Vantiq Edge Reference Guide

Overview

Vantiq Edge installations are non-clustered Vantiq servers designed to run in edge environments, typically very close to data sources. This proximity facilitates low-latency remote application processing, allowing for a more distributed workload across large applications. While it is not a requirement, Vantiq Edge installations usually communicate with a clustered Vantiq cloud installation.

Vantiq Edge installations provide all Vantiq core services, but do not include the following:

- Vantiq's monitoring services (Grafana)
- Integration with an OAuth provider
- Clustered deployment

This guide describes the process for deploying Vantiq Edge installations via:

- an executable file
- as a Docker image *(recommended)*

In addition to the deployment information, this guide also provides instructions for managing the edge installations. General knowledge of Java applications and Docker is assumed (for each respective deployment method).

Requirements

General Requirements

For all deployment methods, a Vantiq license is required to startup the edge server. The license comprises two files:

- `license.key`
- `public.pem`

The license files can be acquired by contacting Vantiq Technical Support via email, at support@vantiq.com.

For the Docker deployment method, a quay.io ID must be registered with Vantiq Technical Support. This is necessary to enable access to the Vantiq Edge, AI-Assistant and GenAI Flow Service Docker images that are stored in a private quay.io repository. To access these images, the developer must create a quay.io account, and then email their quay.io ID to support@vantiq.com. Once the ID has been validated and registered, the developer will then have access to the Docker images.

Hardware Requirements

The edge computing device must meet or exceed the following requirements:

- 64-bit ARM or x86 processor
  - (For Docker deployments, an x86 processor is required)
- 8 GB main memory
- 32 GB permanent storage (to support MongoDB and Qdrant)

Software Requirements

- 64-bit operating system
- Recent version of macOS, Linux (such as Ubuntu), or Windows
  - Ubuntu installation instructions (useful for nano devices like Raspberry Pi, Nvidia Jetson, etc.)
- Hardware Virtualization Technology enabled (Windows)
- Access to OS Administrator privileges
- Java Runtime Environment 1.11.x, (aka Java 11 JRE)
  - Ubuntu installation instructions
  - MacOS installation instructions
  - Windows installation instructions
- MongoDB 4.2 or OLDER support (64-bit)
  - **NOTE**: Vantiq will NOT run with MongoDB 4.3 or newer.
- Docker Engine running 19.03.12 or higher (for Docker deployments)
  - Ubuntu installation instructions
  - MacOS installation instructions
  - Windows installation instructions

Operating system	CPU Arch	Supported configurations
Linux	amd64	Docker Engine or Executable JAR

Operating system	CPU Arch	Supported configurations
Linux	arm64	Executable JAR
Windows	amd64	Docker Desktop on WSL2 or Docker Using apt on WSL2
MacOS Intel	amd64	Docker Desktop or other Docker
MacOS	Apple Silicon	Executable JAR

Deployment Methods

Vantiq Edge can be deployed either using Docker or as an Executable with a JAR.

For production, it is recommended to use docker engine and docker compose on Linux with appropriate x86-64 hardware. Vantiq Edge can be run on 64-bit client operating systems, though usually that is for test and evaluation purposes.

The following sections describe each Vantiq Edge deployment method in detail, offering a simple example for each.

Docker Deployment

> It is assumed that a quay.io ID has been registered with Vantiq Technical Support and that a suitable version of Docker, including either `docker-compose` or `docker compose`, has been installed.

