

This project in Spring Boot revolves around managing UserDetails via CRUD operations. Additionally, we'll construct a complete CI/CD pipeline to automate the deployment process.

Steps:-

1. We Need 3 Aws Instance

Master Server :- Jenkins, Sonar-Server, Nexus Server

Kubernetes Master

Kubernetes Slave

2. We Need Some Devops Tools & Plugins

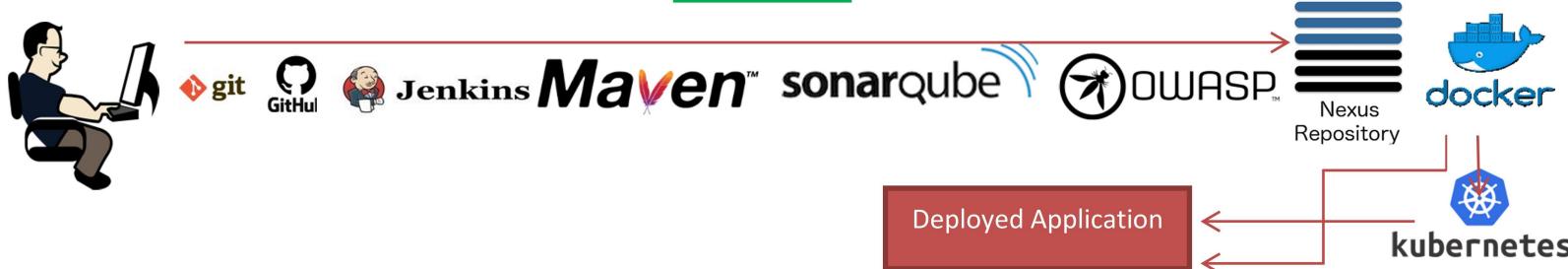
Tools

Git
GitHub
Jenkins
Maven
Sonar-Server
Nexus server
Docker
Kubernetes

Plugins

Owsap (for Dependency Check)
Sonar-Scanner (for code smell and bugs)
ConfigFileProvider (For configure Nexus)
Docker & DockerPipeline (for configure docker)
Kubernters Credential provider (for congigure Kubernetes)

3. FlowChart



Declarative: Tool Install	Git Checkout	Compile	sonar analysis	Package	Dependency check	Maven Deploy	Docker image build	Push Docker image	Docker user network	Docker mysql	Docker deploy
------------------------------	-----------------	---------	-------------------	---------	---------------------	-----------------	--------------------------	-------------------------	---------------------------	-----------------	------------------

- User write and push the code to GitHub.
- Jenkins fetch the code from Github then compile the code using Maven Tools.
- We will perform Sonar Analysis for Code Smell and checking bugs.
- We will Package the code by using maven Tools.
- We will use Owasp tools for Dependencies check .
- We will Nexus Repo to Keep our Artifact.
- We will Create a Dockerfile and build an from Dockerfile using Docker tool.
- Push Docker image to DockerHub Repo.
- Now, we will a Docker Network and attach to mysql container and myapp contaiener to have communication b/w them.
- After this process we will able to launch our application on internet successfully.

1. Create an Instance

2. Login to Jenkins server and Create a Pipeline Project named as :- BootCourse

The screenshot shows the AWS EC2 Instances page. The left sidebar has sections for EC2 Dashboard, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity, Reservations, Images (selected), AMIs, and AMI Catalog. The main area titled 'Instances (3) Info' lists three instances:

Name	Instance ID	Instance state	Instance type	Status check
Master K8s	i-082899dee8e74bac1	Stopped	t2.medium	-
Slave	i-0046b7c4a3eee9978	Stopped	t2.micro	-
Master	i-0c68683f860b4ee48	Running	t2.xlarge	2/2 checks passed

The screenshot shows the Jenkins dashboard. The left sidebar includes links for New Item, People, Build History, Manage Jenkins, My Views, Build Queue (No builds in the queue), and Build Executor Status (built-in-node (0 of 2 executors busy)). The main area displays the 'Build History' for the 'BootCourse' pipeline, which has a status of 'Last Success' (47 min ago #137) and 'Last Failure' (1 hr 7 min ago #135). Other pipelines listed are 'examples' and 'userapp'. At the bottom, there are links for Atom feed for all, Atom feed for failures, and Atom feed for just latest builds.

