Ship Safely on EKS : RollingUpdate, Recreate, Blue/Green & Canary deployments (Hands-On Lab)

Objective: The goal of this project is to set up a small Kubernetes stack on Amazon EKS and then demonstrate the four common deployment strategies—**Rolling Update, Recreate, Blue/Green, and Canary.**

Table of Contents

1) Why deployment strategies matter.

Ship Safely on EKS : RollingUpdate, Recreate, Blue/Green & Canary deployments (Hands-On Lab)

What: Deployment strategies are techniques for shipping new versions of your app with the right balance of safety, speed, and cost.

Why: In production, you need to reduce risk (no/low downtime, fast rollback) while still moving quickly. Knowing these strategies is a must-have skill for platform, DevOps, and SRE roles.

You'll practice:

- **RollingUpdate**: Gradually replace pods; zero/minimal downtime
- **Recreate**: Stop old pods first, then start new (simple, brief downtime)
- **Blue/Green**: Two environments; instant cutover and rollback; safe preview.
- **Canary**: Gradual promotion with pauses (replica weighted in this lab).

2) Prerequisites

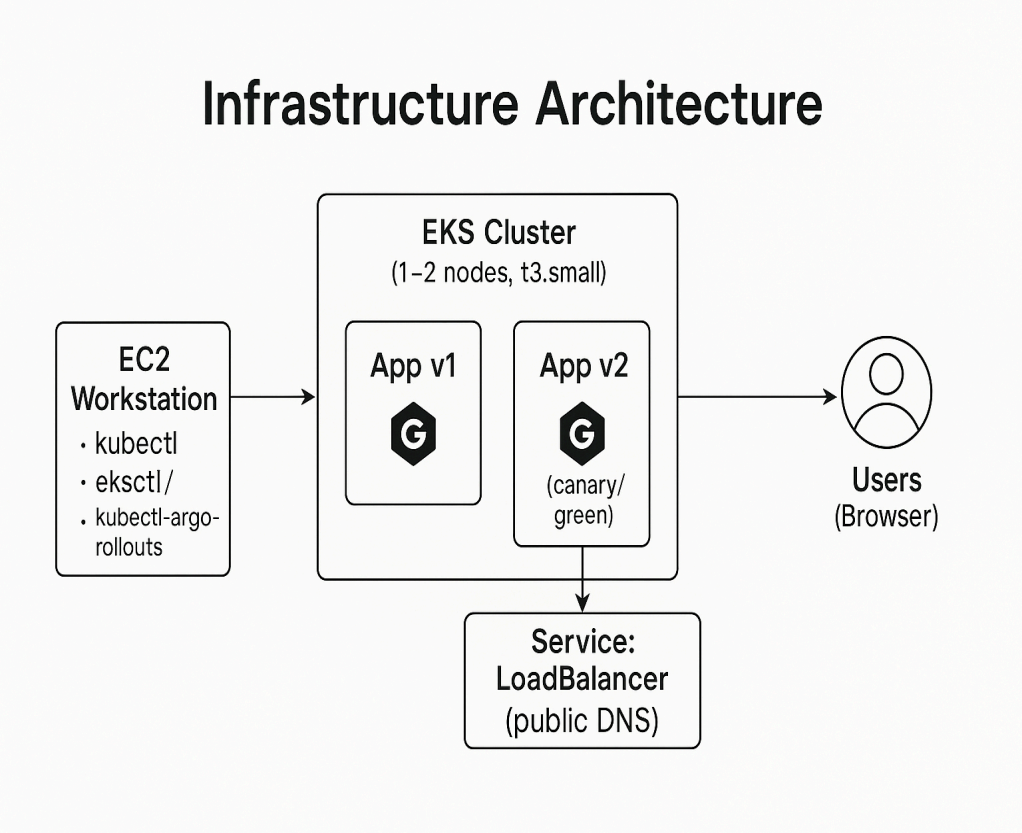
- An AWS account with permission to create EKS, VPC, EC2, ELB, IAM (Admin for labs is fine).
- Basic terminal functionality
- Optional: a GitHub repo to host your manifests

3) Architecture at a glance

What: Cost-aware EKS cluster with a public load balancer via NGINX Ingress

Why: This mirrors a realistic production entry path while staying simple enough to learn quickly.

Ship Safely on EKS : RollingUpdate, Recreate, Blue/Green & Canary deployments (Hands-On Lab)



4) **Set up guardrails (budget & workspace)**

   What: Create a small monthly budget + alerts in AWS billing
   Why: Labs can leak cost; alerts keep surprises away.

Launch EC2 instance with t3.micro AMI

- Name: eks-workstation
- AMI Amazon Linux 2
- Instance type: t3.small
- Key pair: Choose Proceed without key pair if you plan to use EC2 instance connect

Ship Safely on EKS : RollingUpdate, Recreate, Blue/Green & Canary deployments (Hands-On Lab)

- Network: Default VPC, public subnet, auto-assigned public IP enabled. Security group allows SSH port 22 from a single IP address.
- IAM role: Attach a role with Administrator Access (the simplest option for this lab).
- Launch instance

Prepare the workstation:

- Base updates

```
sudo yum -y update
```

- Kubectl

```
curl -sSL -o kubectl
https://storage.googleapis.com/kubernetes-release/release/$(curl -s
https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linu
x/amd64/kubectl
chmod +x kubectl && sudo mv kubectl /usr/local/bin/
```

- Eksctl (install)

```
curl -sSL
"https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_Linux_am
d64.tar.gz" -o eksctl.tgz
tar xzf eksctl.tgz && rm eksctl.tgz
sudo mv eksctl /usr/local/bin/
```

- Helm

```
curl -sSL
https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash
```

- Verify installations

```
kubectl version --client
eksctl version
helm version
```

- Argo Rollouts kubectl plugin

```
curl -sSL -o kubectl-argo-rollouts
https://github.com/argoproj/argo-rollouts/releases/latest/download/kubectl-a
rgo-rollouts-linux-amd64
chmod +x kubectl-argo-rollouts && sudo mv kubectl-argo-rollouts
```

Ship Safely on EKS : RollingUpdate, Recreate, Blue/Green & Canary deployments (Hands-On Lab)

```
/usr/local/bin/
```

Create an EKS cluster (cost-friendly)

What: A tiny EKS cluster with a single spot node

Why: Managed control plane + spot worker keeps costs low while staying realistic.

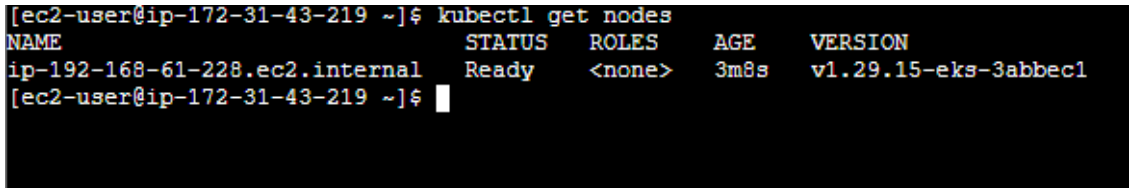
```
cat > cluster.yaml <<'YAML'
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: k8s-deploy-project
  region: us-east-1
  version: "1.29"
iam:
  withOIDC: true
vpc:
  nat:
    gateway: Single # cheaper for lab
managedNodeGroups:
  - name: ng-spot
    instanceTypes: ["t3.small","t3.micro"]
    spot: true
    desiredCapacity: 1
    minSize: 1
    maxSize: 2
    labels: { workload: "general" }
YAML

eksctl create cluster -f cluster.yaml
```

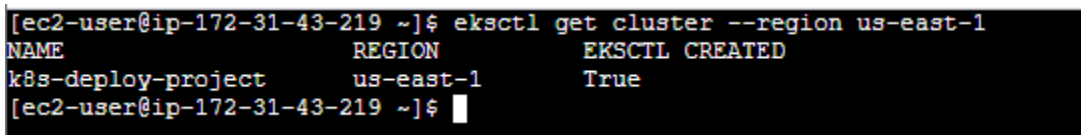
- Verify:

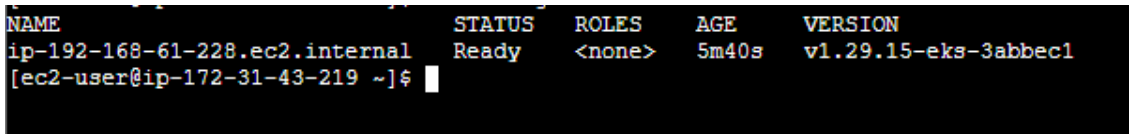
```
Eksctl get cluster –region us-east-1
kubectl get nodes
kubectl get pods -A
**Note** If the node was created successfully, you will get something like the
```

Ship Safely on EKS : RollingUpdate, Recreate, Blue/Green & Canary deployments (Hands-On Lab)

image below







Base namespace + LoadBalancer Service

What: One stable service that frontends our app; AWS will provision a public Load Balancer for it

Why: Service: LoadBalancer gives us a public URL directly

- Create namespace

```
kubectl create namespace app
```

- Stable Service (public)

```
cat > svc-stable.yaml <<'YAML'
apiVersion: v1
kind: Service
metadata:
  name: rollout-svc
  namespace: app
spec:
  type: LoadBalancer
  ports:
  - name: http
    port: 80
    targetPort: 8080
  selector:
    app: rollout-demo
YAML
```

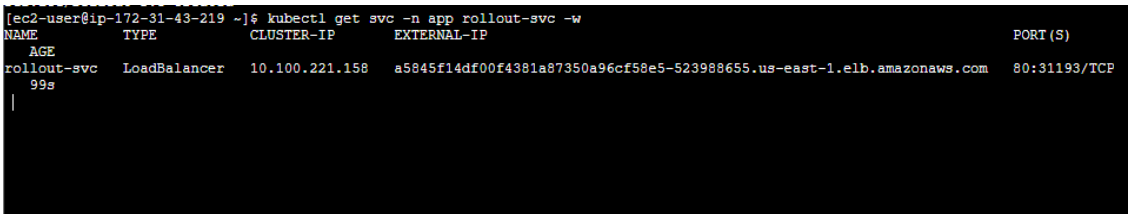
Ship Safely on EKS : RollingUpdate, Recreate, Blue/Green & Canary deployments (Hands-On Lab)

```
kubectl apply -f svc-stable.yaml
```

- Watch until EXTERNAL-IP/DNS appears

```
kubectl get svc -n app rollout-svc -w
LB=$(kubectl get svc -n app rollout-svc -o
jsonpath='{.status.loadBalancer.ingress[0].hostname}')
echo "LoadBalancer: http://$LB"
```

Note It should look like the image below, and copy the EXTERNAL-IP/DNS- you will curl this URL in every step.



-

Strategy 1 — RollingUpdate (native k8s).

What: Gradually replace old pods with new ones.

Why: Default, minimal/o downtime, easy rollback.

- # V1 = yellow

```
cat > deploy-rolling.yaml <<'YAML'
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo-rolling
  namespace: app
spec:
  replicas: 1
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 0
      maxSurge: 1
  selector:
    matchLabels: { app: rollout-demo }
  template:
    metadata: { labels: { app: rollout-demo } }
    spec:
```

Ship Safely on EKS : RollingUpdate, Recreate, Blue/Green & Canary deployments (Hands-On Lab)

```
    containers:
    - name: web
      image: public.ecr.aws/nginx/nginx:alpine   # v1
      ports: [{containerPort: 80}]
YAML

kubectl apply -f deploy-rolling.yaml
kubectl rollout status -n app deploy/demo-rolling

# test
curl -s http://$LB/ | head -n 20
```

```
[ec2-user@ip-172-31-43-219 ~]$ LB=$(kubectl get svc -n app rollout-svc -o jsonpath='{.status.loadBalancer.ingress[0].hostname}')
echo "LoadBalancer DNS: http://$LB"
curl -s http://$LB/ | head -n 20
LoadBalancer DNS: http://a5845f14df00f4381a87350a96cf58e5-523988655.us-east-1.elb.amazonaws.com
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>
```

- Upgrade (Simulate V2) = Blue

```
kubectl -n app set image deploy/demo-rolling
web=public.ecr.aws/nginx/nginx:1.27-alpine
kubectl rollout status -n app deploy/demo-rolling
```

- Rollback if needed

```
kubectl rollout undo -n app deploy/demo-rolling
```

Strategy #2 – Recreate (native k8s)

What: stop old pods, then start new

Why: simple, clean swaps; cause a brief outage; useful when versions can't coexist.

- Pause the rolling deployment to avoid label collisions

Ship Safely on EKS : RollingUpdate, Recreate, Blue/Green & Canary deployments (Hands-On Lab)

```
kubectl -n app scale deploy/demo-rolling --replicas=0

cat > deploy-recreate.yaml <<'YAML'
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo-recreate
  namespace: app
spec:
  replicas: 1
  strategy: { type: Recreate }
  selector:
    matchLabels: { app: rollout-demo }
  template:
    metadata: { labels: { app: rollout-demo } }
    spec:
      containers:
      - name: web
        image: public.ecr.aws/nginx/nginx:alpine
        ports: [{containerPort: 80}]
YAML

kubectl apply -f deploy-recreate.yaml
kubectl rollout status -n app deploy/demo-recreate
```

- Upgrade (expect a short gap)

```
kubectl -n app set image deploy/demo-recreate
web=public.ecr.aws/nginx/nginx:1.27-alpine

kubectl rollout status -n app deploy/demo-recreate
```

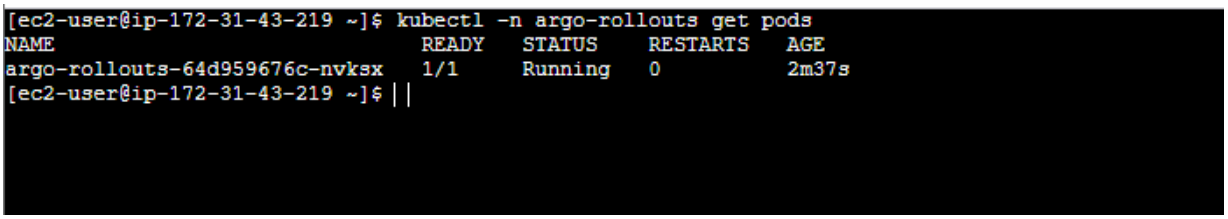
Install Argo Rollouts (enables Blue/green & Canary)

What: a controller that adds a Rollout resource with promotions/pauses/undo.

Ship Safely on EKS : RollingUpdate, Recreate, Blue/Green & Canary deployments (Hands-On Lab)

Why: production style progressive delivery

```
kubectl create namespace argo-rollouts || true
kubectl apply -n argo-rollouts -f
https://github.com/argoproj/argo-rollouts/releases/latest/download/install.yaml
Kubectl -n argo argo-rollouts get pods
**Note** It might take a bit, but it should show the pods running.
```



Strategy #3 – Blue/Green (Argo Rollouts, 2 LoadBalancers)

What: run "blue" and "green" in parallel; send prod traffic to exactly one.

Why: instant cutover and instant rollback; safe preview before promotion.

- Create a preview service (second LB)

```
kubectl -n app delete deploy/demo-recreate --ignore-not-found

cat > svc-preview.yaml <<'YAML'
apiVersion: v1
kind: Service
metadata:
  name: rollout-svc-preview
  namespace: app
spec:
  type: LoadBalancer
  ports:
  - name: http
    port: 80
    targetPort: 80
  selector:
    app: rollout-demo
YAML
```

```
kubectl apply -f svc-preview.yaml
kubectl get svc -n app rollout-svc rollout-svc-preview
```

- Create the Rollout

```
cat > rollout-bluegreen.yaml <<'YAML'
apiVersion: argoproj.io/v1alpha1
kind: Rollout
metadata:
  name: demo-bglab
  namespace: app
spec:
  replicas: 1
  revisionHistoryLimit: 2
  selector:
    matchLabels: { app: rollout-demo }
  template:
    metadata: { labels: { app: rollout-demo } }
    spec:
      containers:
      - name: web
        image: public.ecr.aws/nginx/nginx:alpine
        ports: [{containerPort: 80}]
  strategy:
    blueGreen:
      activeService: rollout-svc
      previewService: rollout-svc-preview
      autoPromotionEnabled: false

YAML

kubectl apply -f rollout-bluegreen.yaml
kubectl argo rollouts get rollout demo-bglab -n app
***Note*** The above command will give out a status update like the image below
```

Ship Safely on EKS : RollingUpdate, Recreate, Blue/Green & Canary deployments (Hands-On Lab)

```
[ec2-user@ip-172-31-43-219 ~]$ kubectl argo rollouts get rollout demo-bglab -n app
Name:            demo-bglab
Namespace:       app
Status:          ✓ Healthy
Strategy:        BlueGreen
Images:          public.ecr.aws/nginx/nginx:alpine (stable, active)
Replicas:
  Desired:       1
  Current:       1
  Updated:       1
  Ready:         1
  Available:     1

NAME                                KIND        STATUS     AGE  INFO
⟳ demo-bglab                        Rollout     ✓ Healthy  9s
└──# revision:1
   └──⧉ demo-bglab-7687d46cd9       ReplicaSet  ✓ Healthy  9s   stable,active
      └──□ demo-bglab-7687d46cd9-s6rtz  Pod     ✓ Running  9s   ready:1/1
[ec2-user@ip-172-31-43-219 ~]$
```

- Try a new version, then promote

```
# propose the new version
kubectl argo rollouts set image demo-bglab
web=public.ecr.aws/nginx/nginx:1.27-alpine -n app
kubectl argo rollouts get rollout demo-bglab -n app -w

LB_STABLE=$(kubectl get svc -n app rollout-svc -o
jsonpath='{.status.loadBalancer.ingress[0].hostname}')
LB_PREVIEW=$(kubectl get svc -n app rollout-svc-preview -o
jsonpath='{.status.loadBalancer.ingress[0].hostname}')
echo "Stable:  http://$LB_STABLE"
echo "Preview: http://$LB_PREVIEW"

# When happy, then promote
Kubectl argo rollouts promote demo-bglab -n app
```

```
[ec2-user@ip-172-31-43-219 ~]$ kubectl argo rollouts promote demo-bglab -n app
rollout 'demo-bglab' promoted
[ec2-user@ip-172-31-43-219 ~]$
```

```
# Instant rollback if needed
```

Strategy #4 — Canary (Argo Rollouts, replica-weighted)

Ship Safely on EKS : RollingUpdate, Recreate, Blue/Green & Canary deployments (Hands-On Lab)

```
cat > rollout-canary.yaml <<'YAML'
apiVersion: argoproj.io/v1alpha1
kind: Rollout
metadata:
  name: demo-canary
  namespace: app
spec:
  replicas: 2                    # need >=2 for 50%/100% steps
  revisionHistoryLimit: 2
  selector:
    matchLabels: { app: rollout-demo }
  template:
    metadata: { labels: { app: rollout-demo } }
    spec:
      containers:
      - name: web
        image: public.ecr.aws/nginx/nginx:alpine
        ports: [{containerPort: 80}]
        volumeMounts:
        - name: site
          mountPath: /usr/share/nginx/html
      volumes:
      - name: site
        configMap:
          name: site-v1
          items:
          - key: index.html
            path: index.html
  strategy:
    canary:
      steps:
      - setWeight: 50
      - pause: { duration: 30 }
      - setWeight: 100

YAML


kubectl apply -f rollout-canary.yaml
```

Ship Safely on EKS : RollingUpdate, Recreate, Blue/Green & Canary deployments (Hands-On Lab)

- Kick off a canary update

```
kubectl argo rollouts set image demo-canary
web=public.ecr.aws/nginx/nginx:1.27-alpine -n app

kubectl argo rollouts get rollout demo-canary -n app -w

kubectl argo rollouts promote demo-canary -n app --full
```

- Roll back if you want to revert

```
kubectl argo rollouts undo demo-canary -n app
```

Observability & Troubleshooting

- Inventory

```
kubectl -n app get pods -o wide
kubectl -n app get svc  -o wide
```

- Rollout status

```
kubectl argo rollouts get rollout demo-bglab  -n app
kubectl argo rollouts get rollout demo-canary -n app -w
```

Ship Safely on EKS : RollingUpdate, Recreate, Blue/Green & Canary deployments (Hands-On Lab)

```
[ec2-user@ip-172-31-43-219 ~]$ kubectl argo rollouts get rollout demo-bglab  -n app
Name:            demo-bglab
Namespace:       app
Status:          ‖ Paused
Message:         BlueGreenPause
Strategy:        BlueGreen
Images:          public.ecr.aws/nginx/nginx:1.27-alpine (stable, active)
                 public.ecr.aws/nginx/nginx:alpine (preview)
Replicas:
  Desired:       1
  Current:       2
  Updated:       1
  Ready:         1
  Available:     1

NAME                                    KIND        STATUS     AGE  INFO
⟳ demo-bglab                            Rollout     ‖ Paused   37m
├──# revision:3
│  └──⧉ demo-bglab-7687d46cd9           ReplicaSet  ✓ Healthy  37m  preview
│     └──□ demo-bglab-7687d46cd9-wstv6  Pod         ✓ Running  21m  ready:1/1
└──# revision:2
   └──⧉ demo-bglab-845dbfbd4b           ReplicaSet  ✓ Healthy  28m  stable,active
      └──□ demo-bglab-845dbfbd4b-sgfk9  Pod         ✓ Running  28m  ready:1/1
[ec2-user@ip-172-31-43-219 ~]$
```

```
Name:            demo-canary
Namespace:       app
Status:          ✖ Degraded
Message:         ProgressDeadlineExceeded: ReplicaSet "demo-canary-6c86994f96" has timed out progressing.
Strategy:        Canary
  Step:          3/3
  SetWeight:     100
  ActualWeight:  100
Images:          public.ecr.aws/nginx/nginx:alpine (stable)
Replicas:
  Desired:       2
  Current:       2
  Updated:       2
  Ready:         0
  Available:     0

NAME                                    KIND        STATUS               AGE  INFO
⟳ demo-canary                           Rollout     ✖ Degraded           20m
├──# revision:3
│  └──⧉ demo-canary-6c86994f96          ReplicaSet  ○ Progressing        20m  stable
│     ├──□ demo-canary-6c86994f96-5dgk4  Pod         ○ ContainerCreating  20m  ready:0/1
│     └──□ demo-canary-6c86994f96-61g79  Pod         ○ ContainerCreating  20m  ready:0/1
└──# revision:2
   └──⧉ demo-canary-97cf5fcff           ReplicaSet  • ScaledDown         13m
```

- Logs & events

```
kubectl -n app logs -l app=rollout-demo --tail=50
kubectl -n app get events --sort-by=.lastTimestamp | tail -30
```

Cleanup (stop meter)

Ship Safely on EKS : RollingUpdate, Recreate, Blue/Green & Canary deployments (Hands-On Lab)

kubectl delete ns app argo-rollouts --ignore-not-found
eksctl delete cluster -f cluster.yaml

```
[ec2-user@ip-172-31-43-219 ~]$ eksctl delete cluster -f cluster.yaml
2025-09-01 04:19:11 [ℹ]  deleting EKS cluster "k8s-deploy-project"
2025-09-01 04:19:11 [ℹ]  will drain 0 unmanaged nodegroup(s) in cluster "k8s-deploy-project"
2025-09-01 04:19:11 [ℹ]  starting parallel draining, max in-flight of 1
2025-09-01 04:19:11 [ℹ]  deleted 0 Fargate profile(s)
2025-09-01 04:19:12 [✓]  kubeconfig has been updated
2025-09-01 04:19:12 [ℹ]  cleaning up AWS load balancers created by Kubernetes objects of Kind Service or Ingress
2025-09-01 04:19:15 [ℹ]
4 sequential tasks: {
    2 parallel sub-tasks: {
        delete nodegroup "ng-small",
        delete nodegroup "ng-spot",
    }, delete IAM OIDC provider, delete addon IAM "eksctl-k8s-deploy-project-addon-vpc-cni", delete cluster control plane "k8s-depl
oy-project" [async]
}
2025-09-01 04:19:15 [ℹ]  will delete stack "eksctl-k8s-deploy-project-nodegroup-ng-small"
2025-09-01 04:19:15 [ℹ]  waiting for stack "eksctl-k8s-deploy-project-nodegroup-ng-small" to get deleted
2025-09-01 04:19:15 [ℹ]  will delete stack "eksctl-k8s-deploy-project-nodegroup-ng-spot"
2025-09-01 04:19:15 [ℹ]  waiting for stack "eksctl-k8s-deploy-project-nodegroup-ng-spot" to get deleted
2025-09-01 04:19:15 [ℹ]  waiting for CloudFormation stack "eksctl-k8s-deploy-project-nodegroup-ng-small"
2025-09-01 04:19:15 [ℹ]  waiting for CloudFormation stack "eksctl-k8s-deploy-project-nodegroup-ng-spot"
2025-09-01 04:19:45 [ℹ]  waiting for CloudFormation stack "eksctl-k8s-deploy-project-nodegroup-ng-small"
2025-09-01 04:19:45 [ℹ]  waiting for CloudFormation stack "eksctl-k8s-deploy-project-nodegroup-ng-spot"
2025-09-01 04:20:43 [ℹ]  waiting for CloudFormation stack "eksctl-k8s-deploy-project-nodegroup-ng-spot"
2025-09-01 04:20:43 [ℹ]  waiting for CloudFormation stack "eksctl-k8s-deploy-project-nodegroup-ng-small"
2025-09-01 04:21:24 [ℹ]  waiting for CloudFormation stack "eksctl-k8s-deploy-project-nodegroup-ng-small"
2025-09-01 04:21:51 [ℹ]  waiting for CloudFormation stack "eksctl-k8s-deploy-project-nodegroup-ng-spot"
2025-09-01 04:22:40 [ℹ]  waiting for CloudFormation stack "eksctl-k8s-deploy-project-nodegroup-ng-small"
2025-09-01 04:23:17 [ℹ]  waiting for CloudFormation stack "eksctl-k8s-deploy-project-nodegroup-ng-spot"
2025-09-01 04:24:25 [ℹ]  waiting for CloudFormation stack "eksctl-k8s-deploy-project-nodegroup-ng-spot"
2025-09-01 04:25:09 [ℹ]  waiting for CloudFormation stack "eksctl-k8s-deploy-project-nodegroup-ng-spot"
2025-09-01 04:26:15 [ℹ]  waiting for CloudFormation stack "eksctl-k8s-deploy-project-nodegroup-ng-spot"
2025-09-01 04:27:40 [ℹ]  waiting for CloudFormation stack "eksctl-k8s-deploy-project-nodegroup-ng-spot"
2025-09-01 04:29:01 [ℹ]  waiting for CloudFormation stack "eksctl-k8s-deploy-project-nodegroup-ng-spot"
2025-09-01 04:29:01 [ℹ]  will delete stack "eksctl-k8s-deploy-project-addon-vpc-cni"
2025-09-01 04:29:02 [ℹ]  will delete stack "eksctl-k8s-deploy-project-cluster"
2025-09-01 04:29:02 [✓]  all cluster resources were deleted
[ec2-user@ip-172-31-43-219 ~]$
```

Conclusion: In this lab

- Built a minimal EKS cluster; exposed via Service: LoadBalancer
- Implemented **RollingUpdate**, **Recreate**, **Blue/Green**, and **Canary**.
- Executed Promotions, pauses, aborts, rollbacks; validated by curls/logs.
- Diagnosed pod density & image pull issues; added node capacity; used AWS public ECR

Ship Safely on EKS : RollingUpdate, Recreate, Blue/Green & Canary deployments (Hands-On Lab)