Jenkins Deep Dive:

 # Why Jenkins:
 # Jenkins wikipedia => https://en.wikipedia.org/wiki/Jenkins_(software)

Formal Introduction of CI, CD, CD:
 # Continuous Integration # is usually the initial part of both Continuous Delivery and Deployment, involving the testing and building of any new or updated source code.
 # Continuous Delivery # involves a manual trigger to production.
 # Continuous Deployment # involves automatic releases to production.

Jenkins Advantages:
 # It is open source and completely free.
 # Issues with tests and builds are detected easily and reported almost immediately.
 # Jenkins is platform independent, available on Windows, macOS, and Linux.
 # It is easily configurable and customisable for any project.

LAB-1:
 # Pre-Requisites of Jenkins
 # Install Java:
 # https://www.oracle.com/technetwork/java/javase/downloads/index.html
 # export JAVA_HOME=/usr/local/java-current # set the Java Path
$ echo $JAVA_HOME # /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home
$ java -version # To check the version of Java, Some projects is version dependent so make sure with Dev Team post configuration in Jenkins

 # Install Tomcat:
 # https://tomcat.apache.org/download-90.cgi
 # Click on Tomcat 9 under Download
 # Go to Binary Distributions And Download
 # Copy the Jenkis.war file which was downloaded from the previous section and copy it to the webapps folder in the tomcat folder.
$ E:\Apps\tomcat7\bin>startup.bat # For MacOS it's different

Install Maven and check the version:
Normal
https://maven.apache.org/download.cgi
Go to Files and Download “Binary.zip”
maven files will be placed in E:\Apps\apache-maven-3.3.3.
Configure System
In the Configure system screen, scroll down till you see the Maven section and then click on the ‘Add Maven’ button.
Uncheck the ‘Install automatically’ option.
Add any name for the setting and the location of the MAVEN_HOME.
Click on the ‘Save’ button at the end of the screen.

Install Maven on MacOSand check the version:
https://www.baeldung.com/install-maven-on-windows-linux-mac
For MacOS
Download apache-maven-3.6.0-bin.tar.gz
#
$ mvn -v

# Install git and check the version and configuration
# Manage Jenkins And Install “Git plugin” And Click on “Install without restart”
# http://localhost:8080/jenkins/restart
$ git —version

# Process to access Jenkins:
# https://jenkins.io/doc/book/installing/ # For MacOS
# Manual Steps:
# https://jenkins.io/ # Download the Jenkins War File
$ cd /Users/javedalam/Documents/Jenkins/ # Moved the place where “jenkins.war” file downloaded
$ java -jar jenkins.war # Start the Jenkins;
# Deep dive into CLI Logs, admin password reflects,
# cd ~; cd /Users/javedalam/.jenkins => Jenkins Database, Never delete unless you want completely uninstall the Jenkins
# Plugin installations steps and so on
# Hit in Browser: http://localhost:8080/ # To access the Jenkins
# Unlock the Jenkins:
$ cat /Users/javedalam/.jenkins/secrets/initialAdminPassword => Administrator Password
# Click on Continue
# Click on Installed suggested Plugin
# Automatically 20 Default Plugins will be installed like: wait for 10 mins
# Folders
# OWASP Markup Formatter

Build Timeout
Credentials Binding
Timestamper
Workspace Cleanup
Ant
Gradle
Pipeline
GitHub Branch Source
Pipeline: GitHub Groovy Libraries
Pipeline: Stage View
Git
Subversion
SSH Slaves
Matrix Authorization Strategy
PAM Authentication
LDAP Email Extension
Mailer

Create first Admin User:
Username: xavy
Password: redhat@123
Confirm password: redhat@123
Full name: xavyaly
E-mail address: wellboy.alam13@gmail.com

Click on Save and Finish
Start using Jenkins

Perfect:
Credentials: xavy/redhat@123 for MacOs (http://localhost:8080)

Verify the default Jenkins Path - Done
http://localhost:8080/ (xavy/redred@123)
http://13.126.38.248:8080/ (bablu/ redhat@123)

LAB-2:
Create a sample new Job
Create a new Job from from the existing Job OR Copy a Job from existing Job
Create a folder in Jenkins - Done
How can we pass the Git URL's in Jenkins Job ?
How we can add the user credentials in Jenkins ?
Kind of Jobs in Jenkins => Normal Jobs, Self Service Jobs, Pipeline Jobs

 # Where exactly Jenkins Keep the build file => Check the default workspace

LAB-X:
Clone the game-of-life Project
Explanation of target folder
Explanation of ".jar" file in Java Project
Manual Execution "$ mvn package install"
Target folder created automatically
Introduction of Maven like pom.xml ?
Create a Maven Job in Jenkins
How to add the GitHub URLs ?
How to add the Maven Operations like compile, test, package etc... ?
How to perform Manual Build ?
Play with it with some errors
Troubleshoot the Jenkins Job via console and CLI Logs ?
How to trace the Errors in Jenkins Jobs ?
How to do Build Triggered Jobs Automatically ?
    Build Triggered Functionality
    Poll SCM
    Web Hook

 # LAB-3:
 # Install & UnInstall the Plugins:
    # Process to install & UnInstall the multiple plugins
 # Important Plugins
    # green balls plugin, Git Plugin, Maven Integration, Digital Ocean, GitHub, version number, Nexus Plugins and so on...
    # Hudson Selenium Plugin # For Automated Testing
    # Verify the Plugins once installed
 # Some default Plugins installed while installation and some need to install as per requirement
    # Option to Install Plugins without restart OR Install with start Jenkins, Need alert while restart option
 # Restart the Jenkins in two ways:
    # CLI: sudo service Jenkins restart -> sudo service <service-name> restart
    # UI: localhost:8080/restart
 # How to stop the running Jenkins Job
    # Abort the running Job but check the workspace first

 # LAB-4:
 # Overview of Downstream & UpStream Jobs in Jenkins ? - Done Job1,

Job2 & Job3

#

Job1 execution successfull -> Build No (1.1.001) -> Job2 (Pass Version a build with parameter) -> Job3

LAB-5:
Introduction of Manage Jenkins
Managing Nodes => Master & Slave
How can manage the matrix permissions in Jenkins ?
How can give the permissions to specific person ?
ect....

LAB-6
Execution of Maven Projects

LAB-7:
Introduction of Jenkins Workspace
Process to Clean the Jenkins Workspace and why it requires ?

LAB-8:
Troubleshoot the Jenkins Job via console and CLI Logs
How to trace the Errors in Jenkins Jobs

LAB-9:
Implementation & Benefits of Jenkins Pipeline Job:

Pipeline basically need a basic understanding of groovy language

No need to install multiple plugins to execute any job # Main pillar

In normal Job tracing error will be challenging part but in Pipeline we break the execution process into stages.

How can we make the stages in Jenkins Pipeline Job which helps to track the error easily

Maintaining of code will be quite easy in Pipeline

Create a sample Pipeline Job

Execute a Hello World Jenkins Pipeline Job

LAB-10:
Automatic deployment of build

Build Trigger

Poll SCM in Jenkins

What is the difference between the SCM Poll and Normal Crontab

Jobs

LAB-11:
How can we create the docker images via Jenkins ?
How can we tag the existing docker images in Jenkins ?
How can we push the 2 docker images in any docker hub registry in Jenkins ?

