How to setup ArgoCD:

ArgoCD is the deployment tool for Kubernetes. We can store our configuration file in github and if we update any file. AgroCD will automatically deployed it.

Pre-requisite:

We need to have running EKS cluster. EC2 instance with kubectl, Terraform, AWS CLI installed.

Install AWS CLI and Configure

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

sudo apt install unzip

unzip awscliv2.zip

sudo ./aws/install

```
You can now run: /usr/local/bin/aws --version
[ec2-user@ip-172-31-9-225 ~]$ aws --version
aws-cli/2.15.7 Python/3.11.6 Linux/6.1.66-91.160.amzn2023.x86_64 exe/x86_64.amzn.2023 prompt/off
[ec2-user@ip-172-31-9-225 ~]$
```

Okay now after installing the AWS CLI, let's configure the ***AWS CLI*** so that it can authenticate and communicate with the AWS environment.

aws configure

```
[ec2-user@ip-172-31-9-225 ~]$ aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]: ap-south-1
Default output format [None]:
[ec2-user@ip-172-31-9-225 ~]$
```

Install and Setup Kubectl

Moving forward now we need to set up the **kubectl** also onto the EC2 instance.

curl -LO "https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl"

chmod +x ./kubectl

sudo mv ./kubectl /usr/local/bin

kubectl version

```
[ec2-user@ip-172-31-9-225 ~]$ kubectl version
Client Version: v1.29.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
The connection to the server localhost:8080 was refused - did you specify the right host or port?
[ec2-user@ip-172-31-9-225 ~]$
```

Follow below steps to install terraform on AmazonLinux.

```
sudo yum install -y yum-utils shadow-utils
sudo yum-config-manager --add-repo
https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
sudo yum -y install terraform
```

```
Complete!
[ec2-user@ip-172-31-9-225 eks-helm]$ terraform version
Terraform v1.6.6
on linux_amd64
[ec2-user@ip-172-31-9-225 eks-helm]$
```

Creating an Amazon EKS cluster using terraform

Code available in https://github.com/ksnithya/blue-green.git

git clone https://github.com/ksnithya/blue-green.git

cd blue-green

terraform init

terraform plan

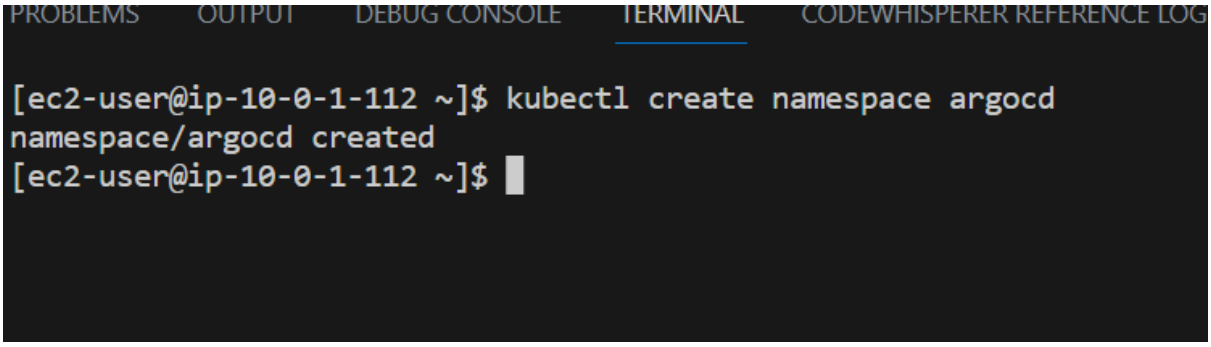
terraform apply

aws eks --region ap-south-1 update-kubeconfig --name eks_cluster_demo

Steps:

1. First we create namespace to install argocd.

   kubectl create namespace argocd



2. Let's run the following command to install agrocd.

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

```
service/argocd-metrics created
service/argocd-notifications-controller-metrics created
service/argocd-redis created
service/argocd-repo-server created
service/argocd-server created
service/argocd-server-metrics created
deployment.apps/argocd-applicationset-controller created
deployment.apps/argocd-dex-server created
deployment.apps/argocd-notifications-controller created
deployment.apps/argocd-redis created
deployment.apps/argocd-repo-server created
deployment.apps/argocd-server created
statefulset.apps/argocd-application-controller created
networkpolicy.networking.k8s.io/argocd-application-controller-network-pol
networkpolicy.networking.k8s.io/argocd-applicationset-controller-network-
networkpolicy.networking.k8s.io/argocd-dex-server-network-policy created
networkpolicy.networking.k8s.io/argocd-notifications-controller-network-p
networkpolicy.networking.k8s.io/argocd-redis-network-policy created
networkpolicy.networking.k8s.io/argocd-repo-server-network-policy created
```

3. Let see what are the resources created.

```
kubectl -n argocd get all
```

```
service/argocd-server-metrics                ClusterIP   172.20.123.129   <none>        8083/TCP
5m38s

NAME                                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/argocd-applicationset-controller    1/1     1            1           5m38s
deployment.apps/argocd-dex-server                   1/1     1            1           5m38s
deployment.apps/argocd-notifications-controller     1/1     1            1           5m38s
deployment.apps/argocd-redis                        1/1     1            1           5m38s
deployment.apps/argocd-repo-server                  1/1     1            1           5m38s
deployment.apps/argocd-server                       1/1     1            1           5m38s

NAME                                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/argocd-applicationset-controller-7786cb7547   1         1         1       5m38s
replicaset.apps/argocd-dex-server-58574dff5f                  1         1         1       5m38s
replicaset.apps/argocd-notifications-controller-7764bb774d    1         1         1       5m38s
replicaset.apps/argocd-redis-77bf5b886                        1         1         1       5m38s
replicaset.apps/argocd-repo-server-5b9977b575                 1         1         1       5m38s
replicaset.apps/argocd-server-6485ccb9c9                      1         1         1       5m38s

NAME                                                READY   AGE
```

