Kubernetes EKS on Max Vibes: Vibe-Code Your React App with Karpathy’s Magic, Then Overcharge It with GitHub Actions, IaC, GitOps, Helmfiles, Grafana and more—Local & Dev Ready! [PDF]"

Check GitHub for helpful DevOps tools:

Michael Robotics

Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats information overload by adhering to the set of principles: simplify, prioritize, and execute.

https://github.com/MichaelRobotics

Ask Personal AI Document assistant to learn interactively (FASTER)!

1 Download PDF

2 Go to website

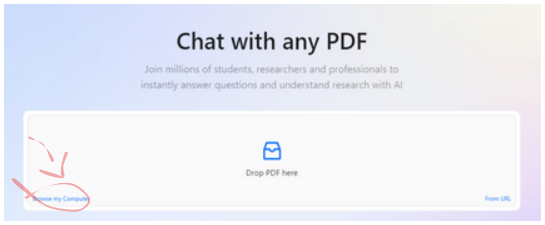
3 Browse file

4 Chat with Document

1 https://github.com/MichaelRobotics/DevOpsTools/blob/main/KubernetesVibes.pdf

Click there to go to ChatPdf website 2

3



Chat with any PDF

Join millions of students, researchers and professionals to instantly answer questions and understand research with AI

Drop PDF here

Browse my Computer

From URL

Ask questions about document! 4

Complety new to Linux and Networking?

Essential for this PDF is a thorough knowledge of networking. I highly recommend the HTB platform's networking module, which offers extensive information to help build a comprehensive understanding.

HTB - Your Cyber Performance Center

We provide a human-first platform creating and maintaining high performing cybersecurity individuals and organizations.

https://www.hackthebox.com/

What is Kubernetes?

Kubernetes is an open-source platform that automates the deployment, scaling, and management of containerized applications. It helps manage clusters of nodes running containers, ensuring efficient and reliable operation.

How Kubernetes clusters are made?

Kubernetes clusters consist of a control plane and multiple worker nodes. The control plane manages cluster operations, while worker nodes run the actual container workloads.

Why and When use Kubernetes

Kubernetes is ideal for deploying scalable, resilient, and automated containerized applications. It is used when managing multiple containers across different environments is necessary.

Example: Running a microservices-based e-commerce platform that scales up during peak hours.

System Requirements

- RAM: 2 GB per node (1 GB can work for testing but may lead to limited performance)
- 10 GB free storage
- Ubuntu

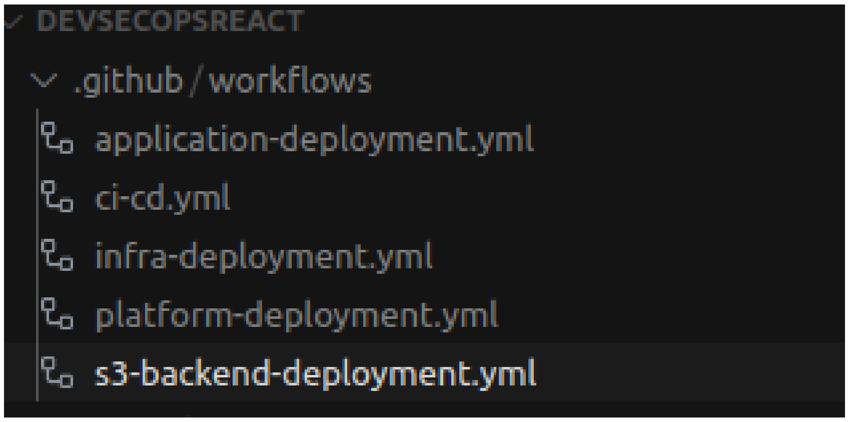
Kubernetes: Main components & packages

- **kube-apiserver:** Central management component that exposes the Kubernetes API; acts as the front-end for the cluster.
- **etcd:** Distributed key-value store for storing all cluster data, ensuring data consistency across nodes.
- **kube-scheduler:** Assigns pods to available nodes based on resource requirements and policies.
- **kube-controller-manager:** Manages core controllers that handle various functions like node status, replication, and endpoints.
- **kubelet:** Agent that runs on each node, responsible for managing pods and their containers.
- **kube-proxy:** Manages networking on each node, ensuring communication between pods and services within the cluster.

