Caltech | Center for Technology & Management Education

Post Graduate Program in DevOps

simplilearn

DevOps

Caltech | Center for Technology & Management Education

CI/CD Pipeline with Jenkins

simplilearn

Jenkins Distributed Builds

Learning Objectives

By the end of this lesson, you will be able to:

- Detail the structure, functions, and working of the Jenkins Master Slave Architecture
- Discuss the process of Jenkins Build Agent configuration
- Set up a new Slave agent in Jenkins
- Outline the process of assigning a specific Jenkins job to a Slave machine

Jenkins Build Management

Introduction to Jenkins Distributed Builds

Jenkins offers distributed Builds to relieve the dependency on the Master to act on all Build requests.

Jenkins distributed Builds can be used to set up a farm of Build servers that can share the load of running Build jobs across multiple environments.

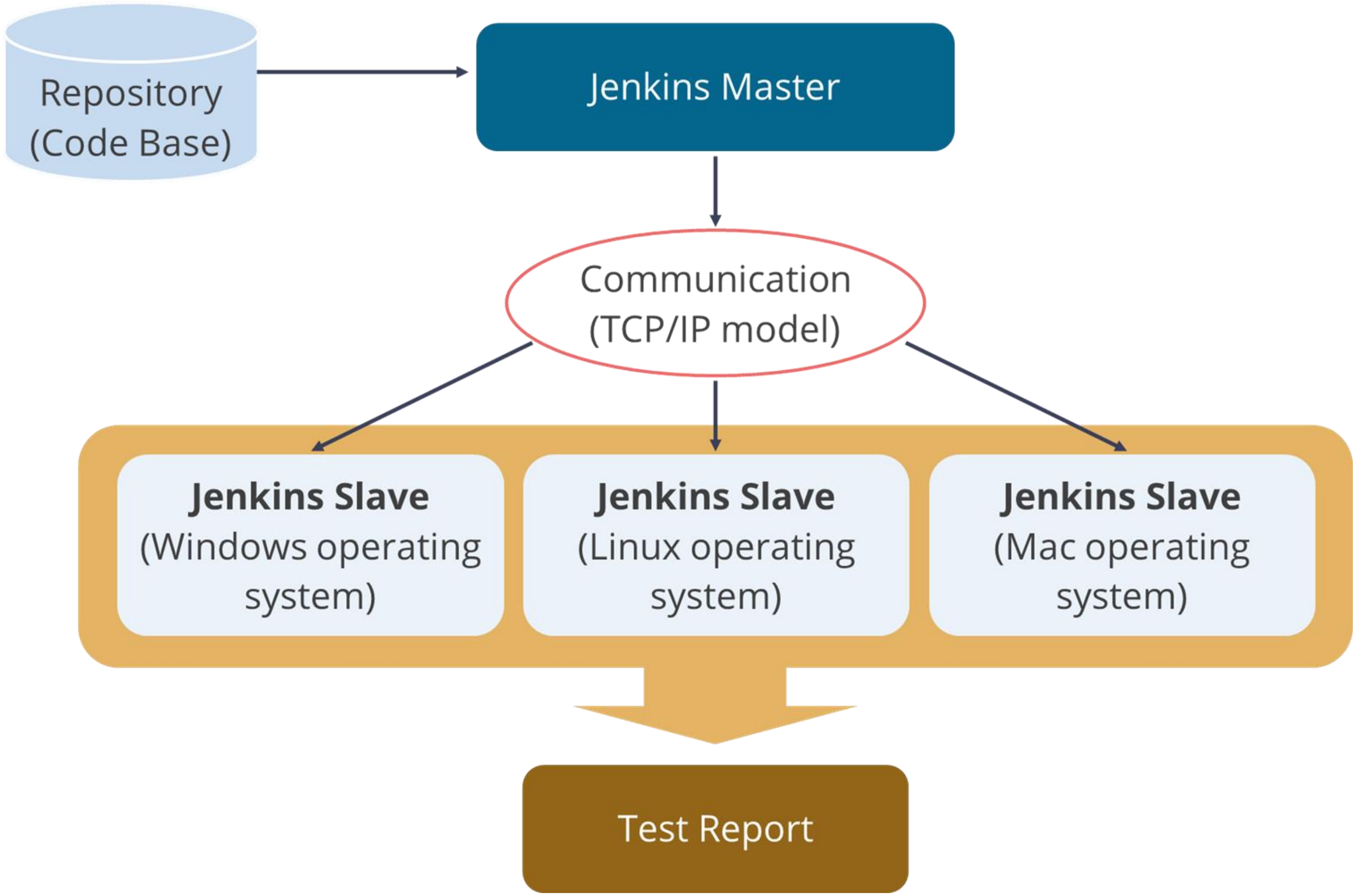
This feature is largely employed by organizations that need to execute Builds in a distributed manner, with failover and load-sharing capability.