LAB-12:
Automation in Jenkins
How can we take the existing Jobs backup in Jenkins (generally we take the backup of ".xml" files) ?
Script for Start and Stop of Jenkins=> https://wiki.jenkins.io/display/JENKINS/Starting+and+Accessing+Jenkins

LAB-13:
Pre-Requisites of Nexus ?
Nexus used to backup the Jenkins Build
How can push the build to our Nexus or JFrog Artifactory ?
How can we manage the Nexus Build ?
How can manage the Nexus Plugins and Configuration of Nexus ?

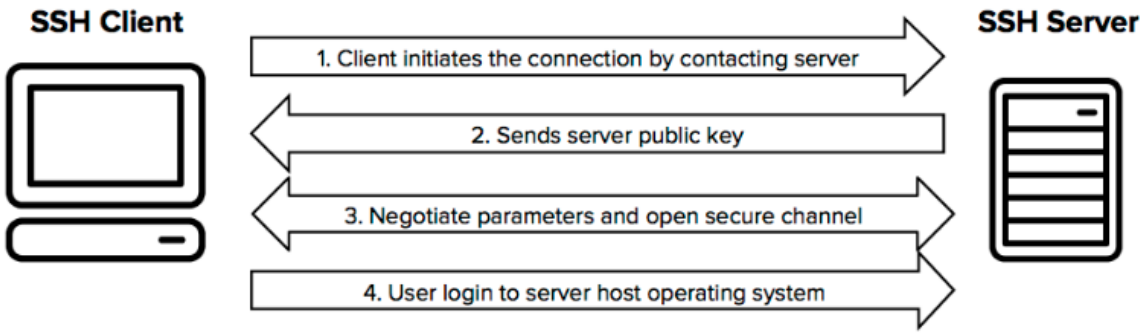
# LAB-14:
# Fork the game-of-life project and clone the Project
    $ git clone <branch-name>
    $ mvn clean install -DskipTest
    # target => <app-name>.tar

LAB-15:
Create a Dockerfile
Build and image
Push that image in docker hub
Create a service

LAB-16:
ssh-keygen: https://www.ssh.com/ssh/download/
Configuration of SSH
    # The SSH protocol uses encryption to secure the connection between a client and a server.
    # All user authentication, commands, output, and file transfers are

encrypted to protect against attacks in the network

Diagram:



UDEMY:

https://www.udemy.com/jenkins-jenkins-unchained-deploy-jenkins-to-cloud-3-in-1/learn/v4/content

#: root
$: user
//: comment

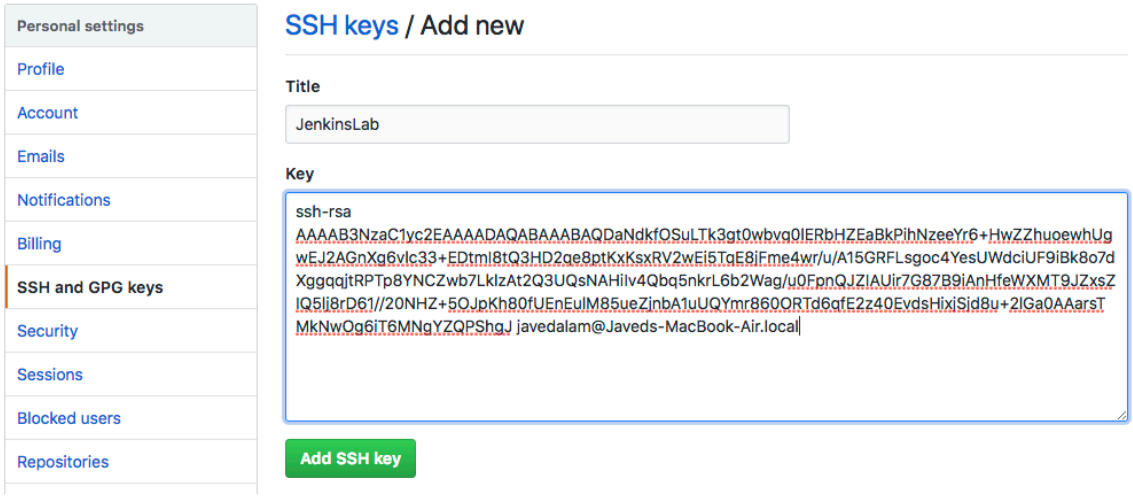
Git: CLI
yum -y install git

GitHub: UI
Create an account and Start the Project

LAB-2:

SSH-KEYGEN:

Create and Add the SSH-KEYGEN in GitHub:
ssh-keygen // Enter + Enter
cat /root/.ssh/id_rsa.pub // copy the public key
// Move to GitHub -> Settings -> SSH and GPG Keys -> Add Title “Lab1” + Paste the content in Keys -> Enter the Password



// Need to add the same key in Jenkins in near future
 # /var/lib/jenkins // Home Directory for Jenkins

LAB-3:

DISASTER RECOVERY:

Configure AWS CLI
// https://docs.aws.amazon.com/cli/latest/userguide/cli-install-macos.html#awscli-install-osx-path
OR
// https://cloudacademy.com/blog/how-to-use-aws-cli/

$ aws —version // aws-cli/1.16.61 Python/3.6.3 Darwin/17.5.0 botocore/1.12.51

$ pwd
/Users/javedalam/Documents/DevOpsLabs/AmazonWebServices

$ aws configure // https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html
AWS Access Key ID [None]: AKIAJNRPZXRQM7L5W7RQ
AWS Secret Access Key [None]:
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
Default region name [None]: ap-south-1 // Critical
Default output format [None]: json

// Cross Verify:
$ aws ec2 describe-instance-status --instance-id <instance-id> **// check the state**

$ aws ec2 start-instances --instance-ids i-0e6d152d8f541798c // Start the Instance

---

LAB-4:

// Create an AWS Instance through CLI or UI
Login to an existing Jenkins Instance:
$ cp ../../../Downloads/devops-key-pair.pem . // copy the key pair
$ ssh -i "devops-key-pair.pem" ubuntu@ec2-52-66-203-141.ap-south-1.compute.amazonaws.com
 # sudo su
 # df -hT
/dev/xvda1 ext4 7.7G 2.8G 5.0G 37% /

LAB-5:

Create a new Volume of 30 GB AND Attach with a running Instance:

$ aws ec2 create-volume --size 80 --region us-east-1 --availability-zone us-east-1a --volume-type gp2
Original-Lab:
$ aws ec2 create-volume --size 30 --region ap-south-1 --availability-zone ap-south-1a --volume-type gp2

```
-------------------------------------------------
|                CreateVolume                   |
+------------------+----------------------------+
|  AvailabilityZone|  ap-south-1a               |
|  CreateTime      |  2018-11-26T10:50:48.000Z  |
|  Encrypted       |  False                     |
|  Iops            |  100                       |
|  Size            |  30                        |
|  SnapshotId      |                            |
|  State           |  creating                  |
|  VolumeId        |  vol-0e03aae194191e3f0     |
|  VolumeType      |  gp2                       |
+------------------+----------------------------+
```

LAB-6:

Add this volume to that instance Volume through CLI or UI

$ aws ec2 attach-volume --volume-id vol-1234567890abcdef0 --instance-id i-01474ef662b89480 --device /dev/sdf
Original-Lab:
$ aws ec2 attach-volume --volume-id vol-0e03aae194191e3f0 --instance-id i-0e6d152d8f541798c --device /dev/sdf


```
{
    "AttachTime": "2018-11-26T10:56:34.061Z",
    "Device": "/dev/sdf",
    "InstanceId": "i-0e6d152d8f541798c",
    "State": "attaching",
    "VolumeId": "vol-0e03aae194191e3f0"
}
```


