ARGO CD

Prerequisites Tools Installed :
- Docker
- Git
- Kubernetes

Task 1: Setup and Configuration

Create a GitRepository: Create a Github repository and host your source code in the same repository.

Github Repo Link: https://github.com/basava1139/setweight

2. Install Argo CD on Your Kubernetes Cluster:
minikube version: v1.32.0

3. Install Argo Rollouts:
kubectl-argo-rollouts: v1.6.0+7eae71e

Task 2: Creating the GitOps Pipeline:

1. Dockerize the Application:
Custom docker image code:

```
From nginx
COPY ./index.html  /usr/share/nginx/html/index.html
```

- From command , you are instructing Docker to use the official NGINX Docker image as the starting point for your custom image.
- COPY command, It copies the index.html file from the current directory of the Docker build context into the /usr/share/nginx/html/ directory within the Docker image.

index.html contains a below code

```
<!DOCTYPE html>
<html>
<head>
    <title>Green Background - Version 2</title>
    <style>
      body {
        background-color: green;
        }

        .version {
          font-size: 40px;
          text-align: center;
          margin-top: 200px;
          color: white;
          text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);
        }
    </style>
</head>
<body>
  <div class="version">Argo-Rollouts Demo by Basava</div>
  <div class="version">Version 2</div>
```

```
</body>
</html>
```

docker build -t basu1139/canarydeploy:v2 .
This command will build a Docker image using the Dockerfile found in the current directory and tag it with
basu1139/canarydeploy:v2.

DockerImage: basu1139/canarydeploy:v2

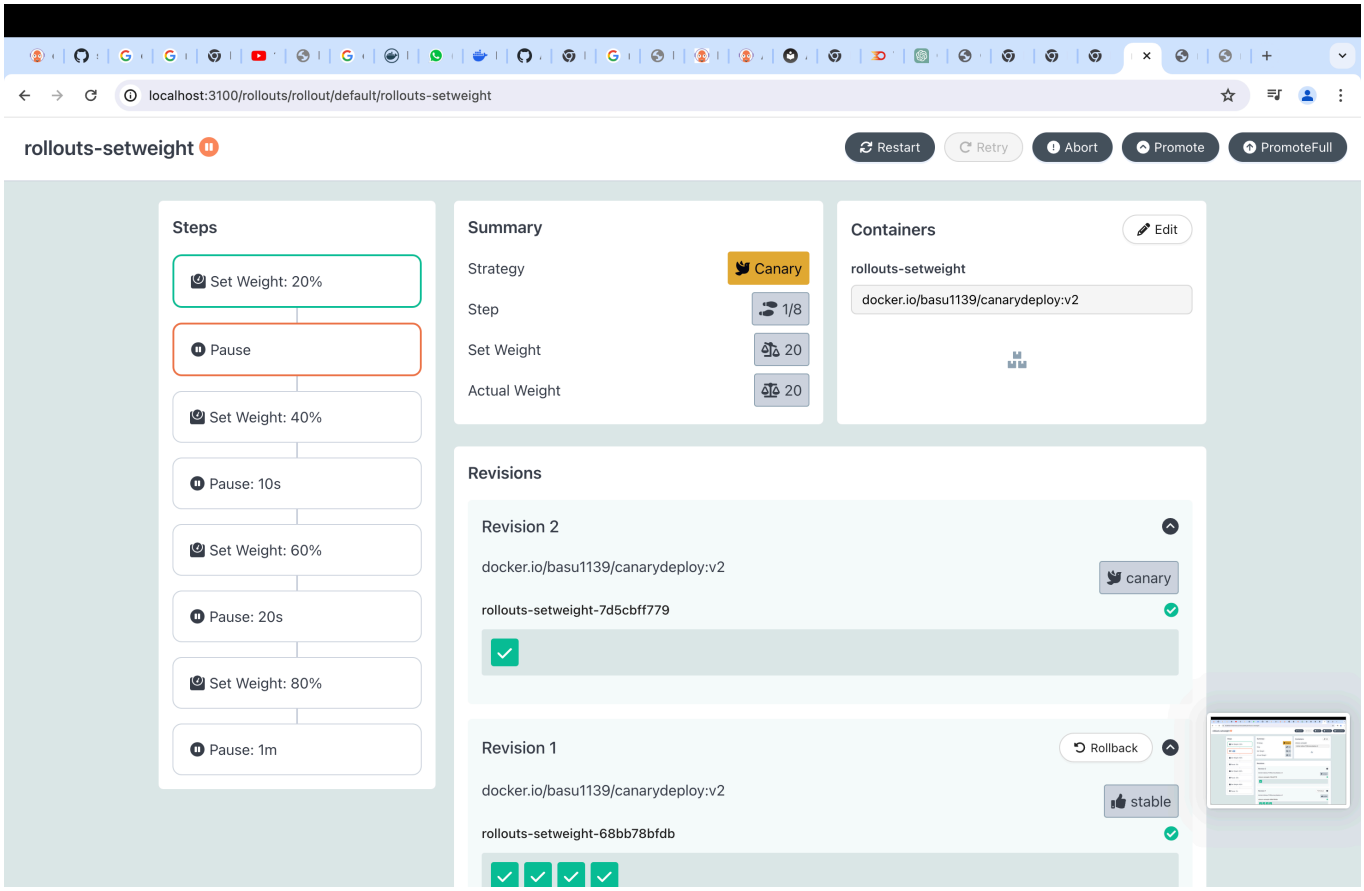
To pull an image to local machine:

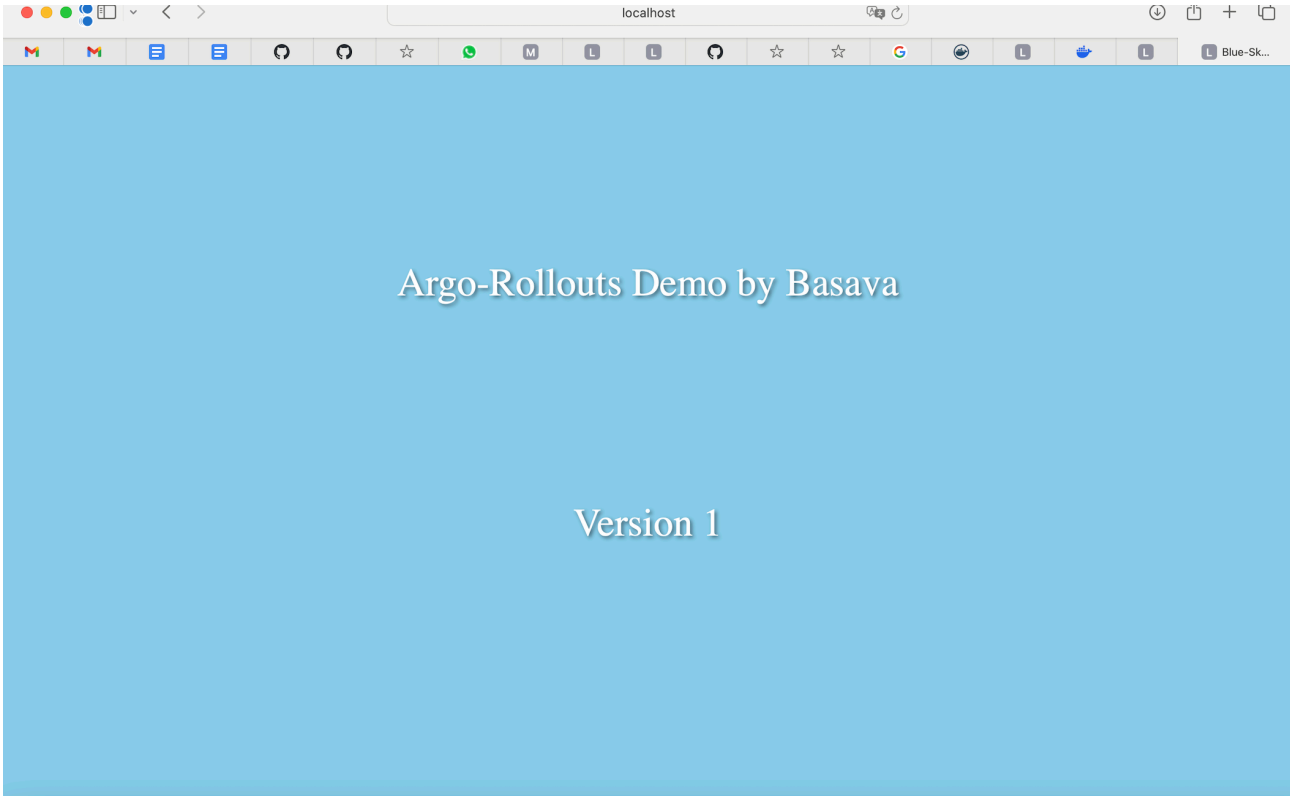
docker pull docker.io/basu1139/canarydeploy:v2

