

Lab (3)

- 1- What is the benefit of using master-slave architecture rather than building on master only?

Increased performance: The slave machines can be used to execute build jobs in parallel, which can significantly improve the performance of your builds.

Improved scalability: The master-slave architecture can be scaled out by adding more slave machines. This allows you to increase the capacity of your Jenkins instance without having to make any changes to the master machine.

Increased availability: If one of the slave machines fails, the other slave machines can continue to execute build jobs. This ensures that your builds will not be interrupted even if one of the slave machines fails.

Reduced load on master machine: The master machine is responsible for scheduling build jobs and managing the slave machines. By offloading the workload of building and testing your software to the slave machines, you can reduce the load on the master machine.

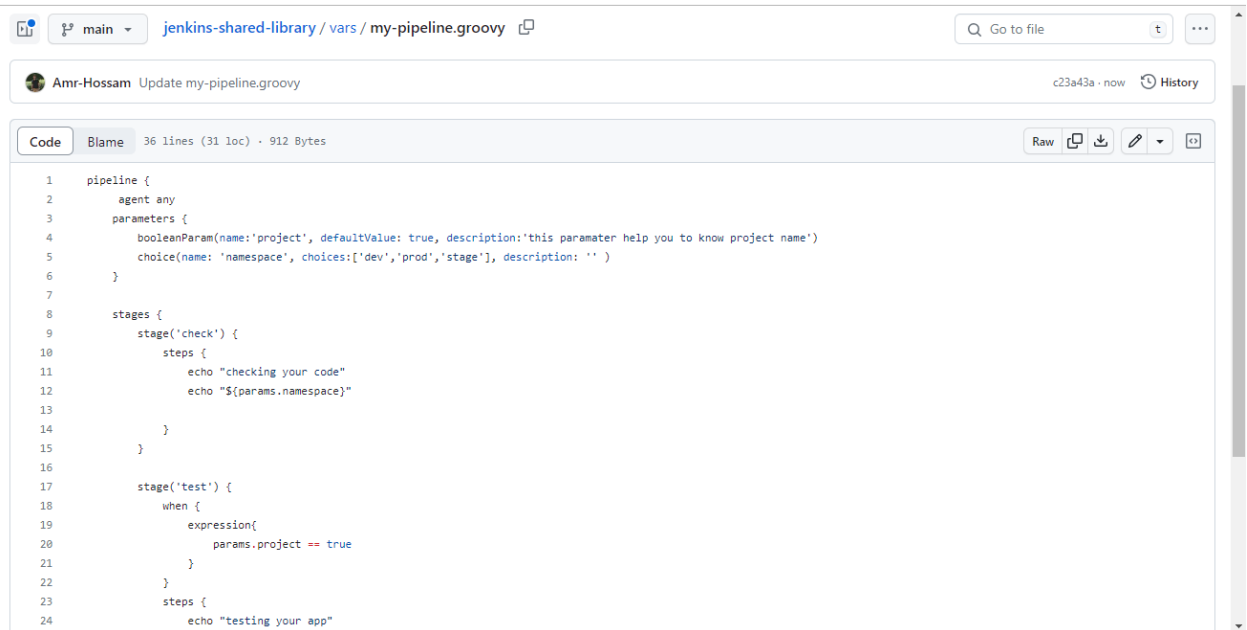
- 2- What is different between authorization and authentication?

Authentication is the process of verifying the identity of a user or system. This is typically done by asking the user to provide something they know, such as a password, or something they have, such as a security token.

Authorization is the process of determining what resources a user or system is allowed to access. This is typically done by assigning permissions to users or groups.

