# 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**

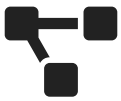
Learn about the different deployment architectures.

**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 Deployment**

Install a multi-node Autonomous Identity installation for evaluation.

**Install Multi-Node Air-Gapped**

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

**Upgrade**

Upgrade to the latest version.

**Appendix: Ports**

Learn about the Autonomous Identity ports.

**Appendix: Vars.yml**

Learn about the main deployment configuration file.

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

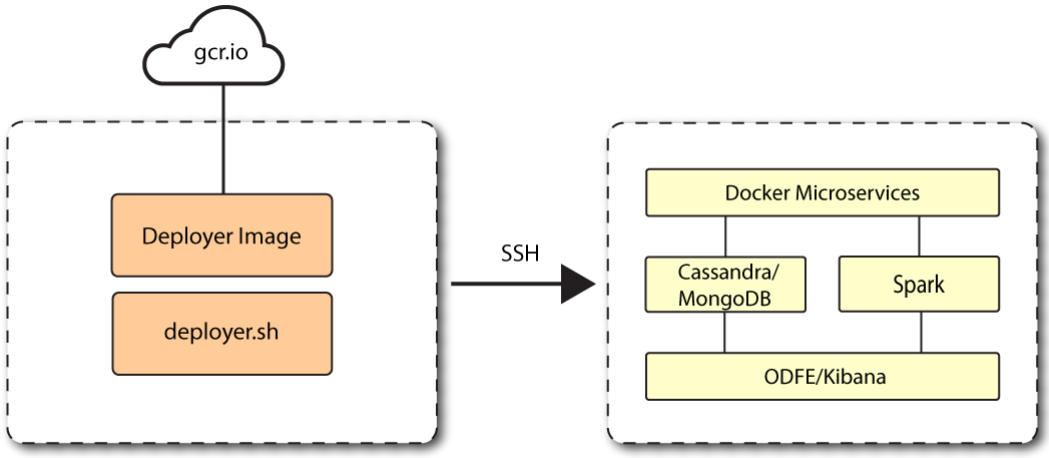
## Deployment Architectures

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

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

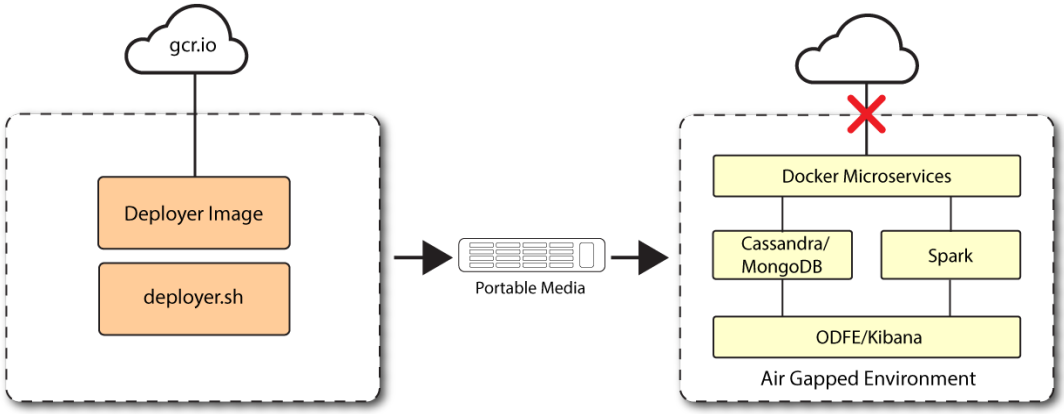
- **Single-Node Target Deployment**. Deploy Autonomous Identity on a single Internet-connected target machine. The deployer script lets you deploy the system from a local laptop or machine or from the target machine itself. The target machine can be on on-prem or on a cloud service, such as Google Cloud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

  Figure 2: A single-node target deployment.



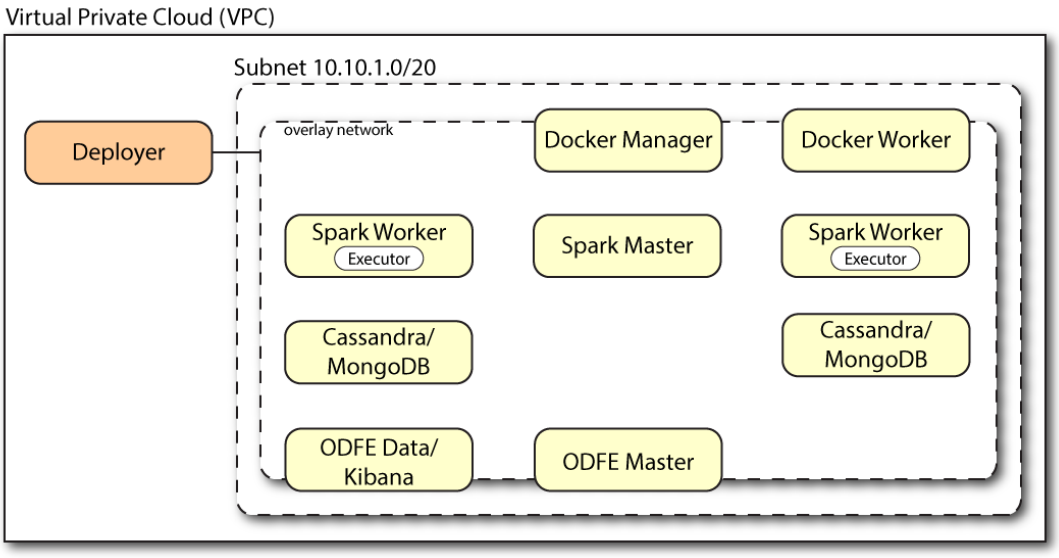

- **Single-Node Air-Gapped Target Deployment**. Deploy Autonomous Identity on a single-node target machine that resides in an air-gapped deployment. In an air-gapped deployment, the target machine is placed in an enhanced security environment where external Internet access is not available. You transfer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

  Figure 3: An air-gapped deployment.



