

[DevOps-Task-6]

# A multi-faceted language for the Java platform

Apacche groovy is a **powerful**, **optionally typed** and **dynamic** language, with **static-typing and static compilation** capabilities, for the Java platform aimed at improving developer productivity thanks to a concise, **familiar and easy to learn syntax**. It integrates smoothly with any Java program, and immediately delivers to your application powerful features, including scripting capabilities, **Domain-Specific Language** authoring, runtime and compile-time **meta-programming** and **functional** programming.

**Perform third task with the help of Jenkins coding file ( called as Jenkins file approach ) and perform the with following phases:**

* **Create container image that’s has Jenkins installed using dockerfile Or You can use the Jenkins Server on RHEL 8/7.**
* **When we launch this image, it should automatically starts Jenkins service in the container.**
* **Create a job chain of job1, job2, job3 and job4 using build pipeline plugin in Jenkins**
* **Job2 ( Seed Job ) : Pull the GitHub repo automatically when some developers push repo to GitHub.**
* **Further on jobs should be pipeline using written code using Groovy language by the developer**
* **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 ).**
* **Expose your pod so that testing team could perform the testing on the pod.**
* **Make the data to remain persistent using PVC ( If server collects some data like logs, other user information ).**
* **Test your app if it is working or not.**
* **if app is not working , then send email to developer with error messages and redeploy the application after code is being edited by the developer**

**Jenkins setup is exact same as Task-3 earlier.**

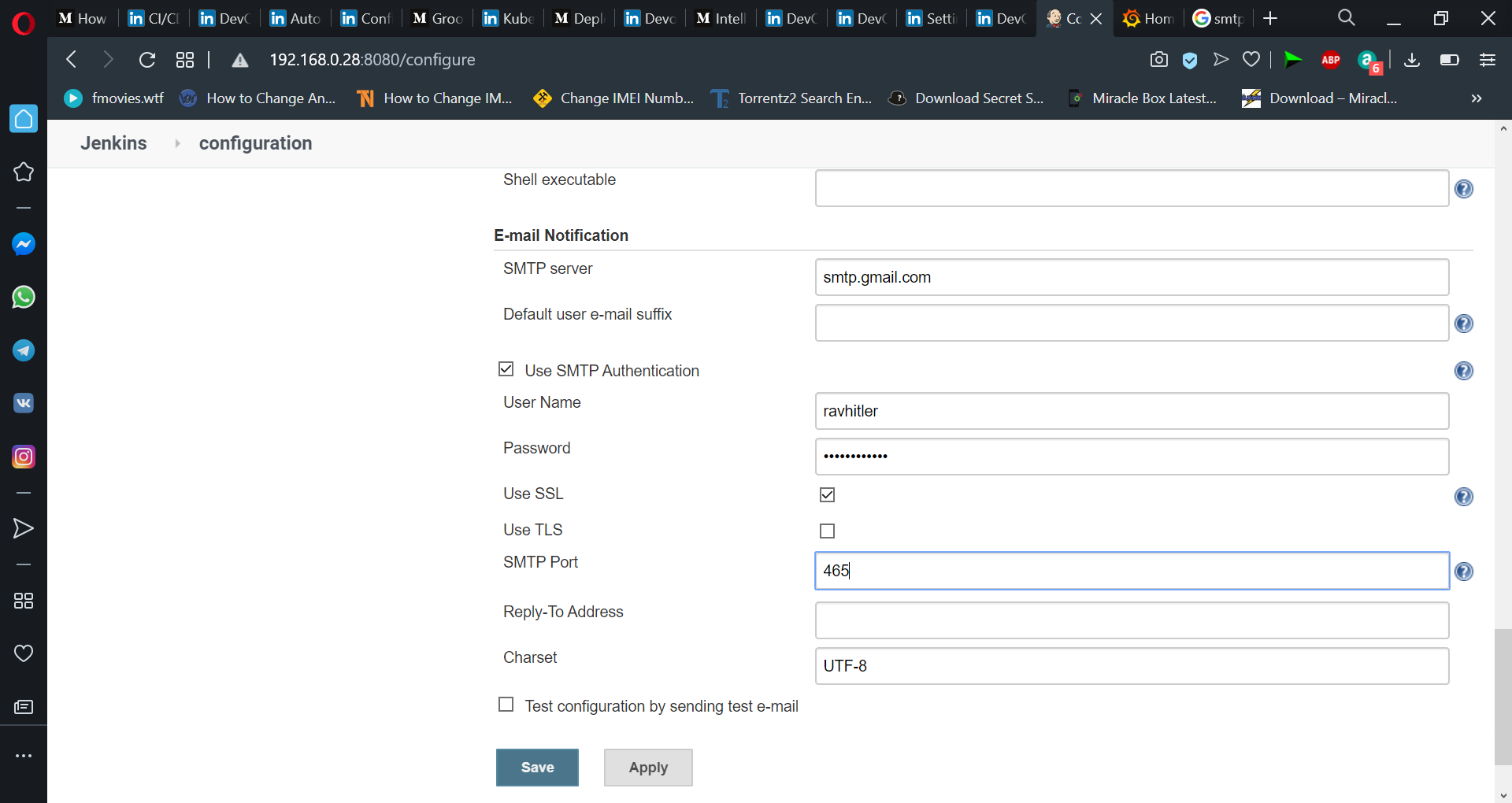
Plugins are required to be installed:

1. [Job DSL](https://plugins.jenkins.io/job-dsl).
2. [Security Script Plugin](https://plugins.jenkins.io/script-security).
3. [GitHub plugin](https://plugins.jenkins.io/github).
4. [Workspace Clean up Plugin](https://plugins.jenkins.io/ws-cleanup).
5. [Cloud Bees Docker Build and Publish plugin](https://plugins.jenkins.io/docker-build-publish).
6. [Email Extension Plugin](https://plugins.jenkins.io/email-ext).
7. [Build Pipeline Plugin](https://plugins.jenkins.io/build-pipeline-plugin).