4. We see the what service mapped to agrocd server.

```
kubectl get svc -n argocd
```

```
NAME                                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/argocd-applicationset-controller-7786cb7547   1         1         1       5m38s
replicaset.apps/argocd-dex-server-58574dff5f          1         1         1       5m38s
replicaset.apps/argocd-notifications-controller-7764bb774d    1         1         1       5m38s
replicaset.apps/argocd-redis-77bf5b886                1         1         1       5m38s
replicaset.apps/argocd-repo-server-5b9977b575         1         1         1       5m38s
replicaset.apps/argocd-server-6485ccb9c9              1         1         1       5m38s

NAME                                             READY   AGE
statefulset.apps/argocd-application-controller   1/1     5m38s
[ec2-user@ip-10-0-1-83 ~]$ kubectl get svc -n argocd
NAME                                      TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
argocd-applicationset-controller          ClusterIP   172.20.233.80    <none>        7000/TCP,8080/TCP
argocd-dex-server                         ClusterIP   172.20.20.250    <none>        5556/TCP,5557/TCP,5558/TCP
argocd-metrics                            ClusterIP   172.20.175.242   <none>        8082/TCP
argocd-notifications-controller-metrics   ClusterIP   172.20.97.31     <none>        9001/TCP
argocd-redis                              ClusterIP   172.20.226.48    <none>        6379/TCP
argocd-repo-server                        ClusterIP   172.20.138.232   <none>        8081/TCP,8084/TCP
argocd-server                             ClusterIP   172.20.203.87    <none>        80/TCP,443/TCP
argocd-server-metrics                     ClusterIP   172.20.123.129   <none>        8083/TCP
[ec2-user@ip-10-0-1-83 ~]$
```

5. Agrocd-server service is mapping to “ClusterIP”. We need to change it to “LoadBalancer” or “NodePort” to access the agrocd UI from outside world.

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

6. Now service will be changed to “LoadBalancer”

```
[ec2-user@ip-10-0-1-83 ~]$ kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'
service/argocd-server patched
[ec2-user@ip-10-0-1-83 ~]$ kubectl get svc -n argocd
NAME                                      TYPE           CLUSTER-IP       EXTERNAL-IP
                        PORT(S)                      AGE
argocd-applicationset-controller          ClusterIP      172.20.233.80    <none>
                        7000/TCP,8080/TCP            29m
argocd-dex-server                         ClusterIP      172.20.20.250    <none>
                        5556/TCP,5557/TCP,5558/TCP   29m
argocd-metrics                            ClusterIP      172.20.175.242   <none>
                        8082/TCP                     29m
argocd-notifications-controller-metrics   ClusterIP      172.20.97.31     <none>
                        9001/TCP                     29m
argocd-redis                              ClusterIP      172.20.226.48    <none>
                        6379/TCP                     29m
argocd-repo-server                        ClusterIP      172.20.138.232   <none>
                        8081/TCP,8084/TCP            29m
argocd-server                             LoadBalancer   172.20.203.87    aeb1d7b98f31441dca402f7f0e031256-388578
st-1.elb.amazonaws.com    80:31087/TCP,443:30947/TCP   29m
argocd-server-metrics                     ClusterIP      172.20.123.129   <none>
                        8083/TCP                     29m
[ec2-user@ip-10-0-1-83 ~]$
```

[ec2-user@ip-172-31-42-60 helm]$ kubectl get svc argocd-server -n argocd
NAME TYPE CLUSTER-IP EXTERNAL-IP
PORT(S) AGE
argocd-server LoadBalancer 172.20.26.38 **ac56ae5e4ee004f588e14968903ea1b2-1457899865.ap-south-1.elb.amazonaws.com** 80:31894/TCP,443:31830/TCP 9m11s

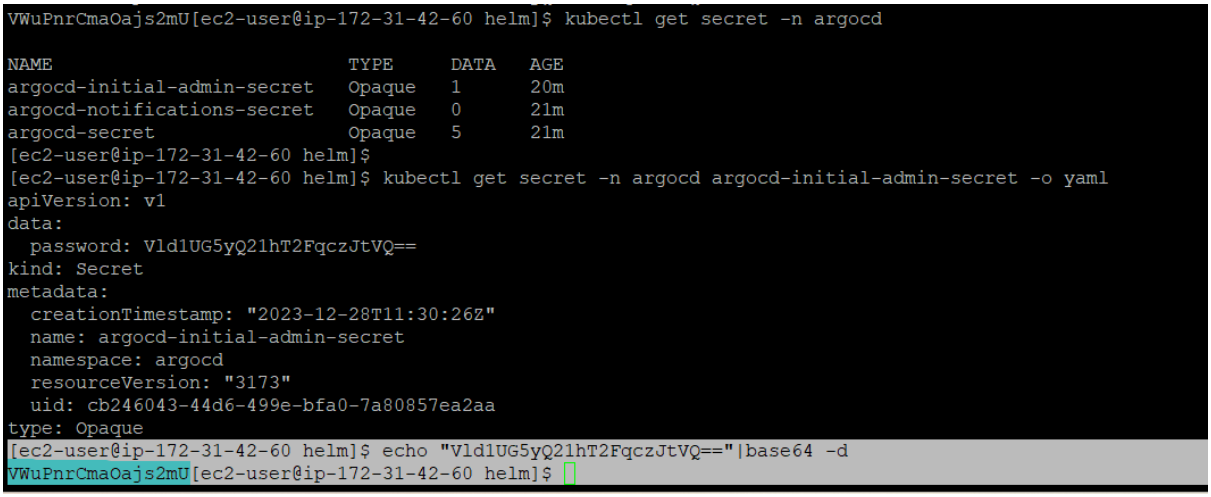
Now we can login to argocd using Loadbalancer.