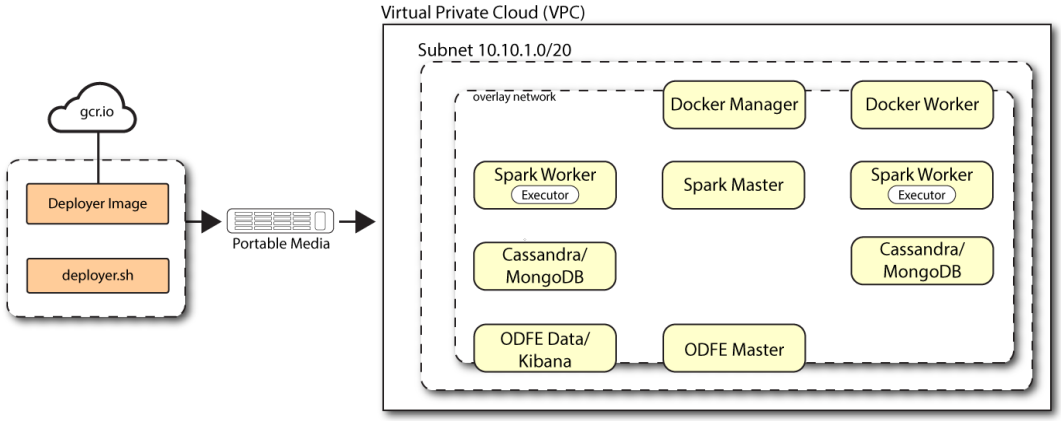

- **Multi-Node Deployment**. Deploy Autonomous Identity on multi-node deployment to distribute the process load on the servers. For installation instruction, see Install a Multi-Node Deployment.

  Figure 4: A multi-node target deployment.




- **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 Deployment.

  Figure 5: A multi-node air-gapped target deployment.

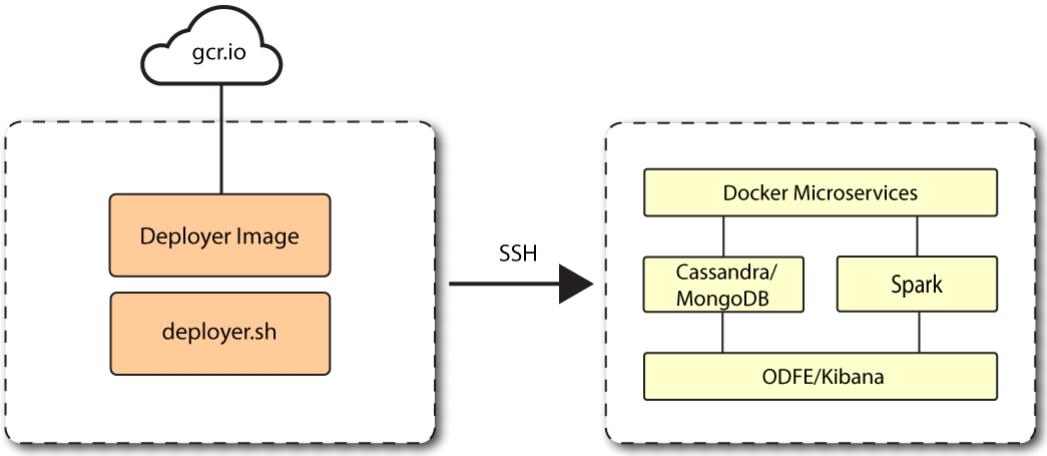



# Install a Single Node Deployment

This section presents instructions on deploying Autonomous Identity in a single-target machine that has Internet connectivity. ForgeRock provides a deployer script that pulls a Docker 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 your run the **`deployer.sh`** script from there. Also, the disk volume where you have the install directory must have at least 8GB free space for the installation.

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

## Install Docker on the Deployer Machine

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

1. On the 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:2021.8.7 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_dif: /data/conf
     ```

   - **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
```

▼ *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
```

4. Open a text editor and set the Autonomous Identity passwords for the configuration service, Cassandra or MongoDB database and their keystore/truststore, and the Elasticsearch service. 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~O>@%^()-_+=|<Y*$$rH&&/m#g{?-
o!z/1}2??3=!*&

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

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

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

## 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
--
```

NOTE

If you see a "data set too large for maximum size" error message while checking the status or starting Apache Cassandra, then you must update the segment size and timeout settings.

To update these settings:

1. Edit the `/opt/autoid/apache-cassandra-3.11.2/conf/cassandra.yaml` file and update the following fields:

```
commitlog_segment_size_in_mb: 32
write_request_timeout_in_ms: 20000
counter_write_request_timeout_in_ms: 20000
```

2. Restart Cassandra.
3. Wait a few minutes for Cassandra to boot back up.

## Check MongoDB

Check the status of MongoDB:

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

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

## Check Apache Spark

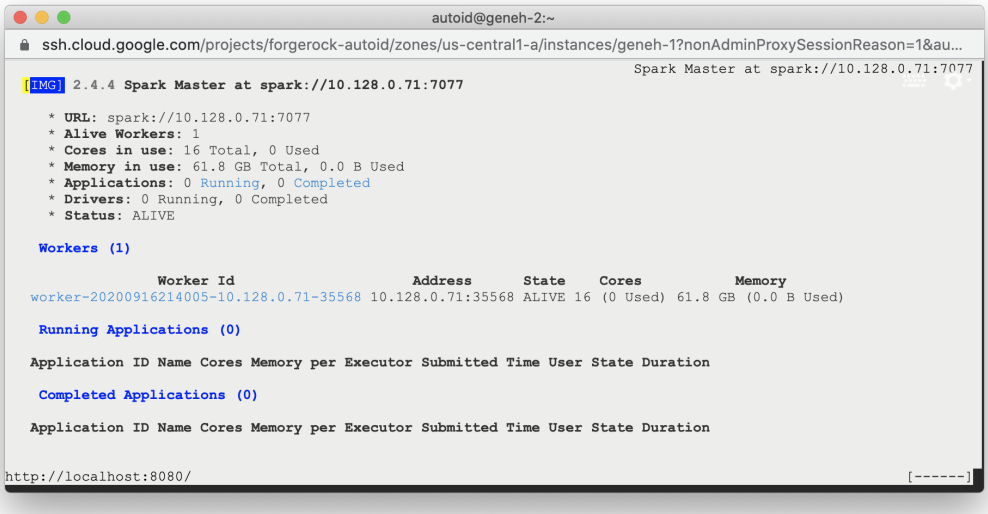
Check Spark:

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

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

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

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




## Start the Analytics

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

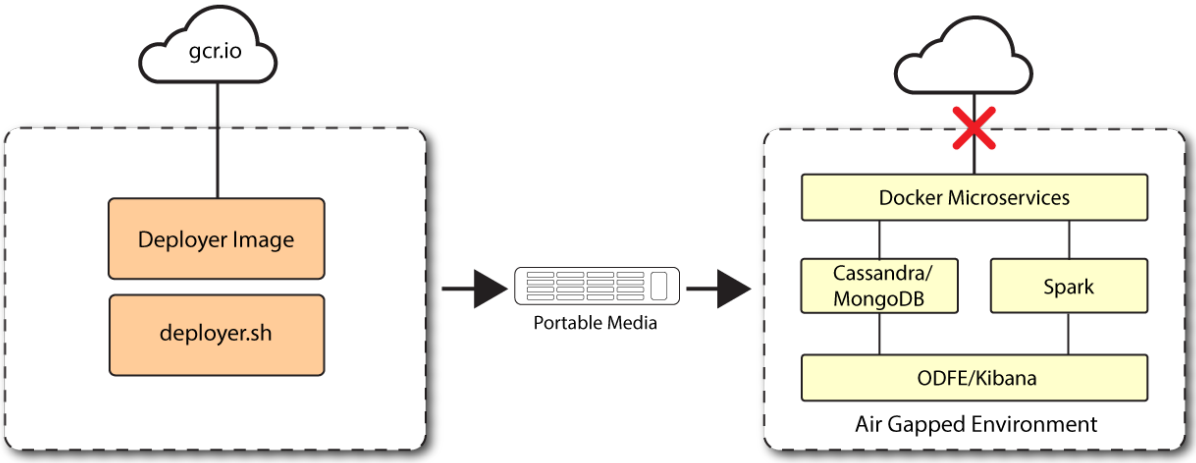
For more information, see Set Entity Definitions.

# Install a Single Node Air-Gapped Deployment

This section 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 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). 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 your run the **`deployer.sh`** script from there. Also, the

disk volume where you have the install directory must have at least 8GB free space for the installation.

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

## Install Docker on the Deployer Machine

1. On the 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:2021.8.7 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 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:2021.8.7
```

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 2021.8.7 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_dif: /data/conf
  ```

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

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

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

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

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

## 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
--
```

NOTE

If you see a "data set too large for maximum size" error message while checking the status or starting Apache Cassandra, then you must update the segment size and timeout settings.

To update these settings:

1. Edit the `/opt/autoid/apache-cassandra-3.11.2/conf/cassandra.yaml` file and update the following fields:

```
commitlog_segment_size_in_mb: 32
write_request_timeout_in_ms: 20000
counter_write_request_timeout_in_ms: 20000
```

2. Restart Cassandra.
3. Wait a few minutes for Cassandra to boot back up.

## Check MongoDB

Check the status of MongoDB:

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

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

## Check Apache Spark

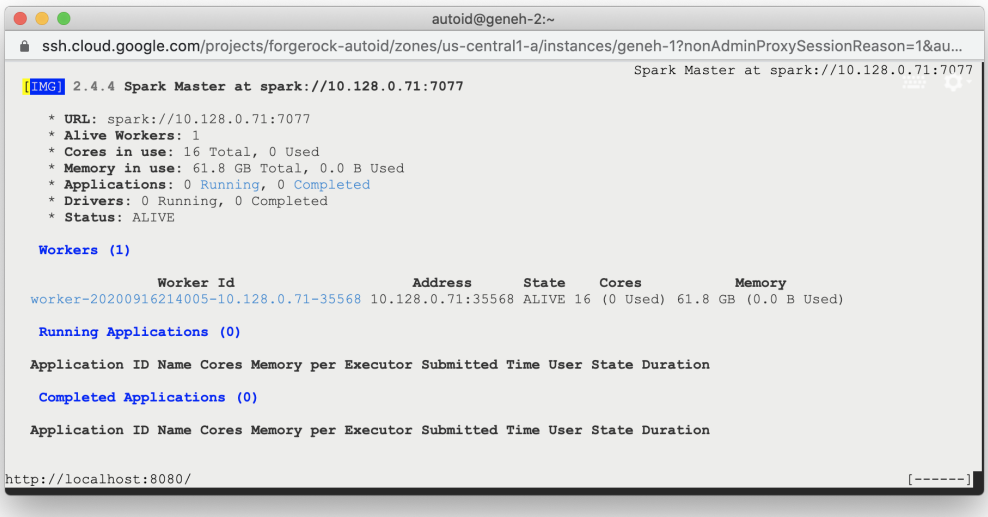
Check Spark:

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

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

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

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




## Start the Analytics

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

For more information, see Set Entity Definitions.

# Install a Multi-Node Deployment

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

Like single-node deployment, ForgeRock provides a deployer script that pulls a Docker image from ForgeRock's Google Cloud Registry (gcr.io) repository. The image contains the microservices, analytics, and backend databases needed for the system. The deployer also uses the node IP addresses specified in your `hosts` file to set up an overlay network and your nodes.