Advantages of Distributed Builds

1. Availability of fault tolerance
2. Reduced latency
3. Maximized availability
4. Great durability
5. Better scalability
6. Increased efficiency

Working of Jenkins Master Slave Architecture

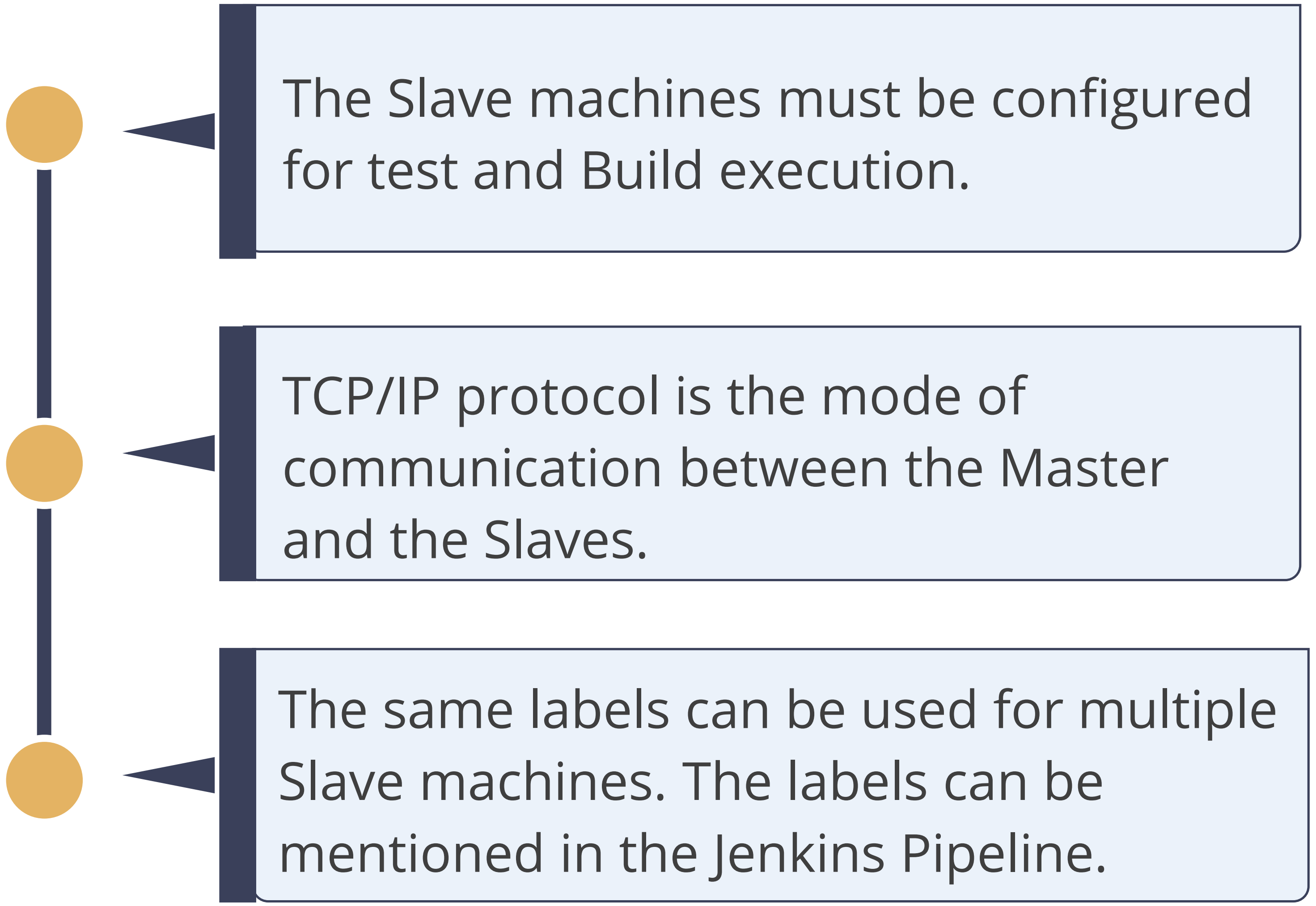
Jenkins manages distributed Builds through Master-Slave architecture.



