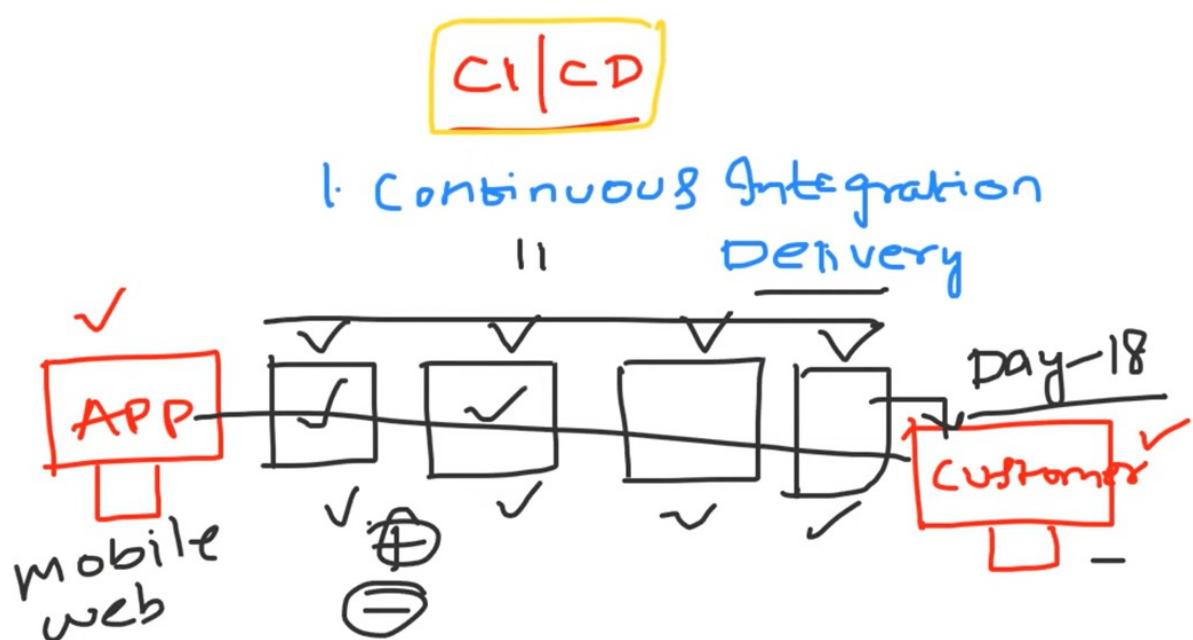


ULTIMATE CI/CD PIPELINE | JENKINS END TO END PROJECT ||



Unit testing

Static Code Analysis

Code Quality /vulnerability

Automation

Reports

Deployment

6:36. Unit testing is nothing but testing code for a block or specific functionality.

7:10. Static code analysis is to verify format, all indentation is right, syntax is right, make sure u didn't declare any unnecessary variables.

Code quality/ vulnerability testing is done to check for vulnerability, that can be misused by hackers

8:51. Automation testing/functional testing /endtoend testing, in this you verify your application in end to end manner such that one functionality doesn't affect other functionality.

Reports - reports for how many units test passed , what is code quality etc.

Deployment stage- deploying your application on a platform so that your customer can access your application.

Unit testing is a critical part of the continuous integration and continuous delivery (CI/CD) pipeline because it **helps identify bugs and issues early in the development process:**

Benefits

Unit testing can:

- Improve code quality
- Reduce development time
- Increase software reliability
- Save time and money

How it works

Unit tests are automated tests that validate the functionality of individual code components, such as functions, methods, or classes. They simulate all external dependencies that are used in the code base.

Best practices

To ensure effective unit testing, you can:

- Write testable code
- Test early and often
- Keep tests small and focused
- Ensure test coverage
- Use mocking
- Monitor and analyze test results

Other types of tests that can be used in CI/CD pipelines include:

- Integration tests
- Acceptance tests
- Synthetic tests
- Performance tests
- Resilience tests
- Static application security tests (SAST)
- Dynamic application security tests (DAST)

Static code analysis is a method of analyzing source code without executing the program to identify security flaws and programming mistakes. It's also known as Static Application Security Testing (SAST) or source code analysis.

Static code analysis is usually performed early in the development process to help developers: Improve code quality, Reduce the risk of security breaches, Comply with coding guidelines and industry standards, and Identify and fix issues early.

Static code analysis tools can identify potential vulnerabilities such as: NULL pointer dereferences, Buffer overflows, Memory usage errors, Injection, and Cross-site scripting.

Static code analysis tools use techniques such as data flow analysis and taint analysis.

Static code analysis is different from dynamic code analysis, which is the process of testing and assessing a program based on execution. Static code analysis is performed before running a program, while dynamic code analysis is performed after running a program.

Dynamic Application Security Testing (DAST) is a security testing method that can be integrated into a continuous integration and continuous delivery (CI/CD) pipeline to identify vulnerabilities in applications: [🔗](#)

What is DAST?

DAST is a cybersecurity testing method that analyzes applications in real-time to identify vulnerabilities. It's also known as outside-in testing or black box testing. [🔗](#)

How does DAST work?

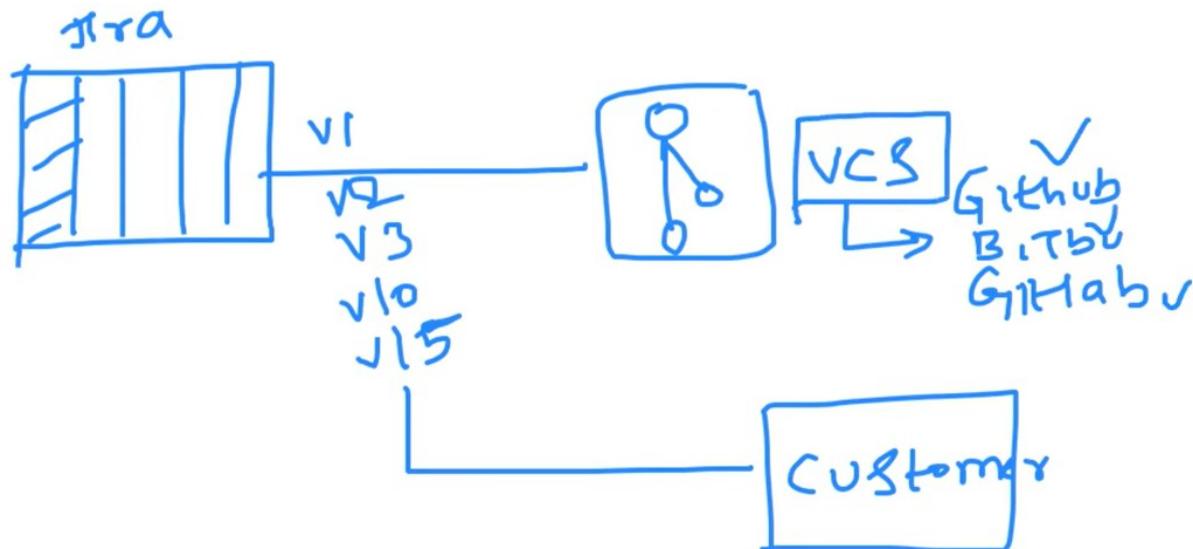
DAST simulates real-world attacks on an application to identify vulnerabilities. It does this by sending HTTP requests to the application, looking for anomalies, and reporting any vulnerabilities it finds. [🔗](#)

Why use DAST in CI/CD?

DAST is an important tool for CI/CD pipelines because it helps to identify vulnerabilities early in the development lifecycle. This allows teams to address vulnerabilities before they become more costly and time-consuming to fix. [🔗](#)

How to integrate DAST into CI/CD?

DAST scans can be automated in a CI/CD pipeline, or run independently on demand. A CI/CD pipeline is a series of automated steps that execute when changes are made to the codebase. [🔗](#)



Define Jenkins

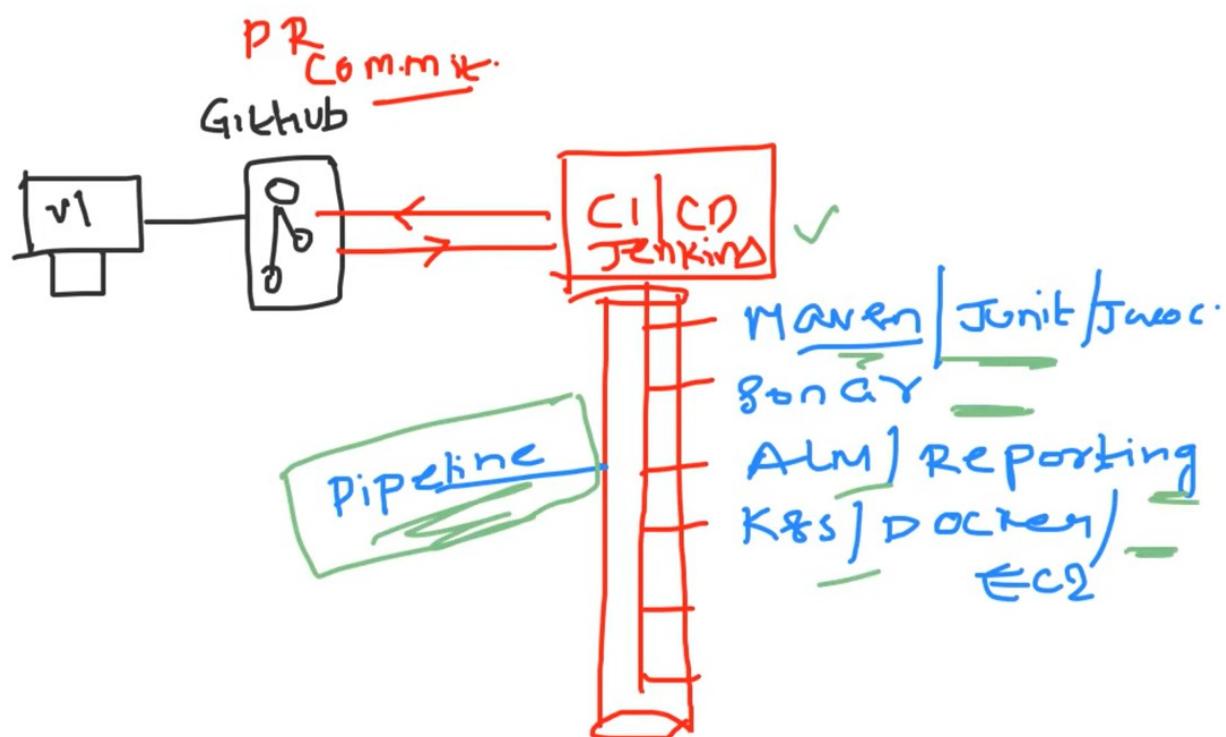
Jenkins is an open-source automation server that helps manage and control the software development process.

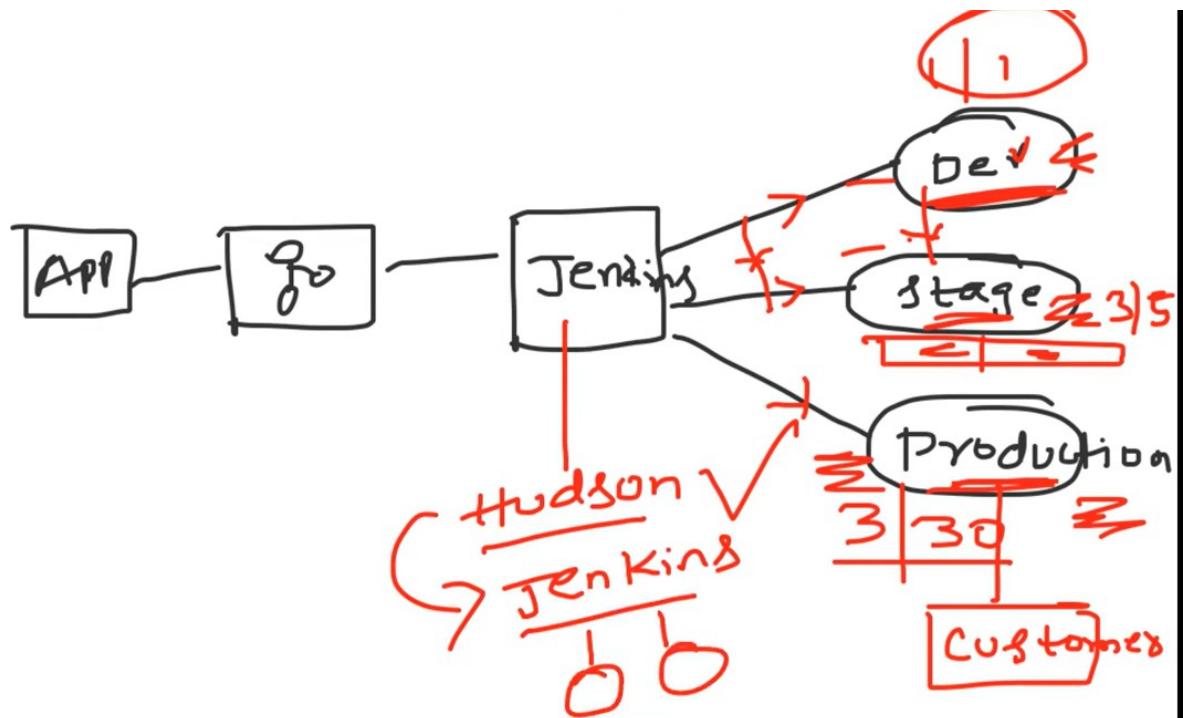
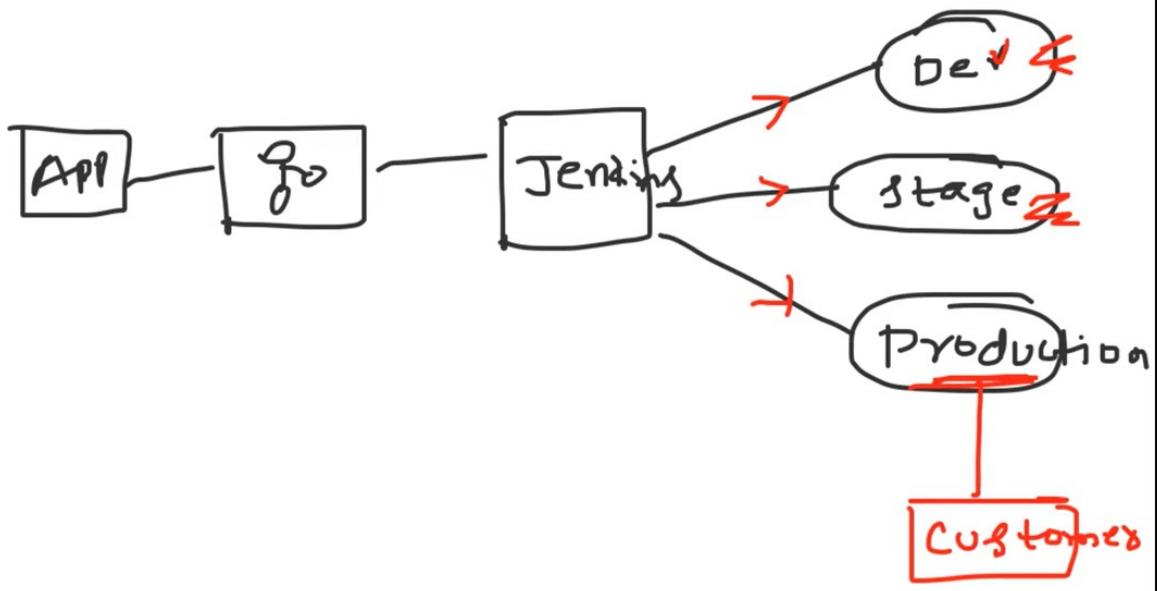
It can be used to automate tasks like building, testing, and deploying software. Jenkins is a popular DevOps tool that can be run on Windows, Linux, macOS, and other Unix-like operating systems. [🔗](#)

