

# DevSecOps Pipeline with AWS, Kubernetes, Jenkins, and Security Integrations

## Project Overview

The project aims to implement a **DevSecOps pipeline** that integrates key tools such as **Jenkins**, **Terraform**, **AWS**, **Kubernetes**, **SonarCloud**, **Snyk**, and **OWASP ZAP** to automate the CI/CD process with a focus on **security**. The goal is to establish a seamless integration between infrastructure provisioning, code quality checks, security scanning, and deployment, all while ensuring that vulnerabilities are tracked and addressed efficiently through **Jira** automation.

## Prerequisites

- **AWS Account:** Active AWS account with IAM user permissions to manage resources.
- **Terraform:** Installed locally for provisioning AWS infrastructure.
- **Jenkins:** Installed with necessary plugins (AWS, Docker, Kubernetes, etc.).
- **GitHub:** Repositories for application code and Jenkinsfiles.
- **SonarCloud:** Account and API token for static code analysis.
- **Snyk:** Account and API token for vulnerability scanning.
- **Docker:** Installed for container image management and deployment.
- **Kubernetes:** kubectl installed for EKS cluster management.
- **Jira:** Account for issue tracking and API access for automation..

## Activities Outline

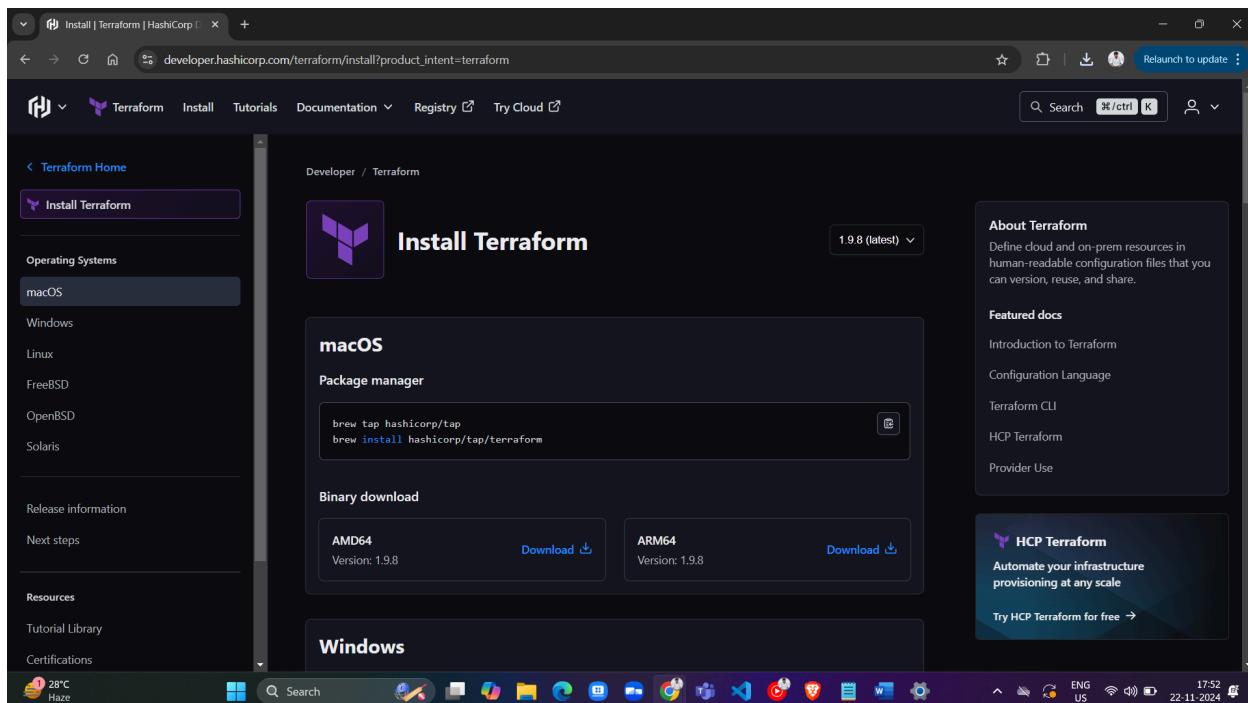
1. **Infrastructure Setup:** Provision AWS resources using Terraform.
2. **Jenkins Setup:** Configure Jenkins as the CI/CD orchestrator.
3. **SAST Integration:** Static Application Security Testing using SonarQube.
4. **SCA Integration:** Software Composition Analysis with Snyk.
5. **Dockerization & ECR:** Containerization and image management with AWS Elastic Container Registry (ECR).
6. **Kubernetes Deployment (EKS):** Deploy and manage applications on EKS.
7. **DAST Integration:** Dynamic Application Security Testing with OWASP ZAP.
8. **Jira Integration:** Automate ticket creation for vulnerabilities.

## Procedures

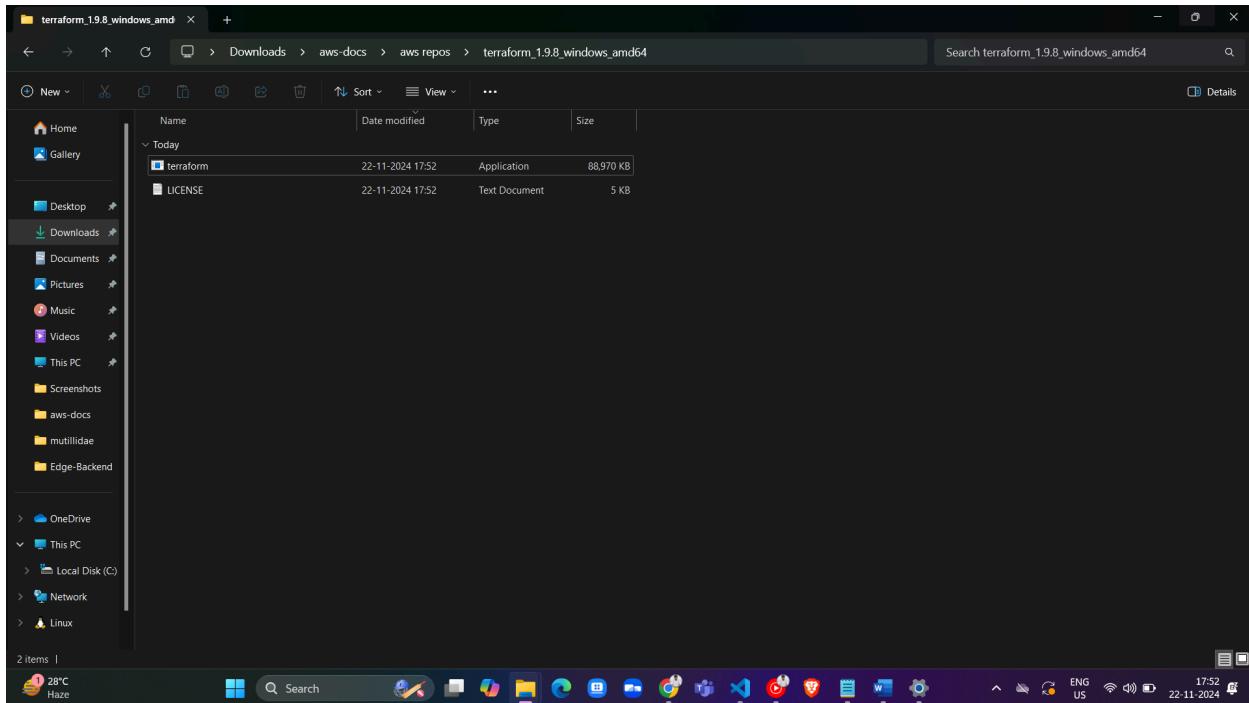
### Step 1. Initial Setup

#### Install Terraform

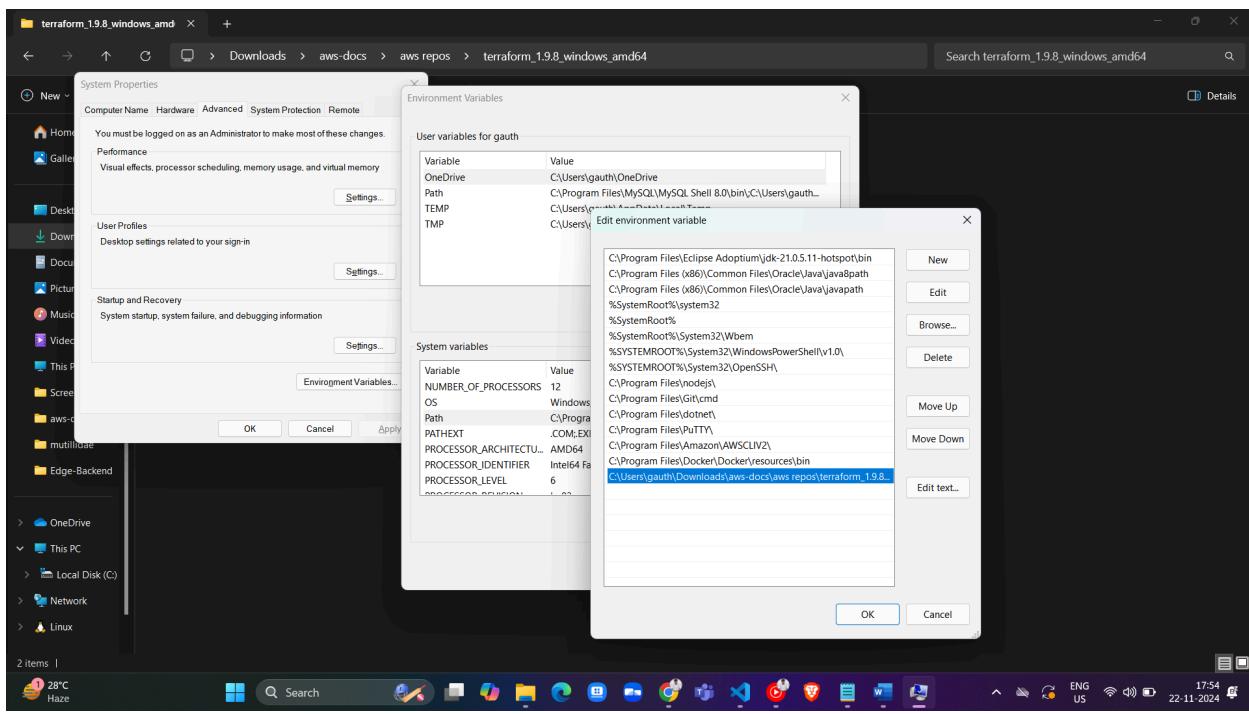
1. Download Terraform from the official Terraform website. ([https://developer.hashicorp.com/terraform/install?product\\_intent=terraform](https://developer.hashicorp.com/terraform/install?product_intent=terraform))



2. Install it on your local machine.

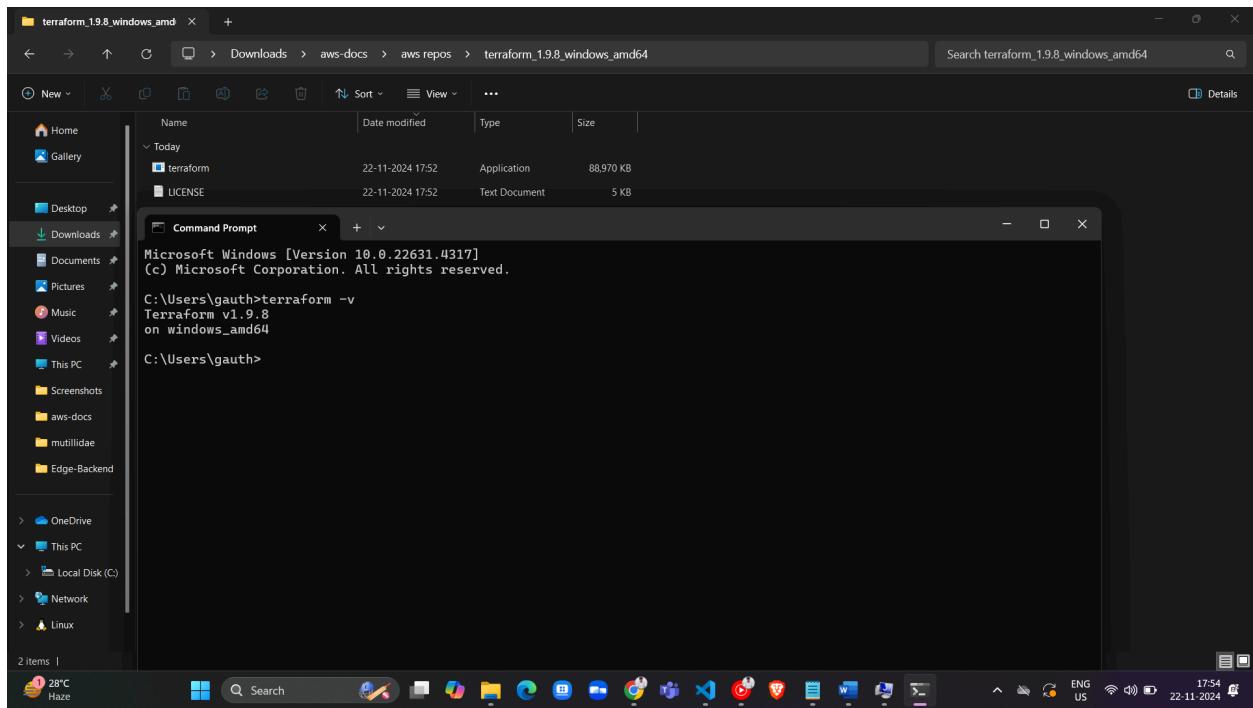


### 3. Add the Terraform executable to your system's environment path.



### 4. Verify the installation:

```
terraform -v
```



## Step 2. Configure AWS CLI

1. Create an IAM user with administrator privileges and CLI access.

us-east-1.console.aws.amazon.com/iam/home?region=us-east-2#users/create

AWS Search [Alt+S] Global Gautham

IAM > Users > Create user

Step 1 Specie user details Step 2 Set permissions Step 3 Review and create

### Specify user details

User details

User name  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

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences ENG US 17:57 22-11-2024

us-east-1.console.aws.amazon.com/iam/home?region=us-east-2#users/create

AWS Search [Alt+S] Global Gautham

IAM > Users > Create user

Step 2 Set permissions Step 3 Review and create

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

### User details

User name Gautham-cli	Console password type None	Require password reset No
--------------------------	-------------------------------	------------------------------

### Permissions summary

Name <a href="#">Edit</a>	Type	Used as
AdministratorAccess	AWS managed - job function	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

2. Click create access key for the created user

The screenshot shows the AWS IAM Access Keys page. On the left, there's a sidebar with navigation links like Dashboard, Access management (User groups, Users, Roles, Policies, Identity providers, Account settings, Root access management), and Access reports (Access Analyzer, External access, Unused access, Analyzer settings, Credential report, Organization activity). The main content area has a heading "Identity and Access Management (IAM)". It displays sections for "Access keys (0)" (with a "Create access key" button) and "SSH public keys for AWS CodeCommit (0)" (with a "Upload SSH public key" button). At the bottom, there's a section for "HTTPS Git credentials for AWS CodeCommit (0)" with a "Generate credentials" button. The top right corner shows "Relaunch to update".

This screenshot shows the first step of the "Create access key" wizard. It starts with a title "Step 1" and a section titled "Access key best practices & alternatives" with a "Info" link. Below it, a note says "Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives." There are four radio button options under "Use case": "Command Line Interface (CLI)" (selected), "Local code", "Application running on an AWS compute service", and "Third-party service". Each option has a brief description. The bottom of the screen shows the standard AWS navigation bar.

3. Download the credentials file (.csv) containing the Access Key ID and Secret Access Key.

**Access key created**  
This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

**Retrieve access keys**

**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
AKIARWPFDIE6HT4QY5H	***** Show

**Access key best practices**

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

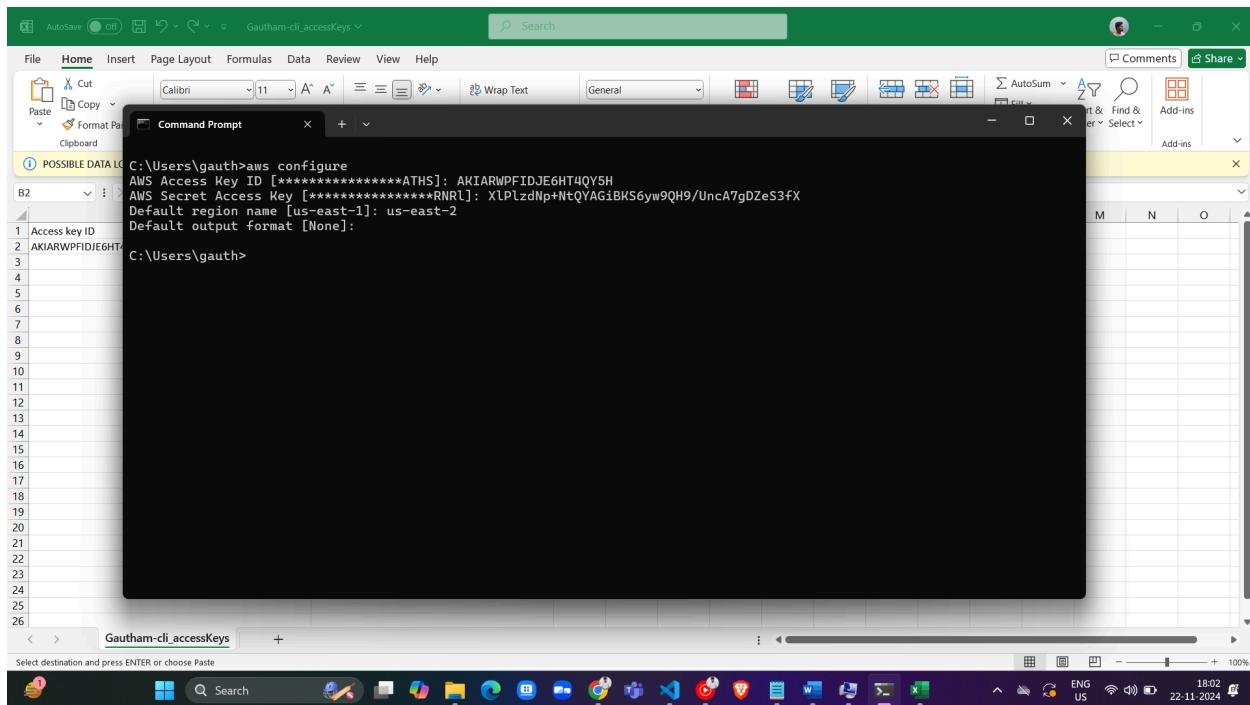
For more details about managing access keys, see the [best practices for managing AWS access keys](#).

