Week 3

This week we will begin to use our source control system. We will also begin to run jenkins in kubernetes. Jenkins is the CI/CD tool we will use for this course. Git is the code versioning system used to track changes to documents and software.

Instructions to Jenkins are stored in the “Jenkinsfile”, which is in the github repository.

The Jenkinsfile will describe work to build software. Jenkins calls this list of work a “pipeline”, which is further divided into “stages”.

This week our Jenkinsfile will create the “build” stage. In later weeks, this will be followed by “test” stages, and then “deploy” stages.

The next few week's experiments will model a workflow:

1. A developer modifies some code, and commits it to github.
2. Jenkins sees this, and begins the build stage.
3. If the build succeeds, the code will be tested in the next stage. The testing can be very sophisticated.
4. Assuming the tests pass, the final “deploy” stage is run. The code is then visible to customers.

We will run both Jenkins and the deployed system that Jenkins builds in kubernetes.

This week’s labs will be performed in the weekly chat session (recorded).

Lab 1: version control using git

Git is a way to manage changes in software. It is capable of merging changes from multiple people. It is good for collaborative projects.

This week github will be used to record Jenkins configurations. When you work with configurations described in text files, it is important to be able to record changes and have a description of why they were made. Others can see what changes made and better understand them, even long after the author left the company.

You must first download it.

```
yum -y install git
```

Next, download this week's material.

```
git clone https://github.com/dlambrig/umlF21.git
```

A directory umlF21 will be created with a sample Jenkins file.

You can store and track your changes in the local repository (residing on your laptop). In a real project you would eventually "push" your changes to the "remote repository" (residing in github's cloud).

Do the following:

```
git log
```

```
git show
```

You should see the changes I made to "Jenkinsfile".

Now you make a change. Using an editor, modify file "Jenkinsfile". Change the "echo" command such that it prints something out.

After you save the file, issue

```
git diff
```

You should see the changes you made.

Next, identify yourself so your commit record will be accurate.

```
git config --global user.name "yourname"

git config --global user.name "youremail@yourcompany.com"
```

Commit your change:

```
git commit -m "Change the echo display"
```

Look at the log of changes.

```
git log
```

Your change is recorded in the local repository, but not the remote repository on github (which is shared by everyone, do not do that).

In the homework, you will create a new repository on github and then use “git init” to create a new local repository and “git revert” to undo a change. To do all this, you will need to

1. Create an account for yourself on github
2. Create a repository
3. Create a personal access token to act as your github password
4. Create a local workspace for the repository, as in below.

Some example instructions for this are here

Branches are a very important concept in git, and you must understand the concept. A branch is a clone of the original repository (the master branch) that can be changed on its own. It can periodically be merged with the master branch. Developers typically work in branches, and once their changes are ready, merge their changes into the master branch for others to use.

You can create a branch, and then look at all the branches you have locally.

```
git checkout -b your-branch

git branch
```

go back to master, and then go back to your branch.

```
git checkout master

git checkout your-branch
```

When you want to push your local changes to github, you can issue:

```
git push origin <branchname>
```

.. where branchname would be either main or whatever you created. In coming weeks we will build into the jenkins pipeline the mechanism to accept changes that are candidates to become part of the master branch.

Lab 2: install Jenkins

From last week, you should have kubernetes running.

First, create a namespace.

```
kubectl create namespace jenkins
```

Next, define a deployment.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jenkins
spec:
  replicas: 1
  selector:
    matchLabels:
      app: jenkins
  template:
    metadata:
      labels:
        app: jenkins
    spec:
      containers:
      - name: jenkins
        image: jenkins/jenkins:lts-jdk11
        ports:
        - containerPort: 8080
        volumeMounts:
        - name: jenkins-home
          mountPath: /var/jenkins_home
      volumes:
      - name: jenkins-home
        emptyDir: { }
```

Now, create the deployment

```
kubectl create -f jenkins.yaml --namespace jenkins
```

Next, create a service. It is made from this yaml file:

```
apiVersion: v1
kind: Service
metadata:
  name: jenkins
spec:
  type: NodePort
  ports:
  - port: 8080
    targetPort: 8080
  selector:
    app: jenkins
```

Create it this way:

```
kubectl create -f jenkins-service.yaml --namespace jenkins

kubectl get services --namespace jenkins
```

You will see a port. In my case it was 30291. Yours will be different. You can point your browser to http://localhost:port. As in:

```
http://localhost:30291/
```

It will ask you for a password. Look at the pod's logs to get it. First get the pod:

```
kubectl get pod -n jenkins
```

Then look at the pod's logs.

```
kubectl logs jenkins-7786bf646-c8gbm
```

Look for the text below (your password will be different)

```
Jenkins initial setup is required. An admin user has been created and a password
generated.
Please use the following password to proceed to installation:

3d5360f8c07b4b4f9c12dbc0033d96d4
```

You should install the default packages and create a default admin user. This will be demoed.

Lab 3: run a simple pipeline

At this point you should not be able to view jenkins from your browser and log into it.

Next, we will create a very simple pipeline. The jenkins file will be on github. This section follows steps described here.

We will use the jenkins file from the umlF21 repository.

```
pipeline {
    agent any
    stages {
        stage('build') {
            steps {
                sh 'uname -a'
            }
        }
    }
}
```

“Agent any” - means the work can run on any available executor. We only have the Jenkins master pod now, which is bad practice. Next week we will set up special "executing pods" to run the pipeline. This will allow us to model “scaling out”. In this way, more people could use Jenkins at once.

There is a single “build” stage in the pipeline. It runs a single step in the pipeline to run the command “uname -a”. This will be run on the jenkins master pod.

Execute through to step 5 of “Through the classic UI”. Then do the section “In SCM”.

Repository URL should be “https://github.com/dlambrig/umlF21/”

Branch specifier should be “*/main”

Script path should be “Jenkinsfile”

Run the pipeline. In the “console output” section of Jenkins, you should see:

```
[Pipeline] stage
[Pipeline] { (build)
[Pipeline] sh
+ uname -a
Linux jenkins-7786bf646-c8gbm 5.10.25-linuxkit #1 SMP Tue Mar 23 09:27:39 UTC 2021
x86_64 GNU/Linux
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
```

Jenkins stores data in the "jenkins home" folder. Next week we will create a "persistent volume" so the data in Jenkins will not be transient.

Useful references

https://www.digitalocean.com/community/tutorials/how-to-install-jenkins-on-kubernetes

https://devopscube.com/jenkins-build-agents-kubernetes/’’