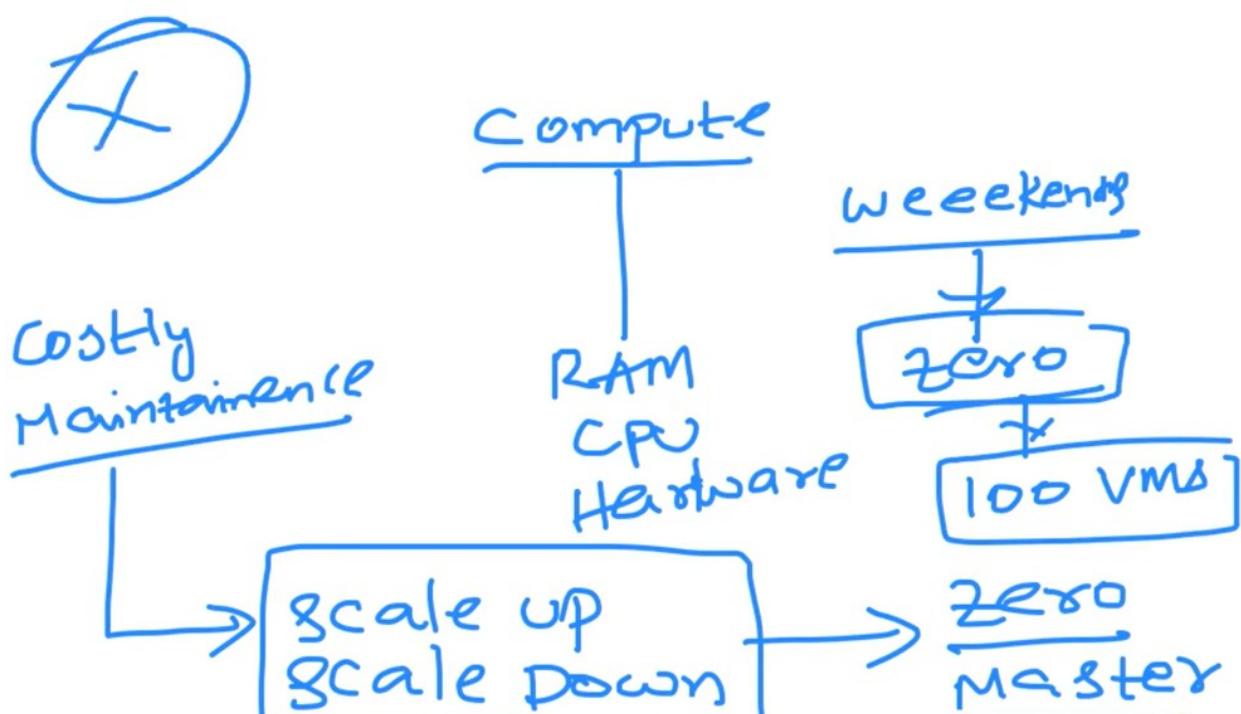
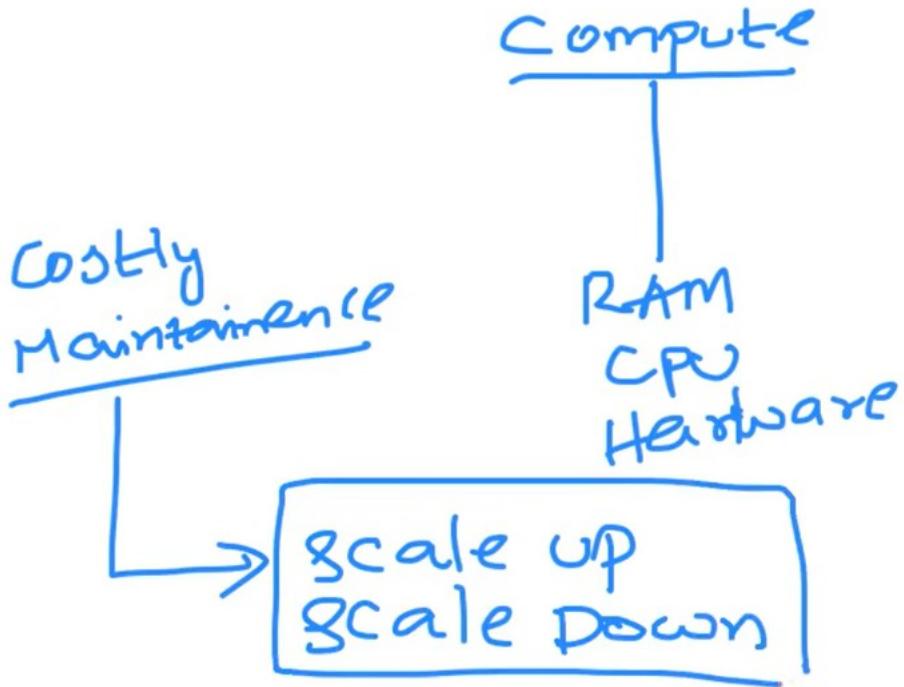


Here are some features of Jenkins:

- **Easy installation:** Jenkins can be installed through native system packages, Docker, or run standalone. [🔗](#)
- **Easy configuration:** Jenkins can be set up and configured via its web interface. [🔗](#)
- **Plugins:** Jenkins provides hundreds of plugins for building, deploying, and automating projects. [🔗](#)
- **CI/CD pipelines:** Jenkins can implement CI/CD workflows called pipelines to automate testing and reporting on code changes. [🔗](#)
- **Docker support:** Jenkins can use Docker images and containers to run inside. [🔗](#)
- **Test recording, reporting, and visualization:** Jenkins can record test failures for reporting and visualization in the web UI. [🔗](#)





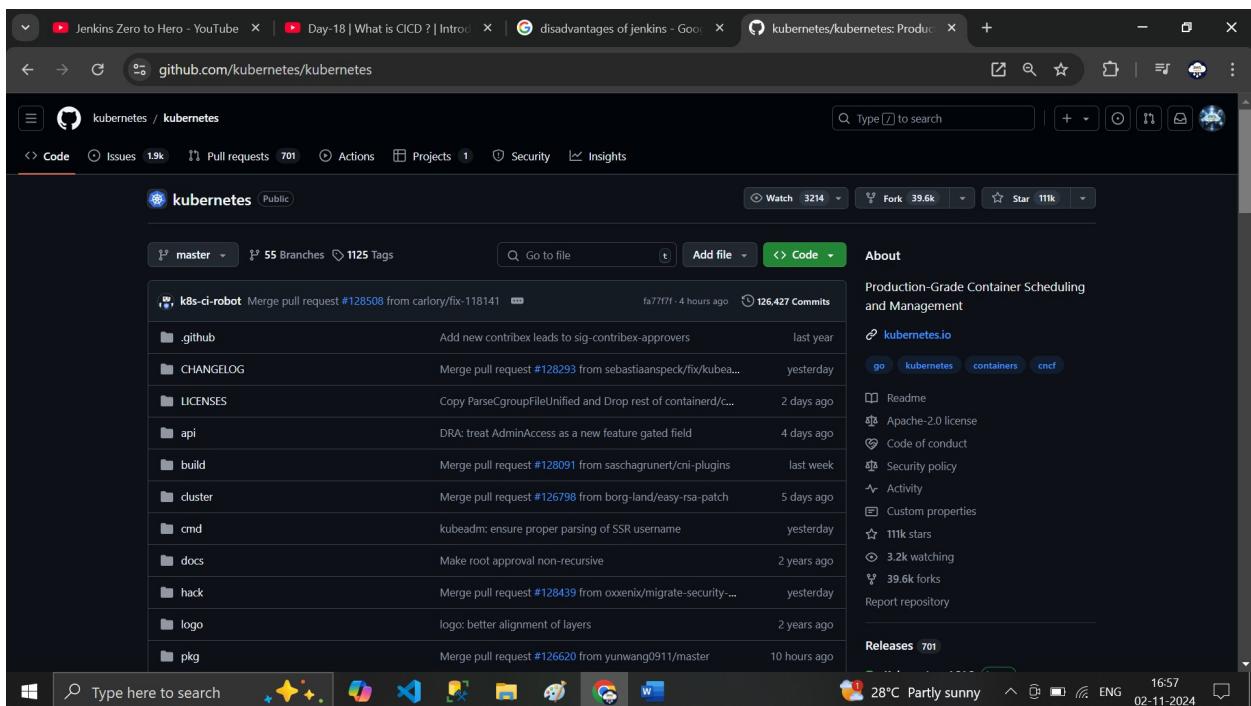


Disadvantage in Jenkins .. When I am in Weekends / holiday Server should not be Running .. Jenkins fails in this

What are the problems with Jenkins?

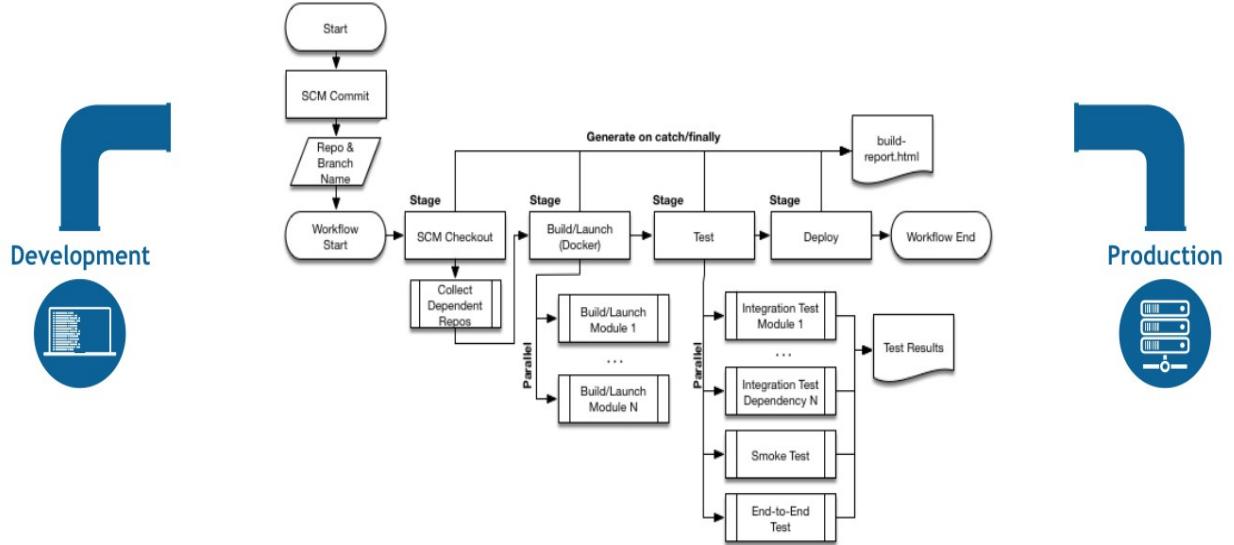
Jenkins doesn't allow server-to-server federation, which can cause performance issues in large-scale environments. Jenkins sprawl—this is a common problem which also stems from lack of federation. Multiple teams using Jenkins can create a large number of standalone Jenkins servers that are difficult to manage.

