

# Lab 4 - Setting up Automated Deployments with ArgoCD

Author: Gourav Shah

Publisher: School of Devops

Version : v2024.06.02.01

---

**Project:** Setup Automated Deployment to Staging and Prod with Argo CD.

## Setup ArgoCD

Install ArgoCD

```
kubectl create namespace argocd

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

Reset admin password to `password`

```
# bcrypt(password)=$2a$10$rRyBsGSHK6.uc8fntPwVIuLVHgsAhAX7TcdrqW/RADU0uh7CaChLa
kubectl -n argocd patch secret argocd-secret \
-p '{"stringData": {
    "admin.password": "$2a$10$rRyBsGSHK6.uc8fntPwVIuLVHgsAhAX7TcdrqW/
RADU0uh7CaChLa",
    "admin.passwordMtime": "'$(date +%FT%T%Z)'"
}}'
```

Source: [reset-argo-password.sh](#) Reference: [argo-cd/faq.md at master · argoproj/argo-cd · GitHub](#)

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

```
kubectl patch svc argocd-server -n argocd --patch \
'{"spec": { "type": "NodePort", "ports": [ { "nodePort": 32100, "port": 443,
```

```
"protocol": "TCP", "targetPort": 8080 } ] } }'
```

source: [patch\\_argo\\_svc.sh](#)

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

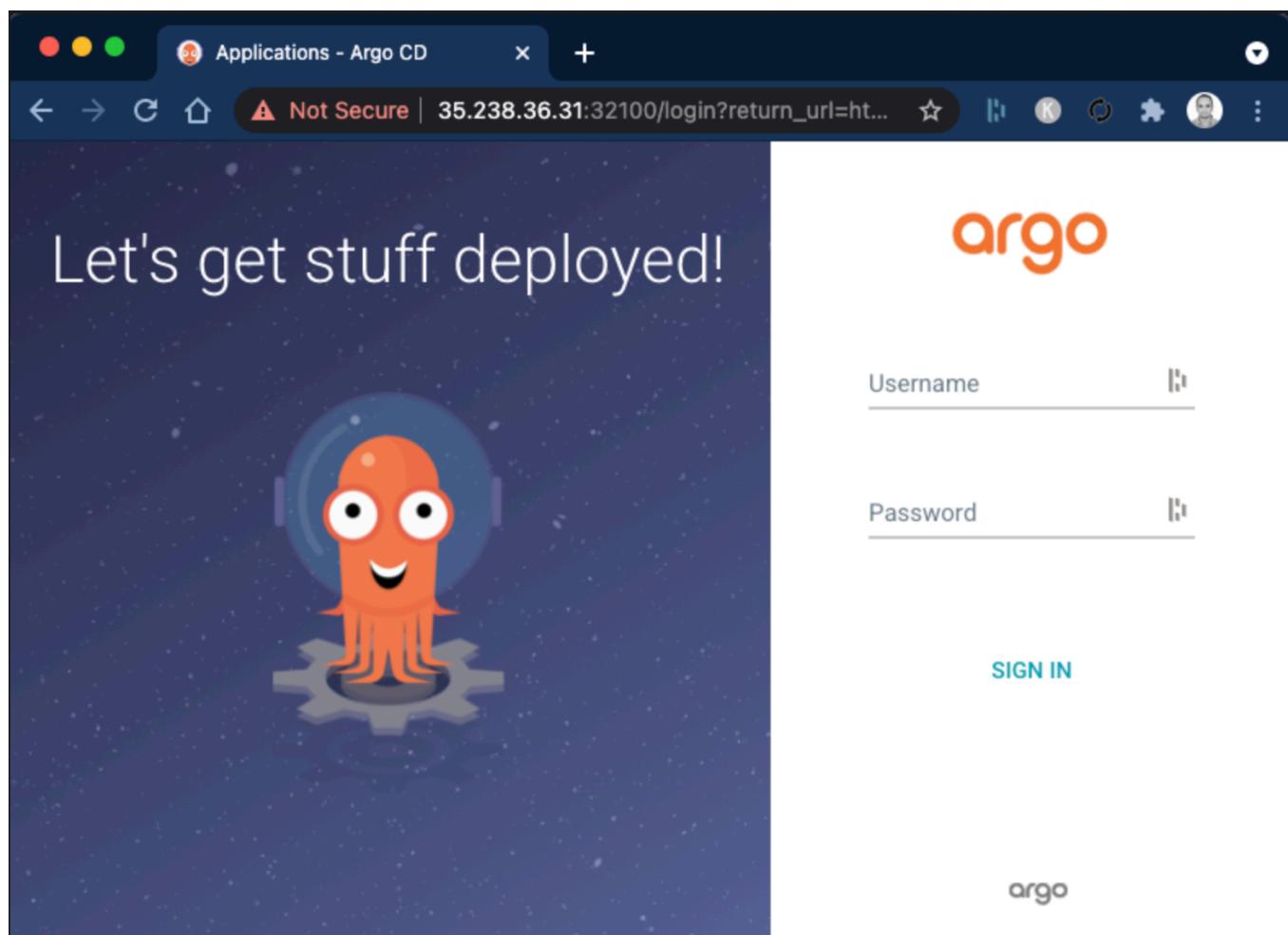
Find out the IP address for one of the nodes. One way to do so is to run the following command,

