

GERRIT Setup

|  |  |
| --- | --- |
| **Author** |  |
| **Approved By** |  |
| **Application Version** |  |
| **Date of Issue** |  |
| **Release Note Number** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Document Control Section** | | | | |
|  | | | | |
| **Revision History:** | | | | |
| **Document version** | **Pages** | **Action** | **Date of revision** | **By** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  | | | | |

Table of Content

[1.0 Confidentiality / Legal notification 3](#_Toc331583772)

[2.0 Introduction 5](#_Toc331583773)

[2.1 why does gerrit fit in 5](#_Toc331583774)

[3.0 Building docker image 5](#_Toc331583773)

[3.1 Creating docker image 3](#_Toc331583772)

[3.2 USING PERSISTENT VOLUMES 5](#_Toc331583773)

[3.3 systemd file 3](#_Toc331583772)

1. Confidentiality / Legal notification

Confidential. Copyright © AI Enterprise., 2018. All rights reserved. This document, including the information contained

herein, is restricted, confidential and proprietary to AI Enterprise, and is to be used only by and disclosed only to those

within AI Enterprise with a need-to-know. DO NOT COPY OR FORWARD INTERNALLY OR RELEASE outside AI Enterprise

without authorization in writing by a Sr. Vice President, Principal, or Director-level manager or a direct designee

thereof who has responsibility for the information contained herein.

1. Introduction

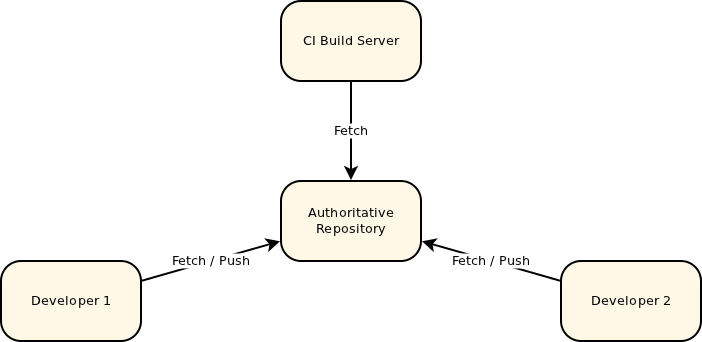
This document describes about create Dockerfile and deploy using yaml file from docker-compose tool. Gerrit provides web based code review and repository management for the Git version control system.

Gerrit is intended to provide a lightweight framework for reviewing every commit before it is accepted into the code base. Changes are uploaded to Gerrit but don’t become a part of the project until they’ve been reviewed and accepted.

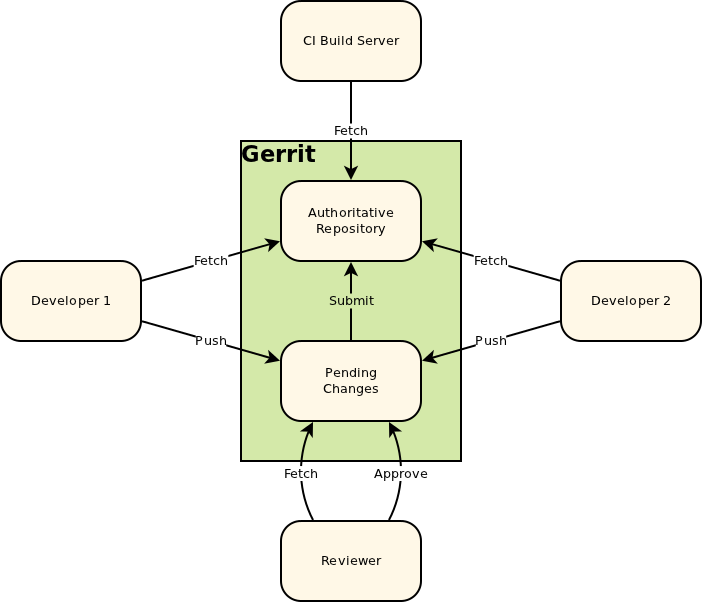
Gerrit is firstly a staging area where changes can be checked over before becoming a part of the code base. It is also an enabler for this review process, capturing notes and comments about the changes to enable discussion of the change. This is particularly useful with distributed teams where this conversation can’t happen face to face. Even with a co-located team having a review tool as an option is beneficial because reviews can be done at a time that is convenient for the reviewer. This allows the developer to create the review and explain the change while it is fresh in their mind. Without such a tool they either need to interrupt someone to review the code or switch context to explain the change when they’ve already moved on to the next task.