- 3- Make jenkins-shared-library and make your jenkinsfile which you used in lab2 to point to this library

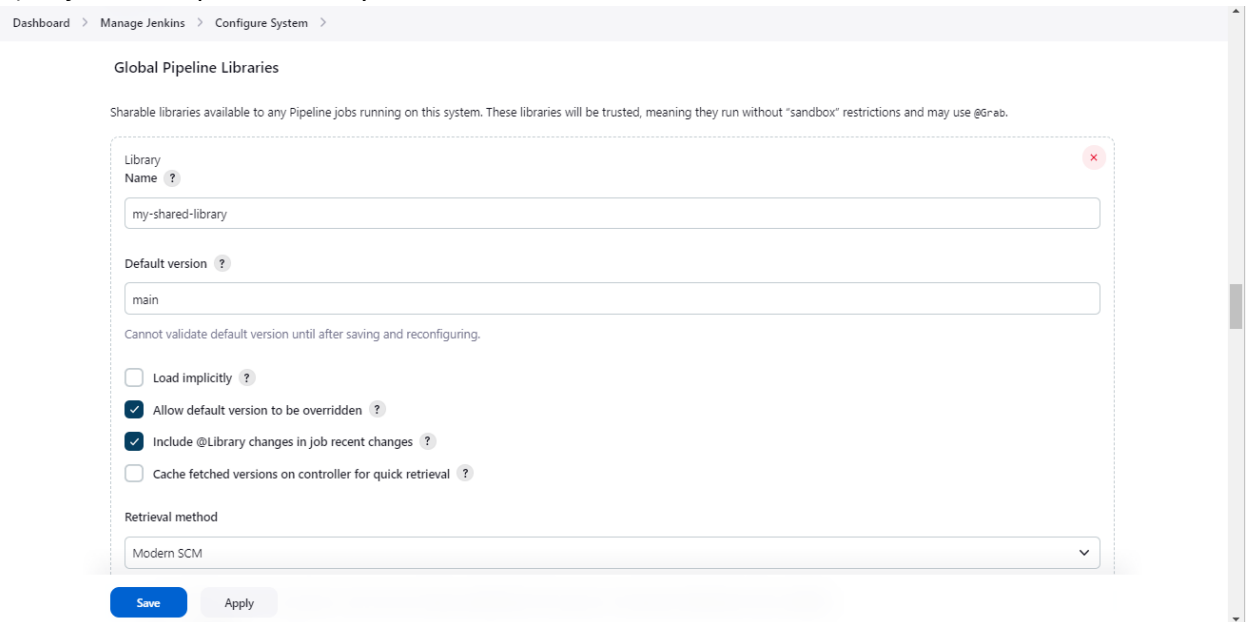
A) Create vars/file.groovy



The screenshot shows the Jenkins web interface for editing a file named `my-pipeline.groovy` within the `jenkins-shared-library / vars` directory. The file is 36 lines long, 31 lines of code, and 912 bytes. The code defines a Jenkins pipeline with two stages: 'check' and 'test'. The 'check' stage has a step that echoes the project name and namespace. The 'test' stage is conditional on the project being 'prod' and has a step that echoes 'testing your app'.

```
1 pipeline {
2   agent any
3   parameters {
4     booleanParam(name: 'project', defaultValue: true, description: 'this parameter help you to know project name')
5     choice(name: 'namespace', choices: ['dev', 'prod', 'stage'], description: '')
6   }
7
8   stages {
9     stage('check') {
10      steps {
11        echo "checking your code"
12        echo "${params.namespace}"
13      }
14    }
15  }
16
17  stage('test') {
18    when {
19      expression {
20        params.project == true
21      }
22    }
23    steps {
24      echo "testing your app"
```

B) Adjust the Pipeline Library



The screenshot shows the Jenkins 'Global Pipeline Libraries' configuration page. The page title is 'Global Pipeline Libraries'. Below the title, there is a description: 'Sharable libraries available to any Pipeline jobs running on this system. These libraries will be trusted, meaning they run without "sandbox" restrictions and may use @Grab.' The configuration form includes a 'Library Name' field with the value 'my-shared-library', a 'Default version' field with the value 'main', and a checkbox 'Load implicitly' which is unchecked. There are also checkboxes for 'Allow default version to be overridden' (checked) and 'Include @Library changes in job recent changes' (checked). A 'Retrieval method' dropdown menu is set to 'Modern SCM'. At the bottom, there are 'Save' and 'Apply' buttons.

Dashboard > Manage Jenkins > Configure System >

Global Pipeline Libraries

Sharable libraries available to any Pipeline jobs running on this system. These libraries will be trusted, meaning they run without "sandbox" restrictions and may use @Grab.

Library Name ?

Default version ?

Cannot validate default version until after saving and reconfiguring.

☐ Load implicitly ?

☒ Allow default version to be overridden ?

☒ Include @Library changes in job recent changes ?

☐ Cache fetched versions on controller for quick retrieval ?

Retrieval method

Dashboard > Manage Jenkins > Configure System >

Retrieval method

Modern SCM

Loads a library from an SCM plugin using newer interfaces optimized for this purpose. The recommended option when available.

Source Code Management

Git

Project Repository ?

`https://github.com/Amr-Hossam/jenkins-shared-library`

Credentials ?

Amr-Hossam/***** (jenkins-cred)

Add

Behaviors

Discover branches ?

Save Apply

C) Now adjust the Jenkinsfile

Amr-Hossam / jenkins-shared-library

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

jenkins-shared-library / Jenkinsfile

Amr-Hossam Create Jenkinsfile b186e62 · 9 minutes ago History

Code Blame 1 lines (1 loc) · 32 Bytes

```
1 @Library('my-shared-library') _
```

D) Create the Pipeline

Dashboard > my-shared-library-pipline > Configuration

Configure

General Advanced Project Options Pipeline

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

`https://github.com/Amr-Hossam/jenkins-shared-library`

Credentials ?

Amr-Hossam/***** (jenkins-cred)

Add

Advanced

Add Repository

Save Apply

E) Output and logs

Dashboard > my-shared-library-pipeline >

Status

Changes

Build Now

Configure

Delete Pipeline

Full Stage View

Rename

Pipeline Syntax

Build History

trend

Filter builds...

#1 Jul 6, 2023, 3:54 AM

Atom feed for all Atom feed for failures

Pipeline my-shared-library-pipeline

Add description

Disable Project

Stage View

Average stage times:
(Average full run time: ~8s)

Declarative: Checkout SCM	check	test	deployment
1s	773ms	714ms	861ms
1s	773ms	714ms	861ms

Permalinks

Checking out Revision 24165c98f4c554b7ab2ab3bca246d84f42298b77 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=18
> git checkout -f 24165c98f4c554b7ab2ab3bca246d84f42298b77 # timeout=18
Commit message: "Delete text.txt"
First time build. Skipping changelog.
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (check)
[Pipeline] echo
checking your code
[Pipeline] echo
dev
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (test)
[Pipeline] echo
testing your app
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (deployment)
[Pipeline] echo
your code is deployed right now
[Pipeline] echo
this build number 1
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

- 4- Try to make new slave as container or ec2 server and configure master to use it

A) Create ec2

The screenshot shows the AWS Management Console for the 'us-east-1' region. The left sidebar contains navigation links for 'New EC2 Experience', 'EC2 Dashboard', 'EC2 Global View', 'Events', 'Instances', 'Instance Types', 'Launch Templates', 'Spot Requests', 'Savings Plans', 'Reserved Instances', 'Dedicated Hosts', 'Scheduled Instances', 'Capacity Reservations', 'Images', 'AMIs', and 'AMI Catalog'. The main content area displays a list of instances with the following table:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availab
ec2-agent	i-07f6c2ebe5531258d	Running	t2.micro	2/2 checks passed	No alarms	us-east-

Below the table, the details for the selected instance 'i-07f6c2ebe5531258d (ec2-agent)' are shown. The 'Instance summary' section includes the following information:

- Instance ID: i-07f6c2ebe5531258d (ec2-agent)
- Public IPv4 address: 54.196.133.137 | [open address](#)
- Private IPv4 addresses: 10.0.0.67
- IPv6 address: -
- Instance state: Running
- Public IPv4 DNS: -
- Hostname type: -
- Private IP DNS name (IPv4 only): -