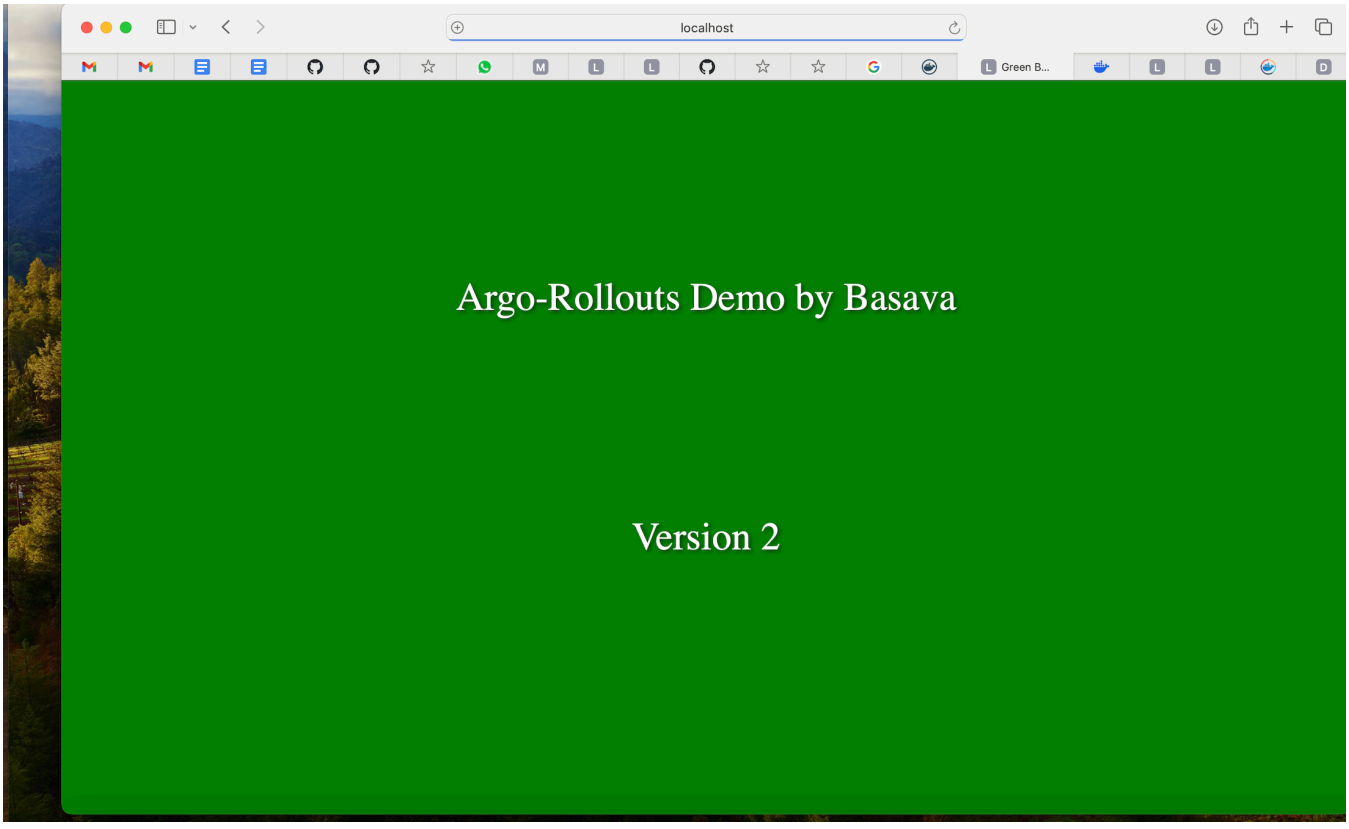
3. Deploy the Application Using Argo CD:

Install argocd by using following command:
-kubectl create namespace argocd
-kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/. argo-cd/stable/manifests/install.yaml

4. Canary deployment:

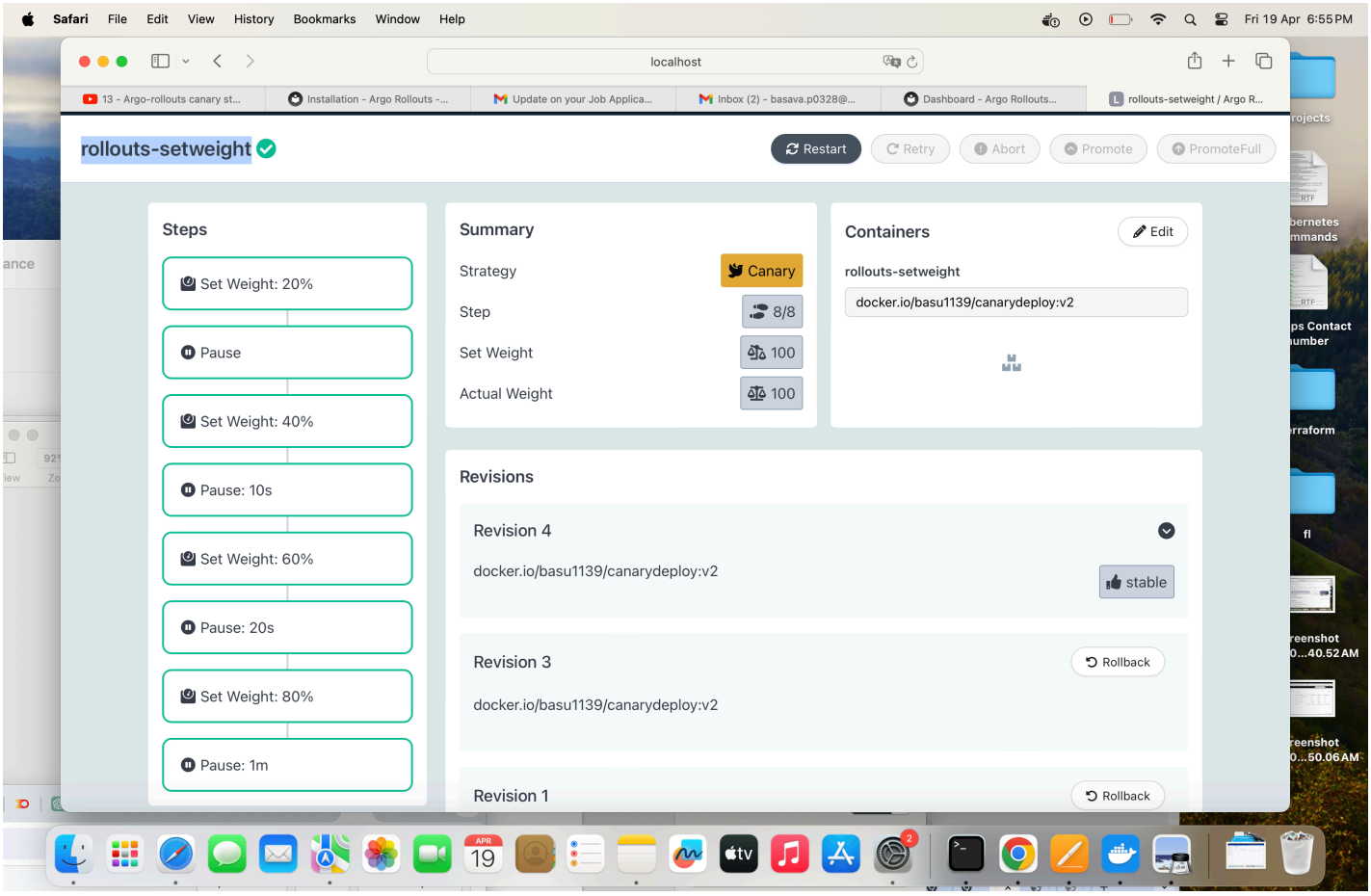



localhost
Blue-Sk...
Argo-Rollouts Demo by Basava
Version 1


localhost
Green B...
Argo-Rollouts Demo by Basava
Version 2

```
mac@MACs-MacBook-Air docker-images % Kubectl Argo Rollouts get rollouts rollouts-setweight
Name:            rollouts-setweight
Namespace:       default
Status:          ✔ Healthy
Strategy:        Canary
  Step:          8/8
  SetWeight:     100
  ActualWeight:  100
Images:          docker.io/basu1139/canarydeploy:v2 (stable)
Replicas:
  Desired:       5
  Current:       5
  Updated:       5
  Ready:         5
  Available:     5

NAME                                       KIND        STATUS        AGE    INFO
⟳ rollouts-setweight                       Rollout     ✔ Healthy     6h31m
├──# revision:4
│  └──⧉ rollouts-setweight-7d5cbff779       ReplicaSet  ✔ Healthy     6h17m  stable
│     ├──□ rollouts-setweight-7d5cbff779-ljphw  Pod     ✔ Running     5h36m  ready:1/1
│     ├──□ rollouts-setweight-7d5cbff779-w75hq  Pod     ✔ Running     5h35m  ready:1/1
│     ├──□ rollouts-setweight-7d5cbff779-796c5  Pod     ✔ Running     5h35m  ready:1/1
│     ├──□ rollouts-setweight-7d5cbff779-drhj6  Pod     ✔ Running     5h35m  ready:1/1
│     └──□ rollouts-setweight-7d5cbff779-mjcvf  Pod     ✔ Running     5h34m  ready:1/1
├──# revision:3
│  └──⧉ rollouts-setweight-9464d99f8       ReplicaSet  • ScaledDown  5h49m
└──# revision:1
   └──⧉ rollouts-setweight-68bb78bfdb       ReplicaSet  • ScaledDown  6h31m
mac@MACs-MacBook-Air docker-images %
```



- **Blue-green deployment** deploys new app versions in separate environments without disrupting the production environment. The production environment is “blue,” and the new version environment is “green.” Traffic shifts gradually from blue to green once green is stable. Issues can be rolled back to minimize impact. High availability and zero-downtime deployment are advantages.