[Download .csv file](#) [Done](#)

Access key ID	Secret access key
AKIARWPFDIE6HT4QY5H	XIPlzdNp+NtQYAGIBK56yw9QH9/UncA7gDZeS3fx
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	
26	

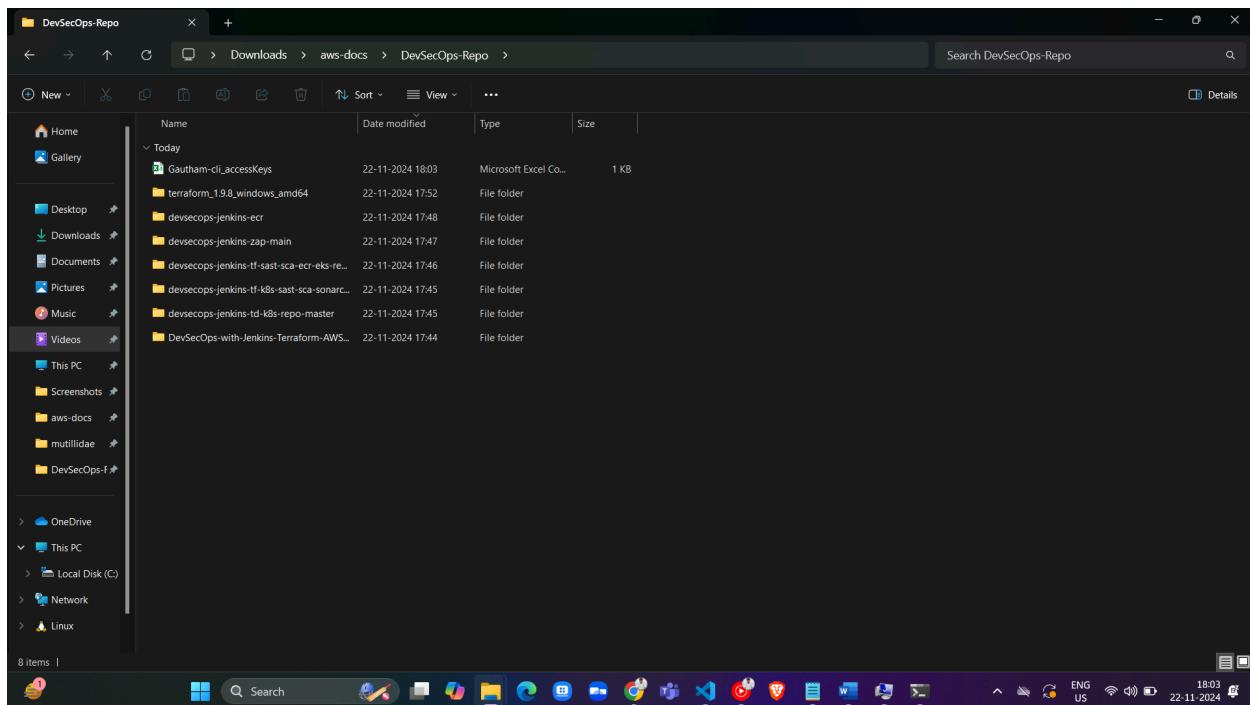
4. Configure AWS CLI: `aws configure`
5. Enter the Access Key ID.
6. Enter the Secret Access Key.
7. Set the default region: `us-east-2`.



## Step 3. Infrastructure Setup with Terraform

### Clone Repository

1. Create a folder **DevSecOps-Repo** on your desktop.

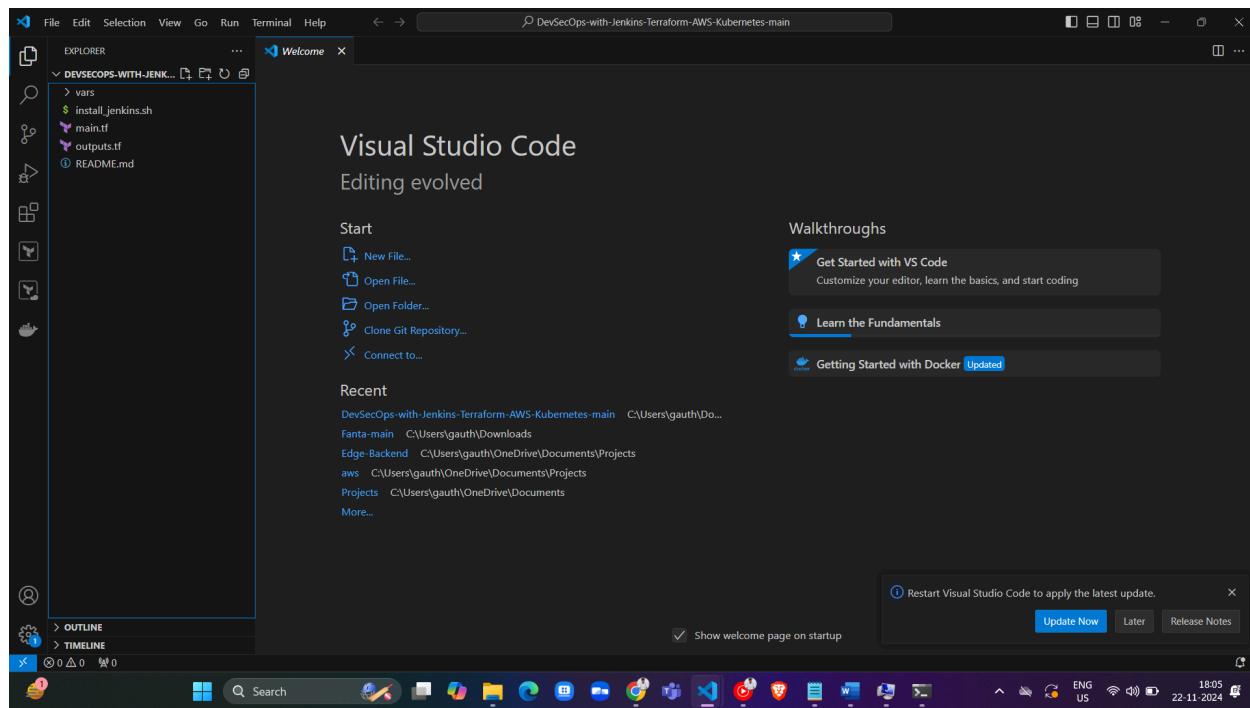


2. Open Git Bash in the folder and clone the repository:

```
git clone
```

```
https://github.com/Akshiv20/DevSecOps-with-Jenkins-Terraform-AWS-Kubernetes.git
```

3. Open the folder in Visual Studio Code.



## Configure Variables

1. Navigate to `vars/dev-east-2.tfvars` and set the following values:

- VPC ID
- IPv4 CIDR block
- KeyPair value

```
dev-east-2.tfvars > main.tf
vars > dev-east-2.tfvars > ...
1 aws_region = "us-east-2"
2
3 vpc_id = "vpc-0a8dcfa7bac90ddaa2"
4
5 cidr_block = "172.31.0.0/16"
6
7 key_name = "proj-lin-pwd"
8
```

The screenshot shows a Windows desktop environment. In the center is a code editor window titled 'DevSecOps-with-Jenkins-Terraform-AWS-Kubernetes-main'. The left sidebar shows a file tree with 'DEVEOPS-WITH-JENKINS-TERRAFORM...', 'vars', 'dev-east-2.tfvars' (which is selected), 'install\_jenkins.sh', 'main.tf', 'outputs.tf', and 'README.md'. The main pane displays a Terraform configuration file named 'main.tf' with the following code:

## Deploy Terraform Resources

- Initialize Terraform:

```
terraform init
```

```
C:\Windows\System32\cmd.exe > + >
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\gauth\Downloads\aws-docs\DevSecOps-Repo\DevSecOps-with-Jenkins-Terraform-AWS-Kubernetes-main>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.77.0...
- Installed hashicorp/aws v5.77.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

C:\Users\gauth\Downloads\aws-docs\DevSecOps-Repo\DevSecOps-with-Jenkins-Terraform-AWS-Kubernetes-main>
```

The screenshot shows a terminal window titled 'C:\Windows\System32\cmd.exe' with the title bar 'DevSecOps-with-Jenkins-Terraform-AWS-Kubernetes-main'. The terminal displays the output of the 'terraform init' command. It starts with the Windows version information, followed by the Terraform initialization process. It shows the download and installation of the 'hashicorp/aws' provider version 5.77.0. It also creates a lock file (.terraform.lock.hcl). Finally, it informs the user that Terraform has been successfully initialized and provides instructions for further use.

- Review the plan:

```
terraform plan -var-file="vars/dev-east-2.tfvars"
```

```
C:\Users\gauth\Downloads\aws-docs\DevSecOps-Repo\DevSecOps-with-Jenkins-Terraform-AWS-Kubernetes-main> terraform plan -var-file="vars/dev-east-2.tfvars"
data.aws_ami.amazon_linux: Reading...
data.aws_ami.amazon_linux: Read complete after 2s [id=ami-09da212cf18033880]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_iam_instance_profile.ec2devsecops_profile will be created
+ resource "aws_iam_instance_profile" "ec2devsecops_profile" {
  + arn          = (known after apply)
  + create_date  = (known after apply)
  + id           = (known after apply)
  + name         = "ec2devsecops_profile"
  + name_prefix  = (known after apply)
  + path         = "/"
  + role         = "devsecops_role"
  + tags_all     = (known after apply)
  + unique_id    = (known after apply)
}

# aws_iam_role.devsecops_role will be created
+ resource "aws_iam_role" "devsecops_role" {
  + arn          = (known after apply)
  + assume_role_policy = jsonencode(
      {
        Statement = [
          +
          {
            + Action      = "sts:AssumeRole"
            + Effect     = "Allow"
            + Principal  = {
                + Service = "ec2.amazonaws.com"
              }
            + Sid        = ""
          ],
        ]
      ],
    )
  + Version     = "2012-10-17"
}
```

- Apply the configuration:

```
terraform apply -var-file="vars/dev-east-2.tfvars"
```

```
C:\Windows\System32\cmd.exe > + ^
}

Plan: 5 to add, 0 to change, 0 to destroy.

Changes to Outputs:
+ output_name = "some resource"

Warning: Value for undeclared variable
The root module does not declare a variable named "cidr_block" but a value was found in file "vars/dev-east-2.tfvars". If you meant to use this value, add a "variable" block to the configuration.

To silence these warnings, use TF_VAR_... environment variables to provide certain "global" settings to all configurations in your organization. To reduce the verbosity of these warnings, use the -compact-warnings option.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_iam_role.devsecops_role: Creating...
aws_security_group.jenkins_sg: Creating...
aws_iam_role.devsecops_role: Creation complete after 2s [id=devsecops_role]
aws_iam_role_policy.devsecops_policy: Creating...
aws_iam_instance_profile.ec2devsecops_profile: Creating...
aws_iam_role_policy.devsecops_policy: Creation complete after 1s [id=devsecops_role:devsecops_policy]
aws_security_group.jenkins_sg: Creation complete after 6s [id=sg-05598c027977005fa]
aws_iam_instance_profile.ec2devsecops_profile: Creation complete after 8s [id=ec2devsecops_profile]
aws_instance.web: Creating...
aws_instance.web: Still creating... [10s elapsed]
aws_instance.web: Creation complete after 15s [id=i-0461426356c36c3ce]

Apply complete! Resources: 5 added, 0 changed, 0 destroyed.

Outputs:
output_name = "some resource"

C:\Users\gauth\Downloads\aws-docs\DevSecOps-Repo\DevSecOps-with-Jenkins-Terraform-AWS-Kubernetes-main>
```

- Verify the created resources in the AWS Console.

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with options like Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), and Network & Security (CloudShell, Feedback). The main area displays a table titled 'Instances (1) Info' with one row. The row details a Jenkins instance named 'Jenkins' with Instance ID 'i-0461426356c36c3ce', which is 'Running' on an 't2.xlarge' instance type. It has an 'Initializing' status check, no alarms, and is located in 'us-east-2a' Availability Zone, Public IP 'ec2-3-1'. A modal window titled 'Select an instance' is open at the bottom, showing the same Jenkins entry. The browser address bar shows 'us-east-2.console.aws.amazon.com/ec2/home?region=us-east-2#Instancesv=3;\$case=tags:true%5C;client:false;\$regex=tags:false%5C;client:false'. The bottom of the screen shows a Windows taskbar with various icons.

## Access Jenkins

1. Open the Jenkins instance via `<public_ip>:8081`.

The screenshot shows a browser window with the URL '3.135.220.22:8081/login?from=%2F'. The page title is 'Getting Started' and the main heading is 'Unlock Jenkins'. It instructs the user that a password has been written to the log and provides the path '/var/lib/jenkins/secrets/initialAdminPassword'. Below this, it says 'Please copy the password from either location and paste it below.' There is a text input field labeled 'Administrator password' and a 'Continue' button at the bottom right. The browser address bar shows 'Not secure' and the full URL. The bottom of the screen shows a Windows taskbar with various icons.

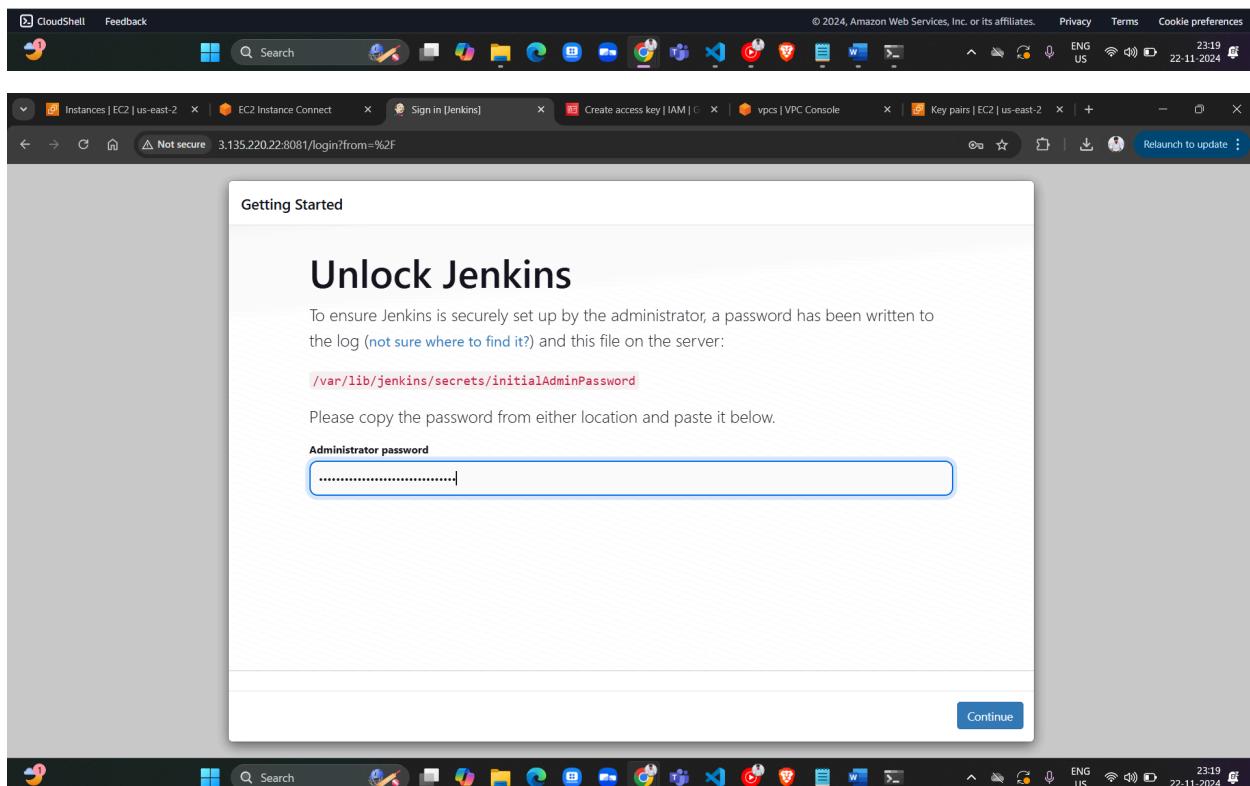
2. Retrieve the Jenkins password from the instance:

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

```
[ec2-user@ip-172-31-15-191 ~]$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword  
2dacccbf55849462b88945d3f1fffdc3d7  
[ec2-user@ip-172-31-15-191 ~]$
```

i-0461426356c36c3ce (Jenkins)

Public IPs: 3.135.220.22 Private IPs: 172.31.15.191



3. Complete Jenkins setup with suggested plugins.

The screenshot shows the Jenkins 'Getting Started' page. At the top, there's a navigation bar with tabs like 'Instances | EC2 | us-east-2', 'EC2 Instance Connect', 'Setup Wizard [Jenkins]', 'Create access key | IAM', 'vpcs | VPC Console', and 'Key pairs | EC2 | us-east-2'. Below the navigation bar is a toolbar with icons for search, refresh, and other functions.

The main content area is titled 'Getting Started' with a sub-section 'API Reference'. It displays a table of Jenkins API endpoints categorized by module:

Module	Endpoint
Folders	OWASP Markup Formatter
Timestamper	Workspace Cleanup
Pipeline	GitHub Branch Source
Git	SSH Build Agents
LDAP	Email Extension
Build Timeout	Ant
Pipeline	Pipeline: GitHub Groovy Libraries
Matrix Authorization Strategy	Pipeline Graph View
Mailer	PAM Authentication
Credentials Binding	Dark Theme
JSON Path API	
Step API	
Pipeline: Step API	
Token Macro	
Build Timeout	
bouncycastle API	
Credentials	
Plain Credentials	
Vault Credentials	
Cloud Credentials	
Credentials Binding	
SCM API	
Pipeline: API	
commons-lang3 v3.x Jenkins API	
Timestamper	
Config API	
String Security	
JavaBeans Activation Framework (JAF) API	
JAXB	
SnakeYAML API	
JSON API	
Jackson 2 API	
commons-text API	
- required dependency	

At the bottom of the page, it says 'Jenkins 2.479.1'.

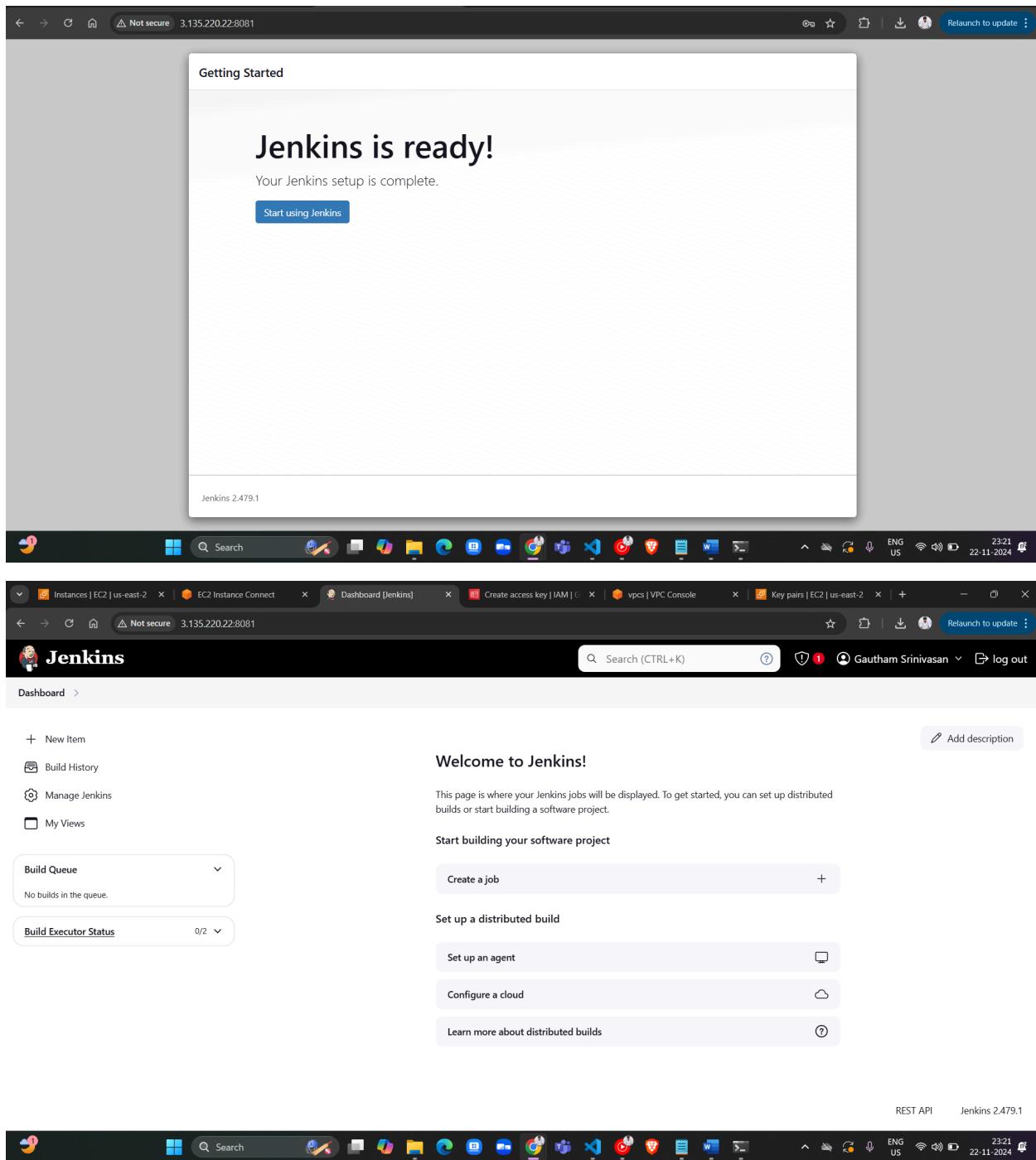
The screenshot shows the 'Create First Admin User' setup page. The URL in the address bar is '3.135.220.22:8081'. The page has a header 'Getting Started' and a sub-section 'Create First Admin User'.

