

Installation

This chapter 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](#).



Deployment Architectures

[Learn about the different deployment architectures.](#)



Install Single Node

[Install a single-node Autonomous Identity installation.](#)



Install Single Node Airgap

[Install a single-node air-gapped Autonomous Identity installation.](#)



Install Multi-Node

[Install a multi-node Autonomous Identity installation.](#)



Install Multi-Node Air-Gapped

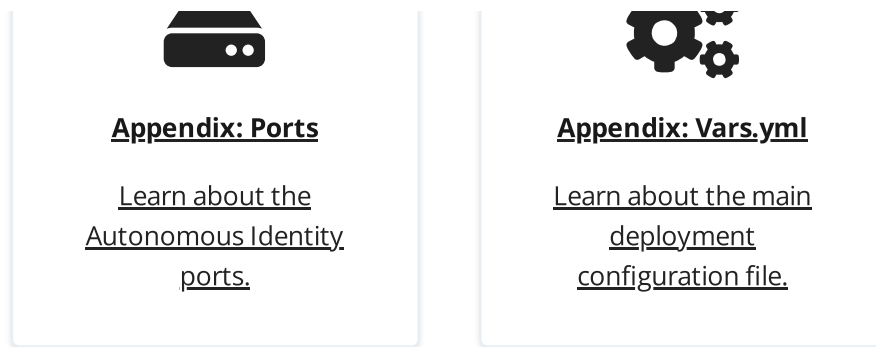
[Install a multi-node Autonomous Identity air-gapped installation.](#)



Upgrade

[Upgrade to the latest version.](#)





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

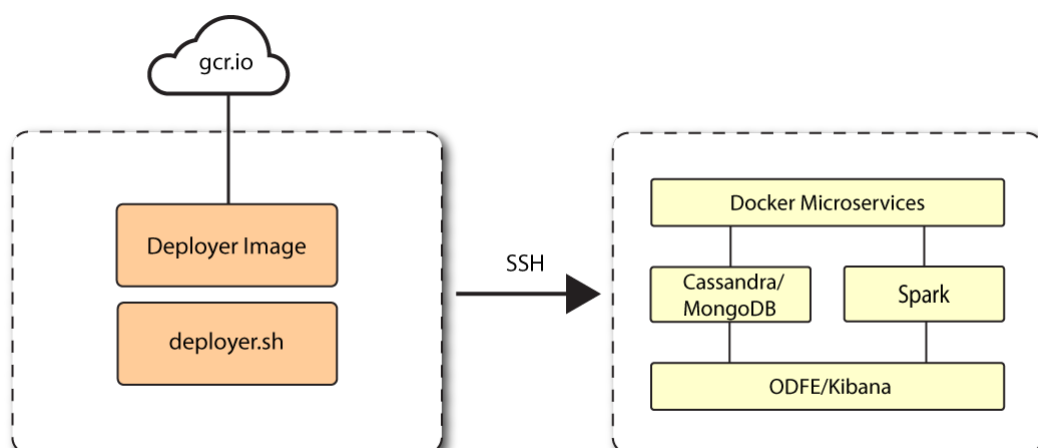
Deployment Architectures

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:

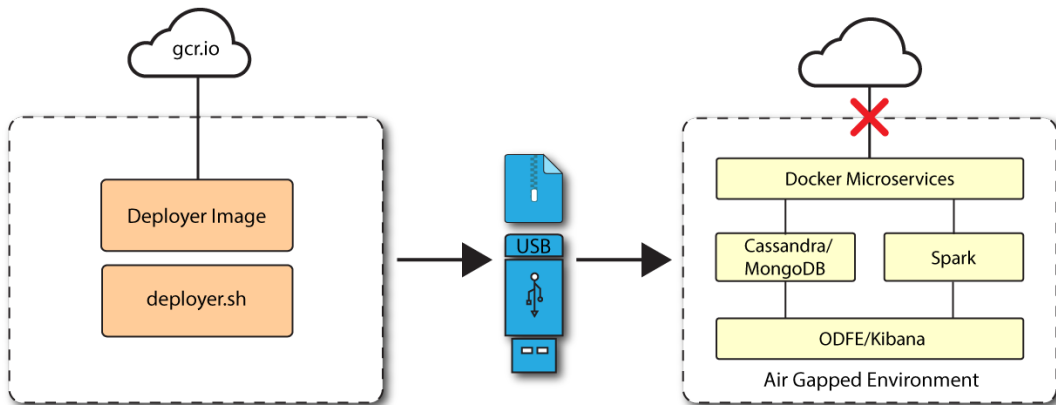
- **Single-Node Target Deployment.** Deploy Autonomous Identity on a single Internet-connected target machine. The deployer script lets you deploy the system from a local laptop or machine or from the target machine itself. The target machine can be on on-prem or on a cloud service, such as Google Cloud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

Figure 2: A single-node deployment.



