How to Install Exalate for GitHub on Docker

Last Modified on 04/08/2025 4:01 pm EDT

You can host Exalate for GitHub on your own server. To do so, you need to install Exalate on Docker.

Note: You need to install Docker. Check the docker documentation for more details.

Steps to Install Exalate for GitHub on Docker

1. Create directory and create docker-compose.yml file

Create a directory to hold the docker-compose file:

```
cd ~
mkdir exalate-gitnode
```

Create a **docker-compose.yml** file

Important: We recommend using the latest version of Exalate for GitHub. It can be found in the Release History.

Enter the latest version in the `image` tag. For example, in `image: idalko/githubnode:5.23.0`, the version of Exalate for GitHub is `5.23.0` .

The ***docker-compose.yml*** file should contain the following information in it:

```
services:
  database:
    restart: unless-stopped
    image: postgres:15.12
    volumes:
      - ./persist/db:/var/lib/postgresql/data
      - ./createdb.sh:/docker-entrypoint-initdb.d/init-user-db.sh
    environment:
      - POSTGRES_PASSWORD=changeme
      - DB_NAME=githubnode
      - DB_USER=exalate
      - DB_PASS=exalate
    networks:
      - dbnet


  githubnode:
    restart: unless-stopped
    ports:
      - 9002:9002


    #
    # Change the image tag to the required version
    # Check Release History on docs.exalate.com for an overview
    #
    image: idalko/githubnode:5.23.0
    depends_on:
      - database #wait for postgres to be started, not for ready
    volumes:
      - ./persist/home:/opt/githubnode/data
    environment:
      # Add your environment settings here
      - GITHUBNODE_PG_HOST=database
      - GITHUBNODE_PG_DB=githubnode?gssEncMode=disable
      - GITHUBNODE_PG_USER=exalate
      - GITHUBNODE_PG_PWD=exalate
      - GITHUBNODE_PORT=9002


     #As part of the security improvements, Exalate 5.6.0 and above validates the origin header
     #that the browser is sending upon every request to Exalate.
     #In order to validate the origin header, Exalate needs to know what is the URL
     #leading to it.
     #When you deploy Exalate onto a server, you configure a DNS rule such that
     #whenever people navigate to foo.com, they reach your server's Exalate.
     #You set up SSL so that https://foo.com leads to your Exalate on your server. once this is      done you need to set an environment variable NODE_SELF_URL=https://foo.co
m
     #for your Exalate docker container.

      - NODE_SELF_URL=https://foo.com

     # You can use following variables to link the node with nginx proxy
     # Replace exa-git.exalate.biz with the appropriate FQDN
     # - LETSENCRYPT_HOST=exa-git.exalate.biz
     # - VIRTUAL_HOST=exa-git.exalate.biz
     # - VIRTUAL_PORT=9002
     # To hanlde SSL termination we suggest following this article https://docs.exalate.com/docs/scripts-how-to-bring-up-a-reverse-proxy-using-the-jwildernginx-proxy

     # CACHE_EXPIRY_DURATION_HOURS variable defines how long the cache will remain in the app.
     # The default value of 8 hours can be changed by specifying the number of hours.
     # - CACHE_EXPIRY_DURATION_HOURS=20


    networks:
      - dbnet
      - default


networks:
  dbnet:
    driver: bridge
  default:
    driver: bridge
```

Note: the **- GITNODE_PG_DB=** and **- DB_NAME=** must match in order to start the db correctly.

Connecting to Postgres 10 or Higher

For unencrypted connections from Exalate to a Postgres version 10 or higher, you need to disable `gssEncMode` with the following setting:

```
# exalate is the name of the database on the postgres instance
#
GITHUBNODE_PG_DB=exalate?gssEncMode=disable
```

2. Ensure that a correct database is setup using a createdb.sh

Create or download a **createdb.sh** file (referenced from docker-compose.yml):

Note: Click createdb.sh to download the file.