IMPORTANT

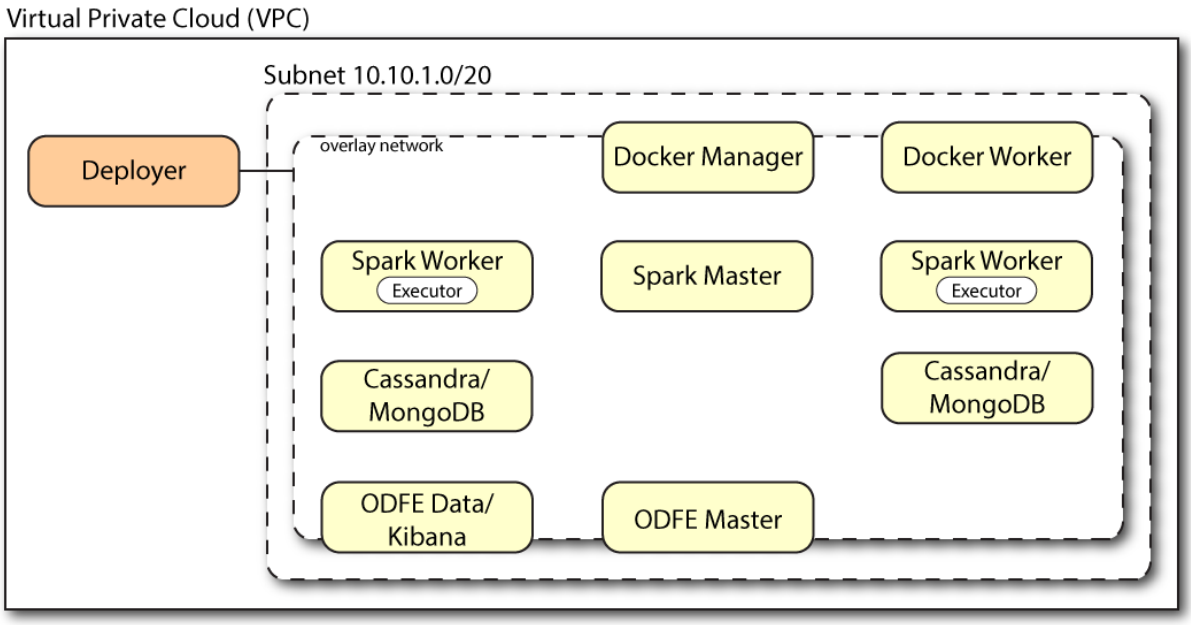
IMPORTANT

The topology presented in this section is a generalized example that is used in our automated testing. Each production deployment is unique and requires proper review prior to implementation.

NOTE

For production, the example assumes that you run the deployer on a dedicated low-spec box. After you set up your environment and back up the `autoid-config` directory, you can recycle the deployer box.

Figure 8: An example 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 multi-node machines within the same subnet. Ports must be open for the installation to succeed. Each instance should be able to communicate to the other instances.

IMPORTANT

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

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

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

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

- **Architecture Requirements**. Make sure that the Spark master is on a separate node from the Spark workers.
- **Database Requirements**. Decide which database you are using: Apache Cassandra or MongoDB. The configuration procedure is slightly different for each database.
- **Deployment Best-Practice**. The example combines the ODFE data and Kibana nodes. For best performance in production, 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
```

IMPORTANT

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

## Example Topology

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

NOTE

Each deployment is unique and should be discussed with your installer and ForgeRock.

For this example, use the following configuration for this example multi-node deployment:

***Suggested Topology***

| | Num Nodes | Cores | Memory |
|---|---|---|---|
| Deployer | 1 | 2 vCPU | 4 GB |
| Docker Manager | 1 | 8 vCPU | 32 GB |
| Docker Worker | 1 | 8 vCPU | 32 GB |
| Cassandra Seeds | 2 | 8 vCPU | 32 GB |
| Spark Master | 1 | 16 vCPU | 64 GB |
| Spark Worker | 2 | 8 vCPU | 32 GB |
| Elasticsearch (ODFE) Master/Kibana | 1 | 8 vCPU | 32 GB |
| ODFE Data | 1 | 8 vCPU | 32 GB |

## Set Up the Nodes

Set up your VMs based on the Example Topology.

1. For each VM, make sure 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 node to a username of your choice:

a. In this example, create user `autoid`.

```
$ sudo adduser autoid
```

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

```
$ sudo passwd autoid
```

c. Configure the user for passwordless sudo.

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

d. Add administrator privileges to the user.

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

3. Change to the user account.

```
$ su - autoid
```

4. 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
```

## 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. Change to the user account.

```
$ su - autoid
```

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

```
$ mkdir autoid-config
```

3. Set up the Docker-CE repository.

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

4. 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
```

5. Enable Docker to start at boot.

```
$ sudo systemctl enable docker
```

6. Start Docker.

```
$ sudo systemctl start docker
```

7. Check that Docker is running.

```
$ systemctl status docker
```

8. Add the user to the Docker group.

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

9. Logout of the user account.

```
$ logout
```

10. 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.

IMPORTANT

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

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

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

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

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

4. Change the privileges to the file.

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

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

NOTE

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

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

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

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

For example, SSH to first node:

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

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

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

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

8. Enter Exit to end your SSH session.
9. Repeat steps 5–8 again for each node.

## Set Up a Shared Directory

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

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

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

2. Create the `/data` directory.

```
$ mkdir -p /data
```

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

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

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

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

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

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

```
$ vi /etc/exports

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

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

6. Start the NFS service.

```
$ systemctl restart nfs-server
```

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

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

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

   a. Install `nfs-utils`:

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

   b. Create the NFS directory mount points:

   ```
   $ mkdir -p /data
   ```

   c. Mount the NFS shared directory:

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

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

   ```
   $ cd /data
   $ touch test
   ```

## Change Network Kernal Settings for Cassandra

The default network kernal setttings require overriding the default values to ensure TCP buffers are properly sized for use with Cassandra.

NOTE

If you are running an evaluation deployment with a small dataset, you can skip this section. For production deployments using Cassandra, run these instructions.

For each Cassandra seed node, run the following steps:

1. SSH to a Cassandra seed machine.
2. Change to the default user. In this example, `autoid`.
3. Open a text editor and edit the `/etc/sysctl.conf` file. Add the following settings to the file:

```
net.core.rmem_max=16777216
net.core.wmem_max=16777216
net.core.rmem_default=16777216
net.core.wmem_default=16777216
net.core.optmem_max=40960
net.ipv4.tcp_rmem=4096 87380 16777216
net.ipv4.tcp_wmem=4096 65536 16777216
vm.max_map_count=1048575
```

4. At runtime, make sure swap is not used. For a permanent change, modify `/etc/fstab` file.

```
$ sudo swapoff -a
```

5. Disable defrag of huge pages. The following command must be executed at each boot:

```
$ echo never | sudo tee
/sys/kernel/mm/transparent_hugepage/defrag
```

6. Set the proper ulimits for the user running Cassandra by setting them in the `/etc/security/limits.conf` file. In this example, the user running Cassandra is `autoid`.

   Open a text editor, and add the following settings in the `/etc/security/limits.conf` file:

```
autoid - memlock unlimited
autoid - nofile 100000
autoid - nproc 32768
autoid - as unlimited
```

## 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. Obtain the registry key for 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.
3. Log in to the ForgeRock Google Cloud Registry (gcr.io) using the registry key.

