



# Training & Certification

## Lab 14B. ArgoCD

Following CI with Tekton, ArgoCD offers the missing piece of how to set up an automated and sophisticated continuous deployment and delivery system which are based on the principles of GitOps. In this lab exercise, you are going to learn:

- How to set up ArgoCD with a web UI inside an existing Kubernetes cluster.
- How to set up an automated deployment to Kubernetes using the principles of GitOps.

### Setting Up ArgoCD

Install ArgoCD:

```
kubectl create namespace argocd
```

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

Source: GitHub, [install\\_argocd.md](#)

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: GitHub, [reset-argo-passwd.md](#)

Reference: GitHub, [argo-cd/faq.md at master · argoproj/argo-cd](#)

```
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: GitHub, [patch\\_argo.md](#)

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

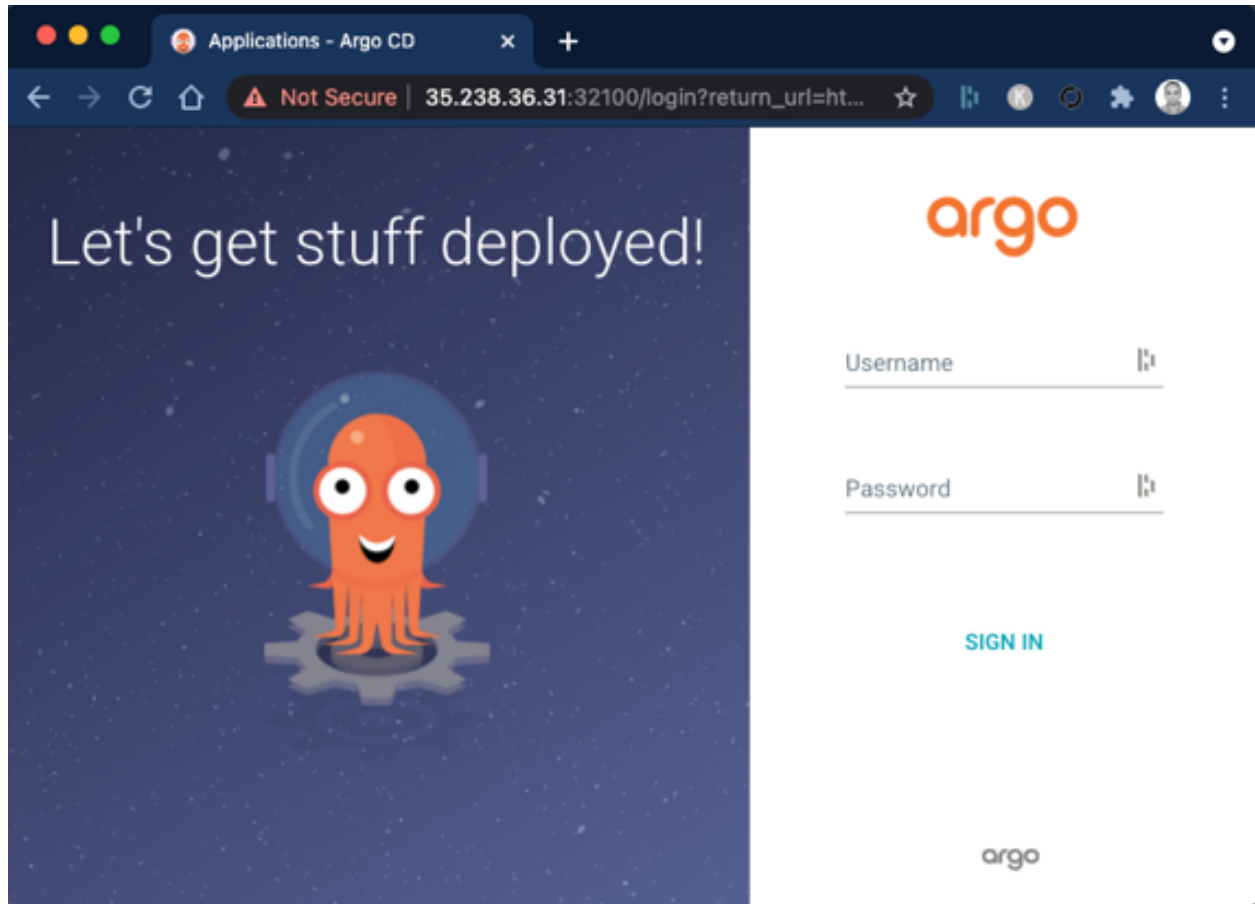
Find out the IP address for one of the nodes. One way to do that, is to run the following command:

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

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

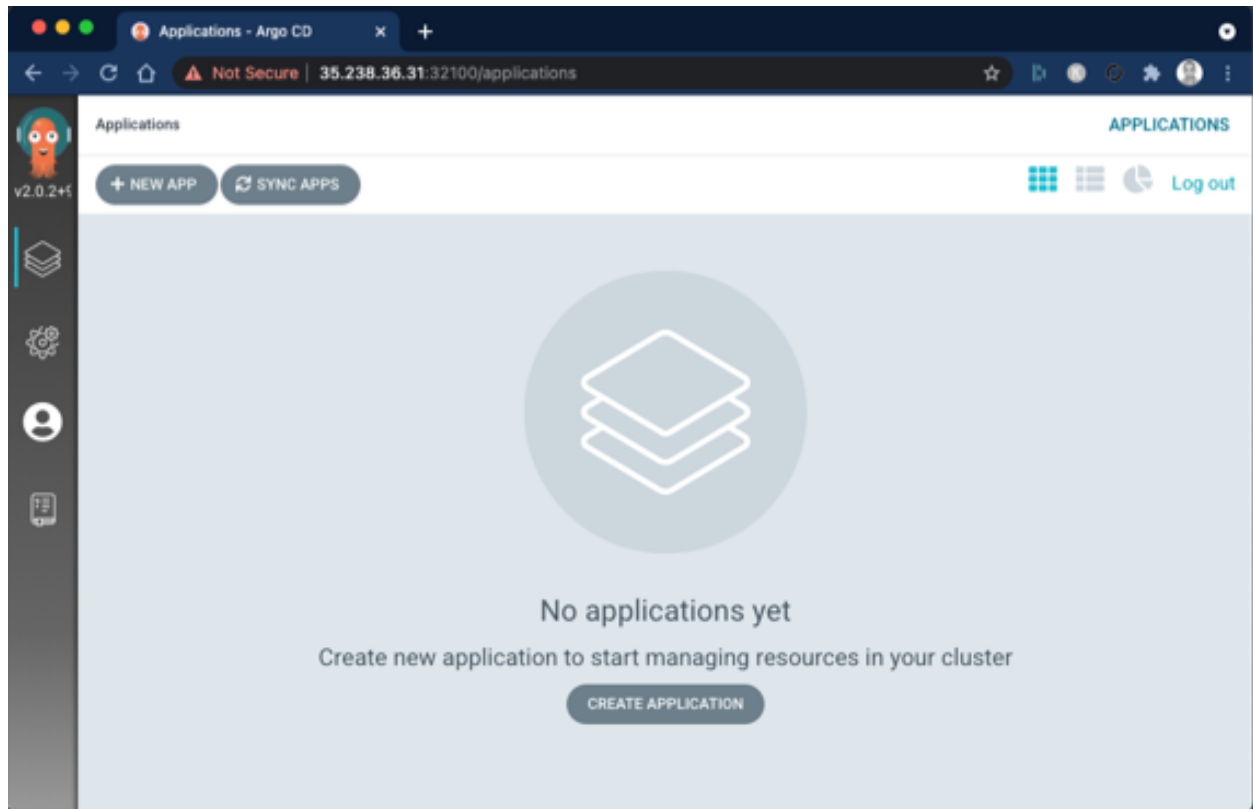
Replace **NODEIP** with the actual IP address of the node listed above.

You should be presented with the login page for ArgoCD:



- username = **admin**
- password = **password**

Once logged in, you should see a screen such as this one:



## Preparing Application Deployment Manifests

Create a fork of GitHub's [schoolofdevops/vote](https://github.com/schoolofdevops/vote) repository. Clone it to the host where `kubectl` is available and switch to it:

```
git clone https://github.com/xxxxxxx/vote.git
cd vote
mkdir deploy
cd deploy
```

Replace `xxxxxxx` with the actual user/organization name.

Generate YAML manifests to deploy `vote` app:

```
kubectl create deployment vote \
  --image=schoolofdevops/vote:v1 \
  --replicas=4 \
  --port=80 \
  --dry-run=client -o yaml | tee vote-deploy.yaml
```

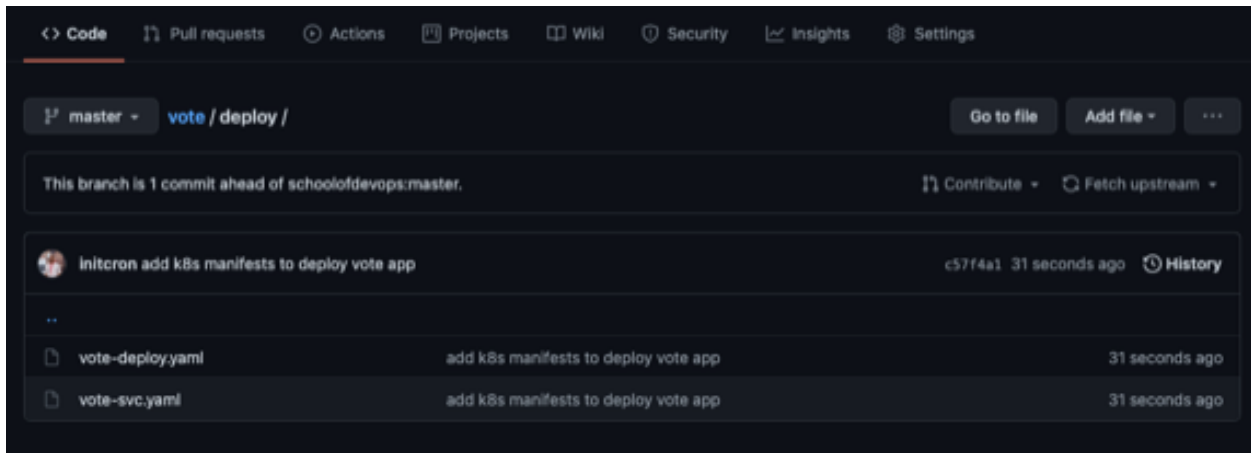
Generate manifest to create a Load Balancer (Service):

```
kubectl create service nodeport \  
  vote --tcp=80 \  
  --node-port=30100 \  
  --dry-run -o yaml | tee vote-svc.yaml
```

Add and commit to the git repository:

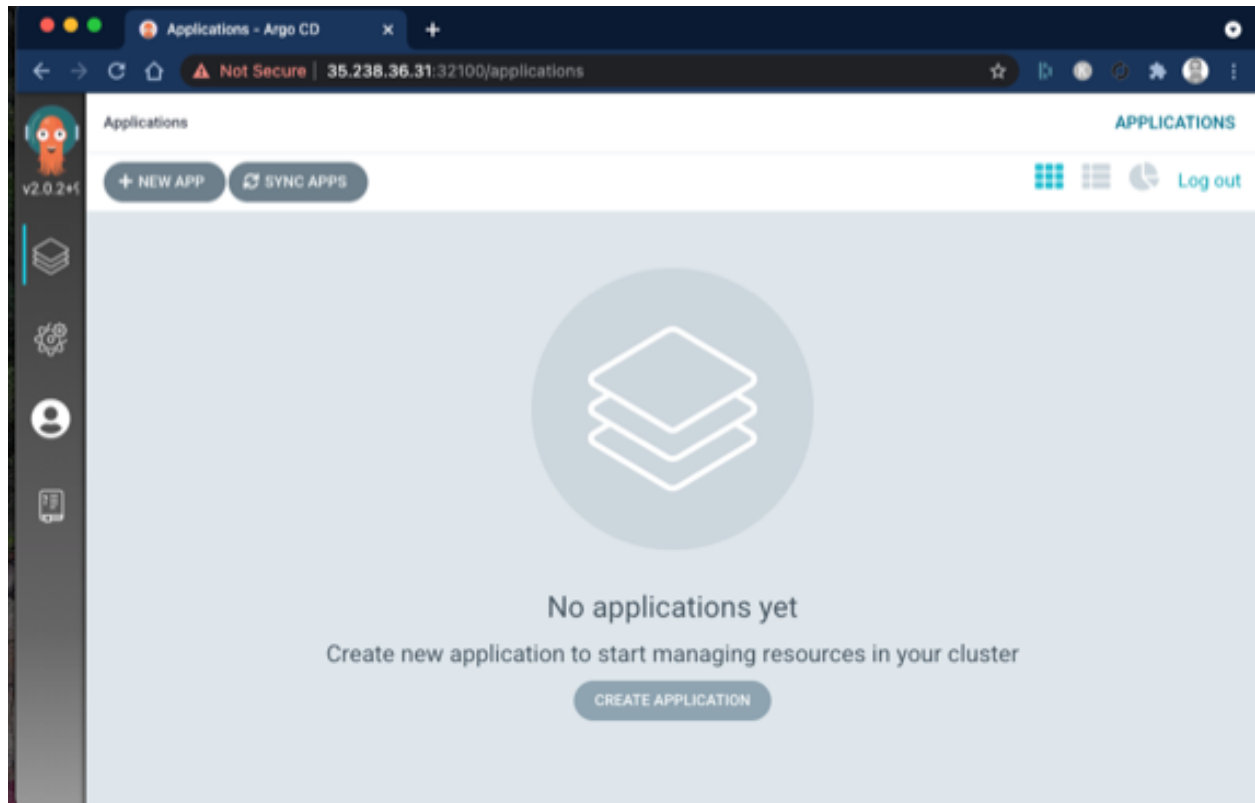
```
git add vote-deploy.yaml vote-svc.yaml  
git commit -am "add k8s manifests to deploy vote app"  
git push origin master
```

Validate that these manifests are reflected in your git repository:



## Setting Up an Automated Deployment with ArgoCD

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



In *General* set the following options:

- Application Name: “vote”
- Project: “default”
- Sync Policy: “Manual”

## GENERAL

Application Name

**vote**

---

Project

**default**

default

SYNC POLICY

**Manual**

---

