**SESSION - 83**

--> Create directory.

--> cd user

**user/Dockerfile**

FROM node:20-alpine3.21 AS builder

WORKDIR /opt/server

COPY package.json .

COPY \*.js .

RUN npm install

FROM node:20-alpine3.21

RUN addgroup -S roboshop && adduser -S roboshop -G roboshop

ENV MONGO\_URL="mongodb://mongodb:27017/users" \

REDIS\_URL="redis://redis:6379" \

MONGO=true

WORKDIR /opt/server

USER roboshop

COPY --from=builder /opt/server /opt/server

CMD ["node","server.js"]

# FROM node:20

# WORKDIR /opt/server

# ENV MONGO="true"

# ENV REDIS\_URL="redis://redis:6379"

# ENV MONGO\_URL="mongodb://mongodb:27017/users"

# COPY package.json .

# COPY \*.js .

# RUN npm install

# RUN apt-get update -y \

# && apt-get install net-tools git -y \

# && apt-get clean

# CMD ["node","server.js"]

--> create user-ci in jenkins

--> create user-cd directory

**user-cd/templates/deployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: user

namespace: roboshop

# deployment labels

labels:

component: user

project: roboshop

tier: app

spec:

replicas: 1

# These are the labels replica set use to create pod replicas, this should match pod labels

selector:

matchLabels:

component: user

project: roboshop

tier: app

# This is pod definition

template:

metadata:

labels:

component: user

project: roboshop

tier: app

spec:

containers:

- name: user

image: "{{ .Values.deployment.imageURL}}:{{ .Values.deployment.imageVersion }}"

command: ["sh", "-c", "echo 'failure' && exit 1"]

resources:

requests:

cpu: "50m"

memory: "128Mi"

limits:

cpu: "100m"

memory: "256Mi"

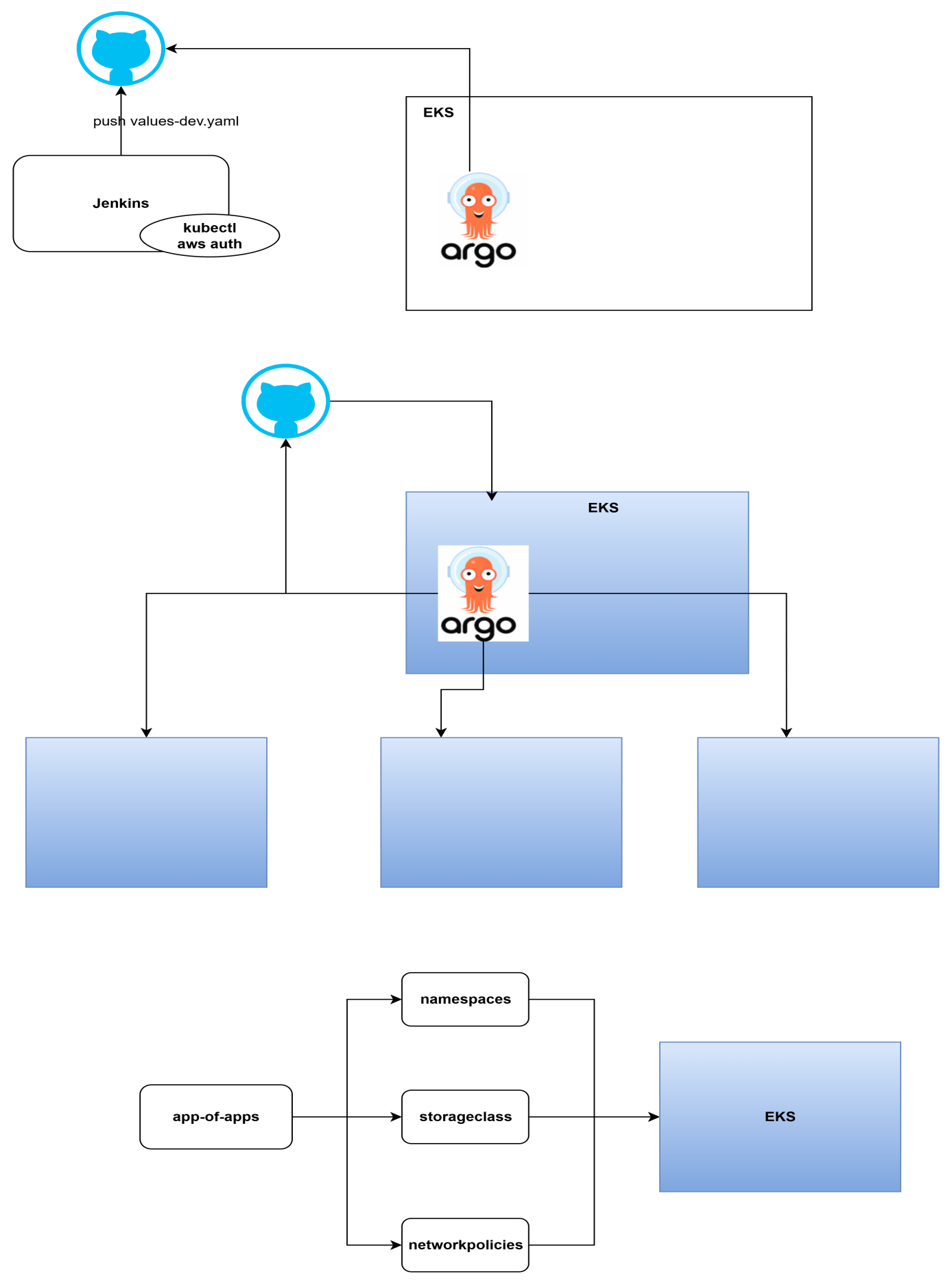
envFrom:

- configMapRef:

name: user

--> push the code

--> console output in jenkins



aws ecr wait image-scan-complete \

--repository-name roboshop/user \

--image-id imageTag=${appVersion} \

--region ${REGION} \

Jenkins -> CICD

Helm

ArgoCD is a deployment tool to deploy applications into kubernetes as well as to manage the cluster

1. Maintan/manage the cluster

2. Deploy/manage applications

**GitOps**

**=========**

Git as a single source of truth

Everything should be inside git

Instead of some need to apply or issues the command, gitops means if something is changed in git repos, it should automatically sync

If we change the image from 1.0.0 in helm values.yaml to 1.0.1 instead we issue helm command, argocd will be looking this git repo and get the changes automatically

--> login bastion host inside.

--> kubectl create namespace argocd

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

--> curl -sSL -o argocd-Linuc-amd64

<https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64>

--> sudo install -m 555 argocd-linux-amd64 /usr/local/bin/agocd

--> rm argocd-linux-amd64

--> kubectl patch svc argocd-server -n argocd -p ‘{“spec”: {“type”; “LoadBalance”}}’

**Advantages**

**============**

1. no need to install extra tools inside jenkins like kubectl

2. no need to provide authenticatioin to jenkins about eks

3. gitops -> git as a single source of truth, we no need to apply the changes manually it will automatically sync

4. revert to the old version in values.yaml then it will restore authentically or raise a PR and change to previous image

5. clean UI

6. ArgoCD is used for cluster management also

kubectl create namespace argocd

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

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

curl -sSL -o argocd-linux-amd64 https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64

sudo install -m 555 argocd-linux-amd64 /usr/local/bin/argocd

rm argocd-linux-amd64

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

argocd admin initial-password -n argocd

--> aws configure - configure the aws

--> aws eks update-kubeconfig --region us-east-1 --name roboshop-dev

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

--> install k9s

--> take duplicate session

--> curl -sS <https://webinstall>.dev/k9s | bash

--> argocd admin initial-password -n argocd

--> curl -sSL -o aargocd-linux-amd64 <https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64>

