

Installation

This chapter shows you how to install and deploy Autonomous Identity for intelligent entitlements management in production environments. For hardware and software requirements, refer to 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, refer to 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 and uses it to deploy the microservices and analytics for Autonomous Identity on a target machine. The target machine only requires the base operating system.

IMPORTANT

If you are upgrading Autonomous Identity on a RHEL 7/CentOS 7, the upgrade to 2022.11 uses RHEL 7/CentOS 7 only. For new and clean installations, Autonomous Identity requires RHEL 8 or CentOS Stream 8 only.

Autonomous Identity 2022.11.0 introduced a new deployer, Deployer Pro, that pulls in the base code from the ForgeRock Google Cloud repository. Customers must now pre-install the third-party software dependencies prior to running deployer pro. For more information, refer to [Install a Single Node Deployment](#).

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, refer to [Install a Single-Node Deployment](#).

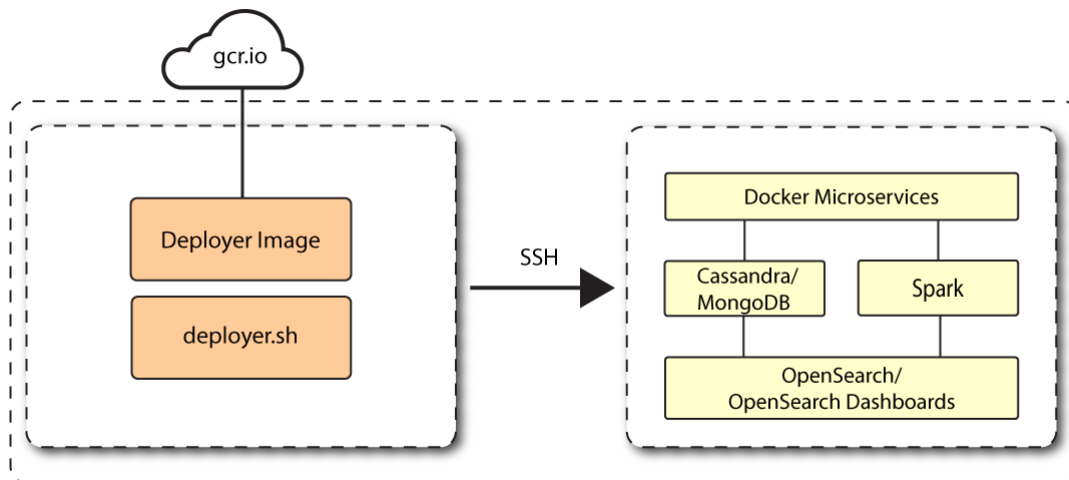


Figure 1. A single-node target deployment.

- **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, refer to [Install a Single-Node Air-Gapped](#).

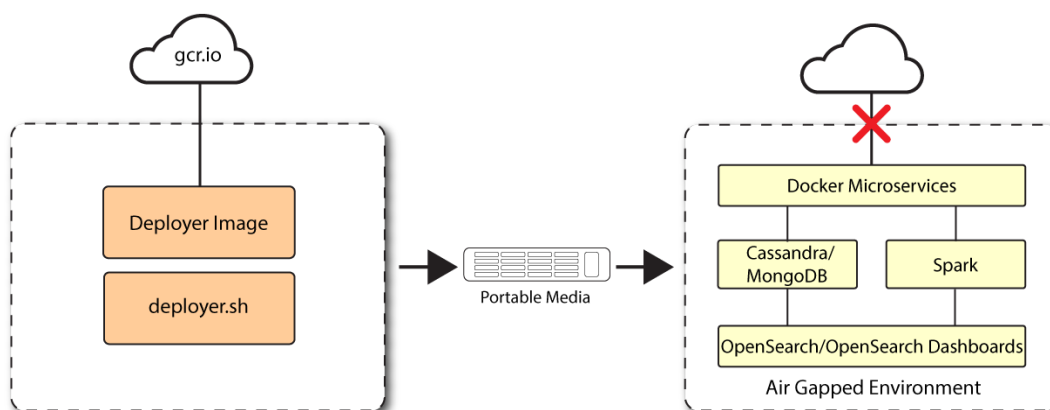


Figure 2. An air-gapped deployment.

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

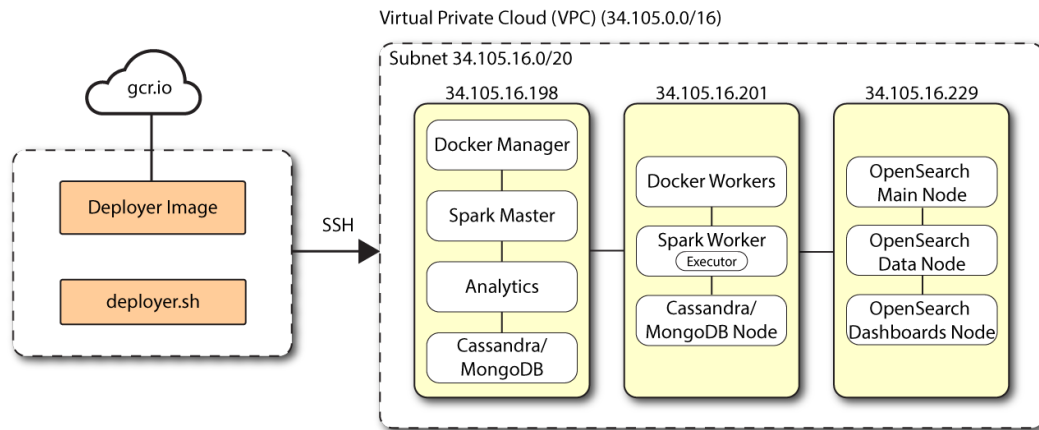


Figure 3. A multi-node target deployment.

- **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, refer to [Install a Multi-Node Air-Gapped Deployment](#).

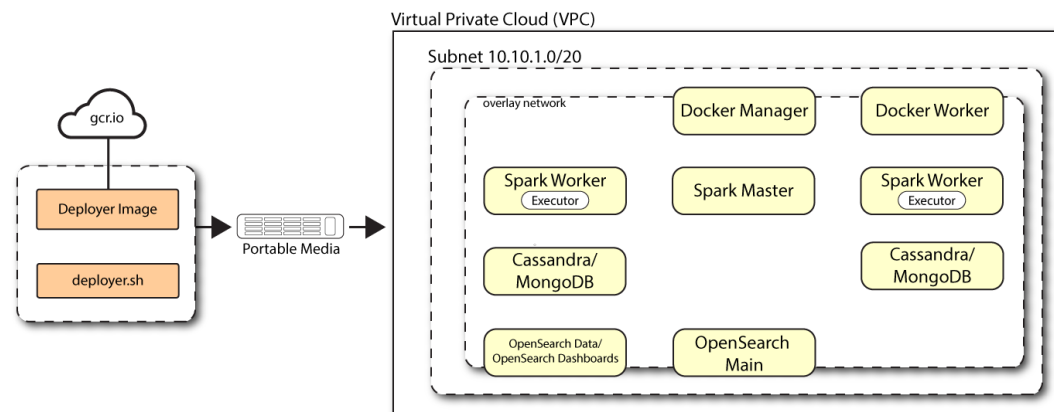


Figure 4. A multi-node air-gapped target deployment.

Install a Single Node Deployment

This section presents instructions on deploying Autonomous Identity in a single-target machine with Internet connectivity.

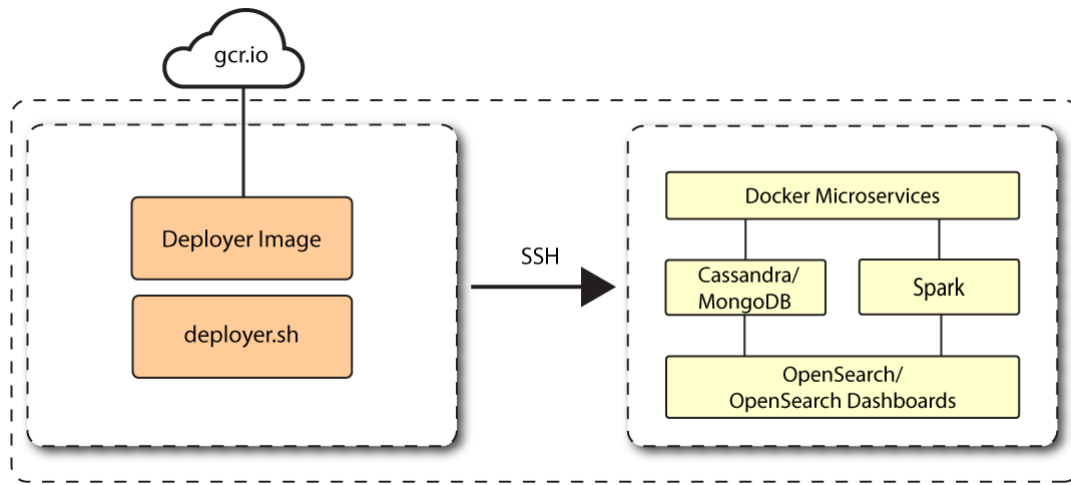


Figure 5. A single-node target deployment.

Autonomous Identity 2022.11.0 introduced a new installation script, *deployer pro* (**Deployer** for **Production**), letting customers manage their third-party software dependencies in their particular Autonomous Identity environments. Autonomous Identity 2022.11.2 continues to use this deployer script. For background information about the deployer, refer to [About the new deployer pro script](#).

NOTE

The procedures presented in this section are generalized examples to help you get acquainted with Autonomous Identity. Consult with your ForgeRock Professional Services or technical partner for specific assistance to install Autonomous Identity within your particular environment.