Check the File Disk
$ **fdisk -l**
Disk /dev/xvdf: 30 GiB, 32212254720 bytes, 62914560 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Format the File System:
$ mkfs.ext4 /dev/xvdf

Mount the new Volume
$ sudo nano /etc/fstab // /dev/xvdf /var/lib/jenkins ext4 defaults 0 0
$ sudo mount /var/lib/jenkins/
$ df -hT

Install the JDK Package:
// Leave it as of now
$ yum -y install java-1.8.0-openjdk

LAB-7:

Install Jenkins on Ubuntu Machine:

// https://linuxize.com/post/how-to-install-jenkins-on-ubuntu-18-04/
// http://52.66.203.141:8080/login?from=%2F // Hit in Browser
// Create an Account // Populate all the required parameters
// Admin => bablu/ redhat@123

Troubleshooting:
// IF Error Like: aly is missing the Overall/Read permission
// Check the config.xml file
$ cd /var/lib/jenkins/
$ nano config.xml

Troubleshooting in CLI:
// https://www.jeffgeerling.com/blog/2018/fixing-jenkins-cli-error-anonymous-missing-overallread-permission

Remove Jenkins from Ubuntu Machine
// https://stackoverflow.com/questions/38604715/how-can-i-remove-jenkins-completely-from-linux
$ sudo service jenkins stop
$ sudo yum remove jenkins
$ sudo apt-get remove --purge jenkins

Handle Jenkins:
$ sudo service jenkins status
$ sudo service jenkins restart
$ sudo service jenkins stop
$ sudo service jenkins start

LAB-8:

Installation of Jenkins from Packages and WAR Files:

jenkins.io // Download Jenkins
https://pkg.jenkins.io/redhat-stable/ // Follow the Process
$ ls -ld /var/lib/jenkins // no outputs
$ systemctl start jenkins // to start the jenkins
$ ps -ef | grep jenkins
$ ls -ld /var/lib/jenkins // can see the output
$ netstat -ntlp // 8080 will be listen to Jenkins Default Port
http://<IP>:8080 // Hit the browser to access the Jenkins

LAB-8:

INSTALL TOMCAT in CentOS:

$ yum -y install tomcat // install tomcat
$ ls -l /usr/share/tomcat // tomcat related files
http://mirrors.jenkins.io/war-stable/latest/jenkins.war // Copy the Link

for Generic Java package (.war) from https://jenkins.io/download/
$ cd /usr/share/tomcat/webapps // move into this directory
$ wget http://mirrors.jenkins.io/war-stable/latest/jenkins.war // Download the war file
$ systemctl start tomcat // start the tomcat
$ ps -ef | grep tomcat // cross check the tomcat started correctly
http://<IP>:8080/jenkins // Hit the browser to access the Jenkins
$ cd /usr/share/tomcat // move to locate the ".jenkins" hidden directory
$ ls -la // ".jenkins" directory will be present
$ ls -la .jenkins // home directory for jenkins
$ systemctl stop jenkins // stop the jenkins
$ mkdir -p /opt/jenkins_home //
$ chown -R tomcat:tomcat /opt/jenkins_home // change the ownership
$ nano /etc/tomcat/context.xml // edit the xml file
<Context>
<Environment name="JENKINS_HOME" value="/opt/jenkins_home" type="java.lang.String" />
<Context/>
// Save the file
$ systemctl start tomcat // start the tomcat
$ ls -l /opt/jenkins_home // change the default directory
$ cat /etc/sysconfig/jenkins // change the path then Jenkins will have a new home directory
JENKINS_HOME="/var/lib/jenkins" // change and restart the tomcat

---

LAB-9:

INSTALLATION OF JENKINS THROUGH DOCKER CONTAINERS:

$ yum install -y yum-utils device-mapper-persistent-data lvm2 // install the additional packages
$ yum-config-manager —add-repo https://download.docker.com/linux/centos/docker-ce.repo // configure the docker repository
$ yum -y install docker-ce // install the docker
$ systemctl start docker // start the docker
$ ps -ef | grep docker // see the process
https://jenkins.io/download/ // Click on Docker
https://github.com/jenkinsci/docker/blob/master/README.md // Click on Documentation; Complete steps to install the Jenkins
$ docker pull jenkins/jenkins:lts // pull the Jenkins Image
$ docker run —name jenkins_master -p 8080:8080 -v jenkins_home:/var/

jenkins_home jenkins/jenkins:lts
// run the jenkins container to launch a docker container with the name jenkins_master; jenkins_home is the volume
OR
$ docker run —name jenkins_master_2 -d -p 8080:8080 -v jenkins_home:/var/jenkins_home jenkins/jenkins:lts // no output
// give the SHA ID ONLY I hope you remember
$ docker ps // list all the containers

// SUCCESSFULLY DONE

LAB-10:

CONFIGURE REVERSE PROXY AND SETTING UP USER INTERFACE FOR JENKINS:

Install nginx from the EPEL repo
Configure the nginx configuration file for jenkins reverse proxy
Access the jenkins user interface and go through the setup wizard
Configure SSH user with the keys and GIT data

Steps:
https://dl.fedoraproject.org/pub/epel/7/x86_64/Packages/e/ // ctrl+F -> releases; copy the Link Address
$ rpm -ivh https://dl.fedoraproject.org/pub/epel/7/x86_64/Packages/e/epel-release-7-11.noarch.rpm // download the rpm file
$ yum -y install nginx // install nginx package
$ nano /etc/nginx/nginx.conf // check the file and remove the server part and save

```
server {
    listen       80 default_server;
    listen       [::]:80 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
        location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
        location = /50x.html {
    }
}
```

$ nano /etc/nginx/conf.d/jenkins.conf // save the changes

```
upstream jenkins {
    server 127.0.0.1:8080;
}

server {
    listen      80 default;
    server_name jenkins.course;

    access_log  /var/log/nginx/jenkins.access.log;
    error_log   /var/log/nginx/jenkins.error.log;

    proxy_buffers 16 64k;
    proxy_buffer_size 128k;

    location / {
        proxy_pass  http://jenkins;
        proxy_next_upstream error timeout invalid_header http_500 http_502 http_503 http_504;
        proxy_redirect off;

        proxy_set_header    Host            $host;
        proxy_set_header    X-Real-IP       $remote_addr;
        proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header    X-Forwarded-Proto https;
    }
```

$ systemctl start nginx // start the nginx
$ ps -ef | grep nginx // cross check the process
http://jenkins.course/login // Unlock jenkins page appears
// populate the password; installed the Plugins; Create an Admin user and Start using
// Manage Jenkins -> Configuration -> Jenkins Location -> http://

jenkins.course/login <should be same>
// Manage Jenkins -> Configuration -> Git Plugin -> Enter user.name and user.email <personal-one>-> Save

---

LAB-11:

CREATE A GLOBAL CREDENTIALS TO ACCESS FROM EVERYWHERE:

$ cat ~/.ssh/id_rsa // copy it
http://localhost:8080/credentials/ -> Credentials -> Jenkins -> Global Credentials -> Add Credentials
// Kind: SSH Username with private key
// Username: xavy
// paste the private key

LAB-12:

AUTOMATING THE JENKINS INSTALLATIONS AND CONFIGURATION PROCESS:

Disabling the manual SetUp Wizard
Automating the SetUp Wizard steps
Creating a puppet configuration to automate the installation and configuration process
Automating the process by applying the Puppet Module