B) Create Public and Private Key at Jenkins container

```
amr@ubuntu-SV: ~  
amr@ubuntu-SV: ~  
amr@ubuntu-SV:~$ docker exec -it -u 0 4db6a89e6eae bash  
root@4db6a89e6eae:/# ssh-keygen  
Generating public/private rsa key pair.  
Enter file in which to save the key (/root/.ssh/id_rsa):  
Created directory '/root/.ssh'.  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /root/.ssh/id_rsa  
Your public key has been saved in /root/.ssh/id_rsa.pub  
The key fingerprint is:  
SHA256:PTvqgi9RFta04T+iKaqzFZ2s59Fp21+iL4I3oHZRGg4 root@4db6a89e6eae  
The key's randomart image is:  
+----[RSA 3072]-----+  
|      oo      |  
|    . O..O    |  
| . + .O       |  
| OE.+ o      |  
| ..++ S =     |  
| O+. + . +   |  
| O.*** + .   |  
| ..O+=+=O.O + |  
| O=...O==+O  |  
+----[SHA256]-----+  
root@4db6a89e6eae:/# cd ~/.ssh/  
root@4db6a89e6eae:~/.ssh# ls  
id_rsa id_rsa.pub  
root@4db6a89e6eae:~/.ssh#
```

C) Copy the Jenkins' public Key to the ec2 agent

```
amr@ubuntu-SV: ~  
ubuntu@ip-10-0-0-67: ~$ cd ~/.ssh/  
ubuntu@ip-10-0-0-67: ~/.ssh$ sudo vim authorized_keys  
ubuntu@ip-10-0-0-67: ~/.ssh$  
  
root@4db6a89e6eae: ~/.ssh#
```

D) Now we can access the ec2 from Jenkins container

```
ubuntu@ip-10-0-0-67: ~  
root@4db6a89e6eae: ~/.ssh# ssh ubuntu@54.196.133.137  
The authenticity of host '54.196.133.137 (54.196.133.137)' can't be established.  
ECDSA key fingerprint is SHA256:/C+Y80R6UGAnEX6Wa6Grbt+yM5d1yti1cAPWqUm3SQk.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '54.196.133.137' (ECDSA) to the list of known hosts.  
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-1025-aws x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/advantage  
  
System information as of Thu Jul  6 05:25:59 UTC 2023  
  
System load:  0.0      Processes:            100  
Usage of /:   20.8% of 7.57GB   Users logged in:     1  
Memory usage: 26%      IPv4 address for eth0: 10.0.0.67  
Swap usage:   0%  
  
Expanded Security Maintenance for Applications is not enabled.  
  
0 updates can be applied immediately.  
  
Enable ESM Apps to receive additional future security updates.  
See https://ubuntu.com/esm or run: sudo pro status  
  
ages      IPv6 address      Instance state
```

E) Create Jenkins home on the agent

```
ubuntu@ip-10-0-0-67: ~  
ubuntu@ip-10-0-0-67:~$ mkdir -p jenkins_home  
ubuntu@ip-10-0-0-67:~$ cd jenkins_home/  
ubuntu@ip-10-0-0-67:~/jenkins_home$ pwd  
/home/ubuntu/jenkins_home  
ubuntu@ip-10-0-0-67:~/jenkins_home$
```

F) Create Agent node (ec2-agent)

The screenshot shows the Jenkins 'Manage Jenkins > Nodes' page. A new agent node named 'ec2-agent' is being configured. The fields are as follows:

- Name:** ec2-agent
- Description:** ec2-agent
- Number of executors:** 1
- Remote root directory:** /home/ubuntu/jenkins_home
- Labels:** ec2-agent

A 'Save' button is visible at the bottom of the configuration form.

G) Create SSH cred.

The screenshot shows the Jenkins 'Manage Jenkins > Nodes' page with a modal window titled 'Jenkins Credentials Provider: Jenkins' open. The modal is for 'Add Credentials' and contains the following fields:

- Domain:** my-domain
- Kind:** SSH Username with private key
- Scope:** Global (Jenkins, nodes, items, all child items, etc)
- ID:** ec2-private-key
- Description:** ec2-private-key

A 'Save' button is visible at the bottom of the modal.

H) Creating agent node

[Dashboard](#) > [Manage Jenkins](#) > [Nodes](#) >

Usage ?

Only build jobs with label expressions matching this node

Launch method ?

Launch agents via SSH

Host ?