Summary of the installation steps

To set up the 2022.11.2 deployment, run the following steps:

- Prerequisites
- Install third-party components
- Set Up SSH on the Deployer
- Install Autonomous Identity

Prerequisites

For new and clean deployments, the following are prerequisites:

- **Operating System.** The target machine requires Red Hat Linux 8/CentOS Stream 8. The deployer machine can use any operating system as long as Docker is installed. For this chapter, we use CentOS Stream 8 as its base operating system.

IMPORTANT

If you are upgrading Autonomous Identity on a RHEL 7/CentOS 7, the upgrade to 2022.11 uses RHEL 7/CentOS 7 only. For new and clean installations, Autonomous Identity requires RHEL 8 or CentOS Stream 8 only.

- **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. The registry key is only available to ForgeRock Autonomous Identity customers. For specific instructions on obtaining the registry key, refer to [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
```

Install third-party components

First, set up your GCP virtual machine and install the third-party package dependencies required for the Autonomous Identity deployment:

Install third-party packages:

1. Create a GCP Red Hat Enterprise Linux (RHEL) 8 or CentOS Stream 8 virtual machine: `n2-standard-4` (4 vCPU and 16GB memory). Refer to <https://www.centos.org/centos-stream/>.
2. Create an `autoid` user with the proper privileges to run the installation. For example:

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

```
sudo usermod -aG wheel autoid
su - autoid
```

3. Install the following packages needed in the Autonomous Identity deployment:

- **Java 11.** For example, `sudo dnf install java-11-openjdk-devel`.
- **wget.** For example, `sudo dnf install wget`.
- **unzip.** For example, `sudo dnf install unzip`.
- **elinks.** For example, `sudo yum install -y elinks`.

4. Install Python 3.8.13.

- Refer to <https://docs.python.org/release/3.8.13/>.
- Make sure no other Python versions are installed on the machine. Remove those versions. For example:

```
sudo rm -rf /usr/bin/python3
sudo rm -rf /usr/bin/python3.6
sudo rm -rf /usr/bin/python3m
sudo rm -rf /usr/bin/pip3
sudo rm -rf /usr/bin/easy_install-3
sudo rm -rf /usr/bin/easy_install-3.6
```

- Create symlinks for python3:

```
sudo ln -s /usr/bin/python 3.8 /usr/bin/python3
sudo ln -s /usr/bin/easy_install-3.8
/usr/bin/easy_install-3
sudo ln -s /usr/bin/pip3.8 /usr/bin/pip3
```

5. Install Cassandra 4.0.6. Refer to

https://cassandra.apache.org/doc/latest/cassandra/getting_started/index.html
[link]. (For MongoDB installations, follow the instructions in Download MongoDB.)

- Log in to the Cassandra shell. For example:

```
cassandra/bin/cqlsh <$ipaddress> -u cassandra -p
cassandra
```


- Create the Cassandra roles for Autonomous Identity. Refer to <https://cassandra.apache.org/doc/latest/cassandra/cql/security.html>
[link]. For example:


```
cassandra/bin/cqlsh <$ipaddress> -u cassandra -p
cassandra -e "CREATE ROLE zoran_dba WITH PASSWORD =
'password' AND SUPERUSER = true AND LOGIN = true;"
```


```

cassandra/bin/cqlsh <$ipaddress> -u cassandra -p
cassandra -e "CREATE ROLE zoranuser WITH PASSWORD =
'password' AND LOGIN = true;"
cassandra/bin/cqlsh <$ipaddress> -u zoran_dba -p
'password -e "ALTER ROLE cassandra WITH
PASSWORD='randompassword123' AND SUPERUSER=false AND
LOGIN = false;"
cassandra/bin/cqlsh <$ipaddress> -u zoran_dba -p
'password -e "ALTER KEYSPACE "system_auth" WITH
REPLICATION = {'class'
:'NetworkTopologyStrategy', 'datacenter1' : 1};"

```

- c. Configure security for Cassandra. Refer to <https://cassandra.apache.org/doc/latest/cassandra/operating/security.htm> .

6. Install MongoDB 4.4. Follow the instructions in <https://www.mongodb.com/docs/v4.4/tutorial/install-mongodb-on-red-hat/> .


- a. Create a MongoDB user with username `mongoadmin` with admin privileges. Follow the instructions in <https://www.mongodb.com/docs/v4.4/core/security-users/> .

For example:

```

db.createUser({ user: "mongoadmin",pwd: "~@C~O>@%^()-
_+=|<Y*$\$rH&&/m#g{?-o!z/1}2??3=!*&", roles: [ { role:
"userAdminAnyDatabase", db: "admin" },
"readWriteAnyDatabase" ]})

```

- b. Set up SSL, refer to <https://www.mongodb.com/docs/v4.4/tutorial/configure-ssl/#procedures—using-net.ssl-settings> . For example, the MongoDB configuration file (`/etc/mongod.conf`) would include a section similar to the following:

```

net:
  tls:
    mode: requireTLS
    certificateKeyFile: /etc/ssl/mongodb.pem
    CAFile: /etc/ssl/rootCA.pem

```

IMPORTANT

Make sure that the CN entry in the `mongodb.pem` certificate is the IP address/hostname of the `mongodb` instance. You need to add this same CN value to the `hosts` file during the Autonomous Identity deployment.

c. Restart the daemon and MongoDB.

```
sudo systemctl daemon-reload
sudo systemctl restart mongod
```


7. Install Apache Spark 3.3.2. Refer to <https://spark.apache.org/downloads.html> .

NOTE

The official release of Apache Livy does not support Apache Spark 3.3.1 or 3.3.2. ForgeRock has re-compiled and packaged Apache Livy to work with Apache Spark 3.3.1 hadoop 3 and Apache Spark 3.3.2 hadoop 3. Use the zip file located at [autoid-config/apache-livy/apache-livy-0.8.0-incubating-SNAPSHOT-bin.zip](#) to install Apache Livy on the Spark-Livy machine.

a. Configure the `SPARK_HOME` in your `bashrc` file. For example:

```
SPARK_HOME=/opt/spark/spark-3.3.2-bin-hadoop3
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
```

b. Configure authentication on Spark, refer to <https://spark.apache.org/docs/latest/security.html#authentication> . For example:

```
spark.authenticate          true
spark.authenticate.secret  <your-secret>
```

c. Enable and start the Spark main and secondary servers:

```
sudo chown -R $USER:USER $SPARK_HOME
```

d. Spark 3.3.1 and 3.3.2 no longer uses `log4j1` and has upgraded to `log4j2`. Copy or move the `log4j` template file to the `log4j2.properties` file. For example:

```
mv /opt/spark/spark-3.3.2-bin-
hadoop3/conf/log4j.properties.template
/opt/spark/spark-3.3.2-bin-
hadoop3/conf/log4j2.properties
```

NOTE

You will install Apache Livy in a later step. Refer to [Install Apache Livy](#).

8. Install OpenSearch 1.3.6 and OpenSearch Dashboards 1.3.6. Refer to <https://opensearch.org/versions/opensearch-1-3-6.html>.
 - a. Configure OpenSearch Dashboards using the `/opensearch-dashboards/config/opensearch_dashboards.yml` file. Refer to <https://opensearch.org/docs/1.3/dashboards/install/index/>.
 - b. Configure TLS/SSL security:
 - Follow the instructions in <https://opensearch.org/docs/latest/security-plugin/configuration/tls/>.
 - Follow the instructions in <https://opensearch.org/docs/2.0/security-plugin/configuration/generate-certificates/>.

IMPORTANT

Make sure that the CN entry in the `esnode.pem` certificate is the IP address/hostname of the OpenSearch instance. You need to add this same CN value to the `hosts` file during the Autonomous Identity deployment.

9. Set up Docker using the procedures in <https://docs.docker.com/engine/install/centos/>.
 - For post-installation Docker steps, follow the instructions in <https://docs.docker.com/engine/install/linux-postinstall/>.

IMPORTANT

Do not use `/opt/autoid` as Docker root as the directory is overwritten during the Autonomous Identity installation and will result in a recursive error.

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

Make sure you have the following prerequisites:

- IP address of machines running OpenSearch, MongoDB, or Cassandra.
- The Autonomous Identity user should have permission to write to `/opt/autoid` on all machines
- To download the deployment images for the install, you still need your registry key to log into the ForgeRock Google Cloud Registry to download the artifacts.
- Make sure you have the proper OpenSearch certificates with the exact names for both pem and JKS files copied to `~/autoid-config/certs/elastic`:
 - `esnode.pem`
 - `esnode-key.pem`
 - `root-ca.pem`
 - `elastic-client-keystore.jks`
 - `elastic-server-truststore.jks`
- Make sure you have the proper MongoDB certificates with exact names for both pem and JKS files copied to `~/autoid-config/certs/mongo`:
 - `mongo-client-keystore.jks`
 - `mongo-server-truststore.jks`
 - `mongodb.pem`
 - `rootCA.pem`
- Make sure you have the proper Cassandra certificates with exact names for both pem and JKS files copied to `~/autoid-config/certs/cassandra`:
 - `Zoran-cassandra-client-cer.pem`
 - `Zoran-cassandra-client-keystore.jks`
 - `Zoran-cassandra-server-cer.pem`
 - `zoran-cassandra-server-keystore.jks`
 - `Zoran-cassandra-client-key.pem`
 - `Zoran-cassandra-client-truststore.jks`
 - `Zoran-cassandra-server-key.pem`
 - `Zoran-cassandra-server-truststore.jks`


Install Autonomous Identity:


1. Create the `autoid-config` directory.

```
mkdir autoid-config
```

2. Change to the directory.

```
cd autoid-config
```

3. Log in to the ForgeRock Google Cloud Registry using the registry key. The registry key is only available to ForgeRock Autonomous Identity customers. For specific instructions on obtaining the registry key, refer to [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  

```

The following output is displayed:

```
Login Succeeded
```

4. 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-pro:2022.11.2 create-  
template
```

5. Create a certificate directory for elastic.

