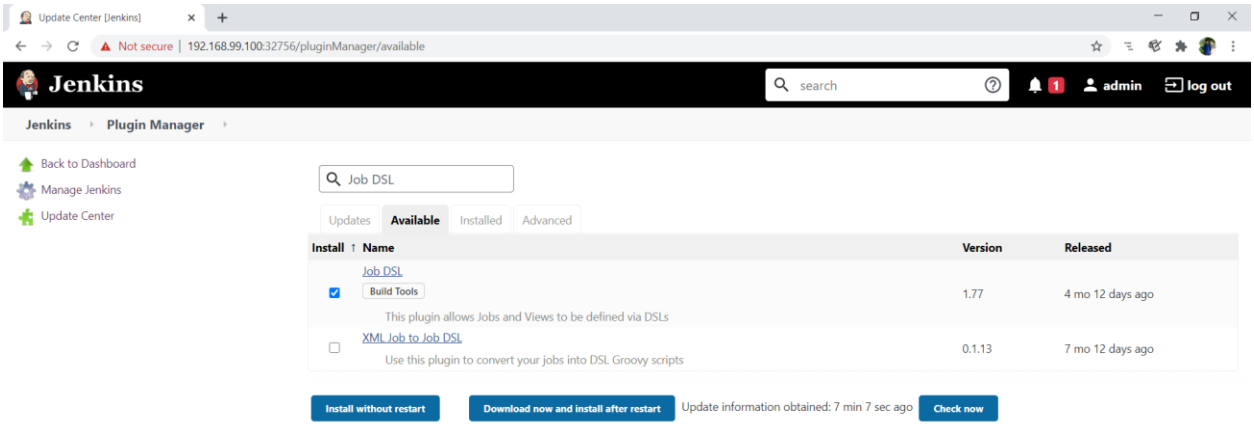
Deploying web-server using Jenkins-Groovy-code and Kubernetes

Jenkins is an open-source automation tool **used to** build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build.

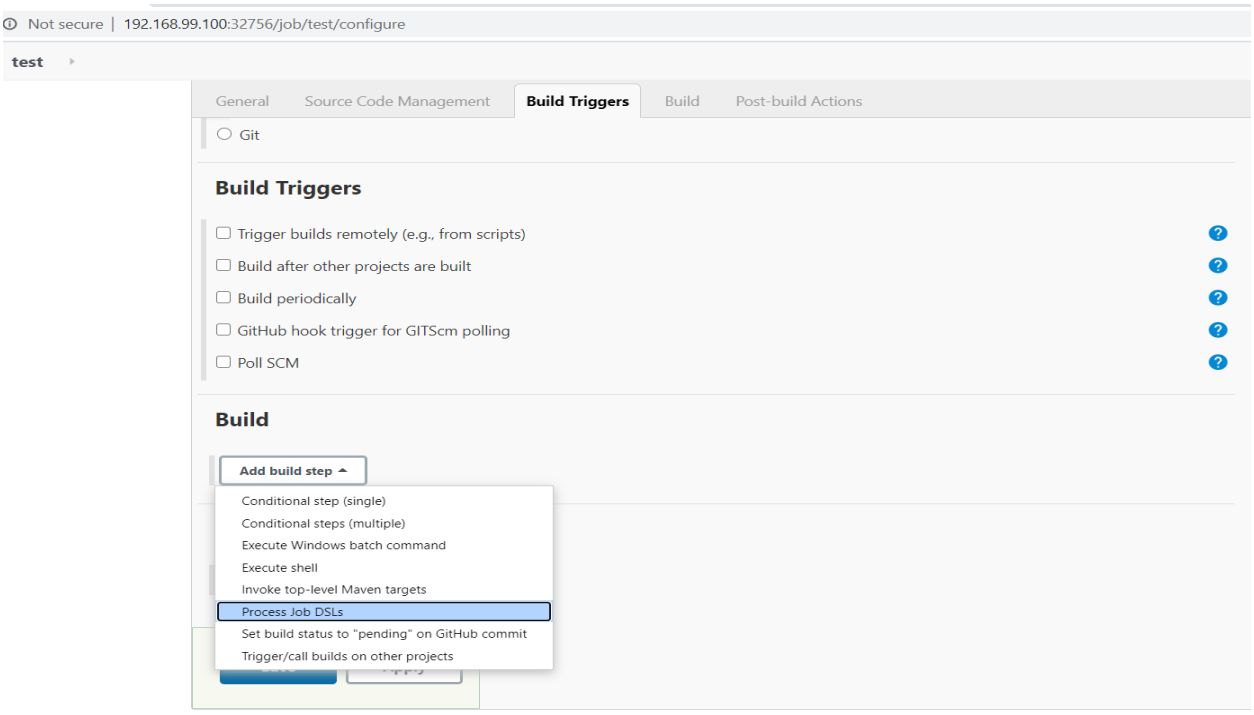
But there is one task of making a job/pipeline which we do manually till now in all my tasks. So somehow, we say that this thing is not automated.

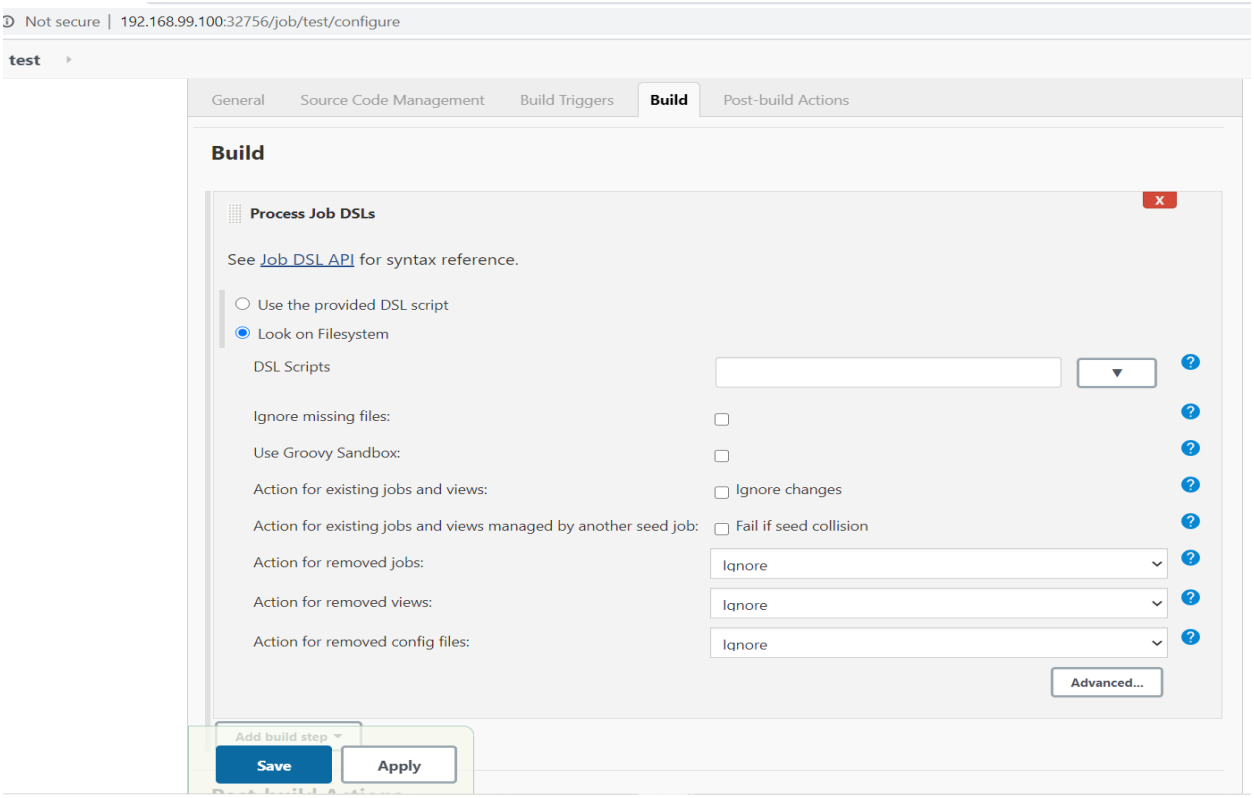
But Jenkins provides one way to make the job by writing code in Groovy language. For this, Jenkins has one plugin named ***Job_DSL.*** You need to install this only.



Now, you are ready to use Groovy code.

After this, Go to create job and in Build, You find one option of Process Job DSL's, click on this.





You have both the option like if you want to give code directly, then choose **Use the provided DSL script** and give the code here. But If you want to run your code as a script, then choose **Look on Filesystem** and give the name of script with ***.groovy*** extension. Make sure you load your file on Jenkins workspace.

Now developers push the code in GitHub, and we need to create only one job for fetch the code from GitHub, this job is also known as ***Seed Job***.

