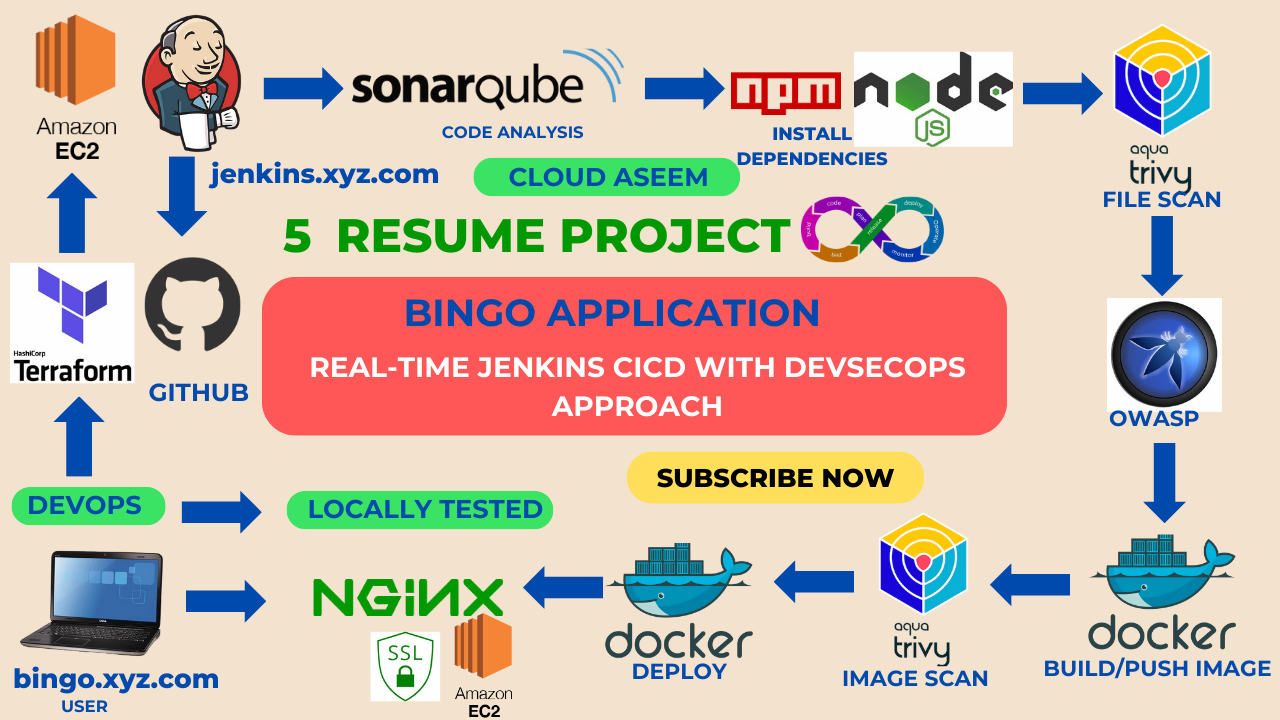
**REAL TIME DEVSECOPS BINGO APP PROJECT| Infrastructure WITH TERRAFORM AND JENKINS CICD PIPELINE |END TO END DEVOPS PROJECT |**

6

project was to learn how to use web sockets and how to create a multiplayer game. It was created using Next.js, React and Sockets.io

Jenkins user = bingouser

Password = bingouser

Sonarqube = bingouser password

Jenkins integration

squ\_02c240a69b9d0045e7043b3f38ce3917c73cad4e

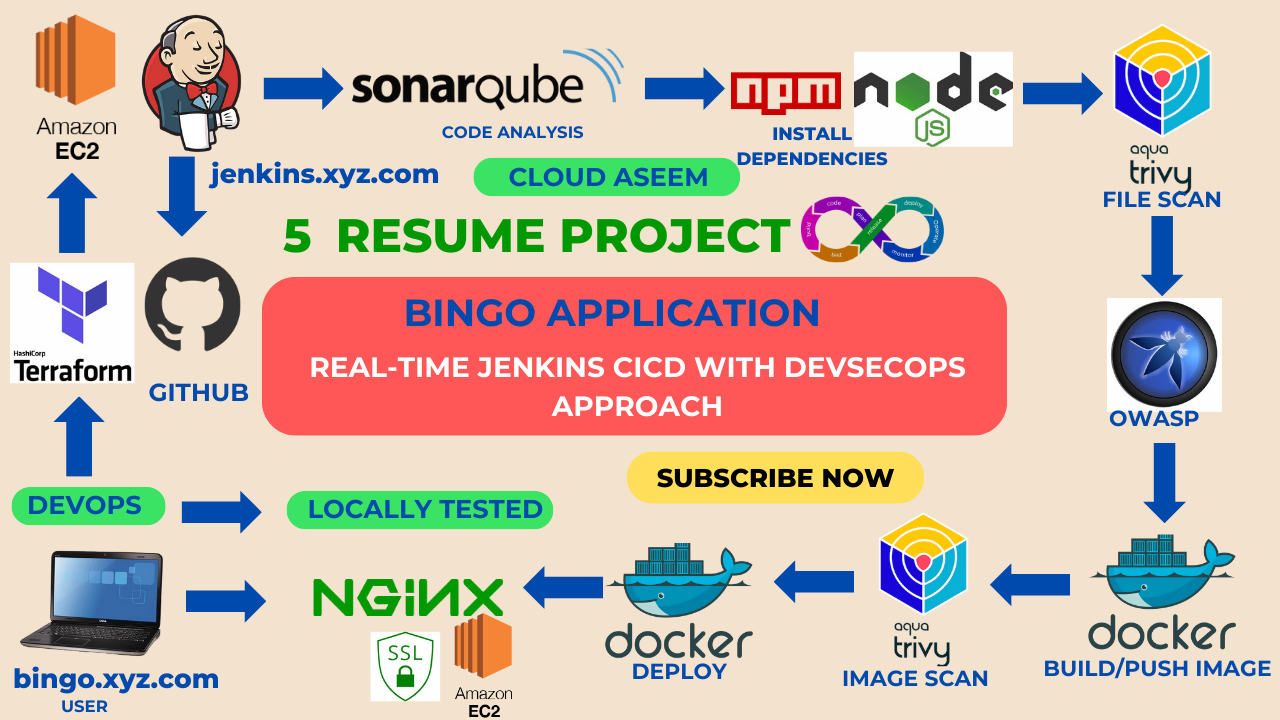
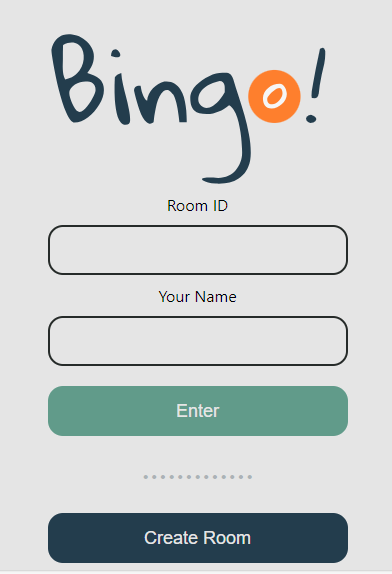
sonar-scanner \

-Dsonar.projectKey=Bingo \

-Dsonar.sources=. \

-Dsonar.host.url=http://65.0.102.160:9000 \

-Dsonar.login=squ\_02c240a69b9d0045e7043b3f38ce3917c73cad4e

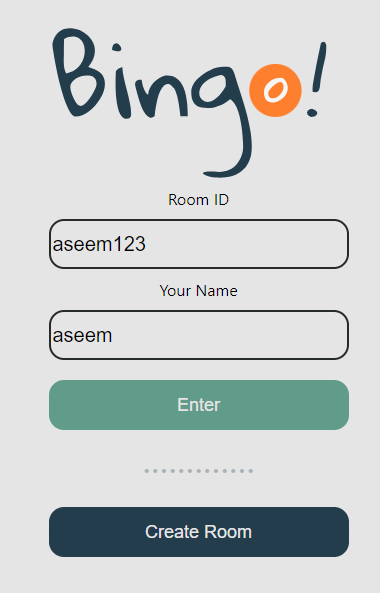


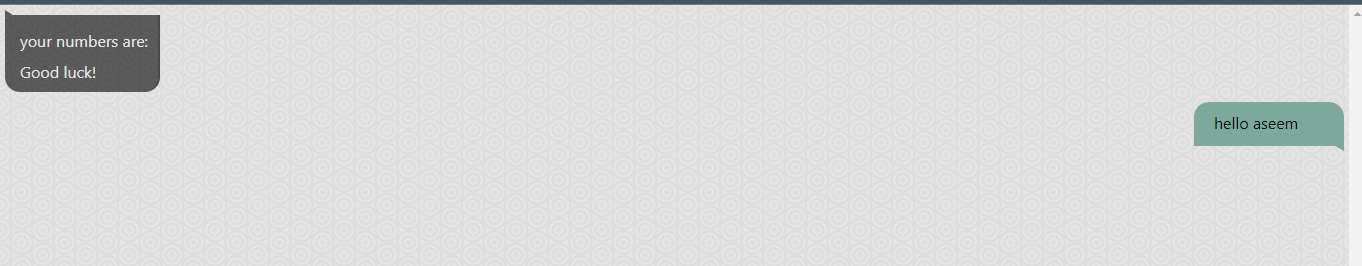
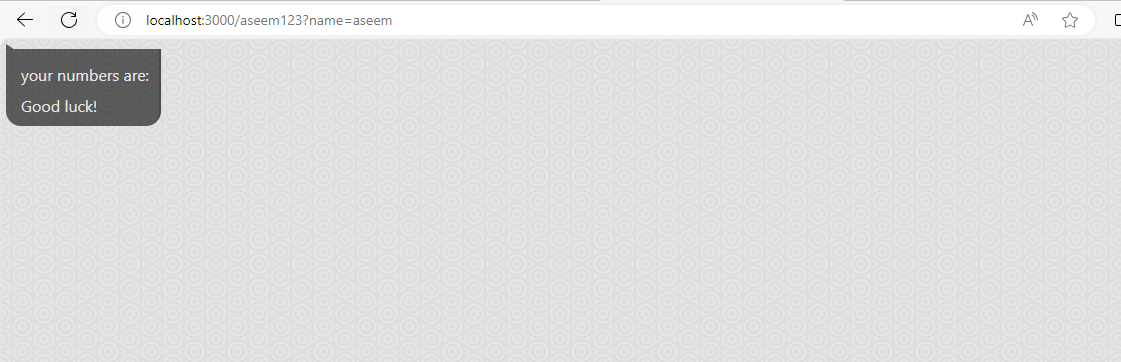
* Enter the room with the new details