```
mkdir -p autoid-config/certs/elastic
```

6. Copy the OpenSearch certificates and JKS files to `autoid-config/certs/elastic`.

7. Create a certificate directory for MongoDB.

```
mkdir -p autoid-config/certs/mongo
```

8. Copy the MongoDB certificates and JKS files to `autoid-config/certs/mongo`.

9. Create a certificate directory for Cassandra.

```
mkdir -p autoid-config/certs/cassandra
```

10. Copy the Cassandra certificates and JKS files to `autoid-config/certs/cassandra`.
11. Update the `hosts` file with the IP addresses of the machines. The `hosts` file must include the IP addresses for Docker nodes, Spark main/livy, and the MongoDB master. While the `deployer pro` does not install or configure the MongoDB main server, the entry is required to run the MongoDB CLI to seed the Autonomous Identity schema.

```
[docker-managers]

[docker-workers]

[docker:children]
docker-managers
docker-workers

[spark-master-livy]

[cassandra-seeds]
#For replica sets, add the IPs of all Cassandra nodes

[mongo_master]
# Add the MongoDB main node in the cluster deployment
# For example: 10.142.15.248 mongodb_master=True

[odfe-master-node]
# Add only the main node in the cluster deployment

[kibana-node]
# Please add only the master node in cluster deployment
```

12. Update the `vars.yml` file:
 - a. Set `db_driver_type` to `mongo` or `cassandra`.
 - b. Set `elastic_host`, `elastic_port`, and `elastic_user` properties.
 - c. Set `kibana_host`.
 - d. Set the Apache livy install directory.
 - e. Ensure the `elastic_user`, `elastic_port`, and `mongo_part` are correctly configured.
 - f. Update the `vault.yml` passwords for elastic and mongo to reflect your installation.
 - g. Set the Cassandra-related parameters in the `vars.yml` file. Default values are:

```
cassandra:
  enable_ssl: "true"
  contact_points: 10.142.15.248 # comma separated
values in case of replication set
  port: 9042
  username: zoran_dba
  cassandra_keystore_password: "Acc#1234"
  cassandra_truststore_password: "Acc#1234"
  ssl_client_key_file: "zoran-cassandra-client-key.pem"
  ssl_client_cert_file: "zoran-cassandra-client-
cer.pem"
  ssl_ca_file: "zoran-cassandra-server-cer.pem"
  server_truststore_jks: "zoran-cassandra-server-
truststore.jks"
  client_truststore_jks: "zoran-cassandra-client-
truststore.jks"
  client_keystore_jks: "zoran-cassandra-client-
keystore.jks"
```

13. Download images:

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

14. Install Apache Livy.

- The official release of Apache Livy does not support Apache Spark 3.3.1 or 3.3.2. ForgeRock has re-compiled and packaged Apache Livy to work with Apache Spark 3.3.1 hadoop 3 and Apache Spark 3.3.2 hadoop 3. Use the zip file located at `autoid-config/apache-livy/apache-livy-0.8.0-incubating-SNAPSHOT-bin.zip` to install Apache Livy on the Spark-Livy machine.
- For Livy configuration, refer to <https://livy.apache.org/get-started/>.

15. On the Spark-Livy machine, run the following commands to install the python package dependencies:

- a. Change to the `/opt/autoid` directory:

```
cd /opt/autoid
```

- b. Create a `requirements.txt` file with the following content:

```
six==1.11
certifi==2019.11.28
python-dateutil==2.8.1
```

```
jsonschema==3.2.0
cassandra-driver
numpy==1.19.5
pyarrow==0.16.0
wrapt==1.11.0
PyYAML==5.4
requests
pymongo
pandas==1.0.5
tabulate
openpyxl
```

c. Install the requirements file:

```
pip3 install -r requirements.txt
```

16. Make sure that the `/opt/autoid` directory exists and that it is both readable and writable.

17. Run the deployer script:

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

18. On the Spark-Livy machine, run the following commands to install the Python egg file:

a. Install the egg file:

```
sudo /usr/local/bin/pip3.8 install setuptools==46.0.0
cd /opt/autoid/eggs
sudo /usr/local/bin/easy_install-3.8 autoid_analytics-
2021.3-py3.6.egg
```

b. Source the `.bashrc` file:

```
source ~/.bashrc
```

c. Restart Spark and Livy.

```
./spark/sbin/stop-all.sh
./livy/bin/livy-server stop

./spark/sbin/start-all.sh
./livy/bin/livy-server start
```


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. Make sure Cassandra is running in cluster mode. For example

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

Check MongoDB

Check MongoDB:

1. Make sure MongoDB is running. For example:

```
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
```

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

▼ [Click to See an Example of the Spark Dashboard](#)

```

autoid@geneh-2:~
ssh.cloud.google.com/projects/forgerock-autoid/zones/us-central1-a/instances/geneh-1?nonAdminProxySessionReason=1&au...
Spark Master at spark://10.128.0.71:7077

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

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

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

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

http://localhost:8080/

```

Start the Analytics

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

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

Install a 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.

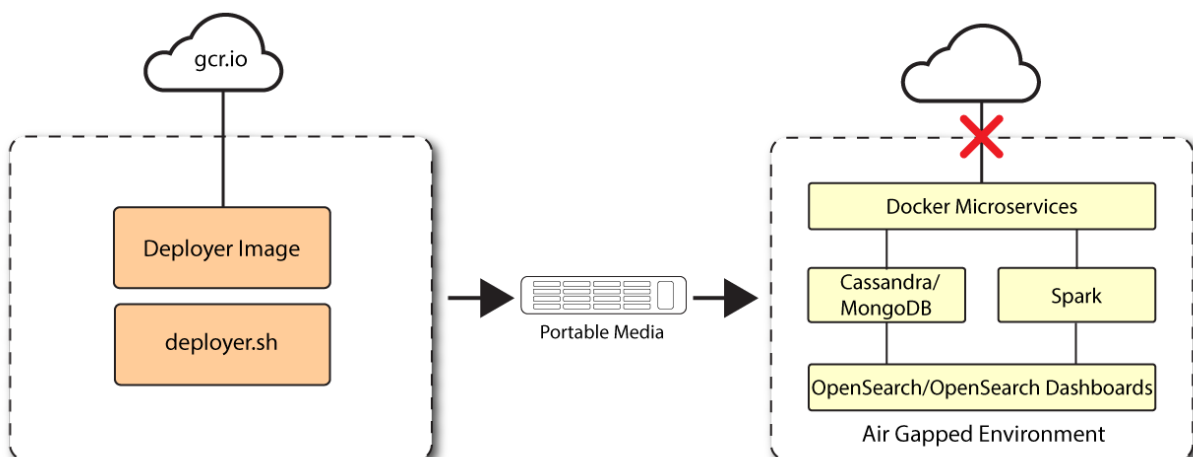


Figure 6. A single-node air-gapped target deployment.

Installation Steps for an airgap deployment

The general procedure for an air-gap deployment is practically identical to that of a single node non-airgapped, except that you must prepare a tar file and copy the files to an air-gap machine.

- Set up the Nodes
- Set up the third-party software dependencies
- Set Up SSH on the Deployer
- Prepare the Tar File
- Install Autonomous Identity

Set up the Nodes

Set up each node as presented in [Install a Single Node Deployment](#).

Make sure you have sufficient storage for your particular deployment. For more information on sizing considerations, refer to [Deployment Planning Guide](#).

Set up the third-party software dependencies

Download and unpack the third-party software dependencies in [Install third-party components](#).

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 using the registry key. The registry key is only available to ForgeRock Autonomous Identity customers. For specific instructions on obtaining the registry key, refer to [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
```

The following output is displayed:

```
Login Succeeded
```

3. Run the **create-template** command to generate the `deployer.sh` script wrapper. 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-pro:2022.11.2 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 Stream 8 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. Import the deployer image.

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

The following output is displayed:

```
...
db631c8b06ee: Loading layer
[=====→]
2.56kB/2.56kB
2d62082e3327: Loading layer
[=====→]
753.2kB/753.2kB
Loaded image: gcr.io/forgerock-autoid/deployer:2022.11.2
```

8. 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
```

The following output is displayed:

```
Config template is copied to host machine directory mapped
to /config
```

Install Autonomous Identity

Make sure you have the following prerequisites:

- IP address of machines running OpenSearch, MongoDB, or Cassandra.
- The Autonomous Identity user should have permission to write to `/opt/autoid` on all machines
- To download the deployment images for the install, you still need your registry key to log into the ForgeRock Google Cloud Registry to download the artifacts.
- Make sure you have the proper OpenSearch certificates with the exact names for both pem and JKS files copied to `~/autoid-config/certs/elastic` :
 - `esnode.pem`
 - `esnode-key.pem`
 - `root-ca.pem`
 - `elastic-client-keystore.jks`
 - `elastic-server-truststore.jks`
- Make sure you have the proper MongoDB certificates with exact names for both pem and JKS files copied to `~/autoid-config/certs/mongo` :
 - `mongo-client-keystore.jks`

- mongo-server-truststore.jks
- mongodb.pem
- rootCA.pem
- Make sure you have the proper Cassandra certificates with exact names for both pem and JKS files copied to ~/autoid-config/certs/cassandra:
 - Zoran-cassandra-client-cer.pem
 - Zoran-cassandra-client-keystore.jks
 - Zoran-cassandra-server-cer.pem
 - zoran-cassandra-server-keystore.jks
 - Zoran-cassandra-client-key.pem
 - Zoran-cassandra-client-truststore.jks
 - Zoran-cassandra-server-key.pem
 - Zoran-cassandra-server-truststore.jks

Install Autonomous Identity:

1. Create a certificate directory for elastic.

