

# **Case Study Report:**

## **DevOps CI/CD Pipeline Implementation Using Git, Jenkins, Maven, Docker, Ansible, and Kubernetes**

### **1. Introduction**

#### **1.1 Overview of DevOps, Continuous Integration (CI), and Continuous Delivery/Deployment (CD)**

##### **DevOps**

DevOps is a modern software engineering approach that emphasizes collaboration between development and operations teams. Its primary objective is to streamline the software development lifecycle by promoting automation, continuous monitoring, and faster delivery of high-quality software. DevOps fosters a culture of shared responsibility, iterative improvement, and operational excellence.

##### **Continuous Integration (CI)**

Continuous Integration refers to the practice where developers frequently commit code changes to a shared repository. Each commit triggers an automated build and testing process, enabling early detection of integration issues. CI helps maintain code stability and ensures that new changes work well with the existing codebase.

##### **Continuous Delivery and Continuous Deployment (CD)**

- Continuous Delivery ensures that applications are always in a deployable state. Validated code changes can be released to production at any time with minimal manual intervention.
- Continuous Deployment extends this concept by automating the entire release process. Every successful code change that passes all tests is automatically deployed to production, eliminating manual steps entirely.

#### **1.2 Importance of CI/CD in Modern Software Development**

In today's fast-paced and highly competitive digital landscape, rapid innovation and frequent feature releases are essential. CI/CD plays a vital role in enabling these capabilities by:

- Automating repetitive tasks such as building, testing, and deployment.
- Reducing integration challenges and shortening release cycles.
- Improving software quality through early bug detection and resolution.
- Enabling faster feedback loops for continuous improvement.

CI/CD is a fundamental component of Agile and DevOps methodologies and is critical for delivering scalable, reliable, and maintainable software solutions

### 1.3 Objective of the Case Study

This case study aims to:

- Design and implement a fully automated DevOps pipeline.
- Integrate industry-standard tools for source control, build automation, configuration management, containerization, and orchestration.
- Demonstrate end-to-end automation of the software delivery process—from code commit to deployment on a Kubernetes cluster.

### 1.4 Tools and Technologies Overview

Tool	Purpose
Git/GitHub	Source code version control and collaboration
Jenkins	Automation server for CI/CD pipeline execution
Maven	Build automation tool for Java-based projects
Tomcat	Web application server for deploying Java apps
Docker	Containerization platform for consistent environments
Ansible	Configuration management and automation
Kubernetes	Container orchestration platform for deployment and scaling

## 2. Scope of the Case Study

This case study serves as a comprehensive guide to understanding and implementing CI/CD pipelines within the DevOps lifecycle. Key areas of focus include:

### 2.1 Fundamentals of CI/CD in the DevOps Lifecycle

- **Continuous Integration (CI):** Explains how developers integrate their code frequently into a shared repository, triggering automated builds and tests to ensure smooth integration with the existing system.
- **Continuous Delivery and Deployment (CD):** Provides insights into the processes that automatically deliver or deploy validated code to staging or

production environments. The distinction between delivery (manual trigger) and deployment (fully automated) highlights different levels of pipeline automation.

- **DevOps Lifecycle:** Describes the broader DevOps lifecycle, which includes planning, coding, building, testing, releasing, deploying, operating, and monitoring software systems.

## 2.2 Required Tools and Resources

To implement an efficient CI/CD pipeline, the following tools and resources are introduced:

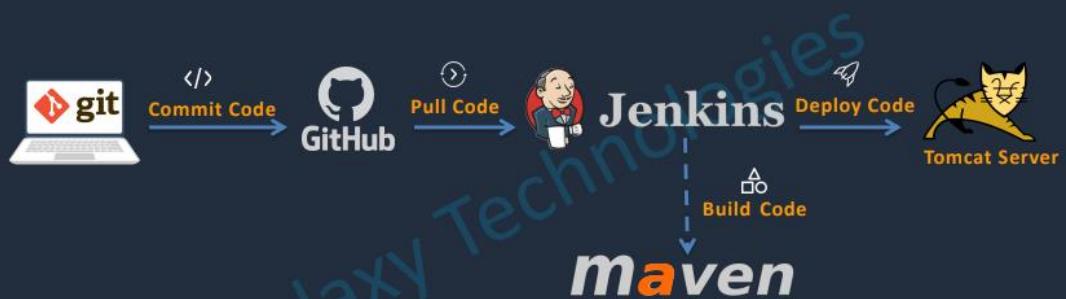
- **Version Control Systems:** Tools such as Git and platforms like GitHub for managing source code and collaboration.
- **Build Automation Tools:** Tools like Maven to automate compiling, testing, and packaging of software.
- **CI/CD Orchestration Tools:** Jenkins or similar tools to automate build, test, and deployment processes.
- **Containerization Tools:** Docker to create portable and consistent environments for development, testing, and production.
- **Configuration Management Tools:** Ansible to automate the setup and configuration of infrastructure and applications.
- **Container Orchestration Platforms:** Kubernetes to deploy, manage, and scale containerized applications efficiently.
- **Cloud Platforms:** AWS, Azure, or Google Cloud for hosting infrastructure and enabling dynamic scalability.

## 1. Deploying Java Application in Tomcat server

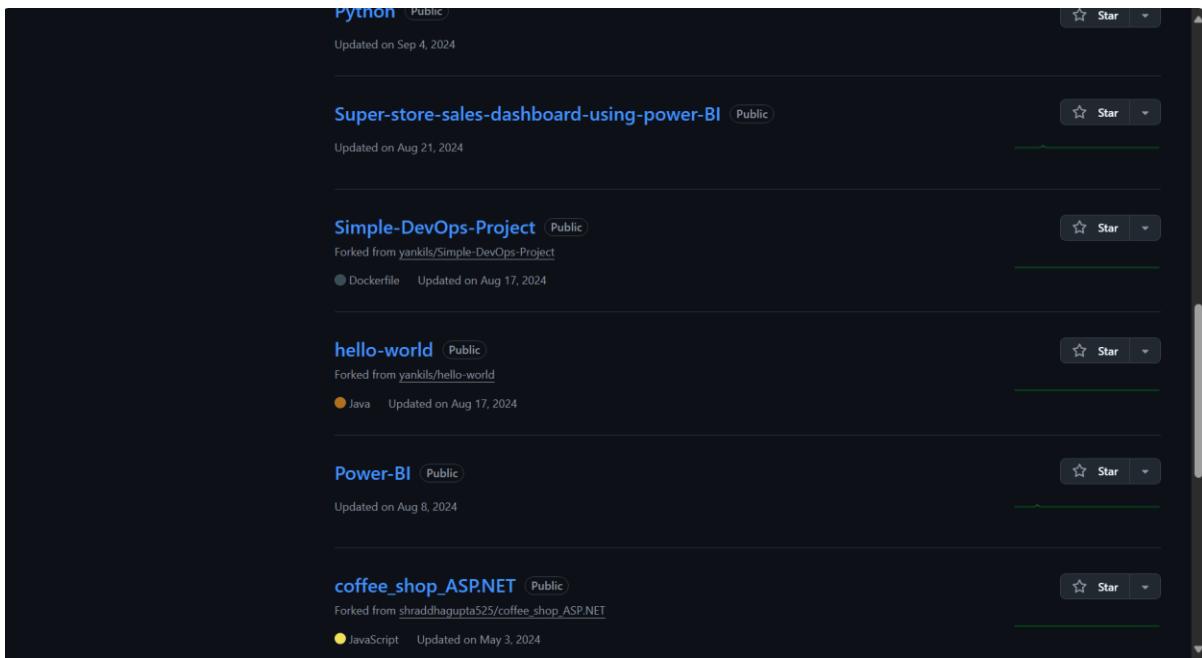
# Build and Deploy on Tomcat Server

- Setup CI/CD with GitHub, Jenkins, Maven and Tomcat
  - Setup Jenkins
  - Setup & configure Maven and Git
  - Setup Tomcat Server
  - Integrating GitHub, Maven, Tomcat Server with Jenkins
  - Create a CI and CD job
  - Test the deployment

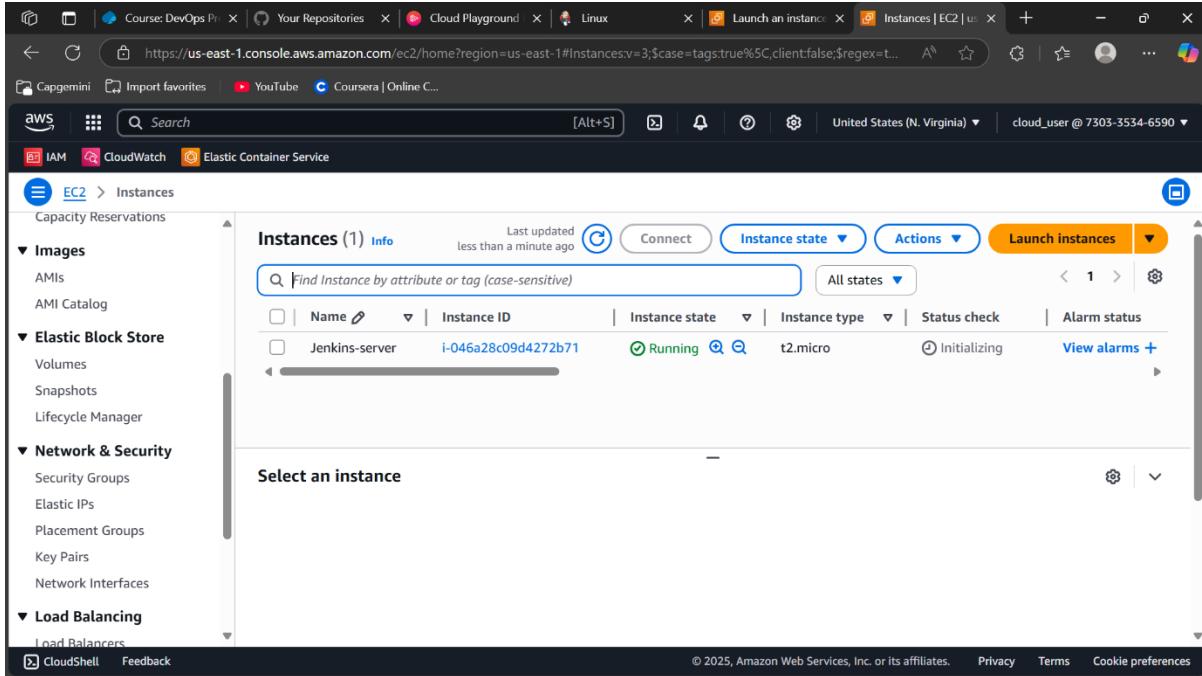
# Deploy Artifacts on a Tomcat Server



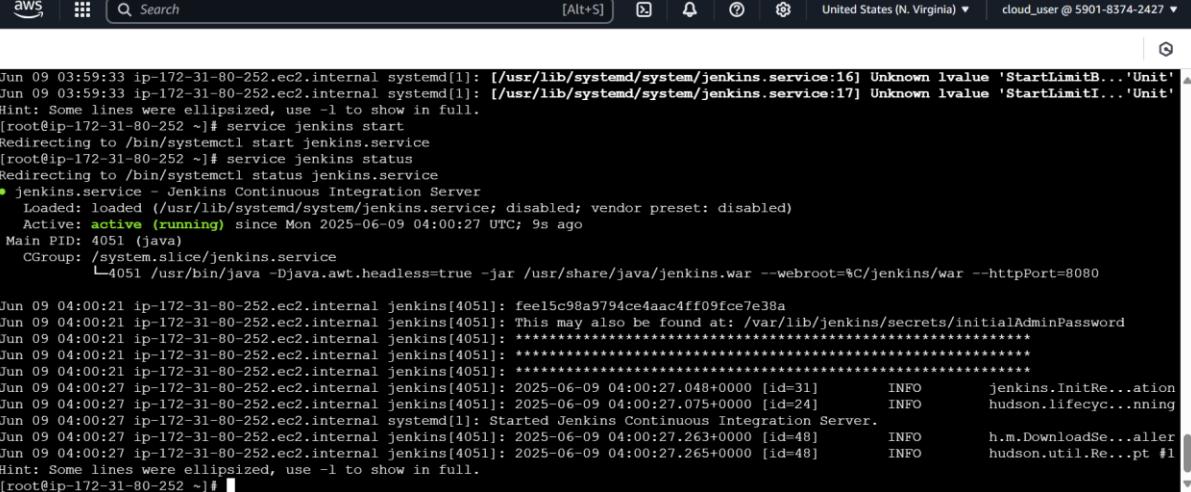
- Technologies Used - Git - Jenkins - Maven - Tomcat Server
- Forked the java application named hello-world in my github account



- Created an ec2 instance named Jenkins server with the configuration and security settings