Task Overview:

1. Create container image that's has Jenkins installed using dockerfile.

2. When we launch this image, it should automatically start Jenkins service in the container.

3. Create a job chain of job1, job2, job3 and job4 using build pipeline plugin in Jenkins

4. Seed Job: Pull the Github repo automatically when some developers push the repo to Github.

5. Further on jobs should be pipeline using written code using Groovy language by the developer

6. Job1 :

1. By looking at the code or program file, Jenkins should automatically start the respective language interpreter installed image container to deploy code on top of Kubernetes (eg. If code is of PHP, then Jenkins should start the container that has PHP already installed)

2. Expose your pod so that testing team could perform the testing on the pod

3. Make the data to remain persistent using PVC (If server collects some data like logs, other user information)

7. Job2: Test your app if it is working or not.

8. Job3: if app is not working, then send email to the developer with error messages and redeploy the application after code is being edited by the developer

Tools used:

- Jenkins
- Kubernetes
- Docker
- **Run Jenkins server using Docker image on the top of Kubernetes:**

Dockerfile for creating image...

```
FROM centos

RUN sudo yum install wget -y

RUN sudo yum install git -y

RUN sudo yum install sudo -y

RUN sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo

RUN sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key

RUN sudo yum install java-11-openjdk.x86_64 -y

RUN sudo yum install jenkins -y

RUN sudo yum install net-tools -y

RUN sudo yum install python36 -y

RUN sudo yum install /sbin/service -y

RUN sudo yum install initscripts -y

RUN sudo yum install httpd -y

RUN sudo yum install git -y

RUN sudo yum install vim -y

RUN sudo yum install openssh-clients -y

RUN echo -e "jenkins ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers

RUN sudo curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s
https://storage.googleapis.com/kubernetes-release/release/stable.txt`/bin/linux/amd64/kubectl

RUN sudo chmod +x ./kubectl
```

RUN sudo mv ./kubectl /usr/bin/

RUN sudo mkdir /kube_cert/

COPY c* /kube_cert/

COPY config /root/.kube/config

COPY yml /root/yml

RUN sudo yum install ncurses -y

CMD java -jar /usr/lib/jenkins/jenkins.war && /bin/bash

EXPOSE 8080

After creating the image, push the image into Docker Hub. Only thing you need to change is files of .kube folder.

You can use my image also by pull using command ***docker pull anuddeeph/Jenkins_k8s:v1***

Now, this is the code for running Jenkins on top of Kubernetes using the YAML file.

```
apiVersion: v1
kind: Service
metadata:
  name: jenkins
  labels:
    app: jenkins
spec:
  ports:
    - port: 8080
  selector:
    app: jenkins
  type: NodePort
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: jenkins-claim
  labels:
    app: jenkins
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
---
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
```

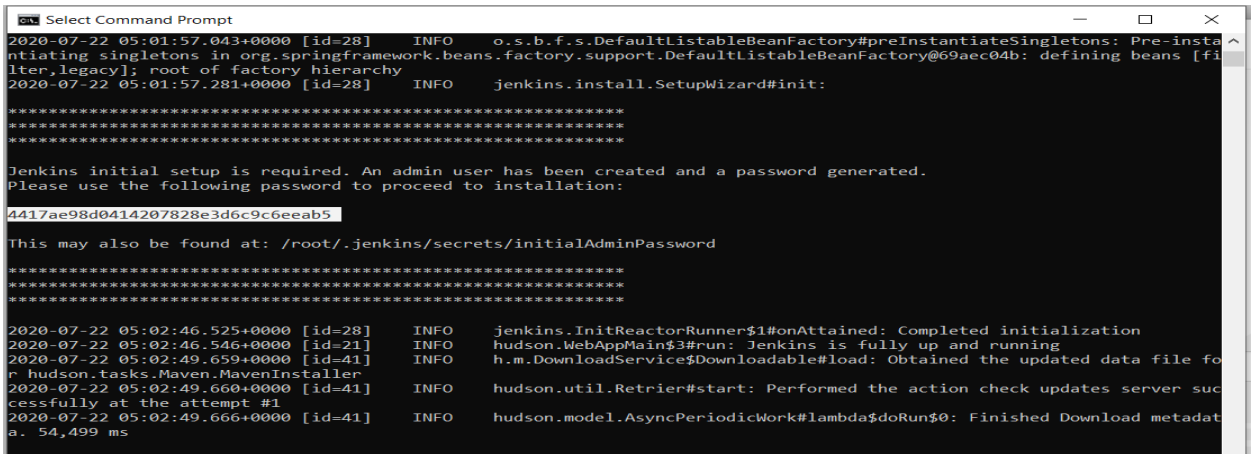
```
metadata:
  name: jenkins
  labels:
    app: jenkins
spec:
  selector:
    matchLabels:
      app: jenkins
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: jenkins
    spec:
      containers:
      - image: anuddeeph/Jenkins_k8s:v1
        name: jenkins
        ports:
        - containerPort: 8080
          name: jenkins
        volumeMounts:
        - name: jenkins-storage
          mountPath: /root/.jenkins/
      volumes:
      - name: jenkins-storage
        persistentVolumeClaim:
          claimName: jenkins-claim
```

To run this, we using the command...

kubectl apply -f file_name

Now Jenkins server is running...

First time when you run, this ask for password, for getting password, RUN ***kubectl logs pod_name***



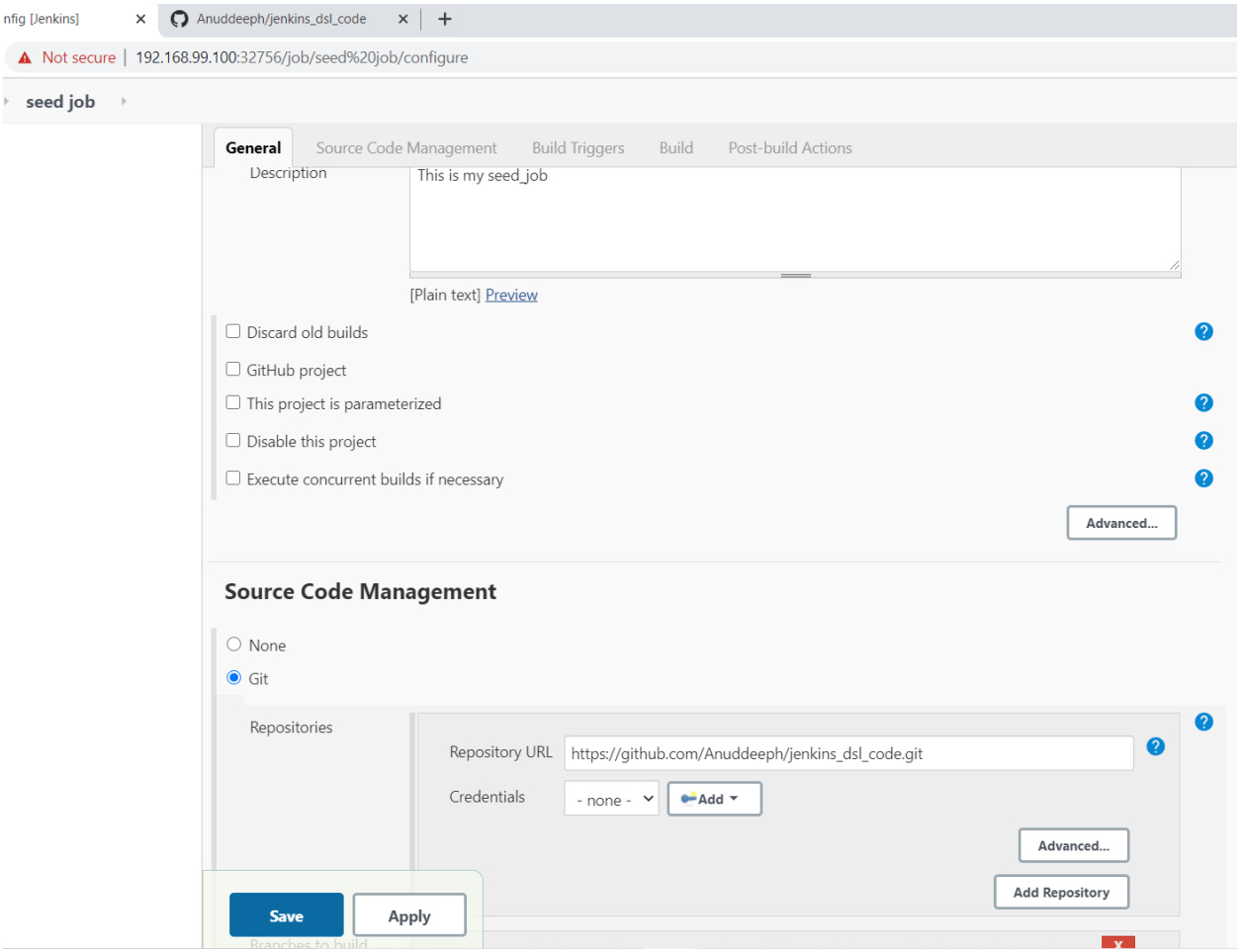
Now come to next step which is main part...