```
mkdir -p autoid-config/certs/elastic
```

2. Copy the OpenSearch certificates and JKS files to autoid-config/certs/elastic .

3. Create a certificate directory for MongoDB.

```
mkdir -p autoid-config/certs/mongo
```

4. Copy the MongoDB certificates and JKS files to autoid-config/certs/mongo .

5. Create a certificate directory for Cassandra.

```
mkdir -p autoid-config/certs/cassandra
```

6. Copy the Cassandra certificates and JKS files to autoid-config/certs/cassandra .

7. Update the hosts file with the IP addresses of the machines. The hosts file must include the IP addresses for Docker nodes, Spark main/livy, and the MongoDB master. While the deployer pro does not install or configure the MongoDB main server, the entry is required to run the MongoDB CLI to seed the Autonomous Identity schema.


```

[docker-managers]

[docker-workers]

[docker:children]
docker-managers
docker-workers

[spark-master-livy]

[cassandra-seeds]
#For replica sets, add the IPs of all Cassandra nodes

[mongo_master]
# Add the MongoDB main node in the cluster deployment
# For example: 10.142.15.248 mongodb_master=True

[odfe-master-node]
# Add only the main node in the cluster deployment

```

8. Update the `vars.yml` file:

- a. Set `offline_mode` to `true`.
- b. Set `db_driver_type` to `mongo` or `cassandra`.
- c. Set `elastic_host`, `elastic_port`, and `elastic_user` properties.
- d. Set `kibana_host`.
- e. Set the Apache livy install directory.
- f. Ensure the `elastic_user`, `elastic_port`, and `mongo_part` are correctly configured.
- g. Update the `vault.yml` passwords for elastic and mongo to reflect your installation.
- h. Set the Cassandra-related parameters in the `vars.yml` file. Default values are:

```

cassandra:
  enable_ssl: "true"
  contact_points: 10.142.15.248 # comma separated
values in case of replication set
  port: 9042
  username: zoran_dba
  cassandra_keystore_password: "Acc#1234"
  cassandra_truststore_password: "Acc#1234"
  ssl_client_key_file: "zoran-cassandra-client-key.pem"

```

```
ssl_client_cert_file: "zoran-cassandra-client-
cer.pem"
ssl_ca_file: "zoran-cassandra-server-cer.pem"
server_truststore_jks: "zoran-cassandra-server-
truststore.jks"
client_truststore_jks: "zoran-cassandra-client-
truststore.jks"
client_keystore_jks: "zoran-cassandra-client-
keystore.jks"
```

9. Install Apache Livy.

- The official release of Apache Livy does not support Apache Spark 3.3.1 or 3.3.2. ForgeRock has re-compiled and packaged Apache Livy to work with Apache Spark 3.3.1 hadoop 3 and Apache Spark 3.3.2 hadoop 3. Use the zip file located at [autoid-config/apache-livy/apache-livy-0.8.0-incubating-SNAPSHOT-bin.zip](#) to install Apache Livy on the Spark-Livy machine.
- For Livy configuration, refer to <https://livy.apache.org/get-started/>.

10. On the Spark-Livy machine, run the following commands to install the python package dependencies:

- a. Change to the /opt/autoid directory:

```
cd /opt/autoid
```

- b. Create a requirements.txt file with the following content:

```
six==1.11
certifi==2019.11.28
python-dateutil==2.8.1
jsonschema==3.2.0
cassandra-driver
numpy==1.19.5
pyarrow==0.16.0
wrapt==1.11.0
PyYAML==5.4
requests
pymongo
pandas==1.0.5
tabulate
openpyxl
```

- c. Install the requirements file:

```
pip3 install -r requirements.txt
```

11. Make sure that the `/opt/autoid` directory exists and that it is both readable and writable.
12. Run the deployer script:

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

13. On the Spark-Livy machine, run the following commands to install the Python egg file:

- a. Install the egg file:

```
sudo /usr/local/bin/pip3.8 install setuptools==46.0.0  
cd /opt/autoid/eggs  
sudo /usr/local/bin/easy_install-3.8 autoid_analytics-  
2021.3-py3.6.egg
```

- b. Source the `.bashrc` file:

```
source ~/.bashrc
```

- c. Restart Spark and Livy.

```
./spark/sbin/stop-all.sh  
./livy/bin/livy-server stop  
  
./spark/sbin/start-all.sh  
./livy/bin/livy-server start
```

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. Make sure Cassandra is running in cluster mode. For example

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

Check MongoDB

Check MongoDB:

1. Make sure MongoDB is running. For example:

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

Check Apache Spark

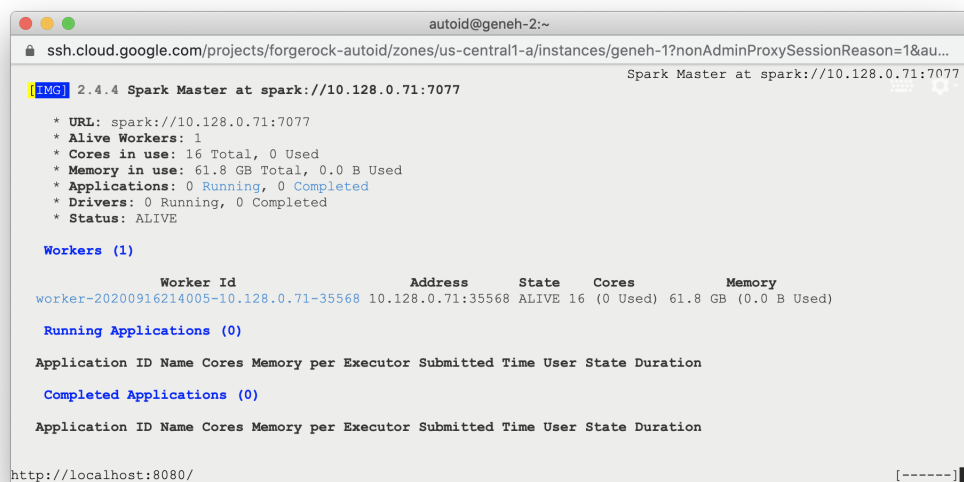
Check Spark:

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

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

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

▼ [Click to See an Example of the Spark Dashboard](#)



```
autoid@geneh-2:~  
ssh.cloud.google.com/projects/forgerock-autoid/zones/us-central1-a/instances/geneh-1?nonAdminProxySessionReason=1&au...  
[IMG] 2.4.4 Spark Master at spark://10.128.0.71:7077 Spark Master at spark://10.128.0.71:7077  
* URL: spark://10.128.0.71:7077  
* Alive Workers: 1  
* Cores in use: 16 Total, 0 Used  
* Memory in use: 61.8 GB Total, 0.0 B Used  
* Applications: 0 Running, 0 Completed  
* Drivers: 0 Running, 0 Completed  
* Status: ALIVE  
  
Workers (1)  


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

  
Running Applications (0)  
Application ID Name Cores Memory per Executor Submitted Time User State Duration  
  
Completed Applications (0)  
Application ID Name Cores Memory per Executor Submitted Time User State Duration  
  
http://localhost:8080/
```

Start the Analytics

If the previous installation steps all succeeded, you must now prepare your data's entity definitions, data sources, and attribute mappings prior to running your analytics jobs.

These steps are required and are critical for a successful analytics process.

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

Install a Multi-Node Deployment

This 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 Pro script to pull a Docker image from ForgeRock's Google Cloud Registry repository with the microservices and analytics 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.

NOTE

The procedures are similar to multinode deployments using older Autonomous Identity release, except that you must install and configure the dependent software packages (for example, Apache Cassandra/MongoDB, Apache Spark and Livy, OpenSearch and OpenSearch Dashboards, and Docker) prior to running Autonomous Identity.

Summary of the installation steps

To set up the 2022.11.2 deployment, run the following steps:

- Prerequisites
- Set Up the Nodes
- Install third-party components
- Set Up SSH on the Deployer
- Set Up a shared data folder
- Install Autonomous Identity
- Set the Cassandra Replication Factor

Prerequisites

Deploy Autonomous Identity on a multi-node target on Redhat Linux Enterprise 8 or CentOS Stream 8. The following are prerequisites:

- **Operating System.** The target machine requires Redhat Linux Enterprise 8 or CentOS Stream 8. The deployer machine can use any operating system as long as

Docker is installed. For this chapter, we use Redhat Linux Enterprise 8 as its base operating system.

IMPORTANT

If you are upgrading Autonomous Identity on a RHEL 7/CentOS 7, the upgrade to 2022.11 uses RHEL 7/CentOS 7 only. For new and clean installations, Autonomous Identity requires RHEL 8 or CentOS Stream 8 only.

- **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. The registry key is only available to ForgeRock Autonomous Identity customers. For specific instructions on obtaining the registry key, refer to [How To Configure Service Credentials \(Push Auth, Docker\) in Backstage](#).
- **Filesystem Requirements.** Autonomous Identity requires a shared filesystem accessible from the Spark main, Spark worker, analytics hosts, and application layer. The shared filesystem should be mounted at the same mount directory on all of those hosts. If the mount directory for the shared filesystem is different from the default, `/data`, update the `/autoid-config/vars.yml` file to point to the correct directories:

```
analytics_data_dir: /data
analytics_conf_dir: /data/conf
```

- **Architecture Requirements.** Make sure that the Spark main 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 OpenSearch data and OpenSearch Dashboards nodes. For best performance in production, dedicate a

separate node to OpenSearch, data nodes, and OpenSearch Dashboards.

- **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
```

NOTE

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 Nodes

Set up three virtual machines.

1. Create a Redhat Linux Enterprise 8 or CentOS Stream 8 virtual machine: N2 4 core and 16 GB. Verify your operating system.

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

IMPORTANT

For multinode deployments, there is a known issue with RHEL 8/CentOS Stream 8 and overlay network configurations. Refer to [Known Issues in 2022.11.0](#).