http:// **ac56ae5e4ee004f588e14968903ea1b2-1457899865.ap-south-1.elb.amazonaws.com**

7. Default username is “admin” and password we can get it from secret.
[ec2-user@ip-172-31-42-60 helm]$ kubectl get secret -n argocd

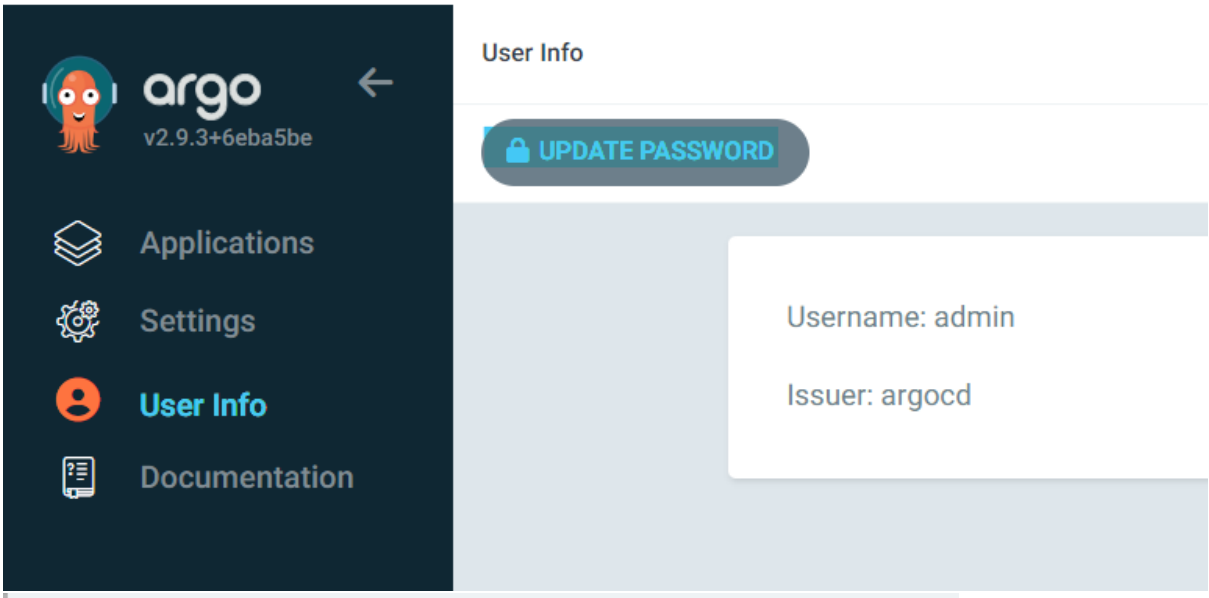
NAME TYPE DATA AGE

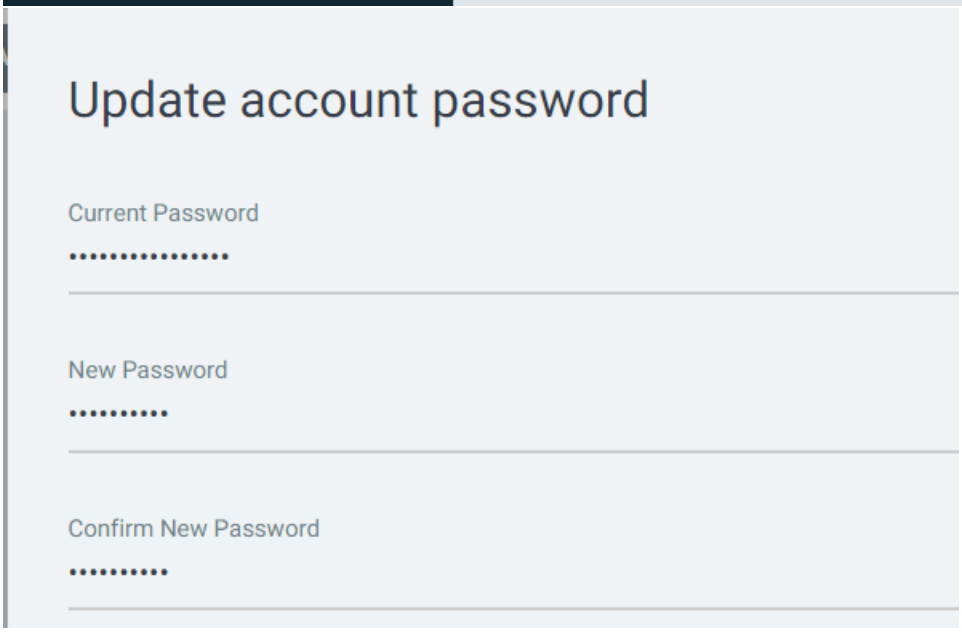
```
argocd-initial-admin-secret   Opaque   1     20m
argocd-notifications-secret   Opaque   0     21m
argocd-secret                 Opaque   5     21m
[ec2-user@ip-172-31-42-60 helm]$
[ec2-user@ip-172-31-42-60 helm]$ kubectl get secret -n argocd argocd-initial-admin-secret -
o yaml
apiVersion: v1
data:
  password: Vld1UG5yQ21hT2FqczJtVQ==
kind: Secret
metadata:
  creationTimestamp: "2023-12-28T11:30:26Z"
  name: argocd-initial-admin-secret
  namespace: argocd
  resourceVersion: "3173"
  uid: cb246043-44d6-499e-bfa0-7a80857ea2aa
type: Opaque
[ec2-user@ip-172-31-42-60 helm]$
[ec2-user@ip-172-31-42-60 helm]$ echo "Vld1UG5yQ21hT2FqczJtVQ==|base64 -d
VWuPnrCmaOajs2mU[ec2-user@ip-172-31-42-60 helm]$
```



We can use this password to login. After login it is recommended to change the password.

http:// **ac56ae5e4ee004f588e14968903ea1b2-1457899865.ap-south-1.elb.amazonaws.com**





Click on “Save password”.

