Predictive Routing Deployment and Operations Guide

Scale AI Core Services

5/7/2025

Contents

Scale AI Core Services

Correct use of scaling enables you to efficiently adapt your environment to changing conditions. The first step in increasing the size of your AI Core Services (AICS) deployment is to add new servers and configure them. You can then allocate instances of the various containers among the new servers. As with the initial deployment, use special consideration when planning how to distribute your instances of MongoDB.

> **Important**
> There is no need to shut down your AICS deployment. You can add servers while AICS is running.

Add New Servers

When GPR is deployed in high availability (HA) mode you can expand that deployment by adding additional hardware to the cluster.

To add servers to an existing cluster, follow these steps:

1. Complete the Installation steps and unpacking steps on the new servers to make them available to the AICS deployment.
2. On the node with hostname *node-1-hostname* execute the following command:

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

   The output of this command should look something like this:

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

3. Copy this command and execute it on the new servers. This adds the new nodes to your existing cluster.
4. On the new servers, execute the following command:

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

You can now scale the services to start using the new servers.

Scaling Individual GPR components

GPR consists of multiple individual components, each with different responsibilities. These components are packaged as Docker images and executed as Docker containers. Every component of GPR can be scaled vertically (configuring already-running GPR Docker containers to use more of the existing hardware resources) or horizontally (adding new hardware and then creating additional GPR containers). Each GPR component scales independently of the others.

The sections below describe each component of GPR, when to scale it, and how to scale it both vertically and horizontally.

Scaling Tango

The Tango container processes all scoring requests and exposes the GPR APIs to the external world. The single Tango container deployed per host during installation can use as many CPUs and as much RAM as you can provide, enabling you to scale vertically to whatever extent your environment requires. The Tango container does not need to scale horizontally.

- In a single-host deployment, only one Tango container is created.
- In an HA deployment, there is one Tango container per host.

Scaling MongoDB

In single-host deployments, you can have only one MongoDB instance and it can only be scaled vertically. In HA deployments, you can have 3 or 5 MongoDB instances running on separate servers and these instances can each be scaled vertically. Genesys does not recommend you to scale MongoDB beyond this point.

> **Important**
>
> Since MongoDB, which stores all your data, is a critical part of GPR, Genesys recommends that you take time to size your MongoDB containers with deliberate care from the beginning, (during installation) rather than adding resources to your existing deployment, because later changes require downtime.

Scaling MongoDB Vertically

MongoDB should be scaled vertically when you have enough hardware but the current MongoDB resource allocation is acting as a bottleneck. By default each MongoDB container can use up to 2 CPUs and up to 8GB of RAM, whether running on a single host or in an HA deployment.

Starting in MongoDB 3.2, WiredTiger is the default storage engine for MongoDB. The `wiredTigerCacheSizeGB` parameter should be configured to use 50% of the maximum memory assigned to the MongoDB container. By default, this is 4 GB.

Single-Server Environments

Scaling MongoDB vertically in a single-host deployment requires downtime.

1. Stop GPR, using the following command:

   ```
   bash stop.sh
   ```

   - To add more CPUs to the MongoDB instance, change the value of the `cpus` setting, found in the **IP_JOP_PRR_<*version_number*>_ENU_linux/scripts/docker-compose.yml** file under the `mongo` container.
   - To add more RAM to the MongoDB instance, change the value of the `mem_limit` setting, found in the **IP_JOP_PRR_<*version_number*>_ENU_linux/scripts/docker-compose.yml** file under the `mongo` container.
   - To change the amount of memory allocated to the wiredTiger storage engine for the MongoDB instance, change the value of the `wiredTigerCacheSizeGB` setting, found in the **IP_JOP_PRR_<*version_number*>_ENU_linux/scripts/docker-compose.yml** file under the `mongo` container.

2. Restart GPR, using the following command:

   ```
   bash start.sh
   ```

HA Environments

Scaling MongoDB vertically in an HA deployment requires downtime.