Repository (Code Base)
Jenkins Master
Communication (TCP/IP model)
Jenkins Slave (Windows operating system)
Jenkins Slave (Linux operating system)
Jenkins Slave (Mac operating system)
Test Report

Working of Jenkins Master Slave Architecture

Jenkins can perform **build automation** across multiple platforms. Testing can be performed across multiple environments, without having to depend on a single machine.



The Slave machines must be configured for test and Build execution.

TCP/IP protocol is the mode of communication between the Master and the Slaves.

The same labels can be used for multiple Slave machines. The labels can be mentioned in the Jenkins Pipeline.

Jenkins Master

The Jenkins main server is known as the Jenkins Master.

It manages **Build** requests and schedules them in a well-defined manner.

It monitors **Build** slave agents and redirects **Build** requests in case of failure.

It supports execution of **Builds** when slave machines are absent.

Jenkins Slave

Jenkins Slave is a program controlled by the Jenkins Master.
Jenkins Master allots Build requests to Jenkins Slaves.

- Any Jenkins task can be configured for Build execution on a specific Slave machine or on any available Slave machine.
- The Slave machine shares the load so that the Jenkins Master can concentrate on managing nodes.
- Jenkins Slaves can be easily installed and configured across multiple platforms including Ubuntu, CentOS, macOS, Windows, and RedHat.

Slave Machines

There are various cloud solutions that deploy slave machines for executing Builds remotely.