8. Now we install argocd CLI.

```
sudo curl -sSL -o /usr/local/bin/argocd
https://github.com/argoproj/argo-cd/releases/latest/download/argocd-
linux-amd64
sudo chmod +x /usr/local/bin/argocd
```

```
[ec2-user@ip-172-31-42-60 helm]$ argocd version
argocd: v2.9.3+6eba5be
  BuildDate: 2023-12-01T23:24:09Z
  GitCommit: 6eba5be864b7e031871ed7698f5233336dfe75c7
  GitTreeState: clean
  GoVersion: go1.21.4
  Compiler: gc
  Platform: linux/amd64
FATA[0000] Failed to establish connection to ac56ae5e4ee004f588e14968903ea1b2-145789986
localhost, argocd-server, argocd-server.argocd, argocd-server.argocd.svc, argocd-server
m
[ec2-user@ip-172-31-42-60 helm]$
```

9. Now we login to argocd CLI.

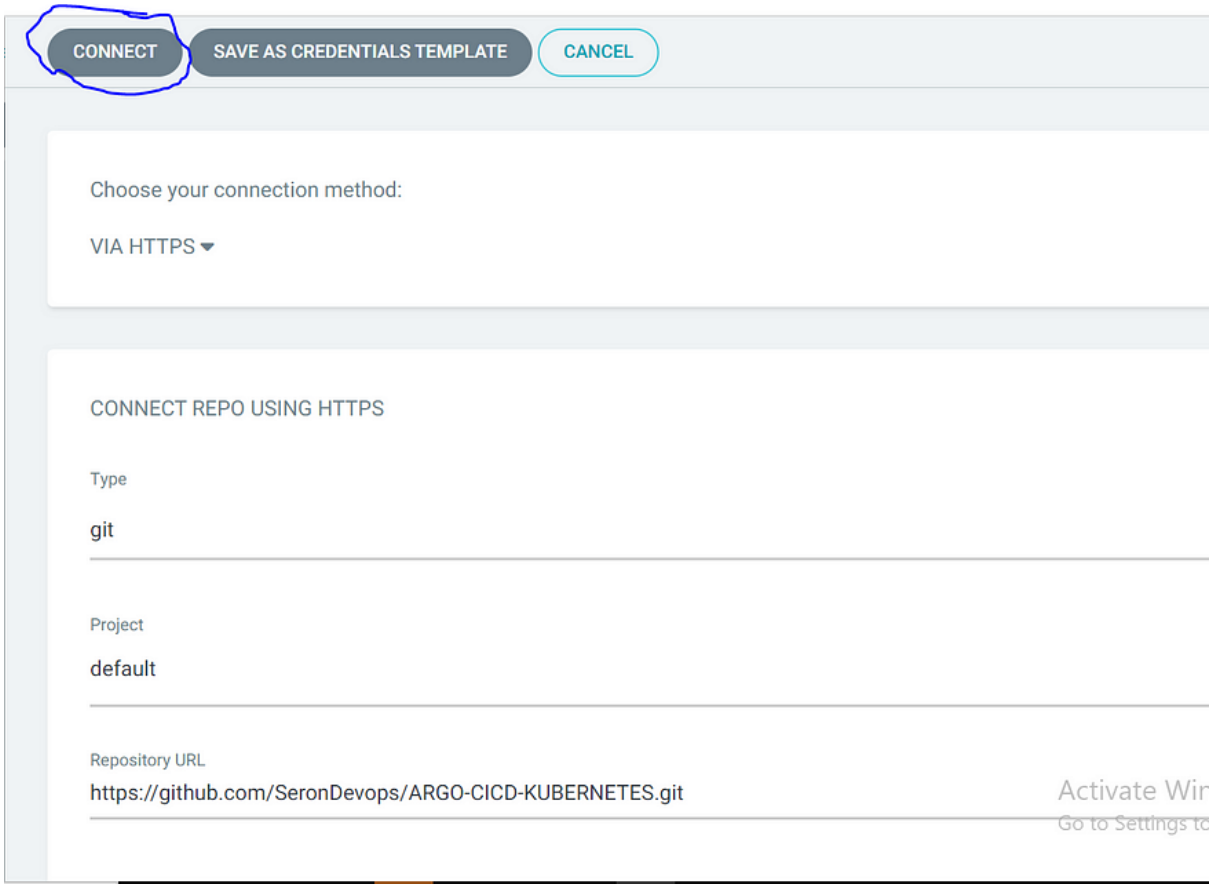
```
argocd login $ARGOCD_SERVER --username admin --password $ARGO_PWD --
insecure
```

```
sonarqube                                                                    latest    dee4f32d6f90   3 weeks ago    725MB
[ec2-user@ip-172-31-42-60 ~]$ sudo argocd login ac56ae5e4ee004f588e14968903ea1b2-1457899865.ap-south-1.elb.amazonaws.com --username admin --password Venkat@123 --insecure
'admin:login' logged in successfully
Context 'ac56ae5e4ee004f588e14968903ea1b2-1457899865.ap-south-1.elb.amazonaws.com' updated
[ec2-user@ip-172-31-42-60 ~]$
```