1. Stop GPR, using the following command:

   ```
   bash stop.sh
   ```

   - To add more CPUs to the MongoDB instances, change the value of the `cpus` setting, found in the **IP_JOP_PRR_<version_number>_ENU_linux/ha-scripts/swarm/mongo-swarm.yml** file under the `mongo1`, `mongo2`, `mongo3` containers and, if you are using five containers, the `mongo4` and `mongo5` containers.
   - To add more RAM to the MongoDB instances, change the value of the `memory` setting, found in the **IP_JOP_PRR_<version_number>_ENU_linux/ha-scripts/swarm/mongo-swarm.yml** file under the `mongo1`, `mongo2`, `mongo3` containers and, if you are using five containers, the `mongo4` and `mongo5` containers.
   - To change the amount of memory allocated to the wiredTiger storage engine for each MongoDB instance, change the value of the `wiredTigerCacheSizeGB` setting, found in the **IP_JOP_PRR_<version_number>_ENU_linux/ha-scripts/swarm/mongo-swarm.yml** file under the `mongo1`, `mongo2`, `mongo3` containers and, if you are using five containers, the `mongo4` and `mongo5` containers.

2. Restart GPR, using the following command:

   ```
   bash start.sh
   ```

Scaling MongoDB Horizontally

You can only have three or five MongoDB instances in an HA deployment. You can scale MongoDB

horizontally only if you currently have three MongoDB instances in the replica set.

1. If you have not already done so, complete the steps in Adding New Servers (above) to provision additional hardware capacity.
2. Label the newly created servers so that they can run additional MongoDB instances. Use the procedure given in Label MongoDB Nodes in the Cluster (above).
3. Restart GPR, using the following command:

```
bash start.sh
```

Scaling MinIO

In release 9.0.013.01 and higher, MinIO is used for dataset uploads. By default, GPR creates one MinIO container for each host, each using two CPUs. There is no need to scale MinIO either vertically or horizontally.

Scaling Workers

GPR includes a number of different containers that run workers, each of which performs a different task in an asynchronous manner. As a result, you can scale the workers of the different types as needed in your environment.

> **Important**
> The exact number and types of containers vary depending on your release. In earlier releases, more of the functionality is performed in the Tango container; in later releases, some of these functions are split into separate containers.

Model Training Workers

Model training workers are responsible for executing model-training jobs.

Scaling Vertically

Model-training workers can be scaled vertically to reduce the time required for individual model-training jobs.

The model-training algorithms cannot use more than four CPUs, which is the default value. If you choose to allocate fewer CPUs, make the changes in the following location (this applies to both single-server and HA deployments):

- The `cpus` setting in the `model_training` section of the **IP_JOP_PRR_<*version_number*>_ENU_linux/scripts/docker-compose.yml** file.

Adding extra RAM to the model-training jobs can speed up the execution of individual jobs. The more RAM you allocate to each model-training container, the faster the execution. Model-training containers use as much RAM as is available on the server where they are deployed.

Scaling Horizontally

Adding more model-training containers ensures that more models can be trained in parallel. Jobs that can not be executed immediately are queued for execution. You can scale model-training containers horizontally, in either HA or single-host deployments, assuming you have enough hardware for the newly created containers.

By default, GPR creates one model-training container that uses up to four CPUs and as much RAM as is available. With this configuration, you can perform only one model-training job at a time. All other jobs are queued.

To scale model-training jobs horizontally, perform the following steps:

1. Stop GPR using the following command:

   ```
   bash stop.sh
   ```

   - For a single-server environment, change the value of the `NUM_OF_MODEL_TRAINING_WORKERS_INSTANCES` configuration variable, found at the beginning of the **IP_JOP_PRR_<*version_number*>_ENU_linux/scripts/start.sh** script.
   - For an HA environment, change the value of the `NUM_OF_MODEL_TRAINING_WORKERS_INSTANCES` configuration variable, found at the beginning of the `IP_JOP_PRR_<version_number>_ENU_linux/ha-scripts/start.sh` script.

2. Restart GPR using the following command:

   ```
   bash start.sh
   ```

Dataset Upload Workers

In releases prior to 9.0.013.01, the dataset-upload workers are responsible for uploading Datasets as well as loading data to MongoDB. (In release 9.0.013.01 and higher, the initial upload is performed by MinIO.)

