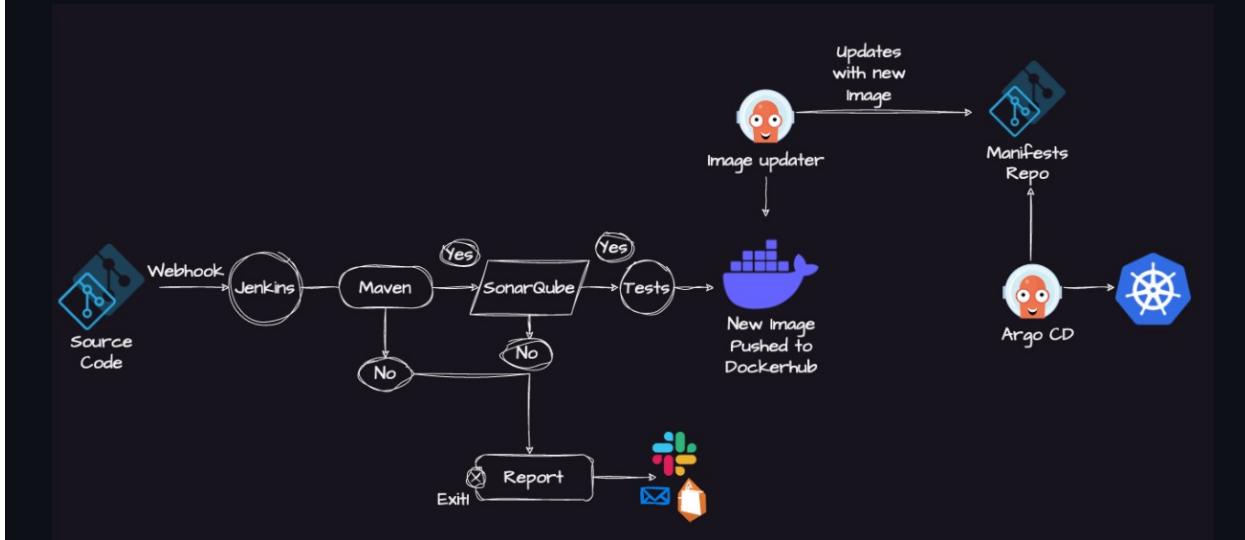


## JENKINS END TO END CICD Implementation with Detailed Notes

### Jenkins Pipeline for Java based application using Maven, SonarQube, Argo CD, Helm and Kubernetes



Here are the step-by-step details to set up an end-to-end Jenkins pipeline for a Java application using SonarQube, Argo CD, Helm, and Kubernetes:

#### Prerequisites:

- Java application code hosted on a Git repository
- Jenkins server
- Kubernetes cluster
- Helm package manager
- Argo CD

```
1. Install the necessary Jenkins plugins:  
    1.1 Git plugin  
    1.2 Maven Integration plugin  
    1.3 Pipeline plugin  
    1.4 Kubernetes Continuous Deploy plugin  
  
2. Create a new Jenkins pipeline:  
    2.1 In Jenkins, create a new pipeline job and configure it with the Git repository URL for the Java application.  
    2.2 Add a Jenkinsfile to the Git repository to define the pipeline stages.  
  
3. Define the pipeline stages:  
    Stage 1: Checkout the source code from Git.  
    Stage 2: Build the Java application using Maven.  
    Stage 3: Run unit tests using JUnit and Mockito.  
    Stage 4: Run SonarQube analysis to check the code quality.  
    Stage 5: Package the application into a JAR file.  
    Stage 6: Deploy the application to a test environment using Helm.  
    Stage 7: Run user acceptance tests on the deployed application.  
    Stage 8: Promote the application to a production environment using Argo CD.  
  
4. Configure Jenkins pipeline stages:  
    Stage 1: Use the Git plugin to check out the source code from the Git repository.  
    Stage 2: Use the Maven Integration plugin to build the Java application.  
    Stage 3: Use the JUnit and Mockito plugins to run unit tests.  
    Stage 4: Use the SonarQube plugin to analyze the code quality of the Java application.  
    Stage 5: Use the Maven Integration plugin to package the application into a JAR file.  
    Stage 6: Use the Kubernetes Continuous Deploy plugin to deploy the application to a test environment using Helm.  
    Stage 7: Use a testing framework like Selenium to run user acceptance tests on the deployed application.  
    Stage 8: Use Argo CD to promote the application to a production environment.  
  
5. Set up Argo CD:  
    Install Argo CD on the Kubernetes cluster.  
    Set up a Git repository for Argo CD to track the changes in the Helm charts and Kubernetes manifests.  
    Create a Helm chart for the Java application that includes the Kubernetes manifests and Helm values.  
    Add the Helm chart to the Git repository that Argo CD is tracking.  
  
6. Configure Jenkins pipeline to integrate with Argo CD:  
    6.1 Add the Argo CD API token to Jenkins credentials.  
    6.2 Update the Jenkins pipeline to include the Argo CD deployment stage.  
  
7. Run the Jenkins pipeline:  
    7.1 Trigger the Jenkins pipeline to start the CI/CD process for the Java application.  
    7.2 Monitor the pipeline stages and fix any issues that arise.
```

This end-to-end Jenkins pipeline will automate the entire CI/CD process for a Java application, from code checkout to production deployment, using popular tools like SonarQube, Argo CD, Helm, and Kubernetes.

# springboot applicaton

Code Blame 24 lines (19 loc) · 771 Bytes GitHub Copilot

```
1 package com.abhishek;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.stereotype.Controller;
6 import org.springframework.ui.Model;
7 import org.springframework.web.bind.annotation.GetMapping;
8
9 @SpringBootApplication
10 @Controller
11 public class StartApplication {
12
13     @GetMapping("/")
14     public String index(Model model) {
15         model.addAttribute("title", "I have successfully built a sprint boot application using Maven");
16         model.addAttribute("msg", "This application is deployed on to Kubernetes using Argo CD");
17         return "index";
18     }
19
20     public static void main(String[] args) {
21         SpringApplication.run(StartApplication.class, args);
22     }
23 }
24 }
```

YouTube JENKINS END TO E Jenkins-Zero-To-Hero Jenkins Zero to He ULTIMATE CI/CD P sre and devops are ... +

github.com/iam-veeramalla/Jenkins-Zero-To-Hero/blob/main/java-maven-sonar-argocd-helm-k8s/spring-boot-app/src/main/resources/templates/index.html

Go to file History

Code Blame 25 lines (21 loc) · 782 Bytes GitHub Copilot

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"/>
    <link data-th-href="@{/css/main.css?{id}}&id=${timestamp}" rel="stylesheet">
    <title>CI/CD Master Class By Abhishek Veeramalla</title>
</head>
<body>
<nav class="navbar navbar-expand-md navbar-dark bg-dark fixed-top">
    <a class="navbar-brand" href="#">Ultimate CI/CD Pipeline using Java</a>
</nav>
<main role="main" class="container">
    <div class="starter-template">
        <h1 th:text="${title}">Default title.</h1>
        <p th:text="${msg}">Default text.</p>
    </div>
</main>
<script data-th-src="@{/js/main.js}"></script>
</body>
</html>
```

# select AWS EC2 instance ....Paid one ... Free tier will not Work with many microservices

**Name and tags** [Info](#)

Name  
 Add additional tags

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

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

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

**Quick Start**

Amazon Linux macOS Ubuntu Windows Red Hat S >  [Browse more AMIs](#)  
Including AMIs from AWS, Marketplace and the Community

Number of instances [Info](#)  
1

Software Image (AMI)  
Amazon Linux 2023 AMI 2023.0.2... [read more](#)  
ami-00c39f71452c08778

Virtual server type (instance type)  
t2.micro

Firewall (security group)  
New security group

Storage (volumes)  
1 volume(s) - 8 GiB

[Cancel](#) [Launch instance](#) [Review commands](#)

# instance type t2.large 2 cpu

Description  
Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2023-03-25

Architecture [AMI ID](#)  
64-bit (x86) ami-007855ac798b5175e [Verified provider](#)

**Instance type** [Info](#)

Instance type  
**t2.large**  
Family: t2 2 vCPU 8 GiB Memory  
On-Demand Windows pricing: 0.1208 USD per Hour  
On-Demand RHEL pricing: 0.1528 USD per Hour  
On-Demand SUSE pricing: 0.1928 USD per Hour  
On-Demand Linux pricing: 0.0928 USD per Hour

[Compare instance types](#)

**Key pair (login)** [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Network [Info](#)

vpc-0895d01f56ccd9b25

Subnet [Info](#)

No preference (Default subnet in any availability zone)

Auto-assign public IP [Info](#)

Enable

**Firewall (security groups)** [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group

Select existing security group

We'll create a new security group called 'launch-wizard-17' with the following rules:

- Allow SSH traffic from  
Helps you connect to your instance

Anywhere  
0.0.0.0/0

- Allow HTTPS traffic from the internet  
To set up an endpoint, for example when creating a web server
- Allow HTTP traffic from the internet  
To set up an endpoint, for example when creating a web server

**⚠️** Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only. **X**

```
> Jenkins-Zero-To-Hero git:(main) cd java-maven-sonar-argocd-helm-k8s
> java-maven-sonar-argocd-helm-k8s git:(main) ls
README.md      spring-boot-app
> java-maven-sonar-argocd-helm-k8s git:(main) cd spring-boot-app
> spring-boot-app git:(main) ls
Dockerfile README.md pom.xml    src
> spring-boot-app git:(main) █
```

This is a simple Spring Boot based Java application that can be built using Maven. Spring Boot dependencies are handled using the pom.xml at the root directory of the repository.

This is a MVC architecture based application where controller returns a page with title and message attributes to the view.

### Execute the application locally and access it using your browser

Checkout the repo and move to the directory

```
git clone https://github.com/iam-veeramalla/Jenkins-Zero-To-Hero/java-maven-sonar-argocd-helm-k8s/spring-boot-app  
cd java-maven-sonar-argocd-helm-k8s/spring-boot-app
```

Execute the Maven targets to generate the artifacts

```
mvn clean package
```

The above maven target stroes the artifacts to the `target` directory. You can either execute the artifact on your local machine (or) run it as a Docker container.

The above maven target stroes the artifacts to the `target` directory. You can either execute the artifact on your local machine (or) run it as a Docker container.

\*\* Note: To avoid issues with local setup, Java versions and other dependencies, I would recommend the docker way. \*\*

### Execute locally (Java 11 needed) and access the application on <http://localhost:8080>

```
java -jar target/spring-boot-web.jar
```

### The Docker way

Build the Docker Image

```
docker build -t ultimate-cicd-pipeline:v1 .
```

```
docker run -d -p 8010:8080 -t ultimate-cicd-pipeline:v1
```

Hurray !! Access the application on <http://<ip-address>:8010>

### Next Steps

YouTube | JENKINS END TO END CI | Jenkins-Zero-To-Hero/ja | ULTIMATE CI/CD PIPELINE | sre and devops are same | +

github.com/iam-veeramalla/Jenkins-Zero-To-Hero/tree/main/java-maven-sonar-argocd-helm-k8s/spring-boot-app

Jenkins-Zero-To-Hero / java-maven-sonar-argocd-helm-k8s / spring-boot-app /

```
docker run -d -p 8010:8080 -t ultimate-cicd-pipeline:v1
```

Hurray !! Access the application on <http://<ip-address>:8010>

## Next Steps

Configure a Sonar Server locally

```
apt install unzip
adduser sonarqube
wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.4.0.54424.zip
unzip *
chmod -R 755 /home/sonarqube/sonarqube-9.4.0.54424
chown -R sonarqube:sonarqube /home/sonarqube/sonarqube-9.4.0.54424
cd sonarqube-9.4.0.54424/bin/linux-x86-64/
./sonar.sh start
```

Hurray !! Now you can access the SonarQube Server on <http://<ip-address>:9000>

Type here to search 27°C Mostly cloudy 15:19 03-11-2024

```
spring-boot-app git:(main) mvn clean package
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.abhishek:spring-boot-demo >-----
[INFO] Building spring-boot-demo 1.0
[INFO]   from pom.xml
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- clean:3.1.0:clean (default-clean) @ spring-boot-demo ---
[INFO]
[INFO] --- resources:3.1.0:resources (default-resources) @ spring-boot-demo ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO] Copying 3 resources
[INFO]
[INFO] --- compiler:3.8.0:compile (default-compile) @ spring-boot-demo ---
[INFO] Changes detected - recompiling the module!
```

```
spring-boot-app git:(main) x docker build -t ultimate-cicd-pipeline:v1 .
+ Building 6.5s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 581B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/adoptopenjdk/openjdk11:alpine-jre
=> [auth] adoptopenjdk/openjdk11:pull token for registry-1.docker.io
=> [internal] load build context
=> => transferring context: 19.07MB
=> [1/3] FROM docker.io/adoptopenjdk/openjdk11:alpine-jre@sha256:246e1fe322c6c5aca91db64db90c129210b581
=> CACHED [2/3] WORKDIR /opt/app
=> [3/3] COPY target/spring-boot-web.jar app.jar
=> exporting to image
=> => exporting layers
=> => writing image sha256:ce6c27fede6033db689dbfad8786890924b92abbcb761e27bd3c594d4b6eaae1
=> => naming to docker.io/library/ultimate-cicd-pipeline:v1
→ spring-boot-app git:(main) x docker run -d -p 8010:8080 -t ultimate-cicd-pipeline:v1
8ec65093671787350fbcc2c025a4828f692b47698452511eec0e03fcc2fee28a
→ spring-boot-app git:(main) x
```

```
# Test @ port 8010
```



```
# AWS EC2 instance ready .. cp the public ip address + pem file & connect to the server
```

A screenshot of the AWS EC2 Instances page. The instance summary for "i-01e981943fdad5201" is displayed. Key details include:

- Instance ID: i-01e981943fdad5201 (ultimate-cicd-demo)
- Public IPv4 address: 34.201.116.83
- Private IPv4 address: 172.31.87.213
- IPv6 address: -
- Instance state: Running
- Private IP DNS name (IPv4 only): ip-172-31-87-213.ec2.internal
- Public IP DNS: ec2-34-201-116-83.compute-1.amazonaws.com
- Hostname type: IP name: ip-172-31-87-213.ec2.internal
- Answer private resource DNS name: IPv4 (A)
- Instance type: t2.large
- Auto-assigned IP address: 34.201.116.83 [Public IP]
- VPC ID: vpc-0895d01f56cc9b25
- IAM Role: -
- Subnet ID: subnet-0d2c3126fb2626991
- IMDSv2: Optional
- Elastic IP addresses: -
- AWS Compute Optimizer finding: Opt-in to AWS Compute Optimizer for recommendations.
- Auto Scaling Group name: -

The sidebar on the left shows navigation links for EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances, Images, and Elastic Block Store.