```
kubectl get nodes -o wide
```

Note IP address for one of the nodes and browse to <https://NODEIP:32100>

where, replace NODEIP with the actual.

You should be presented with the login page for ArgoCD as follows



- username = [admin](#)

- password = password

## Configuring Repository and Project

Ensure that you have checked in all the code from earlier labs and pushed it to your repository.

The screenshot shows a GitHub repository named 'argo-labs'. The repository is public and was forked from 'schoolofdevops/argo-labs'. The main branch is selected. There is 1 branch and 0 tags. A message indicates that the branch is 4 commits ahead of the upstream. The repository contains files: 'base', 'prod', 'staging', 'LICENSE', and 'README.md'. The 'base' file has a note 'added canary release'. The 'prod' file has a note 'added canary release'. The 'staging' file has a note 'updated staging deployment'. The 'LICENSE' and 'README.md' files are labeled as 'Initial commit'.

File	Description
base	added canary release
prod	added canary release
staging	updated staging deployment
LICENSE	Initial commit
README.md	Initial commit

Once logged in to ArgoCD, select `settings` from left menu and browse to `Projects`

The screenshot shows the Argo CD Settings interface. On the left sidebar, the 'Settings' option is selected and highlighted with an orange box. The main content area displays several configuration sections: 'Repositories', 'Repository certificates and known hosts', 'GnuPG keys', 'Clusters', 'Projects' (which is also highlighted with an orange box), 'Accounts', and 'Appearance'. Each section has a brief description below it.

Click on **New Project** → Create and provide Project Name and Description as