Steps:
sudo service jenkins stop // stop the jenkins
sudo nano /etc/sysconfig/jenkins
// JENKINS_JAVA_OPTIONS="-Djava.awt.headless=true" // Initially
// JENKINS_JAVA_OPTIONS="-Djava.awt.headless=true -Djenkins.install.runSetupWizard=false" // Later
// Save the file
mkdir -p /var/lib/jenkins/init.groovy.d
nano /var/lib/jenkins/init.groovy.d/security.groovy

```
import jenkins.model.*
import hudson.security.*
import jenkins.security.s2m.AdminWhitelistRule
import hudson.security.csrf.DefaultCrumbIssuer
import jenkins.model.Jenkins

def instance = Jenkins.getInstance()

def hudsonRealm = new HudsonPrivateSecurityRealm(false)
hudsonRealm.createAccount("admin", "admin")
instance.setSecurityRealm(hudsonRealm)
```

```
def strategy = new FullControlOnceLoggedInAuthorizationStrategy()
strategy.setAllowAnonymousRead(false)
instance.setAuthorizationStrategy(strategy)
instance.save()
```

```
Jenkins.instance.getInjector().getInstance(AdminWhitelistRule.class)
```

```
def j = Jenkins.instance
if(j.getCrumbIssuer() == null) {
    j.setCrumbIssuer(new DefaultCrumbIssuer(true))
    j.save()
    println 'CSRF Protection configuration has changed.  Enabled CSRF Protection.'
}
else {
    println 'Nothing changed.  CSRF Protection already configured.'
}
```

// save this file

 # /var/lib/jenkins/init.groovy.d/installPlugins.groovy

```
import jenkins.model.*
import java.util.logging.Logger

def logger = Logger.getLogger("")
def isInstalled = false
def isInitialized = false

def pluginList = "ws-cleanup timestamper credentials-binding build-timeout antisamy-markup-formatter cloudbees-fold
er pipeline-stage-view pipeline-github-lib github-branch-source workflow-aggregator gradle ant mailer email-ext lda
p pam-auth matrix-auth ssh-slaves github git"

def plugins = pluginList.split()

def masterJenkinsInstance = Jenkins.getInstance()
def pluginManager = instance.getPluginManager()
def updateCenter = instance.getUpdateCenter() {

plugins.each {
  logger.info("Checking " + it)
  if (!pluginManager.getPlugin(it)) {
    logger.info("Looking UpdateCenter for " + it)
    if (!isInitialized) {
      updateCenter.updateAllSites()
      isInitialized = true
    }
    def plugin = updateCenter.getPlugin(it)
    if (plugin) {
      logger.info("Installing " + it)
```

chown -R jenkins:jenkins /var/lib/jenkins/init.groovy.d
systemctl stat jenkins
tail -f /var/lib/jenkins.log // Plugins will install with dependency
// Hit the Jenkins Browser, Wizard should not appeared, Login Page should appears, try Login via Admin and Play
// Manage Jenkins -> Manage Plugins -> All Plugins should installed

---

LAB-13:

AUTOMATION THROUGH PUPPET:

// yum.puppetlabs.com -> puppetlabs-release-e1-7.noarch.rpm -> copy the Link Address
rpm -ivh http://yum.puppetlabs.com/puppetlabs-release-el-7.noarch.rpm
yum -y install puppet-agent
// logout and login
mkdir puppet
cd puppet
mkdir modules
mkdir manifests
cd modules
mkdir jenkins
cd jenkins
mkdir manifests
mkdir files
cd manifests
nano install.pp

```
class jenkins::install {
    file { 'jenkins_repo':
        path   => '/etc/yum.repos.d/jenkins.repo',
        source => 'https://pkg.jenkins.io/redhat-stable/jenkins.repo',
        ensure => present,
        mode   => '0644'
    }

    exec { 'jenkins_repo_key':
        command   => '/bin/rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key',
        unless    => '/bin/rpm -q jenkins',
        subscribe => File['jenkins_repo'],
        require   => File['jenkins_repo']
    }

    package { 'epel-repo':
        name   => 'epel-release',
        ensure => present,
        source => 'https://dl.fedoraproject.org/pub/epel/7/x86_64/Packages/e/epel-release-7-11.noarch.rpm'
    }

    $packages = [ 'jenkins', 'java-1.8.0-openjdk', 'nginx', 'git' ]

    package { $packages:
        ensure    => present,
        require   => [ File['jenkins_repo'], Package['epel-repo'] ]
    }
```

 # nano config.pp

```
class jenkins::config {
    file { 'groovy_script_directory':
        path   => '/var/lib/jenkins/init.groovy.d',
        owner  => 'jenkins',
        group  => 'jenkins',
        mode   => '0755',
        ensure => directory
    }

    file { 'security_groovy_script':
        path    => '/var/lib/jenkins/init.groovy.d/security.groovy',
        owner   => 'jenkins',
        group   => 'jenkins',
        source  => 'puppet:///modules/jenkins/security.groovy',
        mode    => '0644',
        require => File['groovy_script_directory']
    }

    file { 'plugins_groovy_script':
        path    => '/var/lib/jenkins/init.groovy.d/installPlugins.groovy',
        owner   => 'jenkins',
        group   => 'jenkins',
        source  => 'puppet:///modules/jenkins/installPlugins.groovy',
        mode    => '0644',
        require => File['groovy_script_directory']
    }

    file { 'nginx_config_jenkins':
        path   => '/etc/nginx/conf.d/jenkins.conf',
```

```
file { 'nginx_config_jenkins':
    path   => '/etc/nginx/conf.d/jenkins.conf',
    owner  => 'root',
    group  => 'root',
    source => 'puppet:///modules/jenkins/jenkins.conf',
    mode   => '0644'
}

file { 'nginx_config':
    path   => '/etc/nginx/nginx.conf',
    owner  => 'root',
    group  => 'root',
    source => 'puppet:///modules/jenkins/nginx.conf',
    mode   => '0644'
}

file { 'jenkins_sysconfig':
    path   => '/etc/sysconfig/jenkins',
    owner  => 'root',
    group  => 'root',
    source => 'puppet:///modules/jenkins/jenkins',
    mode   => '0644'
}
```

 # nano service.pp

```
class jenkins::service {
    service { 'jenkins':
        ensure => running
    }

    service { 'nginx':
        ensure => running
    }
```

 # nano init.pp

```
class jenkins {
    include jenkins::install
    include jenkins::config
    include jenkins::service

    Class['jenkins::install'] -> Class['jenkins::config'] ~> Class['jenkins::service']
```

 # cd ../files/
 # nano installPlugins.groovy

```
logger.info("" + plugins)
def instance = Jenkins.getInstance()
def pm = instance.getPluginManager()
def uc = instance.getUpdateCenter()
plugins.each {
  logger.info("Checking " + it)
  if (!pm.getPlugin(it)) {
    logger.info("Looking UpdateCenter for " + it)
    if (!initialized) {
      uc.updateAllSites()
      initialized = true
    }
    def plugin = uc.getPlugin(it)
    if (plugin) {
      logger.info("Installing " + it)
        def installFuture = plugin.deploy()
      while(!installFuture.isDone()) {
        logger.info("Waiting for plugin install: " + it)
        sleep(3000)
      }
      installed = true
    }
  }
}
if (installed) {
  logger.info("Plugins installed, initializing a restart!")
  instance.save()
  instance.restart()
}
```