* 1. Where does Gerrit fit in?

Any team with more than one member has a central source repository of some kind (or they should). Git can theoretically work without such a central location but in practice there is usually a central repository. This serves as the authoritative copy of what is actually in the project. This is what everyone fetches from and pushes to and is generally where build servers and other such tools get the source from.



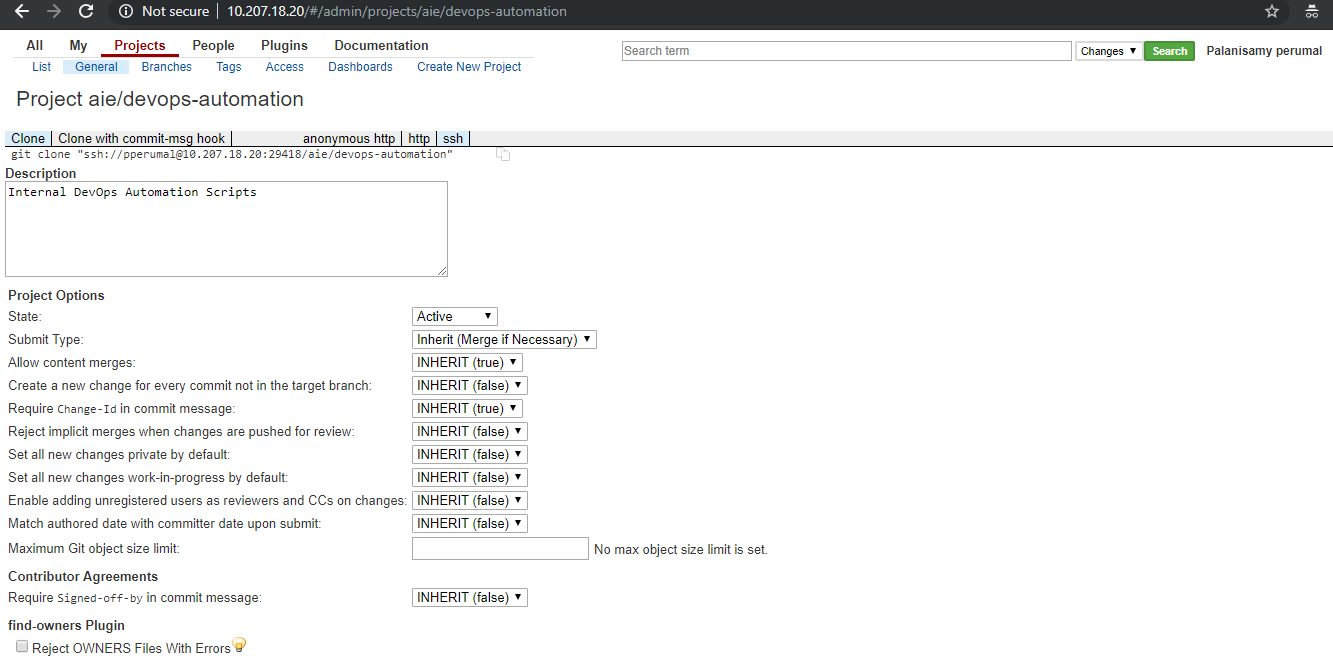
Gerrit is deployed in place of this central repository and adds an additional concept, a store of pending changes. Everyone still fetches from the authoritative repository but instead of pushing back to it, they push to this pending changes location. A change can only be submitted into the authoritative repository and become an accepted part of the project once the change has been reviewed and approved.

