**Devsecops Project**

**Step:1 Create Jenkins server :**

1. Log in to the **AWS Management Console**.
2. Navigate to **EC2 > Instances > Launch Instances**.
3. Configure the instance:
   * Choose **Amazon Linux 2 AMI**.
   * Select **t3.xlarge** as the instance type.
   * Add storage: **60 GiB gp2**.
   * Configure security group:
     + Allow **SSH (port 22)** and **HTTP (port 8080)**.
   * Assign a key pair.
4. Launch the instance and wait for it to initialize.

**Commands to Run After Launch**

# Update the instance

sudo yum update -y

# Install basic utilities

sudo yum install -y wget git

**step 2; Jenkins Installation**

Run these commands on the **Jenkins server instance**:

1. Add the Jenkins repository:

sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat/jenkins.repo

sudo rpm --import https://pkg.jenkins.io/redhat/jenkins.io-2023.key

sudo yum upgrade -y

1. Install Java 17:

amazon-linux-extras enable corretto17

sudo yum install -y java-17-amazon-corretto

java -version

1. Install Jenkins:

sudo yum install jenkins -y

1. Enable and start Jenkins:

sudo systemctl enable jenkins

sudo systemctl start jenkins

sudo systemctl status jenkins

**4. Access Jenkins**

1. Open your browser and access Jenkins at:
2. http://<Jenkins-Instance-IP>:8080
3. Unlock Jenkins:
   * Find the initial admin password

cat in /var/lib/jenkins/secrets/initialAdminPassword.

* + Enter the password and proceed.

1. Install the suggested plugins.
2. Create an admin user:
   * Username: Nuthan
   * Password: Panny@123

**Step 3: Configure Jenkins for CI/CD with Additional Tools**

**1. Install Essential Plugins**

1. Go to **Jenkins Dashboard > Manage Jenkins > Manage Plugins**.
2. Navigate to the **Available** tab and search for these plugins:
   * **Git Plugin**: For integrating Git repositories. (pre-installed)
   * **Pipeline Plugin**: For creating declarative or scripted pipelines.

[Pipeline: Stage View](https://plugins.jenkins.io/pipeline-stage-view)

[Pipeline: Declarative Agent AP](https://plugins.jenkins.io/pipeline-model-declarative-agent)

* + **Terraform Plugin**: For running Terraform commands in Jenkins.
  + **HashiCorp Vault Plugin**: To pull secrets from Vault (optional, based on your goals).
  + **SonarQube Scanner Plugin**: For static code analysis integration.
  + **Docker Plugin**: To run Docker-related commands within Jenkins.
  + **Snyk Security Plugin**: For code and dependency scanning.
  + **Ansible Plugin**: To automate configuration management.

**Install Terraform**

Install Terraform by running these commands:

sudo yum install -y yum-utils

sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo

sudo yum -y install terraform

**Install TFscan**

Install TFscan with the following command:

curl -s https://raw.githubusercontent.com/aquasecurity/tfsec/master/scripts/install\_linux.sh | bash

tfsec –version

**Install Trivy**

1. Install Trivy:

curl -sfL https://raw.githubusercontent.com/aquasecurity/trivy/main/contrib/install.sh | sh

1. Move Trivy to a global directory:

sudo mv /root/bin/trivy /usr/local/bin/trivy

trivy --version

**Why?**: Placing Trivy in /usr/local/bin ensures it is accessible to all users, including Jenkins.

**Updated Steps to Install Snyk CLI**

**Use npm (Recommended)**

If Node.js is installed on your Jenkins server, you can use npm to install Snyk CLI.

1. Install Node.js and npm:

sudo yum install -y nodejs

1. Install Snyk CLI using npm:

npm install -g snyk

1. Verify the installation:

snyk --version

**Configure Global Tools in Jenkins**

1. **Configure Git**:
   * Go to **Manage Jenkins > Global Tool Configuration**.
   * Under **Git**, click **Add Git**.
   * Set the path to /usr/bin/git.
2. **Configure Terraform**:
   * Add Terraform under **Terraform installations**:
     + Go to **Global Tool Configuration**.
     + Add a new Terraform installation and ensure the binary is installed at /usr/bin/
3. **Configure Ansible**:
   * Add Ansible under **Ansible installations**:
     + Go to **Global Tool Configuration**.
     + Add a new Ansible installation and set the path to /usr/bin/

**Create Your First Job to Verify Jenkins**

1. **Create a Freestyle Project**:
   * Go to **Jenkins Dashboard > New Item**.
   * Enter a name (e.g., Verify-Jenkins) and select **Freestyle Project**.
   * Under **Build Steps**, add a shell command:

echo "Jenkins is configured with additional tools!"

tfsec --version

trivy --version

snyk --version

ansible --version

* + Save the job and click **Build Now**.