RoomId : testgame

Your Name : aseem

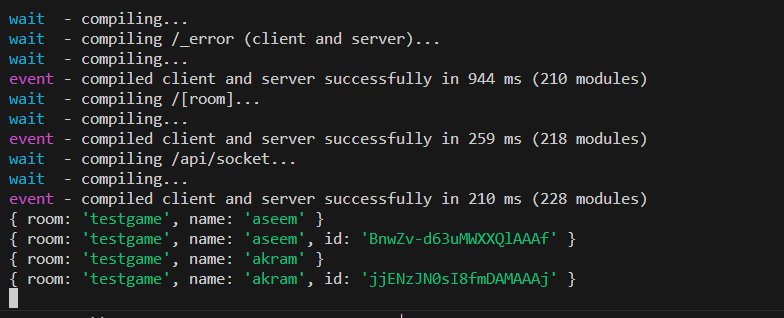
[localhost:3000/testgame?name=aseem](http://localhost:3000/testgame?name=aseem)



If there is no name of the room is exists , it prompt to create a room  
  
  
  
  


Client side :

Chat ID , opens with RoomID and Name   
  
**Server Side :**   
In the application backend logs creates successfully



Login with RoomID with your another user

http://localhost:3000/testgame?name=akram

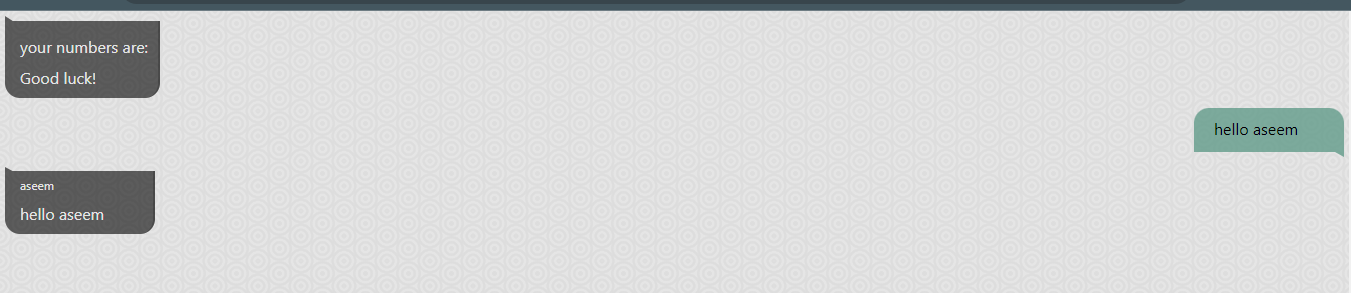


Table of contents

COMPLETE PROJECT VIDEO ON YOUTUBE

Step1: Create directory

Step2: Aws Configure

Step3: Terraform files and Provision Jenkins,sonar

Step 4 — Install Plugins like JDK, Sonarqube Scanner, NodeJs, OWASP Dependency Check

4A — Install Plugin

4B — Configure Java and Nodejs in Global Tool Configuration

Step 5 — Configure Sonar Server in Manage Jenkins

Step 6 — Install OWASP Dependency Check Plugins

Step 7 — Docker Image Build and Push

Embark on a thrillint DevSecOps adventure with our latest project - a real-time Bingo app! 🎉 In this comprehensive tutorial, we'll guide you through the entire lifecycle, from building a robust infrastructure using Terraform to implementing a seamless CI/CD pipeline with Jenkins. 🏗️ Infrastructure with Terraform: Witness the power of Terraform as we construct a scalable and secure environment for our Bingo app. Explore the intricacies of infrastructure as code (IaC) and learn how Terraform streamlines the deployment process. 🔒 DevSecOps Best Practices: Dive deep into the world of DevSecOps with a focus on security. Discover how we integrate Trivy to scan Docker images and containers, ensuring that our app remains fortified against vulnerabilities. We'll also conduct OWASP dependency checks, implement SSL for enhanced encryption, and configure NGINX for optimal web server performance. 🔄 Jenkins CI/CD Pipeline: Join us as we set up a robust Jenkins CI/CD pipeline to automate the entire development lifecycle. Witness the magic of continuous integration as Jenkins builds, tests, and packages our Bingo app. Learn how to seamlessly deploy the app locally for testing and, ultimately, on a server for real-world usage. 👩‍💻 Resume Project Building Locally: Get hands-on experience as we guide you through the process of building the Bingo app on your local machine using Jenkins CI/CD. Understand the nuances of local testing, debugging, and troubleshooting to ensure a smooth development experience. 🚀 Deploy on Server: Witness the culmination of our DevOps journey as we deploy the Bingo app on a production server. Learn the essential steps to ensure a secure and efficient deployment, including best practices for server configuration and maintenance. 🚨 Don't miss out on this immersive tutorial that combines the best of DevOps and Security practices. Elevate your skills and take your DevSecOps game to the next level with our Real-Time DevSecOps Bingo App Project! 🌐💻 #DevSecOps #Terraform #Jenkins #CI/CD #DevOpsProject #BingoApp #SecurityFirst

COMPLETE PROJECT VIDEO ON YOUTUBE

https://www.youtube.com/watch?v=fGaGf3X3s6Q

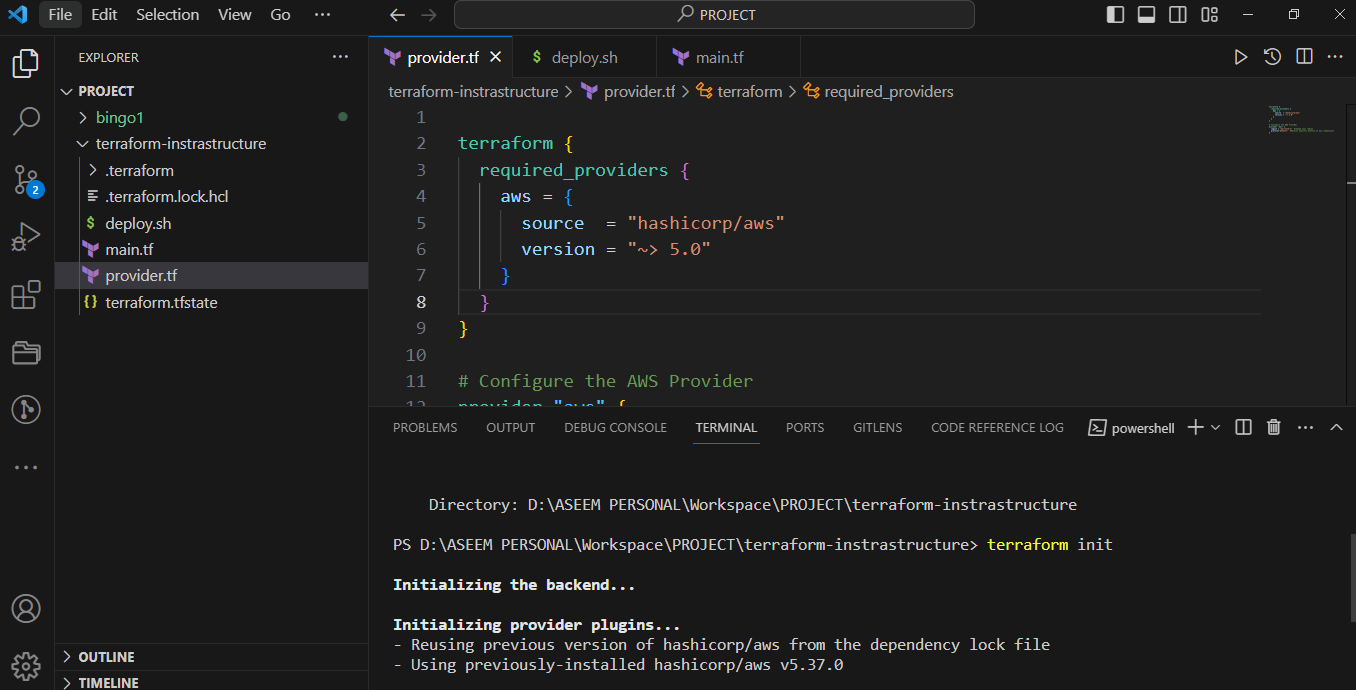
COMPLETE PROJECT VIDEO ON YOUTUBE to install Terraform and Aws cli

