIDC IdP provider. For information, see "*Set Up Single Sign-On*" in the *Admin Guide*.

10. The default Autonomous Identity URL will be: `https://autoid-ui.forgerock.com`. To customize your domain name and target environment, see "*Customize the Domain and Namespace*" in the *Admin Guide*.

11. 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`.

    Note

    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
    ```

12. 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
    ```

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

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

14. Run the deployment.

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

Resolve Hostname

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

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

   For example:

   ```
   34.72.28.214  autoid-ui.forgerock.com autoid-selfservice.forgerock.com
   ```

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

   ```
   34.72.28.214  myid-ui.abc.com  myid-selfservice.abc.com
   ```

   For more information on customizing your domain name, see "*Customize the Domain and Namespace*" in the *Admin Guide*.

Access the Dashboard

You can now access the Autonomous Identity console UI.

1. Open a browser, and point it to `https://autoid-ui.forgerock.com/` (or your customized URL: `https://myid-ui.abc.com`).

2. Log in as a test user: **bob.rodgers@forgerock.com**. Enter the password: **Welcome123**.

Check Apache Cassandra

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

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

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 Apache Spark

- 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

Access Self-Service

The self-service feature lets Autonomous Identity users change their own passwords.

- Open a browser and point it to: `https://autoid-selfservice.forgerock.com/`.

Run the Analytics

If the previous steps all check out successfully, you can start an analytics pipeline run, where association rules, confidence scores, predications, and recommendations are determined. Autonomous Identity provides a small demo data set that lets you run the analytics pipeline on. Note for production runs, prepare your company's dataset as outlined in "*Data Preparation*" in the *Admin Guide*.

1. SSH to the target node.
2. Check that the analytics service is running.

   ```
   $ docker ps | grep analytics
   ```

3. If the previous step returns blank, start the analytics. For more information, see "*Data Preparation*" in the *Admin Guide*.

   ```
   $ docker start analytics
   ```

4. Once you have verified that the analytics service has started, you can run the analytics commands. For more information, see "*Run the Analytics Pipeline*" in the *Admin Guide*.

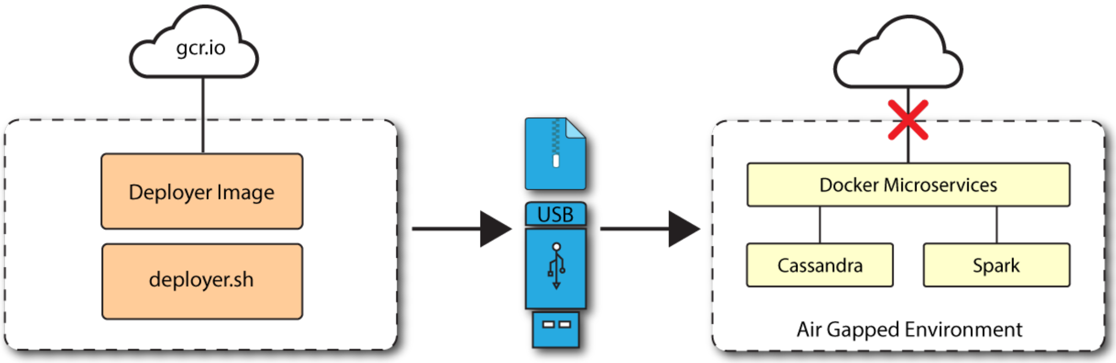
If your analytics pipeline run completes successfully, you have finished your Autonomous Identity installation.

Chapter 5
Install a Single Node Air-Gap Target

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 9: A single-node air-gapped target deployment.



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

- **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 40GB/partition with 14GB used and 27GB free after running the commands.
- **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.

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 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
   ```

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:2020.6.2
    create-template
   ```

4. Make the script executable.

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

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 deployer.sh autoid-packages.tgz 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. 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.