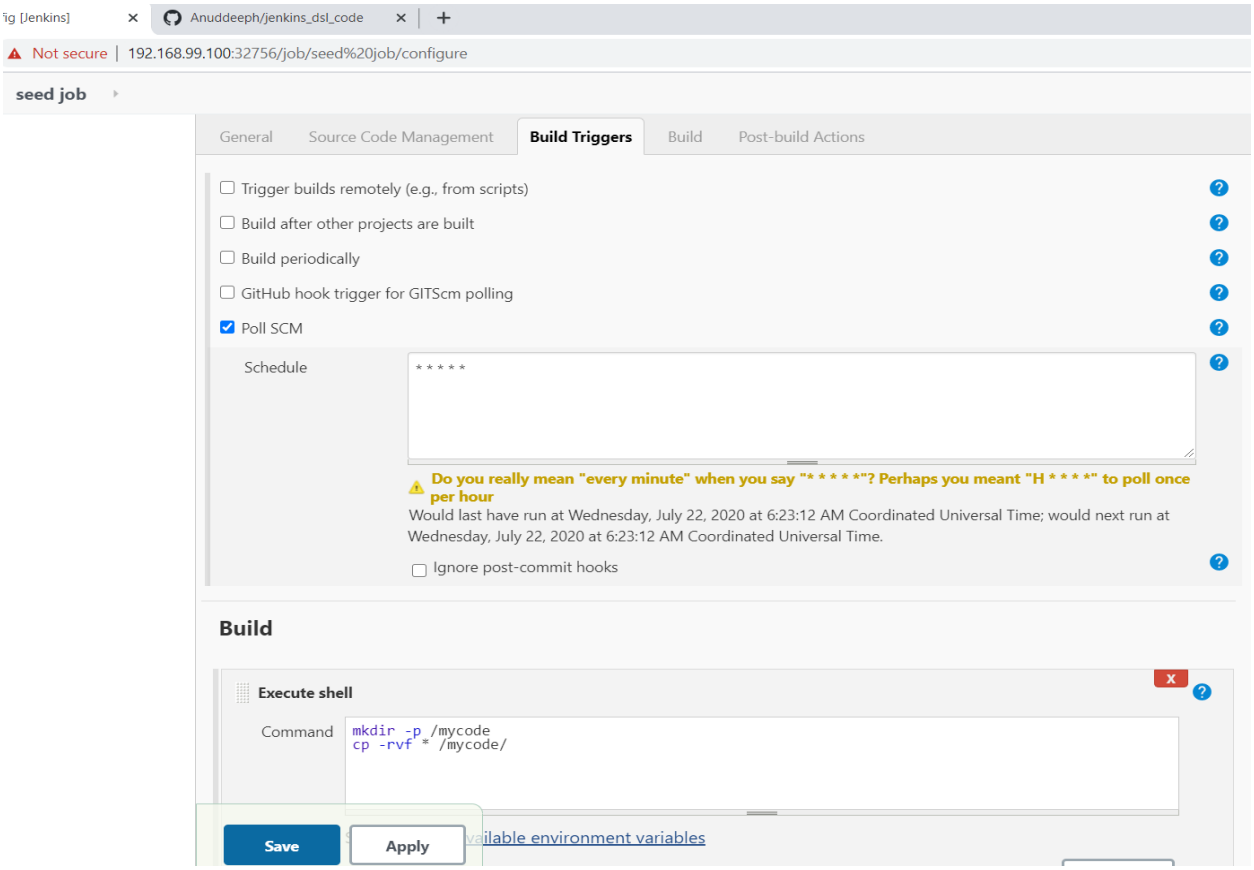
Jobs in Jenkins:

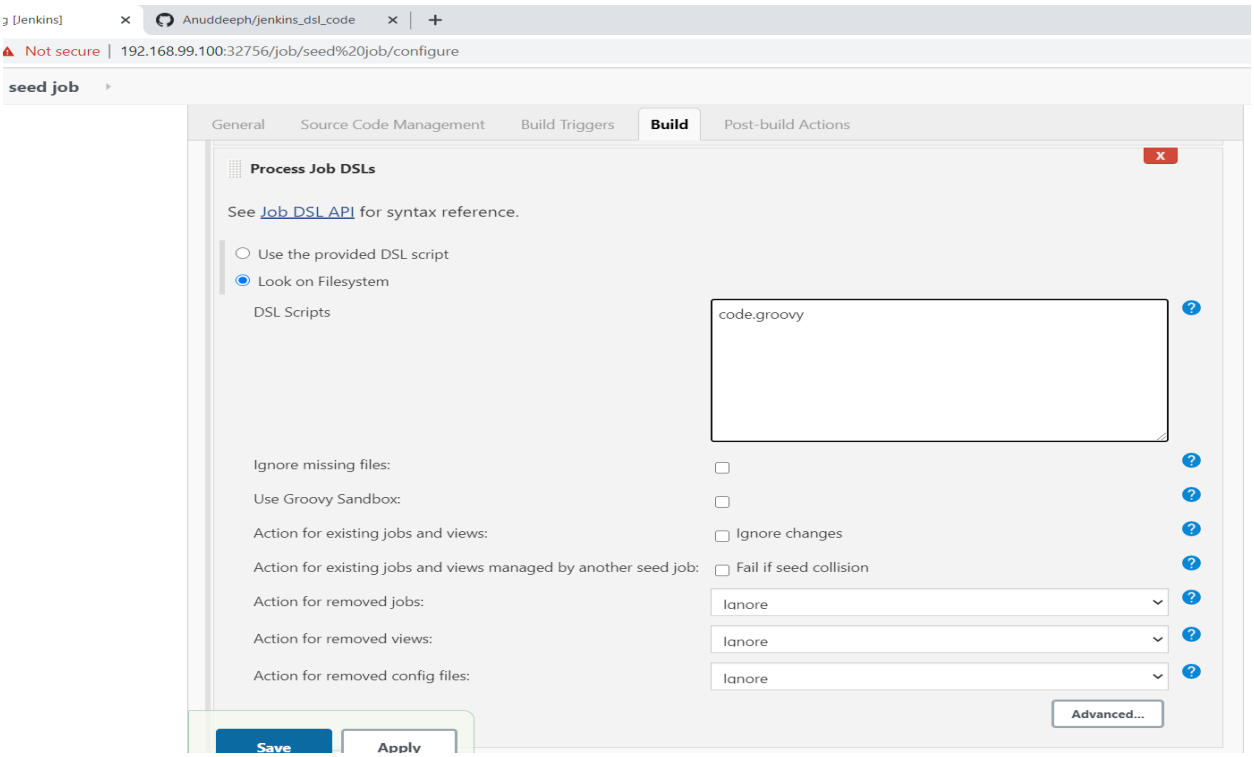
As I say, we need to create only one Seed job only, which create all the job automatically.

- **Seed Job (Pull the GitHub repo when developer pushes some code):**

Whenever the developer pushes any code in GitHub, this job automatically detects and copy in host OS and create other Jobs.

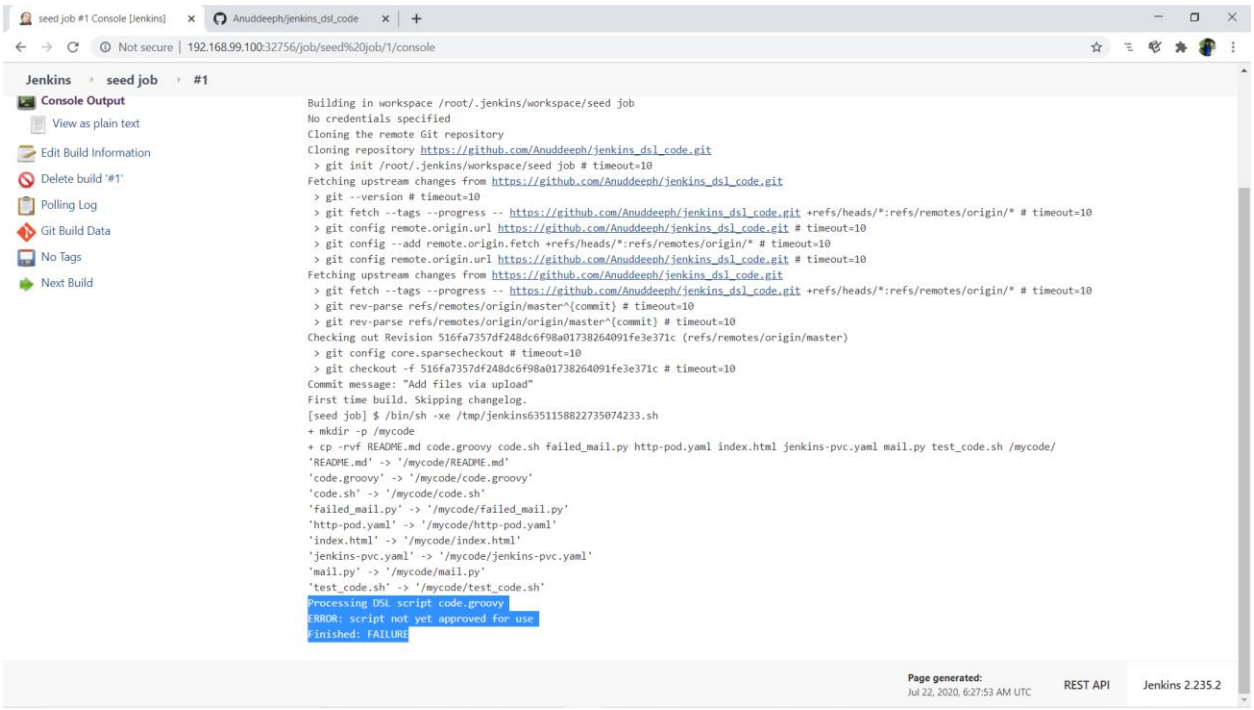



ig [Jenkins]
Anuddeeph/jenkins_dsl_code
Not secure | 192.168.99.100:32756/job/seed%20job/configure
seed job
General
Source Code Management
Build Triggers
Build
Post-build Actions
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Build periodically
GitHub hook trigger for GITScm polling
Poll SCM
Schedule
* * * * *
Do you really mean "every minute" when you say "* * * * *"? Perhaps you meant "H * * * *" to poll once per hour
Would last have run at Wednesday, July 22, 2020 at 6:23:12 AM Coordinated Universal Time; would next run at Wednesday, July 22, 2020 at 6:23:12 AM Coordinated Universal Time.
Ignore post-commit hooks
Build
Execute shell
Command
mkdir -p /mycode
cp -rvf * /mycode/
Save
Apply
ilable environment variables

