Ship Resilient on AWS: Multi-Region EKS + Route 53 Failover + GitHub Automation (Hands-On)

Objective (What we are building)

- Primary & DR EKS clusters (us-east-1 + us-west-2)
- A tiny flask app exposing / and /health — renders “**PRIMARY REGION**” or “**DR REGION**”.
- Route 53 failover DNS (app.<your-domain>) that prefers primary while healthy and flips to DR on outage.
- GitHub Actions for
  1. Build and deploy (image – GHCR — deploy to both clusters)
  2. Route 53 Failover (create/maintain health-checked records).
  3. DR Drill (scale primary – 0, verify DR serves, scale back).

Table of Contents

Ship Resilient on AWS: Multi-Region EKS + Route 53 Failover + GitHub Automation (Hands-On)

6. Base app & Services (K8s)

7. GitHub repo + CI/CD (Build & Deploy)

8. GitHub→AWS OIDC role (keyless) + repo variables

9. Route 53 failover automation (workflow + script)

10. DR drill (automated)

11. Observability & verification

12. Cleanup & costs

1. **Guardrails (budget & workspace)**

   What: Create a small monthly AWS Budget + alert

   Why: Labs can leak costs. Alerts stop surprises

2. **Launch a control-node EC2 (using EC2 instance connect)**

   What: A tiny admin workstation in AWS

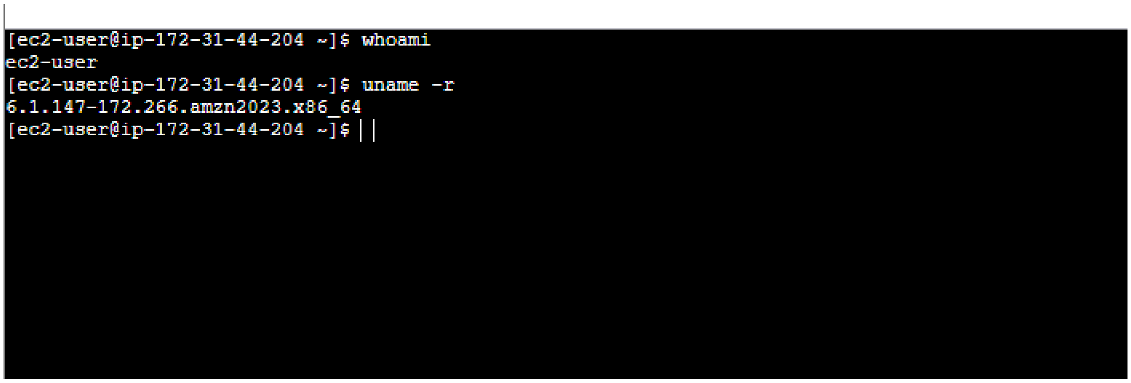
   Why: Keeps tools off your laptop and avoids key management

   - Console —EC2 launch
   - Name: control-node

Ship Resilient on AWS: Multi-Region EKS + Route 53 Failover + GitHub Automation (Hands-On)

- AMI: Amazon Linux
- Type: t3.micro
- Key pair: processed without a key pair
- Network: public subnet, Auto-assigned public IP: enabled
- Security group: SSh(22) from my IP (This is for demo)
- Launch EC2 console

```
# Check on the instance
Whoami
Uname -r
```



```
[ec2-user@ip-172-31-44-204 ~]$ whoami
ec2-user
[ec2-user@ip-172-31-44-204 ~]$ uname -r
6.1.147-172.266.amzn2023.x86_64
[ec2-user@ip-172-31-44-204 ~]$ ||
```

3. **Prepare the workstation**

What: Install the tools we will use

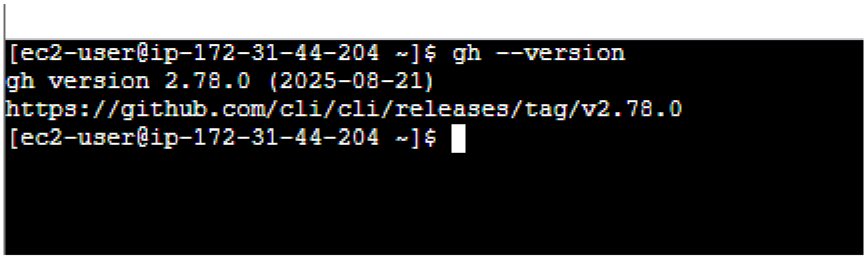
Why: Standard EKS + CI toolchain

```
sudo dnf -y update

#eksctl
curl -sLO
"https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_$(uname
-s)_amd64.tar.gz"
```

```
tar -xzf eksctl_$(uname -s)_amd64.tar.gz -C /tmp && sudo mv /tmp/eksctl
/usr/local/bin

#kubectl
curl -sLO "https://dl.k8s.io/release/$(curl -sL
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
chmod +x kubectl && sudo mv kubectl /usr/local/bin/

# Helm
curl -s https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash

# Git + GitHub CLI
sudo dnf install -y dnf-plugins-core
sudo dnf config-manager --add-repo
https://cli.github.com/packages/rpm/gh-cli.repo

sudo dnf install -y gh
```

- **Check**

```
gh --version
```



```
[ec2-user@ip-172-31-44-204 ~]$ gh --version
gh version 2.78.0 (2025-08-21)
https://github.com/cli/cli/releases/tag/v2.78.0
[ec2-user@ip-172-31-44-204 ~]$
```

