High Availability

The standard WhatsApp Business API Client solution runs on a single Docker container. Having multiple Docker containers running will cause problems and result in your account being temporarily banned. This guide will walk you through how to set up high availability, allowing you to have Docker containers on stand-by in case the primary Docker container goes down.

> This high availability solution **requires an existing WhatsApp Business API Client single-instance installation** to run on top of it. If you haven't set up your WhatsApp Business API Client phone number yet, review the Installation documentation before proceeding with this solution.

This document covers:

Initial Setup

In a High Availability cluster there are three kinds of nodes: Webapp, Master, and Coreapp. They could be started separately in different machines, but they are required to be in the same network so that they can talk to each other.

A Webapp node is responsible for handling API traffic like the original Webapp container. A Coreapp node is responsible for handling messaging traffic to and from WhatsApp. Finally, a Master

node is responsible for monitoring Coreapp nodes in the cluster, so if one Coreapp node dies, it will redirect traffic to another Coreapp node for high availability. There could be multiple Webapp nodes, Coreapp nodes, and Master nodes in the cluster.

Active nodes are no longer referred to as slave nodes. They are called Coreapp nodes.

Note: For production environments, in most cases, the database should be run on a separate physical server from the Coreapp and Webapp containers. For true High Availability, it's recommended to run the Master, Webapp and Coreapp containers on different physical machines.

Set up a shared file system for media messages

If you don't care about media messages, skip this step.

To support sending/receiving media messages, it's required to set up a NFS file system and mount it to a local directory on all Webapp, Master, and Coreapp nodes. Make sure read/write permissions are granted on the shared directory.

Example NFS mount command:

```
mkdir new-local-directory
mount -t nfs nfs_server_IP_addr:/share_directory new-local-directory
```

Installation with Docker Compose

This guide requires Docker, a container platform that lets you run the WhatsApp Business API Client. Docker Compose is also required. Docker Compose is bundled with Docker for macOS and Windows but requires separate installation on Linux.

Developer setup with a single server

1. Install Docker on your system.
2. If Docker Compose is not bundled with your Docker installation, install it.
3. Download the `multiconnect-compose.yml` and `db.env` configuration files: WhatsApp_Configuration_Files.zip.
4. Open a console and navigate to the directory where you have saved the downloaded files.
5. If you have a MySQL installation running, change the values in the `db.env` file to reflect your MySQL configuration. If you do not have MySQL installed, the `multiconnect-compose.yml` and `db.env` files have a default configuration to bring up an instance in a local container.

6. Make sure there are no single-connect containers running before starting multiple containers for High Availability:

```
docker-compose -f your-single-connect-yml-filename stop
```

7. Run the following command in the console:

```
docker-compose -f multiconnect-compose.yml up
```

   You will get some output while the script downloads the Docker images and sets everything up. To run the containers in the background, use the `-d` parameter:

```
docker-compose -f multiconnect-compose.yml up -d
```

Once you have completed these steps, ensure that the containers are running with the following command:

```
docker-compose ps
```

By default, the Webapp container will be running on port 9090.

Production setup with multiple servers for each container

1. Install Docker on your systems.
2. If Docker Compose is not bundled with your Docker installations, install it.
3. Download the `multiconnect-coreapp.yml`, `multiconnect-master.yml`, `multiconnect-webapp.yml`, and `db.env` configuration files: WhatsApp_Configuration_Files.zip, and save each one to its respective server.
4. On each server, open a console and navigate to the directory where you have saved the downloaded files.
5. In a production setup, it is highly recommended that you use a standalone MySQL instance. Once you have one, change the values in the `db.env` file to reflect your MySQL configuration.
6. Make sure there are no single-connect containers running before starting multiple containers for High Availability:

```
docker-compose -f your-single-connect-yml-filename stop
```

7. Run the following commands in the console for each respective machine:

> The environment variable **EXTERNAL_HOSTNAME** should be an IP address or hostname that is accessible **from the machines running other containers**. The ports exposed in a compose file should be **open** to connections from machines running other containers.

```
EXTERNAL_HOSTNAME=MACHINE_HOSTNAME docker-compose -f multiconnect-coreapp
EXTERNAL_HOSTNAME=MACHINE_HOSTNAME docker-compose -f multiconnect-master.
EXTERNAL_HOSTNAME=MACHINE_HOSTNAME docker-compose -f multiconnect-webapp.
```

You will get some output while the script downloads the Docker images and sets everything up. To run the containers in the background, use the `-d` parameter:

```
EXTERNAL_HOSTNAME=MACHINE_HOSTNAME docker-compose -f multiconnect-coreapp
EXTERNAL_HOSTNAME=MACHINE_HOSTNAME docker-compose -f multiconnect-master.
EXTERNAL_HOSTNAME=MACHINE_HOSTNAME docker-compose -f multiconnect-webapp.
```

Once you have completed these steps, ensure that the containers are running with the following command:

```
EXTERNAL_HOSTNAME=MACHINE_HOSTNAME docker-compose -f multiconnect-coreapp.yml
EXTERNAL_HOSTNAME=MACHINE_HOSTNAME docker-compose -f multiconnect-master.yml
EXTERNAL_HOSTNAME=MACHINE_HOSTNAME docker-compose -f multiconnect-webapp.yml
```

By default, the Webapp container will be running on port 9090.

