sonatype/nexus3

By sonatype • Updated 10 days ago

Sonatype Nexus Repository Manager 3

Container

Pulls **10M+**

Overview Tags

Sonatype Nexus3 Docker: sonatype/nexus3

chat on gitter

A Dockerfile for Sonatype Nexus Repository Manager 3, starting with 3.18 the image is based on the Red Hat Universal Base Image while earlier versions used CentOS.

Contribution Guidelines

Go read our contribution guidelines to get a bit more familiar with how we would like things to flow.

Running

To run, binding the exposed port 8081 to the host, use:

```
$ docker run -d -p 8081:8081 --name nexus sonatype/nexus3
```

To test:

```
$ curl http://localhost:8081/
```

Building the Nexus Repository Manager image

To build a docker image from the Docker file you can use this command:

```
$ docker build --rm=true --tag=sonatype/nexus3 .
```

The following optional variables can be used when building the image:

- NEXUS_VERSION: Version of the Nexus Repository Manager
- NEXUS_DOWNLOAD_URL: Download URL for Nexus Repository, alternative to using `NEXUS_VERSION` to download from Sonatype
- NEXUS_DOWNLOAD_SHA256_HASH: Sha256 checksum for the downloaded Nexus Repository Manager archive. Required if `NEXUS_VERSION` or `NEXUS_DOWNLOAD_URL` is provided

Chef Solo for Runtime and Application

Chef Solo is used to build out the runtime and application layers of the Docker image. The Chef cookbook being used is available on GitHub at sonatype/chef-nexus-repository-manager.

Testing the Dockerfile

We are using `rspec` as the test framework. `serverspec` provides a docker backend (see the method `set` in the test code) to run the tests inside the docker container, and abstracts away the difference between distributions in the tests (e.g. yum, apt,...).

```
rspec [--backtrace] spec/Dockerfile_spec.rb
```

Red Hat Certified Image

A Red Hat certified container image can be created using `Dockerfile.rh.el` which is built to be compliant with Red Hat certification. The image includes additional meta data to comform with Kubernetes and OpenShift standards, a directory with the licenses applicable to the software and a man file for help on how to use the software. It also uses an ENTRYPOINT script the ensure the running user has access to the appropriate permissions for OpenShift 'restricted' SCC. In addition to the Red Hat Enterprise Linux image, `Dockerfile.rh.centos` provides the same additions but with a CentOS base.

The Red Hat certified container image is available from the Red Hat Container Catalog and qualified accounts can pull it from registry.connect.redhat.com.

Notes

- Our system requirements should be taken into account when provisioning the Docker container.
- Default user is `admin` and the uniquely generated password can be found in the `admin.password` file inside the volume. See Persistent Data for information about the volume.
- It can take some time (2-3 minutes) for the service to launch in a new container. You can tail the log to determine once Nexus is ready:

```
$ docker logs -f nexus
```

- Installation of Nexus is to `/opt/sonatype/nexus`.
- A persistent directory, `/nexus-data`, is used for configuration, logs, and storage. This directory needs to be writable by the Nexus process, which runs as UID 200.
- There is an environment variable that is being used to pass JVM arguments to the startup script
    - `INSTALL4J_ADD_VM_PARAMS`, passed to the Install4J startup script. Defaults to `-Xms1200m -Xmx1200m -XX:MaxDirectMemorySize=2g -Djava.util.prefs.userRoot=${NEXUS_DATA}/javaprefs`.

  This can be adjusted at runtime:

  ```
  $ docker run -d -p 8081:8081 --name nexus -e INSTALL4J_ADD_VM_PARAMS="-Xms2g -Xmx2g -XX:MaxDirectMemorySize=3g  -Djava.util.prefs.userRoo
  ```

  Of particular note, `-Djava.util.prefs.userRoot=/some-other-dir` can be set to a persistent path, which will maintain the installed Nexus Repository License if the container is restarted.
- Another environment variable can be used to control the Nexus Context Path
    - `NEXUS_CONTEXT`, defaults to /

This can be supplied at runtime:

```
$ docker run -d -p 8081:8081 --name nexus -e NEXUS_CONTEXT=nexus sonatype/nexus3
```

Persistent Data

There are two general approaches to handling persistent storage requirements with Docker. See Managing Data in Containers for additional information.

1. *Use a docker volume*. Since docker volumes are persistent, a volume can be created specifically for this purpose. This is the recommended approach.

   ```
   $ docker volume create --name nexus-data
   $ docker run -d -p 8081:8081 --name nexus -v nexus-data:/nexus-data sonatype/nexus3
   ```

2. *Mount a host directory as the volume*. This is not portable, as it relies on the directory existing with correct permissions on the host. However it can be useful in certain situations where this volume needs to be assigned to certain specific underlying storage.

   ```
   $ mkdir /some/dir/nexus-data && chown -R 200 /some/dir/nexus-data
   $ docker run -d -p 8081:8081 --name nexus -v /some/dir/nexus-data:/nexus-data sonatype/nexus3
   ```

Getting Help

Looking to contribute to our Docker image but need some help? There's a few ways to get information or our attention:

- Chat with us on Gitter
- File an issue on our public JIRA
- Check out the Nexus3 tag on Stack Overflow
- Check out the Nexus Repository User List

Docker Pull Command

```
docker pull sonatype/nexus3
```

Owner

sonatype