

SHRIRAM KARPOORA SUNDRA PANDIAN
DEVSECOPS
TOOLS: JENKINS, SONAR, SNYK, AWS, ZAP, DOCKER, KUBERNETES
INSTRUCTOR: SHIKHAR VERMA

SHRIRAM: I have created this document with step by step guide on mastering the devsecops and it's tools. I got a great instructor for this course and followed his guidance. I am sharing this document for educational purpose and other to learn devsecops and be better at cybersecurity.

DEVSECOPS

DevSecOps integrates security practices into this pipeline, ensuring that security is treated as a shared responsibility throughout the software development lifecycle. By embedding security checks and compliance validations early and frequently, DevSecOps reduces vulnerabilities and enhances application resilience.

CI/CD

A **CI/CD pipeline** (Continuous Integration/Continuous Deployment) is a set of automated processes that enable developers to integrate code changes, test them, and deploy applications seamlessly. It ensures rapid delivery of high-quality software by automating repetitive tasks such as testing, building, and deployment.

AWS

AWS provides an excellent platform for implementing CI/CD pipelines and DevSecOps due to its scalability, comprehensive suite of services, and deep integrations. AWS services like **CodePipeline**, **CodeBuild**, and **CodeDeploy** streamline pipeline automation, while **AWS Security Hub**, **GuardDuty**, and **IAM** offer robust security capabilities. Additionally, AWS's global infrastructure ensures high availability and low-latency deployments, making it a top choice for organizations looking to optimize their development and security workflows.

Enough introduction let's dirty our hands through work.

Starting with Maven, what is maven?

Maven is a build automation and project management tool primarily used in Java-based projects. It is part of the **Apache Software Foundation** and simplifies the process of managing project dependencies, building, packaging, and deploying applications. Maven uses an **XML file (pom.xml)** to describe the project's dependencies, build process, and configuration, enabling consistent and repeatable builds.

MAVEN

1. Kick start the amazon ec2 instance (Amazon Linux) and open using putty or any remote client handler you like.
2. Go to /opt folder and install java 11 and maven through the following tutorial.
3. Install Java 11 or any version of your choice.

<https://docs.aws.amazon.com/corretto/latest/corretto-11-ug/amazon-linux-install.html>

```
sudo yum install java-11-amazon-corretto-headless
```

Option 2: Install the headful Amazon Corretto 11:

```
sudo yum install java-11-amazon-corretto
```

Option 3: Install the JDK for Amazon Corretto 11 (Amazon Linux 2023 only):

```
sudo yum install java-11-amazon-corretto-devel
```

Option 4: Install the JMods for Amazon Corretto 11 (Amazon Linux 2023 only):

```
sudo yum install java-11-amazon-corretto-jmods
```

4. Next step install Maven by visiting the Maven website and download the binary source of tar.gz file. <https://maven.apache.org/download.cgi>

Files

Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow the [installation instructions](#). Use a source archive if you intend to build Maven yourself.

In order to guard against corrupted downloads/installations, it is highly recommended to [verify the signature](#) of the release bundles against the public [KEYS](#) used by the Apache Maven developers.

	Link	Checksums	Signature
Binary tar.gz archive	apache-maven-3.9.9-bin.tar.gz	apache-maven-3.9.9-bin.tar.gz.sha512	apache-maven-3.9.9-bin.tar.gz.asc
Binary zip archive	apache-maven-3.9.9-bin.zip	apache-maven-3.9.9-bin.zip.sha512	apache-maven-3.9.9-bin.zip.asc
Source tar.gz archive	apache-maven-3.9.9-src.tar.gz	apache-maven-3.9.9-src.tar.gz.sha512	apache-maven-3.9.9-src.tar.gz.asc
Source zip archive	apache-maven-3.9.9-erc.zip	apache-maven-3.9.9-erc.zip.sha512	apache-maven-3.9.9-erc.zip.asc

[▪ 3.9.9 Release Notes and Release Reference Documentation](#)
[▪ latest source code from source repository](#)
[▪ Distributed under the Apache License, version 2.0](#)
[▪ other:](#)
[▪ All current release sources \(plugins, shared libraries,...\) available at <https://downloads.apache.org/maven/>](#)

5. Extract the zip using tar -xvf “filename” and extract the folder.
6. Add it to the path through this command “export
PATH=\$PATH:/opt/apache-maven(filename)/bin
7. Once java 11 and maven is successfully installed go to the root folder again.
8. We are going to use the following command for maven utilization which compiles and takes care of the project.
 - `validate` - validate the project is correct and all necessary information is available
 - `compile` - compile the source code of the project
 - `test` - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
 - `package` - take the compiled code and package it in its distributable format, such as a JAR.
 - `verify` - run any checks on results of integration tests to ensure quality criteria are met
 - `install` - install the package into the local repository, for use as a dependency in other projects locally
 - `deploy` - done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.
9. In the root folder take any java project and clone it here using git clone “project.git” command.
10. Once you get the repository. Navigate inside the directory using the cd command and type “maven validate” to start the maven or initialize it on the git folder.
11. Then type “maven package” which does all three actions at once (validate, compile, and test). Once you are done with the command you can see the target folder.

```
[root@ip-172-31-81-176 springboot-hello]# ls -ltr
total 16
drwxr-xr-x. 3 root root 18 Nov 16 19:04 src
-rw-r--r--. 1 root root 731 Nov 16 19:04 azure-pipelines.yml
-rw-r--r--. 1 root root 325 Nov 16 19:04 README.md
-rw-r--r--. 1 root root 95 Nov 16 19:04 Dockerfile
-rw-r--r--. 1 root root 1596 Nov 16 19:18 pom.xml
drwxr-xr-x. 6 root root 159 Nov 16 19:18 target
```

12. Go inside the target folder and you can see .gz file has been created which can be used for the application or be can call it as the application build.
13. Go back to the root directory using cd ~ and find the hidden directories through “ls -ltr” and navigate to .m2/repository and you can the dependencies for the maven project.

```
[root@ip-172-31-81-176 .m2]# cd repository
[root@ip-172-31-81-176 repository]# ls
aopalliance           ch           commons-cli      io          log4j
asm                   classworlds   commons-io       javax       net
backport-util-concurrent com         commons-logging junit      org
```

14. Whatever you make changes to the code in the future to get the fresh and clean bundle from maven simply run “maven clean package”.

SAST (Static Application Security Testing)

SonarQube and Sonarcloud

SonarQube is an open-source platform used for **continuous code quality inspection** and **static code analysis**. It helps developers and teams improve code quality and maintainability by detecting code smells, bugs, security vulnerabilities, and technical debt. SonarQube supports a wide range of programming languages and integrates seamlessly with CI/CD pipelines to automate code reviews.

To Test the source code in the early stage of the process while the continuous integration happens. We can say simply we will test the static code before it pushed into deployment.

Before going deep into the SAST lets start the jenkins server which is the heart of the CI Integration part.

Go to AWS and start the red hat linux with t2.medium for better speed and update the system using dnf update.

```

Authenticating with public key "Imported-OpenSSH-Key"
  • Mobatxt Personal Edition v24.3 +
    (SSH client, X server and network tools)

> SSH session to ec2-user@84.89.55
  • Direct SSH : ✓
  • X11-forwarding : ✓
  • SSH-browser : ✓
  • X11-forwarding : ✘ (disabled or not supported by server)

> For more info, ctrl+click on help or visit our website.

Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.hf/insights-dashboard
[ec2-user@ip-172-31-84-17 ~]$ dnf update
Updating Subscription Management repositories.
Unable to read consumer identity.

This system is not registered with an entitlement server. You can use "rhc" or "subscription-manager" to register.

Red Hat Enterprise Linux 9 for x86_64 - AppStream from RHUI (RPMs)
Red Hat Enterprise Linux 9 for x86_64 - BaseOS from RHUI (RPMs)
Red Hat Enterprise Linux 9 Client Configuration
Dependencies resolved.
=====
Package           Architecture   Version          Repository      Size
=====
Installing:
  Kernel          x86_64        5.14.0-503.14.1.el9_5   rhel-9-baseos-rhui-rpms  2.0 M
  Kernel-core     x86_64        5.14.0-503.14.1.el9_5   rhel-9-baseos-rhui-rpms  18 K
  Kernel-modules  x86_64        5.14.0-503.14.1.el9_5   rhel-9-baseos-rhui-rpms  37 N
  Kernel-modules-core x86_64        5.14.0-503.14.1.el9_5   rhel-9-baseos-rhui-rpms  31 M
Upgrading:
  NetworkManager  x86_64        1:1.48.10-2.el9_5      rhel-9-baseos-rhui-rpms  2.3 M
  NetworkManager-cloud-setup x86_64        1:1.48.10-2.el9_5      rhel-9-appstream-rhui-rpms 76 K
  NetworkManager-libnm x86_64        1:1.48.10-2.el9_5      rhel-9-baseos-rhui-rpms  1.1 M
  NetworkManager-team x86_64        1:1.48.10-2.el9_5      rhel-9-baseos-rhui-rpms  41 K
  NetworkManager-tui  x86_64        1:1.48.10-2.el9_5      rhel-9-baseos-rhui-rpms  250 K
  alternatives     x86_64        1.24-1.el9_5.1       rhel-9-baseos-rhui-rpms  42 K
  audit           x86_64        3.1.5-1.el9         rhel-9-baseos-rhui-rpms  738 K
  audit-libs      x86_64        3.1.5-1.el9         rhel-9-baseos-rhui-rpms  124 K
  bziputils        x86_64        2.35-2-54.el9       rhel-9-baseos-rhui-rpms  4.6 M
  bzip2-gold       x86_64        2.35-2-54.el9       rhel-9-baseos-rhui-rpms  733 K
  ca-certificates noarch        2024.2.09.v8.0.303-91.4.el9_4  rhel-9-baseos-rhui-rpms  1.0 M
  ciknconfig      x86_64        1.24-1.el9_5.1       rhel-9-baseos-rhui-rpms  182 K
  curl             x86_64        4.5-1.el9            rhel-9-baseos-rhui-rpms  362 K
  curl             x86_64        23.4-19.el9         rhel-9-appstream-rhui-rpms 1.3 M
  curl             x86_64        8.32-36.el9         rhel-9-baseos-rhui-rpms  1.2 M
  curl             x86_64        8.32-36.el9         rhel-9-baseos-rhui-rpms  2.0 M
  crypto-common   noarch        20240828-2.git626aa59.el9_5  rhel-9-baseos-rhui-rpms  90 K
  crypto-polices  noarch        20240828-2.git626aa59.el9_5  rhel-9-baseos-rhui-rpms  101 K
  crypto-polices-scripts x86_64        2.7-2.el9            rhel-9-baseos-rhui-rpms  52 K
  cryptsetup-libs x86_64        7.7.1-1.31.el9       rhel-9-baseos-rhui-rpms  257 K
  curl             x86_64        9:1.0.0-198.2.el9    rhel-9-baseos-rhui-rpms  143 K
  device-mapper   x86_64        9:1.0.0-198.2.el9    rhel-9-baseos-rhui-rpms  179 K
  device-mapper-libs x86_64        11:1.0.1.el9         rhel-9-baseos-rhui-rpms  106 K
  readline         x86_64        8.1.1-1.el9          rhel-9-baseos-rhui-rpms  10 K

```

- Install maven like above steps and time to install jenkins
- Go to this folder on linux where you can see all the repositories where you can install the jenkins repository.

```

[root@ip-172-31-84-17 opt]# cd /etc/yum.repos.d
[root@ip-172-31-84-17 yum.repos.d]# ls -ltr
total 36
-rw-r--r-- 1 root root 6088 Jun  7 11:44 redhat-rhui-eus.repo.disabled
-rw-r--r-- 1 root root 4723 Jun  7 11:44 redhat-rhui-beta.repo.disabled
-rw-r--r-- 1 root root 5792 Nov 16 20:12 redhat-rhui.repo.rpmsave
-rw-r--r-- 1 root root 5896 Nov 16 20:16 redhat-rhui.repo
-rw-r--r-- 1 root root 467 Nov 16 20:16 redhat-rhui-client-config.repo
[root@ip-172-31-84-17 yum.repos.d]#

```

- <https://pkg.jenkins.io/redhat-stable/> Navigate to this site and install the first two commands.

Jenkins Redhat Packages

To use this repository, run the following command:

```

sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key

```

```

[root@ip-172-31-84-17 yum.repos.d]# wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
2024-11-17 00:59:05.000 2024-11-17 00:59:05.000 [root@ip-172-31-84-17 yum.repos.d]# wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
Connecting to pkg.jenkins.io (146.75.34.133):443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 85
Saving to: '/etc/yum.repos.d/jenkins.repo'

/etc/yum.repos.d/jenkins.repo      100%[=====]  2.17 MB/s   in 0s

[root@ip-172-31-84-17 yum.repos.d]# rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
[root@ip-172-31-84-17 yum.repos.d]#

```

- Now let's install the jenkins by entering this command: "dnf install jenkins"

```
[root@ip-172-31-84-17 yum.repos.d]# dnf install jenkins
Last metadata expiration check: 0:00:00 ago on Sun Nov 17 01:03:01 UTC 2024.
Dependencies resolved.
Preparing... #################################################
Package           Architecture Version       Repository      Size
jenkins          noarch     2.479.1-1.1.jenkins 91 M
Transaction Summary
Install 1 Package
Total download size: 91 M
Installed size: 92 M
Is this ok [y/N]: y
Downloading Packages:
jenkins-2.479.1-1.1.noarch.rpm
Total
Running transaction check
transaction check succeeded.
Running transaction test
[  0.0% ] 42 MB/s | 91 MB 00:02
[ 100.0% ] 42 MB/s | 91 MB 00:02
[  0.0% ]
```

What is Jenkins?

Jenkins is an open-source automation server used for **continuous integration (CI)** and **continuous delivery (CD)** in software development. It enables developers to automate the building, testing, and deployment of applications, ensuring a faster and more reliable software delivery process. Jenkins is highly extensible with hundreds of plugins to integrate with various tools and technologies in the software development lifecycle.

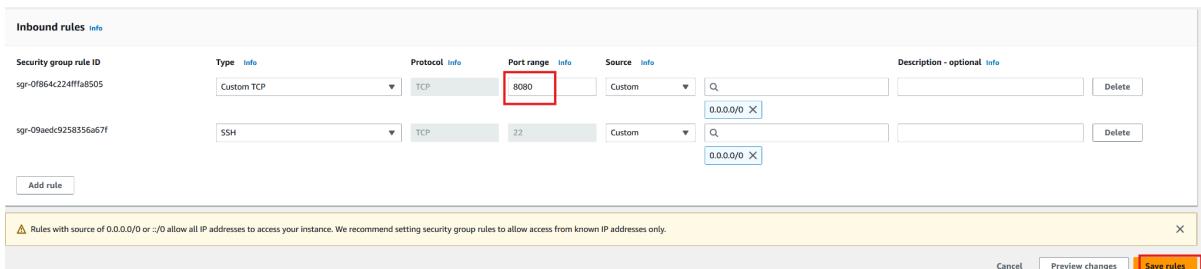
- Start jenkins using the command “`systemctl start jenkins`”.

```
[root@ip-172-31-84-17 yum.repos.d]# systemctl start jenkins
[root@ip-172-31-84-17 yum.repos.d]# systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; disabled; preset: disabled)
   Active: active (running) since Sun 2024-11-17 01:03:01 UTC; 42s ago
     Main PID: 56945 (java)
        Tasks: 51 (llimit: 22900)
       Memory: 617.4M
          CPU: 15.578s
         CGroup: /system.slice/jenkins.service
                 └─56945 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

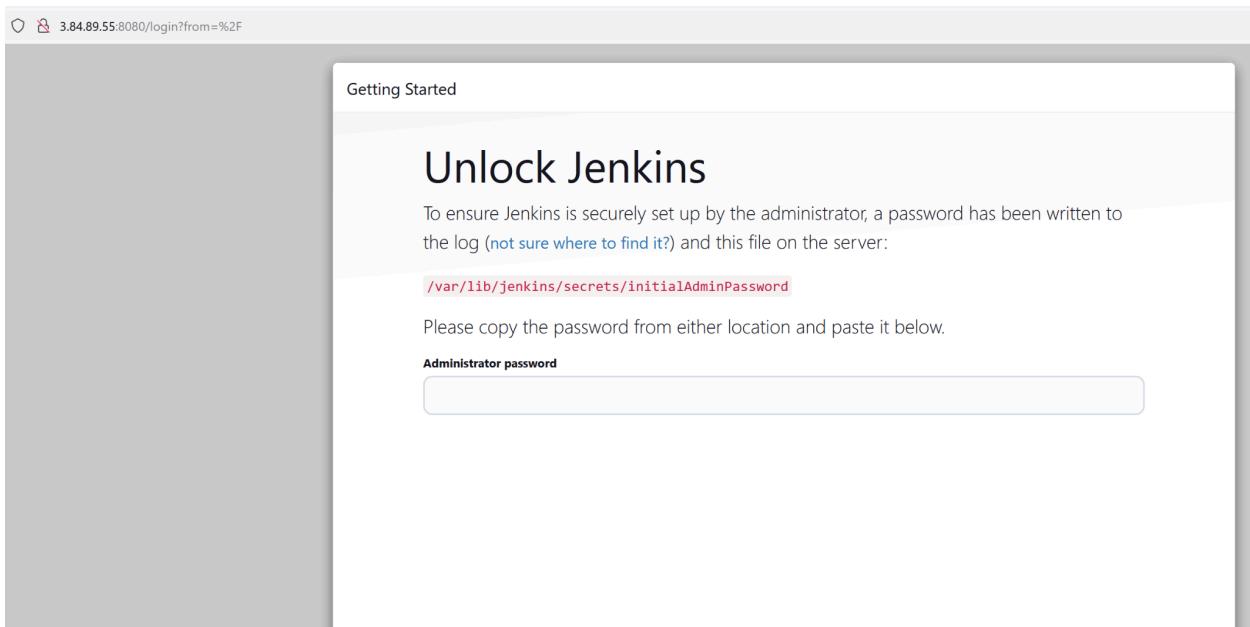
Nov 17 01:02:58 ip-172-31-84-17.ec2.internal jenkins[56945]: 169ff95a1ef14f639a3aff9ecfa329b3
Nov 17 01:02:58 ip-172-31-84-17.ec2.internal jenkins[56945]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Nov 17 01:02:58 ip-172-31-84-17.ec2.internal jenkins[56945]: ****
Nov 17 01:03:01 ip-172-31-84-17.ec2.internal jenkins[56945]: 2024-11-17 01:03:01.230+0000 [id=30]      INFO  jenkins.InitReactorRunner$!#onAttained: Completed initialization
Nov 17 01:03:01 ip-172-31-84-17.ec2.internal jenkins[56945]: 2024-11-17 01:03:01.273+0000 [id=23]      INFO  hudson.lifecycle.LifeCycle#onReady: Jenkins is fully up and running
Nov 17 01:03:01 ip-172-31-84-17.ec2.internal jenkins[56945]: System has started Jenkins Continuous Integration Server.
Nov 17 01:03:01 ip-172-31-84-17.ec2.internal jenkins[56945]: 2024-11-17 01:03:01.392+0000 [id=48]      INFO  h.m.DownloadService#Downloadable#load: Obtained the updated data file for hudson.
Nov 17 01:03:01 ip-172-31-84-17.ec2.internal jenkins[56945]: 2024-11-17 01:03:01.393+0000 [id=48]      INFO  hudson.util.Retry#start: Performed the action check updates server successfully
[root@ip-172-31-84-17 yum.repos.d]# systemctl enable jenkins
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /usr/lib/systemd/system/jenkins.service.
[root@ip-172-31-84-17 yum.repos.d]#
```

Make sure you enable the jenkins as well. (We use enable to make sure the service starts automatically when it boots up by creating symbolic links).

- We need to make sure that port 8080 is running properly so that jenkins can run. In aws configure inbound rules for your ec2 instance to allow port 8080.



- At port 8080 with your public ip you should see the webpage of jenkins.



- As above it mentioned that password is on the following path, navigate to that path and copy the password to unlock.

```
[root@ip-172-31-84-17 ~]# cat /var/lib/jenkins/secrets/initialAdminPassword  
169ff95a1ef14f639a3aff9ecfa329b3
```

- Install the suggested plugins.

Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Getting Started

Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding	** Ionicons API Folders OWASP Markup Formatter ** ASM API ** JSON Path API ** Structs ** Pipeline: Step API ** Token Macro Build Timeout ** Recyclable API ** Credentials ** Plain Credentials ** Variant ** SSH Credentials Credentials Binding ** SCM API ** Pipeline: API ** org.jenkinslang3 v3.x Jenkins API Timestamper ** Cafefine API ** Script Security ** JavaBeans Activation Framework (JAF) API ** JAXB ** SnakeYAML API ** JSON API ** Jackson 2 API ** commons-text API ** Pipeline: Supporting APIs ** plugin Utilities API
✓ Timestamper	⌚ Workspace Cleanup	⌚ Ant	⌚ Gradle	
⌚ Pipeline	⌚ GitHub Branch Source	⌚ Pipeline: GitHub Groovy Libraries	⌚ Pipeline Graph View	
⌚ Git	⌚ SSH Build Agents	⌚ Matrix Authorization Strategy	⌚ PAM Authentication	
⌚ LDAP	⌚ Email Extension	⌚ Mailer	⌚ Dark Theme	

- Add your details for jenkins and save & continue.

Getting Started

Create First Admin User

Username

Password

Confirm password

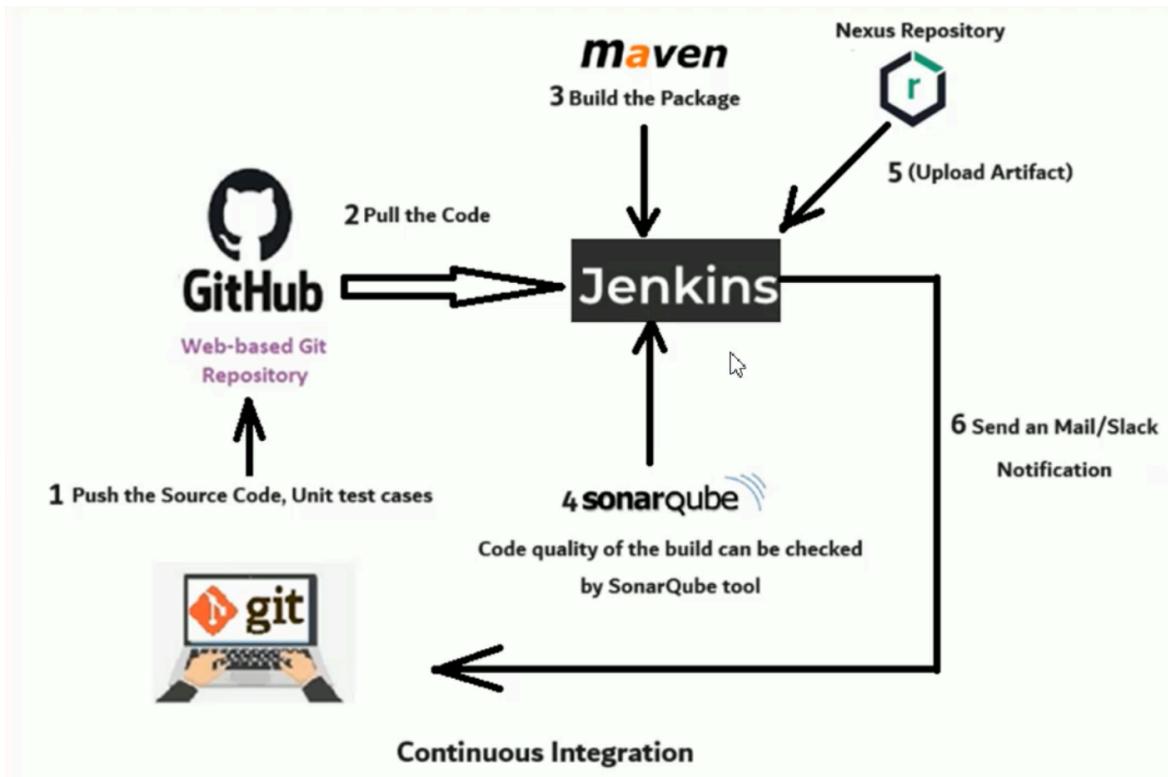
Full name

E-mail address

Jenkins 2.479.1

[Skip and continue as admin](#) [Save and Continue](#)

- Now it's time to integrate maven, sonarqube and everything with jenkins like shown in the diagram.



Maven Integration

- Let's integrate maven with jenkins.
- Click on manage jenkins.

Dashboard >

+ New Item

Build History

Manage Jenkins

My Views

Build Queue

No builds in the queue.

Build Executor Status

0/2

- Now click on tools.