- Installed java and Jenkins in the Jenkins server instance

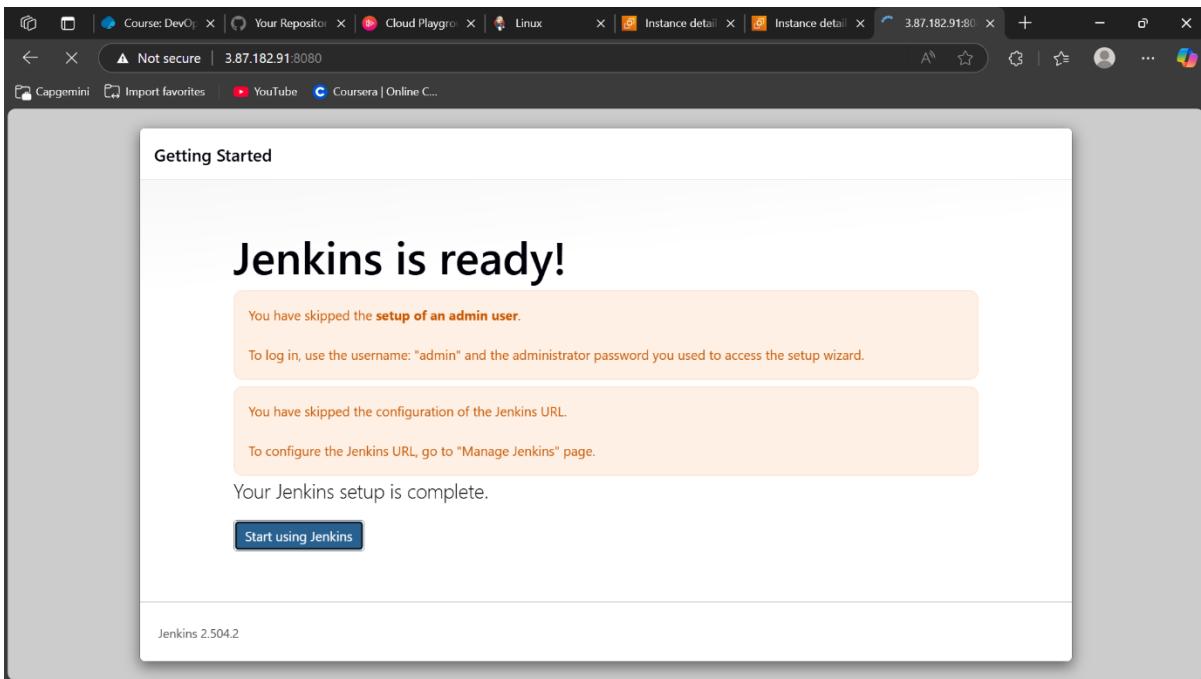


```

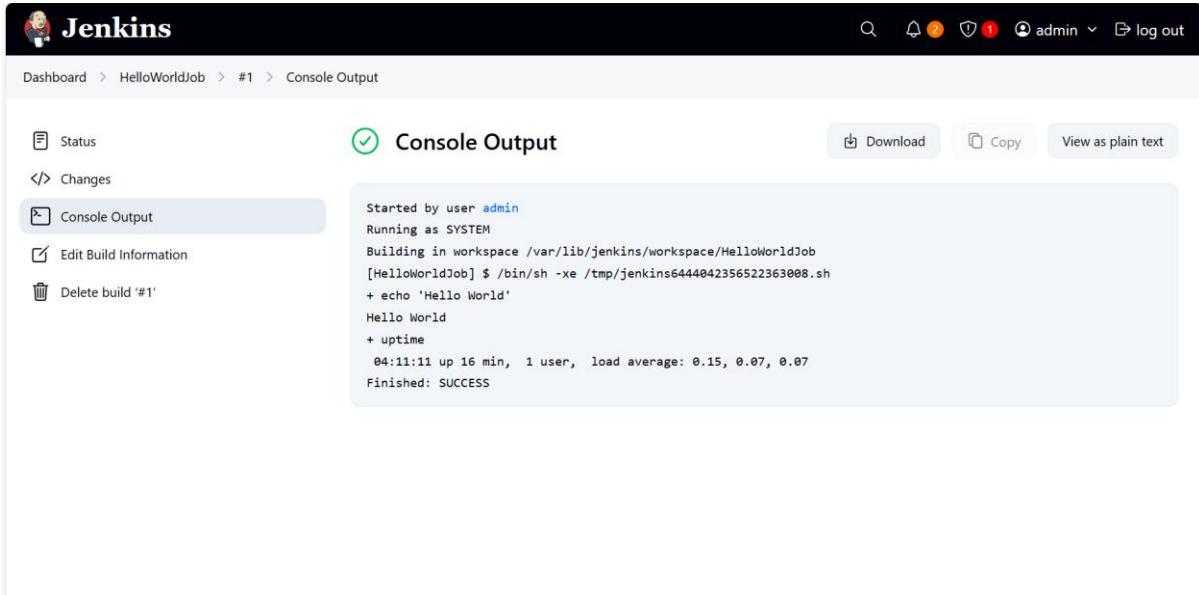
aws AWS CloudWatch Search [Alt+S] United States (N. Virginia) | cloud_user @ 5901-8374-2427
Jun 09 03:59:33 ip-172-31-80-252.ec2.internal systemd[1]: [/usr/lib/systemd/system/jenkins.service:16] Unknown lvalue 'StartLimitB... 'Unit'
Jun 09 03:59:33 ip-172-31-80-252.ec2.internal systemd[1]: [/usr/lib/systemd/system/jenkins.service:17] Unknown lvalue 'StartLimitI... 'Unit'
Hint: Some lines were ellipsized, use -l to show in full.
[root@ip-172-31-80-252 ~]# service jenkins start
Redirecting to /bin/systemctl start jenkins.service
[root@ip-172-31-80-252 ~]# service jenkins status
Redirecting to /bin/systemctl status jenkins.service
● jenkins.service - Jenkins Continuous Integration Server
  Loaded: loaded (/usr/lib/systemd/system/jenkins.service; disabled; vendor preset: disabled)
  Active: active (running) since Mon 2025-06-09 04:00:27 UTC; 9s ago
    Main PID: 4051 (java)
      CGroup: /system.slice/jenkins.service
          └─4051 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=%C/jenkins/war --httpPort=8080

Jun 09 04:00:21 ip-172-31-80-252.ec2.internal jenkins[4051]: feel15c98a9794ce4aac4ff09fce7e38a
Jun 09 04:00:21 ip-172-31-80-252.ec2.internal jenkins[4051]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Jun 09 04:00:21 ip-172-31-80-252.ec2.internal jenkins[4051]: ****
Jun 09 04:00:21 ip-172-31-80-252.ec2.internal jenkins[4051]: ****
Jun 09 04:00:21 ip-172-31-80-252.ec2.internal jenkins[4051]: ****
Jun 09 04:00:27 ip-172-31-80-252.ec2.internal jenkins[4051]: 2025-06-09 04:00:27.048+0000 [id=31]           INFO      jenkins.InitRe...ation
Jun 09 04:00:27 ip-172-31-80-252.ec2.internal jenkins[4051]: 2025-06-09 04:00:27.075+0000 [id=24]           INFO      hudson.lifecyc...nning
Jun 09 04:00:27 ip-172-31-80-252.ec2.internal systemd[1]: Started Jenkins Continuous Integration Server.
Jun 09 04:00:27 ip-172-31-80-252.ec2.internal jenkins[4051]: 2025-06-09 04:00:27.263+0000 [id=48]           INFO      h.m.DownloadSe...aller
Jun 09 04:00:27 ip-172-31-80-252.ec2.internal jenkins[4051]: 2025-06-09 04:00:27.265+0000 [id=48]           INFO      hudson.util.Re...pt #1
Hint: Some lines were ellipsized, use -l to show in full.
[root@ip-172-31-80-252 ~]#

```



- Created my first Jenkins job named Helloworldjob that printed “Hello World”.Created this job to check whether the Jenkins setup was working properly or not



The screenshot shows the Jenkins interface for a build named 'HelloWorldJob' run #1. The 'Console Output' tab is selected. The log output is as follows:

```

Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/HelloWorldJob
[HelloWorldJob] $ /bin/sh -xe /tmp/jenkins6444042356522363008.sh
+ echo 'Hello World'
Hello World
+ uptime
04:11:11 up 16 min, 1 user, load average: 0.15, 0.07, 0.07
Finished: SUCCESS

```

Below this, there is a larger block of log output from a terminal session:

```

Jun 09 04:00:21 ip-172-31-80-252.ec2.internal jenkins[4051]: fee15c98a9794ce4aac4ff09fce7e38a
Jun 09 04:00:21 ip-172-31-80-252.ec2.internal jenkins[4051]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Jun 09 04:00:21 ip-172-31-80-252.ec2.internal jenkins[4051]: ****
Jun 09 04:00:21 ip-172-31-80-252.ec2.internal jenkins[4051]: ****
Jun 09 04:00:21 ip-172-31-80-252.ec2.internal jenkins[4051]: ****
Jun 09 04:00:27 ip-172-31-80-252.ec2.internal jenkins[4051]: 2025-06-09 04:00:27.048+0000 [id=31] INFO jenkins.InitRe...ation
Jun 09 04:00:27 ip-172-31-80-252.ec2.internal jenkins[4051]: 2025-06-09 04:00:27.075+0000 [id=24] INFO hudson.lifecycle...
Jun 09 04:00:27 ip-172-31-80-252.ec2.internal systemd[1]: Started Jenkins Continuous Integration Server.
Jun 09 04:00:27 ip-172-31-80-252.ec2.internal jenkins[4051]: 2025-06-09 04:00:27.263+0000 [id=48] INFO h.m.DownloadSe...aller
Jun 09 04:00:27 ip-172-31-80-252.ec2.internal jenkins[4051]: 2025-06-09 04:00:27.265+0000 [id=48] INFO hudson.util.Re...pt #1
Hint: Some lines were ellipsized, use -l to show in full.
root@ip-172-31-80-252 ~]# cat /var/lib/jenkins/secrets/initialAdminPassword
fee15c98a9794ce4aac4ff09fce7e38a
root@ip-172-31-80-252 ~]# uptime
04:11:27 up 17 min, 1 user, load average: 0.11, 0.07, 0.06
root@ip-172-31-80-252 ~]#

```

- Later I installed github plugin to pull the code from github repository to jenkins

The screenshot shows the Jenkins 'Manage Jenkins > Plugins' page. The left sidebar has tabs for 'Updates', 'Available plugins', 'Installed plugins' (which is selected), and 'Advanced settings'. A search bar at the top right says 'Search installed plugins'. Below it is a list of installed GitHub-related plugins:

- GitHub API Plugin** 1.321-488.v9b\_c0da\_9533f8: This plugin provides GitHub API for other plugins. Status: Enabled (green switch).
- GitHub plugin** 1.43.0: This plugin integrates GitHub to Jenkins. Status: Enabled (green switch).
- Gson API Plugin** 2.13.1-139.v4569c2ef303f: This plugin provides the Gson APIs (v2.13.1) for other plugins. Status: Enabled (green switch).
- Instance Identity** 203.v15e81a\_1b\_7a\_38: Maintains an RSA key pair that can serve as a foundation of authentication when communicating with Jenkins. Status: Enabled (green switch).
- Jackson 2 API Plugin** 2.18.3-402.v74c4eb\_f122b\_2: This plugin exposes the Jackson 2 JSON APIs to other Jenkins plugins. Status: Enabled (green switch).
- Jakarta Activation API** 2.1.3-2

At the bottom left is a link to <https://plugins.jenkins.io/github>.

- And configured it accordingly.

The screenshot shows the Jenkins 'Manage Jenkins > Tools' page. Under 'Git installations', there is a configuration for a 'Git' entry:

- Name**: Git (input field containing 'Git').
- Path to Git executable**: git (input field containing 'git').
- Install automatically**: An unchecked checkbox with a tooltip 'Help for feature: Install automatically'.

At the bottom are 'Save' and 'Apply' buttons.

- Configured maven and java in the configure.xml file in the Jenkins server on aws and installed the plugins named maven in jenkins

**JDK**

Name: java-17

JAVA\_HOME: /usr/lib/jvm/java-17-amazon-corretto.x86\_64

**!** /usr/lib/jvm/java-17-amazon-corretto.x86\_64 doesn't look like a JDK directory

Install automatically ?

Add JDK

Save Apply

- Configured them accordingly in the jenkins

**Maven**

Name: maven

MAVEN\_HOME: /opt/maven

Install automatically ?

Add Maven

Save Apply

Jenkins 2.504.2

- Later created my first maven project to pull the code from github and build the code using maven

New Item

Enter an item name  
FirstMavenProject

Select an item type

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

**Maven project**  
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

If you want to create a new item from other existing, you can use this option:

Copy from **OK**

Configure

Pre Steps

Add pre-build step ▾

General

Source Code Management

Triggers

Environment

**Pre Steps**

Build

Root POM ?  
pom.xml

Goals and options ?  
clean install

Advanced ▾

Post Steps

Save Apply

```

[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 16.027 s
[INFO] Finished at: 2025-06-08T09:23:43Z
[INFO] -----
Waiting for Jenkins to finish collecting data
[JENKINS] Archiving /var/lib/jenkins/workspace/FirstMavenProject/webapp/pom.xml to com.example.maven-project/webapp/1.0-SNAPSHOT/webapp-1.0-SNAPSHOT.pom
[JENKINS] Archiving /var/lib/jenkins/workspace/FirstMavenProject/webapp/target/webapp.war to com.example.maven-project/webapp/1.0-SNAPSHOT/webapp-1.0-SNAPSHOT.war
[JENKINS] Archiving /var/lib/jenkins/workspace/FirstMavenProject/server/pom.xml to com.example.maven-project/server/1.0-SNAPSHOT/server-1.0-SNAPSHOT.pom
[JENKINS] Archiving /var/lib/jenkins/workspace/FirstMavenProject/server/target/server.jar to com.example.maven-project/server/1.0-SNAPSHOT/server-1.0-SNAPSHOT.jar
[JENKINS] Archiving /var/lib/jenkins/workspace/FirstMavenProject/pom.xml to com.example.maven-project/maven-project/1.0-SNAPSHOT/maven-project-1.0-SNAPSHOT.pom
channel stopped
Finished: SUCCESS

```

REST API Jenkins 2.504.2

Dashboard > First maven project > Workspace of First maven project on Built-In Node

**Workspace of First maven project on Built-In Node**

- Status
- </> Changes
- Workspace**
- Wipe Out Current Workspace
- Build Now
- Configure
- Delete Maven project
- Modules
- Rename

First maven project /

- git
- server
- webapp
  - Dockerfile
  - pom.xml
  - README.md
  - regapp-deploy.yml
  - regapp-service.yml

(all files in zip)

Build	Date	Size	Actions
#5	Jun 9, 2025, 5:20:31 AM	130 B	View Log

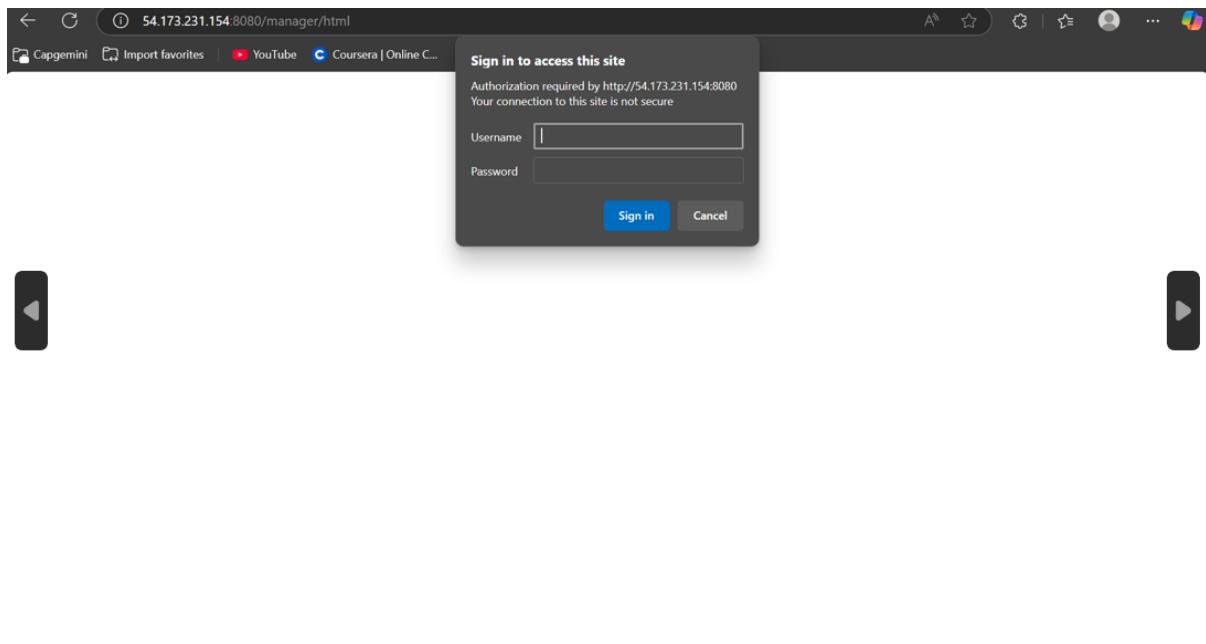
- Created an instance named Tomcat-server and installed java and tomcat to deploy the java application

## 1. Install apache tomcat:

Sudo yum install tomcat

The screenshot shows the 'Launch an instance' configuration page. In the 'Name and tags' section, the name 'Tomcat\_server' is entered. Under 'Application and OS Images (Amazon Machine Image)', a search bar is shown with the placeholder 'Search our full catalog including 1000s of application and OS images'. The 'Software Image (AMI)' is set to 'Amazon Linux 2023 AMI 2023.7.2...'. The 'Virtual server type (instance type)' is 't2.micro'. The 'Firewall (security group)' is 'New security group'. On the right, there's a summary panel with 'Number of instances' set to 1, and a large orange 'Launch instance' button.

The screenshot shows the Apache Tomcat 10.1.41 welcome page at the URL 44.210.126.154:8080. A green banner at the top says 'If you're seeing this, you've successfully installed Tomcat. Congratulations!'. Below it, there's a cartoon cat icon and a list of 'Recommended Reading' links: 'Security Considerations How-To', 'Manager Application How-To', and 'Clustering/Session Replication How-To'. To the right, there are three buttons: 'Server Status', 'Manager App', and 'Host Manager'. The main content area has sections for 'Developer Quick Start', 'Documentation', and 'Getting Help'. The 'Developer Quick Start' section includes links for 'Tomcat Setup', 'First Web Application', 'Realms & AAA', 'JDBC DataSources', 'Examples', and 'Servlet Specifications'. The 'Documentation' section links to 'Tomcat 10.1 Documentation', 'Tomcat 10.1 Configuration', and 'Tomcat Wiki'. The 'Getting Help' section links to 'FAQ and Mailing Lists' and lists several mailing lists: 'tomcat-announce', 'tomcat-users', 'taglibs-user', and 'tomcat-dev'.



- Added manager script credentials in Jenkins to verify successful connection between Jenkins and tomcat.

Path	Version	Display Name	Running	Sessions	Commands
/	<i>None specified</i>	Welcome to Tomcat	true	0	<a href="#">Start</a> <a href="#">Stop</a> <a href="#">Reload</a> <a href="#">Undeploy</a> <a href="#">Expire sessions</a> with idle ≥ 30 minutes
/docs	<i>None specified</i>	Tomcat Documentation	true	0	<a href="#">Start</a> <a href="#">Stop</a> <a href="#">Reload</a> <a href="#">Undeploy</a> <a href="#">Expire sessions</a> with idle ≥ 30 minutes
/examples	<i>None specified</i>	Servlet and JSP Examples	true	0	<a href="#">Start</a> <a href="#">Stop</a> <a href="#">Reload</a> <a href="#">Undeploy</a> <a href="#">Expire sessions</a> with idle ≥ 30 minutes
/host-manager	<i>None specified</i>	Tomcat Host Manager Application	true	0	<a href="#">Start</a> <a href="#">Stop</a> <a href="#">Reload</a> <a href="#">Undeploy</a> <a href="#">Expire sessions</a> with idle ≥ 30 minutes Start Stop Reload Undeploy

- Install the plugin named Deploy to container to deploy our java application.

