Predictive Routing Deployment and Operations Guide

Deploying in High Availability Environments

4/6/2026

Deploying in High Availability Environments

Both AI Core Services (AICS) and Agent State Connector (ASC) support high availability (HA).

High availability (HA) is configured differently for each Predictive Routing component:

- AI Core Services (AICS) uses a multi-server architecture. It can be installed at a single site, or in a multi-site architecture. Genesys recommends that you install AICS on three or five servers. More servers mean higher availability: with three servers, the system can survive the failure of only one machine; with five servers, the system can survive the failure of two machines.

> **Important**
>
> - AICS is installed in Docker containers. Genesys does not ship Docker as a part of AICS. You must install Docker in your environment before you can load the AICS containers.
> - You might need an active internet connection to download additional libraries when installing Docker.

- Agent State Connector (ASC) is deployed in warm-standby mode, with primary and backup servers.
- The URS Strategy Subroutines and Composer Subroutines components run as part of your routing solution, and therefore use the HA architecture established for that solution.

HA for AICS

The HA deployment and operating information for AICS is divided into the following sections:

- Hardware Requirements
- Installing HA AICS - Single Data Center Architecture
- Installing HA AICS - Multiple Data Center Architecture
- Set Values for Environment Variables
- Load Balancing and HTTPS Configuration for HA AICS
- Required Ports for AICS Servers
- Scaling the AICS Deployment (jumps to the Scaling AI Core Services topic in this *Guide*)
- Cleaning Up Disk Space
- Installing into an Existing HA AICS Deployment
- Checking the Logs for the AICS Containers (jumps to the Logging topic in this *Guide*)

- Troubleshooting an AICS HA Deployment
- (Optional) Backing Up Your Data
- (Optional) Installing AICS on a Kubernetes Cluster
- (Optional) Mapping a Local Volume to a Container

Hardware Requirements

- AICS HA requires a cluster of at least three servers. Genesys recommends that you deploy an odd number of servers to be used for hosting highly available AICS system.
- Every server must meet the preconditions stated in Target Server Requirements and Recommendations (in the AI Core Services Single-Host Deployment topic in this *Guide*). This will be verified during installation.
- All servers must have networking set up between them, with the ports opened as specified in Required Ports for AICS Servers.
- All servers must have synchronized system clocks. You can use Network Time Protocol (NTP) for this.
- On every target server, port 3031 must be reachable by the load balancer.
- On every target server, you **MUST** create a separate disk partition for storing MongoDB data. Mount this partition as `/datadir`. The partition size depends on your expected data usage, but must be at least 50 GB. For disk partitioning, use standard Linux tools. The `/datadir` partition **MUST** exist before you install GPR and the user who is executing the GPR installation should have write access to the partition. Preliminary Step: Create a Separate Disk for the MongoDB Database explains how to check the free space in your **mongodb** directory.
- You must have at least 50 GB free disk space on the root partition.

Important

If you are running VMWare VXLAN, you might encounter a port conflict between VMWare VXLAN and Docker, both of which require port 4789. If you encounter this issue, Genesys recommends that you use a networking application such as Weave Net to manage networking among Docker containers. For additional information, consult the documentation for the respective products:

- For the Docker Swarm port requirements: Use swarm mode routing mesh
- For VMWare VXLAN port requirements: Ports and Protocols Required by NSX
- For Weave Net: Introducing Weave Net

Installing HA AICS - Single Data Center Architecture

Important

- The following instructions enable you to set up a **new** AICS HA deployment in a single data center. If you already have a single-server deployment of AICS installed, contact Genesys Customer Care for help migrating to an HA architecture.
- If you need to uninstall AICS from an HA environment, contact Genesys Customer Care for assistance.

Installation Procedure

Important

- Some installation steps require you to know the hostnames of the target servers. You can run the command `hostname` on each server in the cluster to get the hostnames. This document uses the terms *node-1-hostname*, *node-2-hostname*, *node-3-hostname*, and so on, to stand in for the real hostnames of the servers. You must use actual hostnames when executing the example commands shown in the following sections.
- All scripts for installing and operating AICS in an HA setup are located in the **IP_JOP_PRR_<*version_number*>_ENU_linux/ha-scripts/** directory.

1. **Copy the installation binary file:**
   Copy the ***.tar.gz** file to every server in the cluster. Make sure you follow recommendations about the user PR_USER and the installation location described in single-host installation.
2. **Unpack the installation binary file:**

   Unpack the file on every server in the cluster. To unpack, follow these steps:

   1. Copy the **IP_JOP_PRR_<*version_number*>_ENU_linux.tar.gz** installation binary file to the desired installation directory. Genesys recommends that you use the PR_USER home directory as the destination for the AICS installation package.
   2. From a command prompt, unpack the file using the following command to create the `IP_JOP_PRR_<version_number>_ENU_linux` directory:

      ```
      tar -xvzf IP_JOP_PRR_<version_number>_ENU_linux.tar.gz
      ```

   3. **Create a Docker Swarm cluster.**
      AICS uses Docker Swarm technology to ensure high availability of all its components. In order for AICS to be deployed in highly available manner, you must properly format the Docker Swarm cluster on your target servers.

      1. On the target server with the hostname *node-1-hostname*, execute following command to

initiate the Docker Swarm cluster:

```
docker swarm init
```

> **Important**
>
> If the system has multiple IP addresses, specify the **--advertise-addr** parameter so the correct address is chosen for communication between all nodes in the cluster. If you do not specify this parameter, an error similar to the following is generated: `Error response from daemon: could not choose an IP address to advertise since this system has multiple addresses on different interfaces (10.33.181.18 on ens160 and 178.139.129.20 on ens192) - specify one with --advertise-addr`.