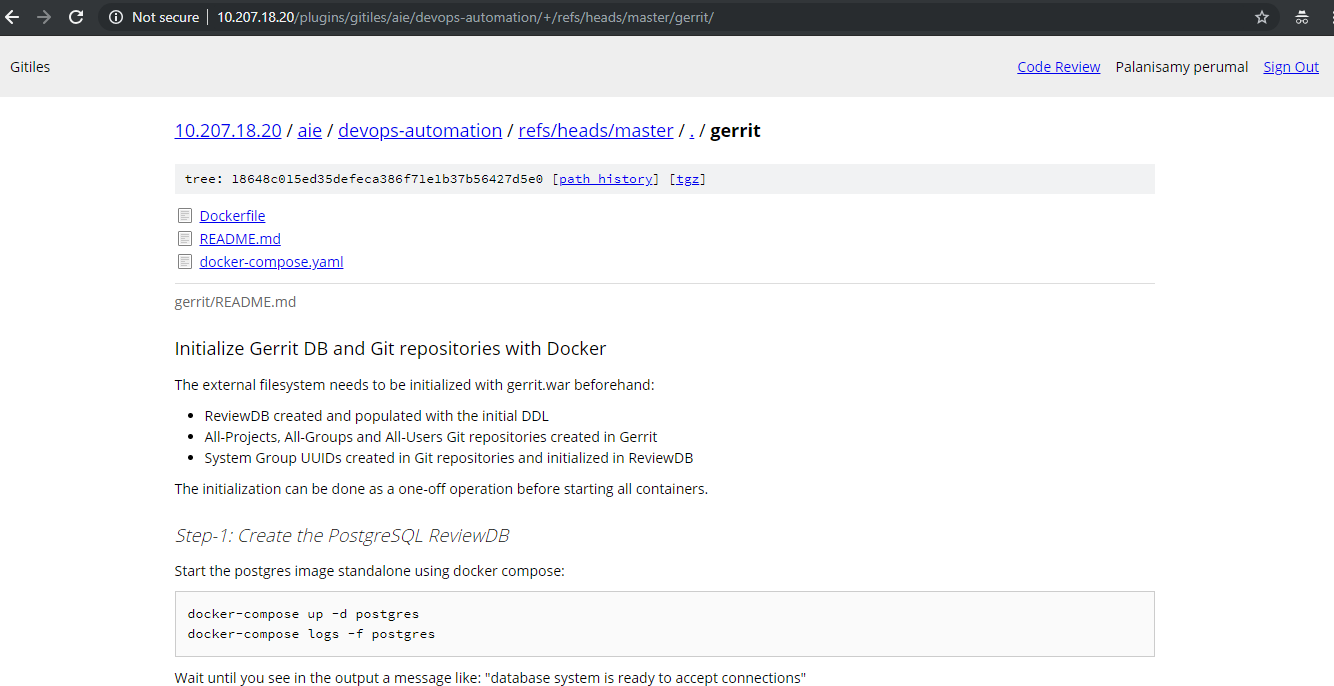


Like any repository hosting solution, Gerrit has a powerful [access control model.](https://git.eclipse.org/r/Documentation/access-control.html) Users can even be granted access to push directly into the central repository, bypassing code review entirely. Gerrit can even be used without code review, used simply to host the repositories and controlling access. But generally it’s just simpler and safer to go through the review process even for users who are allowed to directly push.

1. building Docker image

You can clone the code repository from AIE gerrit which will help you to create a custom docker image from scratch.





About Dockerfile

We’re adding all the configurations including single sign on, service ports etc. You can refer the sample Dockerfile here.