10. Now we create deployment app using argocd UI.

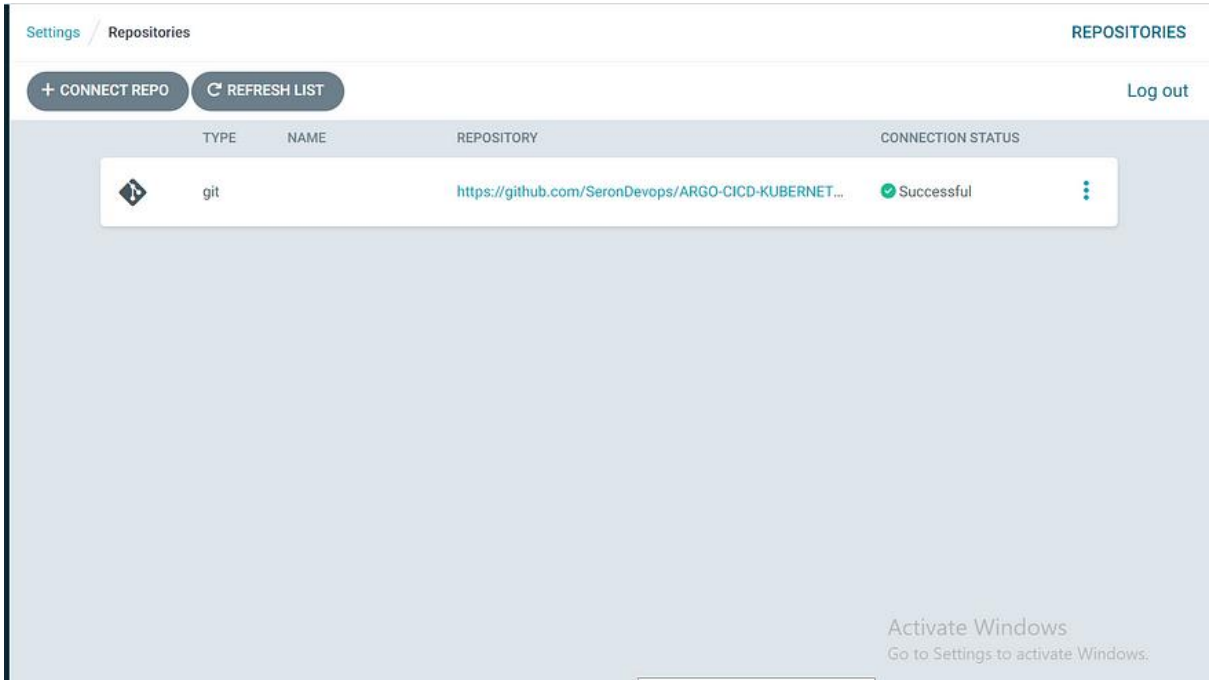
    In the ArgoCD web interface and click on Settings -> Repositories

On the next page click on Connect Repo and fill input the necessary details.



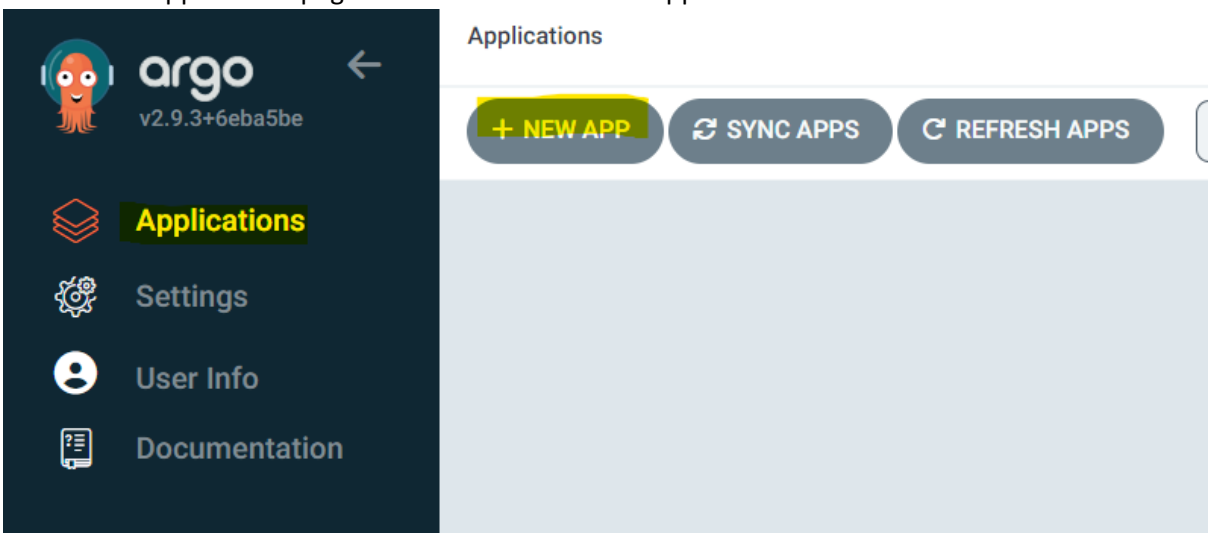
Under “Repository URL” we can give the github link where our code exist.
Repo: https://github.com/ksnithya/blue-green.git

Fill out just these three places and leave the rest options. Head up and click on Connect