Terraform Series Day – 04

https://www.youtube.com/watch?v=Gbyx1dNQyKQ

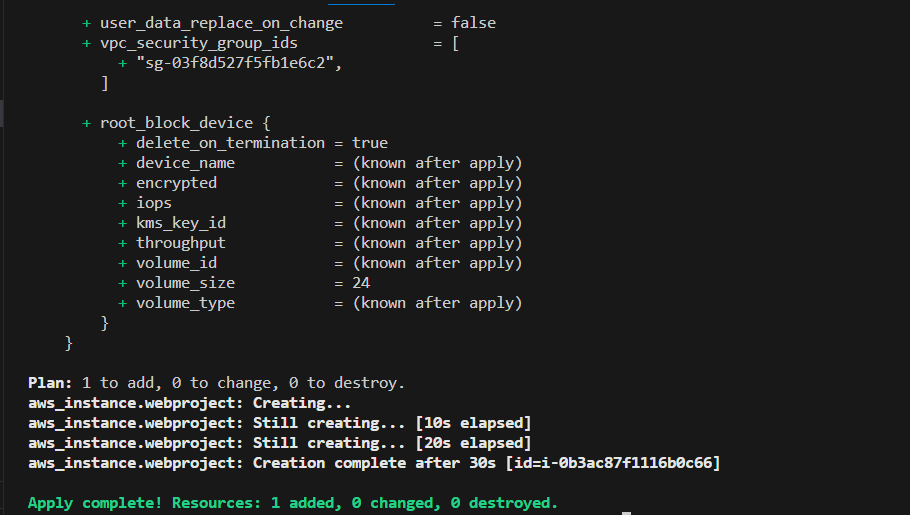
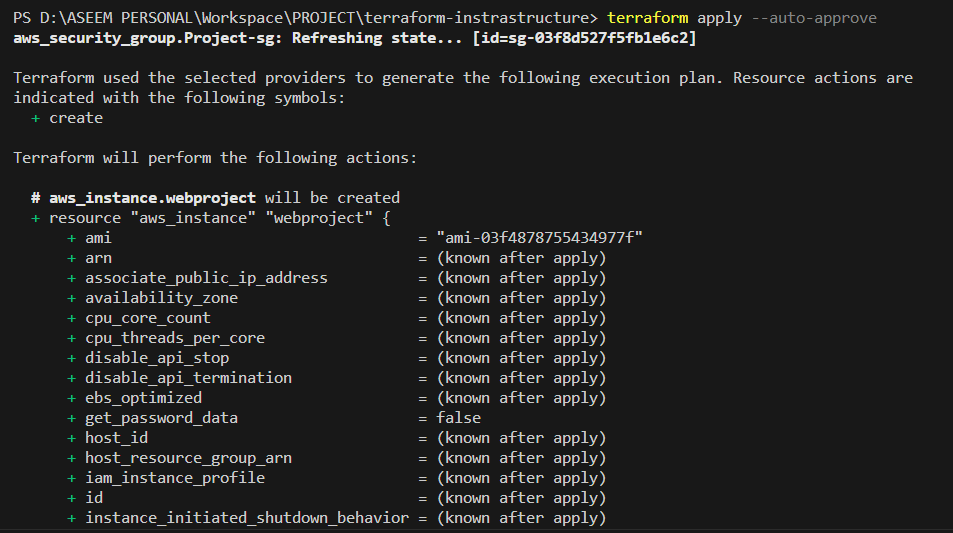
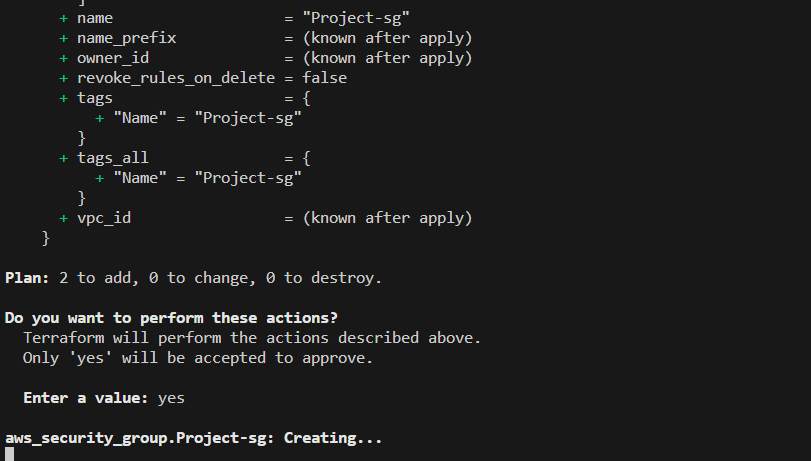
GITHUB LINK

<https://github.com/Aseemakram19/bingo1.git>

1. **Testing locally** 

terraform apply

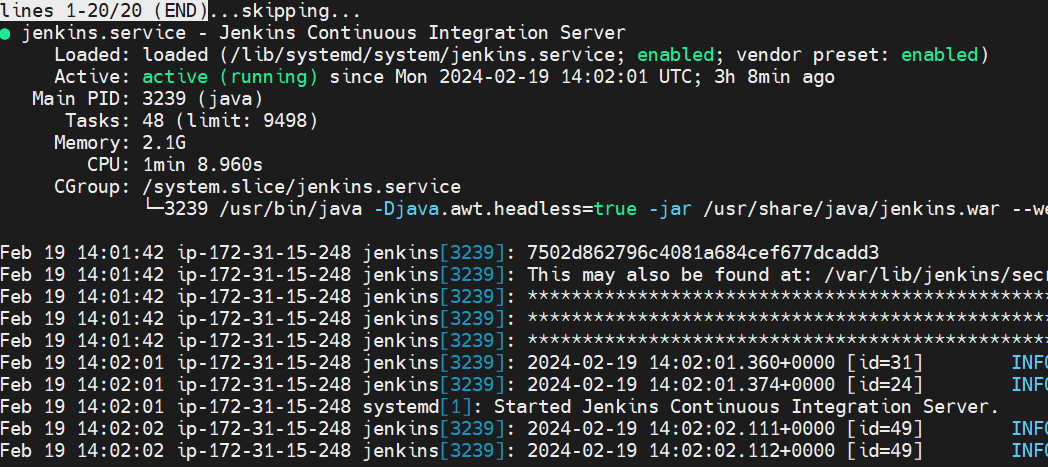
Prompt you to confirm the apply to create resource , enter the value “yes”

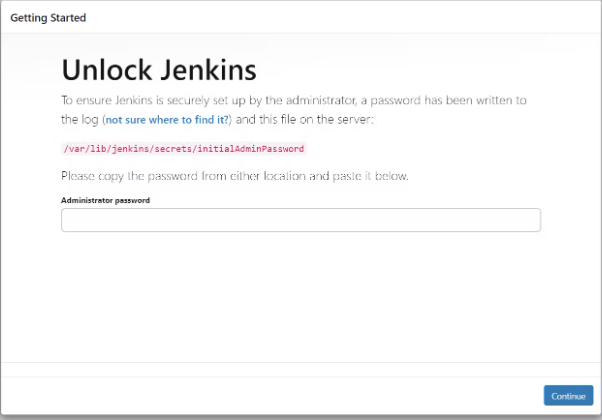
Or else use command to create without prompting the action  
 terraform apply --auto-approve 

Resources has been created validate with IP:8080 for jenkins

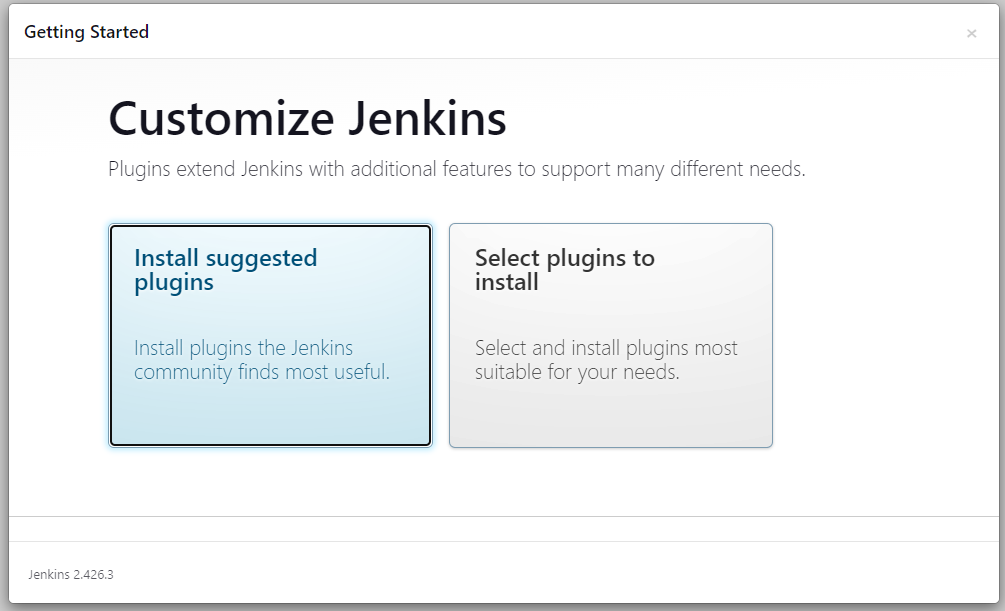
#Login to server with credentials , set the hostname with following command

Resources has been created validate with IP:8080 for jenkins

1. Jenkins 

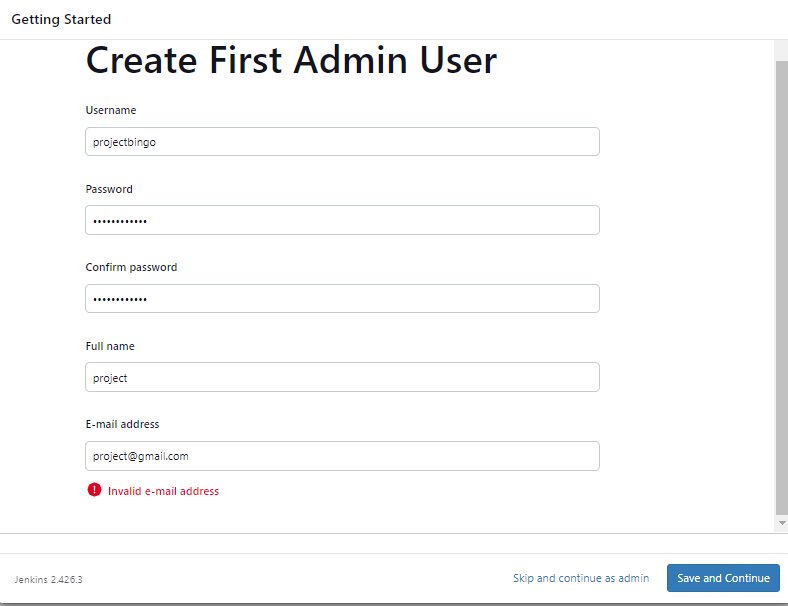


sudo cat /var/lib/jenkins/secrets/initialAdminPassword



Click on install suggested plugin default.

