

Jenkins master slave poc

Use case:

1. In our current architecture, we're using multiple jenkins instances to build and deploy in multiple AWS accounts and multiple environment.
2. We're almost running 4-5 higher performance instances for the jenkins servers and the costs involved in running these instances is also higher.
3. These jenkins instances are running 24*7 whenever the jobs are running or the jenkins setup is idle without running any job.

Requirement:

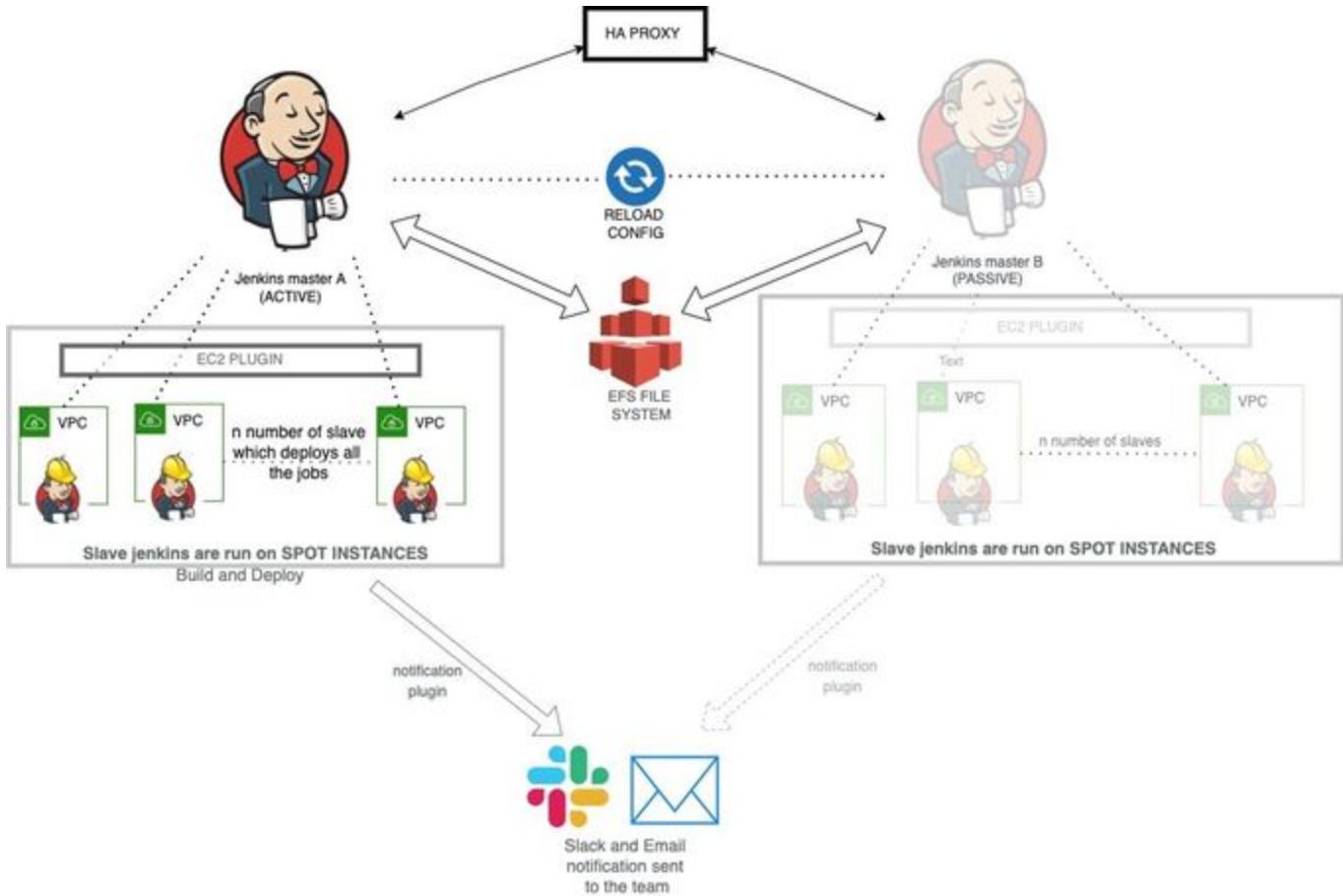
1. Making use of a single jenkins instance which can be used to run jobs in multiple AWS accounts.
2. To make sure we run on instances which has good performance and at the same time is stable.
3. Reducing the Costs involved in the jenkins setup.