3. Install Needed Plugin and Configure tools:- (tools mentioned in flowchart)

SonarQube Scanner installations

SonarQube Scanner installations ^ Edited

Add SonarQube Scanner

SonarQube Scanner

Name: sonar-scanner

Install automatically ?

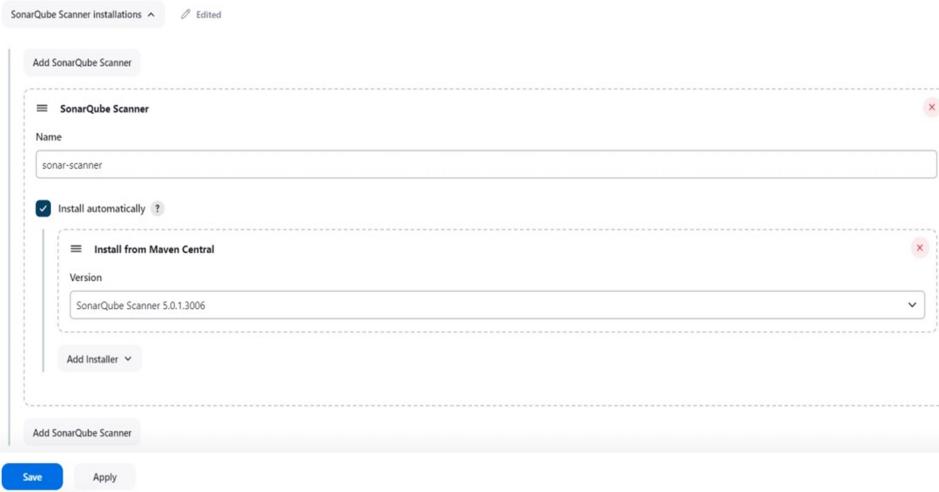
Install from Maven Central

Version: SonarQube Scanner 5.0.1.3006

Add Installer ▾

Add SonarQube Scanner

Save **Apply**



Maven installations

Maven installations ^ Edited

Add Maven

Maven

Name: My Maven

Install automatically ?