<https://github.com/kubernetes/kubernetes>



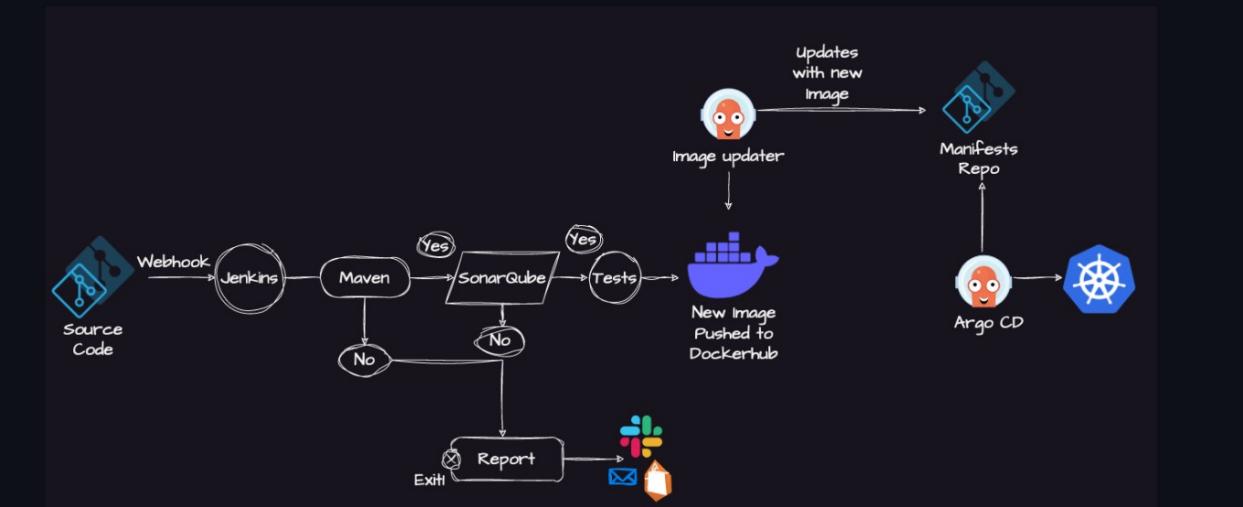
Day-19 | Jenkins ZERO to HERO | 3 Projects Live | Docker Agent | Interview

Jenkins Pipeline Project using Docker as Agent(NodeJS-APP) with AWS EC2- Deployment



1. Install Jenkins on ec2
2. Configure Jenkins and expose to outside world
3. Use Docker as Agents against the VM approach
4. Advantages of using Docker as Agents
5. GitOps approach to deploy applications on to k8s

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



AWS EC2 Instance

- Go to AWS Console
- Instances(running)
- Launch instances

The screenshot shows the AWS EC2 instance creation process. The current step is 'Name and tags'. The 'Name' field contains 'jenkins-master'. The 'Software Image (AMI)' section shows 'Ubuntu Server 22.04 LTS (HVM), SSD Volume Type' selected. The 'Virtual server type (instance type)' is set to 't2.micro'. The 'Launch instance' button is visible at the bottom right.

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

Recent [Quick Start](#)

Amazon Linux  macOS  Ubuntu  Windows  Red Hat  SI 

Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type [Free tier eligible](#)
ami-00874d747dde814fa (64-bit (x86)) / ami-01625be155ee390e9 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description
Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2023-01-15

Number of instances [Info](#)
1

Software Image (AMI)
Canonical, Ubuntu, 22.04 LTS, ...[read more](#)
ami-00874d747dde814fa

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

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

Cancel [Launch instance](#)

#install

Pre-Requisites:

- Java (JDK)

Run the below commands to install Java and Jenkins

Install Java

```
sudo apt update  
sudo apt install openjdk-17-jre
```

Verify Java is Installed

```
java -version
```

Now, you can proceed with installing Jenkins

```
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
```

****Note:** By default, Jenkins will not be accessible to the external world due to the inbound traffic restriction by AWS. Open port 8080 in the inbound traffic rules as show below.

- EC2 > Instances > Click on
- In the bottom tabs -> Click on Security
- Security groups
- Add inbound traffic rules as shown in the image (you can just allow TCP 8080 as well, in my case, I allowed All traffic).

Inbound rules (4)										
	Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description	Actions	
<input type="checkbox"/>	-	sgr-Obaedb63491a557...	IPv4	HTTP	TCP	80	0.0.0.0/0	-		
<input type="checkbox"/>	-	sgr-026c9d83e0574df39	IPv4	All traffic	All	All	0.0.0.0/0	-		
<input type="checkbox"/>	-	sgr-059d7f20d69e56b...	IPv4	HTTPS	TCP	443	0.0.0.0/0	-		

Login to Jenkins using the below URL:

<http://:8080> [You can get the ec2-instance-public-ip-address from your AWS EC2 console page]

Note: If you are not interested in allowing All Traffic to your EC2 instance 1. Delete the inbound traffic rule for your instance 2. Edit the inbound traffic rule to only allow custom TCP port 8080

After you login to Jenkins, - Run the command to copy the Jenkins Admin Password - `sudo cat /var/lib/jenkins/secrets/initialAdminPassword` - Enter the Administrator password

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

Click on Install suggested plugins

Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Wait for the Jenkins to Install suggested plugins

Getting Started

Getting Started

Folders	Formatter	Ant	Gradle	Git
Timestamper	Workspace Cleanup			GitHub
Pipeline	Github Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline: Stage View	Github Branch Source Pipeline: GitHub Groovy Libraries ** Pipeline Graph Analysis ** Pipeline: REST API Pipeline: Stage View
Git	SSH Build Agents	Matrix Authorization Strategy	PAM Authentication	Git SSH Build Agents Matrix Authorization Strategy PAM Authentication
LDAP	Email Extension	Mailer		LDAP Email Extension Mailer

Create First Admin User or Skip the step [If you want to use this Jenkins instance for future use-cases as well, better to create admin user]

Getting Started

Create First Admin User

Username

Password

Confirm password

Full name

E-mail address

Jenkins Installation is Successful. You can now starting using the Jenkins

Getting Started

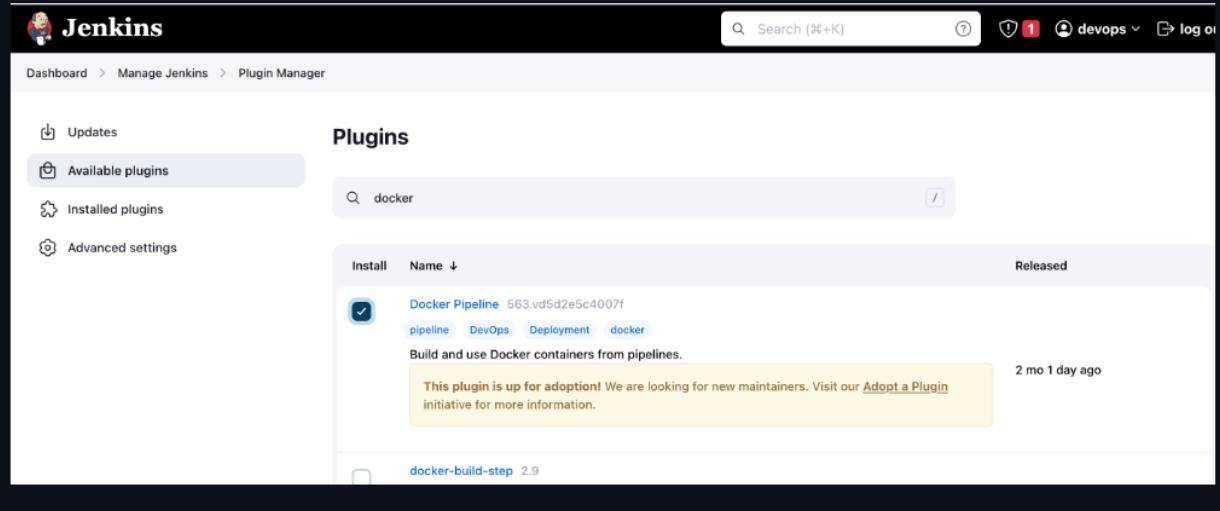
Jenkins is ready!

Your Jenkins setup is complete.

[Start using Jenkins](#)

Install the Docker Pipeline plugin in Jenkins:

- Log in to Jenkins.
- Go to Manage Jenkins > Manage Plugins.
- In the Available tab, search for "Docker Pipeline".
- Select the plugin and click the Install button.
- Restart Jenkins after the plugin is installed.



The screenshot shows the Jenkins Plugin Manager interface. The 'Available plugins' tab is selected. A search bar at the top right contains the text 'docker'. Below the search bar, a list of plugins is shown, with 'Docker Pipeline' highlighted. The 'Docker Pipeline' plugin entry includes a 'Released' timestamp of '2 mo 1 day ago', a brief description, and a note indicating it is up for adoption. Other visible plugins include 'docker-build-step'.

Wait for the Jenkins to be restarted.

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.

----- Practical part

Cp the Public IP

The screenshot shows the AWS EC2 Instances page for an instance named "Jenkins". The instance ID is i-01e71d81f62db5d35. The instance is currently running. Its public IP address is 100.26.160.84. Other details include the private IP DNS name (ip-172-31-53-221.ec2.internal), VPC ID (vpc-0895d01f56cc9b2f), and subnet ID (subnet-01c8c40aad521ac2f). The instance type is t2.micro. The AMI ID is ami-00874d747dde814fa, and the AMI name is ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20230115. The launch time was Wed Feb 01 2023 15:58:51 GMT+0530 (India Standard Time). The instance has no IAM Role or Elastic IP addresses assigned.

```
→ Jenkins-Zero-To-Hero git:(main) ssh -i /Users/aveerama/Downloads/abhishek_devops.pem ubuntu@100.26.160.84
The authenticity of host '100.26.160.84 (100.26.160.84)' can't be established.
ED25519 key fingerprint is SHA256:Vfe7tiQRhSvK3FkwM+HLS+UA8bQi7i+zzskqZaeK3Bk.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '100.26.160.84' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-1028-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 System information as of Wed Feb  1 10:39:11 UTC 2023

 System load:  0.0          Processes:      96
```

updating App packages

```
ubuntu@ip-172-31-53-221:~$ 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 [114 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [99.8 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://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [852 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [189 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [13.1 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [566 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [82.1 kB]
```

```
# install Jenkins
```

```
ubuntu@ip-172-31-53-221:~$ curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.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-stable 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
Ign:4 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:5 https://pkg.jenkins.io/debian-stable binary/ Release [2044 B]
Get:6 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
```

<https://www.jenkins.io/doc/book/installing/>

The screenshot shows a Microsoft Edge browser window with the URL <https://jenkins.io/doc/book/installing/windows/>. The page displays the Jenkins User Handbook for Windows. The left sidebar contains a navigation menu with sections like 'User Handbook' (including 'Installing Jenkins' for Windows), 'Platform Information', 'Using Jenkins', 'Pipeline', 'Blue Ocean', 'Managing Jenkins', and 'Securing Jenkins'. The main content area features a large heading 'Windows' with a sub-section 'Prerequisites' listing hardware requirements (256 MB of RAM, 1 GB of drive space). A 'Table of Contents' sidebar on the right lists various installation steps and troubleshooting topics.

The screenshot shows a web browser window with the URL jenkins.io/doc/book/installing/windows/. The page content includes:

- User Handbook Overview**
- Installing Jenkins** (selected)
 - Docker
 - Kubernetes
 - Linux
 - macOS
 - Windows
 - Other Systems
 - WAR file
 - Other Servlet Containers
 - Offline Installations
 - Initial Settings
- Platform Information**
- Using Jenkins**
- Pipeline**
- Blue Ocean**
- Managing Jenkins**
- Securing Jenkins**

Content on the right side of the page includes:

- if running Jenkins as a Docker container)
- Recommended hardware configuration for a small team:
 - 4 GB+ of RAM
 - 50 GB+ of drive space
- Comprehensive hardware recommendations:
 - Hardware: see the [Hardware Recommendations page](#)
- Software requirements:
 - Java: see the [Java Requirements page](#)
 - Web browser: see the [Web Browser Compatibility page](#)
 - For Windows operating system: [Windows Support Policy](#)
 - For Linux operating system: [Linux Support Policy](#)
 - For servlet containers: [Servlet Container Support Policy](#)

Installation steps using Windows MSI installer

How to install Jenkins on Windows

```
ubuntu@ip-172-31-53-221:~$ ps -ef | grep jenkins
jenkins    4460      1  0 10:40 ?        00:00:42 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080
ubuntu     4634     1328  0 10:41 pts/0    00:00:00 grep --color=auto jenkins
ubuntu@ip-172-31-53-221:~$
```

The screenshot shows the AWS EC2 Instances page with the instance ID **i-01e71d81f62db5d35 (Jenkins)**.