Kubernetes On Vibes: Dev/Prod Environment Setup

1) Project Intro

Idea is to setup every part of Deployment of your Product with GitHub Actions. You know - No matter what change you want to apply to your Dev/Prod environment you run github actions pipelines:



```
DEVSECOPSREACT
  .github / workflows
    application-deployment.yml
    ci-cd.yml
    infra-deployment.yml
    platform-deployment.yml
    s3-backend-deployment.yml
```

s3-backend-deployment.yaml & infra-deployment.yml - Mainly Infra teams

platform-deployment.tml - Mainly platform teams

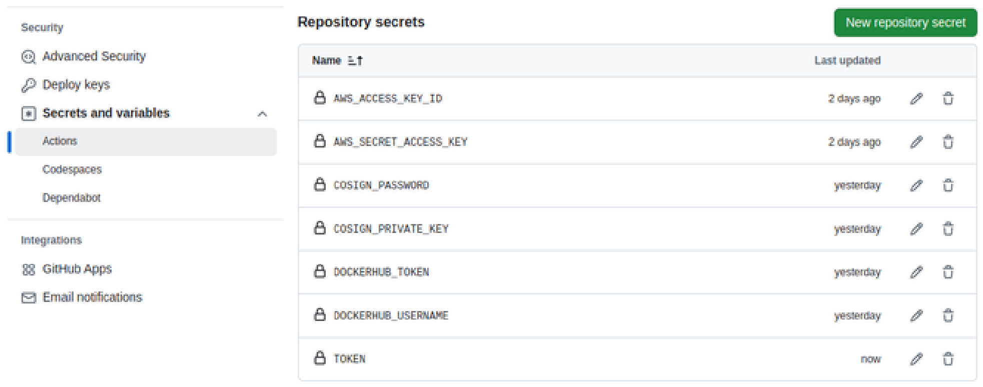
application-deployment.yml - Mainly DevOps&SRE teams

ci-cd.yml - Mainly Dev teams

The setup aligns with best practices for modular IaC and DevSecOps, combining CI/CD with GitOps, Helmfiles for tool management, and Kustomize for local development. To deploy, set up GitHub Secrets, link to Docker or Harbor, and create Cosign keys. While SBOM for compliance and Cosign for zero-trust may seem excessive for small projects, they're valuable additions to your CI for long-term, product-focused teams.

2) Setup Github secrets for Github Actions and cosign keys

You will do there most of your work. Go to your **repository->Settings** and do the work.



Repository secrets

Name	Last updated
AWS_ACCESS_KEY_ID	2 days ago
AWS_SECRET_ACCESS_KEY	2 days ago
COSIGN_PASSWORD	yesterday
COSIGN_PRIVATE_KEY	yesterday
DOCKERHUB_TOKEN	yesterday
DOCKERHUB_USERNAME	yesterday
TOKEN	now

AWS_ACCES_KEY_ID → Go to IAM in your AWS account. Generate credentials and paste there

AWS_SECRET_ACCESS_KEY → Go to IAM in your AWS account. Generate credentials and paste there

DOCKERHUB_TOKEN → Go to your Docker Hub and generate PAT(Personal access token). Paste there

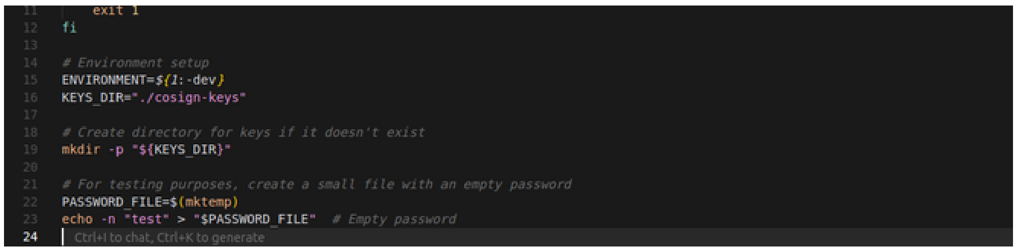
DOCKERHUB_USERNAME → Its just your DockerHub username (mine: robclusterdev)