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

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
```

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

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

12. Open a text editor and edit the `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
    ```

13. 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:

    ```
    [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

    [analytics]
    10.128.0.34
    ```

14. 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 `~/autoid-config/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:
      34.70.190.144:  "10.128.0.34"
    ```

15. Edit other properties in the `~/autoid-config/vars.yml` file, specific to your deployment, such as the following:

    - **Domain name**. For information, see "*Customize the Domain and Namespace*" in the *Admin Guide*.
    - **UI Theme**. Autonomous Identity provides a dark theme mode for its UI. Set the `enable_dark_theme` to `true` to enable it.
    - **Session Duration**. The default session duration is set to 30 minutes. You can alter this value by editing the `jwt_expiry` property to a time value in minutes of your choice.
    - **SSO**. Autonomous Identity provides a single sign-on (SSO) feature that you can configure with an OIDC IdP provider. For information, see "*Set Up Single Sign-On*" in the *Admin Guide*.

16. The default Autonomous Identity URL will be: `https://autoid-ui.forgerock.com`. To customize your domain name and target environment, see "*Customize the Domain and Namespace*" in the *Admin Guide*.

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

    Note

    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
    ```

18. 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
    ```

19. Run the deployment.

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

Access the Dashboard

You can now access the Autonomous Identity console UI.

1. Open a browser, and point it to `https://autoid-ui.forgerock.com/` (or your customized URL: `https://myid-ui.abc.com`).
2. Log in as a test user: **`bob.rodgers@forgerock.com`**. Enter the password: **`Welcome123`**.

Check Apache Cassandra

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

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

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 Apache Spark

- 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

Access Self-Service

The self-service feature lets Autonomous Identity users change their own passwords.

- Open a browser and point it to: `https://autoid-selfservice.forgerock.com/`.

Run the Analytics

If the previous steps all check out successfully, you can start an analytics pipeline run, where association rules, confidence scores, predications, and recommendations are determined. Autonomous Identity provides a small demo data set that lets you run the analytics pipeline on. Note for production runs, prepare your company's dataset as outlined in "*Data Preparation*" in the *Admin Guide*.

1. SSH to the target node.

2. Check that the analytics service is running.

```
$ docker ps | grep analytics
```

3. If the previous step returns blank, start the analytics. For more information, see "*Data Preparation*" in the *Admin Guide*.

```
$ docker start analytics
```

4. Once you have verified that the analytics service has started, you can run the analytics commands. For more information, see "*Run the Analytics Pipeline*" in the *Admin Guide*.

If your analytics pipeline run completes successfully, you have finished your Autonomous Identity installation.

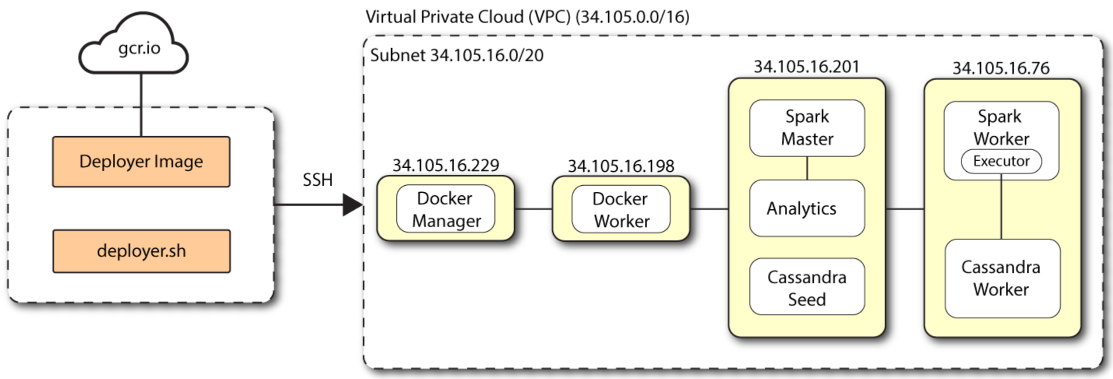
Chapter 6 Install a Multi-Node Deployment

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/or Cassandra nodes. The key is to determine the IP addresses of each node.

Figure 10: A multi-node deployment.



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

- **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 40GB/partition with 14GB used and 27GB free after running the commands.
- **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 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. 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, and analytics hosts. 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.

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, 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
   ```

4. Copy your public SSH key to the target machine's `~/.ssh/authorized_keys` file. If your system does not have an `/authorized_keys` directory, create it using **mkdir -p ~/.ssh/authorized_keys**.