Create a user click on save and continue.

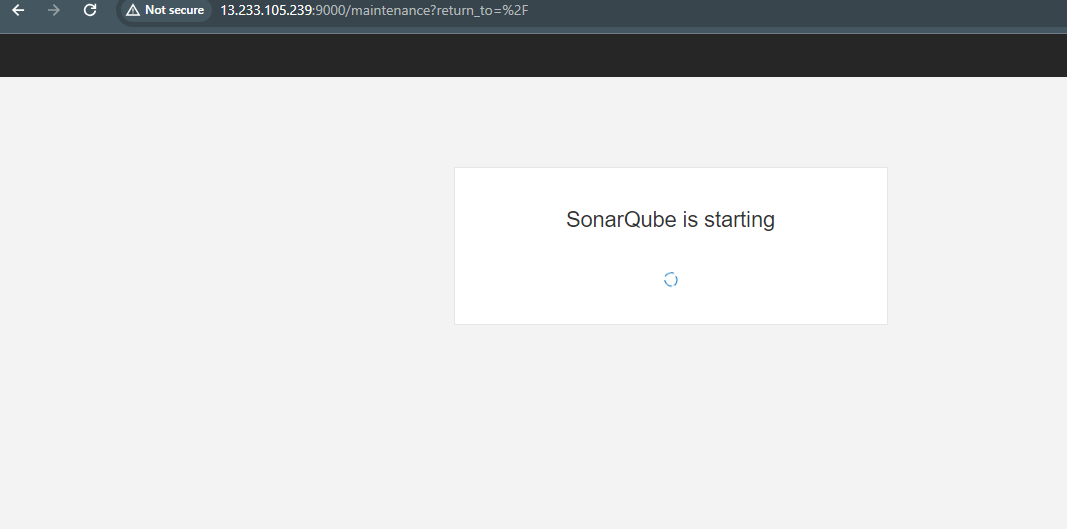


projectbingo as new password

Copy your Public key again and paste it into a new tab

<instance-public-ip:9000>

Now our sonarqube is up and running

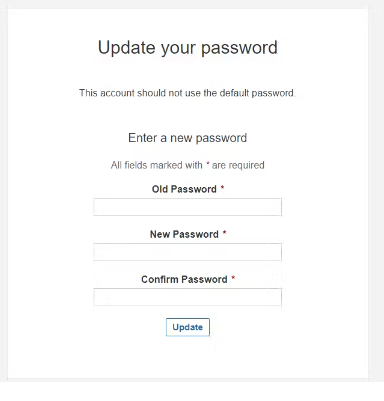


Enter Default username and password, click on login and change password

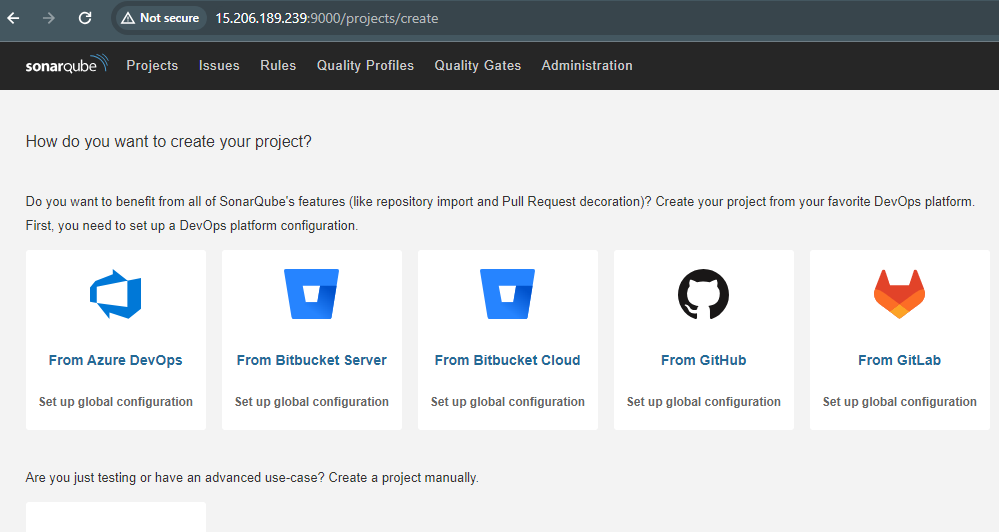
username admin

password admin

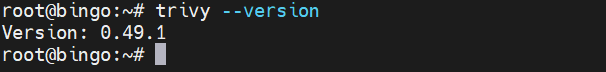
projectbingo as new password



projectbingo as new password



**Check trivy version**



Install Plugins like JDK, Sonarqube Scanner, NodeJs, OWASP Dependency Check

Goto Manage Jenkins →Plugins → Available Plugins →

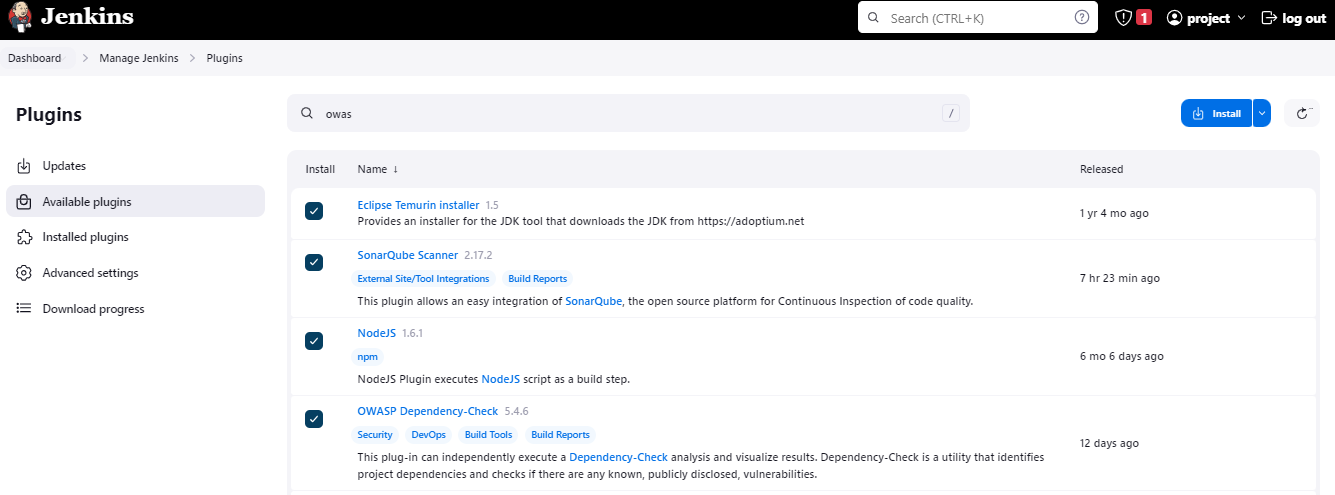
Install below plugins

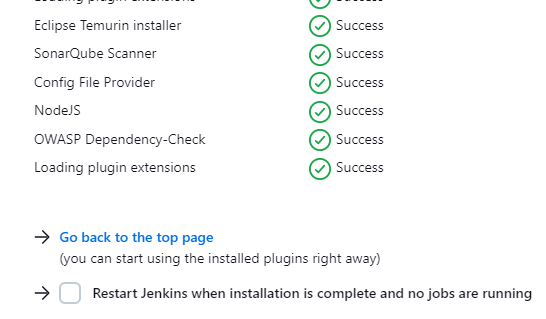
1 . Eclipse Temurin Installer (Install without restart)

2 . SonarQube Scanner (Install without restart)

3 . NodeJs Plugin (Install Without restart) – 16.20.2

4. OWASP Dependency Check Plugins





We need to install the Docker tool in our system, Goto Dashboard → Manage Plugins → Available plugins → Search for Docker and install these plugins