To configure **SSH service** in Jenkins

**Manage Jenkins > Configuration system > SSH remote hosts > set hostname(IP address of base OS) and SSH Port(22) of your base OS.**

**Configuration of E-mail Notification**

****

Creation of yml file for kubernetes for Persistent Volume, Persistent Volume Claim, Result Set for php and html image.

**Start your minikube:-minikube start --driver=virtualbox**

Mount host directory to minikube

minikube ssh

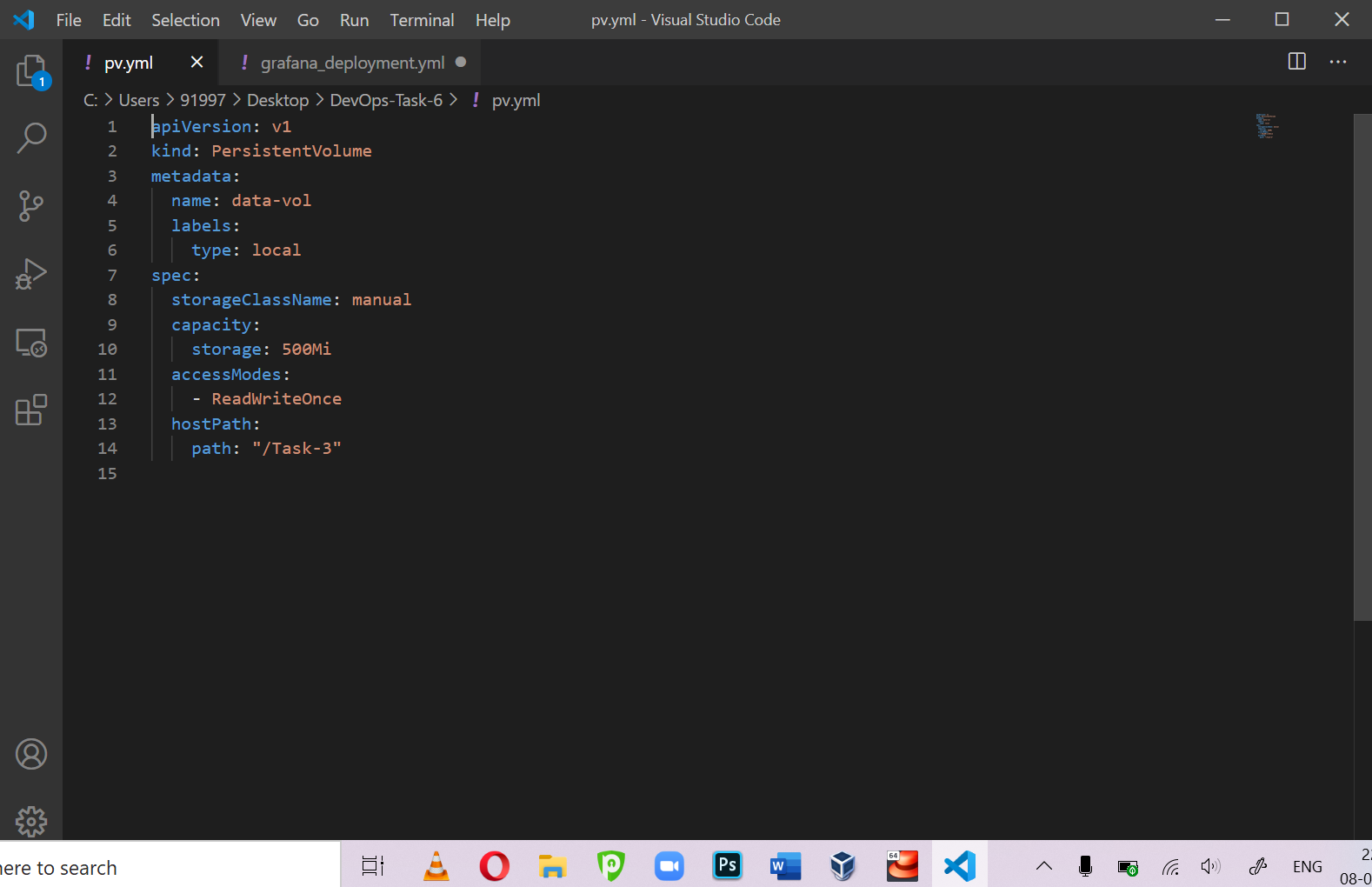
mkdir DevOps-Task-3/

Mount /home/ravi/DevOps-Task-3 directory to minikube Task-3 directory

minikube mount /home/ravi/DevOps-Task-3:/Task-3

**Mounting Task-3 folder from host machine**

**Pv.yml**



apiVersion: v1

kind: PersistentVolume

metadata:

  name: data-vol

  labels:

    type: local

spec:

  storageClassName: manual

  capacity:

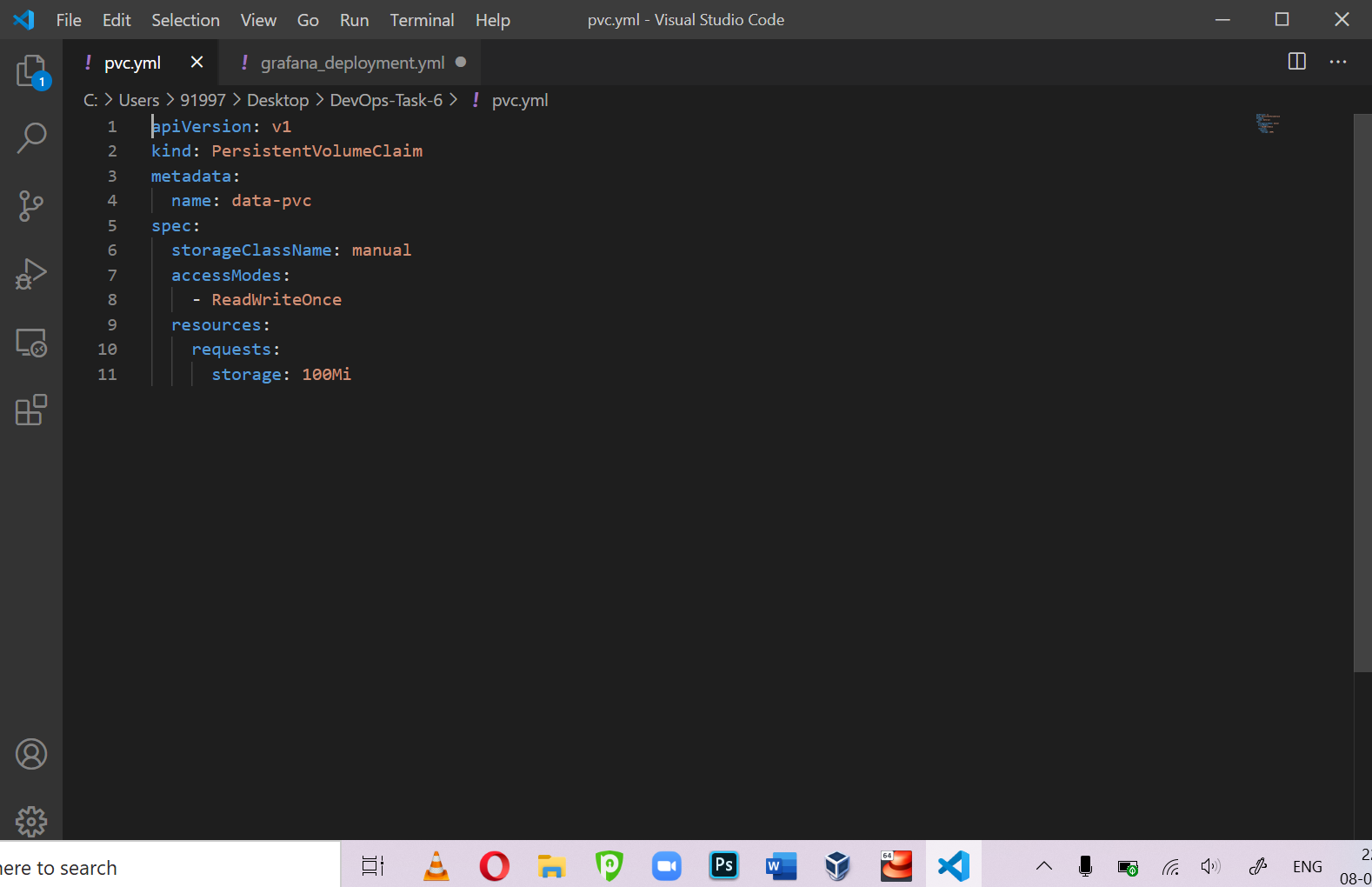
    storage: 500Mi

  accessModes:

    - ReadWriteOnce

  hostPath:

    path: "/Task-3"

****

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

  name: data-pvc

spec:

  storageClassName: manual

  accessModes:

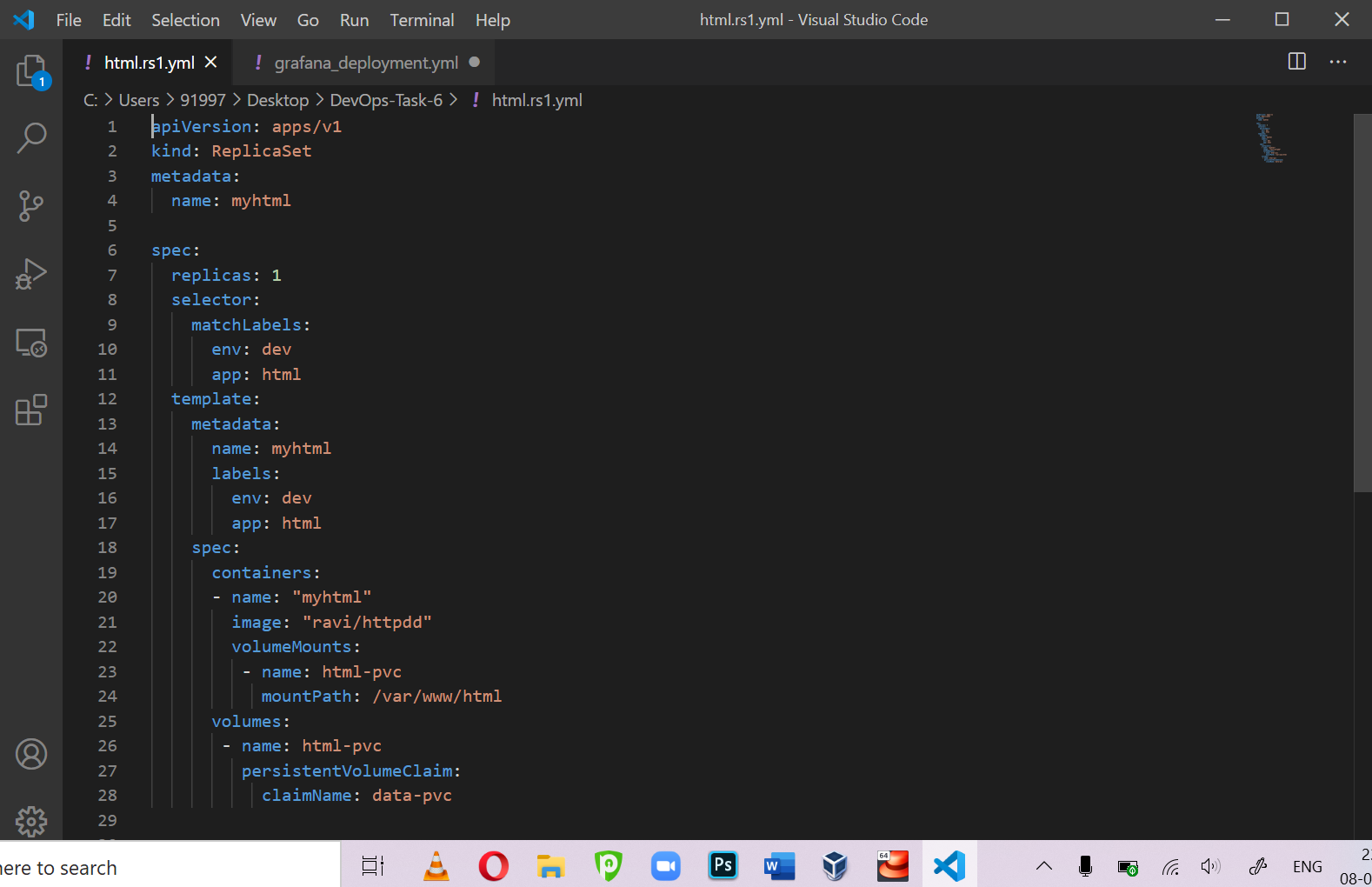
    - ReadWriteOnce

  resources:

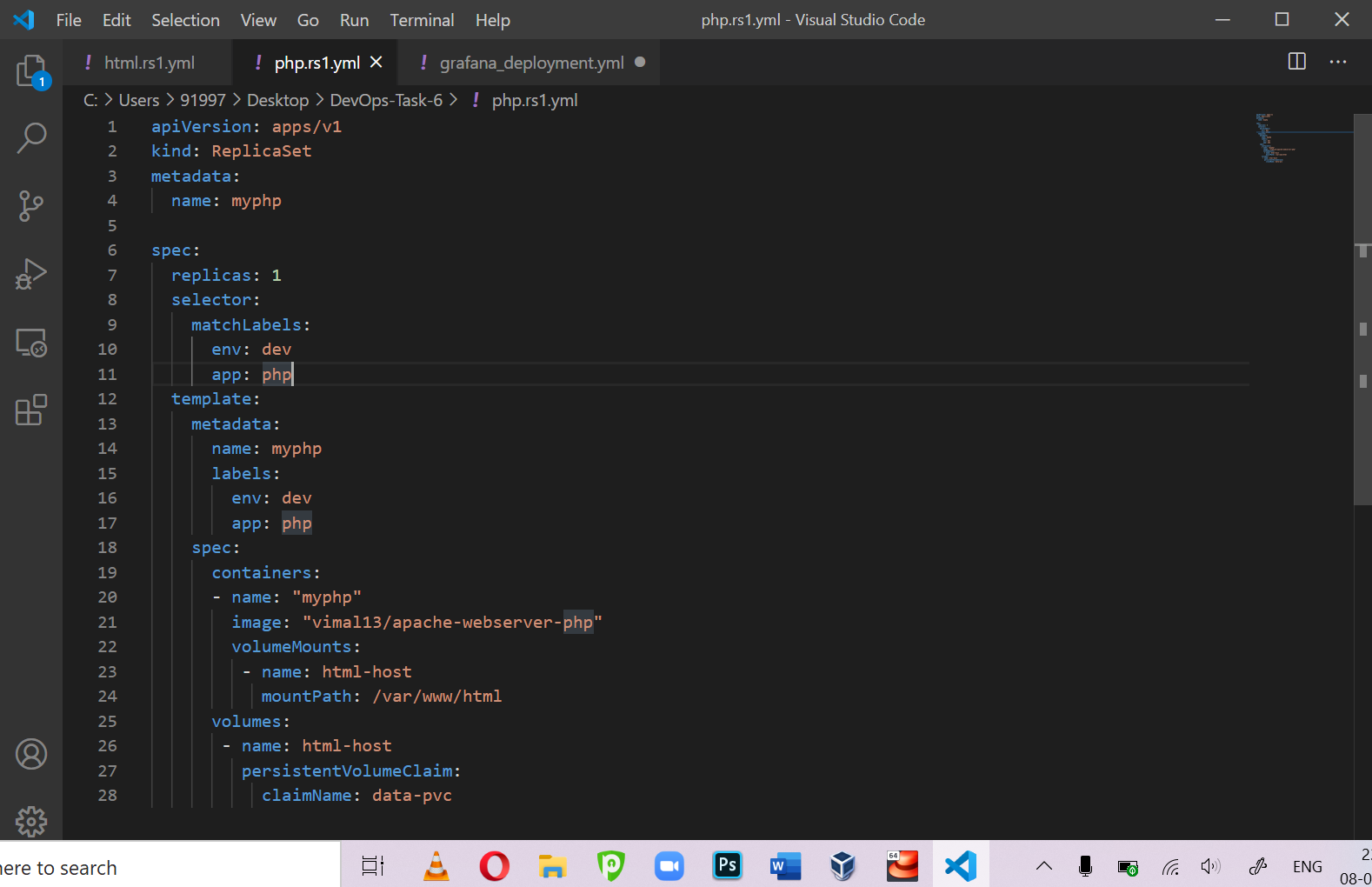
    requests:

      storage: 100Mi

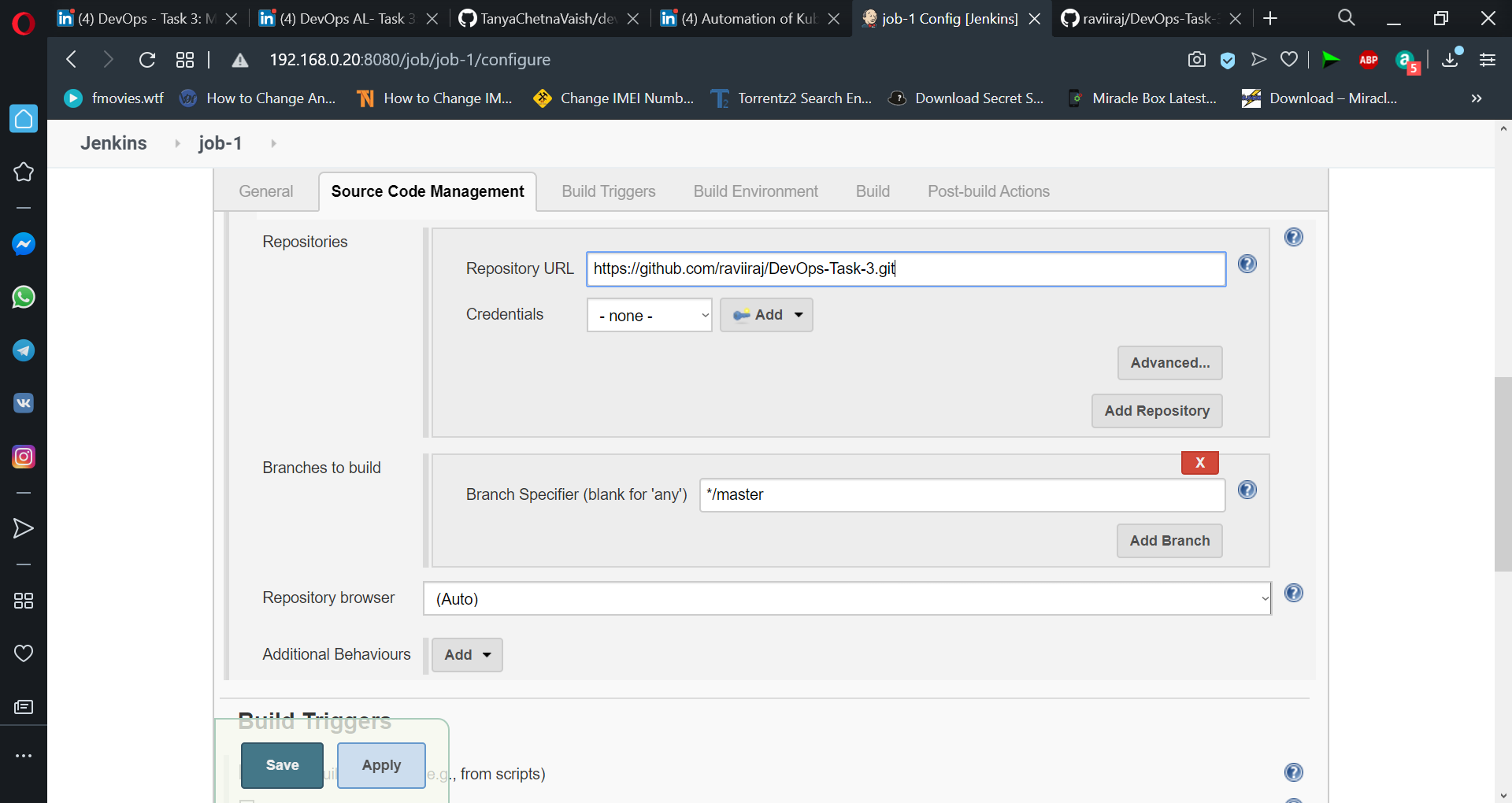
**ReplicaSet for html image**

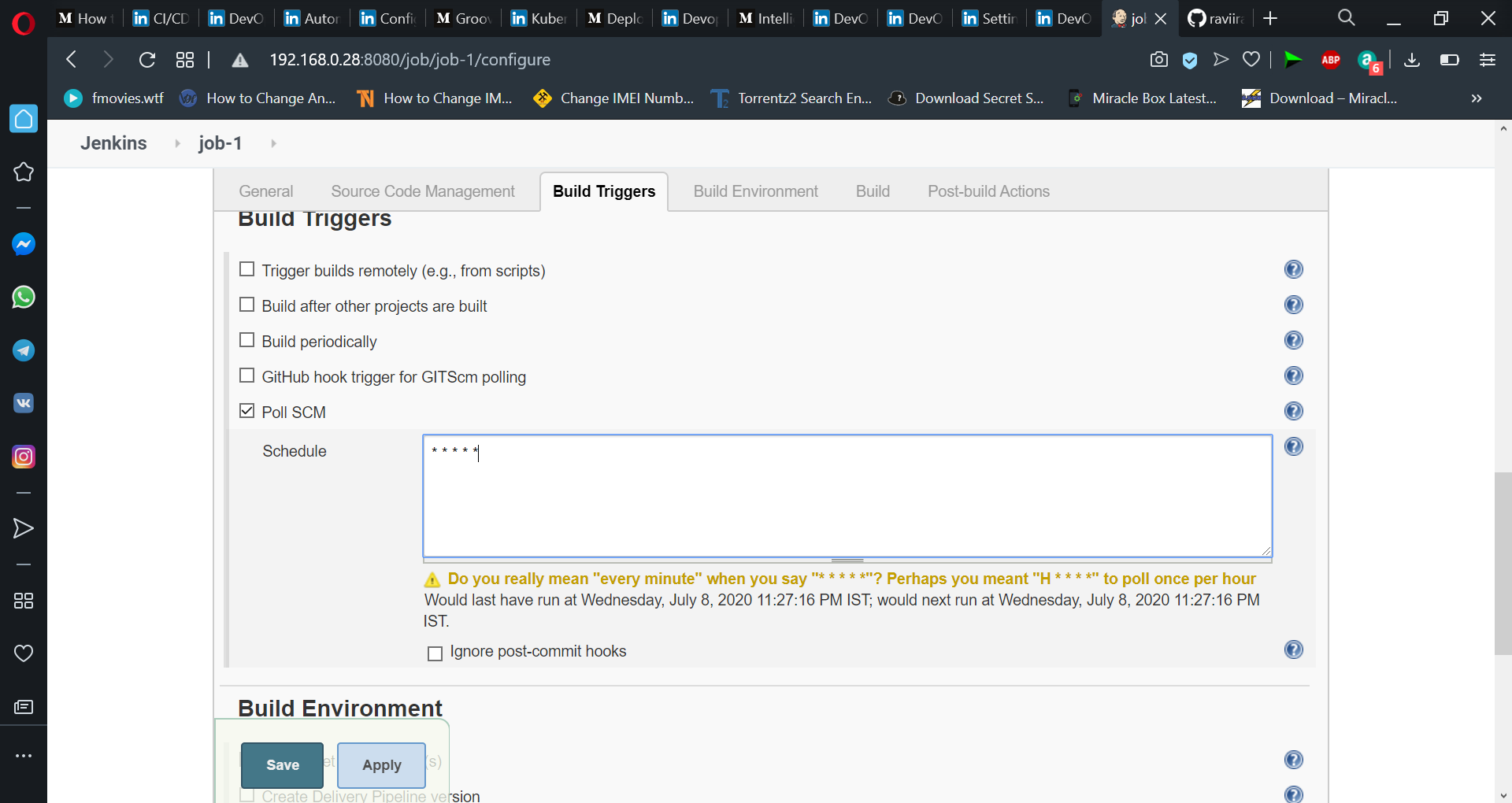
****

**ReplicaSet for php image**



Creation of Job-1 [ seed job ] same as did in Task-3





Now create DSL file of Jenkins

job('TASK-6 JOB-2') {

description("Job-2 task-6")

triggers {

upstream('TASK-6 JOB-1','SUCCESS')

}

steps {

shell('''

if ls /home/ravi/task3 | grep html

then

echo "HTML image launcing..."

kubectl apply -f /home/ravi/task-3-data/html-rs1.yml --kubeconfig /home/ravi/.kube/config

echo "exposing pod HTML"

kubectl expose rs myhtml --port 80 --type=NodePort --kubeconfig /home/ravi/.kube/config

# sleep for 100s to wait for container comes in running state

sleep 100

elif ls /home/ravi/task3 | grep php

then

echo "PHP image launcing..."

kubectl apply -f /home/ravi/task-3-data/php-rs1.yml --kubeconfig /home/ravi/.kube/config

echo "exposing pod PHP"

kubectl expose rs myphp --port 80 --type=NodePort --kubeconfig /home/ravi/.kube/config

# sleep for 100s to wait for container comes in running state

sleep 100

else

echo "Invalid extension JOB failed!!!"