Install	Name	Released
<input checked="" type="checkbox"/>	Deploy to container 1.17	26 days ago
<input type="checkbox"/>	Docker Pipeline 621.va.73f881d9232	8 days 21 hr ago
<input type="checkbox"/>	Artifactory 4.0.8	11 mo ago

New credentials

Kind

Username with password

Scope

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

Username

deployer

Treat username as secret

Password

\*\*\*\*\*

ID

Create

The screenshot shows the Jenkins Global credentials (unrestricted) page. At the top, there is a navigation bar with links to Dashboard, Manage Jenkins, Credentials, System, and Global credentials (unrestricted). A search bar and a user icon for 'admin' are also present. Below the navigation, the title 'Global credentials (unrestricted)' is displayed, along with a blue button '+ Add Credentials'. A table lists a single credential entry:

ID	Name	Kind	Description
tomcat_deployer	deployer/******** (tomcat_deployer)	Username with password	tomcat_deployer

Below the table, there is a legend for icons: S (Small), M (Medium), and L (Large). At the bottom right, there are links for 'REST API' and 'Jenkins 2.504.2'.

- Created an Jenkins pipeline named Build and deploy job to deploy the java application using Jenkins and tomcat server.

The screenshot shows the Jenkins New Item creation page. The title 'New Item' is at the top. A text input field 'Enter an item name' contains 'BuildAndDeployJob'. Below it, a section 'Select an item type' shows two options:

- Freestyle project**: Described as a 'Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.'
- Maven project**: Described as 'Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.'

A note at the bottom says 'If you want to create a new item from other existing, you can use this option:' followed by a 'Create from...' link. A blue 'OK' button is at the bottom left.

Not secure | 3.87.182.91:8080/job/BuildAndDeployJob/configure

C Capgemini Import favorites YouTube Coursera | Online C...

Dashboard > BuildAndDeployJob > Configuration

## Configure

### General

Enabled

Description  
Build code with the help maven and deploy it to Tomcat server

Plain text [Preview](#)

Discard old builds ?

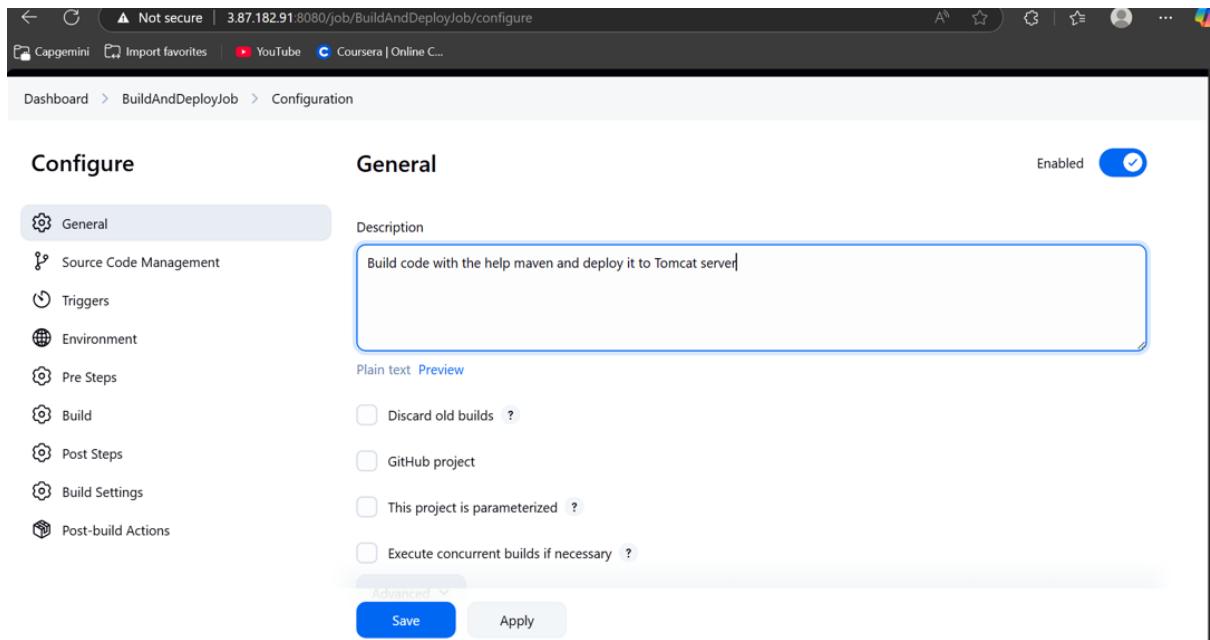
GitHub project

This project is parameterized ?

Execute concurrent builds if necessary ?

Advanced [▼](#)

[Save](#) [Apply](#)



Not secure | 3.87.182.91:8080/job/BuildAndDeployJob/configure

C Capgemini Import favorites YouTube Coursera | Online C...

Dashboard > BuildAndDeployJob > Configuration

## Configure

### Post Steps

Run regardless of build result  
Should the post-build steps run only for successful builds, etc.

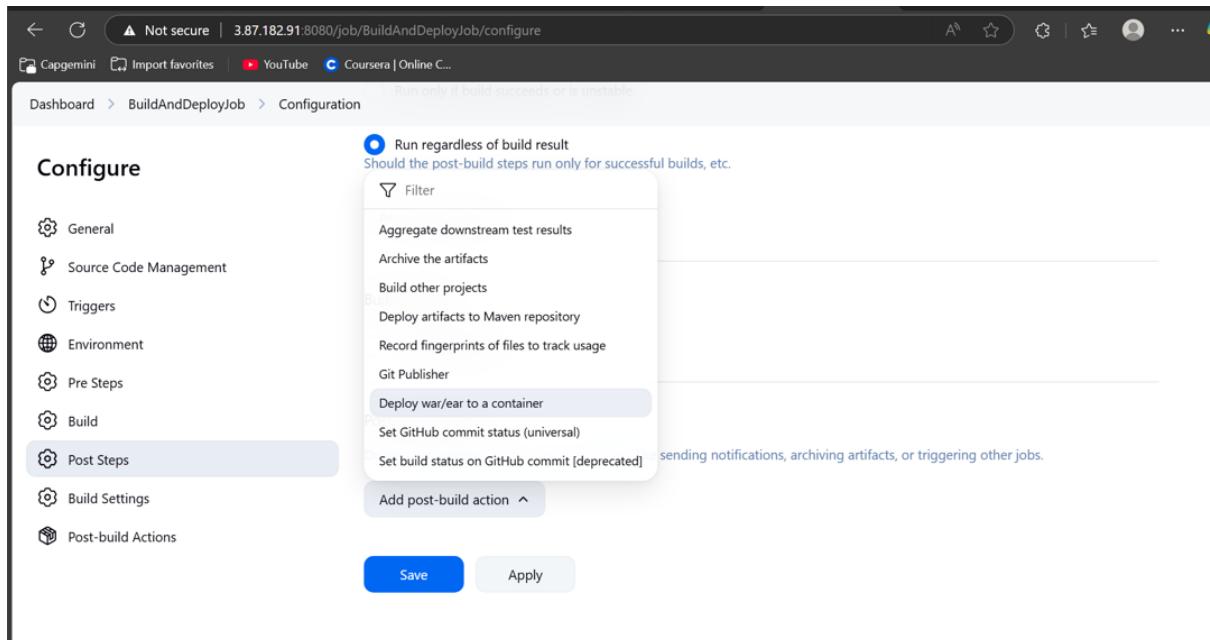
Filter

- Aggregate downstream test results
- Archive the artifacts
- Build other projects
- Deploy artifacts to Maven repository
- Record fingerprints of files to track usage
- Git Publisher
- Deploy war/ear to a container
- Set GitHub commit status (universal)
- Set build status on GitHub commit [deprecated]

Add post-build action ^

sending notifications, archiving artifacts, or triggering other jobs.

[Save](#) [Apply](#)



```

Not secure | 3.87.182.91:8080/job/BuildAndDeployJob/1/console
Capgemini Import favorites YouTube Coursera | Online C...
Dashboard > BuildAndDeployJob > #1 > Console Output

Waiting for Jenkins to finish collecting data
[JENKINS] Archiving /var/lib/jenkins/workspace/BuildAndDeployJob/webapp/pom.xml to com.example.maven-project/webapp/1.0-SNAPSHOT/webapp-1.0-SNAPSHOT.pom
[JENKINS] Archiving /var/lib/jenkins/workspace/BuildAndDeployJob/webapp/target/webapp.war to com.example.maven-project/webapp/1.0-SNAPSHOT/webapp-1.0-SNAPSHOT.war
[JENKINS] Archiving /var/lib/jenkins/workspace/BuildAndDeployJob/server/pom.xml to com.example.maven-project/server/1.0-SNAPSHOT/server-1.0-SNAPSHOT.pom
[JENKINS] Archiving /var/lib/jenkins/workspace/BuildAndDeployJob/server/target/server.jar to com.example.maven-project/server/1.0-SNAPSHOT/server-1.0-SNAPSHOT.jar
[JENKINS] Archiving /var/lib/jenkins/workspace/BuildAndDeployJob/pom.xml to com.example.maven-project/maven-project/1.0-SNAPSHOT/maven-project-1.0-SNAPSHOT.pom
channel stopped
[DeployPublisher][INFO] Attempting to deploy 1 war file(s)
[DeployPublisher][INFO] Deploying /var/lib/jenkins/workspace/BuildAndDeployJob/webapp/target/webapp.war to container Tomcat 8.x Remote with context null
[/var/lib/jenkins/workspace/BuildAndDeployJob/webapp/target/webapp.war] is not deployed. Doing a fresh deployment.
Deploying [/var/lib/jenkins/workspace/BuildAndDeployJob/webapp/target/webapp.war]
Finished: SUCCESS

```

REST API Jenkins 2.504.2

- To check whether the changes are updated or not we use git in this we cloned the java application to our local machine and later committed the changes to github

```

MINGW64/c/Users/valaxy/Desktop/Valaxy/DevOpsProject/hello-world/webapp/src/main/webapp
-rw-r--r-- 1 valaxy 197121 871 Nov 7 14:35 index.jsp

valaxy@LAPTOP-UTQH9GTV MINGW64 ~/Desktop/Valaxy/DevOpsProject/hello-world/webapp/src/main/webapp (master)
$ vi index.jsp

valaxy@LAPTOP-UTQH9GTV MINGW64 ~/Desktop/Valaxy/DevOpsProject/hello-world/webapp/src/main/webapp (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   index.jsp

no changes added to commit (use "git add" and/or "git commit -a")

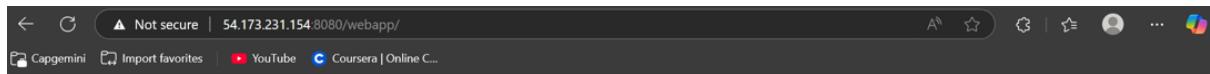
valaxy@LAPTOP-UTQH9GTV MINGW64 ~/Desktop/Valaxy/DevOpsProject/hello-world/webapp/src/main/webapp (master)
$ git add .

valaxy@LAPTOP-UTQH9GTV MINGW64 ~/Desktop/Valaxy/DevOpsProject/hello-world/webapp/src/main/webapp (master)
$ git commit -m "password field in new link in index.jsp"
[master 802c781] password field in new link in index.jsp
1 file changed, 1 insertion(+)

valaxy@LAPTOP-UTQH9GTV MINGW64 ~/Desktop/Valaxy/DevOpsProject/hello-world/webapp/src/main/webapp (master)
$ git push origin master
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.

```

- Now Jenkins pull the code from github repository and build the application



## New user Register for DevOps Learning

Please fill in this form to create an account.

Enter Name	Enter Full Name
Enter mobile	Enter mobile number
Enter Email	Enter Email
Password	Enter Password
Repeat Password	Repeat Password

By creating an account you agree to our [Terms & Privacy](#).

[Register](#)

Already have an account? [Sign in](#).

**Thankyou, Happy Learning**

## 2. Deploying java application on Docker

### Deploy Artifacts on a Container

- Setup CI/CD with GitHub, Jenkins, Maven and Docker
  - Setting up Docker environment
  - Write Dockerfile
  - Create an image and container on docker host
  - Integrate docker host with Jenkins
  - Create CI/CD job on Jenkins to build and deploy on a container

# Deploy Artifacts on a Container



## 1. Setup Docker Environment

- Install Docker Engine on the build or deployment servers (e.g., Jenkins server or AWS EC2 instances).
- Verify Docker installation with basic commands like `docker --version` and `docker run hello-world`.
- Configure Docker daemon and user permissions to allow Jenkins and other tools to interact with Docker.

**i-05f3c25729ae22bf6 (Docker-server)**

**Details** Status and alarms Monitoring Security Networking Storage Tags

**Instance summary**

Instance ID	Public IPv4 address	Private IPv4 addresses
i-05f3c25729ae22bf6	174.129.66.177   open address	172.31.91.102
IPv6 address	Instance state	Public DNS
-	Running	ec2-174-129-66-177.compute-1.amazonaws.com   open address