1. Docker

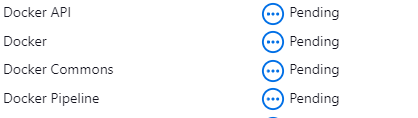
2. Docker Commons

3. Docker Pipeline

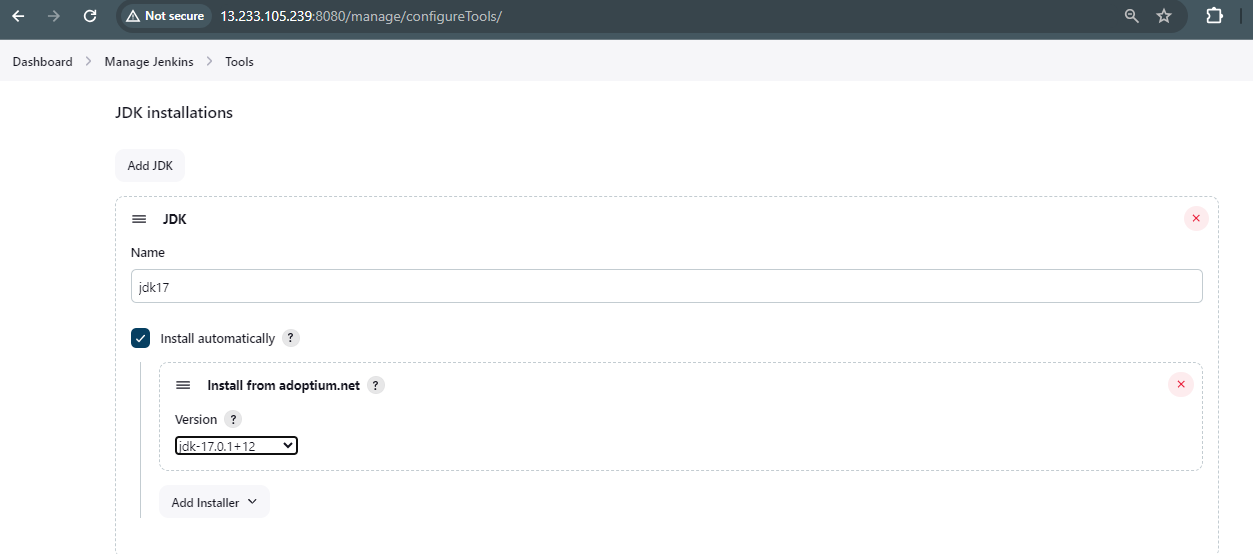
4. Docker API

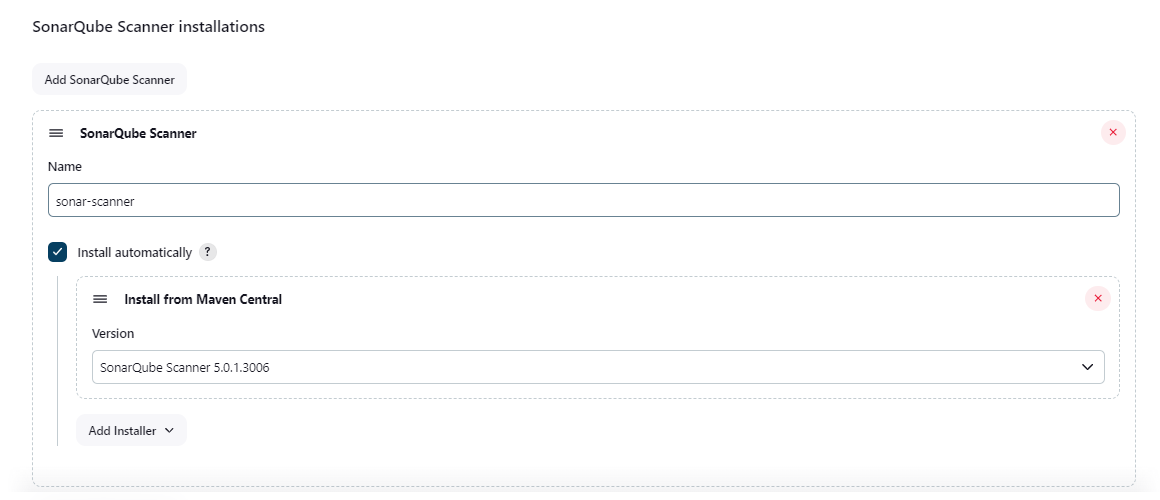
5. docker-build-step

and click on install without restart

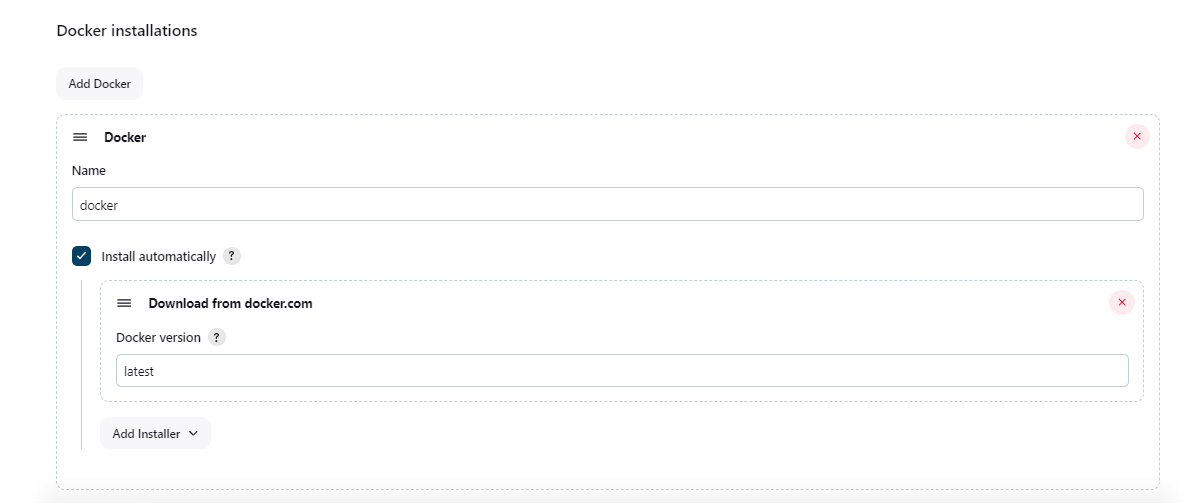


Now, goto Dashboard → Manage Jenkins → Tools →





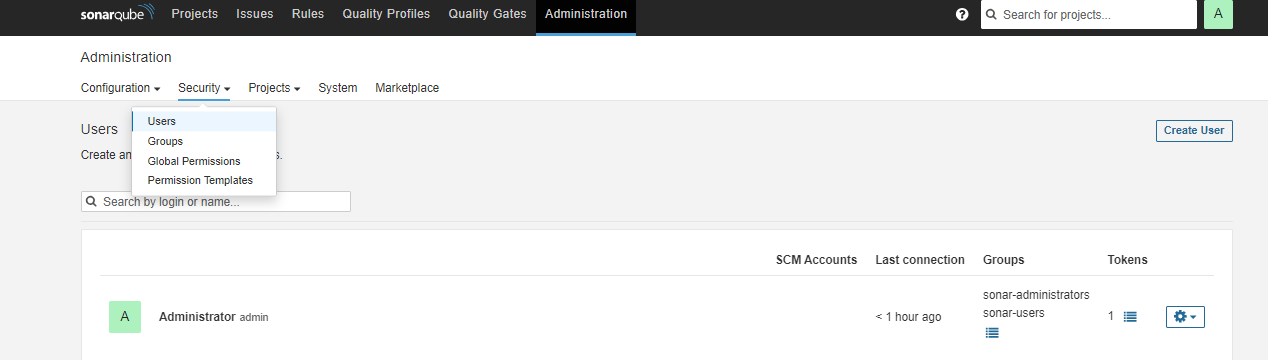
16.20.2



Apply & save the setting

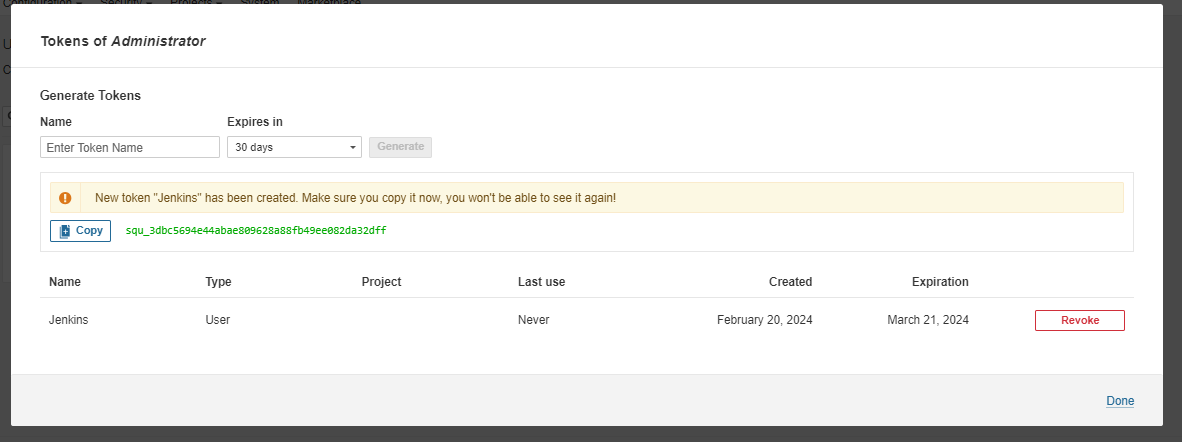
SonarQube token

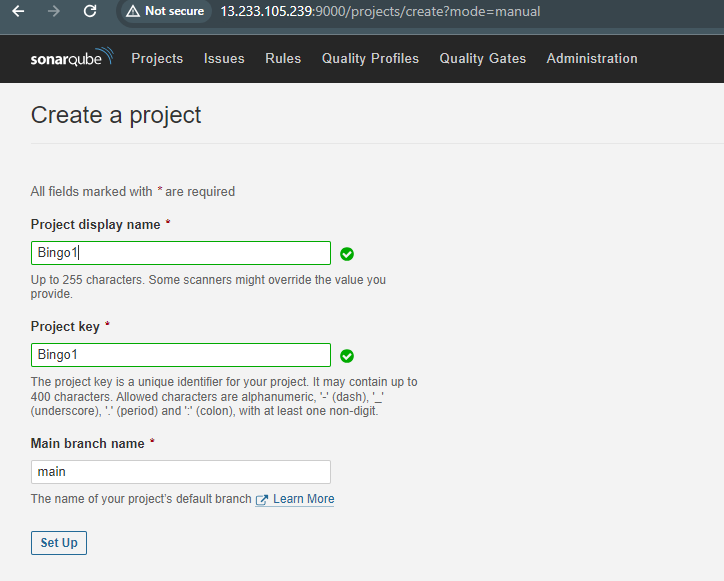
Click on Administration & security & user then click on right 3 dot to create new token