TOKEN → Go to your GitHub account and generate PAT. Paste there

Then clone repository and create cosign keys

```
git clone https://github.com/MichaelRobotics/DevSecOpsReact.git
cd scripts
```

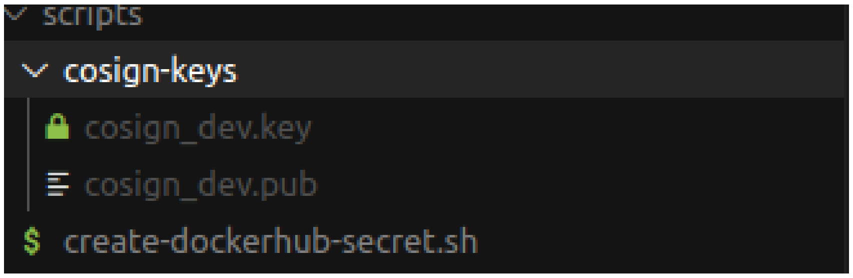
in generate_cosign_keys.sh change password "test" to something else. Will be used to cosing key decryption



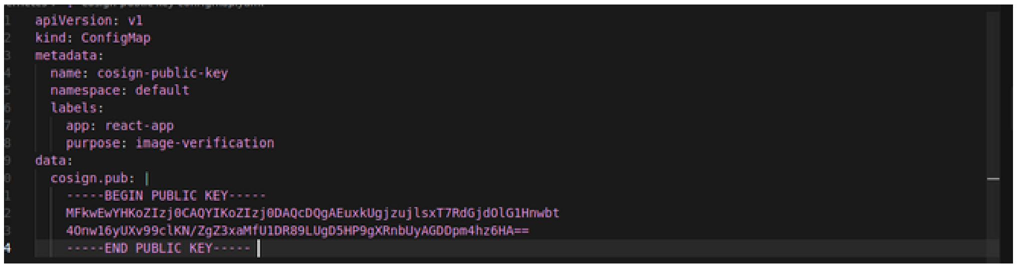
```
    exit 1
fi

# Environment setup
ENVIRONMENT=${1:-dev}
KEYS_DIR="./cosign-keys"

# Create directory for keys if it doesn't exist
mkdir -p "${KEYS_DIR}"

# For testing purposes, create a small file with an empty password
PASSWORD_FILE=$(mktemp)
echo -n "test" > "$PASSWORD_FILE"  # Empty password
```

Save and generate keys

```
./generate_cosign_keys.sh
```

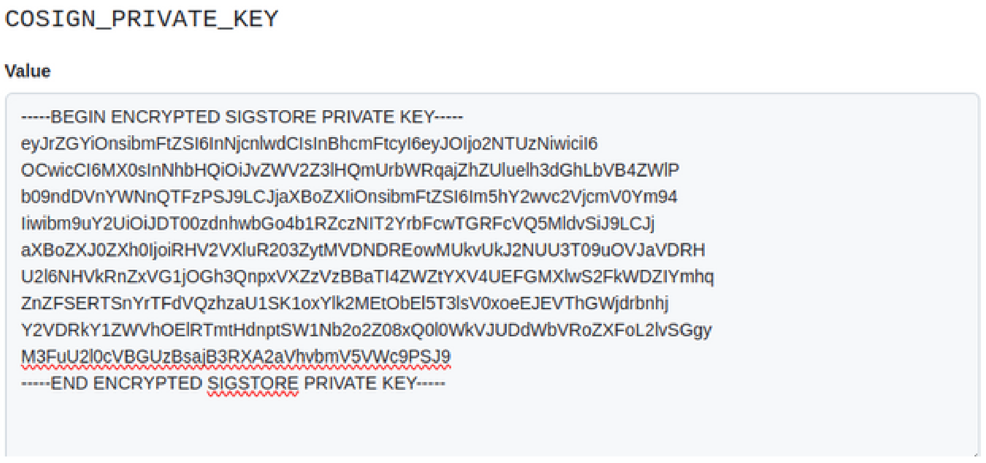


The script has generated a cosign-keys directory containing your public and private Cosign keys. Next, navigate to **kubernetes/cosign-public-key-configmap.yaml**, paste your public key there, and the deployment will use it to verify that the pulled images are signed with the matching key.



```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cosign-public-key
  namespace: default
  labels:
    app: react-app
    purpose: image-verification
data:
  cosign.pub: |
    -----BEGIN PUBLIC KEY-----
    MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEuxkUgjzujlsxT7RdGjdOlG1Hnwbt
    4Onw16yUXv99clKN/ZgZ3xaMfU1DR89LUgD5HP9gXRnbUyAGDDpm4hz6HA==
    -----END PUBLIC KEY-----
```

