Name: Pranav Chaudhari

Task: Argo CD for a sample helloworld application in Kubernetes and achieve below subtasks

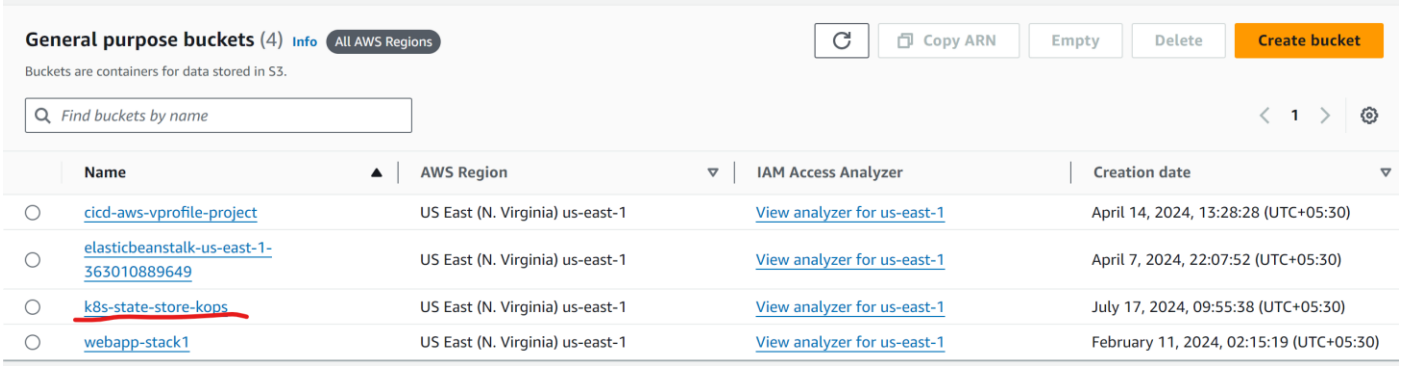
1. Installation and configuration of Argo CD.
2. K8s with one master and 2 worker nodes.
3. How to deploy a sample application using Argo CD into K8S.
4. Create a sample app deployment with Argo CD using custom Docker images.
5. How the Argo CD deploys latest docker images into K8s.

Dockerhub image: https://hub.docker.com/repository/docker/techdecipher/custom-apache

Github Repository: https://github.com/techdecipher/argocd/

1) K8s with one master and 2 worker nodes:

Step1] First, an a cloud bucket is required to store the state information of cluster when we try to create clusters with kops, since we are using AWS, creating S3 bucket in the same region as of the EC2 instance.



Step2] Exporting ENV variables as its beneficial for managing cluster then re-type it.

```
ubuntu@ip-172-31-15-226:~$ export KOPS_STATE_STORE=s3://k8s-state-store-kops
ubuntu@ip-172-31-15-226:~$ export KOPS_CLUSTER_NAME=techdecipher.xyz
ubuntu@ip-172-31-15-226:~$
```

Step3] Since we need 1 master node and 2 worker nodes, the command we use as below.

```
kops create cluster --name=${KOPS_CLUSTER_NAME} --zones=us-east-1a --node-count=2 --node-size=t2.medium --master-size=t2.medium --state=${KOPS_STATE_STORE}
```

```
ubuntu@ip-172-31-15-226:~$ kops create cluster --name=${KOPS_CLUSTER_NAME} --zones=us-east-1a --node-count=2 --node-size
=t2.medium --master-size=t2.medium --state=${KOPS_STATE_STORE}
I0717 04:54:54.461122    2059 new_cluster.go:1354] Cloud Provider ID: "aws"
I0717 04:54:54.579989    2059 subnets.go:185] Assigned CIDR 172.20.32.0/19 to subnet us-east-1a
Previewing changes that will be made:
```