The screenshot shows the Jenkins 'Manage Jenkins' interface. At the top, there is a warning message: "Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation." Below this, there are several sections:

- System Configuration** (with a dropdown arrow):
 - System**: Configure global settings and paths.
 - Clouds**: Add, remove, and configure cloud instances to provision agents on-demand.
- Tools** (highlighted with a red box): Configure tools, their locations and automatic installers.
- Appearance**: Configure the look and feel of Jenkins.
- Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Nodes**: Add, remove, and configure Jenkins nodes.
- Security**:
 - Security**: Secure Jenkins; define who is allowed to access/use the system.
 - Credentials**: Configure credentials.
 - Credential Providers**: Configure the credential providers and types.
 - Users**: Create/delete Jenkins users.
- Status Information**:
 - System Information**: Displays various environmental information to assist trouble-shooting.
 - System Log**: System log captures output from java.util.logging output related to Jenkins.
 - Load Statistics**: Check your resource utilization and see if you need more computers for your builds.
 - About Jenkins**: See the version and build information.

- Go down and add maven.

Maven installations

Add Maven

Save

Apply

- Give any name and remove the install maven option as we already installed it and give the maven path by checking “mvn –version” on the virtual machine.

The screenshot shows the 'Add Maven' dialog. It has a 'Name' field containing 'maven-3.9.9' and a 'MAVEN_HOME' field containing '/opt/apache-maven-3.9.9'. A checkbox for 'install automatically' is present but unchecked. At the bottom, there is a 'Save' button.

```
[root@ip-172-31-84-17 ~]# mvn --version
Apache Maven 3.9.9 (8e8579a9e76f7d015ee5ec7bfcd97d260186937)
Maven home: /opt/apache-maven-3.9.9
Java version: 17.0.13, vendor: Red Hat, Inc., runtime: /usr/lib/jvm/java-17-openjdk-17.0.13.0.11-4.el9.x86_64
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "5.14.0-427.20.1.el9_4.x86_64", arch: "amd64", family: "unix"
```

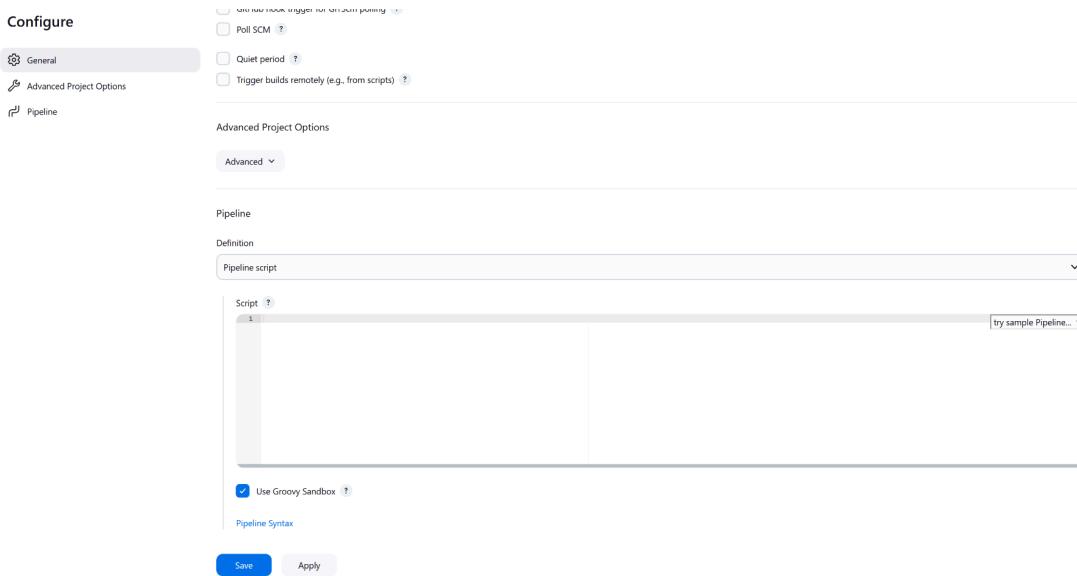
- This is the important step now lets create a declarative pipeline where we are going to write a script about on importing the git project for verifying it's status.
- Click on the new item on the left most part of the screen.

The screenshot shows the Jenkins 'Manage Jenkins' page. At the top left, there is a button labeled '+ New Item' which is highlighted with a red box. To its right, the page title 'Manage Jenkins' is displayed. Below the title, there is a message: 'Building on the built-in node can be a security issue. You should set up distribute'. Under the 'System Configuration' heading, there are two sections: 'System' (with a gear icon) and 'Clouds' (with a cloud icon). Both sections have edit icons to their right. Below these sections is a 'Security' link.

- Give any meaningful name and select pipeline and click ok.

The screenshot shows the 'New Item' dialog box. In the 'Enter an item name' field, the text 'maven-jenkins-git-pipeline' is entered. In the 'Select an item type' section, the 'Pipeline' option is selected and highlighted with a red box. Other options listed include 'Freestyle project', 'Multi-configuration project', 'Folder', 'Multibranch Pipeline', and 'Organization Folder'. At the bottom of the dialog is an 'OK' button.

- Go down and look for the script window.



- Enter this script for declaring the github repository that has java code.

Script ?

```

1  pipeline {
2      agent any
3      tools {
4          maven 'maven-3.9.9'
5      }
6      stages {
7          stage ("Checking out the git project") {
8              steps {
9                  git 'https://github.com/Shikhar82/springboot-maven-micro.git'
10             }
11         }
12     }
13   }
14 }
```

```

pipeline {
    agent any
    tools {
        maven 'maven-3.9.9'
    }
    stages {
        stage ("Checking out the git project") {
            steps {
                git
```

```

        'https://github.com/Shikhar82/springboot-maven-micro.git'
    }
}
}

}

```

- Click on build now after saving the script for executing it.

The screenshot shows the Jenkins pipeline configuration page for a pipeline named 'maven-jenkins-git-pipeline'. The page includes a sidebar with options like Status, Changes, Build Now (which is highlighted with a red box), Configure, Delete Pipeline, Stages, Rename, and Pipeline Syntax. Below the sidebar is a 'Builds' section which is currently empty.

- Once you finish executing it, you will see the log success.

The screenshot shows the Jenkins console output for a pipeline run. The output starts with the command 'git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/maven-jenkins-git-pipeline/.git # timeout=10'. It then fetches changes from the remote Git repository using 'git config remote.origin.url https://github.com/Shikhar82/springboot-maven-micro.git # timeout=10'. It performs a 'git fetch -tags -force --progress -- https://github.com/Shikhar82/springboot-maven-micro.git +refs/heads/*:refs/remotes/origin/* # timeout=10'. Finally, it checks out the master branch with 'git checkout -b master 288ad7abfb45340e5f44e0ba326dc257f02f1b3c # timeout=10'. A commit message 'Create ansible.yaml' is shown, followed by the message 'First time build. Skipping changelog.'

- Add the clean package script on the pipeline script like below for adding that feature.

Script ?

```

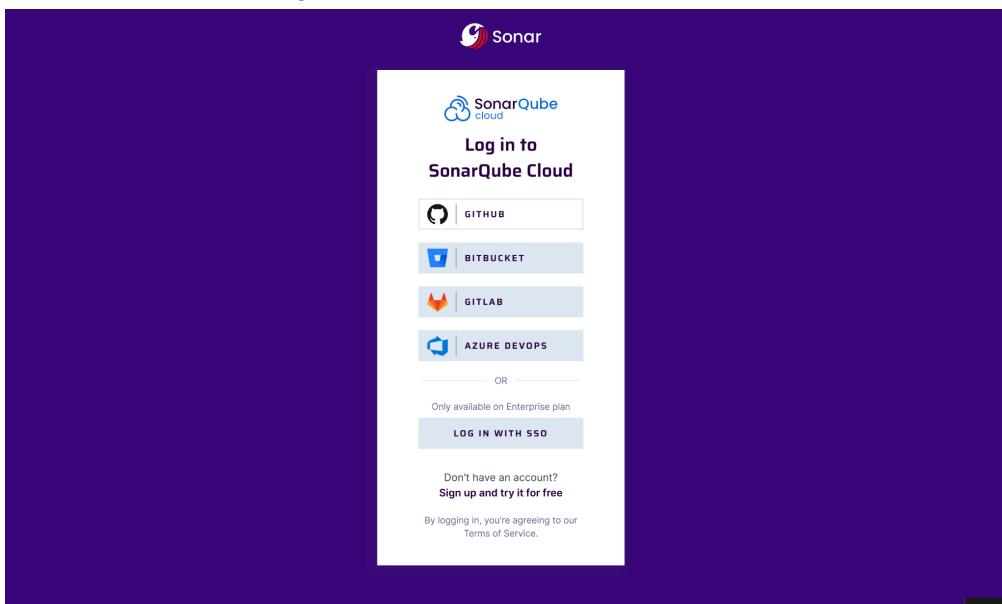
1  pipeline {
2      agent any
3      tools {
4          maven 'maven-3.9.9'
5      }
6      stages {
7          stage ("Checking out the git project") {
8              steps {
9                  git 'https://github.com/Shikhar82/springboot-maven-micro.git'
10             }
11         }
12
13         stage ("Building the package") {
14             steps {
15                 sh 'mvn clean package'
16             }
17         }
18     }

```

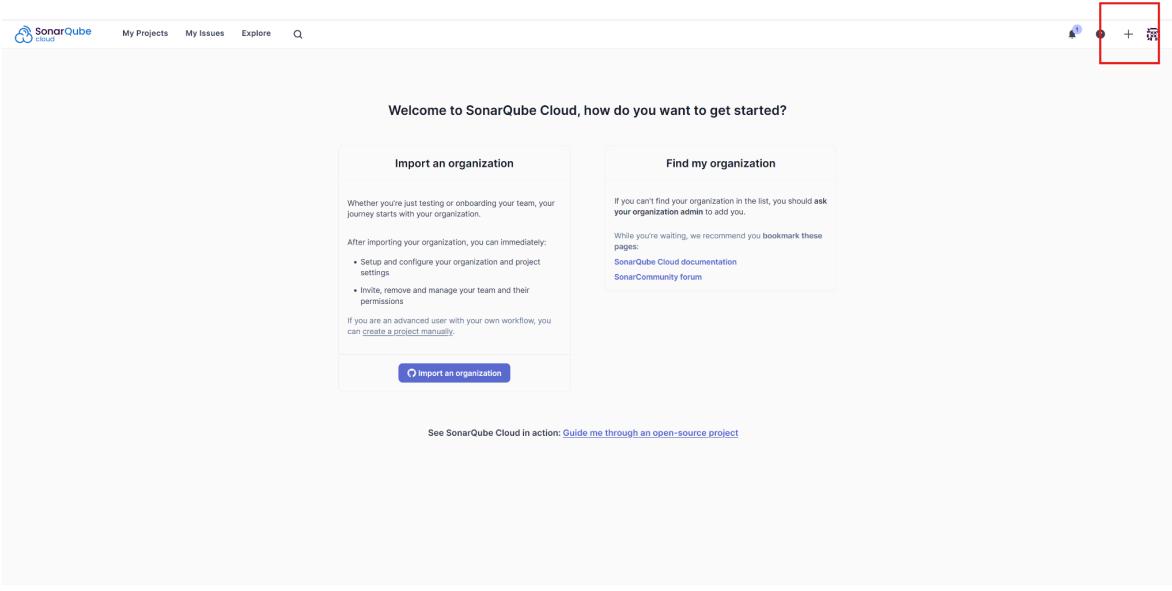
- Now try build again and check whether everything working fine.
- We successfully integrated maven to create the project, but we still haven't integrated the SAST tool for checking vulnerabilities on our code. Let's integrate SAST with jenkins.

SAST Sonarcloud Integration

- Go to sonarcloud website and login using any credentials you have
<https://sonarcloud.io/login>.



- Click on + button on the top right to create you own organization.



- Select create your own.

Create an organization

Organizations enable your team to collaborate across many projects.

Import from a DevOps platform

GitHub

No organization to import? You can [create one manually](#)

Manual setup is not recommended, and leads to missing features like appropriate setup of your project or analysis feedback in the Pull Request.

- Select any name you like for your organization and click on free plan and create your organization.

1 Enter your organization details

Name *

Up to 255 characters

Key * ?

Organization key must start with a lowercase letter or number, followed by lowercase letters, numbers or hyphens, and must end with a letter or number. Maximum length: 255 characters.

> Add additional info

2 Choose a plan

Free

For individual developers, students and open-source projects

\$0

Select Free

Team

Free trial available

Essential capabilities for small teams and business

Starting at

~~\$64~~ \$32/month • 50% off

Start Free Trial

Enterprise

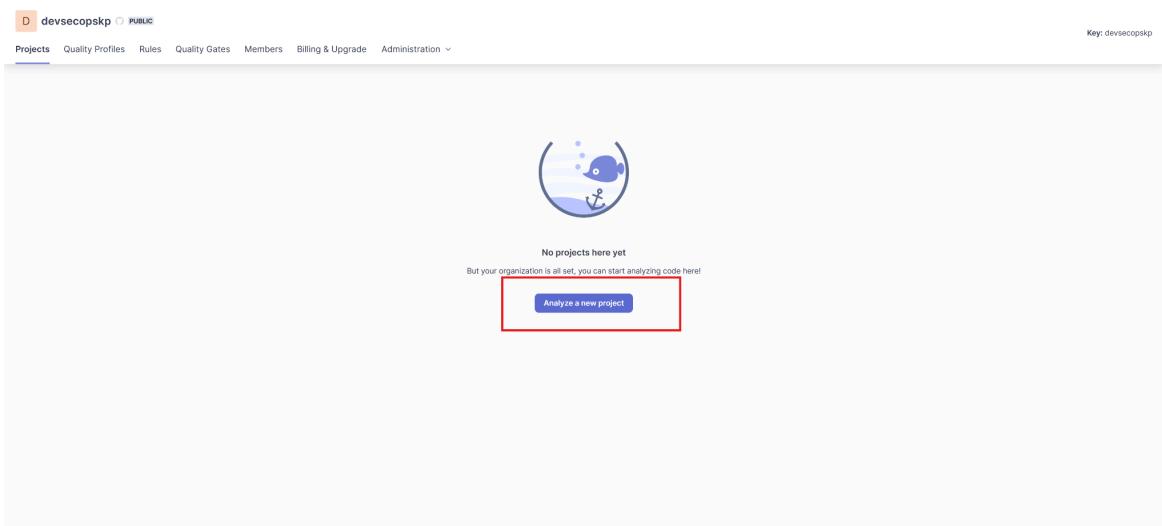
Mission critical flexibility, scalability, and performance.

To get started

Talk to sales

Contact sales

- After entering the organization windows, give analyze new project.



Analyze projects

Manual setup is not recommended, and leads to missing features like appropriate setup of your project or analysis feedback in the Pull Request. We recommend to [import your projects](#)

Organization

devsecopskp

[Create another organization](#)

Display Name * ?

devsecopskp

Up to 255 characters

Project Key * ?

devsecopskp

Up to 400 characters. All letters, digits, dash, underscore, period or colon.

Project visibility *

Public

Anyone will be able to browse your source code and see the result of your analysis.

Private (Upgrade to unlock Private visibility)

Only members of the organization will be able to browse your source code and see the result of your analysis.

[Next](#)

Upgrade to scan private projects 50% off

Starting from ~~\$64~~ \$32 / month with the Team plan.

- ✓ 14-day free trial, cancel any time
- ✓ Private projects
- ✓ Analyze feature branches, maintenance branches, & pull requests
- ✓ Define the quality standard for your team
- ✓ Synchronized user management

[Upgrade devsecopskp](#) [View FAQs](#)

Already have a coupon?

If you've already paid for your plan and were provided with a coupon, [apply it directly here](#).

Set up project for Clean as You Code

The new code definition sets which part of your code will be considered new code.

This helps you focus attention on the most recent changes to your project, enabling you to follow the Clean as You Code methodology.

Learn more: [New Code Definition](#)

Set a new code definition for your organisation to use it by default for all new projects

- ⓘ This can help you use the Clean as You Code methodology consistently across projects.
[devsecopskp - Administration - New Code](#)

The new code for this project will be based on:

Previous version

Any code that has changed since the previous version is considered new code.

Recommended for projects following regular versions or releases.

Number of days

Any code that has changed in the last x days is considered new code. If no action is taken on a new issue after x days, this issue will become part of the overall code.

Recommended for projects following continuous delivery.

ⓘ You can change this at any time in the project administration

[Back](#) [Create project](#)

- After creating a new project.
- Before integrating sonarcloud with our jenkins pipeline we need to generate a token which is used to authenticate the jenkins server on using sonarcloud.

- Go to sonarcloud and click on myaccount>security and click on the token name and generate one.

The screenshot shows the SonarCloud 'Security' page under the 'kp18-cpu' account. At the top, there are tabs for Profile, Security, Notifications, Organizations, and Appearance. The 'Security' tab is selected. Below it, there's a 'Generate Tokens' section with a 'Generate Token' button and a message indicating a new token has been created. The newly created token is displayed as a long string of characters. To the right, there's a sidebar titled 'Upgrade to scan private projects' with a list of benefits and a 'View FAQs' link. Below the token creation message, there's a table for 'Existing Tokens' with one entry: 'jenkins_auth' which was created on November 16, 2024, and has never been used. There's also a 'Revoke' button next to the token name.

Keep the token safe, now lets go back to jenkins server and add script for integrating the sonarcloud.

- This is the script we are going to add.

```
mvn clean verify sonar:sonar -Dsonar.projectKey=devsecopskp
-Dsonar.organization=devsecopskp -Dsonar.host.url=https://sonarcloud.io
-Dsonar.token=d0d2af0eee7f1fdb21e937360084b8ece5a62cb4
```

- Script looks like this. (Make sure you remove clean package and add this for the time being)

The screenshot shows a Jenkins pipeline script. It starts with a 'pipeline' block containing an 'agent any' directive. Inside the pipeline, there's a 'tools' block with a 'maven' entry. The script then defines two stages: 'Checking out the git project' and 'Checking bugs on sonarcloud'. The first stage uses the 'git' step with the URL 'https://github.com/Shikhar82/springboot-maven-micro.git'. The second stage uses the 'sh' step to run the command 'mvn clean verify sonar:sonar -Dsonar.projectKey=devsecopskp -Dsonar.organization=devsecopskp -Dsonar.host.url=https://sonarcloud.io -Dsonar.token=d0d2af0eee7f1fdb21e937360084b8ece5a62cb4'.

```
1 - pipeline {
2   agent any
3   tools {
4     maven 'maven-3.9.9'
5   }
6   stages {
7     stage ("Checking out the git project") {
8       steps {
9         git 'https://github.com/Shikhar82/springboot-maven-micro.git'
10      }
11    }
12    stage ("Checking bugs on sonarcloud") {
13      steps {
14        sh 'mvn clean verify sonar:sonar -Dsonar.projectKey=devsecopskp -Dsonar.organization=devsecopskp -Dsonar.host.url=https://sonarcloud.io -Dsonar.token=d0d2af0eee7f1fdb21e937360084b8ece5a62cb4'
15      }
16    }
17  }
18 }
```

- After build is success we can go to sonarcloud and my projet to see the results.

The screenshot shows the SonarCloud web interface. On the left, there's a sidebar with filters for Quality Gate (Passed: 0, Failed: 0), Reliability (A: 1, B: 0, C: 0, D: 0, E: 0), Security (A: 0, B: 0, C: 0, D: 0, E: 0), and Security Review (A: 1, B: 0, C: 0, D: 0, E: 0). The main area displays a single project: **devsecopspk / springboot-maven-course-micro-svc**. It shows the last analysis was on 11/16/2024 at 9:23 PM, with 71 Lines of Code in XML and Java. The overall status is NEW PUBLIC. Below this, there are metrics: Security (A: 0), Reliability (A: 0), Maintainability (A: 1), Hotspots Reviewed (100%), Coverage (0.0%), and Duplications (0.0%). A note says "Not computed". At the bottom, it says "1 of 1 shown".

- Let's do something fun, go this github and copy the url.
<https://github.com/Shikhar82/devsecops-jenkins-k8s-tf-sast-sca-sonarcloud-snyk-repo.git>
- Go to configuration and click on pipeline syntax.

The screenshot shows the Jenkins Pipeline configuration screen. Under the "Definition" section, the "Pipeline script" tab is selected. The script content is as follows:

```

4   maven 'maven-3.9.9'
5
6   stages {
7     stage ("Checking out the git project") {
8       steps {
9         git branch: 'main', url: 'https://github.com/Shikhar82/devsecops-jenkins-k8s-tf-sast-sca-sonarcloud-snyk-repo.git'
10      }
11    }
12
13    stage ("Checking bugs on sonarcloud") {
14      steps {
15        sh 'mvn clean verify sonar:sonar -Dsonar.projectKey=devsecopspk -Dsonar.organization=devsecopspk -Dsonar.host.url=https://sonarcloud.io -Dsonar.token=d0d2af0eee7f1fdb21e937360084b8ece'
16      }
17    }
18  }
19
20 }

```

Below the script, there is a checkbox labeled "Use Groovy Sandbox" with a question mark icon. A red box highlights the "Pipeline Syntax" button. At the bottom, there are "Save" and "Apply" buttons.

- Click on git:Git and give the git url with mentioned on branch as main.

Overview

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Script**, and you will see a Pipeline Script configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

Steps

Sample Step

archiveArtifacts: Archive the artifacts

archiveArtifacts: Archive the artifacts
bat: Windows Batch Script
build: Build a job
catchError: Catch error and set build result to failure
checkout: Check out from version control
cleanWs: Delete workspace when done
deleteDir: Recursively delete the current directory from the workspace
dir: Change current directory
echo: Print Message
emailext: Extended Email
emailextrcipients: Extended Email Recipients
error: Error signal
fileExists: Verify if file exists in workspace
findBuildScans: Find published build scans
fingerprint: Record fingerprints of files to track usage
git: Git
input: Wait for interactive input
isUnix: Checks if running on a Unix-like node
junit: Archive JUnit-formatted test results
library: Load a library on the fly
libraryResource: Load a resource file from a library
load: Evaluate a Groovy source file into the Pipeline script

Sample Step

git: Git

git ?

Repository URL ?

Branch ?

Credentials ?

+ Add

Include in polling? ?
 Include in changelog? ?

Generate Pipeline Script

```
git branch: 'main', url: 'https://github.com/Shikhar82/devsecops-jenkins-k8s-tf-sast-sca-sonarcloud-snyk-repo.git'
```

- Copy the script from generate pipeline and replace on the script.

```

4      maven 'maven-3.9.9'
5  }
6  }
7  stages {
8    stage ("Checking out the git project") {
9      steps {
10        git branch: 'main', url: 'https://github.com/Shikhar82/devsecops-jenkins-k8s-tf-sast-sca-sonarcloud-snkyk-repo.git'
11      }
12    }
13    stage ("Checking bugs on sonarcloud") {
14      steps {
15        sh 'mvn clean verify sonar:sonar -Dsonar.projectKey=devsecopskp -Dsonar.organization=devsecopskp -Dsonar.host.url=https://sonarcloud.io -Dsonar.token=d0d2af0eee7f1fdb21e93736008'
16      }
17    }
18  }
19 }
20

```

- Click on build, once the build is success go to sonarcloud for finding the vulnerabilities.

devsecopskp / Techstartsnykproject NEW PUBLIC Passed

Last analysis: 11/16/2024, 9:30 PM • 5.6k Lines of Code • Java, JSP, ...

E 52	E 47	A 139	E 0.0%	O 0.0%	D 6.6%
Security	Reliability	Maintainability	Hotspots Reviewed	Coverage	Duplications

- We can see vulnerabilities and click on the result for further analysis.

My Projects My Issues Explore Q

devsecopskp > Techstartsnykproject > master ✓

Summary Issues Security Hotspots Measures Code Activity

Filters

- Clean Code Attribute**
 - Consistency 59
 - Intentionality 103
 - Adaptability 75
 - Responsibility 1
- Software Quality**
 - Security 52
 - Reliability 47
 - Maintainability 139
- Severity**
 - Blocker 0
 - High 110
 - Medium 63
 - Low 65
 - Info 0
- Type**
- Status**
- Security Category**
- Creation Date**

Issues 238 issues 5d 5h effort

src.../t246osslab/easybuggy/core/dao/DBClient.java

src.../t246osslab/easybuggy/core/dao/EmbeddedADS.java

Intentionality: Complete the task associated to this "TODO" comment. (cwe +) 0min effort • 5 minutes ago

Intentionality: Remove this commented out code. (unused +) 5min effort • 5 minutes ago

Intentionality: Remove this commented out code. (unused +) 5min effort • 5 minutes ago

Intentionality: Remove this commented out code. (unused +) 5min effort • 5 minutes ago

Adaptability: Define a constant instead of duplicating this literal "[false, " "]* 4 times. (design +) 10min effort • 2 years ago

Adaptability: Define a constant instead of duplicating this literal "[true, " "]* 10 times. (design +) 22min effort • 2 years ago

SonarQUBE (Own server instead of sonarcloud)

Create your server for static analysis

- Already we have our jenkins server running.