Write your private key into COSIGN_PRIVATE_KEY

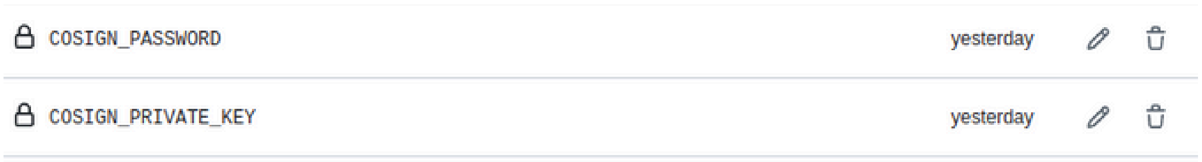


COSIGN_PRIVATE_KEY

Value

```
-----BEGIN ENCRYPTED SIGSTORE PRIVATE KEY-----
eyJrZGYiOnsibmFtZSI6InNjcnlwdCIsInBhcmFtcyI6eyJOIjo2NTUzNiwicil6
OCwicCI6MX0sInNhbHQiOiJvZWV2Z3lHQmUrbWRqajZhZUluelh3dGhLbVB4ZWlP
b09ndDVnYWNnQTFzPSJ9LCJjaXBoZXIiOnsibmFtZSI6Im5hY2wvc2VjcmV0Ym94
IiwibmFtZSI6Im5hY2wvc2VjcmV0Ym94
```

Paste into COSIGN_PASSWORD password from generate_cosign_keys.sh. CI Pipeline will use this password to decrypt encrypted cosign private key

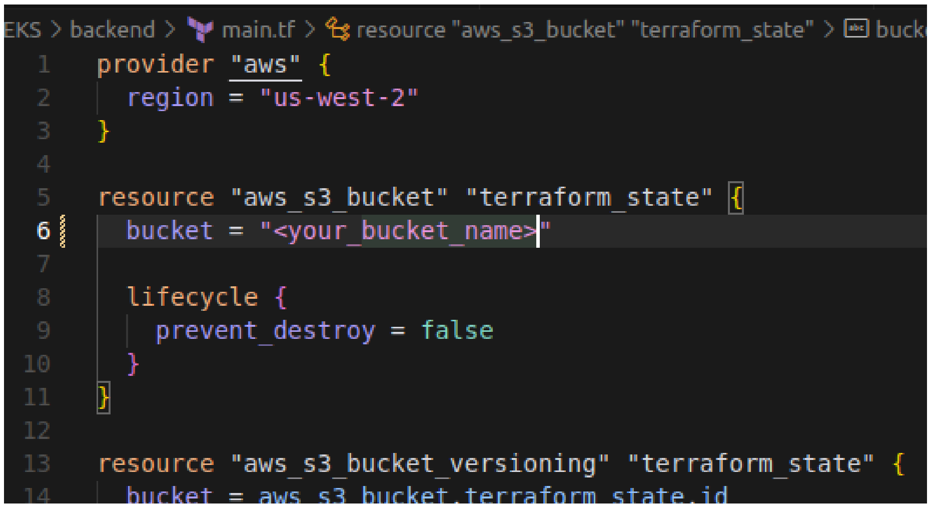


```
COSIGN_PASSWORD        yesterday
COSIGN_PRIVATE_KEY     yesterday
```

Ok now all secrets needed are in their place. Lets go further!

