Blue Green Deployment

Groovy Script:

This is my groovy script to implement blue-green deployment in Kubernetes.
The script contains different stages like

Stages:

1. Git Checkout: In this stage, I am checking out the master branch we can checkout any branch where we are punishing app artifact changes

2. Build Image: I am building the image in this stage and also I am using our last commit ID as an image tag to keep track of the image corresponding to git repo changes.

3. Push Image in image_registry: Then I push the image into my docker hub registry also I push the image with 2 tags one is commit id and the second one is latest.

3. Check current active color: Then I check which color env is active now by listing out active pods attached with a color environment label.

4. Apply new image version in the deactivated deployment: If blue is the active color now I will deploy new image in green and if green is the active color now I will deploy a new image to blue.

5. Shift the traffic to the new deployment: Upon successful deployment, I am shifting the traffic to our new deployment version. Note here I am shifting traffic by just attaching a new env path to my service YAML depending on the active color
6. Post Success: if all the above stages are successful then I am deactivating the deployment with an older image version and post the success msg to flock/slack.
Note: If we don't scale the deactivated deployment replicas to 0 and keep them running with n replicas, It ensures quick rollbacks but incurs additional costs for larger production applications. For my setup I kept it as 0 we can scale it with "n" replicas where "n" depends on application and its traffic.

7. Post Failure: If any of the above stages fails then I am again shifting traffic to the older version and also deactivate the newer deployment and post a failure msg to flock/slack.

Note: If we don't scale the deactivated deployment replicas to 0 and keep them running with n replicas, It ensures quick rollbacks but incurs additional costs for larger production applications.

For my setup I kept it as 0 we can scale it with "n" replicas where "n" depends on the application and its traffic.

In the post section, I am always deleting the directory I am working on. It will delete the pipeline build residue and help us to.

Parameters:

1. Commit_id: string parameter with default value as NA we can specify any commit id from my git repo where we want our app to be rolled back. It's part of the disaster recovery policy.

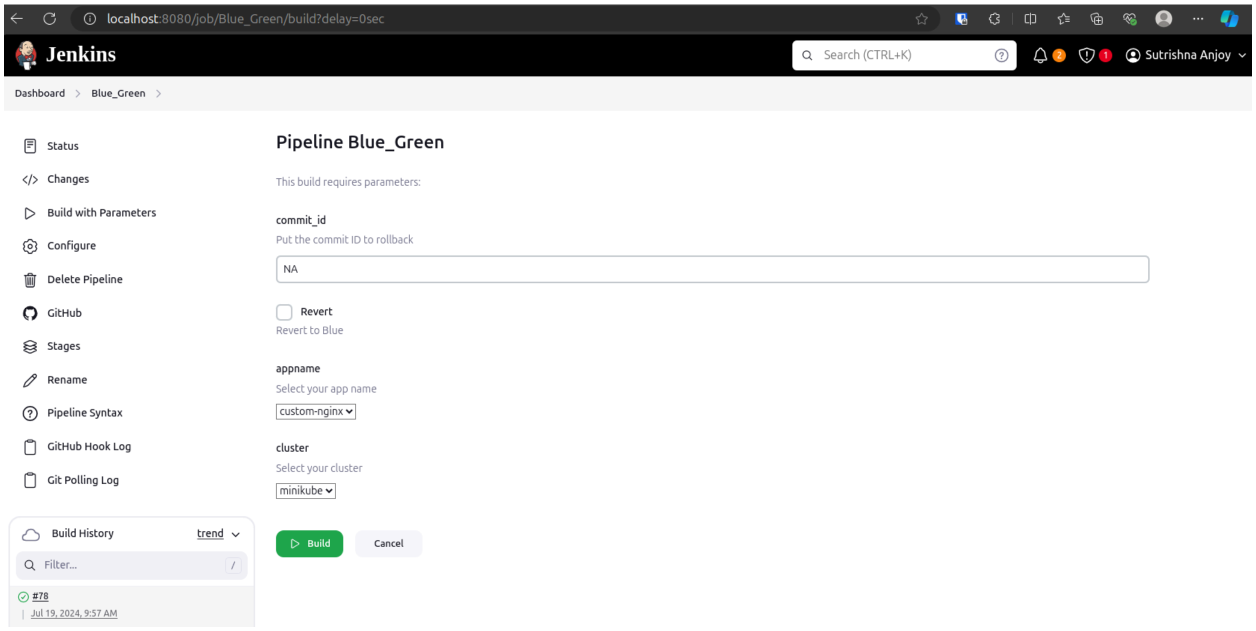
2. Revert: It's a boolean variable if we check the Revert box it will simply revert your deployment to the previous version and I am achieving it by checking out the second last commit. We can run this job as an escalation policy in the alert management system, it will revert us back to the previous version without human intervention.