5. On the target machine, set the privileges on your `~/.ssh` and `~/.ssh/authorized_keys`.

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

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 <username>@34.70.190.144
   Last login: Sun Jun 14 23:23:36 2020 from 74.125.45.78
   ```

7. If you successfully accessed the remote server, enter **exit** to end your SSH session.

Install on the Target Machine

Before you begin, make sure you have CentOS 7 installed on your air-gapped 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
   ```

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:2020.6.2
 create-template
```

4. Make the script executable.

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

5. The **create-template** commands creates a number of configuration files, including `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
```

6. 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:

```
[docker-managers]
34.105.16.229

[docker-workers]
34.105.16.198

[docker:children]
docker-managers
docker-workers

[cassandra-seeds]
34.105.16.201

[cassandra-workers]
34.105.16.76

[spark-master]
34.105.16.201

[spark-workers]
34.105.16.76

[analytics]
34.105.16.201
```

7. 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:
  34.105.16.229:  "10.128.0.34"
  34.105.16.198:  "10.128.0.51"
  34.105.16.201:  "10.128.0.54"
  34.105.16.76:  "10.128.0.55"
```

8. Edit other properties in the `~/autoid-config/vars.yml` file, specific to your deployment, such as the following:

    - **Domain name**. For information, see "*Customize the Domain and Namespace*" in the *Admin Guide*.
    - **UI Theme**. Autonomous Identity provides a dark theme mode for its UI. Set the `enable_dark_theme` to `true` to enable it.
    - **Session Duration**. The default session duration is set to 30 minutes. You can alter this value by editing the `jwt_expiry` property to a time value in minutes of your choice.
    - **SSO**. Autonomous Identity provides a single sign-on (SSO) feature that you can configure with an OIDC IdP provider. For information, see "*Set Up Single Sign-On*" in the *Admin Guide*.

9. The default Autonomous Identity URL will be: `https://autoid-ui.forgerock.com`. To customize your domain name and target environment, see "*Customize the Domain and Namespace*" in the *Admin Guide*.

10. 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`.

    Note

    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
```

11. 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
```

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

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

13. Run the deployment.

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

Access the Dashboard

You can now access the Autonomous Identity console UI.

1. Open a browser, and point it to `https://autoid-ui.forgerock.com/` (or your customized URL: `https://myid-ui.abc.com`).
2. Log in as a test user: **bob.rodgers@forgerock.com**. Enter the password: **Welcome123**.

Check Apache Cassandra

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

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

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 Apache Spark

• 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

Access Self-Service

The self-service feature lets Autonomous Identity users change their own passwords.

- Open a browser and point it to: `https://autoid-selfservice.forgerock.com/`.

Run the Analytics

If the previous steps all check out successfully, you can start an analytics pipeline run, where association rules, confidence scores, predications, and recommendations are determined. Autonomous Identity provides a small demo data set that lets you run the analytics pipeline on. Note for production runs, prepare your company's dataset as outlined in "*Data Preparation*" in the *Admin Guide*.

1. SSH to the target node.
2. Check that the analytics service is running.

   ```
   $ docker ps | grep analytics
   ```

3. If the previous step returns blank, start the analytics. For more information, see "*Data Preparation*" in the *Admin Guide*.

   ```
   $ docker start analytics
   ```

4. Once you have verified that the analytics service has started, you can run the analytics commands. For more information, see "*Run the Analytics Pipeline*" in the *Admin Guide*.

If your analytics pipeline run completes successfully, you have finished your Autonomous Identity installation.

Chapter 7

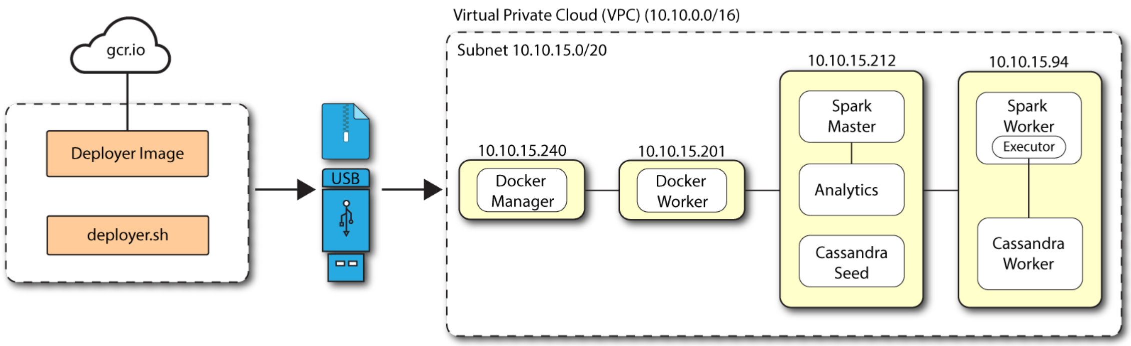
Install a Multi-Node Air-Gapped Deployment

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/or Cassandra nodes. The key is to determine the IP addresses of each node.

Figure 11: A multi-node air-gap deployment.



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

- **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 40GB/partition with 14GB used and 27GB free after running the commands.

- **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 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. 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, and analytics hosts. 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.

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 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
```