The following is an example of the command to initiate the Docker Swarm cluster, specifying the address that is advertised to other members of the cluster:

```
docker swarm init --advertise-addr YOUR_IP_ADDRESS
```

You can also specify a network interface to advertise the interface address, as in the following example:

```
docker swarm init --advertise-addr YOUR_NETWORK_INTERFACE
```

2. Still on the node with the hostname *node-1-hostname*, execute the following command:

   ```
   docker swarm join-token manager
   ```

   The output of this command should look similar to the following:

   ```
   docker swarm join --token
   SWMTKN-1-4d6wgar0nbghws5gx6j912zf2fdawpud42njjwwkso1rf9sy9y-
   dsbdfid1ilds081yyy30rof1t 172.31.18.159:2377
   ```

3. Copy this command and execute it on **all** other nodes in cluster. This ensures that all other nodes join the same cluster and coordinates AICS deployment.
4. Now execute following command on the node with the hostname *node-1-hostname* in order to verify that cluster has been properly formed and that you can continue with installation:

   ```
   docker node ls
   ```

   The output of this command **MUST** show you all target servers in the cluster (*node-1-hostname*, *node-2-hostname*, ..., *node-X-hostname*). If you do not see a complete list of servers, do not proceed with AICS installation. The following is an example of output where all nodes joined the cluster and are all reachable:

   ```
   ID              HOSTNAME  STATUS  AVAILABILITY  MANAGER  STATUS
   vdxn4uzuvaxly9i0je8g0bhps *node-1-hostname Ready Active Leader
   908bvibmyg9w87la6php11q96 node-2-hostname Ready Active Reachable
   ersak4msppm0ymgd2y7lbkgne node-3-hostname Ready Active Reachable
   shzyj970n5932h3z7pdvyvjes node-4-hostname Ready Active Reachable
   zjy3ltqsp3m5uekci7nr06tlj node-5-hostname Ready Active Reachable
   ```

5.

Label MongoDB nodes in the cluster:

Follow the steps below to define your MongoDB nodes:

1. Decide how many MongoDB instances to install in your deployment. You must install odd number of MongoDB instances, and no fewer than three. A higher number means higher availability.

   > **Important**
   >
   > Only one MongoDB instance can run per target server.

2. On the server with the host name *node-1-hostname*, execute following command to see all the nodes currently in the cluster:

   ```
   docker node ls
   ```

3. Choose the servers where MongoDB instances will run. In single data center deployment it does not matter which servers you choose as long as they have fast disks (SSD) and enough disk space.
   The examples assume you chose the servers with the host names *node-1-hostname*, *node-2-hostname*, and *node-3-hostname* to run MongoDB instances.

4. Label the selected nodes appropriately. To do this, execute following commands on *node-1-hostname*:

   ```
   docker node update --label-add mongo.replica=1 $(docker node ls -q -f
   name=node-1-hostname)
   docker node update --label-add mongo.replica=2 $(docker node ls -q -f
   name=node-2-hostname)
   docker node update --label-add mongo.replica=3 $(docker node ls -q -f
   name=node-3-hostname)
   ```

   For a cluster with five MongoDB instances, you would also run these two additional commands (and you would have to have at least five servers in the cluster). Follow the established pattern to label additional nodes if you are using more than five.

   ```
   docker node update --label-add mongo.replica=4 $(docker node ls -q -f
   name=node-4-hostname)
   docker node update --label-add mongo.replica=5 $(docker node ls -q -f
   name=node-5-hostname)
   ```

6. **Label the Worker nodes in the cluster:**

   Decide how many workers you want to run and on which servers.

   - The minimum number of servers marked to run worker instances is two, but you can have more for increased scalability and high availability. This configuration is verified during AICS installation.
   - Each worker container scales independently and you can have multiple instances of the same worker type running on the same server.
   - Workers can be co-located with other containers (such as MongoDB).

   1. Execute following commands on the node with the hostname *node-1-hostname* to ensure that worker instances will run on nodes *node-1-hostname*, *node-2-hostname*, and *node-3-hostname*:

      ```
      docker node update --label-add worker=true $(docker node ls -q -f
      ```

```
name=node-1-hostname)
docker node update --label-add worker=true $(docker node ls -q -f
name=node-2-hostname)
docker node update --label-add worker=true $(docker node ls -q -f
name=node-3-hostname)
```

You can choose to label more nodes and make them available to run worker instances. You cannot label fewer than two nodes with `worker = true`

7. **Label the MinIO nodes in the cluster:**
   You should always label at least one node to run MinIO container. MinIO container is used for faster dataset uploads. We recommend to label two nodes to run MinIO.

   1. Execute following commands on the node with the hostname *node-1-hostname* to ensure that a MinIO instance will run on one of nodes *node-1-hostname* or *node-2-hostname*:

      ```
      docker node update --label-add minio=true $(docker node ls -q -f
      name=node-1-hostname)
      docker node update --label-add minio=true $(docker node ls -q -f
      name=node-2-hostname)
      ```

      - There is always only one MinIO instance running and it only runs on one of the properly-labeled nodes.

   2. Find on what node MinIO container is running by executing following command:

      ```
      docker service ps minio_server_minio --format {{.Node}}
      ```

   3. Find the public IP address of the MinIO node and make sure that the S3_ENDPOINT configuration parameter in **IP_JOP_PRR_<*version_number*>_ENU_linux/conf/tango.env** is configured in the following way:

      ```
      S3_ENDPOINT=https://PUBLIC_IP_OF_NODE_WHERE_MINIO_CONTAINER_RUNS:9000
      ```

      - The MinIO container can be co-located with other containers (such as MongoDB or workers).

8. **Note the Tango instances:**
   There is automatically one Tango instance running on every node (server) in the cluster. As you expand the cluster, new Tango instances are installed and started on the newly-created nodes.

9. **Install AICS in HA mode:**
   Your Docker Swarm cluster is now ready for AICS installation.

   1. To make the Docker images needed by AICS available on every server in the cluster, execute the following command on **every** server in the cluster:

      ```
      bash ha-scripts/install.sh
      ```

      If you are managing your MongoDB deployment externally, run the **install.sh** script with the **-externalMongo** flag, as follows:

      ```
      bash ha-scripts/install.sh -externalMongo
      ```

   2. To initialize the HA AICS deployment and start the application, execute the following command. This command also sets the password for your default user, super_user@genesys.com. Replace the variable `<'my_password'>` in the command below with a strong password, and record it securely for future reference.

```
cd ha-scripts; bash start.sh -l -p <'my_password'>
```

10. **Access AICS in HA mode:**
    Once your fully-installed AICS deployment has started up correctly, you can access AICS by using the IP address of any server in the cluster on port 3031 as: https://<IP_ADDRESS>:3031

> **Important**
>
> Genesys recommends that you install a load balancer in front of the cluster to make it easier to access AICS. See Load Balancing for HA AICS for details.