3. appname: It's an extended choice parameter where we can choose the app name which we want to deploy, the default value is custom-nginx. I am using this app for testing.

4. cluster: Another extended choice parameter to choose your cluster in my case I have one cluster running locally in Minikube that's why Minikube is the default value for this parameter.

Git repo: https://github.com/sutrisnaanjoy19/blue-green

You will find my simple apps artifact for testing in the app folder, Kubernetes YAML in k8s-yaml folder, and jenkinsfile in jenkins folder



Pipeline Blue_Green

This build requires parameters:

commit_id
Put the commit ID to rollback
NA

Revert
Revert to Blue

appname
Select your app name
custom-nginx

cluster
Select your cluster
minikube

```
def image_registry='registry-1.docker.io/anjoysutrisna/'

def app_name= params.appname

def APP_PATH="apps/" + app_name

def image_name=image_registry + app_name

def tag=''

String COLOR_ACTIVE = ''

def YAML_PATH = 'k8s-yaml/' + app_name

def credentials_id = "my_k8s_updated"

def server_url = "https://192.168.49.2:8443"

def n_space = "default"


if(params.cluster == 'minikube' ){

   credentials_id = "my_k8s_updated"

   server_url = "https://192.168.49.2:8443"

   n_space = "default"

}




pipeline {
   agent any

   stages {

       stage('Git Checkout') {

           steps {

               script

               {

                   sh ("mkdir ./$BUILD_NUMBER")

                   dir("$BUILD_NUMBER"){

                   checkout([$class: 'GitSCM',

                       branches: [[name: 'master']],

                       doGenerateSubmoduleConfigurations: false,

                       extensions: [],

                       submoduleCfg: [],

                       userRemoteConfigs: [[url:
'https://github.com/sutrisnaanjoy19/blue-green.git']]])

                   sh 'git pull origin master'

                   sh 'pwd'

                   sh """ls -lah ${APP_PATH}"""

                   if ( params.Revert) {

                       tag = sh(returnStdout: true, script: """git
rev-parse --short @~""").trim()

                   }

                   else if ( params.commit_id == 'NA' ) {
                       tag = sh(returnStdout: true, script: """git log -1
--format=%h""").trim()
                   }
                   else {
                       tag = params.commit_id
                   }

                   echo tag
                   }
               }
           }
       }

       stage('Build Image') {
           steps {
               script
               {
                   dir("$BUILD_NUMBER"){
                       if ( !params.Revert && params.commit_id == 'NA') {
                           sh """docker build -t ${image_name}:${tag}
${APP_PATH}"""
                           sh """docker tag ${image_name}:${tag}
${image_name}:latest"""
                       }
                   }

               }

           }

       }

       stage('Push Image in image_registry') {

           steps {

               script{

                   dir("$BUILD_NUMBER"){

                       docker.withRegistry('https://registry-1.docker.io',
'hub_docker_com'){

                           if ( !params.Revert && params.commit_id ==
'NA') {

                               sh """docker push ${image_name}:${tag}"""

                               sh """docker push ${image_name}:latest"""

                           }

                       }

                   }

               }