## 2. Create a Tomcat Container

- Use the official Tomcat Docker image to spin up a containerized Tomcat server.
- Run commands such as docker run -d -p 8080:8080 tomcat to launch the container.
- Verify the container is running and accessible via the mapped port.

```

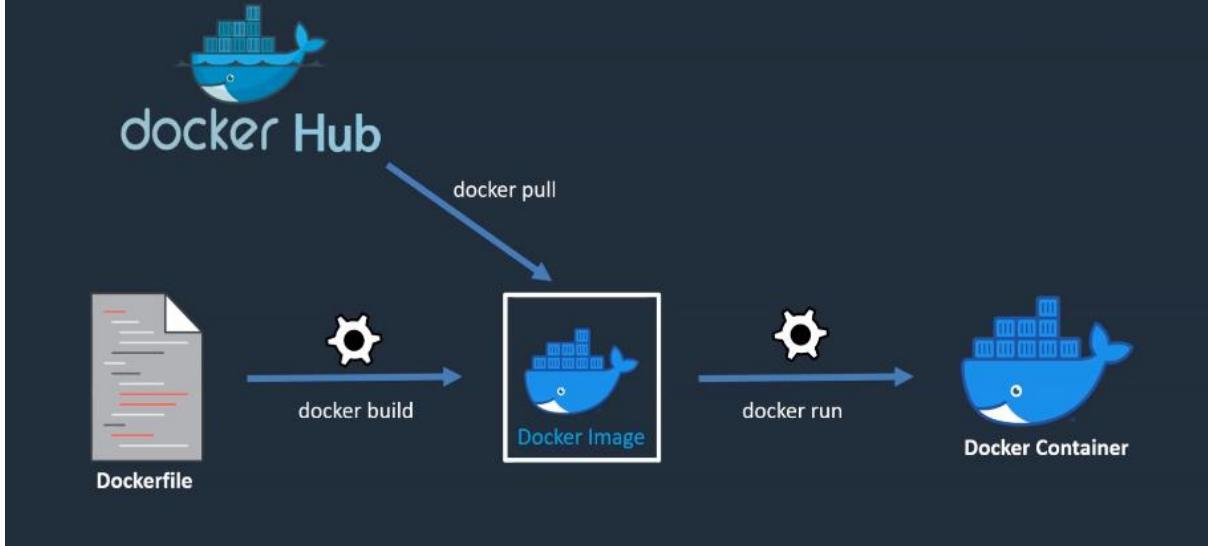
aws | Search [Alt+S] United States (N. Virginia) | cloud_user @ 8517-2516-1858

Amazon Linux 2
AL2 End of Life is 2026-06-30.
A newer version of Amazon Linux is available!
Amazon Linux 2023, GA and supported until 2028-03-15.
https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-172-31-91-102 ~]$ sudo su -
[root@ip-172-31-91-102 ~]# yum install docker -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-
:mtd
Resolving Dependencies
--> Running transaction check
--> Package docker.x86_64 0:25.0.8-1.amzn2.0.4 will be installed
--> Processing Dependency: containerd >= 1.3.2 for package: docker-25.0.8-1.amzn2.0.4.x86_64
--> Processing Dependency: libcgROUP >= 0.40.rc1-5.15 for package: docker-25.0.8-1.amzn2.0.4.x86_64
--> Processing Dependency: runc >= 1.0.0 for package: docker-25.0.8-1.amzn2.0.4.x86_64
--> Processing Dependency: pigz for package: docker-25.0.8-1.amzn2.0.4.x86_64
--> Running transaction check
--> Package containerd.x86_64 0:1.7.27-1.amzn2.0.2 will be installed
--> Package libcgROUP.x86_64 0:0.41-21.amzn2 will be installed
--> Package pigz.x86_64 0:2.3.4-1.amzn2.0.1 will be installed
--> Package runc.x86_64 0:1.2.4-1.amzn2 will be installed

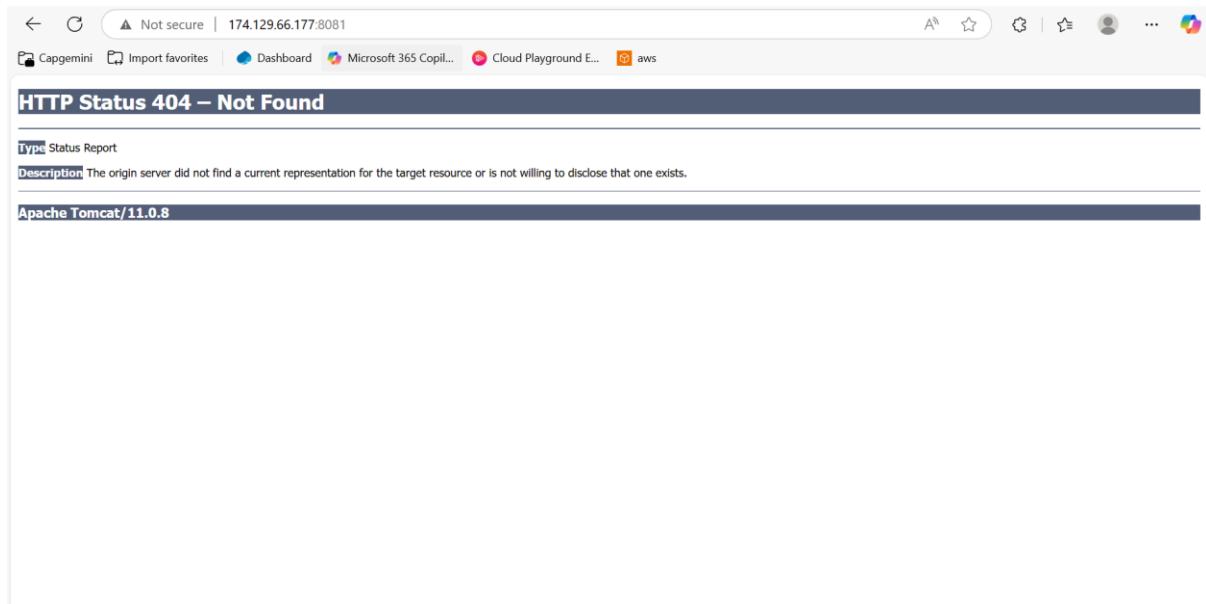
```

# How to create docker container



The screenshot shows the Docker Hub website interface. At the top, there is a search bar with the query "tomcat" and a "Copy" button. Below the search bar, there is a "Recent tags" section listing several tags: latest, jre21-temurin-noble, jre21-temurin-jammy, jre21-temurin, jre21, jre17-temurin-noble, jre17-temurin-jammy, jre17-temurin, jre17, and jdk21-temurin-noble. The main content area displays the "tomcat" image page for the "Apache Tomcat" project. It features a yellow cat icon, the Docker Official Image badge, a download count of 500M+, and a star rating of 3.7K. A brief description states that Apache Tomcat is an open source implementation of the Java Servlet and JavaServer Pages technologies. There are tabs for "Overview" and "Tags". A "Quick reference" sidebar provides links to the Docker Community and Stack Overflow. A cookie consent banner at the bottom asks for acceptance of cookies to enhance site navigation.

```
[root@dockerhost ~]# docker pull tomcat:latest
Digest: sha256:d2f9bdc5b35fc7da231df399a9cda0d49ff402053d47f008dcba499ef3bcf950
Status: Downloaded newer image for tomcat:latest
docker.io/library/tomcat:latest
[root@dockerhost ~]# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
tomcat latest 86433616469a 21 hours ago 469MB
[root@dockerhost ~]# docker run -d --name tomcat-container -p 8081:8080 tomcat
5b48db9b83237e67c3474ff16bbefefed83f75b1bac37892b3973f67daad2266
[root@dockerhost ~]# docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
5b48db9b8323 tomcat "catalina.sh run" 12 seconds ago Up 11 seconds 0.0.0.0:8081->8080/tcp, :::8081->8080/tcp tomcat-containe
r
[root@dockerhost ~]# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
5b48db9b8323 tomcat "catalina.sh run" 3 minutes ago Up 3 minutes 0.0.0.0:8081->8080/tcp, :::8081->8080/tcp tomcat-container
[root@dockerhost ~]#
```



### 3. Fixing Tomcat Container Issues

- Troubleshoot common problems such as port conflicts, volume mounts, or missing dependencies.
- Adjust Dockerfile or container settings to fix deployment errors or startup failures.
- Check container logs with docker logs <container\_id> to identify issues.

```
"docker ps" accepts no arguments.
See 'docker ps --help'.

Usage: docker ps [OPTIONS]

List containers
[root@dockerhost ~]# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
5b48db9b8323 tomcat "catalina.sh run" 8 minutes ago Up 8 minutes 0.0.0.0:8081->8080/tcp, ::8081->8080/tcp tomcat-container
[root@dockerhost ~]# docker exec -it 5b48db9b8323 /bin/bash
root@5b48db9b8323:/usr/local/tomcat# ls
bin lib README.md webapps
BUILDING.txt LICENSE RELEASE-NOTES webapps.dist
conf logs RUNNING.txt cook
CONTRIBUTING.md native-jni-lib
filtered-KEYS NOTICE upstream-KEYS
root@5b48db9b8323:/usr/local/tomcat# cd webapps.dist
root@5b48db9b8323:/usr/local/tomcat/webapps.dist# ls
docs examples host-manager manager ROOT
cp: target './web': No such file or directory
psot@5b48db9b8323:/usr/local/tomcat/webapps.dist# cp -R * ../webap
root@5b48db9b8323:/usr/local/tomcat/webapps.dist# cd ..
root@5b48db9b8323:/usr/local/tomcat# cd webapps
root@5b48db9b8323:/usr/local/tomcat/webapps# ls
docs examples host-manager manager ROOT
root@5b48db9b8323:/usr/local/tomcat/webapps#
```

i-05f3c25729ae22bf6 (Docker-server)

PublicIPs: 174.129.66.177 PrivateIPs: 172.31.91.102

The screenshot shows the Apache Tomcat 11.0.8 homepage. At the top, there's a navigation bar with links for Home, Documentation, Configuration, Examples, Wiki, and Mailing Lists. To the right is a "Find Help" search bar. Below the navigation is the Apache logo and the text "APACHE SOFTWARE FOUNDATION http://www.apache.org/". A green banner at the top says "If you're seeing this, you've successfully installed Tomcat. Congratulations!". To the left is a cartoon cat icon. To the right are three buttons: "Server Status", "Manager App", and "Host Manager". The main content area is divided into sections: "Developer Quick Start" (with links to Tomcat Setup, First Web Application, Realms & AAA, JDBC DataSources, Examples, Servlet Specifications, and Tomcat Versions), "Managing Tomcat" (with links to manager webapp, SCALINA\_HOME/conf/tomcat-users.xml, and a note about access split between users), "Documentation" (with links to Tomcat 11.0 Documentation, Tomcat 11.0 Configuration, Tomcat Wiki, SCALINA\_HOME RUNNING.txt, and developer notes about bug database and JavaDocs), and "Getting Help" (with links to FAQ and Mailing Lists, including tomcat-announce, tomcat-users, taglibs-user, and tomcat-dev).

The screenshot shows an "HTTP Status 404 – Not Found" error page. The title bar indicates the URL is 174.129.66.177:8082. The page has a dark header with "HTTP Status 404 – Not Found". Below it is a "Type" section with "Status Report" and a "Description" section stating "The origin server did not find a current representation for the target resource or is not willing to disclose that one exists." At the bottom is a "Apache Tomcat/11.0.8" footer.

# Write Your 1<sup>st</sup> Docker File

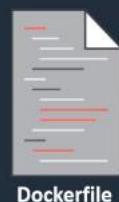


- **FROM:** To pull the base image
- **RUN:** To execute commands
- **CMD:** To provide defaults for an executing container
- **ENTRYPOINT:** To configure a container that will run as an executable
- **WORKDIR:** To sets the working directory
- **COPY:** To copy a directory from your local machine to the docker container
- **ADD:** To copy files and folders from your local machine to docker containers
- **EXPOSE:** Informs Docker that the container listens on the specified network ports at runtime
- **ENV:** To set environment variables

## 4. Create a First Dockerfile

- Write a basic Dockerfile that defines the environment and instructions to build a container image.
- Include steps like setting the base image (e.g., FROM tomcat), copying application artifacts, and exposing necessary ports.
- Build the Docker image locally using docker build -t mytomcat:latest .

# Install tomcat on Centos



- Pull centos from dockerhub
- Install java
- Create /opt/tomcat directory
- Change work directory to /opt/tomcat
- Download tomcat packages
- Extract tar.gz file
- Rename to tomcat directory
- Tell to docker that it runs on port 8080
- Start tomcat services
- **FROM**
- **RUN**
- **RUN**
- **WORKDIR**
- **ADD /RUN**
- **RUN**
- **RUN**
- **EXPOSE**
- **CMD**

## 5. Create a Customized Dockerfile for Tomcat

- Enhance the Dockerfile to include custom configurations, environment variables, or scripts needed for the application.
  - Automate deployment steps like copying WAR files into the Tomcat webapps directory within the image.
  - Optimize the Docker image size and startup time for faster deployments.

```
[root@dockerhost ~]# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
demotomcat latest db95eef31251 9 seconds ago 474MB
tomcat latest 86433616469a 22 hours ago 468MB
8085:8080 demotomcat docker run -d --name demotomcat-container -p
3a0f86423cfabf593cdd20bbfa20e1cd1dbld609afde0ead2bc00f44c7d94c9
[root@dockerhost ~]# docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
3a0f86423cfa demotomcat "catalina.sh run" 5 seconds ago Up 4 seconds 0.0.0.0:8085->8080/tcp, :::8085->8080/tcp
demotomcat-container
f286e17a97de tomcat "catalina.sh run" 27 minutes ago Up 26 minutes 0.0.0.0:8082->8080/tcp, :::8082->8080/tcp
tomcat2
5b48db9b8323 tomcat "catalina.sh run" 42 minutes ago Exited (143) 28 minutes ago
tomcat-container vi Dockerfile
[root@dockerhost ~]#
```

The screenshot shows the Apache Tomcat 11.0 homepage. The URL in the browser is `174.129.66.177:8085`. The page has a green header bar with the text "If you're seeing this, you've successfully installed Tomcat. Congratulations!" and a cartoon cat icon. Below the header, there are three main sections: "Developer Quick Start" (with links to Tomcat Setup, First Web Application, Realms & AAA, JDBC DataSources, Examples, Servlet Specifications, and Tomcat Versions), "Documentation" (with links to Tomcat 11.0 Documentation, Tomcat 11.0 Configuration, Tomcat Wiki, and a link to \$CATALINA\_HOME/running.txt), and "Getting Help" (with links to FAQ and Mailing Lists, including tomcat-announce, tomcat-users, taglibs-user, and tomcat-dev). There are also buttons for Server Status, Manager App, and Host Manager.

# Integrate Docker with Jenkins

- Create a dockeradmin user
- Install “Publish Over SSH” plugin
- Add Dockerhost to Jenkins “configure systems”



## 6. Integrate Docker with Jenkins

- Configure Jenkins with Docker plugins to enable Docker commands within Jenkins jobs.
- Set Jenkins agents with Docker installed or use Docker-in-Docker setups.
- Define Jenkins pipeline steps that build, tag, and push Docker images.

```
drwxr-xr-x 2 root      root      6 Aug 16  2018 rh
[root@dockerhost opt]# ls -ld
drwxr-xr-x 6 root root 59 Jun 10 07:48 .
[root@dockerhost opt]# cd ..
[root@dockerhost ]# cd /root
[root@dockerhost ~]# ll
total 4
-rw-r--r-- 1 root root 88 Jun 10 06:07 Dockerfile
[root@dockerhost ~]# mv Dockerfile /opt/docker/
[root@dockerhost ~]# cd /opt/docker/
[root@dockerhost docker]# ll -l
total 4
-rw-r--r-- 1 root root 88 Jun 10 06:07 Dockerfile
[root@dockerhost docker]# chown -R dockeradmin:dockeradmin /opt/do
cker
[root@dockerhost docker]# ll
total 4
-rw-r--r-- 1 dockeradmin dockeradmin 88 Jun 10 06:07 Dockerfile
[root@dockerhost docker]# ls
Dockerfile  webapp.war
[root@dockerhost docker]# ll
total 8
-rw-r--r-- 1 dockeradmin dockeradmin 88 Jun 10 06:07 Dockerfile
-rw-rw-r-- 1 dockeradmin dockeradmin 2358 Jun 10 07:53 webapp.war
[root@dockerhost docker]# vi Dockerfile
[root@dockerhost docker]# docker build -t webapp .
```

## 7. Jenkins Job to Build and Copy Artifacts onto Docker Host

- Automate the process of building the Java application with Maven.
- Copy the build artifact (WAR/JAR) to the Docker build context or directly to the Docker host.
- Trigger Docker image build and container deployment from Jenkins jobs.

## 8. Update Tomcat Dockerfile to Automate Deployment Process

- Modify the Dockerfile to fully automate application deployment by embedding the build artifact during image creation.
- Ensure that each new Docker image contains the latest version of the application ready for deployment.
- Automate versioning and tagging of Docker images in the pipeline.

```
demotomcat  latest    db95efe31251  2 hours ago   474MB
tomcat     latest    86433616469a  24 hours ago  468MB
[root@dockerhost docker]# docker run -d tomcatv1 -p 8086:8080 tomcat:v1
Unable to find image 'tomcatv1:latest' locally
docker: Error response from daemon: pull access denied for tomcatv1, repository does not exist or may require 'docker login': denied: requested access to the resource is denied.
See 'docker run --help'.
[root@dockerhost docker]# docker run -d --name tomcatv1 -p 8086:80
80 tomcat:v1
621b1653b6def80f136d851551d7134253ace034493c9c28d029f5eac49bad02
[root@dockerhost docker]# docker run -d --name tomcatv1 -p 8086:8080 tomcat:v1
docker: Error response from daemon: Conflict. The container name "/tomcatv1" is already in use by container "621b1653b6def80f136d851551d7134253ace034493c9c28d029f5eac49bad02". You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
[root@dockerhost docker]# docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
 NAMES
621b1653b6de  tomcat:v1  "catalina.sh run"  About a minute ago  Up About a minute  0.0.0.0:8086->8080/tcp, :::8086->8080/tcp
tomcatv1
3a0f86423cfa  demotomcat  "catalina.sh run"  2 hours ago       Up 2 hours      0.0.0.0:8085->8080/tcp, :::8085->8080/tcp
demotomcat-container
f286e17a97de  tomcat     "catalina.sh run"  2 hours ago       Up 2 hours      0.0.0.0:8082->8080/tcp, :::8082->8080/tcp
tomcat2
5b48db9b8323  tomcat     "catalina.sh run"  3 hours ago      Exited (143) 2 hours ago
tomcat-container
[root@dockerhost docker]#
```

## 9. Automate Build and Deployment on Docker Container

- Create Jenkins pipeline stages for continuous building, testing, and deploying Docker containers.
- Use Docker Compose or Kubernetes if needed to manage multi-container deployments.
- Automate container lifecycle management including start, stop, and rollback procedures.