2. Set the user for the target node to `autoid`. In this example, create user `autoid`:

```
sudo adduser autoid
sudo passwd autoid
echo "autoid ALL=(ALL) NOPASSWD:ALL" | sudo tee
/etc/sudoers.d/autoid
sudo usermod -aG wheel autoid
su - autoid
```

3. Optional. 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
```


4. Install the following packages needed in the Autonomous Identity deployment:
 - **Java 11**. For example, `sudo dnf install java-11-openjdk-devel`.
 - **wget**. For example, `sudo dnf install wget`.
 - **unzip**. For example, `sudo dnf install unzip`.
 - **elinks**. For example, `sudo yum install -y elinks`.
 - **Python 3.8.13**. Refer to <https://docs.python.org/release/3.8.13/>.
5. Repeat this procedure for the other nodes.

Install third-party components

Set up a machine with the required third-party software dependencies. Refer to: [Install third-party components](#).

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

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 data folder

The Docker main and worker nodes plus the analytics main and worker nodes require a shared data 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 Main 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, refer to [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
```

Install Autonomous Identity

Make sure you have the following prerequisites:

- IP address of machines running OpenSearch, MongoDB, or Cassandra.
- The Autonomous Identity user should have permission to write to `/opt/autoid` on all machines
- To download the deployment images for the install, you still need your registry key to log into the ForgeRock Google Cloud Registry to download the artifacts.
- Make sure you have the proper OpenSearch certificates with the exact names for both pem and JKS files copied to `~/autoid-config/certs/elastic` :
 - `esnode.pem`
 - `esnode-key.pem`
 - `root-ca.pem`
 - `elastic-client-keystore.jks`
 - `elastic-server-truststore.jks`
- Make sure you have the proper MongoDB certificates with exact names for both pem and JKS files copied to `~/autoid-config/certs/mongo` :
 - `mongo-client-keystore.jks`
 - `mongo-server-truststore.jks`

- mongodb.pem
- rootCA.pem
- Make sure you have the proper Cassandra certificates with exact names for both pem and JKS files copied to ~/autoid-config/certs/cassandra:
 - Zoran-cassandra-client-cer.pem
 - Zoran-cassandra-client-keystore.jks
 - Zoran-cassandra-server-cer.pem
 - zoran-cassandra-server-keystore.jks
 - Zoran-cassandra-client-key.pem
 - Zoran-cassandra-client-truststore.jks
 - Zoran-cassandra-server-key.pem
 - Zoran-cassandra-server-truststore.jks

Install Autonomous Identity:

1. Create the autoid-config directory.

```
mkdir autoid-config
```

2. Change to the directory.

```
cd autoid-config
```

3. Log in to the ForgeRock Google Cloud Registry using the registry key. The registry key is only available to ForgeRock Autonomous Identity customers. For specific instructions on obtaining the registry key, refer to [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
```

The following output is displayed:

```
Login Succeeded
```

4. 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-pro:2022.11.2 create-
template
```

5. Create a certificate directory for elastic.

```
mkdir -p autoid-config/certs/elastic
```

6. Copy the OpenSearch certificates and JKS files to `autoid-config/certs/elastic`.

7. Create a certificate directory for MongoDB.

```
mkdir -p autoid-config/certs/mongo
```

8. Copy the MongoDB certificates and JKS files to `autoid-config/certs/mongo`.

9. Create a certificate directory for Cassandra.

```
mkdir -p autoid-config/certs/cassandra
```

10. Copy the Cassandra certificates and JKS files to `autoid-config/certs/cassandra`.

11. Update the `hosts` file with the IP addresses of the machines. The `hosts` file must include the IP addresses for Docker nodes, Spark main/livy, and the MongoDB master. While the deployer pro does not install or configure the MongoDB main server, the entry is required to run the MongoDB CLI to seed the Autonomous Identity schema.

```
[docker-managers]
```

```
[docker-workers]
```

```
[docker:children]
```

```
docker-managers
```

```
docker-workers
```

```
[spark-master-livy]
```

```
[cassandra-seeds]
```

```
#For replica sets, add the IPs of all Cassandra nodes
```

```
[mongo_master]
```

```
# Add the MongoDB main node in the cluster deployment
```

```
# For example: 10.142.15.248 mongodb_master=True
```

```
[odfe-master-node]
# Add only the main node in the cluster deployment
```

12. Update the `vars.yml` file:

- a. Set `db_driver_type` to `mongo` or `cassandra`.
- b. Set `elastic_host`, `elastic_port`, and `elastic_user` properties.
- c. Set `kibana_host`.
- d. Set the Apache livy install directory.
- e. Ensure the `elastic_user`, `elastic_port`, and `mongo_part` are correctly configured.
- f. Update the `vault.yml` passwords for elastic and mongo to reflect your installation.
- g. Set the Cassandra-related parameters in the `vars.yml` file. Default values are:

```
cassandra:
  enable_ssl: "true"
  contact_points: 10.142.15.248 # comma separated
  values in case of replication set
  port: 9042
  username: zoran_dba
  cassandra_keystore_password: "Acc#1234"
  cassandra_truststore_password: "Acc#1234"
  ssl_client_key_file: "zoran-cassandra-client-key.pem"
  ssl_client_cert_file: "zoran-cassandra-client-
cer.pem"
  ssl_ca_file: "zoran-cassandra-server-cer.pem"
  server_truststore_jks: "zoran-cassandra-server-
truststore.jks"
  client_truststore_jks: "zoran-cassandra-client-
truststore.jks"
  client_keystore_jks: "zoran-cassandra-client-
keystore.jks"
```

13. Download images:

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

14. Install Apache Livy.

- o The official release of Apache Livy does not support Apache Spark 3.3.1 or 3.3.2. ForgeRock has re-compiled and packaged Apache Livy to work with

Apache Spark 3.3.1 hadoop 3 and Apache Spark 3.3.2 hadoop 3. Use the zip file located at `autoid-config/apache-livy/apache-livy-0.8.0-incubating-SNAPSHOT-bin.zip` to install Apache Livy on the Spark-Livy machine.

- For Livy configuration, refer to <https://livy.apache.org/get-started/>.

15. On the Spark-Livy machine, run the following commands to install the python package dependencies:

- a. Change to the `/opt/autoid` directory:

```
cd /opt/autoid
```

- b. Create a `requirements.txt` file with the following content:

```
six==1.11
certifi==2019.11.28
python-dateutil==2.8.1
jsonschema==3.2.0
cassandra-driver
numpy==1.19.5
pyarrow==0.16.0
wrapt==1.11.0
PyYAML==5.4
requests
pymongo
pandas==1.0.5
tabulate
openpyxl
```

- c. Install the requirements file:

```
pip3 install -r requirements.txt
```

16. Make sure that the `/opt/autoid` directory exists and that it is both readable and writable.

17. Run the deployer script:

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

18. On the Spark-Livy machine, run the following commands to install the Python egg file:

- a. Install the egg file:


```
sudo /usr/local/bin/pip3.8 install setuptools==46.00
cd /opt/autoid/eggs
sudo /usr/local/bin/easy_install-3.8 autoid_analytics-
2021.3-py3.6.egg
```

b. Source the `.bashrc` file:

```
source ~/.bashrc
```

c. Restart Spark and Livy.

```
./spark/sbin/stop-all.sh
./livy/bin/livy-server stop

./spark/sbin/start-all.sh
./livy/bin/livy-server start
```

Set the Cassandra 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. Start the Cassandra shell, `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,
  ...
```

2. Restart Cassandra on this node.
3. 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. Make sure Cassandra is running in cluster mode. For example

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

Check MongoDB

Check MongoDB:

1. Make sure MongoDB is running. For example:

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

Check Apache Spark

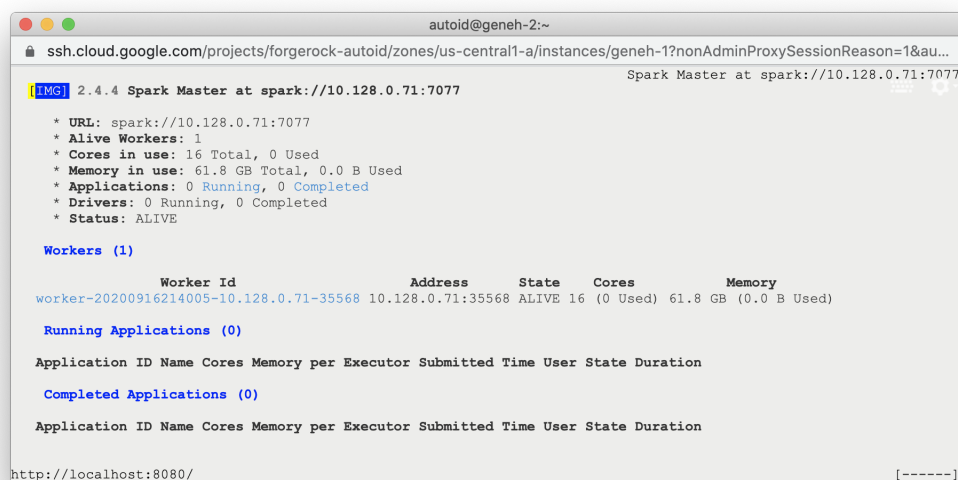
Check Spark:

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

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

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

▼ [Click to See an Example of the Spark Dashboard](#)



```
autoid@geneh-2:~
ssh.cloud.google.com/projects/forgerock-autoid/zones/us-central1-a/instances/geneh-1?nonAdminProxySessionReason=1&au...
Spark Master at spark://10.128.0.71:7077
2.4.4 Spark Master at spark://10.128.0.71:7077
* URL: spark://10.128.0.71:7077
* Alive Workers: 1
* Cores in use: 16 Total, 0 Used
* Memory in use: 61.8 GB Total, 0.0 B Used
* Applications: 0 Running, 0 Completed
* Drivers: 0 Running, 0 Completed
* Status: ALIVE