           }

       }

       stage('Check current active color') {

           steps {

               script{
                   dir("$BUILD_NUMBER"){

                       kubeconfig(credentialsId: "${credentials_id}",
serverUrl: "${server_url}" ) {

                           String test = sh(script: """kubectl get pods
--selector app=${app_name},env=green -n ${n_space} |  head -n1 | cut -d "
" -f1 """, returnStdout: true).trim()

                           echo test

                           if( test == 'NAME' ){

                               COLOR_ACTIVE = 'green'

                               COLOR_DIACTIVATED = 'blue'

                           }

                           else {

                               COLOR_ACTIVE = 'blue'

                               COLOR_DIACTIVATED = 'green'

                           }

                       }

                   }

               }

           }

       }

       stage('apply new image version in deactivated deployment') {

           steps {

               script{

                   dir("$BUILD_NUMBER"){
                       kubeconfig(credentialsId: "${credentials_id}",
serverUrl: "${server_url}" ) {

                           sh """cat ${YAML_PATH}/deployment.yaml |
TARGET_COLOR=${COLOR_DIACTIVATED} TARGET_VERSION=${tag} envsubst | kubectl
apply -f -"""

                           sh """kubectl rollout status
deployment/${app_name}-deployment-${COLOR_DIACTIVATED} --timeout=120s"""

                       }

                   }

               }

           }

       }

       stage('shift the traffic to the new deployment') {

           steps {

               script{

                   dir("$BUILD_NUMBER"){

                       kubeconfig(credentialsId: "${credentials_id}",
serverUrl: "${server_url}" ) {

                           sh """cat ${YAML_PATH}/hpa.yaml |
TARGET_COLOR=${COLOR_DIACTIVATED} envsubst | kubectl apply -f -"""

                           sh """kubectl patch svc ${app_name}-service -p
'{\"spec\":{\"selector\":{\"env\":\"${COLOR_DIACTIVATED}\"}}}' -n default
"""



                       }

                   }

               }
           }

       }

   }

   post {

       success {

           script{

                   kubeconfig(credentialsId: "${credentials_id}",
serverUrl: "${server_url}" ) {

                       sh """kubectl scale
deployment/${app_name}-deployment-${COLOR_ACTIVE} --replicas=0"""



                   }

                   //sh """

                   //    curl -s -X POST --max-time 10 <YOUR_HOOK> -H
"Content-Type: application/json" -d '{ "text": "Deployment SUCCESS
\nBuild: ${BUILD_URL}" }'

                   //"""



           }

       }

       failure {

           script {

                   kubeconfig(credentialsId: "${credentials_id}",
serverUrl: "${server_url}" ) {
```

```
                        sh """kubectl patch svc ${app_name}-service -p
'{\"spec\":{\"selector\":{\"env\":\"${COLOR_ACTIVE}\"}}}' -n ${n_space}
"""

                        sh """kubectl scale
deployment/${app_name}-deployment-${COLOR_DIACTIVATED} --replicas=0"""



                    }

                    //sh """

                    //    curl -s -X POST --max-time 10 <YOUR_HOOK> -H
"Content-Type: application/json" -d '{ "text": "Deployment FAILED rolled
back to previous version \nBuild: ${BUILD_URL}" }'

                    //"""



            }

        }

        always{

            deleteDir()

        }

    }

}
```