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

   You should see:

```
Login Succeeded
```

4. 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:2021.8.7 create-
template
```

## 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`, `vault.yml`,

and `hosts`.

1. On the deployer node, change to the `autoid-config/` directory.
2. Check the `ansible.cfg` file for the remote user and SSH private key file location. If you followed these instruction, the default settings should be in place, which will not require any edits to the `ansible.cfg` file.

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

3. 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.yaml` file to point to the correct location:

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

- **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**. You can skip this step as we use the private IP addresses in the subnet.
- **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/O/1YlRRiW57bWU3zjliW3V
dguJar5i

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

2hwRl7nqPZZMMR+uQThg/zV9rOzHJmkqZtsO4UJSilG9euLCYrzW2hd
oPuCrEDhu

Vsi5+nwAgYR9dP2oWkmGN1dwRe0ixSIM2UzFgpaXZaMOG6VztmFrlVX
h8oFDRGM0

cGrFHcnGF7oUGfWnI2Cekngk64dHA2qD7WxXPbQ/svn9EfTY5aPw5lX
zKA87Ds8p

KHVFUYvmA6wVsxb/riGLwc+XZlb6M9gqHn1XSpsnYRjF6UzfRcRR2Wy
CxLZELaqu

iKxLKB5FYqMBH7Sqg3qBCtE53vZ7T1nefq5RFzmykviYP63Uhu/A2EQ
atrMnaFPl

TTG5CaPjob45CBSyMrheYRWKqxdWN93BTgiTW7p0U6RB0/OCUbsVX6I
G3I9N8Uqt
```

```
l8Kc+7aOmtUqFkwo8w30prIOjStMrokxNsuK9KTUiPu2cj7gwYQ574v
V3hQvQPAr

hhb9ohKr0zoPQt31iTj0FDkJzPepeuzqeq8F51HB56RZKpXdRTfY8G6
OaOT68cV5

vP1O6T/okFKrl41FQ3CyYN5eRHyRTK99zTytrjoP2EbtIZ18z+bg/an
gRHYNzbgk

lc3jpiGzs1ZWHD0nxOmHCMhU4usEcFbV6FlOxzlwrsEhHkeiununlCs
NHatiDgzp

ZWLnP/mXKV992/Jhu0Z577DHlh+3JIYx0PceB9yzACJ8MNARHF7QpBk
htuGMGZpF

T+c73exupZFxItXs1Bnhe3djgE3MKKyYvxNUIbcTJoe7nhVMrwO/7lB
SpVLvC4p3
  wR700U0LDaGGQpslGtiE56SemgoP
```

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

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

- **Elasticsearch Heap Size**. The default heap size for Elasticsearch is 1GB, which is too small for production. For production deployments for large datasets, uncomment the option and enter a value.

```
#elastic_heap_size: 4g   # 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
```

4. Open a text editor and enter the private 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]
10.128.0.90

[docker-workers]
10.128.0.170

[docker:children]
docker-managers
docker-workers

[cassandra-seeds]
10.128.0.175
10.128.0.34

[spark-master]
10.128.0.180

[spark-workers]
10.128.0.176
10.128.0.177

[spark:children]
spark-master
spark-workers

[mongo_master]
#ip#  mongodb_master=True

[mongo_replicas]
#ip-1#
##ip-2#
##…
```

```
[mongo:children]
mongo_replicas
mongo_master

# ELastic Nodes
[odfe-master-node]
10.128.0.178

[odfe-data-nodes]
10.128.0.184

[kibana-node]
10.128.0.184

[notebook]
#ip#
```

5. Open a text editor and set the Autonomous Identity passwords for the configuration service, elasticsearch backend, and Cassandra or MongoDB 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~O>@%^()-_+=|<Y*$$rH&&/m#g{?-
o!z/1}2??3=!*&

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

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

```
    truststore_password: Acc#1234

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

6. 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
```

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

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

8. Run the deployment.

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

## Set the Replication Factor

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

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

1. SSH to a Cassandra seed node.
2. Change to the `/opt/autoid/apache-cassandra-3.11.2/` directory.
3. Start the Cassandra sheel, `cqlsh`, and define the `autoid` keyspace. Change the replication factor to match the number of seed nodes. The default admin user for Cassandra is `zoran_dba`.

```
$ bin/cqlsh -u zoran_dba

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

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

4. Restart Cassandra on this node.
5. Repeat these steps on the other Cassandra seed node(s).

## Resolve Hostname

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

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 `C:\Windows\System32\drivers\etc\hosts` (Windows) for the target node.

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

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

   For example:

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

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

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

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

## Access the Dashboard

Access the Autonomous Identity console UI:

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

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

2. Log in as a test user.

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

## Check Apache Cassandra

Check Cassandra:

1. 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
--
```

NOTE

If you see a "data set too large for maximum size" error message while checking the status or starting Apache Cassandra, then you must update the segment size and timeout settings.

To update these settings:

1. Edit the `/opt/autoid/apache-cassandra-3.11.2/conf/cassandra.yaml` file and update the following fields:

```
commitlog_segment_size_in_mb: 32
write_request_timeout_in_ms: 20000
counter_write_request_timeout_in_ms: 20000
```

2. Restart Cassandra.
3. Wait a few minutes for Cassandra to boot back up.

## Check MongoDB

Check the status of MongoDB:

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

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

## Check Apache Spark

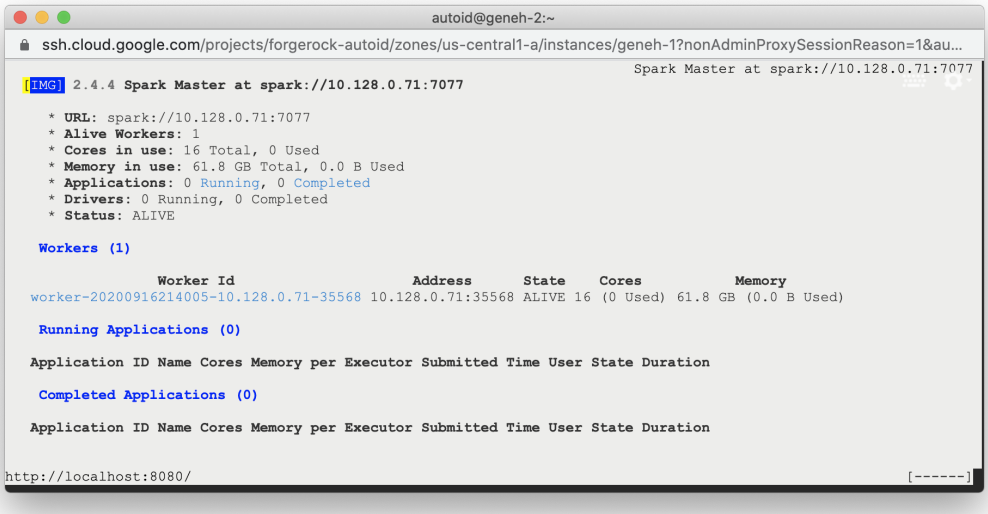
Check Spark:

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

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

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

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




## Start the Analytics

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

For more information, see Set Entity Definitions.

# Install a Multi-Node Air-Gapped Deployment

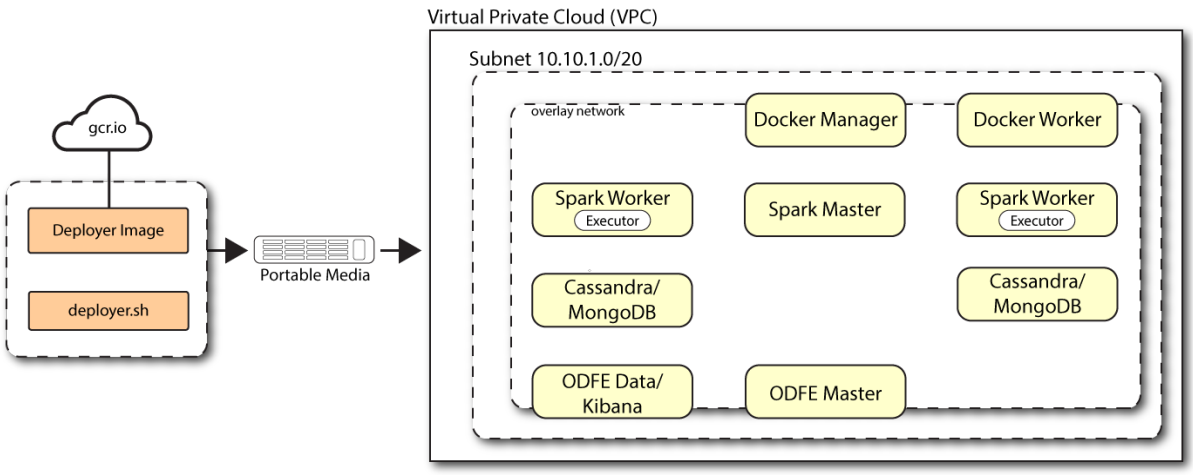
This chapter presents instructions on deploying Autonomous Identity in a multi-node air-gapped or offline target machine that has no external Internet connectivity. ForgeRock provides a deployer script that pulls a Docker image from ForgeRock's Google Cloud Registry (gcr.io) repository. The image contains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

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

The deployment example in this section uses the same multi-node deployment as seen in Example Topology.

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 multi-node machines within the same subnet. Ports must be open for the installation to succeed. Each instance should be able to communicate to the other instances.

  IMPORTANT

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

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

- **Deployment Requirements**. Autonomous Identity provides a `deployer.sh` script that downloads and installs the necessary Docker images. To download the deployment images, you must first obtain a registry key to log into the ForgeRock Google Cloud Registry (gcr.io). The registry key is only available to ForgeRock Autonomous Identity customers. For specific instructions on obtaining the registry key, see How To Configure Service Credentials (Push Auth, Docker) in Backstage.
- **Filesystem Requirements**. Autonomous Identity requires a shared filesystem accessible from the Spark master, Spark worker, analytics hosts, and application

layer. The shared filesystem should be mounted at the same mount directory on all of those hosts. If the mount directory for the shared filesystem is different from the default, `/data`, update the `/autoid-config/vars.yml` file to point to the correct directories:

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

- **Architecture Requirements**. Make sure that the Spark master is on a separate node from the Spark workers.
- **Database Requirements**. Decide which database you are using: Apache Cassandra or MongoDB. The configuration procedure is slightly different for each database.
- **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
```

IMPORTANT

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

## Set Up the Target Nodes

Set up each node as presented in Set Up the Nodes for Non-Airgap.

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

## Set Up the Deployer Machine

Set up the deployer on an Internet-connected machine.

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

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

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

```
$ sudo adduser autoid
```

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

```
$ sudo passwd autoid
```

4. Configure the user for passwordless sudo.

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

5. Add administrator privileges to the user.

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

6. Change to the user account.

```
$ su - autoid
```

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

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

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

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

## Install Docker on the Deployer Machine

Install Docker on the deployer node as presented in Install Docker for Non-Airgap.

## Set Up SSH on the Deployer

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

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

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

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

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

3. Change the privileges to the file.

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

## Prepare the Tar File

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

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

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

2. Log in to the ForgeRock Google Cloud Registry (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:2021.8.7 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 portable hard drive.

## Install on the Air-Gapped 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 he **`create-template`** command. This command creates a configuration file, `ansible.cfg`.

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

11. Make the script executable.

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

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

```
$ ./deployer.sh

Usage: deployer <command>

Commands:
  create-template
  download-images
  import-deployer
  encrypt-vault
  decrypt-vault
  run
  create-tar
```

```
install-docker
install-dbutils
upgrade
```

## Configure Autonomous Identity

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

NOTE

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

IMPORTANT

For air-gapped deployments, you must set the `offline_mode` property to `true` in the `~/autoid-config/vars.yml` file in step 2 below. This is a new change in 2021.8.7 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.yaml` file to point to the correct location:

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

- **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/O/1YlRRiW57bWU3zjliW3V
dguJar5i