Installing HA AICS - Multiple Data Center Architecture

> **Important**
>
> The following instructions enable you to set up a **new** AICS HA deployment in a multiple data center environment. If you already have a single-server deployment of AICS installed, contact Genesys Customer Care for help migrating to an HA architecture.

The basic procedure for installing AICS in multiple data centers is the same as installing AICS in single data center. However, when deploying AICS in an environment with multiple data centers, there are some considerations and requirements in addition to those for a single data center.

- Before starting, ensure that you have a fast LAN/WAN that connects all of the servers and that all ports are open.
- Plan to spread all instances of the AICS components (Workers, MongoDB, Tango, MinIO) across your data centers to ensure that AICS continues to operate correctly if a single data center fails. ***This is most important for servers running MongoDB.***

Special Considerations for MongoDB Instances

- Spread labels across the data centers when labeling servers to run MongoDB replica set members.

> **Important**
>
> The AICS installation procedure does **not** validate whether MongoDB instances are spread across data centers. Failing to ensure this even distribution can compromise overall availability of the AICS deployment.

- Every data center should have similar hardware capacity (RAM, CPU, disk).

- **No data center should have a majority of the MongoDB servers running in it when using three data centers.**

Using Only Two Data Centers

You can use only two data centers when installing AICS in HA mode, but this reduces overall availability of AICS. In this scenario, one data center always has the majority of the MongoDB servers running in it. If that data center fails, the second data center goes into read-only mode. You must then execute a manual recovery action, using the following procedure:

Execute Manual Recovery

To recover if your system enters read-only mode:

1. Find the current status of the MongoDB cluster by entering the following command:

   ```
   docker exec -it $(docker ps -qf label=com.docker.swarm.service.name=mongo_mongo3)
   mongo --ssl --sslCAFile /etc/ssl/mongodb.pem --sslAllowInvalidHostnames --eval "for
   (i=0; i<rs.status().members.length; i++) { member = rs.status().members[i];
   print(member.name + \" : \" + member.stateStr) }"
   ```

   For example, you might enter:

   ```
   [pm@hostname ha-scripts]$ docker exec -it $(docker ps -qf
   label=com.docker.swarm.service.name=mongo_mongo3) mongo --ssl --sslCAFile /etc/ssl/
   mongodb.pem --sslAllowInvalidHostnames --eval "for (i=0;
   i<rs.status().members.length; i++)="" {="" member="rs.status().members[i];"
   print(member.name="" +="" \"="" :="" member.statestr)="" }"<="" tt="">
   ```

   And receive back the following:

   ```
   MongoDB shell version: 3.2.18

   connecting to: test

   mongo_mongo1:27017 : SECONDARY

   mongo_mongo2:27017 : SECONDARY

   mongo_mongo3:27017 : PRIMARY

   [pm@node-3 ha-scripts]$
   ```

   The primary MongoDB node is `mongo_mongo3`. The following command shows the number of members in the MongoDB cluster:

   ```
   rs.status().members.length;
   ```

2. Remove any unreachable MongoDB members. If necessary, use the following command to change to the primary node:

   ```
   com.docker.swarm.service.name=mongo_mongo3
   ```

3. Run the following command on the primary MongoDB node to recover the MongoDB cluster:

   ```
   docker exec -it $(docker ps -qf label=com.docker.swarm.service.name=mongo_mongo3)
   mongo --ssl --sslCAFile /etc/ssl/mongodb.pem --sslAllowInvalidHostnames --eval
   "members = rs.status().members; cfgmembers = rs.conf().members; for
   (i=members.length; i>0; i--) { j = i - 1; if (members[j].health == 0) {
   ```

```
cfgmembers.splice(j,1) } }; cfg = rs.conf(); cfg.members = cfgmembers;
printjson(rs.reconfig(cfg, {force: 1}))"
```

For example, you might enter:

```
[pm@hostname ha-scripts]$ docker exec -it $(docker ps -qf
label=com.docker.swarm.service.name=mongo_mongo3) mongo --ssl --sslCAFile /etc/ssl/
mongodb.pem --sslAllowInvalidHostnames --eval "members = rs.status().members;
cfgmembers = rs.conf().members; for (i=members.length; i>0; i--) { j = i - 1; if
(members[j].health == 0) { cfgmembers.splice(j,1) } }; cfg = rs.conf(); cfg.members
= cfgmembers; printjson(rs.reconfig(cfg, {force: 1}))"
```

And receive back the following:

```
MongoDB shell version: 3.2.18

connecting to: test

{ "ok" : 1 }

[pm@node-3 ha-scripts]$
```

The minority members in the reachable data center can now form a quorum, which returns the running data center to read-write mode.

For other useful commands, including commands for checking node status and removing non-functional nodes, see Troubleshooting Your HA AICS Deployment, below.

Set Values for Environment Variables

This section lists environment variables that must or should be configured for optimal GPR performance, and the recommended values. Adjust these values as necessary, based on your specific environment.

> ### Warning
> The **tango.env** file, which contains the environment variables, is overwritten when you perform a software upgrade. Before upgrading, save a copy of the **tango.env** file and refer to it to reset your variables. Note that if you simply overwrite the new **tango.env** file with your existing one, any environment variables added in the new release are removed.

Environment variables are defined in the **IP_JOP_PRR_<*version_number*>_ENU_linux/conf/tango.env** file. The same file is used for both single node and HA deployments.

To add a new variable:

1. Create a new line in the **tango.env** file.
2. Add the variable and its value, using the following format:
   <NEW_ENV_VAR>=value

> **Important**
>
> - Do not use quotes for string parameters.
> - Remove trailing spaces.

Changes take effect on restart of the tango container (run the `bash scripts/restart.sh` command command). In an HA environment, with multiple instances of the containers running, restart is performed sequentially (a rolling restart), so that there is no downtime of the GPR application.