Step4] Updating it as per next steps in cluster creation and seems to take some time.

```
kops update cluster --name=${KOPS_CLUSTER_NAME} --state=${KOPS_STATE_STORE} --yes --admin
```

```
kOps has set your kubectl context to techdecipher.xyz

Cluster is starting.  It should be ready in a few minutes.

Suggestions:
 * validate cluster: kops validate cluster --wait 10m
 * list nodes: kubectl get nodes --show-labels
 * ssh to a control-plane node: ssh -i ~/.ssh/id_rsa ubuntu@api.techdecipher.xyz
 * the ubuntu user is specific to Ubuntu. If not using Ubuntu please use the appropriate user based on your OS.
 * read about installing addons at: https://kops.sigs.k8s.io/addons.

ubuntu@ip-172-31-15-226:~$
```

Step5] After this, run validate to see if the cluster is ready and it is ready.

```
kops validate cluster --state=${KOPS_STATE_STORE}
```

```
ubuntu@ip-172-31-15-226:~$ kops validate cluster --state=${KOPS_STATE_STORE}
Validating cluster techdecipher.xyz

INSTANCE GROUPS
NAME                          ROLE            MACHINETYPE     MIN     MAX     SUBNETS
control-plane-us-east-1a      ControlPlane    t2.medium       1       1       us-east-1a
nodes-us-east-1a              Node            t2.medium       2       2       us-east-1a

NODE STATUS
NAME                    ROLE            READY
i-0588ba5f481d9f3f3     control-plane   True
i-0a9cd7c4f34cf897a     node            True
i-0d9f0489e3a0b5f2d     node            True

Your cluster techdecipher.xyz is ready
ubuntu@ip-172-31-15-226:~$
```

Step6] Checking the node status for 1 master and 2 worker nodes.

```
kubectl get nodes
```

```
ubuntu@ip-172-31-15-226:~$ kubectl get nodes
NAME                    STATUS   ROLES           AGE     VERSION
i-0588ba5f481d9f3f3     Ready    control-plane   8m5s    v1.26.15
i-0a9cd7c4f34cf897a     Ready    node            5m52s   v1.26.15
i-0d9f0489e3a0b5f2d     Ready    node            5m49s   v1.26.15
ubuntu@ip-172-31-15-226:~$
```

2) Installation and configuration of Argo CD:

Step1] Installation commands right through kubectl for Argo CD installation.

```
kubectl create namespace argocd

kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
```

```
ubuntu@ip-172-31-15-226:~$ kubectl create namespace argocd
namespace/argocd created
ubuntu@ip-172-31-15-226:~$ kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install
.yaml
customresourcedefinition.apiextensions.k8s.io/applications.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/applicationsets.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/appprojects.argoproj.io created
```

Step2] Now, changing the service type from cluster IP to Loadbalancer so we can access the Argo CD UI through browser. Also getting the external IP to access it.

```
kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'

kubectl get svc -n argocd
```

