JENKINS TASKS

BY ANJU M DOMINIC

TASK (PRE-REQUISITE): LAUNCH AN EC2 INSTANCE

- Activity:
  - Launch an ubuntu 18.04 EC2 instance in AWS

#	Steps
1	Go to Ec2 services in AWS Management console
2	Select Launch Instance “t2.xlarge”
3	Choose an Amazon Machine Image (AMI) as Ubuntu Server 18.04 LTS (HVM), SSD Volume Type
4	Go to Configure Security Group and enable access to following ports from anywhere Ssh ->TCP->22 Http ->TCP -> 80 All Tcp ->TCP ->0-65535 All UDP -> UDP ->0-65535 HTTPS->TCP ->443
5	Click Next and go to review and launch
6	Click Launch and select a key
7	View instance
8	Make a note of Public DNS and Public IP

TASK (PRE-REQUISITE): ACCESS AN EC2 INSTANCE AND GET THE JENKINS WORKBOOK CLONED

- Activity:
  - Access the launched EC2 instance and clone the Jenkins Workbook

#	Steps	Commands
1	Go to EC2 services in AWS Management console	
2	Select the instance and click connect	
3	Copy and run the command under example in Git Bash. Make sure you are on the same folder where the key is downloaded	ssh -i “Key Name" <Public DNS name of the Ec2 instance>
4	Clone the Jenkins Workbook	git clone https://github.com/AnjuMeleth/DevOpsMasterJenkins.git

TASK 1: HOW TO INSTALL JENKINS

- Activity:
  - In order to create a continuous delivery pipeline for the application, Install Jenkins on a Ubuntu 18.04 system thus making it your build system.

- Definition of Done:
  - Get the version of Jenkins installed in config.xml
    - cd /var/lib/Jenkins
    - cat config.xml

#	Steps	Commands
1	Go to installation/code directory	cd DevOpsMasterJenkins/installation/code
2	Run the installation script	sh JenkinsInstall.sh
3	If getting error run the command to get the key mentioned in the error	sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 9B7D32F2D50582E6
4	Run the script again	sh JenkinsInstall.sh
5	Go to Jenkins home directory	cd /var/lib/jenkins
6	Cat the config.xml file	cat config.xml

TASK 2: HOW TO CONFIGURE JENKINS

- Activity:
  - Make sure the installed Jenkins is configured to be used as a build server.
- Steps:
  - Access Jenkins using browser

    http://<Ipaddress>:8080
  - Unlock Jenkins using password in /var/lib/jenkins/secrets/initialAdminPassword

    sudo cat /var/lib/jenkins/secrets/initialAdminPassword
  - Install suggested plugins
  - Create admin user
- Definition of Done:
  - Create a JenkinsTask GitHub Repository
  - Create a folder Task2
  - Upload the screenshot of Jenkins dashboard

#	Steps	Commands
1	Access installed Jenkins through browser	http://<Ipaddress>:8080 Use the public IP address of the Ec2 instance
2	Unlock Jenkins using password in /var/lib/jenkins/secrets/initialAdminPassword	sudo cat /var/lib/jenkins/secrets/initialAdminPassword
3	Install suggested plugins	
4	Create admin user	
5	Follow the installer to start using jenkins	

TASK 3: HOW TO CREATE FIRST JENKINS JOB

- Activity:
  - Lets now create a job in Jenkins HelloWorld. Create a freestyle project.
- Steps:
  - Go to new item.
  - Create a Freestyle project and name it as HelloWorld
  - Select “Execute shell” under “Build” section.
  - Enter the commands

```
echo "Hello World "
echo "Welcome to Jenkins“
```

  - Save and Build Now
- Definition of Done:
  - Upload the screenshot of Jenkins dashboard -> Console Output to the GitHub repository under the folder Task3

#	Steps	
1	Create a Freestyle project and name it as HelloWorld	
2	Select “Execute shell” under “Build” section	
3	Enter the commands	echo "Hello World " echo "Welcome to Jenkins“
4	Save and build now	