exit 1

fi

#show output

kubectl get pods --kubeconfig /home/ravi/.kube/config

kubectl get service --kubeconfig /home/ravi/.kube/config

''')

}

}

job('TASK-6 JOB-3') {

description("job-3 Task-6")

triggers {

upstream('TASK-6 JOB-2','SUCCESS')

}

steps {

shell('''

if kubectl get pods --kubeconfig /home/ravi/.kube/config | grep myhtml

then

echo "myhtml running....."

port=$(kubectl get service --kubeconfig /home/ravi/.kube/config | grep myhtml | grep -Eo "[0-9]{5}")

ip=192.168.0.28

status=$(curl -o /dev/null -s -w "%{http\_code}" $ip:$port)

if [[ $status == 200 ]]

then

echo "html page deployed successfully"

exit 0

else

exit 1

fi

elif kubectl get pods --kubeconfig /home/ravi/.kube/config | grep myphp

then

echo "myphp running....."

port=$(kubectl get service --kubeconfig /home/ravi/.kube/config | grep myphp | grep -Eo "[0-9]{5}")

ip=192.168.0.28

status=$(curl -o /dev/null -s -w "%{http\_code}" $ip:$port)

if [[ $status == 200 ]]

then

echo "php page deployed successfully"

exit 0

else

exit 1

fi

fi

''')

}

publishers {

extendedEmail {

recipientList('veerliamg12345@gmail.com')

defaultSubject('Build Failed!!!')

defaultContent('Something broken')

contentType('text/html')

attachBuildLog(attachBuildLog = true)

triggers {

beforeBuild()

stillUnstable {

subject('Subject')

content('Body')

sendTo {

recipientList()

}

}

}

}

}

}

buildPipelineView('TASK6 VIEW') {

filterBuildQueue()

filterExecutors()

title('Task-6 Build Pipeline View')

displayedBuilds(1)

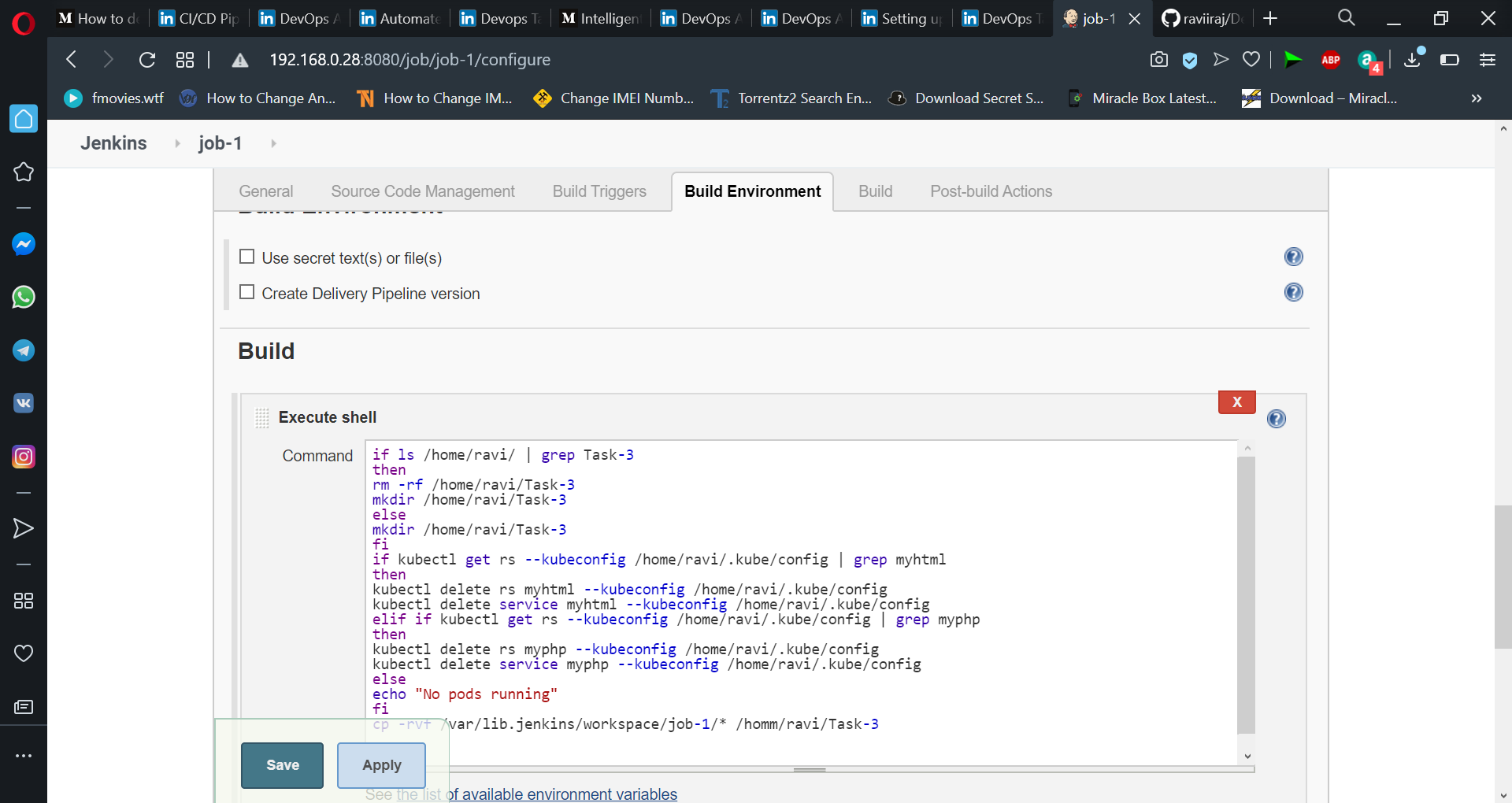
selectedJob('job-1')

alwaysAllowManualTrigger()