Solution:

1. Using a Jenkins master node in one of the AWS accounts and connecting to multiple AWS accounts using the **EC2 plugin** and run the jobs in those particular accounts.
2. We'll be setting up 2 master jenkins (active and passive) with **HA proxy** setup which are high performance instances (t3a.large) and has a stability as well as a backup jenkins job.
3. We'll be using **SPOT instances** to run the jobs in multiple AWS accounts , and these instances will be up and running only when there's a job running which will terminate when automatically after a amount of time the jenkins job is idle.

How to Setup a Jenkins Master and Slave Architecture.

JENKINS-MASTER SLAVE ARCHITECTURE



Jenkins is an essential tool to run CI/CD pipelines for your projects.

However, if you have more and more pipelines running in single Jenkins, you may encounter some problems in availability and scalability.

Given that one Jenkins can cause a single point of failure problem and longer building time when executing the increasing number of CI/CD pipelines

STEPS FOLLOWED FOR THE IMPLEMENTATION:

1. Create a Jenkins master node.
2. Creating a AMI for slave node with dependencies.
3. Setup ec2 plugin for creating slave ec2 instance.
4. Jenkins job to test the Master-Slave working.
5. High Availability of the Master Jenkins.
6. Setup of reload configuration, HA proxy.
7. Setup of slack notification and email notification.

STEP 1: Setup of the Jenkins Master :

1. Login into your AWS console and create an ec2 instance .
 - a. Login into the instance and install java JDK and jenkins on it.

```

sudo apt-get update;          -- updating the package in the
ubuntu machine
sudo apt-get install openjdk-11-jdk;    -- install java jdk

java -version;      -- checking the java version

```

- b. Allow the security groups to enable access to port 22 for ssh and port 8080 for jenkins.

Security group rule ID	Type	Protocol	Port range	Source
	Info	Info	Info	Info
sgr-048d76499bf80e257	SSH	TCP	22	Custom 0.0.0.0/0 X
sgr-0daab860f89b57a22	Custom TCP	TCP	8080	Custom ::/0 X
sgr-05c670022922b35ea	Custom TCP	TCP	8080	Custom 0.0.0.0/0 X

- c. install the Jenkins service.

```
-- (add the repository key to the system)
wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -

-- (append the Debian package repository)
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable
binary/ > /etc/apt/sources.list.d/jenkins.list'

sudo apt update;

sudo apt install jenkins; -- install jenkins

sudo systemctl start jenkins; --start jenkins server

sudo systemctl status jenkins; -- status of the jenkins server
```

```
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; disabled; vendor preset: disabled)
   Active: active (running) since Fri 2022-06-10 09:33:26 UTC; 2 days ago
     Main PID: 12010 (java)
       CGroup: /system.slice/jenkins.service
               └─12010 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/jav...

Jun 13 07:35:14 ip-172-31-10-151.ap-south-1.compute.internal jenkins[12010]: ...
Hint: Some lines were ellipsized, use -l to show in full.
```

2. Open the jenkins home page using the ip address of the instance. <http://your instance ip address:8080>
- Unlock your jenkins

Getting Started

Unlock Jenkins

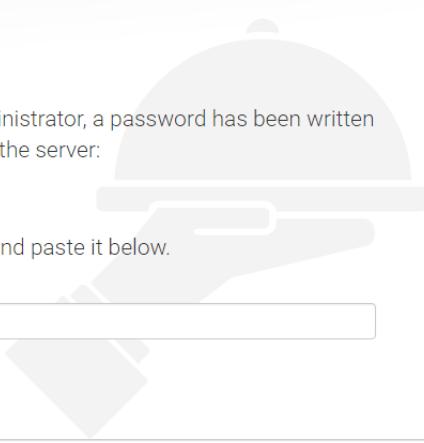
To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

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

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

Administrator password

Continue



- b. you'll find the admin password in the instance path or use this command below.

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

- c. Next will be to install the suggested plugins .

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.

- d. Create your Admin user:

Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

e. Then your jenkins is ready to work.

STEP 2: Creating a AMI for slave node.

1. In the Aws ec2 console, create a ubuntu machine with 20.04 version.
2. Install java latest version.

```
sudo apt-get update;      -- updating the package in the ubuntu  
machine  
sudo apt-get install openjdk-11-jdk;      -- install java jdk  
  
java -version;      -- checking the java version
```

3. Install sonarqube.

```
wget https://binaries.sonarsource.com/Distribution/sonar-scanner-  
cli/sonar-scanner-cli-4.7.0.2747-linux.zip  
unzip sonar-scanner-cli-4.7.0.2747-linux.zip  
mv sonar-scanner-4.7.0.2747-linux/ sonar-scanner
```

4. Go to the AWS ec2 console. and select the above instance click on Actions Image and template Create image.

The screenshot shows the AWS EC2 Instances page. A context menu is open for an instance named "OL-Jenkins-Master". The menu items are: Connect, View details, Manage instance state, Instance settings, Networking, Security, Create image, Create template from instance, and Launch more like this.

- Enter the image name and the description. and Click on create image.

Create image Info

An image (also referred to as an AMI) defines the programs and settings that are applied when you launch an EC2 instance. You can create an image from the configuration of an existing instance.

Instance ID	<input checked="" type="checkbox"/> i-01bbb666d4d1d5488 (OL-Jenkins)
Image name	<input type="text"/>
Image description - optional	<input type="text"/>
No reboot	<input type="checkbox"/> Enable

Instance volumes

Volume type	Device	Snapshot	Size	Volume type	IOPS	Throughput	Delete on termination	Encrypted
EBS	/dev/x...	Create new snapshot fr...	20	EBS General Purpose SS...	3000		<input checked="" type="checkbox"/> Enable	<input type="checkbox"/> Enable

Tags - optional

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Tag image and snapshots together
Tag the image and the snapshots with the same tag.

Tag image and snapshots separately
Tag the image and the snapshots with different tags.

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

Create image

- Now you'll see the AMI created.

Amazon Machine Images (AMIs) (3) Info

Owned by me	Name	AMI ID	AMI name	Source	Owner
<input type="checkbox"/>	OL-Jenkins-Master	ami-0ad5d6ac41a29db95	OL-Jenkins-Master	458146071567/OL-Jenkins-Master	458146071567

7. Now we need to create a security group for the slave which has inbound access only from the jenkins master instance on port 22 (for ssh) and port 8080 (for jenkins) . So here we'll need to add the IP of the jenkins master for both these ports to access.

Security group rule ID	Type	Protocol	Port range	Source
sgr-09a0580e791cee2e5	Custom TCP	TCP	8080	Custom 3.6.160.169/32
sgr-0cf97c39303ac89f1	SSH	TCP	22	Custom 3.6.160.169/32

Add rule

STEP 3: Setup ec2 plugin in jenkins to spin up slave instance.

To setup an ec2 plugin to spin up slave instances we need a Aws user to create slave instance and attach a role which supports the ec2 services.

1. Create a programmatic access user with an access key and secret key. Name the instance as Jenkins-user.
2. Role to be attached to the user

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:*",
        "iam>ListInstanceProfiles",
        "iam>ListRoles",
        "iam>PassRole"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

Once the user and the role is adding we'll login into Jenkins to connect with this user created to check if we can spin up the jenkins-slave instance.

3. Login into Jenkins and go to manage plugin.



Jenkins

Dashboard >

+ New Item

People

Build History

Project Relationship

Check File Fingerprint

Manage Jenkins

My Views

New View

4. In the available tab , click on ec2 plugin download and install the ec2 plugin and restart jenkins once the installation is done.
5. Now go to jenkins Manage Jenkins Manage cloud and node Configure cloud.

↑ Back to Dashboard

 Manage Jenkins

+

New Node

 Configure Clouds

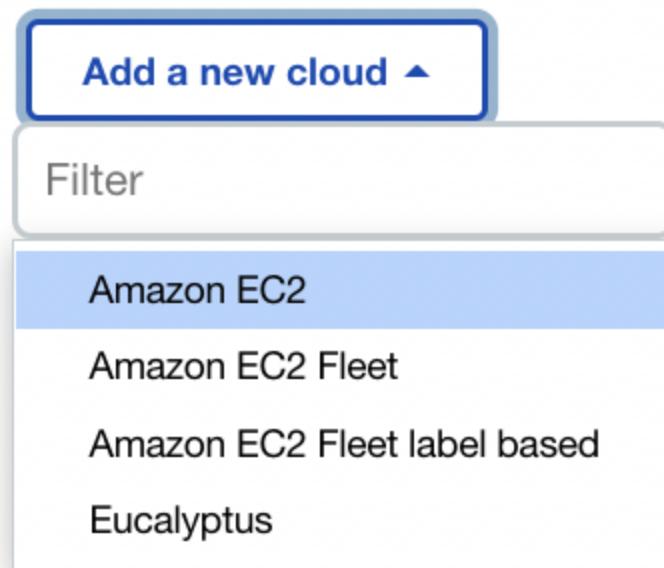
 Node Monitoring

Build Queue



6. Under configure cloud click on Amazon Ec2.

Configure Clouds



7. Specify a name for the cloud: ex- Ol-jenkins-slave

Amazon EC2

Name

No name is specified

Amazon EC2 Credentials ?

AWS IAM Access Key used to connect to EC2. If not specified, implicit authentication mechanisms are used (IAM roles...)

- none -

+ Add

Use EC2 instance profile to obtain credentials ?

Alternate EC2 Endpoint

Used to populate the available regions dropdown. Only set this if you're using a different EC2 endpoint (i.e. operating in govcloud).

The regions will be populated once the keys above are entered.

Region ?

EC2 Key Pair's Private Key ?

- none -

+ Add

No ssh credentials selected

Advanced...

- a. For amazon ec2 credentials click on ADD.

Add Credentials

Domain

Global credentials (unrestricted)

Kind

AWS Credentials

Scope ?

System (Jenkins and nodes only)

ID ?

Description ?

Access Key ID ?

Secret Access Key

- b. domain: global credentials
c. kind: AWS Credentials
d. Scope: system (jenkins and nodes only)
e. Create a Id for it: ol-jenkins-slave
f. Add the Access key Id and the Secret Access key of the user created.
g. Click on Add.
8. select the key added and select the region where we wanna spin up the ec2-instance.

- a. region: ap-south-1
9. Add the ec2-key pair private key (This key is added in order to run the jenkins job in the slave instance)

The screenshot shows the Jenkins configuration interface for adding a new credential. The 'Kind' dropdown is set to 'SSH Username with private key'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'ID' field is empty. The 'Description' field is empty. The 'Username' field contains 'ubuntu'. The 'Treat username as secret' checkbox is unchecked. Under 'Private Key', the 'Enter directly' radio button is selected, and the 'Key' input field shows 'No Stored Value'. A blue 'Add' button is visible to the right of the key field. The 'Passphrase' field is empty.

- a. domain: global credentials
 b. Kind: SSH username with private key
 c. username: ubuntu (for ubuntu instance)(ec2-user for linux instance)
 d. Private key: enter the private key directly
 e. Click on add.

10. Now select the ubuntu user and click on test-connection.

The screenshot shows the Jenkins configuration interface for selecting an EC2 key pair. The 'EC2 Key Pair's Private Key' dropdown is set to 'ubuntu'. Below it is a '+ Add' button and an 'Advanced...' button. At the bottom right is a blue 'Test Connection' button.

11. Select the AMI name of the jenkins which was created in STEP 2. Then give a description to the AMI and click on check AMI and you'll be able to see the AWS account id and the tag name of the AMI created (**accountid/ami-tag-name**).

AMIs

List of AMIs to be launched as agents

The screenshot shows the Jenkins configuration interface for selecting an AMI. It includes fields for 'Description' (set to 'master-slave-jenkins'), 'AMI ID' (set to 'ami-0e801944c4743eebf'), and a preview of the AMI identifier '458146071567/jenkins-test'. At the bottom right is a blue 'Check AMI' button.

12. Select the required instance type based on the number of jenkins jobs running. in our case we're using a t3a.micro instance for the POC instance.
 13. Select the security group which was created for which the instance was created and name the security group.
 14. Under the **Labels:**

Labels ?

Usage ?

Idle termination time ?

Init script ?

 Advanced...

 Add

Add a new cloud ▾

Please mention a label for this particular cloud configuration created, since this label will be used for

- a. When a jenkins deployment project is created we attach the label below to use this particular Ec2 instance type configuration.
- b. Running jenkins on multiple Aws account can be done with the help of the labels.

 **Built-In Node**

Linux (amd64)

Provision via b2c ▾

Provision via ol ▾

last checked

36 sec

- c. **idle termination time:** it determine the time (in minutes) for how long the particular slave instance will be available or idle without performing any jobs after which the instance will get terminated automatically.

15. Click on save and Apply.

We can create a similar configure cloud for Another AWS account and follow the above steps and give a suitable label to identify both the AWS accounts in which the jenkins master will spin up slave instances to run the jenkins job.

STEP 4: JENKINS JOB TO TEST THE MASTER-SLAVE WORKING:

1. In the Jenkins dashboard, click on a new item select Freestyle project enter a name for the project Click ok.

The screenshot shows the Jenkins dashboard with a search bar and a navigation menu. A modal dialog box is open, prompting the user to "Enter an item name". Below the input field, a note says "» Required field". Underneath the input field, there is a section titled "Freestyle project" with a brief description: "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."

2. In the configuration page

- Add label of the cloud configure which we have configured for the ec2 plugin.

The screenshot shows the "General" configuration page for a Jenkins job. The top navigation bar includes tabs for "General", "Source Code Management", "Build Triggers", "Build Environment", "Build", and "Post-build Actions".

The "General" tab contains the following fields:

- Description:** An empty text area with "[Plain text] Preview" links.
- Discard old builds:** An unchecked checkbox.
- GitHub project:** An unchecked checkbox.
- Rebuild options:** A group of checkboxes:
 - Rebuild Without Asking For Parameters
 - Disable Rebuilding for this job
 - This project is parameterised
 - Throttle builds
 - Disable this project
 - Execute concurrent builds if necessary
 - Restrict where this project can be run** (this option is checked)
- Label Expression:** A text input field containing "ol-slave-jenkins". Below it, a note states: "Label **ol-slave-jenkins** matches no nodes and 1 cloud. Permissions or other restrictions provided by plugins may further reduce that list."
- Advanced...** A link to more configuration options.

- Add a build step execute shell (adding basic commands when the job logs into the system)

Build

Execute shell ?

Command

See the [list of available environment variables](#)

```
pwd
df -h
hostname
```

Advanced...

Add build step ▾

Post-build Actions

Add post-build action ▾

Save Apply

- c. Entering the basic shell command to test. and Click on Save Apply.
- d. Now when we navigate to the console output of the jenkins job you can see that new instance has been spun up and the basic execute shell command outputs are visible at the console output.

```
Started by user OL-Jenkins
Running as SYSTEM
Building remotely on EC2 (ol) - master-slave-jenkins (i-037878f7ab8e06ccc) (ol-slave-jenkins) in workspace /home/ec2-user/workspace/Test-Ol-Jenkins
[Test-Ol-Jenkins] $ /bin/sh -xe /tmp/jenkins4618402285426130757.sh
+ pwd
/home/ec2-user/workspace/Test-Ol-Jenkins
+ df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        474M    0  474M   0% /dev
tmpfs          483M    0  483M   0% /dev/shm
tmpfs          483M   60K  483M   1% /run
tmpfs          483M    0  483M   0% /sys/fs/cgroup
/dev/xvda1     8.0G  1.9G  6.2G  23% /
tmpfs          97M    0   97M   0% /run/user/0
tmpfs          97M    0   97M   0% /run/user/1000
+ hostname
ip-172-31-12-73.ap-south-1.compute.internal
```

- e. And we can see the instance initiated and terminated as well after the **idle termination time is set to 5 minutes**.

Name	Instance ID	Instance state	Instance type	Launch time	Instance lifecycle
-	i-037878f7ab8e06ccc	Terminated	t2.micro	2022/06/13 15:27 GMT+5:30	spot

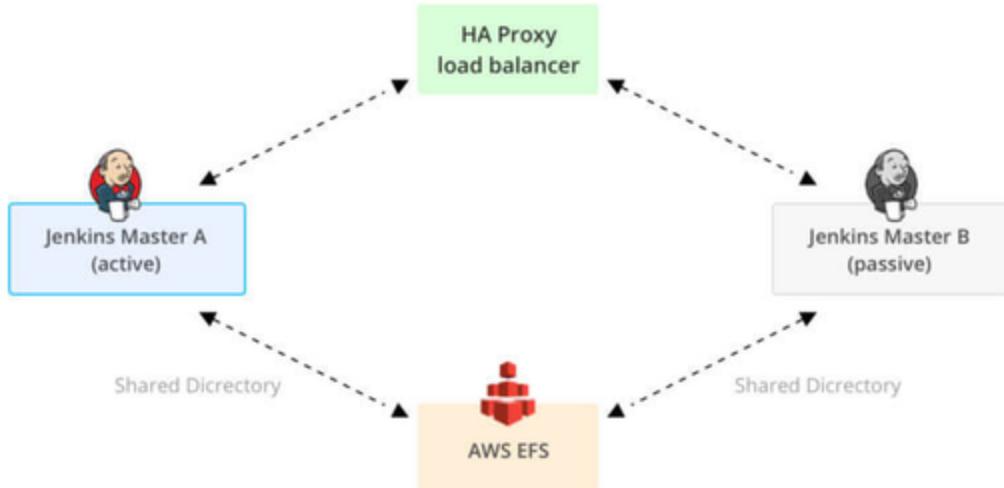
STEP 5: High Availability of the Master Jenkins.

What is High Availability (HA)?

When we say high availability, we are referring to systems that can operate continuously without failure for a long time. The term implies that the system has been tested thoroughly to stand any sort of failure. Jenkins is a crucial component of DevOps and its downtime may have adverse effects on the DevOps environment. To overcome these, we need a high availability setup for Jenkins.

What is Elastic File System (EFS)?

The EFS service on AWS is similar to the NFS service on Unix systems wherein the mount point is created on network topology and can be shared with multiple systems at the same time. We are going to configure EFS with Jenkins master machines so that both masters can store and access Jenkins jobs from EFS.



Note: If Jenkins Master A goes down, Jenkins Master B will become active

We need to create another Jenkins-Master-B i.e. passive node (follow Step 1:) or you can create a AMI of the Jenkins-master-A and launch a new instance.

We need to set up the EFS provisioning before configuring Jenkins. Here is how you can provision EFS on

1. Navigate to Amazon EFS from the Aws management console.
2. click on create File system and enter the following details
 - a. Name of the file system: jenkins-High-Availability.
 - b. Select the VPC ID (same VPC in which the master node is present)
 - c. Select **Regional** in Availability and durability.
 - d. Click on create

Create file system

Create an EFS file system with service recommended settings. [Learn more](#)

Name - optional
Name your file system.

Optional. Apply a name to your file system

Name must not be longer than 256 characters, and must only contain letters, numbers, and these characters: + - = . _ : /

Virtual Private Cloud (VPC)
Choose the VPC where you want EC2 instances to connect to your file system. [Learn more](#)

vpc-3d957d54 ▾
default

Availability and durability
Choose Regional (recommended) to create a file system using regional storage classes. Choose One Zone to create a file system using One Zone storage classes. [Learn more](#)

Regional
Stores data redundantly across multiple AZs

One Zone
Stores data redundantly within a single AZ

Cancel **Customize** **Create**

3. After the EFS is created copy paste the DNS name of the EFS file system.

ol-jenkins-High-Availability (fs-096ad6367ccbf04be)

General	
Performance mode	Automatic backups
General Purpose	<input checked="" type="checkbox"/> Enabled
Throughput mode	Encrypted
Bursting	f1029a89-0fc3-43dd-a3f0-bbe565b73eef (aws/elasticfilesystem)
Lifecycle management	File system state
Transition into IA: 30 days since last access	<input checked="" type="checkbox"/> Available
Transition out of IA: On first access	DNS name
Availability zone	<input type="text"/> fs-096ad6367ccbf04be.efs.ap-south-1.amazonaws.com
Regional	

Once EFS is created, the next step will be to configure Jenkins instances to use the provisioned EFS. The configuration steps are pretty simple, all we need to do is ssh both Jenkins instances, i.e Jenkins A (Active) and Jenkins B (Passive) and add the entry given below in the /etc/fstab file.

```
<region>.fs-******.efs.us-east-2.amazonaws.com: /var/lib/jenkins/ nfs
defaults    1      1
```

fstab is a system configuration file on Linux and other Unix-like operating systems that contains information about major filesystems. It takes its name from the file systems table and it is located in the /etc directory.

Note :

- The first field is the DNS name with “root (/)” of EFS share.
- The second field is the path to be mounted **/var/lib/jenkins/**.
- The third field is a type of filesystem.

Once you complete the above changes in the fstab file, perform the following steps.

- Stop Jenkins Service using the command,

```
systemctl stop jenkins
```

- Mount EFS volume using the command,

```
mount -a
```

- Validate if the EFS is mounted,

```
mount: shows if /var/lib/jenkins/jobs mounted or not
```

- Once EFS is mounted, change the ownership to Jenkins using the following command,

```
chown -R jenkins:jenkins /var/lib/jenkins/jobs
```

- Start Jenkins Service using the command,

```
systemctl start jenkins
```

once you finish the configuration on Jenkins A, ssh to Jenkins B instance and perform the same steps. Then, validate mounts using the “mount -a” command.

So to test the HA, create one test job in Jenkins A and now reload the configuration of jenkins B , you will be able to see the test job there as well. you can write the reload script and run it every minute through cron so that the configuration will always be updated.

STEP 6: Setup of reloading configuration, HA proxy.

write a cron-job to reload the configuration.

The Jenkins Master should keep reloading the configuration being written to the disk (Shared Drive). This may be achieved by configuring a cron job to run a custom reload script on the Jenkins instance.

Install the jenkins cli file.

```
wget http://jenkins-master-ip:8080/jnlpJars/jenkins-cli.jar
```

Create one file in the root folder and paste this script.

```
#!/bin/bash
java -jar /root/jenkins-cli.jar -s http://jenkins-master-ip:8080 -
webSocket -auth username:password reload-configuration
```

Create a cronjob to reload the configuration after every 15 sec.

```
cd /etc/cron.d
sudo vi jenkins_reload
```

Paste the below code in jenkins_reload file

```
* * * * * root /bin/bash /root/script
* * * * * sleep 15; root /bin/bash /root/script
* * * * * sleep 30; root /bin/bash /root/script
* * * * * sleep 45; root /bin/bash /root/script
```

This will run the script after every 15 sec and reload the jenkins configuration.

once you finish the configuration on Jenkins A, ssh to Jenkins B instance and perform the same steps.

Now, Jenkins A and Jenkins B have redundant data on both instances. But what if one Jenkins instance goes down? How can we achieve active-passive synchronization between Jenkins instances? The answer is a High Availability setup wherein both Instances are configured in the HA configuration file. So let's move to the next part configuring HAProxy setup on a separate instance which will be front-facing for both Jenkins instances. Let's see how we can achieve that -

HAProxy Setup

- Go to <https://aws.amazon.com/ec2/>
- Launch instance with the basic configuration required and download PEM key. Once the instance is in the ready state, log in to the instance.

```
sudo apt-get update
sudo apt install -y haproxy
```

edit file “/etc/haproxy/haproxy.cfg” as per the content below:

```

defaults
    log global
    maxconn 2000
    mode http
    option redispatch
    option forwardfor
    option http-server-close
    retries 3
    timeout http-request 10s
    timeout queue 1m
    timeout connect 10s
    timeout client 1m
    timeout server 1m
    timeout check 10s

frontend ft_jenkins
    bind *:80
    default_backend bk_jenkins
    reqadd X-Forwarded-Proto:\ http

backend bk_jenkins
    server jenkins1 <primary_jenkins_ip>:8080 check
    server jenkins2 <secondary_jenkins_ip>:8080 check backup

```

Now start the ha proxy server.

```
systemctl start haproxy
```

Once the high availability setup is done, we will have to test it. For this, we can stop Jenkins A and try to access it with the help of a domain name or IP address, which is common for both machines. If you are able to access the Jenkins console, all your Jenkins jobs/user and complex CI/CD pipelines created are accessible.

STEP 7: SETUP OF SLACK NOTIFICATION

First, let's start by logging into Jenkins and navigating to *Manage Jenkins > Plugin Manager*.

Then, on the *Available* tab, we'll search for *Slack*:

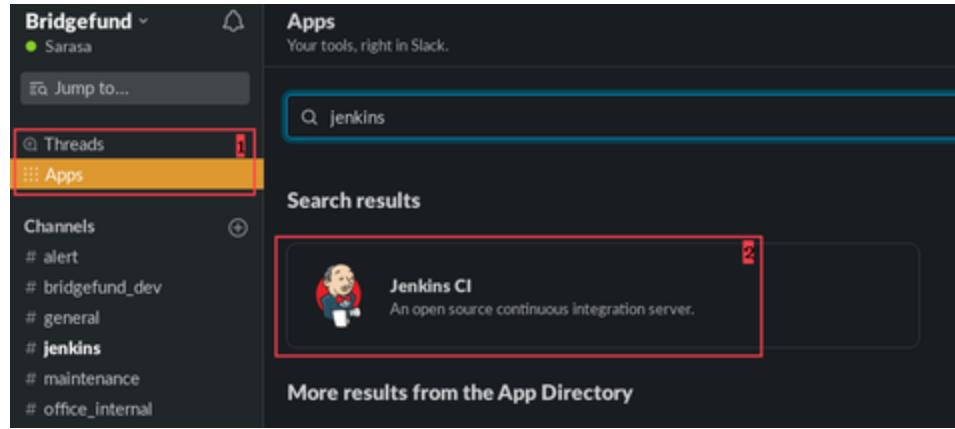
The screenshot shows the Jenkins 'Plugin Manager' interface with the 'Available' tab selected. A search bar at the top contains the word 'Slack'. Below the search bar, a table lists several plugins:

Install	Name	Version
<input type="checkbox"/>	Slack_Upload	1.7
<input type="checkbox"/>	cucumber-slack-notifier	0.8.3
<input checked="" type="checkbox"/>	Slack_Notification	2.35
<input type="checkbox"/>	Build Notifications	1.5.0
<input type="checkbox"/>	Global Slack Notifier	1.5

At the bottom of the screen, there are three buttons: 'Install without restart', 'Download now and install after restart', and 'Check now'. A status message 'Update information obtained: 4 min 57 sec ago' is also visible.

Let's select the checkbox for *Slack Notification* and click *Install without restart*.

1. Go to the slack desktop and click on Add app.



2. Click on Jenkins CI Configuration Add to slack.

Jenkins CI

[App Info](#) [Settings](#)

This app was made by Slack.
This app was made by a member of the Slack team to help connect Slack with a third-party service; these apps may not be tested, documented, or supported by Slack in the way we support our core offerings, like Slack Enterprise Grid and Slack for Teams. You may provide feedback about these apps at feedback@slack.com.

It only uses data Slack already has access to (view our [Privacy Policy](#) to learn more). By enabling and/or using this app, you may be connecting with a service that is not part of Slack.