TASK 4 : HOW TO CREATE A DELIVERY PIPELINE FOR A JAVA APPLICATION

- Activity:
  - Build a delivery pipeline using Jenkins. The source code is kept in https://github.com/AnjuMeleth/hello-world-java.git
- Steps:
  - Create a Freestyle project and name it as HelloWorldJava
  - Select "Execute shell" under "Build" section.
  - Enter the commands

```
git clone https://github.com/AnjuMeleth/hello-world-java.git
cd hello-world-java
javac HelloWorld.java
java HelloWorld
```

  - Save and Build Now

- Definition of Done:
  - Upload the screenshot of Jenkins dashboard -> console Output to the GitHub repository under the folder Task4

TASK 5: HOW TO INTEGRATE JENKINS WITH GIT

- Activity:
  - We have created a simple delivery pipeline in previous task using the source code kept in

    https://github.com/AnjuMeleth/hello-world-java.git
- Steps:
  - Create a Freestyle project and name it as HelloWorldJava
  - Provide the repository path in Source Code Management section
  - Select "Execute shell" under "Build" section.
  - Enter the commands

    ```
    javac HelloWorld.java
    java HelloWorld
    ```
  - Save and Build Now
  - If faced with error you may have to enable "Delete Workspace before the build starts" under "Build Environment"
- Definition of Done:
  - Upload the screenshot of Jenkins dashboard -> console Output to the GitHub under the folder Task5

TASK 6: TRIGGER JENKINS JOB AT A SCHEDULED TIME

- Activity:
  - Create a Jenkins job that gets triggered every 1 minute. The source code is kept in

    https://github.com/AnjuMeleth/hello-world-java.git
- Steps:
  - Create a Freestyle project and name it as HelloWorldCron
  - Provide the repository path in Source Code Management section
  - Enable Build periodically under Build Triggers.Provide the schedule as */1 * * * *
  - Select "Execute shell" under "Build" section.
  - Enter the commands

    ```
    javac HelloWorld.java
    ```
  - Save the configuration

- Definition of Done:
  - We can view history of Builds in the Jenkins Dashboard

TASK 7: HOW TO DISABLE A JENKINS JOB

- Activity:
  - HelloWorldCron is no longer used and hence we need to disable the job.
- Steps:
  - Enable Disable this project under General Tab. Save the configuration.
- Definition of Done:
  - Project shown as disabled in Jenkins dashboard.

TASK 8: TRIGGER JENKINS JOB FROM GIT (DEMO)

- Activity:
  - As a DevOps practitioner you need to trigger a Jenkins job whenever there is a push in the GitHub repository. The source code is kept in

    https://github.com/AnjuMeleth/hello-world-java.git
- Steps:
  - Create a Freestyle project and name it as HelloWorldGitTrigger
  - Provide the repository path in Source Code Management section. You may have to fork the repository and give that URL in the source code management section
  - Enable GitHub hook trigger for GITScm polling under Build Triggers.
  - Select "Execute shell" under "Build" section.
  - Enter the commands

    ```
    javac HelloWorld.java
    ```
  - Save the configuration
  - Enable webhook in GitHub, under settings ->Webhook. Webhook could be found from Manage Jenkins -> Configure System -> GitHub ->Advanced -> Enable Specify another hook URL for GitHub configuration

    http://<ip address>:8080/github-webhook/
- Definition of Done:
  - Upload the screenshot of Jenkins dashboard -> console Output to the GitHub under the folder Task6

TASK 9 :HOW TO CONFIGURE MAVEN IN JENKINS

- Activity:
  - Configure Jenkins to install MAVEN
- Steps:
  - Go to Manage Jenkins -> Global Tool configuration -> Maven. Provide the installation name and select the version for automatic install.
- Definition of Done:
  - None

TASK 10 : HOW TO BUILD A COMPLEX JAVA APPLICATION

- Activity:
  - Build a spring pet clinic application using Jenkins by a continuous delivery pipeline
- Steps:
  - Create a Freestyle project and name it as PetClinicCompile
  - Provide the repository path in Source Code Management section :

    https://github.com/AnjuMeleth/spring-petclinic.git
  - Select “Invoke top level maven target” under “Build” section. Set the goal as “compile” under the correct installation
  - Save the configuration and Build Now
- Definition of Done:
  - Upload the screenshot of Jenkins dashboard -> Output to the GitHub under the folder Task10