 # nano security.groovy

```
import jenkins.model.*
import hudson.security.*
import jenkins.security.s2m.AdminWhitelistRule
import hudson.security.csrf.DefaultCrumbIssuer
import jenkins.model.Jenkins

def instance = Jenkins.getInstance()

def hudsonRealm = new HudsonPrivateSecurityRealm(false)
hudsonRealm.createAccount("admin", "admin")
instance.setSecurityRealm(hudsonRealm)

def strategy = new FullControlOnceLoggedInAuthorizationStrategy()
strategy.setAllowAnonymousRead(false)
instance.setAuthorizationStrategy(strategy)
instance.save()

Jenkins.instance.getInjector().getInstance(AdminWhitelistRule.class)

def j = Jenkins.instance
if(j.getCrumbIssuer() == null) {
    j.setCrumbIssuer(new DefaultCrumbIssuer(true))
    j.save()
    println 'CSRF Protection configuration has changed.  Enabled CSRF Protection.'
}
else {
    println 'Nothing changed.  CSRF Protection already configured.'
}
```

 # nano jenkins

// JENKINS_ARGS=“”
// SAVE THE FILE

 # nano jenkins.config

```
upstream jenkins {
    server 127.0.0.1:8080;
}

server {
    listen      80 default;
    server_name jenkins.course;

    access_log  /var/log/nginx/jenkins.access.log;
    error_log   /var/log/nginx/jenkins.error.log;

    proxy_buffers 16 64k;
    proxy_buffer_size 128k;

    location / {
        proxy_pass  http://jenkins;
        proxy_next_upstream error timeout invalid_header http_500 http_502 http_503 http_504;
        proxy_redirect off;

        proxy_set_header    Host            $host;
        proxy_set_header    X-Real-IP       $remote_addr;
        proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header    X-Forwarded-Proto $scheme;
    }

}
```

 # nano nginx.conf

```
# Load dynamic modules. See /usr/share/nginx/README.dynamic.
include /usr/share/nginx/modules/*.conf;

events {
    worker_connections 1024;
}

http {
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 2048;

    include             /etc/nginx/mime.types;
    default_type        application/octet-stream;

    # Load modular configuration files from the /etc/nginx/conf.d directory.
    # See http://nginx.org/en/docs/ngx_core_module.html#include
    # for more information.
    include /etc/nginx/conf.d/*.conf;
```

 # cd ../../../
 # pwd // puppet directory
 # nano manifests/site.pp

```
node default {
    include jenkins
}
```

 # puppet apply - -modulepath ./modules manifests/site.pp // wait for the compilation

// http://jenkins.course/login

LAB-14:

CREATING BUILD JOBS FROM USER INTERFACE AND SCRIPTS:

Create a small program and writing test for it
Create a basic Jenkins job from the user interface
Learning about Jenkins API call
Using the API call to create a Jenkins Job

Steps:
// create a repository in GitHub and clone in to your Local
 # git clone <ssh-url>
 # cd <project-name>
 # nano account.py

```
class account:
    def check_password_length(self, password):
        if len(password) > 8:
            return True
        else:
            return False

if __name__ == '__main__':
    accVerify = account()
    print('The password length is ' + str(accVerify.check_password_length('offtoschool')))
```

 # nano account_test.py

```
import unittest
import account as AccountClass

class Test(unittest.TestCase):
    accInfo = AccountClass.account()

    def test_check_password_length(self):
        print("Checking possible passwords\n")
        passwordList = [ 'abeautifulday', 'astrictboss', 'alovelyhouse' ]

        for passwd in passwordList:
            print("Checking password " + passwd + "\n")
            passInfo = self.accInfo.check_password_length(passwd)
            self.assertTrue(passInfo)

if __name__ == '__main__':
    unittest.main()
```

 # python account_test.py // execute the script
 # git add -A
 # git commit -m “python code”
 # git push origin master // push the code to the repo
// Create a Free Style project in Jenkins
// Add the git url
// Add the credentials
// Add “python *test.py” under Execute Shell
// Save

// Click on Build and See the results

LAB-15:

 # cp ../sample-python/config.xml . // copy the config.xml from previous execution job
 # pwd // /Users/javedalam/.jenkins/jobs/jenkins_cmd
 # nano config.xml
// Add the below content at the end

```
<publisher>
    <hudson.plugins.ws_cleanup.WsCleanup plugin="ws-cleanup@0.34">
      <patterns class="empty-list"/>
      <deleteDirs>true</deleteDir>
      <skipWhenFailed>false</skipWhenFailed>
      <cleanWhenSuccess>true</cleanWhenSuccess>
      <cleanWhenUnstable>true</cleanWhenUnstable>
      <cleanWhenFailure>true</cleanWhenFailure>
      <cleanWhenNotBuilt>true</cleanWhenNotBuilt>
      <cleanWhenAborted>true</cleanWhenAborted>
      <notFailBuild>false</notFailBuild>
      <cleanupMatrixParent>false</cleanupMatrixParent>
      <externalDelete></externalDelete>
    </hudson.plugins.ws_cleanup.WsCleanup>
  </publishers>
  <buildWrappers/>
</project>
```

// Generate the crumb to handle Jenkins via CLI

// Not working
 # curl -s 'http://localhost:8080/crumbIssuer/api/xml?xpath=concat(//crumbRequestField,":",//crumb)' -u xavy:redhat@123

 # curl -u "admin:admin" 'http://localhost:8080/crumbIssuer/api/xml?xpath=concat(//crumbRequestField,":",//crumb)'
 # curl -u "xavy:redhat@123" 'http://localhost:8080/crumbIssuer/api/xml?xpath=concat(//crumbRequestField,":",//crumb)'

 # wget 'http://localhost:8080/crumbIssuer/api/xml?xpath=concat(//crumbRequestField,":",//crumb)' -u xavy:redhat@123
// Output Should be like below
// Jenkins-Crumb:e2e41f670dc128f378b2a010b4fcb493

OR
// Not Working:
req = requests.get('http://JENKINS_URL/crumbIssuer/api/xml?xpath=concat(//crumbRequestField,":",//crumb)', auth=(username, password))
req = requests.get('http://http://localhost:8080/crumbIssuer/api/xml?xpath=concat(//crumbRequestField,":",//crumb)', auth=(xavy, redhat@123))

Troubleshooting STEPS:
// LIST THE ERRORS:
bash -c 'http://localhost:8080/crumbIssuer/api/xml?xpath=concat(//crumbRequestField,":",//crumb)' -u xavy:redhat@123
-u: -c: line 0: syntax error near unexpected token `//crumbRequestField,":",//crumb'
-u: -c: line 0: `http://localhost:8080/crumbIssuer/api/xml?xpath=concat(//crumbRequestField,":",//crumb)'

VVImp Concept in Jenkins:
Execute the command to create a Job from an exiting Job which automatically clean the workspace:
curl -s -XPOST 'http://localhost:8080/createItem?name=python-project-new' -u admin:admin - -data-binary @config.xml -H "Jenkins-Crumb:e2e41f670dc128f378b2a010b4fcb493" -H "Content-Type:text/xml" // Open your eyes

---

LAB-16:

SKIPPED

LAB-X:

INTEGRATING JENKINS WITH EXTERNAL SERVICES:

INTEGRATING WITH GITHUB
INTEGRATING WITH SONARQUBE // FOR CONTINUOS CODE INSPECTION
INTEGRATING WITH ARTIFACTORY // FOR BUILD BACKUP