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with options like Dashboard, EC2 Global View, Events, Instances (selected), Images, and Capacity Reservations. The main area displays a table of instances. One instance is listed: 'jenkins_server' (Instance ID: i-0a5909ea09225fa95). It is marked as 'Running' and has an 't2.medium' instance type. A status check indicates '2/2 checks passed'. Below the table, the instance details are shown, including its name and ID. At the bottom, there are tabs for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags.

- Now it's time to run our sonarqube, so create a t2.medium ubuntu 22 ec2 instance for our server.



- Once you are in the ubuntu machine using any SSH client i prefer mobaxterm. Update the system.

```
root@ip-172-31-81-238:/home/ubuntu# apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 B]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2152 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [368 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [17.9 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [2636 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [456 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 c-n-f Metadata [612 B]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1135 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [266 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metadata [26.4 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [43.3 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse Translation-en [10.8 kB]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 c-n-f Metadata [440 B]
```

- Time to install java on the system using the command in the screenshot.

```
root@ip-172-31-81-238:/home/ubuntu# apt install openjdk-11-jdk-headless -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  alsatopology-conf alsau-ucm-conf ca-certificates-java fontconfig-config fonts-dejavu-core java-common libasound2
  libasound2-data libavahi-client3 libavahi-common-data libavahi-common3 libcurl2 libfontconfig1 libgraphite2-3 libharfbuzz0b
  libjpeg-turbo8 libjpeg8 liblcms2-2 libpcslite1 openjdk-11-jre-headless
Suggested packages:
  default-jre libasound2-plugins alsau-utils cups-common liblcms2-utils pcscd openjdk-11-demo openjdk-11-source libnss-mdns
  fonts-dejavu-extra fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei | fonts-wqy-zenhei fonts-indic
The following NEW packages will be installed:
  alsatopology-conf alsau-ucm-conf ca-certificates-java fontconfig-config fonts-dejavu-core java-common libasound2
  libasound2-data libavahi-client3 libavahi-common-data libavahi-common3 libcurl2 libfontconfig1 libgraphite2-3 libharfbuzz0b
  libjpeg-turbo8 libjpeg8 liblcms2-2 libpcslite1 openjdk-11-jdk-headless openjdk-11-jre-headless
0 upgraded, 21 newly installed, 0 to remove and 16 not upgraded.
Need to get 119 MB of archives.
After this operation, 268 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 alsatopology-conf all 1.2.5.1-2 [15.5 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 libasound2-data all 1.2.6.1-1ubuntu1 [19.1 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 libasound2 amd64 1.2.6.1-1ubuntu1 [390 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 alsau-ucm-conf all 1.2.6.3-1ubuntu1.11 [43.6 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 java-common all 0.72build2 [6782 B]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libavahi-common-data amd64 0.8-5ubuntu5.2 [23.8 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libavahi-common3 amd64 0.8-5ubuntu5.2 [23.9 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libavahi-client3 amd64 0.8-5ubuntu5.2 [28.0 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libcurl2 amd64 2.4.10p1-1ubuntu4.11 [263 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 fonts-dejavu-core all 2.37-2build1 [1041 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 fontconfig-config all 2.12.1-4.2ubuntu5 [20.1 kB]
root@ip-172-31-81-238:/home/ubuntu# java --version
openjdk 11.0.25 2024-10-15
OpenJDK Runtime Environment (build 11.0.25+9-post-Ubuntu-1ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.25+9-post-Ubuntu-1ubuntu122.04, mixed mode, sharing)
root@ip-172-31-81-238:/home/ubuntu#
```

- Go to binaries.sonarsource.com > distribution > sonarqube > select version 9.3.0 and copy link address.

0	sonarlint-daemon/
0	sonarlint-eclipse-parent/
0	sonarlint-language-server/
0	sonarqube-ant-task/
0	sonarqube/
0	sslr-flex-toolkit/
0	sslr-python-toolkit/
0	test-sonar-xml-parsing/
0	to-remove.org.sonarlint.eclipse.site/
0	xml/
Z 18.3 kB	SonarQube.MSBuild.Runner-0.9.zip
Z 0.5 kB	SonarQube.MSBuild.Runner-0.9.zip

- Navigate to opt and wget to download the package of sonarqube.

```
root@ip-172-31-81-238:/home/ubuntu# cd /opt
root@ip-172-31-81-238:/opt# wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.3.0.51899.zip
--2024-11-18 16:23:59-- https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.3.0.51899.zip
Resolving binaries.sonarsource.com (binaries.sonarsource.com)... 99.84.188.21, 99.84.188.45, 99.84.188.35, ...
Connecting to binaries.sonarsource.com (binaries.sonarsource.com)|99.84.188.21|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 276912542 (264M) [application/zip]
Saving to: 'sonarqube-9.3.0.51899.zip'

sonarqube-9.3.0.51899.zip      51%[=====] 135.72M 27.3MB/s eta 5s
```

- Unzip the package of sonarqube and navigate to the /bin directory and /Linux directory as we use Ubuntu here.
- An important thing to understand is we are unable to start the sonarqube as a root, even if you try to start it, it will stop eventually.

```
root@ip-172-31-81-238:/opt/sonarqube-9.3.0.51899/bin/linux-x86-64# sh sonar.sh start
Starting SonarQube...
Started SonarQube.
root@ip-172-31-81-238:/opt/sonarqube-9.3.0.51899/bin/linux-x86-64# sh sonar.sh status
SonarQube is not running.
```

- Edit the sonarqube file sonar.sh and add run as user = ubuntu.

```
GNU nano 6.2                                     sonar.sh *
# Default values for the Application variables, below.
#
# NOTE: The build for specific applications may override this during the resource-copying
# phase, to fill in a concrete name and avoid the use of the defaults specified here.
DEF_APP_NAME="SonarQube"
DEF_APP_LONG_NAME="SonarQube"

# Application
APP_NAME="${DEF_APP_NAME}"
APP_LONG_NAME="${DEF_APP_LONG_NAME}"

# Wrapper
WRAPPER_CMD="./wrapper"
WRAPPER_CONF="../conf/wrapper.conf"
SHUTDOWNER_LIB_DIR=".../lib"

# Priority at which to run the wrapper. See "man nice" for valid priorities.
# nice is only used if a priority is specified.
PRIORITY=

# Location of the pid file.
PIDDIR="."

# If uncommented, causes the Wrapper to be shutdown using an anchor file.
# When launched with the 'start' command, it will also ignore all INT and
# TERM signals.
#IGNORE_SIGNALS=true

# If specified, the Wrapper will be run as the specified user.
# IMPORTANT - Make sure that the user has the required privileges to write
# the PID file and wrapper.log files. Failure to be able to write the log
# file will cause the Wrapper to exit without any way to write out an error
# message.
# NOTE - This will set the user which is used to run the Wrapper as well as
# the JVM and is not useful in situations where a privileged resource or
# port needs to be allocated prior to the user being changed.
RUN_AS_USER=ubuntu

# The following two lines are used by the chkconfig command. Change as is
# appropriate for your application. They should remain commented.
# chkconfig: 2345 20 80
# description: Test Wrapper Sample Application

# Do not modify anything beyond this point
#-----
# Get the fully qualified path to the script
case $0 in
  /etc/init.d/sonar*) . /etc/init.d/functions; exec $0 "$@";;

  *) . /etc/sonar/sonar.sh; exec $0 "$@";;

esac
```

Save and exit.

- Go back to opt directory and change the ownership to Ubuntu and check whether it's updated.

```

root@ip-172-31-81-238:/opt# chown ubuntu:ubuntu sonarqube-9.3.0.51899 -R
root@ip-172-31-81-238:/opt# ls -ltr
total 270428
drwxr-xr-x 11 ubuntu ubuntu 4096 Jan 28 2022 sonarqube-9.3.0.51899
-rw-r--r-- 1 root root 276912542 Feb 16 2022 sonarqube-9.3.0.51899.zip
root@ip-172-31-81-238:/opt# █

```

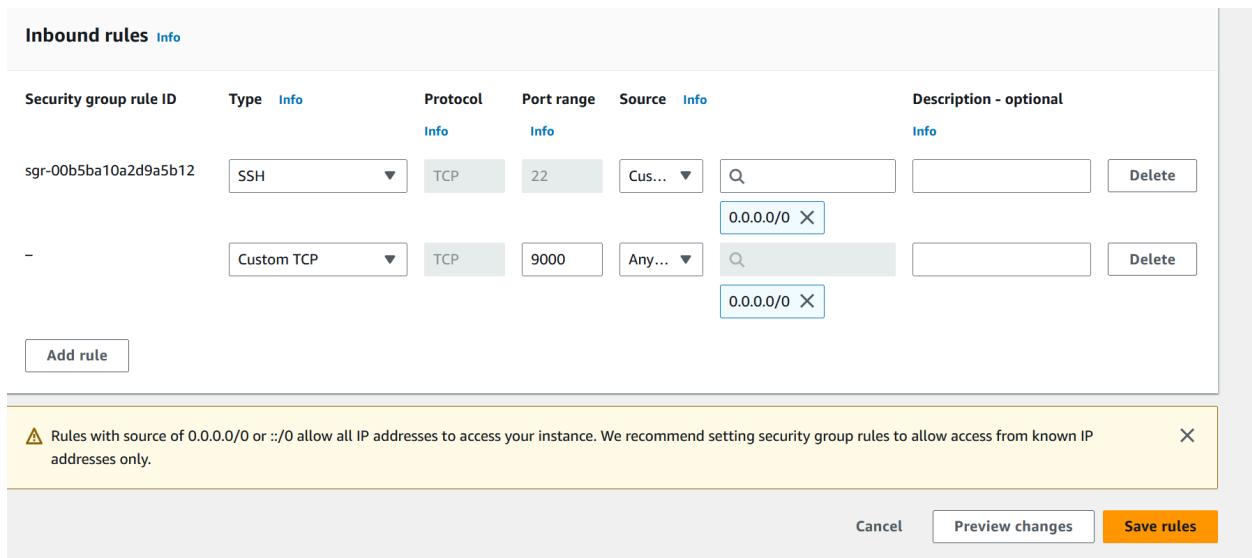
- Navigate to sonar.sh folder and start the sonarqube as you change the ownership to ubuntu.

```

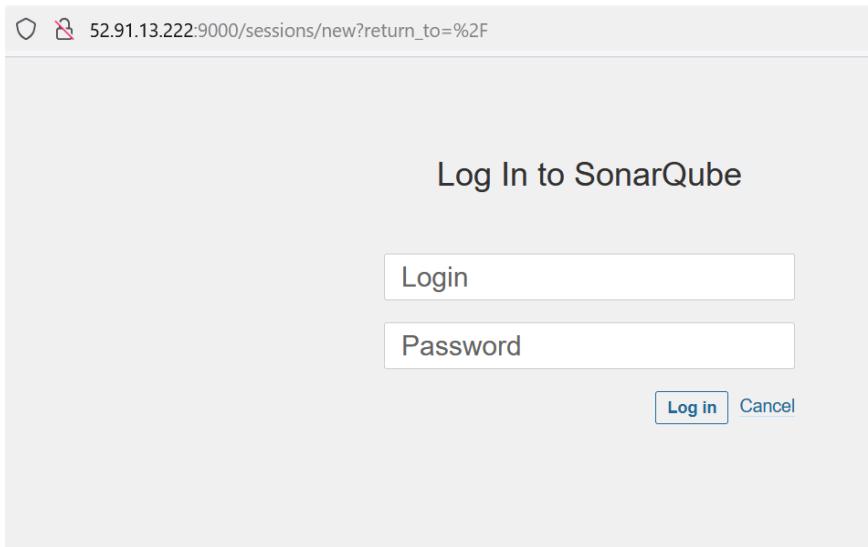
drwxr-xr-x 11 ubuntu ubuntu 4096 Jan 28 2022 sonarqube-9.3.0.51899
-rw-r--r-- 1 root root 276912542 Feb 16 2022 sonarqube-9.3.0.51899.zip
root@ip-172-31-81-238:/opt# cd sonarqube-9.3.0.51899
root@ip-172-31-81-238:/opt/sonarqube-9.3.0.51899# ls
COPYING bin conf data dependency-license.json elasticsearch extensions lib logs temp web
root@ip-172-31-81-238:/opt/sonarqube-9.3.0.51899# cd bin
root@ip-172-31-81-238:/opt/sonarqube-9.3.0.51899/bin# ls
jsw-license linux-x86-64 macosx-universal-64 windows-x86-64
root@ip-172-31-81-238:/opt/sonarqube-9.3.0.51899/bin# cd linux-x86-64/
root@ip-172-31-81-238:/opt/sonarqube-9.3.0.51899/bin/linux-x86-64# ls
lib sonar.sh wrapper
root@ip-172-31-81-238:/opt/sonarqube-9.3.0.51899/bin/linux-x86-64# sh sonar.sh start
Starting SonarQube...
Started SonarQube.
root@ip-172-31-81-238:/opt/sonarqube-9.3.0.51899/bin/linux-x86-64# █

```

- Now let's access sonarqube on the website. Sonarqube uses port 9000, so go to aws and edit the inbound rules and add port 9000.



- Open the URL with port 9000 in the browser and see the login page, default password: admin:admin and change password for your convenience.



- This is how the dashboard looks like.

A screenshot of the SonarQube dashboard. At the top, there is a question: "How do you want to create your project?". Below this, a note says: "Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)? Create your project from your favorite DevOps platform. First, you need to set up a DevOps platform configuration." There are four cards below, each representing a different platform: "From Azure DevOps" (with a blue icon), "From Bitbucket" (with a blue icon), "From GitHub" (with a black icon), and "From GitLab" (with an orange icon). Each card has a "Set up global configuration" button underneath. At the bottom, there is another section with the text: "Are you just testing or have an advanced use-case? Create a project manually." followed by a "Manually" button with a double-angle bracket icon.

- Install the sonarqube scanner plugin before the integration.

The screenshot shows the Jenkins Plugins page. The search bar at the top right contains the text "sonar". Below the search bar, there are tabs for "Available plugins" (which is selected), "Installed plugins", and "Advanced settings". A sidebar on the left lists "Updates", "Available plugins" (selected), "Installed plugins", "Advanced settings", and "Download progress". The main table lists two plugins:

Install	Name	Released
<input checked="" type="checkbox"/>	SonarQube Scanner 2.17.2 External Site/Tool Integrations Build Reports	9 mo 3 days ago
<input type="checkbox"/>	Sonar Quality Gates 328.vf4369b_da_d3c2 Library plugins (for use by other plugins) analysis Other Post-Build Actions	1 mo 1 day ago

A tooltip for the SonarQube Scanner plugin states: "This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality."

- We need to create a authentication token so that jenkins can communication with sonar.

The screenshot shows the SonarQube Administration - Tokens page. The top navigation bar includes links for Rules, Quality Profiles, Quality Gates, Administration, and a search bar. The user is identified as "Administrator". The main content area is titled "Tokens" and contains the following information:

If you want to enforce security by not providing credentials of a real SonarQube user to run your code scan or to invoke web services, you can provide a User Token as a replacement of the user login. This will increase the security of your installation by not letting your analysis user's password going through your network.

Generate Tokens

Enter Token Name Generate

New token "sonartoken" has been created. Make sure you copy it now, you won't be able to see it again!

Copy 87704732a8260314f0d9de22a1fe030e2f1a8aa

Name	Last use	Created
sonartoken	Never	November 18, 2024

Revoke

- Go back to manage jenkins > credentials > Global credentials and click on add credentials.

Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
This credential domain is empty. How about adding some credentials?			

Icon: S M L

- Select secret text and give the token at
New credentials

Kind

Username with password

Username with password

GitHub App

SSH Username with private key

Secret file

Secret text

Certificate

Treat username as secret ?

Password ?

Create

New credentials

Kind

Secret text

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Secret

.....

ID ?

sonartoken

Description ?

This is the security token for integrating sonar server

Create

Global credentials (unrestricted)

+ Add Credentials

Credentials that should be available irrespective of domain specification to requirements matching.

ID

Name

Kind

Description



sonartoken

This is the security token for integrating sonar server

Secret text

This is the security token for integrating sonar server



Icon: S M L

- Now let's add the sonar server on jenkins system.

+ New Item

Build History

Manage Jenkins

My Views

Manage Jenkins

Search settings /

Building on the built-in node can be a security issue. You should set up distributed builds. [Set up agent](#) [Set up cloud](#) [Dismiss](#)

Build Queue

No builds in the queue.

Build Executor Status

System Configuration



System

Configure global settings and paths.



Tools

Configure tools, their locations and automatic installers.



Plugins

Add, remove, disable or enable plugins that can extend the functionality of Jenkins.



Nodes

Add, remove, control and monitor the various nodes that Jenkins runs jobs on.



Clouds

Add, remove, and configure cloud instances to provision agents on-demand.



Appearance

Configure the look and feel of Jenkins

The screenshot shows the Jenkins System configuration page under Manage Jenkins > System. It includes sections for Disable deferred wipeout on this node, Disk Space Monitoring Thresholds, Environment variables, and Tool Locations. Below these is a SonarQube servers section with an Environment variables checkbox. A SonarQube installations section lists one entry and has an Add SonarQube button.

- Give any name, I have given sonar9 because we are using version 9 and give server url and select the token we gave at credentials.

The screenshot shows the SonarQube installation configuration dialog. It has fields for Name (sonar9), Server URL (http://52.91.13.222:9000), and Server authentication token (a placeholder text area). Buttons for Save and Apply are at the bottom.

SonarQube installations
List of SonarQube installations

Name
sonar9

Server URL
Default is http://localhost:9000
http://52.91.13.222:9000/

Server authentication token
SonarQube authentication token. Mandatory when anonymous access is disabled.
This is the security token for integrating sonar server

+ Add

Save Apply

Save and apply.

- Now let's create the pipeline for completion of our sonar pipeline.

- Create a new pipeline for this one specially by selecting new item.

+ New Item

Manage Jenkins

 Build History

 Manage Jenkins

 My Views

Building on the built-in node can be
See [the documentation](#).

Build Queue

No builds in the queue.

Build Executor Status

0/2

System Configuration



System

Configure global settings



Plugins

Add remove disable or enable

Enter an item name

sonarserver-jenkins

Select an item type



Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.



Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK

Click Pipeline above.

- Go to this url for getting the script which we will use in the pipeline code.
<https://www.jenkins.io/doc/pipeline/steps/sonar/>

Script ?

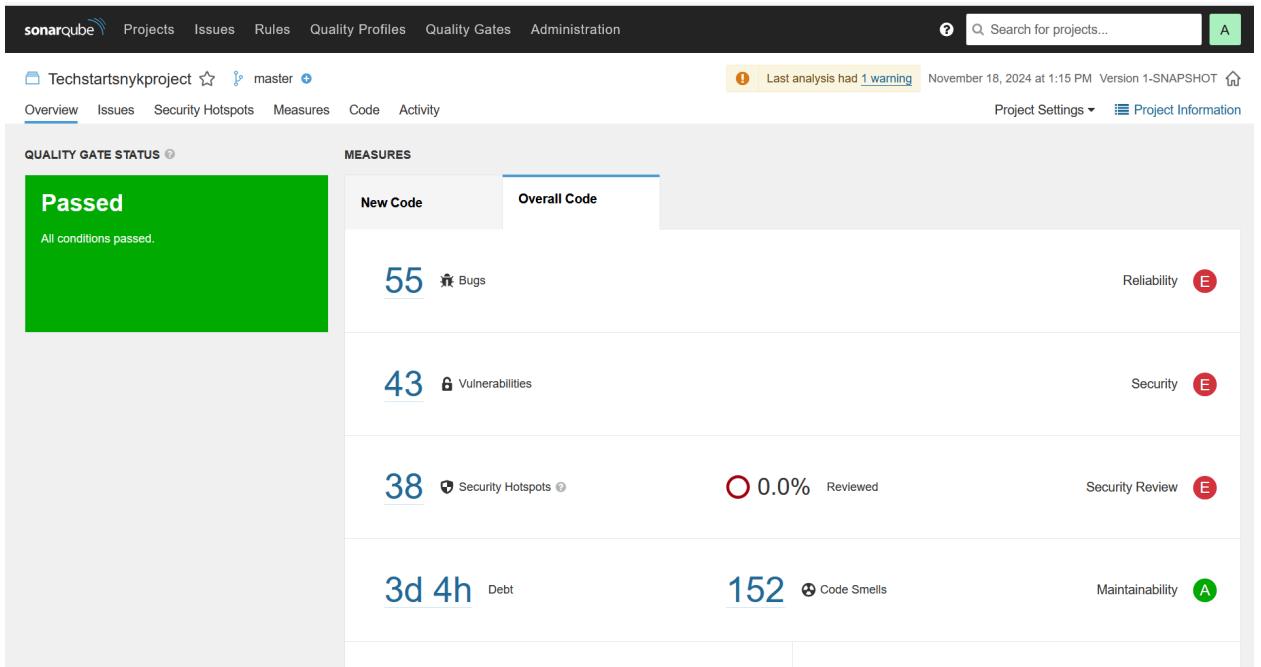
```

1 pipeline {
2     agent any
3     tools {
4         maven 'maven-3.9.9'
5     }
6     stages {
7         stage ("Checking out the git project") {
8             steps {
9                 git branch: 'main', url: 'https://github.com/Shikhar82/devsecops-jenkins-k8s-tf-sast-sca-sonarcloud'
10            }
11        }
12        stage ("Build the package") {
13            steps {
14                sh 'mvn clean package'
15            }
16        }
17        stage ("Integration of sonarcloud server on 9000") {
18            steps {
19                withSonarQubeEnv(installationName: 'sonar9', credentialsId: 'sonartoken') {
20                    sh 'mvn sonar:sonar'
21                }
22            }
23        }
24    }
25 }
26 }
27 }
28 }
```

try sample Pipeline... ▾

Here for installationName is the name you configured for sonar on jenkins system and credentialsID is the name that you used for configuring the sonarqube token on jenkins credentials option.

- Now save the code and select build Now to look for vulnerabilities.
- Once the build is done, go the webpage where your sonarcloud server is running and open the projects to view the vulnerabilities and changes you need to make.



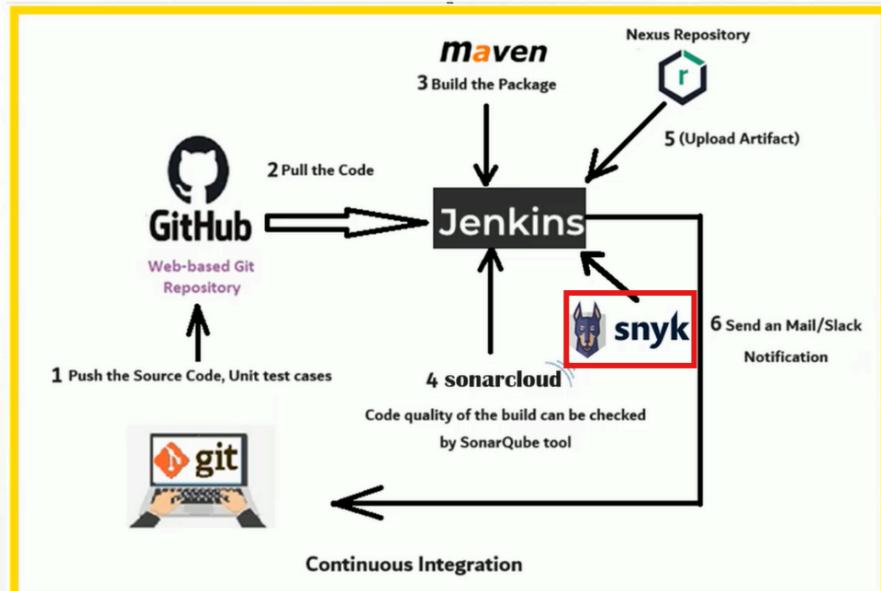
Snyk

Tool to find the vulnerabilities and fix it automatically at code level.

Types:

1. Snyk Code: SAST Tool
2. Snyk opensource: Software composition analysis. (Find vulnerabilities in third party libraries) like log4j.
3. Snyk container: Finding vulnerabilities in kubernetes and container technologies.
4. Snyk is Infrastructure as a code: Finding vulnerabilities on AWS, Terraform and other IAAC Platforms.

Integrating Snyk with Jenkins



- Lets create the account on snyk.io, I would suggest logging in using github account.

The screenshot shows the Snyk.io dashboard for the organization 'kp18-cpu'. The left sidebar includes links for 'Dashboard', 'Projects', 'Integrations', 'Members', and 'Settings'. The main content area features a 'Start securing your code' section with two options:

- Connect your code: 'Connect Snyk to your code to fix issues and vulnerabilities.' A 'Choose integration' button is present.
- Add and scan your first project: 'Import your code to see how Snyk surfaces issues, problematic dependencies, and vulnerabilities.'

Below these sections are 'Invite team members' (with a 'Copy invite link' button), 'Use Snyk in the command line' (with a 'Learn more' button), and notifications for 'Product updates' and 'Help'.

You will see the dashboard like this.

- Important note, whatever code we are going to analyse, we are going to use maven and snyk integration and in that pom.xml file we need to have this plugin script for sure.

```

<plugin>
  <groupId>io.snyk</groupId>
  <artifactId>snyk-maven-plugin</artifactId>
  <version>2.2.0</version>
  <inherited>false</inherited>
  <configuration>
    </configuration>
</plugin>
<!-- Snyk Changes for SCA Ends Here -->
```

- We need to have snyk token for using that and for authenticating it with jenkins. You will find the auth token on synk website, click on your name in the left bottom.

The screenshot shows the Snyk account settings interface. On the left, there's a sidebar with organization details (kp18-cpu), navigation links (Dashboard, Projects, Integrations, Members, Settings), and user notifications (Product updates, Help, Email). The main area is titled 'Account' > 'General'. It has sections for 'ACCOUNT SETTINGS' (with 'General' selected and highlighted in red) and 'Auth Token'. The 'Auth Token' section contains instructions: 'Use this token to authenticate the Snyk CLI and in CI/CD pipelines. Learn more about authenticating CLI in docs.' Below this is a table with one row, showing a token key ('015cdca6-5322-4904-b946-cb83747') and its creation date ('18 November 2024, 15:31:56'). A red box highlights the token key. At the bottom right of the table is a 'Revoke & Regen' button. Other sections visible include 'Authorized Applications' (No applications listed) and 'Preferred Organization' (Choose which organization you are taken to when logging into the site).

Keep this token safe, now navigate to manage jenkins on the jenkins server.

Dashboard > Manage Jenkins

Clouds

Add, remove, and configure cloud instances to provision agents on-demand.

Appearance

Configure the look and feel of Jenkins

Security

Clouds

Add, remove, and configure cloud instances to provision agents on-demand.

Credentials

Configure credentials

Credential Providers

Secure Jenkins; define who is allowed to access/use the system.

Users

Create/delete/modify users that can log Jenkins.

Status Information

System Information

System Log

- Try to add the credentials and enter secret text with the token value and mention the ID and description which we will use it on the pipeline.

New credentials

Kind

Secret text

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Secret

ID ?

SNYK_TOKEN

Description ?

SNYK_TOKEN

Create

- Now let's create a new pipeline.

New Item

Enter an item name

snyk_maven_pipeline

Select an item type



Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

OK

- Use the following script in the pipeline, make sure of the indentation.

```
pipeline {
    agent any
    tools {
        maven 'maven-3.9.9'
    }
    stages {
        stage ("Checking out the git project") {
            steps {
                git branch: 'main', url:
'https://github.com/Shikhar82/devsecops-jenkins-k8s-tf-sast-sca-sonarcloud-snyk-repo.git'
            }
        }

        stage ("Checking bugs on sonarcloud") {
            steps {
                sh 'mvn clean verify sonar:sonar
-Dsonar.projectKey=devsecopskp -Dsonar.organization=devsecopskp
-Dsonar.host.url=https://sonarcloud.io
-Dsonar.token=d0d2af0eee7f1fdb21e937360084b8ece5a62cb4'
            }
        }
    }
}
```

```
stage ("Run SCA scan and analysis using snyk"){
    steps {
        withCredentials([string(credentialsId: 'SNYK_TOKEN',
variable: 'SNYK_TOKEN')]) {
            sh 'mvn snyk:test -fn'
        }
    }
}
```

Script ?

```
17 // }
18 stage ("Build the package") {
19     steps {
20         sh 'mvn clean package'
21     }
22 }
23
24 stage ("Run SCA scan and analysis using snyk"){
25     steps {
26         withCredentials([string(credentialsId: 'SNYK_TOKEN', variable: 'SNYK_TOKEN')]) {
27             sh 'mvn snyk:test -fn'
28         }
29     }
30 }
31 }
32 }
```

Use Groovy Sandbox ?

Save **Apply**

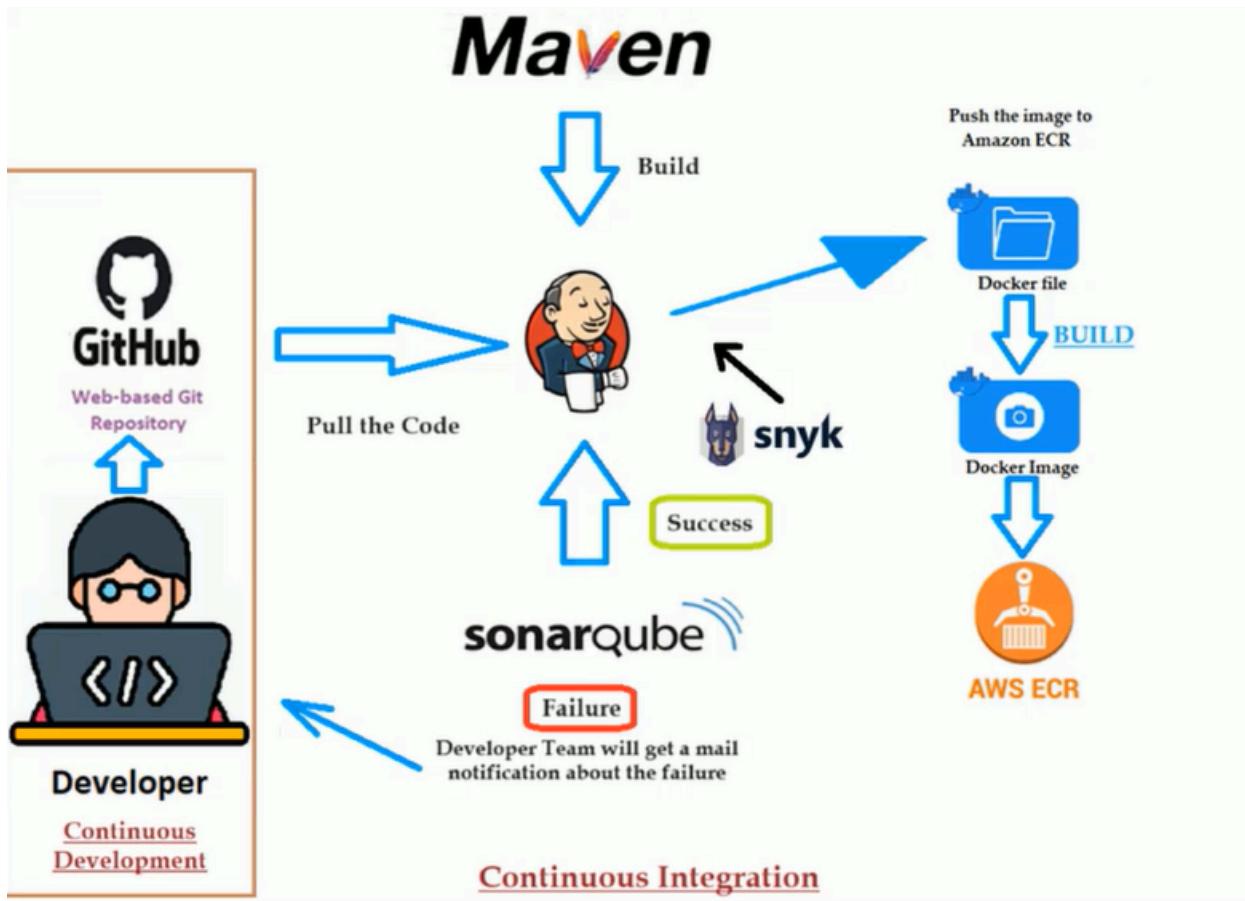
Once you are done with the script apply and save.

```
[INFO] Tested 35 dependencies for known issues, found 51 issues, 51 vulnerable paths.  
[INFO]  
[INFO]  
[INFO] Issues to fix by upgrading:  
[INFO]  
[INFO]   Upgrade mysql:mysql-connector-java@5.1.25 to mysql:mysql-connector-java@8.0.28 to fix  
[INFO]     X Improper Access Control [Low Severity][https://security.snyk.io/vuln/SNYK-JAVA-MYSQL-31449] in  
mysql:mysql-connector-java@5.1.25  
[INFO]       introduced by mysql:mysql-connector-java@5.1.25  
[INFO]     X Improper Authorization [Medium Severity][https://security.snyk.io/vuln/SNYK-JAVA-MYSQL-2386864] in  
mysql:mysql-connector-java@5.1.25  
[INFO]       introduced by mysql:mysql-connector-java@5.1.25  
[INFO]     X XML External Entity (XXE) Injection [Medium Severity][https://security.snyk.io/vuln/SNYK-JAVA-MYSQL-1766958] in mysql:mysql-connector-java@5.1.25  
[INFO]       introduced by mysql:mysql-connector-java@5.1.25  
[INFO]     X Privilege Escalation [Medium Severity][https://security.snyk.io/vuln/SNYK-JAVA-MYSQL-174574] in  
mysql:mysql-connector-java@5.1.25  
[INFO]       introduced by mysql:mysql-connector-java@5.1.25  
[INFO]     X Arbitrary Code Execution [Medium Severity][https://security.snyk.io/vuln/SNYK-JAVA-MYSQL-31580] in  
mysql:mysql-connector-java@5.1.25  
[INFO]       introduced by mysql:mysql-connector-java@5.1.25  
[INFO]     X SQL Injection [Medium Severity][https://security.snyk.io/vuln/SNYK-JAVA-MYSQL-451460] in  
mysql:mysql-connector-java@5.1.25  
[INFO]       introduced by mysql:mysql-connector-java@5.1.25  
[INFO]     X Access Control Bypass [High Severity][https://security.snyk.io/vuln/SNYK-JAVA-MYSQL-451464] in  
mysql:mysql-connector-java@5.1.25  
[INFO]       introduced by mysql:mysql-connector-java@5.1.25
```

We can see all the vulnerabilities and severity on the console output which is report of the pipeline.

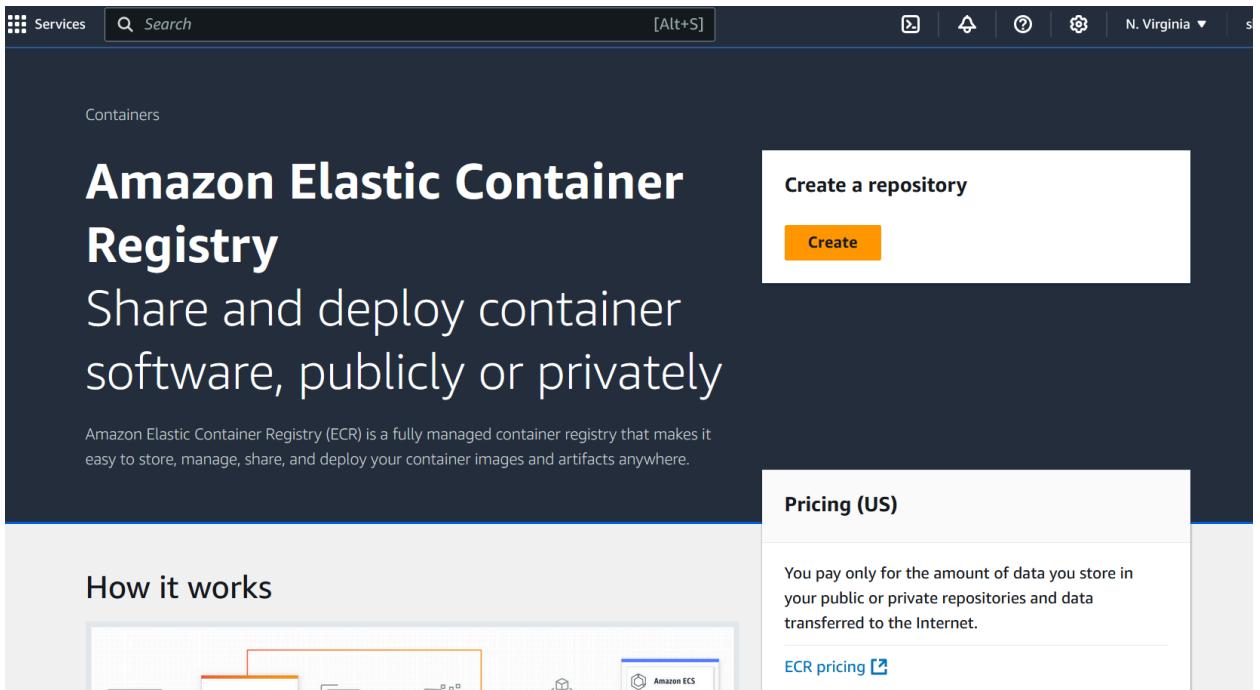
We are successfully worked with Snyk and Sonarcloud.

Building Docker Image using jenkins pipeline



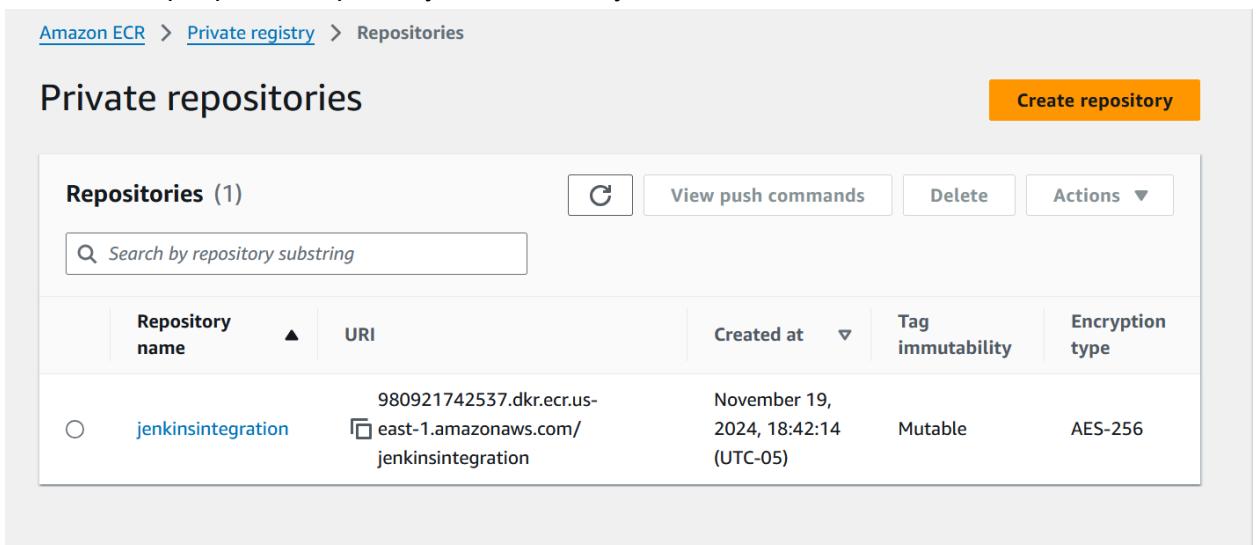
Based on the diagram we are going to build the docker file and deploy the image to AWS ECR (Elastic Container Registry) for better management.

- Go to aws console
- Search for google, build the docker image using jenkins pipeline. Go to this website and copy the two blocks of stages for creation and deployment.



The screenshot shows the Amazon Elastic Container Registry (ECR) homepage. At the top, there's a navigation bar with 'Services' and a search bar. To the right are icons for refresh, filter, and settings, along with the text 'N. Virginia'. Below the header, the title 'Amazon Elastic Container Registry' is prominently displayed with the subtitle 'Share and deploy container software, publicly or privately'. A call-to-action button 'Create a repository' with an orange 'Create' button is visible. On the left, a section titled 'How it works' includes a diagram showing a flow from a developer workstation through a build step to an ECR repository and finally to an Amazon ECS cluster. To the right, a 'Pricing (US)' section states that users pay only for stored data in public or private repositories and data transferred to the Internet, with a link to 'ECR pricing'.

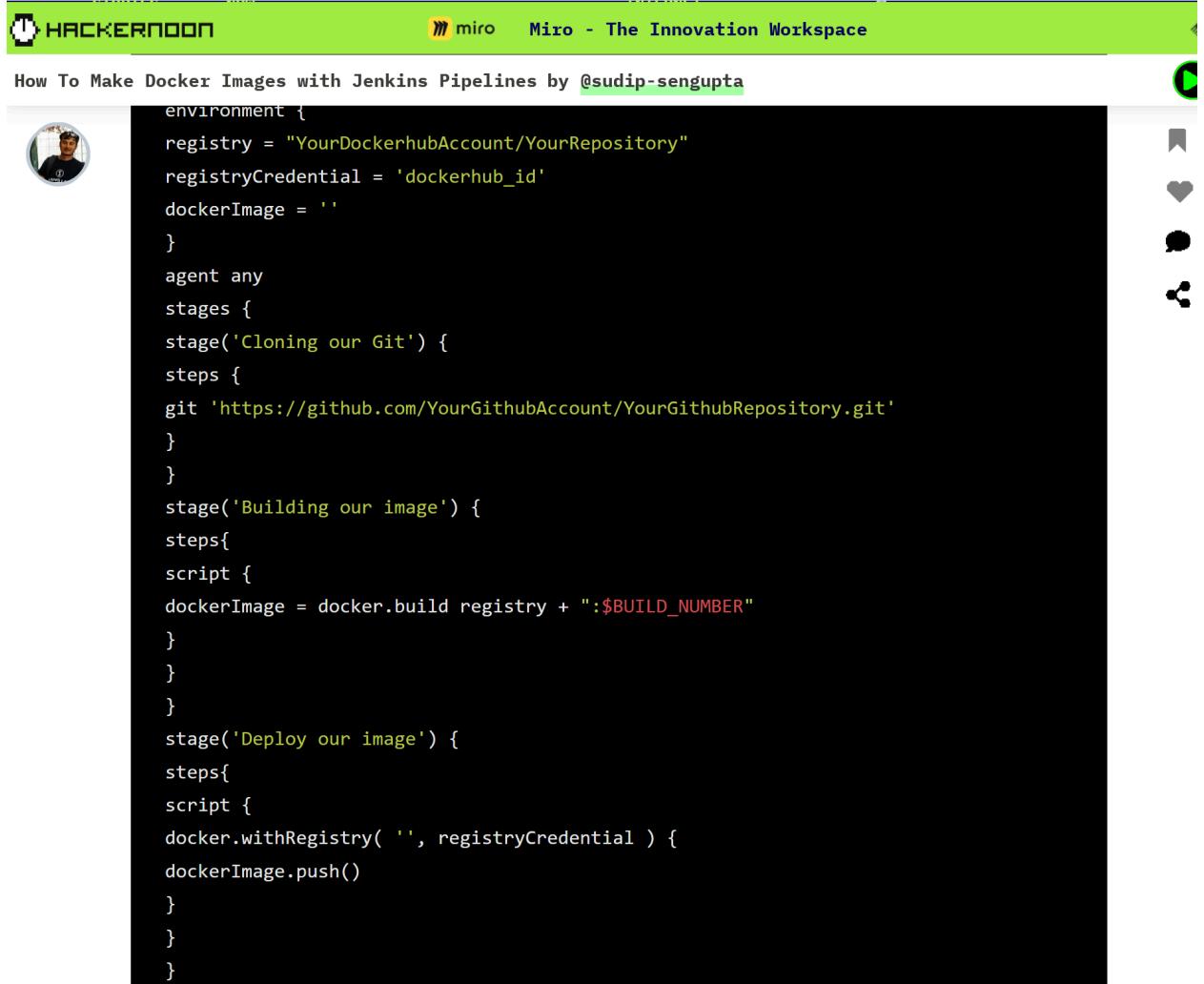
- Create a simple private repository with a name you want to use.



The screenshot shows the 'Private repositories' page within the Amazon ECR service. The URL in the address bar is 'Amazon ECR > Private registry > Repositories'. The main heading is 'Private repositories' with a 'Create repository' button. Below is a table titled 'Repositories (1)'. The table has columns for 'Repository name', 'URI', 'Created at', 'Tag immutability', and 'Encryption type'. One repository is listed: 'jenkinsintegration' with URI '980921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration', created on 'November 19, 2024, 18:42:14 (UTC-05)', 'Mutable' tag immutability, and 'AES-256' encryption type. There are also buttons for 'View push commands', 'Delete', and 'Actions'.

Repository name	URI	Created at	Tag immutability	Encryption type
jenkinsintegration	980921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration	November 19, 2024, 18:42:14 (UTC-05)	Mutable	AES-256

- Go to jenkins server create a new pipeline and copy the old pipeline from above projects and try to add two parts of the code for integrating docker.
- Yes, we are going to build and deploy docker images using jenkins server so copy that part of the code to include in the pipeline.



The screenshot shows a Miro workspace titled "How To Make Docker Images with Jenkins Pipelines by @sudip-sengupta". The workspace contains a Jenkins pipeline script:

```

environment {
    registry = "YourDockerhubAccount/YourRepository"
    registryCredential = 'dockerhub_id'
    dockerImage = ''
}

agent any

stages {
    stage('Cloning our Git') {
        steps {
            git 'https://github.com/YourGithubAccount/YourGithubRepository.git'
        }
    }
    stage('Building our image') {
        steps{
            script {
                dockerImage = docker.build registry + ":$BUILD_NUMBER"
            }
        }
    }
    stage('Deploy our image') {
        steps{
            script {
                docker.withRegistry( '', registryCredential ) {
                    dockerImage.push()
                }
            }
        }
    }
}

```

- Create a new pipeline at jenkins and change the github repository with something without any vulnerabilities.
- Change the github url to "<https://github.com/Shikhar82/springboot-maven-micro.git>" or any git project without any vulnerabilities.

```

        }
        stages {
            stage ("Checking out the git project") {
                steps {
                    git branch: 'master', url: 'https://github.com/Shikhar82/springboot-maven-micro.git'
                }
            }
        }
    }
}

```

- Let's add the docker scripts to our newly created pipeline.

```

stage('Building our image') {
steps{
script {
dockerImage = docker.build registry + ":$BUILD_NUMBER"
}
}
}
}

stage('Deploy our image') {
steps{
script {
docker.withRegistry( " ", registryCredential ) {
dockerImage.push()
}
}
}
}
}

```

- In your pipeline add the registry value as a URI from the ECR repository for example below.

Private repositories

Create repository

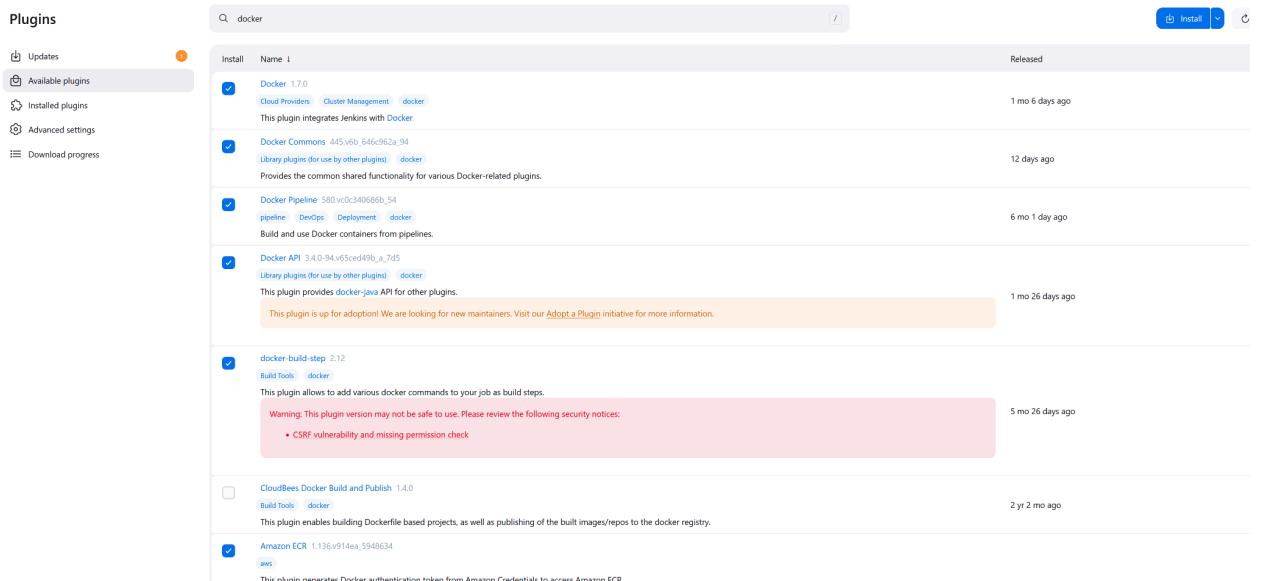
Repositories (1)		C	View push commands	Delete	Actions ▾
<input type="text"/> Search by repository substring					
Repository name	URI	Created at	Tag immutability	Encryption type	
jenkinsintegration	980921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration	November 19, 2024, 18:42:14 (UTC-05)	Mutable	AES-256	

```

` pipeline {
    agent any
    tools {
        maven 'maven-3.9.9'
    }
    environment {
        registry = "980921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration"
    }
    stages {
        stage ("Checking out the git project") {
            steps {
                git branch: 'master', url: 'https://github.com/Shikhar82/springboot-maven-micro.git'
            }
        }
    }
}

```

- We need to install the docker on our jenkins server (Virtual Machine), we can find commands online for installing docker on RHEL or you can simple CHATGPT it.
- Commands for installing docker on RHEL.
 1. sudo yum update -y
 2. yum install -y yum-utils device-mapper-persistent-data lvm2
 3. sudo yum-config-manager --add-repo
<https://download.docker.com/linux/centos/docker-ce.repo>
 4. sudo yum install -y docker-ce docker-ce-cli containerd.io
 5. sudo systemctl start docker
 6. sudo systemctl enable docker
 7. docker --version
 8. sudo docker run hello-world
 9. sudo usermod -aG docker \$USER (Change USER according to your username)
- After installing the docker on the jenkins server go to “manage jenkins” before that save your code and in manage jenkins>plugins>available plugins. Click on all the plugins that has Docker and Amazon ECR as well and install them.



- Once you enabled the required docker plugin simply run you pipeline. I will show how the pipeline looks below.

```
pipeline {
    agent any
    tools {
```

```
maven 'maven-3.9.9'
}
environment {
    registry =
"980921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration"
}
stages {
    stage ("Checking out the git project") {
        steps {
            git branch: 'master', url:
'https://github.com/Shikhar82/springboot-maven-micro.git'
        }
    }

    stage ("Checking bugs on sonarcloud") {
        steps {
            sh 'mvn clean verify sonar:sonar
-Dsonar.projectKey=devsecopskp -Dsonar.organization=devsecopskp
-Dsonar.host.url=https://sonarcloud.io
-Dsonar.token=d0d2af0eee7f1fdb21e937360084b8ece5a62cb4'
        }
    }
    // stage ("Build the package") {
    //     steps {
    //         sh 'mvn clean package'
    //     }
    // }

    stage ("Run SCA scan and analysis using snyk"){
        steps {
            withCredentials([string(credentialsId: 'SNYK_TOKEN',
variable: 'SNYK_TOKEN')]) {
                sh 'mvn snyk:test -fn'
            }
        }
    }
    stage('Building our image') {
        steps{
            script {
                dockerImage = docker.build registry + ":"$BUILD_NUMBER"
            }
        }
    }
}
```

```

// stage('Deploy our image') {
// steps{
// script {
// docker.withRegistry( '', registryCredential ) {
// dockerImage.push()
// }
// }
// }
// }
}

}

```

```

#4 sha256:899046e4a240e349763e42464f501b60a1bd429af9f38cc927d9da2124b98de 62.68MB / 62.68MB 1.0s done
#4 sha256:5ce47bbf12c856f2c3827f573bc33beb2ba86473a4817f4f8c1b87f1990cd6d 56.62MB / 151.61MB 1.1s
#4 sha256:5ce47bbf12c856f2c3827f573bc33beb2ba86473a4817f4f8c1b87f1990cd6d 65.01MB / 151.61MB 1.2s
#4 extracting sha256:899046e4a240e349763e42464f501b60a1bd429af9f38cc927d9da2124b98de 0.1s
#4 sha256:5ce47bbf12c856f2c3827f573bc33beb2ba86473a4817f4f8c1b87f1990cd6d 73.40MB / 151.61MB 1.3s
#4 sha256:5ce47bbf12c856f2c3827f573bc33beb2ba86473a4817f4f8c1b87f1990cd6d 96.47MB / 151.61MB 1.6s
#4 sha256:5ce47bbf12c856f2c3827f573bc33beb2ba86473a4817f4f8c1b87f1990cd6d 105.91MB / 151.61MB 1.7s
#4 sha256:5ce47bbf12c856f2c3827f573bc33beb2ba86473a4817f4f8c1b87f1990cd6d 122.68MB / 151.61MB 1.9s
#4 sha256:5ce47bbf12c856f2c3827f573bc33beb2ba86473a4817f4f8c1b87f1990cd6d 131.07MB / 151.61MB 2.0s
#4 sha256:5ce47bbf12c856f2c3827f573bc33beb2ba86473a4817f4f8c1b87f1990cd6d 145.75MB / 151.61MB 2.2s
#4 sha256:5ce47bbf12c856f2c3827f573bc33beb2ba86473a4817f4f8c1b87f1990cd6d 151.61MB / 151.61MB 2.8s done
#4 extracting sha256:899046e4a240e349763e42464f501b60a1bd429af9f38cc927d9da2124b98de 4.2s done
#4 extracting sha256:5ce47bbf12c856f2c3827f573bc33beb2ba86473a4817f4f8c1b87f1990cd6d
#4 extracting sha256:5ce47bbf12c856f2c3827f573bc33beb2ba86473a4817f4f8c1b87f1990cd6d 3.0s done
#4 DONE 8.6s

#6 [2/2] COPY target/springboot-maven-course-micro-svc-0.0.1-SNAPSHOT.jar app.jar
#6 DONE 0.9s

#7 exporting to image
#7 exporting layers 0.1s done
#7 writing image sha256:0bc277b3dacad71ba5415759812920918f6dfa6ecfb0f54d9c81590518bdc3f
#7 writing image sha256:0bc277b3dacad71ba5415759812920918f6dfa6ecfb0f54d9c81590518bdc3f done
#7 naming to 988921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration:4 done
#7 DONE 0.1s
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

Pipeline script

```
Script ?  
30      |      |      |      |      sh 'mvn snyk:test -fn'  
31      |      |      |      }  
32      |      }  
33      }      stage('Building our image') {  
34      |      steps{  
35      |      |      script {  
36      |      |      |      dockerImage = docker.build registry + ":$BUILD_NUMBER"  
37      |      |      }  
38      |      }  
39      }  
40      }  
41 // stage('Deploy our image') {  
42 // steps{  
43 // script {  
44 // docker.withRegistry( '', registryCredential ) {  
45 // dockerImage.push()  
46 // }  
47 // }
```

- If you encounter any error while build especially building docker go to /var/run/docker.sock on your Jenkins server VM and give the following command: “chmod 777 /var/run/docker.sock”
- Once everything runs successfully you can view your docker image build on the Jenkins server VM through the command “docker image ls”.

```
[root@ip-172-31-84-17 docker_jenkins_maven]# docker image ls  
REPOSITORY          TAG      IMAGE ID      CREATED     SIZE  
980921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration 4        0bc277b3daca   28 minutes ago  483MB
```

Pushing our Docker image to AWS ECR:

- We need to establish communication between AWS and Jenkins server for them to communicate in the first place.
- For this purpose first we are going to install AWS CLI on our Jenkins server VM. Follow these three commands on your Jenkins server VM.
 - curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
 - unzip awscliv2.zip
 - sudo ./aws/install
- Once entered the last command “./aws/install” add the path so that you can aws cli anywhere like shown in the screenshot.

```
[root@ip-172-31-84-17 ~]# ./aws/install
You can now run: /usr/local/bin/aws --version
[root@ip-172-31-84-17 ~]# echo $PATH
/usr/local/bin:/root/bin:/bin:/usr/sbin:/usr/bin
[root@ip-172-31-84-17 ~]# ^C
[root@ip-172-31-84-17 ~]#
[root@ip-172-31-84-17 ~]# /usr/local/bin/aws --version
aws-cli/2.22.1 Python/3.12.6 Linux/5.14.0-427.20.1.el9_4.x86_64 exe/x86_64.rhel.9
[root@ip-172-31-84-17 ~]# export PATH=$PATH:/usr/local/bin
[root@ip-172-31-84-17 ~]# aws --version
aws-cli/2.22.1 Python/3.12.6 Linux/5.14.0-427.20.1.el9_4.x86_64 exe/x86_64.rhel.9
[root@ip-172-31-84-17 ~]#
```

- After installing the aws-cli now our jenkins server is ready to talk with aws ECR. So go to the pipeline, now we will use the deploy stage in our pipeline which we commented out before.

Script ?

```
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
```

The code block shows a Jenkins Pipeline script. A red box highlights the commented-out 'Deploy' stage from lines 41 to 50. The code is as follows:

```
steps{
    script {
        dockerImage = docker.build registry + ":"$BUILD_NUMBER"
    }
}
// stage('Deploy our image') {
// steps{
// script {
// docker.withRegistry( '', registryCredential ) {
// dockerImage.push()
// }
// }
// }
// }
}
```

- Once we enable the script in our code we need to have registryCredential, so we will create a user on AWS with certain policy which aligns with AWS ECR and we will derive the auth token for AWS and use it here at pipeline for the build.

Definition

Pipeline script

Script ?

```
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
```

steps{
 script {
 dockerImage = docker.build registry + ":"\$BUILD_NUMBER"
 }
}
stage('Deploy our image') {
 steps{
 script {
 docker.withRegistry('...', registryCredential) {
 dockerImage.push()
 }
 }
 }
}

Use Groovy Sandbox ?

- Go to AWS management console and search for IAM and create a user.

The screenshot shows the AWS Identity and Access Management (IAM) service in the AWS Management Console. The left sidebar has 'Identity and Access Management (IAM)' selected under 'Access management'. The main area shows the 'Users (0)' page with a heading 'Users (0) Info' and a note: 'An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.' There is a search bar at the top and another one below the heading. On the right, there is a large orange 'Create user' button. The entire 'Create user' button is highlighted with a red rectangle. Below it are navigation controls for pages 1, 2, and 3, and a 'Delete' button.

Specify user details

User details

User name

awsecr_jenkins_user

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)

Provide user access to the AWS Management Console - *optional*

If you're providing console access to a person, it's a [best practice](#) to manage their access in IAM Identity Center.

 If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keypairs, you can generate them after you create this IAM user. [Learn more](#)

Cancel

Next

Give permission as Attach policies and the policy name would be **AmazonEC2ContainerRegistryFullAccess**.

Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Permissions options

Add user to group

Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

Copy permissions

Copy all group memberships, attached managed policies, and inline policies from an existing user.

Attach policies directly

Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (1/1280)

Choose one or more policies to attach to your new user.

Filter by Type

container

X

All types

15 matches

< 1 >

⚙

Policy name

▲ | Type

▼ | Attached entities



 AmazonEC2ContainerRegis...

AWS managed

0



 AmazonEC2ContainerRegis...

AWS managed

0



 AmazonEC2ContainerRegis...

AWS managed

0

User details

User name awsecr_jenkins_user	Console password type None	Require password reset No
----------------------------------	-------------------------------	------------------------------

Permissions summary

Name	Type	Used as
AmazonEC2ContainerRegistryFullAccess	AWS managed	Permissions policy

Tags - optional
 Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

[Cancel](#)
[Previous](#)
Create user

- Once the user is created go to security credentials and navigate to access keys and create a access key.

awsecr_jenkins_user [Info](#) [Delete](#)

Summary

ARN arn:aws:iam::980921742537:user/awsecr_jenkins_user	Console access Disabled	Access key 1 Create access key
Created November 19, 2024, 20:19 (UTC-05:00)	Last console sign-in -	

Permissions	Groups	Tags	Security credentials	Last Accessed
-----------------------------	------------------------	----------------------	----------------------	-------------------------------

Console sign-in

Console sign-in link https://980921742537.signin.aws.amazon.com/console	Console password Not enabled
---	---------------------------------

Multi-factor authentication (MFA) (0)

Use MFA to increase the security of your AWS environment. Signing in with MFA requires an authentication code from an MFA device. Each user can have a maximum of 8 MFA devices assigned. [Learn more](#)

Type	Identifier	Certifications	Created on
------	------------	----------------	------------

- On create access key, click on Command Line Interface (CLI)

[IAM](#) > [Users](#) > [awsecr_jenkins_user](#) > Create access key

alternatives

Step 2 - optional
 Set description tag
 Step 3
 Retrieve access keys

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

Command Line Interface (CLI)
 You plan to use this access key to enable the AWS CLI to access your AWS account.

Local code
 You plan to use this access key to enable application code in a local development environment to access your AWS account.

Application running on an AWS compute service
 You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

Third-party service
 You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

Application running outside AWS
 You plan to use this access key to authenticate workloads running in your

- Save both access keys on the notepad or notes for adding it in the jenkins server.

Access key

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
 AKIA6IY356TE2CKBBM6D	 ***** Show

Access key best practices

- Go to jenkins dashboard and manage jenkins>credentials.

The screenshot shows the Jenkins dashboard with the 'Manage Jenkins' option selected in the top navigation bar. A red box highlights the 'Credentials' link under the 'System Configuration' section, which is described as 'Configure credentials'.

- Add credentials

The screenshot shows the 'Global credentials (unrestricted)' page. A red box highlights the '+ Add Credentials' button in the top right corner. The table below lists two existing credentials:

ID	Name	Kind	Description
sonartoken	This is the security token for integrating sonar server	Secret text	This is the security token for integrating sonar server
SNYK_TOKEN	SNYK_TOKEN	Secret text	SNYK_TOKEN

Click on AWS credentials

The screenshot shows the 'New credentials' page. The 'Kind' dropdown menu is open, and 'AWS Credentials' is selected. Other options in the dropdown include 'Username with password', 'GitHub App', 'SSH Username with private key', 'Secret file', 'Secret text', 'X.509 Client Certificate', and 'Certificate'.

For ID paste the name of user you created on the AWS

IAM > Users > awsecr_jenkins_user

Identity and Access Management (IAM)

Search IAM

Dashboard

▼ Access management

- User groups
- Users**
- Roles
- Policies
- Identity providers
- Account settings

awsecr_jenkins_user Info

Summary

ARN	arn:aws:iam::980921742537:user/awsecr_jenkins_user	Console access Disabled
Created	November 19, 2024, 20:19 (UTC-05:00)	Last console sign-in -

Permissions
Groups
Tags
Security credentials
Last Ac

Console sign-in

Console sign-in URL: [Console sign-in URL](#)

Console sign-in token: [Console sign-in token](#)

New credentials

Kind: AWS Credentials

Scope: Global (Jenkins, nodes, items, all child items, etc)

ID: awsecr_jenkins_user

Description: awsecr_jenkins_user

Access Key ID: AKIA6IY356TE2CKBBM6D

Secret Access Key: XXXXXXXXXXXXXX

These credentials are valid but do not have access to the "AmazonEC2" service in the region "us-east-1". This message is not a problem if you need to access to other services or to other regions. Message: "You are not authorized to perform this operation. User: arn:aws:iam::980921742537:user/awsecr_jenkins_user is not authorized to perform: ec2:DescribeAvailabilityZones because no identity-based policy allows the ec2:DescribeAvailabilityZones action (UnauthorizedOperation)"

IAM Role Support: Advanced ▾

Create

After inputting the access key and other values hit create.

- Now go back to your pipeline and add the registryCredential parameter to the environment code. Credential parameter is the ID of the credential or the username of the AWS user you have created.

Pipeline

Definition

Pipeline script

Script ?

```

1 * pipeline {
2     agent any
3     tools {
4         maven 'maven-3.9.9'
5     }
6     environment {
7         registry = "980921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration"
8         registryCredential = 'awsecr_jenkins_user'
9     }
10    stages {
11        stage ('Checking out the git project') {
12            steps {
13                git branch: 'master', url: 'https://github.com/Shikhar82/springboot-maven-micro.git'
14            }
15        }
16        stage ("Checking bugs on sonarcloud") {
17            steps {
18

```

Use Groovy Sandbox ?

Run this Pipeline

Adding new parameter in the “docker.withRegistry parameter” as “http://” + registry (which is uri of ECR registry), “ecr:us-east-1.” + registryCredential) Note: change you zone accordingly.

```

        }
    }
    stage('Deploy our image') {
        steps{
            script {
                docker.withRegistry("http://" + registry, "ecr:us-east-1:" + registryCredential ) {
                    dockerImage.push()
                }
            }
        }
    }
}

```

- Whole code below for the pipeline.

```

pipeline {
    agent any
    tools {
        maven 'maven-3.9.9'
    }
    environment {
        registry =

```

```

"980921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration"
    registryCredential = 'awsecr_jenkins_user'
}
stages {
    stage ("Checking out the git project") {
        steps {
            git branch: 'master', url:
'https://github.com/Shikhar82/springboot-maven-micro.git'
        }
    }

    stage ("Checking bugs on sonarcloud") {
        steps {
            sh 'mvn clean verify sonar:sonar
-Dsonar.projectKey=devsecopskp -Dsonar.organization=devsecopskp
-Dsonar.host.url=https://sonarcloud.io
-Dsonar.token=d0d2af0eee7f1fdb21e937360084b8ece5a62cb4'
        }
    }
    // stage ("Build the package") {
    //     steps {
    //         sh 'mvn clean package'
    //     }
    // }

    stage ("Run SCA scan and analysis using snyk"){
        steps {
            withCredentials([string(credentialsId: 'SNYK_TOKEN',
variable: 'SNYK_TOKEN')]) {
                sh 'mvn snyk:test -fn'
            }
        }
    }
    stage('Building our image') {
        steps{
            script {
                dockerImage = docker.build registry + ":$BUILD_NUMBER"
            }
        }
    }
    stage('Deploy our image') {
        steps{
            script {

```

```

        docker.withRegistry("http://" + registry,
"ecr:us-east-1:" + registryCredential ) {
            dockerImage.push()
        }
    }

}
}

}

```

- Once everything set in your pipeline, save the code and click build now.

```

[Pipeline] withEnv
[Pipeline] [
[Pipeline] sh
+ docker tag 988921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration:5 988921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration:5
[Pipeline] )
[Pipeline] // withEnv
[Pipeline] isUnix
[Pipeline] withEnv
[Pipeline] [
[Pipeline] sh
+ docker push 988921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration:5
The push refers to repository [988921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration]
b3b7be0bc413: Preparing
d643765cb552: Preparing
3f307074c00f: Preparing
b3b7be0bc413: Pushed
3f307074c00f: Pushed
d643765cb552: Pushed
5: digest: sha256:b2d795dc90a5a838fed4fad4df97f1beb50fb4f0be6414b5a56ebc0b4ffb756 size: 954
[Pipeline] )
[Pipeline] // withEnv
[Pipeline] )
[Pipeline] // withDockerRegistry
[Pipeline] )
[Pipeline] // withEnv
[Pipeline] )
[Pipeline] // script
[Pipeline] )
[Pipeline] // withEnv
[Pipeline] )
[Pipeline] // stage
[Pipeline] )
[Pipeline] // withEnv
[Pipeline] )
[Pipeline] // withEnv
[Pipeline] )
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

- Let's check whether image is successfully pushed to AWS ECR.

	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
<input type="checkbox"/>	5	Image	November 19, 2024, 20:38:34 (UTC-05)	229.86	<input type="button" value="Copy URI"/>	<input type="button" value="sha256:b2d795d..."/>

We got the image pushed successfully.

- Let's also check about it on jenkins server vm.

```
root@ip-172-31-84-17 ~# docker image ls
REPOSITORY                                     TAG      IMAGE ID      CREATED       SIZE
980921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration   5        e90a3e728955   2 minutes ago  483MB
980921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration   4        0be277b5daca   About an hour ago  403MB
[Root@ip-172-31-84-17 ~]#
```

Successfull build and deploy of docker image.

Great work. CI Continuous Integration is ready.

Time to do the Continuous Deployment.

What is Continous Deployment?

Continuous Deployment (CD) is a software development practice where changes to the codebase are automatically deployed to production environments without manual intervention. It is the next step after **Continuous Integration (CI)** and **Continuous Delivery (also CD)**, focusing on delivering software updates directly to end-users as soon as they pass automated testing and quality checks.

CD server

- Go to AWS and create a ec2 instance and we will use UBUNTU as a CD server.

The screenshot shows the AWS EC2 'Create New Instance' wizard. The first step, 'Name and tags', has a 'Name' field containing 'cd_server' and a 'Add additional tags' link. The second step, 'Application and OS Images (Amazon Machine Image)', displays a search bar and a grid of OS options: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, and others. The 'Ubuntu' option is selected. Below the grid, the 'Amazon Machine Image (AMI)' details for 'Ubuntu Server 22.04 LTS (HVM), SSD Volume Type' are shown, including its AMI ID, architecture (64-bit (x86)), and a 'Verified provider' badge. The 'Description' section notes Canonical support and the image build date (2024-09-27). The 'Architecture' dropdown is set to '64-bit (x86)'.

Name and tags [Info](#)

Name
cd_server [Add additional tags](#)

▼ Application and OS Images (Amazon Machine Image) [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Recents [Quick Start](#)

Amazon Linux macOS Ubuntu Windows Red Hat ... SL >

Search [Browse more AMIs](#)
Including AMIs from AWS Marketplace and

Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type [Free tier eligible](#)

ami-005fc0f236362e99f (64-bit (x86)) / ami-07ee04759daf109de (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

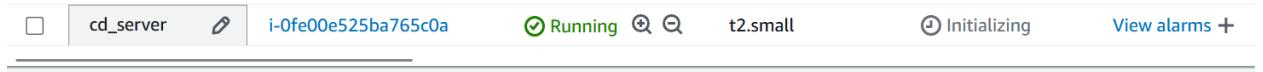
Ubuntu Server 22.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2024-09-27

Architecture [AMI ID](#) Username [i](#)

64-bit (x86) ami-005fc0f236362e99f ubuntu [Verified provider](#)

- Once the ubuntu is running we will access it through ssh client and install docker and aws cli.



- Update the system

```
ubuntu@ip-172-31-83-200:~$ sudo su
root@ip-172-31-83-200:/home/ubuntu# apt update
```

- Install aws cli and add it to the path which we have done above already for jenkins server. Take it as a reference.

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip"
-o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

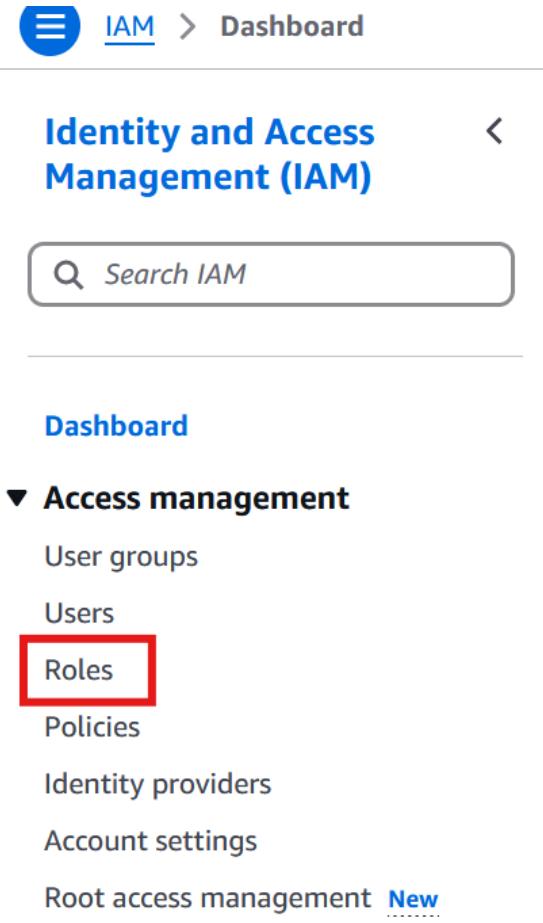
```
aws awscliv2.zip
root@ip-172-31-83-200:/home/ubuntu# cd aws
root@ip-172-31-83-200:/home/ubuntu/aws# ls
README.md  THIRD PARTY LICENSES  dist  install
root@ip-172-31-83-200:/home/ubuntu/aws# ./install
You can now run: /usr/local/bin/aws --version
root@ip-172-31-83-200:/home/ubuntu/aws# export PATH=$PATH:/usr/local/bin
bash: export: `PATH:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/usr/local/bin': not a valid identifier
root@ip-172-31-83-200:/home/ubuntu/aws# aws --version
root@ip-172-31-83-200:/home/ubuntu/aws# aws --version
aws-cli/2.22.1 Python/3.12.6 Linux/6.8.0-1015-aws exe/x86_64.ubuntu.22
root@ip-172-31-83-200:/home/ubuntu/aws#
```

- Now let's install the docker on the Ubuntu which is cd server.