The form fields are as follows:

- Username: admin
- Password: (redacted)
- Confirm password: (redacted)
- Full name: Gautham Srinivasan
- E-mail address: (redacted)

At the bottom right, there are two buttons: 'Skip and continue as admin' and 'Save and Continue'.

At the very bottom of the page, it says 'Jenkins 2.479.1'.



## Step 4. Configure Jenkins

### Set Up Maven

1. Go to Jenkins > Manage Jenkins > Global Tool Configuration.
2. Add a new Maven installation:

- Name: [Maven\\_3\\_2\\_5](#)

Check for maven path using `maven -v`

```
[ec2-user@ip-172-31-15-191 ~]$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
2daccbcf5549462b88945d3f1ffdc3d7
[ec2-user@ip-172-31-15-191 ~]$ mvn -v
Apache Maven 3.2.5 (12a6b3acb947671f09b81f49094c53f426d0ceaf; 2014-12-14T17:29:23+00:00)
Maven home: /usr/share/apache-maven
Java version: 17.0.12, vendor: Amazon.com Inc.
Java home: /usr/lib/jvm/java-17-amazon-corretto.x86_64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.1.109-118.189.amzn2023.x86_64", arch: "amd64", family: "unix"
[ec2-user@ip-172-31-15-191 ~]$
```

[i-0461426356c36c3ce \(Jenkins\)](#)

Public IPs: 3.135.220.22 Private IPs: 172.31.15.191



- Maven home path: [/usr/share/apache-maven](#)

Maven installations	
<a href="#">Add Maven</a>	
<b>Maven</b>	
Name	<input type="text" value="Maven_3_2_5"/>
MAVEN_HOME	<input type="text" value="/usr/share/apache-maven"/>
<input type="checkbox"/> Install automatically	<a href="#">?</a>
<a href="#">Add Maven</a>	
<a href="#">Save</a>	<a href="#">Apply</a>

Jenkins 2.479.1

## Install Plugins

1. Install the following Jenkins plugins:

- Docker Pipeline
- AWS Credentials
- Amazon ECR
- Kubernetes CLI
- Pipeline Stage View

Screenshot of a web browser showing the Jenkins plugin manager interface. The URL is 3.135.220.22:8081/manage/pluginManager/available. The left sidebar shows 'Available plugins' selected. A search bar at the top right contains the text 'pipeline stage'. The main area lists several Jenkins plugins:

- Docker Pipeline** 580.v0c340686b\_54 (Released 6 mo 4 days ago) - Pipeline DevOps Deployment docker. Build and use Docker containers from pipelines.
- AWS Credentials** 231.v08a\_59f17d742 (Released 7 mo 29 days ago) - aws. Allows storing Amazon IAM credentials within the Jenkins Credentials API. Store Amazon IAM access keys (AWSAccessKeyId and AWSSecretKey) within the Jenkins Credentials API. Also support IAM Roles and IAM MFA Token.
- Amazon ECR** 1.136.v914ea\_5948634 (Released 4 mo 28 days ago) - aws. This plugin generates Docker authentication token from Amazon Credentials to access Amazon ECR.

This plugin is up for adoption! We are looking for new maintainers. Visit our [Adopt a Plugin](#) initiative for more information.
- Kubernetes CLI** 1.12.1 (Released 1 yr 2 mo ago) - kubernetes. Configure kubectl for Kubernetes.
- Pipeline: Stage View** 2.34 (Released 1 yr 0 mo ago) - User Interface. Pipeline Stage View Plugin.

The taskbar at the bottom shows various application icons and the date/time 22-11-2024.

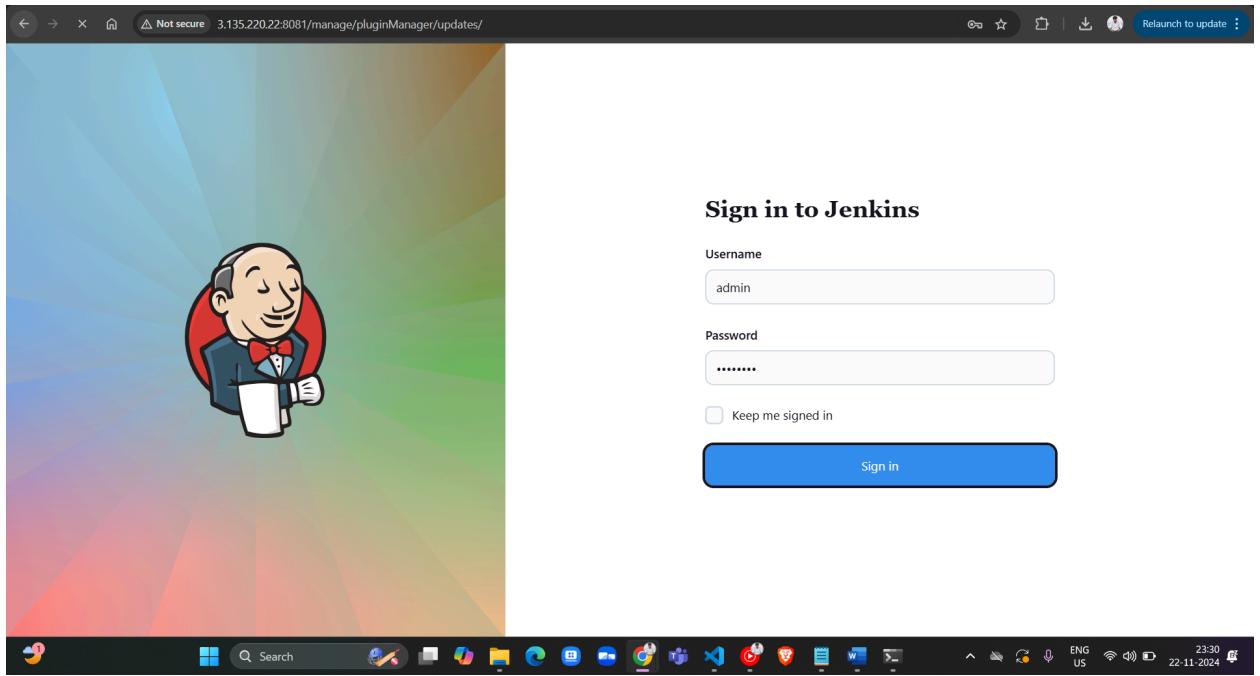


Jenkins is restarting

Your browser will reload automatically when Jenkins is ready

**Safe Restart**  
Builds on agents can usually continue.

Screenshot of a web browser showing the Jenkins plugin manager interface. The URL is 3.135.220.22:8081/manage/pluginManager/updates/. The taskbar at the bottom shows various application icons and the date/time 22-11-2024.



## Step 5. SAST with SonarCloud

### Configure SonarCloud

1. Sign in to [SonarCloud](#) using GitHub.

**SaaS solution for Clean Code. Simple, Scalable, Fast.**

Enable your team to deliver clean code consistently and efficiently with a code review tool that easily integrates into the cloud DevOps platforms and extend your CI/CD workflow.

**Start now**   **Overview**

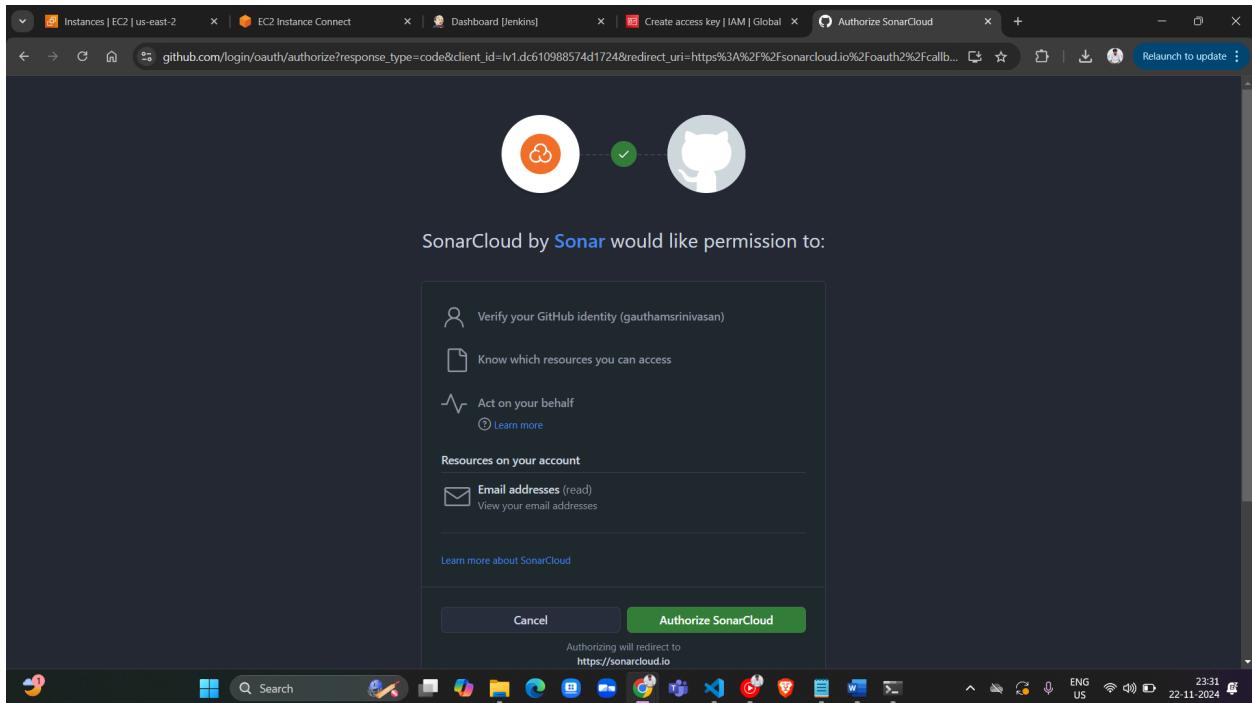
**Reliability**  
2 Open Issues   D  
1 H   1 M   0 L

**Log in to SonarQube Cloud**

GITHUB   BITBUCKET   GITLAB   AZURE DEVOPS

Only available on Enterprise plan

**LOG IN WITH SSO**



2. Create a new organization and project.

**Create an organization**

Organizations enable your team to collaborate across many projects.

**1 Import organization details**

Import Gautham S into a SonarQube Cloud organization

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

**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

Create another organization

Display Name \*

Up to 255 characters

Project Key \*

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.

**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
- ✓ Advance issue detection
- ✓ Deeper SAST
- ✓ Synchronized user management

**Already have a coupon?**

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

The screenshot shows the SonarCloud 'Create project' interface. At the top, there's a note about setting a new code definition for the organization. Below it, two options are presented: 'Previous version' (selected) and 'Number of days'. Both options have descriptions and are marked as recommended for specific project types. A note at the bottom indicates that changes can be made in project administration. At the bottom of the page are 'Back' and 'Create project' buttons.

### 3. Generate a token from the "My Account > Security" tab.

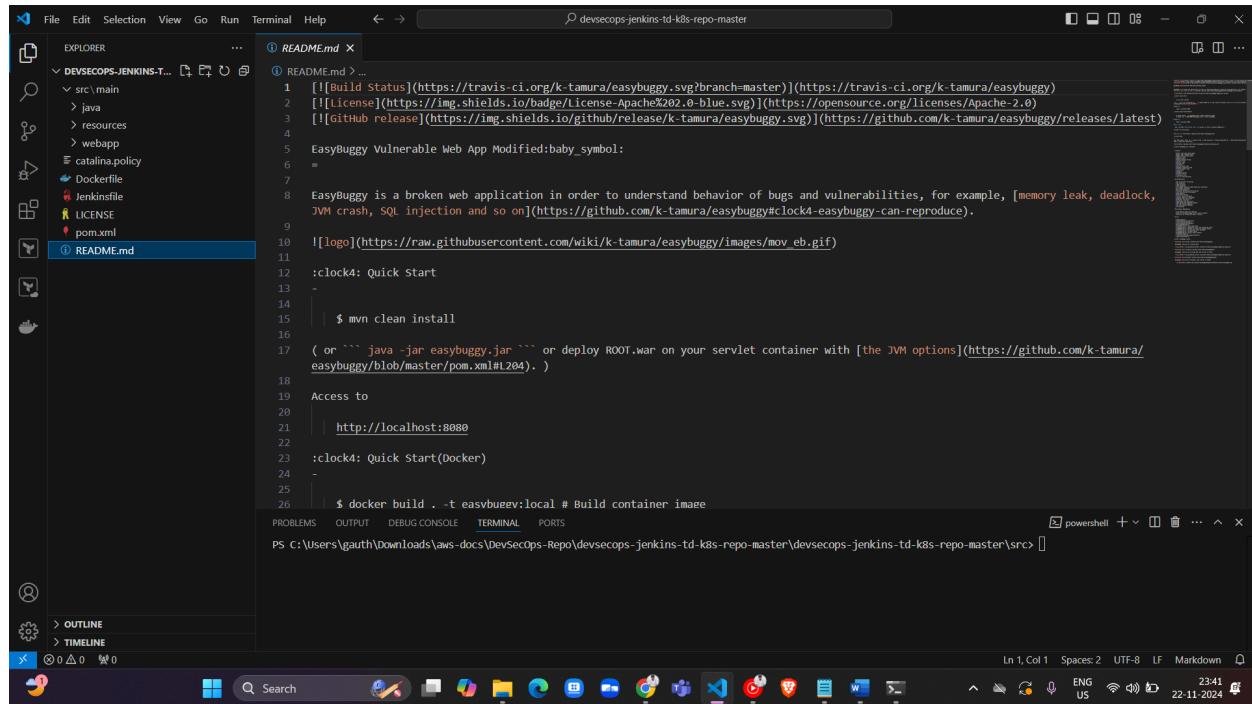
The screenshot shows the SonarCloud 'Security - My Account' page. In the 'Generate Tokens' section, a new token has been created, and its value is displayed. The token value is: 4f1b5bf8dde6e3e3a33f114883cf2a7919a65354. A note says to copy the token now as it won't be seen again. The 'Existing Tokens' table shows one entry: 'token' with 'Never' last use and 'November 22, 2024' creation date, and a 'Revoke' button. On the right, there's a sidebar for upgrading to scan private projects, which includes a 'Upgrade your organizations' button and a 'View FAQs' link.

## Update Jenkins Pipeline

- Clone the repository:

```
git clone
```

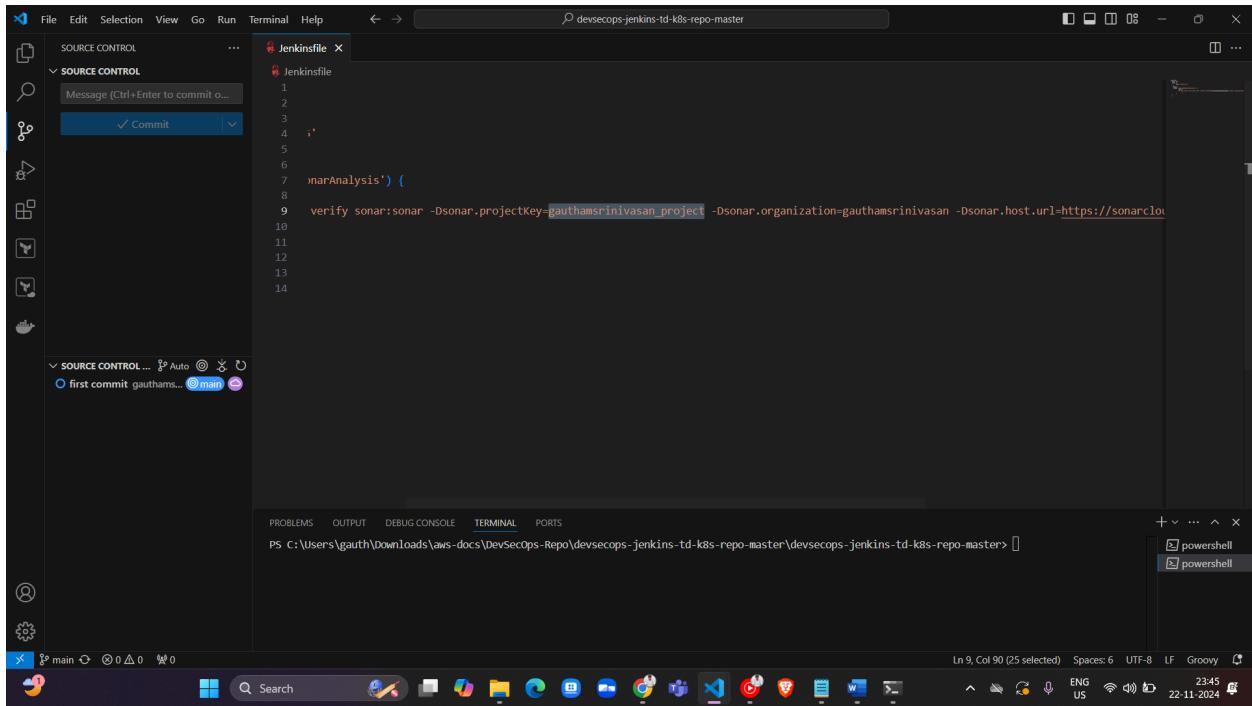
```
https://github.com/Akshiv20/devsecops-jenkins-td-k8s-repo.git
```