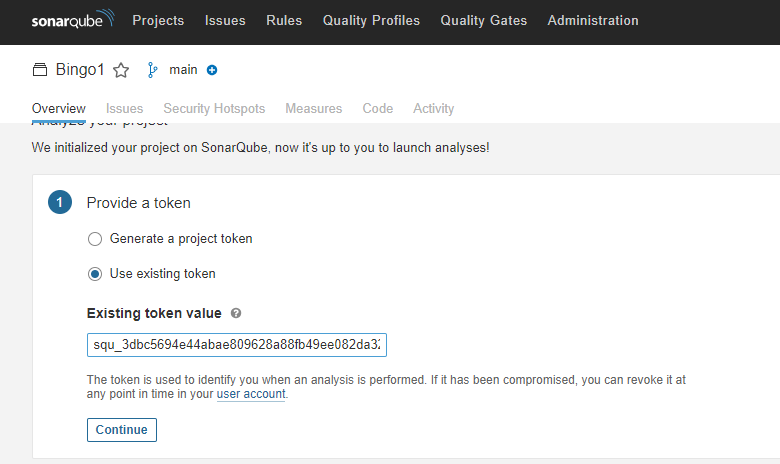
Sonar token for jenkins integration

squ\_3dbc5694e44abae809628a88fb49ee082da32dff





Click on setup , click on locally , provide the token



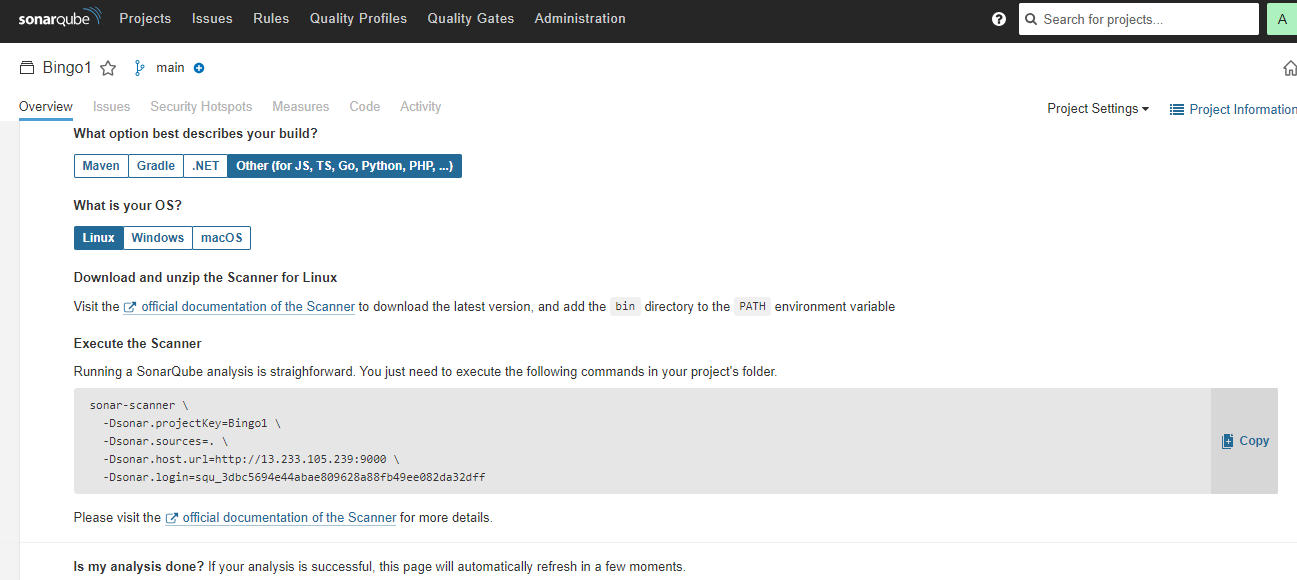
sonar-scanner \

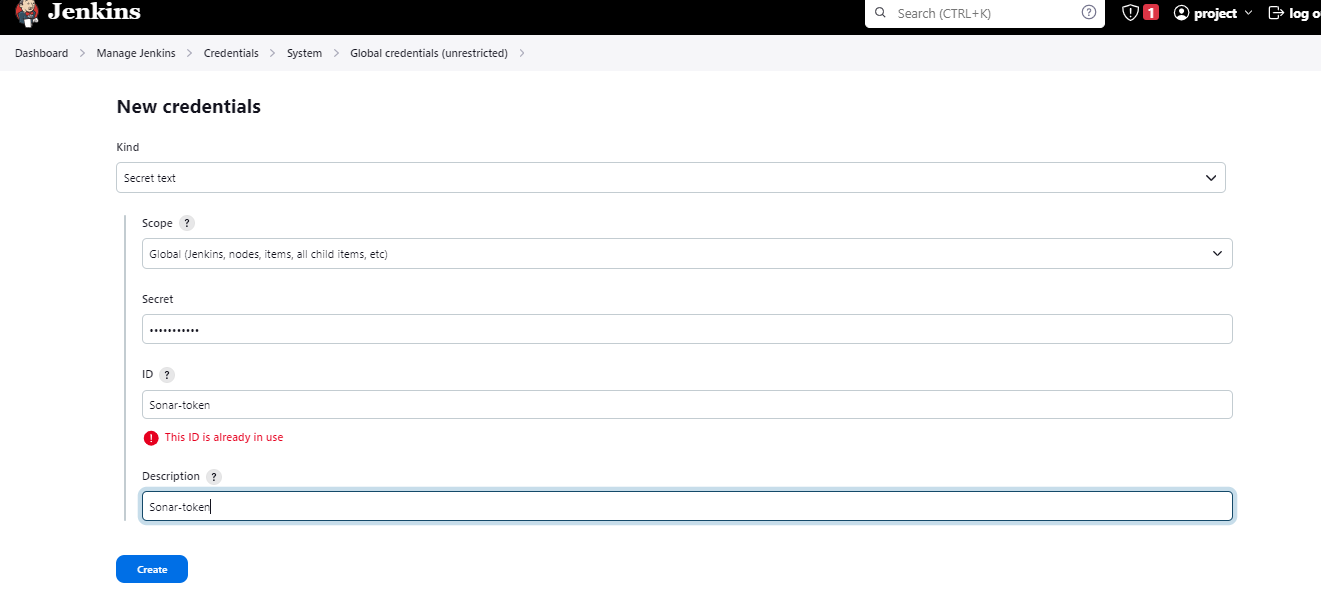
-Dsonar.projectKey=Bingo1 \

-Dsonar.sources=. \

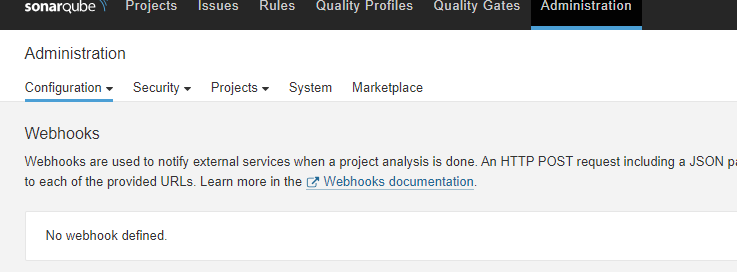
-Dsonar.host.url=http://13.233.105.239:9000 \

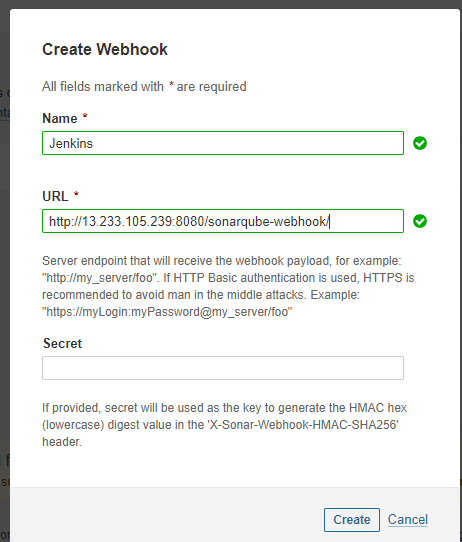
-Dsonar.login=squ\_3dbc5694e44abae809628a88fb49ee082da32dff