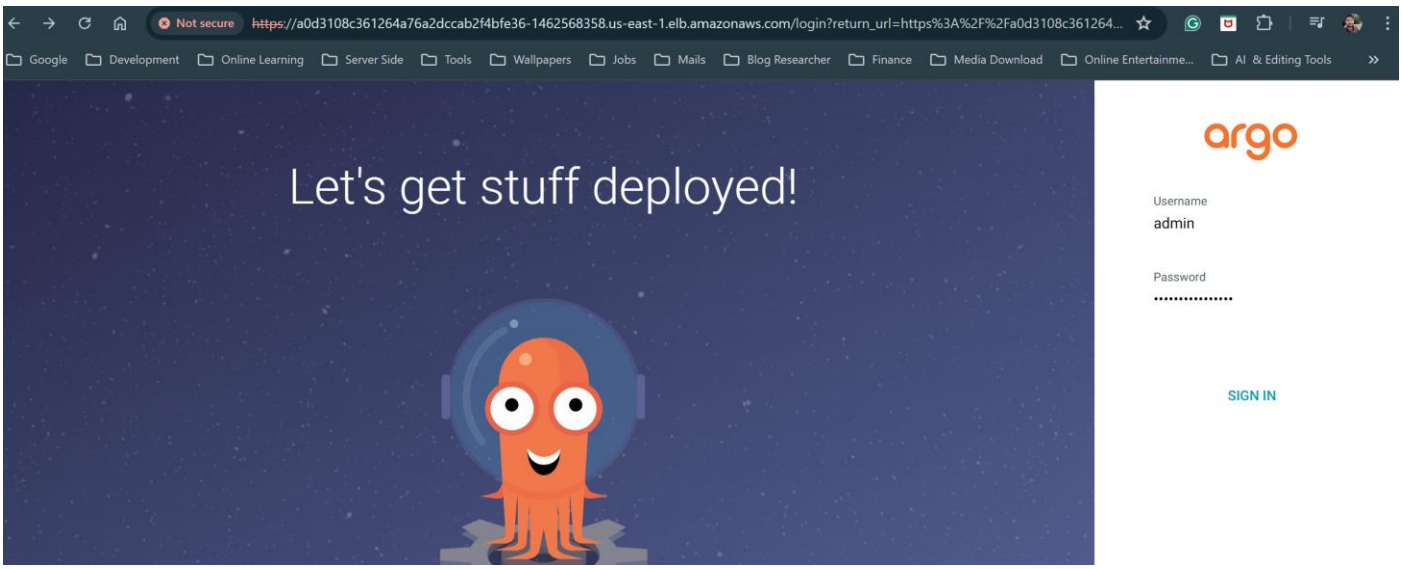
```
ubuntu@ip-172-31-15-226:~$ kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'
service/argocd-server patched
ubuntu@ip-172-31-15-226:~$ kubectl get svc -n argocd
NAME                                      TYPE           CLUSTER-IP       EXTERNAL-IP
                          PORT(S)                        AGE
argocd-applicationset-controller          ClusterIP      100.65.126.143   <none>
                          7000/TCP,8080/TCP              7m46s
argocd-dex-server                         ClusterIP      100.68.11.81     <none>
                          5556/TCP,5557/TCP,5558/TCP     7m46s
argocd-metrics                            ClusterIP      100.67.185.217   <none>
                          8082/TCP                       7m46s
argocd-notifications-controller-metrics   ClusterIP      100.65.181.72    <none>
                          9001/TCP                       7m46s
argocd-redis                              ClusterIP      100.65.43.106    <none>
                          6379/TCP                       7m46s
argocd-repo-server                        ClusterIP      100.71.115.58    <none>
                          8081/TCP,8084/TCP              7m46s
argocd-server                             LoadBalancer   100.65.145.193   a0d3108c361264a76a2dccab2f4bfe36-1462568358.us
-east-1.elb.amazonaws.com   80:32101/TCP,443:31043/TCP   7m46s
argocd-server-metrics                     ClusterIP      100.64.228.117   <none>
                          8083/TCP                       7m46s
ubuntu@ip-172-31-15-226:~$ date
Wed Jul 17 06:08:49 UTC 2024
ubuntu@ip-172-31-15-226:~$
```

Step3] Get the secret password to login to web UI

```
kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 --decode
```

```
ubuntu@ip-172-31-15-226:~$ kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | bas
e64 --decode
X-AW6kZmYmZ3S3zgubuntu@ip-172-31-15-226:~$
```