Output of the Jenkins pipeline:

```
‘”Started by user Sutrishna Anjoy

[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins
in /var/lib/jenkins/workspace/Blue_Green
[Pipeline] {
```

```
[Pipeline] stage
[Pipeline] { (Git Checkout)
[Pipeline] script
[Pipeline] {
[Pipeline] sh
+ mkdir ./78
[Pipeline] dir
Running in /var/lib/jenkins/workspace/Blue_Green/78
[Pipeline] {
[Pipeline] checkout
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/sutrisnaanjoy19/blue-green.git
 > git init /var/lib/jenkins/workspace/Blue_Green/78 # timeout=10
Fetching upstream changes from
https://github.com/sutrisnaanjoy19/blue-green.git
 > git --version # timeout=10
 > git --version # 'git version 2.43.0'
 > git fetch --tags --force --progress --
https://github.com/sutrisnaanjoy19/blue-green.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
 > git config remote.origin.url
https://github.com/sutrisnaanjoy19/blue-green.git # timeout=10
 > git config --add remote.origin.fetch
+refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
 > git rev-parse origin/master^{commit} # timeout=10
Checking out Revision 508ee665f5d2eec7362a43a263fab5a3d6d49fbe
(origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 508ee665f5d2eec7362a43a263fab5a3d6d49fbe #
timeout=10
Commit message: "sooo"
 > git rev-list --no-walk 508ee665f5d2eec7362a43a263fab5a3d6d49fbe #
timeout=10
[Pipeline] sh
+ git pull origin master
From https://github.com/sutrisnaanjoy19/blue-green
 * branch            master     -> FETCH_HEAD
Already up to date.
[Pipeline] sh
+ pwd
/var/lib/jenkins/workspace/Blue_Green/78
[Pipeline] sh
```

```
+ ls -lah apps/custom-nginx
total 16K
drwxr-xr-x 2 jenkins jenkins 4.0K Jul 19 09:58 .
drwxr-xr-x 4 jenkins jenkins 4.0K Jul 19 09:58 ..
-rw-r--r-- 1 jenkins jenkins  108 Jul 19 09:58 Dockerfile
-rw-r--r-- 1 jenkins jenkins  452 Jul 19 09:58 index.html
[Pipeline] sh
+ git log -1 --format=%h
[Pipeline] echo
508ee66
[Pipeline] }
[Pipeline] // dir
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build Image)
[Pipeline] script
[Pipeline] {
[Pipeline] dir
Running in /var/lib/jenkins/workspace/Blue_Green/78
[Pipeline] {
[Pipeline] sh
+ docker build -t
registry-1.docker.io/anjoysutrisna/custom-nginx:508ee66
apps/custom-nginx
#0 building with "default" instance using docker driver

#1 [internal] load build definition from Dockerfile
#1 transferring dockerfile: 145B done
#1 DONE 0.0s

#2 [internal] load metadata for docker.io/library/nginx:1.10.1-alpine
#2 DONE 2.8s

#3 [internal] load .dockerignore
#3 transferring context: 2B done
#3 DONE 0.0s

#4 [1/2] FROM
docker.io/library/nginx:1.10.1-alpine@sha256:dabd1d182f12e2a7d372338d
fd0cde303ef042a6ba01cc829ef464982f9c9e2c
#4 DONE 0.0s
```

```
#5 [internal] load build context
#5 transferring context: 491B done
#5 DONE 0.0s

#6 [2/2] COPY index.html /usr/share/nginx/html
#6 CACHED

#7 exporting to image
#7 exporting layers done
#7 writing image
sha256:8162492dcf55d02bee19b388d41ffa7a9306317cf5fe7cf6d7d9310dbaf137
71 done
#7 naming to registry-1.docker.io/anjoysutrisna/custom-nginx:508ee66
done
#7 DONE 0.0s
[Pipeline] sh
+ docker tag registry-1.docker.io/anjoysutrisna/custom-nginx:508ee66
registry-1.docker.io/anjoysutrisna/custom-nginx:latest
[Pipeline] }
[Pipeline] // dir
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Push Image in image_registry)
[Pipeline] script
[Pipeline] {
[Pipeline] dir
Running in /var/lib/jenkins/workspace/Blue_Green/78
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] withDockerRegistry
$ docker login -u anjoysutrisna -p ********
https://registry-1.docker.io
WARNING! Using --password via the CLI is insecure. Use
--password-stdin.