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

2hwRl7nqPZZMMR+uQThg/zV9rOzHJmkqZtsO4UJSilG9euLCYrzW2hd
oPuCrEDhu

Vsi5+nwAgYR9dP2oWkmGN1dwRe0ixSIM2UzFgpaXZaMOG6VztmFrlVX
h8oFDRGM0

cGrFHcnGF7oUGfWnI2Cekngk64dHA2qD7WxXPbQ/svn9EfTY5aPw5lX
zKA87Ds8p

KHVFUYvmA6wVsxb/riGLwc+XZlb6M9gqHn1XSpsnYRjF6UzfRcRR2Wy
CxLZELaqu

iKxLKB5FYqMBH7Sqg3qBCtE53vZ7T1nefq5RFzmykviYP63Uhu/A2EQ
atrMnaFPl

TTG5CaPjob45CBSyMrheYRWKqxdWN93BTgiTW7p0U6RB0/OCUbsVX6I
G3I9N8Uqt

l8Kc+7aOmtUqFkwo8w30prIOjStMrokxNsuK9KTUiPu2cj7gwYQ574v
V3hQvQPAr
```

```
hhb9ohKr0zoPQt31iTj0FDkJzPepeuzqeq8F51HB56RZKpXdRTfY8G6
OaOT68cV5

vP1O6T/okFKrl41FQ3CyYN5eRHyRTK99zTytrjoP2EbtIZ18z+bg/an
gRHYNzbgk

lc3jpiGzs1ZWHD0nxOmHCMhU4usEcFbV6FlOxzlwrsEhHkeiununlCs
NHatiDgzp

ZWLnP/mXKV992/Jhu0Z577DHlh+3JIYx0PceB9yzACJ8MNARHF7QpBk
htuGMGZpF

T+c73exupZFxItXs1Bnhe3djgE3MKKyYvxNUIbcTJoe7nhVMrwO/7lB
SpVLvC4p3
  wR700U0LDaGGQpslGtiE56SemgoP
```

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

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

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

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

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

```
jas:
    auth_enabled: true
    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 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 multi-node deployments:

```
[docker-managers]
10.128.0.90

[docker-workers]
10.128.0.170

[docker:children]
docker-managers
docker-workers

[cassandra-seeds]
10.128.0.175
10.128.0.34

[spark-master]
10.128.0.180

[spark-workers]
10.128.0.176
10.128.0.177

[spark:children]
spark-master
spark-workers

[mongo_master]
#ip#  mongodb_master=True

[mongo_replicas]
#ip-1#
##ip-2#
##…

[mongo:children]
mongo_replicas
mongo_master

# ELastic Nodes
```

```
[odfe-master-node]
10.128.0.178

[odfe-data-nodes]
10.128.0.184

[kibana-node]
10.128.0.184

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

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

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

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

```
<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
```

## Set the Replication Factor

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

See Set the Replication Factor for Non-Airgap.

## Resolve Hostname

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

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 `C:\Windows\System32\drivers\etc\hosts` (Windows) for the target node.

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

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

For example:

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

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

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

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

### Access the Dashboard

Access the Autonomous Identity console UI:

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

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

2. Log in as a test user.

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

### Start the Analytics

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

For more information, see Set Entity Definitions.

## Upgrade Autonomous Identity

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

## Upgrade Considerations

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

IMPORTANT

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

## Upgrade Paths

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

*Table 1: Upgrade Paths*

| Version | Upgrade To | See |
|---|---|---|
| 2021.8.x (2021.8.0–2021.8.6) | 2021.8.7 | Upgrade from Autonomous Identity 2021.8.x to 2021.8.7 |
| 2021.8.x (2021.8.0–2021.8.6) Air-Gapped | 2021.8.7 Air-Gapped | Upgrade from Autonomous Identity 2021.8.x to 2021.8.7 Air-Gapped |
| 2021.3.x (2021.3.0–2021.3.5) | 2021.3.0–2021.3.4 → 2021.3.5 → 2021.8.0 → 2021.8.7 | Upgrade from Autonomous Identity 2021.3.x to 2021.8.7 |

## Upgrade from Autonomous Identity 2021.8.x to 2021.8.7

The following instructions are for upgrading from Autonomous Identity version **2021.8.x** (**2021.8.0–2021.8.6**) to the latest version **2021.8.7** in non air-gapped deployments.

### *Upgrade from 2021.8.x to 2021.8.7 Non Air-Gap:*

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

```
$ sudo mv /data/conf ~/backup-data-conf-2021.8.x
```

2. Next, if you changed any analytic settings on your deployment, make note of your configuration, so that you can replicate those settings on the upgraded server. Log in to Autonomous Identity, navigate to **Administration** > **Analytic Settings**, and record your settings.
3. On the deployer machine, back up the 2021.8.x `~/autoid-config` directory or move it to another location.

```
$ mv ~/autoid-config ~/backup-2021.8.x
```

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

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

5. Copy your `autoid_registry_key.json`, `ansible.cfg`, and `vault.yml` files from your backup directory to `~/autoid-config`. If your `vault.yml` file is encrypted, copy the `.autoid_vault_password` file to `~/autoid-config`.
6. Copy your original SSH key into the new directory.

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

7. Change the permission on the SSH key.

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

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

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

Last login: Wed Jan 15 18:19:14 2021
```

9. Stop the stack.

NOTE

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

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

You should see:

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

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

```
$ docker swarm leave
```

11. Enter **exit** to end your SSH session.
12. From the deployer, restart Docker command:

```
$ sudo systemctl restart docker
```

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

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

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

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

You should see:

```
Login Succeeded
```

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

```
$ docker run --user=$(id -u) -v ~/autoid-config:/config \
-it gcr.io/forgerock-autoid/deployer:2021.8.7 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.

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. Run the upgrade on versions 2021.8.0–2021.8.3:

```
$ ./deployer.sh debug upgrade_2020_8
```

17. SSH to the target server.
18. On the target server, restore your `/data/conf` configuration file from your previous installation.

```
$ sudo mv ~/backup-data-conf-2021.8.x /data/conf
```

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

You have successfully upgraded your Autonomous Identity server to 2021.8.7.

# Upgrade from Autonomous Identity 2021.8.x to 2021.8.7 Air-Gapped

The following instructions are for upgrading from Autonomous Identity version **2021.8.x** (**2021.8.0–2021.8.6**) to **2021.8.7** on air-gapped deployments.

### *Upgrade from 2021.8.x to 2021.8.7 Air-Gapped:*

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

```
$ sudo mv /data/conf ~/backup-data-conf-2021.8.x
```

2. Next, if you changed any analytic settings on your deployment, make note of your configuration, so that you can replicate those settings on the upgraded server. Log in to Autonomous Identity, navigate to **Administration** > **Analytic Settings**, and record your settings.
3. On the deployer machine, back up the 2021.8.x `~/autoid-config` directory or move it to another location.