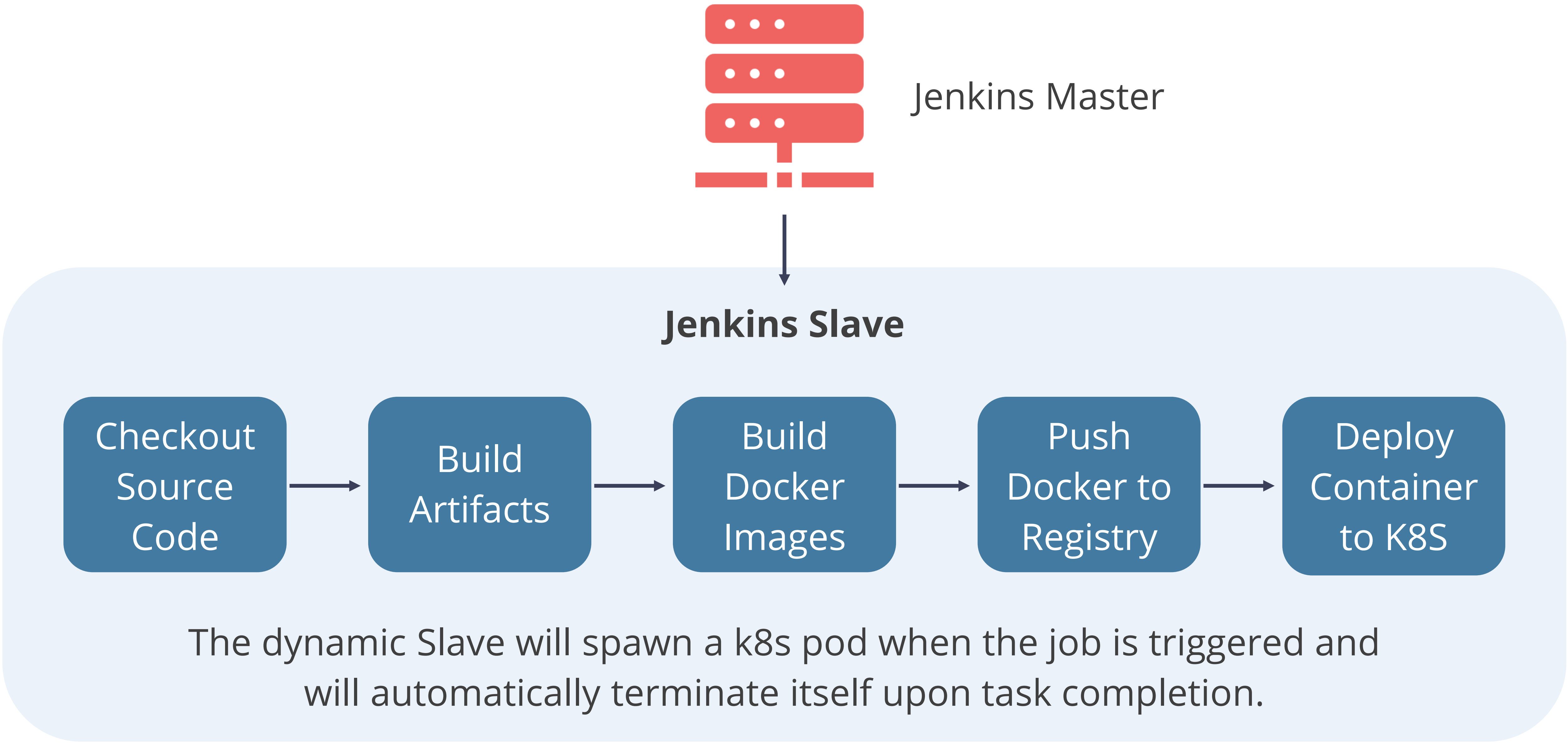
Plugins help in organizing the dynamic slave machines.

Plugins can be discarded upon completion of the Build process.

Examples: EC2 plugin. Docker plugin, and Kubernetes plugin.

Dynamic Slave Machines

Jenkins allows dynamic provisioning of Slaves.



Jenkins Master

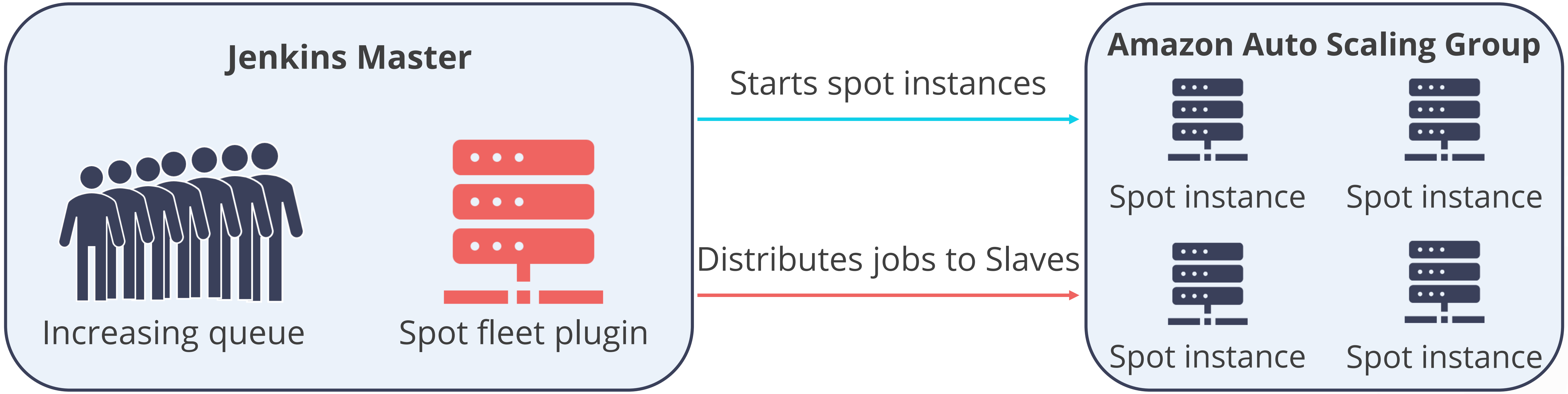
Jenkins Slave

Checkout Source Code → Build Artifacts → Build Docker Images → Push Docker to Registry → Deploy Container to K8S

The dynamic Slave will spawn a k8s pod when the job is triggered and will automatically terminate itself upon task completion.