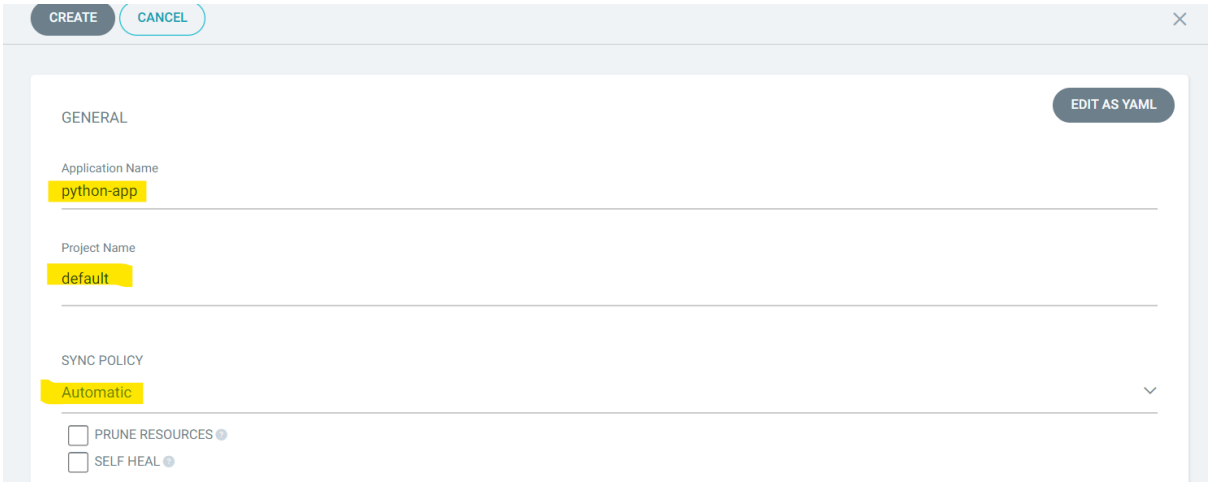


Now we have connected our GitHub repository with ArgoCD. Let's go build our application.

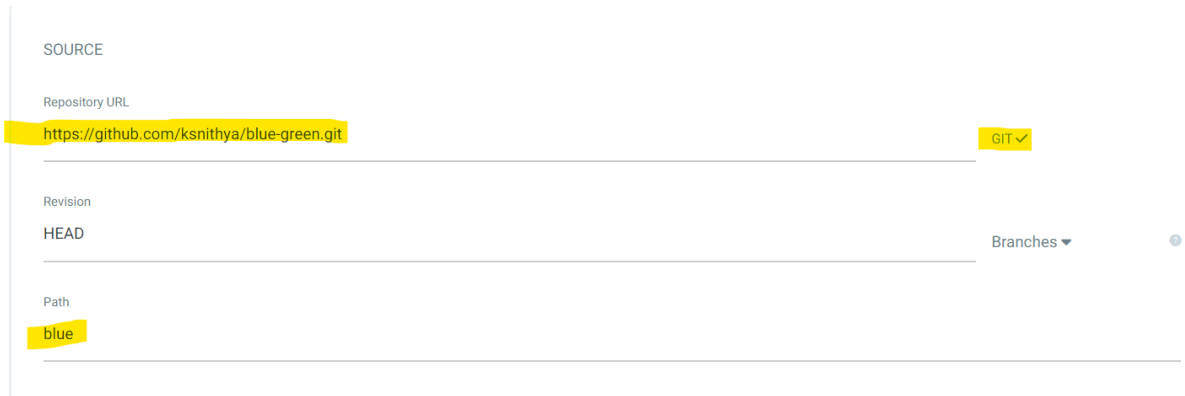
11. Click on the Applications page and click on create new app



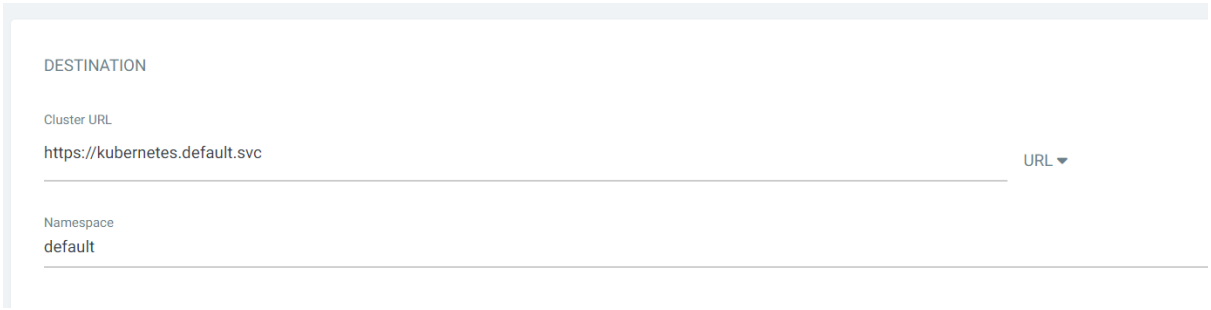
Fill the details as above.

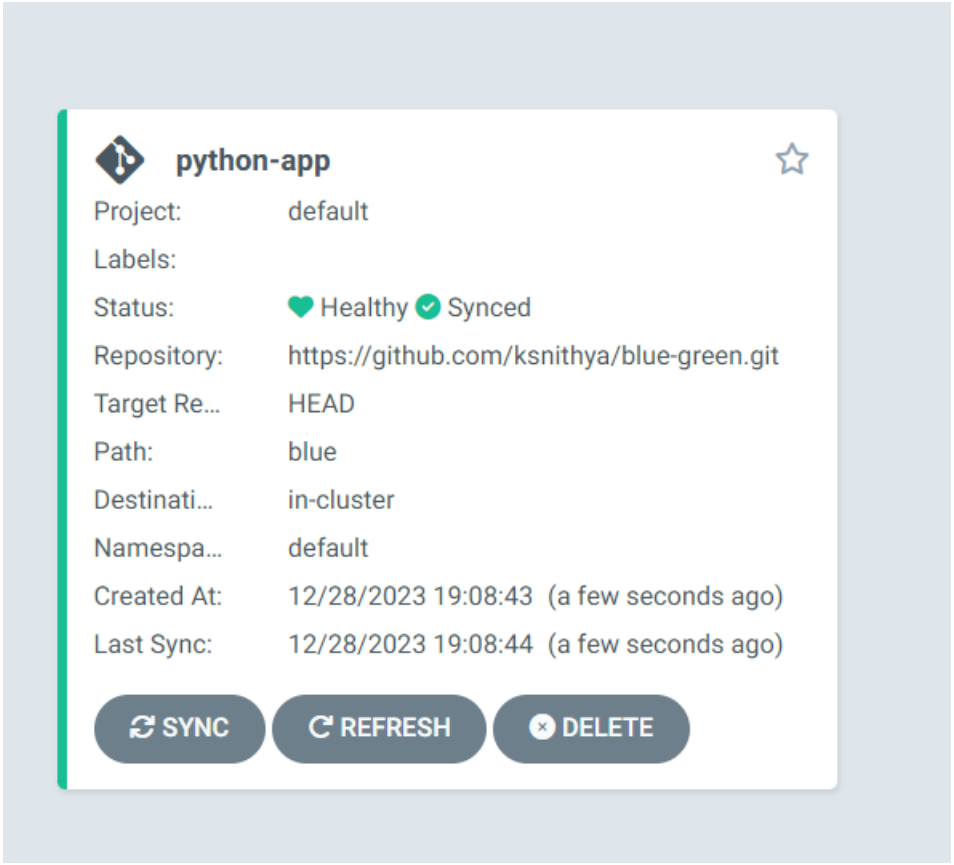


Fill the source details. Code is under blue directory so we need to give the directory path where our code exist.

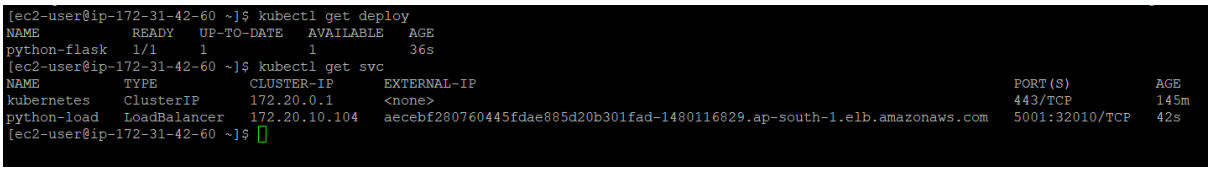


Select the cluster URL and Namespace.Now click on create. It will create the app.

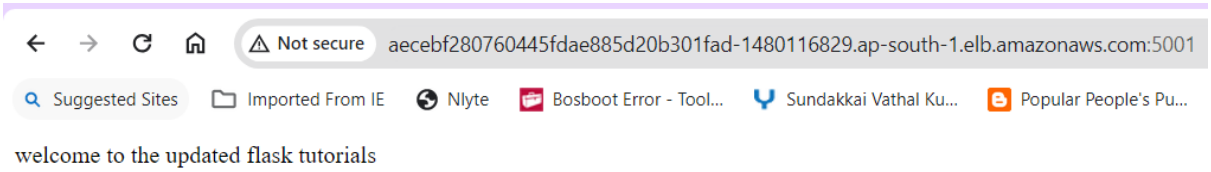




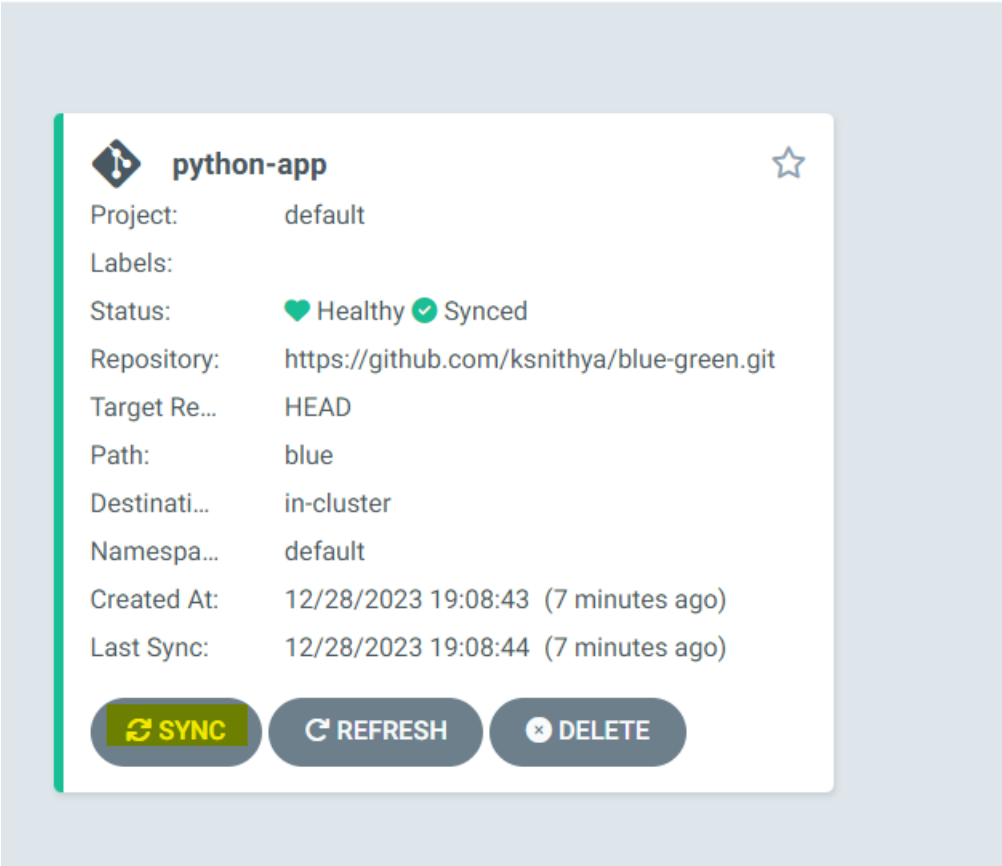
We can see our app deployed in our cluster server.



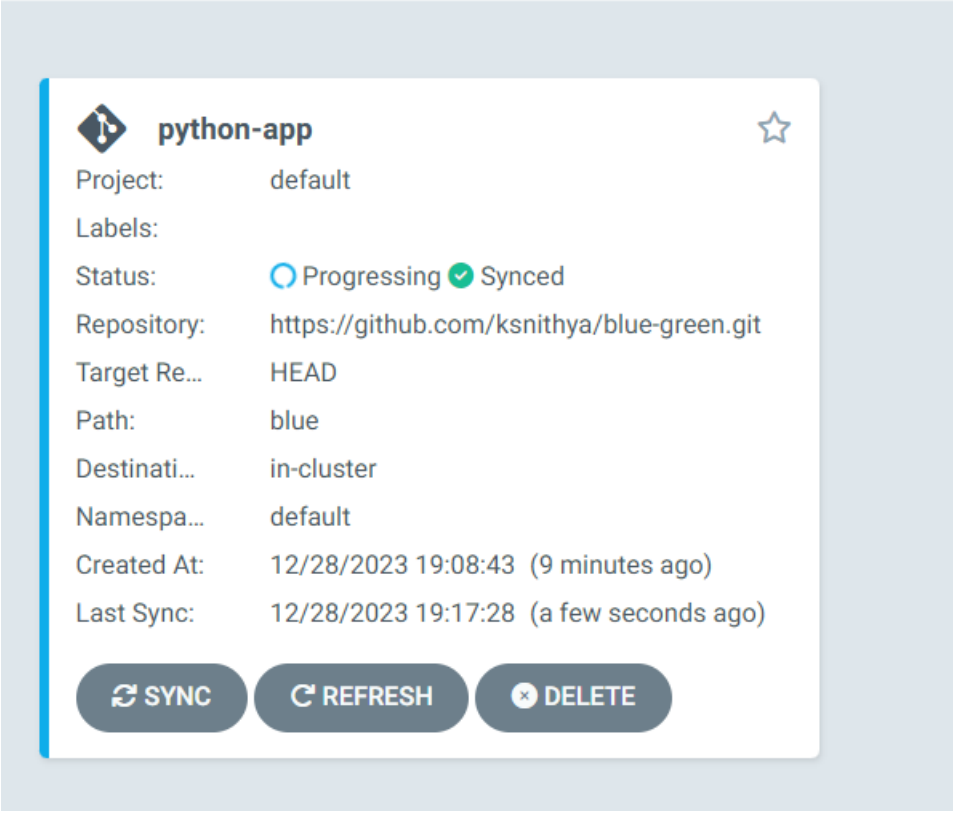
We can access the app using Loadbalancer URL.
http://aecebf280760445fdae885d20b301fad-1480116829.ap-south-1.elb.amazonaws.com:5001

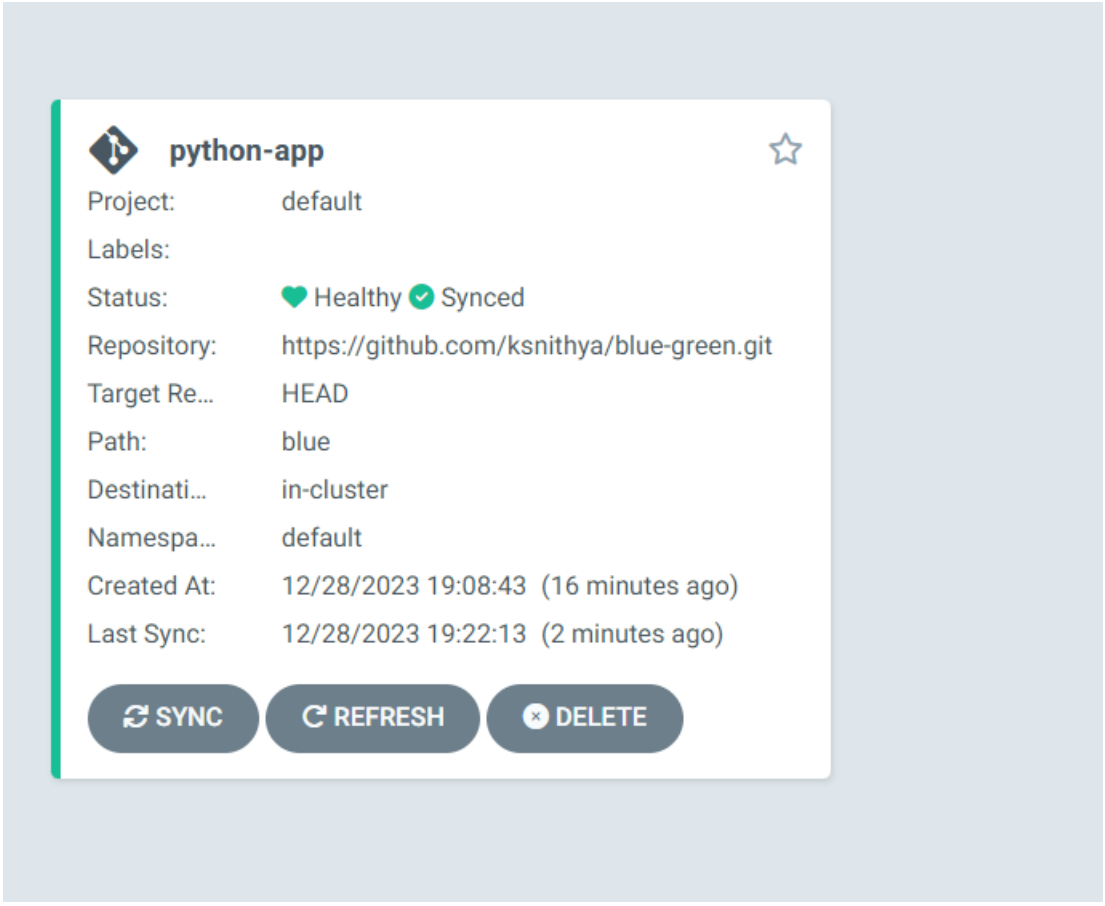


12. Now we can update our python app and create new image and push to dockerhub. Update that image details in our deploy.yml file in our repo and click on sync in app.



It will sync and redeploy it.





Now app is deployed with new image.