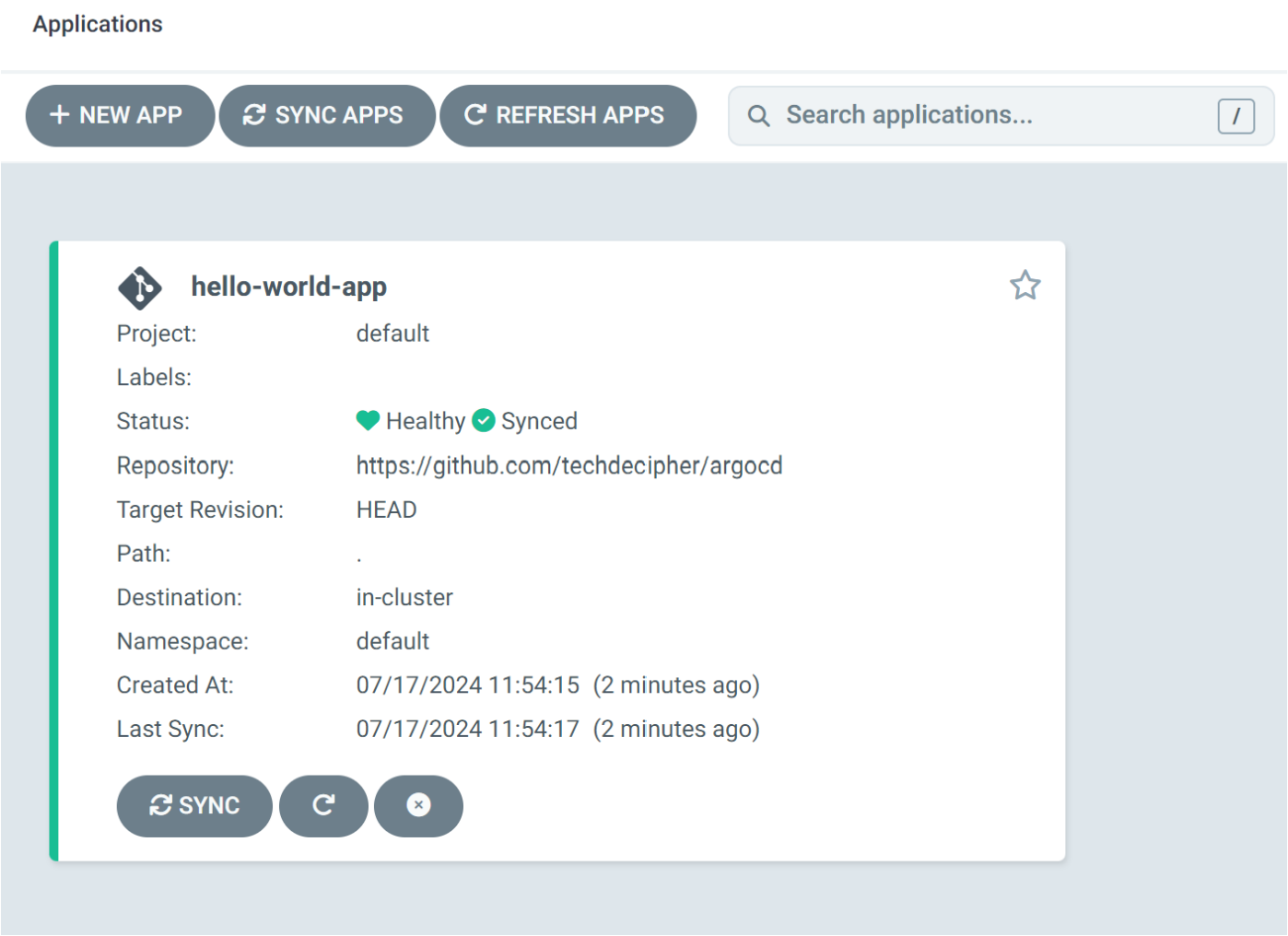
Step4] Login to WebUI of ArgoCD .



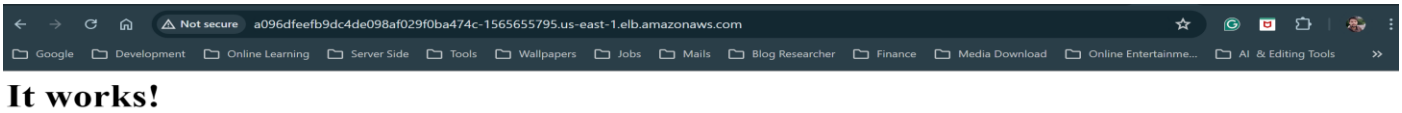
3)How to deploy a sample application using Argo CD into K8S:

Step1] Login to ArgoCD and create new App with respect to below details.

Application Name: hello-world-app
Project: default
Sync Policy: Automatic
Repository URL: https://github.com/techdecipher/argocd
Path: Path to your manifests
Cluster URL: https://kubernetes.default.svc
Namespace: default



Step2] Verify if it is deployed



It works!

4] Create a sample app deployment with Argo CD using custom Docker images:

Step1] install docker and write a sample index file and just build and push it to the dockerhub repo which then is pulled through git as we sync from ArgoCD.