Jenkins EC2 Plugin

The Jenkins EC2 plugin helps to integrate Jenkins with the AWS Cloud.



While configuring this plugin, the EC2 service access to the IAM user needs to be enabled with programmatic access.

Advantages of Jenkins EC2 Plugin

- With the help of Jenkins EC2 plugin, development teams can dynamically launch an instance on AWS cloud without having to build Jenkins Build servers.

- This leads to cost-saving as the dedicated Slave machines are not required for long time periods.

- This plugin supports both on-demand and spot instances for Slave machines.

Jenkins EC2 Fleet Plugin

The Jenkins EC2 Fleet Plugin is similar to the EC2 plugin though it provides more flexibility around spot instances.

Problem

- Spot instances come with the risk of getting terminated when another user bids a higher value.

Solution

- This plugin will maintain fleet by changing the spot prices and maintaining it within the price range.

Jenkins EC2 Fleet Plugin

In case a spot instance gets terminated, this plugin helps to launch a replacement to maintain the minimum set of Slave machines.

EC2 Plugin
All configurations need to be specified within Jenkins itself.

EC2 Fleet Plugin
The plugin supports usage of launch configuration and launch templates.

Jenkins Kubernetes Plugin

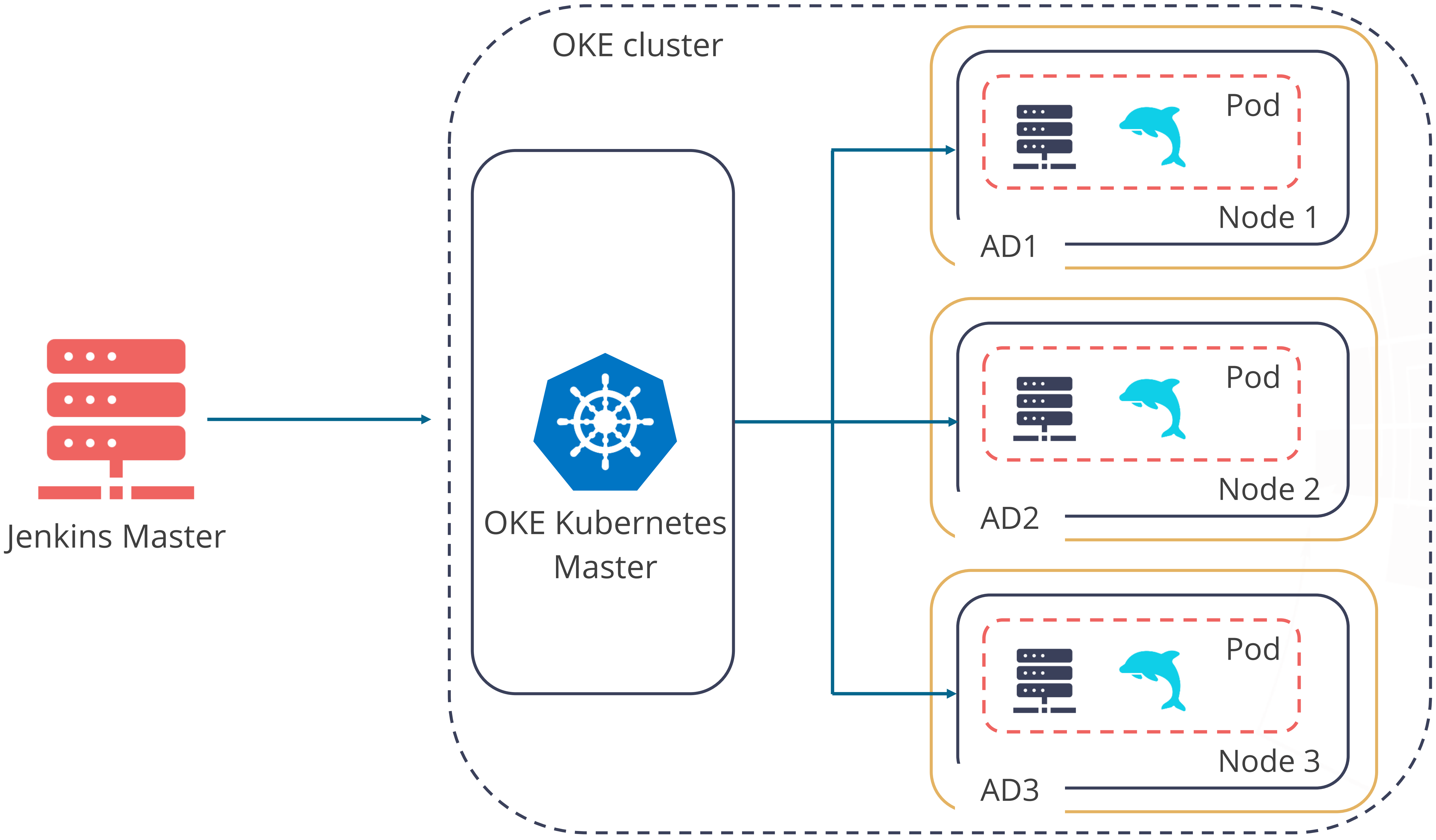
The Jenkins Kubernetes plugin is used to run dynamic slave machines in the Kubernetes cluster.