Instance summary for i-01e71d81f62db5d35 (Jenkins)

Instance ID	i-01e71d81f62db5d35 (Jenkins)	Public IPv4 address	100.26.160.84 open address
IPv6 address	-	Instance state	Running
Hostname type	IP name: ip-172-31-53-221.ec2.internal	Private IP DNS name (IPv4 only)	ip-172-31-53-221.ec2.internal
Answer private resource DNS name	IPv4 (A)	Instance type	t2.micro
Auto-assigned IP address	100.26.160.84 [Public IP]	VPC ID	vpc-0895d01f56cc9b25
IAM Role	-	Subnet ID	subnet-01c8c40aad521ac2f
Private IPv4 addresses			
172.31.53.221			
Public IPv4 DNS			
ec2-100-26-160-84.compute-1.amazonaws.com open address			
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

Instance details

Platform	Ubuntu (Inferred)	AMI ID	ami-00874d747dde814fa
Platform details	Linux/UNIX	AMI name	ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20230115
Stop protection	-	Launch time	-
Monitoring disabled			
Termination protection Disabled			
AMI location			

SG – Edit the inbound rule

New EC2 Experience Tell us what you think

EC2 Dashboard
EC2 Global View
Events
Tags
Limits

Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Scheduled Instances
Capacity Reservations

Images
AMIs
AMI Catalog

Elastic Block Store
Volumes
Snapshots

Amazon Machine Images

EC2 > Security Groups > sg-09f5f369a533dffff - launch-wizard-6

sg-09f5f369a533dffff - launch-wizard-6

Details

Security group name	sg-09f5f369a533dffff	Description	VPC ID
Owner	956919395764	Inbound rules count	vpc-0895d01f56cccd9b25
		1 Permission entry	
		Outbound rules count	
		1 Permission entry	

Inbound rules | Outbound rules | Tags

You can now check network connectivity with Reachability Analyzer [Run Reachability Analyzer](#)

Inbound rules (1/1)

Name	Security group rule...	IP version	Type	Protocol	Port range	Source
sgr-00aed4454d7bef67b	SSH	IPv4	SSH	TCP	22	0.0.0.0/0

Manage tags | Edit inbound rules

EC2 > Security Groups > sg-09f5f369a533dffff - launch-wizard-6 > Edit inbound rules

Edit inbound rules [Info](#)

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

Inbound rules [Info](#)

Security group rule ID	Type	Info	Protocol	Info	Port range	Info	Source	Info	Description - optional	Info
sgr-00aed4454d7bef67b	SSH		TCP		22		Custom		0.0.0.0/0	X
-	All traffic		All		All		Anywhere		0.0.0.0/0	X

Add rule | Cancel | Preview changes | Save rules

now Select Public IP address

New EC2 Experience Tell us what you think

EC2 Dashboard
EC2 Global View
Events
Tags
Limits

Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Scheduled Instances
Capacity Reservations

Images
AMIs

EC2 > Instances > i-01e71d81f62db5d35

Instance summary for i-01e71d81f62db5d35

Updated less than a minute ago

Public IPv4 address copied

Instance ID: i-01e71d81f62db5d35 (Jenkins)

IPv6 address: -

Hostname type: IP name: ip-172-31-53-221.ec2.internal

Answer private resource DNS name: IPv4 (A)

Auto-assigned IP address: 100.26.160.84 [Public IP]

IAM Role: -

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

Private IP: 100.26.160.84 | open address

Instance state: Running

Private IP DNS name (IPv4 only): ip-172-31-53-221.ec2.internal

Instance type: t2.micro

VPC ID: vpc-0895d01f56cccd9b25

Subnet ID: subnet-01c8c40aad521ac2f

Private IPv4 addresses: 172.31.53.221

Public IPv4 DNS: ec2-100-26-160-84.compute-1.amazonaws.com | open address

Elastic IP addresses: -

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

Auto Scaling Group name: -

Instance details [Info](#)

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

```
ubuntu@ip-172-31-53-221:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
c2f1e1f6e94648a99ec5ed91c0034886
ubuntu@ip-172-31-53-221:~$
```

Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Getting Started

+

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding	** Sonar Credentials ** Pipeline: Step API ** Plain Credentials Credentials Binding ** SCM API ** Pipeline: API ** commons-lang3 v3.x Jenkins API Timestamper ** Caffeine API ** Script Security ** JAXB ** SnakeYAML API ** Jackson 2 API ** commons-text API ** Plugin Utilities API ** Font Awesome API ** Popper.js 2 API ** Bootstrap 5 API ** JQuery3 API ** ECharts API ** Display URL API ** Pipeline: Supporting APIs ** Checks API ** JUnit ** Matrix Project ** Resource Disposer Workspace Cleanup Ant ** Apache HttpComponents Client 4.x API ** Oracle Java SE Development Kit Installer ** Command Agent Launcher ** Transferring GIT LFS API
✓ Timestamper	✓ Workspace Cleanup	✓ Ant	✓ Gradle	
⌚ Pipeline	⌚ GitHub Branch Source	⌚ Pipeline: GitHub Groovy Libraries	⌚ Pipeline: Stage View	
⌚ Git	⌚ SSH Build Agents	⌚ Matrix Authorization Strategy	⌚ PAM Authentication	
⌚ LDAP	⌚ Email Extension	⌚ Mailer		

Create First Admin User

Username

Password

Confirm password

Full name

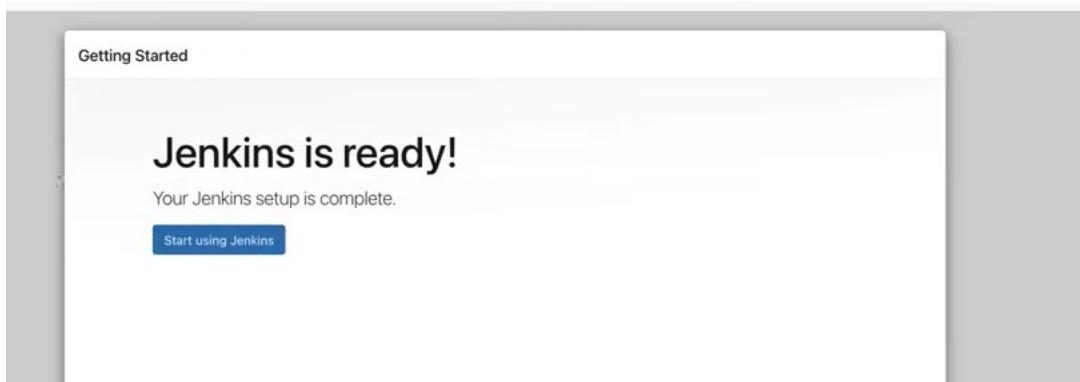
E-mail address

Instance Configuration

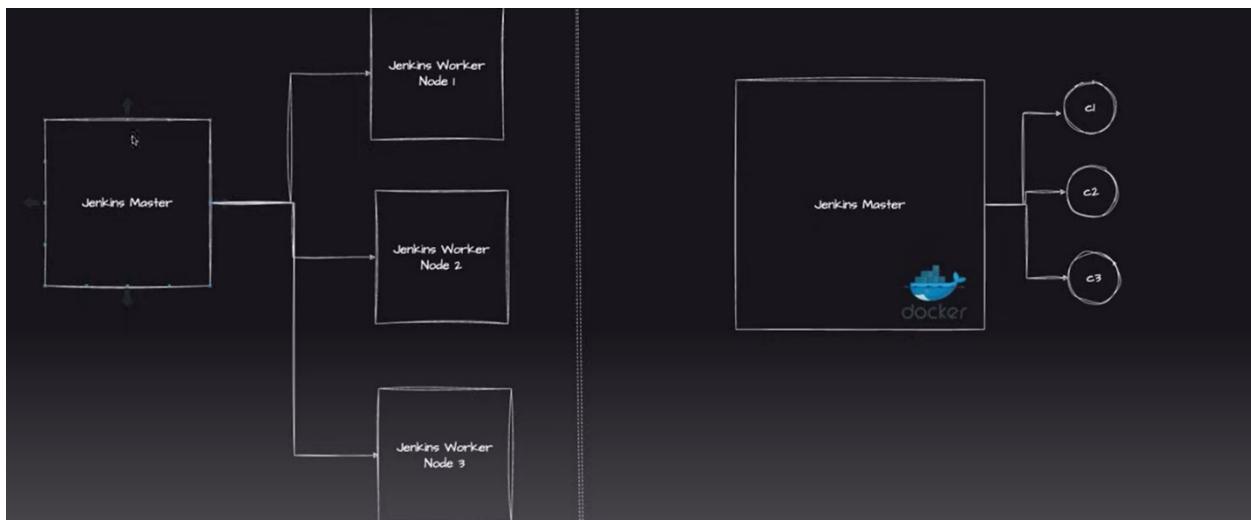
Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

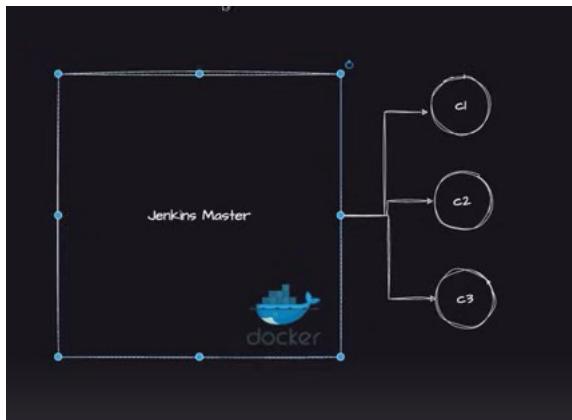


Screenshot of the Jenkins dashboard. The left sidebar includes links for 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. Under 'Build Queue', it says 'No builds in the queue.' On the right, the main area is titled 'Welcome to Jenkins!' with the sub-section 'Start building your software project'. It features a 'Create a job' button, 'Set up a distributed build' section with 'Set up an agent' and 'Configure a cloud' buttons, and a link 'Learn more about distributed builds'. The bottom right corner shows the Jenkins logo and version 'Jenkins 2.375.2'.



Note .. Always use Jenkins master to schedule the task .. & offload

in this Project We use Jenkins with Docker as Agents



now install Docker

```
ubuntu@ip-172-31-53-221:~$ 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
```

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.

```
ubuntu@ip-172-31-53-221:~$ sudo su -
usermod -aG docker jenkins
usermod -aG docker ubuntu
systemctl restart docker
root@ip-172-31-53-221:~# su - jenkins
jenkins@ip-172-31-53-221:~$ docker run hello-world
docker: Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/containers/create": dial unix /var/run/docker.sock: connect: permission denied.
See 'docker run --help'.
jenkins@ip-172-31-53-221:~$
logout
root@ip-172-31-53-221:~# usermod -aG docker jenkins
root@ip-172-31-53-221:~# usermod -aG docker jenkins
root@ip-172-31-53-221:~# su - jenkins
jenkins@ip-172-31-53-221:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:aa0cc8055b82dc2509bed2e19b275c8f463506616377219d9642221ab53cf9fe
Status: Downloaded newer image for hello-world:latest
```

```
2db29710123e: Pull complete
Digest: sha256:aa0cc8055b82dc2509bed2e19b275c8f463506616377219d9642221ab53cf9fe
Status: Downloaded newer image for hello-world:latest
```

Hello from Docker!

This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
 ^(amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

```
$ docker run -it ubuntu bash
```

Share images, automate workflows, and more with a free Docker ID:

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

For more examples and ideas, visit:

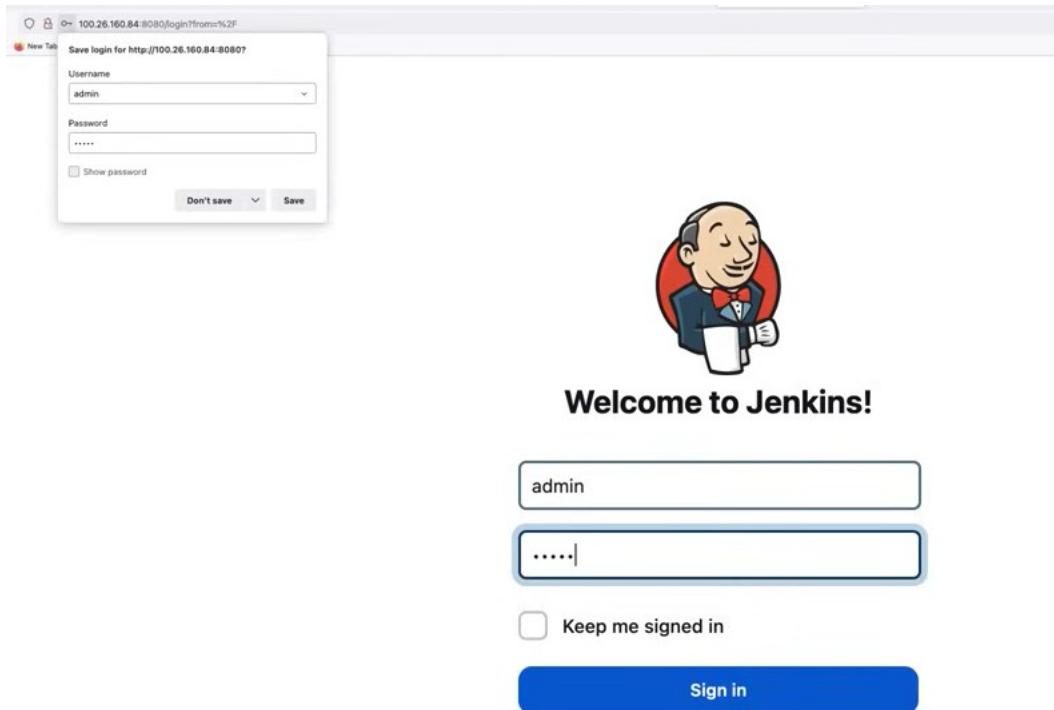
<https://docs.docker.com/get-started/>

```
jenkins@ip-172-31-53-221:~$ ^C
jenkins@ip-172-31-53-221:~$
```

Restart Jenkins

Screenshot of a web browser showing the Jenkins dashboard. A modal dialog box is centered, asking "Are you sure you want to restart Jenkins?". Below the modal, there are several navigation links: "New Item", "People", "Build History", "Manage Jenkins", and "My Views".