1. **Check Console Output**:
   * Ensure the console output confirms the installed versions of tfsec, trivy, snyk, and ansible.

**Installing Maven on the Jenkins Server**

**1. Install Maven**

Run the following commands on your Jenkins server to install Maven:

sudo yum install -y maven

**2. Verify the Maven Installation**

Check the Maven version to confirm the installation:

mvn -version

You should see output like this:

Apache Maven 3.x.x

Maven home: /usr/share/maven

Java version: 11, vendor: Amazon.com Inc.

**Configuring Maven in Jenkins**

**1. Add Maven in Jenkins**

1. Go to **Jenkins Dashboard > Manage Jenkins > Global Tool Configuration**.
2. Scroll to the **Maven** section and click **Add Maven**.
3. Configure Maven:
   * **Name**: Maven (or any name you prefer).
   * **Install Automatically**: Uncheck this if Maven is installed manually (as in this case).
   * **Maven Home Directory**: Provide the path where Maven is installed (e.g., /usr/share/maven).

**Create a Jenkins Freestyle Job**

1. Go to **Jenkins Dashboard > New Item**.
2. Enter a job name (e.g., Verify-Jenkins-Maven) and select **Freestyle Project**.
3. Under **Source Code Management**, select **Git**:
   * Enter the **Repository URL**.
   * Provide **Credentials** if required.
   * Specify the branch (e.g., main).

**Add Maven Build Step**

1. Under **Build Steps**, select **Invoke top-level Maven targets**.
2. Configure the Maven build:
   * **Root POM**: pom.xml**(optional)**
   * **Goals and options**: clean install

**Build and Verify**

1. Save the job and click **Build Now**.
2. Check the console output to ensure:
   * The repository is cloned successfully.
   * The pom.xml file is detected.
   * Maven builds the project without errors.

**Integrating SonarQube**

**1. Install SonarQube**

1. Launch a new **EC2 instance** or use an existing one for SonarQube.
2. Instance specifications:
   * **Instance Type**: t3.medium (minimum 4 GiB RAM).
   * **Volume**: 20 GiB.
3. Install and configure SonarQube:
   * **For Amazon Linux**:

sudo yum update -y

install Java:

**amazon-linux-extras enable corretto17**

**sudo yum install -y java-17-amazon-corretto**

wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.9.0.65466.zip

unzip sonarqube-9.9.0.65466.zip

sudo mv sonarqube-9.9.0.65466 /opt/sonarqube

sudo adduser sonar

sudo chown -R sonar:sonar /opt/sonarqube

sudo su - sonar

/opt/sonarqube/bin/linux-x86-64/sonar.sh start

* + Access SonarQube in a browser using: http://<EC2-public-IP>:9000.

**Configuring SonarQube**

**Note: store projectkey, project display name, token in separate place. Will use those forward.**

**Create a New Project in SonarQube**

1. Log in to SonarQube.
2. Click **Create New Project** and provide the project name (e.g., Sample E-Commerce Project).

**Generate an Authentication Token**

1. Navigate to **My Account > Security**.
2. Under **Generate Tokens**, enter a token name (e.g., SampleProjectToken).
3. Select **Project Analysis** from the dropdown.
4. Click **Generate** and copy the token. (Save it securely; it will not be displayed again.)

**Note : Exit from sonar user before run any commands**

**Installing Sonar Scanner**

**Download and Install Sonar Scanner**

1. **Create a directory for Sonar Scanner:**

mkdir -p /downloads/sonarqube

cd /downloads/sonarqube

1. **Download the latest Sonar Scanner:**

wget https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-5.0.1.3006-linux.zip

unzip sonar-scanner-cli-5.0.1.3006-linux.zip

sudo mv sonar-scanner-5.0.1.3006-linux /opt/sonar-scanner

1. **Add Sonar Scanner to the PATH:**

vi ~/.bashrc

export PATH="/opt/sonar-scanner/bin:$PATH"

source ~/.bashrc

1. **Verify the installation:**

sonar-scanner --version

Ensure “**SonarQube Scanner for Jenkins**” plugin is installed.

1. **Configure SonarQube in Jenkins**:
   * Go to **Manage Jenkins > Configure System**.
   * Find the **SonarQube Servers** section.
   * Add a new SonarQube server:
     + **Name**: SonarQube.
     + **Server URL**: http://<SonarQube-IP>:9000.
     + **Authentication Token**: Add the token you generated earlier.