In *Source* set the following options:

- Repository URL: Your repository URL (https)
- Revision: "HEAD"
- Path: "deploy"

SOURCE

Repository URL

https://github.com/gouravshah/vote.git

GIT ▾

Revision

HEAD

Branches ▾ ⓘ

Path

deploy

In *Destination* set the following options:

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

DESTINATION

Cluster URL

https://kubernetes.default.svc

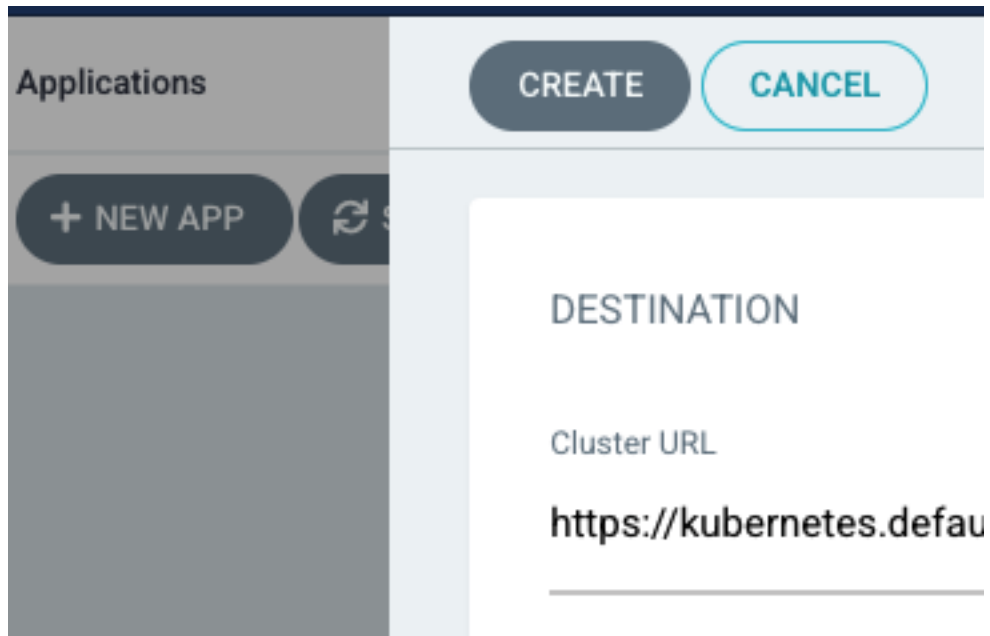
URL ▾

Namespace

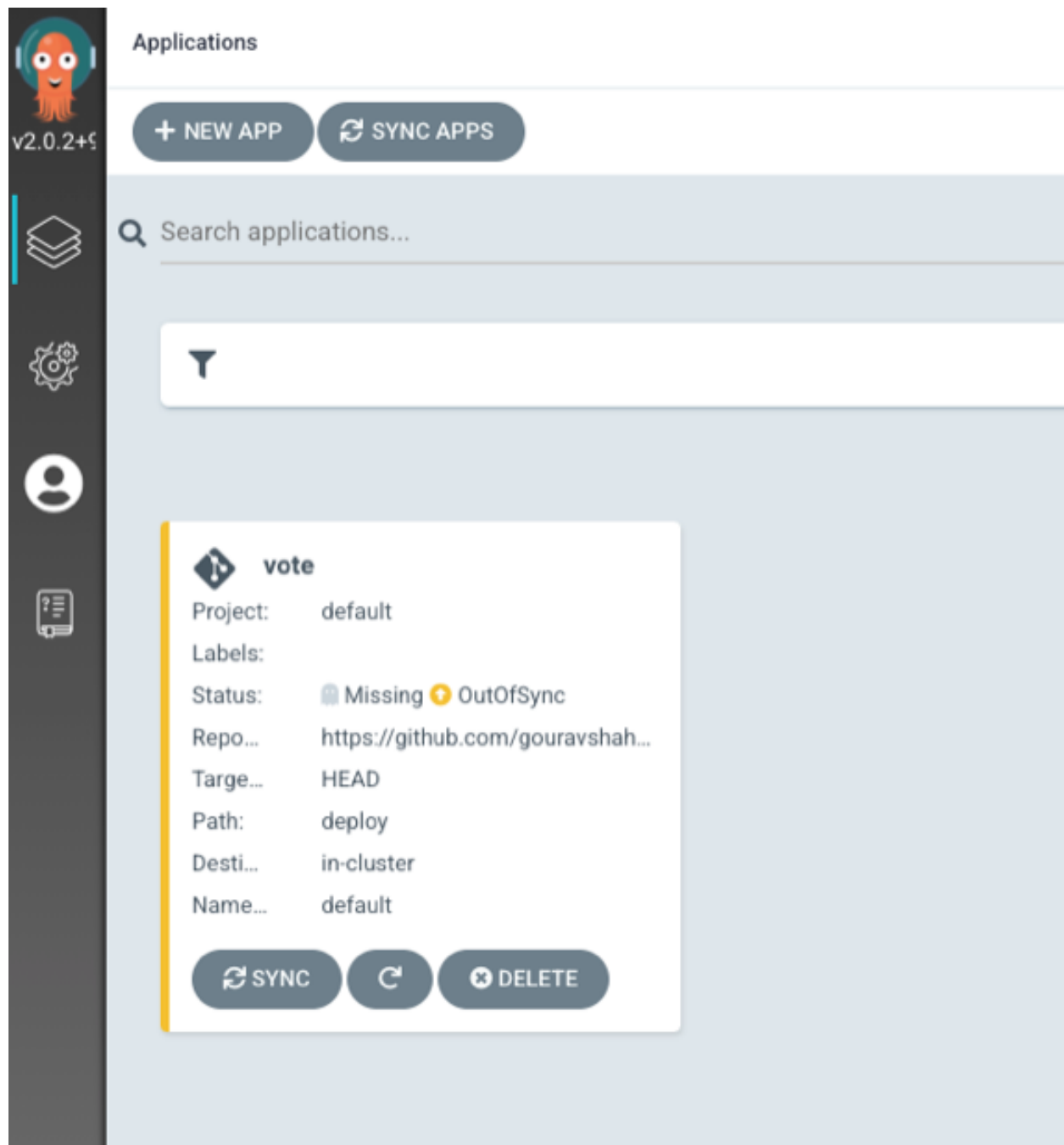
default

Click on the *CREATE* button on the top:





You should now see a `vote` application created.



## Deploy to Kubernetes

Before you begin deployment, go to the `kubectl` console and start looking for changes in the default namespace:

```
watch kubectl get all -n default
```

Select the `vote` application on ArgoCD to access more details:

The screenshot displays the ArgoCD web interface for the 'vote' application. The top section shows the application's metadata:

- Project:** default
- Labels:**
- Status:** Missing (ghost icon) OutOfSync (yellow exclamation mark icon)
- Repo...** https://github.com/gouravshah...
- Target...** HEAD
- Path:** deploy
- Desti...** in-cluster
- Name...** default

Below the metadata are three buttons: **SYNC** (refresh icon), **RETRY** (refresh icon), and **DELETE** (trash icon).

The bottom section, titled 'APPLICATION DETAILS', contains a navigation bar with buttons: **APP DETAILS**, **APP DIFF**, **SYNC**, **SYNC STATUS**, **HISTORY AND ROLLBACK**, **DELETE**, and **REFRESH**. On the right, there are icons for user, roles, clusters, and a **Log out** link.

The main content area shows the **APP HEALTH** as **Missing** (ghost icon) and the **CURRENT SYNC STATUS** as **OutOfSync** (yellow exclamation mark icon) from **HEAD (c57f4a1)**. A **MORE** link is available. The sync status details include:  
Author: Gourav Shah <gs@initcron.org>  
Comment: add k8s manifests to deploy vote app

At the bottom, a diagram illustrates the application's structure. It shows a 'vote' application (ghost icon) with a '3 minutes' sync interval, which is linked to a 'vote svc' (service) and a 'vote deploy' (deployment). Both the service and deployment are shown as 'OutOfSync' (yellow exclamation mark icon).



Go ahead and click on the *SYNC* button. You should see screen such as one below:

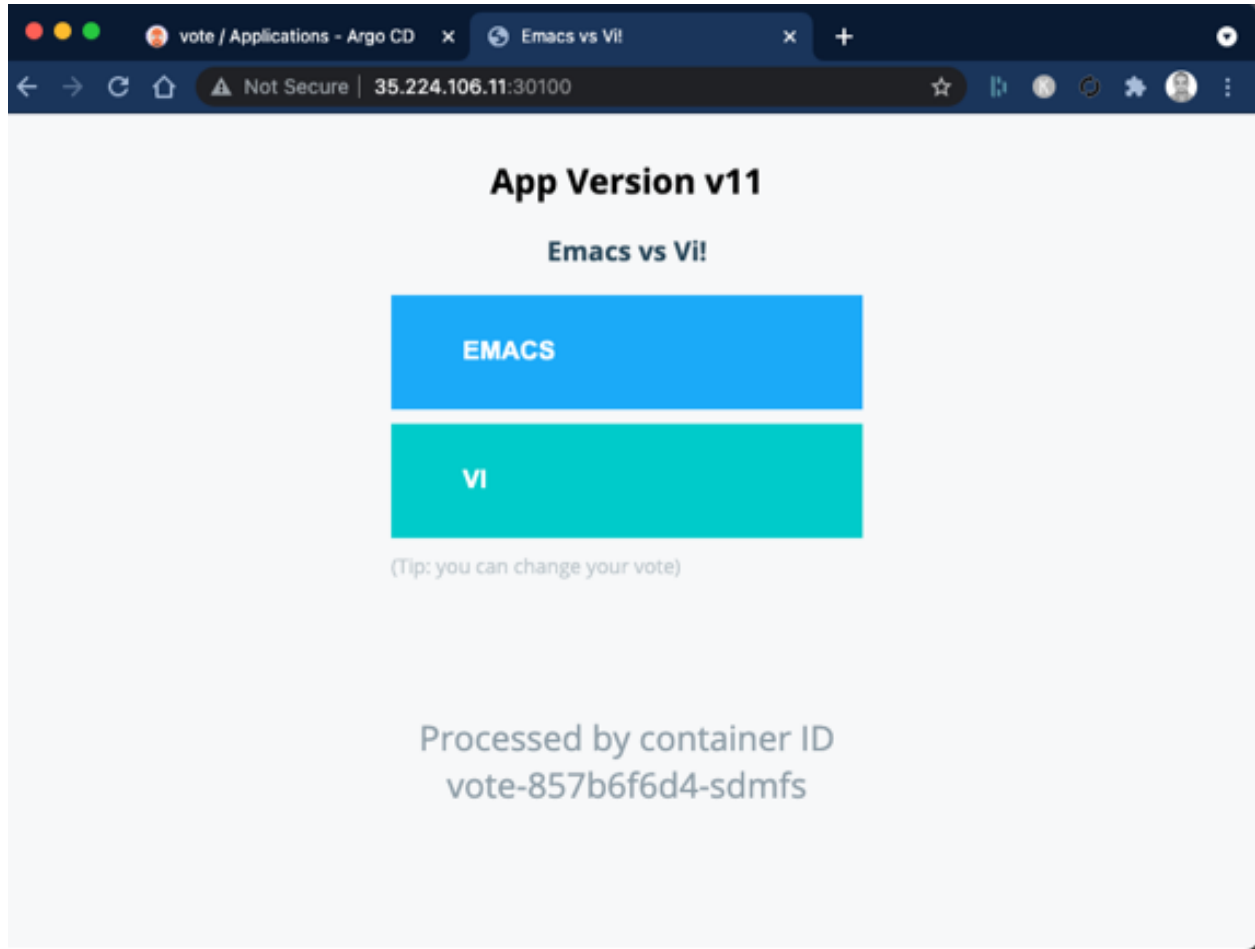