54.196.133.137

Credentials ?

ubuntu (ec2-private-key)

Add

Host Key Verification Strategy ?

Non verifying Verification Strategy

Advanced

Availability ?

Keep this agent online as much as possible

Save

I) Agent successfully connected and online

[Dashboard](#) > [Manage Jenkins](#) > [Nodes](#) > [ec2-agent](#) > [Log](#)


```
Couldn't figure out the Java version of /home/ubuntu/jenkins_home/jdk/bin/java
bash: line 1: /home/ubuntu/jenkins_home/jdk/bin/java: No such file or directory





[07/06/23 05:48:39] [SSH] Checking java version of java
[07/06/23 05:48:39] [SSH] java -version returned 11.0.19.
[07/06/23 05:48:39] [SSH] Starting sftp client.
[07/06/23 05:48:40] [SSH] Copying latest remoting.jar...
[07/06/23 05:48:51] [SSH] Copied 1,370,647 bytes.
Expanded the channel window size to 4MB
[07/06/23 05:48:52] [SSH] Starting agent process: cd "/home/ubuntu/jenkins_home" && java -jar remoting.jar -workDir /home/ubuntu/jenkins_home
-jar-cache /home/ubuntu/jenkins_home/remoting/jarCache
Jul 06, 2023 5:48:52 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/ubuntu/jenkins_home/remoting as a remoting work directory
Jul 06, 2023 5:48:52 AM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to /home/ubuntu/jenkins_home/remoting
<===[JENKINS REMOTING CAPACITY]==>channel started
Remoting version: 3107.v665000b_51092
Launcher: SSHLauncher
Communication Protocol: Standard in/out
This is a Unix agent
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by jenkins.slaves.StandardOutputSwapper$ChannelSwapper to constructor java.io.FileDescriptor(int)
WARNING: Please consider reporting this to the maintainers of jenkins.slaves.StandardOutputSwapper$ChannelSwapper
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
Evacuated stdout
Agent successfully connected and online
```

REST API


Jenkins 2.387.3


G) The Agent Node is ready

 **Jenkins**

   Amr Hossam  log out

[Dashboard](#) > [Manage Jenkins](#) > [Nodes](#) >

 Configure Clouds

 Node Monitoring

Build Queue

No builds in the queue.

Build Executor Status


Built-In Node




1 Idle

ec2-agent

2 Idle

Nodes


[+ New Node](#) 


S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	401.86 GB	2.65 GB	401.86 GB	0ms
	ec2-agent	Linux (amd64)	4.1 sec ahead	4.96 GB	 0 B	4.96 GB	7639ms
	last checked	1 min 34 sec	1 min 35 sec	1 min 23 sec	1 min 29 sec	1 min 23 sec	1 min 36 sec


[REST API](#) [Jenkins 2.387.3](#)

K) Create Sample pipeline with ec2-agent as executer

[Dashboard](#) > [my-agent-pipeline](#) > [Configuration](#)

 General

 Advanced Project Options

 Pipeline

Configure

Pipeline

Definition

Pipeline script

Script

```
1 pipeline {
2   agent { label 'ec2-agent' }
3
4   stages {
5     stage('Build') {
6       steps {
7         echo 'ec2-agent is executing your pipeline!'
8       }
9     }
10  }
11 }
12
13
```

try sample Pipeline...

☒ Use Groovy Sandbox


[Pipeline Syntax](#)


Save





Apply

[REST API](#) [Jenkins 2.387.3](#)


L) Console Output log


 **Jenkins**


Search (CTRL+K) 


 1  2  Amr Hossam  log out


Dashboard > my-agent-pipeline > #1


 Status


 Changes


 **Console Output**


 View as plain text


 Edit Build Information


 Delete build '#1'

 Restart from Stage

 Replay

 Pipeline Steps

 Workspaces

 **Console Output**

Started by user [Amr Hossam](#)
[Pipeline] Start of Pipeline
[Pipeline] node
Running on [ec2-agent](#) in /home/ubuntu/jenkins_home/workspace/my-agent-pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] echo
ec2-agent is executing your pipeline
[Pipeline] }
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

REST API Jenkins 2.387.3