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

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

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

http://localhost:8080/ [-----]
```

Start the Analytics

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

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

Install a Multi-Node Air-Gapped Deployment

This chapter presents instructions on deploying Autonomous Identity in a multi-node air-gapped or offline target machine with no external Internet connectivity. ForgeRock provides a deployer script that pulls a Docker image from ForgeRock's Google Cloud Registry 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 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.

Summary of the installation steps

To set up the 2022.11.2 deployment, run the following steps:

- Prerequisites
- Set up the Nodes
- Set Up SSH on the Deployer
- Prepare the Tar File
- Install third-party components
- Install Autonomous Identity Air-Gapped
- Set the Replication Factor

Prerequisites

Deploy Autonomous Identity on a multi-node air-gapped target on Redhat Linux Enterprise 8 or CentOS Stream 8. The following are prerequisites:

- **Operating System.** The target machine requires Redhat Linux Enterprise 8 or CentOS Stream 8. The deployer machine can use any operating system as long as Docker is installed. For this chapter, we use Redhat Linux Enterprise 8 as its base operating system.

IMPORTANT

If you are upgrading Autonomous Identity on a RHEL 7/CentOS 7, the upgrade to 2022.11 uses RHEL 7/CentOS 7 only. For new and clean installations, Autonomous Identity requires RHEL 8 or CentOS Stream 8 only.

- **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. The registry key is only available to ForgeRock Autonomous

Identity customers. For specific instructions on obtaining the registry key, refer to [How To Configure Service Credentials \(Push Auth, Docker\) in Backstage](#).

- **Filesystem Requirements.** Autonomous Identity requires a shared filesystem accessible from the Spark main, Spark worker, analytics hosts, and application layer. The shared filesystem should be mounted at the same mount directory on all of those hosts. If the mount directory for the shared filesystem is different from the default, /data , update the /autoid-config/vars.yml file to point to the correct directories:

```
analytics_data_dir: /data
analytics_conf_dir: /data/conf
```

- **Architecture Requirements.** Make sure that the Spark main 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 OpenSearch data and OpenSearch Dashboards nodes. For best performance in production, dedicate a separate node to OpenSearch, data nodes, and OpenSearch Dashboards.
- **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
```

NOTE

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 Nodes

Set up each node as presented in [Set Up the Nodes](#).

Make sure you have sufficient storage for your particular deployment. For more information on sizing considerations, refer to [Deployment Planning Guide](#).

IMPORTANT

For multinode deployments, there is a known issue with RHEL 8/CentOS Stream 8 and overlay network configurations. Refer to [Known Issues in 2022.11.0](#).

Install third-party components

Set up a machine with the required third-party software dependencies. Refer to: [Install third-party components](#).

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 using the registry key. The registry key is only available to ForgeRock Autonomous Identity customers. For specific instructions on obtaining the registry key, refer to [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  
↗
```

The following output is displayed:

```
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-pro:2022.11.2 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 Autonomous Identity Air-Gapped

Make sure you have the following prerequisites:

- IP address of machines running OpenSearch, MongoDB, or Cassandra.
- The Autonomous Identity user should have permission to write to `/opt/autoid` on all machines
- To download the deployment images for the install, you still need your registry key to log into the ForgeRock Google Cloud Registry to download the artifacts.
- Make sure you have the proper OpenSearch certificates with the exact names for both pem and JKS files copied to `~/autoid-config/certs/elastic`:
 - `esnode.pem`
 - `esnode-key.pem`
 - `root-ca.pem`

- elastic-client-keystore.jks
- elastic-server-truststore.jks
- Make sure you have the proper MongoDB certificates with exact names for both pem and JKS files copied to `~/autoid-config/certs/mongo`:
 - mongo-client-keystore.jks
 - mongo-server-truststore.jks
 - mongodb.pem
 - rootCA.pem
- Make sure you have the proper Cassandra certificates with exact names for both pem and JKS files copied to `~/autoid-config/certs/cassandra`:
 - Zoran-cassandra-client-cer.pem
 - Zoran-cassandra-client-keystore.jks
 - Zoran-cassandra-server-cer.pem
 - zoran-cassandra-server-keystore.jks
 - Zoran-cassandra-client-key.pem
 - Zoran-cassandra-client-truststore.jks
 - Zoran-cassandra-server-key.pem
 - Zoran-cassandra-server-truststore.jks

Install Autonomous Identity:

1. Change to the directory.

```
cd autoid-config
```

2. 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-pro:2022.11.2 create-  
template
```

3. Create a certificate directory for elastic.

```
mkdir -p autoid-config/certs/elastic
```

4. Copy the OpenSearch certificates and JKS files to `autoid-config/certs/elastic`.
5. Create a certificate directory for MongoDB.

```
mkdir -p autoid-config/certs/mongo
```

6. Copy the MongoDB certificates and JKS files to `autoid-config/certs/mongo`.
7. Create a certificate directory for Cassandra.

```
mkdir -p autoid-config/certs/cassandra
```

8. Copy the Cassandra certificates and JKS files to `autoid-config/certs/cassandra`.
9. Update the `hosts` file with the IP addresses of the machines. The `hosts` file must include the IP addresses for Docker nodes, Spark main/livy, and the MongoDB master. While the deployer `pro` does not install or configure the MongoDB main server, the entry is required to run the MongoDB CLI to seed the Autonomous Identity schema.

```
[docker-managers]

[docker-workers]

[docker:children]
docker-managers
docker-workers

[spark-master-livy]

[cassandra-seeds]
#For replica sets, add the IPs of all Cassandra nodes

[mongo_master]
# Add the MongoDB main node in the cluster deployment
# For example: 10.142.15.248 mongodb_master=True

[odfe-master-node]
# Add only the main node in the cluster deployment
```

10. Update the `vars.yml` file:
 - a. Set `offline_mode` to `true`.
 - b. Set `db_driver_type` to `mongo` or `cassandra`.
 - c. Set `elastic_host`, `elastic_port`, and `elastic_user` properties.

- d. Set `kibana_host`.
- e. Set the Apache livy install directory.
- f. Ensure the `elastic_user`, `elastic_port`, and `mongo_part` are correctly configured.
- g. Update the `vault.yml` passwords for elastic and mongo to reflect your installation.
- h. Set the Cassandra-related parameters in the `vars.yml` file. Default values are:

```
cassandra:
  enable_ssl: "true"
  contact_points: 10.142.15.248 # comma separated
  values in case of replication set
  port: 9042
  username: zoran_dba
  cassandra_keystore_password: "Acc#1234"
  cassandra_truststore_password: "Acc#1234"
  ssl_client_key_file: "zoran-cassandra-client-key.pem"
  ssl_client_cert_file: "zoran-cassandra-client-
cer.pem"
  ssl_ca_file: "zoran-cassandra-server-cer.pem"
  server_truststore_jks: "zoran-cassandra-server-
truststore.jks"
  client_truststore_jks: "zoran-cassandra-client-
truststore.jks"
  client_keystore_jks: "zoran-cassandra-client-
keystore.jks"
```

11. Install Apache Livy.

- The official release of Apache Livy does not support Apache Spark 3.3.1 or 3.3.2. ForgeRock has re-compiled and packaged Apache Livy to work with Apache Spark 3.3.1 hadoop 3 and Apache Spark 3.3.2 hadoop 3. Use the zip file located at `autoid-config/apache-livy/apache-livy-0.8.0-incubating-SNAPSHOT-bin.zip` to install Apache Livy on the Spark-Livy machine.
- For Livy configuration, refer to <https://livy.apache.org/get-started/>.

12. On the Spark-Livy machine, run the following commands to install the python package dependencies:

- a. Change to the `/opt/autoid` directory:

```
cd /opt/autoid
```

b. Create a `requirements.txt` file with the following content:

```
six==1.11
certifi==2019.11.28
python-dateutil==2.8.1
jsonschema==3.2.0
cassandra-driver
numpy==1.19.5
pyarrow==0.16.0
wrapt==1.11.0
PyYAML==5.4
requests
pymongo
pandas==1.0.5
tabulate
openpyxl
```

c. Install the requirements file:

```
pip3 install -r requirements.txt
```

13. Make sure that the `/opt/autoid` directory exists and that it is both readable and writable.

14. Run the deployer script:

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

15. On the Spark-Livy machine, run the following commands to install the Python egg file:

a. Install the egg file:

```
sudo /usr/local/bin/pip3.8 install setuptools==46.00
cd /opt/autoid/eggs
sudo /usr/local/bin/easy_install-3.8 autoid_analytics-
2021.3-py3.6.egg
```

b. Source the `.bashrc` file:

```
source ~/.bashrc
```

c. Restart Spark and Livy.