The screenshot shows the 'Projects' creation dialog. It features a 'CREATE' button at the top right and a 'CANCEL' button. In the center, there's a 'NAME' field containing 'default'. Below it is a 'GENERAL' section with two input fields: 'Project Name' set to 'instavote' and 'Description' set to 'Example Voting App'. The 'New Project' button in the main projects list and the 'Project Name' and 'Description' fields in the dialog are all highlighted with orange boxes.

Proceed to create the project.

From Project Configuration page that you are redirected to, edit DESTINATIONS

- Select default cluster name from dropdown
- Select `in-cluster` as Name
- Add two entries, one for `staging` and another for `prod` Namespace
- Save

Server	Name	Namespace
https://kubernetes.default.svc	in-cluster	staging
https://kubernetes.default.svc	in-cluster	prod

ADD DESTINATION

From `settings` from left menu and browse to `Repositories`

Select `Connet Repo` and provide the following configuration

- Via: HTTPS
- Type: git
- Project: instavote
- Repository URL: <https://github.com/xxxx/argo-labs.git> (replace with actual)
- Username: GitHub Username (If Private Repo)
- Password: GitHub Password or Token (If Private Repo)

Choose your connection method:

VIA HTTPS ▾

### CONNECT REPO USING HTTPS

Type

git

Project

instavote

#### Repository URL

<https://github.com/devops-0002/argo-labs.git>

Username (optional)

Password (optional)

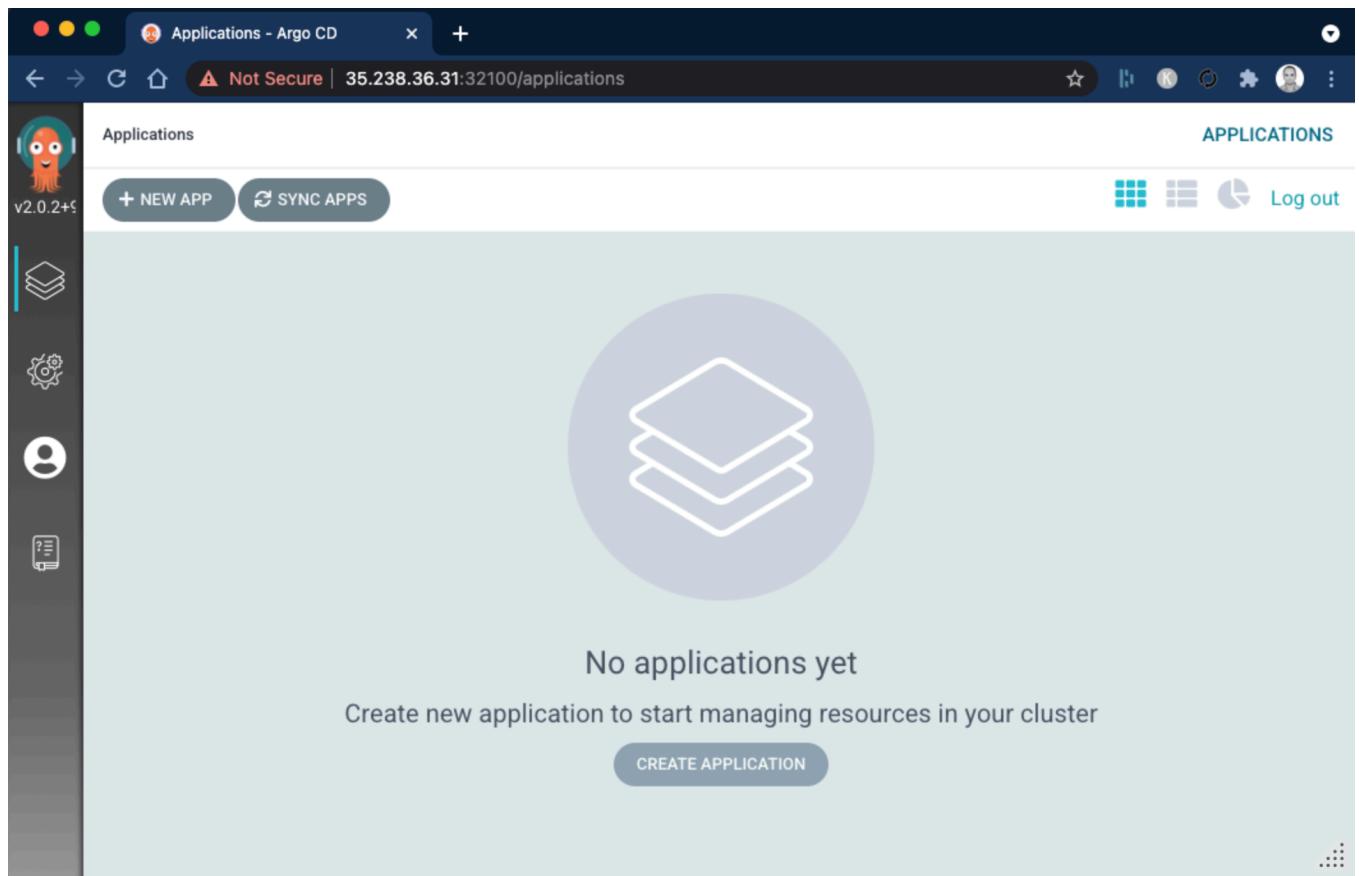
Finally click on **Connect** to add the repo.

# Setup Staging and Prod Deployments with ArgoCD

Clean up resources in the namespaces for `staging` and `prod` environments if you haven't already,