```
→ spring-boot-app git:(main) ✘ ssh -i /Users/aveerama/Downloads/abhishek-aws-pem-file.pem ubuntu@34.201.116.83

The authenticity of host '34.201.116.83 (34.201.116.83)' can't be established.
ED25519 key fingerprint is SHA256:PuZGB703HKG1B6g2xLZYNcrXBjB8eDj64bYKfn0AngU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '34.201.116.83' (ED25519) to the list of known hosts.
```

```
# update packages + java in sudo
```

```
ubuntu@ip-172-31-87-213:~$ sudo apt update
sudo apt install openjdk-11-jre
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [108 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [728 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [147 kB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [9020 B]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [701 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [109 kB]
Get:11 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 c-n-f Metadata [588 B]
Get:12 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [715 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
61% [6 Packages store 0 B] [13 Translation-en 66.9 kB/5652 kB 1%] [12 Packages 327 kB/715 kB 46%]
```

```
# install Jenkins
```

```
ubuntu@ip-172-31-87-213:~$ curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee \
/usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
https://pkg.jenkins.io/debian binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Ign:5 https://pkg.jenkins.io/debian binary/ InRelease
Get:6 https://pkg.jenkins.io/debian binary/ Release [2044 B]
Get:7 https://pkg.jenkins.io/debian binary/ Release.gpg [833 B]
Get:8 https://pkg.jenkins.io/debian binary/ Packages [52.5 kB]
Fetched 55.3 kB in 1s (107 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... 0%
```

```
# Jenkins Server will start @ port 8080 -- @ EC2 instance configure inbound traffic rules
```

The screenshot shows the AWS EC2 Instance Details page for an instance with ID i-01e981943fdad5201. The instance is running in the N. Virginia region and is associated with a subnet subnet-0d2c3126fb2626991. The security group sg-07283a6335f133fb6 (launch-wizard-17) is attached. Inbound rules allow SSH (TCP port 22) from 0.0.0.0/0 and All traffic (All protocol, All port range) from 0.0.0.0/0. Outbound rules allow All traffic (All protocol, All port range) to 0.0.0.0/0.

EC2 > Security Groups > sg-07283a6335f133fb6 - launch-wizard-17 > Edit inbound rules

### Edit inbound rules Info

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

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0593c581c74081155	SSH	TCP	22	Custom 0.0.0.0/0	
-	All traffic	All	All	Custom 0.0.0.0/0	

**Add rule** Cancel Preview changes Save rules

```
ubuntu@ip-172-31-87-213:~$ ps -ef | grep jenkins
jenkins      4769      1 54 19:12 ?        00:00:52 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080
ubuntu      4939     1430  0 19:13 pts/0    00:00:00 grep --color=auto jenkins
ubuntu@ip-172-31-87-213:~$
```

# use ip address Public Ec2 instance & check the running Server

**Instance summary for i-01e981943fdad5201**

Updated less than a minute ago

**Instance ID**: i-01e981943fdad5201 (ultimate-cicd-demo)

**IPv6 address**: -

**Hostname type**: IP name: ip-172-31-87-213.ec2.internal

**Answer private resource DNS name**: IPv4 (A)

**Auto-assigned IP address**: 34.201.116.83 [Public IP]

**IAM Role**: -

**IMDSv2**: Optional

**Public IPv4 address copied**: 34.201.116.83 | open address

**Private IP DNS name (IPv4 only)**: ip-172-31-87-213.ec2.internal

**Instance state**: Running

**Private IPv4 addresses**: 172.31.87.213

**Instance type**: t2.large

**VPC ID**: vpc-0895d01f56cc9b25

**Subnet ID**: subnet-0d2c3126fb2626991

**Elastic IP addresses**: -

**AWS Compute Optimizer finding**: Opt-in to AWS Compute Optimizer for recommendations. | Learn more

**Auto Scaling Group name**: -

**Details** | **Security** | **Networking** | **Storage** | **Status checks** | **Monitoring** | **Tags**

JENKINS IS NOW UP AND RUNNING! Implementation with detailed notes | BEST PRACTICES PROJECT

Getting Started

## Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

**Administrator password**

```
# cp the initial password
```

```
ubuntu@ip-172-31-87-213:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
bcfd5a47692141eabaefb1884f018390
ubuntu@ip-172-31-87-213:~$
```

# Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

This connection is not secure. Logins entered here could be compromised. [Learn More](#)

# Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding	OWASP Markup Formatter ** Structs ** Token Macro Build Timeout ** Credentials ** Trilead API ** SSH Credentials ** Pipeline: Step API ** Plain Credentials Credentials Binding ** SCM API ** Pipeline: API ** commons-lang3 v3.x Jenkins API
✓ Timestamper	⌚ Workspace Cleanup	⌚ Ant	⌚ Gradle	Timestamper ** Caffeine API ** Script Security ** JAXB ** SnakeYAML API ** Jackson 2 API ** commons-text API ** Pipeline: Supporting APIs ** Plugin Utilities API ** Font Awesome API
⌚ Pipeline	⌚ GitHub Branch Source	⌚ Pipeline: GitHub Groovy Libraries	⌚ Pipeline: Stage View	
⌚ Git	⌚ SSH Build Agents	⌚ Matrix Authorization Strategy	⌚ PAM Authentication	
⌚ LDAP	⌚ Email Extension	⌚ Mailer		** – required dependency

# Jenkins is ready!

You have skipped the **setup of an admin user**.

To log in, use the username: "admin" and the administrator password you used to access the setup wizard.

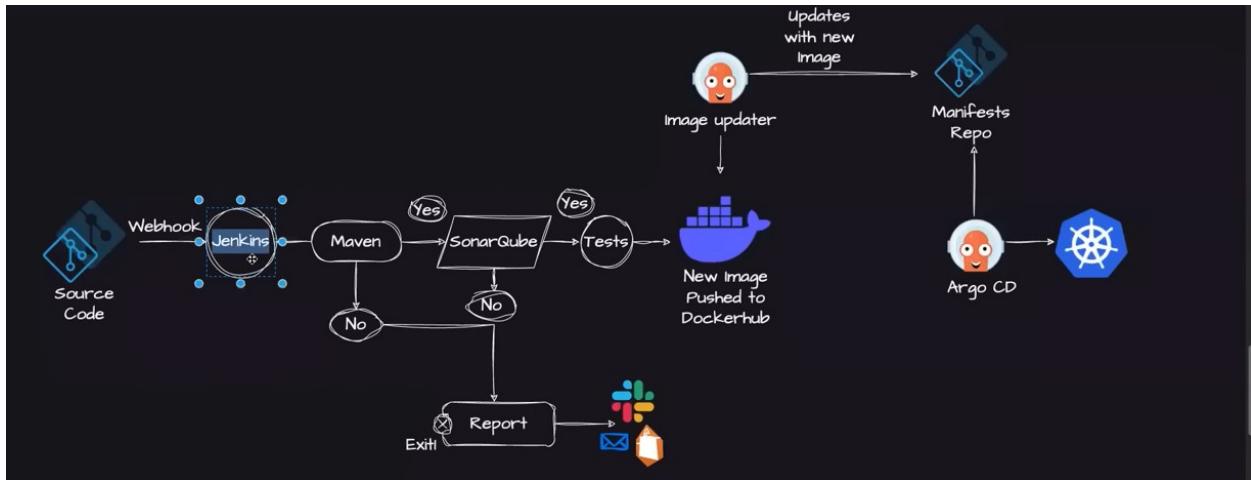
+

Your Jenkins setup is complete.

[Start using Jenkins](#)

The screenshot shows a Firefox browser window with multiple tabs open. The active tab is the Jenkins dashboard at 34.201.116.83:8080. The dashboard has a dark header with the Jenkins logo and the word 'Jenkins'. Below the header, there's a navigation bar with links like 'Dashboard', 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. A 'Build Queue' section shows 'No builds in the queue.' A 'Build Executor Status' section shows '1 Idle' and '2 Idle'. The main content area features a 'Welcome to Jenkins!' message, a 'Start building your software project' button, and sections for 'Create a job', 'Set up a distributed build', 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'. The status bar at the bottom right indicates 'Tue 4 Apr 12:45 AM'.

# next Step



# now write a Pipeline using jenkin ui - click on newitem –select - pipeline

Dashboard >

- + New Item
- People
- Build History
- Manage Jenkins
- My Views

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Build Queue

No builds in the queue.

Build Executor Status

1 Idle  
2 Idle

Create a job →

Set up a distributed build

Set up an agent →  
Configure a cloud →  
Learn more about distributed builds →

ultimate-demo  
» Required field

**Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

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

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

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

**Multibranch Pipeline**  
Creates a set of Pipeline projects according to detected branches in one SCM repository.

**Organization Folder**  
Creates a set of multibranch project subfolders by scanning for repositories.  
**OK**

## # Groovy script or Go to Repository

### Configure

General

Advanced Project Options

Pipeline

Pipeline script

Script ?

1

try sample Pipeline... ▾

Use Groovy Sandbox ?

Pipeline Syntax

Save

Apply

Dashboard > ultimate-demo > Configuration

## Configure

Definition

- Pipeline script
- Pipeline script from SCM

General

Advanced Project Options

Pipeline

Script ?

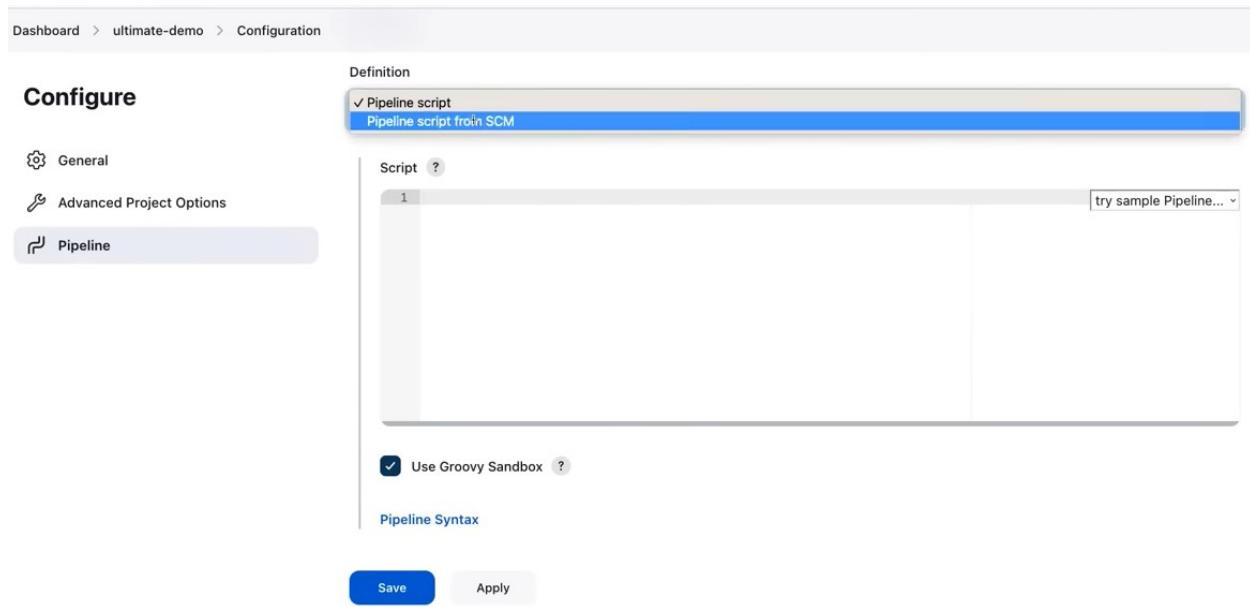
try sample Pipeline... ▾

1

Use Groovy Sandbox ?

Pipeline Syntax

Save Apply



## # Jenkinsfile

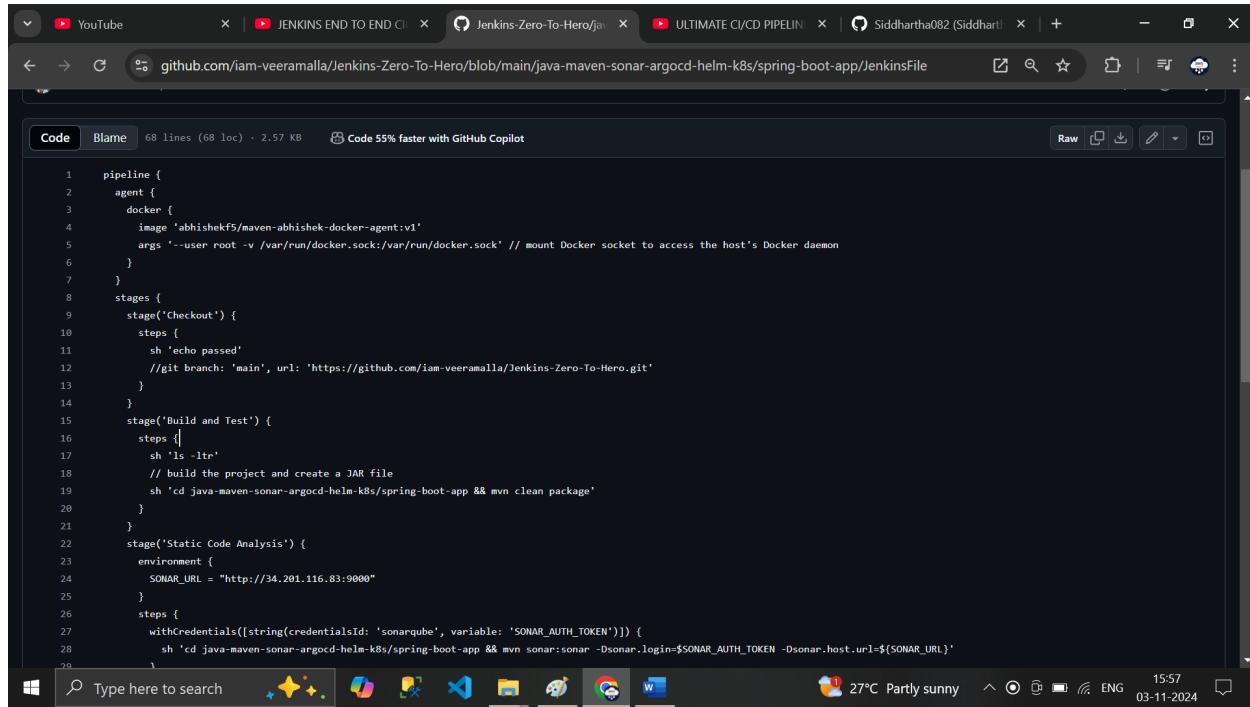
YouTube JENKINS END TO END CI Jenkins-Zero-To-Hero/jenkinsfile ULTIMATE CI/CD PIPELINE Siddhartha082 (Siddhart) - +

github.com/iam-veeramalla/Jenkins-Zero-To-Hero/blob/main/java-maven-sonar-argocd-helm-k8s/spring-boot-app/Jenkinsfile

Code Blame 68 lines (68 loc) · 2.57 KB Code 55% faster with GitHub Copilot Raw

```
1 pipeline {
2     agent {
3         docker {
4             image 'abhishekfs/maven-abhishek-docker-agent:v1'
5             args '--user root -v /var/run/docker.sock:/var/run/docker.sock' // mount Docker socket to access the host's Docker daemon
6         }
7     }
8     stages {
9         stage('Checkout') {
10            steps {
11                sh 'echo passed'
12                //git branch: 'main', url: 'https://github.com/iam-veeramalla/Jenkins-Zero-To-Hero.git'
13            }
14        }
15        stage('Build and Test') {
16            steps {
17                sh 'ls -ltr'
18                // build the project and create a JAR file
19                sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && mvn clean package'
20            }
21        }
22        stage('Static Code Analysis') {
23            environment {
24                SONAR_URL = "http://34.201.116.83:9000"
25            }
26            steps {
27                withCredentials([string(credentialsId: 'sonarqube', variable: 'SONAR_AUTH_TOKEN')])
28                sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && mvn sonar:sonar -Dsonar.login=${SONAR_AUTH_TOKEN} -Dsonar.host.url=${SONAR_URL}'
29            }
30        }
31    }
32 }
```

Type here to search



```

steps {
    script {
        sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && docker build -t ${DOCKER_IMAGE} .'
        def dockerImage = docker.image("${DOCKER_IMAGE}")
        docker.withRegistry('https://index.docker.io/v1/', "docker-cred") {
            dockerImage.push()
        }
    }
}
stage('Update Deployment File') {
    environment {
        GIT_REPO_NAME = "Jenkins-Zero-to-Hero"
        GIT_USER_NAME = "iam-veeramalla"
    }
    steps {
        withCredentials([string(credentialsId: 'github', variable: 'GITHUB_TOKEN')]) {
            sh '''
                git config user.email "abhishek.xyz@gmail.com"
                git config user.name "Abhishek Veeramalla"
                BUILD_NUMBER=${BUILD_NUMBER}
                sed -i "/${replaceImageTag}/${BUILD_NUMBER}/g" java-maven-sonar-argocd-helm-k8s/spring-boot-app-manifests/deployment.yaml
                git add java-maven-sonar-argocd-helm-k8s/spring-boot-app-manifests/deployment.yaml
                git commit -m "Update deployment image to version ${BUILD_NUMBER}"
                git push https://${GITHUB_TOKEN}@github.com/${GIT_USER_NAME}/${GIT_REPO_NAME} HEAD:main
            '''
        }
    }
}

```

**Definition**

General

Advanced Project Options

Pipeline

**SCM**

Git

Repositories

Repository URL:

Please enter Git repository.

Credentials:

Add

**Configure**

General

Advanced Project Options

Pipeline

**Repositories**

Repository URL:

Credentials:

Add

Advanced

Add Repository

**Branches to build**

Branch Specifier (blank for 'any')

**General**

**Advanced Project Options**

**Pipeline**

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

\*/main

Add Branch

Repository browser ?

(Auto)

Additional Behaviours

Add ▾

**General**

**Advanced Project Options**

**Pipeline**

Add Branch

Repository browser ?

(Auto)

Additional Behaviours

Add ▾

Script Path ?

+ java-maven-sonar-argocd-helm-k8s/spring-boot-app/JenkinsFile

Lightweight checkout ?

[Pipeline Syntax](#)

Dashboard > ultimate-demo >

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

## Pipeline ultimate-demo

Add description Disable Project

**Stage View**

No data available. This Pipeline has not yet run.

**Permalinks**

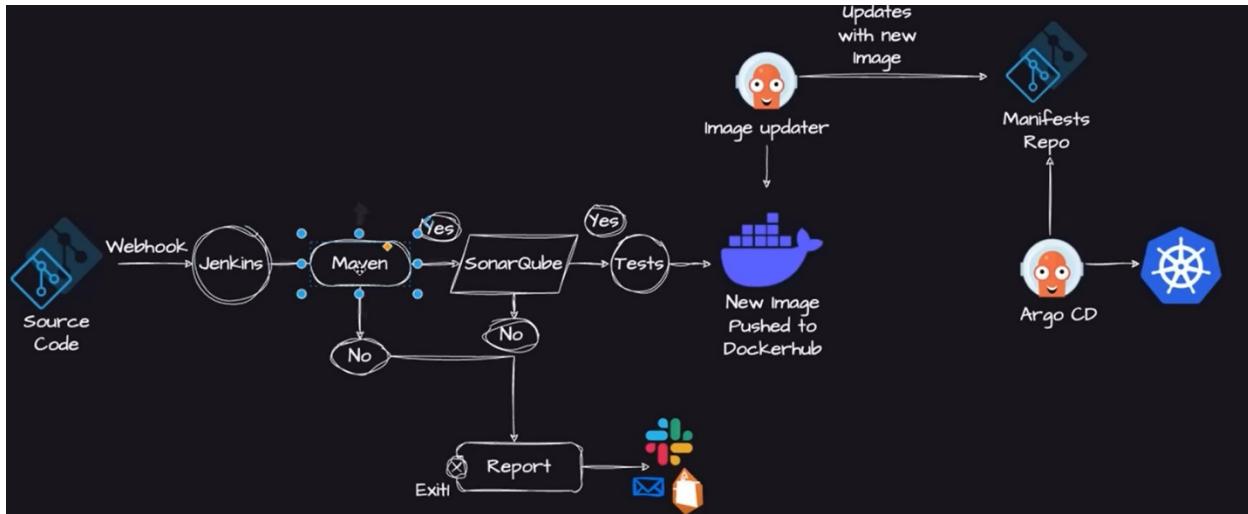
Build History trend / Filter builds... No builds Atom feed for all Atom feed for failures

---- install Docker Container in the Jenkins – using Docker Plugins

-- note Docker container default has maven---

The screenshot shows the Jenkins 'Plugins' page under 'Manage Jenkins'. The left sidebar includes 'Updates', 'Available plugins' (which is selected), 'Installed plugins', 'Advanced settings', and 'Download progress'. A search bar at the top right contains the text 'docker pipeline'. The main area displays a table for the 'Docker Pipeline' plugin, version 563.vd5d2e5c4007f. The table columns are 'Install', 'Name', and 'Released'. The plugin is marked as installed. A tooltip indicates it is up for adoption. The 'Released' date is 4 mo 2 days ago.

The screenshot shows the Jenkins 'Download progress' page under 'Manage Jenkins'. The left sidebar includes 'Updates', 'Available plugins' (selected), 'Installed plugins', 'Advanced settings', and 'Download progress' (selected). The main area displays a table titled 'Download progress' with two sections: 'Preparation' and a list of various Jenkins components. The preparation section lists 'Checking internet connectivity', 'Checking update center connectivity', and 'Success'. The component list includes Ionicons API, Folders, JavaBeans Activation Framework (JAF) API, JavaMail API, bouncycastle API, Instance Identity, Mina SSHD API :: Common, Mina SSHD API :: Core, SSH server, OWASP Markup Formatter, Structs, Token Macro, Build Timeout, Credentials, Trilead API, and several entries ending in 'done' or '...'. All items show a green checkmark and the word 'Success'.