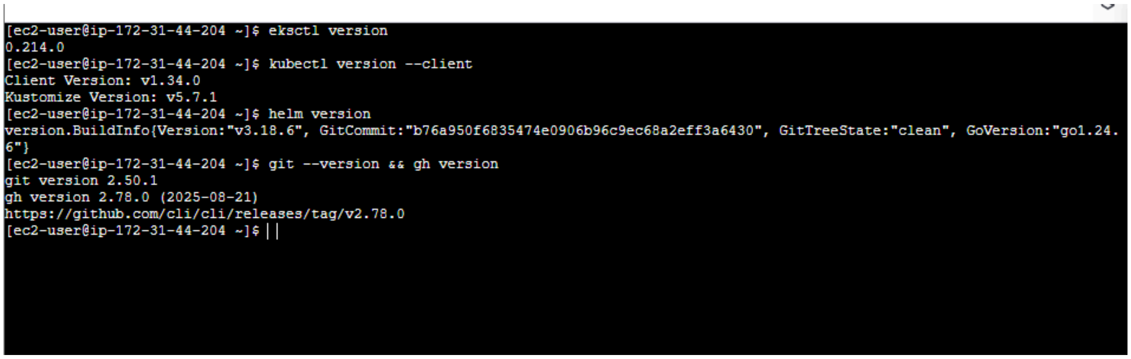
- **Then, log in and authenticate to your GitHub.**

```
gh auth login
**Note** Remember to grab your token from your GitHub account
```

Ship Resilient on AWS: Multi-Region EKS + Route 53 Failover + GitHub Automation (Hands-On)

- **Verify your installations**

```
eksctl version
kubectl version --client
helm version
git --version && gh --version
```



```
[ec2-user@ip-172-31-44-204 ~]$ eksctl version
0.214.0
[ec2-user@ip-172-31-44-204 ~]$ kubectl version --client
Client Version: v1.34.0
Kustomize Version: v5.7.1
[ec2-user@ip-172-31-44-204 ~]$ helm version
version.BuildInfo{Version:"v3.18.6", GitCommit:"b76a950f6835474e0906b96c9ec68a2eff3a6430", GitTreeState:"clean", GoVersion:"go1.24.
6"}
[ec2-user@ip-172-31-44-204 ~]$ git --version && gh version
git version 2.50.1
gh version 2.78.0 (2025-08-21)
https://github.com/cli/cli/releases/tag/v2.78.0
[ec2-user@ip-172-31-44-204 ~]$ ||
```

4. **Route 53 domain**

What: Register our domain in Route 53 and use a subdomain for fallover

Why: Keeps DNS + health checks in one place. We will use app.<domain> (CNAME)

- In console, Route 53 —- create Hosted zone — “name” . Route 53 auto-creates the public hosted zone.

```
aws route53 list-hosted-zones-by-name --dns-name lab.henrydisasterproject \
  --query 'HostedZones[0].Id' --output text | sed 's|/hostedzone/||'
```

5. **Create two EKS clusters (primary & DR)**

What: One cluster in us-east-1 (primary), one in us-west-2 (DR)

Ship Resilient on AWS: Multi-Region EKS + Route 53 Failover + GitHub Automation (Hands-On)

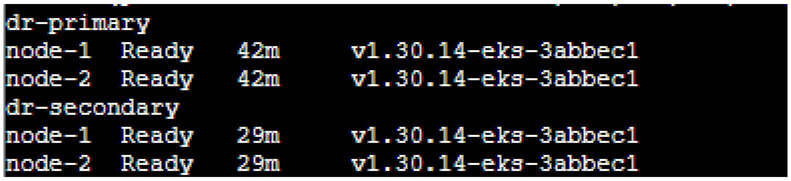
Why: Multi-region footprint is the core of DR.

```
# Primary (us-east-1)
eksctl create cluster \
  --name dr-primary \
  --version 1.30 \
  --region us-east-1 \
  --nodegroup-name ng-primary \
  --node-type t3.small \
  --nodes 2 --nodes-min 1 --nodes-max 3 \
  --managed


# DR (us-west-2)
eksctl create cluster \
  --name dr-secondary \
  --version 1.30 \
  --region us-west-2 \
  --nodegroup-name ng-dr \
  --node-type t3.small \
  --nodes 2 --nodes-min 1 --nodes-max 3 \
  --managed
```

- **Add kube contexts**

```
aws eks update-kubeconfig --region us-east-1 --name dr-primary   --alias dr-primary

aws eks update-kubeconfig --region us-west-2 --name dr-secondary --alias dr-secondary
```

Note if your commands and everything was successful, it should give a result like the image below



```
dr-primary
node-1  Ready   42m     v1.30.14-eks-3abbec1
node-2  Ready   42m     v1.30.14-eks-3abbec1
dr-secondary
node-1  Ready   29m     v1.30.14-eks-3abbec1
node-2  Ready   29m     v1.30.14-eks-3abbec1
```

6. **Base app & Services (Kubernetes)**

What: A tiny flask service with /health, exposed by **Service:LoadBalancer** in each other

Why: Gives us a visible endpoint for DNS failover + health checks.

```
# Namespaces

kubectl --context dr-primary create namespace app
kubectl --context dr-secondary create namespace app
```

- **Deployment**

```
cat > deployment.yaml <<'YAML'
apiVersion: apps/v1
kind: Deployment
metadata: {name: demo, namespace: app}
spec:
  replicas: 2
  selector: {matchLabels: {app: demo}}
  template:
    metadata: {labels: {app: demo}}
    spec:
      containers:
      - name: demo
        image: ghcr.io/henry-ibe/aws-eks-dr/demo:dev
        ports: [{containerPort: 8080}]
        readinessProbe: {httpGet: {path: /health, port: 8080}, initialDelaySeconds: 3, periodSeconds: 5}
        livenessProbe:  {httpGet: {path: /health, port: 8080}, initialDelaySeconds: 10, periodSeconds: 10}

YAML
```

- **Service**

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

YAML
```

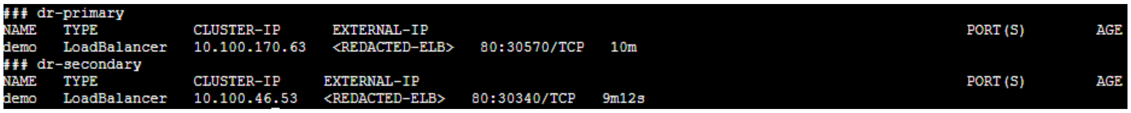
- **Apply yaml file to see the changes**

```
kubectl --context dr-primary apply -f deployment.yaml -f service.yaml
kubectl --context dr-secondary apply -f deployment.yaml -f service.yaml
```

- Check: By running these commands, you should see that the services for both primary & DR regions are working.