```
cd argo-labs
kubectl delete -k staging/ -n staging
kubectl delete -k prod/ -n prod
```

Browse to ArgoCD web console and click on **Create Application**



From General ,

- Application Name : `vote-staging`
- Project : `instavote`
- Sync Policy : `Automatic`
- Prune Resources: Checked

## GENERAL

Application Name

vote-staging

---

Project Name

instavote

---

## SYNC POLICY

Automatic

---



PRUNE RESOURCES ?



SELF HEAL ?

From Source,

- Repository URL : Your Repo URL (<https://>)
- Revision : main / HEAD
- Path : staging

## SOURCE

Repository URL

<https://github.com/devops-0002/argo-labs.git>

GIT ✓

Revision

HEAD

Branches ▾



Path

staging

From Destination,

- Cluster URL : <https://kubernetes.default.svc> (default)
- Namespace : staging

## DESTINATION

Cluster URL

`https://kubernetes.default.svc`

---

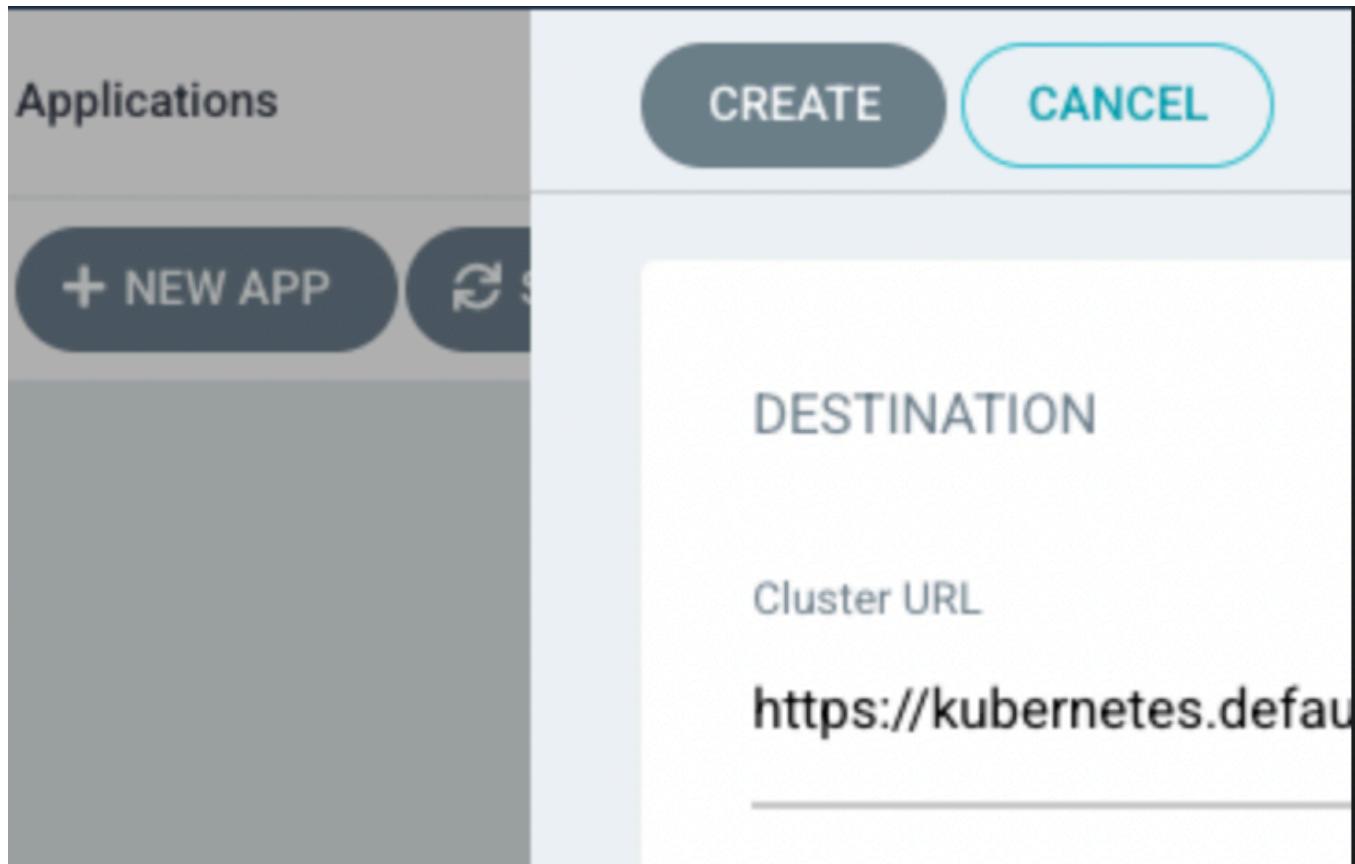
Namespace

`staging`

---



Click on **CREATE** button on the top



validate with