```
# now install SonarQube
```

```
# Go back to Jenkins – Plugins
```

Dashboard > Manage Jenkins

**Manage Jenkins**

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

[Set up agent](#) [Set up cloud](#) [Dismiss](#)

**System Configuration**

- System**: Configure global settings and paths.
- Tools**: Configure tools, their locations and automatic installers.
- Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

**Build Queue**: No builds in the queue.

**Build Executor Status**: 1 Idle, 2 Idle

**Nodes and Clouds**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

**Security**

Firefox - Jenkins - Plugins - Tue 4 Apr 12:58 AM

34.201.116.83:8080/manage/pluginManager/available

## Plugins

Available plugins

Search: sonar

Install	Name	Released
<input checked="" type="checkbox"/>	SonarQube Scanner 2.15 External Site/Tool Integrations Build Reports	4 mo 12 days ago
<input type="checkbox"/>	Sonar Quality Gates 1.3.1 Fails the build whenever the Quality Gates criteria in the Sonar 5.6+ analysis aren't met (the project Quality Gates status is different than "Passed")  Warning: This plugin version may not be safe to use. Please review the following security notices: + • Credentials transmitted in plain text	4 yr 7 mo ago

Install without restart   Download now and install after restart   Update information obtained: 15 min ago   Check now

Firefox - Jenkins - Plugins - Tue 4 Apr 12:58 AM

34.201.116.83:8080/manage/pluginManager/updates/

## Download progress

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

Ionicons API	Success
Folders	Success
JavaBeans Activation Framework (JAF) API	Success
JavaMail API	Success
bouncycastle API	Success
Instance Identity	Success
Mina SSHD API :: Common	Success
Mina SSHD API :: Core	Success
SSH server	Success
OWASP Markup Formatter	Success
Structs	Success
Token Macro	Success
Build Timeout	Success
Credentials	Success
Trilead API	Success
...	...

## Next Steps

### Configure a Sonar Server locally

```
apt install unzip  
adduser sonarqube  
wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.4.0.54424.zip  
unzip *  
chmod -R 755 /home/sonarqube/sonarqube-9.4.0.54424  
chown -R sonarqube:sonarqube /home/sonarqube/sonarqube-9.4.0.54424  
cd sonarqube-9.4.0.54424/bin/linux-x86-64/  
../sonar.sh start
```

Hurray !! Now you can access the `SonarQube Server` on `http://<ip-address>:9000`

```
ubuntu@ip-172-31-87-213:~$ adduser sonarqube  
  
adduser: Only root may add a user or group to the system.  
ubuntu@ip-172-31-87-213:~$ sudo su -  
root@ip-172-31-87-213:~# adduser sonarqube  
^C  
root@ip-172-31-87-213:~# adduser sonarqube  
Adding user `sonarqube' ...  
Adding new group `sonarqube' (1001) ...  
Adding new user `sonarqube' (1001) with group `sonarqube' ...  
Creating home directory `/home/sonarqube' ...  
Copying files from `/etc/skel' ...  
New password:  
Retype new password:  
passwd: password updated successfully  
Changing the user information for sonarqube  
Enter the new value, or press ENTER for the default  
    Full Name []:  
    Room Number []:  
    Work Phone []:  
    Home Phone []:  
    Other []:  
Is the information correct? [Y/n] y  
root@ip-172-31-87-213:~#
```

```
root@ip-172-31-87-213:~# sudo su - sonarqube  
sonarqube@ip-172-31-87-213:~$
```

# Download sonar Binary

```
root@ip-172-31-87-213:~# sudo su - sonarqube
sonarqube@ip-172-31-87-213:~$ wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.4.0.54424.zip
--2023-04-03 19:30:59-- https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.4.0.54424.zip
Resolving binaries.sonarsource.com (binaries.sonarsource.com)... 18.160.10.89, 18.160.10.93, 18.160.10.111, ...
Connecting to binaries.sonarsource.com (binaries.sonarsource.com)|18.160.10.89|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 287666040 (274M) [binary/octet-stream]
Saving to: 'sonarqube-9.4.0.54424.zip'

sonarqube-9.4.0.54424.zip 100%[=====] 274.34M 102MB/s in 2.7s

2023-04-03 19:31:02 (102 MB/s) - 'sonarqube-9.4.0.54424.zip' saved [287666040/287666040]

sonarqube@ip-172-31-87-213:~$
```

```
sonarqube@ip-172-31-87-213:~$ unzip *
Command 'unzip' not found, but can be installed with:
apt install unzip
Please ask your administrator.
sonarqube@ip-172-31-87-213:~$ logout
root@ip-172-31-87-213:~# apt install unzip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  zip
The following NEW packages will be installed:
  unzip
0 upgraded, 1 newly installed, 0 to remove and 6 not upgraded.
Need to get 174 kB of archives.
After this operation, 385 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 unzip amd64 6.0-26ubuntu3.1 [174 kB]
Fetched 174 kB in 0s (9201 kB/s)
```

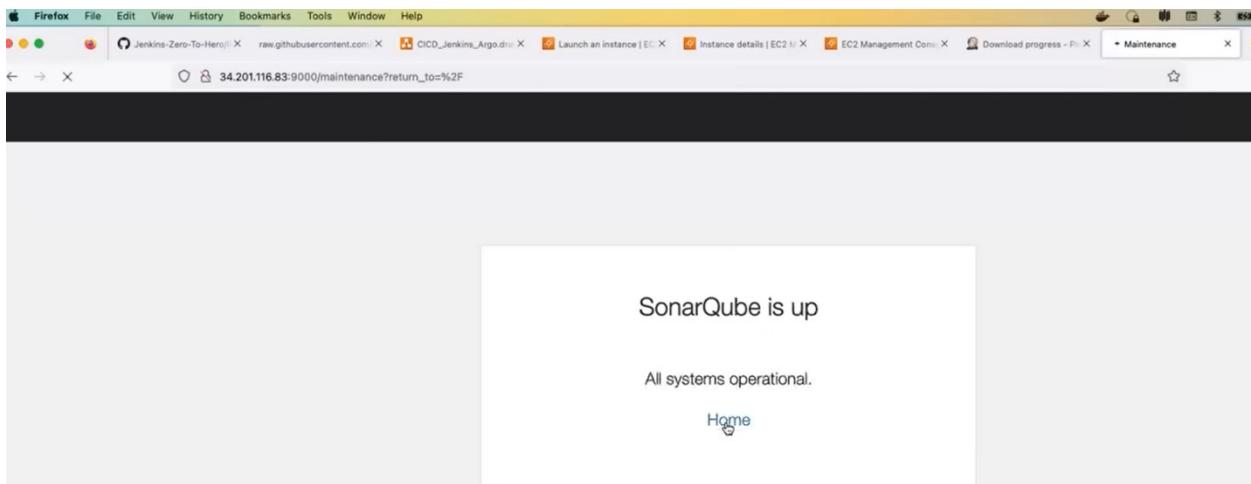
```
No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-172-31-87-213:~# sudo su -
root@ip-172-31-87-213:~# sudo su - sonarqube
sonarqube@ip-172-31-87-213:~$ ls
sonarqube-9.4.0.54424.zip
sonarqube@ip-172-31-87-213:~$ unzip *
```

```
creating: sonarqube-9.4.0.54424/elasticsearch/
creating: sonarqube-9.4.0.54424/elasticsearch/lib/
inflating: sonarqube-9.4.0.54424/elasticsearch/lib/elasticsearch-7.17.1.jar
inflating: sonarqube-9.4.0.54424/elasticsearch/lib/java-version-checker-7.17.1.jar
inflating: sonarqube-9.4.0.54424/elasticsearch/lib/elasticsearch-launchers-7.17.1.jar
inflating: sonarqube-9.4.0.54424/elasticsearch/lib/elasticsearch-x-content-7.17.1.jar
inflating: sonarqube-9.4.0.54424/elasticsearch/lib/elasticsearch-lz4-7.17.1.jar
inflating: sonarqube-9.4.0.54424/elasticsearch/lib/elasticsearch-cli-7.17.1.jar
inflating: sonarqube-9.4.0.54424/elasticsearch/lib/elasticsearch-core-7.17.1.jar
inflating: sonarqube-9.4.0.54424/elasticsearch/lib/elasticsearch-secure-sm-7.17.1.jar
inflating: sonarqube-9.4.0.54424/elasticsearch/lib/elasticsearch-geo-7.17.1.jar
inflating: sonarqube-9.4.0.54424/elasticsearch/lib/lucene-core-8.11.1.jar
inflating: sonarqube-9.4.0.54424/elasticsearch/lib/lucene-analyzers-common-8.11.1.jar
inflating: sonarqube-9.4.0.54424/elasticsearch/lib/lucene-backward-codecs-8.11.1.jar
inflating: sonarqube-9.4.0.54424/elasticsearch/lib/lucene-grouping-8.11.1.jar
inflating: sonarqube-9.4.0.54424/elasticsearch/lib/lucene-highlighter-8.11.1.jar
inflating: sonarqube-9.4.0.54424/elasticsearch/lib/lucene-join-8.11.1.jar
inflating: sonarqube-9.4.0.54424/elasticsearch/lib/lucene-memory-8.11.1.jar
inflating: sonarqube-9.4.0.54424/elasticsearch/lib/lucene-misc-8.11.1.jar
inflating: sonarqube-9.4.0.54424/elasticsearch/lib/lucene-queries-8.11.1.jar
inflating: sonarqube-9.4.0.54424/elasticsearch/lib/lucene-queryparser-8.11.1.jar
inflating: sonarqube-9.4.0.54424/elasticsearch/lib/lucene-sandbox-8.11.1.jar
```

```
sonarqube@ip-172-31-87-213:~$ ls
sonarqube-9.4.0.54424  sonarqube-9.4.0.54424.zip
sonarqube@ip-172-31-87-213:~$ cd sonarqube-9.4.0.54424
sonarqube@ip-172-31-87-213:~/sonarqube-9.4.0.54424$ cd ..
sonarqube@ip-172-31-87-213:~$ chmod -R 755 /home/sonarqube/sonarqube-9.4.0.54424
sonarqube@ip-172-31-87-213:~$ chown -R sonarqube:sonarqube /home/sonarqube/sonarqube-9.4.0.54424
sonarqube@ip-172-31-87-213:~$ cd sonarqube-9.4.0.54424/bin/
jsw-license/          linux-x86-64/          macosx-universal-64/ windows-x86-64/
sonarqube@ip-172-31-87-213:~$ cd sonarqube-9.4.0.54424/bin/linux-x86-64/
sonarqube@ip-172-31-87-213:~/sonarqube-9.4.0.54424/bin/linux-x86-64$ ls
lib  sonar.sh  wrapper
sonarqube@ip-172-31-87-213:~/sonarqube-9.4.0.54424/bin/linux-x86-64$ ./sonar.sh start
Starting SonarQube...
Started SonarQube.
sonarqube@ip-172-31-87-213:~/sonarqube-9.4.0.54424/bin/linux-x86-64$ █
```

# note Sonar Server Starts @ Port 9000



# First time Login + password – is Admin.. later Change the password

Update your password

This account should not use the default password.

Enter a new password

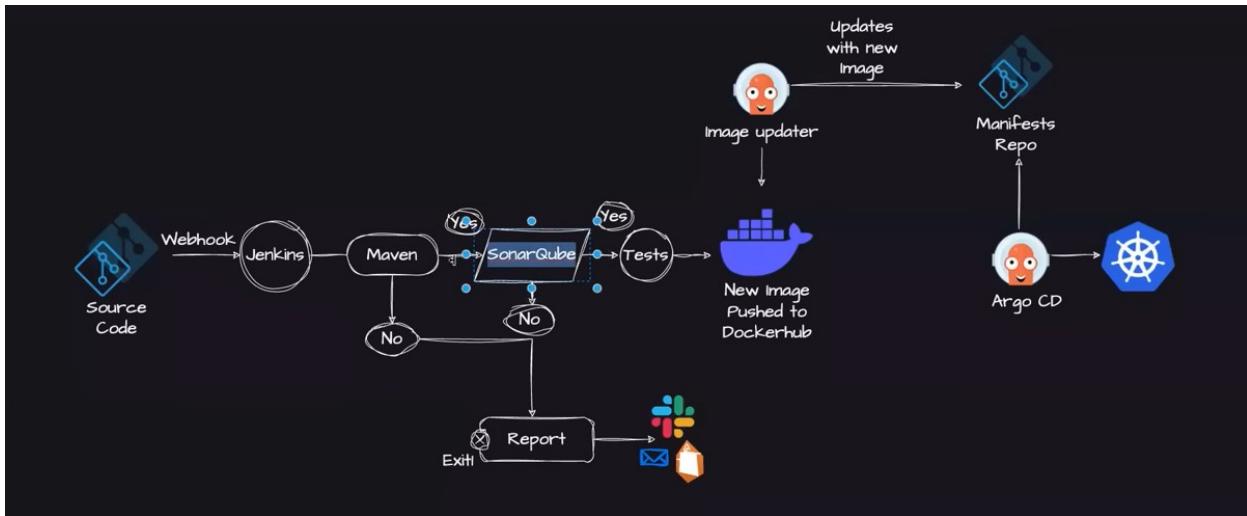
All fields marked with \* are required

**Old Password \***

This connection is not secure. Logins entered here could be compromised. Learn More

**Confirm Password \***

**Update**



# now Jenkins would talk / Connect to SonarCube.. – Go to my Account

# Click on Security – Generate New Tokens

## Tokens

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

### Generate Tokens

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

Copy 0176b6f4169ba05d54a14c116d735792e42797c7

Name	Last use	Created
jenkins	Never	April 4, 2023

Revoke

Enter a new password

Get the most out of SonarQube  
Take advantage of the whole ecosystem

# Cp the Token ID .. Go to Jenkins .. click on Manage Jenkins – Click on Manage Credentials

Dashboard > Manage Jenkins

Build History Manage Jenkins My Views

Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.

Set up agent Set up cloud Dismiss

System Configuration

System Configure global settings and paths.

Tools Configure tools, their locations and automatic installers.

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

Build Queue No builds in the queue.

Build Executor Status 1 idle 2 idle

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

Security

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

Manage Credentials Configure credentials

Configure Credential Providers Configure the credential providers and types

Users

The screenshot shows the Jenkins 'Credentials' page. At the top, there's a navigation bar with 'Dashboard > Manage Jenkins > Credentials'. The main title is 'Credentials'. Below it, there's a table header with columns: T, P, Store ↓, Domain, ID, and Name. A search bar at the top right contains 'Search (⌘+K)' and other icons. The main content area is titled 'Stores scoped to Jenkins' and shows a single entry: 'System' under 'Domains' with '(global)' next to it. Below the table, there are size icons: S, M, and L.

# Click on System

The screenshot shows the 'System' credentials page. The title is 'System' and there's a 'Add domain' button. The table has columns: Domain ↓ and Description. A single entry is shown: 'Global credentials (unrestricted)' with a description: 'Credentials that should be available irrespective of domain specification to requirements matching.' Below the table are size icons: S, M, and L.

## New credentials

Kind

Secret text

Scope ?

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

Secret

.....

ID ?

sonarqube

Description ?

Create

This part of the screenshot shows the 'New credentials' form. It includes fields for 'Kind' (set to 'Secret text'), 'Scope' (set to 'Global'), 'Secret' (containing several dots), 'ID' (set to 'sonarqube'), and 'Description' (an empty field). At the bottom is a blue 'Create' button.

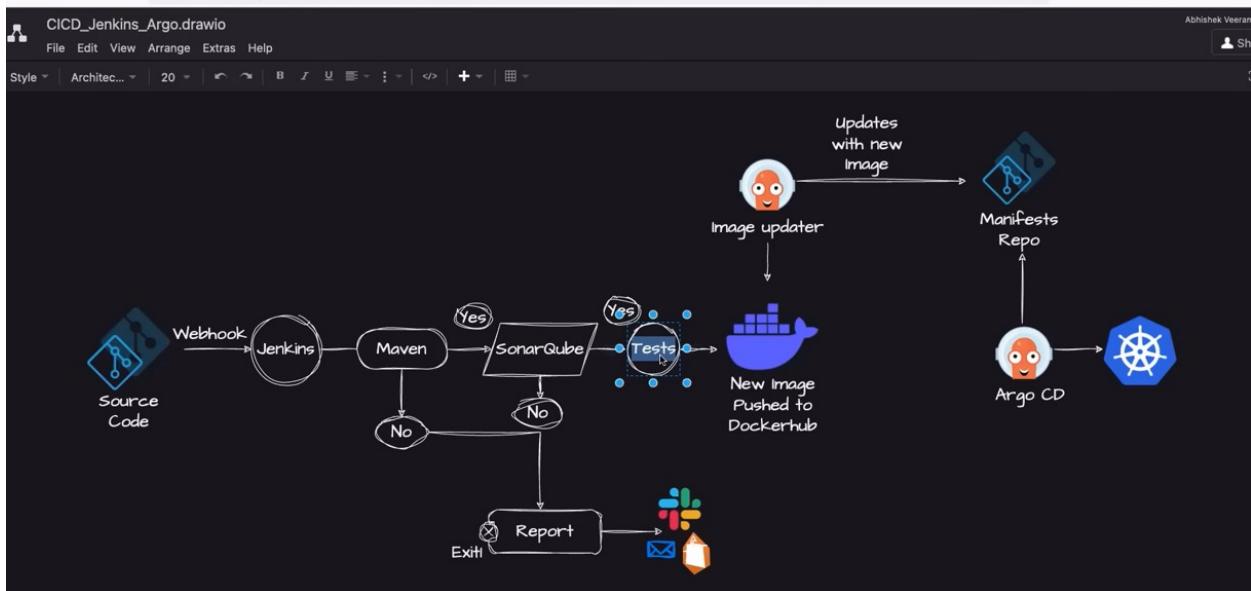
Global credentials (unrestricted)

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

ID	Name	Kind	Description
sonarqube	sonarqube	Secret text	

Icon: S M L

# Finally SonarQube Configuration Completed



# now install Docker in EC2 Instance – root user