## 10. Jenkins Job to Automate CI/CD to Deploy Application on Docker Container

- Combine all steps into a fully automated Jenkins pipeline job that listens for code changes.
- Upon each commit, Jenkins triggers build, test, Docker image creation, and deployment to the Docker environment.
- Achieve continuous integration and continuous deployment with minimal manual intervention.

New user Register for DevOps Learning

Please fill in this form to create an account.

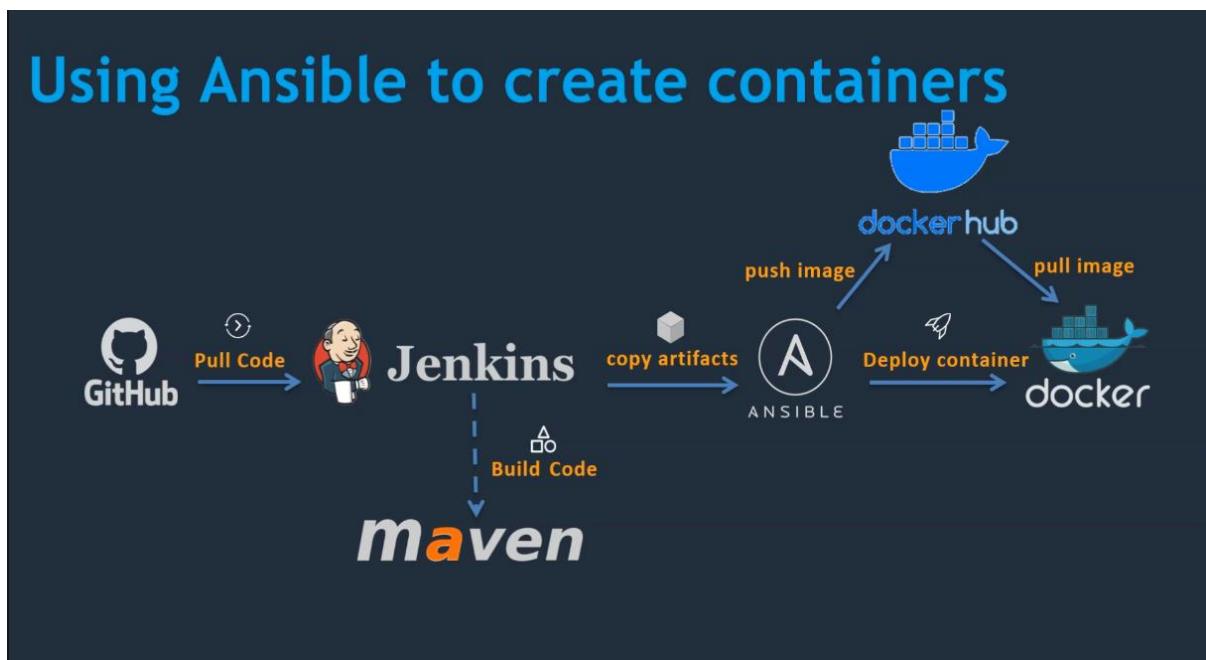
Enter Name	<input type="text"/>
Enter mobile	<input type="text"/>
Enter Email	<input type="text"/>
Password	<input type="password"/>
Repeat Password	<input type="password"/>

By creating an account you agree to our [Terms & Privacy](#).

Already have an account? [Sign in](#).

**Thankyou, Happy Learning**

### 3. Integrating Ansible in CI/CD Pipeline



# Prepare Ansible Server

- Setup EC2 instance
- Setup hostname
- Create ansadmin user
- Add user to sudoers file
- Generate ssh keys
- Enable password based login
- Install ansible



ANSIBLE

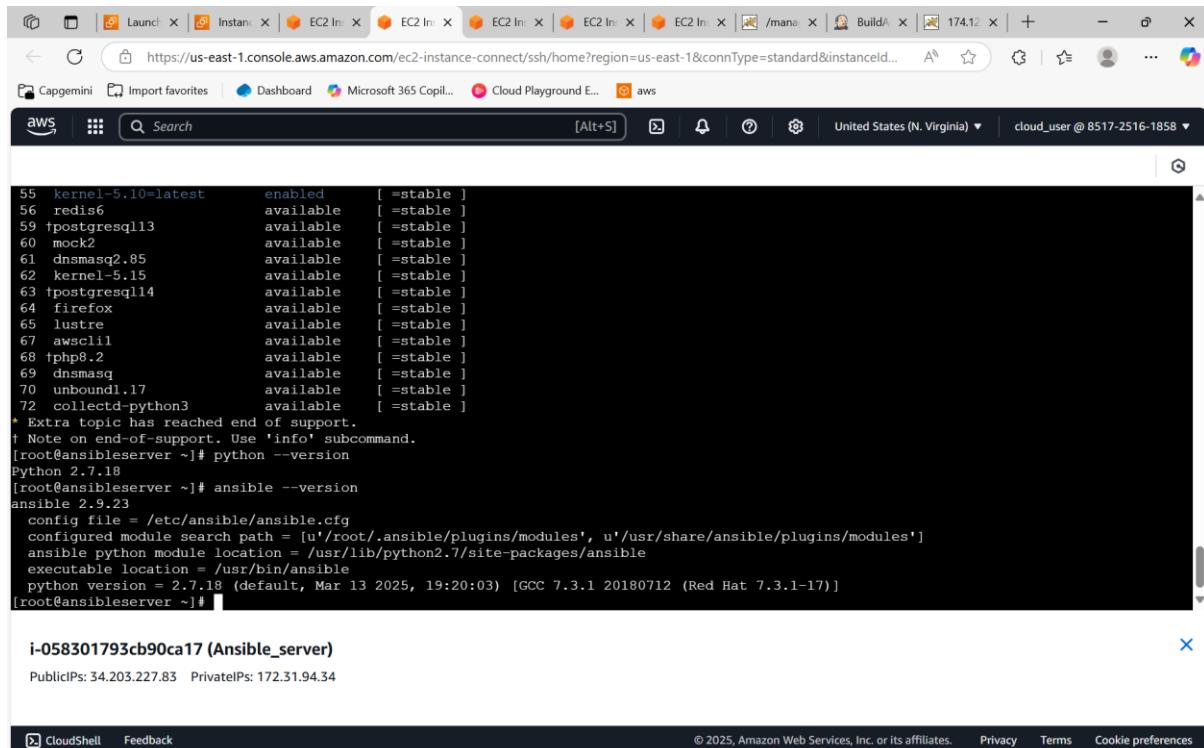
## 1. Section Introduction – Why Do We Need Ansible?

- Ansible is a powerful automation tool that simplifies configuration management, application deployment, and orchestration.
- It helps manage infrastructure as code with easy-to-write YAML playbooks, reducing manual intervention and errors.
- Integrating Ansible into CI/CD pipelines allows automated, repeatable, and consistent deployment processes across environments.

The screenshot shows the AWS CloudWatch Metrics Insights interface. At the top, there's a search bar with the query: `CloudWatch Metrics Insights usage`. Below the search bar, there are two tabs: `Metrics` (selected) and `Logs`. The main pane displays a table of metrics with the following columns: `CloudWatch Metrics Insights usage`, and `CloudWatch Metrics Insights usage`. The data shows a single row with values: 1, 1, 1, 1, 1, and 1 respectively. At the bottom of the table, there are buttons for `Next` and `Previous`.

## 2. Ansible Installation

- Install Ansible on the control node (e.g., Jenkins server or a dedicated automation server).
- Verify the installation with commands like ansible --version and configure necessary dependencies such as Python and SSH.
- Set up inventory files defining the target hosts and configure SSH access for passwordless login.



The screenshot shows a CloudShell terminal window with multiple tabs at the top. The main terminal area displays the following command-line session:

```
55 kernel-5.10=latest      enabled      [ =stable ]
56 redis6                  available    [ =stable ]
59 tpostgresql13           available    [ =stable ]
60 mock2                   available    [ =stable ]
61 dnsmasq2.85             available    [ =stable ]
62 kernel-5.15              available    [ =stable ]
63 tpostgresql14           available    [ =stable ]
64 firefox                 available    [ =stable ]
65 lustre                  available    [ =stable ]
67 awscilil                available    [ =stable ]
68 tphp8.2                 available    [ =stable ]
69 dnsmasq                 available    [ =stable ]
70 unbound1.17              available    [ =stable ]
72 collectd-python3         available    [ =stable ]
* Extra topic has reached end of support.
† Note on end-of-support. Use 'info' subcommand.
[root@ansibleserver ~]# python --version
Python 2.7.18
[root@ansibleserver ~]# ansible --version
ansible 2.9.23
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.18 (default, Mar 13 2025, 19:20:03) [GCC 7.3.1 20180712 (Red Hat 7.3.1-17)]
[root@ansibleserver ~]#
```

Below the terminal, the CloudShell interface shows the instance details:

i-058301793cb90ca17 (Ansible\_server)  
Public IPs: 34.203.227.83 Private IPs: 172.31.94.34

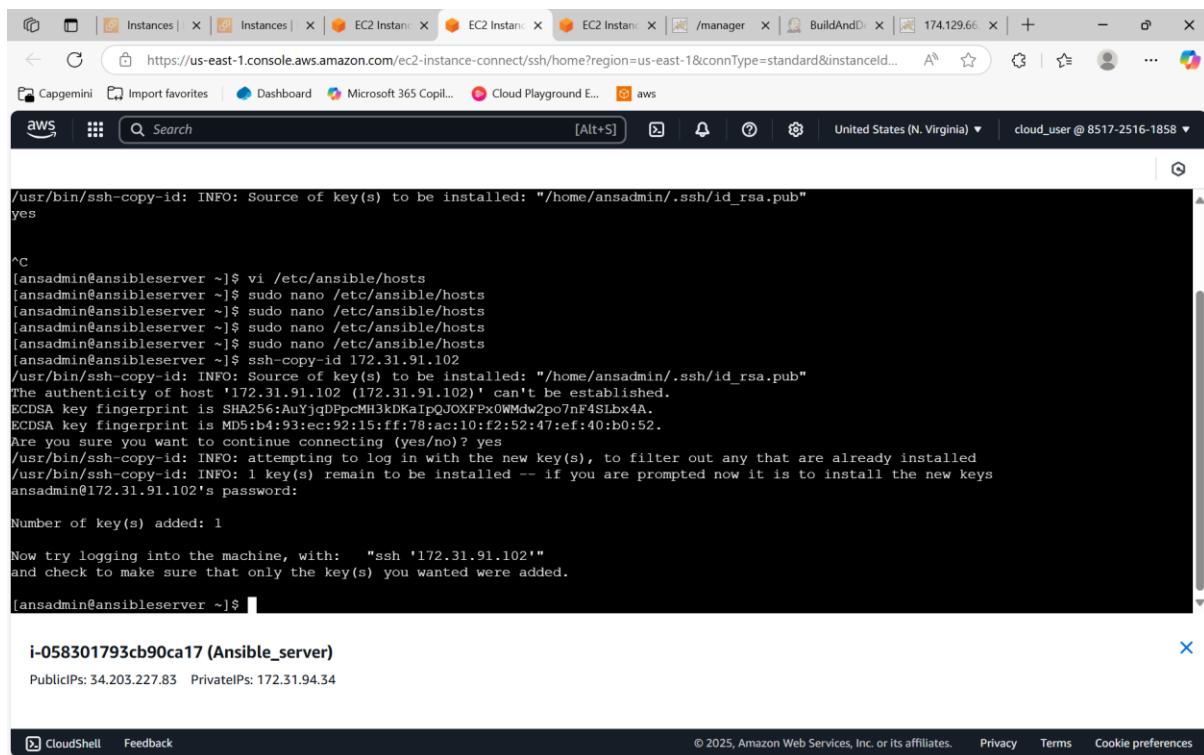
At the bottom, there are links for CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

## 3. Integrate Docker with Ansible

- Use Ansible Docker modules to manage Docker images, containers, and networks programmatically.
- Create playbooks that automate Docker tasks such as building images, starting containers, and cleaning up old containers.
- Enable Ansible to interact with Docker daemons on remote or local hosts.

## 4. Integrate Ansible with Jenkins

- Configure Jenkins to run Ansible playbooks as build or post-build steps using plugins like "Ansible plugin".
- Pass variables and parameters from Jenkins to Ansible playbooks for flexible deployments.
- Use Jenkins to trigger Ansible-driven deployments as part of the CI/CD pipeline automatically.



```
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ansadmin/.ssh/id_rsa.pub"
yes

^C
[ansadmin@ansibleserver ~]$ vi /etc/ansible/hosts
[ansadmin@ansibleserver ~]$ sudo nano /etc/ansible/hosts
[ansadmin@ansibleserver ~]$ ssh-copy-id 172.31.91.102
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ansadmin/.ssh/id_rsa.pub"
The authenticity of host '172.31.91.102 (172.31.91.102)' can't be established.
ECDSA key fingerprint is SHA256:AuYjqbPpcMH3kRaIpqJOXFpx0WMdwpo/nF4SLbx4A.
ECDSA key fingerprint is MD5:b4:93:ec:92:15:ff:78:a:c:10:f:2:52:47:e:f:40:b0:52.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
ansadmin@172.31.91.102's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh '172.31.91.102'"
and check to make sure that only the key(s) you wanted were added.

[ansadmin@ansibleserver ~]$
```

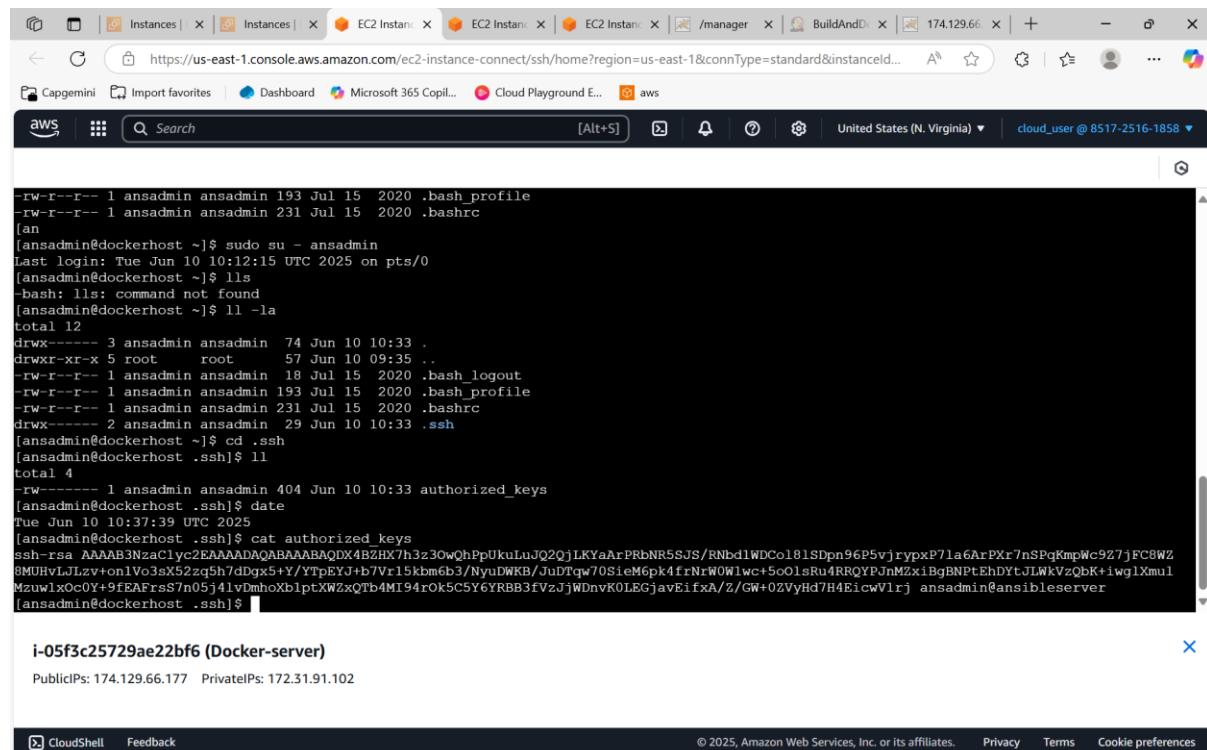
i-058301793cb90ca17 (Ansible\_server)  
PublicIPs: 34.203.227.83 PrivateIPs: 172.31.94.34

## 5. Build an Image and Create Container on Ansible

- Write Ansible playbooks that build Docker images from Dockerfiles on specified hosts.
- Automate the creation and startup of containers based on the newly built images.
- Manage container lifecycle through Ansible tasks, including start, stop, and removal.