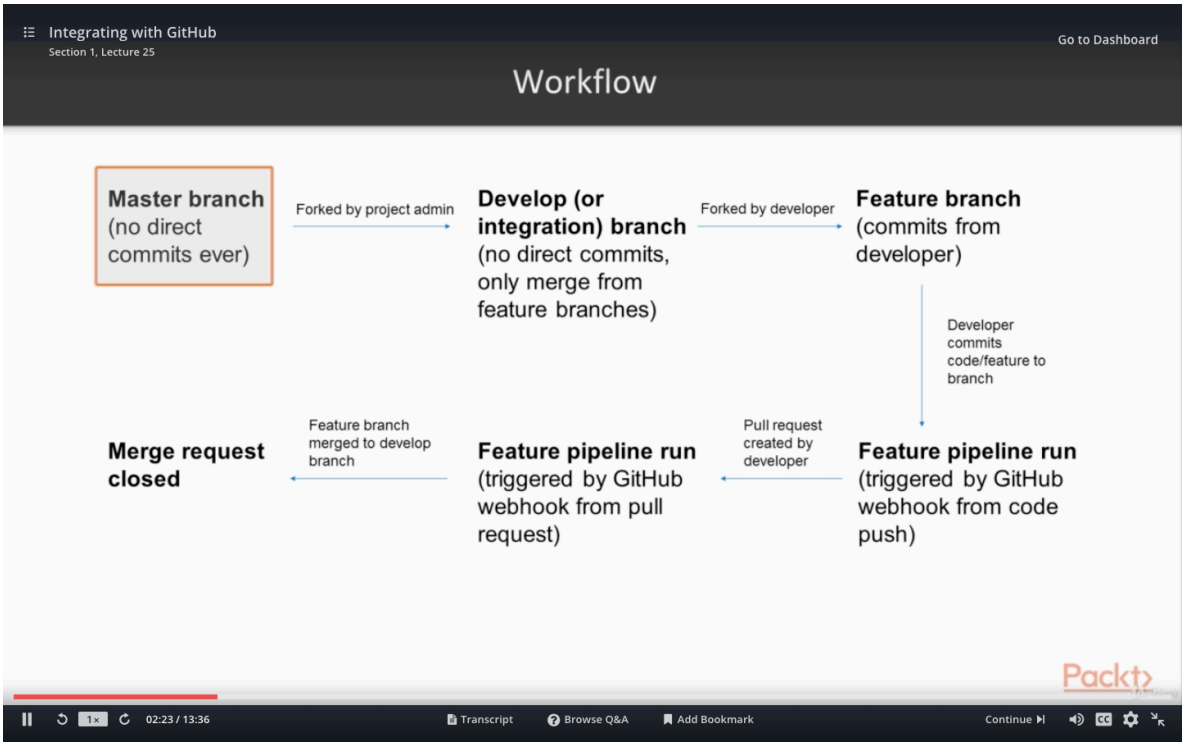
INTEGRATING WITH JIRA // FOR ISSUE TRACKER
INTEGRATING WITH SLACK // NOTIFICATION SOLUTION

INTEGRATING WITH GITHUB:

INTEGRATING WITH GITHUB // UNSUCCESSFUL

Explore the Plugins required
Preparing GitHub and configuring Jenkins to work together
Configuring the Pipeline to create a code build workflow
Testing the Pipeline with different Scenarios

WORKFLOW:



Steps:

IN GITHUB:
// GitHub -> Settings -> Developer Settings -> Personal Access Tokens -> Click on "Generate New Token"
// Token Description: IntergrationWithGitHub
// Select "repo", "admin:org_hook" and "admin:repo_hook" permissions
// Click on "Generate token"
// cb3bc36b0f8fc7a7aa48d4e90dbe4681a1c9cd50 // Done

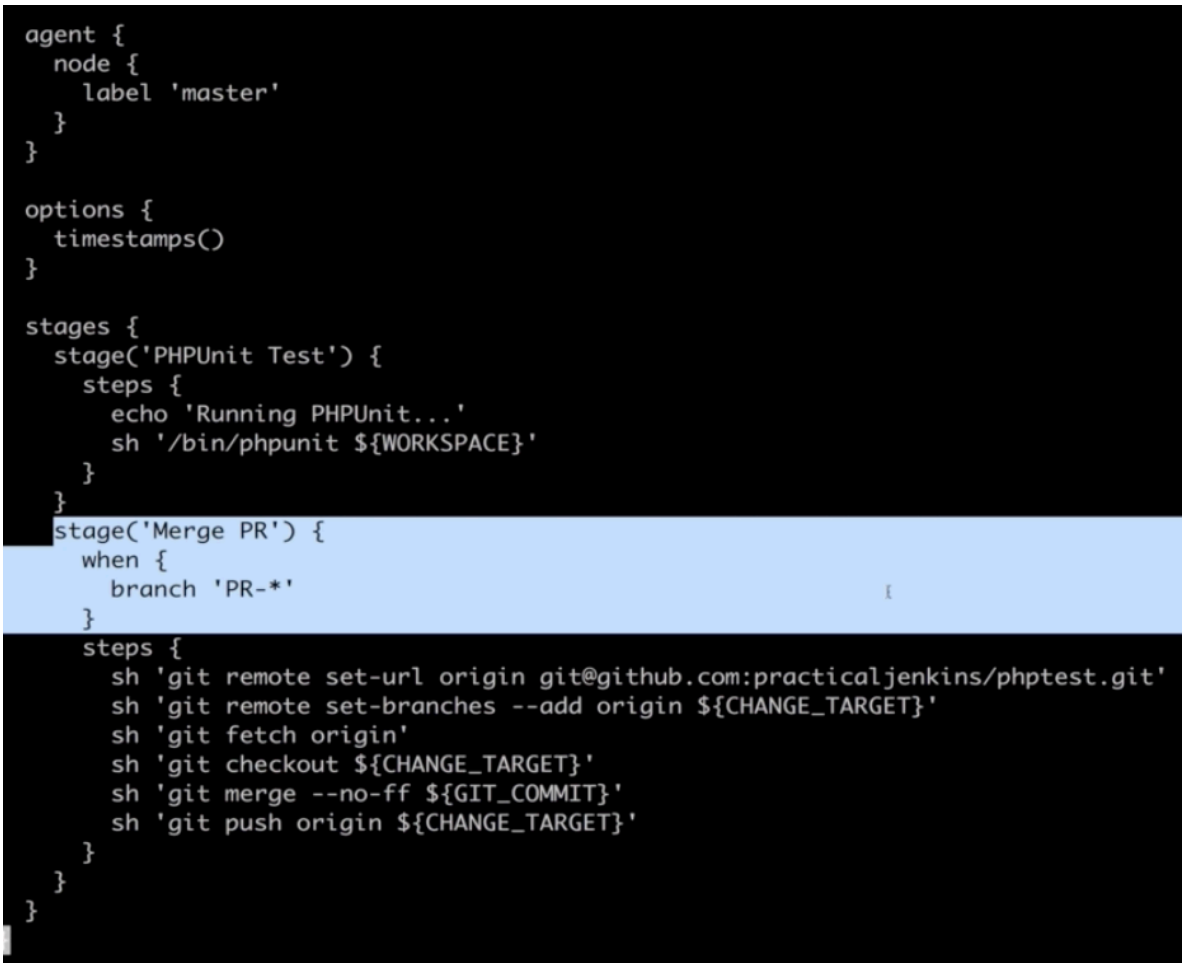
IN JENKINS:
// Jenkins -> Credentials -> Jenkins -> Global Credentials ->

// Add Credentials -> Kind: Username -> IntergrationWithGitHub; Password: cb3bc36b0f8fc7a7aa48d4e90dbe4681a1c9cd50; ID: jenkins-secret
// Add Credentials -> Kind: Secret text; Secret: cb3bc36b0f8fc7a7aa48d4e90dbe4681a1c9cd50; ID: jenkins-token
// Done