Custom Index file

```
<!DOCTYPE html><html lang="en"><head> <meta charset="UTF-8">
  <title> Hello World!</title> </head>
<body>
  <h1>Hello World!</h1>
  <p>This is my Apache server running on Kubernetes!</p>
  <p>I did one change now.</p>
  <p>This is new demo.</p>
  <p>2 changes made.</p>
  <p>3 changes made.</p>
  <p>4 changes made.</p>
  <p>5 changes made.</p>
  <p>55 changes made.</p>
</body>
</html>
```

Docker file

```
FROM httpd:latest
COPY index.html /usr/local/apache2/htdocs/
```

Step2] Building the custom image and pushing it to dockerhub repo

```
docker build -t techdecipher/custom-apache:v6 .

docker push techdecipher/custom-apache:v6
```

```
ubuntu@ip-172-31-15-226:~$ sudo docker build -t techdecipher/custom-apache:v6 .
[+] Building 0.3s (7/7) FINISHED
 => [internal] load build definition from dockerfile
 => => transferring dockerfile: 99B
 => [internal] load metadata for docker.io/library/httpd:latest
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load build context
```

Step3] The 2 files I put in the docker hub for the Argo CD to monitor repository and automatically deploy changes to Kubernetes cluster.

argocd / hello-world-deployment.yaml

techdecipher custom-docker-image hello-world-deployment.yaml 3eafe38 · 18 minutes ago History