- apt install docker
- apt install podman-docker

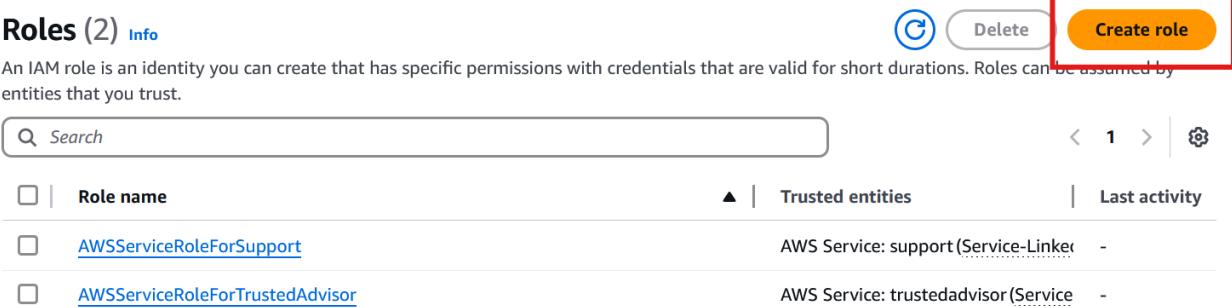
```
root@ip-172-31-83-200:~# apt install podman-docker
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  buildah catatonit common containerNetworking-plugins crun dconf-gsettings-backend dconf-service dns-root-data dnsmasq-base
  docker-compose fuse-overlayfs glib-networking glib-networking-common glib-networking-services
  golang-github-containernetworking-plugin-dnsname golang-github-containers-common golang-github-containers-image
  gsettings-desktop-schemas libavahi-client3 libavahi-common-data libavahi-common3 libavahi-glib1 libdconf1 libostree-1-1
  libproxy1v5 libslirp0 libsoup2.4-1 libsoup2.4-4-common libyajl2 podman python3-docker python3-dockerpty python3-docopt
  python3-dotenv python3-texttable python3-websocket session-migration slirp4netns uidmap
Suggested packages:
  containers-storage
Recommended packages:
  docker.io
The following NEW packages will be installed:
  buildah catatonit common containerNetworking-plugins crun dconf-gsettings-backend dconf-service dns-root-data dnsmasq-base
  docker-compose fuse-overlayfs glib-networking glib-networking-common glib-networking-services
  golang-github-containernetworking-plugin-dnsname golang-github-containers-common golang-github-containers-image
  gsettings-desktop-schemas libavahi-client3 libavahi-common-data libavahi-common3 libavahi-glib1 libdconf1 libostree-1-1
  libproxy1v5 libslirp0 libsoup2.4-1 libsoup2.4-4-common libyajl2 podman python3-docker python3-dockerpty python3-docopt
  python3-dotenv python3-texttable python3-websocket session-migration slirp4netns uidmap
```

- We have to create a role now like we have created a policy for a user. Role is important for connecting the ec2 with the AWS ECR.
- Go to aws management console and navigate to IAM and click on roles.



The screenshot shows the AWS IAM Dashboard. In the left sidebar under 'Access management', the 'Roles' option is highlighted with a red box. Other options like 'User groups', 'Users', 'Policies', 'Identity providers', 'Account settings', and 'Root access management' are also listed. The main content area is titled 'IAM Dashboard' and includes sections for 'Security recommendations' (with two items) and 'IAM resources' (with tabs for 'User groups', 'Users', and 'Roles').

- Create role



The screenshot shows the 'Roles' list page with 2 items. The 'Create role' button at the top right is highlighted with a red box. The table lists two roles: 'AWSServiceRoleForSupport' and 'AWSServiceRoleForTrustedAdvisor'. The columns include 'Role name', 'Trusted entities', and 'Last activity'.

Role name	Trusted entities	Last activity
AWSServiceRoleForSupport	AWS Service: support (Service-Linker)	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linker)	-

- Select Trusted entity type as AWS service and use case is EC2.
- Trusted entity type**

<input checked="" type="radio"/> AWS service Allow AWS services like EC2, Lambda, or others to perform actions in this account.	<input type="radio"/> AWS account Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
<input type="radio"/> Web identity Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.	<input type="radio"/> SAML 2.0 federation Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
<input type="radio"/> Custom trust policy Create a custom trust policy to enable others to perform actions in this account.	

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

EC2

Choose a use case for the specified service.

Use case

EC2

Allows EC2 instances to call AWS services on your behalf.

- Click on AmazonEC2ContainerRegistryFullAccess

Add permissions Info

Permissions policies (1/990) Info

Choose one or more policies to attach to your new role.

Filter by Type

contain



All types

14 matches

Policy name

Type

Description



AmazonEC2ContainerRegi...

AWS managed

Provides administrative

AmazonEC2ContainerRegistryFullAccess

Provides administrative access to Amazon ECR resources

```

1  {
2      "Version": "2012-10-17",
3      "Statement": [
4          {
5              "Effect": "Allow",
6              "Action": [
7                  "ecr:*",
8                  "cloudtrail:LookupEvents"

```

- Give a role name and make sure you remember this role name or note it somewhere and click on create role.

Role details

Role name

Enter a meaningful name to identify this role.

Maximum 64 characters. Use alphanumeric and '+,-,_' characters.

Description

- Once the role is created go to ec2 instances and locate your cd_server instance and click on actions on the top and click on modify IAM role.

The screenshot shows the AWS EC2 Instances page. There are two instances listed: 'jenkins_server' (running) and 'cd_server' (running). The 'Actions' menu is open for the 'cd_server' instance, with several options like 'Connect', 'View details', 'Manage instance state', 'Instance settings', 'Networking', 'Security' (which is also highlighted with a red box), 'Image and templates', and 'Monitor and troubleshoot'. The 'Modify IAM role' option is specifically highlighted with a red box.

- Choose your newly create role and update IAM role.

Modify IAM role Info

Attach an IAM role to your instance.

Instance ID

i-0fe00e525ba765c0a (cd_server)

IAM role

Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

ec2ecrrole

Create new IAM role



No IAM Role

Choose this option to detach an IAM role

ec2ecrrole

arn:aws:iam::980921742537:instance-profile/ec2ecrrole

Cancel

Update IAM role

- Now everything is set time to push ECR into the CD server which is ubuntu in our case.
- Navigate to ECR on aws management console and click on view push commands.

The screenshot shows the AWS ECR console with the path: Amazon ECR > Private registry > Repositories > jenkinsintegration. The 'Images (1)' section is displayed, showing one image entry. The 'View push commands' button is highlighted with a red box.

	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
<input type="checkbox"/>	5	Image	November 19, 2024, 20:38:34 (UTC-05)	229.86	<input type="checkbox"/> Copy URI	<input type="checkbox"/> sha256:b2d795d...

- We got certain commands for the integration, copy the first command and paste it on your ubuntu terminal which is cd server.

macOS / Linux Windows

Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see [Getting Started with Amazon ECR](#).

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see [Registry Authentication](#).

1. Retrieve an authentication token and authenticate your Docker client to your registry. Use the AWS CLI:

```
 aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin  
980921742537.dkr.ecr.us-east-1.amazonaws.com
```

Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.

2. Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions [here](#). You can skip this step if your image is already built:

```
 docker build -t jenkinsintegration .
```

3. After the build completes, tag your image so you can push the image to this repository:

```
 docker tag jenkinsintegration:latest 980921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration:latest
```

4. Run the following command to push this image to your newly created AWS repository:

```
 docker push 980921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration:latest
```

```
root@ip-172-31-83-200:~# aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 980921742537.dkr.ecr.us-east-1.amazonaws.com
Emulate Docker CLI using podman. Create /etc/containers/nodocker to quiet msg.
Login Succeeded!
```

Now I am able to access the aws ecr.

- Time to run the command to transfer the container and image to cd server.
Use following command where you can modify the app name, port and URI.

```
docker container run --name=firstapp -p 8000:8000 --detach
980921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration:5
```

I got the uri from here in the screenshot below.

The screenshot shows the Amazon ECR console interface. In the top navigation bar, the path is: [Amazon ECR](#) > [Private registry](#) > [Repositories](#) > [jenkinsintegration](#) > sha256:b2d795dc90a55a838fed4fad4df97f1beb50fb4ff. Below this, there's a large "Image" section. Under "Details", it shows "Image tags" with "5" listed. The "URI" field contains "980921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration:5", which is highlighted with a red rectangular border. Below the URI, the "Digest" field shows "sha256:b2d795dc90a55a838fed4fad4df97f1beb50fb4f0be6414b5a56ebc6b4ffb756".

- Once the command run successfully try to look for container through the two commands.
 - docker image ls
 - docker container ls

```
root@ip-172-31-83-200:~# docker container run --name=firstapp -p 8080:8080 --detach 980921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration:5
Emulate Docker CLI using podman. Create /etc/containers/nodocker to quiet msg.
Trying to pull 980921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration:5...
Getting image source signatures
Copying blob 899046e4a240 done
Copying blob 849fae9bc522 done
Copying blob 5ce47bbf12c8 done
Copying config e90a3e7289 done
Writing manifest to image destination
Storing signatures
cf25e3cd91fae58f041f5893b93a44056d8233e4fcf2f3886a5cc4a35a55a601
root@ip-172-31-83-200:~# docker container ls
Emulate Docker CLI using podman. Create /etc/containers/nodocker to quiet msg.
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
cf25e3cd91fa 980921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration:5 5 seconds ago Up 6 seconds ago 0.0.
root@ip-172-31-83-200:~#
```

As we can see our container is running on port 8000 successfully.

- As we are using the port 8080 (we are using port 8080 because docker container on aws ecr is using 8080 so we have to use the same port) we need to allow it on the inbound rules of security groups of our ubuntu instance on the aws management console.

sg-01de45c83eb4da420 - launch-wizard-4 Actions ▾

Details			
Security group name launch-wizard-4	Security group ID sg-01de45c83eb4da420	Description launch-wizard-4 created 2024-11-20T01:51:16.270Z	VPC ID vpc-07fc7cf613b687ce6
Owner 980921742537	Inbound rules count 1 Permission entry	Outbound rules count 1 Permission entry	

[Inbound rules](#) [Outbound rules](#) [Sharing - new](#) [VPC associations - new](#) [Tags](#)

Inbound rules (1)					
Edit inbound rules C Manage tags Search < 1 > ⚙️					
<input type="checkbox"/>	Name	Security group rule...	IP version	Type	Protocol
<input type="checkbox"/>	-	sgr-0223130ab7f7343...	IPv4	SSH	TCP

Edit inbound rules Info
 Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules <small>Info</small>						
Security group rule ID	Type	Protocol	Port range	Source	Description - optional	
	Info	Info		Info	Info	
sgr-03d45b9c286ce481b	Custom TCP	TCP	8080	Cus... ▼	<input type="text" value="0.0.0.0/0"/> X	Delete
sgr-0b7db6b0af261c9c6	Custom TCP	TCP	8000	Cus... ▼	<input type="text" value="0.0.0.0/0"/> X	Delete
sgr-0223130ab7f734341	SSH	TCP	22	Cus... ▼	<input type="text" value="0.0.0.0/0"/> X	Delete
Add rule						

[Cancel](#) [Preview changes](#) [Save rules](#)

- As we enabled the port 8080 the application is running right now.
- We are going to access our application through the url which is IP address of the ubuntu vm and port 8080 but we need more parameters to it.

- How to look for it? Go to jenkins server vm go to “cd /var/lib/jenkins/workspace” and choose your pipeline name, in our case it is docker_jenkins_maven.

```
[root@ip-172-31-84-17 ~]# cd /var/lib/jenkins/workspace
[root@ip-172-31-84-17 workspace]# ls
docker_jenkins_maven  maven-jenkins-git-pipeline  snyk_maven_pipeline  sonarserver-jenkins
docker_jenkins_maven@tmp  maven-jenkins-git-pipeline@tmp  snyk_maven_pipeline@tmp  sonarserver-jenkins@tmp
[root@ip-172-31-84-17 workspace]# cd docker_jenkins_maven
[root@ip-172-31-84-17 docker_jenkins_maven]# ls
ansible.yaml  generate_kubeconfig_eks.sh  mvnw  sonar-project.properties  springboot-deployment.yml  src
deployment.yaml  Jenkinsfile  mvnw.cmd  springboot-deployment-acr.yml  springboot-service-acr.yaml  target
Dockerfile  kubernetes  pom.xml  springboot-deployment-ecr.yml  springboot-service.yaml
[root@ip-172-31-84-17 docker_jenkins_maven]# cd src
[root@ip-172-31-84-17 src]# ls
main  test
[root@ip-172-31-84-17 src]# cd main/resources
[root@ip-172-31-84-17 resources]# ls
application.properties
[root@ip-172-31-84-17 resources]# cat application.properties
server.port=8080
server.servlet.context-path=/course-svc
```

- Once you try to access the webpage through this url for this application you can access it successfully.



AWS EKS Cluster



- EKS cluster Elastic Kubernetes Cluster has bunch of nodes running the docker images which will be controlled by the master node.
- Let's create a EKS cluster on AWS. Navigate to aws console and search for EKS.
- Click on create cluster for creating a new cluster, if you already created a cluster you can simply register it. But I am creating a new one.
- Give a name for your cluster and it is asking for IAM role which we need to create for AWS EKS.

Configure cluster

Cluster configuration [Info](#)

Name
Enter a unique name for this cluster. This property cannot be changed after the cluster is created.

The cluster name should begin with letter or digit and can have any of the following characters: the set of Unicode letters, digits, hyphens and underscores. Maximum length of 100.

Cluster IAM role [Info](#)
Select the Cluster IAM role to allow the Kubernetes control plane to manage AWS resources on your behalf. This cannot be changed after the cluster is created. To create a new custom role, follow the instructions in the [Amazon EKS User Guide](#).

Select role

[Filter roles](#)

No roles found. Follow the link above to create a new role.

Create recommended role

- Open a new windows and search for IAM and click on roles and create role.

The screenshot shows the AWS IAM Roles management interface. On the left, there's a sidebar with 'Identity and Access Management (IAM)' selected. The main area is titled 'Roles (3)'. It lists three roles: 'AWSServiceRoleForSupport' (trusted by 'support (Service-Linked)'), 'AWSServiceRoleForTrustedAdvisor' (trusted by 'trustedadvisor (Service)'), and 'ec2crole' (trusted by 'ec2'). A red box highlights the 'Create role' button at the top right of the table header.

- Select AWS service and use case will be EKS> EKS-cluster like below.

Select trusted entity Info

The screenshot shows the 'Select trusted entity' step of a wizard. The title is 'Trusted entity type'. There are five options: 'AWS service' (selected), 'AWS account', 'Web identity', 'SAML 2.0 federation', and 'Custom trust policy'. Each option has a brief description below it. The 'AWS service' option is highlighted with a blue border.

Trusted entity type	Description
<input checked="" type="radio"/> AWS service	Allow AWS services like EC2, Lambda, or others to perform actions in this account.
<input type="radio"/> AWS account	Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
<input type="radio"/> Web identity	Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
<input type="radio"/> SAML 2.0 federation	Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
<input type="radio"/> Custom trust policy	Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

EKS



Choose a use case for the specified service.

Use case

- EKS - Service
Allows EKS to manage clusters on your behalf.
- EKS - Cluster
Allows the cluster Kubernetes control plane to manage AWS resources on your behalf.
- EKS - Nodegroup
Allows EKS to manage nodegroups on your behalf.
- EKS - Fargate pod
Allows access to other AWS service resources that are required to run Amazon EKS pods on AWS Fargate.
- EKS - Fargate profile
Allows EKS to run Fargate tasks.
- EKS - Connector
Allows access to other AWS service resources that are required to connect to external clusters
- EKS Local - Outpost
Allows Amazon EKS Local to call AWS services on your behalf.
- EKS - Pod Identity

Keep everything default on add permissions

Add permissions Info

Permissions policies (1) Info

The type of role that you selected requires the following policy.

Policy name

▲ | Type



AmazonEKSClusterPolicy

AWS managed

► Set permissions boundary - optional

[Cancel](#)

[Previous](#)

[Next](#)

Give role name

Name, review, and create

Role details

Role name

Enter a meaningful name to identify this role.

Maximum 64 characters. Use alphanumeric and '+=, @-' characters.

Description

Add a short explanation for this role.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=,. @-/[\[]!#\$%^*();":'

Click on create role

Step 3: Add tags

Add tags - optional Info

Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

You can add up to 50 more tags.

Once the role is create head back to EKS cluster creation process.

- Now select the role you have created for EKS cluster.

Cluster configuration Info

Name
Enter a unique name for this cluster. This property cannot be changed after the cluster is created.

The cluster name should begin with letter or digit and can have any of the following characters: the set of Unicode letters, digits, hyphens and underscores. Maximum length of 100.

Cluster IAM role Info
Select the Cluster IAM role to allow the Kubernetes control plane to manage AWS resources on your behalf. This cannot be changed after the cluster is created. To create a new custom role, follow the instructions in the [Amazon EKS User Guide](#).

<input type="text" value="eksclusterrole"/> <input type="button" value="Filter roles"/>	<input checked="" type="checkbox" value="arn:aws:iam::980921742537:role/eksclusterrole"/>	<input type="button" value="Create recommended role"/>
--	---	--

- Keep cluster endpoint access as public and leave everything default for this project, you can modify networking and other policies for your use case in the future.

Cluster endpoint access [Info](#)

Configure access to the Kubernetes API server endpoint.

Public

The cluster endpoint is accessible from outside of your VPC. Worker node traffic will leave your VPC to connect to the endpoint.

Public and private

The cluster endpoint is accessible from outside of your VPC. Worker node traffic to the endpoint will stay within your VPC.

Private

The cluster endpoint is only accessible through your VPC. Worker node traffic to the endpoint will stay within your VPC.

► **Advanced settings**

- You can play around here on control plane logs if you want those pertaining logs.

Control plane logs [Info](#)

Send audit and diagnostic logs from the Amazon EKS control plane to CloudWatch Logs.

API server

Logs pertaining to API requests to the cluster.

Audit

Logs pertaining to cluster access via the Kubernetes API.

Authenticator

Logs pertaining to authentication requests into the cluster.

Controller manager

Logs pertaining to state of cluster controllers.

Scheduler

Logs pertaining to scheduling decisions.

I am leaving everything default here as I don't need those.

- Add-on if you need anything example guardDuty for protection you can use, I am going to destroy this application later so I don't need these.

Select add-ons

Review the add-ons from multiple categories, then select add-ons to enhance your cluster.

Amazon EKS add-ons (11) [Info](#)

 kube-proxy Info Enable service networking within your cluster. Category: networking	 CoreDNS Info Enable service discovery within your cluster. Category: networking	 Amazon VPC CNI Info Enable pod networking within your cluster. Category: networking
 Amazon GuardDuty EKS Runtime	 CSI Snapshot Controller Info	 AWS Distro for OpenTelemetry Info

- Keep everything default and hit create cluster, Note: we can modify certain things if we need to.

Selected add-ons version (4)

Add-on name	Version
coredns	v1.11.3-eksbuild.1
eks-pod-identity-agent	v1.3.4-eksbuild.1
kube-proxy	v1.31.2-eksbuild.3
vpc-cni	v1.19.0-eksbuild.1

EKS Pod Identity (1)

Add-on name	IAM role	Service account
vpc-cni	Not set	aws-node

[Cancel](#) [Previous](#) [Create](#)

- I got the error while creating saying that one of subnet don't support creation control plane on subnet us-east-1c so I will remove that from my subnet list and click on create cluster again.

✖ Cannot create cluster 'firstekscluster' because EKS does not support creating control plane instances in us-east-1e, the targeted availability zone. Retry cluster creation using control plane subnets that span at least two of these availability zones: us-east-1a, us-east-1b, us-east-1c, us-east-1d, us-east-1f. Note, post cluster creation, you can run worker nodes in separate subnets/availability zones from control plane subnets/availability zones passed during cluster creation

Subnets [Info](#)

Choose the subnets in your VPC where the control plane may place elastic network interfaces (ENIs) to facilitate communication with your cluster. To create a new subnet, go to the corresponding page in the [VPC console](#).

Select subnets

▼

C

Clear selected subnets

subnet-0197abf956d9ef180 ✖ us-east-1a 172.31.0.0/20	subnet-018e5b9d80ace0361 ✖ us-east-1c 172.31.16.0/20
subnet-09a63e9da9eb0299a ✖ us-east-1b 172.31.80.0/20	subnet-0c46efd5de2ea1772 ✖ us-east-1f 172.31.64.0/20
subnet-0ee53f209eba290e8 ✖ us-east-1d 172.31.32.0/20	

Now I only have 5 subnets, which is enough and fine.

- My first cluster is starting and it is in creation process.

[EKS](#) > [Clusters](#) > [firstekscluster](#)

firstekscluster

C

Delete cluster

View dashboard

▼ Cluster info [Info](#)

Status
Creating

Kubernetes version
[Info](#)
1.31

Support period
Standard support until November 26, 2025

Provider
EKS

Cluster health issues

0

Upgrade insights

0

Wait until it creates it completely.

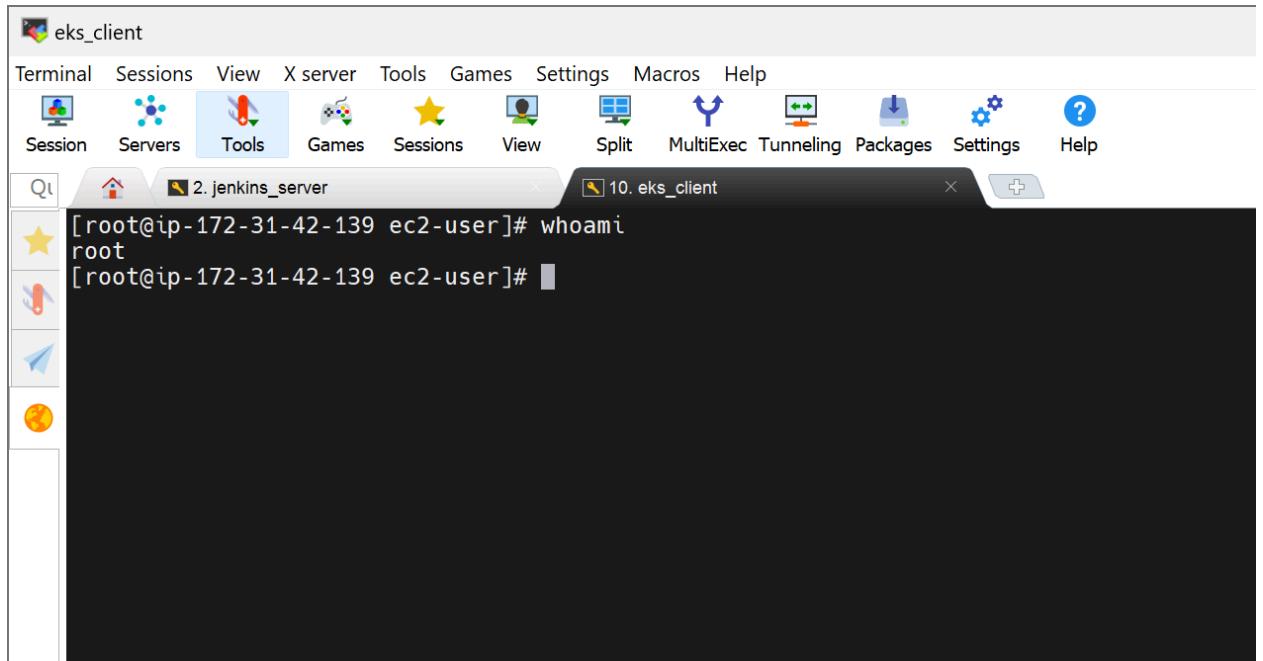
- We need to have a new instance or EKS client which can interact with the cluster so we will create one client machine which is ec2 instance.

The screenshot shows the AWS Launch Wizard interface for creating a new Amazon Machine Image (AMI) and instance type. The process is divided into several sections:

- Name:** The name is set to "eks_client".
- Application and OS Images (Amazon Machine Image):** A search bar is available to find application and OS images. Previews of various AMIs are shown, including Amazon Linux, macOS, Ubuntu, Windows, Red Hat, and SUSE. A "Browse more AMIs" link is provided.
- Instance type:** The selected instance type is "t2.micro", which is listed as "Free tier eligible". It includes details about its family, vCPUs, memory, and current generation. A dropdown arrow indicates more options are available. Other generation options are shown as "All generations".
- Key pair (login):** A key pair named "jenkins_server" is selected. An option to "Create new key pair" is also present.
- Summary:** On the right, a summary panel shows the configuration: 1 instance of the Amazon Linux 2023 AMI (ami-012967cc5a8c9f891), using the t2.micro instance type, and 1 volume (8 GiB). It also notes the "Free tier" availability for the instance type.

Keep everything default and launch the instance.

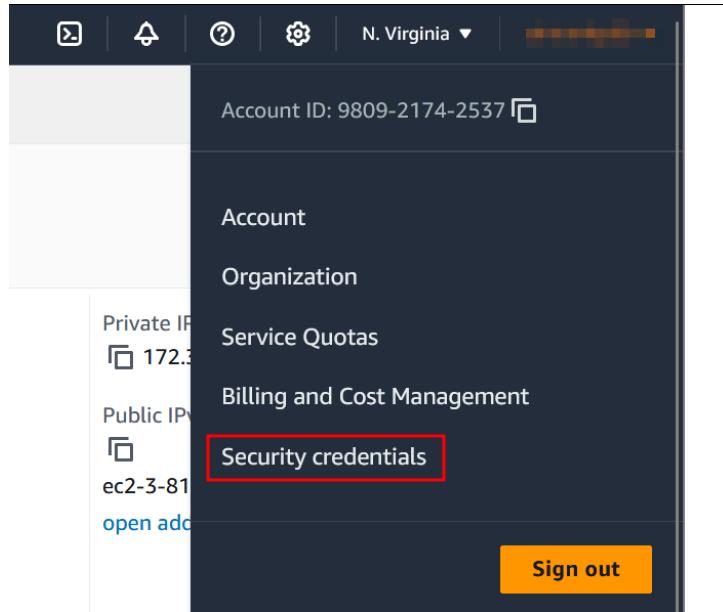
- Once the instance is running we will access it using mobaxterm which is ssh client to access the machine.