Upgrading

Developer setup with a single server

The `multiconnect-compose.yml` file has fields indicating container versions. For example:

```
services:
  ...
  waweb:
      image: docker.whatsapp.biz/web:v2.19.4
  ...
  master:
```

```
    image: docker.whatsapp.biz/coreapp:v2.19.4
  ...
  wacore:
      image: docker.whatsapp.biz/coreapp:v2.19.4
```

To upgrade an installation, change the version numbers in the `multiconnect-compose.yml` file:

```
services:
  ...
  waweb:
      image: docker.whatsapp.biz/web:v2.19.7
  ...
  master:
    image: docker.whatsapp.biz/coreapp:v2.19.7
  ...
  wacore:
      image: docker.whatsapp.biz/coreapp:v2.19.7
```

Then, restart the Docker containers:

```
docker-compose -f multiconnect-compose.yml up
```

Production set up with multiple servers for each container

The YAML files have fields indicating container versions. For example:

```
services:
  ...
  waweb:
      image: docker.whatsapp.biz/web:v2.19.4
```

```
services:
  ...
  wacore:
      image: docker.whatsapp.biz/coreapp:v2.19.4
```

```
services:
  ...
  master:
    image: docker.whatsapp.biz/coreapp:v2.19.4
```

To upgrade an installation, change the version numbers in the respective files:

```
services:
  ...
  waweb:
      image: docker.whatsapp.biz/web:v2.19.7
```

```
services:
  ...
  wacore:
      image: docker.whatsapp.biz/coreapp:v2.19.7
```

```
services:
  ...
  master:
    image: docker.whatsapp.biz/coreapp:v2.19.7
```

Then, restart the Docker containers:

```
EXTERNAL_HOSTNAME=MACHINE_HOSTNAME docker-compose -f multiconnect-coreapp.yml
EXTERNAL_HOSTNAME=MACHINE_HOSTNAME docker-compose -f multiconnect-master.yml u
EXTERNAL_HOSTNAME=MACHINE_HOSTNAME docker-compose -f multiconnect-webapp.yml u
```

Attaching existing volumes

If you have media volumes from a previous installation, replace the following volume definition in the YAML files:

```
volumes:
  whatsappData:
      driver: local
  whatsappMedia:
      driver: local
```

with:

```
volumes:
  whatsappData:
```

```
    external: true
  whatsappMedia:
    external: true
```

Bind mounts

> This is only recommended if you want to maintain an existing bind mount volume.

If you wish to directly mount a host path (an existing location on your host) into the container, you can do that by changing the volume line inside the service section to point to the host path.

```
wacore:
    volumes:
        /filepath/waent/data:/usr/local/waent/data
        /filepath/wamedia:/usr/local/wamedia
```

Uninstalling

Developer setup with a single server

> You'll have to repeat this for all the machines where you have nodes running.

If you need to reset your development environment by removing all containers, run the following command from the directory containing the `multiconnect-compose.yml` file:

```
docker-compose -f multiconnect-compose.yml down
```

In order to get rid of all volumes defined in the `multiconnect-compose.yml` file in addition to the containers, run `down` with the `-v` parameter:

```
docker-compose -f multiconnect-compose.yml down -v
```

Production set up with multiple servers for each container

If you need to reset your development environment by removing all containers, run the following command from the directory containing the YAML file on each server:

```
EXTERNAL_HOSTNAME=MACHINE_HOSTNAME docker-compose -f multiconnect-coreapp.yml
EXTERNAL_HOSTNAME=MACHINE_HOSTNAME docker-compose -f multiconnect-master.yml c
EXTERNAL_HOSTNAME=MACHINE_HOSTNAME docker-compose -f multiconnect-webapp.yml c
```

In order to get rid of all volumes defined in the YAML files in addition to the containers, run `down` with the `-v` parameter:

```
EXTERNAL_HOSTNAME=MACHINE_HOSTNAME docker-compose -f multiconnect-coreapp.yml
EXTERNAL_HOSTNAME=MACHINE_HOSTNAME docker-compose -f multiconnect-master.yml c
EXTERNAL_HOSTNAME=MACHINE_HOSTNAME docker-compose -f multiconnect-webapp.yml c
```

Troubleshooting Logs

To obtain logs for troubleshooting, run the following command on your servers:

```
  docker-compose logs > debug_output.txt
```