WARNING! Your password will be stored unencrypted in
/var/lib/jenkins/workspace/Blue_Green/78@tmp/c4bb7037-d45f-47e1-930b-
7ff38b473e26/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentia
l-stores
Login Succeeded
[Pipeline] {
[Pipeline] sh
+ docker push registry-1.docker.io/anjoysutrisna/custom-nginx:508ee66
The push refers to repository
[registry-1.docker.io/anjoysutrisna/custom-nginx]
68c30669bc19: Preparing
3e214109dd04: Preparing
5080acfc4b83: Preparing
0fd716f781a0: Preparing
011b303988d2: Preparing
011b303988d2: Layer already exists
3e214109dd04: Layer already exists
0fd716f781a0: Layer already exists
68c30669bc19: Layer already exists
5080acfc4b83: Layer already exists
508ee66: digest:
sha256:51226406c3f6091ac88da99344feac68df43e82e3eb1ce42034d12ad47c384
35 size: 1361
[Pipeline] sh
+ docker push registry-1.docker.io/anjoysutrisna/custom-nginx:latest
The push refers to repository
[registry-1.docker.io/anjoysutrisna/custom-nginx]
68c30669bc19: Preparing
3e214109dd04: Preparing
5080acfc4b83: Preparing
0fd716f781a0: Preparing
011b303988d2: Preparing
0fd716f781a0: Layer already exists
68c30669bc19: Layer already exists
5080acfc4b83: Layer already exists
3e214109dd04: Layer already exists
011b303988d2: Layer already exists
latest: digest:
sha256:51226406c3f6091ac88da99344feac68df43e82e3eb1ce42034d12ad47c384
35 size: 1361
[Pipeline] }
[Pipeline] // withDockerRegistry
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // dir
[Pipeline] }
[Pipeline] // script
[Pipeline] }
```

```
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Check current active color)
[Pipeline] script
[Pipeline] {
[Pipeline] dir
Running in /var/lib/jenkins/workspace/Blue_Green/78
[Pipeline] {
[Pipeline] kubeconfig
[Pipeline] {
[Pipeline] sh
+ kubectl get pods --selector app=myapp,env=green -n default
+ + cut -d  head -f1 -n1

No resources found in the default namespace.
[Pipeline] echo

[Pipeline] }
[Pipeline] // kubeconfig
[Pipeline] }
[Pipeline] // dir
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (apply new image version in deactivated deployment)
[Pipeline] script
[Pipeline] {
[Pipeline] dir
Running in /var/lib/jenkins/workspace/Blue_Green/78
[Pipeline] {
[Pipeline] kubeconfig
[Pipeline] {
[Pipeline] sh
+ cat k8s-yaml/custom-nginx/deployment.yaml
+ TARGET_COLOR=green TARGET_VERSION=508ee66 envsubst
+ kubectl apply -f -
deployment.apps/custom-nginx-deployment-green configured
[Pipeline] sh
+ kubectl rollout status deployment/custom-nginx-deployment-green
--timeout=120s
deployment "custom-nginx-deployment-green" successfully rolled out
[Pipeline] }
[Pipeline] // kubeconfig
[Pipeline] }
[Pipeline] // dir
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (shift the traffic to the new deployment)
[Pipeline] script
[Pipeline] {
[Pipeline] dir
Running in /var/lib/jenkins/workspace/Blue_Green/78
[Pipeline] {
[Pipeline] kubeconfig
[Pipeline] {
[Pipeline] sh
+ cat k8s-yaml/custom-nginx/hpa.yaml
+ TARGET_COLOR=green envsubst
+ kubectl apply -f -
horizontalpodautoscaler.autoscaling/myapp-hpa unchanged
[Pipeline] sh
+ kubectl patch svc custom-nginx-service -p
{"spec":{"selector":{"env":"green"}}} -n default
service/custom-nginx-service patched (no change)
[Pipeline] }
[Pipeline] // kubeconfig
[Pipeline] }
[Pipeline] // dir
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] deleteDir
[Pipeline] script
[Pipeline] {
[Pipeline] kubeconfig
[Pipeline] {
[Pipeline] sh
+ kubectl scale deployment/custom-nginx-deployment-blue --replicas=0
deployment.apps/custom-nginx-deployment-blue scaled
[Pipeline] }
[Pipeline] // kubeconfig
[Pipeline] }
```

```
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS'"
```