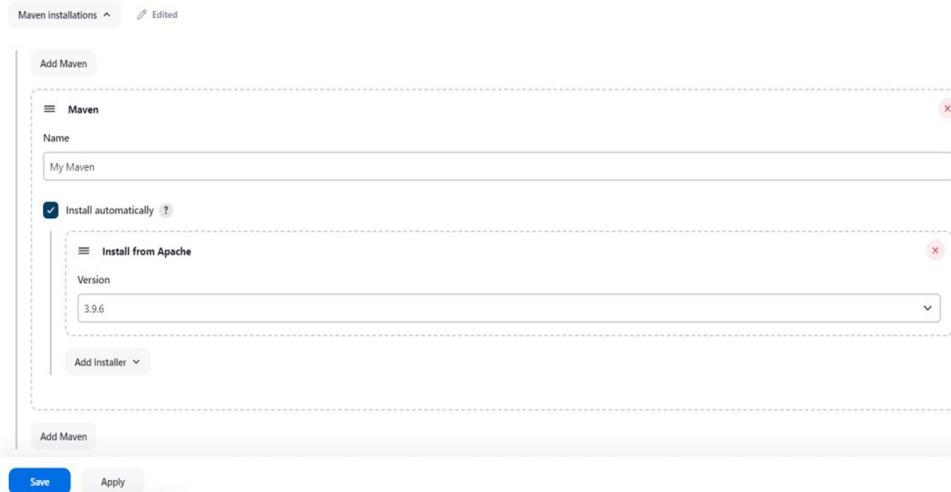
Install from Apache

Version: 3.9.6

Add Installer ▾

Add Maven

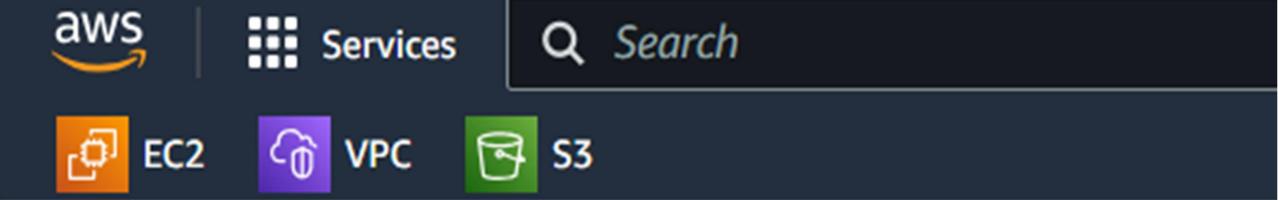
Save **Apply**



4. Create a Nexus and Sonar server using containerization on Jenkins server

```
034287f2e5d0 sonatype/nexus3      "/opt/sonatype/nexus..."  7 days ago      Up 3 hours      0.0.0.0:8081->8081/tcp, :::8081->8081/tcp
nexus
2583ee97207e sonarqube:lts-community  "/opt/sonarqube/dock..."  8 days ago      Up 3 hours      0.0.0.0:9000->9000/tcp, :::9000->9000/tcp
sonar
root@ip-172-31-17-44:~#
```

5. Create a network



```
aws | Services | Search
EC2 VPC S3

root@ip-172-31-17-44:~# docker network ls
NETWORK ID     NAME      DRIVER      SCOPE
7f8030e71e2d   bridge    bridge      local
c0fa551b5135   host      host       local
daac64f849e1   none      null       local
0d3bad5f5b34   usernetwork  bridge      local
root@ip-172-31-17-44:~#
```

6. Create Mysql container and App container and attached with created network (usernetwork)

```
root@ip-172-31-17-44:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
 NAMES
8b9918a2f5c2      rajatkumar01/users1.0:latest   "java -jar /app/app..."   About an hour ago   Up 59 minutes    0.0.0.0:9096->9096/tcp, :::9096->9096/tcp
user
9e44b4682fcf      mysql:latest           "docker-entrypoint.s..."   About an hour ago   Up About an hour   0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp
mysql
034287f2e5d0      sonatype/nexus3       "/opt/sonatype/nexus3..."  7 days ago        Up 3 hours       0.0.0.0:8081->8081/tcp, :::8081->8081/tcp
nexus
2583ee97207e      sonarqube:lts-community  "/opt/sonarqube/dock..."  8 days ago        Up 3 hours       0.0.0.0:9000->9000/tcp, :::9000->9000/tcp
sonar
root@ip-172-31-17-44:~#
```

7. Both Container is Up and running we can use Jenkins Public ip and port 9096 to launch app.



8. As our project is having only backend and database so we use Postman to send data to app.

```
{"id":2,"name":"RAUSHAN","email_id":"raushan@gmail.com","about":"Java Developer"}, {"id":3,"name":"RAJAT","email_id":"raajat.deveng@gmail.com","about":"Devops Engineer"}
```

The Postman interface shows the following details:

- Request Method: POST
- URL: http://54.205.226.99:9096/users/
- Body Type: JSON
- Body Content:

```
1  {
2   ...
3   ...
4 }
```
- Response Status: 200 OK
- Response Body:

```
1  {
2   ...
3   ...
4 }
```

9. To check Dependency check result we have to login sonar server.

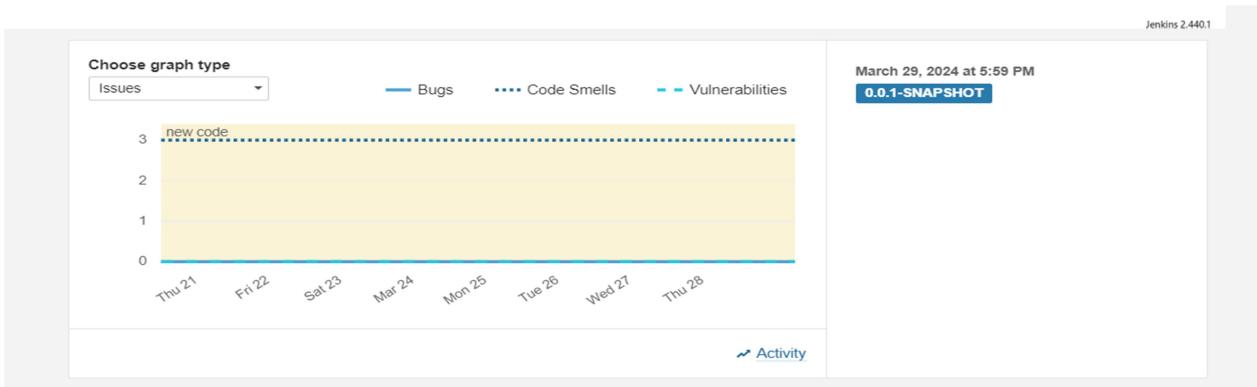
The screenshot shows the SonarQube dashboard for the project 'UserExample' in the 'main' branch. The 'QUALITY GATE STATUS' is 'Passed' with the message 'All conditions passed.' The 'MEASURES' section includes:

- New Code**: Since March 20, 2024, Started 8 days ago
- Bugs**: 0
- Vulnerabilities**: 0
- Security Hotspots**: 0
- Code Smells**: 3
- Debt**: 14min
- Maintainability**: A (green)
- Reliability**: A (green)
- Security**: A (green)
- Security Review**: A (green)

At the bottom, there are two circular progress bars: one at 0.0% and another at 0.0%.

The screenshot shows the Jenkins 'Dependency-Check Results' page. The left sidebar shows build-related links like Status, Changes, Console Output, Edit Build Information, Delete build #137, Git Build Data, and Dependency-Check (which is selected). The main content area is titled 'Dependency-Check Results' and shows the 'SEVERITY DISTRIBUTION' with a chart. Below the chart is a table of vulnerabilities:

File Name	Vulnerability	Severity	Weakness
↳ UserExample-0.0.1-SNAPSHOT.jar; angus-activation-2.0.1.jar	NVD CVE-2008-7271	Medium	CWE-79
↳ UserExample-0.0.1-SNAPSHOT.jar; angus-activation-2.0.1.jar	NVD CVE-2010-4647	Medium	CWE-79
↳ UserExample-0.0.1-SNAPSHOT.jar; angus-activation-2.0.1.jar	NVD CVE-2023-4218	Medium	CWE-611
↳ UserExample-0.0.1-SNAPSHOT.jar; jackson-core-2.15.4.jar	NVD CVE-2022-45688	High	CWE-787
↳ UserExample-0.0.1-SNAPSHOT.jar; jackson-core-2.15.4.jar	NVD CVE-2023-5072	High	CWE-770
↳ UserExample-0.0.1-SNAPSHOT.jar; jackson-databind-2.15.4.jar	NVD CVE-2023-35116	Medium	CWE-770
↳ UserExample-0.0.1-SNAPSHOT.jar; jakarta.activation-api-2.1.2.jar	NVD CVE-2008-7271	Medium	CWE-79
↳ UserExample-0.0.1-SNAPSHOT.jar; jakarta.activation-api-2.1.2.jar	NVD CVE-2010-4647	Medium	CWE-79
↳ UserExample-0.0.1-SNAPSHOT.jar; jakarta.activation-api-2.1.2.jar	NVD CVE-2023-4218	Medium	CWE-611



10. To check artifact on Nexus We will have to login on Nexus Server.

The screenshot shows the Sonatype Nexus Repository interface. The left sidebar has options: Welcome, Search, Browse (which is selected), and Upload. The main area shows a tree view under 'Browse / maven-snapshots'. At the top right are links for Jenkins and Sign out. A search bar at the top says 'Search components'. The tree view shows several artifact versions under 'com/user/example/UserExample/0.0.1-SNAPSHOT'. One version, '0.0.1-20240329122931-46', is expanded to show its contents: UserExample-0.0.1-20240329122931-46.jar, UserExample-0.0.1-20240329122931-46.jar.md5, UserExample-0.0.1-20240329122931-46.jar.sha1, UserExample-0.0.1-20240329122931-46.pom, UserExample-0.0.1-20240329122931-46.pom.md5, UserExample-0.0.1-20240329122931-46.pom.sha1, maven-metadata.xml, maven-metadata.xml.md5, and maven-metadata.xml.sha1. There is also a 'Delete folder' button.

Total Stage in this Pipeline :-

The screenshot shows the Jenkins Pipeline Step 'Restart from Stage' menu. On the left, there's a sidebar with various Jenkins navigation items like Status, Changes, Console Output, Edit Build Information, Delete build '#137', Git Build Data, Dependency-Check, Restart from Stage (which is highlighted in blue), Replay, Pipeline Steps, Workspaces, and Previous Build. On the right, the 'Restart #137 from Stage' section is displayed. It has a 'Stage Name' dropdown menu open, showing options: Git Checkout (selected), Git Checkout, Compile, sonar analysis, Package, Dependency check, Maven Deploy, Docker image build, Push Docker image, Docker user network, Docker mysql, and Docker deploy.