- In Amazon linux aws cli is configured by default for communicating it with aws.

```
[root@ip-172-31-42-139 ec2-user]# aws --version
aws-cli/2.15.30 Python/3.9.16 Linux/6.1.115-126.197.amzn2023.x86_64 source/x86_64.amzn.2023 prompt/off
[root@ip-172-31-42-139 ec2-user]#
```

- Lets create a access key for our aws and allow this ec2 instance to communicate with our AWS account.
- Navigate to aws management console, and click on your account name and select security credentials.



- Click on create access key.

Access keys (1)

Actions ▾ **Create access key**

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. [Learn more](#)

Access key ID	Created on	Access key last used	Region last used	Service last used	Status
[REDACTED]	16 days ago	None	N/A	N/A	Active

- Copying the root access key is not recommended but I am going to terminate this instance, but if you are using for production and important work, make sure your create a user at IAM and configure the policies and create access key for the user rather than using with the root account.
- Save the access key and secret access key for using it at ec2.

Retrieve access key Info

Access key

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key



Secret access key

***** [Show](#)

- Use AWS configure and add the access and secret access key to enable the communication between ec2 and aws account.

```
[root@ip-172-31-42-139 ec2-user]# aws configure
AWS Access Key ID [None]: [REDACTED]
AWS Secret Access Key [None]: [REDACTED]
Default region name [None]: us-east-1
Default output format [None]: json
[root@ip-172-31-42-139 ec2-user]#
```

- We can check eks cluster status from ec2 instance for checking whether the ec2 is properly connected with ec2.

```
aws eks --region us-east-1 describe-cluster --name firstekscluster --query cluster.status
```

```
[root@ip-172-31-42-139 ec2-user]# aws eks --region us-east-1 describe-cluster --name firstekscluster --query cluster.status
"ACTIVE"
[root@ip-172-31-42-139 ec2-user]#
```

Change the cluster name parameter with your own cluster and region as well.

- We will add a config file for kubernetes here using the command.

```
[root@ip-172-31-42-139 ec2-user]# aws eks --region us-east-1 update-kubeconfig --name firstekscluster
Added new context arn:aws:eks:us-east-1:980921742537:cluster/firstekscluster to /root/.kube/config
[root@ip-172-31-42-139 ec2-user]# ls -ltra
total 12
-rw-r--r--. 1 ec2-user ec2-user 492 Jan 28 2023 .bashrc
-rw-r--r--. 1 ec2-user ec2-user 141 Jan 28 2023 .bash_profile
-rw-r--r--. 1 ec2-user ec2-user 18 Jan 28 2023 .bash_logout
drwxr-xr-x. 3 root root 22 Nov 20 23:49 ..
drwx----- 2 ec2-user ec2-user 29 Nov 20 23:49 .ssh
drwx----- 3 ec2-user ec2-user 74 Nov 20 23:49 .
[root@ip-172-31-42-139 ec2-user]# cd ~
[root@ip-172-31-42-139 ~]# ls -ltra
total 24
-rw-r--r--. 1 root root 129 Feb 2 2023 .tcshrc
-rw-r--r--. 1 root root 100 Feb 2 2023 .cshrc
-rw-r--r--. 1 root root 429 Feb 2 2023 .bashrc
-rw-r--r--. 1 root root 141 Feb 2 2023 .bash_profile
-rw-r--r--. 1 root root 18 Feb 2 2023 .bash_logout
dr-xr-xr-x. 18 root root 237 Nov 13 18:25 ..
drwx----- 2 root root 29 Nov 20 23:49 .ssh
drwxr-xr-x. 2 root root 39 Nov 21 00:02 .aws
-rw----- 1 root root 20 Nov 21 00:12 .lessht
drwxr-xr-x. 2 root root 20 Nov 21 00:13 .kube
dr-xr-x--- 5 root root 144 Nov 21 00:13 .
[root@ip-172-31-42-139 ~]#
```

- We need to install kubectl, we need kubectl to interact with kubernetes API. So navigate to this website and copy the curl command based on your version of EKS cluster you have chosen while creating it.

[“\[https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html#linux_amd64_kubectl\]\(https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html#linux_amd64_kubectl\)”](https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html#linux_amd64_kubectl)

Linux (amd64)

1. Download the `kubectl` binary for your cluster's Kubernetes version from Amazon S3.

- Kubernetes 1.31

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.2/2024-11-15/bin/linux/amd64/kubectl
```

- Kubernetes 1.30

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.6/2024-11-15/bin/linux/amd64/kubectl
```

- Kubernetes 1.29

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.10/2024-11-15/bin/linux/amd64/kubectl
```

- Kubernetes 1.28

- Follow these two steps as well for adding it to path and including it in home directory.

3. Apply execute permissions to the binary.

```
chmod +x ./kubectl
```

4. Copy the binary to a folder in your PATH. If you have already installed a version of `kubectl`, then we recommend creating a `$HOME/bin/kubectl` and ensuring that `$HOME/bin` comes first in your `$PATH`.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

```
[root@ip-172-31-42-139 ~]# curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.2/2024-11-15/bin/linux/amd64/kubectl
% Total    % Received % Xferd  Average Speed   Time   Time  Current
          Dload  Upload   Total Spent  Left  Speed
100 53.7M  100 53.7M    0     0  8413k      0  0:00:06  0:00:06  --- 9517k
[root@ip-172-31-42-139 ~]# ls
kubectl
[root@ip-172-31-42-139 ~]# chmod +x kubectl
[root@ip-172-31-42-139 ~]# mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
[root@ip-172-31-42-139 ~]#
```

```
[root@ip-172-31-42-139 ~]# kubectl version --client
Client Version: v1.31.2-eks-94953ac
Kustomize Version: v5.4.2
[root@ip-172-31-42-139 ~]
```

- EKSctl how to install it? Let's look for the amazon website and do the following steps.

```
# Download the eksctl binary
```

```
curl -s --location
```

```
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
```

```
# Move the binary to a directory in your PATH
```

```
sudo mv /tmp/eksctl /usr/local/bin
```

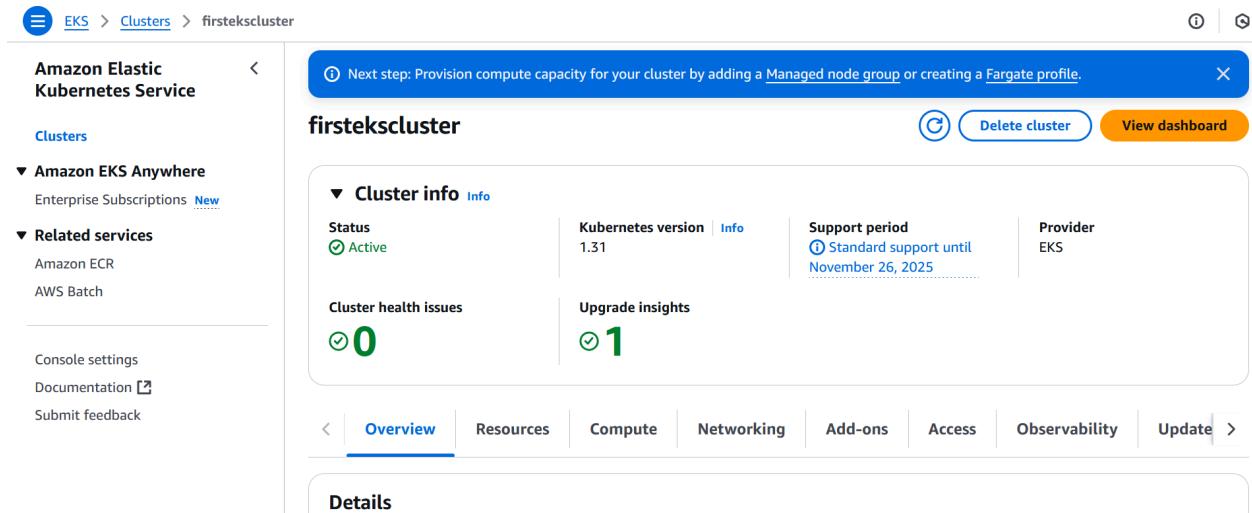
eksctl version

```
[root@ip-172-31-42-139 ~]# curl -s --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
[root@ip-172-31-42-139 ~]# sudo mv /tmp/eksctl /usr/local/bin
[root@ip-172-31-42-139 ~]# eksctl version
0.194.0
```

- Check whether the kubectl able to communicate with kubernetes.

```
[root@ip-172-31-42-139 ~]# kubectl get svc
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes     ClusterIP  10.100.0.1  <none>        443/TCP   6h4m
```

- We successfully deployed the master node, now we need to add the worker nodes and node group.
- Go to aws console and open the EKS service.



Navigate to compute

The screenshot shows the AWS EKS Cluster Overview page for a cluster named 'firstekscluster'. At the top right are three buttons: a blue 'C' icon, 'Delete cluster', and 'View dashboard'. Below the cluster name is a section titled 'Cluster info' with tabs for Status (Active), Kubernetes version (1.31), Support period (Standard support until November 26, 2025), and Provider (EKS). Under 'Cluster health issues', there are two items: '0' (green) and '1' (green). Below this is a navigation bar with tabs: Overview, Resources, Compute (which is selected and highlighted with a red box), Networking, Add-ons, Access, Observability, and Update. The 'Compute' tab has a sub-section titled 'Nodes (0)' with a search bar labeled 'Filter Nodes by property or value'. A message at the bottom says 'No node groups'.

Scroll down and click on add node group for a node group where we will run the worker nodes which is ec2 instances.

The screenshot shows the 'Node groups' page for the same cluster. It includes a header with 'Edit', 'Delete', and 'Add node group' buttons. Below is a table with columns: Group name, Desired size, AMI release version, Launch template, and Status. A message states 'No node groups' and 'This cluster does not have any node groups.' At the bottom is a large red-bordered button labeled 'Add node group'.

Add name to your node group and now we will need to create a IAM role for this node group.

Node group configuration

These properties cannot be changed after the node group is created.

Name

Assign a unique name for this node group.

group1

The node group name should begin with letter or digit and can have any of the following characters: the set of Unicode letters, digits, hyphens and underscores. Maximum length of 63.

Node IAM role Info

Select the IAM role that will be used by the nodes. To create a new role, go to the [IAM console](#).

ec2ecrrole

Filter roles

No roles found. Follow the link above to create a new role.

[Learn more](#)



[Create recommended](#)

- Navigate to AWS IAM and hit create role

Roles (5) <small>Info</small>		
<input type="checkbox"/> Role name	Trusted entities	Last activity
<input type="checkbox"/> AWSServiceRoleForAmazonEKS	AWS Service: eks (Service-Linked Role)	19 minutes ago
<input type="checkbox"/> AWSServiceRoleForSupport	AWS Service: support (Service-Linked Role)	-
<input type="checkbox"/> AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linked Role)	-
<input type="checkbox"/> ec2ecrrole	AWS Service: ec2	1 hour ago
<input type="checkbox"/> eksclusterrole	AWS Service: eks	1 hour ago

- Click on AWS service as trusted entity type and use case as EC2 as we are going to deploy EC2 as a worker nodes.

Trusted entity type

AWS service

Allow AWS services like EC2, Lambda, or others to perform actions in this account.

AWS account

Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

Web identity

Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

SAML 2.0 federation

Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

Custom trust policy

Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

EC2

Choose a use case for the specified service.

Use case

EC2

Allows EC2 instances to call AWS services on your behalf.

- Type 'eks' and add these two roles.

Filter by type			
	Policy name	Type	Description
<input checked="" type="checkbox"/>	AmazonEKS_CNI_Policy	AWS managed	This policy provides the Amazon VPC ...
<input type="checkbox"/>	AmazonEKSBlockStorageP...	AWS managed	Policy attached to the EKS Cluster Rol...
<input type="checkbox"/>	AmazonEKSClusterPolicy	AWS managed	This policy provides Kubernetes the pe...
<input type="checkbox"/>	AmazonEKSComputePolicy	AWS managed	Policy attached to the EKS Cluster Rol...
<input type="checkbox"/>	AmazonEKSFargatePodEx...	AWS managed	Provides access to other AWS service r...
<input type="checkbox"/>	AmazonEKSLoadBalancing...	AWS managed	Policy attached to the EKS Cluster Rol...
<input type="checkbox"/>	AmazonEKSLocalOutpostC...	AWS managed	This policy provides permissions to EK...
<input type="checkbox"/>	AmazonEKSNetworkingPo...	AWS managed	Policy attached to the EKS Cluster Rol...
<input type="checkbox"/>	AmazonEKSServicePolicy	AWS managed	This policy allows Amazon Elastic Cont...
<input type="checkbox"/>	AmazonEKSVPCCResourceC...	AWS managed	Policy used by VPC Resource Controlle...
<input type="checkbox"/>	AmazonEKSWorkerNodeM...	AWS managed	This policy allows Amazon EKS worker ...
<input checked="" type="checkbox"/>	AmazonEKSWorkerNodeP...	AWS managed	This policy allows Amazon EKS worker ...

- Add this role as well for ec2 container registry read only.

Search bar: ec2container

Filter: All types

Results: 8 matches

Policy name	Type	Description
<input type="checkbox"/> AmazonEC2ContainerRegi...	AWS managed	Provides administrative...
<input type="checkbox"/> AmazonEC2ContainerRegi...	AWS managed	Provides full access to...
<input type="checkbox"/> AmazonEC2ContainerRegi...	AWS managed	Provides access to pul...
<input checked="" type="checkbox"/> AmazonEC2ContainerRegi...	AWS managed	Provides read-only ac...

AmazonEC2ContainerRegistryReadOnly

Provides read-only access to Amazon EC2 Container Registry repositories.

```

1  {
2      "Version": "2012-10-17",
3      "Statement": [
4          {
5              "Effect": "Allow",
6              "Action": [
7                  "ecr:GetAuthorizationToken",
8                  "ecr:BatchCheckLayerAvailability",
9                  "ecr:GetDownloadUrlForLayer",
10                 "ecr:getRepositoryPolicy",

```

- Give a role name and create role.

Role details

Role name

Enter a meaningful name to identify this role.

Maximum 64 characters. Use alphanumeric and '+=-,.@-_' characters.

Description

Add a short explanation for this role.

Allows EC2 instances to call AWS services on your behalf.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=,. @-/[\{\}]!#\$%

Step 1: Select trusted entities

Step 2. Add permissions

Permissions policy summary

Policy name	Type	Attached as
AmazonEC2ContainerRegistryReadOnly	AWS managed	Permissions policy
AmazonEKS_CNI_Policy	AWS managed	Permissions policy
AmazonEKSWorkerNodePolicy	AWS managed	Permissions policy

Step 3: Add tags

Add tags - optional Info

Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

[Cancel](#)

[Previous](#)

[Create role](#)

- Once the role is created add it here.

Node group configuration

These properties cannot be changed after the node group is created.

Name

Assign a unique name for this node group.

group1

The node group name should begin with letter or digit and can have any of the following characters: the set of Unicode letters, digits, hyphens and underscores. Maximum length of 63.

Node IAM role Info

Select the IAM role that will be used by the nodes. To create a new role, go to the [IAM console](#).

eksgroup1role



[Create recommended role](#)

ⓘ The selected role must not be used by a self-managed node group as this could lead to a service interruption upon managed node group deletion.

[Learn more](#)

- Choose the image based on your needs and keep everything default and create the node group.

Node group compute configuration

These properties cannot be changed after the node group is created.

AMI type [Info](#)

Select the EKS-optimized Amazon Machine Image for nodes.

Amazon Linux 2 (AL2_x86_64)



Capacity type

Select the capacity purchase option for this node group.

On-Demand



Instance types [Info](#)

Select instance types you prefer for this node group.

Enter an instance type

t2.micro

vCPU: 1 vCPU Memory: 1 GiB Network: Low to Moderate Max ENI: 2 Max IPs: 4



Disk size

Select the size of the attached EBS volume for each node.

20



GiB

Update here, don't use t2.micro it will produce error so use t3.medium

Node group creation in progress

group1 is now being created. This process may take several minutes.



group1



Edit

Delete

Node group configuration [Info](#)

Kubernetes version

1.31

AMI type [Info](#)

AL2_x86_64

Status

Creating

AMI release version [Info](#)

1.31.2-20241115

Instance types

t2.micro

Disk size

20 GiB

< Details

Nodes

Health issues 0

Kubernetes labels

Update config

Kubernetes taints

U >

- Once the node group is created successfully, we can see 2 more ec2 instances are running on EC2 which are worker nodes.

Node group configuration Info

Kubernetes version 1.31	AMI type <small>Info</small> AL2_x86_64	Status Active
AMI release version <small>Info</small> 1.31.2-20241115	Instance types t2.micro	Disk size 20 GiB

Details Nodes Health issues 0 Kubernetes labels Update config Kubernetes taints U >

We can see two ec2 worker nodes are running successfully.

	i-096d4b4abc21e5b12	Running <small>View logs</small> <small>Metrics</small>	t2.micro	2/2 checks passed <small>View alarms</small> +
	i-0da7c23c7116dc8fa	Running <small>View logs</small> <small>Metrics</small>	t2.micro	2/2 checks passed <small>View alarms</small> +

```
[root@ip-172-31-42-139 ~]# kubectl get nodes
NAME           STATUS   ROLES      AGE    VERSION
ip-172-31-24-180.ec2.internal   Ready    <none>    26m   v1.31.2-eks-94953ac
ip-172-31-34-228.ec2.internal   Ready    <none>    26m   v1.31.2-eks-94953ac
```

- Now we will do the complete process of building the project, creating the image and adding it to our EKS cluster.
- Go the jenkins server and create a new pipeline and modify the github repo with this one, "<https://github.com/Shikhar82/techstart-jenkins-deploy-to-k8-repo.git>".

Script ?

```

8   registryCredential = 'awsecr_jenkins_user'
9 }
10 stages {
11   stage ("Checking out the git project") {
12     steps {
13       git branch: 'main', url: 'https://github.com/Shikhar82/techstart-jenkins-deploy-to-k8-repo.git'
14     }
15   }
16
17   stage ("Checking bugs on sonarcloud") {
18     steps {
19       sh 'mvn clean verify sonar:sonar -Dsonar.projectKey=devsecopskp -Dsonar.organization=devsecopskp -Dsonar.host.url=https://sonarcloud.io'
20     }
21   }
22   // stage ("Build the package") {
23   //   steps {

```

Change it to main branch as well as that github repository belongs to main branch.

Click build now and go to next step once the build is finished, It will take some time 😕

```
> finaleksdeployment < > #1 <

92a4e8a3140f: Pushed
954be880ce7c: Pushed
1: digest: sha256:eb25440a618105674db545c1629eff045f18d54e2281b5462ea146a29c0e362f size: 1372
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withDockerRegistry
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

- Great, we have to focus on the .git repository, depends on your own project, where we need to have two files, one is deployment.yaml and service.yaml files. The deployment.yaml file has the URI and app information
- After this I installed the docker at eks_client before that I changed the URI on the deployment.yaml file. You will find this file at the github repository
[“https://github.com/Shikhar82/techstart-jenkins-deploy-to-k8-repo.git”](https://github.com/Shikhar82/techstart-jenkins-deploy-to-k8-repo.git)
For editing the deployment file, you can fork the above repository and modify the deployment file based on your URI that you find in you images at AWS ECR.

```

2. jenkins_server 10. eks_client
(9/10): docker-25.0.6-1.amzn2023.0.2.x86_64.rpm 29 MB/s | 44 MB 00:01
(10/10): containerd-1.7.23-1.amzn2023.0.1.x86_64.rpm 20 MB/s | 36 MB 00:01
Total 47 MB/s | 84 MB 00:01
-----+-----+-----+
Preparing: 1/1
Installing : runc-1.1.14-1.amzn2023.0.1.x86_64 1/10
Installing : containerd-1.7.23-1.amzn2023.0.1.x86_64 2/10
Running scriptlet: containerd-1.7.23-1.amzn2023.0.1.x86_64 2/10
Installing : pigz-2.5-1.amzn2023.0.3.x86_64 3/10
Installing : libnftnl-1.2.2-2.amzn2023.0.2.x86_64 4/10
Installing : libnfnetwork-1.0.1-19.amzn2023.0.2.x86_64 5/10
Installing : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64 6/10
Installing : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64 7/10
Installing : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64 8/10
Running scriptlet: iptables-nft-1.8.8-3.amzn2023.0.2.x86_64 8/10
Installing : libcgroup-3.0-1.amzn2023.0.1.x86_64 9/10
Running scriptlet: docker-25.0.6-1.amzn2023.0.2.x86_64 10/10
Installing : docker-25.0.6-1.amzn2023.0.2.x86_64 10/10
Running scriptlet: docker-25.0.6-1.amzn2023.0.2.x86_64 10/10
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.

Verifying : containerd-1.7.23-1.amzn2023.0.1.x86_64 1/10
Verifying : docker-25.0.6-1.amzn2023.0.2.x86_64 2/10
Verifying : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64 3/10
Verifying : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64 4/10
Verifying : libcgroup-3.0-1.amzn2023.0.1.x86_64 5/10
Verifying : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64 6/10
Verifying : libnfnetwork-1.0.1-19.amzn2023.0.2.x86_64 7/10
Verifying : libnftnl-1.2.2-2.amzn2023.0.2.x86_64 8/10
Verifying : pigz-2.5-1.amzn2023.0.3.x86_64 9/10
Verifying : runc-1.1.14-1.amzn2023.0.1.x86_64 10/10

Installed:
containerd-1.7.23-1.amzn2023.0.1.x86_64 docker-25.0.6-1.amzn2023.0.2.x86_64
iptables-libs-1.8.8-3.amzn2023.0.2.x86_64 iptables-nft-1.8.8-3.amzn2023.0.2.x86_64
libcgroup-3.0-1.amzn2023.0.1.x86_64 libnfnetwork-1.0.1-19.amzn2023.0.2.x86_64
libnftnl-1.2.2-2.amzn2023.0.2.x86_64 libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64
pigz-2.5-1.amzn2023.0.3.x86_64 runc-1.1.14-1.amzn2023.0.1.x86_64

Complete!
[root@ip-172-31-42-139 ~]# cat /etc/os-release
NAME="Amazon Linux"
VERSION="2023"
ID="amzn"
ID_LIKE="fedora"
VERSION_ID="2023"
PLATFORM_ID="platform:el2023"
PRETTY_NAME="Amazon Linux 2023.6.20241111"
ANSI_COLOR="0;33"
CPE_NAME="cpe:2.3:o:amazon:amazon_linux:2023"
HOME_URL="https://aws.amazon.com/linux/amazon-linux-2023/"
DOCUMENTATION_URL="https://docs.aws.amazon.com/linux/"
SUPPORT_URL="https://aws.amazon.com/premiumsupport/"
BUG_REPORT_URL="https://github.com/amazonlinux/amazon-linux-2023"
VENDOR_NAME="AWS"
VENDOR_URL="https://aws.amazon.com/"
SUPPORT_END="2028-03-15"
[root@ip-172-31-42-139 ~]# service docker start
Redirecting to /bin/systemctl start docker.service
[root@ip-172-31-42-139 ~]# usermod -a -G docker ec2-user
[root@ip-172-31-42-139 ~]# aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 9809217425

```

- Add ssh plugin at jenkins server dashboard after that write the pipeline using ssh where you can transfer the deployment and service.yaml files.
- Go to create pipeline syntax and give ssh values and use that for pipeline.
- I added this stage at my pipeline for transferring the files.

```
stage("Transferring deployment.yaml and service.yaml to EKS_client Machine"){
    steps{
```

```
        sshagent(['eksclientssh']) {
            sh 'ssh -o StrictHostKeyChecking=no ec2-user@172.31.84.17'
            sh 'scp -o StrictHostKeyChecking=no
/var/lib/jenkins/workspace/finaleksdeployment/deployment.yaml
ec2-user@172.31.42.139:/tmp'
            sh 'scp -o StrictHostKeyChecking=no
```

```
/var/lib/jenkins/workspace/finaleksdeployment/service.yaml
ec2-user@172.31.42.139:/tmp'
    }
}
}
```

- We successfully transferred both files.
- Now time to create a secret key at client EKS for verification to pull the image from ECR.

```
[root@ip-172-31-42-139 tmp]# cat deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp-deployment3
  labels:
    app: myapp
spec:
  replicas: 1
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      imagePullSecrets:
      - name: ecr-key
      containers:
      - name: myapp
        image: 980921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration:5
        imagePullPolicy: Always
        ports:
        - containerPort: 8080
[ec2-user@ip-172-31-42-139 tmp]$
```

- To create the secret key use the command
- Create a namespace and after creating that execute the command for creating the secret key.

```
kubectl create secret docker-registry ecr-key
--docker-server=980921742537.dkr.ecr.us-east-1.amazonaws.com
--docker-username=AWS --docker-password=$(aws ecr get-login-password)
--namespace=securityapp
```

Note: The bold text is the part of the URI of the image at ECR till amazonaws.com

- Try to run these commands just for verification whether everything is working fine.

```
[root@ip-172-31-42-139 tmp]# kubectl get nodes
NAME           STATUS   ROLES      AGE   VERSION
ip-172-31-70-1.ec2.internal   Ready    <none>    34m   v1.31.2-eks-94953ac
ip-172-31-83-251.ec2.internal   Ready    <none>    34m   v1.31.2-eks-94953ac
[root@ip-172-31-42-139 tmp]# kubectl get namespace
NAME     STATUS   AGE
default  Active   9h
kube-node-lease  Active   9h
kube-public   Active   9h
kube-system   Active   9h
securityapp  Active   6m5s
[root@ip-172-31-42-139 tmp]# kubectl get deploy
No resources found in default namespace.
[root@ip-172-31-42-139 tmp]# kubectl create -f deployment.yaml -n securityapp
deployment.apps/myapp-deployment3 created
[root@ip-172-31-42-139 tmp]# kubectl get deploy
No resources found in default namespace.
[root@ip-172-31-42-139 tmp]# kubectl get deploy -n securityapp
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
myapp-deployment3  1/1       1          1          16s
[root@ip-172-31-42-139 tmp]# kubectl delete deploy -n securityapp
error: resource(s) were provided, but no name was specified
[root@ip-172-31-42-139 tmp]# kubectl delete deploy myapp-deployment3 -n securityapp
deployment.apps "myapp-deployment3" deleted
[root@ip-172-31-42-139 tmp]#
```

- We are going to add the stage in our pipeline to deploy the deployment.yaml and service.yaml automatically.
- Navigate to jenkins pipeline and the stages.

```
[root@ip-172-31-42-139 tmp]# whereis kubectl
kubectl: /root/bin/kubectl
[root@ip-172-31-42-139 tmp]#
```

```
stage("Executing the application"){
    steps{
        sshagent(['eksclientssh']) {
            sh 'ssh -o StrictHostKeyChecking=no ec2-user@172.31.84.17 cd /tmp'
            sh 'ssh -o StrictHostKeyChecking=no ec2-user@172.31.84.17 /root/bin/kubectl delete all --all -n securityapp'
        }
    }
}
```

- Click on build now and run it for checking whether it is working fine. Then we will add two more lines.
- If kubectl is on the root directory you will get error as the permission is denied, so make sure you transfer the kubectl to ec2-user through the following code step.

```

sudo su - ec2-user
mkdir -p ~/bin
exit
sudo mv /root/bin/kubectl /home/ec2-user/bin/
sudo chown ec2-user:ec2-user /home/ec2-user/bin/kubectl
sudo chmod +x /home/ec2-user/bin/kubectl
echo 'export PATH=$PATH:/home/ec2-user/bin' >> ~/.bashrc
source ~/.bashrc
kubectl version --client

```

Once you run these commands you have successfully transferred the kubectl.