## 6. Ansible Playbook to Create Image and Container

- Develop modular playbooks with clear tasks to build Docker images and deploy containers.
- Include steps to pull code, copy artifacts, and configure container environment variables.
- Use handlers and conditional tasks for efficient automation and error handling.



The screenshot shows a browser-based AWS CloudShell interface. The terminal window displays the following command-line session:

```
-rw-r--r-- 1 ansadmin ansadmin 193 Jul 15 2020 .bash_profile
-rw-r--r-- 1 ansadmin ansadmin 231 Jul 15 2020 .bashrc
[an
[ansadmin@dockerhost ~]$ sudo su - ansadmin
Last login: Tue Jun 10 10:12:15 UTC 2025 on pts/0
[ansadmin@dockerhost ~]$ ll
-bash: ll: command not found
[ansadmin@dockerhost ~]$ ll -la
total 12
drwx----- 3 ansadmin ansadmin 74 Jun 10 10:33 .
drwxr-xr-x 5 root root 57 Jun 10 09:35 ..
-rw-r--r-- 1 ansadmin ansadmin 18 Jul 15 2020 .bash_logout
-rw-r--r-- 1 ansadmin ansadmin 193 Jul 15 2020 .bash_profile
-rw-r--r-- 1 ansadmin ansadmin 231 Jul 15 2020 .bashrc
drwx----- 2 ansadmin ansadmin 29 Jun 10 10:33 .ssh
[ansadmin@dockerhost ~]$ cd .ssh
[ansadmin@dockerhost .ssh]$ ll
total 4
-rw----- 1 ansadmin ansadmin 404 Jun 10 10:33 authorized_keys
[ansadmin@dockerhost .ssh]$ date
Tue Jun 10 10:37:39 UTC 2025
[ansadmin@dockerhost .ssh]$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQDX4BZXH7h3z3OwQhPpUkLuJQ2QjLKYArPRbNR5SJS/RNbdlWDCo181SDpn96P5vjrypxP7la6ArPXr7nSPqKmpWc9Z7jFC8WZ
8MUHvLJLzv+on1Vo3sX52zq5h7dgx5tY/YTp6YJ+b7Vr15kbm6b3/Nyu0WRB/JuDtqw70SieM6pk4frNrWOWIwc+5cOlsRu4RRQYJPJnMxiBgBNPtEhDYtJLWkVzQbK+iwglXmul
Mzu1x0cOY+9fEAFrssTn05j41vbmboXb1ptXWzxQTb4MI94rOk5C5Y6YRBB3fVzJjWDnvK0LEGjavEifxA/Z/GW+0ZVvHd7H4EicwVlrj ansadmin@ansibleserver
[ansadmin@dockerhost .ssh]$
```

Below the terminal, a summary of the Docker server is shown:

i-05f3c25729ae22bf6 (Docker-server)  
PublicIPs: 174.129.66.177 PrivateIPs: 172.31.91.102

## 7. Copy Image onto Docker Hub

- Automate pushing Docker images to Docker Hub or other container registries using Ansible tasks.
- Manage authentication securely within playbooks or Jenkins credentials.
- Facilitate image distribution and version control across different environments.

## 8. Jenkins Job to Build an Image onto Ansible

- Configure Jenkins pipelines to invoke Ansible playbooks that build Docker images.
- Use Jenkins environment variables to control playbook behavior and image tags.
- Integrate image build step with source code changes to maintain continuous integration.

## 9. How to Create Container on Docker Host Using Ansible Playbook – DevOps Project

- Use Ansible playbooks to launch containers on remote Docker hosts, ensuring environment consistency.
- Configure network, volumes, and ports in playbooks to match application requirements.
- Validate container status and logs through automated post-deployment checks.

```
7ytCxB02rxIn8QP2fxlv1vtGvch3ex+BInMFda4wIDAQABoIBACKh7rEqqDWAcJzs
X0lnrHsD57Mo5VESD/nkwrn2ug75uotUn8xgMmKEQj114ymmnssjOR07x3FK701C9
c18GdDtcRXh151s13MH3o1R3p1tu1BhTzG/Bt/ozeKj3trT5EZ-nvHfTvDGw1j
hw5D1LwWPTi0vZFd6ougi80u0d010mfl.r3PVaum7eiQzKT1.L1bYz8DRXBclrdEN
WDg+hScHAAHPE//nA3T3lGPmgLnqhy8sN7FBmwtfEsGys7AcIhdDjzxNhDA/Yv13'N
CagT6WDvEjLPArbrAd+s02uqh2vaqj1JbuT0i1+6iiRoMfeFbvbi01glqYggEcE+
wxyLPvEcgYEAE/2F9/U16eebeo3WPx1Hb93C2zppJ218x+TXfjAbun+Land17d
n0x8zE3jogheVeKmu9g8sAtafyDmk1w5RxHGs1CBxuYo1rnCL8kdfsAu
mzjks9gBemtpkn3khbp8C7zn4PG/6rPpaLptdAnHDEF2aqyQgrKQWKBUcgYE6i3x
GpYD+QCXEXbcoENM2nJttNjg250hHP7NTMRb36Luapa0aDVE4jk0QC53DRG4bkXbDT
0mwBhT1L1Gx7SN35wQ14R7cpPjbYGg2/NTT2TREUkPTGFRg9PHCTAXYPERH/OxsP
vWBWrYz1IKO/qf1ICPNUKTSVsSh0pcqhK5Qj/RccgYBkh2lnPY0KubDUfIt0aA7M
GONNbpu0w204dnqdQ05wH1A3jzsrsGhz2ppbbkTKBVm6CbVznwQOzL05fEpMm2s
btQUAPlxny2f+xbjwrysYp/7RMN31k1C2xFkkbu706TA/lcv17QS1UfCPx/1U
Ej0CKYs5jK8yz07Luyl8QKbgB22JIAGB0CKSibFxtYvPAukd41ldmNmVsAq05
w5pVNsgnQLV16oeVunSY2L8QU/6dpedAnhfm0YhWWb+NuA7Vr8ARmhbg9qF9t3RO
og5Pd1e1qVPN6NTLDS4/E/Jlw1j+cxlCmt3ies5gsv1X/fcc88DgmLPj2SDr25UG
3RmAoGATAnbuTHB GhdzL1+2feC98vBGSFNWBqOioXnfPPEQTSY4CcWjM1RahhdWc
brskXp2fV8iIIUGDkv8Xwa3bKosAs4tDkWBxtv7KFu07g+E+fvcwZ0mPG9kVXRt
JEEmcggnmJ2EXT0zVGHHO7UN8Kz2RDSGKWLj2hJ5uQKc/Emj8=
-----END RSA PRIVATE KEY-----
[ansadmin@ansibleserver ~]$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAQABAAQABQDX4B2HX7h3z30wQhpPukluLuJQ2QjLKYaArPRbNR5SJS/RnbdiWDCol81SDpn96P5vJrypxP7la6ArPxr7nSpqKmpWc9Z7jFC8WZ
8MUHvLJLzv+on1Vo3sX52zq5h7d9x5t+YTpEYJ+b7Vrl5kbm6b3/NyuDWKB/JuD7qw805ieM6pk4frNrWOWlwC5oOlsRu4RRQYJmMzXiBgNPtEhDytJlkVzQbK+iwg1Xmul
Mzuwi0c0Y+9fEAFrS7/n05j41vDmhoXb1ptXWzXQtB4M194r0k5C5Y6YRBB3fVzJjWDnvK0LEGjavEifxa/Z/GW+0ZVvHd7H4EicwVlrj ansadmin@ansibleserver
[ansadmin@ansibleserver ~]$
```

i-058301793cb90ca17 (Ansible\_server)  
PublicIPs: 34.203.227.83 PrivateIPs: 172.31.94.34

## 10. Continuous Deployment of Docker Container Using Ansible Playbook

- Set up Ansible playbooks to handle rolling updates, zero-downtime deployments, or blue-green deployment strategies.
- Automate rollback processes in case of failures using Ansible's idempotency features.
- Incorporate monitoring and alerting hooks within the playbooks.
- Ansible executions.

```
[ansadmin@ansibleserver ~]$ cd ..
[ansadmin@ansibleserver home]$ ansible all -m ping
[WARNING]: Platform linux on host 172.31.91.102 is using the discovered Python
interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/referen
ce_appendices/interpreter_discovery.html for more information.
172.31.91.102 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
[ansadmin@ansibleserver home]$ ansible [REDACTED]
```

The screenshot shows a browser-based AWS CloudShell interface. The terminal window displays the following command-line session:

```
Last login: Tue Jun 10 10:12:15 UTC 2025 on pts/0
[ansadmin@dockerhost ~]$ ll
-bash: ll: command not found
[ansadmin@dockerhost ~]$ ll -la
total 12
drwx----- 3 ansadmin ansadmin 74 Jun 10 10:33 .
drwxr-xr-x 5 root      root   57 Jun 10 09:35 ..
-rw-r--r-- 1 ansadmin ansadmin 18 Jul 15 2020 .bash_logout
-rw-r--r-- 1 ansadmin ansadmin 193 Jul 15 2020 .bash_profile
-rw-r--r-- 1 ansadmin ansadmin 231 Jul 15 2020 .bashrc
drwx----- 2 ansadmin ansadmin 29 Jun 10 10:33 .ssh
[ansadmin@dockerhost ~]$ cd .ssh
[ansadmin@dockerhost .ssh]$ ll
total 4
-rw----- 1 ansadmin ansadmin 404 Jun 10 10:33 authorized_keys
[ansadmin@dockerhost .ssh]$ date
Tue Jun 10 10:37:39 UTC 2025
[ansadmin@dockerhost .ssh]$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAQABQDX4BZHX7h3z30wQpUkuLuJQ2QjLKYaArPRbNR5SJS/RNbdiWDCo181SDpn96P5vjrypxP7la6ArPXr7nSPqKmpWc9Z7jFC8WZ
8MUHvLJLzv+on1Vo3sX52zq5h7dDgX5+Y/YTpEYJ+b7Vr15kbm6b3/NyuDWKB/JuDtqw70SieM6pk4frNrW0Wlwc+5oOlRu4RRQYYPJnMZxiBgBNPtEhDytJLWkVzQbK+iwg1Xmul
Mzw1xOcOY+9tEAFrS7n05j41vbmhoXb1ptXWZxQTb4MI94rok5C5Y6YRBB3fVzJjWDnvnR0LEGjavEifxa/A/GW+02VvHd7H4EicwVirj ansadmin@ansibleserver
[ansadmin@dockerhost .ssh]$
[ansadmin@dockerhost ~]$ uptime
10:46:13 up 5:25, 1 user, load average: 0.01, 0.02, 0.00
[ansadmin@dockerhost ~]$
```

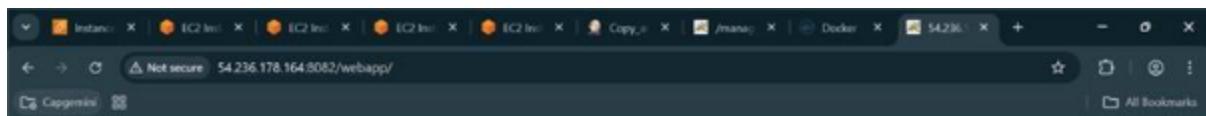
Below the terminal, the CloudShell summary shows:

**i-05f3c25729ae22bf6 (Docker-server)**  
 PublicIPs: 174.129.66.177 PrivateIPs: 172.31.91.102

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

## 11. Jenkins CI/CD to Deploy on Container Using Ansible

- Combine Jenkins and Ansible for a fully automated pipeline where code commits trigger builds and deployments.
- Use Jenkins to orchestrate the entire process from building code, creating Docker images, and deploying containers via Ansible.
- Ensure pipeline reliability and auditability with logs from both Jenkins and Ansible executions.



## New user Register for DevOps Learning

Please fill in this form to create an account.

Enter Name	<input type="text"/>
Enter mobile	<input type="text"/>
Enter Email	<input type="text"/>
Password	<input type="password"/>
Repeat Password	<input type="password"/>

By creating an account you agree to our [Terms & Privacy](#).

[Register](#)

Already have an account? [Sign in](#).

**Thankyou, Happy Learning**



## 4.Kubernetes on AWS

### 1. Why Use Kubernetes?

Kubernetes is an open-source platform that automates the deployment, scaling, and management of containerized applications. It helps ensure applications are highly available, scalable, and use cloud resources efficiently.

Running Kubernetes on AWS combines the flexibility of Kubernetes with the power of AWS infrastructure. This setup makes it easier to manage large, reliable applications in the cloud.

### 2. Ways to Install Kubernetes

You can install Kubernetes in different ways, such as:

- **kubeadm** – Manual setup for more control
- **minikube** – Best for local testing
- **kops** – Used for setting up Kubernetes on AWS

- **Amazon EKS (Elastic Kubernetes Service)** – A **managed service**, recommended for production environments on AWS because it automates much of the setup and maintenance

### 3. Installing Kubernetes with Amazon EKS

To set up an EKS cluster:

- Use the AWS Management Console or CLI to create a cluster
- Set up networking (VPC, subnets) and IAM roles
- Launch EC2 worker nodes to run workloads
- Use **kubectl** to manage the cluster by updating the local kubeconfig file

```

root@ip-172-31-28-76 ~]#
[root@ip-172-31-28-76 ~]# curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.21.2/2021-07-05/bin/linux/amd64/kubectl
% Total    % Received: % Xferd  Average Speed   Time   Time  Current
          Dload  Upload   Total Spent   Left  Speed
100 44.2M  100 44.2M    0   16.7M    0  0:00:02  0:00:02  --:--:-- 16.7M
[root@ip-172-31-28-76 ~]# ll
total 89340
drwxr-xr-x 3 root root    78 Nov 11 23:13 aws
-rw-r--r-- 1 root root 45078827 Nov 12 18:52 awscli2.zip
-rw-r--r-- 1 root root 46403584 Nov 12 14:01 kubectl
[root@ip-172-31-28-76 ~]# chmod +x kubectl
[root@ip-172-31-28-76 ~]# mv kubectl /usr/local/bin
[root@ip-172-31-28-76 ~]# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin
[root@ip-172-31-28-76 ~]# /usr/local/bin/eksctl version
Client Version: version.Info{Major:"1", Minor:"21+", GitVersion:"v1.21.2-13+d2965f0db1071203c6f5bc662c2827c71fc0b20...", GitTreeState:"clean", BuildDate:"2021-06-26T01:02:11Z", GoVersion:"go1.16.5", Compiler:"gc", Platform:"linux/amd64"}
The connection to the server localhost:8080 was refused - did you specify the right host or port?
[root@ip-172-31-28-76 ~]# curl -sS https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz | tar xz -C /tmp
[root@ip-172-31-28-76 tmp]# cd /tmp
[root@ip-172-31-28-76 tmp]# ll
total 78268
-rwxr-xr-x 1 root root 80146432 Nov 10 16:05 eksctl
drwxr-xr-x 3 root root   17 Nov 12 13:46 systemd-private-861d73d301a54dfbbbe16fa9674d9eb4-chronyd.service-M0zbFk
[root@ip-172-31-28-76 tmp]# mv eksctl /usr/local/bin
[root@ip-172-31-28-76 tmp]# 

```

### 4. Set Up a Bootstrap Server

To use eksctl (a CLI tool for EKS), you need a **bootstrap server**—this could be your local machine or an EC2 instance.