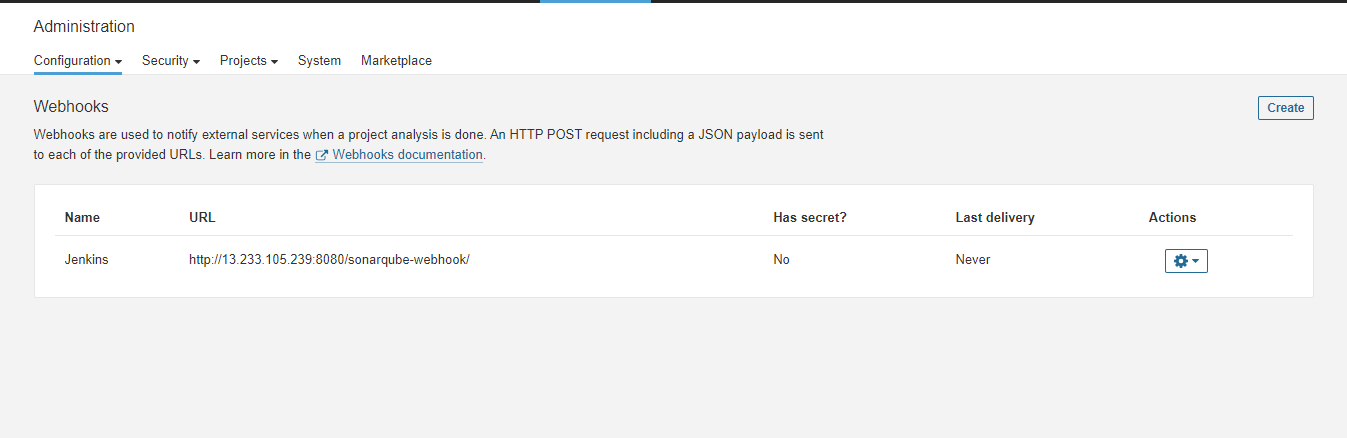


Jenkins in order to add SonarQube token in Jenkins credentials sections 

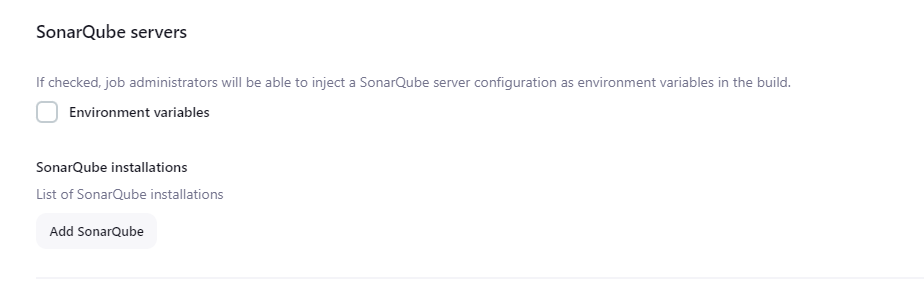
Quality Gate

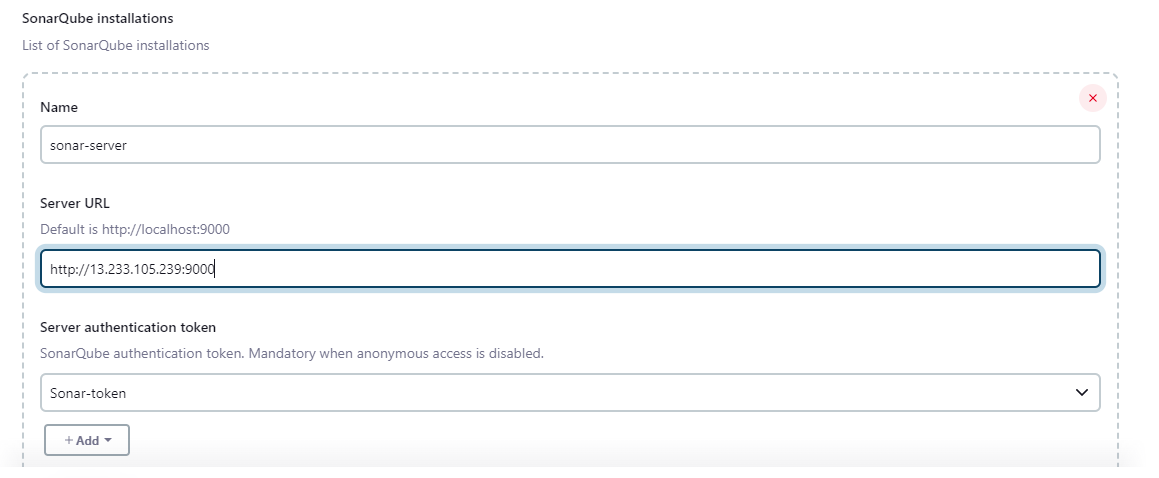




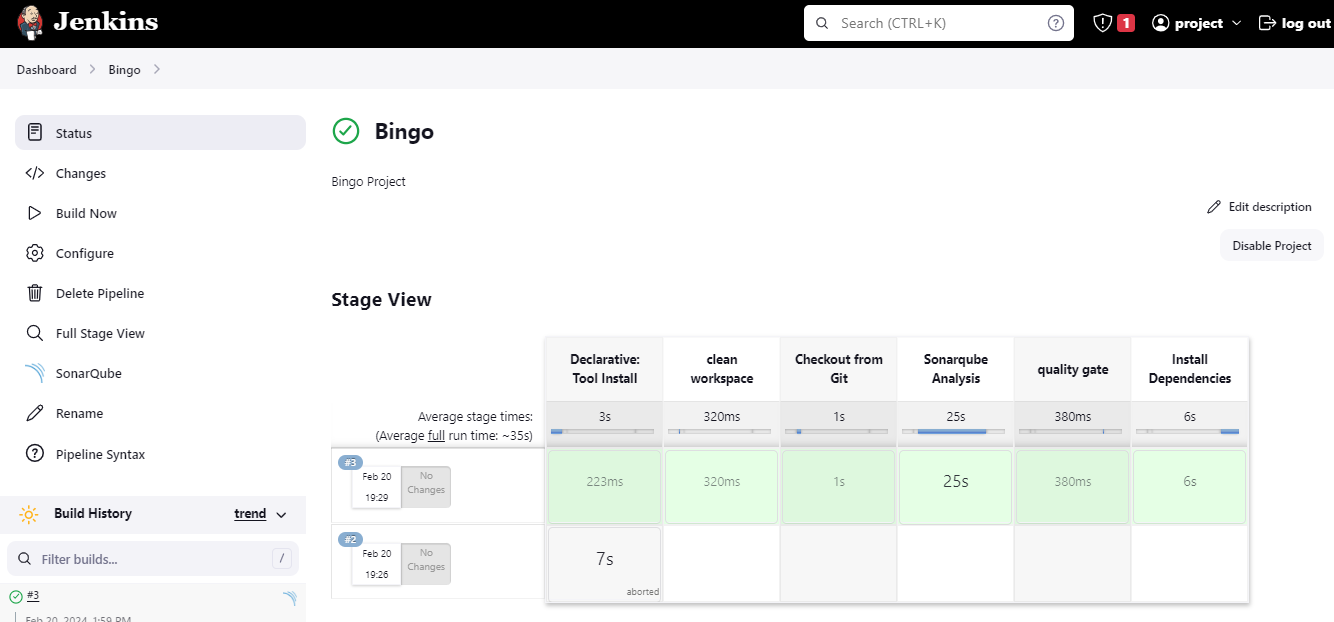


Configure the sonarqube server





Create new pipeline in for bingo project



pipeline{

agent any

tools{

jdk 'jdk17'

nodejs 'node16'

}

#environment {

#SCANNER\_HOME=tool 'sonar-scanner'

#}

stages {

stage('clean workspace'){

steps{

cleanWs()

}

}

stage('Checkout from Git'){

steps{

git branch: 'main', url: 'https://github.com/Aseemakram19/bingo1.git'

}

}

stage("Sonarqube Analysis "){

steps{

withSonarQubeEnv('sonar-server') {

sh ''' $SCANNER\_HOME/bin/sonar-scanner -Dsonar.projectName=Bingo \

-Dsonar.projectKey=Bingo '''

}

}

}

stage("quality gate"){

steps {

script {

waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'

}

}

}

stage('Install Dependencies') {

steps {

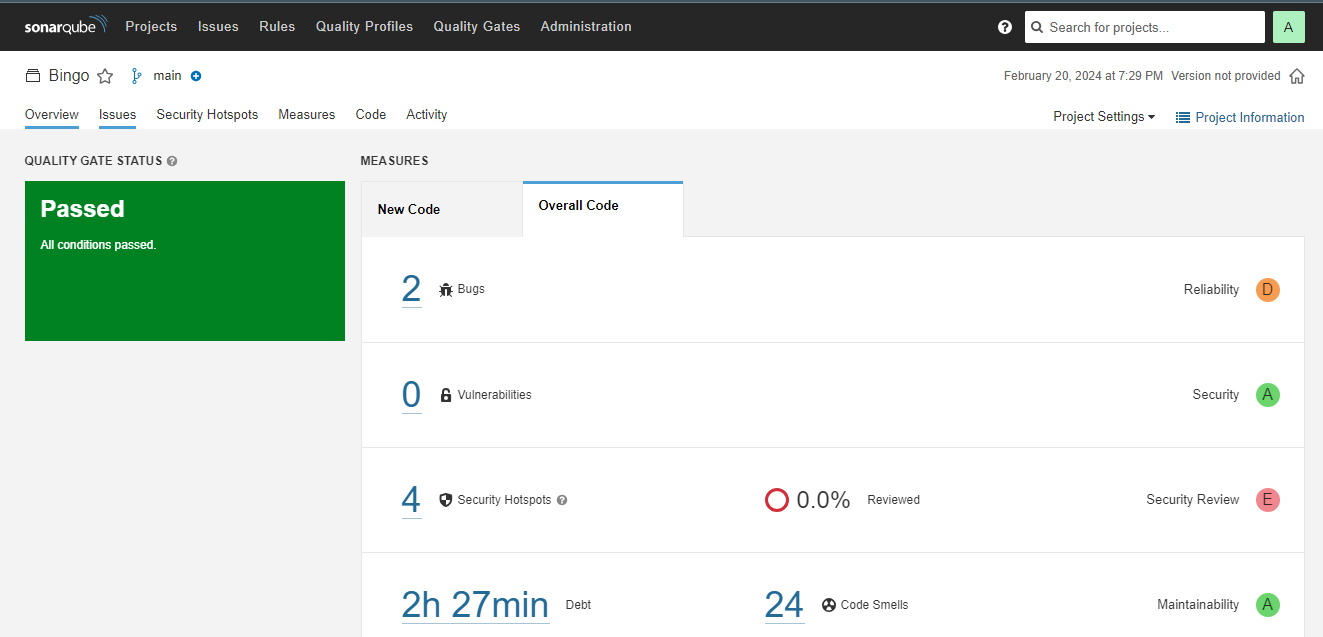
sh "npm install"

}

}

}

}



Add remaining tools checks in our pipelines , Now go configure → Pipeline and add this stage to your pipeline and build.

**OWASP Dependency-Check** is an open-source tool that identifies project dependencies and checks if there are any known, publicly disclosed, vulnerabilities in those dependencies. It is part of the Open Web Application Security Project (OWASP) and is designed to help organizations and developers identify and mitigate security risks related to third-party libraries and components used in their software projects.