|  |  |
| --- | --- |
|  | FROM ubuntu:16.04  MAINTAINER AIEnterprise DevOps |
|  |  |
|  | # Add Gerrit packages repository |
|  | RUN echo "deb mirror://mirrorlist.gerritforge.com/deb gerrit contrib" > /etc/apt/sources.list.d/GerritForge.list |
|  |  |
|  | RUN apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 1871F775 |
|  | RUN apt-get update |
|  | RUN apt-key update |
|  | RUN apt-get -y install sudo |
|  |  |
|  | # Install OpenJDK and Gerrit in two subsequent transactions |
|  | # (pre-trans Gerrit script needs to have access to the Java command) |
|  | RUN apt-get -y install openjdk-8-jdk |
|  | RUN apt-get -y install gerrit=2.16.4-2 && rm -f /var/gerrit/logs/\* |
|  |  |
|  | USER gerrit |
|  | RUN java -jar /var/gerrit/bin/gerrit.war init --batch --install-all-plugins -d /var/gerrit |
|  | RUN java -jar /var/gerrit/bin/gerrit.war reindex -d /var/gerrit |
|  | RUN git config -f /var/gerrit/etc/gerrit.config --add container.javaOptions "-Djava.security.egd=file:/dev/./urandom" |
|  |  |
|  | RUN git config -f /var/gerrit/etc/gerrit.config --unset auth.type "DEVELOPMENT\_BECOME\_ANY\_ACCOUNT" |
|  | RUN git config -f /var/gerrit/etc/gerrit.config --add auth.type "LDAP" |
|  | RUN git config -f /var/gerrit/etc/gerrit.config --add auth.gitBasicAuthPolicy "LDAP" |
|  | RUN git config -f /var/gerrit/etc/gerrit.config --add ldap.server "${LDAP\_SERVER}" |
|  | RUN git config -f /var/gerrit/etc/gerrit.config --add ldap.accountBase "cn=accounts,dc=aienterprise,dc=com" |
|  | RUN git config -f /var/gerrit/etc/gerrit.config --add ldap.groupBase "cn=accounts,dc=aienterprise,dc=com" |
|  | RUN git config -f /var/gerrit/etc/gerrit.config --add ldap.accountFullName "displayName" |
|  | RUN git config -f /var/gerrit/etc/gerrit.config --add ldap.accountEmailAddress "mail" |
|  | RUN git config -f /var/gerrit/etc/secure.config --add ldap.password "bang@2018AIE" |
|  |  |
|  | ENV CANONICAL\_WEB\_URL= |
|  | ENV LDAP\_SERVER=ldap://10.207.18.24 |
|  |  |
|  | # Allow incoming traffic |
|  | EXPOSE 29418 8080 |
|  |  |
|  | VOLUME ["/var/gerrit/git", "/var/gerrit/index", "/var/gerrit/cache", "/var/gerrit/db", "/var/gerrit/etc"] |
|  |  |
|  | # Start Gerrit |
|  | CMD git config -f /var/gerrit/etc/gerrit.config gerrit.canonicalWebUrl "${CANONICAL\_WEB\_URL:-http://10.207.18.20}" && \ |
|  | git config -f /var/gerrit/etc/gerrit.config noteDb.changes.autoMigrate true && \ |
|  | /var/gerrit/bin/gerrit.sh run |

* 1. Creating docker image

Navidate to Gerrit directory and run the below command.

Docker build -t aie-gerrit:1.0 .

This command will build a docker image and storage the image as locally. We can define this image name in the docker-compose.yaml for run the container

* 1. Using persistent volumes

Use docker persistent volumes to keep Gerrit data across restarts. See below a sample docker-compose.yaml for persisting the H2 Database, Lucene indexes, Caches and Git repositories.

Example of /docker-compose.yaml

version: '3'

services:

gerrit:

image: aie-gerrit:1.0

volumes:

- git-volume:/var/gerrit/git:rw

- etc-volume:/var/gerrit/etc:rw

- index-volume:/var/gerrit/index:rw

- cache-volume:/var/gerrit/cache:rw

entrypoint: java -jar /var/gerrit/bin/gerrit.war init -d /var/gerrit

ports:

- "29418:29418"

- "8080:8080"

volumes:

git-volume: {}

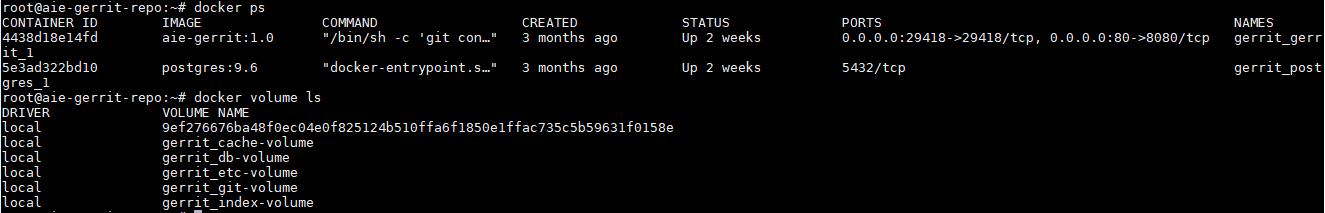
etc-volume: {}

db-volume: {}

index-volume: {}

cache-volume: {}

Run docker-compose up to trigger the build and execution of your custom Gerrit docker setup. Docker container will show as below.



* 1. Systemd file

we’ve created systemd file for automatically bringup docker containers when the system goes down.

Example of systemd service file

[Unit]

Description=gerrit service

After=docker.service

[Service]

Type=oneshot

RemainAfterExit=yes

StandardError=null

StandardOutput=null

WorkingDirectory=/root/gerrit

ExecStart=/usr/local/bin/docker-compose -f docker-compose.yaml start

ExecStop=/usr/local/bin/docker-compose -f docker-compose.yaml stop

[Install]

WantedBy=multi-user.target

We can restart the gerrit container using systemctl restart gerrit