The file **must be executable** and should contain the following information:

```
#!/bin/bash

TEST=`psql -U postgres <<-EOSQL
   SELECT 1 FROM pg_database WHERE datname='$DB_NAME';
EOSQL`

echo "******CREATING DOCKER DATABASE******"
if [[ $TEST == "1" ]]; then
    # database exists
    # $? is 0
    exit 0
else
psql -U postgres <<-EOSQL
   CREATE ROLE $DB_USER WITH LOGIN ENCRYPTED PASSWORD '${DB_PASS}' SUPERUSER;
EOSQL

psql -U postgres <<-EOSQL
   CREATE DATABASE $DB_NAME WITH OWNER $DB_USER ENCODING 'UNICODE' LC_COLLATE 'C' LC_CTYPE 'C' TEMPLATE template0
;
EOSQL

psql -U postgres <<-EOSQL
   GRANT ALL PRIVILEGES ON DATABASE $DB_NAME TO $DB_USER;
EOSQL
fi

echo ""
echo "******DOCKER DATABASE CREATED******"
```

Ensure that the volumes are included in your backup strategy:

- persist

3. Set Environment Variables if necessary

Below, you can find the environment variables used for the app container. All are optional, and in the given example, we've overridden GITHUBNODE_PG_DB, GITHUBNODE_PG_USER, and GITHUBNODE_PG_PWD just to show how to pass different credentials to the Exalate application.

Full list of environment variables:

Variable name	Default value	Example	Description
GITHUBNODE_PG_HOST	GITHUBNODE_PG_HOST=database	GITHUBNODE_PG_HOST=db.acme.com	Tells the exal where is the p database to c hosted
GITHUBNODE_PG_DB	GITHUBNODE_PG_DB=exalate	GITHUBNODE_PG_DB=exalate	Tells the exal what is the po database nan exalate applic
GITHUBNODE_PG_USER	GITHUBNODE_PG_USER=idalko	GITHUBNODE_PG_USER=exalate	Tells the exal what is the po database use exalate applic perform quer
GITHUBNODE_PG_PWD	GITHUBNODE_PG_PWD=idalko	GITHUBNODE_PG_PWD=secret	Tells the exal what is the po database use the exalate ap perform quer

Variable name	Default value	Example	Description

GITHUBNODE_PORT	GITHUBNODE_PORT=9000	GITHUBNODE_PORT=8080	Tells what wh to start the e application or is the port wit exalategitnod container, thu variable is ch example to 8
			`ports:`
			`  - 9000:90`
			should also b
			`ports:`
			`  - 8080:80`

GITHUBNODE_SMTP_HOST_NAME	GITHUBNODE_SMTP_HOST_NAME=mail.server.com	GITHUBNODE_SMTP_HOST_NAME=smtp.gmail.com	Is used to ser notifications a blocking sync
GITHUBNODE_SMTP_PORT	GITHUBNODE_SMTP_PORT=465	GITHUBODE_SMTP_PORT=587	is used to ser notifications a blocking sync
GITHUBNODE_SMTP_FROM	GITHUBNODE_SMTP_FROM=admin@admin.com	GITHUBNODE_SMTP_FROM=my.name@gmail.com	Is used to ser notifications a blocking sync
GITHUBNODE_SMTP_USER	GITHUBNODE_SMTP_USER=admin	GITHUBNODE_SMTP_USER=my.name	Is used to ser notifications a blocking sync
GITHUBNODE_SMTP_PASS	GITHUBNODE_SMTP_PASS=1234567	GITHUBNODE_SMTP_PASS=secret	Is used to ser notifications a blocking sync
GITHUBNODE_SMTP_TLS	GITHUBNODE_SMTP_TLS=true	GITHUBNODE_SMTP_TLS=true	Is used to ser notifications a blocking sync Can be set to the GITHUBNODE should be set that accepts non-TLS conn
HTTP_HEADERS	n/a	HTTP_HEADERS="TestName1: testAddHeader1"	Allows additio to pass betwe and the serve request head
FEATURE_AI_ASSIST_ENABLED	n/a	FEATURE_AI_ASSIST_ENABLED=true	Switches on A in Exalate adr When enable Exalate AI to rules. **The AI Assis requires a rea internet conn*

Using a Proxy for Outgoing Connections

Whenever the Exalate node needs to use a proxy to establish outgoing connections, use the following parameters in the environment (naming should be obvious):