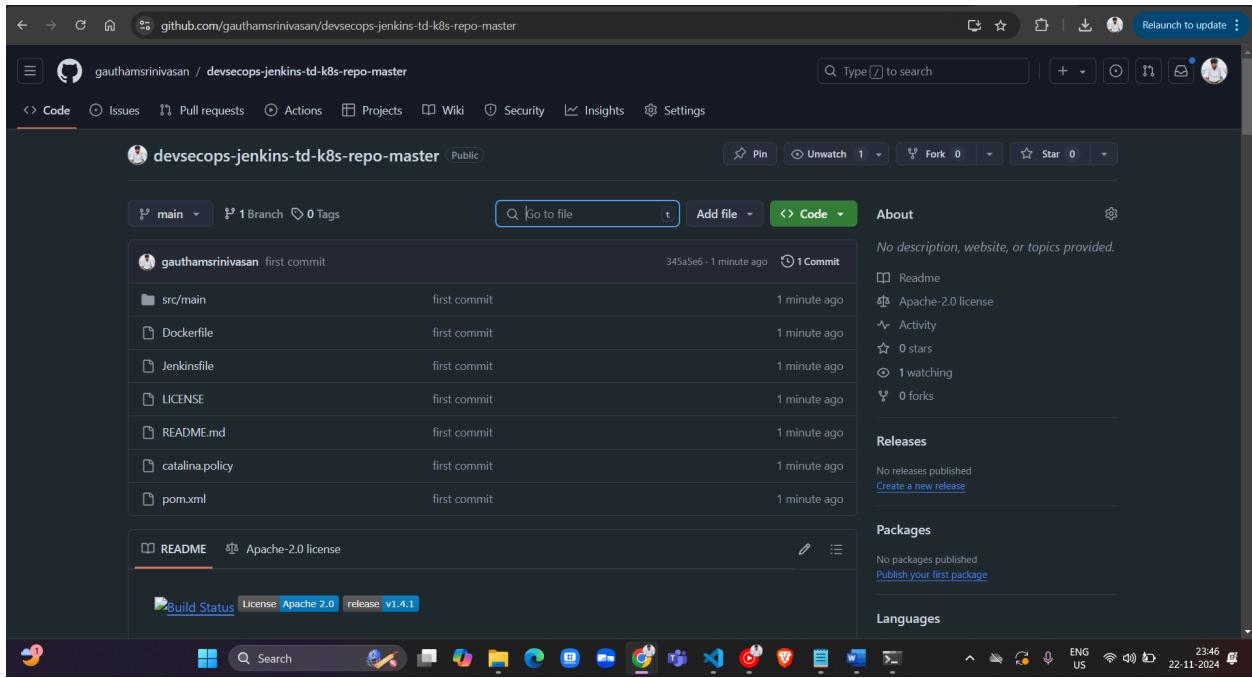
```
READEMe.md × ...
1  [[Build Status]](https://travis-ci.org/k-tamura/easybuggy.svg?branch=master)](https://travis-ci.org/k-tamura/easybuggy)
2  [[License]](https://img.shields.io/badge/License-Apache%202.0-blue.svg)](https://opensource.org/licenses/Apache-2.0)
3  [[GitHub Release]](https://img.shields.io/github/release/k-tamura/easybuggy.svg)](https://github.com/k-tamura/easybuggy/releases/latest)
4
5 EasyBuggy Vulnerable Web App Modified:baby_symbol:
6 =
7
8 EasyBuggy is a broken web application in order to understand behavior of bugs and vulnerabilities, for example, [memory leak, deadlock, JVM crash, SQL injection and so on](https://github.com/k-tamura/easybuggy#clock4-easybuggy-can-reproduce).
9
10 
11 :clock4: Quick Start
12 -
13
14 | $ mvn clean install
15
16 ( Or ```` java -jar easybuggy.jar ```` or deploy ROOT.war on your servlet container with [the JVM options](https://github.com/k-tamura/easybuggy/blob/master/pom.xml#L204). )
17
18 Access to
19
20 | http://localhost:8080
21
22 :clock4: Quick Start(Docker)
23 -
24
25
26 $ docker build . -t easvbuggev:local # Build container imagee
```

- Update the following values in the [Jenkinsfile](#):

- Sonar project key
- Organization key
- Token

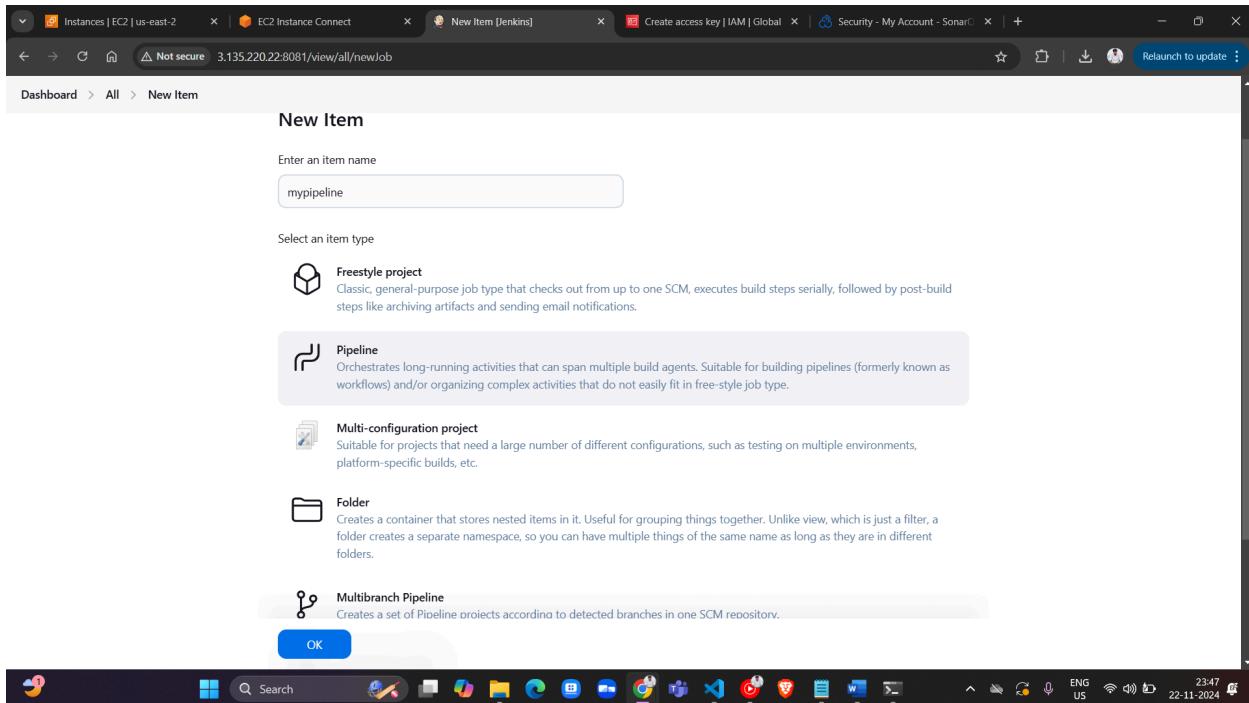


Save and publish the repo to your github account and verify



## Run SAST Scan

1. Create a new Jenkins pipeline.



## 2. Configure the pipeline to use the [devsecops-jenkins-td-k8s-repo](#) repository.

Definition  
Pipeline script from SCM

SCM  
Git

Repositories

Repository URL  
https://github.com/gauthamsrinivasan/devsecops-jenkins-td-k8s-repo-master.git

Credentials  
- none -

Advanced

Add Repository

Save Apply

## 3. Trigger a build and verify the results on SonarCloud.

Dashboard > mypipeline >

Status

Changes: creating the devsecops pipeline

Build Now

Configure

Delete Pipeline

Full Stage View

Stages

Rename

Pipeline Syntax

Stage View

Average stage times: (Average full run time: ~1min 10s)

Declarative: Checkout SCM	Declarative: Tool Install	Compile and Run Sonar Analysis
435ms	162ms	1min 3s

#1 Nov 22 23:52 No Changes 435ms 162ms 1min 3s

Builds

Filter

Today #1 6:22 PM

Permalinks

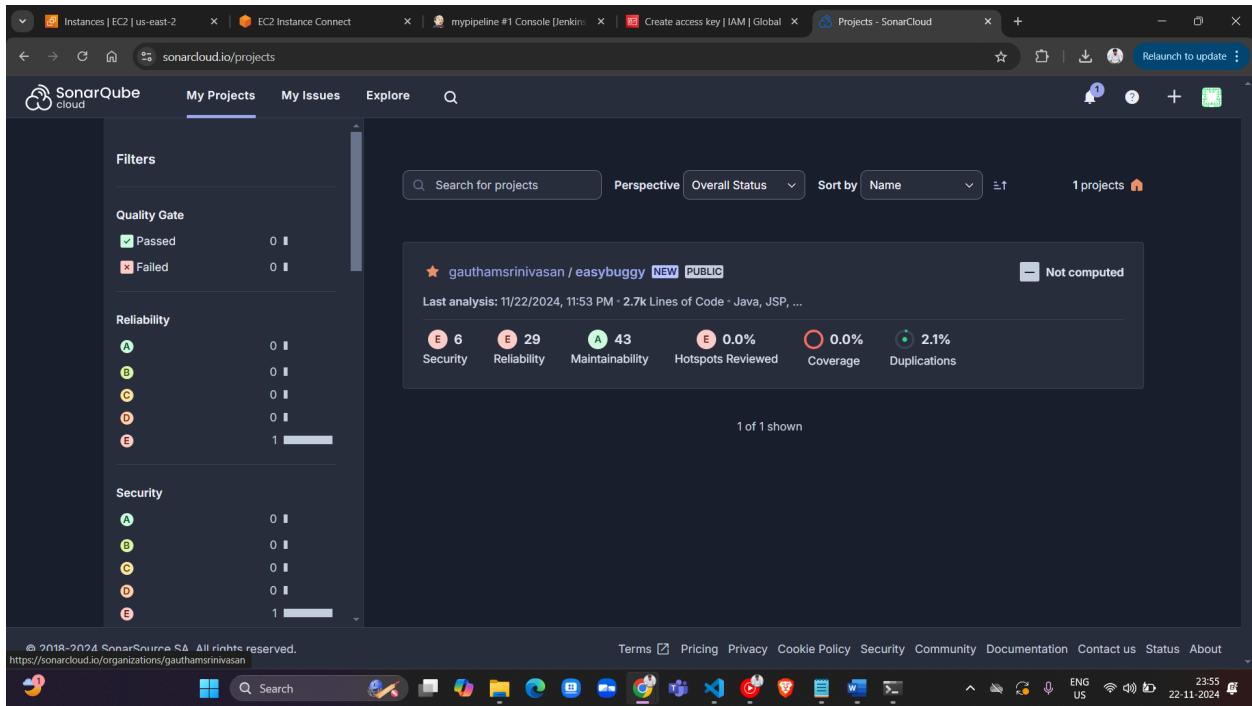
- Last build (#1), 22 sec ago

REST API Jenkins 2.479.1

Dashboard > mypipeline > #1

```
[INFO] 18:23:58.965 ANALYSIS SUCCESSFUL, you can find the results at: https://sonarcloud.io/dashboard?id=gauthamsrinivasan\_project
[INFO] 18:23:58.965 Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
[INFO] 18:23:58.965 More about the report processing at https://sonarcloud.io/api/ce/task?id=A2NWhdVbMExfhLDnaFCC
[INFO] 18:24:00.082 Sensor cache published successfully
[INFO] 18:24:00.668 Analysis total time: 36.374 s
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 01:01 min
[INFO] Finished at: 2024-11-22T18:24:00+00:00
[INFO] Final Memory: 88M/308M
[INFO] -----
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // stage
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

REST API Jenkins 2.479.1



## Step 6. SCA with Snyk

### 1. Configure Snyk

1. Sign up for [Snyk](#) using GitHub.

snyk.io/product/open-source-security-management/?utm\_medium=paid-search&utm\_source=google&utm\_campaign=gs\_sn-brand-ind&utm\_content=br\_sca&...

Products Resources Company Pricing

EN Login Sign up Book a live demo

Snyk Open Source Overview Docs

Snyk's 2023 State of Open Source Security report now available →

 SNYK OPEN SOURCE

# Open source risk management made for developers

Snyk Open Source provides advanced software composition analysis (SCA) backed by industry-leading security and application intelligence.

Book a live demo



23:56 ENG US 22-11-2024

app.snyk.io/login

Relaunch to update



## Automatically find and fix vulnerabilities in your code, open source, and containers

-  Seamlessly integrate your projects
-  Instantly scan for vulnerabilities
-  Fix quickly with a pull request

Get started for free  
No credit card required

 GitHub

 Google

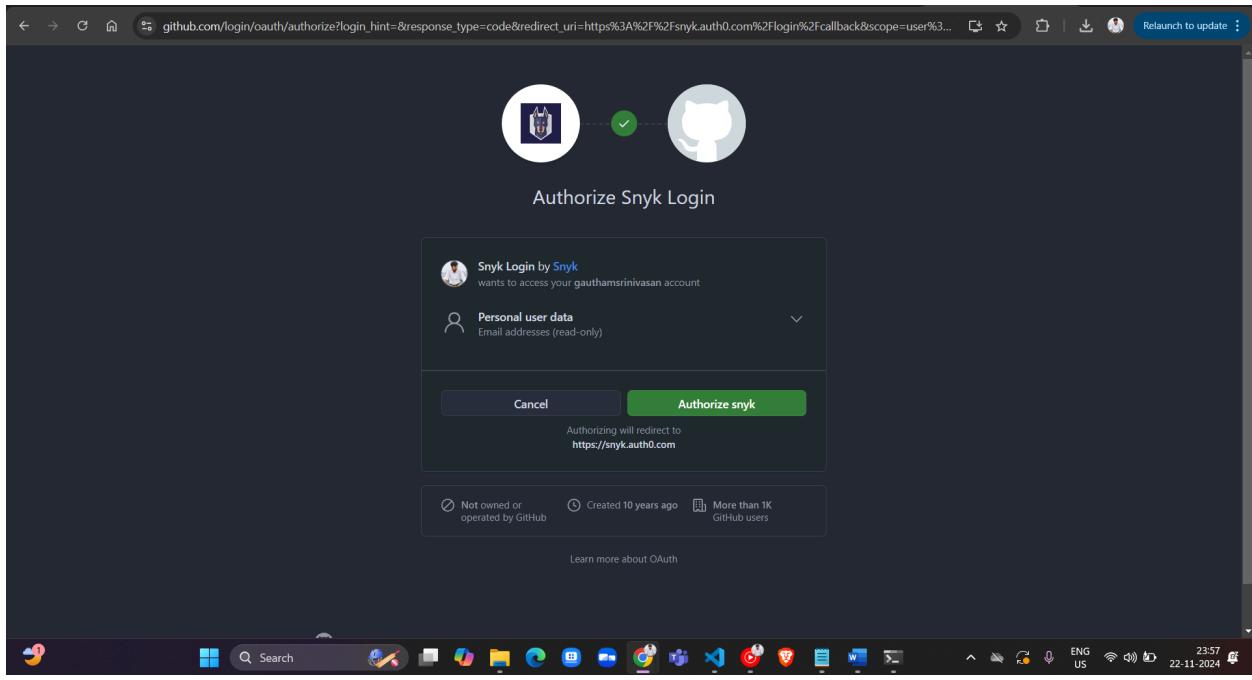
Or Sign up with: Bitbucket | Azure AD | Docker ID

Log in with your company SSO

We will not make any use of the auth provider without your permission.  
By logging in or signing up, you agree to abide by our policies, including our Terms of Service and Privacy Policy

JOIN THE MILLIONS OF DEVELOPERS WHO BUILD SECURELY WITH SNYK

23:56 ENG US 22-11-2024



The screenshot shows a web browser window with the URL app.snyk.io/org/gauthamsrinivasan. The page has a dark header with the Snyk logo and the organization name "gauthamsrinivasan". The main content area is titled "Start securing your code" and contains two items: "Connect your code" (with a sub-note "Connect Snyk to your code to fix issues and vulnerabilities.") and "Add and scan your first project" (with a sub-note "Import your code to see how Snyk surfaces issues, problematic dependencies, and vulnerabilities."). Below these is a button "Add projects". Further down, there's a section "Invite team members" with a "Copy invite link" button, and another section "Use Snyk in the command line" with a "Learn more" button. On the left side, there's a sidebar with links for "Dashboard", "Projects", "Integrations", "Members", and "Settings". At the bottom left, there's a notification for "Product updates". The bottom of the screen shows a taskbar with various application icons and system status indicators.

2. Generate an authentication token from "Account Settings > Auth Tokens".

The screenshot shows the Snyk account settings page for the organization "gauthamsrinivasan". The "General" tab is selected under "ACCOUNT SETTINGS". The "Auth Token" section contains a token key "28eb52f7-2189-4aec-8dc8-1f1529afe0fc" created on "22 November 2024, 23:57:07". A "Revoke & Regenerate" button is available. Below this is the "Authorized Applications" section, which currently shows "No applications". The "Preferred Organization" section has "gauthamsrinivasan" selected. The browser toolbar at the bottom includes icons for search, file operations, and system status.

This screenshot is identical to the one above, displaying the Snyk account settings page for the organization "gauthamsrinivasan". The "General" tab is selected, and the "Auth Token" section shows the same token details and interface. The "Authorized Applications" and "Preferred Organization" sections are also present. The browser toolbar at the bottom is identical.

## 2. Add Credentials in Jenkins

1. Go to Jenkins > Manage Jenkins > Credentials.
2. Add a new secret text:
  - ID: [SNYK\\_TOKEN](#)
  - Secret: Paste the Snyk token.

New credentials

Kind

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

Secret

ID ? SNYK\_TOKEN

Description ? SNYK\_TOKEN

Create

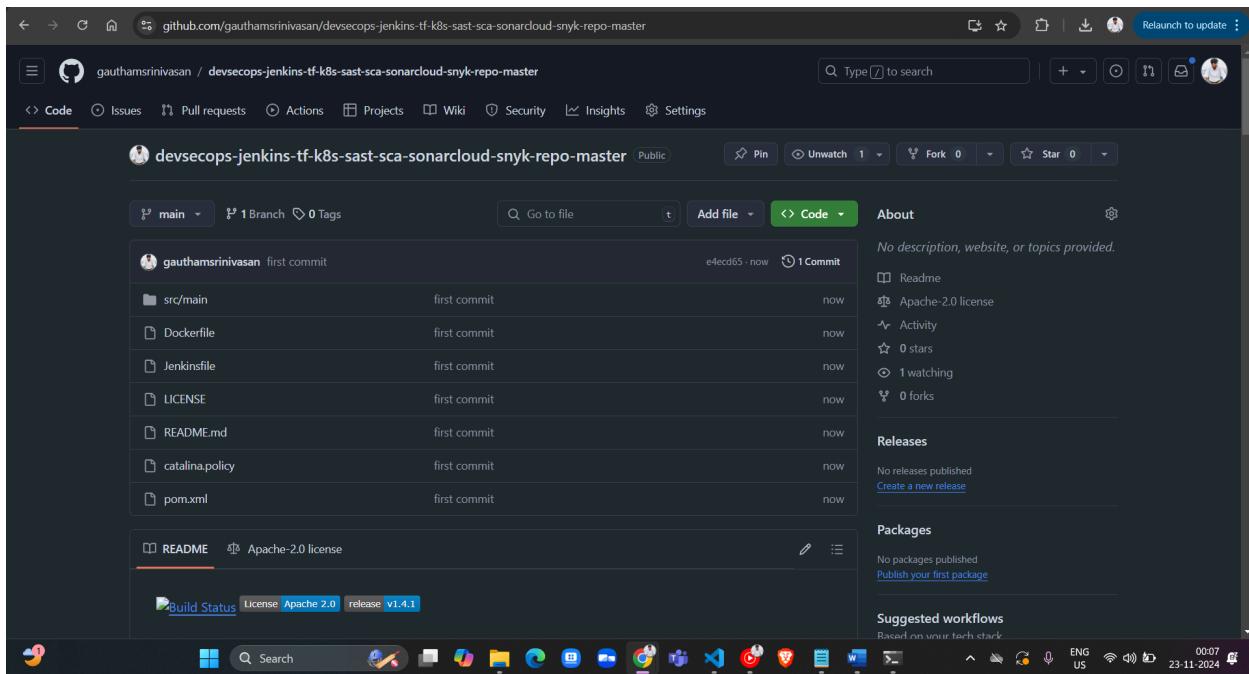
### 3. Update Jenkins Pipeline

1. Clone the repository:

```
git clone  
https://github.com/Akshiv20/devsecops-jenkins-tf-k8s-sast-sca-son  
arccloud-snyk-repo.git
```

2. Update the [Jenkinsfile](#) with the Sonar token and Snyk token.

Save and publish the repo to your github account and verify



- Change the github url in the pipeline and build now and verify

Not secure 3.135.220.22:8081/job/mypipeline/configure

Dashboard > mypipeline > Configuration

### Configure

General Advanced Project Options Pipeline

SCM ?

Repositories ?

Repository URL ? https://github.com/gauthamsrinivasan/devsecops-jenkins-tf-k8s-sast-sca-sonarcloud-snyk-repo-master.git

Credentials ? - none - + Add Advanced

Add Repository Branches to build ? Branch Specifier (blank for 'any') ?

Save Apply

Not secure 3.135.220.22:8081/job/mypipeline/

Jenkins

Dashboard > mypipeline >

### mypipeline

Status Changes Build Now Configure Delete Pipeline Full Stage View Stages Rename Pipeline Syntax

Average stage times: (Average full run time: ~1min 8s)

Declarative: Checkout SCM	Declarative: Tool Install	CompileandRunSonarAnalysis	RunSCAAnalysisUsingSnyk
440ms	139ms	54s	18s
#2 Nov 23 00:09 No Changes	445ms	117ms	45s
#1 Nov 22 23:52 No Changes	435ms	162ms	1min 3s

Builds Filter Permalinks

- Last build (#1), 16 min ago
- Last stable build (#1), 16 min ago

The screenshot shows a Jenkins pipeline console output. The log starts with an [INFO] message indicating a failed goal execution. It then lists several [Pipeline] steps, likely related to configuration or environment setup. Finally, it concludes with a 'Finished: SUCCESS' message. The Jenkins version is 2.479.1.