**Key aspects of OWASP Dependency-Check:**

1. **Identification of Vulnerabilities:**
   * Dependency-Check analyzes project dependencies, including libraries, frameworks, and components.
   * It checks these dependencies against a database of known vulnerabilities, including the National Vulnerability Database (NVD) and other sources.
2. **Automatic Scanning:**
   * Dependency-Check can be integrated into the development process and CI/CD pipelines.
   * It automatically scans dependencies to identify and report vulnerabilities, allowing for early detection and remediation.5
3. **Multiple Language Support:**
   * Dependency-Check supports multiple programming languages and package managers, making it versatile for a wide range of projects.
   * It can analyze dependencies in Java, JavaScript, .NET, Python, Ruby, and more.
4. **Integration with Build Tools:**
   * It integrates with popular build tools such as Maven, Gradle, npm, and others, making it easy to incorporate into existing development workflows.
5. **Usefulness:**
   * **Early Vulnerability Detection:** Dependency-Check enables developers to identify and address security issues early in the development lifecycle.
   * **Continuous Monitoring:** It supports continuous monitoring of dependencies, helping to keep software secure over time as new vulnerabilities are discovered.
6. **Applicability:**
   * **Web Applications:** Especially relevant for web applications where security vulnerabilities in dependencies can lead to serious security issues.
   * **Any Software with Dependencies:** Applicable to any software project that relies on third-party libraries or components.
7. **Complementary to SCA Practices:**
   * Dependency-Check is part of Software Composition Analysis (SCA) practices, focusing on the security aspects of third-party software components.
8. **Open Source and Actively Maintained:**
   * As an open-source project, Dependency-Check benefits from community contributions and is actively maintained to keep the vulnerability database up-to-date.

stage('OWASP FS SCAN') {

steps {

dependencyCheck additionalArguments: '--scan ./ --disableYarnAudit --disableNodeAudit', odcInstallation: 'DP'

dependencyCheckPublisher pattern: '\*\*/dependency-check-report.xml'

}

}

stage('TRIVY FS SCAN') {

steps {

sh "trivy fs . > trivyfs.txt"

}

}

**Trivy is an open-source vulnerability scanner designed for containerized environments. Here's a breakdown of its key aspects:**

What is Trivy?

Vulnerability Scanner: Trivy is primarily used as a vulnerability scanner, focusing on container images. It helps identify security vulnerabilities within the components (libraries, packages, dependencies) of containerized applications.

Container Security: Trivy is specifically tailored for container security, making it a valuable tool in DevSecOps (Development, Security, and Operations) workflows. It's commonly used with container orchestration platforms like Kubernetes and containerization technologies like Docker.

Why use Trivy?

Early Detection of Vulnerabilities: Trivy aids in the early detection of vulnerabilities in container images. By scanning container images during the development and CI/CD (Continuous Integration/Continuous Deployment) processes, security issues can be addressed before deployment, reducing the risk of exposing vulnerabilities in production.

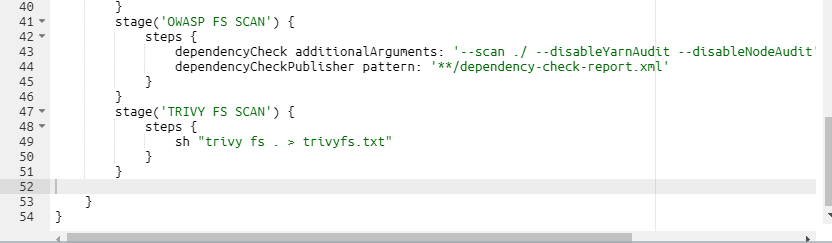
Automation: Trivy supports automation, allowing developers and DevOps teams to integrate vulnerability scanning seamlessly into their CI/CD pipelines. This ensures that security checks are part of the automated build and deployment processes.

Where is Trivy used?

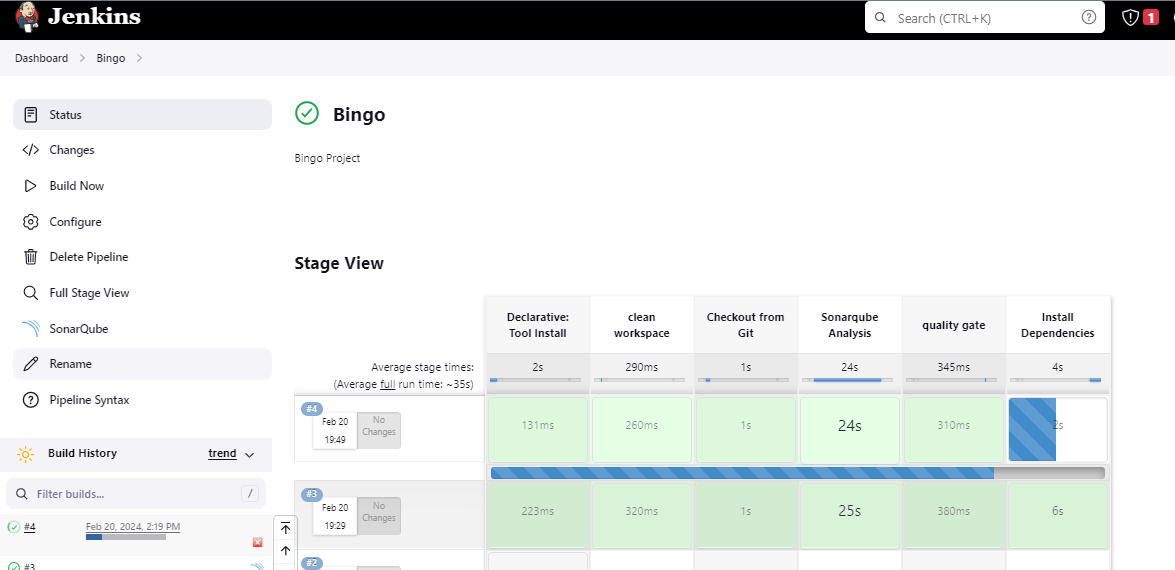
Containerized Environments: Trivy is primarily used in containerized environments, where applications are packaged along with their dependencies into containers. This includes popular containerization platforms such as Docker and container orchestration systems like Kubernetes.

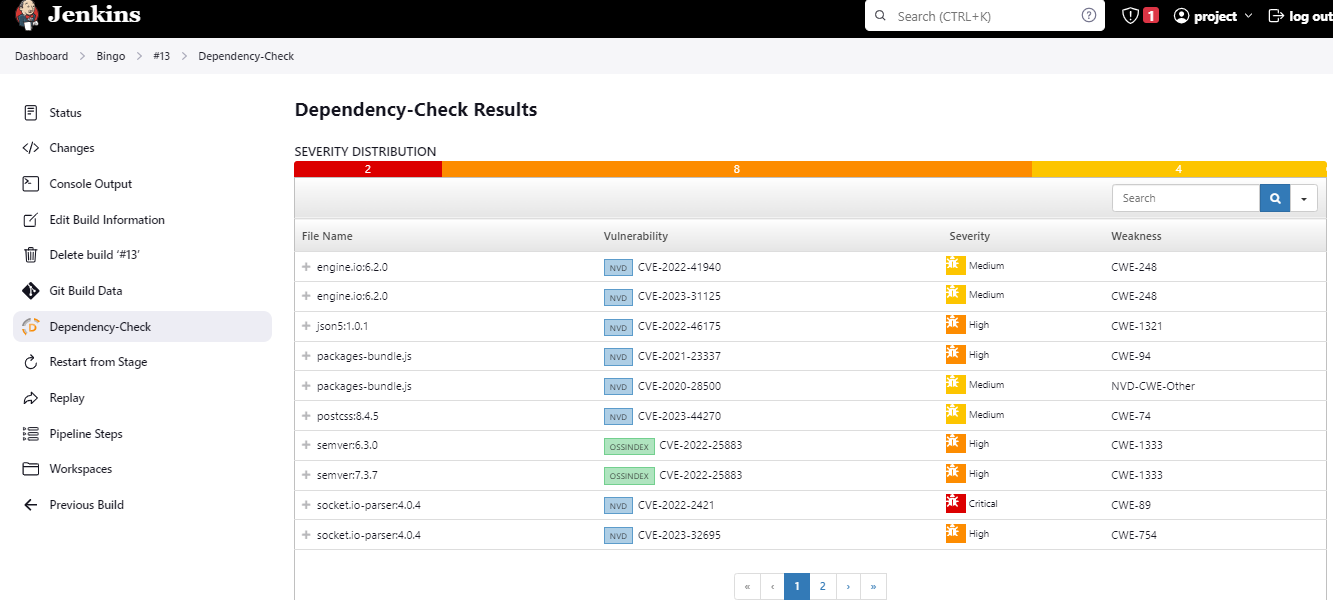
DevSecOps Pipelines: Trivy is integrated into DevSecOps pipelines to automate security checks throughout the software development lifecycle. It helps teams identify and remediate vulnerabilities early in the development process.

Security Audits: Trivy can be used for regular security audits of container images in production environments. This ongoing monitoring helps ensure that the deployed containers remain secure against the latest vulnerabilities.



Build now the pipeline





Go to docker hub & created a access token to integrate with Jenkins for Docker build & docker Push ,& docker pull to deploy

<https://hub.docker.com/>

Add docker credentials in credentials

Username – ##########

Password - ##########



Add this stage to Pipeline Script

stage("Docker Build & Push") {

steps {

script {

withDockerRegistry(credentialsId: 'docker', toolName: 'docker') {

sh "docker build -t bingo ."

sh "docker tag bingo aseemakram19/bingo:latest "

sh "docker push aseemakram19/bingo:latest "

}

}

}

}

stage("TRIVY"){

steps{

sh "trivy image aseemakram19/bingo:latest > trivyimage.txt"

}

}

You will see the output below, with a dependency trend.

Now Run the container to see if the bingo app is live with adding the below stage