```

sonarqube@ip-172-31-87-213:~/sonarqube-9.4.0.54424/bin/linux-x86-64$ sudo su -
[sudo] password for sonarqube:
sudo: a password is required
sonarqube@ip-172-31-87-213:~/sonarqube-9.4.0.54424/bin/linux-x86-64$
logout
root@ip-172-31-87-213:~# cl

```

## Docker Slave Configuration

Run the below command to Install Docker

```
sudo apt update  
sudo apt install docker.io
```

**Grant Jenkins user and Ubuntu user permission to docker deamon.**

```
sudo su -  
usermod -aG docker jenkins  
usermod -aG docker ubuntu  
systemctl restart docker
```

Once you are done with the above steps, it is better to restart Jenkins.

```
http://<ec2-instance-public-ip>:8080/restart
```

The docker agent configuration is now successful.

```
root@ip-172-31-87-213:~# sudo apt install docker.io  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan  
Suggested packages:  
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils  
The following NEW packages will be installed:  
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan  
0 upgraded, 8 newly installed, 0 to remove and 6 not upgraded.  
Need to get 72.4 MB of archives.  
After this operation, 287 MB of additional disk space will be used.  
Do you want to continue? [Y/n] ■
```

**# grant Permissions**

```
root@ip-172-31-87-213:~# usermod -aG docker jenkins  
usermod -aG docker ubuntu  
systemctl restart docker ■
```

**# Restart - Jenkins**

34.201.116.83:8080/restart

Jenkins

Manage Jenkins [Jenkins] — http://34.201.116.83:8080/manage/

Dashboard > Manage

System » Global credentials (unrestricted) [Jenkins] — http://34.201.116.83:8080/manage/credentials/store/system/domain/\_/

Download progress - Plugins [Jenkins] — http://34.201.116.83:8080/manage/pluginManager/updates/

Available plugins - Plugins [Jenkins] — http://34.201.116.83:8080/manage/pluginManager/available

Updates - Plugins [Jenkins] — http://34.201.116.83:8080/manage/pluginManager/

System » Global credentials (unrestricted) » sonarqube [Jenkins] — http://34.201.116.83:8080/manage/credentials/store/system/domain/\_/credential/sonarqube/

New credentials [Jenkins] — http://34.201.116.83:8080/manage/credentials/store/system/domain/\_/newCredentials

Jenkins » Credentials [Jenkins] — http://34.201.116.83:8080/manage/credentials/

+ Add domain

This time, search with: G b D W Y S C

Icon: S M L

+



Please wait while Jenkins is restarting ...

Your browser will reload automatically when Jenkins is ready.



## Welcome to Jenkins!

admin

.....

Keep me signed in

**Sign in**

Dashboard >

+ New Item      Add description

People      All +

Build History

Manage Jenkins

My Views

S	W	Name ↓	Last Success	Last Failure	Last Duration
...	...	ultimate-demo	N/A	N/A	N/A

Build Queue      ▾

No builds in the queue.

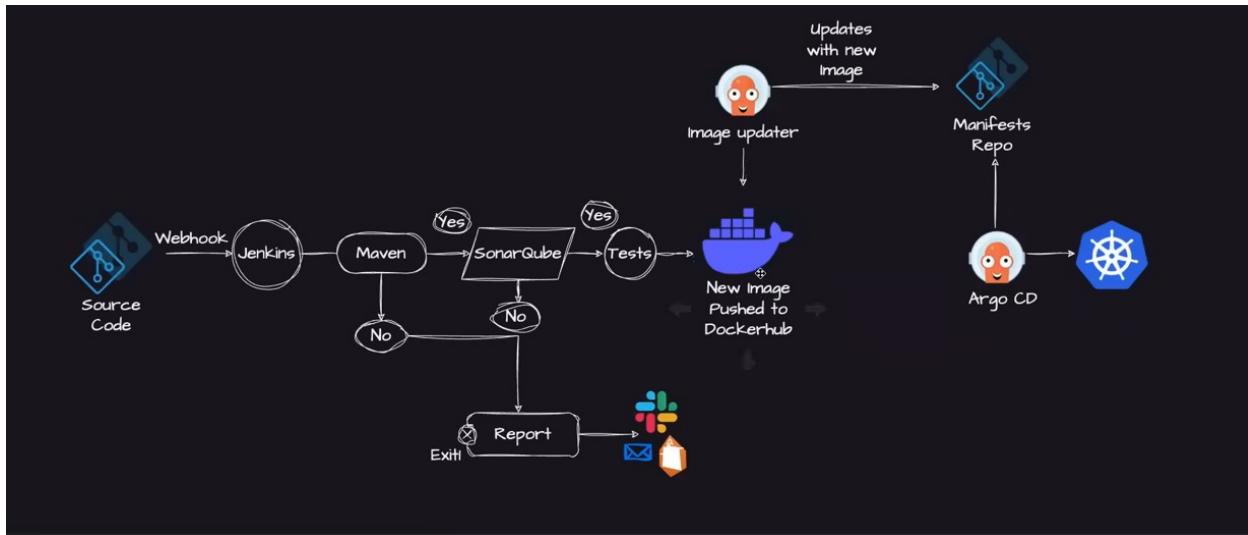
Icon: S M L      Icon legend      Atom feed for all      Atom feed for failures      Atom feed for just latest builds

Build Executor Status      ▾

1 Idle  
2 Idle

REST API      Jenkins 2.397

# Docker step is Configured



# Final step .. install ArgoCD + Kubernetes Cluster ...

```

→ ~ git:(argo-all-namespaces) ✘ minikube start --memory=4098 --driver=hyperkit
minikube v1.25.2 on Darwin 13.2.1
⠄ Using the hyperkit driver based on user configuration
👍 Starting control plane node minikube in cluster minikube
🔥 Creating hyperkit VM (CPUs=2, Memory=4098MB, Disk=20000MB) ...
🌐 Preparing Kubernetes v1.23.3 on Docker 20.10.12 ...
  ■ kubelet.housekeeping-interval=5m
  ■ Generating certificates and keys ...
  ■ Booting up control plane ...
  ■ Configuring RBAC rules ...
🌐 Verifying Kubernetes components...
  ■ Using image gcr.io/k8s-minikube/storage-provisioner:v5
⭐ Enabled addons: storage-provisioner, default-storageclass

❗ /usr/local/bin/kubectl is version 1.21.3, which may have incompatibilities with Kubernetes 1.23.3.
  ■ Want kubectl v1.23.3? Try 'minikube kubectl -- get pods -A'
⚡ Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
→ ~ git:(argo-all-namespaces) ✘

```

<https://operatorhub.io/operator/argocd-operator>

The screenshot shows the Argo CD operator page on OperatorHub.io. The page features a large blue header with the Argo CD logo and a brief description: "Argo CD is a declarative, GitOps continuous delivery tool for Kubernetes.". Below the header, there's a breadcrumb navigation: "Home > Argo CD". On the right side, there's a prominent "Install" button. To the right of the button, there are dropdown menus for "CHANNEL" (set to "alpha"), "VERSION" (set to "0.12.0 (Current)"), and "CAPABILITY LEVEL" (set to "Basic Install"). The main content area contains an "Overview" section with a brief description of the operator's role in managing the Argo CD lifecycle. At the bottom of the page, there's a "Type here to search" bar and a Windows taskbar at the bottom.

This screenshot shows the "Install on Kubernetes" modal window from the Argo CD operator page. The modal is titled "Install on Kubernetes" and contains three numbered steps: 1. Install Operator Lifecycle Manager (OLM), 2. Install the operator by running a command, and 3. Watch your operator come up. Each step includes a code snippet and a "What happens when I execute this command?" link. The modal has a dark background and a light foreground. The "Install" button is located at the bottom right of the modal. The rest of the Argo CD operator page is visible in the background, along with the Windows taskbar at the bottom.

## Install on Kubernetes

1. Install Operator Lifecycle Manager (OLM), a tool to help manage the Operators running on your cluster.

```
$ curl -sL https://github.com/operator-framework/operator-lifecycle-manager/releases/download/v0.29.0/install.sh | bash -s v0.29.0
```

2. Install the operator by running the following command:

[What happens when I execute this command?](#)

```
$ kubectl create -f https://operatorhub.io/install/argocd-operator.yaml
```

This Operator will be installed in the "operators" namespace and will be usable from all namespaces in the cluster.

3. After install, watch your operator come up using next command.

```
$ kubectl get csv -n operators
```

To use it, checkout the custom resource definitions (CRDs) introduced by this operator to start using it.

```
→ ~ git:(argo-all-namespaces) ✘ curl -sL https://github.com/operator-framework/operator-lifecycle-manager/releases/download/v0.24.0/install.sh | bash -s v0.24.0
customresourcedefinition.apiextensions.k8s.io/catalogsources.operators.coreos.com created
customresourcedefinition.apiextensions.k8s.io/clusterserviceversions.operators.coreos.com created
customresourcedefinition.apiextensions.k8s.io/installplans.operators.coreos.com created
customresourcedefinition.apiextensions.k8s.io/olmconfigs.operators.coreos.com created
customresourcedefinition.apiextensions.k8s.io/operatorconditions.operators.coreos.com created
customresourcedefinition.apiextensions.k8s.io/operatorgroups.operators.coreos.com created
customresourcedefinition.apiextensions.k8s.io/operators.operators.coreos.com created
customresourcedefinition.apiextensions.k8s.io/subscriptions.operators.coreos.com created
customresourcedefinition.apiextensions.k8s.io/catalogsources.operators.coreos.com condition met
customresourcedefinition.apiextensions.k8s.io/clusterserviceversions.operators.coreos.com condition met
customresourcedefinition.apiextensions.k8s.io/installplans.operators.coreos.com condition met
customresourcedefinition.apiextensions.k8s.io/olmconfigs.operators.coreos.com condition met
customresourcedefinition.apiextensions.k8s.io/operatorconditions.operators.coreos.com condition met
```

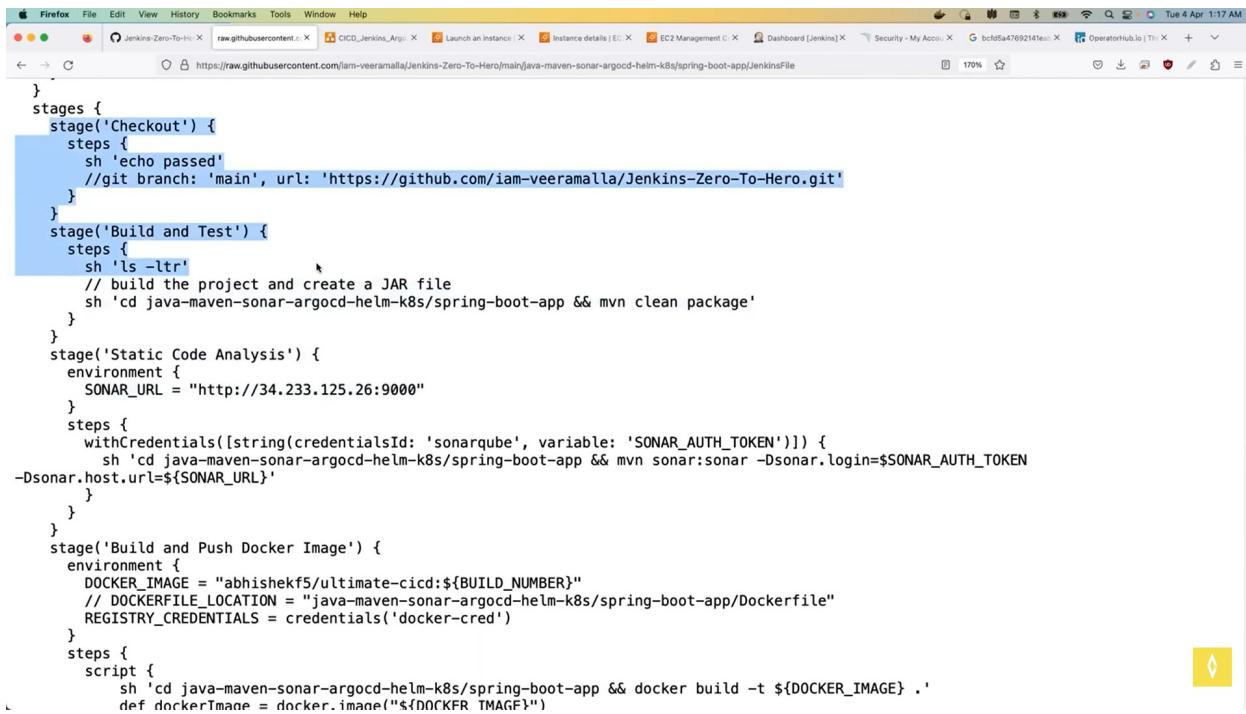
```

customresourcedefinition.apiextensions.k8s.io/operators.operators.coreos.com condition met
customresourcedefinition.apiextensions.k8s.io/subscriptions.operators.coreos.com condition met
namespace/olm created
namespace/operators created
serviceaccount/olm-operator-serviceaccount created
clusterrole.rbac.authorization.k8s.io/system:controller:operator-lifecycle-manager created
clusterrolebinding.rbac.authorization.k8s.io/olm-operator-binding-olm created
olmconfig.operators.coreos.com/cluster created
deployment.apps/olm-operator created
deployment.apps/catalog-operator created
clusterrole.rbac.authorization.k8s.io/aggregate-olm-edit created
clusterrole.rbac.authorization.k8s.io/aggregate-olm-view created
operatorgroup.operators.coreos.com/global-operators created
operatorgroup.operators.coreos.com/olm-operators created
clusterserviceversion.operators.coreos.com/packageserver created
catalogsource.operators.coreos.com/operatorhubio-catalog created
Waiting for deployment "olm-operator" rollout to finish: 0 of 1 updated replicas are available...

deployment "olm-operator" successfully rolled out
deployment "catalog-operator" successfully rolled out

```

Package server phase: Installing



```

}
stages {
    stage('Checkout') {
        steps {
            sh 'echo passed'
            //git branch: 'main', url: 'https://github.com/iam-veeramalla/Jenkins-Zero-To-Hero.git'
        }
    }
    stage('Build and Test') {
        steps {
            sh 'ls -ltr'
            // build the project and create a JAR file
            sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && mvn clean package'
        }
    }
    stage('Static Code Analysis') {
        environment {
            SONAR_URL = "http://34.233.125.26:9000"
        }
        steps {
            withCredentials([string(credentialsId: 'sonarqube', variable: 'SONAR_AUTH_TOKEN')]) {
                sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && mvn sonar:sonar -Dsonar.login=$SONAR_AUTH_TOKEN -Dsonar.host.url=${SONAR_URL}'
            }
        }
    }
    stage('Build and Push Docker Image') {
        environment {
            DOCKER_IMAGE = "abhishekf5/ultimate-cicd:${BUILD_NUMBER}"
            // DOCKERFILE_LOCATION = "java-maven-sonar-argocd-helm-k8s/spring-boot-app/Dockerfile"
            REGISTRY_CREDENTIALS = credentials('docker-cred')
        }
        steps {
            script {
                sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && docker build -t ${DOCKER_IMAGE} .'
                def dockerImageName = docker.image("${DOCKER_IMAGE}")

```

```

→ ~ git:(argo-all-namespaces) ✘ kubectl create -f https://operatorhub.io/install/argocd-operator.yaml
subscription.coreos.com/my-argocd-operator created
→ ~ git:(argo-all-namespaces) ✘ kubectl get pods -n operators
No resources found in operators namespace.
→ ~ git:(argo-all-namespaces) ✘ kubectl get pods -n operators -w
^C
→ ~ git:(argo-all-namespaces) ✘ kubectl get pods -n operators
No resources found in operators namespace.
→ ~ git:(argo-all-namespaces) ✘ kubectl get pods -n operators
No resources found in operators namespace.
→ ~ git:(argo-all-namespaces) ✘ kubectl get pods -n operators
NAME                                READY   STATUS        RESTARTS   AGE
argocd-operator-controller-manager-c775dbd54-vqmc4   0/2     ContainerCreating   0          35s
→ ~ git:(argo-all-namespaces) ✘

```

```

→ ~ git:(argo-all-namespaces) ✘ cd ~/go/src/github.com/
F5Networks/           example.com/      redhat-developer/  test/
argoproj/             iam-veeramalla/    secure-mid-stream/

```

→ spring-boot-app git:(main) ✘ ls	Dockerfile	README.md	pom.xml
→ spring-boot-app git:(main) ✘ ls target	src	target	
classes	maven-archiver		spring-boot-web.jar
generated-sources	maven-status		spring-boot-web.jar.original
→ spring-boot-app git:(main) ✘			

```

# For Java 8, try this
# FROM openjdk:8-jdk-alpine

# For Java 11, try this
FROM adoptopenjdk/openjdk11:alpine-jre

# Refer to Maven build -> finalName
ARG JAR_FILE=target/spring-boot-web.jar

# cd /opt/app
WORKDIR /opt/app

# cp target/spring-boot-web.jar /opt/app/app.jar
COPY ${JAR_FILE} app.jar

# java -jar /opt/app/app.jar
ENTRYPOINT ["java","-jar","app.jar"]

## sudo docker run -p 8080:8080 -t docker-spring-boot:1.0
## sudo docker run -p 80:8080 -t docker-spring-boot:1.0
## sudo docker run -p 443:8443 -t docker-spring-boot:1.0
~
```

```
  steps {
    sh 'echo passed'
    //git branch: 'main', url: 'https://github.com/iam-veeramalla/Jenkins-Zero-To-Hero.git'
  }
}
stage('Build and Test') {
  steps {
    sh 'ls -ltr'
    // build the project and create a JAR file
    sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && mvn clean package'
  }
}
stage('Static Code Analysis') {
  environment {
    SONAR_URL = "http://34.233.125.26:9000"
  }
  steps {
    withCredentials([string(credentialsId: 'sonarqube', variable: 'SONAR_AUTH_TOKEN')]) {
      sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && mvn sonar:sonar -Dsonar.login=$SONAR_AUTH_TOKEN
-Dsonar.host.url=${SONAR_URL}'
    }
  }
}
stage('Build and Push Docker Image') {
  environment {
    DOCKER_IMAGE = "abhishekf5/ultimate-cicd:${BUILD_NUMBER}"
    // DOCKERFILE_LOCATION = "java-maven-sonar-argocd-helm-k8s/spring-boot-app/Dockerfile"
    REGISTRY_CREDENTIALS = credentials('docker-cred')
  }
  steps {
    script {
      sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && docker build -t ${DOCKER_IMAGE} .'
      def dockerImage = docker.image("${DOCKER_IMAGE}")
      docker.withRegistry('https://index.docker.io/v1/', "docker-cred") {
        dockerImage.push()
      }
    }
    SONAR_URL = "http://34.233.125.26:9000"
  }
  steps {
    withCredentials([string(credentialsId: 'sonarqube', variable: 'SONAR_AUTH_TOKEN')]) {
      sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && mvn sonar:sonar -Dsonar.login=$SONAR_AUTH_TOKEN
-Dsonar.host.url=${SONAR_URL}'
    }
  }
}
stage('Build and Push Docker Image') {
  environment {
    DOCKER_IMAGE = "abhishekf5/ultimate-cicd:${BUILD_NUMBER}"
    // DOCKERFILE_LOCATION = "java-maven-sonar-argocd-helm-k8s/spring-boot-app/Dockerfile"
    REGISTRY_CREDENTIALS = credentials('docker-cred')
  }
  steps {
    script {
      sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && docker build -t ${DOCKER_IMAGE} .'
      def dockerImage = docker.image("${DOCKER_IMAGE}")
      docker.withRegistry('https://index.docker.io/v1/', "docker-cred") {
        dockerImage.push()
      }
    }
  }
}
stage('Update Deployment File') {
  environment {
    GIT_REPO_NAME = "Jenkins-Zero-To-Hero"
    GIT_USER_NAME = "iam-veeramalla"
  }
  steps {
    withCredentials([string(credentialsId: 'github', variable: 'GITHUB_TOKEN')]) {
      sh '''
        git config user.email "abhishek.xyz@gmail.com"
        git config user.name "Abhishek Veeramalla"
        BUILD_NUMBER=${BUILD_NUMBER}
      '''
    }
  }
}
```

```

        }
    }
    stage('Update Deployment File') {
        environment {
            GIT_REPO_NAME = "Jenkins-Zero-To-Hero"
            GIT_USER_NAME = "iam-veeramalla"
        }
        steps {
            withCredentials([string(credentialsId: 'github', variable: 'GITHUB_TOKEN')]) {
                sh """
                    git config user.email "abhishek.xyz@gmail.com"
                    git config user.name "Abhishek Veeramalla"
                    BUILD_NUMBER=${BUILD_NUMBER}
                    sed -i "s/replaceImageTag/${BUILD_NUMBER}/g" java-maven-sonar-argocd-helm-k8s/spring-boot-app-manifests/deployment.yml
                    git add java-maven-sonar-argocd-helm-k8s/spring-boot-app-manifests/deployment.yml
                    git commit -m "Update deployment image to version ${BUILD_NUMBER}"
                    git push https://${GITHUB_TOKEN}@github.com/${GIT_USER_NAME}/${GIT_REPO_NAME} HEAD:main
                """
            }
        }
    }
}

```

## # spring Boot Manifest

The screenshot shows a GitHub commit page for the repository Jenkins-Zero-To-Hero / java-maven-sonar-argocd-helm-k8s / spring-boot-app-manifests. The commit message is "fix indentation issues in service". It contains two files: deployment.yml and service.yml.

Name	Last commit message
deployment.yml	Update deployment.yml
service.yml	fix indentation issues in service

The screenshot shows the deployment.yml file content in GitHub Copilot. The file is a Kubernetes Deployment manifest with the following code:

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-boot-app
  labels:
    app: spring-boot-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: spring-boot-app
  template:
    metadata:
      labels:
        app: spring-boot-app
    spec:
      containers:
        - name: spring-boot-app
          image: abhishekf5/ultimate-cicd:replaceImageTag
          ports:
            - containerPort: 8080

```

YouTube | JENKINS END TO END CICD | Jenkins-Zero-To-Hero/java-maven-sonar-argocd-helm-k8s/spring-boot-app-manifests/service... | +

Code Issues 16 Pull requests 208 Actions Projects Security Insights

**Jenkins-Zero-To-Hero / java-maven-sonar-argocd-helm-k8s / spring-boot-app-manifests / service.yml**

iam-veeramalla fix indentation issues in service 5e5c2e0 · last year History

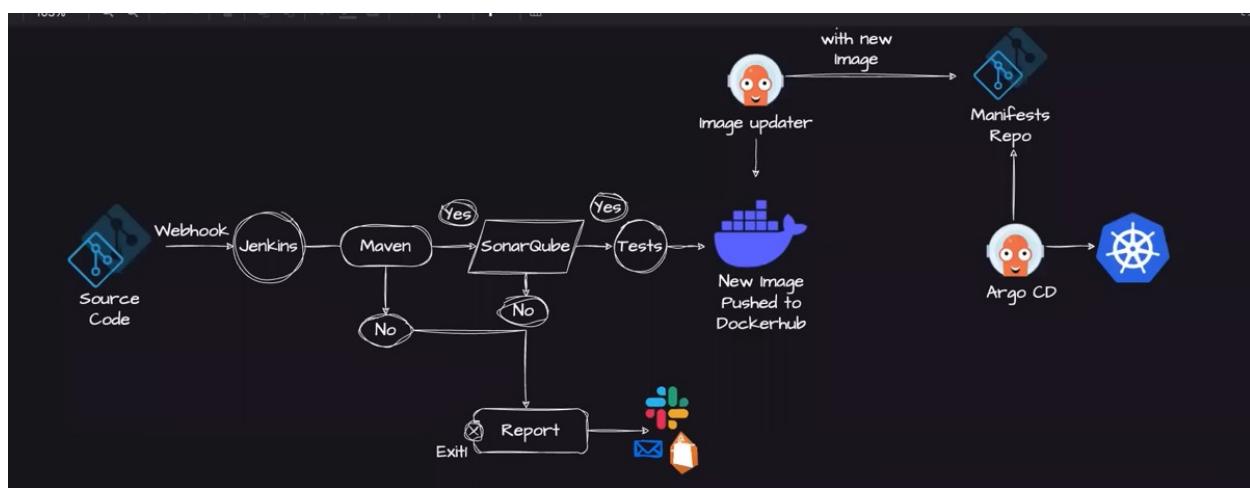
**Code** Blame 13 lines (13 loc) · 207 Bytes

```

1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: spring-boot-app-service
5 spec:
6   type: NodePort
7   ports:
8     - name: http
9       port: 80
10      targetPort: 8080
11      protocol: TCP
12   selector:
13     app: spring-boot-app

```

Type here to search 27°C Partly sunny 17:13 03-11-2024



The screenshot shows the Jenkins 'Credentials' management interface. At the top, there's a navigation bar with links for 'Dashboard', 'Manage Jenkins', and 'Credentials'. The main title is 'Credentials'. Below the title is a table with columns: T, P, Store ↓, Domain, ID, and Name. There is one entry: an icon of a key, a user icon, 'System' selected in a dropdown, '(global)' under Domain, 'sonarqube' under ID, and 'sonarqube' under Name. A search bar at the top right contains the placeholder 'Search (⌘+K)'. On the far right, there are icons for help, security, user status (1), and log out.

## Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	sonarqube	sonarqube

### Stores scoped to Jenkins

P	Store ↓	Domains
	System	(global)

Icon: S M L

# username + password – docker hub

The screenshot shows the 'New credentials' creation page. The title is 'New credentials'. Under 'Kind', 'Username with password' is selected. In the 'Scope' section, 'Global (Jenkins, nodes, items, all child items, etc)' is chosen. The 'Username' field is empty. An unchecked checkbox 'Treat username as secret' is present. The 'Password' field is also empty. At the bottom is a blue 'Create' button. The browser's address bar shows the full URL of the Jenkins instance.

```

        steps {
            withCredentials([string(credentialsId: 'sonarqube', variable: 'SONAR_AUTH_TOKEN')]) {
                sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && mvn sonar:sonar -Dsonar.login=$SONAR_AUTH_TOKEN
-Dsonar.host.url=${SONAR_URL}'
            }
        }
    stage('Build and Push Docker Image') {
        environment {
            DOCKER_IMAGE = "abhishekf5/ultimate-cicd:${BUILD_NUMBER}"
            // DOCKERFILE_LOCATION = "java-maven-sonar-argocd-helm-k8s/spring-boot-app/Dockerfile"
            REGISTRY_CREDENTIALS = credentials('docker-cred')
        }
        steps {
            script {
                sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && docker build -t ${DOCKER_IMAGE} .'
                def dockerImage = docker.image("${DOCKER_IMAGE}")
                docker.withRegistry('https://index.docker.io/v1/', "docker-cred") {
                    dockerImage.push()
                }
            }
        }
    }
    stage('Update Deployment File') {
        environment {
            GIT_REPO_NAME = "Jenkins-Zero-To-Hero"
            GIT_USER_NAME = "iam-veeramalla"
        }
        steps {
            withCredentials([string(credentialsId: 'github', variable: 'GITHUB_TOKEN')]) {
                sh '''
                    git config user.email "abhishek.xyz@gmail.com"
                    git config user.name "Abhishek Veeramalla"
                    BUILD_NUMBER=${BUILD_NUMBER}
                    sed -i "s/replaceImageTag/${BUILD_NUMBER}/g" java-maven-sonar-argocd-helm-k8s/spring-boot-app-manifests/deployment.yaml
                '''
            }
        }
    }
}

```

The screenshot shows the Jenkins 'Create New Credential' dialog for a Docker Registry credential. The fields filled in are:

- Username:** abhishekf5
- Treat username as secret:** (unchecked)
- Password:** (redacted)
- ID:** docker-cred
- Description:** (empty)

At the bottom left is a blue **Create** button, and at the bottom right are links for **REST API** and **Jenkins 2.397**.

# note – password must be DockerHub only

**Global credentials (unrestricted)**
+ Add Credentials

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

ID	Name	Kind	Description
 sonarqube	sonarqube	Secret text	
 docker-cred	abhishek15/*****	Username with password	

Icon: S M L

## # now its for GITHUb Credentials – Choose Secret text

**New credentials**

Kind

Scope ?

Secret

ID ?

Description ?

Create

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

### New credentials

Kind

Secret text

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

Secret

ID ?  
github

Description ?

**Create**

# Access Key from Github – Settings- developer settings-personal- access-tokens – Generate new-token

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

Expiration \*  
30 days

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> <b>repo</b>	Full control of private repositories
<input type="checkbox"/> <b>repo:status</b>	Access commit status
<input type="checkbox"/> <b>repo_deployment</b>	Access deployment status
<input type="checkbox"/> <b>public_repo</b>	Access public repositories

## New credentials

Kind

Secret text

Scope ?

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

Secret

.....

ID ?

+ github

Description ?

Save password for http://34.201.116.83:8080?

Username: No username

Password: .....

Show password

Don't save Save

Global credentials (unrestricted) >

ID	Name	Kind	Description
🔑 sonarqube	sonarqube	Secret text	<input type="button" value="Edit"/>
💻 docker-cred	abhishekf5/*****	Username with password	<input type="button" value="Edit"/>
🔑 github	github	Secret text	<input type="button" value="Edit"/>

Icon: S M L

# Restart

Jenkins-Zero-To-X | raw.githubusercontent.com | CI/CD\_Jenkins\_4 | Launch an instance | Instance details | EC2 Management | Restart Jenkins | Security - My Account | bcf05a4780214 | OperatorHub.io | Persona

34.201.116.83:8080/restart/

# Jenkins

Dashboard > Restart

+ New Item Are you sure you want to restart Jenkins? Yes

People +

Build History

Manage Jenkins

My Views

**Build Queue** ▾  
No builds in the queue.

**Build Executor Status** ▾  
1 Idle  
2 Idle

```

}
stage('Static Code Analysis') {
    environment {
        SONAR_URL = "http://34.201.116.83:9000"
    }
    steps {
        withCredentials([string(credentialsId: 'sonarqube', variable: 'SONAR_AUTH_TOKEN')]) {
            sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && mvn sonar:sonar -Dsonar.login=$SONAR_AUTH_TOKEN -Dsonar.host.url=${SONAR_URL}'
        }
    }
}
stage('Build and Push Docker Image') {
    environment {
        DOCKER_IMAGE = "abhishekf5/ultimate-cicd:${BUILD_NUMBER}"
        // DOCKERFILE_LOCATION = "java-maven-sonar-argocd-helm-k8s/spring-boot-app/Dockerfile"
        REGISTRY_CREDENTIALS = credentials('docker-cred')
    }
}

```

```

→ spring-boot-app git:(main) ✘ kubectl get pods -n operators
NAME                                     READY   STATUS    RESTARTS   AGE
argocd-operator-controller-manager-c775dbd54-vqmc4   2/2     Running   0          17m
→ spring-boot-app git:(main) ✘

```

# now time for CD – PART - Click on Build now on Jenkins

Dashboard > ultimate-demo >

**Pipeline ultimate-demo**

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

Build History trend Filter builds... #1 Apr 3, 2023, 8:06 PM Atom feed for all Atom feed for failures

Stage View

Average stage times: + Declarative: Checkout SCM 55ms

#1 Apr 04 01:36 No Changes 55ms

Declarative: Checkout SCM 55ms

Permalinks

Atom feed for all Atom feed for failures

Dashboard > ultimate-demo >

**Pipeline ultimate-demo**

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

Build History trend Filter builds... #1 Apr 3, 2023, 8:06 PM Atom feed for all Atom feed for failures

Stage View

Average stage times: + Declarative: Checkout SCM 664ms Checkout 408ms Build and Test 13s Static Code Analysis 21s Build and Push Docker Image 6s

#1 Apr 04 01:36 No Changes 664ms 408ms 13s 21s 6s

Declarative: Checkout SCM 664ms

Checkout 408ms

Build and Test 13s

Static Code Analysis 21s

Build and Push Docker Image 6s

Permalinks

Atom feed for all Atom feed for failures

Firefox - Tue 4 Apr 1:38 AM

Dashboard > ultimate-demo >

## Pipeline ultimate-demo

- Status
- </> Changes
- ▷ Build Now
- ⚙ Configure
- >Delete Pipeline
- Full Stage View
- Rename
- Pipeline Syntax

**Stage View**

Declarative: Checkout SCM	Checkout	Build and Test	Static Code Analysis	Build and Push Docker Image	Update Deployment File
664ms	408ms	13s	21s	22s	1s
664ms	408ms	13s	21s	22s	1s

Average stage times: (Average full run time: ~1min 33s)

#1 Apr 04 01:36 No Changes

**Build History** trend ▾

Filter builds... /

#1 Apr 3, 2023, 8:06 PM

Atom feed for all Atom feed for failures

**Permalinks**

- Last build (#1), 1 min 7 sec ago

crede

Highlight All Match Case Match Diacritics Whole Words 1 of 4 matches

Firefox - Tue 4 Apr 1:40 AM

Dashboard > ultimate-demo > #1

```
[INFO] Finished at: 2023-04-03T20:07:07Z
[INFO] -----
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Static Code Analysis)
[Pipeline] withEnv
[Pipeline] {
[Pipeline] withCredentials
Masking supported pattern matches of $SONAR_AUTH_TOKEN
[Pipeline] {
[Pipeline] sh
+ cd java-maven-sonar-argocd-helm-k8s/spring-boot-app
+ mvn sonar:sonar -Dsonar.login=**** -Dsonar.host.url=http://34.201.116.83:9000
[INFO] Scanning for projects...
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-install-plugin/2.5.2/maven-
install-plugin-2.5.2.pom
Progress (1): 2.7/6.4 kB
Progress (1): 5.5/6.4 kB
Progress (1): 6.4 kB

Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-install-plugin/2.5.2/maven-
install-plugin-2.5.2.pom (6.4 kB at 17 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins/25/maven-plugins-
25.pom
Progress (1): 2.7/9.6 kB
Progress (1): 5.5/9.6 kB
Progress (1): 8.2/9.6 kB
Progress (1): 9.6 kB
```

sonarsonar

Highlight All Match Case Match Diacritics Whole Words 1 of 1 match

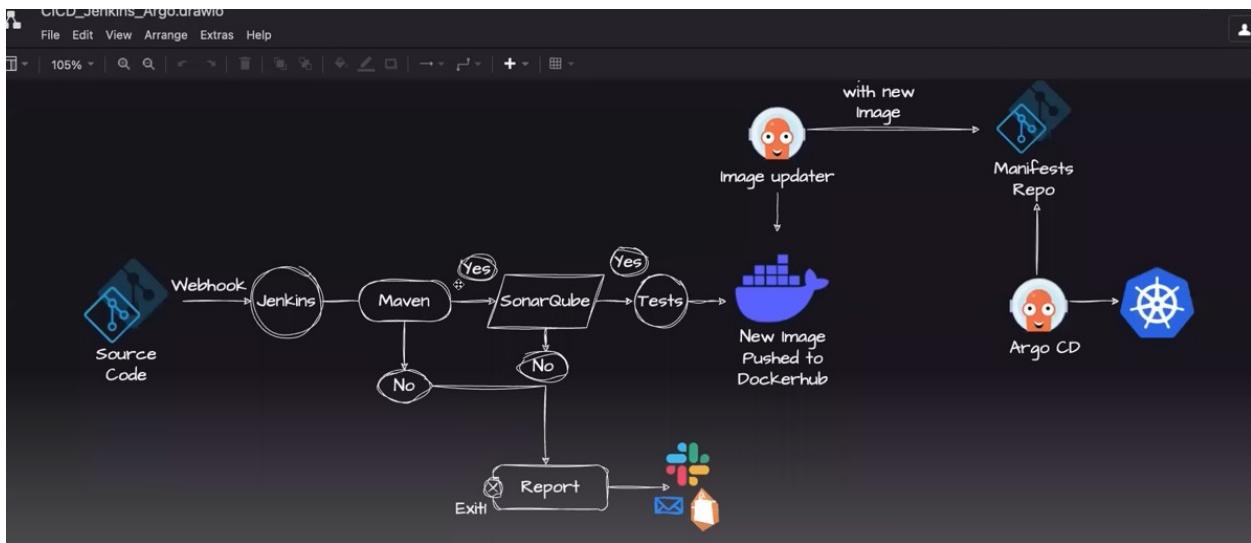
# Go to SonarCube – Refresh it –

The screenshot shows the SonarQube Security page. At the top, there is a banner indicating that the version of SonarQube is at end-of-life. Below this, the 'Tokens' section allows users to generate tokens for code scans or web services. A table lists a single token named 'jenkins', which was last used 'less than 1 hour ago' and created on 'April 4, 2023'. A 'Revoke' button is available for this token. Below the tokens section, there is a form to 'Enter a new password'. A note states that all fields marked with an asterisk (\*) are required. An 'Old Password' field is shown with a red asterisk. To the right of the password form is a promotional banner for SonarLint, which is described as a free IDE plugin for finding and fixing issues earlier in the workflow. It includes a 'Learn More' link and a 'Dismiss' button.

## # Go to Projects

The screenshot shows the SonarQube Projects page. On the left, there is a sidebar with various filters: Quality Gate (Passed: 1, Failed: 0), Reliability (Bugs: A, B, C, D, E), Security (Vulnerabilities: A, B, C, D, E), and Security Review (Security Hotspots: A, B, C, D, E). The main area displays a single project named 'spring-boot-demo' with a status of 'Passed'. The project details include: Bugs (0 A), Vulnerabilities (0 A), Hotspots Reviewed (- A), Code Smells (0 A), Coverage (0.0%), Duplications (0.0%), and Lines (79 XML, Java). A note indicates that the last analysis was 3 minutes ago. Below the project details, there is a warning about using an embedded database for evaluation purposes. To the right of the project details is a promotional banner for SonarLint, similar to the one on the previous page. It includes a 'Learn More' link and a 'Dismiss' button.

## # now check Docker image is Created or not ??

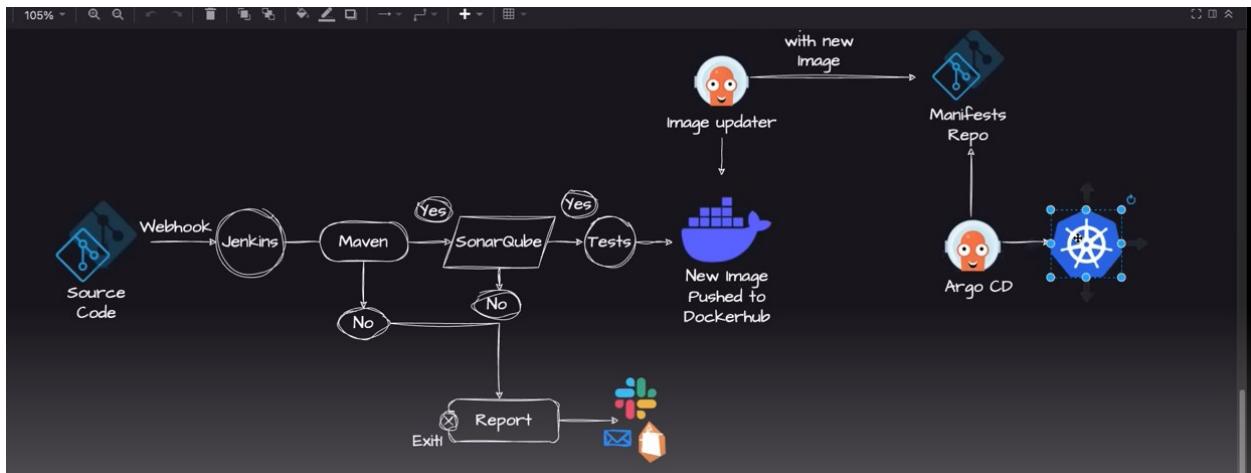


```

root@ip-172-31-87-213:~# docker images
REPOSITORY                                TAG      IMAGE ID      CREATED        SIZE
abhishekf5/ultimate-ci_cd                1        21eef10a77c5  4 minutes ago  169MB
abhishekf5/maven-abhishek-docker-agent    v1       3fb9145e2467  9 hours ago   913MB
adoptopenjdk/openjdk11                     alpine-jre 41f668e6b55c  13 days ago   150MB
root@ip-172-31-87-213:~#

```

### # Final Stage on ArgoCD deploy on Kubernetes



```

→ spring-boot-app git:(main) ✘ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

→ spring-boot-app git:(main) ✘

```

# <https://operatorhub.io/operator/argocd-operator>

The screenshot shows a web browser window with the URL <https://operatorhub.io/operator/argocd-operator>. The page is titled "Argo CD" and features a large "Install" button. To the right, there are detailed sections for the operator's configuration, including "CHANNEL: alpha", "VERSION: 0.12.0 (Current)", and "CAPABILITY LEVEL" which includes "Basic Install", "Seamless Upgrades", "Full Lifecycle" (which is selected), and "Deep Insights". The "PROVIDER" section lists "Argo CD Community". The browser's taskbar at the bottom shows other open tabs like YouTube, Jenkins, and Jenkins-Zero-To-Hero/java-maven, along with system icons for battery, signal, and date/time.

# Click on operator documentation – usage- Basic ----# Create ArgoCD Controller

The screenshot shows a web browser window with multiple tabs open, including YouTube, Jenkins, OperatorHub.io, and Argo CD Operator. The main content is the Argo CD Operator documentation for the 'Create' section. The left sidebar has a 'Basics' category selected. The right sidebar contains a 'Table of contents' with various sections like Create, ConfigMaps, Secrets, Deployments, Services, and more. The central content area includes a note about the ArgoCD Reference, a code example for creating an ArgoCD cluster, and a command-line example:

```
apiVersion: argoproj.io/v1alpha1
kind: Argocd
metadata:
  name: example-argocd
  labels:
    example: basic
spec: {}
```

**Create**

Create a new Argo CD cluster in the `argocd` namespace using the provided basic example.

```
kubectl create -n argocd -f examples/argocd-basic.yaml
```

This screenshot is identical to the one above, showing the 'Create' section of the Argo CD Operator documentation. The left sidebar has the 'Basics' category selected. The right sidebar shows the same 'Table of contents'. The central content area includes the creation note, the YAML code example, and the command-line example. Additionally, it includes a table of resources created by the deployment:

NAME	DATA	AGE
configmap/argocd-cm	14	2m9s
configmap/argocd-rbac-cm	3	2m9s
configmap/argocd-ssh-known-hosts-cm	1	2m9s
configmap/argocd-tls-certs-cm	0	2m9s

There will be several Argo CD resources created that should be familiar to anyone who has deployed Argo CD.

```
kubectl get cm,secret,deploy -n argocd
```

Some unrelated items have been removed for clarity.

NAME	TYPE
secret/argocd-secret	Opaque
secret/example-argocd-ca	kubernetes.io/tls

The screenshot shows the 'ConfigMaps' section of the Argo CD Operator Basics documentation. It displays two tables: one for ConfigMaps and one for Secrets.

**ConfigMaps**

NAME	DATA	AGE
configmap/argocd-cm	14	2m9s
configmap/argocd-rbac-cm	3	2m9s
configmap/argocd-ssh-known-hosts-cm	1	2m9s
configmap/argocd-tls-certs-cm	0	2m9s

**Secrets**

NAME	TYPE
secret/argocd-secret	Opaque
secret/example-argocd-ca	kubernetes.io/tls
secret/example-argocd-cluster	Opaque
secret/example-argocd-tls	kubernetes.io/tls

The screenshot shows the 'ConfigMaps' section of the Argo CD Operator Basics documentation. It displays a table for ConfigMaps and a section for Secrets.

**ConfigMaps**

NAME	DATA	AGE
configmap/argocd-cm	14	33s
configmap/argocd-rbac-cm	3	33s
configmap/argocd-ssh-known-hosts-cm	1	33s
configmap/argocd-tls-certs-cm	0	33s

**Secrets**

There is a Secret that is used by Argo CD named `argocd-secret`. The `argocd-server` component reads this secret to obtain the admin password for authentication. NOTE: Upon initial deployment, the initial password for the `admin` user is stored in the `argocd-cluster` secret instead.

The screenshot shows a web browser window with multiple tabs open. The active tab is 'argocd-operator.readthedocs.io/en/latest/usage/basics/'. The page content discusses the 'Secrets' managed by the operator. It mentions a secret named 'argocd-secret' used for authentication and lists other secrets like 'example-argocd-ca', 'example-argocd-cluster', and 'example-argocd-tls'. A table shows the details of these secrets.

NAME	TYPE
secret/argocd-secret	Opaque
secret/example-argocd-ca	kubernetes.io/tls
secret/example-argocd-cluster	Opaque
secret/example-argocd-tls	kubernetes.io/tls

The page also states that the cluster Secret contains the admin password for authenticating with Argo CD.

This screenshot shows the same documentation page as the previous one, but with a different section highlighted. It focuses on the YAML representation of a cluster Secret and how changes to the admin password are synchronized automatically.

```
apiVersion: v1
data:
  admin.password: ...
kind: Secret
metadata:
  labels:
    app.kubernetes.io/name: example-argocd-cluster
    app.kubernetes.io/part-of: argocd
  example: basic
  name: example-argocd-cluster
  namespace: argocd
type: Opaque
```

The page explains that the operator will watch for changes to the `admin.password` value and synchronize it automatically. It also provides a command to fetch the admin password from the cluster Secret.

```
→ spring-boot-app git:(main) ✘ vim argocd-basic.yml
→ spring-boot-app git:(main) ✘ kubectl apply -f argocd-basic.yml
argocd.argoproj.io/example-argocd created
→ spring-boot-app git:(main) ✘
```

```
apiVersion: argoproj.io/v1alpha1
kind: ArgoCD
metadata:
  name: example-argocd
  labels:
    example: basic
spec: {}
[REDACTED]
```

NAME	READY	STATUS	RESTARTS	AGE
example-argocd-application-controller-0	0/1	ContainerCreating	0	17s
example-argocd-redis-6db56fc79f-fps8w	0/1	ContainerCreating	0	18s
example-argocd-repo-server-6b78f58dc4-vq9t8	0/1	Init:0/1	0	18s
example-argocd-server-846cdb99cc-p4jdc	0/1	ContainerCreating	0	18s

```
# now Final Step .. argocd pull latest image from the git Deployment.yml file
```

A screenshot of a GitHub code editor window. The URL in the address bar is [github.com/iam-veeramalla/Jenkins-Zero-To-Hero/blob/main/java-maven-sonar-argocd-helm-k8s/spring-boot-app-manifests/deployment.yaml](https://github.com/iam-veeramalla/Jenkins-Zero-To-Hero/blob/main/java-maven-sonar-argocd-helm-k8s/spring-boot-app-manifests/deployment.yaml). The file content is a Kubernetes Deployment manifest named deployment.yaml. It defines a deployment for a Spring Boot application with two replicas, selecting pods by the app label. The containers use an image named abhishek5/ultimate-cicd:replaceImageTag, port 8080, and expose port 8080.

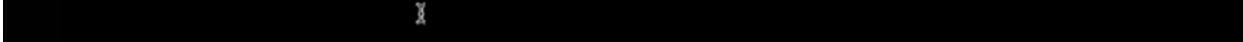
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-boot-app
  labels:
    app: spring-boot-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: spring-boot-app
  template:
    metadata:
      labels:
        app: spring-boot-app
    spec:
      containers:
        - name: spring-boot-app
          image: abhishek5/ultimate-cicd:replaceImageTag
          ports:
            - containerPort: 8080
```

A terminal window showing Kubernetes command outputs. The first command, `kubectl get pods`, lists several pods in the `spring-boot-app` namespace, all in a `ContainerCreating` state. The second command, `kubectl get pods -w`, shows the same pods transitioning through various stages. The third command, `kubectl get svc`, lists services with their corresponding IP addresses and port mappings.

```
NAME           READY   STATUS             RESTARTS   AGE
example-argocd-application-controller-0   0/1     ContainerCreating   0          17s
example-argocd-redis-6db56fc79f-fps8w     0/1     ContainerCreating   0          18s
example-argocd-repo-server-6b78f58dc4-vq9t8 0/1     Init:0/1          0          18s
example-argocd-server-846cdb99cc-p4jdc     0/1     ContainerCreating   0          18s
→ spring-boot-app git:(main) ✘ kubectl get pods -w
NAME           READY   STATUS             RESTARTS   AGE
example-argocd-application-controller-0   0/1     ContainerCreating   0          50s
example-argocd-redis-6db56fc79f-fps8w     0/1     ContainerCreating   0          51s
example-argocd-repo-server-6b78f58dc4-vq9t8 0/1     Init:0/1          0          51s
example-argocd-server-846cdb99cc-p4jdc     0/1     ContainerCreating   0          51s
^C
→ spring-boot-app git:(main) ✘ kubectl get svc
NAME           TYPE      CLUSTER-IP      EXTERNAL-IP   PORT(S)      AGE
example-argocd-metrics   ClusterIP  10.104.135.129 <none>       8082/TCP    65s
example-argocd-redis     ClusterIP  10.109.78.67  <none>       6379/TCP    64s
example-argocd-repo-server ClusterIP  10.109.56.71  <none>       8081/TCP,8084/TCP  64s
example-argocd-server    ClusterIP  10.97.80.124  <none>       80/TCP,443/TCP  64s
example-argocd-server-metrics ClusterIP  10.100.182.196 <none>       8083/TCP    64s
kubernetes           ClusterIP  10.96.0.1       <none>       443/TCP    33m
```

# Change type: from Clusterip to nodeport --mode

```
→ spring-boot-app git:(main) ✘ kubectl edit svc example-argocd-server
service/example-argocd-server edited
→ spring-boot-app git:(main) ✘
```



```
iTerm2 Shell Edit View Session Scripts Profiles Toolbar Window Help
root@ip-172-31-87-213: ~ (ssh)
kubectl edit svc example-argocd-server
kubectl (v1)

- apiVersion: argoproj.io/v1alpha1
  blockOwnerDeletion: true
  controller: true
  kind: ArgoCD
  name: example-argocd
  uid: 31b2244e-5db8-4bb0-a9ea-29a510f26743
  resourceVersion: "3966"
  uid: 8e8e01bc-0a3f-4fc1-889c-7780dd9e3718
spec:
  clusterIP: 10.97.80.124
  clusterIPs:
  - 10.97.80.124
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: 8080
  - name: https
    port: 443
    protocol: TCP
    targetPort: 8080
  selector:
    app.kubernetes.io/name: example-argocd-server
  sessionAffinity: None
  type: 
status:
  loadBalancer: {}
-- INSERT --
```

```
→ spring-boot-app git:(main) ✘ kubectl get svc
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP  PORT(S)          AGE
example-argocd-metrics  ClusterIP  10.104.135.129 <none>        8082/TCP        100s
example-argocd-redis  ClusterIP  10.109.78.67    <none>        6379/TCP        99s
example-argocd-repo-server  ClusterIP  10.109.56.71    <none>        8081/TCP,8084/TCP  99s
example-argocd-server  NodePort   10.97.80.124    <none>        80:31271/TCP,443:32471/TCP  99s
example-argocd-server-metrics  ClusterIP  10.100.182.196 <none>        8083/TCP        99s
kubernetes       ClusterIP  10.96.0.1     <none>        443/TCP         33m
→ spring-boot-app git:(main) ✘
```

# now Execute it on Browser - by using minikube port forwarding

```

→ spring-boot-app git:(main) ✘ minikube service argocd-server
→ spring-boot-app git:(main) ✘ minikube service list
|-----|-----|-----|-----|
| NAMESPACE | NAME | TARGET PORT | URL |
|-----|-----|-----|-----|
| default | example-argocd-metrics | No node port |
| default | example-argocd-redis | No node port |
| default | example-argocd-repo-server | No node port |
| default | example-argocd-server | http/80 | http://192.168.64.19:31271 |
| | https/443 | https://192.168.64.19:32471 |
| default | example-argocd-server-metrics | No node port |
| default | kubernetes | No node port |
| kube-system | kube-dns | No node port |
| olm | operatorhubio-catalog | No node port |
| olm | packageserver-service | No node port |
| operators | argocd-operator-controller-manager-metrics-service | No node port |
|-----|-----|-----|-----|
→ spring-boot-app git:(main) ✘

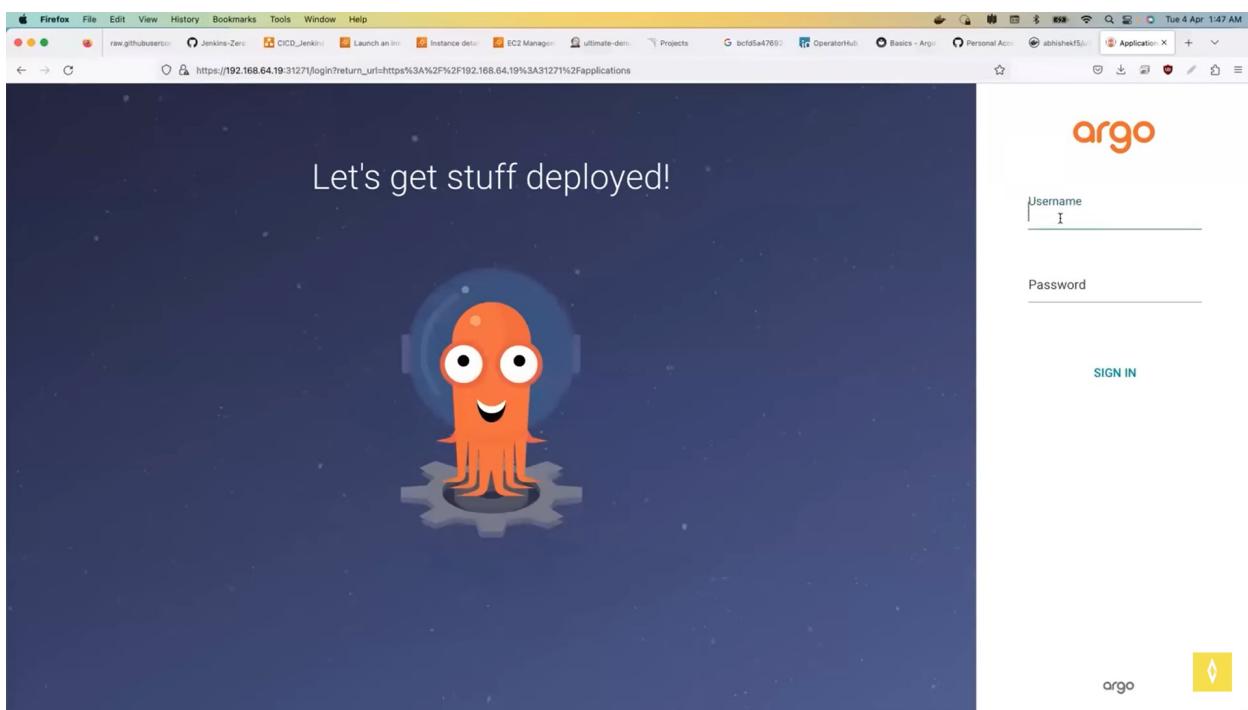
```

# Check pods are up

```

→ spring-boot-app git:(main) ✘ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
example-argocd-application-controller-0   1/1     Running   0          2m26s
example-argocd-redis-6db56fc79f-fps8w   1/1     Running   0          2m27s
example-argocd-repo-server-6b78f58dc4-vq9t8   1/1     Running   0          2m27s
example-argocd-server-846cdb99cc-p4jdc   1/1     Running   0          2m27s
→ spring-boot-app git:(main) ✘

```

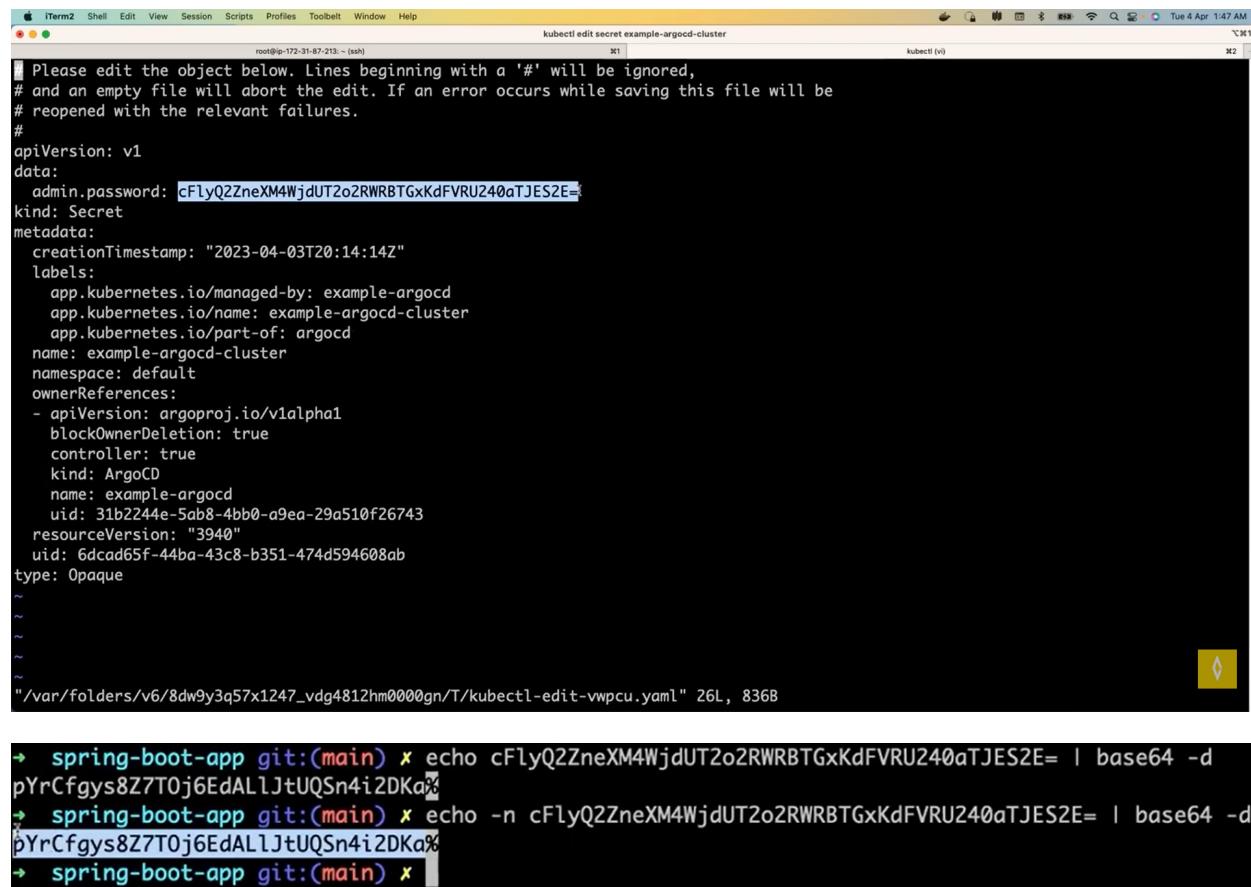


# password

## Argocd stores password in cluster

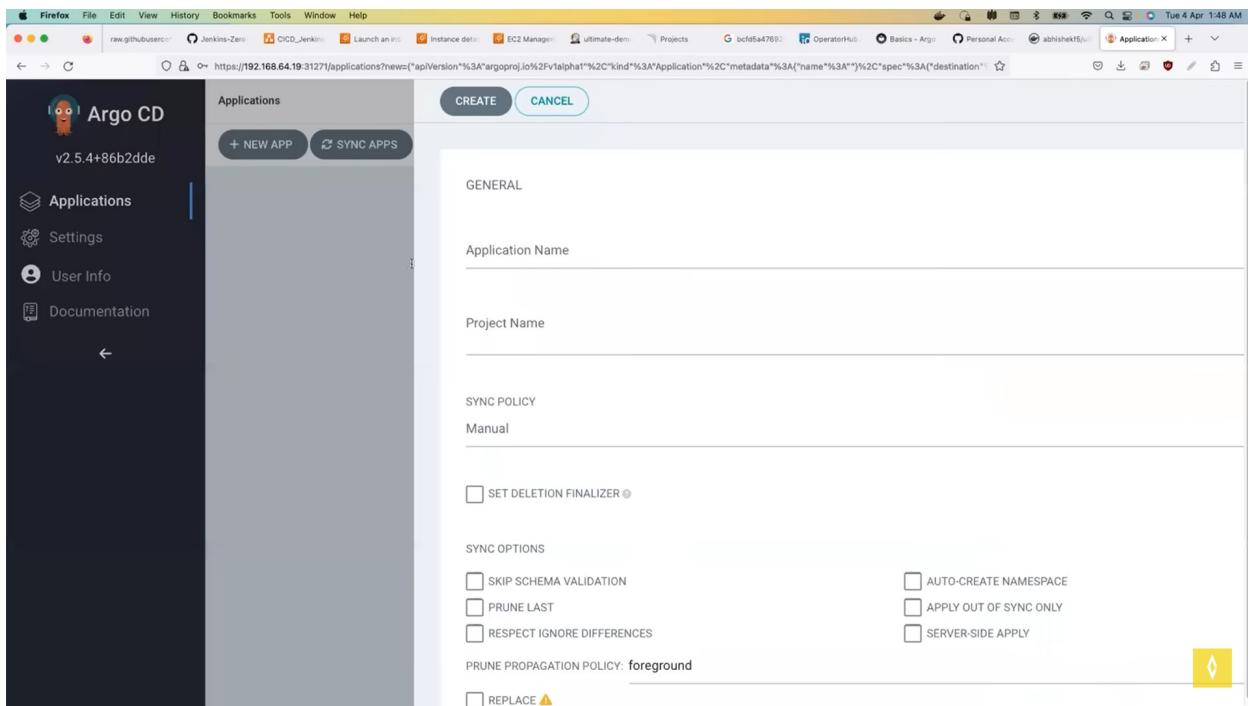
```
→ spring-boot-app git:(main) ✘ kubectl get secret
NAME                                     TYPE        DATA   AGE
argocd-secret                           Opaque      5      3m6s
default-token-s9zf5                     kubernetes.io/service-account-token 3      34m
example-argocd-argocd-application-controller-token-k9q4v
example-argocd-argocd-grafana-token-gl9qp
example-argocd-argocd-redis-ha-token-59fwd
example-argocd-argocd-redis-token-jkd8q
example-argocd-argocd-server-token-jf87l
example-argocd-ca                        Opaque      1      3m6s
example-argocd-cluster                   Opaque      4      3m6s
example-argocd-default-cluster-config
example-argocd-tls                       kubernetes.io/tls                    2      3m6s
→ spring-boot-app git:(main) ✘
```

```
spring-boot-app git:(main) ✘ kubectl edit secret example-argocd-cluster
```

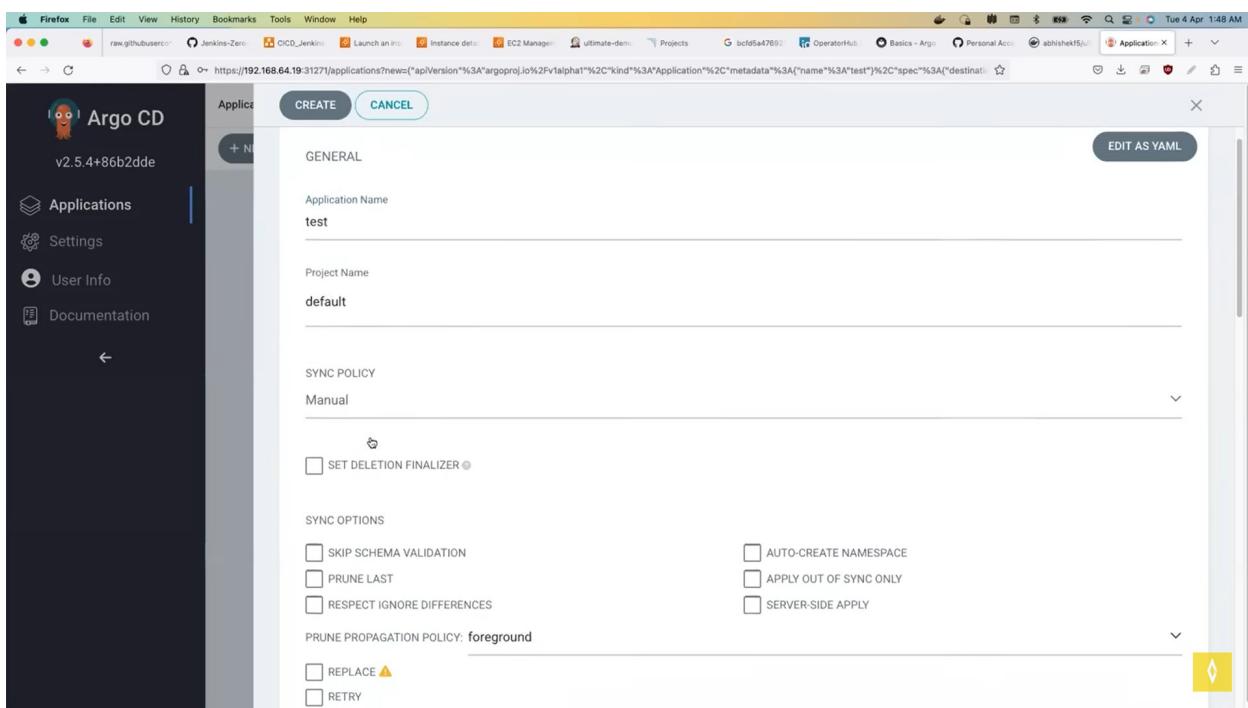


```
root@ip-172-31-87-213:~ (oh)
[kubectl edit secret example-argocd-cluster]
root@ip-172-31-87-213:~ (oh)
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  admin.password: cFlyQ2ZneXM4WjdUT2o2RWRBTGxKdFVRU240aTJES2E=
kind: Secret
metadata:
  creationTimestamp: "2023-04-03T20:14:14Z"
  labels:
    app.kubernetes.io/managed-by: example-argocd
    app.kubernetes.io/name: example-argocd-cluster
    app.kubernetes.io/part-of: argocd
  name: example-argocd-cluster
  namespace: default
  ownerReferences:
  - apiVersion: argoproj.io/v1alpha1
    blockOwnerDeletion: true
    controller: true
    kind: ArgoCD
    name: example-argocd
    uid: 31b2244e-5ab8-4bb0-a9ea-29a510f26743
  resourceVersion: "3949"
  uid: 6dcad65f-44ba-43c8-b351-474d594608ab
type: Opaque
~
~
~
~
~/var/folders/v6/8dw9y3q57x1247_vdg4812hm0000gn/T/kubectl-edit-vwpcu.yaml" 26L, 836B

→ spring-boot-app git:(main) ✘ echo cFlyQ2ZneXM4WjdUT2o2RWRBTGxKdFVRU240aTJES2E= | base64 -d
pYrCfgy8Z7T0j6EdALLjtUQSn4i2DKa%
→ spring-boot-app git:(main) ✘ echo -n cFlyQ2ZneXM4WjdUT2o2RWRBTGxKdFVRU240aTJES2E= | base64 -d
pYrCfgy8Z7T0j6EdALLjtUQSn4i2DKa%
→ spring-boot-app git:(main) ✘
```



## # Create app



Repository URL  
<https://github.com/iam-veeramalla/Jenkins-Zero-To-Hero> GIT ▾

Revision  
HEAD Branches ▾ (

Path  
java-maven-sonar-argocd-helm-k8s/spring-boot-app-manifests

+ DESTINATION

Cluster URL  
<https://kubernetes.default.svc> URL ▾  
https://kubernetes.default.svc

Namespace  
+

Directory ▾

DIRECTORY  
DIRECTORY RECURSE  DESTINATION

# Click on Create

The screenshot shows the Argo CD application configuration interface. On the left, there's a sidebar with navigation links like Applications, Settings, User Info, Documentation, and filters for Name, Kind, Sync Status, and Health Status. The main area is titled 'TEST' and contains fields for PROJECT (set to 'default'), LABELS (empty), ANNOTATIONS (empty), and NOTIFICATION SUBSCRIPTIONS (empty). It also includes fields for CLUSTER (URL: https://kubernetes.default.svc), NAMESPACE (default), CREATED\_AT (04/04/2023 01:49:29), REPO URL (https://github.com/iam-veeramalla/Jenkins-Zero-To-Hero), TARGET REVISION (HEAD), PATH (java-maven-sonar-argocd-helm-k8s/spring-boot-app-manifests), and REVISION HISTORY LIMIT (10). There are 'SAVE' and 'CANCEL' buttons at the top right.

## # Click on Sync

The screenshot shows the Argo CD sync interface. The left sidebar is identical to the previous screenshot. The main area displays an application named 'test' with a status of 'Missing'. The 'SYNC' tab is selected, showing 'CURRENT SYNC STATUS' as 'OutOfSync' (From HEAD (e27ae28) to HEAD (e27ae28)). The 'LAST SYNC RESULT' shows a 'Syncing' status with a note: 'Running a few seconds ago (Tue Apr 04 2023 01:50:10 GMT+0530)' and 'waiting to start'. On the right, there are buttons for 'SYNCHRONIZE' and 'CANCEL'. A large panel on the right lists sync options: PRUNE (unchecked), DRY RUN (unchecked), SYNC OPTIONS (SKIP\_SCHEMA\_VALIDATION (unchecked), PRUNE\_LAST (unchecked), RESPECT\_IGNORE\_DIFFERENCES (unchecked)), PRUNE PROPAGATION POLICY (REPLACE (unchecked), RETRY (unchecked)), SYNCHRONIZE RESOURCES (APPS/DEPLOYMENT/DEFAULT (checked)), and a yellow 'refresh' icon.

```
→ spring-boot-app git:(main) ✘ kubectl get deploy
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
example-argocd-redis	1/1	1	1	6m55s
example-argocd-repo-server	1/1	1	1	6m55s
example-argocd-server	1/1	1	1	6m55s
sprint-boot-app	2/2	2	2	59s

```
→ spring-boot-app git:(main) ✘
```

```
→ spring-boot-app git:(main) ✘ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
example-argocd-application-controller-0	1/1	Running	0	7m
example-argocd-redis-6db56fc79f-fps8w	1/1	Running	0	7m1s
example-argocd-repo-server-6b78f58dc4-vq9t8	1/1	Running	0	7m1s
example-argocd-server-846cdb99cc-p4jdc	1/1	Running	0	7m1s
sprint-boot-app-844c94fb56-72t67	1/1	Running	0	65s
sprint-boot-app-844c94fb56-lhmvb	1/1	Running	0	65s

```
→ spring-boot-app git:(main) ✘
```

```
# test ..
```

```
→ spring-boot-app git:(main) ✘ ls
Dockerfile      README.md      argocd-basic.yml pom.xml      src      target
→ spring-boot-app git:(main) ✘ kubectl edit deploy sprint-boot-app
```

# in spring boot app .. will change some data .. see the reflection in ArgoCD

```

metadata:
  creationTimestamp: null
  labels:
    app: sprint-boot-app
spec:
  containers:
    - image: abhishekf5/ultimate-cicd:2
      imagePullPolicy: IfNotPresent
      name: sprint-boot-app
      ports:
        - containerPort: 8080
          protocol: TCP
      resources: {}
      terminationMessagePath: /dev/termination-log
      terminationMessagePolicy: File
  dnsPolicy: ClusterFirst
  restartPolicy: Always
  schedulerName: default-scheduler
  securityContext: {}
  terminationGracePeriodSeconds: 30
status:
  availableReplicas: 2
  conditions:
    - lastTransitionTime: "2023-04-03T20:21:01Z"
      lastUpdateTime: "2023-04-03T20:21:01Z"
      message: Deployment has minimum availability.
      reason: MinimumReplicasAvailable
      status: "True"
      type: Available
    - lastTransitionTime: "2023-04-03T20:20:11Z"
      lastUpdateTime: "2023-04-03T20:21:01Z"

```

## # Refresh ArgoCD – Error

The screenshot shows the Argo CD web interface. On the left, there's a sidebar with navigation links: Argo CD (v2.5.4+86b2dde), Applications, Settings, User Info, Documentation, FILTERS (NAME, KINDS, SYNC STATUS, HEALTH STATUS), and a yellow 'Sync' button at the bottom.

The main content area is titled 'Application conditions'. It displays a single error message card:

- ComparisonError**
- rpc error: code = Unavailable desc = connection error: desc = "transport: authentication handshake failed"+ context deadline exceeded"
- a few seconds ago (Tue Apr 04 2023 01:52:09 GMT+0530)

The screenshot shows the Argo CD interface for an application named "test". The left sidebar includes sections for Applications, Settings, User Info, Documentation, Filters (NAME, KINDS, SYNC STATUS, HEALTH STATUS), and a GitHub integration section. The main panel displays the "APPLICATION DETAILS TREE" for the "test" application. It shows the current sync status as "Progressing" with an "OutOfSync" icon. The last sync result was "Sync OK" from HEAD (e27ae28) 3 minutes ago, with a comment from Abhishek Veeramalla about updating the deployment image to version 1. Below this, a deployment graph illustrates the flow from a "test" deployment target to two "sprint-boot-app" instances, each with its own revision (rev.2 and rev.1) and corresponding pods.

## # Click on test Dig deep

This screenshot shows the "TEST" tab of the Argo CD application configuration for the "test" deployment. The tab is part of a larger interface with tabs for SUMMARY, PARAMETERS, MANIFEST, DIFF, and EVENTS. The TEST section contains the following configuration details:

- PROJECT:** default
- ANNOTATIONS:** in-cluster (<https://kubernetes.default.svc>)
- CLUSTER:** in-cluster (<https://kubernetes.default.svc>)
- NAMESPACE:** default
- CREATED\_AT:** 04/04/2023 01:49:29 (4 minutes ago)
- REPO URL:** <https://github.com/iam-veeramalla/Jenkins-Zero-To-Hero>
- TARGET REVISION:** HEAD
- PATH:** java-maven-sonar-argocd-helm-k8s/spring-boot-app-manifests
- SYNC OPTIONS:** (empty)
- RETRY OPTIONS:** Retry disabled
- STATUS:** OutOfSync From HEAD (e27ae28)

## #- Click Diff

Firefox - https://192.168.64.19:31271/applications/default/test?view=tree&conditions=false&node=argoproj.io%2FApplication%2Fdefault%2Ftest%2F0&resource=&tab=diff

```

1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4 annotations:
5 deployment.kubernetes.io/revision: '2'
6 kubectl.kubernetes.io/last-applied-configuration: >
7   {"apiVersion":"apps/v1","kind":"Deployment","metadata":{"annotations":{},"labels":{"app":"sprint-boot-app","app.kubernetes.io/instance":"test","name":"test","namespace":"default"},"spec":{"replicas":2,"selector":{"matchLabels":{"app":"sprint-boot-app"}}, "template":{"metadata":{"labels":{"app":"sprint-boot-app"}}, "spec":{"containers":[{"image": "abhishekf5/ultimate-cicd:1","name":"sprint-boot-app","ports":[{"containerPort":8080}]}]}}}
8   generation: 2
9   labels:
10  app: sprint-boot-app
11  app.kubernetes.io/instance: test
12  managedFields:
13    - apiVersion: apps/v1
14      fieldsType: FieldsV1
15      fieldsV1:

```

```

1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4 annotations:
5 deployment.kubernetes.io/revision: '2'
6 kubectl.kubernetes.io/last-applied-configuration: >
7   {"apiVersion":"apps/v1","kind":"Deployment","metadata":{"annotations":{},"labels":{"app":"sprint-boot-app","app.kubernetes.io/instance":"test","name":"test","namespace":"default"},"spec":{"replicas":2,"selector":{"matchLabels":{"app":"sprint-boot-app"}}, "template":{"metadata":{"labels":{"app":"sprint-boot-app"}}, "spec":{"containers":[{"image": "abhishekf5/ultimate-cicd:1","name":"sprint-boot-app","ports":[{"containerPort":8080}]}]}}}
8   generation: 2
9   labels:
10  app: sprint-boot-app
11  app.kubernetes.io/instance: test
12  managedFields:
13    - apiVersion: apps/v1
14      fieldsType: FieldsV1
15      fieldsV1:

```

Firefox - https://192.168.64.19:31271/applications/default/test?view=tree&conditions=false&node=argoproj.io%2FApplication%2Fdefault%2Ftest%2F0&resource=&tab=diff

```

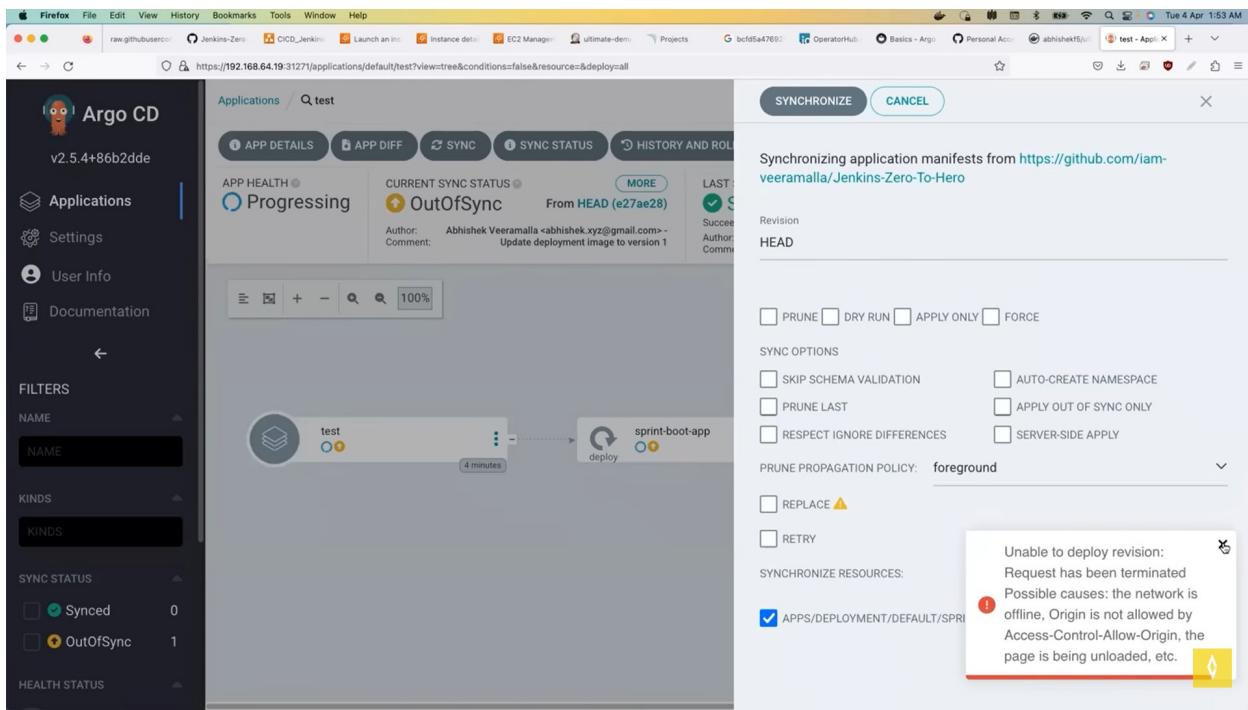
123  maxSurge: 25%
124  maxUnavailable: 25%
125  type: RollingUpdate
126  template:
127  metadata:
128  creationTimestamp: null
129  labels:
130  app: sprint-boot-app
131  spec:
132  containers:
133    - image: 'abhishekf5/ultimate-cicd:2'
134    imagePullPolicy: IfNotPresent
135    name: sprint-boot-app
136    ports:
137      - containerPort: 8080
138      protocol: TCP
139    resources: {}
140    terminationMessagePath: /dev/termination-log
141    terminationMessagePolicy: File
142    dnsPolicy: ClusterFirst
143    restartPolicy: Always
144    schedulerName: default-scheduler
145    securityContext: {}
146    terminationGracePeriodSeconds: 30
147    status:
148    availableReplicas: 2
149    conditions:
150      - lastTransitionTime: '2023-04-03T20:21:01Z'
151    lastUpdateTime: '2023-04-03T20:21:01Z'

```

```

123  maxSurge: 25%
124  maxUnavailable: 25%
125  type: RollingUpdate
126  template:
127  metadata:
128  creationTimestamp: null
129  labels:
130  app: sprint-boot-app
131  spec:
132  containers:
133    - image: 'abhishekf5/ultimate-cicd:1'
134    imagePullPolicy: IfNotPresent
135    name: sprint-boot-app
136    ports:
137      - containerPort: 8080
138      protocol: TCP
139    resources: {}
140    terminationMessagePath: /dev/termination-log
141    terminationMessagePolicy: File
142    dnsPolicy: ClusterFirst
143    restartPolicy: Always
144    schedulerName: default-scheduler
145    securityContext: {}
146    terminationGracePeriodSeconds: 30
147    status:
148    availableReplicas: 2
149    conditions:
150      - lastTransitionTime: '2023-04-03T20:21:01Z'
151    lastUpdateTime: '2023-04-03T20:21:01Z'

```



## # Argocd Processing will revoke any changes made before – Automatically