```
./spark/sbin/stop-all.sh
./livy/bin/livy-server stop
```

```
./spark/sbin/start-all.sh
./livy/bin/livy-server start
```

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.

Refer to [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 steps 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. Refer to 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. 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.

IMPORTANT

Clean installations of Autonomous Identity 2022.11.0 and subsequent upgrade to 2022.11.1 to 2022.11.2 use the new deployer pro script. Upgrades from version 2021.8.7 to 2022.11.0 to 2022.11.2 use the older deployer script. The upgrade procedures differ slightly between the deployer pro and deployer versions, primarily in certificates directory creation (deployer versions) and using the proper image name during the `create-template` command (deployer pro and deployer versions).

The following chart summarizes these upgrade paths:

Table 1: Upgrade Paths

| Version | Upgrade To | Refer to |
|-------------------------------------|-------------------------------------|---|
| 2022.11.x (deployer-pro) | 2022.11.2 (deployer-pro) | <ul style="list-style-type: none"> • Upgrade from Autonomous Identity 2022.11.x to 2022.11.2 using deployer pro |
| 2022.11.x Air-Gapped (deployer-pro) | 2022.11.2 Air-Gapped (deployer-pro) | <ul style="list-style-type: none"> • Upgrade from Autonomous Identity 2022.11.x to 2022.11.2 Air-Gapped using deployer pro |
| 2022.11.x (deployer) | 2022.11.2 (deployer) | <ul style="list-style-type: none"> • Upgrade from Autonomous Identity 2022.11.x to 2022.11.2 using the deployer |

| Version | Upgrade To | Refer to |
|---------------------------------|---------------------------------|---|
| 2022.11.x Air-Gapped (deployer) | 2022.11.2 Air Gapped (deployer) | <ul style="list-style-type: none"> Upgrade from Autonomous Identity 2022.11.x to 2022.11.2 Air-Gapped using the deployer |

Upgrade from Autonomous Identity 2022.11.x to 2022.11.2 using deployer pro

The following instructions are for upgrading from Autonomous Identity version 2022.11.0 or 2022.11.1 to the latest version **2022.11.2** in non air-gapped deployments using the **deployer pro**.

IMPORTANT

The following steps assume you ran a fresh install of Autonomous Identity 2022.11.0, which uses deployer pro. Make sure you have upgraded your third-party software packages to the supported versions prior to upgrade.

Upgrade from 2022.11.x to 2022.11.2 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-2022.11.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 2022.11.x `~/autoid-config` directory or move it to another location.

```
mv ~/autoid-config ~/backup-2022.11.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. Set up your certificate directories for OpenSearch, MongoDB, or Cassandra for the deployer:

a. Create a certificate directory OpenSearch:

```
mkdir -p autoid-config/certs/elastic
```

b. Copy the OpenSearch certificates and JKS files to `autoid-config/certs/elastic`.

c. Create a certificate directory for MongoDB (if you use MongoDB):

```
mkdir -p autoid-config/certs/mongo
```

d. Copy the MongoDB certificates and JKS files to `autoid-config/certs/mongo`.

e. Create a certificate directory for Cassandra (if you use Cassandra):

```
mkdir -p autoid-config/certs/cassandra
```

f. Copy the Cassandra certificates and JKS files to `autoid-config/certs/cassandra`.

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

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

8. Change the permission on the SSH key.

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

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


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


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

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

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

11. Log in to the ForgeRock Google Cloud Registry 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
```

12. 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-pro:2022.11.2 create-
template
```

13. 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.

14. 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
```

15. Prune old Docker images before running the upgrade command:

a. Get all of the Docker images:

```
docker images
```

b. Identify the images that are Autonomous Identity-related. They start with the URL of the ForgeRock Google cloud registry (ForgeRock GCR). For example:

| REPOSITORY | IMAGE ID | CREATED | SIZE | TAG |
|---|--------------|-------------|--------|-----------|
| <ForgeRock GCR>/ci/develop/deployer | 075481cea4c2 | 2 hours ago | 823MB | 650879186 |
| <ForgeRock GCR>/ci/develop/offline-packages | e1a90f389ccc | 2 hours ago | 3.03GB | 650879186 |
| <ForgeRock GCR>/ci/develop/zoran-ui | bd303a28b5df | 2 hours ago | 35.3MB | 650879186 |
| <ForgeRock GCR>/ci/develop/zoran-api | 114d1aca5b0a | 2 hours ago | 421MB | 650879186 |
| <ForgeRock GCR>/ci/develop/nginx | 43b410661269 | 2 hours ago | 16.7MB | 650879186 |
| <ForgeRock GCR>/ci/develop/jas | 2821e5c365d8 | 2 hours ago | 491MB | 650879186 |

c. Remove the old images using the `docker rmi` command. For example:

```
docker rmi -f <image ID>
```

Example:

```
docker rmi -f 075481cea4c2
```

d. Repeat the previous command to remove all of the Autonomous Identity-related Docker images.

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

```
docker swarm leave
```

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

```
sudo systemctl restart docker
```

19. 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
```

20. Run the upgrade:

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

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

```
sudo mv ~/backup-data-conf-2022.11.x /data/conf
```

23. 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.
24. Log out, and then log back in to Autonomous Identity.

You have successfully upgraded your Autonomous Identity server to 2022.11.2.

Upgrade from Autonomous Identity 2022.11.x to 2022.11.2 Air-Gapped using deployer pro

The following instructions are for upgrading from Autonomous Identity version 2022.11.0 or 2022.11.1 to **2022.11.2** on air-gapped deployments using the **deployer pro**.

IMPORTANT

The following steps assume you ran a fresh install of Autonomous Identity 2022.11.0, which uses `deployer pro`. Make sure you have upgraded your third-party software packages to the supported versions prior to upgrade.

Upgrade from 2022.11.x to 2022.11.2 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-2022.11.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 2022.11.x `~/autoid-config` directory or move it to another location.

```
mv ~/autoid-config ~/backup-2022.11.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. Set up your certificate directories for OpenSearch, MongoDB, or Cassandra for the deployer:

- a. Create a certificate directory OpenSearch:

```
mkdir -p autoid-config/certs/elastic
```

- b. Copy the OpenSearch certificates and JKS files to `autoid-config/certs/elastic`.

- c. Create a certificate directory for MongoDB (if you use MongoDB):

```
mkdir -p autoid-config/certs/mongo
```

- d. Copy the MongoDB certificates and JKS files to `autoid-config/certs/mongo`.

- e. Create a certificate directory for Cassandra (if you use Cassandra):

```
mkdir -p autoid-config/certs/cassandra
```

- f. Copy the Cassandra certificates and JKS files to `autoid-config/certs/cassandra`.

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

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

8. Change the permission on the SSH key.

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

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

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

10. Log in to the ForgeRock Google Cloud Registry 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
```

11. 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-pro:2022.11.2 create-  
template
```

12. 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.

13. 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
```

14. 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
```

15. Prune old Docker images before running the upgrade command:

- a. Get all of the Docker images:

```
docker images
```

- b. Identify the images that are Autonomous Identity-related. They start with the URL of the ForgeRock Google cloud registry (ForgeRock GCR). For example:

| REPOSITORY | IMAGE ID | CREATED | SIZE | TAG |
|---|--------------|-------------|--------|-----------|
| <ForgeRock GCR>/ci/develop/deployer | 075481cea4c2 | 2 hours ago | 823MB | 650879186 |
| <ForgeRock GCR>/ci/develop/offline-packages | e1a90f389ccc | 2 hours ago | 3.03GB | 650879186 |
| <ForgeRock GCR>/ci/develop/zoran-ui | bd303a28b5df | 2 hours ago | 35.3MB | 650879186 |

```
<ForgeRock GCR>/ci/develop/zoran-api      650879186
114d1aca5b0a  2 hours ago  421MB
<ForgeRock GCR>/ci/develop/nginx          650879186
43b410661269  2 hours ago  16.7MB
<ForgeRock GCR>/ci/develop/jas           650879186
2821e5c365d8  2 hours ago  491MB
```

c. Remove the old images using the `docker rmi` command. For example:

```
docker rmi -f <image ID>
```

Example:

```
docker rmi -f 075481cea4c2
```

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

```
docker swarm leave
```

17. From the deployer, restart Docker:

```
sudo systemctl restart docker
```

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

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

19. Copy the `autoid-packages.tgz`, `deployer.sh`, and SSH key (`id_rsa`) to a portable hard drive.

20. On the air-gapped target machine, backup your previous `~/autoid-config` directory, and then create a new `~/autoid-config` directory.

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

21. Copy the `autoid-package.tgz` tar file, `deployer.sh`, and SSH key from the portable storage device to the `/autoid-config` folder.

22. Unpack the tar file.

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

23. Set up your certificate directories for OpenSearch, MongoDB, or Cassandra for the deployer:

a. Create a certificate directory OpenSearch:


```
mkdir -p autoid-config/certs/elastic
```

b. Copy the OpenSearch certificates and JKS files to `autoid-config/certs/elastic`.

c. Create a certificate directory for MongoDB (if you use MongoDB):

```
mkdir -p autoid-config/certs/mongo
```

d. Copy the MongoDB certificates and JKS files to `autoid-config/certs/mongo`.

e. Create a certificate directory for Cassandra (if you use Cassandra):

```
mkdir -p autoid-config/certs/cassandra
```

f. Copy the Cassandra certificates and JKS files to `autoid-config/certs/cassandra`.

24. Copy the SSH key to the `~/autoid-config` directory.

25. Change the privileges to the file.

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

26. Change to the configuration directory.

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

27. 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: <ForgeRock Google cloud registry
URL>/deployer:2022.11.2
```

28. 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
```

29. 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.

30. Run the upgrade:

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

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

```
sudo mv ~/backup-data-conf-2022.11.x /data/conf
```

32. 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.
33. Log out, and then log back in to Autonomous Identity.

You have successfully upgraded your Autonomous Identity server to 2022.11.2.

Upgrade from Autonomous Identity 2022.11.x to 2022.11.2 using the deployer

The following instructions are for upgrading from Autonomous Identity version 2022.11.0 or 2022.11.1 to the latest version **2022.11.2** in non air-gapped deployments using the **deployer**.

Upgrade from 2022.11.x to 2022.11.2 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-2022.11.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 2022.11.x `~/autoid-config` directory or move it to another location.