Configurable Environment Variables

- **ADD_CARDINALITIES_EVERY_N_RECORDS** - When you append data to an Agent or Customer Profile via the API, cardinalities are computed only for the appended data portion and only when the number of agents or customers set in the ADD_CARDINALITIES_EVERY_N_RECORDS parameter is reached. The results of computation are added to the already-stored cardinality values. This significantly improves speed when loading new data by avoiding simultaneous recomputations on the full data collection when there are multiple frequent appends done in small batches.enables you to specify how many appended records are added to an Agent or Customer Profile before GPR recalculates cardinalities. The default value is `1000`.
  - Notes:
    - This functionality is available only when you use the Predictive Routing API. If you append using the Predictive Routing application interface, all cardinalities are recalculated.
    - Full automatic computation happens only once, when an Agent or Customer Profile is uploaded the first time for schema discovery.
    - You can force recomputation of cardinalities on the full Agent or Customer Profiles collection using the POST **compute_cardinalities** API endpoint. For details, see the *Predictive Routing API Reference*. (This file requires a password to open it. Contact your Genesys representative if you need access.)
- **AUTOGENERATE_INDEXES** - Instructs GPR to create indexes on all Datasets, Agent Profile schemas, and Customer Profile schemas. By default, set to `True`.

  > **Important**
  >
  > Genesys strongly recommends you to leave the default value for this variable.

- **HOST_DOMAIN** - Use this variable to specify the public IP address or host name used for your deployment. The value should be one of the following, depending on your environment type:
  - For single-server deployments, specify the public IP address or the host name of the host where GPR is deployed.
  - For high availability (HA) deployments, specify the IP address of your load balancer.
- **LOG_LEVEL**
  - **INFO** - Informational messages that highlight the progress of the application: **LOG_LEVEL**=`INFO`.

This setting is recommended for production deployments.

- **DEBUG** - Fine-grained informational events that are most useful to debug the application: **LOG_LEVEL**=DEBUG. This setting should be used only for short periods of time because it can fill the disk.

- **LOGIN_MESSAGES** enables you have the Predictive Routing application display a custom message on the login screen.
  - When you enter this message, make sure that all special characters are properly escaped. *Special characters* are ones not part of the standard English alphabet, such as symbols, letters with umlauts, cedillas, and other such marks, and letters from other alphabets, such as the Greek or Cyrillic alphabets.
  - To simplify the task of converting characters, Genesys recommends an online conversion tool, such as **https://www.freeformatter.com/html-escape.html**.
  - For example, make the following substitutions:
    - & becomes &
    - < becomes <
    - > becomes >
    - " becomes "
    - ' becomes '
- **OMP_NUM_THREADS** (required for releases prior to 9.0.011.00; in releases 9.0.011.00 and higher, this parameter is set automatically)
  - Genesys recommends that you set the value to OMP_NUM_THREADS=1 for the best performance.
  - If you do not specify a value, GPR spawns one thread for each core it detects in your environment. The system assumes it can use all available cores for tasks such as analysis and model training, leaving no CPU resources for other processes running on the same machine, such as reading/writing to the database. The result is an overall slowdown of the application. Set this variable to allow the operating system to properly distribute CPU threads among the various running processes.
- **S3_ENDPOINT** - (Mandatory) The endpoint for the Minio container, introduced in AICS release 9.0.013.01 for Dataset uploads and expanded to Agent and Customer Profile uploads in AICS 9.0.015.03. Specifies the public IP address or domain name of the server where AI Core Services is installed, followed, optionally, by the port number.
  - The port number must always be 9000, which is the mandatory port value allocated for the Minio container.
  - In HA environments, locate the server on which the Minio container is running and use the public IP address or the domain name of that server. For example:

    For an IP address - `S3_ENDPOINT=https://<public_ip_address>:9000`

    For a domain name - `S3_ENDPOINT=https://<your_domain_name>:9000`
  - The S3_ENDPOINT value must always use the HTTPS protocol. If you do not configure this variable, the **start.sh** script generates a warning message and stops deploying AICS.
- (Optional) **GUNICORN_TIMEOUT**
  - Adjust the timeout if you need to accommodate a large Dataset. The current default value is `600` seconds.

Load Balancing and HTTPS Configuration for HA AICS

Once AICS has been installed and started, you can access it using the IP address of any node in the cluster on port 3031. To enable load balancing:

1. Your load balancer should have its health-check functionality turned on.
2. The load balancer should check for HTTP code `200` to be returned on `https://IP:3031/login`.

> **Important**
>
> - Genesys recommends a third-party highly-available load balancer, such as F5, to ensure all requests to AICS platform are spread evenly across all nodes in the AICS cluster.
> - If you need SSL, set it up on the third-party load balancer.
> - If you are using a domain name instead of a numeric IP address, configure the S3_ENDPOINT environment variable in the **tango.env** file as follows: **S3_ENDPOINT**= `https://<your_domain_name>:9000`

Configure HTTPS in an HA Environment

In a high availability (HA) environment, follow the instructions provided in the documentation for your load balancer. In an HA environment, you do not need to deploy the certificates on the individual nodes.

Unusual HTTP/S Deployment Scenarios

- Default local certificate - Genesys ships AICS with a default local certificate. You can use that local certificate to access the GPR web application for internal testing purposes. Note the following points when using the local certificate:
  - The browser displays a Not Secure connection warning.
  - You **cannot** use the default certificate to configure connections from Agent State Connector or the Subroutines components to AICS.
- Self-signed certificate - *For lab environments only* - You can use OpenSSL to generate a self-signed certificate. You can use this self-signed certificate to configure secure connection between AICS and the other GPR components, as explained in the instructions for configuring HTTPS for ASC and the URS Strategy Subroutines. Generate a self-signed certificate by executing a command following the format in the following example:

```
$ openssl req -new -newkey rsa:4096 -days 365 -nodes -x509 -subj "/C=US/ST=US/L=US/
O=IT/OU=IT Department/CN=<ip address of the server where GPR is deployed>" -keyout
tango.key -out tango.crt
```

- HTTP connections in test environments - In AICS release 9.0.015.04 and higher, you can optionally configure HTTP connections.

> **Warning**
> HTTP connections are supported only in test environments. Genesys strongly recommends using the default HTTPS configuration in production environments and in lab environments that contain sensitive data. Genesys is not responsible for any potential damage and/or data loss if the solution is implemented without the recommended security practices and protocols.

