How to Install Exalate for Azure DevOps on Docker

Last Modified on 03/13/2024 8:16 am EDT

You can host Exalate for Azure DevOps 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 Azure DevOps on Docker

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

Create a directory to hold the docker-compose file:

```
cd ~
mkdir exalate-azurenode
```

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

> **Note**: Click docker-compose.yml to download the file.

> **Important**: We recommend using the latest version of Exalate for Azure DevOps. It can be found in the Release History. Enter the latest version in the `image` tag. For example, in `image: idalko/azurenode:5.6.0`, the version of Exalate for Azure DevOps is `5.6.0`.

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

```
version: '2'
services:
  database:
    restart: unless-stopped
    image: postgres:9.4
    volumes:
      - ./persist/db:/var/lib/postgresql/data
      - ./createdb.sh:/docker-entrypoint-initdb.d/init-user-db.sh
    environment:
      # adapt the default passwords
      - DB_NAME=azurenode
      - DB_USER=idalko
      - DB_PASS=idalko
      - POSTGRES_PASSWORD=changeme
    networks:
      - database
  azurenode:
    restart: unless-stopped

    # use the latest version https://hub.docker.com/r/idalko/azurenode
    image: idalko/azurenode:5.6.0
    depends_on:
      - database #wait for postgres to be started, not for ready
    volumes:
      - ./persist/home:/opt/azurenode/data
    environment:
      # Add your environment settings here, check documentation for details
      - AZURENODE_PG_HOST=database
      - AZURENODE_PG_DB=azurenode?gssEncMode=disable
      - AZURENODE_PG_USER=idalko
      - AZURENODE_PG_PWD=idalko
      - AZURENODE_PORT=9002


      # 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.
      #Now you need to set environment variable NODE_SELF_URL=https://foo.com
      #for your Exalate docker container.

      # Use following variables to link the node with jwilder/nginx proxy
      # Replace francis-ado.exalate.biz with the appropriate FQDN
      - LETSENCRYPT_HOST=francis-ado.exalate.biz
      - VIRTUAL_HOST=francis-ado.exalate.biz
      - VIRTUAL_PORT=9002
      - NODE_SELF_URL=https://foo.com


    networks:
      - database
      - proxy
networks:
  database:
    driver: bridge
  default:
    driver: bridge
  proxy:
    external:
      name: proxy
```

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
#
AZURENODE_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 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' TEMPL
ATE 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 of them are optional, and in the given example, we've overridden AZURENODE_PG_DB, AZURENODE_PG_USER, and AZURENODE_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
AZURENODE_PG_HOST	AZURENODE_PG_HOST=database	AZURENODE_PG_HOST=db.acme.com	Tells the exala where is the p to connect to
AZURENODE_PG_DB	AZURENODE_PG_DB=exalate	AZURENODE_PG_DB=exalate	Tells the exala what is the po name for the e application
AZURENODE_PG_USER	AZURENODE_PG_USER=idalko	AZURENODE_PG_USER=exalate	Tells the exala what is the po User name for application to with
AZURENODE_PG_PWD	AZURENODE_PG_PWD=idalko	AZURENODE_PG_PWD=secret	Tells the exala what is the po user's passwo application to with

Variable name	Default value	Example	Description

AZURENODE_PORT	AZURENODE_PORT=9000	AZURENODE_PORT=8080	Tells what whi start the exala on. Note that t within the exalateazuren container, thus is changed (fo the
			`ports: - 9000:9000`
			should also be
			`ports: - 8080:8080`

AZURENODE_SMTP_HOST_NAME	AZURENODE_SMTP_HOST_NAME=mail.server.com	AZURENODE_SMTP_HOST_NAME=smtp.gmail.com	Is used to send notifications a blocking synch
AZURENODE_SMTP_PORT	AZURENODE_SMTP_PORT=465	AZURENODE_SMTP_PORT=587	Is used to send notifications a blocking synch
AZURENODE_SMTP_FROM	AZURENODE_SMTP_FROM=admin@admin.com	AZURENODE_SMTP_FROM=my.name@gmail.com	Is used to send notifications a blocking synch
AZURENODE_SMTP_USER	AZURENODE_SMTP_USER=admin	AZURENODE_SMTP_USER=my.name	Is used to send notifications a blocking synch
AZURENODE_SMTP_PASS	AZURENODE_SMTP_PASS=1234567	AZURENODE_SMTP_PASS=secret	Is used to send notifications a blocking synch
AZURENODE_SMTP_TLS	AZURENODE_SMTP_TLS=true	AZURENODE_SMTP_TLS=true	Is used to send notifications a blocking synch be set to false AZURENODE_S should be set accepts non-S connections
HTTP_HEADER	n/a	HTTP_HEADERS="TestName1: testAddHeader1"	Allows additio pass between the server thr header.

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-azurenode
docker-compose up -d
```

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.
This is required to be able to install the **ADO extension** on the organization's site. Whenever

deploying the extension, the extension requests the mapper where the node serving the ADO organization is located.

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

- URL of the ADO organization
- URL of the Exalate node which has been deployed on-premise

About the Exalate Node URL

The exalate node needs to be reachable by:

- **The Azure DevOps instance**.
  - Exalate configures webhooks on the ADO project, which is used to notify Exalate whenever a web item is modified
  - Exalate needs to be reachable for the OAuth protocol to set up a trust relationship between the ADO instance and the Exalate node
- **The ADO users**
  - The sync panel and the Exalate console are web properties that need to be reachable by the users.

How to Manage the Application on Docker

Run Queries to the Application's Database

```
cd ~/exalate-azurenode
docker exec -it exalateazurenode_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 exalateazurenode_database_1 bash session
```

Inspect the Application's Filesystem

```
cd ~/exalate-azurenode
docker exec -it exalateazurenode_azurenode_1 bash
```

Remove the Application

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

Remove the Application Data

> **Warning**: Do this only if you wish 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-azurenode
# docker volume ls | grep exalateazurenode_vol |  awk '{ print $2 }' | xargs docker volume rm
docker volume rm exalateazurenode_voldatabase
docker volume rm exalateazurenode_volazurenode
```

System Administration Tasks

With the Exalate for Azure DevOps 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/azurenode
   image: idalko/azurenode: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 for Azure DevOps

If you have issues during the installation of the Exalate app for Azure DevOps, you can find logs describing possible problems 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 Azure DevOps

Logs are generated under the directory: `/opt/azurenode/data/logs (in the docker container)`

Refer to these logs to get more information about possible problems, and contact our support team if you need any assistance.

Support

Please read our Support options.

ON THIS PAGE

Steps to Install Exalate for Azure DevOps on Docker

How to Manage the Application on Docker

Upgrading Exalate on Docker

YouTube Channel

Ebooks

Still need help?

Join our Community

Visit our Service Desk

Find a Partner