```
$ mv ~/autoid-config ~/backup-2021.8.x
```

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

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

5. Copy your `autoid_registry_key.json`, `ansible.cfg`, and `vault.yml` files from your backup directory to `~/autoid-config`. If your `vault.yml` file is encrypted, copy the `.autoid_vault_password` file to `~/autoid-config`.
6. Copy your original SSH key into the new directory.

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

7. Change the permission on the SSH key.

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

8. Stop the stack.

NOTE

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

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

You should see:

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

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

```
$ docker swarm leave
```

10. From the deployer, restart Docker:

```
$ sudo systemctl restart docker
```

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

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

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

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

You should see:

```
Login Succeeded
```

13. Run the **`create-template`** command to generate the `deployer.sh` script wrapper and configuration files. Note that the command sets the configuration

directory on the target node to `/config`. The **`--user`** parameter eliminates the need to use **`sudo`** while editing the hosts file and other configuration files.

```
$ docker run --user=$(id -u) -v ~/autoid-config:/config \
-it gcr.io/forgerock-autoid/deployer:2021.8.7 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.

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

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

17. Copy the autoid-packages.tgz, deployer.sh, and SSH key (id_rsa) to a portable hard drive.
18. On the air-gapped target machine, backup your previous `~/autoid-config` directory, and then create a new `~/autoid-config` directory.

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

19. Copy the `autoid-package.tgz` tar file from the portable storage device.
20. Unpack the tar file.

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

21. Copy the SSH key to the `~/autoid-config` directory.
22. Change the privileges to the file.

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

23. Change to the configuration directory.

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

24. 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:2021.8.7
```

25. 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
```

26. 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.

27. Run the upgrade on versions 2021.8.0–2021.8.3:

```
$ ./deployer.sh debug upgrade_2020_8
```

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

```
$ sudo mv ~/backup-data-conf-2021.8.x /data/conf
```

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

You have successfully upgraded your Autonomous Identity server to 2021.8.7.

### Upgrade from Autonomous Identity 2021.3.x to 2021.8.7

The following instruction is for an upgrade from Autonomous Identity **2021.3.x** (2021.3.0–2021.3.5) to version **2021.8.7** in non air-gapped deployments.

As a reminder, upgrade from 2021.3.x to 2021.8.7 requires multiple updates to account for numerous backend component and feature changes for each major release:

- 2021.3.0–2021.3.4 → 2021.3.5
- 2021.3.5 → 2021.8.0
- 2021.8.0 → 2021.8.7

> ***Upgrade from 2021.3.x to version 2021.8.7:***
>
> 1. If you are on version 2021.3.0–2021.3.4, you must upgrade to the latest patch release to version 2021.3.5. See Upgrading Autonomous Identity to 2021.3.5.
> 2. From version 2021.3.5, upgrade to version 2021.8.0. Follow the instructions in Upgrading Autonomous Identity from 2021.3.5 to 2021.8.0.
> 3. From version 2021.8.0, upgrade to the latest patch version, 2021.8.7. Follow the instructions in Upgrade from Autonomous Identity 2021.8.x to 2021.8.7.

## 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, private IP address mapping, LDAP/SSO options, and session duration during installation. The file is created when running the **`create-template`** command during the installation and is located in the `/autoid-config` directory.

The file is as follows:

```
ai_product: auto-id                          # Product name
domain_name: forgerock.com                   # Default domain name
target_environment: autoid                   # Default namespace
analytics_data_dir: /data                    # Default data
directory
analytics_conf_dir: /data/conf               # Default config
directory for analytics


# set to true for air-gap installation
offline_mode: false

# choose the DB Type : cassandra| mongo
db_driver_type: cassandra

# Needed only if private and public IP address of
# target nodes are different. If cloud VMs the private
# is different than the IP address (public ip) used for
# SSH. Private IP addresses are used by various services
# to reach other services in the cluster
# Example:
# private_ip_address_mapping:
#   35.223.33.21: "10.128.0.5"
#   108.59.83.132: "10.128.0.37"
#   ...
private_ip_address_mapping:                          # private and
external IP mapping
#private_ip_address_mapping-ip-addesses#

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/O/1YlRRiW57bWU3zjliW3VdguJar5i9
#
Z+1a8lI+0S9pWynbv9+Ao0aXFjSJYVxAm/w7DJbVRGcPhsPmExiSBDw8szfQ8PAU
#
2hwRl7nqPZZMMR+uQThg/zV9rOzHJmkqZtsO4UJSilG9euLCYrzW2hdoPuCrEDhu
#
Vsi5+nwAgYR9dP2oWkmGN1dwRe0ixSIM2UzFgpaXZaMOG6VztmFrlVXh8oFDRGM0
#
cGrFHcnGF7oUGfWnI2Cekngk64dHA2qD7WxXPbQ/svn9EfTY5aPw5lXzKA87Ds8p
#
KHVFUYvmA6wVsxb/riGLwc+XZlb6M9gqHn1XSpsnYRjF6UzfRcRR2WyCxLZELaqu
#
iKxLKB5FYqMBH7Sqg3qBCtE53vZ7T1nefq5RFzmykviYP63Uhu/A2EQatrMnaFPl
#
TTG5CaPjob45CBSyMrheYRWKqxdWN93BTgiTW7p0U6RB0/OCUbsVX6IG3I9N8Uqt
#
l8Kc+7aOmtUqFkwo8w30prIOjStMrokxNsuK9KTUiPu2cj7gwYQ574vV3hQvQPAr
#
hhb9ohKr0zoPQt31iTj0FDkJzPepeuzqeq8F51HB56RZKpXdRTfY8G6OaOT68cV5
#
vP1O6T/okFKrl41FQ3CyYN5eRHyRTK99zTytrjoP2EbtIZ18z+bg/angRHYNzbgk
#
lc3jpiGzs1ZWHD0nxOmHCMhU4usEcFbV6FlOxzlwrsEhHkeiununlCsNHatiDgzp
#
ZWLnP/mXKV992/Jhu0Z577DHlh+3JIYx0PceB9yzACJ8MNARHF7QpBkhtuGMGZpF
#
T+c73exupZFxItXs1Bnhe3djgE3MKKyYvxNUIbcTJoe7nhVMrwO/7lBSpVLvC4p3
#  wR700U0LDaGGQpslGtiE56SemgoP

# mongo config ends

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

jas:
  auth_enabled: true
  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?