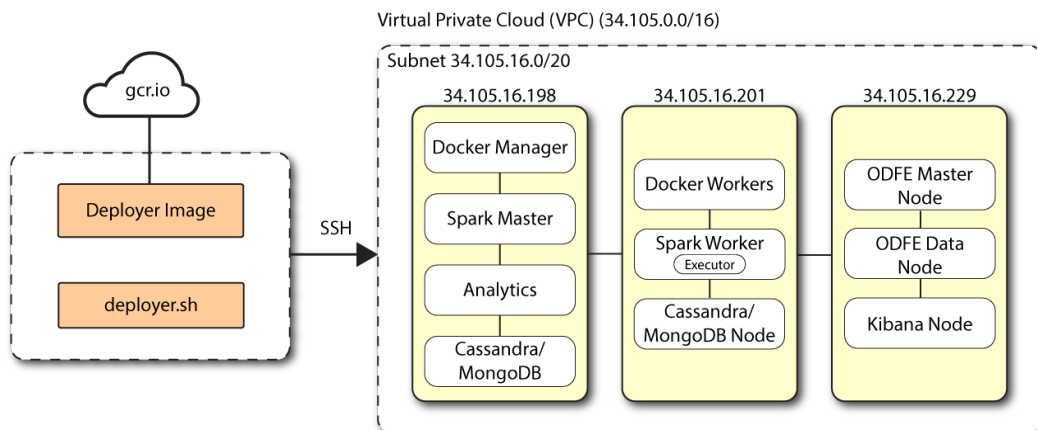
- **Single-Node Air-Gapped Target Deployment.** Deploy Autonomous Identity on a single-node target machine that resides in an air-gapped environment. In an air-gapped environment, the target machine is placed in an enhanced security environment where external Internet access is not available. You transfer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



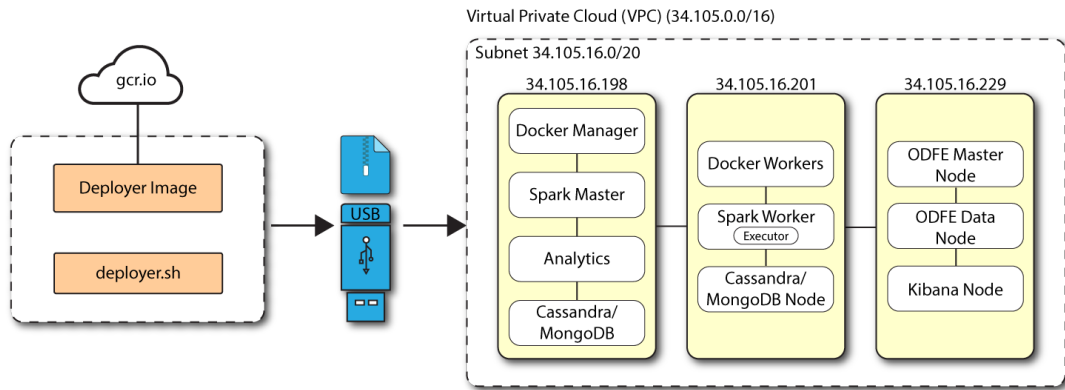
- **Multi-Node Deployment.** Deploy Autonomous Identity on multi-node deployment to distribute the process load on the servers. For installation instruction, see [Install a Multi-Node](#).

Figure 4: A multi-node 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-gap environment.

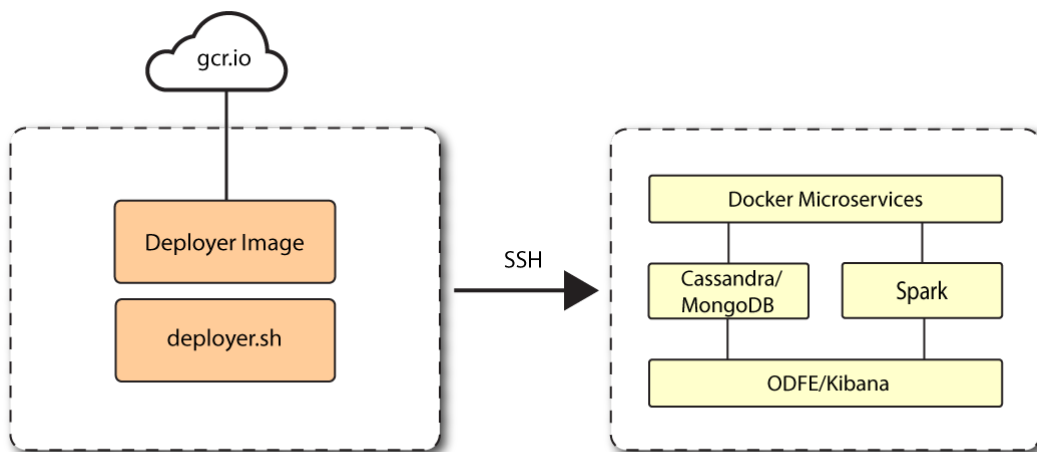


Install a Single Node Deployment

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 chapter, 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 you 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 deployer 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}
```

7. Logout of the user account.

```
$ logout
```

8. Re-login using created user. Login with the user created for the deployer machine. For example, autoid.

```
$ su - autoid
```

Set Up SSH on the Deployer

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

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

```
$ cd ~/.ssh
```

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

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

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

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

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


4. Change the privileges and owner to the file.

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

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

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

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

NOTE

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

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

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

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

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

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

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

Install 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:2021.8.0 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. 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 `ansible.cfg` file in step 1, 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:
 - **AI Product.** Do not change this property.

```
ai_product: auto-id
```

- **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_dir: /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.