Begin by creating a directory for the edge deployment in the target machine's filesystem (that is, the machine on which the image will be running). Then, navigate into the directory and create a sub-directory named `config`. Copy the `license.key` and `public.pem` files into the `config` sub-directory. Next, from the top-level edge deployment directory (the parent directory of `config`), create a file named `docker-compose.yml`. Paste the following content into the new `docker-compose.yml` file:

```yaml
services:
  vantiq_edge:
    container_name: vantiq_edge_server
    image: quay.io/vantiq/vantiq-edge:1.43
    ports:
      - 8080:8080
    depends_on:
      - vantiq_edge_mongo
      - vantiq_edge_qdrant
    restart: unless-stopped
    volumes:
      - ./config/license.key:/opt/vantiq/config/license.key
      - ./config/public.pem:/opt/vantiq/config/public.pem
    networks:
      - vantiq_edge

  vantiq_edge_mongo:
    container_name: vantiq_edge_mongo
    image: bitnamilegacy/mongodb:4.2.21
    restart: unless-stopped
    environment:
      - MONGODB_USERNAME=ars
      - MONGODB_PASSWORD=ars
      - MONGODB_DATABASE=ars02
      - MONGODB_ROOT_USER=root
      - MONGODB_ROOT_PASSWORD=ars
    volumes:
      - vantiq_edge_data:/bitnami:rw
    networks:
      vantiq_edge:
        aliases: [edge-mongo]

  vantiq_ai_assistant:
    container_name: vantiq_ai_assistant
    image: quay.io/vantiq/ai-assistant:1.43
    restart: unless-stopped
    network_mode: "service:vantiq_edge"

  vantiq_genai_flow_service:
    container_name: vantiq_genai_flow_service
    image: quay.io/vantiq/genaiflowservice:1.43
    restart: unless-stopped
    command: ["uvicorn", "app.genaiflow_service:app", "--host", "0.0.0.0", "--port", "8889"]
    network_mode: "service:vantiq_edge"

  vantiq_edge_qdrant:
    container_name: vantiq_edge_qdrant
    image: qdrant/qdrant:v1.13.4
    restart: unless-stopped
    volumes:
      - qdrantData:/qdrant/storage
    networks:
      vantiq_edge:
        aliases: [edge-qdrant]

  vantiq_unstructured_api:
    container_name: vantiq_unstructured_api
    image: quay.io/vantiq/unstructured-api:0.0.82
    restart: unless-stopped
    environment:
      - PORT=18000
      - UNSTRUCTURED_PARALLEL_MODE_ENABLED=true
      - UNSTRUCTURED_PARALLEL_MODE_URL=http://localhost:18000/general/v0/general
      - UNSTRUCTURED_PARALLEL_MODE_SPLIT_SIZE=20
      - UNSTRUCTURED_PARALLEL_MODE_THREADS=4
      - UNSTRUCTURED_DOWNLOAD_THREADS=4
    network_mode: "service:vantiq_edge"

networks:
  vantiq_edge:
    ipam:
      config: []
volumes:
  vantiq_edge_data: {}
  qdrantData: {}
```

The value of `MONGODB_ROOT_PASSWORD` is used during the first image run to set the password of `MONGODB_ROOT_USER`. You can set the `MONGODB_ROOT_PASSWORD` to a different value if you prefer. Do not change the values of `MONGODB_USERNAME`, `MONGODB_PASSWORD`, and `MONGODB_DATABASE`.

The tag for the vantiq-edge image (the `1.43` in `image: quay.io/vantiq/vantiq-edge:1.43`) can be changed to whichever suitable version the developer would like to use. The version tags are listed in the quay.io repository. The tag for the `ai-assistant` and `genaiflowservice` images must match the tag for the vantiq-edge image for the major and minor versions (e.g., 1.43). The patch version (e.g., 1.43.1), if specified, might differ but should preferably match.

If the tag specified has the `major.minor` format, the latest patch version will be used. If a specific patch version is required, the tag should be specified as `1.43.1` or similar.

> Do NOT use any tag containing `SNAPSHOT`.

The same is true for the MongoDB version tag, so long as it meets the version requirements noted above in the Software Requirements section.

Once the `docker-compose.yml` file has been created and saved, make sure that Docker CLI has been logged into the quay.io container registry. This can be done by running `docker login quay.io` and entering the credentials when prompted. With that done, the edge installation can be started up using the following command in the same directory as the `docker-compose.yml` file:

```
docker compose up -d
```

or if you have `docker-compose` installed,

```
docker-compose up -d
```

This will run the docker compose services in the background. To check that they are up and running, you can examine the running docker processes using the following command:

```
docker compose ps
```

The initial startup may take a few minutes. You can check the logs of the edge server using the command,

```
docker compose logs -f vantiq_edge
```

> The name `vantiq_edge` is the name of the service in the `docker-compose.yml` file, service name also listed in the `docker compose ps` output.