The main content area displays a large Jenkins logo with the text "Please wait while Jenkins is restarting . . ." and a message below it stating "Your browser will reload automatically when Jenkins is ready."



```
# now Install Docker Plugins inside the Jenkins
```

☞ **Install the Docker Pipeline plugin in Jenkins:**

- Log in to Jenkins.
- Go to Manage Jenkins > Manage Plugins.
- In the Available tab, search for "Docker Pipeline".
- Select the plugin and click the Install button.
- Restart Jenkins after the plugin is installed.

```
# Click on Manage Jenkins
```

The screenshot shows the Jenkins dashboard. On the left, there's a sidebar with links: 'New Item', 'People', 'Build History', 'Manage Jenkins' (which is highlighted), and 'My Views'. The main content area has a heading 'Welcome to Jenkins!' and a sub-section 'Start building your software project'. It features a 'Create a job' button and several links for setting up distributed builds: 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'. On the far right, there's a yellow gear icon.

Click on manage Plugins

This screenshot shows the 'Manage Jenkins' page. It includes sections for 'Build Queue' (with a note 'No builds in the queue.'), 'Build Executor Status' (showing 1 Idle and 2 Idle), and 'Security'. The 'Manage Plugins' section is highlighted with a light gray background. Other sections shown include 'Configure System', 'Global Tool Configuration', 'Manage Nodes and Clouds', 'Configure Global Security', 'Manage Credentials', 'Configure Credential Providers', and 'Manage Users'.

Updates

Available plugins

Installed plugins

Advanced settings

Plugins

Search: docker pipeline

Name ↓	Released	Installed
No updates		

Update information obtained: 16 min ago [Check now](#)

Updates

Available plugins

Installed plugins

Advanced settings

Plugins

Search: docker pipeline

Install	Name ↓	Released
<input checked="" type="checkbox"/>	Docker Pipeline 563.vd5d2e5c4007f pipeline DevOps Deployment docker Build and use Docker containers from pipelines.	2 mo 1 day ago

This plugin is up for adoption! We are looking for new maintainers. Visit our [Adopt a Plugin](#) initiative for more information.

[Install without restart](#) [Download now and install after restart](#) Update information obtained: 17 min ago [Check now](#)

Updates

Available plugins

Installed plugins

Advanced settings

Download progress

Download progress

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

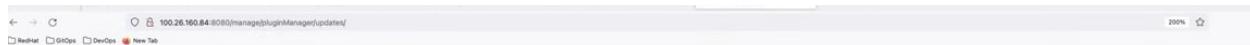
Authentication Tokens API Downloaded Successfully. Will be activated during the next boot

Docker Commons Downloaded Successfully. Will be activated during the next boot

Docker Pipeline Downloaded Successfully. Will be activated during the next boot

→ [Go back to the top page](#)
(you can start using the installed plugins right away)

→ Restart Jenkins when installation is complete and no jobs are running



Save login for http://100.26.160.84:8080/

Username: admin

Password:

Show password

Don't save Save

Welcome to Jenkins!

admin

.....

Keep me signed in

Sign in

Click on New Item

Jenkins

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.

Create a job

Build Executor Status

1 Idle

2 Idle

Set up a distributed build

Set up an agent

Configure a cloud

Learn more about distributed builds

Enter an item name

» 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 separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK

Cancel



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.

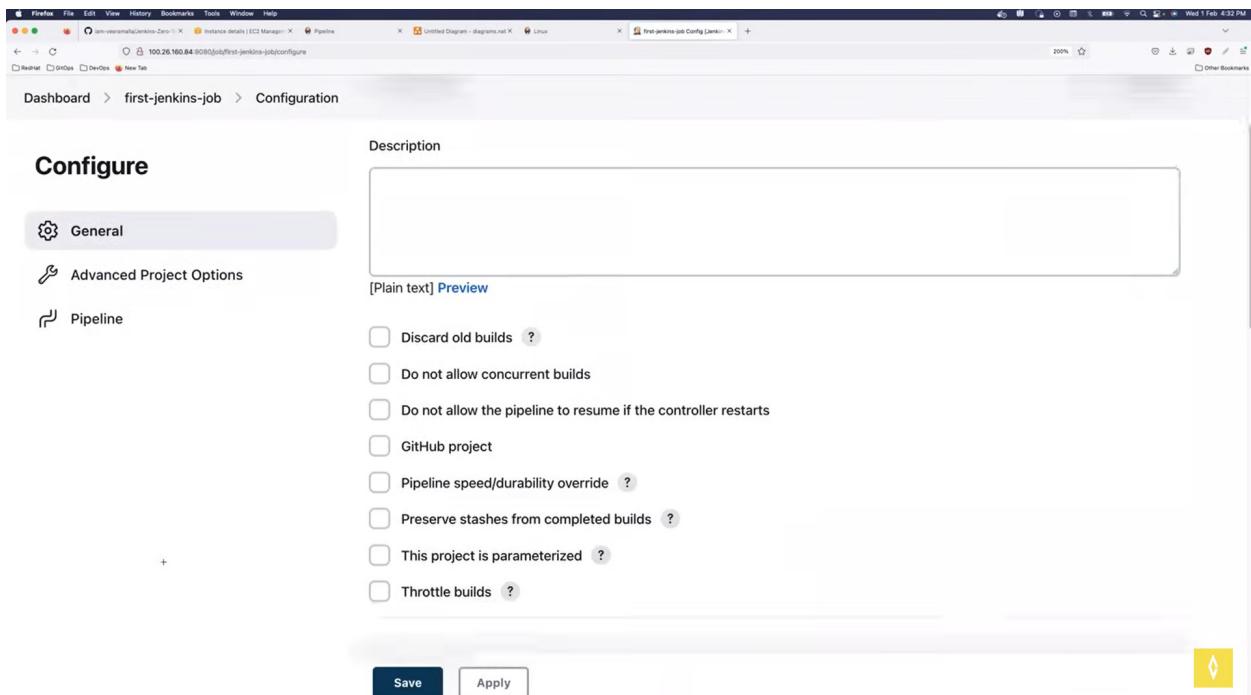
select



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.

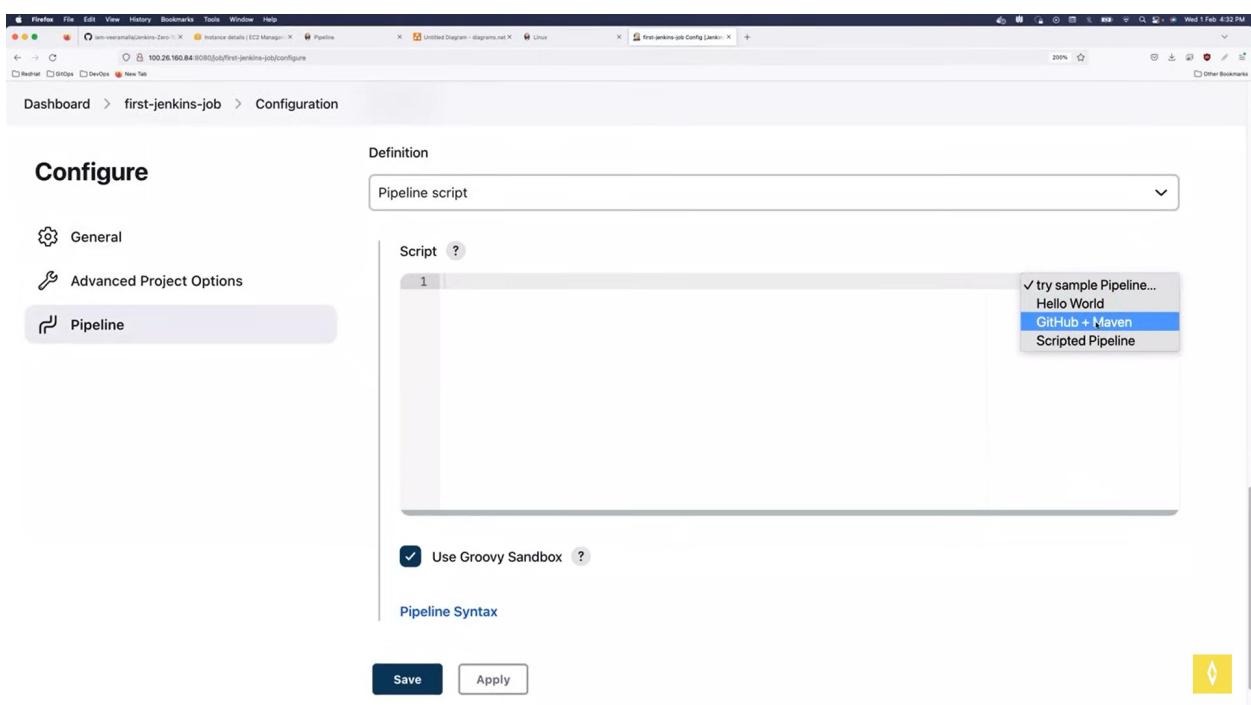
Advantage of selecting pipeline is we can write our own code (declarative or scripted pipelines)



The screenshot shows the Jenkins job configuration page for a job named "first-jenkins-job". The "General" tab is selected. The "Description" field is empty. Under the "Pipeline" section, several checkboxes are listed:

- Discard old builds
- Do not allow concurrent builds
- Do not allow the pipeline to resume if the controller restarts
- GitHub project
- Pipeline speed/durability override
- Preserve stashes from completed builds
- This project is parameterized
- Throttle builds

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



The screenshot shows the Jenkins job configuration page for a job named "first-jenkins-job". The "Pipeline" tab is selected. The "Definition" dropdown is set to "Pipeline script". The "Script" area contains a code editor with a single line of Groovy script: "1". A dropdown menu is open next to the script, showing options: "try sample Pipeline...", "Hello World", "GitHub + Maven" (which is selected), and "Scripted Pipeline". Below the script editor is a checked checkbox for "Use Groovy Sandbox". At the bottom are "Save" and "Apply" buttons.

inbuilt template Draft

Dashboard > first-jenkins-job > Configuration

Configure

Definition

Pipeline script

Script ?

```
1* pipeline {
2    agent any
3
4*   tools {
5       // Install the Maven version configured as "M3" and add it to the path.
6       maven "M3"
7   }
8
9*   stages {
10*     stage('Build') {
11*       steps {
12           // Get some code from a GitHub repository
13           git 'https://github.com/jglick/simple-maven-project-with-tests.git'
14
15           // Run Maven on a Unix agent.
16       }
17     }
18   }
19 }
```

GitHub + Maven

Use Groovy Sandbox ?

Save Apply

Configure

Pipeline script

Script ?

```
1* pipeline {
2    agent any
3
4*   tools {
5       // Install the Maven version configured as "M3" and add it to the path.
6       maven "M3"
7   }
8
9*   stages {
10*     stage('Build') {
11*       steps {
12           // Get some code from a GitHub repository
13           git 'https://github.com/jglick/simple-maven-project-with-tests.git'
14
15           // Run Maven on a Unix agent.
16       }
17     }
18   }
19 }
```

GitHub + Maven

Use Groovy Sandbox ?

The screenshot shows the Jenkins job configuration interface. The top navigation bar includes tabs for Dashboard, first-jenkins-job, Configuration, Pipeline, and Build. The main area is titled 'Configure' under 'Pipeline'. The 'Pipeline script' dropdown is set to 'Script'. Below it is a code editor with the following Groovy script:

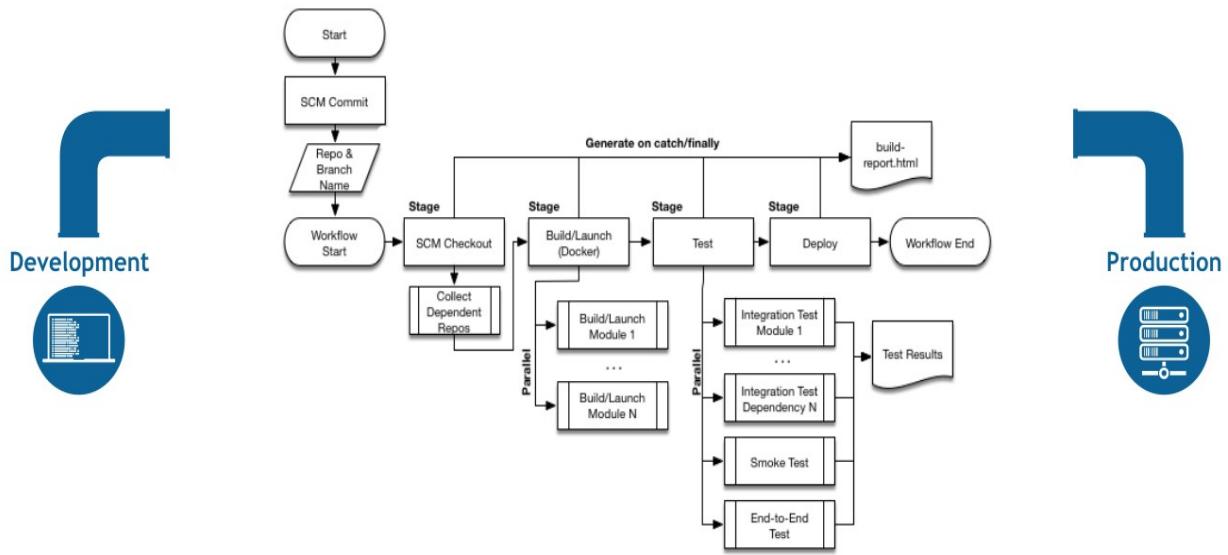
```
1 try sample Pipeline...
```

Below the code editor is a checkbox labeled 'Use Groovy Sandbox'. At the bottom are 'Save' and 'Apply' buttons.

The screenshot shows a code editor window with a dark theme. The title bar says 'Jenkins-Zero-To-Hero'. The left sidebar has icons for file operations, search, and settings. The main area displays a Jenkinsfile with the following content:

```
1 # A simple jenkins pipeline to verify if the docker Agent
2 # configuration is working as expected.
3
4 pipeline {
5     agent {
6         docker { image 'node:16-alpine' }
7     }
8     stages {
9         stage('Test') {
10            steps {
11                sh 'node --version'
12            }
13        }
14    }
15 }
```

The code editor also shows tabs for 'Welcome', 'Jenkinsfile', 'README.md', and 'ESP-IDF: Search Error Hint'. The status bar at the bottom shows various system icons and the date/time '02-11-2024 19:14'.



```

5      // Install the Maven version configured as "M3" and add it to the path.
6      maven "M3"
7  }
8
9  stages {
10     stage("Build") {
11         steps {
12             git 'https://github.com/jglick/simple-maven-project-with-tests.git'
13
14             sh "mvn -Dmaven.test.failure.ignore=true clean package"
15
16             sh "mvn -Dmaven.test.failure.ignore=true clean package"
17
18             sh "mvn -Dmaven.test.failure.ignore=true clean package"
19
20         }
21     }
22 }

```

Sample

The screenshot shows the Jenkins Pipeline configuration page. The pipeline script is defined as follows:

```
1 pipeline {  
2     agent any  
3  
4     stages {  
5         stage('Hello') {  
6             steps {  
7                 echo 'Hello World'  
8             }  
9         }  
10    }  
11}
```

Below the script, there is a checkbox labeled "Use Groovy Sandbox". At the bottom, there are "Save" and "Apply" buttons.

Click on Pipeline Syntax

The screenshot shows the Jenkins Pipeline Syntax Snippet Generator. The search bar at the top contains the text "archiveArtifacts: Archive the artifacts". Below the search bar, the "archiveArtifacts" step is expanded, showing its parameters:

- Files to archive
- Advanced...

At the bottom, there is a "Generate Pipeline Script" button.

The screenshot shows a Jenkins Pipeline Syntax configuration page. The URL in the address bar is `100.26.160.84:8080/jenkins/job/first-jenkins-job/pipeline-syntax`. The page title is "Pipeline Syntax". On the left, there's a sidebar with links: "Global Variables Reference", "Online Documentation", "Examples Reference", and "IntelliJ IDEA GDSL". The main area has a "Sample Step" section with a dropdown menu set to "git: Git". Below it is a "Repository URL" field containing `https://github.com/argoproj/argo-cd`. A "Branch" field is set to "main". Under "Credentials", there's a dropdown set to "- none -" and a "+ Add" button. Two checkboxes are checked: "Include in polling?" and "Include in changelog?". A yellow "Generate Pipeline Script" button is visible. The script output is shown in a large text area:

```
git 'https://github.com/argoproj/argo-cd'
```

Global Variables

There are many features of the Pipeline that are not steps. These are often exposed via global variables, which are not supported by the

Shell Script

The screenshot shows the Jenkins Pipeline Syntax Snippet Generator interface. On the left, there's a sidebar with links: Declarative Online Documentation, Steps Reference, Global Variables Reference, Online Documentation, Examples Reference, and IntelliJ IDEA GDSL. The main area has a title "Steps" and a sub-section "Sample Step". A text input field contains "sh: Shell Script". Below it is a code editor window with the following content:

```
sh
  cd foo
  ./hack/test.
```

Below the code editor is a "Generate Pipeline Script" button. Underneath it, a text input field contains "git 'https://github.com/argoproj/argo-cd'".

The screenshot shows the Jenkins Pipeline Syntax Snippet Generator interface. The main area has a title "Shell Script" and a code editor window with the following content:

```
cd foo
./hack/test.sh
```

Below the code editor is a "Generate Pipeline Script" button. Underneath it, a text input field contains the generated pipeline script:

```
sh '''cd foo
./hack/test.sh'''
```

Global Variables

There are many features of the Pipeline that are not steps. These are often exposed via global variables, which are not supported by the snippet generator. See the [Global Variables Reference](#) for details.

Pipeline script from SCM – Source Code Management Repo

The screenshot shows the Jenkins Pipeline configuration page for a job named "first-jenkins-job". The "Pipeline" tab is selected. The "Definition" dropdown is set to "Pipeline script from SCM". The "Script" pane contains the following Groovy code:

```
1 pipeline {  
2     agent any  
3  
4     tools {  
5         // Install the Maven version configured as "M3" and add it to the path.  
6         maven "M3"  
7     }  
8  
9     stages {  
10        stage('Build') {  
11            steps {  
12                // Get some code from a GitHub repository  
13                git 'https://github.com/jglick/simple-maven-project-with-tests.git'  
14  
15                // Run Maven on a Unix agent.  
16            }  
17        }  
18    }  
19}
```

A checkbox labeled "Use Groovy Sandbox" is checked. At the bottom are "Save" and "Apply" buttons.

The screenshot shows the Jenkins Pipeline configuration page for the same job. The "Pipeline" tab is selected. The "Definition" dropdown is set to "Pipeline script from SCM". The "SCM" section is expanded, showing "Git" selected in the dropdown. The "Repositories" section has a "Repository URL" input field containing "https://github.com/iam-veeramalla/Jenkins-Zero-To-Hero". A red error message below the field says "Please enter Git repository." The "Credentials" section is collapsed.

```
# change Branch to Main
```

The screenshot shows the Jenkins Pipeline configuration page for a job named "first-jenkins-job". The "Pipeline" tab is selected. Under "Branches to build", the "Branch Specifier" is set to "*/*main". Under "Repository browser", it is set to "(Auto)". There are "Save" and "Apply" buttons at the bottom.

Path

The screenshot shows the Jenkins Pipeline configuration page for a job named "first-jenkins-job". The "Pipeline" tab is selected. Under "Script Path", the value is "my-first-pipeline/Jenkinsfile". The "Lightweight checkout" checkbox is checked. There are "Save" and "Apply" buttons at the bottom. The footer includes links for "REST API" and "Jenkins 2.375.2".

EC2 instance

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with links like EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances (selected), Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, and Advanced Instances. The main area has a heading 'Instances (1) Info' and a search bar 'Find instance by attribute or tag (case-sensitive)'. A table lists one instance: Jenkins, with Instance ID i-01e71d81f62db5d35, State Running, Type t2.micro, Status 2/2 checks passed, and Zone us-east-1e. A public IP ec2-100-26-160-84.co... is also listed. Buttons for Connect, Actions, and Launch Instance are at the top right.

Go back to Jenkins – Click on Build now

The screenshot shows the Jenkins Pipeline configuration page for 'Pipeline first-jenkins-job'. The left sidebar includes options like Status, Changes, Build Now (which is highlighted in blue), Configure, Delete Pipeline, Full Stage View, Rename, and Pipeline Syntax. The main area has a 'Stage View' section with a message 'No data available. This Pipeline has not yet run.' and a 'Permalinks' section. At the bottom, there's a 'Build History' table with one entry: '#1 Feb 1, 2023, 11:12 AM'. Buttons for Add description and Disable Project are on the right.

now Fetching the Code from Github

```
Dashboard > first-jenkins-job > #1
Workspaces

Cloning repository https://github.com/iam-veeramalla/Jenkins-Zero-To-Hero
> git init /var/lib/jenkins/workspace/first-jenkins-job # timeout=10
Fetching upstream changes from https://github.com/iam-veeramalla/Jenkins-Zero-To-Hero
> git --version # timeout=10
> git --version # 'git version 2.34.1'
> git fetch --tags --force --progress -- https://github.com/iam-veeramalla/Jenkins-Zero-To-Hero
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/iam-veeramalla/Jenkins-Zero-To-Hero # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 936b83ed3b97013987df16c8ddf122dc82c6d2eb (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 936b83ed3b97013987df16c8ddf122dc82c6d2eb # timeout=10
Commit message: "Update README.md"
First time build. Skipping changelog.
[Pipeline]
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
```

• • •

REST API Jenkins 2.375.2

in Docker , node-16 Alpine img

```
[Pipeline] $+
+ docker pull node:16-alpine
16-alpine: Pulling from library/node
8921db27df28: Pulling fs layer
4e08f9bea56d: Pulling fs layer
b8243c93ff91: Pulling fs layer
2ed4f044afde: Pulling fs layer
2ed4f044afde: Waiting
b8243c93ff91: Verifying Checksum
b8243c93ff91: Download complete
8921db27df28: Verifying Checksum
8921db27df28: Download complete
2ed4f044afde: Verifying Checksum
2ed4f044afde: Download complete
8921db27df28: Pull complete
4e08f9bea56d: Verifying Checksum
4e08f9bea56d: Download complete
```

```

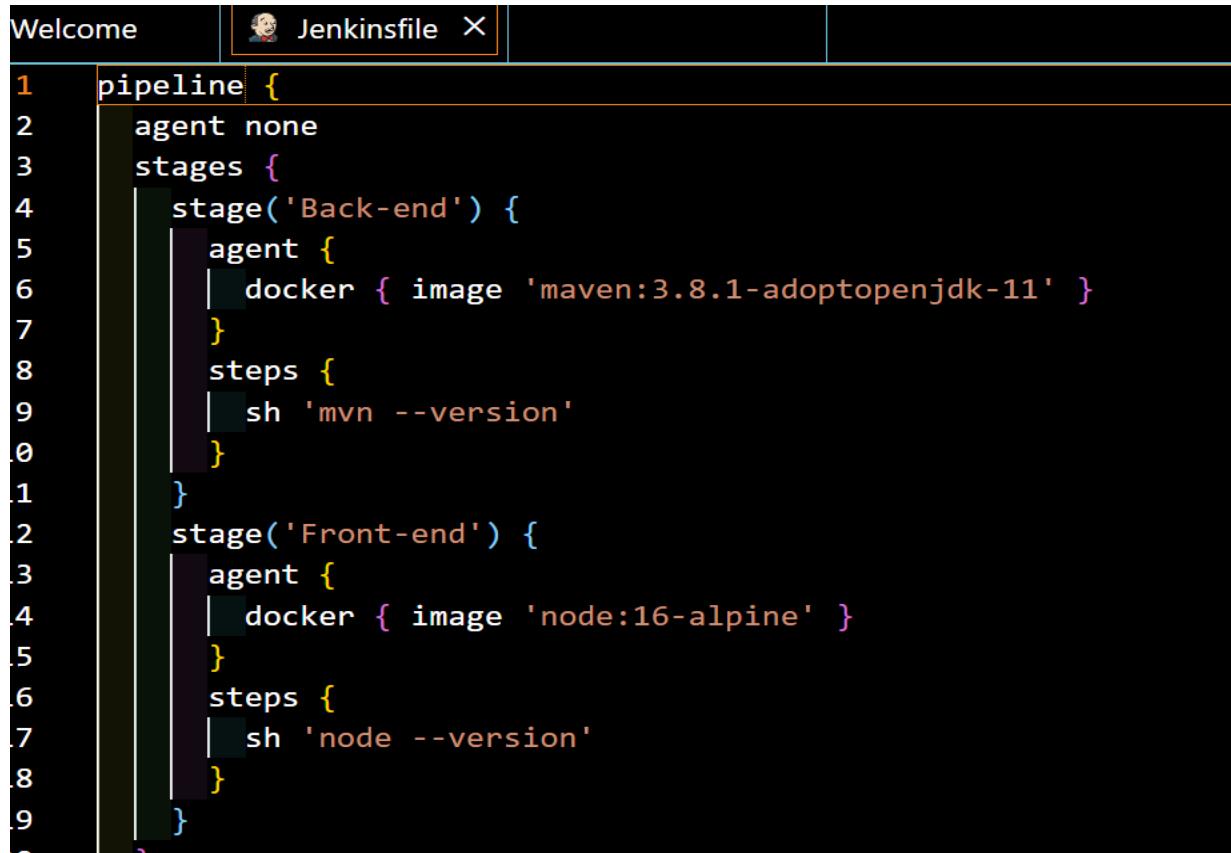
-e ***** node:16-
alpine cat
$ docker top e7cb3165715254ccb15fa46456261e80b50bb6e7b208044fa8125accb3508c9e -eo pid,comm
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Test)
[Pipeline] sh (hide)
+ node --version
v16.19.0
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
$ docker stop --time=1 e7cb3165715254ccb15fa46456261e80b50bb6e7b208044fa8125accb3508c9e
$ docker rm -f --volumes e7cb3165715254ccb15fa46456261e80b50bb6e7b208044fa8125accb3508c9e
[Pipeline] // withDockerContainer
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

Project 2 .. Multistage- MultiAgent - Maven as Backend & node- 16 alpine as Front end

Multi Stage Multi Agent

Set up a multi stage jenkins pipeline where each stage is run on a unique agent. This is a very useful approach when you have multi language application or application that has conflicting dependencies.



The screenshot shows the Jenkinsfile editor interface. The title bar says "Welcome" and "Jenkinsfile X". The main area displays a Groovy script for a Jenkins pipeline:

```

1 pipeline {
2     agent none
3     stages {
4         stage('Back-end') {
5             agent {
6                 docker { image 'maven:3.8.1-adoptopenjdk-11' }
7             }
8             steps {
9                 sh 'mvn --version'
10            }
11        }
12        stage('Front-end') {
13            agent {
14                docker { image 'node:16-alpine' }
15            }
16            steps {
17                sh 'node --version'
18            }
19        }
20    }
21 }

```

Note .. two Containers will be Created

Click on Configure

The screenshot shows the Jenkins interface for a pipeline named "Pipeline first-jenkins-job".

Stage View: Shows a message: "No data available. This Pipeline has not yet run."

Permalinks: Links for Atom feed for all and Atom feed for failures.

Build History: One build listed: #1, Feb 1, 2023, 11:12 AM.

Configuration Tab: Selected tab. Sub-sections include General, Advanced Project Options, and Pipeline.

Definition: Set to "Pipeline script from SCM".

SCM: Set to "Git".

Repositories: Repository URL: `https://github.com/iam-veeramalla/Jenkins-Zero-To-Hero`.

Credentials: None.

Buttons: Save, Apply.

Dashboard > first-jenkins-job > Configuration

Configure

 General

 Advanced Project Options

 Pipeline

Repository browser ?

(Auto) ▼

Additional Behaviours

Add ▾

Script Path ?

multi-stage-multi-agent/Jenkinsfile

Lightweight checkout ?

Pipeline Syntax

Save Apply

Dashboard > first-jenkins-job >

Build Now Disable Project

Configure

Delete Pipeline

Full Stage View

Rename

Pipeline Syntax

Stage View

Average stage times:
(Average full run time: ~25s)

Declarative: Checkout SCM	Test
1s	712ms

#1 Feb 01 16:42 No Changes

1s 712ms

Build History trend ▾

Filter builds... /

#1 Feb 1, 2023, 11:12 AM

Atom feed for all Atom feed for failures

↑ ↓ +

Permalinks

- Last build (#1), 9 min 56 sec ago
- Last stable build (#1), 9 min 56 sec ago
- Last successful build (#1), 9 min 56 sec ago
- Last completed build (#1), 9 min 56 sec ago

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
q3c94435e4bf	maven:3.8.1-adoptopenjdk-11	/usr/local/bin/mvn -	4 seconds ago	Up 1 second		inter
esting_chandrasekhar						
jenkins@ip-172-31-53-221:~\$ docker ps						
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
7958bbd67307	node:16-alpine	docker-entrypoint.s...	2 seconds ago	Up Less than a second		pedantic
_boyd						
jenkins@ip-172-31-53-221:~\$ docker ps						

```

jenkins@ip-172-31-53-221:~$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
a3c94435e4bf maven:3.8.1-adoptopenjdk-11 "/usr/local/bin/mvn..." 4 seconds ago Up 1 second
esting_chandrasekhar
jenkins@ip-172-31-53-221:~$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
7958bbd67307 node:16-alpine "docker-entrypoint.s..." 2 seconds ago Up Less than a second
_pedantic_boyd
jenkins@ip-172-31-53-221:~$ docker ps

```

now docker Container are Terminated

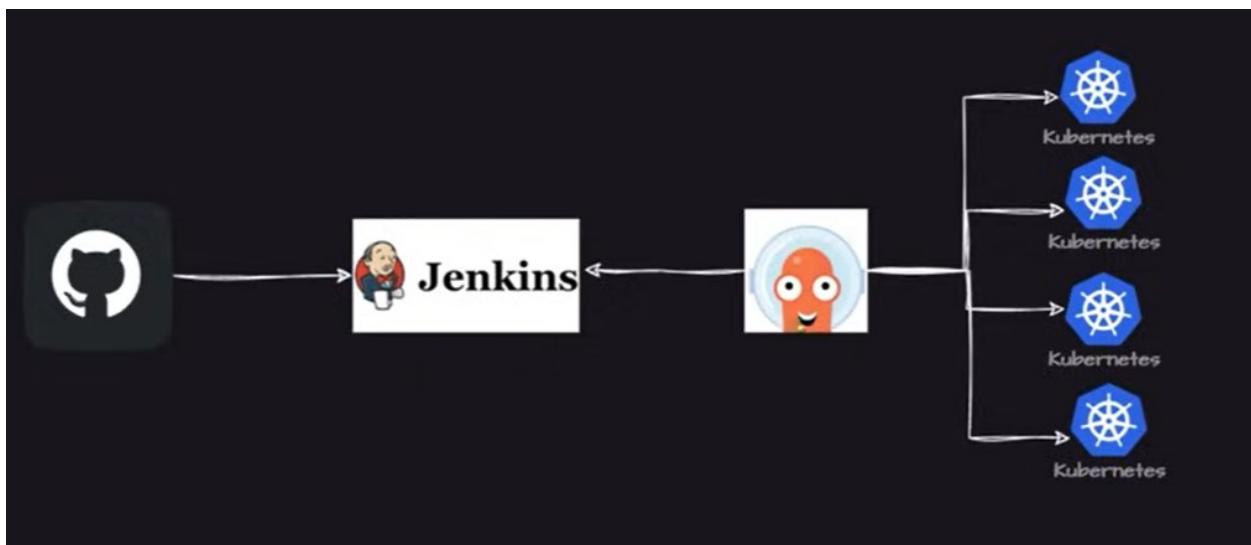
```

_poya
jenkins@ip-172-31-53-221:~$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
jenkins@ip-172-31-53-221:~$ 

```

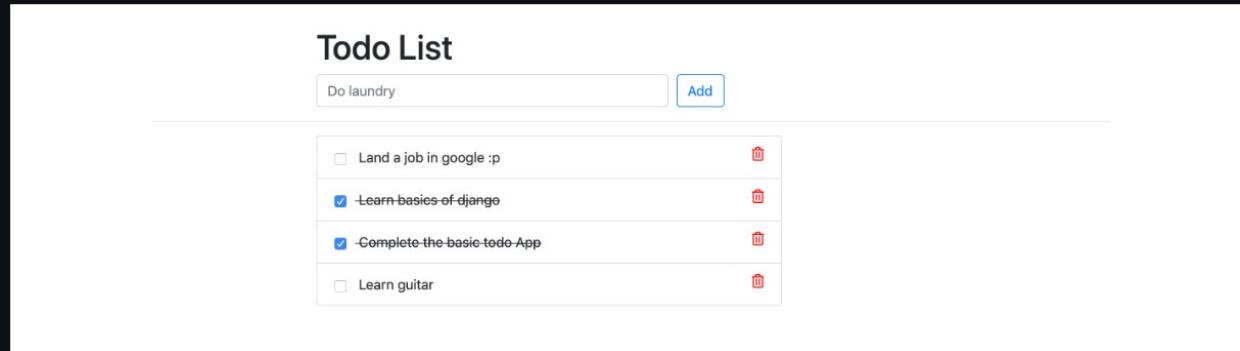
3rd Project – Python Jenkins – argocd-Kubernetes

CICD Architecture [GitHub -> Jenkins -> k8s Manifests -> Argo CD -> k8s cluster]



django-todo

A simple todo app built with django



```
→ cicd-end-to-end git:(main) kubectl get svc
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP  PORT(S)        AGE
example-argocd-metrics  ClusterIP  10.108.124.138 <none>       8082/TCP      99m
example-argocd-redis    ClusterIP  10.107.179.212 <none>       6379/TCP      99m
example-argocd-repo-server ClusterIP  10.96.90.219 <none>       8081/TCP,8084/TCP 99m
example-argocd-server   ClusterIP  10.109.202.90  <none>       80/TCP,443/TCP 99m
example-argocd-server-metrics ClusterIP  10.101.192.62 <none>       8083/TCP      99m
kubernetes          ClusterIP  10.96.0.1     <none>       443/TCP      13d
todo-service         NodePort   10.111.83.255 <none>       80:31000/TCP  64m
→ cicd-end-to-end git:(main)
```

```
→ cicd-end-to-end git:(main) minikube service todo-service
→ cicd-end-to-end git:(main) minikube service list
|-----|-----|-----|-----|-----|
| NAMESPACE | NAME          | TARGET PORT | URL          | |
|---|---|---|---|---|
| argo-rollouts | argo-rollouts-metrics | No node port | |
| 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 | No node port | |
| default | example-argocd-server-metrics | No node port | |
| default | kubernetes | No node port | |
| default | todo-service | 80 | http://192.168.64.4:31000 | |
| ingress-nginx | ingress-nginx-controller | http/80 | http://192.168.64.4:31515 | |
| | | https/443 | http://192.168.64.4:31288 | |
| ingress-nginx | ingress-nginx-controller-admission | No node port | |
| istio-system | istio-egressgateway | No node port | |
| istio-system | istio-ingressgateway | status-port/15021 | http://192.168.64.4:31173 | |
| | | http2/80 | http://192.168.64.4:30257 | |
| | | https/443 | http://192.168.64.4:31342 | |
| | | tcp/31400 | http://192.168.64.4:30280 | |
| | | tls/15443 | http://192.168.64.4:30045 | |
| istio-system | istiod | No node port | |
| kube-system | kube-dns | No node port | |
|-----|-----|-----|-----|-----|
→ cicd-end-to-end git:(main)
```

The screenshot shows a web-based todo list application. At the top, there's a navigation bar with links like 'Getting Started', 'GitOps', 'AWS', 'DevOps', 'Youtube', 'operators', 'Argo CD', 'terraform', 'golang', 'CrossPlane', 'Security Guild', 'RedHat', 'OpenSource', 'GITLAB', 'Flux', and 'ZAP'. Below the navigation, the title 'Todo List - Modified once again' is displayed. A search bar contains the text 'Do laundry' with an 'Add' button next to it. The main area lists three tasks:

- Docker file banao
- Send Resume Google now !
- Hacktoberfest Updates

The screenshot shows a GitHub repository page for 'Jenkins-Zero-To-Hero/python-jenkins-argocd-k8s'. The repository has 11 commits from 'iam-veeramalla' with the message 'Update README.md'. The commits are listed as follows:

Name	Last commit message	Last commit date
..		
deploy	add cicd pipeline with argocd and k8s	last year
staticfiles	add cicd pipeline with argocd and k8s	last year
todoApp	add cicd pipeline with argocd and k8s	last year
todos	add cicd pipeline with argocd and k8s	last year
Dockerfile	add cicd pipeline with argocd and k8s	last year
Jenkinsfile	add cicd pipeline with argocd and k8s	last year
LICENSE	add cicd pipeline with argocd and k8s	last year
README.md	Update README.md	last year
db.sqlite3	add cicd pipeline with argocd and k8s	last year
docker-compose.yml	add cicd pipeline with argocd and k8s	last year
manage.py	add cicd pipeline with argocd and k8s	last year

At the bottom of the repository page, there's a 'README.md' file with a preview and edit options. The Windows taskbar at the bottom shows various pinned icons and the system tray with weather information (26°C Sunny), battery level (11:28), and network status (03-11-2024).

Pipeline cicd-end-to-end-demo

This Pipeline demonstrates an End to End CI/CD Process.

Stage View

	Declarative: Checkout SCM	Checkout	Build Docker	Push the artifacts	Checkout K8S manifest SCM	Update K8S manifest & push to Repo
Average stage times: (Average full run time: ~1min 3s)	2s	1s	10s	18s	1s	3s
#32 Oct 19 19:13 6 commits	3s	1s	45s	23s	3s	6s
#31 Oct 19 01:22 1 commit	1s	1s	4s	17s	1s	3s
#30 Oct 19 01:20 1 commit	1s	1s	8s	18s	1s	3s

make Change in index.html

```
% extends 'todos/base.html'
{
    % block title %
        <title>Todo list</title>
    {%
        % endblock %
    }
    % block content %
        <div class="container">
            <!-- title row -->
            <div class="row">
                <div class="offset-md-2 col-lg-9">
                    <div class="page-header">
                        <h1>
                            Todo List - Abhishek
                        </h1>
                    </div>
                </div>
            </div>
            <!-- Add a todo row -->
            <div class="row">
                <div class="offset-md-2 col-lg-9">
                    <form method="post" action="{% url 'todos:add' %}">
                        {% csrf_token %}
                        <div class="form-row">
                            <div class="col-md-6">
                                <input type="text" class="form-control" name="title" placeholder="Do laundry" required>
                            </div>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    {%
        % endblock %
    }
}
```

Click on build now in Jenkins

The screenshot shows a Jenkins console output page. The left sidebar has a 'Console Output' section selected. The main area displays the build log:

```

Started by user unknown or anonymous
Obtained Jenkinsfile from git https://github.com/iam-veeramalla/cicd-end-to-end
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /Users/aveerama/.jenkins/workspace/cicd-end-to-end-demo
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /Users/aveerama/.jenkins/workspace/cicd-end-to-end-demo/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/iam-veeramalla/cicd-end-to-end # timeout=10
Fetching upstream changes from https://github.com/iam-veeramalla/cicd-end-to-end
> git --version # timeout=10
> git --version # 'git version 2.37.0 (Apple Git-136)'
> git fetch --tags --force --progress -- https://github.com/iam-veeramalla/cicd-end-to-end +refs/heads/*:refs/remotes/origin/* # timeout=10

```

At the bottom right, it says 'REST API' and 'Jenkins 2.303.1'.

Jenkinsfile

The screenshot shows a GitHub code editor displaying a Jenkinsfile. The file content is as follows:

```

pipeline {
    agent any
    environment {
        IMAGE_TAG = "${BUILD_NUMBER}"
    }
    stages {
        stage('Checkout'){
            steps {
                git credentialsId: 'f87a3a8-0e09-45e7-b9cf-6dc08feac070',
                url: 'https://github.com/iam-veeramalla/cicd-end-to-end',
                branch: 'main'
            }
        }
        stage('Build Docker'){
            steps{
                script{
                    sh '''
                        echo 'Build Docker Image'
                        docker build -t abhishekfs/cicd-e2e:${BUILD_NUMBER} .
                    '''
                }
            }
        }
    }
}

```

A screenshot of a Windows desktop environment. In the center is a GitHub code editor window displaying a Jenkinsfile. The file contains several stages: 'Build Docker', 'Push the artifacts', 'Checkout K8S manifest SCM', and 'Update K8S manifest & push to Repo'. Each stage includes steps for running shell scripts, pushing Docker images, and managing Git repositories. The GitHub interface shows 67 lines of code with 59 loc and a size of 2.05 KB. A GitHub Copilot icon is visible in the top right. The taskbar at the bottom shows various pinned icons and the date/time as 03-11-2024.