To use HTTP connections, perform the following steps:

1. Comment out the following lines in the **ha-scripts/swarm/tango-swarm.yml** file *on every node*:

   ```
   #       - ../../conf/tango.key:/data/ssl/tango.key
   #       - ../../conf/tango.crt:/data/ssl/tango.crt
   ```

2. Save your changes.
3. Restart GPR by running the following command on any node:

   ```
   $ bash ha-scripts/restart.sh
   ```

   You do not need to restart each node separately. Running the restart command on one node restarts the entire system.

4. On the load balancer in front of your Docker Swarm cluster, change *https* to *http* in your load balancer configuration. For example, make a change similar to the following:
   Change

   https://your_node_ip:3031

   to

   http://your_node_ip:3031

Using the NGINX Load Balancer

> **Important**
> The NGINX container was removed from AICS in release 9.0.013.01.

In releases through 9.0.012.01, Genesys shipped the NGINX load balancer as part of AICS. *It is intended for use only in prototype scenarios.*

> **Important**
> The NGINX load balancer is a single point of failure and should not be used in production deployments.

To use the NGINX, follow the procedure below:

1. Edit the `ha-scripts/nginx/nginx.conf` file by putting the IP addresses of all nodes in your cluster into the `upstream tango` section using syntax such as IP1:3031, IP2:3031, IP3:3031. For example, your command might look similar to the following:

   ```
   upstream tango {
        server 18.220.11.120:3031;
        server 18.216.235.201:3031;
        server 13.59.93.192:3031;
        }
   ```

2. Execute the following command in order to start the NGINX container:

   ```
   bash ha-scripts/nginx/start.sh
   ```

3. Verify that you can access AICS by pointing your browser to IP address where NGINX is running.

To stop NGINX, run the following command:

```
bash ha-scripts/nginx/stop.sh
```

To fix a `413 (Request Entity Too Large)` NGINX error, follow these steps:

1. Open the **nginx.conf** file.
2. Increase the value for the **client_max_body_size** parameter to 3g.
3. Restart NGINX using the command:

```
docker restart nginx
```

Required Ports for AICS Servers

The following ports are those required for communication between all target servers in the cluster. Note that some ports are specific to high availability (HA) environments (such as the Docker swarm port), while others apply to all deployments.

Component	Protocol	Port Number	Type	Description
Docker	TCP	2377	Inbound/Outbound	Cluster management communications
Docker swarm	TCP/UDP	7946	Inbound/Outbound	Required for Docker Swarm for communication among nodes
Docker swarm	UDP	4789	Outbound/Inbound	For overlay network traffic
MongoDB	TCP	27017	Inbound/Outbound	Default port for MongoDB
Tango container	TCP	3031	Inbound	Required to access the Predictive Routing API and Predictive Routing

Component	Protocol	Port Number	Type	Description
				web application
MinIO container	TCP	32646	Inbound/Outbound	Default port for MinIO
SSH	TCP	22	Inbound/Outbound	Required to access all target servers using SSH

To open a port, use the following syntax:

```
firewall-cmd --zone=public --add-port=<''port_number''>/<''protocol''> --permanent
```

Important

If you are running VMWare VXLAN, you might encounter a port conflict between VMWare VXLAN and Docker, both of which require port 4789. If you encounter this issue, Genesys recommends that you use a networking application such as Weave Net to manage networking among Docker containers. For additional information, consult the documentation for the respective products:

- For the Docker Swarm port requirements: Use swarm mode routing mesh
- For VMWare VXLAN port requirements: Ports and Protocols Required by NSX
- For Weave Net: Introducing Weave Net

Clean Up Disk Space

Starting in release 9.0.013.01, GPR performs automatic cleanup processes which should maintain an adequate amount of free disk space. However, if you are running an earlier version of AICS, or are running 9.0.013.01 or higher and continue to encounter disk space problems, refer to the instructions in this section.

You might encounter performance issues if you do not clean up Docker data that is no longer required. The Docker `prune` command enables you to clean up your Docker environment. The Docker user documentation provides a detailed discussion of the `prune` command and how to use it to clean up images, containers, and volumes; see Prune unused Docker objects.

Important

The clean-up process does not affect normal GPR operation. It does not require downtime, there is no need to restart any component, and performance is unaffected.

Clean-Up Procedure

Genesys recommends that you use the following commands to remove unnecessary Docker data:

```
docker container prune -f
docker volume prune -f
docker network prune -f
```

To schedule regular cleanup jobs, use the `crontab` functionality to execute the appropriate command on every server where GPR is installed. The following example schedules the cleanup job for every Saturday at 1:00 am:

```
echo "0 1 * * Sat (docker container prune -f; docker volume prune -f; docker network prune
-f)" ) | crontab -
```

In an HA environment, Genesys recommends that you perform the cleanup on each node in turn.

If you need to configure your logging settings to avoid unacceptable log file sizes, see the following information:

- The LOG_LEVEL environment variable
- Configure AICS Log Settings

Installing into an Existing HA AICS Deployment

It is easy to install a different version of AICS on your target servers. Use the standard procedure described below to install either a newer or an older version of AICS. For some releases, listed below, you must also run additional scripts to complete the upgrade.

Special Upgrade Procedures

Some releases require special upgrade scripts or procedures. These procedures appear in the Upgrade Notes section of the RN for that release.

- Review the Upgrade Notes section of the Release Notes for *all* releases later than your starting release, including your target release.
- If special upgrade scripts are required for any releases between your current version and your target version, they are all included in the IP for your target version.
- Follow any procedures specified for the interim releases, such as running scripts.
- If there is no Upgrade Notes section, or the section is empty, no additional steps are required for the associated release.
- The following AICS releases *do* require special upgrade procedures:
  - 9.0.007.00
  - 9.0.007.01
  - 9.0.007.03

- 9.0.011.00
- 9.0.013.01
- 9.0.014.00
- 9.0.014.02

Standard Upgrade Procedure

Follow the steps in this section to perform the standard upgrade:

> **Important**
>
> The standard process described below requires downtime, but does not result in loss of data. The upgrade script updates only the various services running in the Tango container and the Workers containers. It does not stop or upgrade MongoDB.

