############################# Pre-requisite ####################################

# # AWS CLI – On Windows

* Download AWSCLIV2.msi file from awscli.amazonaws.com/AWSCLIV2.msi

# # kubectl – On Windows using chocolatey

* choco install kubernetes-cli

# # eksctl - On Windows using chocolatey

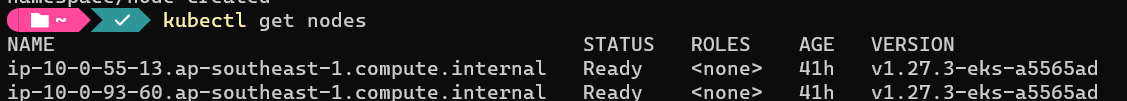
* chocolatey install eksctl

# # argocd-cli

* chocolatey install argocd-cli

############################# Install Argo CD on EKS cluster #############################

1. Check if kubectl is working as expected   
   kubectl get nodes

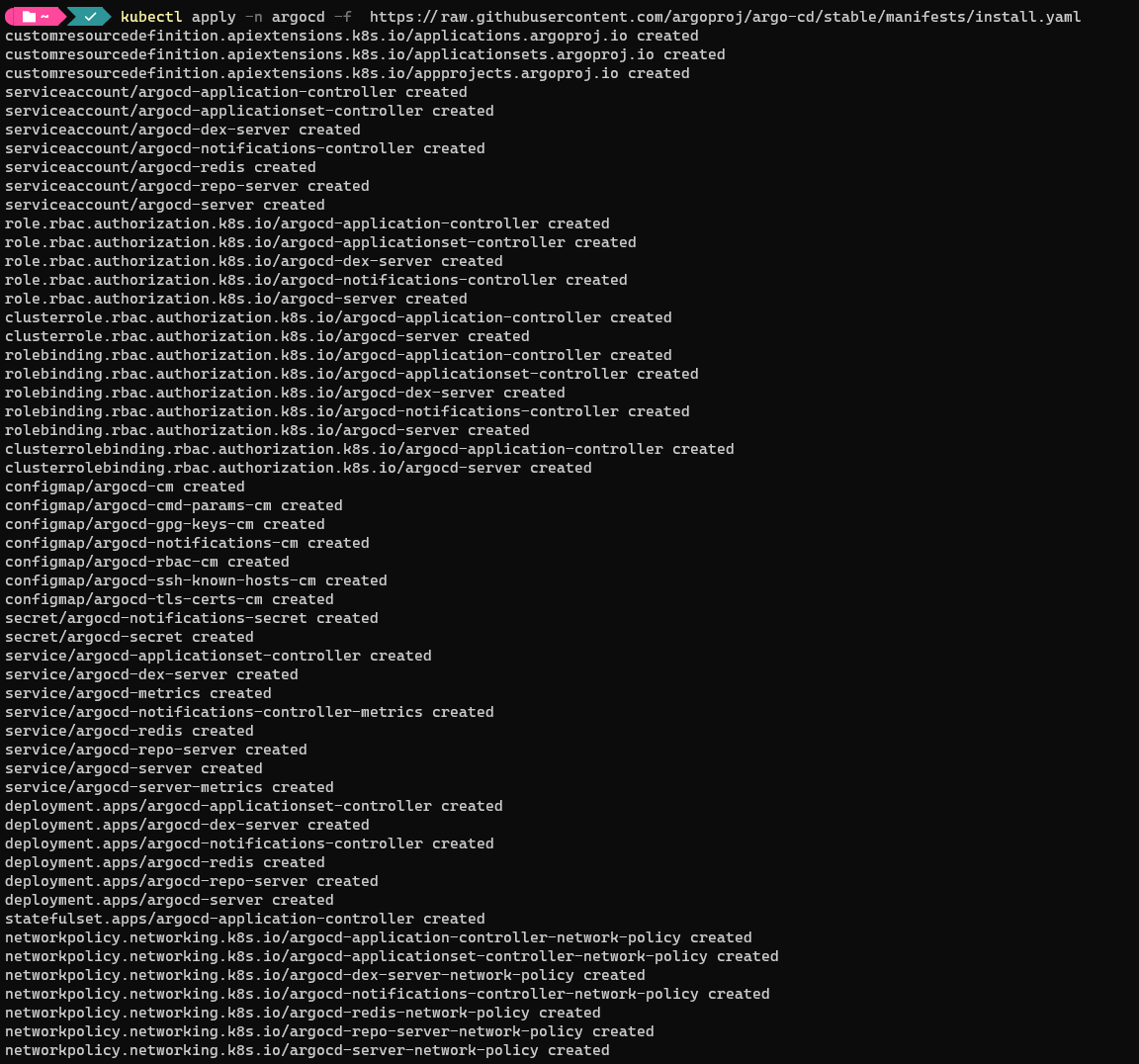


1. Create namespace  
   kubectl create namespace argocd



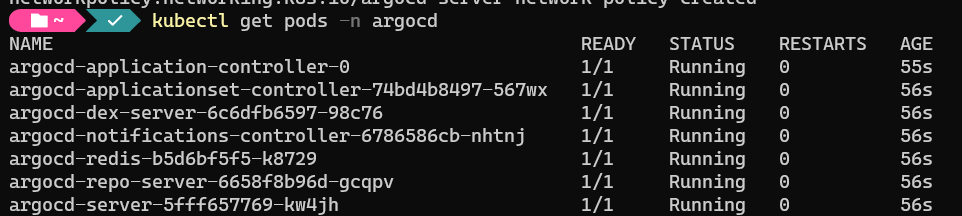
1. Run the Argo CD install script provided by the project maintainers

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



1. Check the status of your Kubernetes pods.

kubectl get pods -n argocd



############################# Access to Argo CD ###########################

# # Forwarding Ports to Access Argo CD

Retrieve the admin password which was automatically generated during installation and decode from base64 from online.

kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath=”{.data.password}”



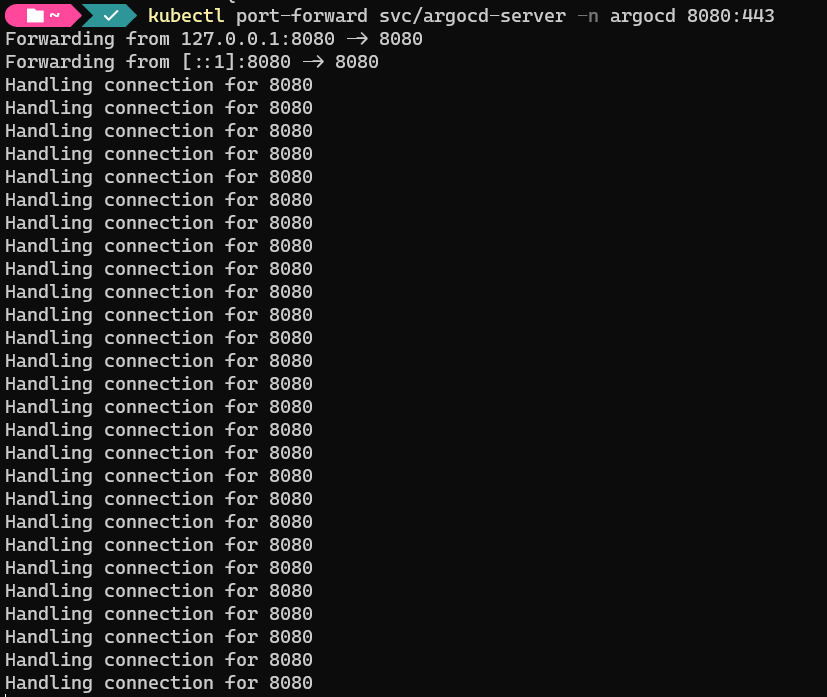
decode base64: aQOOteoI6UV9di9C

A screenshot of a computer

Description automatically generated

Forward those to arbitrarily chosen other ports, like 8080

kubectl port-forward svc/argocd-server -n argocd 8080:443



Access from internet from browser

[http://localhost:8080](http://localhost:8080/)

use id: admin

password: <base64 decoded password>

A picture containing text, screenshot, software, multimedia software

Description automatically generated

A screenshot of a web browser

Description automatically generated with low confidence

########################### Create an application in Argo CD using GUI #####################

1. Create a new project named sd4871-pi-sharp

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Add source repositories.

A white rectangular object with a black border

Description automatically generated

Add destination.

A screenshot of a computer

Description automatically generated

Add cluster resource allow list

A screenshot of a computer

Description automatically generated

1. Settings → Repositories → Connect Repo → Create a new token in github and add devops repo

A screenshot of a computer

Description automatically generated

1. Click the "New app" button on Argo CD UI and fill the following details:
   * application name: msa-helm
   * project: sd4871-pi-sharp
   * repository URL: https://github.com/hieptran228/sd4871\_devops
   * path: deploy/helm/apps
   * Cluster: https://kubernetes.default.svc (this is the same cluster where ArgoCD is installed)
   * Namespace: helm-app
   * Leave all the other values empty or with default selections. Finally click the Create button. The application entry will appear in the main dashboard. Click on it.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Syncing the application

When the application created, the status should be OutOfSync because we are using Manual in Sync Policy.