```
kubectl get all -n staging
```

## Set up Deploy to Prod Configuration

You will deploy to prod based on a specific git branch e.g. `release`. Create a release branch to deploy the application to prod with,

```
cd argo-labs/
git pull origin main
git checkout -b release
git push origin release
```

You will see a new branch created on GitHub for this repository. Alternately, you could also create the branch from GitHub Web UI.

Create another application, repeat the same configurations with the following changes,

- Application Name: vote-prod
- Project Name: instavote
- Sync Policy: Automatic
- Prune Resources: Checked
- Repository: Same Repo
- Revision: release (Branches)
- Path: prod
- Cluster URL: default from dropdown.
- Namespace : prod

Create and Sync manually.

The screenshot shows the ArgoCD application management interface. At the top, there's a header with buttons for '+ NEW APP', 'SYNC APPS', 'REFRESH APPS', a search bar ('Search applications...'), and a refresh icon. Below the header, there are two application cards:

- vote-prod** (Project: default)
 

Status:	Healthy Synced
Repository:	https://github.com/initcron/vote-deploy.git
Target Revisi...	release
Path:	prod
Destination:	in-cluster
Namespace:	prod
Created At:	10/04/2023 22:24:05 (21 hours ago)
Last Sync:	10/04/2023 22:32:22 (21 hours ago)

 Buttons: SYNC, REFRESH, DELETE
- vote-staging** (Project: default)
 

Status:	Healthy Synced
Repository:	https://github.com/initcron/vote-deploy.git
Target Revisi...	HEAD
Path:	staging
Destination:	in-cluster
Namespace:	staging
Created At:	10/04/2023 22:22:20 (21 hours ago)
Last Sync:	10/04/2023 22:28:12 (21 hours ago)

 Buttons: SYNC, REFRESH, DELETE

Once synced, you should see two applications configured on ArgoCD tracking two environments.

You could also check the applications using kubectl as