Install the following on it:

- eksctl
- AWS CLI
- kubectl

## 5. Set Up Kubernetes Using eksctl

Using eksctl simplifies creating a cluster. It automatically handles:

- Creating node groups
- Setting up networking
- Connecting all components

The screenshot shows a terminal window titled 'Terminal Sessions' with a single session. The session content is as follows:

```
root@ip-172-31-28-76 ~# curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.21.2/2021-07-05/bin/linux/amd64/kubectl
% Total % Received % Xferd Average Speed Time Time Current
          0  0  0  0  0  0  0:00:02  0:00:02 --:-- 16.7M
Lroot@ip-172-31-28-76 ~]# ll
total 89348
drwxr-xr-x 3 root root    78 Nov 11 23:13 aws
-rw-r--r-- 1 root root 45078821 Nov 12 12:52 awscliv2.zip
-rw-r--r-- 1 root root 46403584 Nov 12 14:01 kubectl
[root@ip-172-31-28-76 ~]# chmod +x kubectl
[root@ip-172-31-28-76 ~]# mv kubectl /usr/local/bin
[root@ip-172-31-28-76 ~]# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin
[root@ip-172-31-28-76 ~]# kubectl version
Client Version: version.Info{Major:"1", Minor:"21+", GitVersion:"v1.21.2-13+dd2965f0db1071203c6f5be662c2827c71fc0b20
", GitTreeState:"clean", BuildDate:"2021-06-26T01:02:11Z", GoVersion:"go1.18.5", Compiler:"gc", Platform:"linux/amd64"}
The connection to the server localhost:8000 was refused - did you specify the right host or port?
[root@ip-172-31-28-76 ~]# curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
[root@ip-172-31-28-76 tmp]# ll
total 78268
-rwxr-xr-x 1 root root 80146432 Nov 10 16:05 eksctl
drwx----- 3 root root      17 Nov 12 13:46 systemd-private-861d736301a34dfbbee16ta9674d9eb4-chronyd.service-Mzbfx
[root@ip-172-31-28-76 tmp]# eksctl /usr/local/bin
[root@ip-172-31-28-76 tmp]#
```

## 6. Run Basic Kubernetes Commands

Interact with your cluster using kubectl:

- kubectl get nodes – Check running nodes
- kubectl get pods – View application pods
- kubectl describe <resource> – Get details about any resource

```
[root@ip-172-31-28-76 tmp]# eksctl create cluster --name valaxy \
> --region us-east-1 \
> --node-type t2.small
```

## • 7. Create Your First Manifest File

- Write a **YAML** file that defines your application (Pod, Deployment, etc.).  
For example, create a Pod with a container image and resource limits.

```
2021-11-12 14:29:49 [ ] building managed nodegroup stack "eksctl-valaxy-nodegroup-ng-daf4191c"
2021-11-12 14:29:49 [ ] deploying stack "eksctl-valaxy-nodegroup-ng-daf4191c"
2021-11-12 14:29:49 [ ] waiting for CloudFormation stack "eksctl-valaxy-nodegroup-ng-daf4191c"
2021-11-12 14:30:05 [ ] waiting for CloudFormation stack "eksctl-valaxy-nodegroup-ng-daf4191c"
2021-11-12 14:30:23 [ ] waiting for CloudFormation stack "eksctl-valaxy-nodegroup-ng-daf4191c"
2021-11-12 14:30:42 [ ] waiting for CloudFormation stack "eksctl-valaxy-nodegroup-ng-daf4191c"
2021-11-12 14:30:59 [ ] waiting for CloudFormation stack "eksctl-valaxy-nodegroup-ng-daf4191c"
2021-11-12 14:31:19 [ ] waiting for CloudFormation stack "eksctl-valaxy-nodegroup-ng-daf4191c"
2021-11-12 14:31:36 [ ] waiting for CloudFormation stack "eksctl-valaxy-nodegroup-ng-daf4191c"
2021-11-12 14:31:57 [ ] waiting for CloudFormation stack "eksctl-valaxy-nodegroup-ng-daf4191c"
2021-11-12 14:32:14 [ ] waiting for CloudFormation stack "eksctl-valaxy-nodegroup-ng-daf4191c"
2021-11-12 14:32:31 [ ] waiting for CloudFormation stack "eksctl-valaxy-nodegroup-ng-daf4191c"
2021-11-12 14:32:48 [ ] waiting for CloudFormation stack "eksctl-valaxy-nodegroup-ng-daf4191c"
2021-11-12 14:33:04 [ ] waiting for CloudFormation stack "eksctl-valaxy-nodegroup-ng-daf4191c"
2021-11-12 14:33:22 [ ] waiting for CloudFormation stack "eksctl-valaxy-nodegroup-ng-daf4191c"
2021-11-12 14:33:38 [ ] waiting for CloudFormation stack "eksctl-valaxy-nodegroup-ng-daf4191c"
2021-11-12 14:33:54 [ ] waiting for CloudFormation stack "eksctl-valaxy-nodegroup-ng-daf4191c"
2021-11-12 14:34:13 [ ] waiting for CloudFormation stack "eksctl-valaxy-nodegroup-ng-daf4191c"
2021-11-12 14:34:28 [ ] waiting for CloudFormation stack "eksctl-valaxy-nodegroup-ng-daf4191c"
2021-11-12 14:34:45 [ ] waiting for CloudFormation stack "eksctl-valaxy-nodegroup-ng-daf4191c"
2021-11-12 14:34:45 [ ] waiting for the control plane availability...
2021-11-12 14:34:45 [✓] saved kubeconfig as '/root/.kube/config'
2021-11-12 14:34:45 [ ] no tasks
2021-11-12 14:34:45 [✓] All EKS cluster resources for "valaxy" have been created
2021-11-12 14:34:45 [ ] nodegroup "ng-daf4191c" has 2 node(s)
2021-11-12 14:34:45 [ ] node "ip-192-168-26-149.ec2.internal" is ready
2021-11-12 14:34:45 [ ] node "ip-192-168-37-222.ec2.internal" is ready
2021-11-12 14:34:45 [ ] waiting for at least 2 node(s) to become ready in "ng-daf4191c"
2021-11-12 14:34:45 [ ] nodegroup "ng-daf4191c" has 2 node(s)
2021-11-12 14:34:45 [ ] node "ip-192-168-26-149.ec2.internal" is ready
2021-11-12 14:34:45 [ ] node "ip-192-168-37-222.ec2.internal" is ready
2021-11-12 14:34:46 [ ] kubectl command should work with "/root/.kube/config", try 'kubectl get nodes'
2021-11-12 14:34:46 [✓] ENS cluster "valaxy" in "us-east-1" region is ready
[root@ip-172-31-28-76 tmp]#
```

## 8. Create a Service Manifest

To make your application accessible, create a **Service**. Choose a type:

- ClusterIP – Internal access
- NodePort – Exposes app on a node port
- LoadBalancer – Public access through AWS load balancer

The screenshot shows a terminal window titled 'Quick connect...' with the URL 'http://192.168.88.111:8080/xterm'. The terminal content displays a series of commands run on a host with IP 172.31.28.76, likely a K8s master node. The commands include:

```
[root@ip-172-31-28-76 ~]# kubectl get all
[root@ip-172-31-28-76 ~]# kubectl create deployment demo-nginx --image=nginx --port=88 --replicas=2
deployment.apps/demo-nginx created
[root@ip-172-31-28-76 ~]# kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
demo-nginx   2/2     2           2           17s
[root@ip-172-31-28-76 ~]# kubectl get deploy
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
demo-nginx   2/2     2           2           21s
[root@ip-172-31-28-76 ~]# kubectl get replicaset
NAME          DESIRED   CURRENT   READY   AGE
demo-nginx-848d469579  2         2         2         61s
[root@ip-172-31-28-76 ~]# kubectl get pod
NAME          READY   STATUS    RESTARTS   AGE
demo-nginx-848d469579-djzng  1/1     Running   0          84s
demo-nginx-848d469579-zkkc2  1/1     Running   0          84s
[root@ip-172-31-28-76 ~]# kubectl get all
```

## • 9. Labels and Selectors

- Use **labels** (key-value pairs) to tag resources.

**Selectors** connect Pods to Services or Deployments based on these labels, allowing better grouping and dynamic management.

The screenshot shows a terminal window titled 'Quick connect...' with the URL 'http://192.168.88.111:8080/xterm'. The terminal content displays the creation of a Kubernetes Service named 'demo-service'. The commands include:

```
[root@ip-172-31-28-76 ~]# kubectl get all
[root@ip-172-31-28-76 ~]# kubectl get all
[root@ip-172-31-28-76 ~]# ls
drwxrwsr-x 2 root kubelet 4096 May  1 10:15 awscliv2.zip
[root@ip-172-31-28-76 ~]# vi service.yml
[root@ip-172-31-28-76 ~]# cat service.yml
apiVersion: v1
kind: Service
metadata:
  name: demo-service

spec:
  ports:
    - name: nginx-port
      port: 88
      targetPort: 80
  type: LoadBalancer
[root@ip-172-31-28-76 ~]#
```

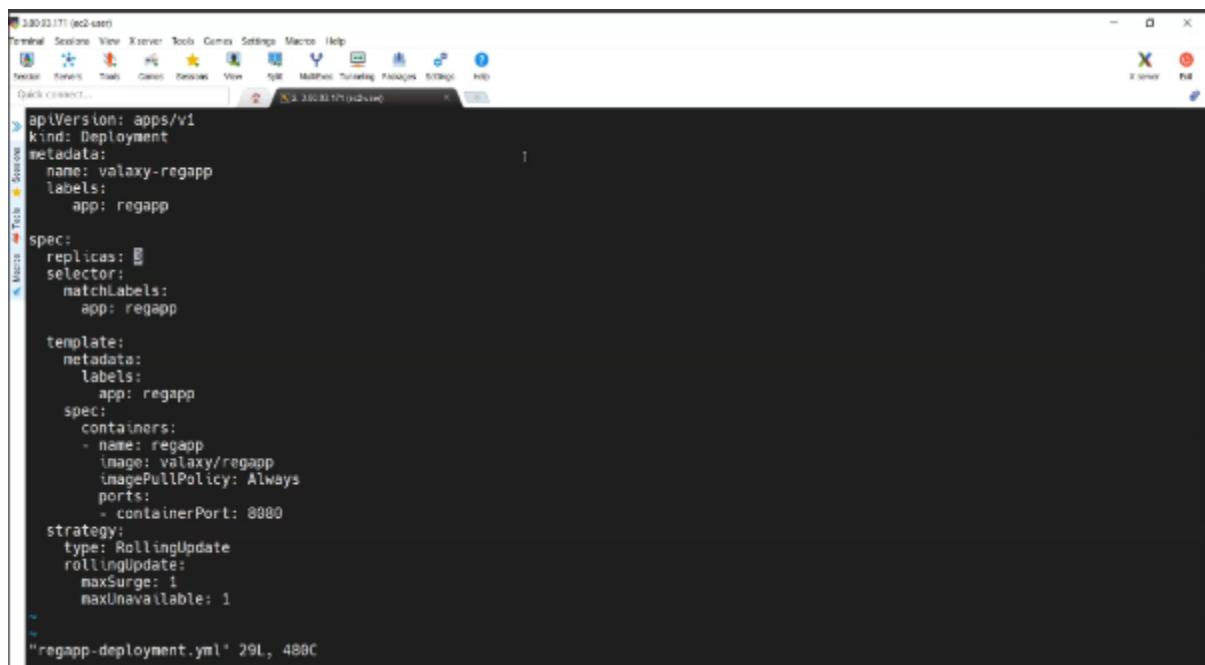
## Integrating Kubernetes with CI/CD

### 1. Write a Deployment File

Create a deployment YAML file defining:

- Container image
- Number of replicas
- Update strategy
- Resource limits

This ensures automatic scaling and reliable app updates.

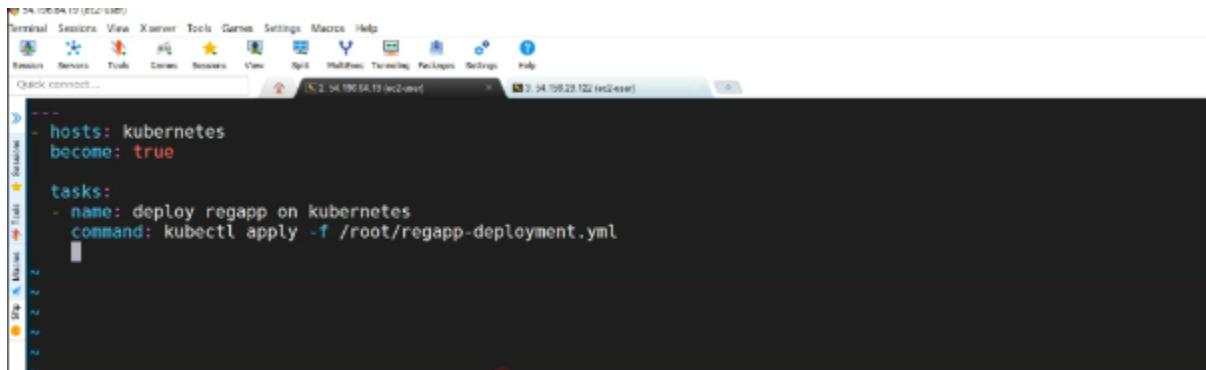


A screenshot of a terminal window titled "130.33.171 (ec2-user)". The window shows a deployment YAML file named "regapp-deployment.yaml". The file defines a deployment named "valaxy-regapp" with one replica, using the container image "valaxy/regapp". It specifies a rolling update strategy with a maxSurge of 1 and a maxUnavailable of 1. The terminal also shows the file's size as 29L and 488C.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: valaxy-regapp
  labels:
    app: regapp
spec:
  replicas: 1
  selector:
    matchLabels:
      app: regapp
  template:
    metadata:
      labels:
        app: regapp
    spec:
      containers:
        - name: regapp
          image: valaxy/regapp
          imagePullPolicy: Always
          ports:
            - containerPort: 8080
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
"regapp-deployment.yaml" 29L, 488C
```

### 2. Apply Deployment and Service

- Apply deployment and service manifests using `kubectl apply -f` commands.
- The deployment creates pods running the application, while the service exposes pods to internal or external traffic.
- Validate pod status and service accessibility through Kubernetes commands.

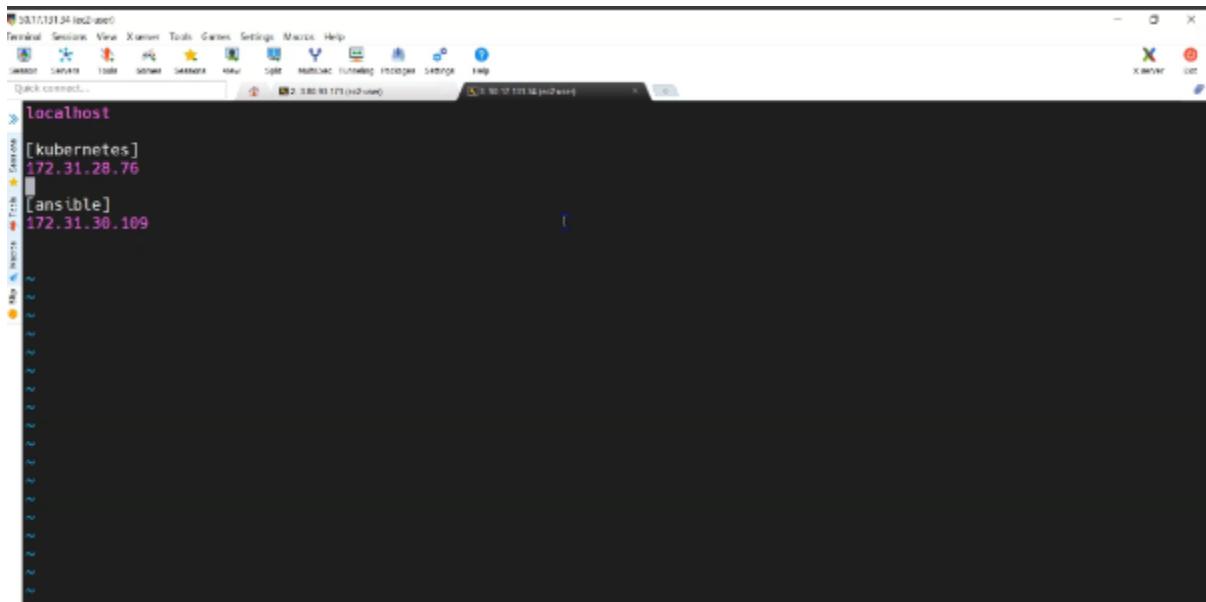