```
enable_dark_theme: false
```

- **Offline Mode.** Do not change this property. The property is for air-gap deployments only and should be kept to false.
- **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: cassandra
```

- **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.** This property has three possible values:
 - **Local.** Local indicates that sets up elasticsearch with local accounts and enables the Autonomous Identity UI features: self-service and manage identities. Local auth mode should be enabled for demo environments only.
 - **SSO.** SSO indicates that single sign-on (SSO) is in use. With SSO only, the Autonomous Identity UI features, self-service and manage identities pages, is not available on the system but is managed by the SSO provider. The login page displays "Sign in using OpenID." For more information, see [Set Up SSO](#).
 - **LocalAndSSO.** LocalAndSSO indicates that SSO is used and local account features, like self-service and manage identities are available to the user. The login page displays "Sign in using OpenID" and a link "Or sign in via email".

```
authentication_option: "Local"
```

- **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".

```
access_log_enabled: true
```

- **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"
```

- **Local Auth Mode Password.** When `authentication_option` is set to `Local`, the `local_auth_mode_password` sets the password for the login user.
- **SSO.** Use these properties to set up SSO. For more information, see [Set Up SSO](#).
- **MongoDB Settings.** Use these settings to configure a MongoDB cluster. These settings are not needed for single node deployments.
- **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
  auth_type: 'jwt'
  signiture_key_id: 'service1-hmac'
  signiture_algorithm: 'hmac-sha256'
  max_memory: 4096M
  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
```

```
[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]

[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

Despite the presence of special characters in the examples below, do not include special characters, such as `&` or `$`, in your production `vault.yml` passwords as it will result in a failed deployer process.

```

configuration_service_vault:
  basic_auth_password: ~@C~0>@%^() -_+= |<Y*$rH&&/m#g{?-

```



```
o!z/1}2??3=!*&
```

```
cassandra_vault:
```

```
  cassandra_password: ~@C~0>@%^() -_+=|<Y*$$rH&&/m#g{?-  
o!z/1}2??3=!*&
```

```
  cassandra_admin_password: ~@C~0>@%^() -_+=|  
<Y*$$rH&&/m#g{?-o!z/1}2??3=!*&
```

```
  keystore_password: Acc#1234  
  truststore_password: Acc#1234
```

```
mongo_vault:
```

```
  mongo_admin_password: ~@C~0>@%^() -_+=|<Y*$$rH&&/m#g{?-  
o!z/1}2??3=!*&
```

```
  mongo_root_password: ~@C~0>@%^() -_+=|<Y*$$rH&&/m#g{?-  
o!z/1}2??3=!*&
```

```
  keystore_password: Acc#1234  
  truststore_password: Acc#1234
```

```
elastic_vault:
```

```
  elastic_admin_password: ~@C~0>@%^() -_+=|  
<Y*$$rH&&/m#g{?-o!z/1}2??3=!*&
```

```
  elasticsearch_password: ~@C~0>@%^() -_+=|  
<Y*$$rH&&/m#g{?-o!z/1}2??3=!*&
```

```
  keystore_password: Acc#1234  
  truststore_password: Acc#1234
```

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

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

For example:

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

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

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

For more information on customizing your domain name, 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](#)

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

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

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

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

http://localhost:8080/
```

Start the Analytics

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

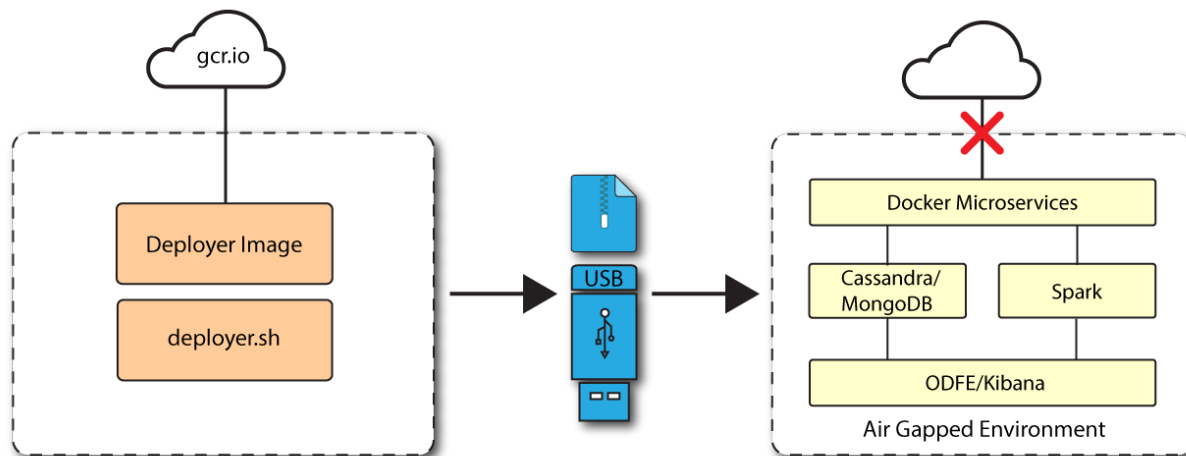
For more information, see [Set Entity Definitions](#).

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 chapter, 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 \(gcr.io\)](https://www.forgerock.org/docs/autoid/latest/forge-rock/google-cloud-registry.html). 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.
- **Docker Required on Air-Gap Machines.** When installing the Autonomous Identity binaries on the air-gap machine using a tar file, you must also manually install Docker 20.10.7 onto the machine.
- **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 you 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 deployer 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}
```

7. Logout of the user account.

```
$ logout
```

8. Re-login using created user. Login with the user created for the deployer machine. For example, autoid.

```
$ su - autoid
```


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:2021.8.0 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 on the Air-Gap Target

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

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

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

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

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

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

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

6. Change to the configuration directory.

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

7. 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:2021.8.0
```

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 `ansible.cfg` file in step 1, 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.