```
kubectl get applications -n argocd
kubectl describe application vote-prod -n argocd
```

# Deployments in Action

Open two terminals and start watching for changes in the staging namespace

Terminal 1

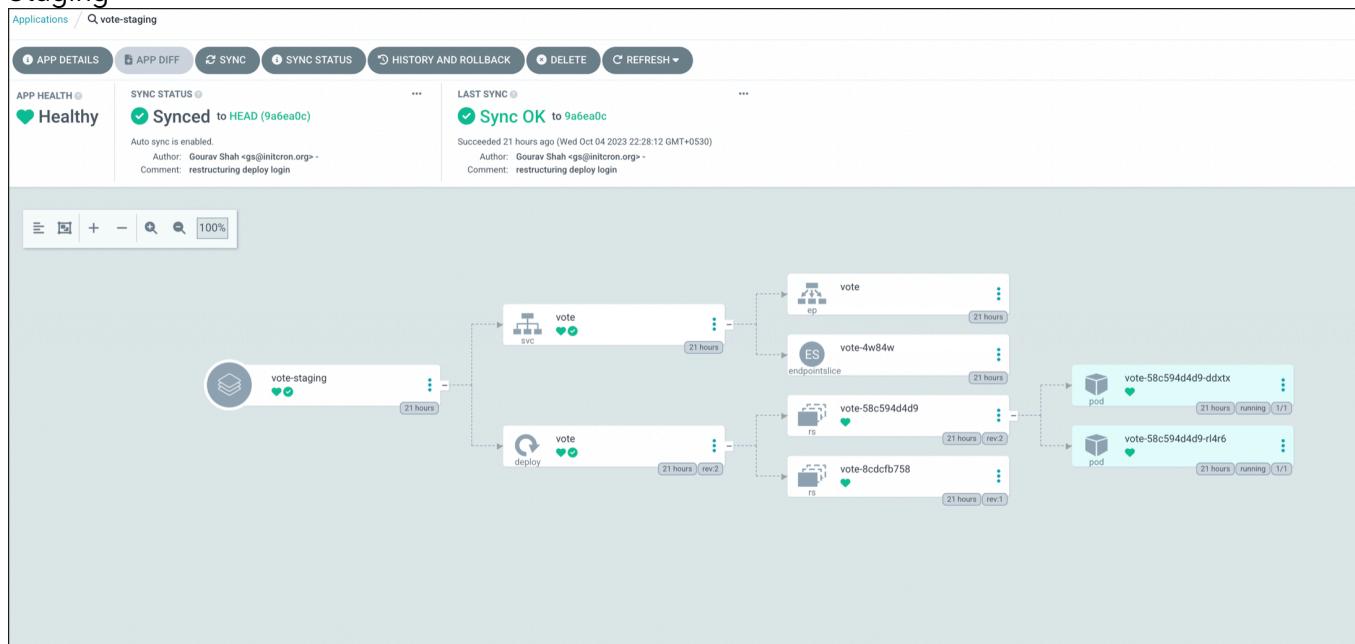
```
watch kubectl get ro,all -n staging
```

Terminal 2

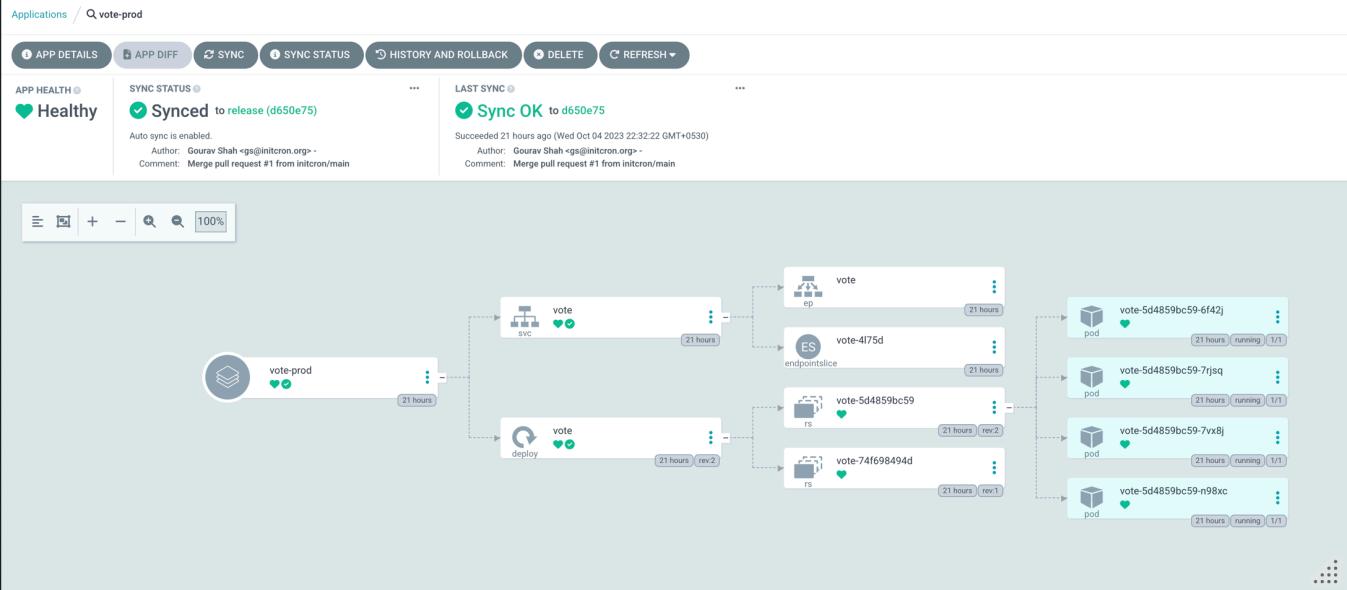
```
watch kubectl get ro,all -n prod
```

Watch for the changes in the console as well as on Argo. You shall see the application synced from the Git Repo to the Kubernetes Cluster in a few seconds.

Staging



Prod

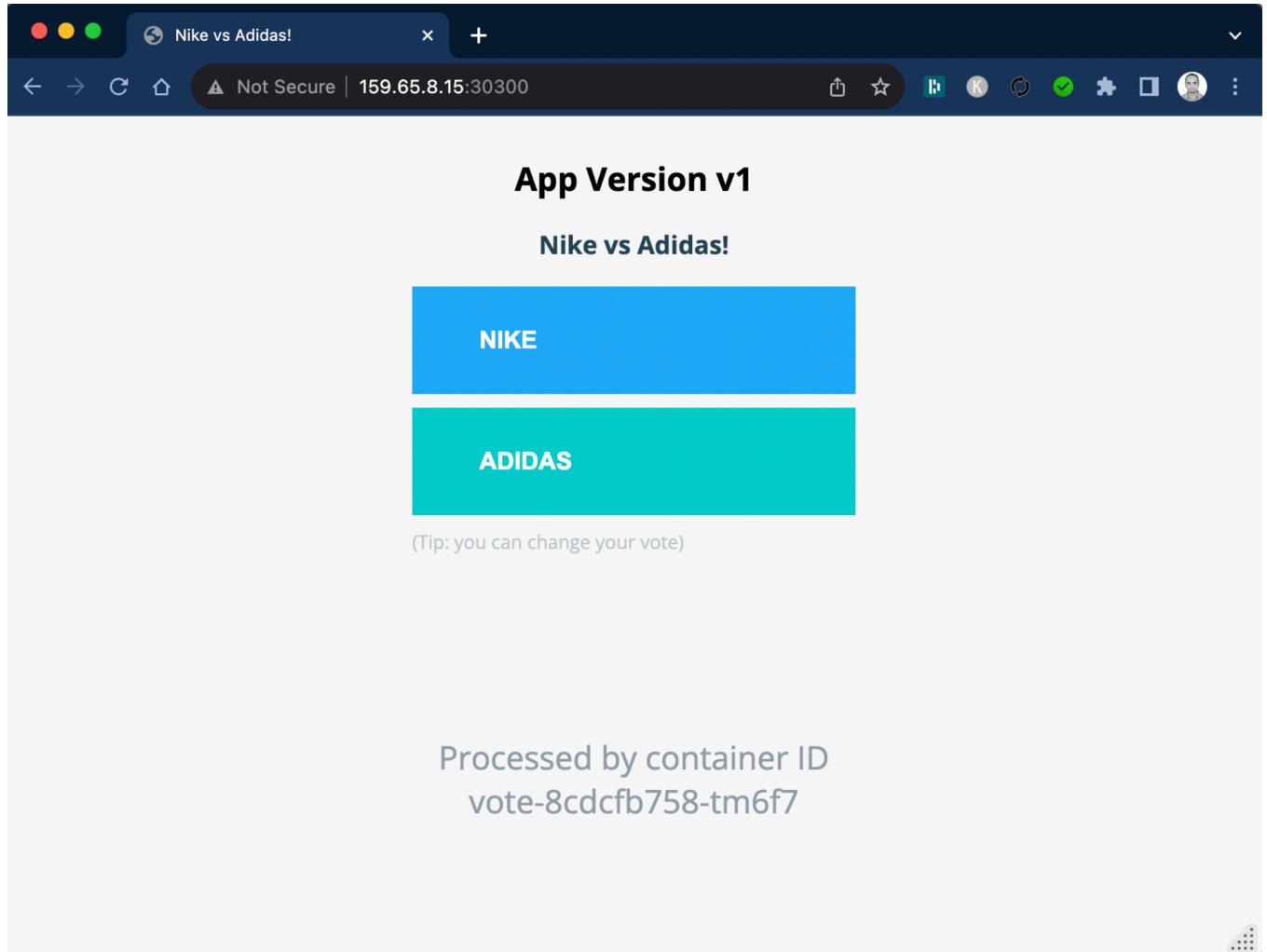


Validate by accessing the vote apps on

- Staging : <http://NODEIP:30000>
- Prod : <http://NODEIP:30200>

where, replace NODEIP with actual IP or Hostname of your cluster node.

e.g.



## Exercises

- Set up branch protection rule to lock down `release` branch and allow changes via pull requests. You can experiment by adding additional policies as well.
- Try modifying YAML manifests in Deploy Repo in Git in main branch by changing the image tag in `staging/kustomization.yaml` and wait for the staging deployment. Then raise the pull request to merge it into release and see it deployed to prod.

## References

- Getting Started with Argo [Getting Started - Argo CD - Declarative GitOps CD for Kubernetes](#)
- Reset admin password [argo-cd/faq.md at master · argoproj/argo-cd · GitHub](#)

#courses/argo/labs/v1