Mohammad Saeed Pourmazar

https://github.com/MohammadSaeedPourmazar

https://gitlab.com/MohammadSaeedPourmazar

https://medium.com/@MohammadSaeedPourmazar

https://dev.to/MohammadSaeedPourmazar

https://www.youtube.com/@MohammadSaeedPourmazar

https://www.instagram.com/MohammadSaeedPourmazar

https://www.facebook.com/MohammadSaeedPourmazar

https://www.linkedin.com/in/MohammadSaeedPourmazar/

https://orcid.org/0009-0008-9383-419X

Installing Sonatype Nexus, SonarQube, and Jenkins using Docker

Installing Sonatype Nexus, SonarQube, and Jenkins using Docker is a great way to build a CI/CD pipeline and manage artifacts and code quality. I'll walk you through the entire process from scratch.

Prerequisites

Ensure your system has the following installed:

- *Docker (Install from https://docs.docker.com/get-docker/)*
- *Docker Compose (Install from https://docs.docker.com/compose/install/)*

You can verify installation with:

```
docker --version
docker compose version
```

Step-by-Step Setup Using Docker

We'll use Docker Compose to run:

1. *Jenkins – CI/CD automation tool*
2. *Sonatype Nexus – Artifact Repository Manager*
3. *SonarQube – Code Quality & Static Analysis tool*

Step 1: Create a Project Directory

```
mkdir devops-stack
```

```
cd devops-stack
```

Create a docker-compose.yml file:

```
touch docker-compose.yml
```

Step 2: Add Docker Compose Configuration

```
nano docker-compose.yml
```

Paste the following content into docker-compose.yml:

```
version: '3.8'

services:
  jenkins:
    image: jenkins/jenkins:lts
    container_name: jenkins
    ports:
      - "8081:8080"
      - "50000:50000"
    volumes:
      - jenkins_home:/var/jenkins_home
    restart: unless-stopped

  nexus:
    image: sonatype/nexus3
    container_name: nexus
    ports:
      - "8082:8081"
    volumes:
      - nexus_data:/nexus-data
    restart: unless-stopped

  sonarqube:
    image: sonarqube:lts
    container_name: sonarqube
    ports:
      - "9000:9000"
    environment:
      - SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true
    volumes:
      - sonarqube_data:/opt/sonarqube/data
      - sonarqube_extensions:/opt/sonarqube/extensions
    restart: unless-stopped

volumes:
  jenkins_home:
  nexus_data:
  sonarqube_data:
  sonarqube_extensions:
```

Step 3: Run the Containers

Run the following command inside your devops-stack directory:

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

This will download and start:

- *Jenkins on port **8081***
- *Nexus on port **8082***
- *SonarQube on port **9000***

Step 4: Access the Services

Tool	***URL***	***Default Login Info***
Jenkins	*http://localhost:8081*	*Get admin password from logs (explained below)*
Nexus	*http://localhost:8082*	*admin / password from log (/nexus-data/admin.password)*
SonarQube	*http://localhost:9000*	*admin / admin*

Step 5: Get Initial Admin Passwords

Jenkins:

```
sudo docker exec -it jenkins cat /var/jenkins_home/secrets/initialAdminPassword
```

Nexus:

```
sudo docker exec -it nexus cat /nexus-data/admin.password
```

Use those passwords to log into the web UI and set your own password.

Step 6: Configure Jenkins with SonarQube & Nexus

1. Install Required Plugins in Jenkins:

*Go to **Manage Jenkins > Manage Plugins**, then install:*

- ***SonarQube Scanner***
- ***Pipeline***
- ***Docker Pipeline***
- ***Nexus Artifact Uploader***
- ***Git***

2. Configure SonarQube in Jenkins:

*Go to **Manage Jenkins > Configure System**, then:*

- *Scroll to **SonarQube Servers** section*
- *Add Name and Server URL (**http://sonarqube:9000**)*
- *Generate token in SonarQube UI (**My Account > Security**) and use it in Jenkins*

3. Configure Nexus in Jenkins (Optional):

*Use **Nexus Artifact Uploader Plugin** in Jenkinsfile or freestyle jobs to push artifacts.*

Step 7: Example Jenkins Pipeline with SonarQube

*Create a new pipeline project with this sample **Jenkinsfile**:*

```
pipeline {
  agent any

  tools {
    maven 'Maven 3.8.5' // Configure Maven under Global Tools
  }

  environment {
    SONARQUBE = 'SonarQube' // Match the name in Jenkins config
  }

  stages {
    stage('Checkout') {
      steps {
        git 'https://github.com/your/repo.git'
      }
    }

    stage('Build') {
      steps {
        sh 'mvn clean install'
      }
```

```
        }

        stage('SonarQube Analysis') {
            steps {
                withSonarQubeEnv("${SONARQUBE}") {
                    sh 'mvn sonar:sonar'
                }
            }
        }
    }
}
```

Tips

- *Use named volumes to persist data.*
- *Use **docker-compose down** to stop services and **docker-compose up -d** to restart.*
- *You can also define networks explicitly if services need custom communication rules.*