Add this pipeline code to your overall pipeline:

```

pipeline {
    agent any
    tools {
        maven 'maven-3.9.9'
    }
    environment {
        registry =
"980921742537.dkr.ecr.us-east-1.amazonaws.com/jenkinsintegration"
        registryCredential = 'awsecr_jenkins_user'
    }
    stages {
        stage ("Checking out the git project") {
            steps {
                git branch: 'main', url:
'https://github.com/kp18-cpu/techstart-jenkins-deploy-to-k8-repo.git'
            }
        }

        stage ("Checking bugs on sonarcloud") {
            steps {
                sh 'mvn clean verify sonar:sonar
-Dsonar.projectKey=devsecopskp -Dsonar.organization=devsecopskp
-Dsonar.host.url=https://sonarcloud.io
-Dsonar.token=d0d2af0eee7f1fdb21e937360084b8ece5a62cb4'
            }
        }
        // stage ("Build the package") {
        //     steps {
        //         sh 'mvn clean package'
        //     }
    }
}

```

```

// }

stage ("Run SCA scan and analysis using snyk"){
    steps {
        withCredentials([string(credentialsId: 'SNYK_TOKEN',
variable: 'SNYK_TOKEN')]) {
            sh 'mvn snyk:test -fn'
        }
    }
}

stage('Building our image') {
    steps{
        script {
            dockerImage = docker.build registry + ":"$BUILD_NUMBER"
        }
    }
}

stage('Deploy our image') {
    steps{
        script {
            docker.withRegistry("http://" + registry,
"ecr:us-east-1:" + registryCredential ) {
                dockerImage.push()
            }
        }
    }
}

stage("Transferring deployment.yaml and service.yaml to EKS_client Machine"){
    steps{
        sshagent(['eksclientssh']) {
            sh 'ssh -o StrictHostKeyChecking=no
ec2-user@172.31.84.17'
            sh 'scp -o StrictHostKeyChecking=no
/var/lib/jenkins/workspace/finaleksdeployment/deployment.yaml
ec2-user@172.31.37.81:/tmp'
            sh 'scp -o StrictHostKeyChecking=no
/var/lib/jenkins/workspace/finaleksdeployment/service.yaml
ec2-user@172.31.37.81:/tmp'
        }
    }
}

stage("Executing the application"){

```

```

steps{
    sshagent(['eksclientssh']) {
        sh 'ssh -o StrictHostKeyChecking=no ec2-user@172.31.37.81
cd /tmp'
            sh 'ssh -o StrictHostKeyChecking=no ec2-user@172.31.37.81
/home/ec2-user/bin/kubectl delete all --all -n securityapp'
            sh 'ssh -o StrictHostKeyChecking=no ec2-user@172.31.37.81
/home/ec2-user/bin/kubectl create -f /tmp/deployment.yaml -n securityapp'
            sh 'ssh -o StrictHostKeyChecking=no ec2-user@172.31.37.81
/home/ec2-user/bin/kubectl create -f /tmp/service.yaml -n securityapp'
        }
    }
}
}

```

172.21.37.81 is the private IP eks_client machine.

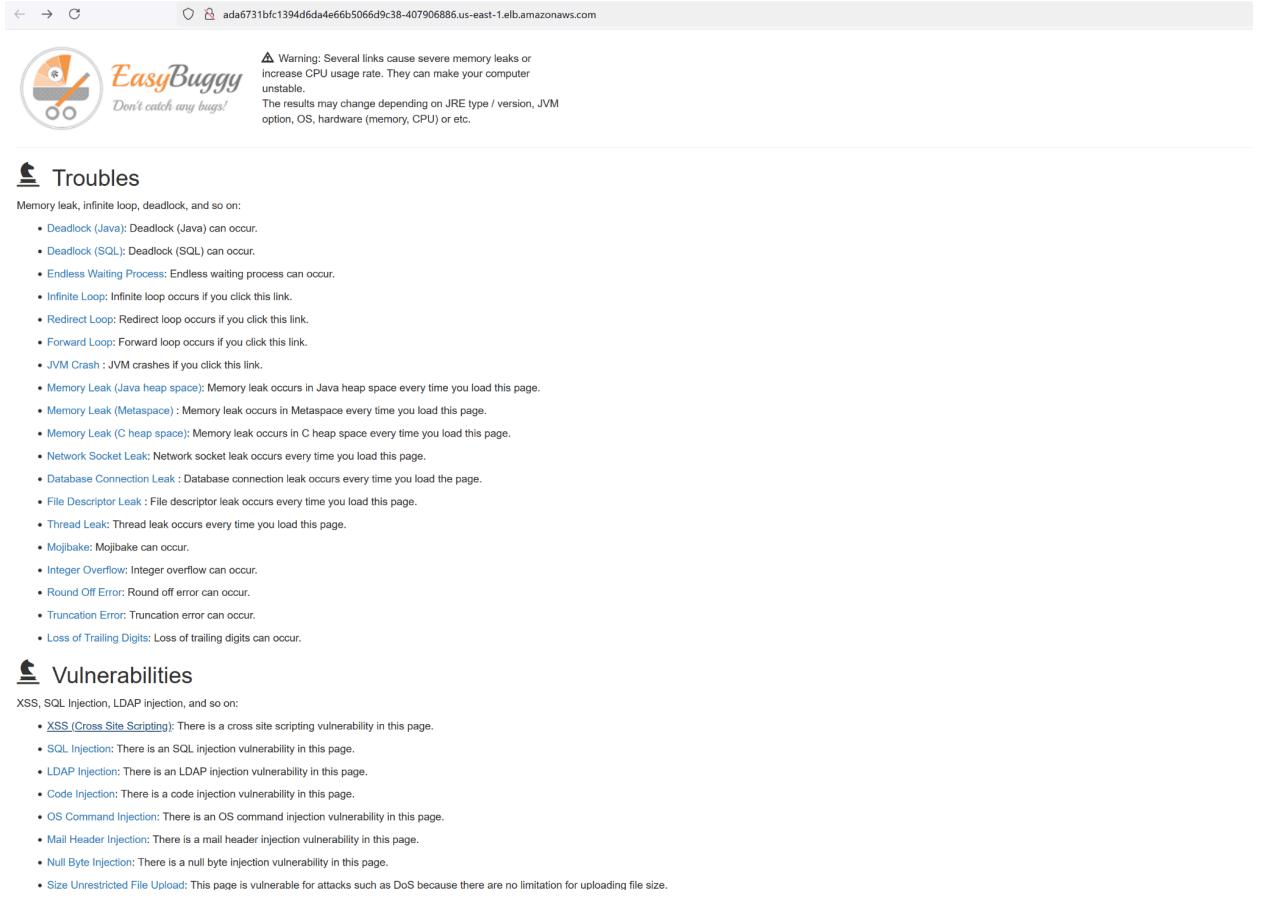
Where we try to deploy the deployment.yaml and service.yaml files to run on the cluster.

- We can verify that our hosted application is running on our aws cluster.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
myapp	LoadBalancer	10.100.141.1	ada6731bfcc1394d6da4e66b5066d9c38-407906886.us-east-1.elb.amazonaws.com	80:30167/TCP	57s

Copy the external ip and paste it on the browser to check whether it is working properly.

- We successfully created and hosted our application on the AWS Kubernetes service.



The screenshot shows a web browser window with the URL ada6731bfc1394d6da4e66b5066d9c38-407906886.us-east-1.elb.amazonaws.com. The page has a header with the EasyBuggy logo and the slogan "Don't catch any bugs!". A warning message at the top states: "⚠ Warning: Several links cause severe memory leaks or increase CPU usage rate. They can make your computer unstable. The results may change depending on JRE type / version, JVM option, OS, hardware (memory, CPU) or etc." Below the warning, there are two sections: "Troubles" and "Vulnerabilities".

Troubles

Memory leak, infinite loop, deadlock, and so on:

- [Deadlock \(Java\)](#): Deadlock (Java) can occur.
- [Deadlock \(SQL\)](#): Deadlock (SQL) can occur.
- [Endless Waiting Process](#): Endless waiting process can occur.
- [Infinite Loop](#): Infinite loop occurs if you click this link.
- [Redirect Loop](#): Redirect loop occurs if you click this link.
- [Forward Loop](#): Forward loop occurs if you click this link.
- [JVM Crash](#): JVM crashes if you click this link.
- [Memory Leak \(Java heap space\)](#): Memory leak occurs in Java heap space every time you load this page.
- [Memory Leak \(Metaspace\)](#): Memory leak occurs in Metaspace every time you load this page.
- [Memory Leak \(C heap space\)](#): Memory leak occurs in C heap space every time you load this page.
- [Network Socket Leak](#): Network socket leak occurs every time you load this page.
- [Database Connection Leak](#): Database connection leak occurs every time you load the page.
- [File Descriptor Leak](#): File descriptor leak occurs every time you load this page.
- [Thread Leak](#): Thread leak occurs every time you load this page.
- [Mojibake](#): Mojibake can occur.
- [Integer Overflow](#): Integer overflow can occur.
- [Round Off Error](#): Round off error can occur.
- [Truncation Error](#): Truncation error can occur.
- [Loss of Trailing Digits](#): Loss of trailing digits can occur.

Vulnerabilities

XSS, SQL Injection, LDAP injection, and so on:

- [XSS \(Cross Site Scripting\)](#): There is a cross site scripting vulnerability in this page.
- [SQL Injection](#): There is an SQL injection vulnerability in this page.
- [LDAP Injection](#): There is an LDAP injection vulnerability in this page.
- [Code Injection](#): There is a code injection vulnerability in this page.
- [OS Command Injection](#): There is an OS command injection vulnerability in this page.
- [Mail Header Injection](#): There is a mail header injection vulnerability in this page.
- [Null Byte Injection](#): There is a null byte injection vulnerability in this page.
- [Size Unrestricted File Upload](#): This page is vulnerable for attacks such as DoS because there are no limitation for uploading file size.

I am excited it is time to integrate the Zap tool which is a dynamic testing tool used for testing the deployed application. We are going to analyze the above-listed application and look for vulnerabilities.

What is Dynamic Testing?

Dynamic Testing is a software testing methodology that involves executing a program or application to identify defects. It focuses on the behavior of the software during execution, ensuring that it works as intended in a live environment. Unlike static testing, which reviews code without running it, dynamic testing involves actual execution.

What is Zap Tool ?

OWASP ZAP (Zed Attack Proxy) is an open-source web application security testing tool developed by the Open Web Application Security Project (OWASP). It is widely used for dynamic application security testing (DAST) to identify vulnerabilities in web applications.

Key Features of ZAP

1. Intercepts and inspects HTTP(S) traffic between the client and server.
2. Performs automated scans for security vulnerabilities.
3. Includes tools for manual security testing (e.g., fuzzing, forced browsing).
4. Integrates with CI/CD pipelines for automated security testing.
5. Provides an easy-to-use GUI, API, and command-line interface.

Now let's start integrating the zap with our application for the dynamic testing.

- Navigate to eks_client VM and install zap.
- GO to /tmp directory and create a zap directory through mkdir zap.
- Before installing make sure you have java installed on your eks_client.
- To install java:

```
wget https://corretto.aws/downloads/latest/amazon-corretto-11-x64-linux-jdk.rpm  
sudo yum localinstall -y amazon-corretto-11-x64-linux-jdk.rpm
```

```
java -version
```

Once installed:

```
readlink -f $(which java)  
export JAVA_HOME=/usr/lib/jvm/java-11-amazon-corretto  
export PATH=$JAVA_HOME/bin:$PATH  
echo 'export JAVA_HOME=/usr/lib/jvm/java-11-amazon-corretto' >> ~/.bashrc  
echo 'export PATH=$JAVA_HOME/bin:$PATH' >> ~/.bashrc  
source ~/.bashrc
```

```
[ec2-user@ip-172-31-37-81 zap]$ java --version  
openjdk 11.0.25 2024-10-15 LTS  
OpenJDK Runtime Environment Corretto-11.0.25.9.1 (build 11.0.25+9-LTS)  
OpenJDK 64-Bit Server VM Corretto-11.0.25.9.1 (build 11.0.25+9-LTS, mixed mode)  
[ec2-user@ip-172-31-37-81 zap]$ readlink -f $(which java)  
/usr/lib/jvm/java-11-amazon-corretto/bin/java  
[ec2-user@ip-172-31-37-81 zap]$ export JAVA_HOME=/usr/lib/jvm/java-11-amazon-corretto  
[ec2-user@ip-172-31-37-81 zap]$ export PATH=$JAVA_HOME/bin:$PATH  
[ec2-user@ip-172-31-37-81 zap]$ echo 'export JAVA_HOME=/usr/lib/jvm/java-11-amazon-corretto' >> ~/.bashrc  
[ec2-user@ip-172-31-37-81 zap]$ echo 'export PATH=$JAVA_HOME/bin:$PATH' >> ~/.bashrc  
[ec2-user@ip-172-31-37-81 zap]$ 'export PATH=$JAVA_HOME/bin:$PATH' >> ~/.bashrc  
[ec2-user@ip-172-31-37-81 zap]$  
-bash: export PATH=$JAVA_HOME/bin:$PATH: No such file or directory  
-bash: [ec2-user@ip-172-31-37-81: command not found  
[ec2-user@ip-172-31-37-81 zap]$ source ~/.bashrc  
[ec2-user@ip-172-31-37-81 zap]$ ls
```

- After that install zap using wget command:

```
wget  
https://github.com/zaproxy/zaproxy/releases/download/v2.15.0/ZAP\_2\_15\_0\_unix.sh  
chmod +x ZAP.sh (change name accordingly)  
./zap.sh -y (change name accordingly)
```

```
[ec2-user@ip-172-31-37-81 zap]$ sudo ./ZAP_2_15_0_unix.sh
Starting Installer ...
This will install Zed Attack Proxy on your computer.
OK [o, Enter], Cancel [c]
o
Click Next to continue, or Cancel to exit Setup.
Please read the following License Agreement. You must accept the terms of
this agreement before continuing with the installation.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

[Enter]

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.
```

Keep everything default and complete the installation.

```
[ec2-user@ip-172-31-37-81 zap]$ zap.sh
Found Java version 11.0.25
Available memory: 949 MB
Using JVM args: Xmx237m
1654 [main] INFO org.parosproxy.paros.Constant - Copying default configuration to /home/ec2-user/.ZAP/config.xml
1879 [main] INFO org.parosproxy.paros.Constant - Creating directory /home/ec2-user/.ZAP/session
1880 [main] INFO org.parosproxy.paros.Constant - Creating directory /home/ec2-user/.ZAP/dirbuster
1880 [main] INFO org.parosproxy.paros.Constant - Creating directory /home/ec2-user/.ZAP/fuzzers
1880 [main] INFO org.parosproxy.paros.Constant - Creating directory /home/ec2-user/.ZAP/plugins
2082 [main] INFO org.zaproxy.zap.GuiBootstrap - ZAP 2.15.0 started 21/11/2024, 06:26:21 with home: /home/ec2-user/.ZAP/ cores: 1
maxMemory: 230 MB
2084 [main] FATAL org.zaproxy.zap.GuiBootstrap - ZAP GUI is not supported on a headless environment.
Run ZAP inline or in daemon mode, use -help command line argument for more details.
ZAP GUI is not supported on a headless environment.
Run ZAP inline or in daemon mode, use -help command line argument for more details.
[ec2-user@ip-172-31-37-81 zap]$
```

Zap is successfully installed.

- We can see our url running on the container like this.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
myapp	LoadBalancer	10.100.141.1	ada6731bfc1394d6da4e66b5066d9c38-407906886.us-east-1.elb.amazonaws.com	80:30167/TCP	50m

- We need to retrieve that url in which our app is running deployed on the cluster. Time to think how to retrieve this url

```
[ec2-user@ip-172-31-37-81 zap]$ kubectl get svc --namespace securityapp
NAME      TYPE        CLUSTER-IP   EXTERNAL-IP
myapp    LoadBalancer  10.100.141.1  ada6731bfc1394d6da4e66b5066d9c38-407906886.us-east-1.elb.amazonaws.com
[ec2-user@ip-172-31-37-81 zap]$ /home/ec2-user/bin/kubectl get services/myapp --namespace=securityapp -o json
{
  "apiVersion": "v1",
  "kind": "Service",
  "metadata": {
    "creationTimestamp": "2024-11-21T05:38:51Z",
    "finalizers": [
      "service.kubernetes.io/load-balancer-cleanup"
    ],
    "labels": {
      "app": "myapp",
      "k8s-app": "myapp"
    },
    "name": "myapp",
    "namespace": "securityapp",
    "resourceVersion": "106644",
    "uid": "da6731bf-c139-4d6d-a4e6-6b5066d9c38e"
  },
  "spec": {
    "allocateLoadBalancerNodePorts": true,
    "clusterIP": "10.100.141.1",
    "clusterIPs": [
      "10.100.141.1"
    ],
    "externalTrafficPolicy": "Cluster",
    "internalTrafficPolicy": "Cluster",
    "ipFamilies": [
      "IPv4"
    ],
    "ipFamilyPolicy": "SingleStack",
    "ports": [
      {
        "name": "http",
        "nodePort": 30167,
        "port": 80,
        "protocol": "TCP",
        "targetPort": 8080
      }
    ],
    "selector": {
      "app": "myapp"
    },
    "sessionAffinity": "None",
    "type": "LoadBalancer"
  },
  "status": {
    "loadBalancer": {
      "ingress": [
        {
          "hostname": "ada6731bfc1394d6da4e66b5066d9c38-407906886.us-east-1.elb.amazonaws.com"
        }
      ]
    }
  }
}
```

- Using this command we can retrieve the url dynamically, Make sure you change the values of –namespace, in our case its “securityapp”.

```
/home/ec2-user/bin/kubectl get services/myapp --namespace=securityapp -o json | /usr/bin/jq -r ".status.loadBalancer.ingress[] | .hostname"
```

```
[ec2-user@ip-172-31-37-81 zap]$ /home/ec2-user/bin/kubectl get services/myapp --namespace=securityapp -o json | /usr/bin/jq -r ".status.loadBalancer.ingress[] | .hostname"
ada6731bfc1394d6da4e66b5066d9c38-407906886.us-east-1.elb.amazonaws.com
```

If you have any error because of jq (yum install jq).

- The url is dynamic so we are try to add dynamic ways to bind it to the url and get the analysis through zap.

The script of complete zap is here:

```
/usr/local/bin/zap.sh -cmd -quickurl http://$(/home/ec2-user/bin/kubectl get services/myapp --namespace=securityapp -o json | /usr/bin/jq -r ".status.loadBalancer.ingress[] | .hostname") -quickprogress -quickout /tmp/zap_report.html
```

Note: Very Important code above for integrating zap.

- We can integrate this script into the pipeline for automation, but I just run it on the terminal and we will get the report in the form of an HTML file.
- The testing will run for a while so have patience and we will get the file successfully.

```
[ec2-user@ip-172-31-37-81 zap]$ /usr/local/bin/zap.sh -cmd -quickurl http://$(/home/ec2-user/bin/kubectl get services/myapp --name=securityapp -o json | /usr/bin/jq -r ".status.loadBalancer.ingress[] | .hostname") -quickprogress -quickout /tmp/zap_report.html
Found Java version 11.0.25
Available memory: 949 MB
Using JVM args: -Xmx237m
Accessing URL
Using traditional spider
[=====] 36%
```

It's going to be 50 min, and the progress is still at 73 percent.

```
[ec2-user@ip-172-31-37-81 zap]$ /usr/local/bin/zap.sh -cmd -quickurl http://$(/home/ec2-user/bin/kubectl get services/myapp --name=securityapp -o json | /usr/bin/jq -r ".status.loadBalancer.ingress[] | .hostname") -quickprogress -quickout /tmp/zap_report.html
Found Java version 11.0.25
Available memory: 949 MB
Using JVM args: -Xmx237m
Accessing URL
Using traditional spider
[=====■====] 73%
```

- Once the .html file is saved in the /tmp folder look for it.
- Time to transfer the file.
- cd to the /tmp folder where our file will be stored.
- Start the Python server, before that make sure you have Python installed.
- python3 -m http.server 8080
- Go to the web browser and access the server using the IP address and port number and you will see the file. You can download it into your local system.
- Click on zap_report.html and view the report of vulnerabilities and in-depth report.

The screenshot shows the ZAP Report interface. At the top, there is a navigation bar with icons for Home, Help, and Logout. Below the navigation bar, the text "ZAP Version: 2.14.0" is displayed. The main content area is titled "Summary of Alerts". It features a table with four columns: Risk Level (High, Medium, Low, Informational), Number of Alerts (0, 3, 3, 1), and a status bar indicating the total count. Below this, a section titled "Alerts" displays a table with columns for Name, Risk Level, and Number of Instances. The table lists various security issues found during the scan.

Risk Level	Number of Alerts
High	0
Medium	3
Low	3
Informational	1
False Positives	0

Name	Risk Level	Number of Instances
Absence of Anti-CSRF Tokens	Medium	19
Content Security Policy (CSP) Header Not Set	Medium	37
Session ID in URL Rewrite	Medium	1
Application Error Disclosure	Low	3
Cooke without SameSite Attribute	Low	5
Server Leaks Version Information via "Server" HTTP Response Header Field	Low	46
Session Management Response Identified	Informational	10

This is just an overview of the report shared here, we can explore more of it and use it for further testing and reducing the bugs and code smell.

DEVSECOPS has so much and lot to learn and explore, especially playing around with different options on the AWS.

I want to sincerely thank Mr. Verma for his great guidance in learning and being on the path of DEVSECOPS.

Glad you guys learnt through this document.