TASK 11 : TESTING A COMPLEX JAVA APPLICATION

- Activity:
  - Test a spring pet clinic application using Jenkins
- Steps:
  - Create a Freestyle project and name it as PetClinicTest
  - Provide the repository path in Source Code Management section :

    https://github.com/AnjuMeleth/spring-petclinic.git
  - Select “Invoke top level maven target” under “Build” section. Set the goal as “test” under the correct installation
  - Save the configuration and Build Now
- Definition of Done:
  - Upload the screenshot of Jenkins dashboard -> console Output to the GitHub under the folder Task11

TASK 12 : DISCARD OLD BUILDS(ADMIN TASK)

- Activity:
  - You have lot of builds happening that causes memory issues in the Jenkins server. Limit the number of builds to 1 and delete other builds. Also delete builds older than one day.
- Steps:
  - Go to configure in PetClinicTest job.
  - Enable discard old builds in General tab. Give number of builds as two and delete builds older than one day.
- Definition of Done:
  - None

TASK 13 : CLEANUP THE WORKSPACE(ADMIN TASK)

- Activity:
  - As a best practice you need to clean up the workspace before running a new task.
- Steps:
  - Go to configure in PetClinicTest job.
  - Enable “Delete workspace before build starts” in “Build environment”.
  - Workspace could be accessed at /var/lib/Jenkins/workspace/PetClinicTest
- Definition of Done:
  - None

TASK 14 : HOW TO INSTALL A PLUGIN IN JENKINS

- Activity:
  - Install a “Build pipeline” plugin in Jenkins.
- Steps:
  - Go to Manage Jenkins -> Manage Plugins -> Available
  - Select Build Pipeline Plugin and install without restart
  - Go back to the top page is clicked
- Definition of Done:
  - Build Pipeline could be viewed under the Installed tab

TASK 15 :HOW TO BUILD A TWO STAGE PIPELINE – BUILD AND TEST

- Activity:
  - Create a pipeline for spring pet clinic application with two stages , build and test
- Steps:
  - Trigger PetClinicTest after the PetClinicCompile job using “Build after other projects are built” in “Build Trigger”
  - Create a pipeline view using Build Pipeline plugin
- Definition of Done:
  - Upload the screenshot of pipeline view to the GitHub under the folder Task15

TASK 16 : HOW TO PACKAGE CODE OF A COMPLEX JAVA APPLICATION

- Activity:
  - Package code of a spring pet clinic application using Jenkins
- Steps:
  - Create a Freestyle project and name it as PetClinicPackage
  - Provide the repository path in Source Code Management section :

    https://github.com/AnjuMeleth/spring-petclinic.git
  - Select "Invoke top level maven target" under "Build" section. Set the goal as "package" under the correct installation
  - Save the configuration and Build Now
- Definition of Done:
  - Upload the screenshot of Jenkins dashboard -> console output of the project to the GitHub under the folder Task16

TASK 17 : INSTALL A PLUGIN IN JENKINS

- Activity:
  - Install a “HTML Publisher” plugin in Jenkins.
- Steps:
  - Go to Manage Jenkins -> Manage Plugins -> Available
  - Select HTML Publisher and install without restart
- Definition of Done:
  - HTML Publisher could be viewed under the Installed tab

TASK 18 : PUBLISH HTML REPORT OF A COMPLEX JAVA APPLICATION

- Activity:
  - Publish HTML reports of a spring pet clinic application using Jenkins
- Steps:
  - Create a Freestyle project and name it as PetClinicReports
  - Provide the repository path in Source Code Management section :

    https://github.com/AnjuMeleth/spring-petclinic.git
  - Select "Invoke top level maven target" under "Build" section. Set the goal as "verify" under the correct installation
  - Enable Publish HTML Reports under Post-Build actions. Provide the HTML directory as target/site/jacoco and index page as index.html
  - Save the configuration and Build Now
- Definition of Done:
  - Upload the screenshot of Jenkins dashboard of the project to the GitHub under the folder Task18