Scaling Vertically

Dataset-upload workers can be scaled vertically to reduce the time required for individual Dataset upload jobs. The Dataset-upload containers use as many CPUs as you allocate. By default, this value is set to two CPUs.

To change this number, edit the `cpus` setting in the `dataset_upload` section of the appropriate file for your environment:

- For single-server deployments: **IP_JOP_PRR_<*version_number*>_ENU_linux/scripts/docker-compose.yml**
- For HA deployments: **IP_JOP_PRR_<*version_number*>_ENU_linux/ha-scripts/swarm/worker-swarm.yml**

Adding extra RAM to the dataset-upload jobs can speed up the execution of individual jobs. The more RAM you allocate to each dataset-upload container, the faster the execution. Dataset-upload containers can use as much RAM as is available on the server where they are deployed.

Scaling Horizontally

Adding more dataset-upload containers ensures that more Datasets can be uploaded in parallel. Jobs that cannot be executed immediately are queued for execution. You can scale dataset-upload containers horizontally regardless of whether GPR is deployed in an HA or a single-server environment. The only requirement is that there must be enough hardware for the newly created containers.

By default, GPR creates one dataset-upload worker for each container, each using up to two CPUs and as much RAM as is available. With this configuration, you can upload only one dataset in parallel. All other jobs are queued.

In order to scale dataset-upload jobs horizontally, perform the following steps:

1. Stop GPR using the following command:

   ```
   bash stop.sh
   ```

   - For a single-server environment, change the value of the `NUM_OF_DATASET_UPLOAD_WORKERS_INSTANCES` configuration variable, found at the beginning of the **IP_JOP_PRR_<*version_number*>_ENU_linux/scripts/start.sh** script.
   - For an HA environment, change the value of the `NUM_OF_DATASET_UPLOAD_WORKERS_INSTANCES` configuration variable, found at the beginning of the `IP_JOP_PRR_<version_number>_ENU_linux/ha-scripts/start.sh` script.

2. Restart GPR using the following command:

   ```
   bash start.sh
   ```

Analysis Workers

Analysis containers are used for various analysis jobs, supporting creation of the Lift Estimation report and the Feature Analysis report, among others. By default, GPR creates one analysis container for a single-host deployment and two analysis containers for an HA deployment. Each container uses up to two CPUs (the default setting) and as much RAM as is available.

In a single-host deployment, by default, you have only one container, which means you can only run one analysis job at a time. All other analysis jobs are queued. To run analysis jobs in parallel, perform the following steps:

1. Stop GPR using the following command:

   ```
   bash stop.sh
   ```

2. Change the value of the `NUM_OF_ANALYSIS_WORKERS_INSTANCES` configuration variable, found in the **IP_JOP_PRR_<*version_number*>_ENU_linux/scripts/start.sh** script from 1 to 2.
3. Restart GPR using the following command:

   ```
   bash start.sh
   ```

In an HA deployment, by default, you have two analysis containers running, which means you can run two analysis jobs at the same time. All other analysis jobs are queued. To change this value, perform the following steps:

1. Stop GPR using the following command:

   ```
   bash stop.sh
   ```

2. Change the value of the `NUM_OF_ANALYSIS_WORKERS_INSTANCES` configuration variable, found in the **IP_JOP_PRR_<*version_number*>_ENU_linux/ha-scripts/start.sh** script, from 2 to your desired number.
3. Restart GPR using the following command:

   ```
   bash start.sh
   ```

To increase the processing speed for each analysis job, increase the number of CPUs the analysis container can use.

To change this value, perform the following steps:

1. Stop GPR using the following command:

   ```
   bash stop.sh
   ```

2. Edit the `cpus` setting in the `analysis` section of the appropriate file for your environment:

- For single-server deployments: **IP_JOP_PRR_<*version_number*>_ENU_linux/scripts/docker-compose.yml**
- For HA deployments: **IP_JOP_PRR_<*version_number*>_ENU_linux/ha-scripts/swarm/worker-swarm.yml**

1. Restart GPR using the following command:

   ```
   bash start.sh
   ```

Purge Workers

By default, GPR creates one purge container for each host, each using at most one CPU and at most two GB RAM.

There is no need to scale this type of worker either vertically or horizontally.