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:2020.6.2
 create-template
```

4. Make the script executable.

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

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 deployer.sh autoid-packages.tgz 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. 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
    ```

12. Open a text editor and enter the IP addresses of the target machines in the `~/autoid-config/hosts` file. Make sure the ports are open. The following is an example of the `~/autoid-config/hosts` file:

```
[docker-managers]
10.10.15.240

[docker-workers]
10.10.15.201

[docker:children]
docker-managers
docker-workers

[cassandra-seeds]
10.10.15.212

[cassandra-workers]
10.10.15.94

[spark-master]
10.10.15.212

[spark-workers]
10.10.15.94

[analytics]
10.10.15.212
```

13. 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:
      34.70.190.144:  "10.128.0.34"
    ```

14. Edit other properties in the `~/autoid-config/vars.yml` file, specific to your deployment, such as the following:

    - **Domain name**. For information, see "*Customize the Domain and Namespace*" in the *Admin Guide*.

    - **UI Theme**. Autonomous Identity provides a dark theme mode for its UI. Set the `enable_dark_theme` to `true` to enable it.

    - **Session Duration**. The default session duration is set to 30 minutes. You can alter this value by editing the `jwt_expiry` property to a time value in minutes of your choice.

    - **SSO**. Autonomous Identity provides a single sign-on (SSO) feature that you can configure with an OIDC IdP provider. For information, see "*Set Up Single Sign-On*" in the *Admin Guide*.

15. The default Autonomous Identity URL will be: `https://autoid-ui.forgerock.com`. To customize your domain name and target environment, see "*Customize the Domain and Namespace*" in the *Admin Guide*.

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

    Note

    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
    ```

17. 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
    ```

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

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

19. Run the deployment.

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

Access the Dashboard

You can now access the Autonomous Identity console UI.

1. Open a browser, and point it to `https://autoid-ui.forgerock.com/` (or your customized URL: `https://myid-ui.abc.com`).

2. Log in as a test user: **`bob.rodgers@forgerock.com`**. Enter the password: **`Welcome123`**.

Check Apache Cassandra

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

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

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 Apache Spark

- 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

Access Self-Service

The self-service feature lets Autonomous Identity users change their own passwords.

- Open a browser and point it to: `https://autoid-selfservice.forgerock.com/`.

Run the Analytics

If the previous steps all check out successfully, you can start an analytics pipeline run, where association rules, confidence scores, predications, and recommendations are determined. Autonomous Identity provides a small demo data set that lets you run the analytics pipeline on. Note for production runs, prepare your company's dataset as outlined in "*Data Preparation*" in the *Admin Guide*.

1. SSH to the target node.
2. Check that the analytics service is running.

   ```
   $ docker ps | grep analytics
   ```

3. If the previous step returns blank, start the analytics. For more information, see "*Data Preparation*" in the *Admin Guide*.

   ```
   $ docker start analytics
   ```

4. Once you have verified that the analytics service has started, you can run the analytics commands. For more information, see "*Run the Analytics Pipeline*" in the *Admin Guide*.

If your analytics pipeline run completes successfully, you have finished your Autonomous Identity installation.

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.

Autonomous Identity uses the following ports:

Autonomous Identity Ports

Port	Protocol	Machine	Description
2376	TCP	Docker	Secure Docker client communication. This port is required for the Docker machine, which orchestrates the Docker hosts.
2377	TCP	Docker	Communication between the nodes of a Docker swarm cluster. Only needed on manager nodes.
7946	TCP/UDP	Docker	Communication among nodes for container network discovery.
4789	TCP	Docker	Overlay network traffic.
7001	TCP	Cassandra	Internode communication.
9042	TCP	Cassandra	CQL native transport.
7077	TCP	Spark	Spark master internode communication port.
40040-40045	TCP	Analytics	Spark driver ports for Spark workers to callback.
443	TCP	Autonomous Identity	Port to access the dashboard and API.

Chapter 8
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:

```
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

# 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

# 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:
```

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.