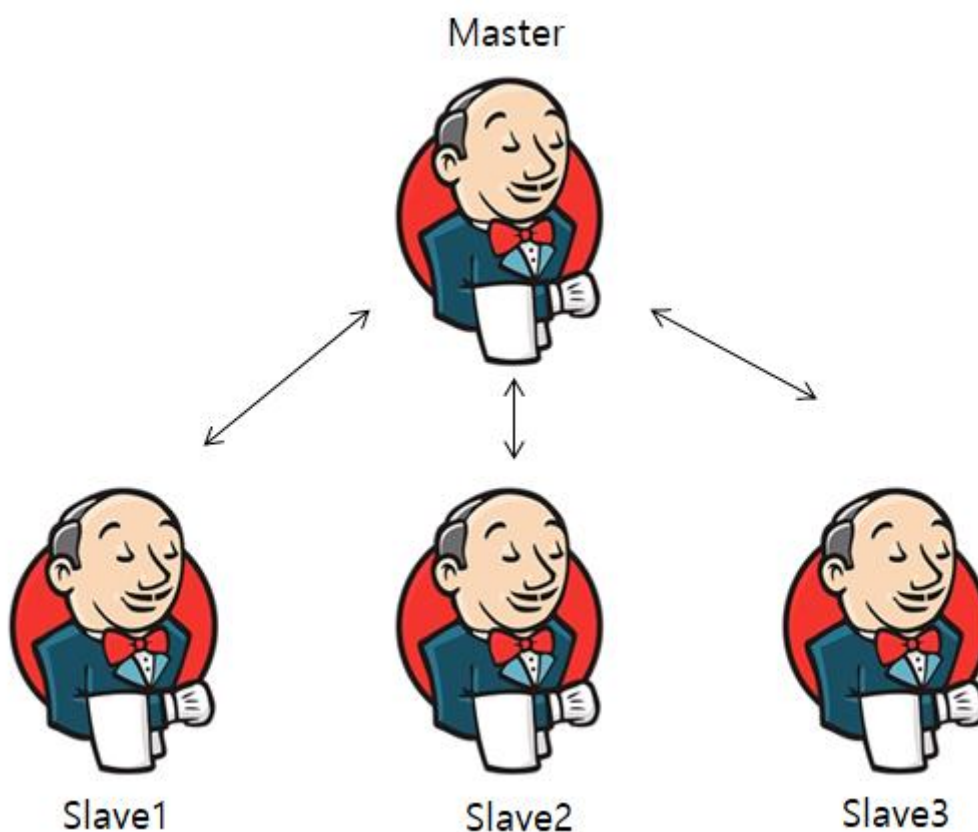


JENKINS MASTER AND SLAVE

INTRODUCTION

In today's fast-paced software development landscape, continuous integration and delivery (CI/CD) have become indispensable practices for ensuring rapid and reliable software releases. At the heart of CI/CD automation lies Jenkins, an open-source automation server that enables developers to automate various stages of the software development lifecycle. In this blog, we'll delve into the concept of Jenkins and explore the significance of its master-slave architecture in scaling CI/CD pipelines.



Understanding Jenkins

Jenkins is a widely-used automation server that facilitates the automation of building, testing, and deploying software applications. It

provides a flexible and extensible platform for orchestrating CI/CD workflows, allowing teams to automate repetitive tasks and streamline the software delivery process. Jenkins supports integration with a myriad of tools and technologies, making it a versatile solution for modern software development teams.

Master-Slave Architecture

The master-slave architecture in Jenkins enables the distribution of workload across multiple nodes, thereby improving efficiency and scalability. In this setup, the Jenkins master node serves as the central controller responsible for managing job execution and distributing tasks to one or more slave nodes. Each slave node operates independently and executes build jobs assigned to it by the master. This distributed approach allows for parallel execution of tasks, reducing build times and enhancing resource utilization.

Benefits and Use Cases

The master-slave architecture offers several benefits, including:

- **Scalability:** Easily scale Jenkins infrastructure by adding or removing slave nodes based on workload demands.
- **Fault Tolerance:** Distributed architecture enhances fault tolerance and resilience against node failures.
- **Resource Optimization:** Efficiently utilize computing resources by distributing workload across multiple nodes.
- **Parallel Execution:** Execute multiple build jobs concurrently, accelerating the software delivery pipeline.

Use cases for Jenkins master-slave architecture include:

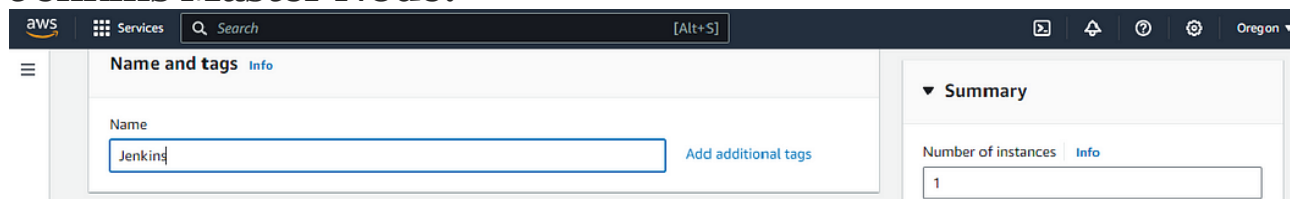
- **Large-Scale Builds:** Handle large and complex builds more effectively by distributing workload across multiple nodes.
- **Geographic Distribution:** Deploy slave nodes in different geographical regions to minimize latency and improve performance.
- **Specialized Environments:** Utilize dedicated slave nodes for specific tasks such as integration testing, deployment, or performance testing.

Practical Guide: Setting Up Jenkins Master-Slave Architecture on AWS EC2

Prepare AWS Environment:

Launch EC2 instances for Jenkins master and slave nodes, ensuring proper network configuration and security group settings.

Jenkins Master Node:



The screenshot shows the AWS Management Console interface for creating a new EC2 instance. The top navigation bar includes the AWS logo, 'Services', a search bar, and the region 'Oregon'. The main content area is divided into two panels. The left panel, titled 'Name and tags', contains a text input field for the instance name, which is set to 'Jenkins', and a link to 'Add additional tags'. The right panel, titled 'Summary', shows the 'Number of instances' set to '1'.

Recents

Quick Start

Amazon Linux
aws

macOS
Mac

Ubuntu
ubuntu

Windows
Microsoft

Red Hat
Red Hat

SUSE Linux
SUS

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI

ami-01e82af4e524a0aa3 (64-bit (x86), uefi-preferred) / ami-01c3c55948a949a52 (64-bit (Arm), uefi)

Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

Description

Amazon Linux 2023 AMI 2023.3.20240205.2 x86_64 HVM kernel-6.1

Architecture

64-bit (x86)

Boot mode

uefi-preferred

AMI ID

ami-01e82af4e524a0aa3

Verified provider

▼ Instance type Info | Get advice

Instance type

t2.medium

Family: t2 2 vCPU 4 GiB Memory Current generation: true

On-Demand SUSE base pricing: 0.1464 USD per Hour

On-Demand Linux base pricing: 0.0464 USD per Hour

On-Demand RHEL base pricing: 0.1064 USD per Hour

On-Demand Windows base pricing: 0.0644 USD per Hour

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

Create the new key pair for these instances also:

▼ Key pair (login) Info

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

Key pair name - required

Select

Create new key pair

Firewall (security groups) | Info

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

☒ Create security group

☐ Select existing security group

We'll create a new security group called '**launch-wizard-2**' with the following rules:

- ☒ Allow SSH traffic from
Helps you connect to your instance
- ☒ Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server
- ☒ Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

Anywhere
0.0.0.0/0



After successfully launching instance, install the Jenkins in master node and also install java.

Jenkins slave nodes

With the same instance configurations, we will launch the 1 Jenkins slave node and configure it from inside to launch the slave node

Basic Pre-requisite in slave node for its successful configuration:

```

_ / m /
[ec2-user@ip-172-31-22-79 ~]$ sudo yum install java -y
Last metadata expiration check: 0:39:15 ago on Sun Feb 18 18:55:45 2024.
Dependencies resolved.
=====
Package                                Architecture  Version
=====
Installing:
  java-21-amazon-corretto              x86_64        1:21.0.2+13-1.amzn2023.1
Installing dependencies:
  alsa-lib                             x86_64        1.2.7.2-1.amzn2023.0.2
  cairo                                x86_64        1.17.6-2.amzn2023.0.1
  dejavu-sans-fonts                    noarch        2.37-16.amzn2023.0.2
  dejavu-sans-mono-fonts               noarch        2.37-16.amzn2023.0.2
  dejavu-serif-fonts                  noarch        2.37-16.amzn2023.0.2
  fontconfig                           x86_64        2.13.94-2.amzn2023.0.2
  fonts-filesystem                    noarch        1:2.0.5-12.amzn2023.0.2
  freetype                             x86_64        2.13.0-2.amzn2023.0.1
  giflib                               x86_64        5.2.1-9.amzn2023.0.1
  google-noto-fonts-common             noarch        20201206-2.amzn2023.0.2
  google-noto-sans-vf-fonts            noarch        20201206-2.amzn2023.0.2
  graphite2                            x86_64        1.3.14-7.amzn2023.0.2
=====

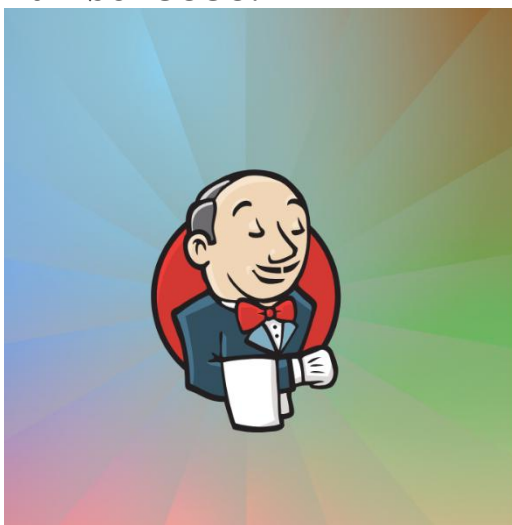
```

Now our plan is to set up one slave node for **Docker** builds.
So first set up node as docker slave, run below command in that slave node:

- `sudo yum install docker -y #linux 2023`
- `sudo usermod -aG docker ec2-user`
- `newgrp docker`
- `sudo service docker start`
- `sudo chmod 777 /var/run/docker.sock`

Access the Jenkins Master Node from WebUI

Access the Jenkins master node on the browser using its public IP at port number 8080.



Sign in to Jenkins

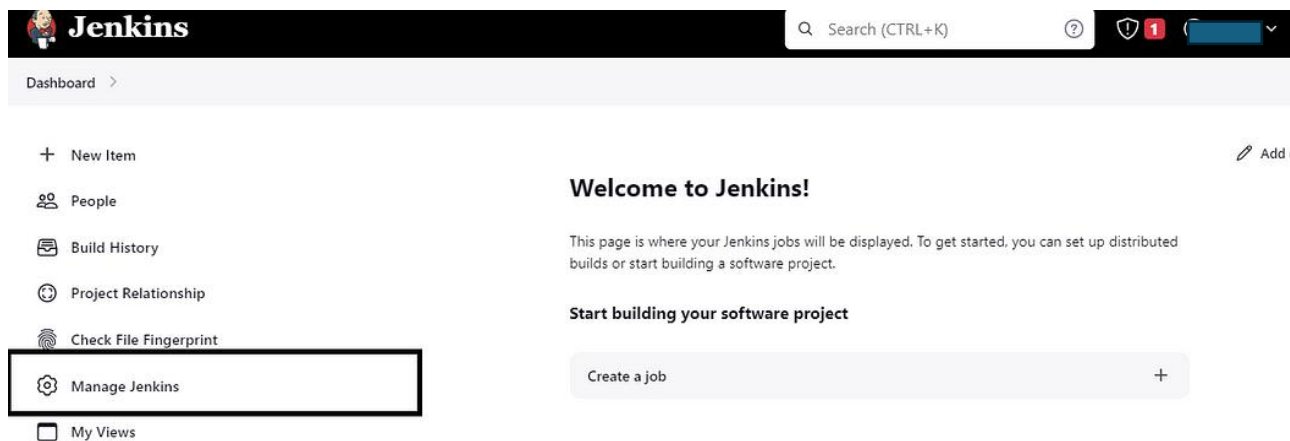
Username

Password

☐ Keep me signed in

Sign in

Go to the “Manage Jenkins” option from your dashboard :

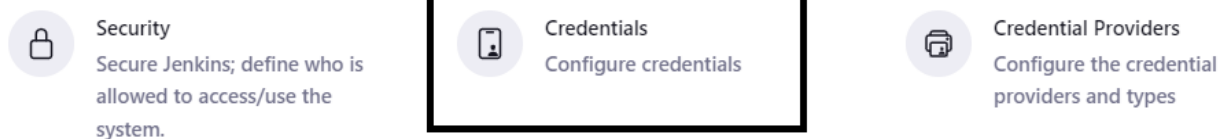


--Go to plugins install ssh agent.

Setting Credentials :

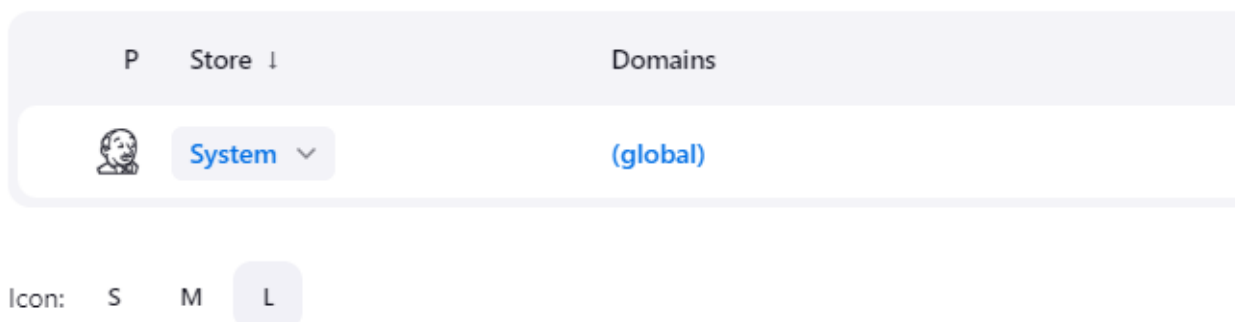
On the “Manage Jenkins” section, click on the “Credentials” in security section :

Security




Click on “system” :

Stores scoped to Jenkins



System

Domain ↓	Description
 Global credentials (unrestricted) ▼	Credentials that should be available irrespective of domain specification to requirements matching.

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

Global credentials (unrestricted) [+ Add Credentials](#)

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

Choose “SSH Username with private key” on this option:

New credentials

Kind

SSH Username with private key

Scope ?

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

Then give the username you want to access, here we will give “ec2-user” and then copy the same key that we have created while launching the slave node.

Private Key

☒ Enter directly

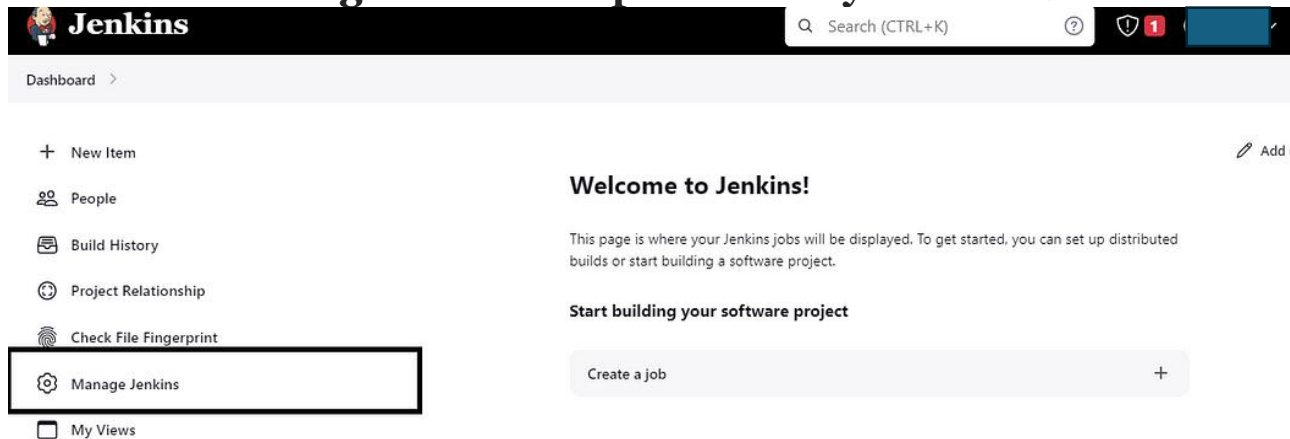
Key

Enter New Secret Below

```
MPDBA0GAE7LHWLP0C7gqEK1YELQCAU0CD1TVZ#80L0zy0ZQTAX0VEDyDKE0MVZS  
e3LNgox9wdn9E0Ccc52kwjGMnCw45jY+tQuYqrytQT1wW+sUApeQ33bs2r9riogR  
wr4d4L1D37fZX7DomdIboeezrN62w38oR4t1vA/AX06CJb5vru0=  
-----END RSA PRIVATE KEY-----
```

And create this credentials, so that we can use it while adding the slave node on the Jenkins.

Go to the “Manage Jenkins” option from your dashboard :



The screenshot shows the Jenkins dashboard. On the left sidebar, the 'Manage Jenkins' option is highlighted with a black box. The main content area displays a 'Welcome to Jenkins!' message and a 'Create a job' button.

Go and search java path in master node and copy the java path

```
[ec2-user@ip-172-31-54-32 ~]$ cd /usr/lib/jvm/java-17-amazon-corretto.x86_64/  
[ec2-user@ip-172-31-54-32 java-17-amazon-corretto.x86_64]$ ls  
bin  conf  legal  lib  man  release  
[ec2-user@ip-172-31-54-32 java-17-amazon-corretto.x86_64]$ pwd  
/usr/lib/jvm/java-17-amazon-corretto.x86_64  
[ec2-user@ip-172-31-54-32 java-17-amazon-corretto.x86_64]$
```

Manage Jenkins—tools- add jdk {java path}

Paste java path here and save

Dashboard > Manage Jenkins > Tools

JDK installations ^ Edited

Add JDK

JDK

Name

JAVA_HOME

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 ?

Save Apply

Click on “Nodes” option on your screen :
Manage Jenkins

Search settings /

Building on the built-in node can be a security issue. You should set the number of executors on the built-in node to 0. See [the documentation](#). **Manage** **Dismiss**

System Configuration

- System**
Configure global settings and paths.
- Tools**
Configure tools, their locations and automatic installers.
- Plugins**
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Nodes**
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Clouds**
Add, remove, and configure cloud instances to provision agents on-demand.

Then click on “+New Node” button.

Nodes **+ New Node** Node Monitoring ↻

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	7.69 GB	! 0 B	1.90 GB	0ms

Then write the name you want to give to this slave node:

New node

Node name

Slave 1 - Docker

Type



Permanent Agent

Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

Then we will add further details for that node, i.e, number of executors and that workspace remote root directory is default other wise we can create and pass here:

Number of executors ?

2

Remote root directory ?

/home/ec2-user

Then we will set label for that Slave node :

Labels ?

Node_For_Docker

Usage ?

Use this node as much as possible

Then finally we will give the slave host details, i.e, its private IP and method of launching agent :

The screenshot shows the Jenkins configuration page for adding a new node. The 'Launch method' is set to 'Launch agents via SSH'. The 'Host' field contains the IP address '172.31.53.48'. The 'Credentials' field contains 'ec2-user (slave)'. The 'Host Key Verification Strategy' is set to 'Known hosts file Verification Strategy'.

Launch method ?

Launch agents via SSH

Host ?

172.31.53.48

Credentials ?

ec2-user (slave)

+ Add

Host Key Verification Strategy ?

Known hosts file Verification Strategy

Keep the Host-key Verification Strategy to “Non verifying Strategy”.

For giving the “ec2-user” credentials that we can see here, we first have to create it separately on “Credentials” section of Jenkins :

And finally our setup would be look like this :

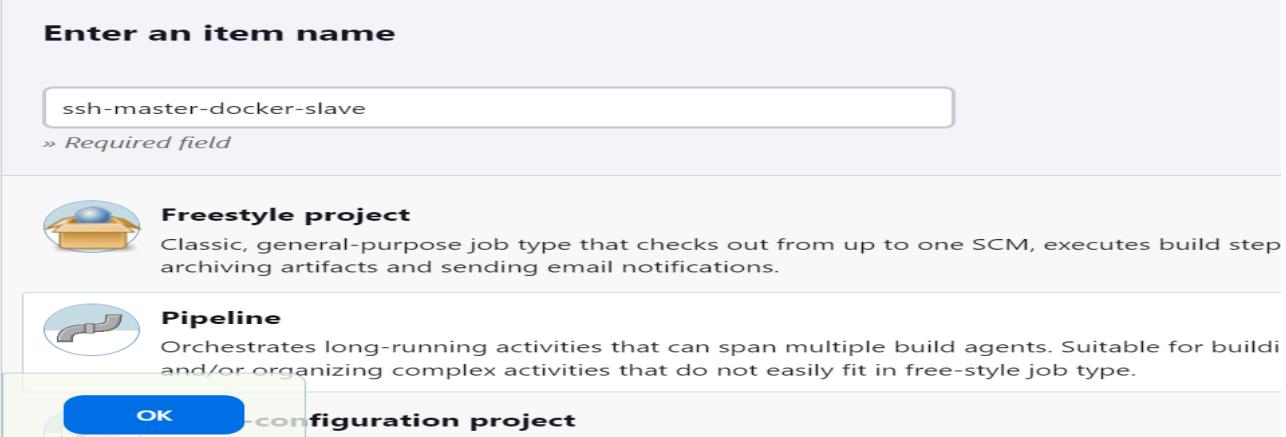
The screenshot shows the Jenkins Nodes page. At the top, there is a '+ New Node' button and a 'Configure Monitors' button. Below the buttons is a table with columns: S, Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, Free Temp Space, and Response Time. The table has two rows: 'Built-In Node' and 'Slave 1 - Docker'. The 'Built-In Node' row shows 'Linux (amd64)' architecture, 'In sync' clock difference, '5.99 GiB' free disk space, '0 B' free swap space (with a red exclamation mark icon), '1.90 GiB' free temp space (with a yellow warning icon), and '0ms' response time. The 'Slave 1 - Docker' row shows 'N/A' for all metrics except 'Response Time' which is 'N/A'. At the bottom of the table, there is a row for 'Data obtained' with a value of '31 min' for each of the last six columns.

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	5.99 GiB	0 B	1.90 GiB	0ms
	Slave 1 - Docker		N/A	N/A	N/A	N/A	N/A
Data obtained		31 min	31 min	31 min	31 min	31 min	31 min

By implementing Jenkins master-slave architecture on AWS EC2, organizations can harness the power of distributed computing to accelerate their CI/CD pipelines and deliver high-quality software with greater efficiency and reliability.

equipped to leverage Jenkins' distributed capabilities and propel your software delivery process to new heights of efficiency and agility.

■ Build the Jenkins pipe line



The screenshot shows the Jenkins 'New Item' configuration page. At the top, there is a section titled 'Enter an item name' with a text input field containing 'ssh-master-docker-slave' and a note '» Required field'. Below this, there are two job type options: 'Freestyle project' (described as a classic, general-purpose job type) and 'Pipeline' (described as orchestrating long-running activities). The 'Pipeline' option is selected and highlighted with a green box. At the bottom, there is a blue 'OK' button and a partially visible 'configuration project' label.

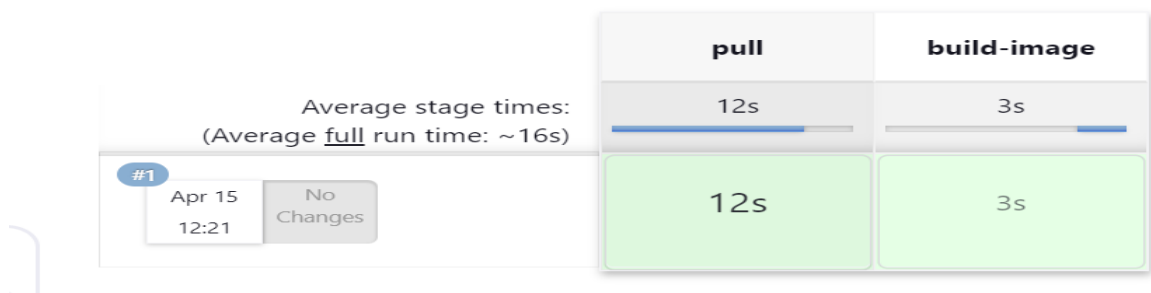
Write the pipe line script

```
pipeline {  
  
    agent {  
  
        label 'Node_For_Docker'  
  
    }  
  
    stages {  
  
        stage('pull') {  
  
            steps {  
  
                sh 'docker pull jenkins/jenkins'  
  
            }  
  
        }  
  
        stage('build-image') {  
  
            steps {  
  
                sh 'docker run -dt -p 8081:8080 jenkins/jenkins'  
  
            }  
  
        }  
  
    }  
}
```

```
}  
  
}
```

- Build the code

Stage View



- Go to slave node and check workspace, docker image, docker container created or not.

```
[ec2-user@ip-172-31-53-48 ~]$ pwd  
/home/ec2-user  
[ec2-user@ip-172-31-53-48 ~]$ ls  
caches  remoting  remoting.jar  workspace  
[ec2-user@ip-172-31-53-48 ~]$ cd workspace/  
[ec2-user@ip-172-31-53-48 workspace]$ ls  
ssh-master-docker-slave  ssh-master-docker-slave@tmp  
[ec2-user@ip-172-31-53-48 workspace]$ docker images  
REPOSITORY      TAG         IMAGE ID      CREATED        SIZE  
jenkins/jenkins latest      8e7bd57a40fa  5 days ago    470MB  
[ec2-user@ip-172-31-53-48 workspace]$ docker ps -a  
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS  
12383e99a049   jenkins/jenkins  "/usr/bin/tini -- /u..."  51 seconds ago  Up 48 seconds  50000/tcp, 0.0.0.0:8081->8080/tcp, :::8081->8080/tcp  
tcp            objective_archimedes
```

- The master and slave is completed by ssh agent method.

➔ **Master and slave by other approach (launch agent connecting it to the controller.**

Node name

slave2-by-docker-tcp

Type



Permanent Agent


Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.





Copy Existing Node


[Create](#)


Step 5: Add a description of the node.


 Status

 Delete Agent

 Configure

 Build History

 Load Statistics

 Log

Build Executor Status

Name ?

slave2-docker-by-tcp

Description ?

Plain text [Preview](#)

Number of executors ?

2

Step 6: Select the Number of executors to 2, generally equal to the number of vCPUs your machine has. It means how many parallel tasks this node can handle. If a task/ process requires one thread, then it consumes one core of your machine, which is how many vCPUs your machine has. So, we will be selecting this based on vCPUs.

Next, that workspace remote root directory is default otherwise we can create and pass here:.

Number of executors ?

2

Remote root directory ?

/home/ec2-user

Step 7: Add labels, for now, we have added docker-by-tcp-controller as the label; it is generally given as dev, stage, etc. Select Usage to Use this node as much as possible and the **Launch method to Launch the agent by connecting it to the controller.**

Labels ?

docker-by-tcp-controller

Usage ?

Use this node as much as possible

Launch method ?

Launch agent by connecting it to the controller

Availability ?

Keep this agent online as much as possible

.. . - ..

Step 10: Click on "Save".

--When you click "enter", you will see a red cross in front of our Slave node. This means our agent is not yet connected. To connect our agent with the master node,

-- we need to follow some steps.

Step-1: click on manage Jenkins → click on security → go to agents → by default it is disabled → click on fixed → give any port number (ex:5000) → then apply and save.

Dashboard > Manage Jenkins > Security

Agents

TCP port for inbound agents ?

☒ Fixed

5000

☐ Random



☐ Disable

Agent protocols ▾

CSRF Protection


Save Apply


Step-2 : Click on “Slave”.


	slave2-docker-by-tcp	N/A	N/A	N/A	N/A	N/A	
Data obtained	3.2 sec	3.2 sec	3.2 sec	3.2 sec	3.2 sec	3.2 sec	


Step-3: To connect with our master node, we need to install agent.jar on our master node. To install agent.jar, copy the command according to your environment and enter it onto your master node.


Dashboard > Nodes > slave2-docker-by-tcp


 Status

 Delete Agent

 Configure

 Build History

 Load Statistics

 Log

Agent slave2-docker-by-tcp

Mark this node temporarily offline ?

Add description

Run from agent command line: (Unix)

```
curl -sO http://54.210.123.181:8080/jnlpJars/agent.jar
java -jar agent.jar -url http://54.210.123.181:8080/ -secret 0e9e4b35621aeaa43adcc7da4b738307e4367ec5d162118f967ffa7b8075b2ef -name "slave2-docker-by-tcp" -workDir "/home/ec2-user"
```

You will see something like this –

INFO: “Connected” indicates that your master node is successfully connected with your slave node.

```
ec2-user@ip-172-31-53-48 ~]$ curl -sO http://54.210.123.181:8080/jnlpJars/agent.jar
ava -jar agent.jar -url http://54.210.123.181:8080/ -secret 0e9e4b35621aeaa43adcc7da4b738307e4367ec5d162118f967ffa8b8075b2ef -name "slave2-docker-by-tcp" -workDir "/home/ec2-user"
pr 15, 2024 7:15:31 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/ec2-user/remoting as a remoting work directory
pr 15, 2024 7:15:31 AM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to /home/ec2-user/remoting
pr 15, 2024 7:15:31 AM hudson.remoting.Launcher createEngine
INFO: Setting up agent: slave2-docker-by-tcp
pr 15, 2024 7:15:31 AM hudson.remoting.Engine startEngine
Apr 15, 2024 7:15:32 AM hudson.remoting.Launcher$CuiListener status
INFO: Trying protocol: JNLP4-connect
Apr 15, 2024 7:15:32 AM org.jenkinsci.remoting.protocol.impl.BIONetworkLayer$Reader run
INFO: Waiting for ProtocolStack to start.
Apr 15, 2024 7:15:32 AM hudson.remoting.Launcher$CuiListener status
INFO: Remote identity confirmed: d7:fb:2d:4a:4f:18:47:ab:07:5c:06:41:e8:96:43:6c
Apr 15, 2024 7:15:32 AM hudson.remoting.Launcher$CuiListener status
INFO: Connected
```

Step-4: Now, to check whether it is connected or not, refresh the slave2 node.

Status

Agent slave2-docker-by-tcp

Delete Agent

Configure

Build History

Agent is connected.

Step-5: create a pipeline

Enter an item name

» Required field

Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, archiving artifacts and sending email notifications.

Pipeline

Orchestrates long-running activities that can span multiple build agents and/or organizing complex activities that do not easily fit in free-style jobs.

Multi-configuration project

Use for projects that need a large number of different configurations.

Step-6: create a pipeline script for any sample application{httpd}

```
pipeline {
    agent {
        label 'docker-by-tcp-controller'
```

```

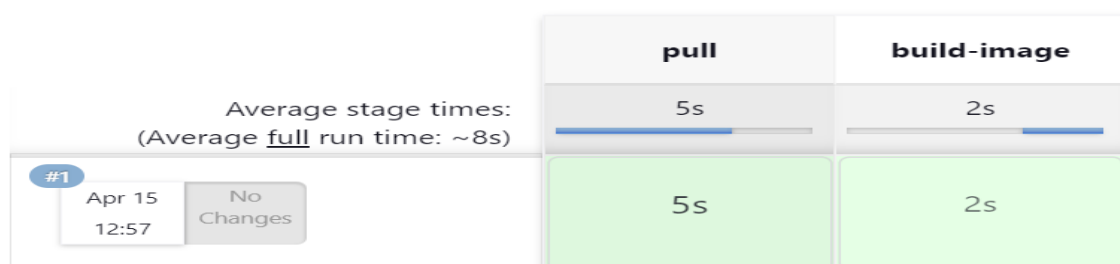
}

stages {
  stage('pull') {
    steps {
      sh 'docker pull httpd'
    }
  }
  stage('build-image') {
    steps {
      sh 'docker run -dt -p 81:80 httpd'
    }
  }
}
}

```

Step-7: build the pipe line

Stage View



Step-8 : go to slave node and check docker image and container is created or not.

```
INFO: Trying protocol: JNLP4-connect
Apr 15, 2024 7:15:32 AM org.jenkinsci.remoting.protocol.impl.BIONetworkLayer$Reader run
INFO: Waiting for ProtocolStack to start.
Apr 15, 2024 7:15:32 AM hudson.remoting.Launcher$CuiListener status
INFO: Remote identity confirmed: d7:fb:2d:4a:4f:18:47:ab:07:5c:06:41:e8:96:43:6c
Apr 15, 2024 7:15:32 AM hudson.remoting.Launcher$CuiListener status
INFO: Connected
[ec2-user@ip-172-31-53-48 ~]$
[ec2-user@ip-172-31-53-48 ~]$ ls
agent.jar  caches  remoting  remoting.jar  workspace
[ec2-user@ip-172-31-53-48 ~]$ cd workspace/
[ec2-user@ip-172-31-53-48 workspace]$ ls
docker-by-tcp-controller  docker-by-tcp-controller@tmp  ssh-master-docker-slave  ssh-master-docker-slave@tmp
[ec2-user@ip-172-31-53-48 workspace]$ docker images
REPOSITORY          TAG         IMAGE ID      CREATED      SIZE
jenkins/jenkins     latest     8e7bd57a40fa  5 days ago  470MB
httpd               latest     fa0099f1c09d  9 days ago  148MB
[ec2-user@ip-172-31-53-48 workspace]$ docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS
b0609373dd68  httpd     "httpd-foreground"      50 seconds ago Up 48 seconds  0.0.0.0:81->80/tcp, :::81->80/tcp
12383e99a049  jenkins/jenkins  "/usr/bin/tini -- /u..." 37 minutes ago Up 37 minutes  50000/tcp, 0.0.0.0:8081->8080/tcp, :::8081->8080/tcp
objective/archimedes
```

- Master and slave other approach by (launch agent connecting it to the controller. Is competed successfully