A screenshot of a computer

Description automatically generated

Note: At this point, I can’t sync because argocd can’t access ECR repo in private mode. I will solve this after install ArgoCD Image Updater and configure AWS and ECR credential.

A screenshot of a computer

Description automatically generated

########################### Install ArgoCD Image updater #####################

1. Install argocd image updater

kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj-labs/argocd-image-updater/stable/manifests/install.yaml

2. Create aws secret contains accesskey. Edit your information from this file

[raw.githubusercontent.com/hieptran228/sd4871\_devops/main/deploy/aws/aws-secret.yaml](https://raw.githubusercontent.com/hieptran228/sd4871_devops/main/deploy/aws/aws-secret.yaml)

And apply: kubectl apply –f aws-secret.yaml -n argocd

3. Create cronjob to refresh ECR token from this file

[raw.githubusercontent.com/hieptran228/sd4871\_devops/main/deploy/ecr/ecr-cronjob.yaml](https://raw.githubusercontent.com/hieptran228/sd4871_devops/main/deploy/ecr/ecr-cronjob.yaml)

kubectl apply –f [ecr-cronjob.yaml](https://raw.githubusercontent.com/hieptran228/sd4871_devops/main/deploy/ecr/ecr-cronjob.yaml) -n argocd

4. Update ArgoCD configmap:

kubectl get configmap --namespace argocd argocd-image-updater-config -o yaml > argocd-image-updater-config.yml

code ./argocd-image-updater-config.yml

Add to bottom

data:

registries.conf: |

registries:

- name: Elastic Container Registry

prefix: 913820192915.dkr.ecr.ap-southeast-1.amazonaws.com

api\_url: https://913820192915.dkr.ecr.ap-southeast-1.amazonaws.com

credentials: secret:argocd/aws-ecr-creds#creds

Then apply: kubectl apply -f ./argocd-image-updater-config.yml -n argocd

5. Add git token for git writeback:

kubectl --namespace argocd create secret generic github-token --from-literal=username=<github\_username> --from-literal=password=<github\_token>

6. Restart

kubectl -n argocd rollout restart deployment argocd-image-updater

7. Sync app in argocd

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

### 8. Annotate our msa-helm

kubectl annotate application msa-helm \

argocd-image-updater.argoproj.io/write-back-method: git:secret:argocd/github-token\

argocd-image-updater.argoproj.io/git-branch: main\

argocd-image-updater.argoproj.io/image-list="backend=913820192915.dkr.ecr.ap-southeast-1.amazonaws.com/backend-nashtech-hieptran-sd4871, frontend=913820192915.dkr.ecr.ap-southeast-1.amazonaws.com/frontend-nashtech-hieptran-sd4871" \

argocd-image-updater.argoproj.io/backend.helm.image-name= backend.image.name\

argocd-image-updater.argoproj.io/backend.helm.image-tag= backend.image.tag\

argocd-image-updater.argoproj.io/frontend.helm.image-name= frontend.image.name\

argocd-image-updater.argoproj.io/ frontend.helm.image-tag= frontend.image.tag

### 9. Commit to msa app (change title to Todos – V3.0.1 - PiSharp)

A screenshot of a computer

Description automatically generatedArgoCD will out-of-sync

A screenshot of a browser window

Description automatically generated

We need to manual sync. After sync, we will have latest version up.

A screenshot of a computer

Description automatically generated

A screenshot of a phone

Description automatically generated

ArgoCD Rollouts – Blue Green strategy

1. Install Argo Rollouts to argo-rollouts namespace.

kubectl create namespace argo-rollouts

kubectl apply -n argo-rollouts -f <https://github.com/argoproj/argo-rollouts/releases/latest/download/install.yaml>

A screenshot of a computer

Description automatically generated

1. Add rollout.yaml to frontend and backend
2. Update MSA app to trigger Jenkins pipeline and new tag is pushed.

Rollouts now control and create pods in new tag (tag: 18) and keep old pods running (tag: 17). Argo status: Suspended.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Now, we can check and promote tag 18 if pass tests.

A screenshot of a computer

Description automatically generated