```
[INFO] -----
[ERROR] Failed to execute goal io.snyk:snyk-maven-plugin:2.0.0:test (default-cli) on project easybuggy: snyk command exited with non-zero exit code (1). See output for details. -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -x switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://wiki.apache.org/confluence/display/MAVEN/MojoFailureException
[INFO] Build failures were ignored.
[Pipeline]
[Pipeline] // withCredentials
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // stage
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

REST API Jenkins 2.479.1

## Step 7. Build and Push to ECR

### 1. Configure Docker Credentials

#### 1. Add Docker Hub credentials in Jenkins:

- ID: [dockerlogin](#)
- Username and Password: Your Docker Hub credentials.

The screenshot shows the 'New credentials' configuration page in Jenkins. The 'Kind' dropdown is set to 'Username with password'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Username' field contains 'gauthamsrinivasan'. The 'Password' field contains '\*\*\*\*\*'. The 'ID' field is set to 'dockerlogin'. The 'Create' button is visible at the bottom.

New credentials

Kind

Username with password

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

Username ? gauthamsrinivasan

Treat username as secret ?

Password ? \*\*\*\*\*

ID ? dockerlogin

Description ?

Create

## 2.Create ECR

1. Go to amazon ecr service and create a new repository.

The screenshot shows the Amazon ECR service landing page. At the top right, there is a prominent 'Create a repository' button with an orange 'Create' button below it. To the left of the button, there is descriptive text about ECR: 'Amazon Elastic Container Registry (ECR) is a fully managed container registry that makes it easy to store, manage, share, and deploy your container images and artifacts anywhere.' Below this, there is a section titled 'How it works' with a diagram illustrating the workflow from writing code to running containers. On the right side, there is a 'Pricing (US)' section stating 'You pay only for the amount of data you store in your public or private repositories and data transferred to the Internet.' Below this, there is a link to 'ECR pricing'. At the bottom of the page, there is a navigation bar with links for CloudShell, Feedback, Search, and various AWS services like Lambda, S3, and CloudWatch. The status bar at the bottom right shows the date as 23-11-2024 and the time as 00:27.

The screenshot shows the 'Create private repository' wizard. The first step, 'General settings', is displayed. It includes fields for 'Repository name' (containing '116981766729.dkr.ecr.us-east-2.amazonaws.com/myimg') and 'Image tag mutability' (set to 'Mutable'). A note states that tag mutability can't be changed once the repository is created. The second step, 'Encryption settings', is shown below with 'Encryption configuration' set to 'AES-256'. A note indicates that encryption settings are不可变 (cannot be changed). At the bottom, there is a 'Next Step' button. The status bar at the bottom right shows the date as 23-11-2024 and the time as 00:27.

2. Copy the uri ("116981766729.dkr.ecr.us-east-2.amazonaws.com/myimg")

The screenshot shows the AWS ECR console interface. On the left, there's a navigation sidebar with sections for Amazon Elastic Container Registry (Private registry and Public registry), ECR public gallery, Amazon ECS, and Amazon EKS. The main area is titled "Private repositories" and shows a table of repositories. A green success message at the top says "Successfully created myimg". The repository table has columns for Repository name, URI, Created at, Tag immutability, and Encryption type. One row is listed: "myimg" with URI "116981766729.dkr.ecr.us-east-2.amazonaws.com/myimg", created on "November 23, 2024, 00:27:39 (UTC+05.5)", mutable tags, and AES-256 encryption. A tooltip "Copied URI to clipboard" is shown over the URI column. At the bottom of the page, there's a standard Windows taskbar with various icons.

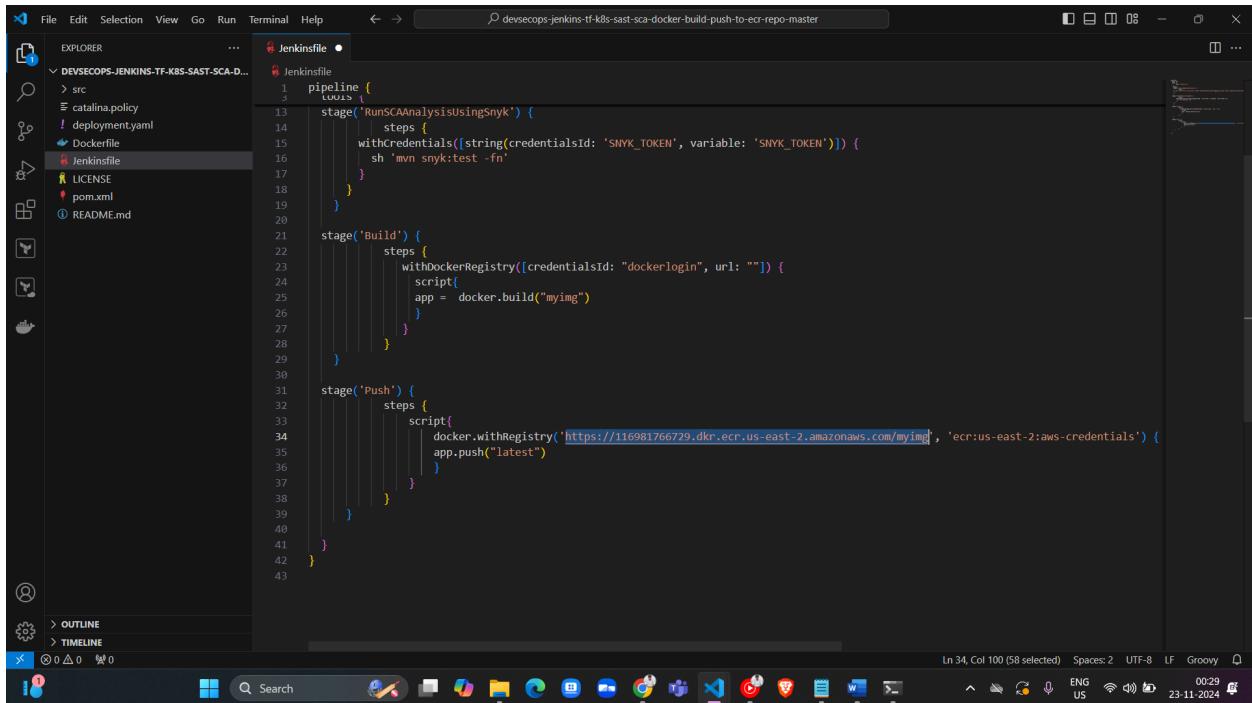
### 3. Update Jenkins Pipeline

1. Clone the repository:

```
git clone
```

```
https://github.com/Akshiv20/devsecops-jenkins-tf-k8s-sast-sca-doc  
ker-build-push-to-ecr-repo.git
```

2. Update the `Jenkinsfile` with the ECR repository URI.

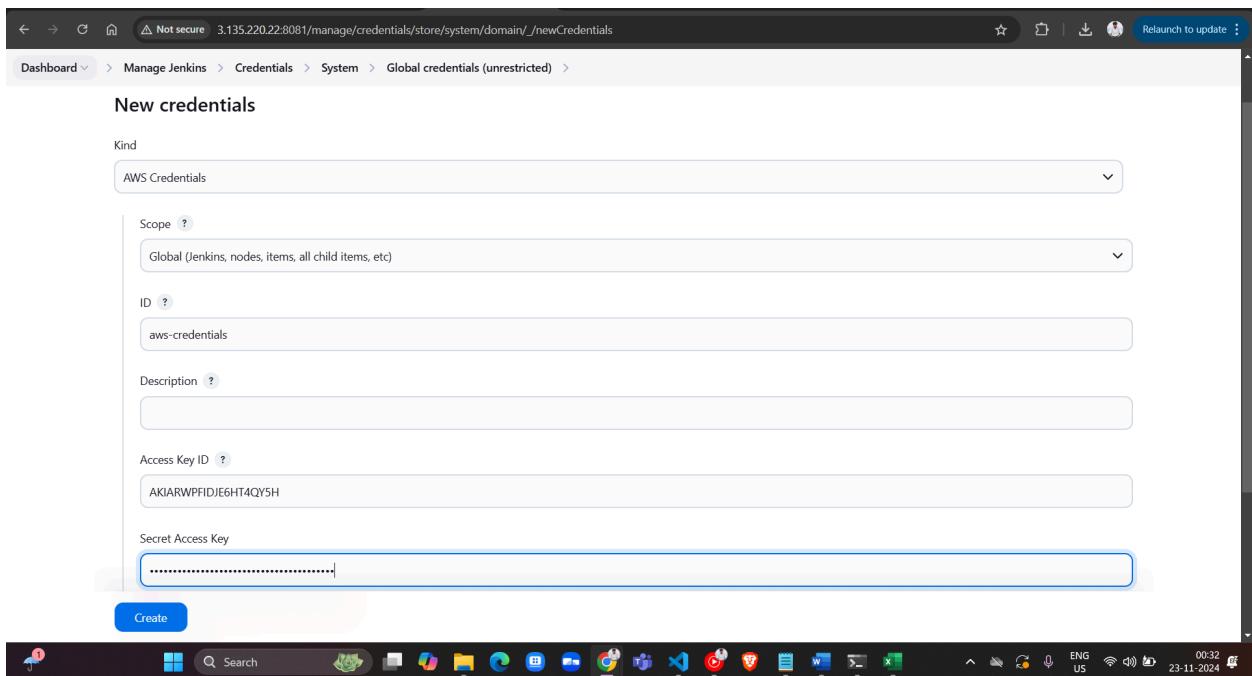


```
File Edit Selection View Go Run Terminal Help devsecops-jenkins-tf-k8s-sast-sca-docker-build-push-to-ecr-repo-master
EXPLORER
DEVSECOPS-JENKINS-TF-K8S-SAST-SCA-D...
src
catalina.policy
deployment.yaml
Dockerfile
Jenkinsfile
LICENSE
pom.xml
README.md
Jenkinsfile
1 Jenkinsfile
2
3 pipeline {
4     tools {
5         stage('RunSCAAnalysisUsingSnyk') {
6             steps {
7                 withCredentials([string(credentialsId: 'SNYK_TOKEN', variable: 'SNYK_TOKEN')]) {
8                     sh 'mvn snyk:test -fn'
9                 }
10            }
11        }
12        stage('Build') {
13            steps {
14                withDockerRegistry([credentialsId: "dockerlogin", url: ""]) {
15                    script{
16                        app = docker.build("myimg")
17                    }
18                }
19            }
20        }
21        stage('Push') {
22            steps {
23                script{
24                    docker.withRegistry('https://116981766729.dkr.ecr.us-east-2.amazonaws.com/myimg', 'ecr:us-east-2:aws-credentials') {
25                        app.push("latest")
26                    }
27                }
28            }
29        }
30    }
31 }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }
```

Ln 34, Col 100 (58 selected) Spaces: 2 UTF-8 LF Groovy

## 4. Add AWS Credentials to Jenkins

1. Go to Jenkins > Credentials > Add AWS Credentials.



New credentials

Kind: AWS Credentials

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

ID: aws-credentials

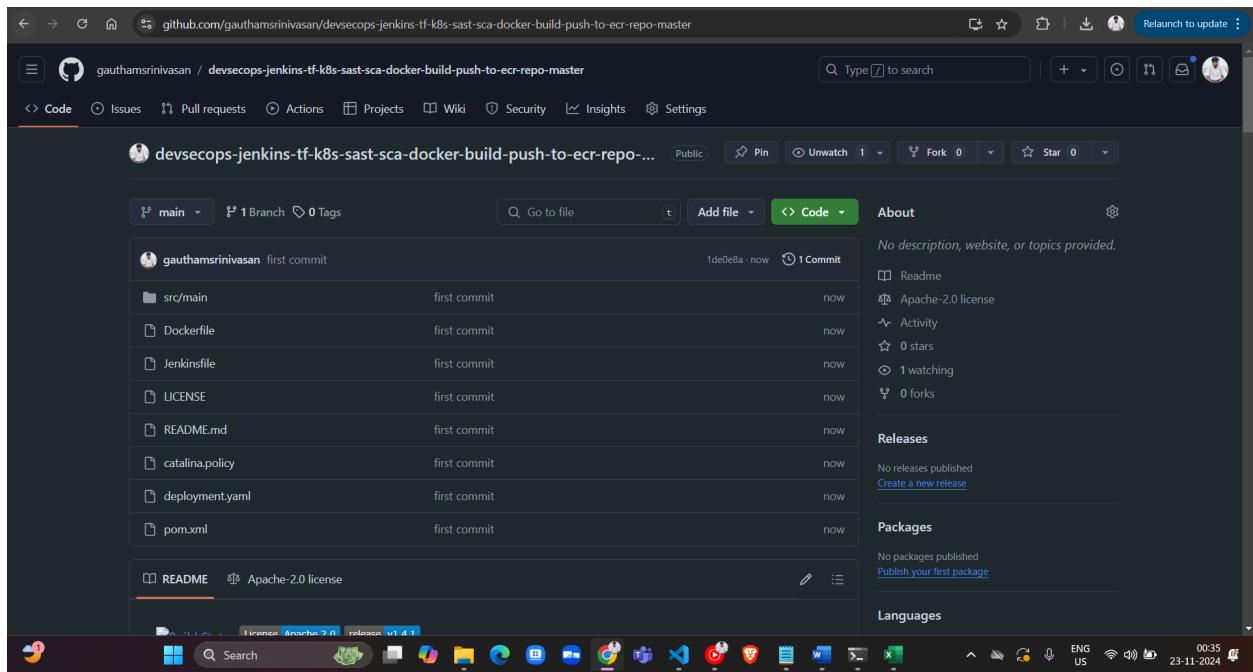
Description:

Access Key ID: AKIARWPFIJDIE6HT4QY5H

Secret Access Key: .....|

Create

Save and publish the folder to github and verify the repository.



## 5. Push Image to ECR

1. Trigger the pipeline in Jenkins.

The screenshot shows the Jenkins Pipeline configuration page. The left sidebar has tabs for General, Advanced Project Options, and Pipeline, with Pipeline selected. The main area is titled 'Pipeline' and 'Definition'. It shows a dropdown for 'Pipeline script from SCM' and a 'SCM' section. In the 'SCM' section, 'Git' is selected, and a 'Repositories' section is expanded, showing a 'Repository URL' input field with the value 'https://github.com/gauthamsrinivasan/devsecops-jenkins-tf-k8s-sast-sca-docker-build-push-to-ecr-repo-master.git'. Below this are 'Credentials' and 'Advanced' dropdowns. At the bottom are 'Save' and 'Apply' buttons.

The screenshot shows the Jenkins Pipeline status page for 'mypiipeline'. The left sidebar has tabs for Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Stages, Rename, and Pipeline Syntax, with Status selected. The main area shows a green checkmark icon and the pipeline name 'mypiipeline'. A message says 'creating the devsecops pipeline'. Below this is a 'Stage View' table:

	Declarative: Checkout SCM	Declarative: Tool Install	CompileandRunSonarAnalysis	RunSCAAnalysisUsingSnyk	Build	Push
Average stage times: (Average full run time: ~1min 23s)	420ms	132ms	52s	14s	43s	10s
#3 Nov 23 00:39 No Changes	380ms	118ms	48s	10s	43s	10s
#2 Nov 23 00:09 No Changes	445ms	117ms	45s	18s		
#1 Nov 22 23:52 No Changes	435ms	162ms	1min 3s			

Below the table is a 'Builds' section with three entries: #3 7:09 PM, #2 6:39 PM, and #1 6:22 PM. At the bottom is a 'Permalinks' section with links for the last build and the last stable build.

2. Verify the image in the AWS ECR Console.

The screenshot shows the AWS ECR console interface. On the left, there's a navigation sidebar with sections for Private registry (Repositories, Images, Permissions, Lifecycle Policy, Repository tags, Features & Settings) and Public registry (ECR public gallery, Amazon ECS, Amazon EKS). The main area is titled 'Images (1)' and shows a table with one row. The table columns are Image tag, Artifact type, Pushed at, Size (MB), Image URI, and Digest. The single entry is 'latest' under 'Image tag', 'Image' under 'Artifact type', 'November 23, 2024, 00:40:59 (UTC+05.5)' under 'Pushed at', '166.76' under 'Size (MB)', and 'sha256:993ab76869cea0...' under 'Image URI'. There are buttons for Delete, Details, Scan, and View push commands.

