How to Install Exalate for Salesforce on Docker

Last Modified on 04/08/2025 4:02 pm EDT

You can host Exalate for Salesforce 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.

How to Install Exalate for Salesforce on Docker

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

Create a directory to hold the docker-compose file:

```
cd ~
mkdir exalate-salesforce
```

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

> **Important**: We recommend using the latest version of Exalate for Salesforce. It can be found in the Release History. Enter the latest version in the `image` tag. For example, in `image: idalko/salesforcenode:5.23.0`, the version of Exalate for Salesforce 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=exalate
      - DB_NAME=salesforcenode
      - DB_USER=exalate
      - DB_PASS=exalate
    networks:
      - dbnet

  salesforce:
    restart: unless-stopped
    ports:
      - 9000:9002

    #
    # Change the image tag to the required version
    # Check Release History on docs.exalate.com for an overview
    #
```

```
    #
    image: idalko/salesforcenode:5.23.0
    depends_on:
      - database #wait for postgres to be started, not for ready
    volumes:
      - ./persist/home:/opt/salesforcenode/data
    environment:
      # Add your environment settings here
      - SALESFORCENODE_PG_HOST=database
      - SALESFORCENODE_PG_DB=salesforcenode?gssEncMode=disable
      - SALESFORCENODE_PG_USER=exalate
      - SALESFORCENODE_PG_PWD=exalate


        #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.com
      #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-snow.exalate.biz with the appropriate FQDN
      # - LETSENCRYPT_HOST=exa-sf.exalate.biz
      # - VIRTUAL_HOST=exa-sf.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-jwilderngінx-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 **- SALESFORCENODE_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:

```
# in this example 'exalate' is the name of the database on the postgres instance
#
SALESFORCENODE_PG_DB=exalate?gssEncMode=disable
```

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

Create or download a **createdb.sh** file:

Note: Click createdb.sh to download the file.

The file **must be executable** (you can use the command: **chmod +x createdb.sh** to make the file 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' 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 SALESFORCENODE_PG_DB, SALESFORCENODE_PG_USER, and SALESFORCENODE_PG_PWD just to show how to pass different credentials to the Exalate application.

Full list of environment variables:

Variable name	Example	Description

Variable name	Example	Description
HTTP_HEADER	HTTP_HEADERS="TestName1: testAddHeader1"	Allows additional information to pass between the clients and the server through the request header.
SALESFORCENODE_PG_HOST	SALESFORCENODE_PG_HOST=database	Tells Exalate where is the Postgres database to connect is hosted
SALESFORCENODE_PG_DB	SALESFORCENODE_PG_DB=exalate	Tells Exalate what is the Postgres database name for the Exalate application
SALESFORCENODE_PG_USER	SALESFORCENODE_PG_USER=exalate	Tells the Exalate application what is the Postgres database user name for the Exalate application to perform queries with
SALESFORCENODE_PG_PWD	SALESFORCENODE_PG_PWD=secret	Tells the Exalate application what is the Postgres database user's password for the Exalate application to perform queries with
SMTP_HOST_NAME	SMTP_HOST_NAME=smtp.gmail.com	The host name of the SMTP server used to send error notifications
SMTP_PORT	SMTP_PORT=587	Port (also check the TLS setting)

Variable name	Example	Description
SMTP_FROM	SMTP_FROM=my.name@gmail.com	Email that is used to send error notifications
SMTP_LOGIN	SMTP_LOGIN=my.name	Login to the SMTP service
SMTP_PASS	SMTP_PASS=secret	Password to the SMTP service
SMTP_TLS	SMTP_TLS=true	Can be set to false, but then the SMTP_PORT should be set to the port, that accepts non-SSL and non-TLS connections
FEATURE_AI_ASSIST_ENABLED	FEATURE_AI_ASSIST_ENABLED=true	AI Assist feature in Exalate admin console. When enabled, users can use Exalate AI to generate sync rules. **The AI Assist feature requires a real-time internet connection.*

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-salesforce
docker-compose up -d
```

Once the container is started you should be able to access it using the port specified in the **ports:** section of your docker-compose file(for example:).

To complete the app Oauth registration flow with Salesforce is required that your container's Fully Qualified Domain Name (NODE_SELF_URL specified in the docker-compose file) has a A+ rating on https://www.ssllabs.com/ssltest/analyze.html.
To manage SSL termination in our experience the simplest way is using the jwilder/nginx-proxy approach.

Once this is done you should be able to access the application registration page using the Fully Qualified Domain Name(FQDN).

5. Install the bridge app on the Salesforce Org

Follow the normal installation flow in the Salesforce Installation Guide, this will create the connected app that will provide you the Consumer Key and Consumer Secret that the you will need to complete the registration.

6. 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 our support portal providing the following:

- Salesforce instance URL
- Exalate node URL/Fully Qualified Domain Name(FQDN)

Once this request is raised our team will take the necessary steps to register you container and provide an update statement, this will be used in the container DB to provide the registration detail to the container.

To access the container's DB you can use the following command(`exalate-salesforce-database-1` is the example db name):

```
docker exec -it exalate-salesforce-database-1 psql -U postgres
```

The update statement will look something like this:

```
INSERT INTO general_settings (id, url, exalate_url, instance_uid, field_values)VALUES (...
```

To ensure the details are correctly updated you can restart the container after updating the DB:

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

Once the container is up you should be able to access the locally running application in your Salesforce Org.

How to Manage the Application on Docker

Run Queries to the Application's Database

```
cd ~/exalate-salesforce
docker exec -it exalatesalesforcenode_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 exalatesalesforcenode_database_1 bash session
```

Inspect the Application's Filesystem

```
cd ~/exalate-salesforce
docker exec -it exalatesalesforcenode_salesforcenode_1 bash
```

Remove the Application

```
cd ~/exalate-salesforce
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-salesforce
# docker volume ls | grep exalatesalesforcenode_vol |  awk '{ print $2 }' | xargs docker volume rm
docker volume rm exalatesalesforcenode_voldatabase
docker volume rm exalatesalesforcenode_volsalesforcenode
```

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/salesforcenode
   image: idalko/salesforcenode: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 Salesforce

If you have issues during the installation of the Exalate app for Salesforce, 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.

Problems while running the Exalate server for Salesforce

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

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

Support

Please read our Support options.

ON THIS PAGE