TASK 19 : INSTALL A PLUGIN IN JENKINS

- Activity:
  - Install a “Warnings Next Generation” plugin in Jenkins.
- Steps:
  - Go to Manage Jenkins -> Manage Plugins -> Available
  - Select Warnings Next Generation and install without restart
- Definition of Done:
  - Warnings Next Generation could be viewed under the Installed tab

TASK 20 : HOW TO RECORD WARNINGS IN JENKINS PIPELINE

- Activity:
  - In the previous Jenkins job PetClinicTest, create a report on the warnings generated
- Steps:
  - Select "Record compiler warnings and static analysis results" from Post build actions.
  - Select Maven as tool.
  - Mention the report file as “**/target/surefire-reports/*.xml”
  - Save the configuration and Build Now
- Definition of Done:
  - Upload the screenshot of Jenkins dashboard of the project to the GitHub under the folder Task20

TASK 21 :HOW TO BUILD A MULTI-STAGE PIPELINE

- Activity:
  - Create a pipeline for spring pet clinic application with stages build , test , verify and package
- Steps:
  - Trigger PetClinicTest after the PetClinicCompile job using “Build after other projects are built” in “Build Trigger”. Similarly trigger “PetClinicPackage” after “PetClinicTest” and “PetClinicReports” after “PetClinicPackage”
  - Create a pipeline view using Build Pipeline plugin
- Definition of Done:
  - Upload the screenshot of pipeline view to the GitHub under the folder Task21

TASK 22 : SCRIPTED PIPELINE FOR A SIMPLE HELLO WORLD

- Activity:
  - Create a scripted pipeline for a HelloWorld Application
- Steps:
  - Create a pipeline project ScriptedHelloWorld. Code is available in

DevOpsMasterJenkins/ScriptedHelloWorld/pipeline.txt

- Definition of Done:
  - Upload the screenshot of Jenkins dashboard of the project to the GitHub under the folder Task22

TASK 23 : INSTALL A PLUGIN IN JENKINS (ADMIN TASK)

- Activity:
  - Install a “Pipeline Maven Integration” plugin in Jenkins.
- Steps:
  - Go to Manage Jenkins -> Manage Plugins -> Available
  - Select Pipeline Maven Integration and install without restart
- Definition of Done:
  - Pipeline Maven Integration could be viewed under the Installed tab

TASK 24 :SCRIPTED PIPELINE FOR BUILDING A COMPLEX JAVA APPLICATION

- Activity:
  - Create a scripted pipeline to build a spring pet clinic application
- Steps:
  - Create a pipeline project ScriptedPetClinic
  - Code is available in DevOpsMasterJenkins/ScriptedPetClinic/pipeline.txt
- Definition of Done:
  - Upload the screenshot of Jenkins dashboard of the project to the GitHub under the folder Task24

TASK 25 :SCRIPTED PIPELINE FOR BUILDING AND TESTING A COMPLEX JAVA APPLICATION

- Activity:
  - Create a scripted pipeline to build and test a spring pet clinic application
- Steps:
  - Create a pipeline project ScriptedPetClinic
  - Code is available in DevOpsMasterJenkins/ScriptedPetClinic/pipeline.txt
- Definition of Done:
  - Upload the screenshot of Jenkins dashboard of the project to the GitHub under the folder Task25

TASK 26 :SCRIPTED PIPELINE FOR BUILDING, TESTING AND PACKAGE A COMPLEX JAVA APPLICATION

- Activity:
  - Create a scripted pipeline to build, test and package a spring pet clinic application
- Steps:
  - Create a pipeline project ScriptedPetClinic
  - Code is available in DevOpsMasterJenkins/ScriptedPetClinic/pipeline.txt
- Definition of Done:
  - Upload the screenshot of Jenkins dashboard of the project to the GitHub under the folder Task26