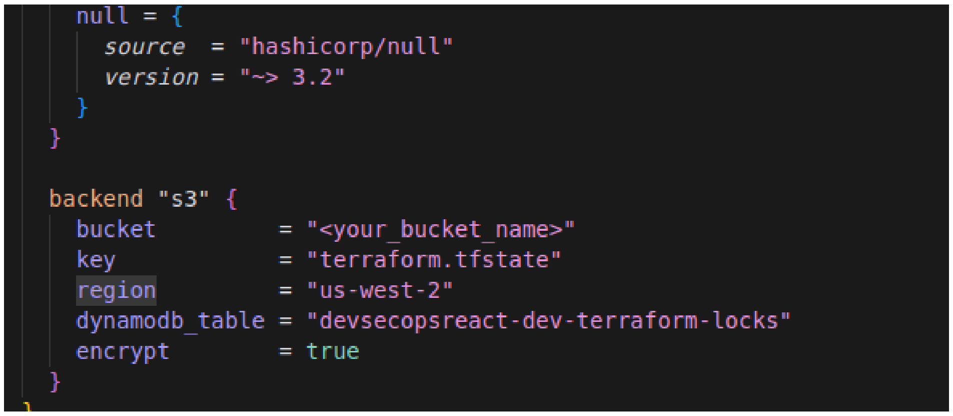
3) Setup s3 backend

Go to **EKS/backend/main.tf.** Customize your bucket name to one, that nobody uses - If somebody already uses name you want to provide, crash will happen. Bucket example: "your_bucket_name_random_5_digit_number"



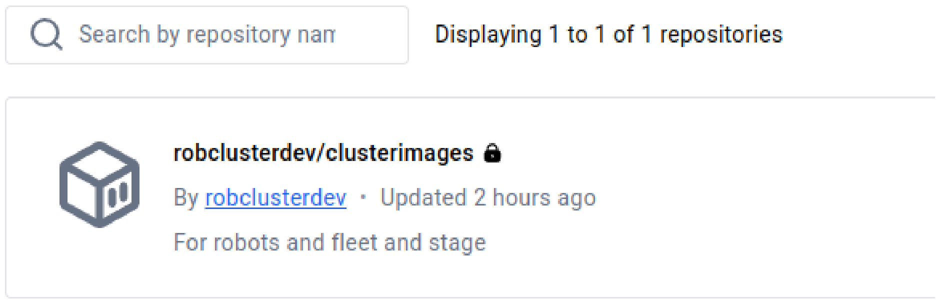
```
EKS > backend > main.tf > resource "aws_s3_bucket" "terraform_state" > buck
provider "aws" {
  region = "us-west-2"
}

resource "aws_s3_bucket" "terraform_state" {
  bucket = "<your_bucket_name>"

  lifecycle {
    prevent_destroy = false
  }
}

resource "aws_s3_bucket_versioning" "terraform_state" {
  bucket = aws_s3_bucket.terraform_state.id
```

Navigate to EKS/main.tf and setup connection to s3 backend. Paste there yout bucket name



```
    null = {
      source  = "hashicorp/null"
      version = "~> 3.2"
    }
  }

  backend "s3" {
    bucket         = "<your_bucket_name>"
    key            = "terraform.tfstate"
    region         = "us-west-2"
    dynamodb_table = "devsecopsreact-dev-terraform-locks"
    encrypt        = true
  }
```