## Step 8. Kubernetes Deployment

### Create EKS Cluster

1. Create the cluster:

```
eksctl create cluster --name kubernetes-cluster --version 1.31  
--region us-east-2 --nodegroup-name linux-node --node-type  
t2.xlarge --nodes 2
```

(Wait approximately 10 mins for the process to complete..)

```
[ec2-user@ip-172-31-15-191 ~]$ eksctl create cluster --name kubernetes-cluster --version 1.31 --region us-east-2 --nodegroup-name linux-node --node-type t2.xlarge --no-des 2
2024-11-22 19:15:50 [I] eksctl version 0.194.0
2024-11-22 19:15:50 [I] using region us-east-2
2024-11-22 19:15:50 [I] setting availability zones to [us-east-2b us-east-2a us-east-2c]
2024-11-22 19:15:50 [I] subnets for us-east-2b - public:192.168.0.0/19 private:192.168.96.0/19
2024-11-22 19:15:50 [I] subnets for us-east-2a - public:192.168.32.0/19 private:192.168.128.0/19
2024-11-22 19:15:50 [I] subnets for us-east-2c - public:192.168.64.0/19 private:192.168.160.0/19
2024-11-22 19:15:50 [I] nodegroup "linux-node" will use "[AmazonLinux2/1.31]"
2024-11-22 19:15:50 [I] using Kubernetes version 1.31
2024-11-22 19:15:50 [I] creating EKS cluster "kubernetes-cluster" in "us-east-2" region with managed nodes
2024-11-22 19:15:50 [I] will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2024-11-22 19:15:50 [I] if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=us-east-2 --cluster=kubernetes-cluster'
2024-11-22 19:15:50 [I] Kubernetes API endpoint access will use default of (publicAccess=true, privateAccess=false) for cluster "kubernetes-cluster" in "us-east-2"
2024-11-22 19:15:50 [I] CloudWatch logging will not be enabled for cluster "kubernetes-cluster" in "us-east-2"
2024-11-22 19:15:50 [I] you can enable it with 'eksctl utils update-cluster-logging --enable-types=[SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)] --region=us-east-2 --cluster=kubernetes-cluster'
2024-11-22 19:15:50 [I] default addons 'vpc-cni', 'kube-proxy', 'coredns' were not specified, will install them as EKS addons
2024-11-22 19:15:50 [I]
2 sequential tasks: { create cluster control plane "kubernetes-cluster",
  2 sequential sub-tasks: {
    2 sequential sub-tasks: {
      1 task: { create addons },
      wait for control plane to become ready,
    },
    create managed nodegroup "linux-node",
  }
}
```

i-0461426356c36c3ce (Jenkins)

PublicIPs: 3.135.220.22 PrivateIPs: 172.31.15.191

```
2024-11-22 19:29:52 [I] creating addon
2024-11-22 19:29:52 [I] successfully created addon
2024-11-22 19:29:52 [I] creating addon
2024-11-22 19:29:53 [I] successfully created addon
2024-11-22 19:27:53 [I] building managed nodegroup stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 19:27:53 [I] deploying stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 19:27:53 [I] waiting for CloudFormation stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 19:28:23 [I] waiting for CloudFormation stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 19:29:01 [I] waiting for CloudFormation stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 19:29:38 [I] waiting for CloudFormation stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 19:30:10 [I] waiting for CloudFormation stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 19:31:32 [I] waiting for CloudFormation stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 19:31:32 [I] waiting for the control plane to become ready
2024-11-22 19:31:34 [V] saved kubeconfig as "/home/ec2-user/.kube/config"
2024-11-22 19:31:34 [I] no tasks
2024-11-22 19:31:34 [V] all EKS cluster resources for "kubernetes-cluster" have been created
2024-11-22 19:31:34 [V] created 0 nodegroup(s) in cluster "kubernetes-cluster"
2024-11-22 19:31:34 [I] nodegroup "linux-node" has 2 node(s)
2024-11-22 19:31:34 [I] node "ip-192-168-63-199.us-east-2.compute.internal" is ready
2024-11-22 19:31:34 [I] node "ip-192-168-69-198.us-east-2.compute.internal" is ready
2024-11-22 19:31:34 [I] waiting for at least 2 node(s) to become ready in "linux-node"
2024-11-22 19:31:34 [I] nodegroup "linux-node" has 2 node(s)
2024-11-22 19:31:34 [I] node "ip-192-168-63-199.us-east-2.compute.internal" is ready
2024-11-22 19:31:34 [I] node "ip-192-168-69-198.us-east-2.compute.internal" is ready
2024-11-22 19:31:35 [V] created 1 managed nodegroup(s) in cluster "kubernetes-cluster"
2024-11-22 19:31:35 [I] kubectl command should work with "/home/ec2-user/.kube/config", try 'kubectl get nodes'
2024-11-22 19:31:35 [V] EKS cluster "kubernetes-cluster" in "us-east-2" region is ready
[ec2-user@ip-172-31-15-191 ~]$
```

i-0461426356c36c3ce (Jenkins)

PublicIPs: 3.135.220.22 PrivateIPs: 172.31.15.191

```
2024-11-22 19:29:52 [I] creating addon
2024-11-22 19:29:52 [I] successfully created addon
2024-11-22 19:29:52 [I] creating addon
2024-11-22 19:29:53 [I] successfully created addon
2024-11-22 19:27:53 [I] building managed nodegroup stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 19:27:53 [I] deploying stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 19:27:53 [I] waiting for CloudFormation stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 19:28:23 [I] waiting for CloudFormation stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 19:29:01 [I] waiting for CloudFormation stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 19:29:38 [I] waiting for CloudFormation stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 19:30:10 [I] waiting for CloudFormation stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 19:31:32 [I] waiting for CloudFormation stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 19:31:32 [I] waiting for the control plane to become ready
2024-11-22 19:31:34 [V] saved kubeconfig as "/home/ec2-user/.kube/config"
2024-11-22 19:31:34 [I] no tasks
2024-11-22 19:31:34 [V] all EKS cluster resources for "kubernetes-cluster" have been created
2024-11-22 19:31:34 [V] created 0 nodegroup(s) in cluster "kubernetes-cluster"
2024-11-22 19:31:34 [I] nodegroup "linux-node" has 2 node(s)
2024-11-22 19:31:34 [I] node "ip-192-168-63-199.us-east-2.compute.internal" is ready
2024-11-22 19:31:34 [I] node "ip-192-168-69-198.us-east-2.compute.internal" is ready
2024-11-22 19:31:34 [I] waiting for at least 2 node(s) to become ready in "linux-node"
2024-11-22 19:31:34 [I] nodegroup "linux-node" has 2 node(s)
2024-11-22 19:31:34 [I] node "ip-192-168-63-199.us-east-2.compute.internal" is ready
2024-11-22 19:31:34 [I] node "ip-192-168-69-198.us-east-2.compute.internal" is ready
2024-11-22 19:31:35 [V] created 1 managed nodegroup(s) in cluster "kubernetes-cluster"
2024-11-22 19:31:35 [I] kubectl command should work with "/home/ec2-user/.kube/config", try 'kubectl get nodes'
2024-11-22 19:31:35 [V] EKS cluster "kubernetes-cluster" in "us-east-2" region is ready
[ec2-user@ip-172-31-15-191 ~]$
```

2. Go the “/home/ec2-user/.kube/config” and view the file

```

2024-11-22 19:31:35 [i] kubectl command should work with "/home/ec2-user/.kube/config", try 'kubectl get nodes'
2024-11-22 19:31:35 [v] EKS cluster "kubernetes-cluster" in "us-east-2" region is ready
[ec2-user@ip-172-31-15-191 ~]$ cd /home/ec2-user/.kube/config
[bash: cd: /home/ec2-user/.kube/config: Not a directory]
[ec2-user@ip-172-31-15-191 ~]$ cat /home/ec2-user/.kube/config
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdjTibDRVJSUZJQOFURS0tLS0tCKlJSURCVENDQWUy20F3SUJBZ01J%jLSY2dMUFovYTBJRFFZSkvWklodmNQQVEETEJRQX4GVEVUTUJFROExVUUKQXhNS2EzVm
aWEpI1lhSbGN6QWVCDz85TkrF8rlqSxhPvUyTURKyUz3MPrOEV47WpBcEUSXhNRephTUJvEApfekF5QmdVKEJBTvRDxQXhW1WwJtVjbawE_324fU1BMsdu3FHU01lM0REBRUJSUVPQ0TRJQkR3QXdn20VLCKrVgU
UBUURFS2JRCjJQ19NWd4mtTg013M5bUVMWJTr0W-6TEFocGJDMOUwPFpk1Vm7dtaU1K0hLZS9kRMnLd3XKwNlJ0Xh2aTlRzWVNUddtU0M5cVpaay9WUVRzSVJzRmk1V0d1r2zrCmDThGEV0J0FKOUkEKO1mVsWUzgymMz1s
wp1UVLz184enkrl0H2W56WUh3Vkf5b2UuQ3U2UmR0em9obId0b04EM11jN0FvG1lMml1ksTltd1lwC1DW58Ck0rYTTyNgZp12JRRlt55230Vm9UW7Fu1Nvc22zrSHVMVtg4albhpeEdxeEVFV24MM3BFskRc2hF0T1t
VmhwebxVm91ZEEzJYp60C2Mnrcf0Rghq2FFfa2qjx12q2zDRVNmU4TmVtGcs20DU5QjNGVnhCWTVtv1zcz23RWVVFtL7cpXb9kogpzaRcvU0YeikhNyJz1ckZEYmabzJWWvdnqkFB2pXVEJYTTbE0R0ExVWREd0VcL3d
RROf3UNWREpQCKJn1l1z1UK1C0WY4RUJUQRURUdgV01WR08xVRE21PxQkJUVG1z0FpwM1dyd21ln019F20F1SVFxcWNpd1kU0VYK0mdOVkhISUW8akFNz2dcmRXSmxJbV52Edw1kBMdgDU3FHU01M0RRR0Ud1lVb9QT
RJQkFRQ22M0pBSggfApj2ic0NjNRTxpuXhW3htdn2fCW28MDxVSkSvccodnejBlAgcx1GtEqV1vMw021grnxXpqll1PbEWYUxtQn4CjRWY55cnVch0UNy2fobGxq202H0Ma3puMng4eirOBWJhN0prKzJxu5rbhdhsXp2h1segpFdhrTtV3d1EWmlq1K1mcstkcOpZWBlieHwbllicGKRT2f23K3
aHdsQkZ0d0dRxUVd4WzTuktNc01BY2t0Ch0b1zu0j14Yndus25RTd11U2kWmXVzXKR1BeXNu0ajfYznyrQldIejdrkTqkRfa0hHVUxmRFNzV04WJMRUhpWTGdSSDEyaGxxci0tLS0tRU5E1ENFU1RJRKdQVRLFLSo
TLSOK
server: https://BD19DBD2023E6736C067A1710E6B9F8.y14.us-east-2.eks.amazonaws.com
name: kubernetes-cluster.us-east-2.eksctl.io
contexts:
- context:
    cluster: kubernetes-cluster.us-east-2.eksctl.io
    user: i-0461426356c36c3ce@kubernetes-cluster.us-east-2.eksctl.io
    name: i-0461426356c36c3ce@kubernetes-cluster.us-east-2.eksctl.io
current-context: i-0461426356c36c3ce@kubernetes-cluster.us-east-2.eksctl.io
kind: Config

```

**i-0461426356c36c3ce (Jenkins)**

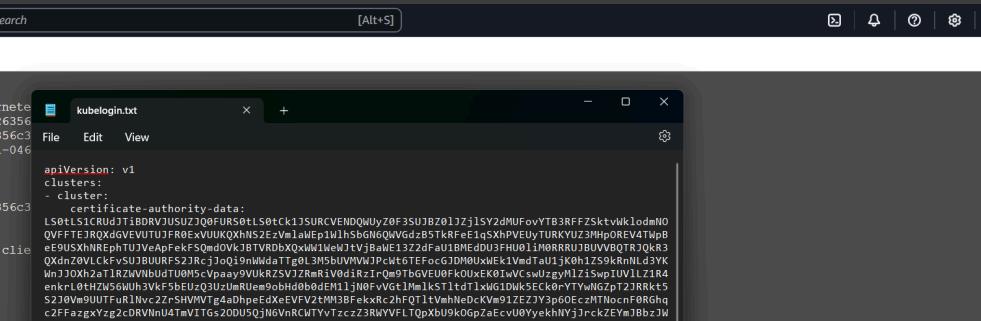
PublicIPs: 3.135.220.22 PrivateIPs: 172.31.15.191

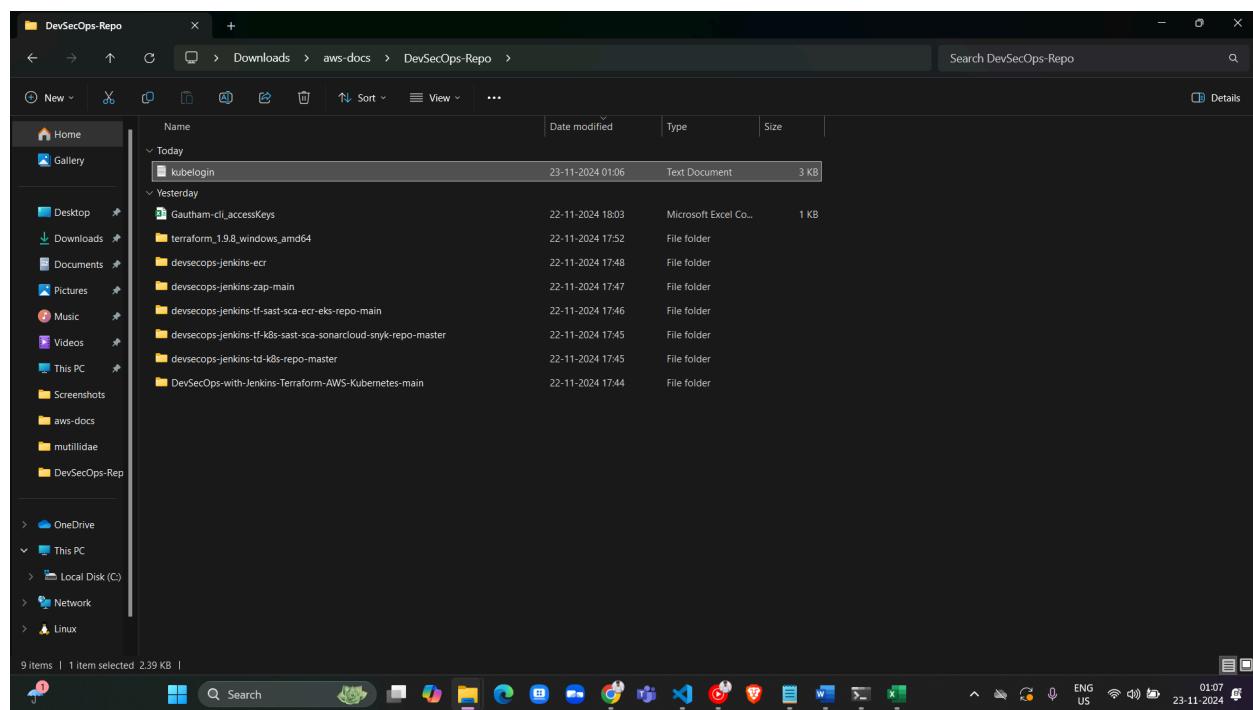


## 2. Configure Jenkins with Kubeconfig

1. Copy the kubeconfig instance content to the “kubelogin” file in the local machine.

```
contexts:
- context:
  cluster: kubernetes
  user: i-0461426356
  name: i-0461426356c3
current-context: i-0461426356c3
kind: config
preferences: {}
users:
- name: i-0461426356c3
  user:
    exec:
      apiVersion: cli
      args:
      - eks
    - get-token
    - --output
    - json
    - --cluster-name
    - kubernetes-clu
    - --region
    - us-east-2
    command: aws
    env:
    - name: AWS_STS
      value: regional
      providerclusterIn
[ec2-user@ip-172-31-15 ~]
```





2. Add it as a secret file in Jenkins credentials.

The screenshot shows the Jenkins 'New credentials' interface. The 'Kind' dropdown is set to 'Secret file'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'File' section shows a chosen file named 'kubelogin.txt'. The 'ID' field contains 'kubelogin'. The 'Description' field also contains 'kubelogin'. A blue 'Create' button is at the bottom.

### 3. Deploy to Kubernetes

1. Create a namespace using “`kubectl create namespace devsecops`”

```
cluster: kubernetes-cluster.us-east-2.eksctl.io
user: i-0461426356c36c3ce@kubernetes-cluster.us-east-2.eksctl.io
name: i-0461426356c36c3ce@kubernetes-cluster.us-east-2.eksctl.io
current-context: i-0461426356c36c3ce@kubernetes-cluster.us-east-2.eksctl.io
kind: Config
preferences: {}
users:
- name: i-0461426356c36c3ce@kubernetes-cluster.us-east-2.eksctl.io
  user:
    exec:
      apiVersion: client.authentication.k8s.io/v1beta1
      args:
        - eks
        - get-token
        - --output
        - json
        - --cluster-name
        - kubernetes-cluster
        - --region
        - us-east-2
      command: aws
      env:
        - name: AWS_STS_REGIONAL_ENDPOINTS
          value: regional
        - name: AWS_EKS_CLUSTER_NAME
          value: kubernetes-cluster
      provideClusterInfo: false
[ec2-user@ip-172-31-15-191 ~]$ kubectl create namespace devsecops
namespace/devsecops created
[ec2-user@ip-172-31-15-191 ~]$
```

i-0461426356c36c3ce (Jenkins)

Public IPs: 3.135.220.22 Private IPs: 172.31.15.191



2. Update the `deployment.yaml` file with the ECR image URI.

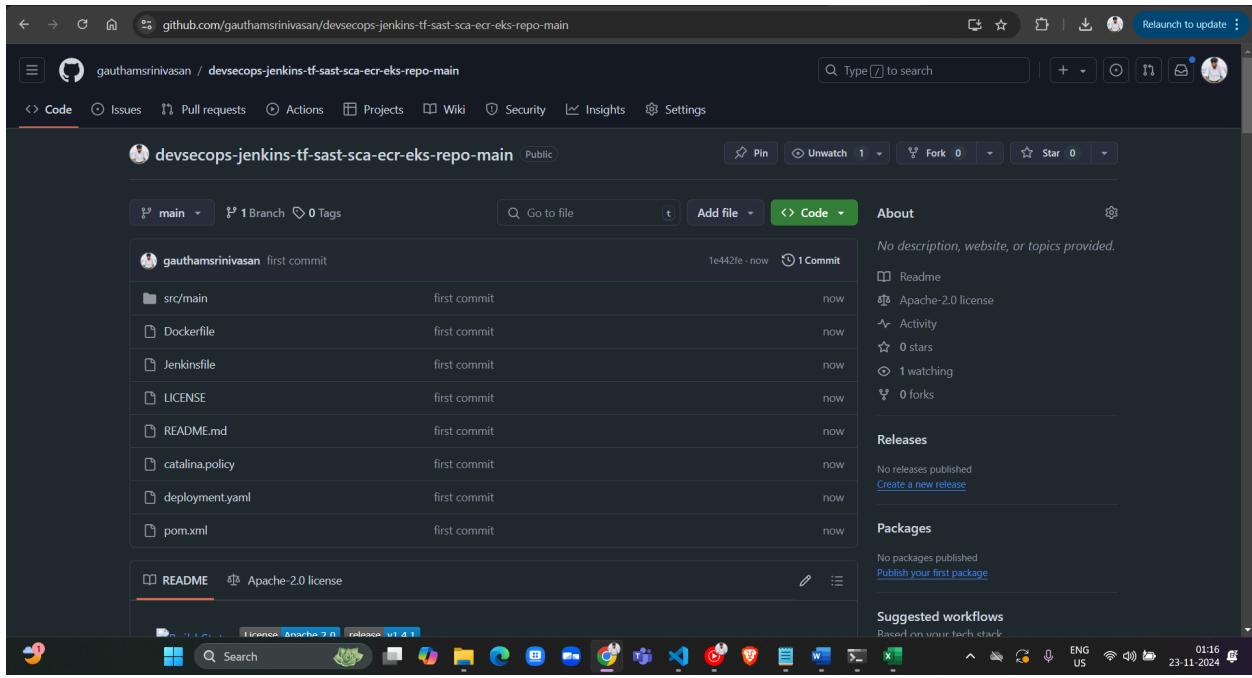
The screenshot shows the Jenkinsfile in the VS Code editor. The file contains Groovy code defining a pipeline with stages for building, pushing to ECR, and deploying to Kubernetes. The Jenkinsfile is located in a folder named 'DEVESECOPS-JENKINS-TF-SAST-SCA-ECR-E...'.

```
1 Jenkinsfile
  2 pipeline {
  3     stage('Build') {
  4         steps {
  5             withDockerRegistry([credentialsId: 'dockerlogin', url: ""])
  6                 script{
  7                     app = docker.build("myimg")
  8                 }
  9             }
 10         }
 11         stage('Push') {
 12             steps {
 13                 script{
 14                     docker.withRegistry('https://116981766729.dkr.ecr.us-east-2.amazonaws.com/myimg', 'ecr:us-east-2:aws-credentials') {
 15                         app.push("latest")
 16                     }
 17                 }
 18             }
 19         }
 20     stage('Kubernetes Deployment of Gautham Web Application') {
 21         steps {
 22             withKubeConfig([credentialsId: 'kubelogin']) {
 23                 sh('kubectl delete all --all -n devsecops')
 24                 sh('kubectl apply -f deployment.yaml --namespace=devsecops')
 25             }
 26         }
 27     }
 28 }
 29 }
```