Setups:

Jenkins setup: I am running Jenkins in my system and have installed it using the official installation specified in https://www.jenkins.io/doc/book/installing/linux/
After installation, I need to check if it's running fine by checking systemctl status

```
sudo systemctl enable jenkins
sudo systemctl start jenkins
sudo systemctl status jenkins
```

If everything looks, the file opens http://localhost:8080 and installs required plugins like git, docker, Kubernetes, and extended choice parameter plugin.

Minikube setup: I have used the Minikube official document to install Minikube https://minikube.sigs.k8s.io/docs/start/?arch=%2Flinux%2Fx86-64%2Fstable%2Fbinary+download after setting up I start the Minikube cluster using docker as a driver like below

```
minikube start --driver=docker
```

Also to add the Minikube credential in Jenkins we need to modify our .kube/config file and upload it as a secret file.
Copy your .kube/config file into another file like below and get the base64 encoded value of my client.crt.client.key and ca.crt file

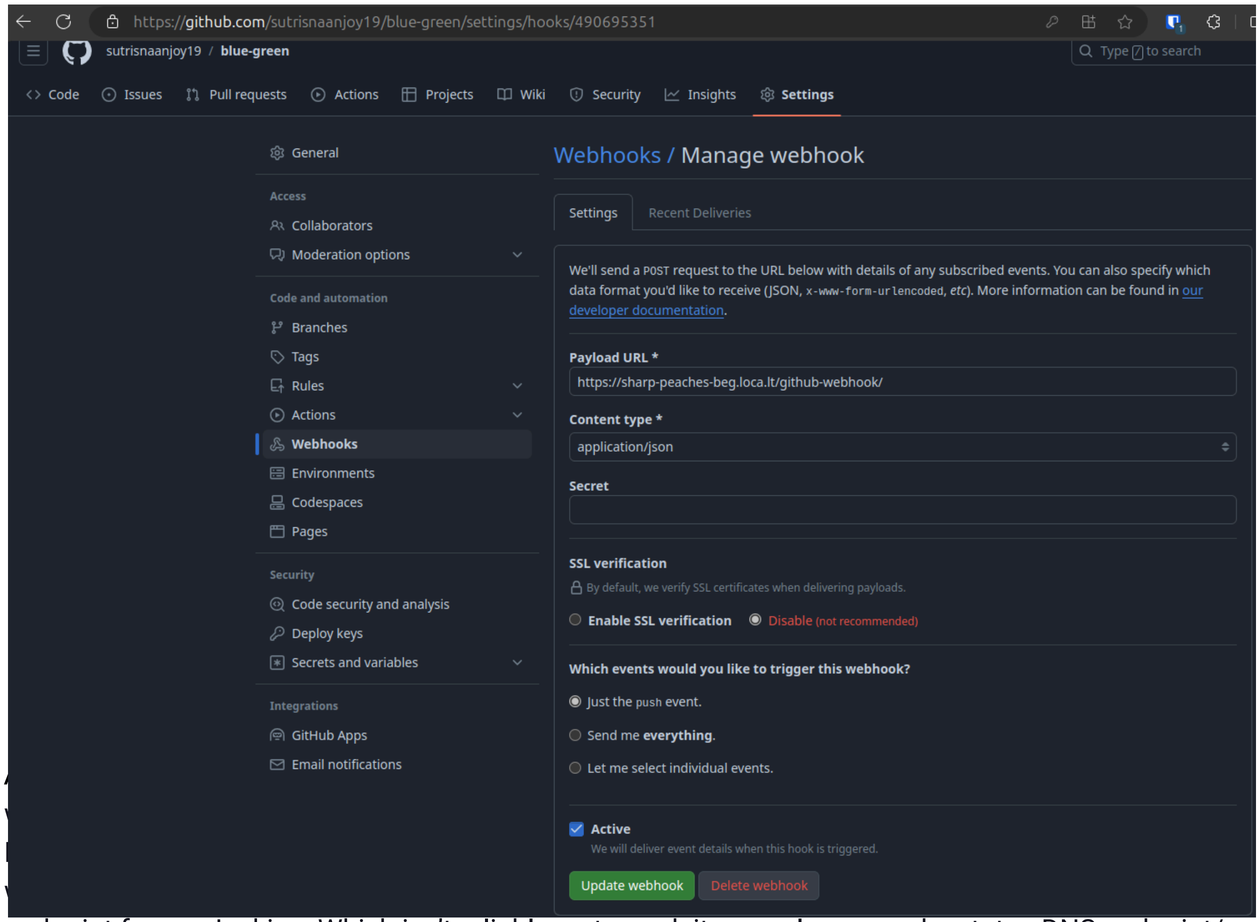
```
cp .kube/config config
cat /home/sushi/.minikube/ca.crt | base64 -w 0
cat /home/sushi/.minikube/profiles/minikube/client.crt | base64 -w 0
cat /home/sushi/.minikube/profiles/minikube/client.key | base64 -w 0
```