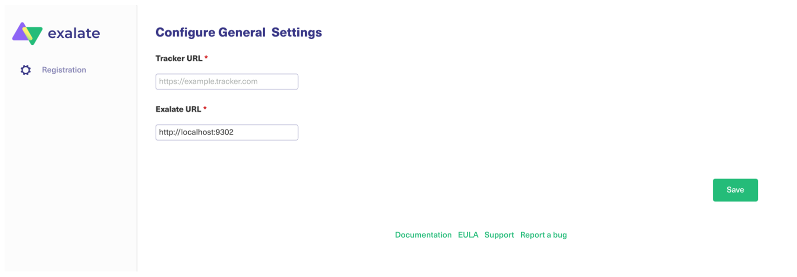
- PROXY_HTTP_HOST
- PROXY_HTTP_PORT
- PROXY_HTTPS_HOST
- PROXY_HTTPS_PORT

4. Start the Application

```
cd ~/exalate-gitnode
docker-compose up -d
```

Verify the installation

After performing these steps and checking that the container is up, you should be able to access the Exalate console via http://localhost:9302



Configure General Settings

Tracker URL *

https://example.tracker.com

Exalate URL *

http://localhost:9302

Note: You might need to set up local port forwarding in order to get this to work.

5. Register the Node

To be able to fully use the functionality of your new node, it needs to be registered on the mapper. This mapper acts as a DNS server, mapping tracker URLs to node URLs.

Please raise a ticket on the support portal providing the following:

- URL of the GitHub instance
- URL of the Exalate node which has been deployed on premise

How to Manage the Application on Docker

Run Queries to the Application's Database

```
cd ~/exalate-gitnode
docker exec -it exalate-gitnode_database_1 bash
su postgres
psql -A $DB_NAME
```

You can find all tables using PSQLs \dt+ command:

```
\dt+
```

All the Postgres SQL queries are permitted

To exit the application's DB:

```
\q
# \q exits the psql
exit
# exits the postgres user session
exit
# exits the exalate-gitnode_database_1 bash session
```

Inspect the Application's Filesystem

```
cd ~/exalate-gitnode
docker exec -it exalate-gitnode_githubnode_1 bash
```

Remove the application

```
cd ~/exalate-gitnode
docker-compose rm
```

Remove the application data

Warning: Do this only if you want to delete all the synchronization information, including the current synchronizations enqueued to be performed, and synchronization status. Ensure that the remote side you exalate issues with knows that you're stopping synchronization and are ready to handle synchronization errors.

```
cd ~/exalate-gitnode
# docker volume ls | grep exalate-gitnode_vol |   awk '{ print $2 }' | xargs docker volume r
m
docker volume rm exalate-gitnode_voldatabase
docker volume rm exalate-gitnode_volgithubnode
```

System Administration Tasks

With the Exalate for Jira Cloud is running on your environment, you are also required to do the mandatory system administration tasks

- Backup (& restore tests)
- Disaster recovery procedure
- Upgrades whenever needed

Note: Please note that an Exalate version has a lifespan of 2 years. This is to ensure backward compatibility over the whole platform. There are regular new versions deployed which contain bug fixes, security-related improvements, and even new features. Watch the release notes page for any new versions.

Upgrading Exalate on Docker

If you need to upgrade Exalate on Docker, here are the steps to follow:

1. **Edit the YAML File**:

   Open the `docker-compose.yml` file in a text editor and modify the image tag for the service you wish to upgrade.

   ```
   # use the latest version https://hub.docker.com/r/idalko/githubnode
   image: idalko/githubnode:latest
   depends_on:
   - database #wait for postgres to be started, not for ready
   ```

   Replace `latest` with the latest or desired version tag.

2. **Pull the Latest Image**:

   From the directory containing your `docker-compose.yml` file, pull the latest image.

   ```
   docker-compose pull
   ```

3. **Recreate the Container**:

Using Docker Compose, you can easily recreate the container with the new image.

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

The `-d` flag runs the containers in detached mode. Docker Compose automatically stops the old container and starts a new one based on the updated image.

4. **Post-Upgrade Checks**:

After starting the upgraded container, check to make sure everything is running as expected:

- Log into the Exalate interface and verify that all your configurations, connections are intact.
- Test out a few synchronizations to make sure they work as expected.
- Check for any errors in the Docker logs or the Exalate logs.

Troubleshooting

Issues during the installation of the Exalate server for GitHub

If you have issues during the installation of the Exalate app for GitHub, you can find logs describing possible problem inside `/tmp`.

The name for the file is generated randomly and automatically by the OS, but you can find the file by the creation date.

Issues while running the Exalate server for GitHub

Logs are generated under the directory: `/opt/githubnode/data/logs`.

Refer to these logs to get more information about possible issues and communicate with our support if you need any assistance.

Support

Check our Support options.

ON THIS PAGE