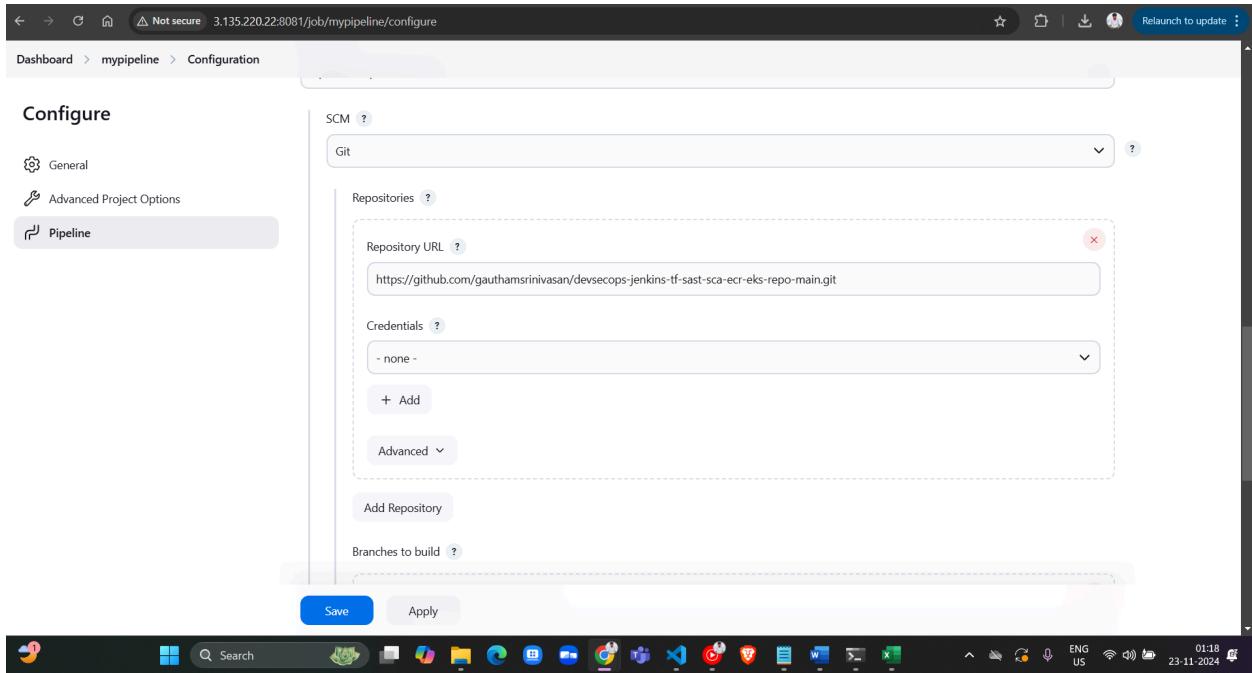
The screenshot shows the deployment.yaml file in the VS Code editor. The file defines a Kubernetes deployment for an application named 'easybuggy'. It specifies one replica, selects pods with the label 'app: easybuggy', and uses a Docker image from the AWS ECR repository. The deployment.yaml file is part of a commit message.

```
1 deployment.yaml
  2 metadata:
  3     name: easybuggy-deployment
  4 spec:
  5     replicas: 1
  6     selector:
  7         matchLabels:
  8             app: easybuggy
  9     template:
 10         metadata:
 11             labels:
 12                 app: easybuggy
 13         spec:
 14             containers:
 15                 - name: easybuggy
 16                   image: 116981766729.dkr.ecr.us-east-2.amazonaws.com/myimg
 17                   imagePullPolicy: Always
 18             ports:
 19                 - containerPort: 8080
 20             # service type loadbalancer
 21             ...
 22             apiVersion: v1
 23             kind: Service
 24             metadata:
 25                 labels:
 26                     app: easybuggy
 27                     k8s-app: easybuggy
 28                     name: easybuggy
 29             spec:
 30                 ports:
 31                     - name: http
 32                     port: 80
 33                     protocol: TCP
 34                     targetPort: 8080
 35                     type: LoadBalancer
 36             selector:
 37                 app: easybuggy
 38 }
```

### 3. Push the folder to the github and verify



#### 4. Change the jenkins github url



#### 5. Build now and verify the output

## 6. Go to the Instance verify with the command “kubectl get deployments -n devsecops”

```

current-context: i-0461426356c36c3ce@kubernetes-cluster.us-east-2.eksctl.io
kind: Config
preferences: {}
users:
- name: i-0461426356c36c3ce@kubernetes-cluster.us-east-2.eksctl.io
  user:
    exec:
      apiVersion: client.authentication.k8s.io/v1beta1
      args:
      - eks
      - get-token
      - --output
      - json
      - --cluster-name
      - kubernetes-cluster
      - --region
      - us-east-2
      command: aws
      env:
      - name: AWS_STS_REGIONAL_ENDPOINTS
        value: regional
      provideClusterInfo: false
[ec2-user@ip-172-31-15-191 ~]$ kubectl create namespace devsecops
namespace/devsecops created
[ec2-user@ip-172-31-15-191 ~]$ kubectl get deployments -n devsecops
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
easybuggy-deployment   0/1     1           0          2m22s
[ec2-user@ip-172-31-15-191 ~]$ 
```

i-0461426356c36c3ce (Jenkins)

Public IPs: 3.135.220.22 Private IPs: 172.31.15.191

## 7. Check the service of the deployment using the service “kubectl get svc -n devsecops”

```

- us-east-2
  command: aws
  env:
    - name: AWS_STS_REGIONAL_ENDPOINTS
      value: regional
      provideClusterInfo: false
[ec2-user@ip-172-31-15-191 ~]$ kubectl create namespace devsecops
namespace/devsecops created
[ec2-user@ip-172-31-15-191 ~]$ kubectl get deployments -n devsecops
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
easybuggy-deployment   0/1     1           0          2m22s
[ec2-user@ip-172-31-15-191 ~]$ kubectl get svc -n devsecops
NAME            TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
easybuggy       LoadBalancer   10.100.28.224   af7268c0e8c12450eb5cdff0ab5d7aa3-1838981700.us-east-2.elb.amazonaws.com   80:31051/TCP   2m38s
[ec2-user@ip-172-31-15-191 ~]$ ^C
[ec2-user@ip-172-31-15-191 ~]$ kubectl get deployments -n devsecops
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
easybuggy-deployment   0/1     1           0          7m28s
[ec2-user@ip-172-31-15-191 ~]$ kubectl get svc -n devsecops
NAME            TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
easybuggy       LoadBalancer   10.100.28.224   af7268c0e8c12450eb5cdff0ab5d7aa3-1838981700.us-east-2.elb.amazonaws.com   80:31051/TCP   7m41s
[ec2-user@ip-172-31-15-191 ~]$ kubectl get svc -n devsecops
NAME            TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
easybuggy       LoadBalancer   10.100.174.231   aa0df6d34811c46a9bf4e5ac608050e10-1105435064.us-east-2.elb.amazonaws.com   80:31523/TCP   29s
[ec2-user@ip-172-31-15-191 ~]$ kubectl get svc -n devsecops
NAME            TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
easybuggy       LoadBalancer   10.100.236.67   ad65b6ca2472d41f69f3373f658f6838-1486206385.us-east-2.elb.amazonaws.com   80:31632/TCP   11s
[ec2-user@ip-172-31-15-191 ~]$ 

```

i-0461426356c36c3ce (Jenkins)

Public IPs: 3.135.220.22 Private IPs: 172.31.15.191

The screenshot shows the Jenkins dashboard for the Jenkins instance running on the specified IP. The dashboard includes a summary of the Jenkins environment, a list of available jobs, and various monitoring and configuration links.

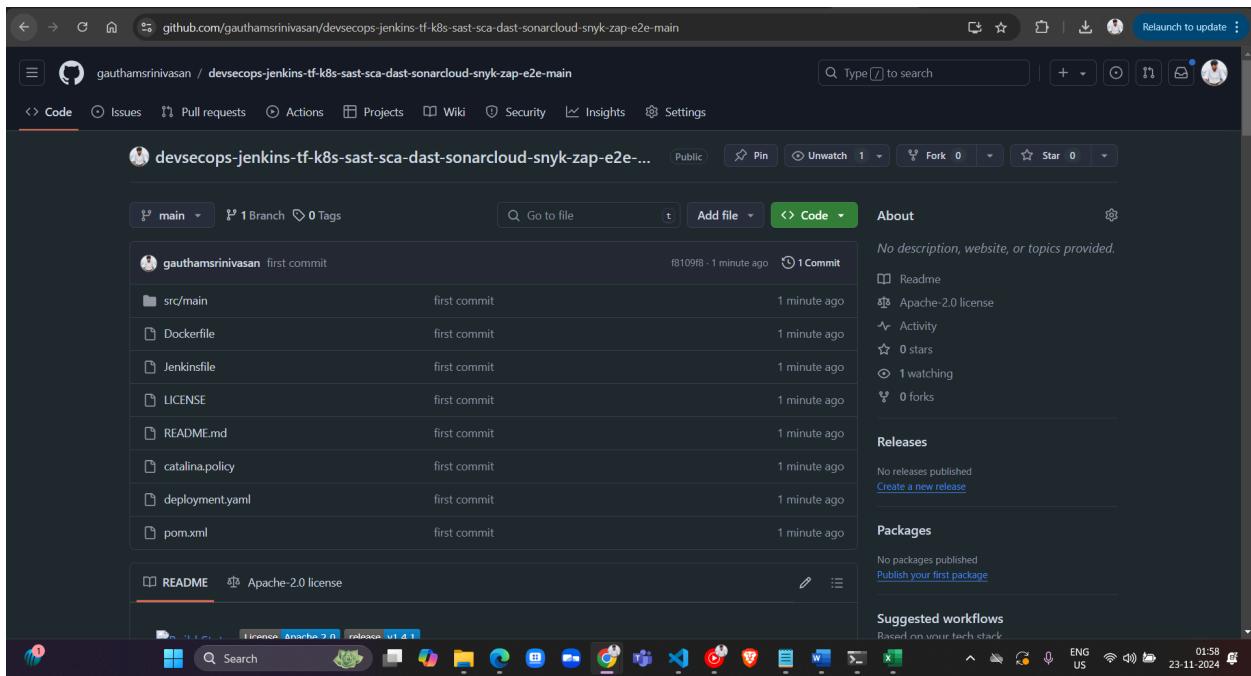
## Step 9. DAST with OWASP ZAP

- Add an OWASP ZAP stage to the pipeline.

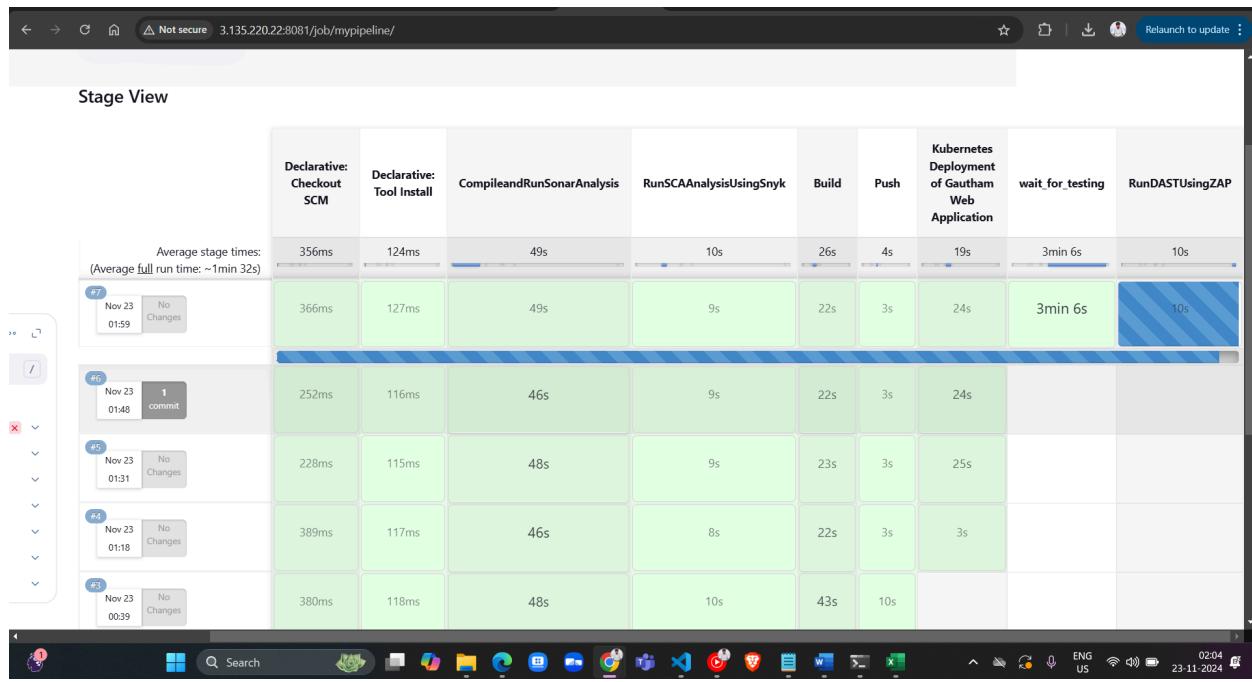
A screenshot of a code editor window titled "Jenkinsfile". The file contains Groovy script for a Jenkins pipeline. The pipeline defines stages for Kubernetes deployment, waiting for testing, and running DAST using Zap. The code uses Jenkins Pipeline syntax with steps like `sh` for shell commands and `withKubeConfig` for Kubernetes authentication.

```
pipeline {
    tools {
        stages {
            stage('Push') {
                stage('Kubernetes Deployment of Gautham Web Application') {
                    steps {
                        withKubeConfig([credentialsId: 'kubelogin']) {
                            sh('kubectl delete all --all -n devsecops')
                            sh ('kubectl apply -f deployment.yaml --namespace=devsecops')
                        }
                    }
                }
                stage ('wait_for_testing'){
                    steps {
                        sh 'pwd; sleep 180; echo "Application has been deployed on K8S"'
                    }
                }
                stage ('RunDASTUsingZAP'){
                    steps {
                        withKubeConfig([credentialsId: 'kubelogin']){
                            sh("zap.sh -cmd quickrun http://$kubeconfig get services/easybuggy --namespace=devsecops -o json| jq -r '.status.loadBalancer.ingress[0].ip' > zap_report.html")
                            archiveArtifacts artifacts: 'zap_report.html'
                        }
                    }
                }
            }
        }
    }
}
```