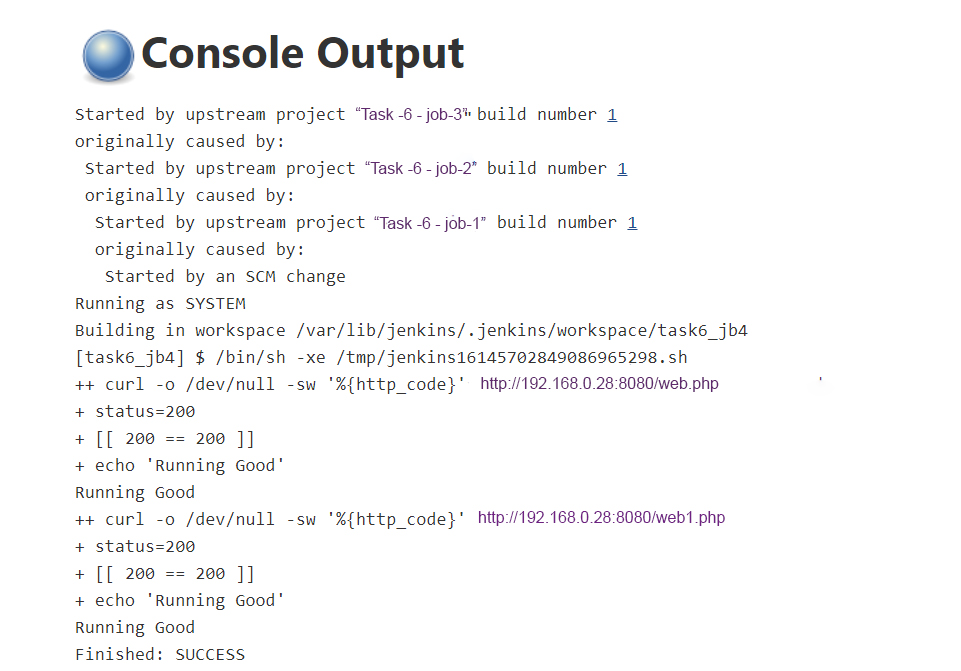
showPipelineParameters()

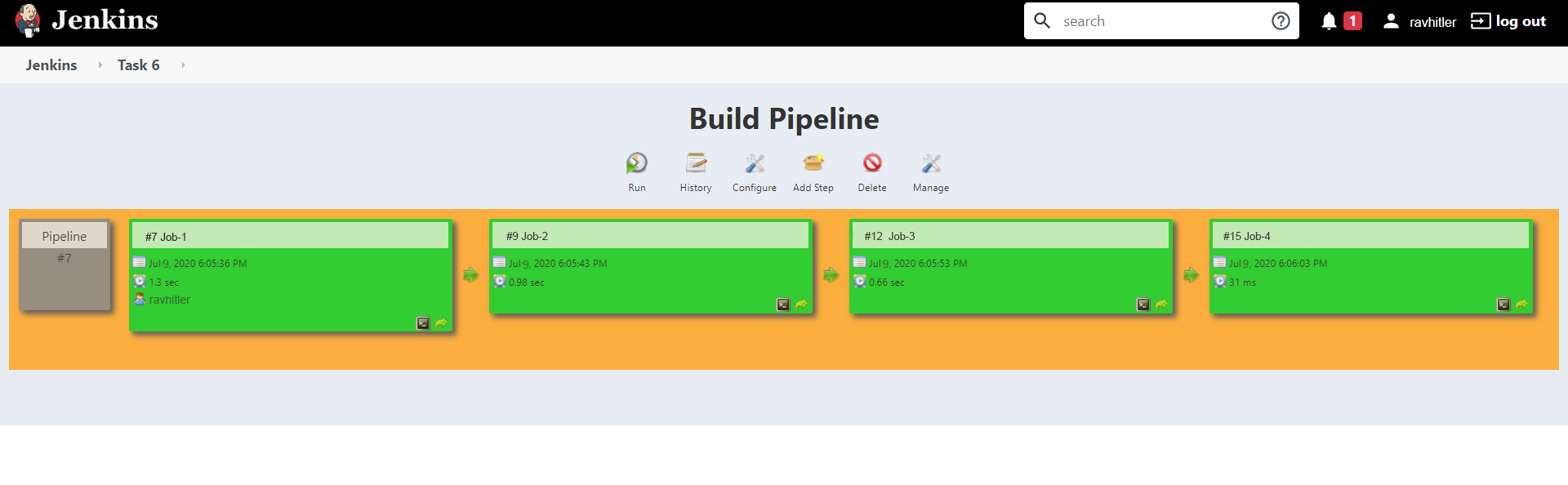
refreshFrequency(60)

}



/Job-1 is created finally. This is seed job so jenkinsfile will create all other jobs automatically.





Lastly completed Task-6 of DevOps AL

For more info and documents related to this project, checkout my GitHub link below: