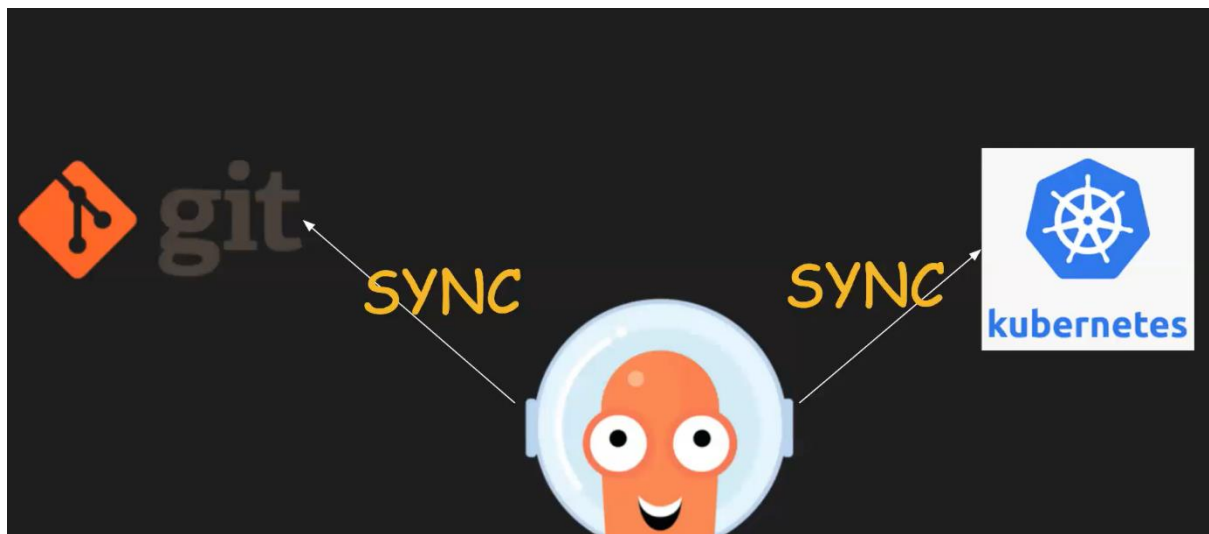
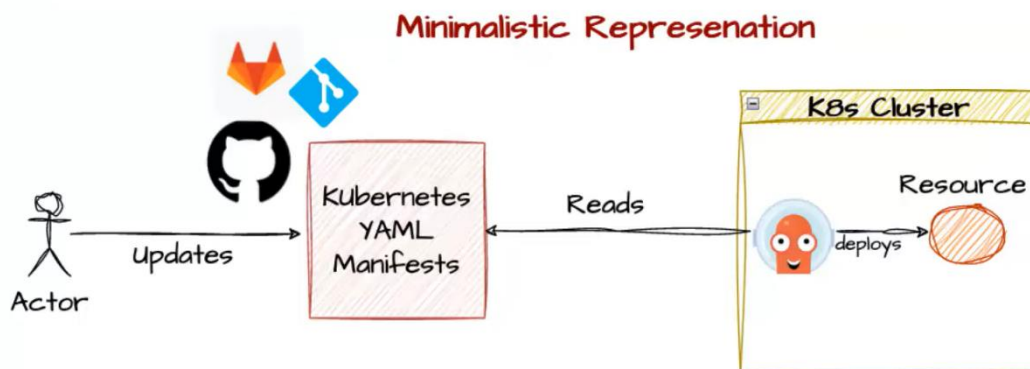


WHAT IS GITOPS ?

GITOPS uses GIT as a Single source of truth to deliver applications and infrastructure.



[Argo CD](#) is a Kubernetes-native continuous deployment (CD) tool. Unlike external CD tools that only enable push-based deployments, Argo CD can pull updated code from Git repositories and deploy it directly to Kubernetes resources. It enables developers to manage both infrastructure configuration and application updates in one system.

Argo CD offers the following key features and capabilities:

- Manual or automatic deployment of applications to a Kubernetes cluster.
- Automatic synchronization of application state to the current version of declarative configuration.
- Web user interface and command-line interface (CLI).

- Ability to visualize deployment issues, detect and remediate configuration drift.
- Role-based access control (RBAC) enabling multi-cluster management.
- Single sign-on (SSO) with providers such as GitLab, GitHub, Microsoft, OAuth2, OIDC, LinkedIn, LDAP, and SAML 2.0
- Support for webhooks triggering actions in GitLab, GitHub, and BitBucket.

GitOps with Argo CD

GitOps is a software engineering practice that uses a Git repository as its single source of truth. Teams commit declarative configurations into Git, and these configurations are used to create environments needed for the continuous delivery process. There is no manual setup of environments and no use of standalone scripts—everything is defined through the Git repository.

A basic part of the GitOps process is a pull request. New versions of a configuration are introduced via pull request, merged with the main branch in the Git repository, and then the new version is automatically deployed. The Git repository contains a full record of all changes, including all details of the environment at every stage of the process.

Argo CD handles the latter stages of the GitOps process, ensuring that new configurations are correctly deployed to a Kubernetes cluster.

At a high level, the Argo CD process works like this:

1. A developer makes changes to an application, pushing a new version of Kubernetes resource definitions to a Git repo.
2. Continuous integration is triggered, resulting in a new container image saved to a registry.
3. A developer issues a pull request, changing Kubernetes manifests, which are created either manually or automatically.
4. The pull request is reviewed and changes are merged to the main branch. This triggers a webhook which tells Argo CD a change was made.
5. Argo CD clones the repo and compares the application state with the current state of the Kubernetes cluster. It applies the required changes to cluster configuration.
6. Kubernetes uses its controllers to reconcile the changes required to cluster resources, until it achieves the desired configuration.
7. Argo CD monitors progress and when the Kubernetes cluster is ready, reports that the application is in sync.
8. ArgoCD also works in the other direction, monitoring changes in the Kubernetes cluster and discarding them if they don't match the current configuration in Git.

How does Argo CD make it happen?

- **GitOps agent**—Argo CD is responsible for pulling updated code from Git repositories and deploying it directly to Kubernetes resources. It manages both infrastructure configuration and application updates in one system.
- **Custom Resource Definitions (CRD)**—Argo CD operates in its own namespace within a Kubernetes cluster. It provides its own CRDs that extend the Kubernetes API and make it possible to define the desired application state in a declarative way. Based on the instructions in a Git repo or a Helm repo, Argo CD uses its CRDs to implement the changes within its dedicated namespace.

- CLI—Argo CD offers a powerful CLI that lets you create YAML resource definitions with a few simple commands. For example, the

Argo CD app create

command lets you specify a few flags and create a valid

Application

object that describes your application, with no need to write YAML by hand.

STEP :1

Installation of ArgoCD:

Go to Official Documentation of ArgoCD:

https://argo-cd.readthedocs.io/en/stable/getting_started/

kubectl create namespace argocd

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

1. Install Argo CD

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

This will create a new namespace, `argocd`, where Argo CD services and application resources will live.

Warning

The installation manifests include `ClusterRoleBinding` resources that reference `argocd` namespace. If you are installing Argo CD into a different namespace then make sure to update the namespace reference.

getting_started/#requirements

STEP 2:

After installing Argocd . We get below pods

```
[root@ip-172-31-9-228 ~]# kubectl get pods -n argocd
NAME                                READY   STATUS    RESTARTS   AGE
argocd-application-controller-0     1/1     Running   0           92m
argocd-applicationset-controller-744b76d7fd-4kthf  1/1     Running   0           92m
argocd-dex-server-5bf5dbc64d-qqrjd  1/1     Running   0           92m
argocd-notifications-controller-84f5bf6896-kg8zp  1/1     Running   0           92m
argocd-redis-74b8999f94-rg8r7       1/1     Running   0           92m
argocd-repo-server-57f4899557-z7lfc  1/1     Running   0           92m
argocd-server-7bc7b97977-nzjf8      1/1     Running   0           92m
[root@ip-172-31-9-228 ~]#
```

EXPOSE ARGOCD-SERVER SERVICE TO NODE PORT

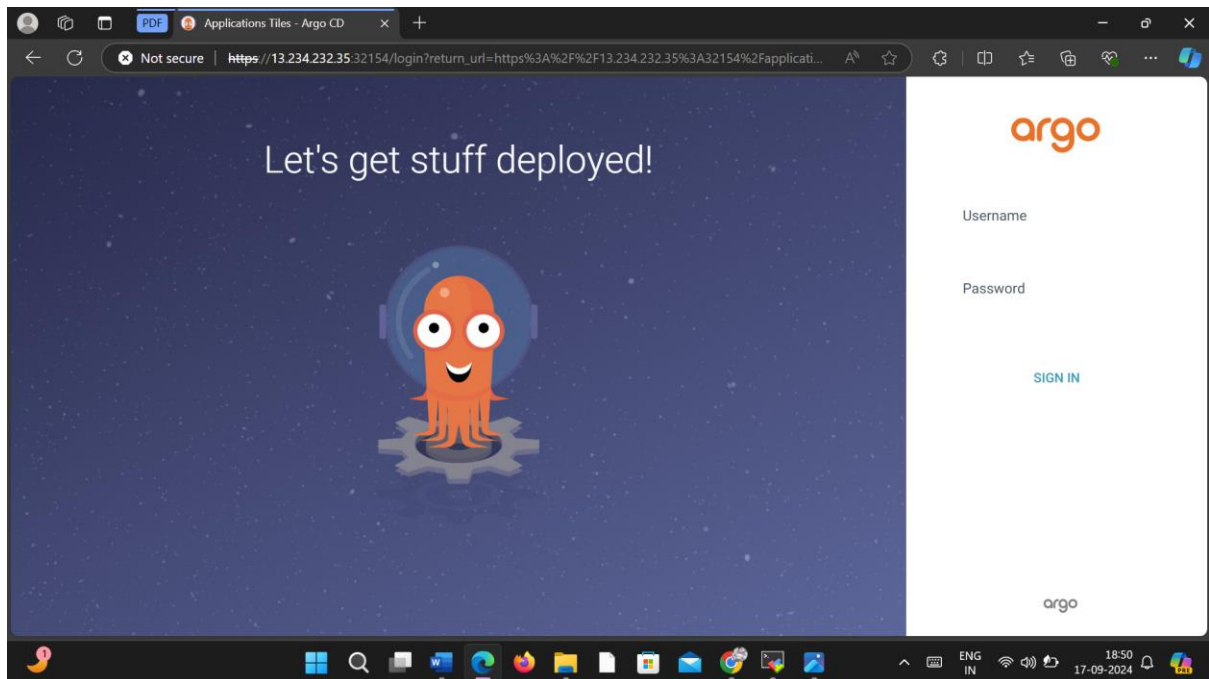
```
[root@ip-172-31-9-228 ~]# kubectl get pods -n argocd
NAME                                READY   STATUS    RESTARTS   AGE
argocd-application-controller-0     1/1     Running   0           92m
argocd-applicationset-controller-744b76d7fd-4kthf  1/1     Running   0           92m
argocd-dex-server-5bf5dbc64d-qqrjd  1/1     Running   0           92m
argocd-notifications-controller-84f5bf6896-kg8zp  1/1     Running   0           92m
argocd-redis-74b8999f94-rg8r7       1/1     Running   0           92m
argocd-repo-server-57f4899557-z7lfc  1/1     Running   0           92m
argocd-server-7bc7b97977-nzjf8      1/1     Running   0           92m
[root@ip-172-31-9-228 ~]# kubectl get svc -n argocd
NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S)                                AGE
argocd-applicationset-controller    ClusterIP           100.66.147.127  <none>           7000/TCP,8080/TCP                     93m
argocd-dex-server                  ClusterIP           100.66.144.33   <none>           5556/TCP,5557/TCP,5558/TCP            93m
argocd-metrics                     ClusterIP           100.70.157.227  <none>           8082/TCP                              93m
argocd-notifications-controller-metrics  ClusterIP           100.66.211.27   <none>           9001/TCP                              93m
argocd-redis                       ClusterIP           100.71.229.234  <none>           6379/TCP                              93m
argocd-repo-server                 ClusterIP           100.66.194.179  <none>           8081/TCP,8084/TCP                     93m
argocd-server                      NodePort            100.65.43.13    <none>           80:30716/TCP,443:32154/TCP            93m
argocd-server-metrics              ClusterIP           100.66.232.166  <none>           8083/TCP                              93m
[root@ip-172-31-9-228 ~]# kubectl edit svc argocd-server -n argocd
```

```
app.kubernetes.io/component: server
app.kubernetes.io/name: argocd-server
app.kubernetes.io/part-of: argocd
name: argocd-server
namespace: argocd
resourceVersion: "7680"
uid: bb718d00-8ce3-4cb2-b49f-3e1210f7b8be
spec:
  clusterIP: 100.65.43.13
  clusterIPs:
  - 100.65.43.13
  externalTrafficPolicy: Cluster
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - name: http
    nodePort: 30716
    port: 80
    protocol: TCP
    targetPort: 8080
  - name: https
    nodePort: 32154
    port: 443
    protocol: TCP
    targetPort: 8080
  selector:
    app.kubernetes.io/name: argocd-server
  sessionAffinity: None
  type: NodePort
status: {}
loadBalancer: {}
```

STEP 3:

To Login into ArgoCD UI

Copy PublicIP of EC2 instance along with NodePort and Paste in browser.

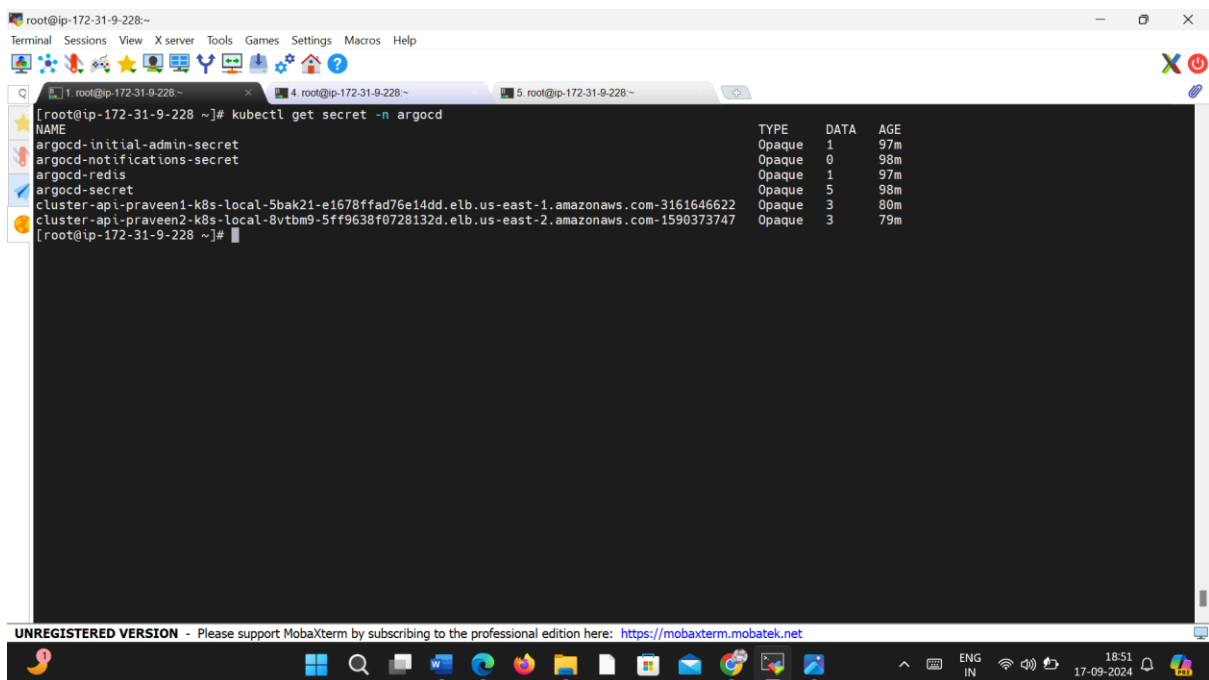


This is argocd DashBoard

UserName: admin

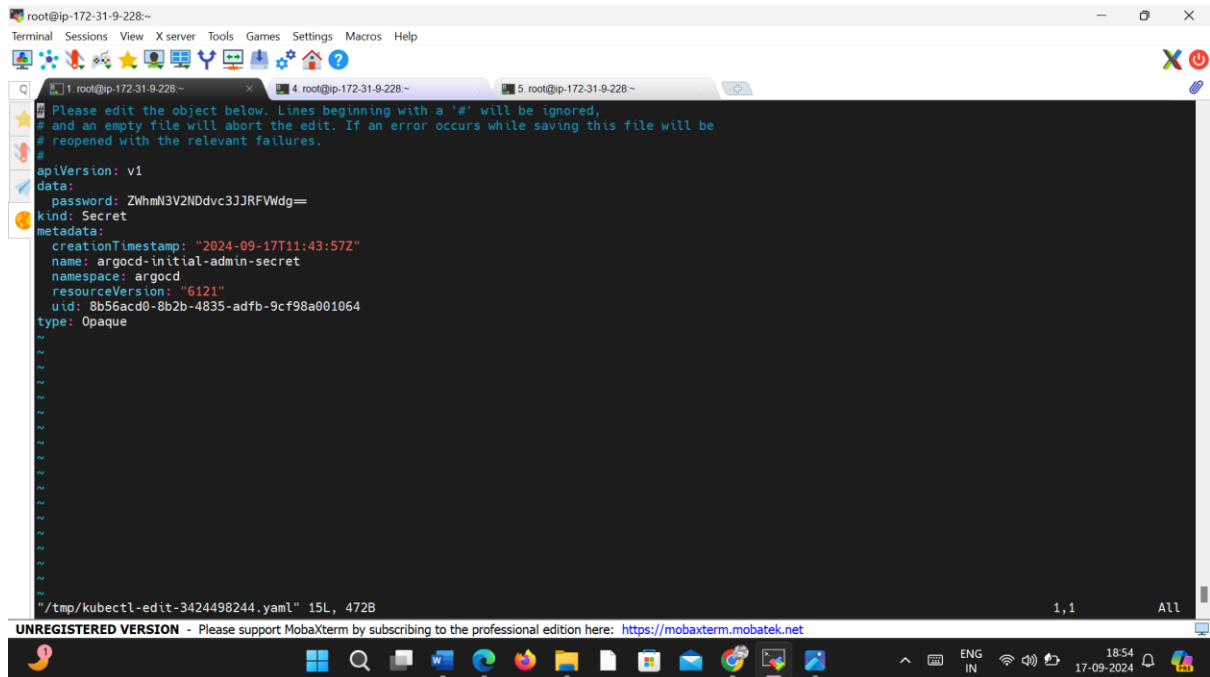
To get Password:

Go to argocd secret



Edit **argocd-initial-admin-secret**

Kubectl edit secret **argocd-initial-admin-secret** -n argocd

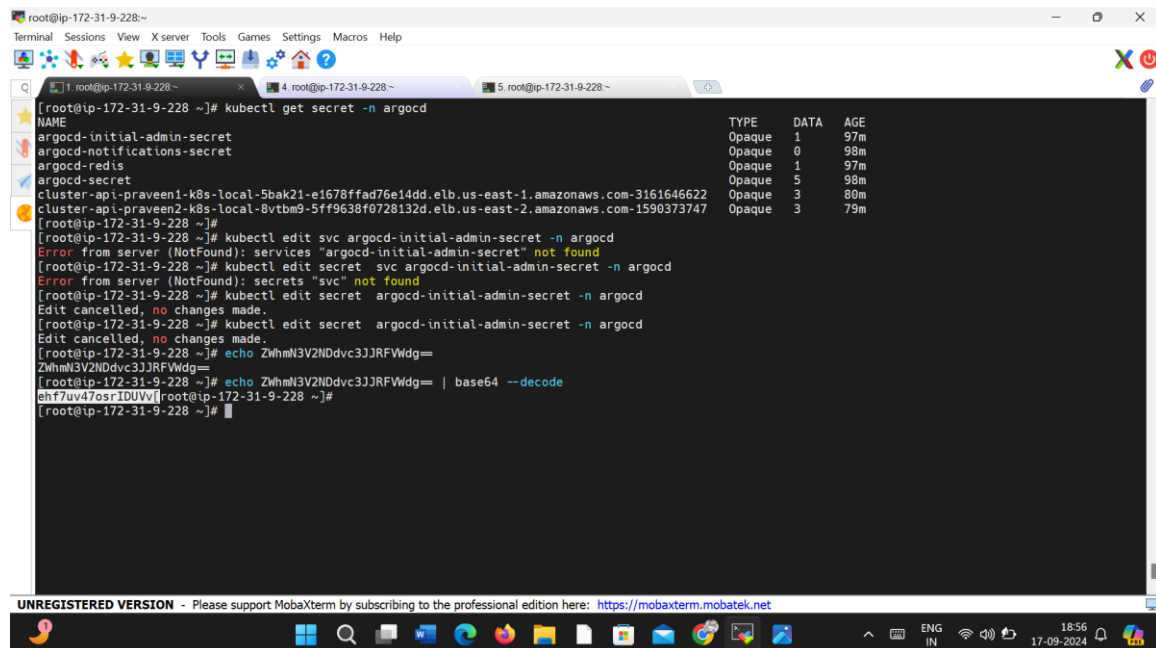


```
root@ip-172-31-9-228:~
Terminal Sessions View X server Tools Games Settings Macros Help

Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  password: ZWhmN3V2NDdvc3JJRFVWdg==
kind: Secret
metadata:
  creationTimestamp: "2024-09-17T11:43:57Z"
  name: argocd-initial-admin-secret
  namespace: argocd
  resourceVersion: "6121"
  uid: 8b56acd0-8b2b-4835-adfb-9cf98a001064
type: Opaque

~/tmp/kubectrl-edit-3424498244.yaml 15L, 472B
1,1 All
```

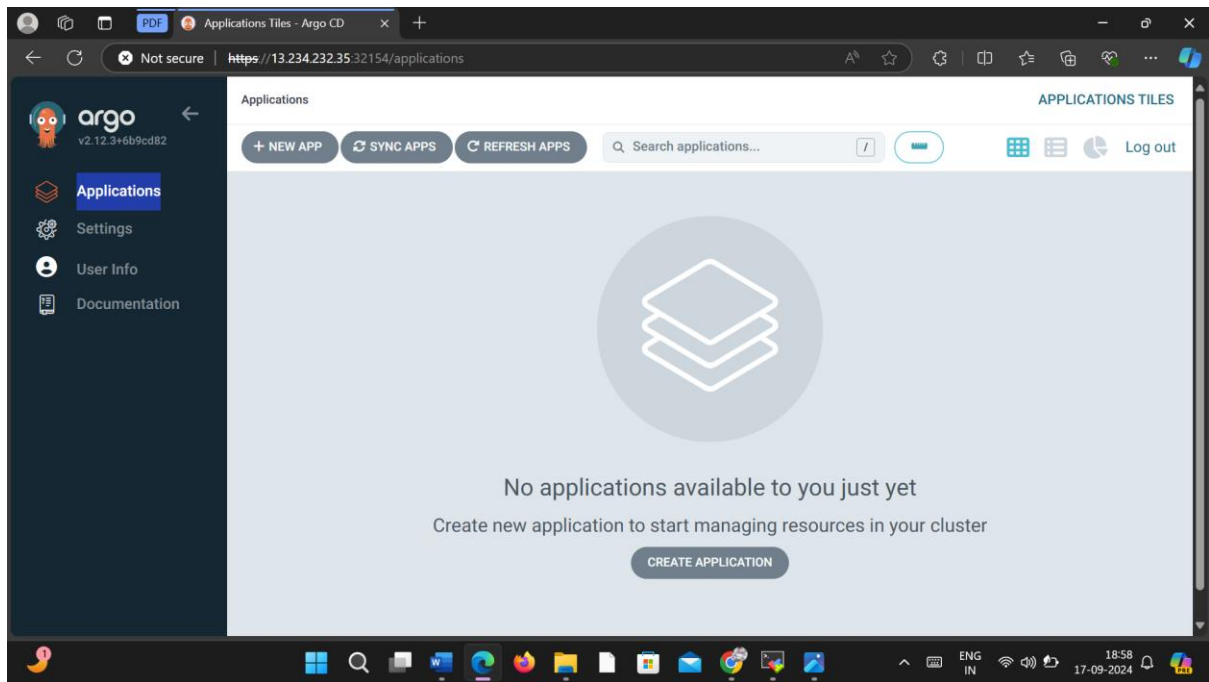
Copy the password and decode using Base64



```
root@ip-172-31-9-228:~
Terminal Sessions View X server Tools Games Settings Macros Help

[root@ip-172-31-9-228 ~]# kubectl get secret -n argocd
NAME                                     TYPE      DATA      AGE
argocd-initial-admin-secret             Opaque    1           97m
argocd-notifications-secret             Opaque    0           98m
argocd-redis                            Opaque    1           97m
argocd-secret                           Opaque    5           98m
cluster-api-praveen1-k8s-local-5bak21-e1678ffad76e1dd.elb.us-east-1.amazonaws.com-3161646622 Opaque    3           88m
cluster-api-praveen2-k8s-local-8vtbm9-5ff9638f0728132d.elb.us-east-2.amazonaws.com-1590373747 Opaque    3           79m

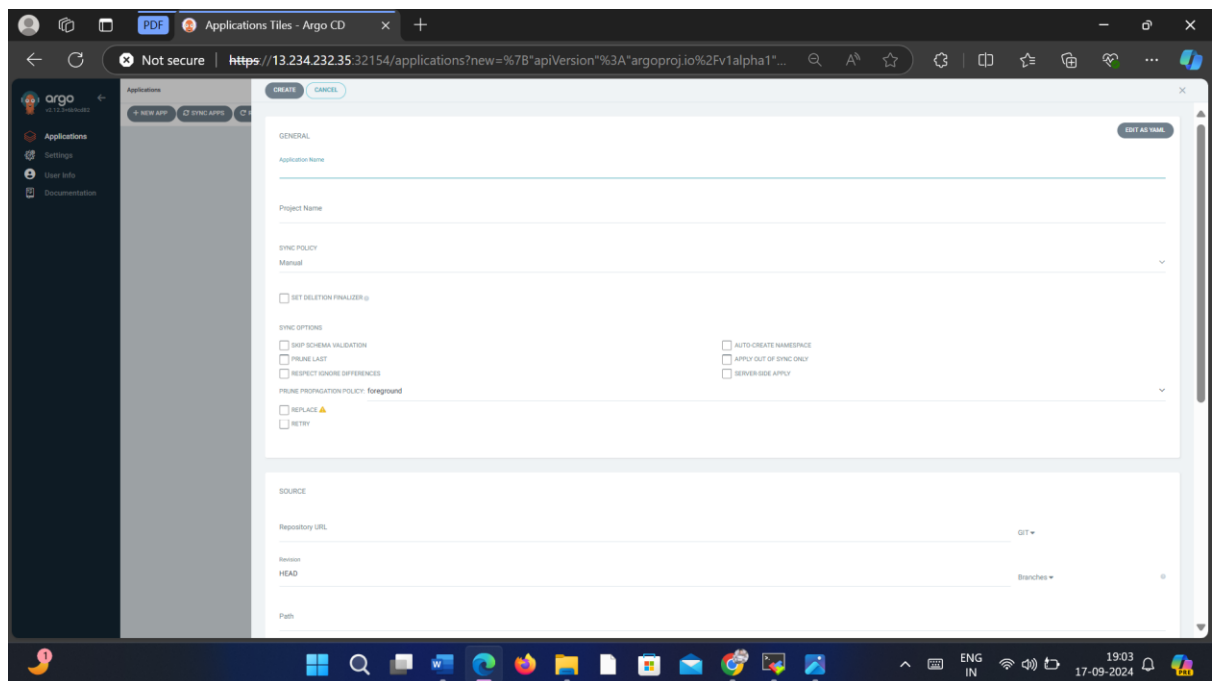
[root@ip-172-31-9-228 ~]# kubectl edit svc argocd-initial-admin-secret -n argocd
Error from server (NotFound): services "argocd-initial-admin-secret" not found
[root@ip-172-31-9-228 ~]# kubectl edit secret svc argocd-initial-admin-secret -n argocd
Error from server (NotFound): secrets "svc" not found
[root@ip-172-31-9-228 ~]# kubectl edit secret argocd-initial-admin-secret -n argocd
Edit cancelled, no changes made.
[root@ip-172-31-9-228 ~]# kubectl edit secret argocd-initial-admin-secret -n argocd
Edit cancelled, no changes made.
[root@ip-172-31-9-228 ~]# echo ZWhmN3V2NDdvc3JJRFVWdg==
ZWhmN3V2NDdvc3JJRFVWdg==
[root@ip-172-31-9-228 ~]# echo ZWhmN3V2NDdvc3JJRFVWdg== | base64 --decode
ehf7uv47osrIDUVv[
root@ip-172-31-9-228 ~]#
```



STEP 4:

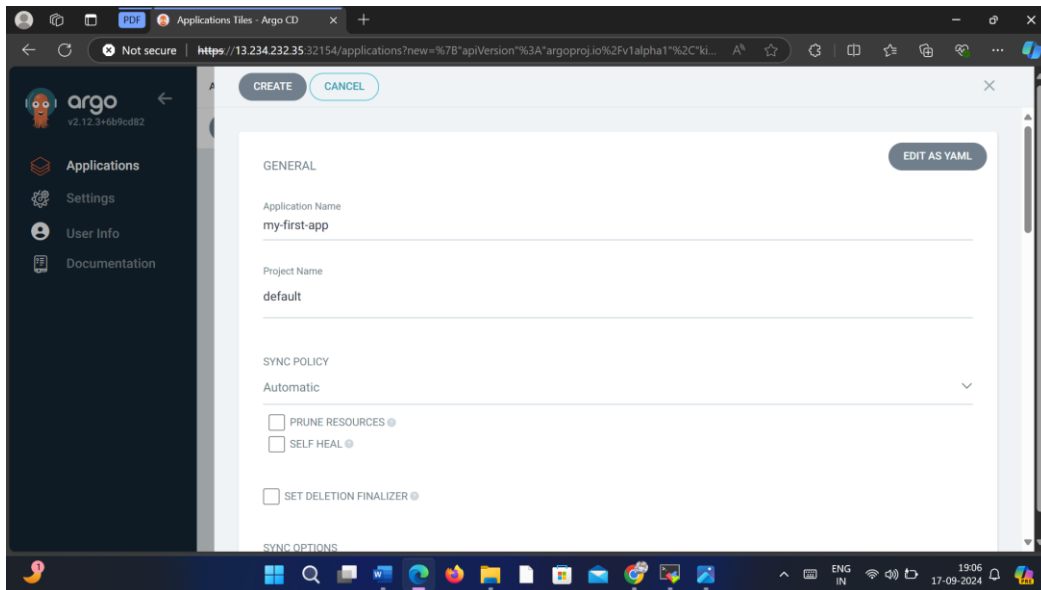
Create a Application:

- Click On New app



- After Click on application . We get above UI.
- Configure the details.

1. **Application Name: firstapp**
2. **Project Name: Default (In my case)**
3. **SYNC POLICY : Automatic**

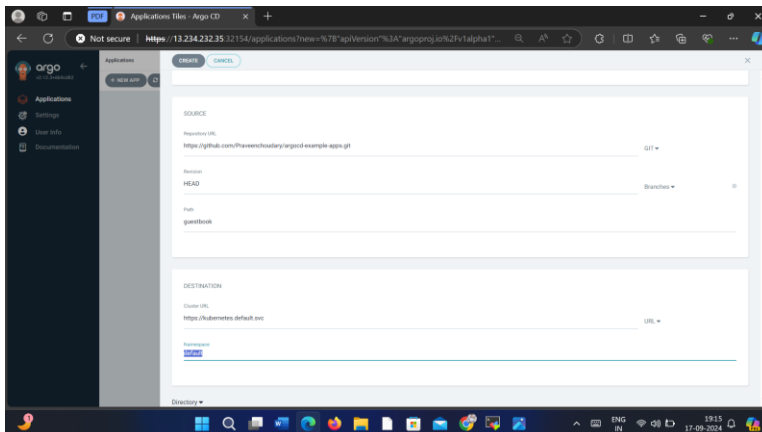


The screenshot shows the Argo CD web interface in a browser. The 'CREATE' modal is open, displaying the 'GENERAL' tab. The 'Application Name' field is filled with 'my-first-app'. The 'Project Name' field is filled with 'default'. The 'SYNC POLICY' dropdown is set to 'Automatic'. Below this, there are three unchecked checkboxes: 'PRUNE RESOURCES', 'SELF HEAL', and 'SET DELETION FINALIZER'. An 'EDIT AS YAML' button is visible in the top right corner of the form. The left sidebar shows the 'Applications' menu item selected.

4. **Source: Paste Github repo url**

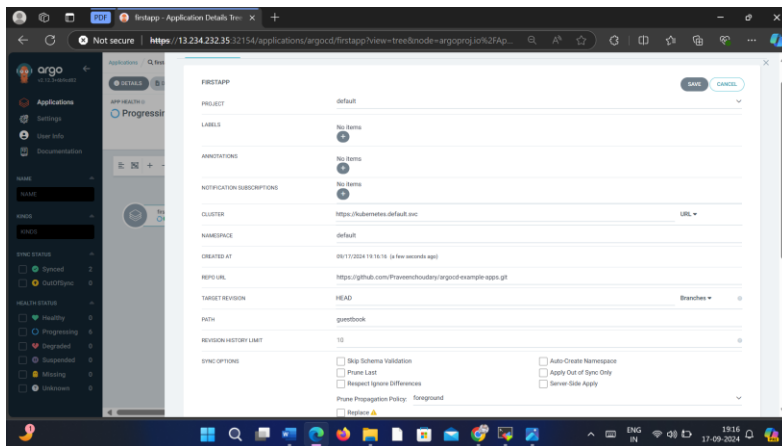
<https://github.com/Praveenchoudary/argocd-example-apps.git>

5. **Path: guestbook**
6. **Cluster url: <https://kubernetes.default.svc> (default url)**
7. **Name Space: default**

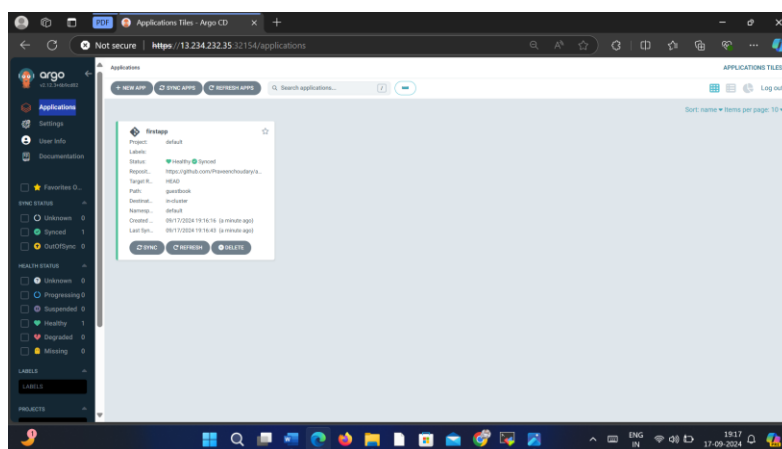


The screenshot shows the Argo CD web interface with the 'CREATE' modal open, displaying the 'SOURCE' and 'DESTINATION' tabs. In the 'SOURCE' section, the 'Repository URL' field is filled with 'https://github.com/Praveenchoudary/argocd-example-apps.git', the 'Revision' is set to 'HEAD', and the 'Path' is 'guestbook'. In the 'DESTINATION' section, the 'Cluster URL' is 'https://kubernetes.default.svc' and the 'Namespace' is 'default'. The left sidebar shows the 'Applications' menu item selected.

8. **Click on create**



Application is cretaed Successfully



deployment.yml

apiVersion: apps/v1

kind: Deployment

metadata:

name: guestbook-ui

spec:

replicas: 3

revisionHistoryLimit: 3

selector:

matchLabels:

app: guestbook-ui

template:

metadata:

labels:

app: guestbook-ui

spec:

containers:

- image: praveen22233/netflix

name: guestbook-ui

ports:

- containerPort: 80

Three Pods are created as I mentioned replica count=3 in deployment.yml file and service LoadBalancer.

Service.yml

apiVersion: v1

kind: Service

metadata:

name: guestbook-ui

spec:

type: LoadBalancer

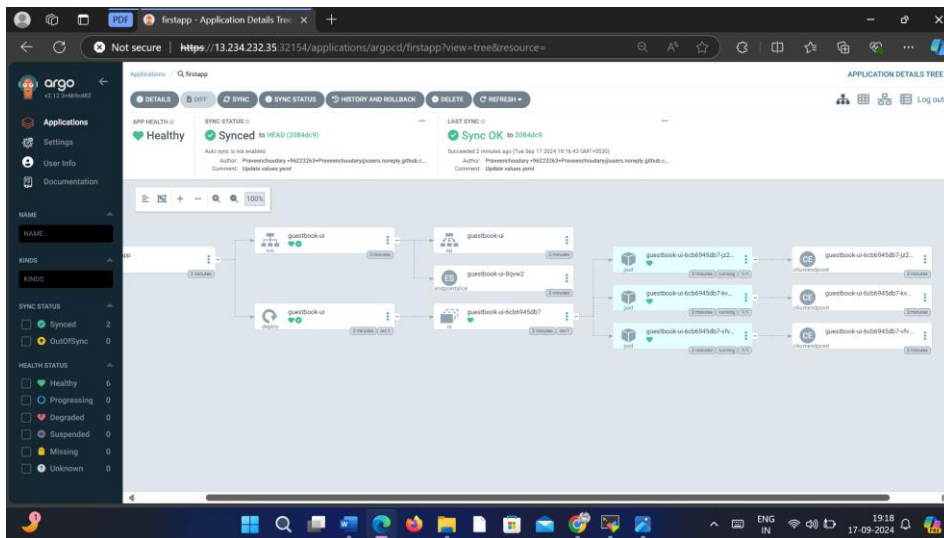
selector:

app: guestbook-ui

ports:

- port: 80

targetPort: 80



3 pods and service is created in k8s cluster. This is done by using argocd

```

root@ip-172-31-9-228 ~
[root@ip-172-31-9-228 ~]# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
guestbook-ui-6cb6945db7-j2q9        1/1     Running   0           8m31s
guestbook-ui-6cb6945db7-kvxxm       1/1     Running   0           8m31s
guestbook-ui-6cb6945db7-vfvqg       1/1     Running   0           8m31s
[root@ip-172-31-9-228 ~]# kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/guestbook-ui-6cb6945db7-j2q9    1/1     Running   0           8m36s
pod/guestbook-ui-6cb6945db7-kvxxm    1/1     Running   0           8m36s
pod/guestbook-ui-6cb6945db7-vfvqg    1/1     Running   0           8m36s

NAME                                TYPE          CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
service/guestbook-ui                LoadBalancer  100.68.176.136  a76ecc72d42ba4f65a55c43b1763a3fd-1858565982.ap-south-1.elb.amazonaws.com  80:30958/TCP    8m36s
service/kubernetes                   ClusterIP      100.64.0.1      <none>            443/TCP          156m

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/guestbook-ui        3/3     3             3           8m36s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/guestbook-ui-6cb6945db7  3         3         3       8m36s
[root@ip-172-31-9-228 ~]#

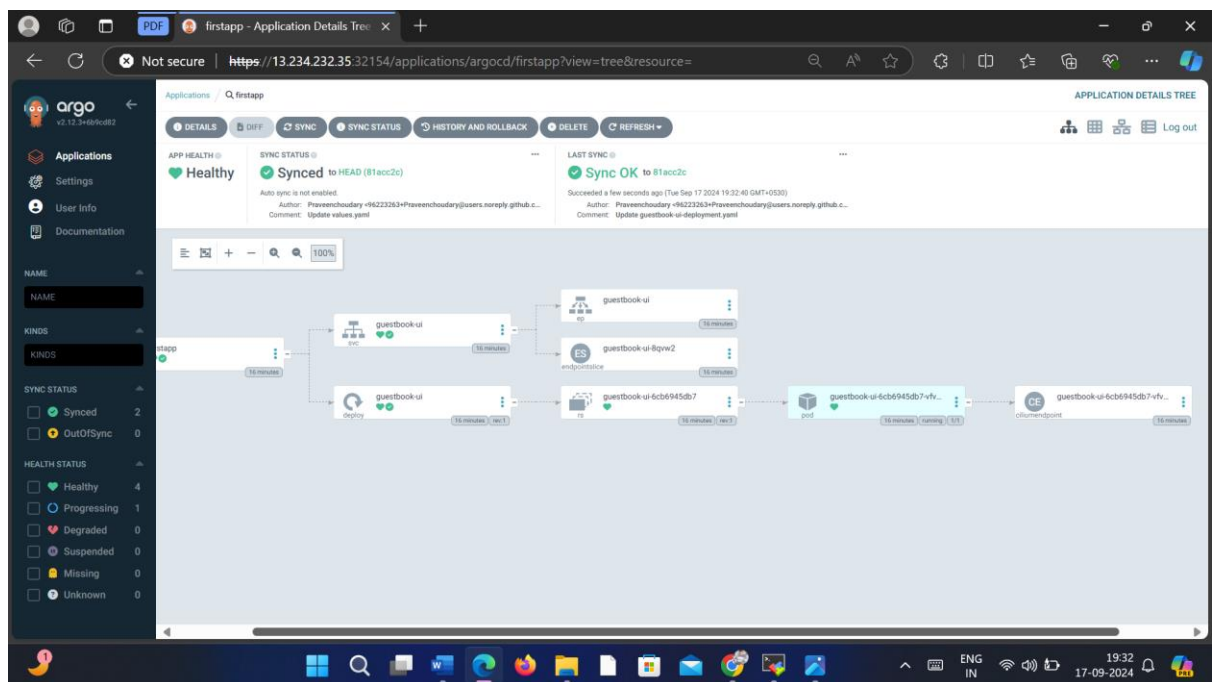
```

If we make any changes in yml files which are in github. Automatically the changes will reflected in k8s cluster.

```
root@ip-172-31-9-228:~  
Terminal Sessions View X server Tools Games Settings Macros Help  
[root@ip-172-31-9-228 ~]# kubectl get pods  
NAME READY STATUS RESTARTS AGE  
guestbook-ui-6cb6945db7-jz2q9 1/1 Running 0 12m  
guestbook-ui-6cb6945db7-kvxxm 1/1 Running 0 12m  
guestbook-ui-6cb6945db7-vfvqg 1/1 Running 0 12m  
[root@ip-172-31-9-228 ~]#
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

- In K8s cluster there is 3 pods are running.
- Now I make changes in replicas count to 1 in deployment.yml file in github and commit the changes .
- Automatically The pod size will be decreased to 1.



The replica count is decreased to one.

```
root@ip-172-31-9-228:~  
Terminal Sessions View X server Tools Games Settings Macros Help  
[root@ip-172-31-9-228 ~]# kubectl get pods  
NAME READY STATUS RESTARTS AGE  
guestbook-ui-6cb6945db7-vfvqg 1/1 Running 0 17m  
[root@ip-172-31-9-228 ~]#
```

Note:

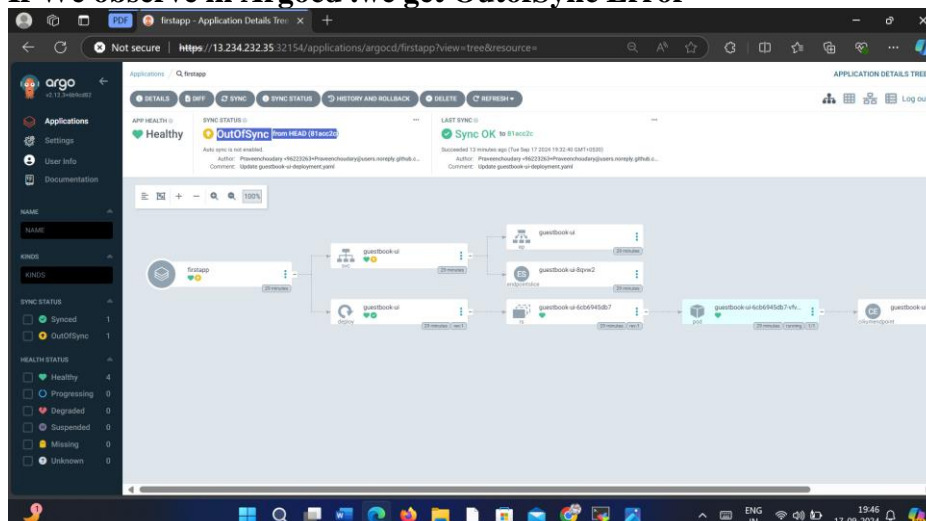
1. Once if we implement ArgoCD, if we make any changes manually in our cluster using the kubectl command, Kubernetes will reject those request from that user. Because when we apply changes manually, ArgoCD will check the actual state of the cluster with the desired state of the cluster (GitHub)

```
root@ip-172-31-9-228:~# kubectl get svc
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
guestbook-ui        LoadBalancer 100.68.176.136  a76ecc72d42ba4f65a55c43b1763a3fd-1858565982.ap-south-1.elb.amazonaws.com 80:30958/TCP    23m
kubernetes           ClusterIP    100.64.0.1      <none>            443/TCP          172m
```

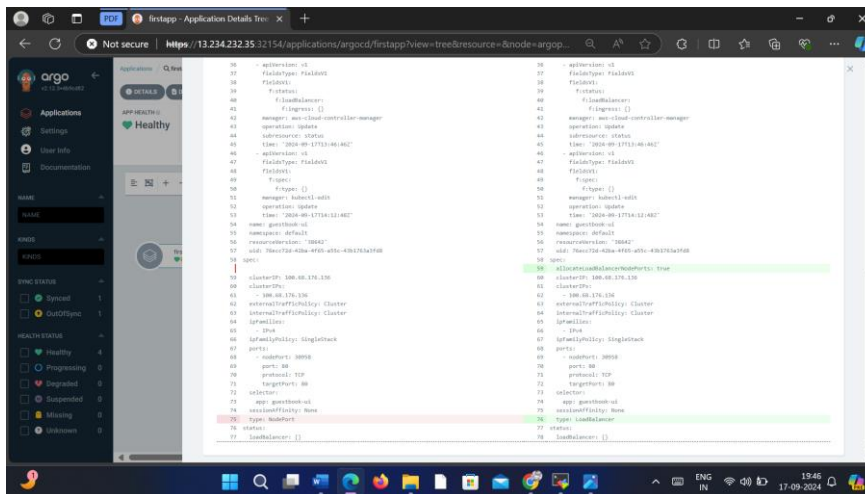
Here The svc type is LoadBalancer.so now I am going to change to NodePort directly in K8s cluster.

```
root@ip-172-31-9-228:~# kubectl get svc
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
guestbook-ui        LoadBalancer 100.68.176.136  a76ecc72d42ba4f65a55c43b1763a3fd-1858565982.ap-south-1.elb.amazonaws.com 80:30958/TCP    23m
kubernetes           ClusterIP    100.64.0.1      <none>            443/TCP          172m
```

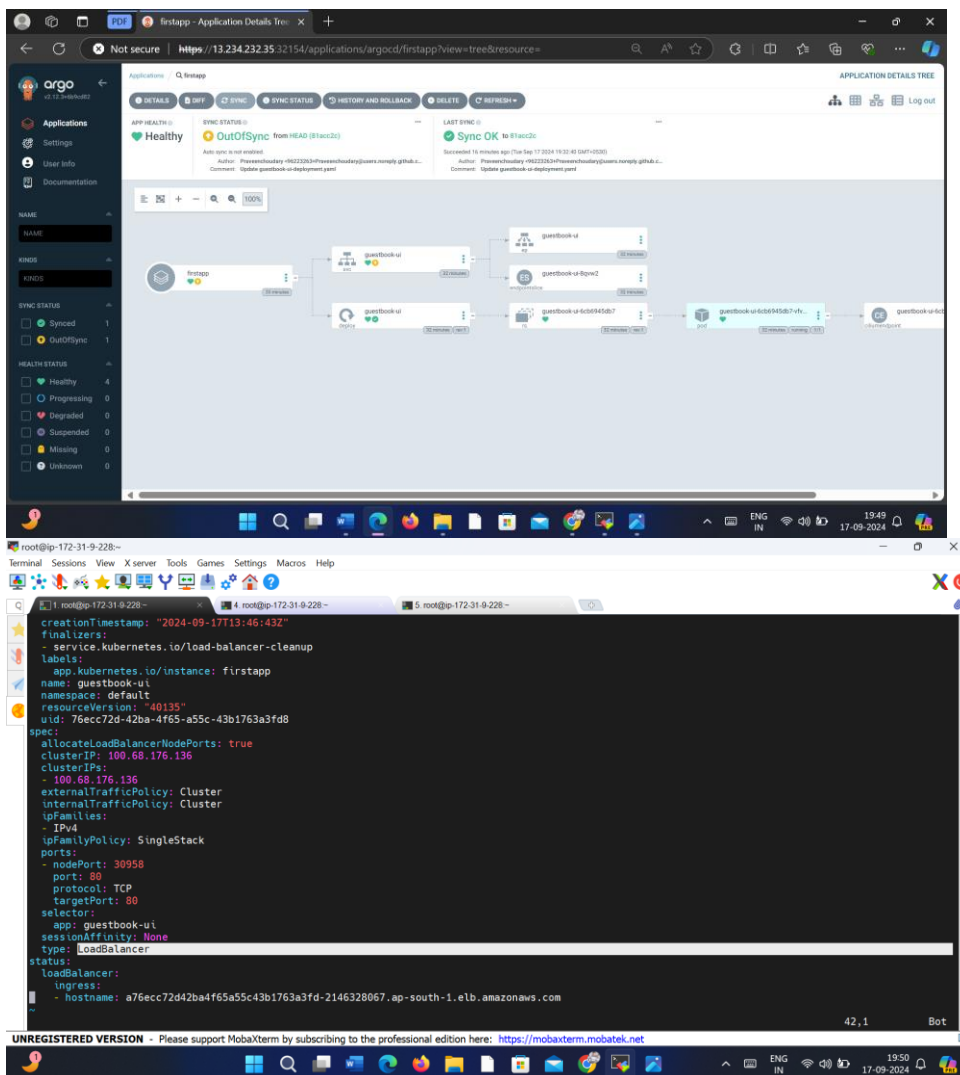
If we observe in ArgoCD .we get OutofSync Error



Argocd Clearly showing Changes in service.yml file



Suppose if You click sync in argocd. The changes will revert back in service .yaml file.



- Previously I Changed type to NodePort from LoadBalancer.
- The Argocd Reject that change We get error OutOfSync.and I clicked manually on sync it the changes is revertback

STEP 5: Access application In my case the type is LoadBalancer.so I can access by using LB.

```
root@ip-172-31-9-228:~
Terminal Sessions View X server Tools Games Settings Macros Help
1 root@ip-172-31-9-228 ~ 4 root@ip-172-31-9-228 ~ 5 root@ip-172-31-9-228 ~
[root@ip-172-31-9-228 ~]# kubectl get svc
NAME          TYPE          CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
guestbook-ui  LoadBalancer 100.68.176.136   a76ecc72d42ba4f65a55c43b1763a3fd-2146328067.ap-south-1.elb.amazonaws.com 80:30958/TCP    39m
kubernetes    ClusterIP      100.64.0.1       <none>           443/TCP          3h7m
[root@ip-172-31-9-228 ~]#
```