Code Blame 19 lines (19 loc) · 326 Bytes Code 55% faster with GitHub Copilot Raw

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: apache
spec:
  replicas: 1
  selector:
    matchLabels:
      app: apache
  template:
    metadata:
      labels:
        app: apache
    spec:
      containers:
      - name: apache
        image: techdecipher/custom-apache:v6
        ports:
        - containerPort: 80
```

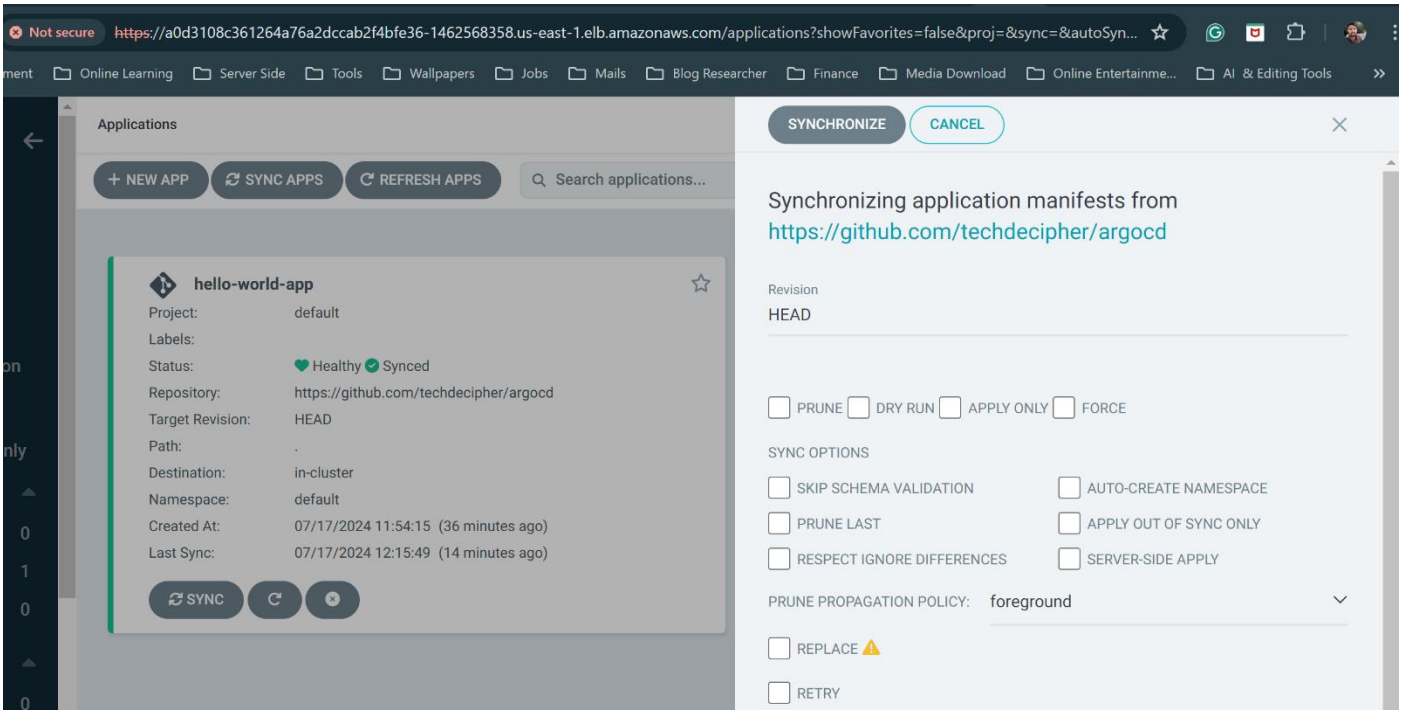
argocd / hello-world-service.yaml

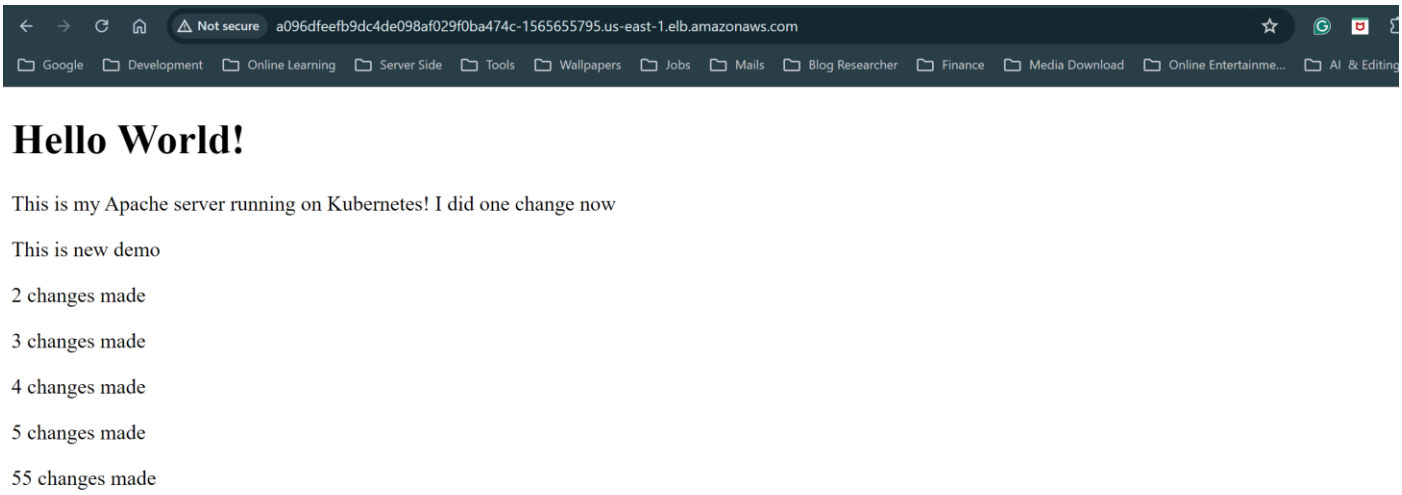
techdecipher Update hello-world-service.yaml - using apache

Code Blame 11 lines (11 loc) · 158 Bytes Code 55% faster with GitHub Copilot

```yaml
apiVersion: v1
kind: Service
metadata:
  name: apache-service
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 80
  selector:
    app: apache
```

Step4] Just committed the changes from Git repository as added new custom image which I built, and syncing option from ArgoCD shows the latest changes.





5)How the Argo CD deploys latest docker images into K8s:

So it is like when the new docker image is created, one can go around and push the changes to docker hub repo, post which the user can update the git file, with new image and commit the changes. Once that is done, just goto ArgoCD web UI and then hit sync changes button to see the changes, latest changes are available.