```
hosts: kubernetes
become: true

tasks:
- name: deploy regapp on kubernetes
  command: kubectl apply -f /root/regapp-deployment.yml
```

### 3. Integrate Kubernetes Bootstrap Server with Ansible

- Use Ansible to automate management of Kubernetes bootstrap server where kubectl and cluster tools are configured.
- Playbooks can handle setup tasks such as configuration, authentication, and cluster maintenance.
- Enables automated deployment workflows integrated with infrastructure provisioning.



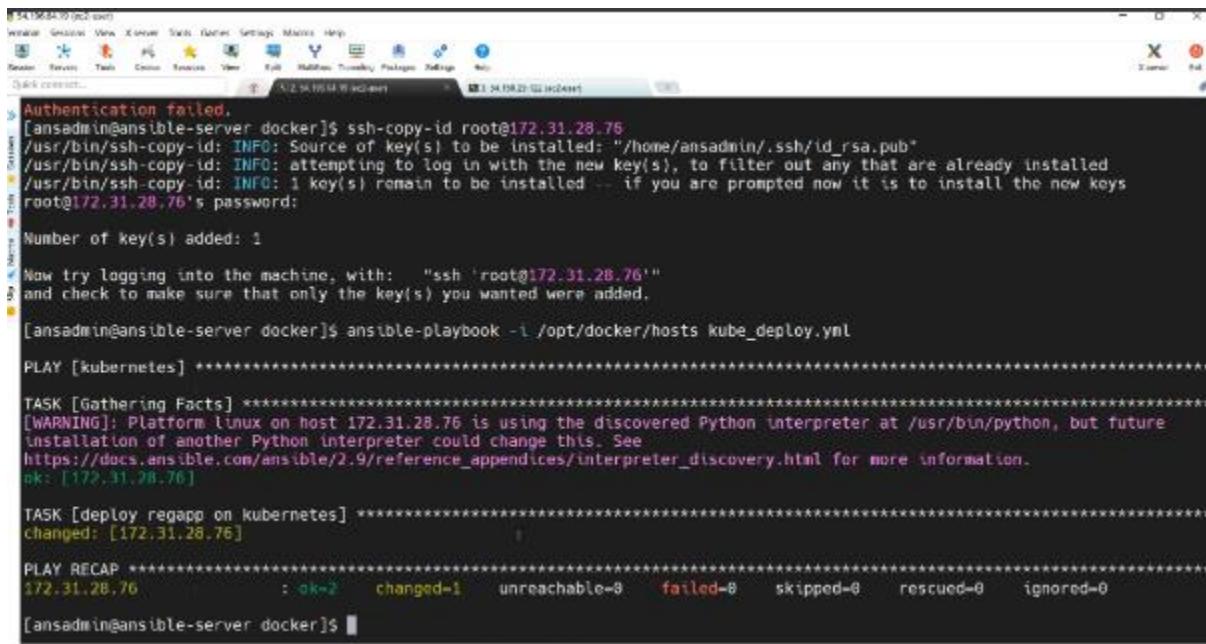
```
[kubernetes]
172.31.28.76
[ansible]
172.31.30.109
```

### 4. Use Ansible Playbooks to Apply Manifests

Create Ansible playbooks that:

- Run kubectl apply for Deployment and Service YAMLS
- Manage app updates and rollbacks

This promotes **Infrastructure as Code** practices.



A screenshot of a terminal window titled 'Terminal' showing the execution of an Ansible playbook. The terminal shows the following output:

```
[ansadmin@ansible-server docker]$ ssh-copy-id root@172.31.28.76
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ansadmin/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
root@172.31.28.76's password:
Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'root@172.31.28.76'"
and check to make sure that only the key(s) you wanted were added.

[ansadmin@ansible-server docker]$ ansible-playbook -i /opt/docker/hosts kube_deploy.yml
PLAY [kubernetes] ****
TASK [Gathering Facts] ****
[WARNING]: Platform linux on host 172.31.28.76 is using the discovered Python interpreter at /usr/bin/python, but future
installation of another Python interpreter could change this. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
ok: [172.31.28.76]

TASK [deploy regapp on kubernetes] ****
changed: [172.31.28.76]

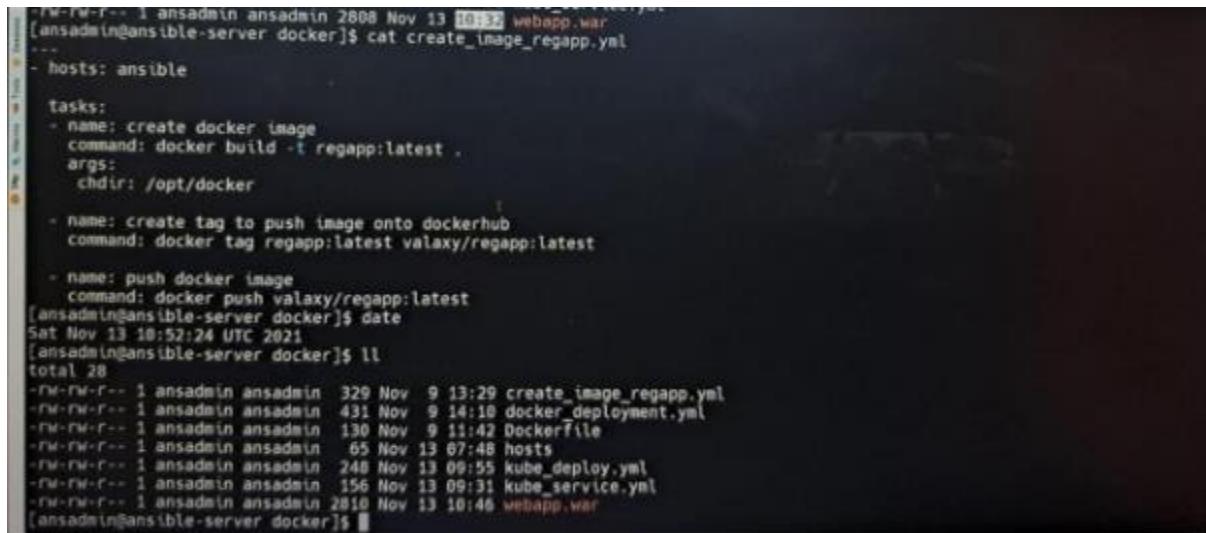
PLAY RECAP ****
172.31.28.76 : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
[ansadmin@ansible-server docker]$
```

## 5. Automate Deployment with Jenkins

Use Jenkins to:

- Trigger Ansible playbooks or kubectl commands
- Automate app build, test, and deployment
- Run these pipelines on every code commit or pull request

Monitor success or failure right from Jenkins.



A screenshot of a terminal window titled 'Terminal' showing the execution of a Jenkins pipeline. The terminal shows the following output:

```
[ansadmin@ansible-server docker]$ cat create_image_regapp.yml
---
hosts: ansible
tasks:
- name: create docker image
  command: docker build -t regapp:latest .
  args:
    chdir: /opt/docker
- name: create tag to push image onto dockerhub
  command: docker tag regapp:latest valaxy/regapp:latest
- name: push docker image
  command: docker push valaxy/regapp:latest
[ansadmin@ansible-server docker]$ date
Sat Nov 13 10:52:24 UTC 2021
[ansadmin@ansible-server docker]$ ll
total 28
-rw-rw-r-- 1 ansadmin ansadmin 329 Nov  9 13:29 create_image_regapp.yml
-rw-rw-r-- 1 ansadmin ansadmin 431 Nov  9 14:10 docker_deployment.yml
-rw-rw-r-- 1 ansadmin ansadmin 130 Nov  9 11:42 Dockerfile
-rw-rw-r-- 1 ansadmin ansadmin   65 Nov 13 07:48 hosts
-rw-rw-r-- 1 ansadmin ansadmin 248 Nov 13 09:55 kube_deploy.yml
-rw-rw-r-- 1 ansadmin ansadmin 156 Nov 13 09:31 kube_service.yml
-rw-rw-r-- 1 ansadmin ansadmin 2810 Nov 13 10:46 webapp.war
[ansadmin@ansible-server docker]$
```

## 6. Build Docker Image for Kubernetes

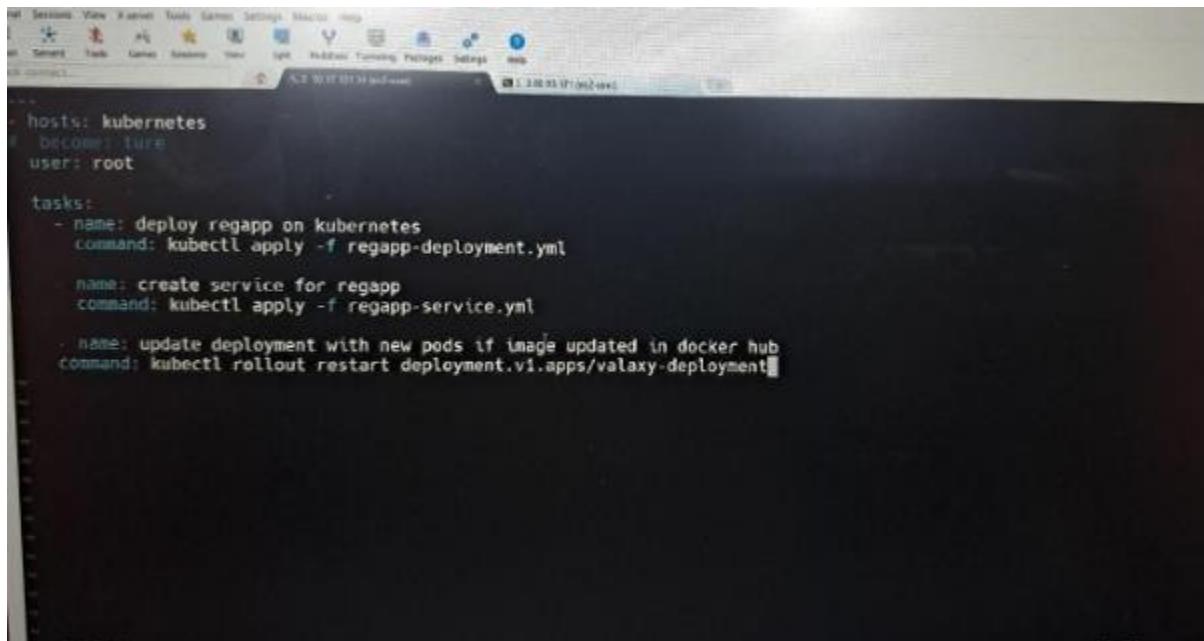
Set up a Jenkins job to:

- Build Docker images from the latest code
- Push them to a container registry (Docker Hub or AWS ECR)

This ensures your Kubernetes deployment always uses the latest version.

## 7. Enable Rolling Update to Create Pod from Latest Docker Image

- Configure deployment strategy in Kubernetes manifest to support rolling updates.
- Allows zero downtime deployments by gradually replacing old pods with new ones running updated images.
- Monitors rollout progress and automatically rolls back on failures.



```
host: kubernetes
become: true
user: root

tasks:
  - name: deploy regapp on kubernetes
    command: kubectl apply -f regapp-deployment.yml

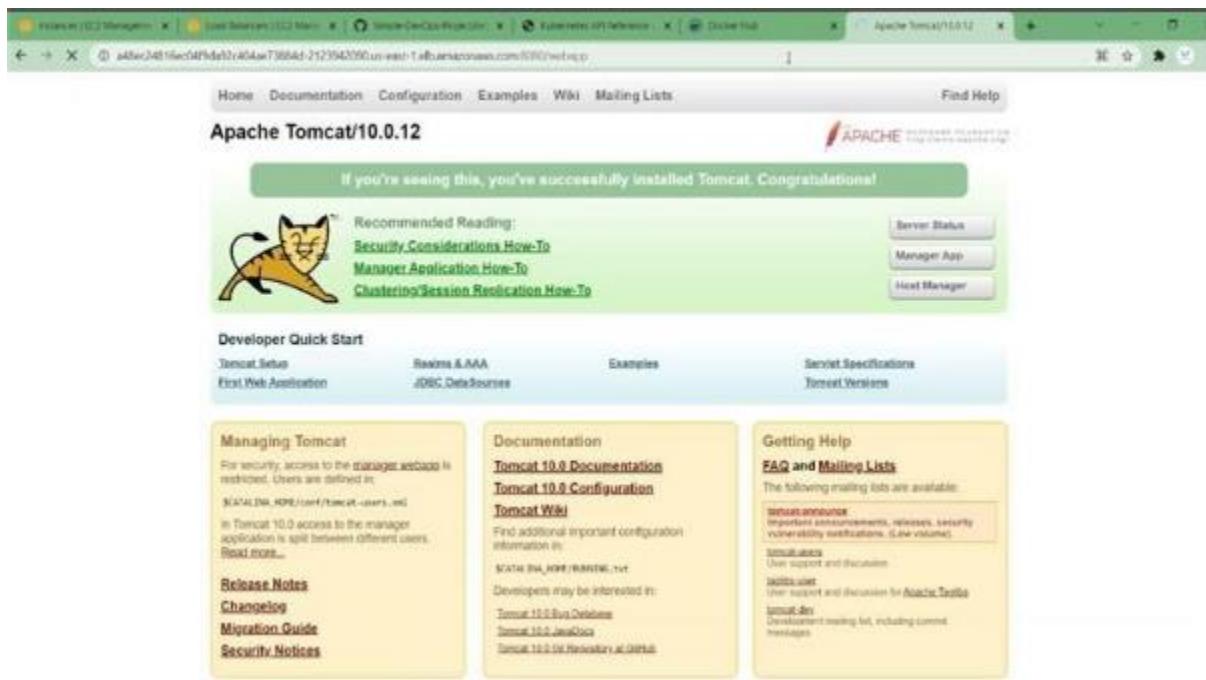
  - name: create service for regapp
    command: kubectl apply -f regapp-service.yml

  - name: update deployment with new pods if image updated in docker hub
    command: kubectl rollout restart deployment.v1.apps/vataxy-deployment
```

## 8. End-to-End CI/CD Pipeline

Combine all CI/CD steps:

- Build and test code
- Create Docker image
- Deploy to Kubernetes
- Roll out updates automatically



## 9. Clean Up Resources

Use Jenkins or Ansible to:

- Remove unused Pods, Services, and Deployments
- Tear down test clusters when no longer needed

```
[root@ip-172-31-25-112 ~]# cd tmp/
[root@ip-172-31-25-112 tmp]# ll
total 16
-rw----- 1 root root 0 Jun 13 07:29 6229lb61.eksctl.lock
-rw-r--r-- 1 root root 203 Jun 13 07:34 pod.yml
-rw-r--r-- 1 root root 481 Jun 13 07:36 regapp-deploy.yml
-rw-r--r-- 1 root root 197 Jun 13 07:36 regapp-service.yml
-rw-r--r-- 1 root root 183 Jun 13 07:34 service.yml
drwx----- 3 root root 17 Jun 13 07:08 systemd-private-72ab7dd853ea47c9be9b90aefc2d7eal-chronyd.service-pgkTY6
[root@ip-172-31-25-112 tmp]# ls
6229lb61.eksctl.lock regapp-deploy.yml    service.yml
pod.yml      regapp-service.yml  systemd-private-72ab7dd853ea47c9be9b90aefc2d7eal-chronyd.service-pgkTY6
[root@ip-172-31-25-112 tmp]# kubectl delete -f regapp-service.yml
service "valaxy-service" deleted
[root@ip-172-31-25-112 tmp]# kubectl delete -f regapp-deploy.yml
deployment.apps "valaxy-regapp" deleted
[root@ip-172-31-25-112 tmp]#
```

## **Conclusion**

This case study successfully demonstrates the implementation of a fully automated DevOps CI/CD pipeline using widely adopted tools such as Git, Jenkins, Maven, Docker, Ansible, and Kubernetes on AWS infrastructure. By integrating these technologies, the pipeline enabled continuous integration and continuous deployment, significantly improving the speed, reliability, and efficiency of software delivery with minimal manual intervention. The process began with Jenkins automating build and test tasks triggered by code commits to Git, followed by Maven handling Java project builds, and Docker containerizing the applications for consistent deployment. Ansible further streamlined configuration management, while Kubernetes—deployed via Amazon EKS—provided scalable orchestration and seamless rollout of application updates. This automation-first approach allowed early detection of bugs, reduced deployment errors, and supported rolling updates, thereby enhancing software quality and accelerating release cycles. Leveraging AWS cloud services and infrastructure-as-code practices ensured scalability, flexibility, and consistency across environments. Overall, this project highlights the tangible benefits of adopting DevOps methodologies and CI/CD pipelines in modern software development. Future improvements could include integrating advanced monitoring tools, enabling multi-cloud deployment, and adopting more sophisticated rollback and deployment strategies to further optimize the delivery process.