**4. Update Jenkins Job**

1. Open the Jenkins job for your project.
2. Add a **SonarQube Analysis Build Step**:
   * Go to **build-steps**
   * Select **"Execute SonarQube Scanner"**.
   * Configure:
     + **Task**: Add the following analysis properties:

sonar.projectKey=<your-project-key>

sonar.projectName=<your-project-name>

sonar.projectVersion=1.0

sonar.sources=src

sonar.java.binaries=target/classes

sonar.login=<token>

* + Replace <your-project-key> and <your-project-name> with the details of your project.

1. **Trigger the Jenkins Job**:
   * Run the job and verify that SonarQube analyzes the code and generates a report.

Here we are sharing hashicorp vault with Jenkins instance:

Open port for vault : 8200

**Next Step: HashiCorp Vault Integration**

Before moving further, we need to integrate **HashiCorp Vault** for securely managing AWS credentials and other secrets required by Terraform.

Install vault: :

**sudo yum install -y yum-utils shadow-utils**

**sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo**

**sudo yum -y install vault**

Start Vault:

* Run Vault in dev mode:

vault server -dev -dev-listen-address="0.0.0.0:8200"

**Open a Separate Terminal for Configure Vault for AWS Credential Management Setup**

1. **Right-click** on the tab of your terminal session.
2. From the context menu, select the option **'Duplicate Session'**.
3. This will open a new tab with a duplicate of your current terminal session, which you can use to continue the setup process.
4. After entering into the duplicate terminal, get sudo access and navigate to:

 Set Vault's environment variables:

export VAULT\_ADDR=http://0.0.0.0:8200

export VAULT\_TOKEN=<root-token>

 Enable the AWS secrets engine:

vault secrets enable -path=aws aws

 Configure AWS credentials in Vault:

vault write aws/config/root \

access\_key=<YOUR\_AWS\_ACCESS\_KEY> \

secret\_key=<YOUR\_AWS\_SECRET\_KEY> \

region=us-east-1

 Create a Vault role for AWS credentials:

vault write aws/roles/dev-role \

credential\_type=iam\_user \

policy\_document=-<<EOF

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": ["ec2:\*", "sts:GetCallerIdentity"],

"Resource": "\*"

}

]

}

EOF

**Testing HashiCorp Vault in Freestyle**

To test Vault in a freestyle Jenkins job:

**Steps:**

1. **Create a New Freestyle Job**:
   * Go to Jenkins Dashboard → New Item → Enter a Job Name (e.g., Test-Vault) → Select "Freestyle Project" → Click "OK."
2. **Add Build Step**:
   * Under **Build**, click on "Add Build Step" → Select "Execute Shell."
3. **Add the Following Shell Script**:

# Export Vault address and token

export VAULT\_ADDR=http://<VAULT\_SERVER\_IP>:8200

export VAULT\_TOKEN=<YOUR\_VAULT\_TOKEN>

echo "Testing Vault Connection..."

# Read AWS credentials from Vault

vault read -format=json aws/creds/dev-role > aws\_creds.json

jq -r '.data.access\_key' aws\_creds.json

jq -r '.data.secret\_key' aws\_creds.json

1. **Run the Job**:
   * Click **Save** and then **Build Now**.
   * Check the **Console Output** to verify:
     + Vault connection is successful.
     + AWS credentials are retrieved and displayed.

**Integrating Trivy for Container Image Scanning**

Install docker :

Yum install docker -y

**Steps to Integrate Trivy for Image Scanning**

1. **Create the Dockerfile**:

Vi Dockerfile

# Use an official OpenJDK runtime as a parent image

FROM openjdk:17-jdk-slim

# Set the working directory inside the container

WORKDIR /app

# Copy the built JAR file into the container

COPY target/project-0.0.1-SNAPSHOT.jar app.jar

# Expose the port your application listens on (default Spring Boot port)

EXPOSE 8081

# Run the JAR file

CMD ["java", "-jar", "app.jar"]

* + Save the above Dockerfile in the root directory of your project (/path-to-your-project).

1. **Build the Docker Image**:
   * Navigate to your project directory and run:

docker build -t sample-ecommerce-app .

1. **Run the Docker Container** (Optional for testing):
   * To test the container, run:

docker run -p 8081:8081 sample-ecommerce-app

* + Access the application in your browser at http://<your-server-ip>:8081.