--> sudo install -m 555 argocd-linux-amd64 /usr/local/bin/agocd

--> rm argocd-linux-amd64

--> argocd admin initial-password -n argocd

--> connect argo

**Administration**

**=================**

namespaces

storageclasses

network policies

ingress controller drivers

ebs drivers

efs drivers

pv

**argo-cd/namespaces/flipkart.yaml**

apiVersion: v1

kind: Namespace

metadata:

name: flipkart

labels:

project: flipkart

environment: dev

**applications/namespaces.yaml**

apiVersion: argoproj.io/v1alpha1

kind: Application

metadata:

name: namespaces

namespace: argocd # Namespace where the Argo CD Application resource lives

spec:

project: default # The Argo CD Project this application belongs to

source:

repoURL: https://github.com/daws-84s/eks-argocd.git # URL of the Git repository

targetRevision: main # The specific branch, tag, or commit to sync from

path: namespaces # The path within the repository where the application manifests are located

destination:

server: https://kubernetes.default.svc # The API server URL of the target Kubernetes cluster

namespace: argocd # The namespace in the target cluster where the application will be deployed

syncPolicy:

automated: # Enable automated synchronization

prune: true # Delete resources that are no longer in Git

selfHeal: true # Revert changes made manually in the cluster to match Git

syncOptions:

- CreateNamespace=true # Automatically create the target namespace if it doesn't exist

--> push the code

--> pull the code

--> cd eks-argocd

--> cd applications

--> kubectl apply -f namespace.yaml

**storageclasses/ebs-sc.yaml**

apiVersion: storage.k8s.io/v1

kind: StorageClass

metadata:

name: roboshop-ebs

reclaimPolicy: Retain

provisioner: ebs.csi.aws.com

volumeBindingMode: WaitForFirstConsumer # disk will be created when pod is getting created

**applications/storage-classes.yaml**

apiVersion: argoproj.io/v1alpha1

kind: Application

metadata:

name: storageclasses

namespace: argocd # Namespace where the Argo CD Application resource lives

spec:

project: default # The Argo CD Project this application belongs to

source:

repoURL: https://github.com/daws-84s/eks-argocd.git # URL of the Git repository

targetRevision: main # The specific branch, tag, or commit to sync from

path: storageclasses # The path within the repository where the application manifests are located

destination:

server: https://kubernetes.default.svc # The API server URL of the target Kubernetes cluster

namespace: argocd # The namespace in the target cluster where the application will be deployed

syncPolicy:

automated: # Enable automated synchronization

prune: true # Delete resources that are no longer in Git

selfHeal: true # Revert changes made manually in the cluster to match Git

syncOptions:

- CreateNamespace=true # Automatically create the target namespace if it doesn't exist

**app-of-apps.yaml**

apiVersion: argoproj.io/v1alpha1

kind: Application

metadata:

name: platform-root

namespace: argocd # Namespace where the Argo CD Application resource lives

finalizers:

# The default behaviour is foreground cascading deletion

- resources-finalizer.argocd.argoproj.io

spec:

project: default # The Argo CD Project this application belongs to

source:

repoURL: https://github.com/daws-84s/eks-argocd.git # URL of the Git repository

targetRevision: main # The specific branch, tag, or commit to sync from

path: applications # The path within the repository where the application manifests are located

destination:

server: https://kubernetes.default.svc # The API server URL of the target Kubernetes cluster

namespace: argocd # The namespace in the target cluster where the application will be deployed

syncPolicy:

automated: # Enable automated synchronization

prune: true # Delete resources that are no longer in Git

selfHeal: true # Revert changes made manually in the cluster to match Git

syncOptions:

- CreateNamespace=true # Automatically create the target namespace if it doesn't exist

--> cd argo-cd

--> kubectl apply -f app-of-apps.yaml

--> webhooks - add details

-->

everything in argocd is called as application.

1. source

2. destination

group of groups in ansible

[web]

frontend.daws84s.site

[user]

user.daws84s.site

[roboshop:children]

web

user

**EKS Admins**

**=============**