## 2. Publish your folder to the repo and verify



## 3. Trigger the pipeline to generate a report



The screenshot shows a web browser window with the URL <https://3.135.220.22:8081/job/mypipeline/7/console>. The page displays a progress bar for a task named "Active scanning". The progress bar consists of a series of brackets and percentage symbols, indicating a completion rate of 100%. The browser interface includes a top navigation bar with back, forward, search, and download icons, as well as a status bar at the bottom showing network connectivity and system information.

```
Dashboard > mypipeline > #7

[=====] 92% /
[=====] 92% \
[=====] 93% /
[=====] 97% -
[=====] 100%
Attack complete
Writing results to /var/lib/jenkins/workspace/mypipeline/zap_report.html
[Pipeline] archiveArtifacts
Archiving artifacts
[Pipeline] }
[kubernetes-cli] kubectl configuration cleaned up
[Pipeline] // withKubeConfig
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

**ZAP Report**

Site: <http://a0c70e93bc3d94ab2af5cb56bd0d9c1a-1697270124.us-east-2.elb.amazonaws.com>

Generated on Fri, 22 Nov 2024 20:42:40

ZAP Version: 2.14.0

### Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	3
Low	4
Informational	1
False Positives:	0

### Alerts

Name	Risk Level	Number of Instances
<a href="#">Absence of Anti-CSRF Tokens</a>	Medium	16
<a href="#">Content Security Policy (CSP) Header Not Set</a>	Medium	45
<a href="#">Session ID in URL Rewrite</a>	Medium	1
<a href="#">Application Error Disclosure</a>	Low	24
<a href="#">Cookie without SameSite Attribute</a>	Low	9
<a href="#">Server Leaks Version Information via "Server" HTTP Response Header Field</a>	Low	54
<a href="#">Timestamp Disclosure - Unix</a>	Low	8
<a href="#">Session Management Response Identified</a>	Informational	13

### Alert Detail

**Absence of Anti-CSRF Tokens**

This control includes a component for CSRF.

Do not use the GET method for any request that triggers a state change.

Phase: Implementation

Check the HTTP Referer header to see if the request originated from an expected page. This could break legitimate functionality, because users or proxies may have disabled sending the Referer for privacy reasons.

Reference: <http://projекты.webappsec.org/Cross-Site-Request-Forgery>  
<https://cwe.mitre.org/data/definitions/352.html>

CWE Id: 352  
WASC Id: 9  
Plugin Id: 10202

**Content Security Policy (CSP) Header Not Set**

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.

URL: <http://a0c70e93bc3d94ab2af5cb56bd0d9c1a-1697270124.us-east-2.elb.amazonaws.com>  
Method: GET

URL: <http://a0c70e93bc3d94ab2af5cb56bd0d9c1a-1697270124.us-east-2.elb.amazonaws.com/ae>  
Method: GET

URL: <http://a0c70e93bc3d94ab2af5cb56bd0d9c1a-1697270124.us-east-2.elb.amazonaws.com/nioobe>  
Method: GET

## Step 10. Integrate Snyk with Jira to Assign Tickets for Vulnerabilities

1. Go to Snyk Console > Settings > Plans and pricing > Enable the Free trial for 14 days
2. Navigate to Integrations > Jira and enter the base url of your project board.

The screenshot shows the Snyk integration page for the organization 'gauthamsrinivasan'. On the left, there's a sidebar with options like Dashboard, Projects, Reports, Dependencies, Integrations (which is selected), Members, and Settings. The main area displays several integration cards:

- PHPStorm
- PyCharm
- Rider
- RubyMine
- WebStorm

Below these are sections for 'Gatekeeper plugins' (Artifactory Plugin, Nexus) and 'Notifications' (Slack, Slack App, Jira). A 'View Documentation' link is also present.

The screenshot shows the 'Jira Settings' page under the 'Settings' section of the Snyk interface. The left sidebar includes 'General', 'Integrations', 'Authorized Snyk Apps', 'Snyk Open Source', 'Snyk Code', 'Snyk IaC', 'Usage', 'Notifications', 'Snyk Preview', 'DeepCode AI Fix', 'Integrations settings', and 'GitHub'. The right side shows the 'Jira Settings' configuration form:

**Jira Settings**  
To use Snyk with Jira, you'll need to provide some configuration. See our [Jira integration documentation](#) for more details.

**Base URL**  
e.g.

**Username/email**

**API token**

**Save and continue**

For installation of Jira within a private network click [here](#).

3. Go to Jira > Profile > Security > Create Tokens and copy it.

The screenshot shows the 'API Tokens' page from the Atlassian account settings. A single API token named 'snyk' is listed, created on Nov 23, 2024, and last accessed 21 minutes ago. There is a 'Revoke' button next to it.

4. Verify your connection and jira project and select the type and click finish

The screenshot shows the 'Jira Settings' page in the Snyk application. It displays the 'Jira Integration Url' as <https://cloudops-1.atlassian.net>. Other settings include 'Ignored Fields' (e.g. components, security, customfield\_10006), 'Default Project' (Cloud Demons), and 'Default Issue Type' (Task). A 'Finish' button is at the bottom. Below it is a 'Disconnect from Jira' section with a 'Disconnect' button.

5. Go to your project in snyk and select a vulnerability and a new button “**Create a Jira Issue**” will be enabled

The screenshot shows the Snyk interface for analyzing a Dockerfile. The left sidebar shows the user's organization and project details. The main content area displays a summary of the analysis, including priority score, fixed status, and exploit maturity. A specific vulnerability for zlib/zlib1g is highlighted, categorized as CRITICAL with a score of 714. The vulnerability details include its introduction through the package, fixed versions, and no known exploits.

**PRIORITY SCORE**  
Scored between 0 - 1000

**"FIXED IN" AVAILABLE**

Yes	58
No	50

**COMPUTED FIXABILITY**

Fixable	0
Partially fixable	0
No supported fix	108

**EXPLOIT MATURITY**

Mature	1
Proof of concept	0
No known exploit	107
No data	0

**VULNERABILITY** zlib/zlib1g - Out-of-bounds Write ⚠  
CWE-787 | CVE-2022-37434 | CVSS 9.8 | CRITICAL | SNYK-DEBIAN11-ZLIB-2976151

**SCORE** 714

**INTRODUCED THROUGH** zlib/zlib1g@1:1.2.11.dfsg-2+deb11u1  
**FIXED IN** zlib/zlib1g@1:1.2.11.dfsg-2+deb11u2, @1:1.2.11.dfsg-2+deb11u2

**EXPLOIT MATURITY** NO KNOWN EXPLOIT

**Ignore** **Create a Jira issue**

6. Assign the jira issue to your team member and flag the issue type and click create

The screenshot shows the Snyk interface for a project named "gauthamsrinivasan/devsecops-jenkins-tf-k8s-sast-sca-dast-sonarcloud-snyk-zap-e2e-main". A modal window titled "Create a Jira Issue" is open. In the "Select issue type" dropdown, "Task" is selected. The "Reporter" field contains "Gautham Srinivasan". The "Assignee (optional)" dropdown shows a list of users: Gautham Srinivasan, Nandini, Priyadarshini R, and sanjushrivarshini. The "Description (optional)" text area contains the following text:

[Vulnerability in gauthamsrinivasan/devsecops-jenkins-tf-k8s-sast-sca-dast-sonarcloud-snyk-zap-e2e-]

The screenshot shows the same Snyk interface and Jira issue creation dialog. The "Summary" field now contains "gauthamsrinivasan/devsecops-jenkins-tf-k8s-sast-sca-dast-sonarcloud-snyk-". The "Description (optional)" text area contains the following text:

[More about this issue: https://app.snyk.io/vuln/SNYK-DEBIAN11-ZLIB-2976151]

[Vulnerability in gauthamsrinivasan/devsecops-jenkins-tf-k8s-sast-sca-dast-sonarcloud-snyk-zap-e2e-]  
main Dockerfile [https://app.snyk.io/org/gauthamsrinivasan/project/e902e447-0db2-4752-aafa-add4d279110#issue-SNYK-DEBIAN11-ZLIB-2976151]

Introduced through: zlib/zlib1g

The screenshot shows the Snyk interface for a Dockerfile project. On the left, there's a sidebar with navigation links like Dashboard, Projects (which is selected), Reports, Dependencies, Integrations, Members, and Settings. Below that are Product updates, Help, and a user profile for Gautham S. The main content area shows a 'Create a Jira Issue' modal. The modal has fields for Development (optional) with 'Immediate effect required' checked, Team (optional) empty, Labels (optional) with 'BUG' entered, and Rank (optional) empty. At the bottom is a 'Create Jira issue' button. In the background, there's a dark panel with a 'SCORE 714' and a 'NO KNOWN EXPLOIT' message.

7. Verify the jira issue is assigned in your jira board console

The screenshot shows the Jira interface for the 'Cloud Demons' project. On the left, the sidebar includes sections for PLANNING (Timeline, Board, List, Forms), DEVELOPMENT (Code, Project pages, Project settings, Archived issues), and a navigation bar with 'Your work', 'Projects', 'Filters', 'Dashboards', 'Teams', 'Plans', 'Apps', and a 'Create' button. The main area displays a 'CD board' with three columns: 'TO DO', 'IN PROGRESS', and 'DONE'. A card for 'CD-9' is visible in the 'IN PROGRESS' column, labeled 'BUG'. A tooltip for this card provides the detailed description: 'gauthamsrinivasan/devsecops-jenkins-tf-k8s-sast-sca-dast-sonarcloud-snky-zap-e2e-main:Dockerfile - Out-of-bounds Write in zlib/zlib1g'. A 'Show me' button is present in the top right of the board area.

This screenshot shows the detailed view of the Jira ticket for 'CD-9'. The ticket title is 'gauthamsrinivasan/devsecops-jenkins-tf-k8s-sast-sca-dast-sonarcloud-snky-zap-e2e-main:Dockerfile - Out-of-bounds Write in zlib/zlib1g'. The 'Details' panel on the right lists the assignee as 'Gautham Srinivasan', the label as 'BUG', and the reporter as 'Gautham Srinivasan'. It also shows the development status with options to 'Create branch' and 'Create commit'. The ticket description includes a link to 'More about this issue' and a 'Snyk' section. The bottom of the screen shows a Windows taskbar with various application icons.

8. Navigate the snyk issue from the jira issued ticket by clicking the embedded link

The screenshot shows the Snyk interface for a project named 'gauthamsrinivasan/devsecops-jenkins-tf-k8s-sast-sca-dast-sonarcloud-snyk-zap-e2e-main'. The main view is for the 'Dockerfile' file. A prominent red box highlights a critical vulnerability:

- CVE-2022-37434**: zlib/zlib1g - Out-of-bounds Write (CWE-787)
- CVSS 9.8**
- CRITICAL**
- SNYK-DEBIAN11-ZLIB-2976151**
- Score: 714**

The interface includes a sidebar with navigation links like 'Dashboard', 'Projects', 'Reports', 'Dependencies', 'Integrations', 'Members', and 'Settings'. The 'Projects' link is highlighted. The 'Overview' tab is selected. On the right, there are filters for priority (High, Medium, Low) and exploit maturity (No known exploit). Buttons for 'Ignore', 'Jira CD-9', and 'Create a Jira issue' are also present.

## Step 11. Cleanup

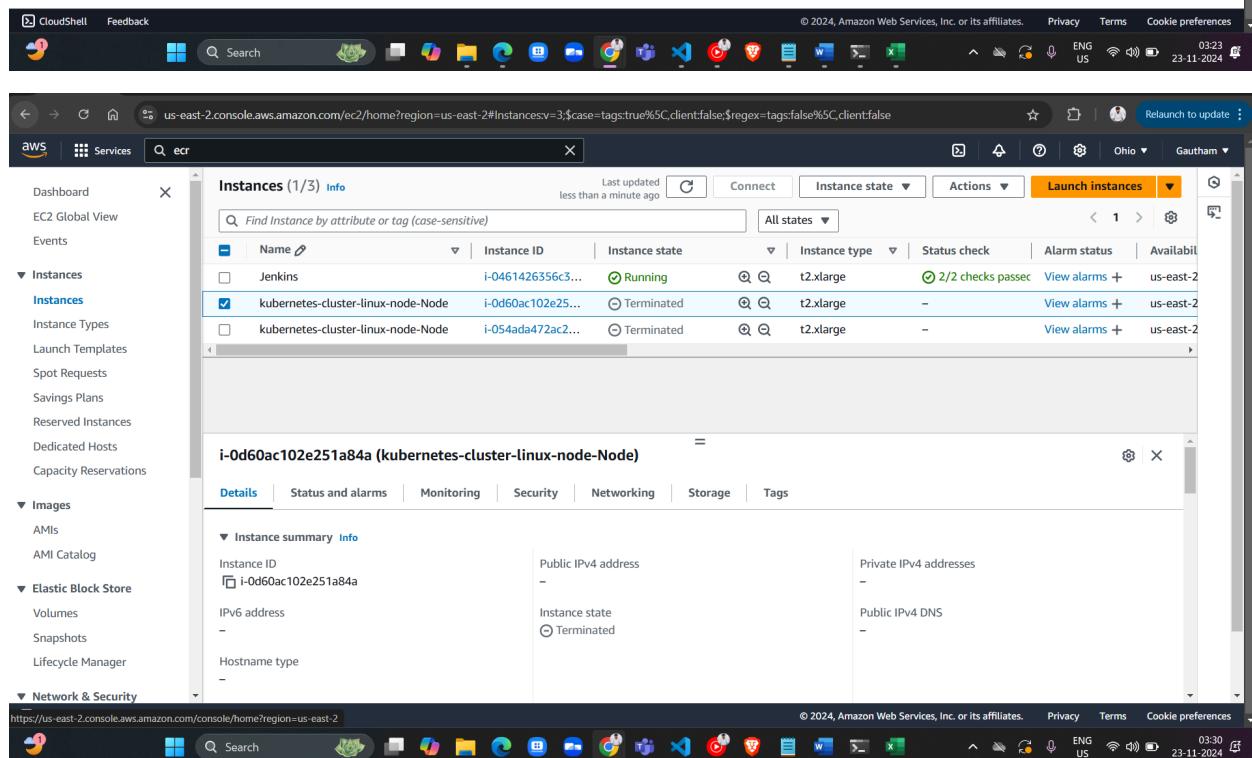
1. Delete the EKS cluster:

```
eksctl delete cluster --name kubernetes-cluster --region us-east-2
```

```
[ec2-user@ip-172-31-15-191 ~]$ eksctl delete cluster --name kubernetes-cluster --region us-east-2
2024-11-22 21:53:07 [i] deleting EKS cluster "kubernetes-cluster"
2024-11-22 21:53:07 [i] will drain 0 unmanaged nodegroup(s) in cluster "kubernetes-cluster"
2024-11-22 21:53:07 [i] starting parallel draining, max in-flight of 1
2024-11-22 21:53:07 [X] failed to acquire semaphore while waiting for all routines to finish: context canceled
2024-11-22 21:53:07 [i] deleted 0 Fargate profile(s)
2024-11-22 21:53:07 [V] kubeconfig has been updated
2024-11-22 21:53:07 [i] cleaning up AWS load balancers created by Kubernetes objects of Kind Service or Ingress
```

i-0461426356c36c3ce (Jenkins)

Public IPs: 3.135.220.22 Private IPs: 172.31.15.191



```
[ec2-user@ip-172-31-15-191 ~]$ eksctl delete cluster --name kubernetes-cluster --region us-east-2
2024-11-22 21:53:07 [i] deleting EKS cluster "kubernetes-cluster"
2024-11-22 21:53:07 [i] will drain 0 unmanaged nodegroup(s) in cluster "kubernetes-cluster"
2024-11-22 21:53:07 [i] starting parallel draining, max in-flight of 1
2024-11-22 21:53:07 [x] failed to acquire semaphore while waiting for all routines to finish: context canceled
2024-11-22 21:53:07 [i] deleted 0 Fargate profile(s)
2024-11-22 21:53:07 [v] kubeconfig has been updated
2024-11-22 21:53:07 [i] cleaning up AWS load balancers created by Kubernetes objects of Kind Service or Ingress
2024-11-22 21:53:33 [i]
2 sequential tasks: { delete nodegroup "linux-node", delete cluster control plane "kubernetes-cluster" [async]
}
2024-11-22 21:53:33 [i] will delete stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 21:53:33 [i] waiting for stack "eksctl-kubernetes-cluster-nodegroup-linux-node" to get deleted
2024-11-22 21:53:33 [i] waiting for CloudFormation stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 21:54:03 [i] waiting for CloudFormation stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 21:54:36 [i] waiting for CloudFormation stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 21:56:07 [i] waiting for CloudFormation stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 21:57:25 [i] waiting for CloudFormation stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 21:59:01 [i] waiting for CloudFormation stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 22:00:16 [i] waiting for CloudFormation stack "eksctl-kubernetes-cluster-nodegroup-linux-node"
2024-11-22 22:00:16 [i] will delete stack "eksctl-kubernetes-cluster-cluster"
2024-11-22 22:00:16 [v] all cluster resources were deleted
[ec2-user@ip-172-31-15-191 ~]$
```

i-0461426356c36c3ce (Jenkins)  
PublicIPs: 3.135.220.22 PrivateIPs: 172.31.15.191

2. Remove other resources as required by using  
`-var-file="vars/dev-east-2.tfvars"`

```

C:\Users\gauth\Downloads\aws-docs\DevSecOps-Repo\DevSecOps-with-Jenkins-Terraform-AWS-Kubernetes-main\DevSecOps-with-Jenkins-Terraform-AWS-Kubernetes-main> terraform destroy -var-file="vars/dev-east-2.tfvars"
data.aws_ami.amazon_linux: Reading...
aws_iam_role.devsecops_role: Refreshing state... [id=devsecops_role]
aws_security_group.jenkins_sg: Refreshing state... [id=sg-05598c027977005fa]
data.aws_ami.amazon_linux: Read complete after 2s [id=ami-09da212cf18033880]
aws_iam_instance_profile.ec2devsecops_profile: Refreshing state... [id=ec2devsecops_profile]
aws_iam_role_policy.devsecops_policy: Refreshing state... [id=devsecops_role:devsecops_policy]
aws_instance.web: Refreshing state... [id=i-0461426356c36c3ce]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

# aws_iam_instance_profile.ec2devsecops_profile will be destroyed
- resource "aws_iam_instance_profile" "ec2devsecops_profile" {
    - arn      = "arn:aws:iam::116981766729:instance-profile/ec2devsecops_profile" -> null
    - create_date = "2024-11-22T17:43:42Z" -> null
    - id       = "ec2devsecops_profile" -> null
    - name     = "ec2devsecops_profile" -> null
    - path     = "/" -> null
    - role     = "devsecops_role" -> null
    - tags     = "{}" -> null
    - tags_all = "{}" -> null
    - unique_id = "AIPARWPFIJDJE6V15KPVMN" -> null
    # (1 unchanged attribute hidden)
}

# aws_iam_role.devsecops_role will be destroyed
- resource "aws_iam_role" "devsecops_role" {
    - arn      = "arn:aws:iam::116981766729:role/devsecops_role" -> null
    - assume_role_policy = jsonencode(
        {
            Statement = [
                {
                    Action      = "sts:AssumeRole"
                    Effect     = "Allow"
                    Principal  = {
                        Service = "ec2.amazonaws.com"
                    }
                }
            ]
        }
    )
}

```

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Available
Jenkins	i-0461426356c3...	Terminated	t2.xlarge	-	<a href="#">View alarms</a> +	us-east-2
<b>kubernetes-cluster-linux-node-Node</b>	<b>i-0d60ac102e25...</b>	<b>Terminated</b>	<b>t2.xlarge</b>	<b>-</b>	<a href="#">View alarms</a> +	<b>us-east-2</b>
kubernetes-cluster-linux-node-Node	i-054ada472ac2...	Terminated	t2.xlarge	-	<a href="#">View alarms</a> +	us-east-2

- Delete the Registry

The screenshot shows the AWS ECR (Amazon Elastic Container Registry) console. On the left, there's a navigation sidebar with options like 'Amazon Elastic Container Registry', 'Private registry', 'Public registry', and links to 'Getting started' and 'Documentation'. The main area is titled 'Private repositories' and shows a table with one repository entry: 'myimg'. A modal dialog box is overlaid on the page, prompting the user to confirm the deletion of the repository. The dialog contains the text: 'Delete myimg', 'Are you sure you want to delete myimg? This will also delete any container images and artifacts in this repository.', and two buttons: 'Cancel' and 'Delete'.

- Delete the created user

The screenshot shows the AWS IAM (Identity and Access Management) console. The left sidebar includes sections for 'Identity and Access Management (IAM)', 'Access management' (with 'Users' selected), and 'Access reports'. The main area displays a table for 'Users (1/1)' with one user listed: 'Gautham-cli'. A modal dialog box is overlaid, prompting the user to delete the user 'Gautham-cli'. The dialog includes the text: 'Delete Gautham-cli?', 'Delete Gautham-cli permanently? This will also delete all its user data, security credentials and inline policies.', a table showing 'User name' (Gautham-cli) and 'Last activity' (4 minutes ago), a note about recent activity, and a text input field containing 'Gautham-cli'. There are 'Cancel' and 'Delete user' buttons at the bottom.

- Delete SonarQube organization

The screenshot shows the SonarCloud organization settings page for 'gauthamsrinivasan'. The top navigation bar includes links for 'My Projects', 'My Issues', 'Explore', and a search icon. The main content area has a dark background with white text. It features two sections: 'Only allow private projects' and 'Delete Organization'. The 'Only allow private projects' section contains a note about removing the ability to create public projects and a checkbox followed by a 'Save' button. The 'Delete Organization' section contains a note about deleting the organization and all its projects, followed by a 'Delete' button. At the bottom, there's a footer with copyright information and links to various SonarCloud pages like Terms, Pricing, Privacy, and Documentation. The taskbar at the bottom of the screen shows several pinned application icons.

## Conclusion

This project successfully demonstrates the integration of Jenkins, Terraform, AWS, Kubernetes, SonarCloud, Snyk, OWASP ZAP, and Jira for a comprehensive DevSecOps pipeline. By automating the entire workflow, from infrastructure provisioning to vulnerability scanning and automated issue tracking, we ensure seamless security, scalability, and efficiency in the development lifecycle. The process improves software quality, enhances security posture, and facilitates faster, automated delivery with minimal manual intervention, aligning with best practices in modern DevSecOps.