To check the overall startup logs,

```
docker compose logs -f
```

As a best practice, always use `docker compose` for lifecycle management commands such as `start`, `stop`, `restart`, `up`, and `down`. This approach ensures proper handling of service dependencies while preventing the mismanagement or orphaning of compose-managed resources, including volumes and networks.

For example, to restart the edge server, run the following command,

```
docker compose restart vantiq_edge
```

Navigate to the Setting Up Vantiq Edge section for next steps.

Update Service Images

To update one or more service images, typically for a patch release, modify the `image` tag in the `docker-compose.yml` file for the relevant services. After making the changes, run the following commands:

```
docker compose pull
docker compose up -d
```

> If the image tags are specified with the `major.minor` format (e.g., 1.43), the latest patch version is automatically used for the update. If a specific patch version is required, the tag should be specified as `1.43.1` or similar.

After the update, you must refresh any open browser IDE sessions to ensure the latest UI version is loaded.

Upgrade

To upgrade a Vantiq Edge installation from a `1.42` to `1.43` release, you must perform a migration of the QDrant vector database. You can either download and run the migration script, or follow the manual steps below.

Create the upgrade directory

Create a directory named `upgrade` alongside your existing `config` directory within your Edge deployment directory. This directory will hold the script and configuration files needed for the migration.

Create configuration files

Within the `upgrade` directory, create the following files with the specified content:

`mongoDbService.json`:

```
{
  "hosts": [
    { "host": "vantiq_edge_mongo" }
  ]
}
```

`io.vantiq.aimanager.AiManager.json`:

```
{
  "config": {
    "semanticIndexService": {
      "vectorDB": {
        "host": "edge-qdrant"
      }
    }
  }
}
```

`vectorDbService.json`:

```
{
  "service": {
    "hostname": "edge-qdrant"
  }
}
```

Create the migration script

Create a script named `upgrade.sh` with the following content:

```
#!/bin/bash

network_name=$(docker network ls --format '{{.Name}}' | grep 'vantiq_edge')

docker run --rm \
  --name qdrant_migration_143 \
  --network $network_name \
  -v ./mongoDbService.json:/opt/vantiq/config/mongoDbService.json \
  -v ./vectorDbService.json:/opt/vantiq/config/vectorDbService.json \
  -v ./io.vantiq.aimanager.AiManager.json:/opt/vantiq/config/io.vantiq.aimanager.AiManager.json \
  quay.io/vantiq/qdrant-migration:1.43
```

Make the script executable

```
chmod +x upgrade.sh
```

Verify Edge Services are running

This should list all the Vantiq services, including `vantiq_edge`:

```
docker compose -f <path_to_docker_compose_file> ps
```

Stop the Edge service:

Stop only the `vantiq_edge` service (not the entire Docker Compose stack):

```
docker compose -f <path_to_docker_compose_file> stop vantiq_edge
```

Run the migration script:

```
./upgrade.sh
```

> Note: During migration, different 1.42 semantic indexes may be migrated into the same 1.43 collection. As a result, the migration process may attempt to create the same collection more than once. In this case you will see 'ALREADY_EXISTS' messages, which are expected and can safely be ignored."

Update Image Tags & Restart

After the migration is complete, stop the Edge services using:

```
docker compose -f <path_to_docker_compose_file> down
```

Then update the `docker-compose.yml` file to reference the appropriate 1.43 image tags (see the documentation above for guidance).

Once updated, restart the Edge services with:

```
docker compose -f <path_to_docker_compose_file> up -d
```

The QDrant migration is a one time operation, so you can remove the `upgrade` directory and its contents after the upgrade is complete.

> Note: If you are upgrading from version 1.41, you must first upgrade to 1.42 before proceeding to 1.43. The upgrade from 1.41 to 1.42 includes a required QDrant migration, similar to the one described above. If you want to use the above instructions to perform a migration from 1.41 to 1.42, use the `qdrant-migration:1.42.3` image in your upgrade script and adjust the `docker-compose.yml` service image tags accordingly (e.g., 1.42 vs 1.43).

Vantiq Edge Logs

If you need to customize the Vantiq Edge logs, such as when instructed by Vantiq Support, follow these steps:

- create a `log` directory alongside the existing `config` directory
- place the `logback.xml` file within the `config` directory
- in the `docker-compose.yml` file, locate the `volumes` section of the `vantiq_edge` service and add the following two lines:

```
      - ./config/logback.xml:/opt/vantiq/config/logback.xml
      - ./log:/var/log/vantiq
```

Upon restarting the Vantiq Edge server, the new log configuration will take effect, and logs will be written to the log directory.

Executable JAR Deployment

This deployment method only supports the deployment of the Vantiq Edge server with MongoDB. It does not support the deployment of the AI Assistant, GenAI Flow Service, or the Qdrant vector database. We recommend using Docker to deploy the Edge Server with AI capabilities.

The JAR deployment can be used on any platform that supports the Java version noted above, and does not require Docker or Hardware Virtualization support.

> The steps below assume that a suitable version of Java has already been installed.

MongoDB

The Vantiq Edge installation requires a running MongoDB instance (version 4.2 or earlier). To download MongoDB, follow the instructions below:

- In all cases, be certain to install *MongoDB 4.2 or OLDER*. The following MongoDB documentation links describe the installation process, but offer examples with newer versions of MongoDB.
- Ubuntu installation instructions
- macOS installation instructions
- Windows installation instructions

After successfully installing and starting up, the next step involves properly configuring the database. To do so, first open a command line connection to mongo.

- This is typically done by running the `mongo` command from the command line
  - In some cases, the command may not be recognized. The solution usually involves editing your `PATH` environment variable to include the directory in which the `mongo` executable is installed.
- For more details, visit the MongoDB documentation

Once inside the `mongo` command, create a new user in the ars02 database as follows:

```
use ars02

db.createUser({user: "ars", pwd: "ars", roles: ["readWrite", "dbAdmin"]})
```

- To verify success:

```
show users
```

- This should display information about the new user in json format

Running the Vantiq Executable

With MongoDB running and configured, the Vantiq Edge installation can now be started up. The Vantiq Edge executable is packaged as a `.zip` file, and can be retrieved by emailing Vantiq Technical Support (support@vantiq.com).

Once downloaded, create a directory for the edge deployment in the machine's file-system, and copy/unzip the `.zip` file there. There should be 3 resulting directories:

- `bin`
- `lib`
- `config`

Copy the license files (the `license.key` and `public.pem` files) into the `config` directory. Then, create the log directory: `/var/log/vantiq` and ensure that the log directory is writable by the Vantiq Edge installation process (*i.e.*, by the user under which the Vantiq executable will run).

Finally, to run the Vantiq Edge installation, execute the following command:

```
./<pathToInstallDirectory>/bin/vantiq.sh
```

Alternatively, the following command can run the executable in the background for Unix-based operating systems:

```
./<pathToInstallDirectory>/bin/vantiq.sh > /dev/null 2>&1 &
```

> Be sure to validate that the executable can run successfully *before* attempting to run in the background.

If either step fails, examine the logs in `/var/log/vantiq` or contact Vantiq Technical Support (support@vantiq.com).

> If no log files are present in this directory, it may be that that directory is not writable by the user running the Vantiq executable.

Navigate to the Setting Up Vantiq Edge section for next steps.

Windows bat file

The steps for using the *vantiq.bat* file for Windows are similar to those above. You will need to copy the license files and create the log directory, as described above.

Then, to run the Vantiq Edge installation on Windows, execute the following command:

```
<pathToInstallDirectory>\bin\vantiq.bat
```

Setting Up Vantiq Edge

With the Vantiq Edge installation now running, the next step is to properly configure the installation. This can be done through the Vantiq IDE.

The Edge IDE should look similar to that of the cloud-based installations. To access it, go to the URL of the deployed system (often times `http://localhost:8080/ui/ide/index.html`).

The IDE should initially display a window requesting login credentials. Login as the `system` user with the following credentials:

```
Username: system
Password: fxtrt$1492
```

After logging in, the next steps are:

- change the system password
- set the default LLMs API key
- create a new Organization and Namespace
- create a GenAIFlowService Service Connector

Changing the System Password

Select the menu `Administer/Users` and select the `system` user. Enter a new password and click `Save`, then type a second time to confirm the new password.

Setting the default LLMs API key

Open the LLMs pane by selecting `Add/LLMs` from the menu. In the configuration description column, you will find the names of the required API keys, represented as secrets (e.g., `@secrets(OPENAI_API_KEY)`) for the generative LLMs. To edit or add the necessary secret(s), go to `Administer/Advanced/Secrets`. For instance, if the secret for `OPENAI_API_KEY` is already defined to specify an OpenAI key, simply update it by entering your API key value in the Secret field.

Creating a New Organization and Namespace

It is imperative that no applications run in the `system` Namespace, so creating a new Organization and Namespace is **critical**. This will involve creating a new user with a different username/password.

For more information, refer to the Administrators Reference Guide or follow the User and Namespace Administration Tutorial

Creating a GenAIFlowService Service Connector

After creating a new Organization and Namespace, log in as the new user. Ensure that the GenAIFlowService Service Connector is present in the organization namespace. This connector is essential for the GenAI Flow Service to function. If the connector does not exist, you will need to create it using the default port of `8889` by importing the GenAIFlowService project.

If you want to create the connector yourself, use these steps:

- Select `Administer/Advanced/Service Connectors` from the menu, click the + *Connector* button, and specify the following values:
    - **Name** – `GenAIFlowService`.
    - **Is Organization-wide** – checked (aka `true`).
    - **Is Connector External** – checked (aka `true`).
    - **Connector Hostname** – `localhost`
    - **Connector Port** – `8889` (use the `--port` value specified in the `vantiq_genai_flow_service` service in the `docker-compose.yml` file).
    - **Request Timeout** – `120`

Click on `Create` to save the new Service Connector.

Creating a Video Assistant Service Connector

After creating a new Organization and Namespace, log in as the new user. If VIDEO sources are expected to be used, ensure that the VideoAssistant Service Connector is present in the organization namespace. This connector is essential for the VIDEO source(s) to function. If the connector does not exist, you will need to create it using the default port of `8890` by importing the VideoAssistant project.

If you want to create the connector yourself, use these steps:

- Select `Administer/Advanced/Service Connectors` from the menu, click the + *Connector* button, and specify the following values:
    - **Name** – `VideoAssistantService`.
    - **Is Organization-wide** – checked (aka `true`).
    - **Is Connector External** – checked (aka `true`).
    - **Connector Hostname** – `localhost`
    - **Connector Port** – `8890`.
    - **Request Timeout** – `120`

Click on `Create` to save the new Service Connector.

Vantiq Edge Self Node

The Vantiq Edge system self Node URI is set during initial startup with the default value `localhost:8080`. It is possible to override this default value by defining `vantiqExternalUri` within a file named `io.vantiq.vertx.BootstrapVerticle.json` placed in the Vantiq Edge `config` directory.

For example,

```
{
    "vantiqExternalUri": "<my server’s URI>"
}
```

For the override to take effect this file must be present during the initial Vantiq Edge startup.

> If the system has already been started once, you will have to manually edit the URI of the Node named *self* while logged into the *system* namespace.

Edge Installation Management

As all the Vantiq Edge installation’s data is stored in the adjacent MongoDB instance, it is crucial that production-quality deployments backup the database. The following notes provide high-level guidelines for this process:

- Backup MongoDB on a frequent basis.
  - Documentation for Backing up and Restoring with MongoDB Tools
  - The Vantiq Edge installation must be shutdown before the backup can be taken.
  - It is up to the developer to ensure the integrity of the database before taking a backup. Backing up a corrupted database is not advised.
- Store the MongoDB backup on a machine separate from the edge compute device.
- To restore MongoDB, you must start with an empty *ars02* database.
- After restoring MongoDB, manually check that the Vantiq Edge installation contains the data that is expected.
- To backup and restore semantic indexes and Qdrant data, refer to the Vantiq CLI `dump` and `load` commands for the `semanticindexes` resource.