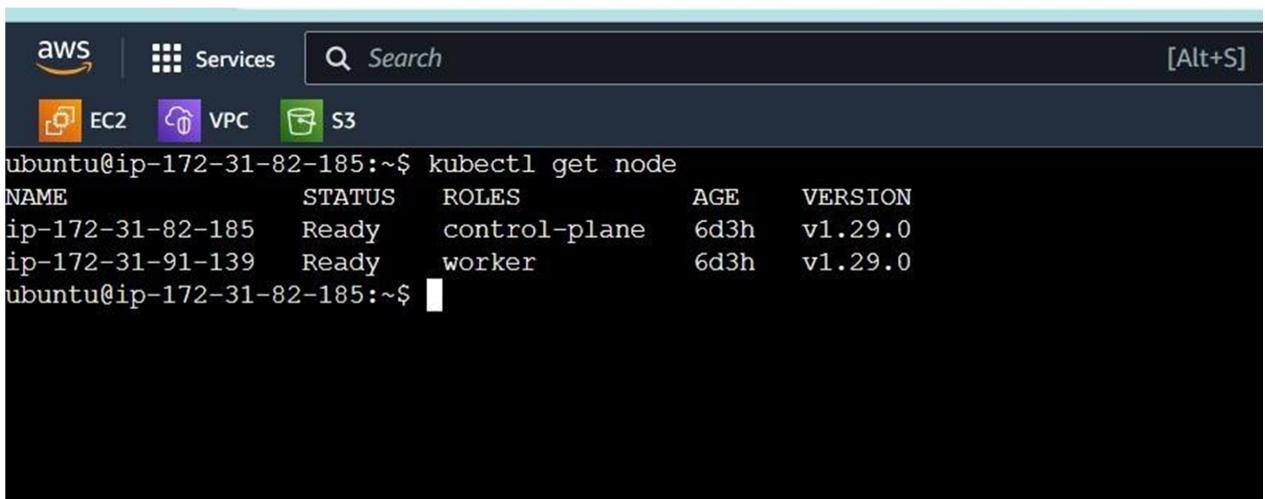
Deployment From Docker is Done.

Now we will deploy on k8s using Host Machine

- For k8s deployment we need k8s master and node Connected with kubeadm setup:
- We will use same docker image which is used in docker deployment.
- We will use yaml file for k8s deployment.

Steps:-

- We will verify Our k8s master and nodes is Ready or not.
- We will apply our yaml file on master.
- We will verify the status of pod weather it is running or not. If our backend and Database pod is running then our deployment is successfully deployed.
- Take Slave Ip and hit the url our project is up and running.



The screenshot shows a terminal window within the AWS Cloud9 IDE. The terminal output is as follows:

```
ubuntu@ip-172-31-82-185:~$ kubectl get node
NAME           STATUS    ROLES      AGE     VERSION
ip-172-31-82-185   Ready    control-plane   6d3h    v1.29.0
ip-172-31-91-139   Ready    worker       6d3h    v1.29.0
ubuntu@ip-172-31-82-185:~$ █
```

S master UserExample / k8s

RajatK01 Update UserExample-svc.yml 861ff09 · 2 days ago History

This branch is 66 commits ahead of, 15 commits behind Raushanray/UserExample:master.

Contribute Sync fork

Name	Last commit message	Last commit date
..		
UserExample-Pod.yml	k8s commit	last week
UserExample-deployment	k8s commit	last week
UserExample-svc.yml	Update UserExample-svc.yml	2 days ago
mysql-deployment.yml	Update mysql-deployment.yml	5 days ago
mysql-pv.yml	Update mysql-pv.yml	2 days ago
mysql-pvc.yml	k8s commit	last week
mysql-svc.yml	k8s commit	last week
storageclass.yaml	Add files via upload	5 days ago
storageclassclaim.yaml	Add files via upload	5 days ago

```
ubuntu@ip-172-31-28-210:~/UserExample/k8s$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
mysql-6c96f755db-fsqcz   1/1    Running   0          5d22h
userexample-5d8555997-jqj6g   1/1    Running   13 (17m ago)  55m
userexample-5d8555997-tnsjn   1/1    Running   13 (17m ago)  55m
userexample-pod      1/1    Running   13 (15m ago)  54m
ubuntu@ip-172-31-28-210:~/UserExample/k8s$
```

Pretty-print []

[]

Deployment Done And Working.