TASK 27 :SCRIPTED PIPELINE FOR BUILDING, TESTING PACKAGE AND VERIFY A COMPLEX JAVA APPLICATION

- Activity:
  - Create a scripted pipeline to build, test , package and verify a spring pet clinic application. Also create HTML Reports for the application.
- Steps:
  - Create a pipeline project ScriptedPetClinic
  - Code is available in DevOpsMasterJenkins/ScriptedPetClinic/pipeline.txt
- Definition of Done:
  - Upload the screenshot of Jenkins dashboard of the project to the GitHub under the folder Task27

TASK 28 : DECLARATIVE PIPELINE FOR A SIMPLE HELLO WORLD

- Activity:
  - Create a declarative pipeline for a HelloWorld Application
- Steps:
  - Create a pipeline project DeclarativeHelloWorld. Code is available in Jenkinsfile of

    https://github.com/AnjuMeleth/hello-world-java.git
- Definition of Done:
  - Upload the screenshot of Jenkins dashboard of the project to the GitHub under the folder Task28

TASK 29 :DECLARATIVE PIPELINE FOR BUILDING A COMPLEX JAVA APPLICATION

- Activity:
  - Create a declarative pipeline to build a spring pet clinic application
- Steps:
  - Create a pipeline project DeclarativePetClinic

  Code is available in Jenkinsfile of https://github.com/AnjuMeleth/spring-petclinic.git
- Definition of Done:
  - Upload the screenshot of Jenkins dashboard of the project to the GitHub under the folder Task29

TASK 30 :DECLARATIVE PIPELINE FOR BUILDING, TESTING A COMPLEX JAVA APPLICATION

- Activity:
  - Create a declarative pipeline to build, test a spring pet clinic application
- Steps:
  - Create a pipeline project DeclarativePetClinic

  Code is available in Jenkinsfile of https://github.com/AnjuMeleth/spring-petclinic.git
- Definition of Done:
  - Upload the screenshot of Jenkins dashboard of the project to the GitHub under the folder Task30

TASK 31 :DECLARATIVE PIPELINE FOR BUILDING, TESTING, PACKAGING A COMPLEX JAVA APPLICATION

- Activity:
  - Create a declarative pipeline to build, test and package a spring pet clinic application
- Steps:
  - Create a pipeline project DeclarativePetClinic

  Code is available in Jenkinsfile of https://github.com/AnjuMeleth/spring-petclinic.git
- Definition of Done:
  - Upload the screenshot of Jenkins dashboard of the project to the GitHub under the folder Task31

TASK 32 :DECLARATIVE PIPELINE FOR BUILDING, TESTING, PACKAGING, VERIFY A COMPLEX JAVA APPLICATION

- Activity:
  - Create a declarative pipeline to build,test, package, verify a spring pet clinic application
- Steps:
  - Create a pipeline project DeclarativePetClinic

  Code is available in Jenkinsfile of https://github.com/AnjuMeleth/spring-petclinic.git
- Definition of Done:
  - Upload the screenshot of Jenkins dashboard of the project to the GitHub under the folder Task32

TASK 33 : ADD A SLAVE NODE

- Activity:
  - Add a slave node (Ubuntu instance) to the Jenkins master
- Steps:
  - Go to Manage Jenkins -> Manage Nodes and clouds -> New Node
  - Give a node name and select permanent agent
  - Enter the Host details and directory as /home/ubuntu. Please note to enter IP
  - Select launch method to be launch with ssh.
  - Select Add credentials and ssh username with key option provide the pem file details there.
  - Select non verifying strategy.
  - Make sure to install java on the slave node.

```
sudo apt-get update
sudo apt-get install default-jdk
```

- Definition of Done:
  - New node will be seen online in Jenkins Dashboard

TASK 34 : RUN AN APPLICATION ON JENKINS SLAVE

- Activity:
  - Execute the job PetClinicBuild in a Jenkins slave
- Steps:
  - Go to project configuration and enable “restrict where this project is run” and provide a slave label
  - Code in https://github.com/AnjuMeleth/spring-petclinic.git
- Definition of Done:
  - Upload the screenshot of Jenkins dashboard -> console output of the project to the GitHub under the folder Task34