1. If you edited your **tango.env** file, save a copy of it in a separate location. Also save any other files you might have customized, such as the **docker-compose.yml** file and the contents of your **/datadir** directory.
2. Copy the new AICS release package (the ***.tar.gz** file) to all servers in the cluster. Use the same user and procedure as if you are installing AICS for the first time. All the recommendations about the user who performs the installation and operates AICS still apply.
3. After unpacking the new version of AICS in the PR_USER home directory that contains *all* target servers, you will have multiple different subdirectories named **IP_JOP_PRR_<*version_number*>_ENU_linux**. For example you might have two subdirectories:
   - **IP_JOP_PRR_<*old_version_number*>_ENU_linux**
   - **IP_JOP_PRR_<*new_version_number*>_ENU_linux**
4. Assuming you are installing *new_version* of the application and removing *old_version*, execute the following command in the **IP_JOP_PRR_<*new_version_number*>_ENU_linux** directory on *all* target servers:

   ```
   bash ha-scripts/install.sh
   ```

5. Then in any one of the servers, execute the following command in the **IP_JOP_PRR_<*new_version_number*>_ENU_linux** directory:

   ```
   bash ha-scripts/upgrade_gpr_services.sh
   ```

   This command executes the upgrade of Tango (AICS) on all nodes in the cluster, one by one, and rolls back the change if there is a problem. There is no downtime during this upgrade, and no data loss.
6. To restore your custom environment values, paste the copy you made of your previous **tango.env** file over the new one, as well as any other files you might have customized, such as the **docker-compose.yml** file and the contents of your **/datadir** directory.

Your upgrade should now be complete.

Troubleshooting a AICS HA Deployment

The following sections offer information that can help you identify issues without your deployment.

Handling Server Failure

If a server (node) restarts, the HA deployment recovers automatically as long as the server keeps its previous IP address and the data on the disk is not corrupted.

The following command identifies a non-working node as **unreachable** node:

```
docker node ls
```

If a server needs to be decommissioned and replaced with new one, the following manual step is necessary to preserve the health of the cluster. After shutting down the server that is to be decommissioned, execute the following two commands, where **NODE_ID** is the unique node identifier of the server to be decommissioned:

```
docker node demote <NODE_ID>
docker node rm <NODE_ID>
```

After this, you can add a new server to your environment. Label it the same way as the decommissioned server and execute the procedure for joining that server to the cluster as described in Installation Procedure, above.

Handling Failover

When a server hosting MongoDB and the AICS application (the Tango container) experiences a failover, a certain number of API requests to AICS might fail during the few seconds it takes for the system to recover. The routing strategy attempts to resend any failed request, but Agent State Connector (ASC) does not have this capability. As a result, there is a risk of a small data loss.

Note that error messages appear in the logs for both MongoDB and the AICS application when a failover occurs.

Health Checks for Your Deployment

To check the health of your Predictive Routing HA deployment, perform the following steps:

1. Verify that all nodes are up and running. On any node in the cluster, execute the following command:

   ```
   docker node ls
   ```

   You should receive output similar to the following:

   ```
   [pm@hostname ~]$ docker node ls
   ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS
   mc0bgyueb3c0h9drsy3j0i2ty node-1-hostname Ready Active Leader
   ```

```
vm1csljly66vwguxzaz8ly98r *node-2-hostname Ready Active Reachable

z2vlnldcyh0y57jwns0bz9jxe node-3-hostname Ready Active Reachable
```

All nodes should be reachable.

2. Check that all services are running by executing the following command on any node in the cluster:

```
docker service ls
```

You should receive output similar to the following:

```
[pm@hostname ~]$ docker service ls

ID NAME MODE REPLICAS IMAGE PORTS

jzjitn8lp78t mongo_mongo1 replicated 1/1 mongo:3.2

iqntp5eabfnw mongo_mongo2 replicated 1/1 mongo:3.2

whw05twosi9s mongo_mongo3 replicated 1/1 mongo:3.2

1jp3sgt16czw tango_tango global 3/3 jop_tango:2017_12_12_15_17

hu3kvkzxn88r workers_workers replicated 2/2 jop_tango:2017_12_12_15_17
```

- The important column here is REPLICAS.
- The Tango service should always be global and reachable on port 3031 on every node in cluster.
- The MongoDB service is replicated. The value 1/1 in the REPLICAS column for MongoDB indicates that a replica exists for each instance. See Checking the Health of MongoDB (below) for how to check health of MongoDB database.
- The Workers service is replicated and should show as many replicas as there are nodes labeled with the Workers label. See Label the Worker Nodes in the Cluster (above) for how to label nodes.

Checking the Health of MongoDB

All the commands listed below should show your MongoDB cluster with one PRIMARY instance and all other instances should be healthy SECONDARY instances.

- To check the health of the MongoDB cluster while logged into node with hostname *node-1-hostname* execute following command on *node-1-hostname*:

```
[pm@node-1-hostname ~]$ docker exec -it $(docker ps -qf
label=com.docker.swarm.service.name=mongo_mongo1) mongo --ssl --sslCAFile /etc/ssl/
mongodb.pem --sslAllowInvalidHostnames --eval 'rs.status()'
```

- To check the health of MongoDB cluster while logged into node with hostname *node-2-hostname* execute following command on *node-2-hostname*:

```
[pm@node-2-hostname ~]$ docker exec -it $(docker ps -qf
```

```
label=com.docker.swarm.service.name=mongo_mongo2) mongo --ssl --sslCAFile /etc/ssl/
mongodb.pem --sslAllowInvalidHostnames --eval 'rs.status()'
```

- To check the health of MongoDB cluster while logged into node with hostname *node-3-hostname* execute following command on *node-3-hostname*:

  ```
  [pm@node-3-hostname ~]$ docker exec -it $(docker ps -qf
  label=com.docker.swarm.service.name=mongo_mongo3) mongo --ssl --sslCAFile /etc/ssl/
  mongodb.pem --sslAllowInvalidHostnames --eval 'rs.status()'
  ```

  Similarly, you can check the health of MongoDB cluster from any other node where a MongoDB replica is running.