A screenshot of a 'Synchronization' dialog box. At the top, there are two buttons: 'SYNCHRONIZE' and 'CANCEL', with a close 'X' button on the right. The main text says 'Synchronizing application manifests from https://github.com/gouravshah/vote.git'. Below this, it shows 'Revision HEAD'. There are four checkboxes: 'PRUNE', 'DRY RUN', 'APPLY ONLY', and 'FORCE'. Under 'SYNC OPTIONS', there are four more checkboxes: 'SKIP SCHEMA VALIDATION', 'AUTO-CREATE NAMESPACE', 'PRUNE LAST', and 'APPLY OUT OF SYNC ONLY'. There is also a 'REPLACE' checkbox with a yellow warning triangle icon. Below these is a dropdown menu for 'PRUNE PROPAGATION POLICY' currently set to 'foreground'. At the bottom, it says 'SYNCHRONIZE RESOURCES:' followed by 'all / out of sync / none'. Two resources are listed with checkboxes and a plus icon: '/SERVICE/DEFAULT/VOTE' and 'APPS/DEPLOYMENT/DEFAULT/VOTE', both of which are checked.

Keep everything as is and click on the *SYNCHRONIZE* button.

Watch for the changes in the console as well as in Argo. You should see the application synced from the git repository to the Kubernetes cluster in a few seconds.



Validate by accessing the `vote` application on `http://NODEIP:30100`.



Now:

- Enable auto-sync by browsing to *App Details* → *ENABLE AUTO-SYNC*.
- Try modifying YAML manifests in git and either do a Manual sync, or wait for the Auto sync to see if changes in git are reflected in the cluster. Some configurations you can play with include:
  - image: `schoolfodevops/vote: vxx` (you can use tags v1 to v9 which are available on Docker Hub)
  - replicas
  - nodePort in service
- Create YAML manifests for a `sysfoo` application, add it to the repository and have it synced automatically with ArgoCD.

## Additional Resources

- [“Getting Started with Argo”](#)

## Summary

GitOps is an up and coming technology, with ArgoCD and FluxCD being the two most popular options to implement it. In this exercise, you have learned just enough about GitOps to get you curious and explore this topic further. You can start diving deeper into the world of GitOps with the Linux Foundation's course: [\*"GitOps: Continuous Delivery on Kubernetes with Flux"\*](#) ([\*LFS269\*](#)).