```
kubectl --context dr-primary   -n app get svc demo -w
kubectl --context dr-secondary -n app get svc demo -w
**Note** I redacted some info for security purposes, but you should get it when
```

these commands are run.



```
### dr-primary
NAME   TYPE           CLUSTER-IP      EXTERNAL-IP                                                          PORT(S)        AGE
demo   LoadBalancer   10.100.170.63   <REDACTED-ELB>   80:30570/TCP   10m
### dr-secondary
NAME   TYPE           CLUSTER-IP     EXTERNAL-IP                                                           PORT(S)        AGE
demo   LoadBalancer   10.100.46.53   <REDACTED-ELB>   80:30340/TCP   9m12s
```

7. **GitHub repo + CI/CD (Build & Deploy)**

What: Put app + manifests in a repo and create a pipeline that builds/pushes a container to GHCR and deploys to both clusters.

Why: Demonstrates automation beyond manual kubectl

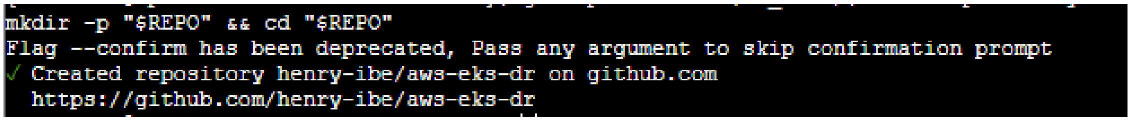
- **Log in to GitHub**

```
gh auth login
***You will need to log in using the GitHub https login with a web browser. ***
Put the code provided for authentication ***
```

- **Create repo**

```
export GH_USER=<github Username>
export REPO=aws-eks-dr # your repo name
gh repo create "$GH_USER/$REPO" --public --y
mkdir $REPO && cd $REPO
```

**If everything goes perfectly, it should be like the image below.



```
mkdir -p "$REPO" && cd "$REPO"
Flag --confirm has been deprecated, Pass any argument to skip confirmation prompt
✓ Created repository henry-ibe/aws-eks-dr on github.com
  https://github.com/henry-ibe/aws-eks-dr
```

- **App (flask + /health)**

Ship Resilient on AWS: Multi-Region EKS + Route 53 Failover + GitHub Automation (Hands-On)

```
mkdir -p app
```

```
cat > app/app.py <<'PY'
from flask import Flask, jsonify
import os
app = Flask(__name__)
ROLE = os.getenv("ROLE","unknown").upper()
REGION = os.getenv("REGION","unknown")
@app.get("/")
def index(): return f"<h1>{ROLE} REGION</h1><p>Region: {REGION}</p>"
@app.get("/health")
def health(): return jsonify({"status":"ok","role":ROLE,"region":REGION})

PY
```

```
cat > app/requirements.txt <<'REQ'
flask==3.0.3
gunicorn==22.0.0

REQ
```

```
cat > app/Dockerfile <<'DOCKER'
FROM python:3.11-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY app.py .
ENV PORT=8080
EXPOSE 8080
CMD ["gunicorn","--bind","0.0.0.0:8080","app:app"]

DOCKER
```

Ship Resilient on AWS: Multi-Region EKS + Route 53 Failover + GitHub Automation (Hands-On)

- **K8s (base + overlays)**

```
mkdir -p k8s/base k8s/overlays/primary k8s/overlays/dr
```

```
cat > k8s/base/deployment.yaml <<'YAML'
apiVersion: apps/v1
kind: Deployment
metadata: {name: demo}
spec:
  replicas: 2
  selector: {matchLabels: {app: demo}}
  template:
    metadata: {labels: {app: demo}}
    spec:
      containers:
      - name: demo
        image: ghcr.io/henry-ibe/aws-eks-dr/demo:dev
        ports: [{containerPort: 8080}]
        readinessProbe: {httpGet: {path: /health, port: 8080}, initialDelaySeconds: 3, periodSeconds: 5}
        livenessProbe:  {httpGet: {path: /health, port: 8080}, initialDelaySeconds: 10, periodSeconds: 10}

YAML
```

```
cat > k8s/base/service.yaml <<'YAML'
apiVersion: v1
kind: Service
metadata: {name: demo}
spec:
  type: LoadBalancer
  selector: {app: demo}
  ports:
    - name: http
      port: 80
      targetPort: 8080
```

```
YAML
```

```
cat > k8s/base/kustomization.yaml <<'YAML'
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
resources: [deployment.yaml, service.yaml]

YAML
```

```
cat > k8s/overlays/primary/kustomization.yaml <<'YAML'
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
resources: [../../base]
patches:
- path: deployment.patch.yaml

YAML
```

```
cat > k8s/overlays/primary/deployment.patch.yaml <<'YAML'
apiVersion: apps/v1
kind: Deployment
metadata: {name: demo}
spec:
  template:
    spec:
      containers:
      - name: demo
        env:
        - {name: ROLE, value: primary}
        - {name: REGION, value: us-east-1}
```

```
YAML
```

```
cat > k8s/overlays/dr/kustomization.yaml <<'YAML'
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
resources: [../../base]
patches:
- path: deployment.patch.yaml

YAML
```

```
cat > k8s/overlays/dr/deployment.patch.yaml <<'YAML'
apiVersion: apps/v1
kind: Deployment
metadata: {name: demo}
spec:
  template:
    spec:
      containers:
      - name: demo
        env:
        - {name: ROLE, value: dr}
        - {name: REGION, value: us-west-2}

YAML
```

- **Workflows**

Ship Resilient on AWS: Multi-Region EKS + Route 53 Failover + GitHub Automation (Hands-On)

```
mkdir -p .github/workflows
cat > .github/workflows/build-deploy.yml <<'YAML'
name: Build & Deploy
on:
  push:
    paths: ["app/**","k8s/**",".github/workflows/build-deploy.yml"]
  workflow_dispatch: {}
permissions:
  contents: write
  packages: write
  id-token: write
jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    env:
      AWS_REGION_PRIMARY: ${{ vars.PRIMARY_REGION }}
      AWS_REGION_DR: ${{ vars.DR_REGION }}
      EKS_PRIMARY_NAME: ${{ vars.EKS_PRIMARY_NAME || 'dr-primary' }}
      EKS_DR_NAME: ${{ vars.EKS_DR_NAME || 'dr-secondary' }}
      AWS_ROLE_ARN: ${{ vars.AWS_ROLE_ARN }}
    steps:
      - uses: actions/checkout@v4
      - uses: docker/login-action@v3
        with: { registry: ghcr.io, username: ${{ github.actor }}, password: ${{
secrets.GITHUB_TOKEN }} }
      - name: Build & Push
        run: |
          IMAGE=ghcr.io/${{ github.repository }}/demo
          TAG=${{ github.sha }}
          docker build -t "$IMAGE:$TAG" app
          docker push "$IMAGE:$TAG"
          echo "IMAGE=$IMAGE" >> $GITHUB_ENV
          echo "TAG=$TAG" >> $GITHUB_ENV
      - uses: aws-actions/configure-aws-credentials@v4
        with: { role-to-assume: ${{ env.AWS_ROLE_ARN }}, aws-region: ${{
env.AWS_REGION_PRIMARY }} }
```

```
      - name: Install kubectl
        run: |
          curl -sLO "https://dl.k8s.io/release/$(curl -sL https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
          chmod +x kubectl && sudo mv kubectl /usr/local/bin/kubectl
      - name: Update kubeconfigs
        run: |
          aws eks update-kubeconfig --region "$AWS_REGION_PRIMARY" --name "$EKS_PRIMARY_NAME" --alias "$EKS_PRIMARY_NAME"
          aws eks update-kubeconfig --region "$AWS_REGION_DR" --name "$EKS_DR_NAME" --alias "$EKS_DR_NAME"
      - name: Deploy to PRIMARY
        run: |
          sed -i "s#ghcr.io/henry-ibe/aws-eks-dr/demo:dev#$IMAGE:$TAG#g" k8s/base/deployment.yaml
          kubectl --context "$EKS_PRIMARY_NAME" create ns app --dry-run=client -o yaml | kubectl apply -f -
          kubectl --context "$EKS_PRIMARY_NAME" -n app apply -k k8s/overlays/primary
          kubectl --context "$EKS_PRIMARY_NAME" -n app rollout status deploy/demo --timeout=180s
      - name: Deploy to DR
        run: |
          kubectl --context "$EKS_DR_NAME" create ns app --dry-run=client -o yaml | kubectl apply -f -
          kubectl --context "$EKS_DR_NAME" -n app apply -k k8s/overlays/dr
          kubectl --context "$EKS_DR_NAME" -n app rollout status deploy/demo --timeout=180s
      - name: Show Service LBs
        run: |
          for CTX in "$EKS_PRIMARY_NAME" "$EKS_DR_NAME"; do
            echo "=== $CTX ==="
            kubectl --context "$CTX" -n app get svc demo -o wide
          done

YAML
```

```
git init && git add . && git commit -m "feat: app, k8s, CI (build & deploy)"
git branch -M main
git remote add origin https://github.com/$GH_USER/$REPO.git
git push -u origin main
```

8. GitHub — AWS OIDC role (keyless) + repo variables

What: Let Action assume an AWS role with OIDC (no long-lived keys)

Why: Secure automation best practice

1. In the console IAM — Identity providers _— Add provider:
    - Provider type : OpenID connect
    - Provider URL: https://token.actions.githubusercontent.com
    - Audience: sts.amazonaws.com
    - IAM — Roles —-- Create role (Web identity — that provider)
    - Trust policy (scope to your repo/branch)

```
{
  "Version":"2012-10-17",
  "Statement":[{
    "Effect":"Allow",

"Principal":{"Federated":"arn:aws:iam::<ACCOUNT_ID>:oidc-provider/tok
en.actions.githubusercontent.com"},
    "Action":"sts:AssumeRoleWithWebIdentity",
    "Condition":{

"StringEquals":{"token.actions.githubusercontent.com:aud":"sts.amazona
ws.com"},

"StringLike":{"token.actions.githubusercontent.com:sub":"repo:henry-ibe/
aws-eks-dr:ref:refs/heads/main"}
    }
  }]
}
```

- **Attach AdministratorAccess (demo for this lab only, not recommended in production environment)**

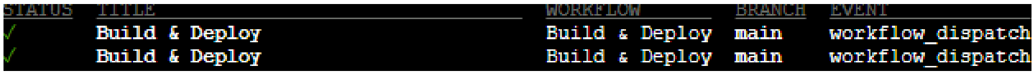
```
AWS_ROLE_ARN = arn:aws:iam::<acct>:role/<your-role>
PRIMARY_REGION = us-east-1
DR_REGION      = us-west-2
EKS_PRIMARY_NAME = dr-primary
EKS_DR_NAME      = dr-secondary
ROOT_DOMAIN     = example.com
APP_SUBDOMAIN  = app

**Note** To get your AWS_ROLE_ARN, you can run the command
aws iam get-role --role-name gh-oidc-eks-deployer --query 'Role.Arn'
--output text
```

- **Grant that role access to both EKS clusters (RBAC)**

```
eksctl create iamidentitymapping \
  --cluster dr-primary --region us-east-1 \
  --arn arn:aws:iam::<ACCOUNT_ID>:role/gh-oidc-aks-deployer \
  --group system:masters --username github-oidc || true

eksctl create iamidentitymapping \
  --cluster dr-secondary --region us-west-2 \
  --arn arn:aws:iam::<ACCOUNT_ID>:role/gh-oidc-aks-deployer \
  --group system:masters --username github-oidc || true
```

- **Check work to see if the deployment was successful**

```
gh run list -L 5
```



```
STATUS  TITLE                                          WORKFLOW        BRANCH  EVENT
✓       Build & Deploy                                 Build & Deploy  main    workflow_dispatch
✓       Build & Deploy                                 Build & Deploy  main    workflow_dispatch
```

9. **Route 53 failover – Automation (workflow + script)**

What: A workflow that reads both clusters' LB hostnames and writes failover CNAME + HTTP health checks for the primary

Why: No manual DNS edits; idempotent, repeatable, and demo-friendly.

```
# Script
mkdir -p scripts
cat > scripts/route53_failover.sh <<'BASH'
#!/usr/bin/env bash
set -euo pipefail
APP_SUBDOMAIN="${APP_SUBDOMAIN:-app}"
PRIMARY_REGION="${PRIMARY_REGION:-us-east-1}"
DR_REGION="${DR_REGION:-us-west-2}"
EKS_PRIMARY_NAME="${EKS_PRIMARY_NAME:-dr-primary}"
EKS_DR_NAME="${EKS_DR_NAME:-dr-secondary}"
if [[ -z "${ROOT_DOMAIN:-}" ]]; then echo "ROOT_DOMAIN not set"; exit 1; fi
RECORD_NAME="${APP_SUBDOMAIN}.${ROOT_DOMAIN}."
aws eks update-kubeconfig --region "$PRIMARY_REGION" --name
"$EKS_PRIMARY_NAME" --alias "$EKS_PRIMARY_NAME" >/dev/null
aws eks update-kubeconfig --region "$DR_REGION" --name "$EKS_DR_NAME" --alias
"$EKS_DR_NAME" >/dev/null
```

Ship Resilient on AWS: Multi-Region EKS + Route 53 Failover + GitHub Automation (Hands-On)

```
PRIMARY_LB=$(kubectl --context="$EKS_PRIMARY_NAME" -n app get svc demo -o
jsonpath='{.status.loadBalancer.ingress[0].hostname}')
DR_LB=$(kubectl --context="$EKS_DR_NAME" -n app get svc demo -o
jsonpath='{.status.loadBalancer.ingress[0].hostname}')
[[ -z "$PRIMARY_LB" || -z "$DR_LB" ]] && { echo "LB hostnames not ready"; exit 2; }
HZID=$(aws route53 list-hosted-zones --query
"HostedZones[?Name=='${ROOT_DOMAIN}.'].Id" --output text | sed
's|/hostedzone/||')
[[ -z "$HZID" ]] && { echo "Hosted zone not found"; exit 3; }
HCID=$(aws route53 create-health-check --caller-reference "$(date +%s)" \
 --health-check-config
"{\"FullyQualifiedDomainName\":\"${PRIMARY_LB}\",\"Port\":80,\"Type\":\"HTTP\",\"Re
sourcePath\":\"/health\",\"RequestInterval\":30,\"FailureThreshold\":3}" \
 --query 'HealthCheck.Id' --output text)
cat > /tmp/r53.json <<EOF
{"Comment":"Failover CNAME for ${RECORD_NAME}","Changes":[
{"Action":"UPSERT","ResourceRecordSet":{"Name":"${RECORD_NAME}","Type":"CNAME","
SetIdentifier":"primary-record","Failover":"PRIMARY","TTL":30,"HealthCheckId":"${HCID}","R
esourceRecords":[{"Value":"${PRIMARY_LB}"}]}},
{"Action":"UPSERT","ResourceRecordSet":{"Name":"${RECORD_NAME}","Type":"CNAME","
SetIdentifier":"dr-record","Failover":"SECONDARY","TTL":30,"ResourceRecords":[{"Value":"$
{DR_LB}"}]}}
]}
EOF
aws route53 change-resource-record-sets --hosted-zone-id "$HZID" --change-batch
file:///tmp/r53.json
echo "✅ Created/updated failover DNS for ${RECORD_NAME}"
BASH
chmod +x scripts/route53_failover.sh

# Workflow
cat > .github/workflows/route53-failover.yml <<'YAML'
name: Route53 Failover
on: { workflow_dispatch: {} }
permissions: { id-token: write, contents: read }
jobs:
 r53:
   runs-on: ubuntu-latest
   env:
     AWS_ROLE_ARN: ${{ vars.AWS_ROLE_ARN }}
     ROOT_DOMAIN: ${{ vars.ROOT_DOMAIN }}
     APP_SUBDOMAIN: ${{ vars.APP_SUBDOMAIN }}
     PRIMARY_REGION: ${{ vars.PRIMARY_REGION }}
     DR_REGION: ${{ vars.DR_REGION }}
     EKS_PRIMARY_NAME: ${{ vars.EKS_PRIMARY_NAME || 'dr-primary' }}
     EKS_DR_NAME: ${{ vars.EKS_DR_NAME || 'dr-secondary' }}
   steps:
     - uses: actions/checkout@v4
     - uses: aws-actions/configure-aws-credentials@v4
       with: { role-to-assume: ${{ env.AWS_ROLE_ARN }}, aws-region: ${{
env.PRIMARY_REGION }} }
     - name: Install kubectl
       run: |
         curl -sLO "https://dl.k8s.io/release/$(curl -sL
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
         chmod +x kubectl && sudo mv kubectl /usr/local/bin/kubectl
     - name: Create/Update Route53 failover
       run: |
         ./scripts/route53_failover.sh

YAML

git add . && git commit -m "feat: Route53 failover automation" && git push
```

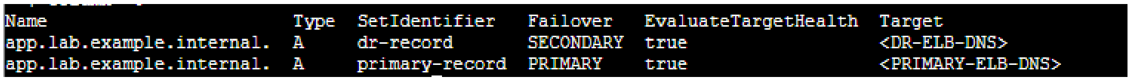
- **Confirm the two failover records exist**

```
HZ=<Your hosted zone> ** Note use your own hosted zone**
HZID=$(aws route53 list-hosted-zones-by-name --dns-name "${HZ}." \
  --query 'HostedZones[0].Id' --output text | sed 's:.*/::')
```

Ship Resilient on AWS: Multi-Region EKS + Route 53 Failover + GitHub Automation (Hands-On)

```
aws route53 list-resource-record-sets --hosted-zone-id "$HZID" \
  --query "ResourceRecordSets[?Name=='app.${HZ}.' &&
Type=='A'].[Name,SetIdentifier,Failover,AliasTarget.DNSName,AliasTarget.HostedZoneId,AliasTarget.EvaluateTargetHealth]" \
  --output table
```

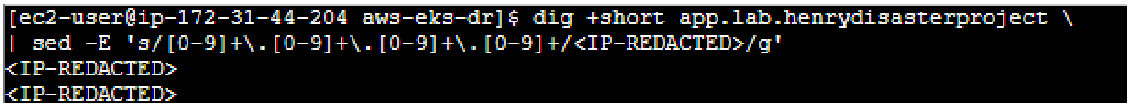
Note If everything is good, you should see similar results like the image below.



```
Name                       Type  SetIdentifier   Failover   EvaluateTargetHealth  Target
app.lab.example.internal.  A     dr-record       SECONDARY  true                  <DR-ELB-DNS>
app.lab.example.internal.  A     primary-record  PRIMARY    true                  <PRIMARY-ELB-DNS>
```

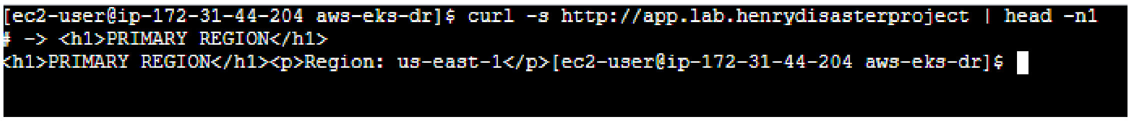
- **Check your work**

```
# Show DNS resolution
dig +short app.lab.henrydisasterproject \
| sed -E 's/[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+/<IP-REDACTED>/g'
```



```
[ec2-user@ip-172-31-44-204 aws-eks-dr]$ dig +short app.lab.henrydisasterproject \
| sed -E 's/[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+/<IP-REDACTED>/g'
<IP-REDACTED>
<IP-REDACTED>
```

```
#Show the app response
curl -s http://app.lab.henrydisasterproject | head -n1
# -> <h1>PRIMARY REGION</h1>
```



```
[ec2-user@ip-172-31-44-204 aws-eks-dr]$ curl -s http://app.lab.henrydisasterproject | head -n1
# -> <h1>PRIMARY REGION</h1>
<h1>PRIMARY REGION</h1><p>Region: us-east-1</p>[ec2-user@ip-172-31-44-204 aws-eks-dr]$
```

10. **Observability & Verification**

What: Prove real failover by taking the primary down and watching DNS flip to DR

Why: DR only counts when tested. We need to see our work play out

Ship Resilient on AWS: Multi-Region EKS + Route 53 Failover + GitHub Automation (Hands-On)

- **Inventory**

```
# from the repo root (mine is aws-eks-dr)
mkdir -p .github/workflows
```

```
cat > .github/workflows/dr-drill.yml <<'YAML'
name: DR Drill
on: { workflow_dispatch: {} }
permissions: { id-token: write, contents: read }

jobs:
  drill:
    runs-on: ubuntu-latest
    env:
      AWS_ROLE_ARN: ${{ vars.AWS_ROLE_ARN }}
      PRIMARY_REGION: ${{ vars.PRIMARY_REGION }}
      DR_REGION: ${{ vars.DR_REGION }}
      EKS_PRIMARY_NAME: ${{ vars.EKS_PRIMARY_NAME || 'dr-primary' }}
      EKS_DR_NAME: ${{ vars.EKS_DR_NAME || 'dr-secondary' }}
      ROOT_DOMAIN: ${{ vars.ROOT_DOMAIN }}
      APP_SUBDOMAIN: ${{ vars.APP_SUBDOMAIN }}
      PUBLIC_HOST: ${{ format('{0}.{1}', vars.APP_SUBDOMAIN,
vars.ROOT_DOMAIN) }}

    steps:
      - uses: actions/checkout@v4

      - uses: aws-actions/configure-aws-credentials@v4
        with:
          role-to-assume: ${{ env.AWS_ROLE_ARN }}
          aws-region: ${{ env.PRIMARY_REGION }}

      - name: Install kubectl + dnsutils
        run: |
```

Ship Resilient on AWS: Multi-Region EKS + Route 53 Failover + GitHub Automation (Hands-On)

```
          curl -sLO "https://dl.k8s.io/release/$(curl -sL https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
          chmod +x kubectl && sudo mv kubectl /usr/local/bin/kubectl
          sudo apt-get update -y && sudo apt-get install -y dnsutils

      - name: Kubeconfigs
        run: |
          aws eks update-kubeconfig --region "$PRIMARY_REGION" --name "$EKS_PRIMARY_NAME" --alias "$EKS_PRIMARY_NAME"
          aws eks update-kubeconfig --region "$DR_REGION" --name "$EKS_DR_NAME" --alias "$EKS_DR_NAME"

      - name: Discover LB hostnames & hosted zone
        id: disc
        run: |
          set -euo pipefail
          P=$(kubectl --context "$EKS_PRIMARY_NAME" -n app get svc demo -o jsonpath='{.status.loadBalancer.ingress[0].hostname}')
          D=$(kubectl --context "$EKS_DR_NAME" -n app get svc demo -o jsonpath='{.status.loadBalancer.ingress[0].hostname}')
          HZID=$(aws route53 list-hosted-zones-by-name --dns-name "${ROOT_DOMAIN}." --query 'HostedZones[0].Id' --output text | sed 's:.*/::')
          echo "primary_lb=$P" >> $GITHUB_OUTPUT
          echo "dr_lb=$D"      >> $GITHUB_OUTPUT
          echo "hzid=$HZID"   >> $GITHUB_OUTPUT
          echo "Primary:  $P"
          echo "DR:       $D"
          echo "HostedZone: $HZID"

      - name: Detect zone type
        id: zone
        run: |
          ZPRIVATE=$(aws route53 get-hosted-zone --id "${{ steps.disc.outputs.hzid }}" --query 'HostedZone.Config.PrivateZone' --output text)
          echo "private=$ZPRIVATE" >> $GITHUB_OUTPUT
          echo "Zone is private? $ZPRIVATE"

      - name: Scale PRIMARY to 0
        run: |
          kubectl --context "$EKS_PRIMARY_NAME" -n app scale deploy/demo --replicas=0

      # Public zones: use Route 53 authoritative answer to watch the flip
      - name: Show current authoritative answer (public)
        if: ${{ steps.zone.outputs.private != 'True' && steps.zone.outputs.private != 'true' }}
        run: |
          aws route53 test-dns-answer \
            --hosted-zone-id "${{ steps.disc.outputs.hzid }}" \
            --record-name "${PUBLIC_HOST}." \
            --record-type A

      - name: Wait until Route53 prefers DR (public)
        if: ${{ steps.zone.outputs.private != 'True' && steps.zone.outputs.private != 'true' }}
        run: |
          set -euo pipefail
          DR_IPS=$(dig +short "${{ steps.disc.outputs.dr_lb }}")
          for i in $(seq 1 18); do
            ANSWER=$(aws route53 test-dns-answer \
              --hosted-zone-id "${{ steps.disc.outputs.hzid }}" \
              --record-name "${PUBLIC_HOST}." \
              --record-type A --query 'RecordData' --output text || true)
            echo "Authoritative answer: $ANSWER"
            if comm -12 <(echo "$ANSWER" | tr ' ' '\n' | sort) <(echo "$DR_IPS" | sort) | grep -q .; then
              echo "✅ Route53 is answering with DR"; exit 0
            fi
            sleep 10
          done
          echo "❌ Route53 did not flip to DR in time"; exit 1
```

Ship Resilient on AWS: Multi-Region EKS + Route 53 Failover + GitHub Automation (Hands-On)

```
   # Private zones: verify from inside the VPC (pod in DR cluster)
   - name: Verify DR serves (private; in-cluster curl)
    if: ${{ steps.zone.outputs.private == 'True' || steps.zone.outputs.private
== 'true' }}
    run: |
     set -euo pipefail
     for i in $(seq 1 18); do
      kubectl --context "$EKS_DR_NAME" -n app run drill-curl --restart=Never
--image=curlimages/curl:8.10.1 --rm -i -- \
       sh -lc 'curl -s "http://'${PUBLIC_HOST}'" || true' | tee /tmp/out.txt ||
true
      if grep -q "DR REGION" /tmp/out.txt; then
       echo "✅ DR is serving via ${PUBLIC_HOST}"; exit 0
      fi
      sleep 10
     done
     echo "❌ DR not serving in time"; exit 1

   - name: Scale PRIMARY back
    if: always()
    run: |
     kubectl --context "$EKS_PRIMARY_NAME" -n app scale deploy/demo
--replicas=2
YAML
```

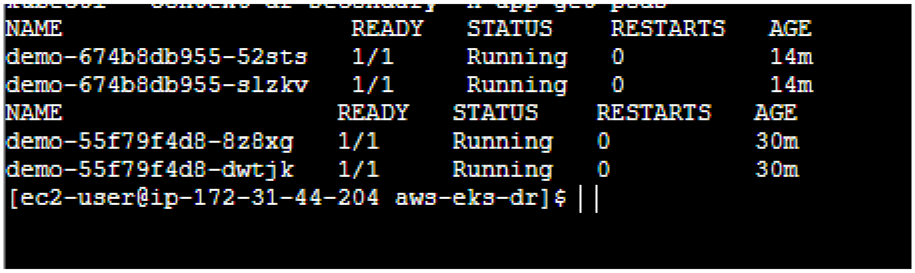
```
git add .github/workflows/dr-drill.yml
git commit -m "fix(dr): handle private hosted zone in DR drill"
git push origin main
```

```
#Kick it off and stream logs

gh workflow run ".github/workflows/dr-drill.yml" --ref main
gh run view --repo henry-ibe/aws-eks-dr --log     # use your own repo
```

12. **Verification (Manual)**

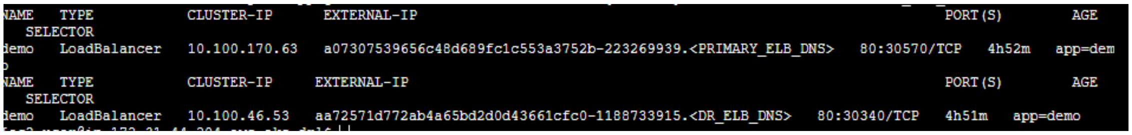
- **Inventory**

```
kubectl --context dr-primary   -n app get pods,svc -o wide
kubectl --context dr-secondary -n app get pods,svc -o wide
```



```
NAME                      READY   STATUS    RESTARTS   AGE
demo-674b8db955-52sts     1/1     Running   0          14m
demo-674b8db955-slzkv     1/1     Running   0          14m
NAME                    READY   STATUS    RESTARTS   AGE
demo-55f79f4d8-8z8xg    1/1     Running   0          30m
demo-55f79f4d8-dwtjk    1/1     Running   0          30m
[ec2-user@ip-172-31-44-204 aws-eks-dr]$ ||
```

- **Endpoints**



```
NAME   TYPE           CLUSTER-IP      EXTERNAL-IP                                                          PORT(S)        AGE
  SELECTOR
demo   LoadBalancer   10.100.170.63   a07307539656c48d689fc1c553a3752b-223269939.<PRIMARY_ELB_DNS>   80:30570/TCP   4h52m   app=dem
o
NAME   TYPE           CLUSTER-IP     EXTERNAL-IP                                                        PORT(S)        AGE
  SELECTOR
demo   LoadBalancer   10.100.46.53   aa72571d772ab4a65bd2d0d43661cfc0-1188733915.<DR_ELB_DNS>   80:30340/TCP   4h51m   app=demo
```

** Note** The ELB DNS was masked for security reasons but this shows everything is working perfectly.

- **Quick proof**

Ship Resilient on AWS: Multi-Region EKS + Route 53 Failover + GitHub Automation (Hands-On)

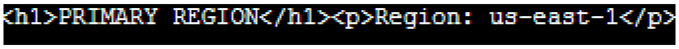
To prove failover, you must show that your public hostname now resolves/serves from the DR ELB when the primary is down

```
PRIMARY_CTX=dr-primary
DR_CTX=dr-secondary
ROOT_DOMAIN=${ROOT_DOMAIN:-lab.henrydisasterproject}
FQDN="app.${ROOT_DOMAIN}"
**Note** Adjust yours to the correct values
```

```
# discover each ELB hostname

PRIMARY_LB=$(kubectl --context "$PRIMARY_CTX" -n app get svc demo -o jsonpath='{.status.loadBalancer.ingress[0].hostname}')
DR_LB=$(kubectl --context "$DR_CTX" -n app get svc demo -o jsonpath='{.status.loadBalancer.ingress[0].hostname}')
echo "PRIMARY_LB: $PRIMARY_LB"
echo "DR_LB:      $DR_LB"
```

```
# What does DNS resolve to ** from inside the VPC**

echo "FQDN A records:"
dig +short "$FQDN"

echo "Primary ELB A records:"
dig +short "$PRIMARY_LB"

echo "DR ELB A records:"
dig +short "$DR_LB"
```

```
# – app says which region?
```



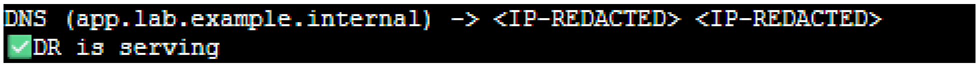
```
<h1>PRIMARY REGION</h1><p>Region: us-east-1</p>
```

Ship Resilient on AWS: Multi-Region EKS + Route 53 Failover + GitHub Automation (Hands-On)

- **Mini failover drill (Manual)**

```
# take primary down
kubectl --context "$PRIMARY_CTX" -n app scale deploy/demo --replicas=0
```

```
# Poll until DR serves (TTL is 30s in your workflow)
for i in {1..18}; do
  ans=$(dig +short "$FQDN" | xargs echo)
  echo "DNS -> $ans"
  page=$(curl -s "http://${FQDN}" || true)
  if echo "$page" | grep -q "DR REGION"; then
    echo "✅ DR is serving"
    break
  fi
  sleep 10
done
```



```
DNS (app.lab.example.internal) -> <IP-REDACTED> <IP-REDACTED>
✅DR is serving
```

Note As you can see here, our failover is working perfectly, but the IP was redacted for security reasons.

```
# Bring primary back
kubectl --context "$PRIMARY_CTX" -n app scale deploy/demo --replicas=2
```

Conclusion:

Ship Resilient on AWS: Multi-Region EKS + Route 53 Failover + GitHub Automation (Hands-On)

This project demonstrates that disaster recovery on AWS can be automated, tested, and cost-effective. By combining a multi-region EKS cluster, Route 53 health-check failover, and GitHub Actions with OIDC authentication, we built a resilient platform that continues serving traffic even when an entire region fails.

Key takeaways:

- Resilience is measurable: DR drills validated that DNS flipped to the DR site within TTL windows.
- Security is enhanced: No long-lived credentials; GitHub OIDC ensures least privilege, short-lived access.
- Operations are simplified: Kustomize overlays, GitHub workflows, and automated failover scripts make DR reproducible and maintainable.
- Scalability is built in: The approach can expand from simple demo apps to production workloads with observability, backups, and cost-optimized scale-to-Zero DR clusters.

Ultimately, this lab demonstrates how to design for failure and ship a resilient system. It’s not just about building infrastructure; it’s about proving, through automation and testing, that your workloads can withstand regional outages without customer disruption.