- For example, if you send the following commands:

  ```
  docker exec -it $(docker ps -qf name=mongo_mongo) mongo --ssl --sslCAFile /etc/ssl/
  mongodb.pem --sslAllowInvalidHostnames --eval "rs.status()" | grep "stateStr"
  ```

  You should receive a response similar to the following:

  ```
  "stateStr" : "SECONDARY",
  "stateStr" : "SECONDARY",
  "stateStr" : "PRIMARY",
  "stateStr" : "SECONDARY",
  "stateStr" : "SECONDARY",
  [pm@node-1 IP_JOP_PRR_gpr_rc_ENU_linux]$
  ```

Other Useful Commands

Here are few more useful commands to troubleshoot MongoDB:

To find out the status of all members in the replica set, use the following command:

```
docker exec -it $(docker ps -qf label=com.docker.swarm.service.name=mongo_mongo3) mongo --ssl
--sslCAFile /etc/ssl/mongodb.pem --sslAllowInvalidHostnames --eval "rs.status().members"
```

To remove an unreachable member, execute the following command (this has to be repeated for each unreachable member in a failed data center):

```
docker exec -it $(docker ps -qf label=com.docker.swarm.service.name=mongo_mongo3) mongo --ssl
--sslCAFile /etc/ssl/mongodb.pem --sslAllowInvalidHostnames --eval 'rs.remove("HOST:PORT")'
```

(Optional) Backing Up Your Data

This section applies specifically to backing up and restoring in an HA environment. For instructions to back up and restore MongoDB in a single-site/single-server AICS deployment, see Backing Up and Restoring Your Data.

Although HA greatly reduces the likelihood of data loss, Genesys recommends that you back up your data to safeguard it. This section explains how to back up and restore your data in an HA environment.

> **Important**
>
> All MongoDB backup and restore operations should be performed on the PRIMARY MongoDB instance.

Using SSL with MongoDB

The procedure below is for MongoDB with SSL enabled. *Genesys recommends that you use SSL.*

- To use SSL, add the `--ssl` parameter to your commands.
  In test environments, you can optionally add `--sslAllowInvalidCertificates` following the `--ssl` parameter.

  In test environments ONLY, if you need to maintain an environment without SSL connections, omit the `--ssl` and `--sslAllowInvalidCertificates` parameters.

Backing Up

On every server where MongoDB is running, there is one important directory:

- The `/data/db` directory in every MongoDB container is mapped to the `/datadir` directory on the server file system.

Use the `mongodump` command from inside the container to back up your MongoDB data, using the following command:

```
mongodump --ssl --out /data/db/`date +"%m-%d-%Y"`
```

This command backs up all databases in the `/data/db/<date +"%m-%d-%Y">` directory located in the container. For example, you might back up the `/data/db/08-18-2019` directory.

The backed-up data is located in the `/datadir/<date +"%m-%d-%Y">` directory on the server host computer. For the example backup command above, the output would be located in the `/datadir/08-18-2019` directory.

Restoring

In order to restore data you must first make data files available in the appropriate directory on the server host computer.

Use the following command inside of the container:

```
mongorestore --ssl --drop /data/db/''PATH_TO_SPECIFIC_BACKUP_DIRECTORY''
```

For example, you might run the command:

```
mongorestore --ssl --drop /data/db/08-18-2019
```

For extra information about backing up MongoDB and data preservation strategies, see the following topic on the MongoDB site: https://docs.mongodb.com/manual/core/backups/.

(Optional) Installing AICS on a Kubernetes Cluster

The following instructions provide optional deployment and configuration procedures that guide you through setting up AI Core Services (AICS) on Kubernetes. These instructions assume that you have already installed Kubernetes in your environment. Genesys supplies scripts to orchestrate AICS on the Kubernetes cluster.

For information about Kubernetes and how to deploy it, see the Kubernetes web site.

System and Architecture Requirements

In addition to the standard set of system requirements, the following apply specifically to AICS running on a Kubernetes cluster:

- Kubernetes version 1.10 or higher.
- Helm installed on the master node of the Kubernetes cluster.
- At least four nodes comprising the cluster, with the following roles:
  - 1 *master* node with minimum of 4 CPUs and 8 GB RAM.
  - 3 *worker* nodes each with a minimum of 8 CPUs and 16 GB RAM.
- AICS has specific CPU and RAM requirements depending on your environment. Use the Sizing Worksheet to determine what hardware resources you need for your environment. When running AICS on a Kubernetes cluster, keep in mind that Kubernetes also has CPU and RAM requirements.
- Kubernetes configured to use either local storage or Dynamic Volume Provisioning on all nodes, including the *master* node.

Installing AICS on the Kubernetes Cluster

To install AICS on Kubernetes, perform the following procedures:

1. Guided by the instructions in Installation Procedure (above), upload and unpack the **IP_JOP_PRR_<*version_number*>_ENU_linux.tar.gz** installation binary file to all Kubernetes nodes. When you unpack this file, it creates the **IP_JOP_PRR_<*version_number*>_ENU_linux** directory.
2. On the host that is the master node of the Kubernetes cluster, navigate to the **IP_JOP_PRR_<*version_number*>_ENU_linux** directory.
3. Locate and modify the following two files to include the version of AICS you are deploying. Do this by replacing `jop_tango:CHANGE_ME` in each of the two files with `jop_tango:$VERSION`, which can be found in the **jop.version** file in the same folder.
   - **helm/gpr/values.yml**
   - **helm/worker/values.yml**

     The AICS component was previously known as JOP. These files retain the old naming convention.
4. On each Kubernetes *worker* node (these are the Kubernetes *worker* nodes, not to be confused with AICS worker containers), navigate to the **IP_JOP_PRR_<*version_number*>_ENU_linux** directory and then run the following command to load the AICS Docker images:

```
bash ha-scripts/install.sh -s
```

Labeling Nodes

Run the followng commands from the Kubernetes cluster *master* node.

1. Check the existing labels on Kubernetes nodes:

   ```
   sudo kubectl get nodes --show-labels
   ```

2. For every node where you want an AICS worker container to run, execute:

   ```
   sudo kubectl label nodes <node-name> worker=true
   ```

3. For every node where you want a Tango container to run, execute:

   ```
   sudo kubectl label nodes <node-name> gpr-apps=true
   ```

Deploying AICS Using Helm with Local Storage

To use local storage for the MinIO and MongoDB data store, execute the following script on master node to deploy AICS using Helm charts. This deploys the Tango container and the various AICS worker containers:

```
bash kubernetes/deploy-gpr-services.sh
```

Deploying AICS using Helm with Dynamic Volume Provisioning

The following optional command can be used to provide details for the storage class and the type of provisioner. When you run the command, replace <provisioner_name> with an actual name of a supported provisioner, such as glusterfs.

```
bash kubernetes/deploy-gpr-services.sh -s <provisioner_name> -m mongodb-ssd -c minio-ssd
```

This ensures that the appropriate storage class is selected when making a persistent volume claim.