1. **Scan the Image with Trivy**:
   * Use Trivy to scan for vulnerabilities in the image:

trivy image sample-ecommerce-app

1. **Analyze the Output**:
   * Review the vulnerabilities found in the image (if any) and fix them by updating dependencies or modifying the Dockerfile.
2. **Clean Up**:
   * Stop and remove the running container (if any):

docker stop <container-id>

docker rm <container-id>

**Push Docker Image to a Container Registry**

* Store your Docker image in a secure and accessible container registry, such as Docker Hub, Amazon ECR, or Azure ACR.
* Steps:
  1. Log in to Docker Hub

docker login -u <your-dockerhub-username> -p <your-dockerhub-password>

* 1. Tag your Docker image:

docker tag <Image-name > <your-registry>/sample-ecommerce-app:java

* 1. Push the image:

docker push <your-registry>/sample-ecommerce-app:java

**Install and Configure Nexus Repository:**

Create an EC2 instance for Nexus:

Give a name of instance(eg : nexus)

Instabce type: t3.xlarge

Open port : 8081

Installing java **:** amazon-linux-extras enable corretto17

sudo yum install -y java-17-amazon-corretto

wget https://download.sonatype.com/nexus/3/nexus-3.76.0-03-unix.tar.gz

tar -xvf nexus-3.76.0-03-unix.tar.gz

mv nexus-3.76.0-03 /opt/nexus

mv sonatype-work /opt/sonatype-work

useradd nexus

chown -R nexus:nexus /opt/nexus /opt/sonatype-work

vi /opt/nexus/bin/nexus.rc

run\_as\_user="nexus"

su – nexus

cd /opt/nexus/bin

./nexus start

Access nexus: your ip:8081

Click on sign in:

Admin: admin

Pass : cat /opt/sonatype-work/nexus3/admin.password

Change your password to continue.

Your choice :

**Enable the KV Secrets Engine**

First, ensure that the KV secrets engine is enabled. This engine will store your Nexus credentials securely.

bash

CopyEdit

vault secrets enable -path=nexus kv

**2. Store Nexus Credentials**

Use the vault kv put command to securely store the Nexus username and password.

vault kv put nexus/credentials username="your-nexus-username" password="your-nexus-password"

You can also include other information, such as the Nexus repository URL or token, if required.

vault kv put nexus/credentials \

username="your-nexus-username" \

password="your-nexus-password" \

repo\_url="https://nexus.example.com"

**3. Retrieve Nexus Credentials**

To fetch the credentials, use the vault kv get command.

* Retrieve all credentials:

vault kv get nexus/credentials

**Install Ansible:**

**For Amazon Linux:**

Amazon Linux typically does not require EPEL as Ansible is available directly from the Amazon Linux repository. Use the following commands:

1. **Update your system:**

sudo yum update -y

1. **Install Ansible directly:**

sudo amazon-linux-extras enable ansible2

sudo yum install ansible -y

1. **Verify Ansible installation:**

ansible –version

**Use the Generic Key-Value (KV) Secrets Engine**

* Store Docker credentials in the KV secrets engine and retrieve them when needed.
* Example:

vault kv put secret/docker username=nuthan0530 password=

Retrieve credentials:

vault kv get -field=username secret/docker

vault kv get -field=password secret/docker

**Steps to Store and Retrieve a Snyk Token in Vault**

**1. Enable the KV Secrets Engine (if not already enabled)**

vault secrets enable -path=snyk kv

* -path=snyk: Specifies a custom path for storing Snyk-related secrets. You can customize this as needed.

**2. Store the Snyk Token**

vault kv put snyk/token api\_token="your-snyk-token"

Replace your-snyk-token with the actual Snyk token.

**3. Retrieve the Snyk Token**

To fetch the token programmatically or manually:

vault kv get -field=api\_token snyk/token

The -field=api\_token flag extracts only the token value.

**Install Sendmail or Postfix**

You can choose to install either sendmail or postfix as the MTA.

1. **Install Sendmail**

sudo yum install sendmail -y

sudo systemctl start sendmail

sudo systemctl enable sendmail

**Integrate Vault with Jenkins:**

Since you have installed HashiCorp Vault Plugin, add vault credentials to Jenkins

Go to **Manage Jenkins > Credentials**.

Select global scope> kind : secret text

 Add a **Vault Address** and same with token

Name it accordingly VAULT-ADDR & VAULT\_TOKEN

Configure Vault in Jenkins:

* Navigate to **Manage Jenkins > Configure System**.
* Under the **Vault Plugin** section:
  + Set the Vault URL: <http://YOUR> IP:8200.
  + Add the credentials created earlier.