```
17     }
18
19     stage('Build Docker'){
20         steps{
21             script{
22                 sh """
23                     echo 'Build Docker Image'
24                     docker build -t abhishekfs/cicd-e2e:${BUILD_NUMBER} .
25                 """
26             }
27         }
28     }
29
30     stage('Push the artifacts'){
31         steps{
32             script{
33                 sh """
34                     echo 'Push to Repo'
35                     docker push abhishekfs/cicd-e2e:${BUILD_NUMBER}
36                 """
37             }
38         }
39     }
40
41     stage('Checkout K8S manifest SCM'){
42         steps {
43             git credentialsId: 'f87a34a8-0e09-45e7-b9cf-6dc68feac670',
44             url: 'https://github.com/iam-veeramalla/cicd-demo-manifests-repo.git',
45             branch: 'main'
46         }
47     }
48 }
```

A second screenshot of a Windows desktop environment, showing the same GitHub code editor window with the Jenkinsfile. The code is identical to the one in the first screenshot, containing four stages: 'Build Docker', 'Push the artifacts', 'Checkout K8S manifest SCM', and 'Update K8S manifest & push to Repo'. The GitHub interface shows 67 lines of code with 59 loc and a size of 2.05 KB. A GitHub Copilot icon is visible in the top right. The taskbar at the bottom shows various pinned icons and the date/time as 03-11-2024.

```
38     }
39 }
40
41 stage('Checkout K8S manifest SCM'){
42     steps {
43         git credentialsId: 'f87a34a8-0e09-45e7-b9cf-6dc68feac670',
44         url: 'https://github.com/iam-veeramalla/cicd-demo-manifests-repo.git',
45         branch: 'main'
46     }
47 }
48
49 stage('Update K8S manifest & push to Repo'){
50     steps {
51         script{
52             withCredentials([usernamePassword(credentialsId: 'f87a34a8-0e09-45e7-b9cf-6dc68feac670', passwordVariable: 'GIT_PASSWORD', usernameVariable: 'GIT_USERNAME')]) {
53                 sh """
54                     cat deploy.yaml
55                     sed -i '' "s/32/${BUILD_NUMBER}/g" deploy.yaml
56                     cat deploy.yaml
57                     git add deploy.yaml
58                     git commit -m 'Updated the deploy yaml | Jenkins Pipeline'
59                     git remote -v
60                     git push https://github.com/iam-veeramalla/cicd-demo-manifests-repo.git HEAD:main
61                 """
62             }
63         }
64     }
65 }
66 }
67 }
```

now for Code Generator snippet .. in Jenkins .. it gives You Git ID ...

 Jenkins

Dashboard > cicd-end-to-end-demo > Pipeline Syntax

[Back](#)

[Snippet Generator](#)

[Declarative Directive Generator](#)

[Declarative Online Documentation](#)

[Steps Reference](#)

[Global Variables Reference](#)

[Online Documentation](#)

[Examples Reference](#)

[IntelliJ IDEA GDSL](#)

Overview

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Script**, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

Steps

Sample Step

git: Git

git

Repository URL

`https://github.com/iam-veeramalla/test.git`

Please enter Git repository.

Branch

master

Credentials

`https://github.com/iam-veeramalla/test.git`

Failed to connect to repository : Command "git ls-remote -h -- https://github.com/iam-veeramalla/test.git HEAD" returned status code 128:

`stdout:
stderr: remote: Repository not found.
fatal: repository 'https://github.com/iam-veeramalla/test.git/' not found`

Branch

main

Credentials

iam-veeramalla/******** [Add](#)

Include in polling?

Include in changelog?

Generate Pipeline Script

```
git branch: 'main', credentialsId: 'f87a34a8-0e09-45e7-b9cf-6dc68feac670', url: 'https://github.com/iam-veeramalla/test.git'
```

```

Dashboard > cicd-end-to-end-demo > #33
288cf3a46e32: Preparing
186da83755d: Preparing
955c9335e041: Preparing
8e079fee2186: Preparing
57b220df0ae0: Waiting
0c7daf9a72c8: Waiting
75ba02937496: Waiting
288cf3a46e32: Waiting
186da83755d: Waiting
955c9335e041: Waiting
8e079fee2186: Waiting
1bd09c3779f2: Layer already exists
dcc5479aae4e: Layer already exists
d91c7bf5ffb3: Layer already exists
57b220df0ae0: Layer already exists
0c7daf9a72c8: Layer already exists
75ba02937496: Layer already exists
186da83755d: Layer already exists
288cf3a46e32: Layer already exists
955c9335e041: Layer already exists
8e079fee2186: Layer already exists
beec54e011f1: Pushed
f77ab5e3e09: Pushed
33: digest: sha256:5cf48395cb2815e069d7df8325d7be5b1bfe0b9eba41b79f47df3fc912024506 size: 2850
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage {
[Pipeline] { (Checkout K8S manifest SCM)
[Pipeline] git
The recommended git tool is: NONE
using credential f87a34a8-0e09-45e7-b9cf-6dc68feac670
> git rev-parse --resolve-git-dir /Users/aveerama/.jenkins/workspace/cicd-end-to-end-demo/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/iam-veeramalla/cicd-demo-manifests-repo.git # timeout=10
Fetching upstream changes from https://github.com/iam-veeramalla/cicd-demo-manifests-repo.git
> git --version # timeout=10

```

Showing 1 changed file with 1 addition and 1 deletion.

```

@@ -16,7 +16,7 @@ spec:
16   spec:
17     containers:
18       - name: todo
19 -      image: abhishekf5/cicd-e2e:32
20     ports:
21       - containerPort: 8000
22

```

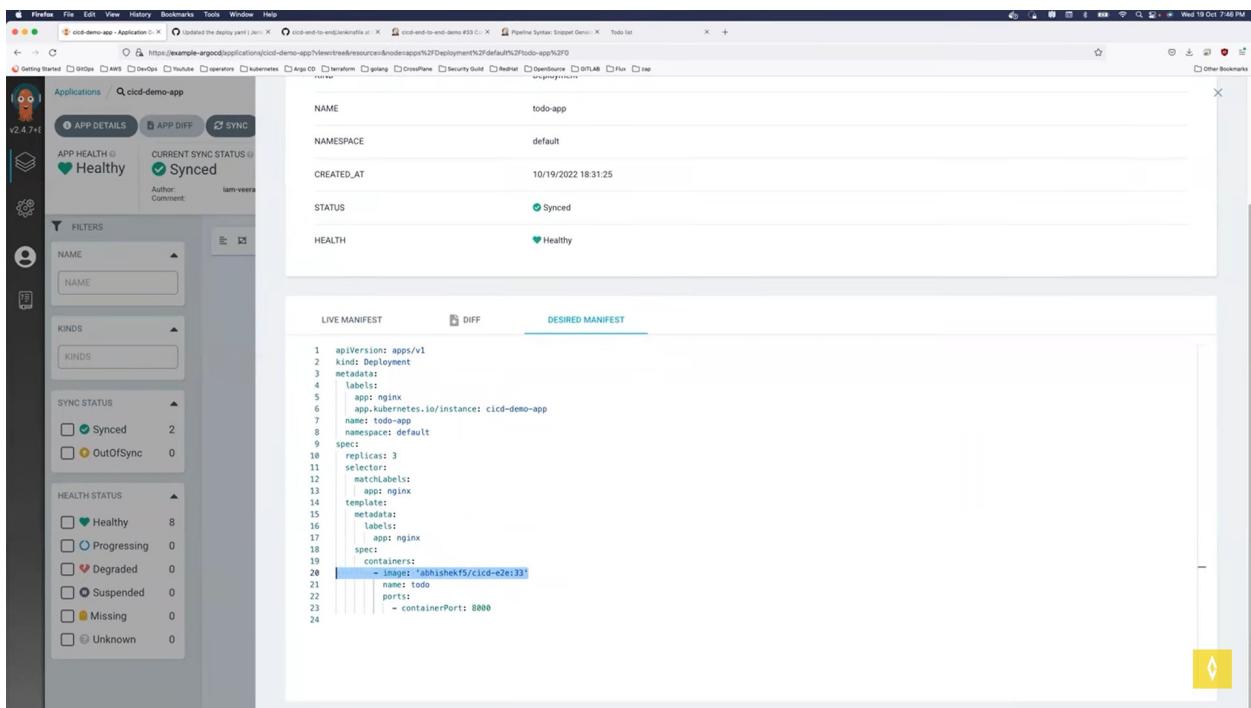
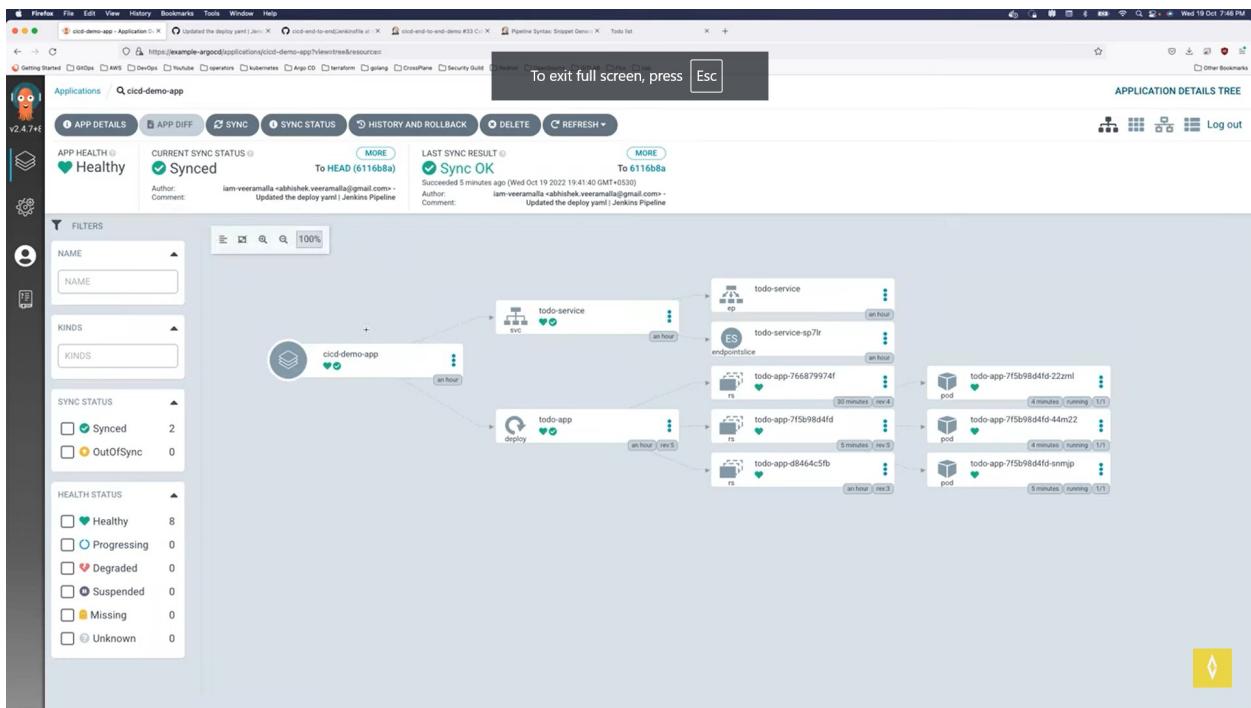
0 comments on commit 6116b8a

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Comment on this commit



```

→ cicd-end-to-end git:(main) kubectl get deploy
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
example-argocd-redis   1/1     1           1           110m
example-argocd-repo-server 1/1     1           1           110m
example-argocd-server    1/1     1           1           110m
todo-app        3/3     3           3           75m
→ cicd-end-to-end git:(main) kubectl edit deploy todo-app

```

#img tag updated

```

app: nginx
strategy:
  rollingUpdate:
    maxSurge: 25%
    maxUnavailable: 25%
  type: RollingUpdate
template:
  metadata:
    creationTimestamp: null
    labels:
      app: nginx
spec:
  containers:
    - image: abhishekf5/cicd-e2e:33
      imagePullPolicy: IfNotPresent
      name: todo
      ports:
        - containerPort: 8000
          protocol: TCP
      resources: {}
      terminationMessagePath: /dev/termination-log
      terminationMessagePolicy: File
    dnsPolicy: ClusterFirst
    restartPolicy: Always
    schedulerName: default-scheduler
    securityContext: {}
    terminationGracePeriodSeconds: 30

```

Clipboard Image Tools Shapes Colors

Firefox File Edit View History Bookmarks Tools Window Help

Complete CI/CD setup with Live Demo | #devops #jenkins| Write CI/CD w...

Todo List

+ Add

Do laundry

Docker file banao

Send Resume Google now !

Hacktoberfest Updates