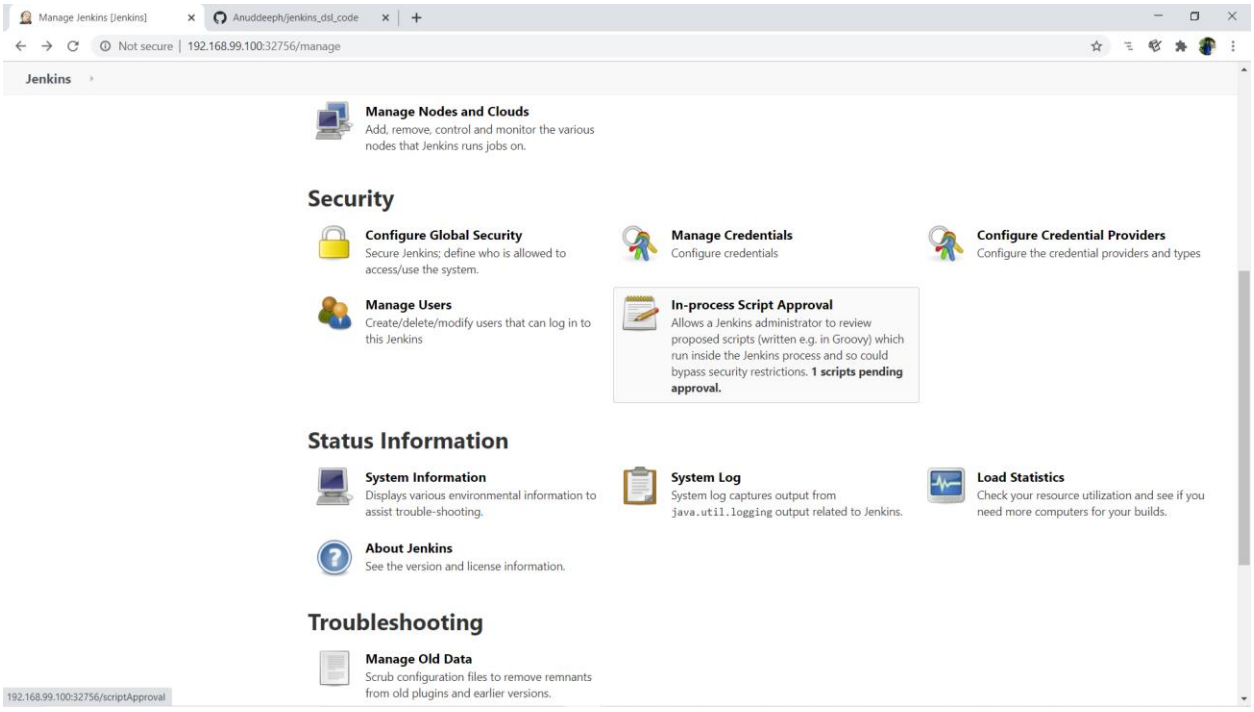

g [Jenkins]
Anuddeeph/jenkins_dsl_code
Not secure | 192.168.99.100:32756/job/seed%20job/configure
seed job
General
Source Code Management
Build Triggers
Build
Post-build Actions
Process Job DSLs
See Job DSL API for syntax reference.
Use the provided DSL script
Look on Filesystem
DSL Scripts
code.groovy
Ignore missing files:
Use Groovy Sandbox:
Action for existing jobs and views:
Ignore changes
Action for existing jobs and views managed by another seed job:
Fail if seed collision
Action for removed jobs:
Ignore
Action for removed views:
Ignore
Action for removed config files:
Ignore
Advanced...
Save
Apply

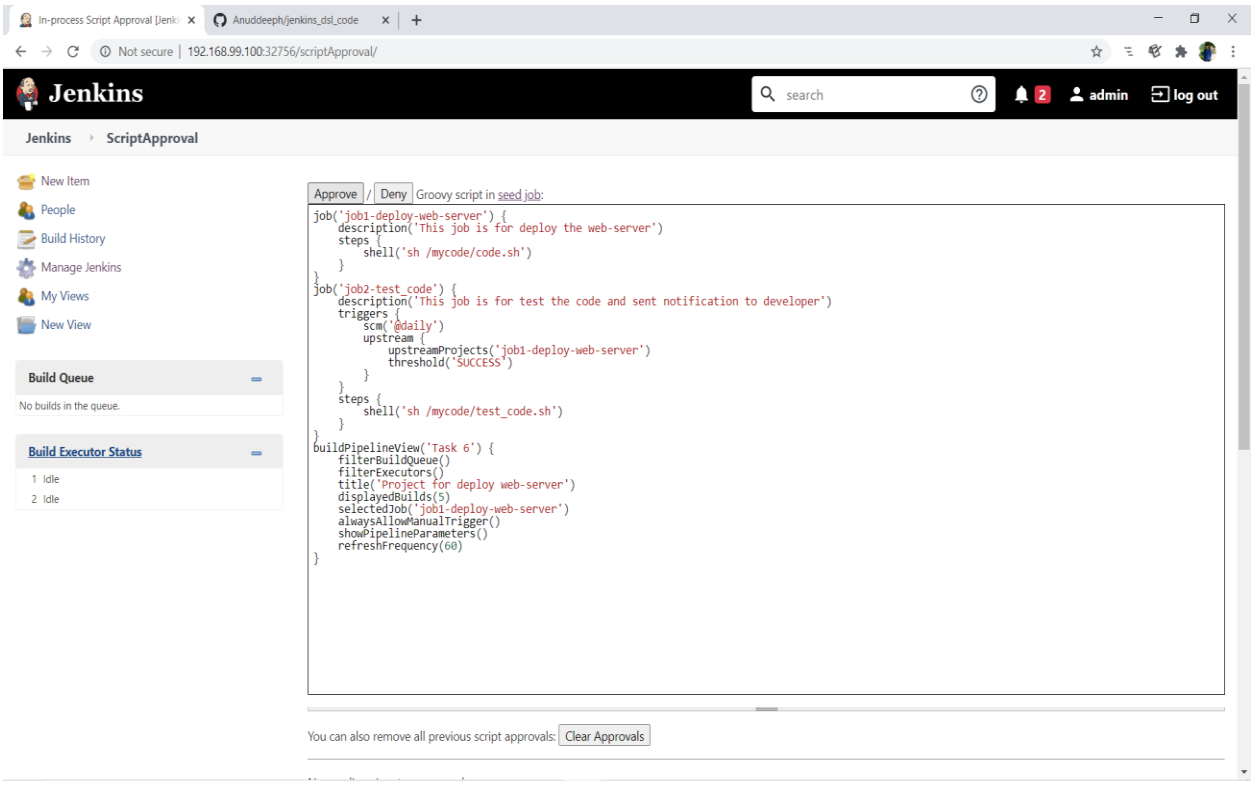
So let's see what happens when we build this job...

First time, when you run it, It gives an error...

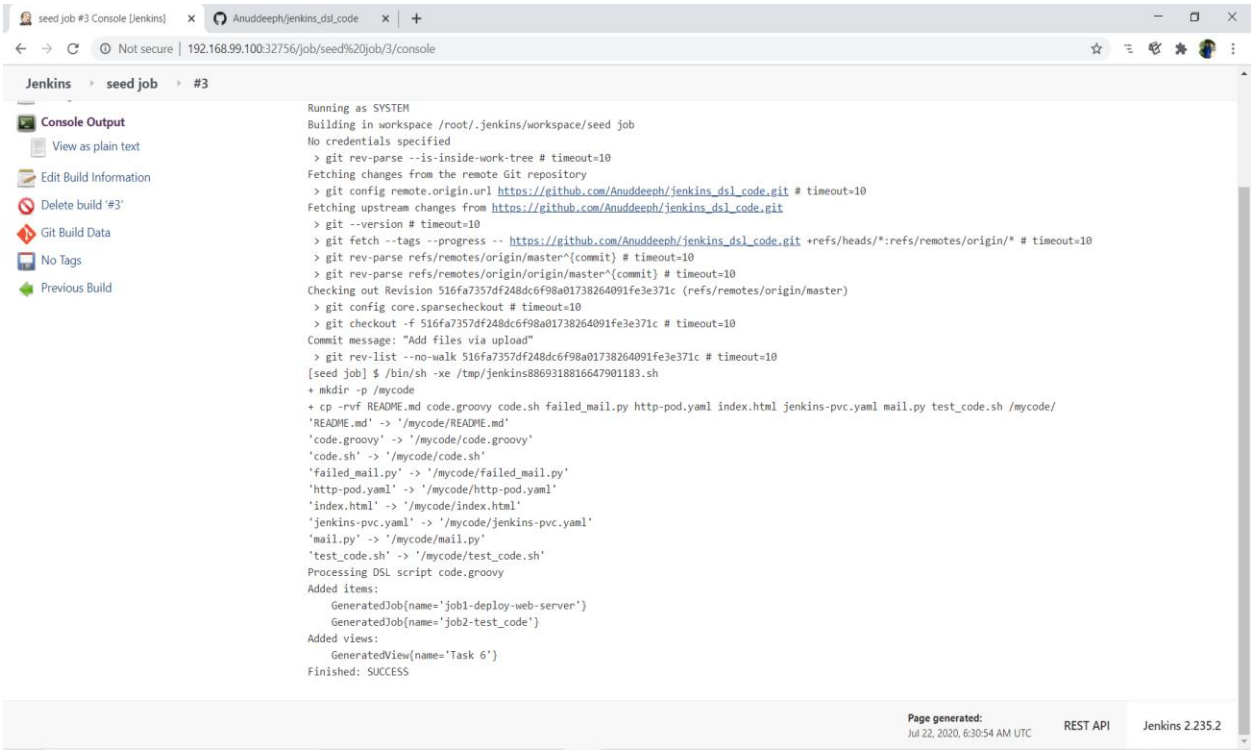


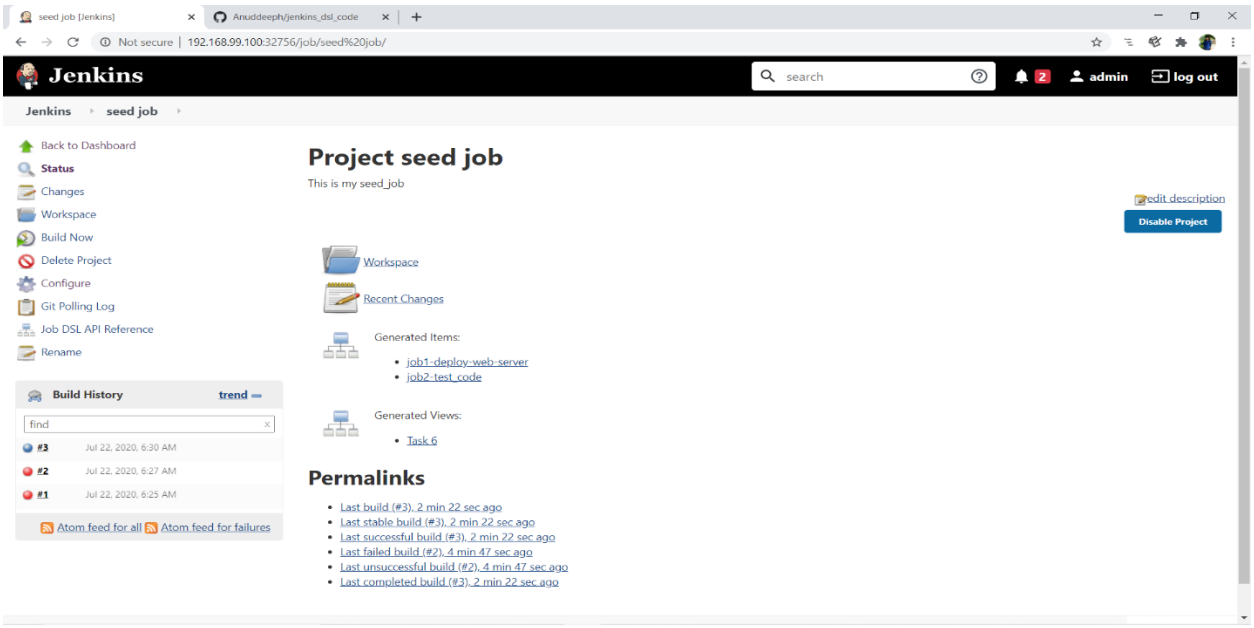
To remove this, go to **Jenkins configuration -> In Process Script Approval** and approve the script.



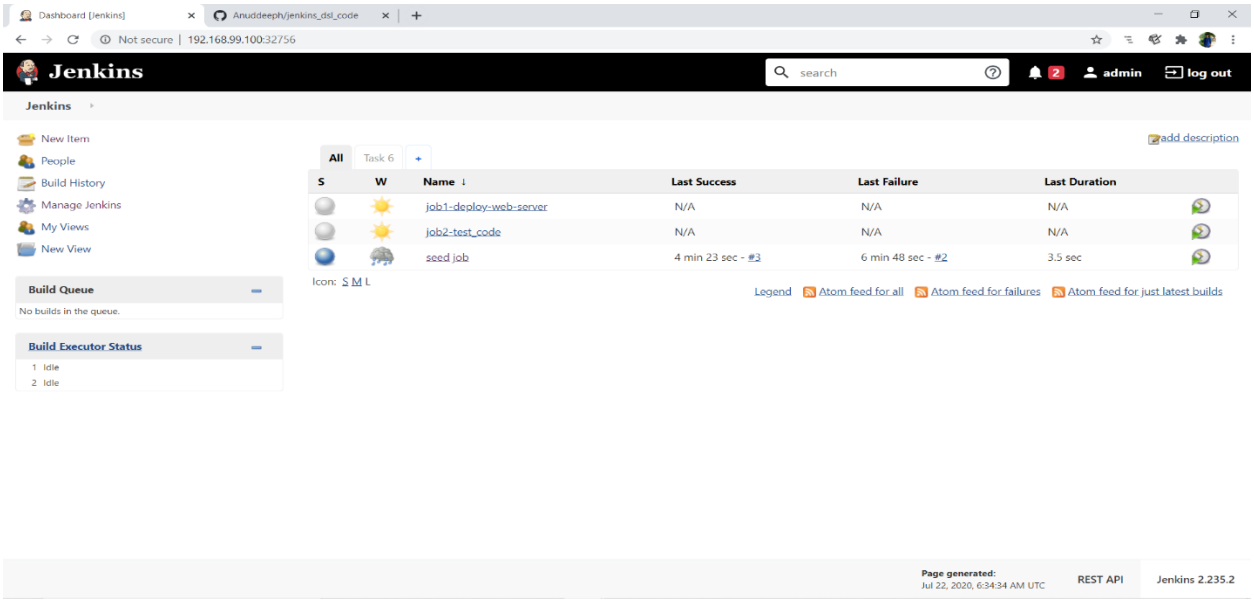


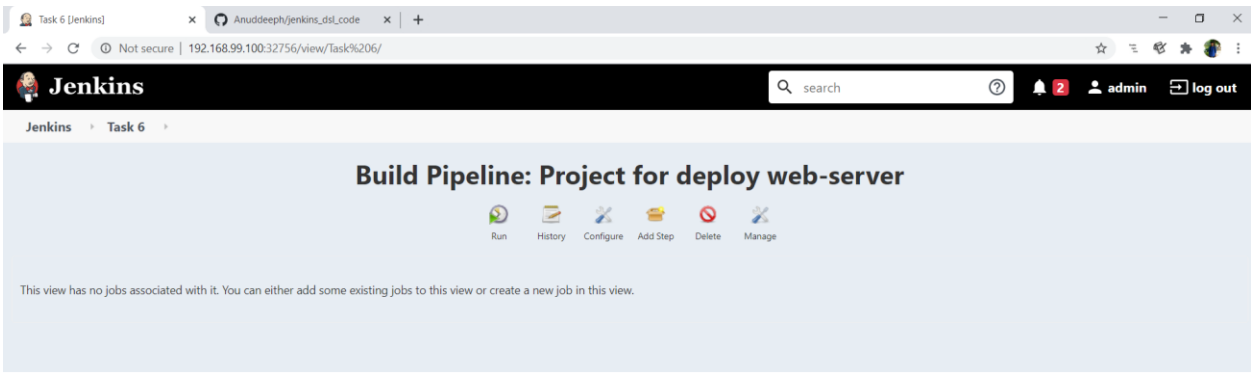
Now again build and this time output is...





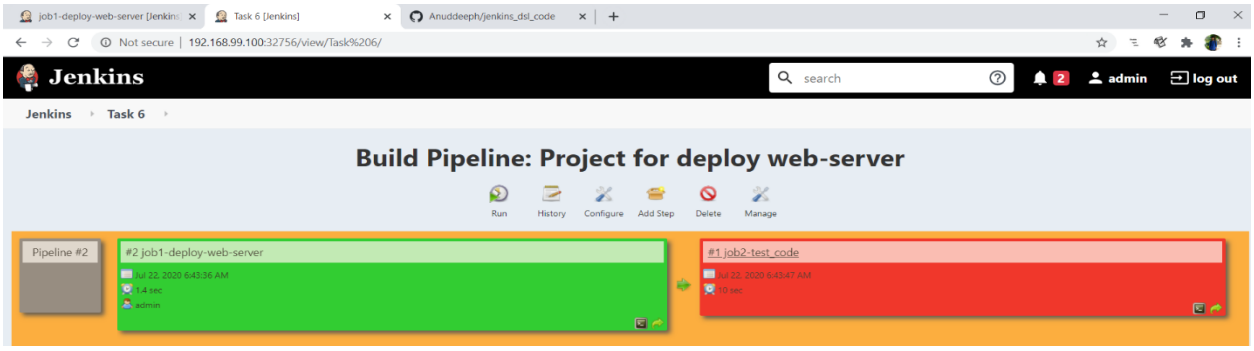
You can see that two jobs and one view are created here.



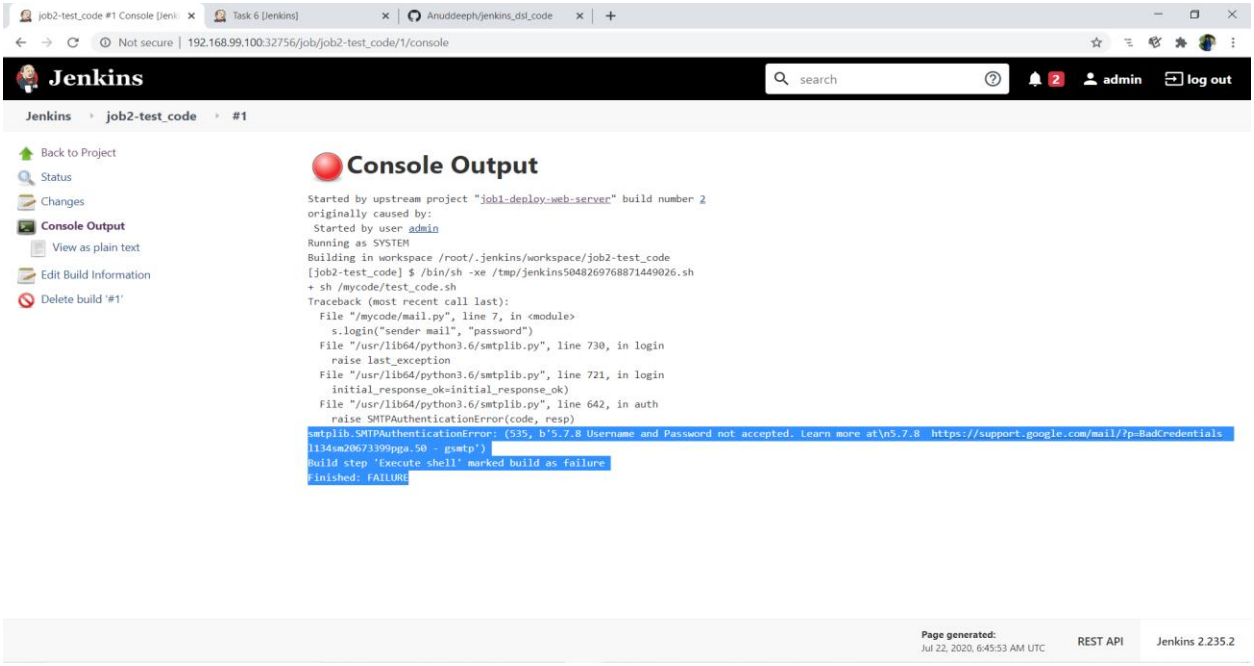


And this is the pipeline view...

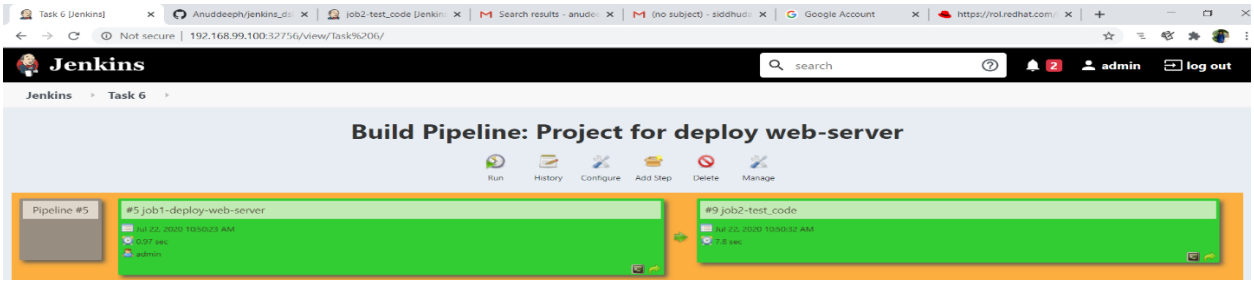
Push on RUN and your all jobs are running now...😍

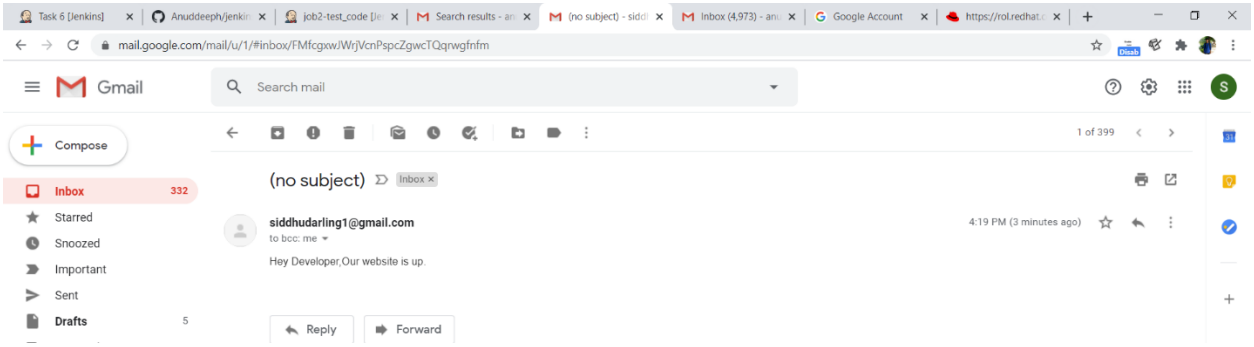


Job 2 is not running now; this conflict comes because I am not uploaded my mail password in GitHub. If you provided the right id/password, it works...👍



After giving the required credentials,





Yo...This completed and now this is much automated than previous tasks...

Now let's see what I write in my groovy file(code.groovy)...

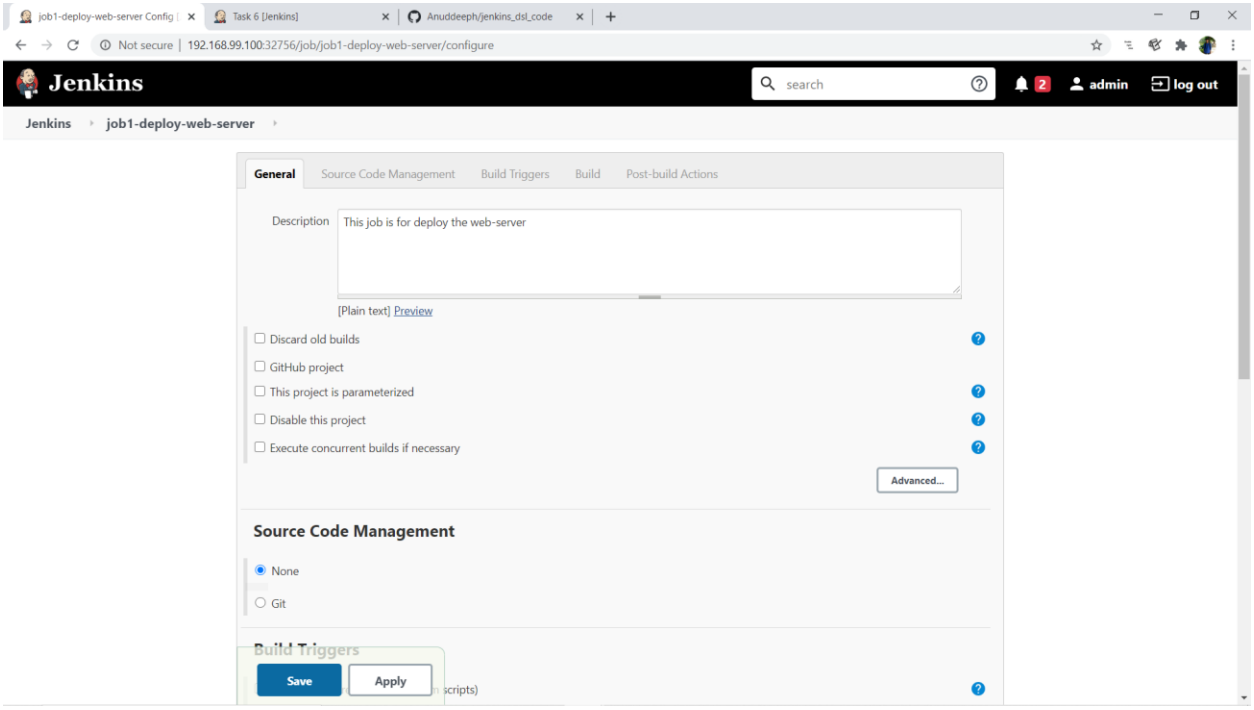
- **Job1(Deploy deployment for website):**

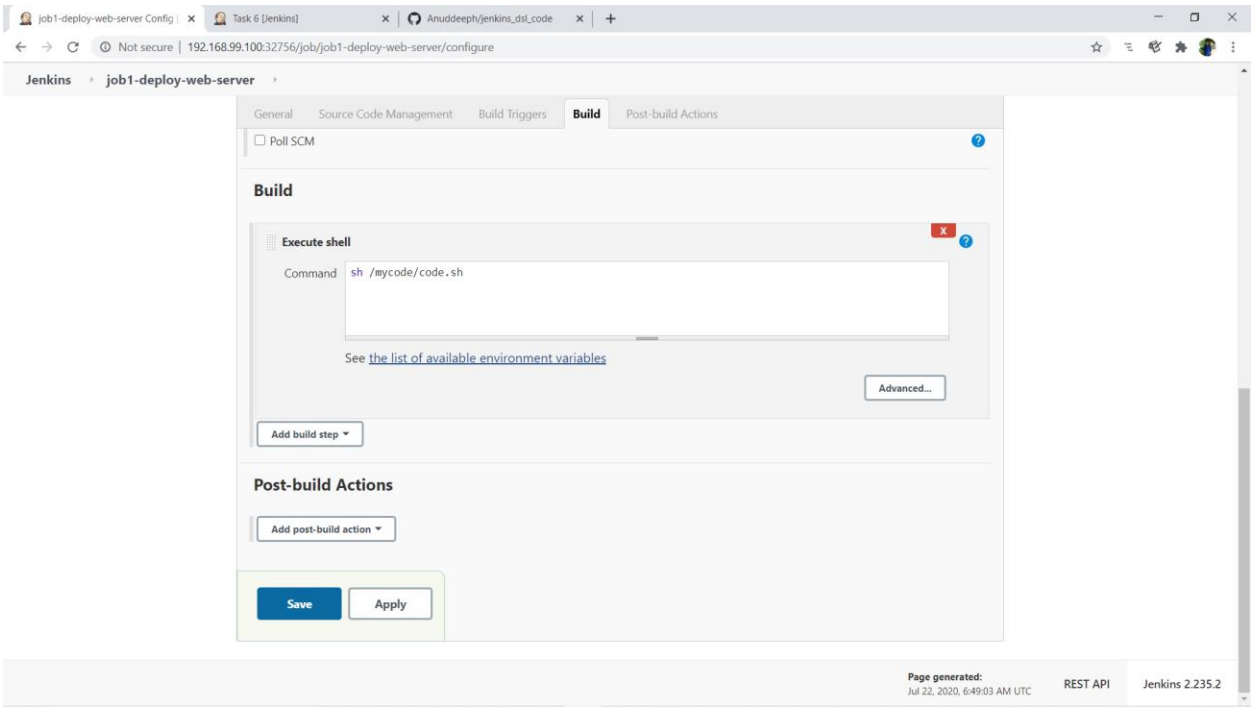
It launches the deployment with PVC and services. Here I have taken the example of webpages, so I have used the apache webserver.

```
job('job1-deploy-web-server') {
            description('This job is for deploy the web-server')
            steps {
                shell('sh /mycode/code.sh')
            }
        }
```

And code.sh is...

```
if kubectl get pods | grep httpd
        then
        echo "service is running"
        else
        kubectl create -f /mycode/http-pod.yaml
        sleep 20
        fi
        POD=$(kubectl get pod -l app=httpd -o
jsonpath="{.items[0].metadata.name}")
        kubectl cp /mycode/*.html $POD:/var/www/html/
```





The YAML code is launching for the website is below...

```
apiVersion: v1
kind: Service
metadata:
  name: httpd
  labels:
    app: httpd
spec:
  ports:
    - nodePort: 80
      port: 80
      protocol: TCP
      targetPort: 80
  selector:
    app: httpd
  type: NodePort
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: httpd-claim
  labels:
    app: httpd
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
---
```

```
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: httpd
  labels:
    app: httpd
spec:
  selector:
    matchLabels:
      app: httpd
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: httpd
    spec:
      containers:
      - image: anuddeeph/apache-webserver-php
        name: httpd
        ports:
        - containerPort: 8080
          name: httpd
        volumeMounts:
        - name: httpd-storage
          mountPath: /var/www/html/
      volumes:
      - name: httpd-storage
        persistentVolumeClaim:
          claimName: httpd-claim
```

- **Job2(Testing that website is running or not) and Job3(Sent a mail to the developer):**

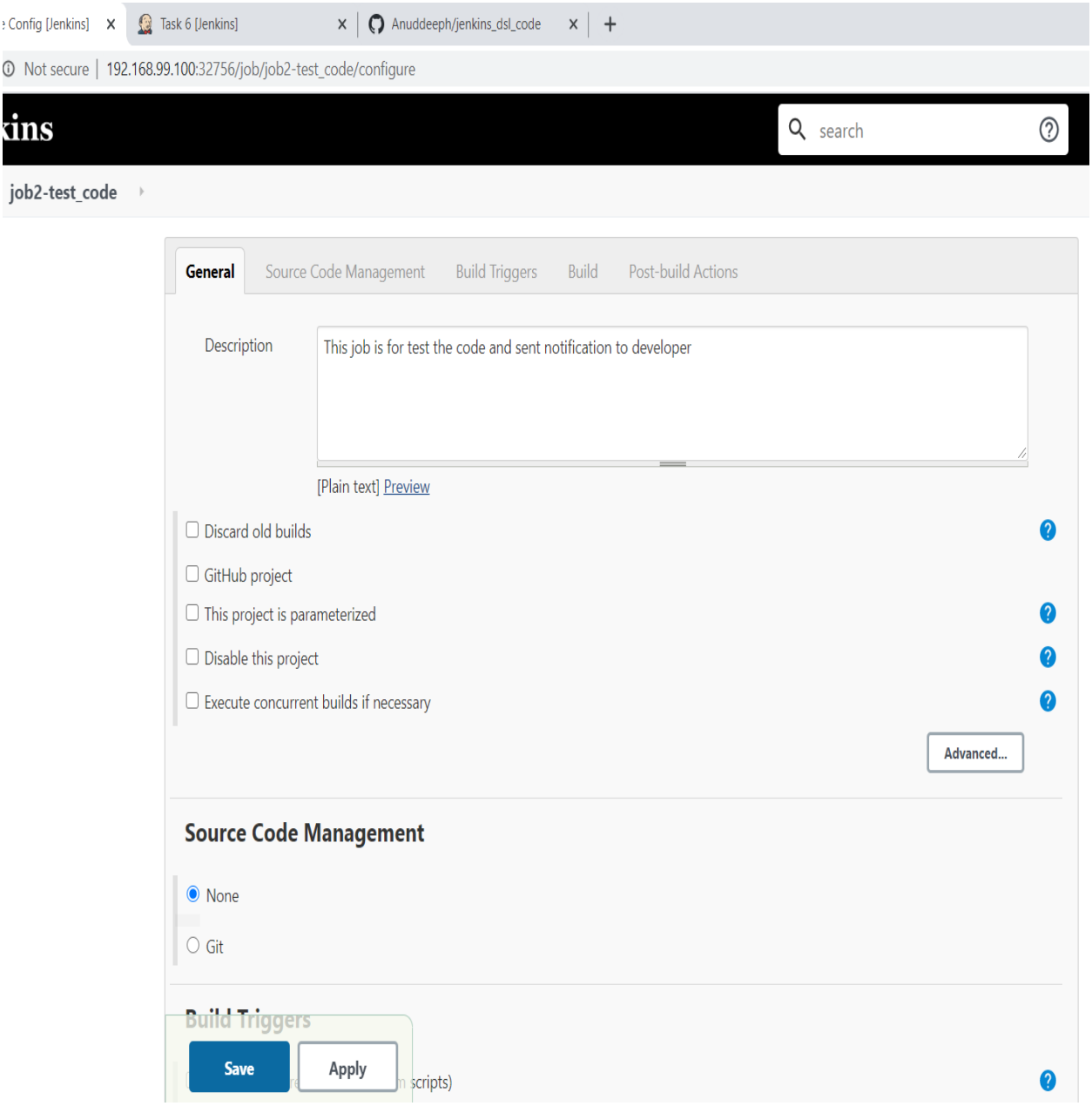
After successfully build, Job1 will trigger Job2 and it checks the website that is working or not. If not working, it sent a notification to the developer.

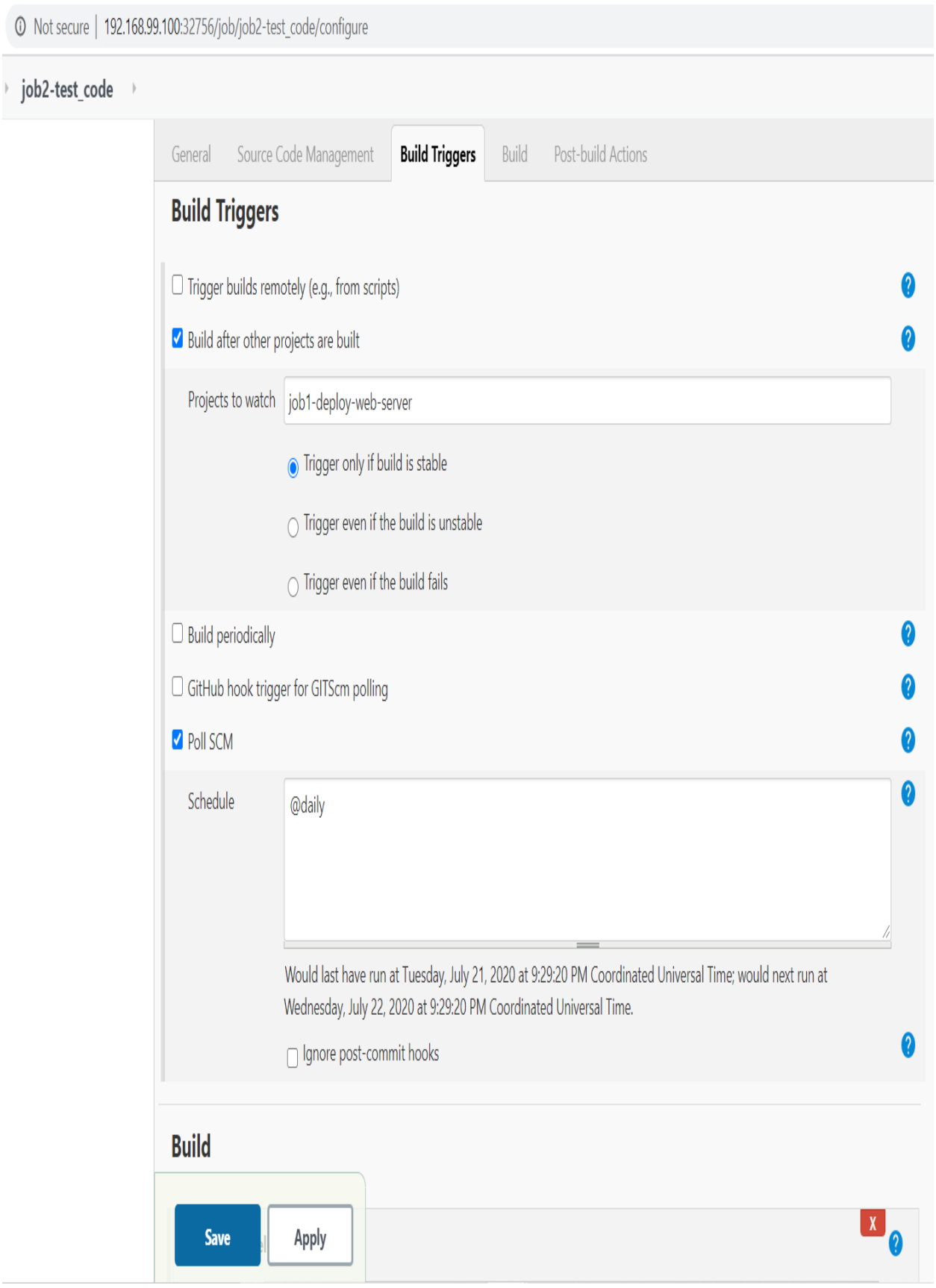
Groovy code for this...

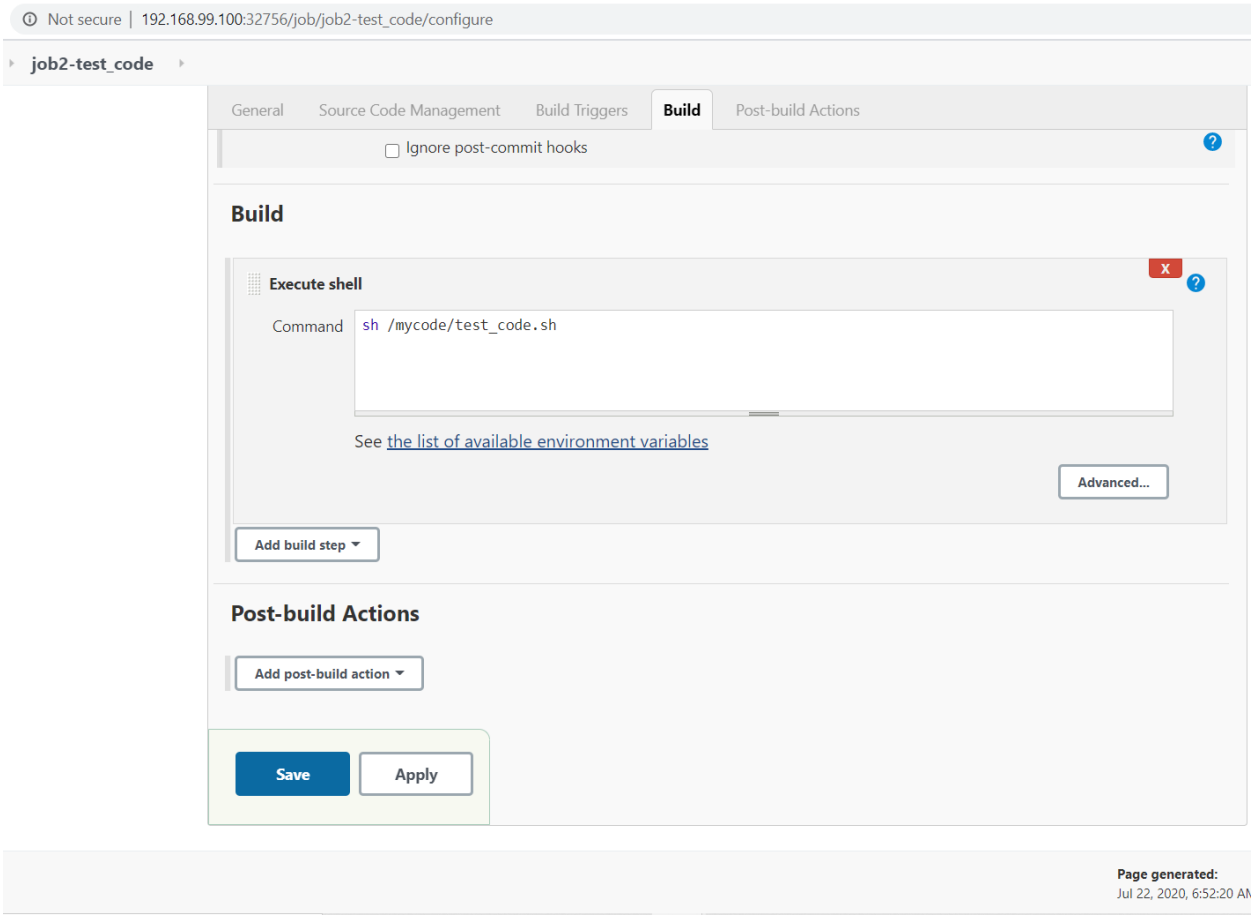
```
job('job2-test_code') {
            description('This job is for test the code and sent notification
to developer')
            triggers {
                scm('@daily')
                upstream {
                    upstreamProjects('job1-deploy-web-server')
                    threshold('SUCCESS')
                }
            }
            steps {
                shell('sh /mycode/test_code.sh')
            }
        }
```

I write one script for do test here(test_code.sh)...

```
#!/bin/bash
        status=$(curl -o /dev/null -s -w "{%http_code}"
http://192.168.99.101:32033)
        if [[ $status==200 ]]
        then
        python3 /mycode/mail.py
        else
        python3 /mycode/failed_mail.py
        fi
```




Not secure | 192.168.99.100:32756/job/job2-test_code/configure
job2-test_code
General
Source Code Management
Build Triggers
Build
Post-build Actions
Build Triggers
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Projects to watch
job1-deploy-web-server
Trigger only if build is stable
Trigger even if the build is unstable
Trigger even if the build fails
Build periodically
GitHub hook trigger for GITScm polling
Poll SCM
Schedule
@daily
Would last have run at Tuesday, July 21, 2020 at 9:29:20 PM Coordinated Universal Time; would next run at Wednesday, July 22, 2020 at 9:29:20 PM Coordinated Universal Time.
Ignore post-commit hooks
Build
Save
Apply



For creating a pipeline view, we use the below code...

```
buildPipelineView('Task 6') {
            filterBuildQueue()
            filterExecutors()
            title('Project for deploy web-server')
            displayedBuilds(5)
            selectedJob('job1-deploy-web-server')
            alwaysAllowManualTrigger()
            showPipelineParameters()
            refreshFrequency(60)
        }
```

Pull all the code in one file and your groovy script is ready😍

GitHub Link... https://github.com/Anuddeeph/jenkins_dsl_code.git

Thanks for reading...😍

Give a thumbsup👍, if you like it...

If have any query, please DM me...

Let's connect with each other...😃

This is the submission of the **#6th** task given by Mr. **Vimal Daga** Sir under the DevOps assembly line training.

Normally we create Job in Jenkins manually, but there is one more way to create job using code. I try to deploy web-server using groovy code on the top of Kubernetes.

For more detail, please check the article once..

. GitHub Link: https://github.com/Anuddeeph/jenkins_dsl_code.git

#vimaldaga #automation #jenkins #dsl #kubernetes #docker #github #righteducation #devops