- **Canary deployment** gradually introduces new versions or features to production.
- The new version is first deployed to a small group of users called “canary users”. The development team monitors feedback and performance indicators from canary users to evaluate the stability and reliability of the new feature.
- If no problems arise, more users are gradually added until all users use the new version.
- If a problem is found, it can be quickly rolled back or fixed to avoid negative effects on the entire user group. A canary deployment quickly identifies problems and adjusts to a small impact area.

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Rollout
metadata:
  name: example-rollout
spec:
  replicas: 10
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.15.4
        ports:
        - containerPort: 80
  minReadySeconds: 30
  revisionHistoryLimit: 3
  strategy:
    canary: #Indicates that the rollout should use the Canary strategy
      maxSurge: "25%"
      maxUnavailable: 0
      steps:
      - setWeight: 10
      - pause:
          duration: 1h # 1 hour
      - setWeight: 20
      - pause: {} # pause indefinitely
```

SetWeight field dictates the percentage of traffic that should be sent to the canary

pause struct instructs the rollout to pause.

Pause Duration

Pause duration can be specified with an optional time unit suffix. Valid time units are "s", "m", "h". Defaults to "s" if not specified.

```
spec:
  strategy:
    canary:
      steps:
        - pause: { duration: 10 }  # 10 seconds
        - pause: { duration: 10s } # 10 seconds
        - pause: { duration: 10m } # 10 minutes
        - pause: { duration: 10h } # 10 hours
        - pause: {}                # pause indefinitely
```

If no duration is specified for a pause step, the rollout will be paused indefinitely

promote to the next step

kubectl argo rollouts promote <rollout>