IN JENKINS:
// Manage Jenkins -> Configuration Security -> GitHub -> Add GitHub Server -> GitHub Server
// Name: IntergrationWithGitHub
// Credentials: Jenkins-token
// Click on Test Connection
// Select the checkbox for "Manage Hooks" -> Helps in Web Hooks
// Click on Save

IN JENKINS:
// Install the Plugin: "GitHub Branch Source Plugin"

IN CLI:
 # git clone git@github.com:practicaljenkins/sample-php-project.git
 # nano Jenkinsfile



```
agent {
  node {
    label 'master'
  }
}

options {
  timestamps()
}

stages {
  stage('PHPUnit Test') {
    steps {
      echo 'Running PHPUnit...'
      sh '/bin/phpunit ${WORKSPACE}'
    }
  }
  stage('Merge PR') {
    when {
      branch 'PR-*'
    }
    steps {
      sh 'git remote set-url origin git@github.com:practicaljenkins/phptest.git'
      sh 'git remote set-branches --add origin ${CHANGE_TARGET}'
      sh 'git fetch origin'
      sh 'git checkout ${CHANGE_TARGET}'
      sh 'git merge --no-ff ${GIT_COMMIT}'
      sh 'git push origin ${CHANGE_TARGET}'
    }
  }
}
```

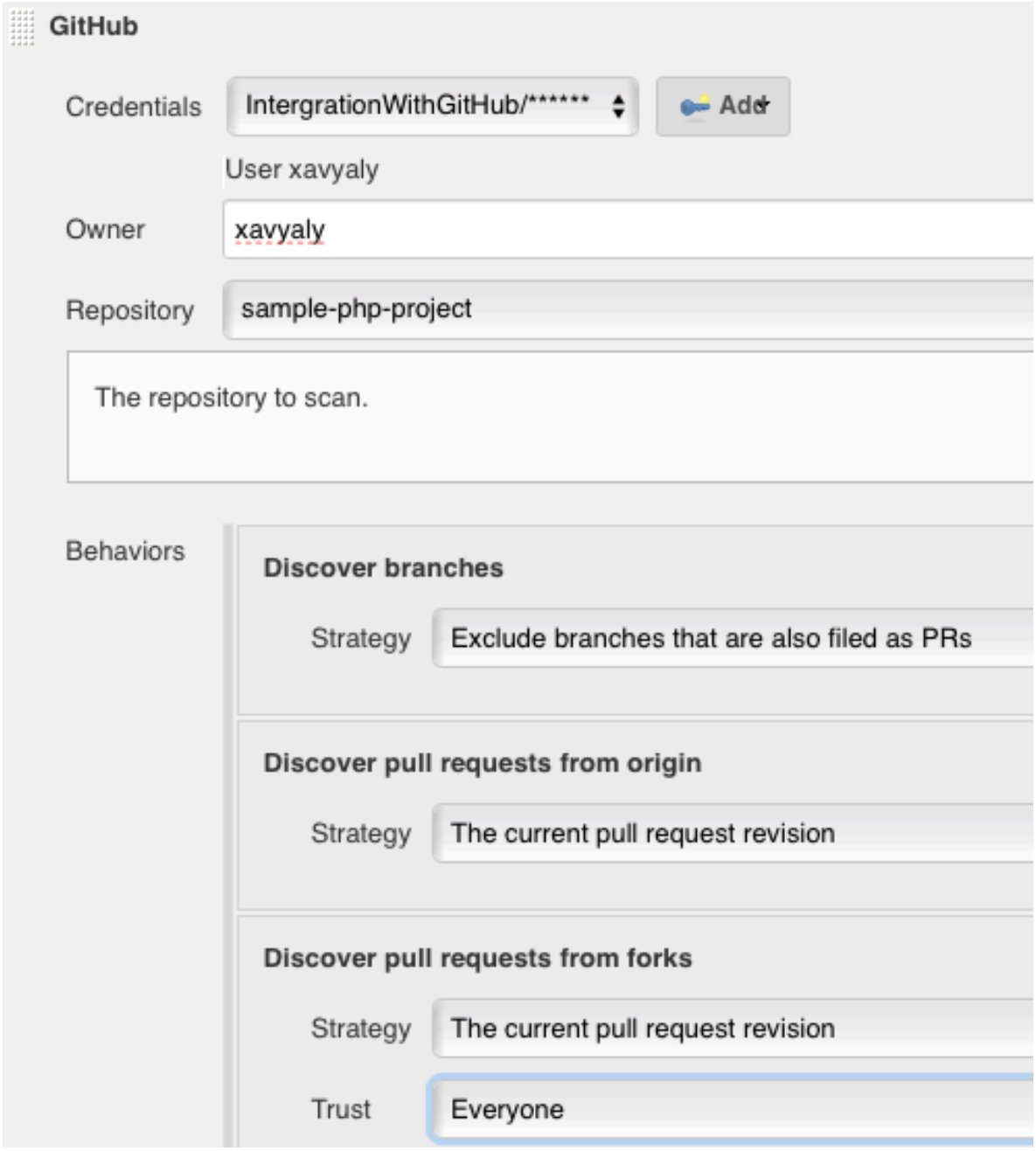
OR

```
pipeline {

  agent {
    node {
      label 'master'
    }
  }

  options {
    timestamps()
  }

  stages {
    stage('PHPUnit Test') {
      steps {
        echo 'Running PHPUnit...'
        sh '/bin/phpunit ${WORKSPACE}'
      }
    }
    stage('Merge PR') {
      when {
        branch 'PR-*'
      }
      steps {
        sh 'git remote set-url origin git@github.com:practicaljenkins/phptest.git'
        sh 'git remote set-branches - -add origin ${CHANGE_TARGET}'
        sh 'git fetch origin'
        sh 'git checkout ${CHANGE_TARGET}'
        sh 'git merge - -no-ff ${GIT_COMMIT}'
        sh 'git push origin ${CHANGE_TARGET}'
      }
    }
  }
}
```

IN GIT:
// Fork the Project "https://github.com/xavyaly/sample-php-project"
 # git clone https://github.com/xavyaly/sample-php-project
 # git branch // master*
 # git checkout -b develop
 # git add .
 # git commit -m "added a new php project"
 # git push origin develop

IN JENKINS:
// Create a new Job “sample-php-project”
// Select the “Multibranch Pipeline”
// Select “OK”
// Branch Sources: GitHub
// Credentials: IntegrationWithGitHub/***
// Owner: xavyaly
// Repository: sample-php-project
// Save // Scan Repository Log -> SUCCESS

Sample:



// Save to See the changes

// Click on develop branch // Deep dive

WEBHOOK VERIFICATION:
// GitHub -> <sample-php-project> -> Settings -> Webhooks ->
// http://localhost:8080/github-webhook/ (Generated Automatically) -> Click on Link
// Pull and Push both supports

IN GIT:
git checkout -b featurebranch
nano src/ConnectTest.php // change from google to yahoo
git add .
git commit -m "update the code"
git push origin featurebranch

IN JENKINS:
// Need to change the Jenkinsfile unless new branch will not been appeared under Jenkins Job
// Create some new branches and push the code, you can see the changes
// Raise a PULL Requests and Compare the code from different branches
// Done

LAB-X:

INTEGRATING JENKINS WITH EXTERNAL SERVICES:

INTEGRATING WITH SONARQUBE // FOR CONTINUOS CODE INSPECTION

SETTING UP THE PREREQUISITES OF JENKINS
INSTALL AND CONFIGURE SONARQUBE PLUGIN
CONFIGURE JENKINS PIPELINE FOR SONARQUBE ACTIONS
GENERATE SONARQUBE ANALYSIS REPORT FROM JENKINS PIPELINE

// Expalanation of SonarQube: Quality Analysis Tool

Steps:

SonarQube UI:
// https://www.sonarqube.org/downloads/ -> COMMUNITY EDITION 7.4 -> sonarqube-7.4.zip
pwd // /Users/javedalam/Documents/DevOpsLabs/

SonarQube4Days-3/sonarqube-7.4/bin/macosx-universal-64
./sonar.sh start // Start this script
// http://localhost:9000 // 9000 Sonarqube default port
// Default Credentials?
// admin/admin

// If wanted to Change the default port
pwd // /Users/javedalam/Documents/DevOpsLabs/SonarQube4Days-3/sonarqube-7.4/conf

SonarQube SignUp:
// Accont -> Security -> Generate New token "SonarQubeImplementation"

// Hold your token "aurefbskfsf9943y2g" for future use

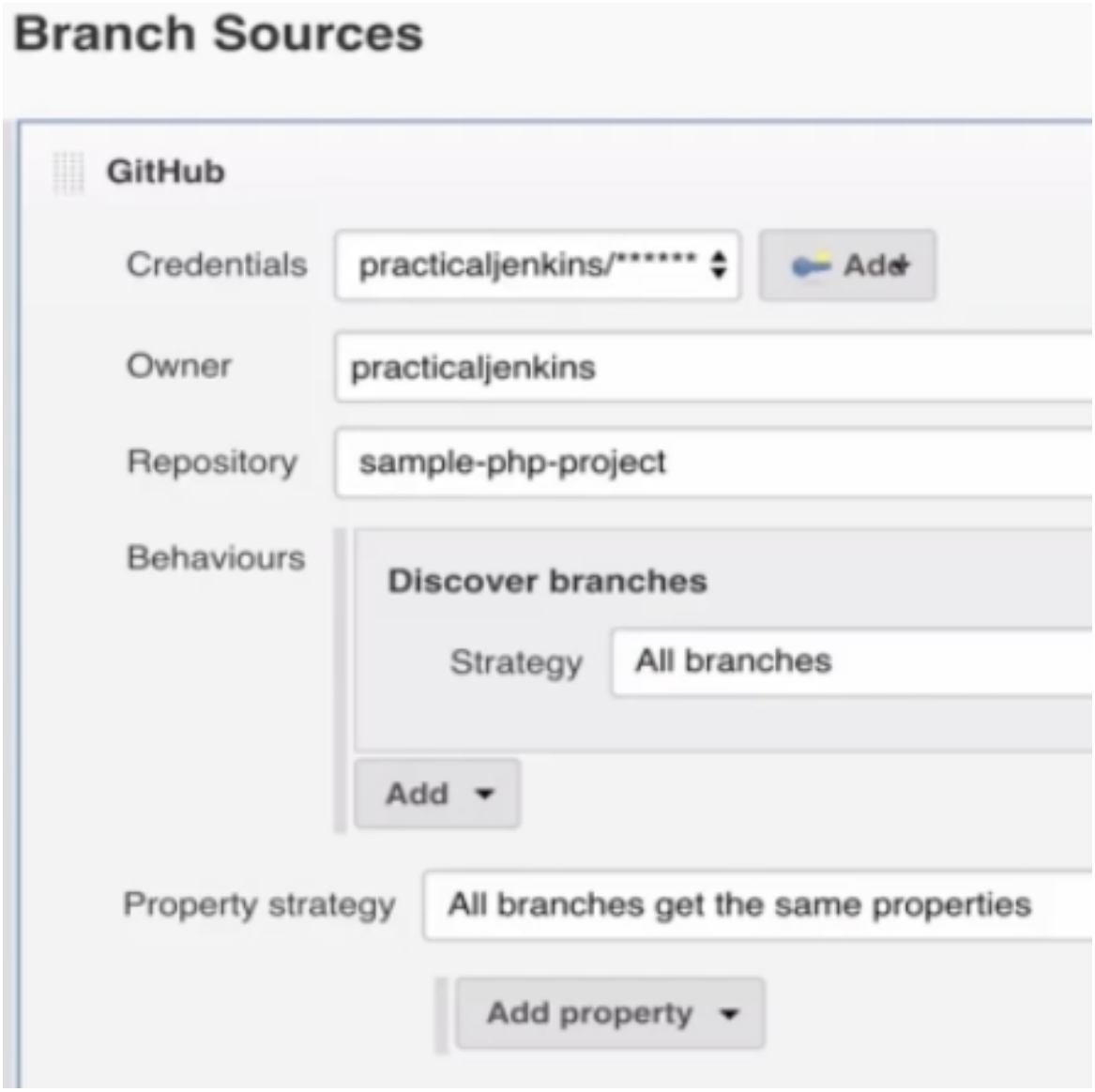
Jenkins:
// Manage Jenkins -> Manage Plugins -> Install "SonarQube Scanner" Plugin
// Restart the Jenkins
// Manage Jenkins -> Configure -> Search for "SonarQube servers"
// Select the checkbox for "Enable injection of SonarQube server configuration as build environment variables"
// Click on Add SonarQube // It expanded
// Name: SonarQubeImplementation
// Server URL: SonarQube Server URL like "http://<ip>"
// Server authentication token: aurefbskfsf9943y2g
// Click on Save the configuration

Installation:
// Download the SonarQube Scanner // Scanner is for CLI
// https://www.sonarqube.org/downloads/ -> Documentation and Download -> Analyzing with SonarQube Scanner ->
// Mac OS X 64 bit -> https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-3.2.0.1227-macosx.zip
# cd /opt
$ sudo wget https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-3.2.0.1227-macosx.zip
$ sudo unzip sonar-scanner-3.2.0.1227-macosx
$ sudo mv sonar-scanner-3.2.0.1227-macosx sonarqube-scanner
# ls -l sonarqube-scanner/bin/sonar-scanner
# yum install php-phpunit-PHPUnit // For Linux
# brew install phpunit // MacOs; https://stackoverflow.com/questions/3301300/setting-up-phpunit-on-osx
# cd ~/Documents/MyGit/sample-php-project/
# cp Jenkinsfile ../
# nano Jenkinsfile // Edit the Jenkinsfile

```
pipeline {
  agent {
    node {
      label 'master'
    }
  }

  options {
    timestamps()
  }

  stages {
    stage('PHPUnit Test') {
      steps {
        echo 'Running PHPUnit...'
        sh '/bin/phpunit ${WORKSPACE}'
      }
    }
    stage('SonarQube analysis') {
      steps {
        withSonarQubeEnv('Practical Jenkins Sonarqube') {
          sh 'echo "sonar.projectKey=production:phptest" > ${WORKSPACE}/sonar-project.properties'
          sh 'echo "sonar.sources=." >> ${WORKSPACE}/sonar-project.properties'
          sh '/opt/sonarqube-scanner/bin/sonar-scanner'
        }
      }
    }
  }
```

 # git add -A
 # git commit -m “updated the Jenkinsfile”
 # git push origin master

JENKINS:
// Create a Multibranch pipeline Job



// Click on Save // SUCCESS

SONARQUBE:
// http://<ip>/projects // List with Project name and details

SONARQUBE Plugins Details:
// https://docs/sonarqube.org/display/SCAN/Analyzing+with+SonarQube+Scanner+for+Jenkins

---