**Jenkins - Pipelines**

**Beginning**

We will run everything on just the master node (building and sonarqube). To make sure that we can do this, let's adjust the master to be a t2.large instance so that it won't die when we run these projects. You can do this without losing anything by stopping the ec2 instance -> actions -> instance settings -> change instance type. Then boot it up. Make sure that you can access jenkins again. If you had sonarqube installed on the slave, you will need to unzip and run it on the master. (Make sure you update sonarqube setting within jenkins to point to the correct ip address if necessary). Remember that you have to restart sonarqube everytime your machine restarts. (Double check the status of sonarqube with something like `./sonarqube-9.0.1.46107/bin/linux-x86-64/sonar.sh status`).

Install maven on your main node. (ubuntu: sudo apt install maven, amazon: sudo yum install maven)

Before beginning chapter 3, the author assumes that you have created 3 different freestyle projects (which were not covered in the last chapter), so before creating pipelines, I created the following additional projects (leave your existing project alone):

- Create PetClinic-Compile (for building)
  - Add git link under source code management
  - single build step, invoke top-level maven targets, goal is a package
- Create PetClinic-Code (for sonarqube)
  - Add git link under source code management
  - single build step, invoke top-level maven targets, goal is a package
  - execute sonarqube scanner (all blank)
- Create PetClinic-Test (for junit tests)
  - Add git link under source code management
  - single build step, invoke top-level maven targets, goals are `test verify`
  - Post-build action to publish junit test
  - Test report XMLS `**/target/surefire-reports/**.xml`

Make sure that each of those will build correctly before proceeding!

**Pipelines**

Read through the overview and Elements of a CI/CD pipeline [here](#).

Most of the following was pilfered from chapter 3 of the text, but it seems fairly outdated, so I have tried to recreate the steps for a pipeline below, though you could still get the general idea from the textbook.

Begin by clicking on **new item** -> **pipeline**. Give it a name like `PetClinic-Pipeline`. Down in the pipeline script section, you could type some code to define the steps of your pipeline. This is a great reference. I have included some code in the git repository that you forked called `joespipeline`. You should take a look at it. You should configure your pipeline script to build from SCM (click the dropdown under Definition). Paste in your git repo and credentials. Make sure to identify your script path. (I.e. `joespipeline`). When you click on **Build Now** you should see two stages that run.

Make sure that your pipeline build succeeds before moving on. See if you can modify the pipeline script to have it run on your slave node(optional). (Hint: you will probably have to install maven on it).

Note that your pipeline should generate an artifact called `petclinic.war`. You should be able to see this after the pipeline completes under the status screen. The war file is what will eventually be sent to production.

**Another pipeline**

In this section, you can pretty much follow the section from chapter 3 on “Using the build pipeline plugin”. (Stop where it talks about deployment). Or you can view my instructions here below.

We have already created multiple build project in part one of this assignment. Now we will chain them together to form a **pipeline**. Navigate to your PetClinic-Compile project -> Add post-build action to build other projects. In the **projects to build** section. Add PetClinic-Code and PetClinic-Test. (Only if the build is stable). Back on the main status page for the Compile project you should see the two downstream projects.

Install a new plugin **Build Pipeline** (back under manage jenkins -> install plugins).
Back on the main Dashboard, you should now be able to click the plus sign and see a Build Pipeline View. Name it. Set the initial build job to be PetClinic-Compile. Click the OK button. You should now have a cool visual of your chained projects or pipeline. You should run it.

**Hint**

When you configure your downstream projects for PetClinic-Compile, make sure you selected ‘Trigger parameterized build on other projects’, instead of ‘Build other projects’.

**Check off procedure:**

In a series of screenshots, or to your instructor:

- Demonstrate that your new (non-plugin) pipeline successfully completed (on the main node and/or slave)
- Demonstrate that your pipeline view has all jobs finishing