TASK 35 : RUN AN APPLICATION ON JENKINS SLAVE IN SCRIPTED PIPELINE

- Activity:
  - Execute the spring pet clinic application in a Jenkins slave through scripted pipeline
- Steps:
  - Go to the configuration of ScriptedPetClinic job
  - Modify the scripted pipeline from

```
node { -> node ('slave1') {
```

- Definition of Done:
  - Upload the screenshot of Jenkins dashboard -> console output of the project to the GitHub under the folder Task34

TASK 36 : RUN AN APPLICATION ON JENKINS SLAVE(DEMO) IN DECLARATIVE PIPELINE

- Activity:
  - Execute the spring pet clinic application in a Jenkins slave through declarative pipeline
- Steps:
  - You may have to fork the repository https://github.com/AnjuMeleth/spring-petclinic.git
  - Modify the Jenkinsfile in the forked GItHub repository from

    agent{label 'master’} -> agent{label ‘slave1”}

    And also provide your GitHub repository path

  Build now
- Definition of Done:
  - Upload the screenshot of Jenkins dashboard -> console output of the project to the GitHub under the folder Task36

TASK 37 : INSTALL A PLUGIN IN JENKINS (BONUS TASK)

- Activity:
  - Install a “NodeJS” plugin in Jenkins.
- Steps:
  - Go to Manage Jenkins -> Manage Plugins -> Available
  - Select NodeJS and install without restart
  - Go to Manage Jenkins -> Global Tool configuration
  - Global npm packages to install give as -> bower@~1.8.0 grunt-cli@~1.2.0
- Definition of Done:
  - NodeJS could be viewed under the Installed tab

TASK 38 : RUN A NODE JS APPLICATION(DEMO)

- Activity:
  - Execute building of a Node JS application
- Steps:
  - Create a pipeline project NodeSample
  - Repository is at https://github.com/AnjuMeleth/simple-node-js-react-npm-app.git
- Definition of Done:
  - No screenshots to be uploaded

TASK 39 : HOW TO SEND AN EMAIL FROM JENKINS (DEMO)

- Activity:
  - Send an email from Jenkins about the status of our builds
- Steps:
  - Go to PetClinicBuild project
  - Select Editable Email notification in post build actions and provide the recipient name
  - Go to Advanced settings and set the trigger as always
  - Go to Manage Jenkins -> Configure system -> Extended Email Notification
  - For gmail , smtp server is smtp.gmail.com and suffix @gmail.com.
  - Select Advanced section , Use SMTP authentication. Give user name and password.(Password could be ideally app password)
  - Go to gmail account -> security -> App passwords -> Select app and generate password
  - Use SSL port 465
  - Try sending a test mail also for Email notification section.

- Definition of Done:
  - Upload the screenshot of Jenkins dashboard -> console output of the project to the GitHub under the folder Task36

TASK 40: HOW TO ADD A NEW USER IN JENKINS AND GIVE RESTRICTED PERMISSIONS

- Activity:
  - Add a new user and set only the overall read permissions in Jenkins
- Steps:
  - Go to Manage Jenkins -> Manage Users -> Create user
  - Provide the user details and save.
  - Go to Manage Jenkins -> Configure Global security -> Authorisation -> .Matrix based security
  - Select overall read.
- Definition of Done:
  - New user cannot create a job. No screenshots to be uploaded.

TASK 41 : INSTALL A PLUGIN IN JENKINS (ADMIN TASK)

- Activity:
  - Install a “Backup” plugin in Jenkins.
- Steps:
  - Go to Manage Jenkins -> Manage Plugins -> Available
  - Select Backup and install without restart
- Definition of Done:
  - Backup could be viewed under the Installed tab

TASK 42 : BACKUP JENKINS (DEMO)

- Activity:
  - Backup Jenkins server with all it’s jobs.
- Steps:
  - Go to Manage Jenkins -> Backup Manager -> Setup
  - Provide backup directory /tmp
  - Backup Hudson configuration
- Definition of Done:
  - Backup could be found in /tmp folder. No screenshots required.

Q & A

THANK YOU