The plugin creates a new pod every time a new Build request is raised, and terminates it once the Build is completed.

- These pods contain a special container JNLP that runs the Jenkins Agent process.
- This plugin receives user input for labels that invoke a new pod.

The plugin supports Kubernetes cluster 1.14 or above.
For OpenShift, it supports OpenShift 4.x platforms.

Jenkins Kubernetes Plugin



Assisted Practice

Setting up New Linux VM as Slave Machine

Duration: 15 min

Problem Statement:

1. Set up a new Linux VM as a slave machine.

Assisted Practice: Guidelines

Steps to Set up a new Linux VM as a slave machine:

1. Login to Jenkins and navigate to Manage Jenkins and then Manage Nodes.
2. Click on New Node and provide the Node name.
3. Provide Agent name, Number of executors and remote home directory where Jenkins workspace and other required files would be deployed.
4. To connect with the slave from Jenkins master, select Launch method as SSH and provide hostname or IP, credentials to login using SSH. Disable the host key.
5. Save the Slave configuration.
6. Configure the user on slave VM.
7. Modify the configuration file to allow Password Based SSH and restart the SSH service.
8. Install Java on the Slave machine.
9. Perform restart of the sshd service.
10. Post that you need to relaunch slave in Jenkins.

Assisted Practice

Run a New Freestyle Jenkins Job on Slave

Duration: 10 min

Problem Statement:

1. Run a new Freestyle Jenkins Job on slave.

Assisted Practice: Guidelines

Steps to run a new Freestyle Jenkins Job on slave:

1. Login to Jenkins and navigate to Free Style job and select desired job.
2. Click on Configure option to configure Jenkins job and execute builds on slave machines.
3. Navigate to General tab and scroll down to configuration named as Restrict.
4. Provide the label name which will define the slave machine.
5. Trigger fresh build to understand if the new build is getting triggered on the remote slave machine.

Key Takeaways

- Jenkins offers distributed Builds to relieve the dependency on the Master to act on all Build requests.

- Jenkins supports multiple programming languages, which enables it to perform Build automation across multiple platforms.

- Several cloud solutions including EC2 plugin, Docker plugin, and the Kubernetes plugin can be used for organizing dynamic Slave machines.

Configure a Windows Slave Machine to Trigger Build on a Slave

Lesson-End Project

Problem Statement:

Perform the following:

- Create a Freestyle Job to send an email upon a successful Build.

Access: Click on the **Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Thank You