Upgrading

Follow the steps below to upgrade the Tango container and the Workers containers.

Upgrading the Tango Container Using Helm

1. Download the new AICS IP, unpack it, and navigate to the **IP_JOP_PRR_<*version_number*>_ENU_linux** directory for the *new* IP.
2. Manually modify the following two files to include the version of AICS you are deploying. Do this by replacing `jop_tango:CHANGE_ME` in each of the two files with `jop_tango:$VERSION`, which can be found in the **jop.version** file in the folder for the *new* IP.
   - **helm/gpr/values.yml**
   - **helm/worker/values.yml**
3. On each Kubernetes *master* and *worker* node, navigate to the **IP_JOP_PRR_<*version_number*>_ENU_linux** directory for the *new* version and then run the following command to load the new AICS Docker images:

```
bash ha-scripts/install.sh -s
```

4. On the master node, run the following command:

```
helm upgrade --install gpr -f ./helm/gpr/values.yaml ./helm/gpr
```

- To check the upgrade history of the AICS Tango container, run the following command:

```
helm history gpr
```

- To roll back the upgrade of the AICS Tango container, run the following command:

```
helm rollback gpr <revision_number>
```

Upgrading AICS Worker Containers Using Helm

To upgrade the Worker containers, run the following commands:

```
helm upgrade --install worker-analysis --set workerDeploymentName=worker-
analysis,workerTopic=analysis -f /helm/worker/values.yaml /helm/worker
 helm upgrade --install worker-dataset-upload --set workerDeploymentName=worker-dataset-
upload,workerTopic=dataset_upload -f /helm/worker/values.yaml /helm/worker
 helm upgrade --install worker-model-training --set workerDeploymentName=worker-model-
training,workerTopic=model_training -f /helm/worker/values.yaml /helm/worker
 helm upgrade --install worker-purging --set workerDeploymentName=worker-
purging,workerTopic=purging -f /helm/worker/values.yaml /helm/worker
```

To list the containers deployed using Helm, run the following command:

```
helm ls
```

To check the upgrade history of the AICS Worker containers, run the following command:

```
helm history <worker_name>
```

To roll back the upgrade of the AICS Worker containers, run the following command:

```
helm rollback <worker_name> <revision_name>
```

(Optional) Mapping a Local Volume into a Container

Local directories or files can be mapped on any of the containers user by the application in an HA deployment: tango, workers. or mongo.

> **Tip**
> An HA deployment in Production mode should not use NGINX.

To mount a volume, update the file corresponding to the desired container to a local directory or file by editing the volumes declaration:

- tango: <IP_JOP_PRR_<*version_number*>_ENU_linux/ha-scripts/swarm/tango-swarm.yml
- mongo: <IP_JOP_PRR_<*version_number*>_ENU_linux/ha-scripts/swarm/mongo-swarm5.yml / <IP_JOP_PRR_<version_number>_ENU_linux/ha-scripts/swarm/mongo-swarm.yml
- workers: <IP_JOP_PRR_<*version_number*>_ENU_linux/ha-scripts/swarm/worker-swarm.yml

> **Important**
>
> Mapping a directory or file on a node makes it available only on that host. It does not create or imply any type of file replication.

To mount a local directory, follow the format presented in the following example:

- To mount /some_local_directory, into /custom_mount_point in the mongo container on node-1, edit the <IP_JOP_PRR_<*version_number*>_ENU_linux/ha-scripts/mongo-swarm.yml file as follows:

```
volumes:
   - mongodata1:/data/db
   - mongoconfig1:/data/configdb
   - ../conf/mongodb.pem:/etc/ssl/mongodb.pem
   - /some_local_directory:/custom_mount_point
```

To make the changes take effect restart the application:

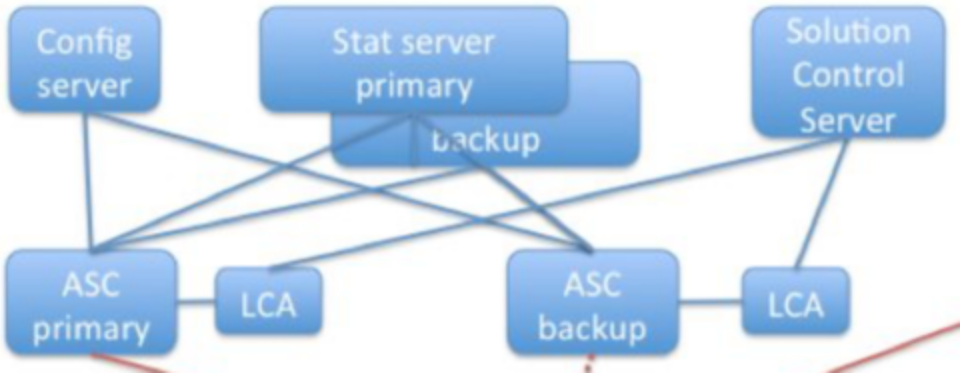
```
bash <IP_JOP_PRR_<''version_number''>_ENU_linux/ha-scripts/restart.sh
```

> **Important**
>
> Additional information can be found at https://docs.docker.com/compose/compose-file/compose-file-v2/#volumes

HA for ASC

Agent State Connector (ASC) has a standard primary-backup warm-standby high availability configuration. The backup server application remains initialized and ready to take over the operations of the primary server. It maintains connections to Configuration Server and Stat Server, but does not send agent profile updates to AICS.

To configure a primary-backup pair of ASC instances, create two ASC Application objects. Open the **Server Info** tab for the backup ASC set warm standby as the redundancy mode. When Local Control Agent (LCA) determines that the primary ASC is unavailable, it implements a changeover of the backup to primary mode.



Important

If the Stat Server instance you are using for Predictive Routing is release 8.5.100.10 or higher, you must set the value for the accept-clients-in-backup-mode configuration option in the Stat Server `Application` object to no to ensure normal backup switchover between ASC instances.