[Add to Slack](#)

[App help](#) [Terms](#)

3. Select a channel or click on create new slack channel. and click on Add jenkins CI integration.

Jenkins CI
An open source continuous integration server.

Jenkins CI is a customizable continuous integration server with over 600 plugins, allowing you to configure it to meet your needs.

This integration will post build notifications to a channel in Slack.

Post to Channel
Start by choosing a channel where Jenkins notifications will be posted.

Choose a channel... or create a new channel

[Add Jenkins CI integration](#)

we need to configure new credentials. Let's navigate to *Jenkins > Credentials > System > Global Credentials* and add a new *Secret text* credential:

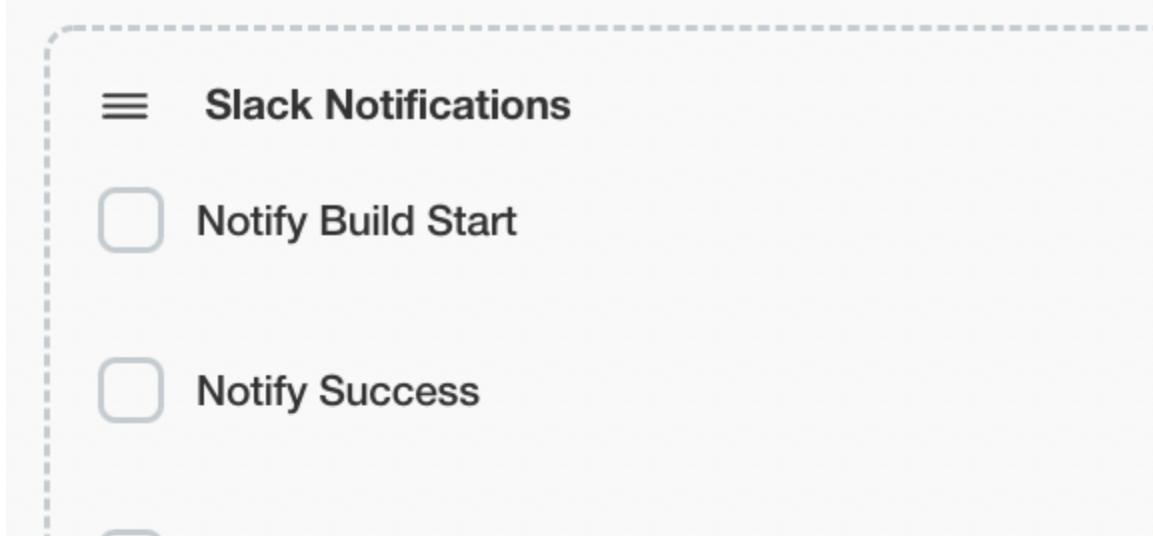
Scope	Global (Jenkins, nodes, items, all child items, etc)
Secret
ID	slack-token
Description	OAuth token for Slack

Save

1. Now go to the Jenkins dashboard Manage Jenkins Configure System → Search for **Global Slack Notifier Settings**
 - a. Add Team subdomain: go-betterplace
 - b. Integration token id:
 - c. Save the credentials
2. In the Jenkins job go to the Post-build actions and select slack notification.



and select action for which you want to be notified in slack.





Notify Aborted



Notify Not Built



Notify Unstable



Notify Regression



Notify Every Failure



Notify Back To Normal

Advanced...

Add post-build action ▾

- Which build phases to send messages for (start, success, failure, etc)
- The name of the credentials to use – the ones we added previously
- The Slack channel name or member ID to send messages to

We can also specify additional fields if desired, such as commit information used for the Jenkins job, custom messages, custom bot icons, and more:

Notification message includes	nothing about commits
	What commit information to include into notification message
Workspace	
Override url	
Credential	OAuth token for Slack ▾ Add ▾
Slack Token (deprecated)	
Custom slack app bot user	<input type="checkbox"/>
Icon Emoji	
Username	
Channel / member id	#jenkins

When setting things up via the UI, we can use the *Test Connection* button to ensure that Jenkins can reach Slack.

3. Click on Save and Apply.

4. And once the Build is built we'll get notification on slack for the same.

The screenshot shows three messages from the Jenkins app in Slack:

- jenkins APP 1:21 PM**
Slack/Jenkins plugin: you're all set on
<http://3.108.65.143:8080/>
- jenkins APP 1:27 PM**
test123 - #7 Started by user OL-Jenkins ([Open](#))
test123 - #7 Success after 2.7 sec ([Open](#))
- jenkins APP 2:05 PM**
test123 - #8 Started by user OL-Jenkins ([Open](#))
test123 - #8 Success after 3.1 sec ([Open](#))