```
mv ~/autoid-config ~/backup-2022.11.x
```

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

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

5. Copy your `autoid_registry_key.json` from your backup directory 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. On the deployer node, change to the `~/autoid-config` directory.

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

10. Log in to the ForgeRock Google Cloud Registry 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
```

11. 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:2022.11.2 create-
template
```

12. 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.

13. 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
```

14. Prune old Docker images before running the upgrade command:

a. Get all of the Docker images:

```
docker images
```

b. Identify the images that are Autonomous Identity-related. They start with the URL of the ForgeRock Google cloud registry (ForgeRock GCR). For example:

| REPOSITORY | IMAGE ID | CREATED | SIZE | TAG |
|---|--------------|-------------|--------|-----------|
| <ForgeRock GCR>/ci/develop/deployer | 075481cea4c2 | 2 hours ago | 823MB | 650879186 |
| <ForgeRock GCR>/ci/develop/offline-packages | e1a90f389ccc | 2 hours ago | 3.03GB | 650879186 |
| <ForgeRock GCR>/ci/develop/zoran-ui | bd303a28b5df | 2 hours ago | 35.3MB | 650879186 |
| <ForgeRock GCR>/ci/develop/zoran-api | 114d1aca5b0a | 2 hours ago | 421MB | 650879186 |
| <ForgeRock GCR>/ci/develop/nginx | 43b410661269 | 2 hours ago | 16.7MB | 650879186 |
| <ForgeRock GCR>/ci/develop/jas | 2821e5c365d8 | 2 hours ago | 491MB | 650879186 |

c. Remove the old images using the `docker rmi` command. For example:

```
docker rmi -f <image ID>
```

Example:

```
docker rmi -f 075481cea4c2
```

d. Repeat the previous command to remove all of the Autonomous Identity-related Docker images.

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

```
docker swarm leave
```

16. Enter `exit` to end your SSH session.
17. From the deployer, restart Docker command:

```
sudo systemctl restart docker
```

18. 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
```

19. Run the upgrade:

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

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

```
sudo mv ~/backup-data-conf-2022.11.x /data/conf
```

22. 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.
23. Log out, and then log back in to Autonomous Identity.

You have successfully upgraded your Autonomous Identity server to 2022.11.2.

Upgrade from Autonomous Identity 2022.11.x to 2022.11.2 Air-Gapped using the deployer

The following instructions are for upgrading from Autonomous Identity version 2022.11.0 or 2022.11.1 to 2022.11.2 on air-gapped deployments using the **deployer**.

Upgrade from 2022.11.x to 2022.11.2 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-2022.11.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 2022.11.x ~/autoid-config directory or move it to another location.

```
mv ~/autoid-config ~/backup-2022.11.x
```

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

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

5. Copy your autoid_registry_key.json from your backup directory 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. On the deployer node, change to the ~/autoid-config directory.

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

9. Log in to the ForgeRock Google Cloud Registry 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
```

10. 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:2022.11.2 create-
template
```

11. 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.

12. 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
```

13. 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
```

14. Prune old Docker images before running the upgrade command:

a. Get all of the Docker images:

```
docker images
```

b. Identify the images that are Autonomous Identity-related. They start with the URL of the ForgeRock Google cloud registry (ForgeRock GCR). For example:

| REPOSITORY | IMAGE ID | CREATED | SIZE | TAG |
|---|--------------|-------------|--------|-----------|
| <ForgeRock GCR>/ci/develop/deployer | 075481cea4c2 | 2 hours ago | 823MB | 650879186 |
| <ForgeRock GCR>/ci/develop/offline-packages | e1a90f389ccc | 2 hours ago | 3.03GB | 650879186 |
| <ForgeRock GCR>/ci/develop/zoran-ui | bd303a28b5df | 2 hours ago | 35.3MB | 650879186 |
| <ForgeRock GCR>/ci/develop/zoran-api | 114d1aca5b0a | 2 hours ago | 421MB | 650879186 |
| <ForgeRock GCR>/ci/develop/nginx | 43b410661269 | 2 hours ago | 16.7MB | 650879186 |
| <ForgeRock GCR>/ci/develop/jas | 2821e5c365d8 | 2 hours ago | 491MB | 650879186 |

c. Remove the old images using the `docker rmi` command. For example:

```
docker rmi -f <image ID>
```

Example:

```
docker rmi -f 075481cea4c2
```

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

```
docker swarm leave
```

16. From the deployer, restart Docker:

```
sudo systemctl restart docker
```

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

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

18. Copy the `autoid-packages.tgz`, `deployer.sh`, and SSH key (`id_rsa`) to a portable hard drive.

19. On the air-gapped target machine, backup your previous `~/autoid-config` directory, and then create a new `~/autoid-config` directory.

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

20. Copy the `autoid-package.tgz` tar file, `deployer.sh`, and SSH key from the portable storage device to the `/autoid-config` folder.

21. Unpack the tar file.

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

22. Copy the SSH key to the `~/autoid-config` directory.

23. Change the privileges to the file.

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

24. Change to the configuration directory.

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

25. 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: https://gcr.io/forgerock-
autoid/deployer:2022.11.2
```

26. 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
```

27. 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.

28. Run the upgrade:

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

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

```
sudo mv ~/backup-data-conf-2022.11.x /data/conf
```

30. 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.
31. Log out, and then log back in to Autonomous Identity.

You have successfully upgraded your Autonomous Identity server to 2022.11.2.

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 refer to the available Autonomous Identity ports, refer to [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-addresses#

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

```

# set the following API parameters when # SSO and LdapAndSSO
properties
# authentication_option is SSO or LdapAndSSO
# oidc_issuer:
# oidc_auth_url
# oidc_token_url:
# oidc_user_info_url:
# oidc_callback_url:
# oidc_jwks_url:
# oidc_client_scope:
# oidc_groups_attribute:
# oidc_uid_attribute:
# oidc_client_id:
# oidc_client_secret:
# admin_object_id:
# entitlement_owner_object_id:
# executive_object_id:
# supervisor_object_id:
# user_object_id:
# application_owner_object_id:
# role_owner_object_id:
# role_engineer_object_id:
# oidc_end_session_endpoint:
# oidc_logout_redirect_url:

# mongo config starts

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

# custom key
# password for inter-process authentication
#
# please regenerate this file on production environment with
command 'openssl rand -base64 741'
#mongodb_keyfile_content: |
#
8pYcxvCqoe89kcp33KuTtKVf5MoHGefjTnudrq5BosvWRoIxLowmdjrmUpVfAivh
#
CHjqM6w0zVBytAxH1lW+7teMYe6eDn2S/0/1YlRRiW57bWU3zjliW3VdguJar5i9
#

```

```
Z+1a8lI+0S9pWynbv9+Ao0aXFjSJYVxAm/w7DJbVRGcPhsPmExiSBDw8szfQ8PAU
#
2hwRl7nqPZZMMR+uQThg/zV9r0zHJmkqZts04UJSilG9euLCYrzW2hdoPuCrEDhu
#
Vsi5+nwAgYR9dP2oWkmGN1dwRe0ixSIM2UzFgpaXZaM0G6VztmFr1VXh8oFDRGM0
#
cGrFHcnGF7oUGfWnI2Cekngk64dHA2qd7WxXPbQ/svn9EfTY5aPw5lXzKA87Ds8p
#
KHVFUYvmA6wVsxb/riGLwc+XZl6M9gqHn1XSpsnYRjF6UzfRcRR2WyCxLZELaqu
#
iKxLKB5FYqMBH7Sqq3qBCtE53vZ7T1nefq5RFzmykviYP63Uhu/A2EQatrMnaFP1
#
TTG5CaPjob45CBSyMrheYRWKqxdWN93BTgiTW7p0U6RB0/OCUbsVX6IG3I9N8Uqt
#
l8Kc+7a0mtUqFkwo8w30prIOjStMrokxNsuK9KTUiPu2cj7gwYQ574vV3hQvQPAr
#
hnb9ohKr0zoPQt31iTj0FDkJzPepezqeq8F51HB56RZKpXdRTfY8G60a0T68cV5
#
vP106T/okFKr141FQ3CyYN5eRHyRTK99zTytrjoP2EbtIZ18z+bg/angRHYNzbgk
#
lc3jpiGzs1ZWHD0nx0mHCMhU4usEcFbV6F10xzlwrSEhHkeiununlCsNHatiDgzp
#
ZWLnP/mXKV992/Jhu0Z577DHlh+3JIYx0PceB9yzACJ8MNARHF7QpBkhtuGMGZpF
#
T+c73exupZFxItXs1Bnhe3djgE3MKKyYvxNUIbcTJoe7nhVMrw0/7lBSpVLvC4p3
#  wR700U0LdaGGQpslGtiE56SemgoP
```

```
# mongo config ends
```

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

```
jas:
```

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

```
mongo_port: 27017 # Port where Mongo is running
```

```
elastic_host: 10.128.0.28 # IP Address of master node where
OpenSearch is running
```

```
elastic_port: 9200          # Port of master node where
OpenSearch is running
elastic_user: elasticadmin # Opensearch username

kibana_host: 10.128.0.28   # IP Address of node where
OpenSearch Dashboard is running

apache_livy:
  dest_dir: /home/ansible/livy # Folder where livy is installed.
AutoID copies analytics files to this directory.

cassandra:
# Cassandra Nodes details.
  enable_ssl: "true"
# Set if SSL is enabled.
  contact_points:
# Comma seperated list of ip addresses - first ip is master#
  port: 9042
# Port where cassandra node is running
  username: zoranuser
# User created for AutoID to seed Schema
  cassandra_keystore_password: "Acc#1234"
# Keystore Password
  cassandra_truststore_password: "Acc#1234"
# Truststore Password
  ssl_client_key_file: "zoran-cassandra-client-key.pem"
# Cassandra Client Key File
  ssl_client_cert_file: "zoran-cassandra-client-cer.pem"
# Cassandra Client Cert File
  ssl_ca_file: "zoran-cassandra-server-cer.pem"
# Cassandra Server Root CA File
  server_truststore_jks: "zoran-cassandra-server-truststore.jks"
# Server Truststore file for services to connect
  client_truststore_jks: "zoran-cassandra-client-truststore.jks"
# Client Truststore file for services to connect
  client_keystore_jks: "zoran-cassandra-client-keystore.jks"
# Client Keystore file for services to use
```