Paste the encoded secret into my config file by replacing the file path and also change the key name with an extension of -data.
Then manage Jenkins> credentials > new credential > secret file
Uploaded my config file here. Also, I need to create a cloud go to manage jenkins > cloud > new cloud > Kubernetes> select my config file. And finally, test the connection.

Image registry: I already have a docker hub account just create a API token by Account settings > Personal Access Token > create my token with read and write permission copy the token and go to Jenkins manage Jenkins > credentials > new credentials > username/password Paste the API token in the password field and fill in the username with myr docker hub username.

Git webhook: I already have a GitHub account I go to setting of my GitHub profile settings > develop settings > personal access token > classic token create a new token for my Jenkins with specific permission and then create a username/password based credential my manage Jenkins> credentials > new credential > username/password here use GitHub API token as a password and my GitHub username as username

To run the job automatically when a commit happens in a specific branch or a new branch/tag is created in the git repo we need to create a webhook by blue-green repo > settings > webhook > new webhook in the payload section we need to paster the Jenkins URL and add GitHub-webhook and select content-type as application/json and choose when we want to trigger the hook and make it active like below



Webhooks / Manage webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in our developer documentation.

Payload URL *
https://sharp-peaches-beg.loca.lt/github-webhook/

Content type *
application/json

Secret

SSL verification
By default, we verify SSL certificates when delivering payloads.
Enable SSL verification [✓ Disable (not recommended)]

Which events would you like to trigger this webhook?
[✓ Just the push event.]
Send me everything.
Let me select individual events.

[✓ Active]
We will deliver event details when this hook is triggered.

endpoint for my Jenkins. Which isn't reliable so to work it properly we need a status DNS endpoint/ public IP for the Jankins host.

Disaster recovery plan and rollback procedure: If any stage of the pipeline fails it will automatically be rolled back to the previous version it is handled by the post-failure stage and you will get the msg of failure and rolled back.
If after some time of deploying it fails we can always revert back to the previous version using the revert parameter if the previous version isn't also working we can go back to a much older version using an older commit ID which we can collect from git.

Test Cases and Validation: Checked the pipeline for every test case revert or using commit ID or automatic trigger of manual trigger pipeline is working fine for all cases. In case of a new version failure rollback is also working fine.

Monitoring and Logging: We can use the Kube Prometheus stack with Grafana to monitor the time series data of our application infra. Also, we should monitor the traffic of our load balancer. We need to set alerts of Prometheus if service is unavailable or pod isn't up or the endpoint isn't reachable we are receiving HTTP codes like 3xx or 4xx or 5xx the node is down or we are seeing crashloopbackoff ot imagepullbackoff error. And we should also set the alert receiver as pager duty or flock/slack. We can also automate escalation policy over pager duty if the service is unavailable we can call the Jenkins pipeline with Revert option as true it will revert it to the previous state without any human intervention.