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 2021.8.0 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:
 - **AI Product.** Do not change this property.

```
ai_product: auto-id
```

- **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_dir: /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`.

```
enable_dark_theme: false
```

- **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: cassandra
```

- **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.** This property has three possible values:
 - **Local.** Local indicates that sets up elasticsearch with local accounts and enables the Autonomous Identity UI features: self-service and manage identities. Local auth mode should be enabled for demo environments only.
 - **SSO.** SSO indicates that single sign-on (SSO) is in use. With SSO only, the Autonomous Identity UI features, self-service and manage identities pages, is not available on the system but is managed by the SSO provider. The login page displays "Sign in using OpenID." For more information, see [Set Up SSO](#).
 - **LocalAndSSO.** LocalAndSSO indicates that SSO is used and local account features, like self-service and manage identities are available to the user. The login page displays "Sign in using OpenID" and a link "Or sign in via email".

```
authentication_option: "Local"
```

- **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"
```

- **Local Auth Mode Password.** When `authentication_option` is set to Local, the `local_auth_mode_password` sets the password for the login user.
- **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
  auth_type: 'jwt'
  signiture_key_id: 'service1-hmac'
  signiture_algorithm: 'hmac-sha256'
  max_memory: 4096M
  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
```

```
[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]
```

```
[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

Despite the presence of special characters in the examples below, do not include special characters, such as `&` or `$`, in your production `vault.yml` passwords as it will result in a failed deployer process.

```
configuration_service_vault:
```

```
  basic_auth_password: ~@C~0>@%^()_+|=|<Y*$$rH&&/m#g{?-o!z/1}2??3=!*&
```

```
cassandra_vault:
```

```
  cassandra_password: ~@C~0>@%^()_+|=|<Y*$$rH&&/m#g{?-o!z/1}2??3=!*&
```

```
  cassandra_admin_password: ~@C~0>@%^()_+|=|<Y*$$rH&&/m#g{?-o!z/1}2??3=!*&
```

```
  keystore_password: Acc#1234
```

```
  truststore_password: Acc#1234
```

```
mongo_vault:
```

```
  mongo_admin_password: ~@C~0>@%^()_+|=|<Y*$$rH&&/m#g{?-o!z/1}2??3=!*&
```

```

mongo_root_password: ~@C~0>@%^() -_+=| <Y*$$rH&&/m#g{?-
o!z/1}2??3=!*&
keystore_password: Acc#1234
truststore_password: Acc#1234

elastic_vault:
  elastic_admin_password: ~@C~0>@%^() -_+=|
  <Y*$$rH&&/m#g{?-o!z/1}2??3=!*&
  elasticsearch_password: ~@C~0>@%^() -_+=|
  <Y*$$rH&&/m#g{?-o!z/1}2??3=!*&
  keystore_password: Acc#1234
  truststore_password: Acc#1234

```

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

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

For example:

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

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

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

For more information on customizing your domain name, 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 steps are required and are critical for a successful analytics process.

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

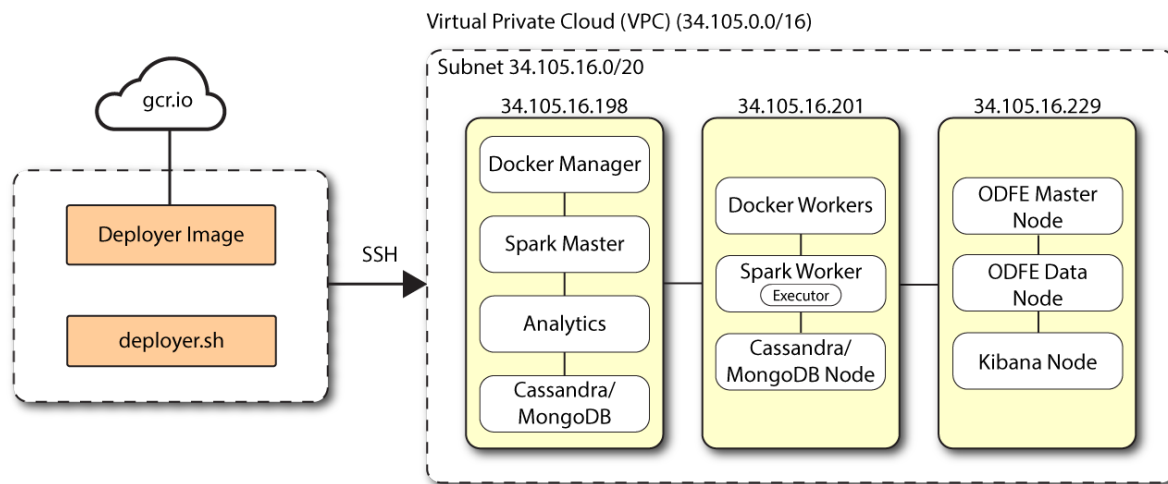
Install a Multi-Node Deployment

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 chapter, we use CentOS 7 as its base operating system.
- **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_dir: /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 you 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}
```

7. Logout of the user account.

```
$ logout
```

8. Re-login using created user. Login with the user created for the deployer machine. For example, autoid.

```
$ su - autoid
```

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 `~/.ssh/authorized_keys`, create it using `sudo mkdir -p ~/.ssh`, then `sudo touch ~/.ssh/authorized_keys`.

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

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

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

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

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

7. 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`.

- o 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.

- o 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.

- o 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:2021.8.0 create-
template
```

4. 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:

- **AI Product.** Do not change this property.

```
ai_product: auto-id
```

- **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.yml` 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`.

```
enable_dark_theme: false
```

- **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: cassandra
```

- **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.** This property has three possible values:
 - **Local.** `Local` indicates that sets up elasticsearch with local accounts and enables the Autonomous Identity UI features: self-service and manage identities. Local auth mode should be enabled for demo environments only.
 - **SSO.** `SSO` indicates that single sign-on (SSO) is in use. With SSO only, the Autonomous Identity UI features, self-service and manage identities pages, is not available on the system but is managed by the SSO provider. The login page displays "Sign in using OpenID." For more information, see [Set Up SSO](#).
 - **LocalAndSSO.** `LocalAndSSO` indicates that SSO is used and local account features, like self-service and manage identities are available to the user. The login page displays "Sign in using OpenID" and a link "Or sign in via email".

```
authentication_option: "Local"
```

- **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"
```

- **Local Auth Mode Password.** When `authentication_option` is set to `Local`, the `local_auth_mode_password` sets the password for the login user.
- **SSO.** Use these properties to set up SSO. For more information, see [Set Up SSO](#).
- **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/0/1YlRRiW57bWU3zjliW3V
dguJar5i

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

2hwR17nqPZZMMR+uQThg/zV9r0zHJmkqZts04UJSi1G9euLCYrzW2hd
oPuCrEDhu

Vsi5+nwAgYR9dP2oWkmGN1dwRe0ixSIM2UzFgpaXZaMOG6VztmFr1VX
```



```
h8oFDRGM0
```

```
cGrFHcnGF7oUGfWnI2Cekngk64dHA2qD7WxXPbQ/svn9EfTY5aPw51X  
zKA87Ds8p
```

```
KHVFUYvmA6wVsb/rIGLwc+XZ1b6M9gqHn1XSpsnYRjF6UzfRcRR2Wy  
CxLZELaqu
```

```
iKxLKB5FYqMBH7Sgg3qBCtE53vZ7T1nefq5RFzmykviYP63Uhu/A2EQ  
atrMnaFP1
```

```
TTG5CaPjob45CBSyMrheYRWKqxdWN93BTgiTW7p0U6RB0/OCUbsVX6I  
G3I9N8Uqt
```

```
l8Kc+7a0mtUqFkwo8w30prIOjStMroknNsuK9KTUiPu2cj7gwYQ574v  
V3hQvQPAr
```

```
hhb9ohKr0zoPQt31iTj0FDkJzPepezqeq8F51HB56RZKpXdRTfY8G6  
0a0T68cV5
```

```
vP106T/okFKr141FQ3CyYN5eRHyRTK99zTytrjoP2EbtIZ18z+bg/an  
gRHYNzbgk
```

```
lc3jpiGzs1ZWHD0nx0mHCMhU4usEcFbV6F10xzlwrSEhHkeiununlCs  
NHatiDgzp
```

```
ZWLnP/mXKV992/Jhu0Z577DH1h+3JIYx0PceB9yzACJ8MNARHF7QpBk  
htuGMGZpF
```

```
T+c73exupZFxItXs1Bnhe3djgE3MKKyYvxNUIbcTJoe7nhVMrw0/71B  
SpVLvC4p3  
wR700U0LDaGGQps1GtiE56SemgoP
```

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
  auth_type: 'jwt'
  signature_key_id: 'service1-hmac'
  signature_algorithm: 'hmac-sha256'
  max_memory: 4096M
  mapping_entity_type: /common/mappings
  datasource_entity_type: /common/datasources
```

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

[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]

[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

Despite the presence of special characters in the examples below, do not include special characters, such as `&` or `$`, in your production `vault.yml` passwords as it will result in a failed deployer process.

```
configuration_service_vault:
```

```
  basic_auth_password: ~@C~0>@%^()-_+=|<Y*$$rH&&/m#g{?-o!z/1}2??3=!*&
```

```
cassandra_vault:
```

```
  cassandra_password: ~@C~0>@%^()-_+=|<Y*$$rH&&/m#g{?-o!z/1}2??3=!*&
```

```
  cassandra_admin_password: ~@C~0>@%^()-_+=|<Y*$$rH&&/m#g{?-o!z/1}2??3=!*&
```

```
  keystore_password: Acc#1234
```

```
  truststore_password: Acc#1234
```

```
mongo_vault:
```

```
  mongo_admin_password: ~@C~0>@%^()-_+=|<Y*$$rH&&/m#g{?-o!z/1}2??3=!*&
```

```
mongo_root_password: ~@C~0>@%^()-_+=|<Y*$$rH&&/m#g{?-
o!z/1}2??3=!*&
keystore_password: Acc#1234
truststore_password: Acc#1234

elastic_vault:
  elastic_admin_password: ~@C~0>@%^()-_+=|
  <Y*$$rH&&/m#g{?-o!z/1}2??3=!*&
  elasticsearch_password: ~@C~0>@%^()-_+=|
  <Y*$$rH&&/m#g{?-o!z/1}2??3=!*&
  keystore_password: Acc#1234
  truststore_password: Acc#1234
```

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 on the target node. The following domain names must resolve to the IP address of the target node: `<target-environment>-ui.<domain-name>`.
2. If DNS cannot resolve target node hostname, edit it locally on the machine that you want to access Autonomous Identity using a browser. Open a text editor and add an entry in the `/etc/hosts` (Linux/Unix) file or

[path]`C:\Windows\System32\drivers\etc\hosts" (Windows) for the self-service and UI services for each managed target node.

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

For example:

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

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

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

For more information on customizing your domain name, 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 steps are required and are critical for a successful analytics process.

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

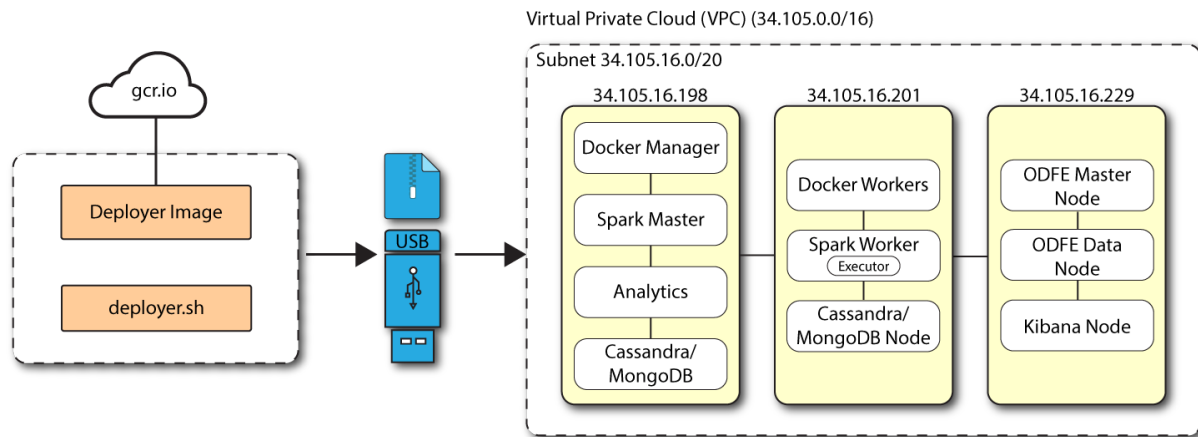
Install a Multi-Node Air-Gapped Deployment

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 chapter, we use CentOS 7 as its base operating system.
- **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_dir: /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.
- **Docker Required on Air-Gap Machines.** When installing the Autonomous Identity binaries on the air-gap machine using a tar file, you must also manually install Docker 20.10.7 onto the machine.
- **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 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 you 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
```

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:2021.8.0 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 on the Air-Gap Target

Before you begin, make sure you have CentOS 7 and Docker 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 the **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 2021.8.0 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:
 - **AI Product.** Do not change this property.

```
ai_product: auto-id
```

- **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.yml` file to point to the correct location:

```
analytics_data_dir: /data
analytics_conf_dir: /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`.

```
enable_dark_theme: false
```

- **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: cassandra
```

- **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.** This property has three possible values:
 - **Local.** Local indicates that sets up elasticsearch with local accounts and enables the Autonomous Identity UI features: self-service and manage identities. Local auth mode should be enabled for demo environments only.
 - **SSO.** SSO indicates that single sign-on (SSO) is in use. With SSO only, the Autonomous Identity UI features, self-service and manage identities pages, is not available on the system but is managed by the SSO provider. The login page displays "Sign in using OpenID." For more information, see [Set Up SSO](#).
 - **LocalAndSSO.** LocalAndSSO indicates that SSO is used and local account features, like self-service and manage identities are available to the user. The login page displays "Sign in using OpenID" and a link "Or sign in via email".

```
authentication_option: "Local"
```

- **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"
```

- **Local Auth Mode Password.** When `authentication_option` is set to Local, the `local_auth_mode_password` sets the password for the login user.
- **SSO.** Use these properties to set up SSO. For more information, see [Set Up SSO](#).
- **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/0/1YlRRiW57bWU3zjliW3V  
dguJar5i  
  
Z+1a8lI+0S9pWynbv9+Ao0aXFjSJYVxAm/w7DJbVRGcPhsPmExiSBDw  
8szfQ8PAU  
  
2hwR17nqPZZMMR+uQThg/zV9r0zHJmkqZts04UJSilG9euLCYrzW2hd  
oPuCrEDhu  
  
Vsi5+nwAgYR9dP2oWkmGN1dwRe0ixSIM2UzFgpaXZaM0G6VztmFr1VX  
h8oFDRGM0  
  
cGrFHcnGF7oUGfWnI2Cekngk64dHA2qD7WxXPbQ/svn9EfTY5aPw5lX  
zKA87Ds8p  
  
KHVFUYvmA6wVsxbr/riGLwc+XZ1b6M9gqHn1XSpsnYRjF6UzfrCRR2Wy  
CxLZELaqu  
  
iKxLKB5FYqMBH7Sqq3qBCtE53vZ7T1nefq5RFzmykviYP63Uhu/A2EQ  
atrMnaFP1  
  
TTG5CaPjob45CBSyMrheYRWKqxdWN93BTgiTW7p0U6RB0/OCUbsVX6I  
G3I9N8Uqt  
  
l8Kc+7a0mtUqFkwo8w30prIOjStMroKxNsuK9KTUiPu2cj7gwYQ574v  
V3hQvQPAr
```

```
hhb9ohKr0zoPQt31iTj0FDkJzPepezqeq8F51HB56RZKpXdRTfY8G6  
0a0T68cV5
```

```
vP106T/okFKr141FQ3CyYN5eRHyRTK99zTytrjoP2EbtIZ18z+bg/an  
gRHYNzbgk
```

```
lc3jpiGzs1ZWHD0nx0mHCMhU4usEcFbV6F10xzlwrSEhHkeiununlCs  
NHatiDgzp
```

```
ZWLnP/mXKV992/Jhu0Z577DH1h+3JIYx0PceB9yzACJ8MNARHF7QpBk  
htuGMGZpF
```

```
T+c73exupZFxItXs1Bnhe3djgE3MKKyYvxNUIbcTJoe7nhVMrw0/71B  
SpVLvC4p3  
wR700U0LDaGGQps1GtiE56SemgoP
```

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  
  auth_type: 'jwt'  
  signature_key_id: 'service1-hmac'  
  signature_algorithm: 'hmac-sha256'  
  max_memory: 4096M  
  mapping_entity_type: /common/mappings  
  datasource_entity_type: /common/datasources
```

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
```

```
[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]
```

```
[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

Despite the presence of special characters in the examples below, do not include special characters, such as `&` or `$`, in your production `vault.yml` passwords as it will result in a failed deployer process.

```
configuration_service_vault:
  basic_auth_password: ~@C~0>@%^()-_+=|<Y*$$rH&&/m#g{?-o!z/1}2??3=!*&

cassandra_vault:
  cassandra_password: ~@C~0>@%^()-_+=|<Y*$$rH&&/m#g{?-o!z/1}2??3=!*&
  cassandra_admin_password: ~@C~0>@%^()-_+=|<Y*$$rH&&/m#g{?-o!z/1}2??3=!*&
  keystore_password: Acc#1234
  truststore_password: Acc#1234

mongo_vault:
  mongo_admin_password: ~@C~0>@%^()-_+=|<Y*$$rH&&/m#g{?-o!z/1}2??3=!*&
  mongo_root_password: ~@C~0>@%^()-_+=|<Y*$$rH&&/m#g{?-o!z/1}2??3=!*&
  keystore_password: Acc#1234
  truststore_password: Acc#1234

elastic_vault:
  elastic_admin_password: ~@C~0>@%^()-_+=|<Y*$$rH&&/m#g{?-o!z/1}2??3=!*&
  elasticsearch_password: ~@C~0>@%^()-_+=|<Y*$$rH&&/m#g{?-o!z/1}2??3=!*&
  keystore_password: Acc#1234
  truststore_password: Acc#1234
```

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
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

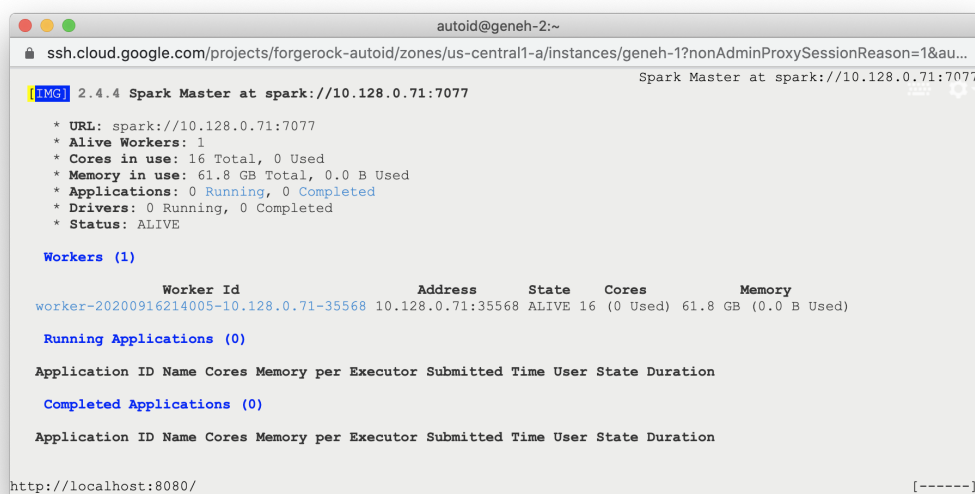
Check Spark:

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](#)



The screenshot shows a terminal window with the following output:

```
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
```

Start the Analytics

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

For more information, 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](#).

WARNING

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

Upgrade Paths

The upgrade assumes the following upgrade paths depends on your current deployment version. The preferred upgrade path is from your current version to the next major release version to the latest patch version. The following chart summarizes these upgrade paths:

Table 1: Upgrade Paths

Version	Upgrade To	See
---------	------------	-----

Version	Upgrade To	See
2021.3.x (2021.3.0–2021.3.5)	2021.3.0–2021.3.4 → 2021.3.5 → 2021.8.0	Upgrading Autonomous Identity to 2021.3.5 , Upgrade from Autonomous Identity 2021.3.5 to 2021.8.0

IMPORTANT

After you upgrade your Autonomous Identity version to 2021.8.0, ForgeRock strongly encourages you to upgrade your Autonomous Identity version to 2021.8.4, which includes important security fixes and features.

Upgrade from Autonomous Identity 2021.3.5 to 2021.8.0

The following instructions are for an upgrade from Autonomous Identity version **2021.3.5** to **2021.8.0**.

NOTE

If you are upgrading from Autonomous Identity version 2021.3.0 to 2021.3.4, you must first upgrade to 2021.3.5, then to version 2021.8.0. See [Upgrading Autonomous Identity to 2021.3.5](#).

Upgrade to version 2021.3.5 to 2021.8.0:

1. On the deployer machine, back up the 2021.3.5 `~/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. 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 autoid@<Target-IP-Address>
```

```
Last login: Tue Aug 15 18:19:14 2021
```

7. Stop the stack.

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

You should see:

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

9. Enter **exit** to end your SSH session.
10. From the deployer, repeat the restart Docker command:

```
$ 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:2021.8.0 create-template
```

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.

IMPORTANT

You must keep your configuration settings consistent from one system to another. For 2021.3.x to 2021.8.x upgrades, do **not** copy-n-paste your hosts file as they are slightly different.

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
```

16. SSH to the target node.
17. Stop the Spark master and workers.

```
$ /opt/autoid/spark/spark-3.0.1-bin-hadoop2.7/sbin/stop-all.sh
```

18. Rename two `log4j` jar files in the `/opt/autoid/elastic/opendistroforelasticsearch-1.9.0/lib` folder. The upgrade installs an older version of the files, and another upgrade to the latest patch release installs the new version of these files.

```
$ cd /opt/autoid/elastic/opedistroforelasticsearch-1.9.0/lib/

$ sudo mv log4j-api-2.17.0.jar log4j-api-2.17.0.jar.bak

$ sudo mv log4j-core-2.17.0.jar log4j-core-2.17.0.jar.bak
```

19. Exit your SSH session.

20. Run the upgrade.

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

21. SSH to the target server.

IMPORTANT

If you freshly installed 2021.8.0 or run an upgrade where the log4j patch was not applied previously, you can skip steps 22–25.

22. On the target server, edit the `/opt/autoid/res/jas/docker-compose.yml` and set the `JAS_AUTH_ENABLED` to `true`.

```
JAS_AUTH_ENABLED=true
```

23. Restart the JAS container.

```
$ docker stack rm jas

$ docker stack deploy -c /opt/autoid/res/jas/docker-
compose.yml jas
```

24. Remove the nginx container.

```
$ docker stack rm nginx
```

25. Redeploy the stack.

```
$ docker stack deploy -c /opt/autoid/res/nginx/docker-
compose.yml nginx
```

26. Log out and then log back in to Autonomous Identity.

27. Add a reference to the Autonomous Identity UI to your `/etc/hosts` or DNS server.

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

28. Redo the attribute mappings. See [Set Attribute Mappings](#).
29. Then, run the analytics pipeline with a new role mining analytics step, *analytics mine*:

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

You have successfully upgraded your Autonomous Identity server to 2021.8.0. Upgrade your system to the latest patch release for the important new features and security fixes.

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-addresses#

api:
  authentication_option: "Local" # Values:
  "Local", "SSO", "LocalAndSSO"
  access_log_enabled: true # Enable
access logs
  jwt_expiry: "30 minutes" # Default
session duration
  jwt_secret_file: "{{ install_path }}/jwt/secret.txt" #
Location of JWT secret file
  jwt_audience: "http://my.service"
  oidc_jwks_url: "na"
  local_auth_mode_password: Welcome123

# set the following API parameters when # SSO and LdapAndSSO
properties
# authentication_option is SSO or LdapAndSSO
# oidc_issuer:
# oidc_auth_url
# oidc_token_url:

```



```
# oidc_user_info_url:
# oidc_callback_url:
# oidc_jwks_url:
# oidc_client_scope:
# oidc_groups_attribute:
# oidc_uid_attribute:
# oidc_client_id:
# oidc_client_secret:
# admin_object_id:
# entitlement_owner_object_id:
# executive_object_id:
# supervisor_object_id:
# user_object_id:
# application_owner_object_id:
# role_owner_object_id:
# role_engineer_object_id:
# oidc_end_session_endpoint:
# oidc_logout_redirect_url:

# mongo config starts

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

# custom key
# password for inter-process authentication
#
# please regenerate this file on production environment with
command 'openssl rand -base64 741'
#mongodb_keyfile_content: |
#
8pYcxvCqoe89kcp33KuTtKVf5MoHGEFjTnudrq5BosvWRoIxLowmdjrmUpVfAivh
#
CHjqM6w0zVBytAxH1lW+7teMYe6eDn2S/0/1YlRRiW57bWU3zjliW3VdguJar5i9
#
Z+1a8lI+0S9pWynbv9+Ao0aXFjSJYVxAm/w7DJbVRGcPhsPmExiSBDw8szfQ8PAU
#
2hwRl7nqPZZMMR+uQThg/zV9r0zHJmkqZts04UJSilG9euLCYrzW2hdoPuCrEDhu
#
Vsi5+nwAgYR9dP2oWkmGN1dwRe0ixSIM2UzFgpaXZaM0G6VztmFr1VXh8oFDRGM0
#
cGrFHcnGF7oUGfWnI2Cekngk64dHA2qD7WxXPbQ/svn9EfTY5aPw5lXzKA87Ds8p
```

```
#
KHVFUYvmA6wVsxbr/riGLwc+XZ1b6M9gqHn1XSpsnYRjF6UzfRcRR2WyCxLZELaqu
#
iKxLKB5FYqMBH7Sqq3qBCtE53vZ7T1nefq5RFzmykviYP63Uhu/A2EQatrMnaFP1
#
TTG5CaPjob45CBSyMrheYRWKqxdWN93BTgiTW7p0U6RB0/OCUbsVX6IG3I9N8Uqt
#
18Kc+7a0mtUqFkwo8w30prIOjStMrokxNsuK9KTUiPu2cj7gwYQ574vV3hQvQPAR
#
hnb9ohKr0zoPQt31iTj0FDkJzPepezqeq8F51HB56RZKpXdRTfy8G60a0T68cV5
#
vP106T/okFKr141FQ3CyYN5eRHyRTK99zTytrjoP2EbtIZ18z+bg/angRHYNzbgk
#
lc3jpiGzs1ZWHD0nx0mHCMhU4usEcFbV6F10xzlwrseHhkeiununlCsNHatiDgzp
#
ZWLnP/mXKV992/Jhu0Z577DH1h+3JIYx0PceB9yzACJ8MNARHF7QpBkhtuGMGZpF
#
T+c73exupZFxItXs1Bnhe3djgE3MKKyYvxNUIbcTJoe7nhVMrw0/7lBSpVLvC4p3
#  wR700U0LdaGGQpslGtiE56SemgoP
```

```
# mongo config ends
```

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

```
jas:
```

```
  auth_enabled: true
  auth_type: 'jwt'
  signiture_key_id: 'service1-hmac'
  signiture_algorithm: 'hmac-sha256'
  max_memory: 4096M
  mapping_entity_type: /common/mappings
  datasource_entity_type: /common/datasources
```

Was this helpful?  