Setup Your docker registry variables



Search by repository nam

Displaying 1 to 1 of 1 repositories

robclusterdev/clusterimages

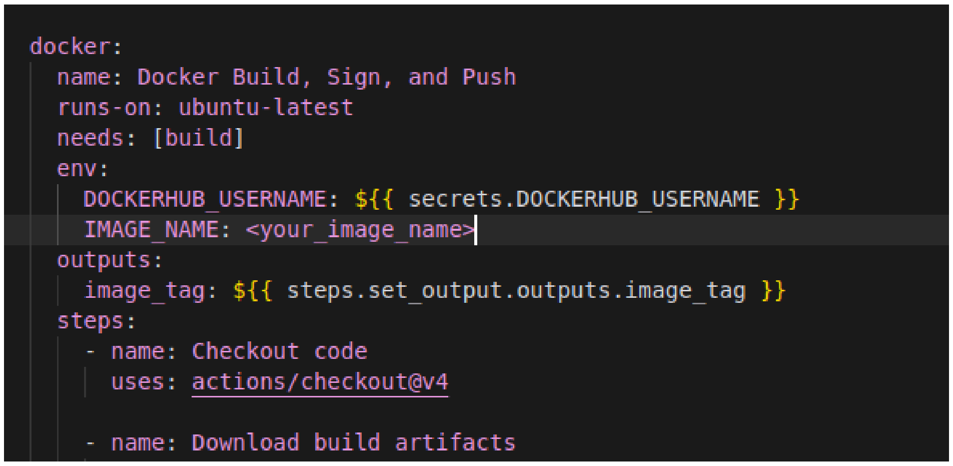
By robclusterdev • Updated 2 hours ago

For robots and fleet and stage

My repo name: robclusterdev

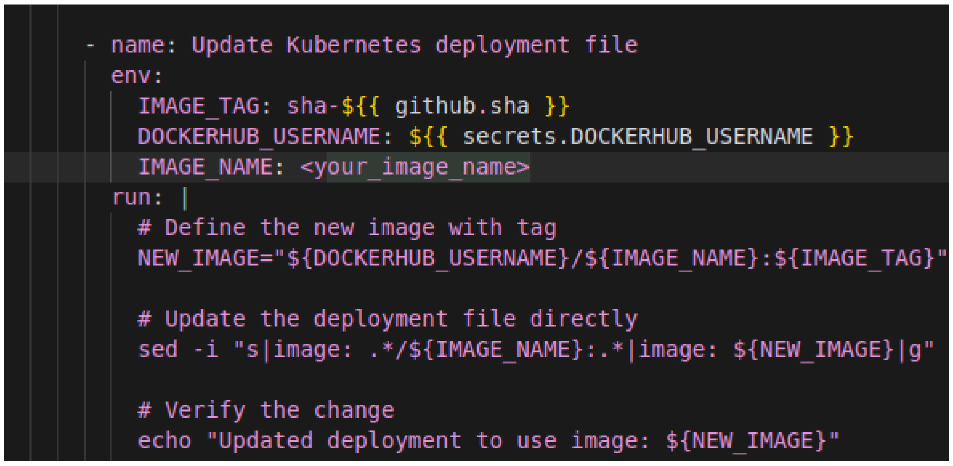
My images name: clusterimages

Got to **ci-cd.yml** and set how your images are named in your repo. In my case it is **clusterimages**. Pate your images name to <your_image_name>



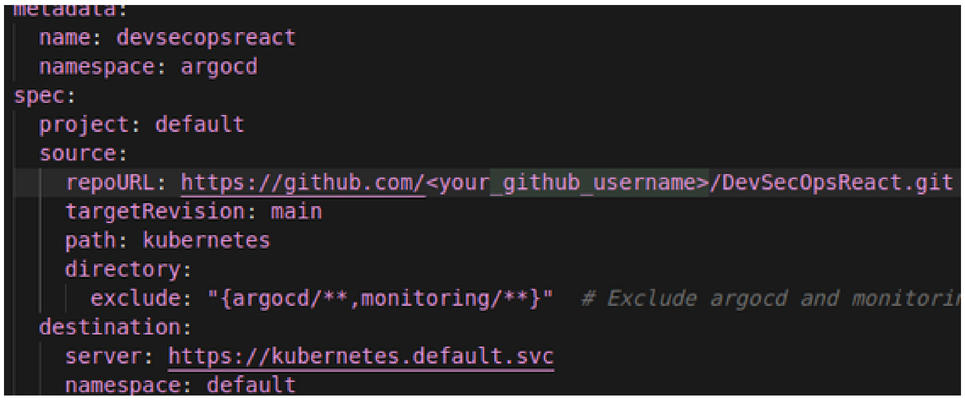
```
docker:
  name: Docker Build, Sign, and Push
  runs-on: ubuntu-latest
  needs: [build]
  env:
    DOCKERHUB_USERNAME: ${{ secrets.DOCKERHUB_USERNAME }}
    IMAGE_NAME: <your_image_name>
  outputs:
    image_tag: ${{ steps.set_output.outputs.image_tag }}
  steps:
    - name: Checkout code
      uses: actions/checkout@v4

    - name: Download build artifacts
```

and here



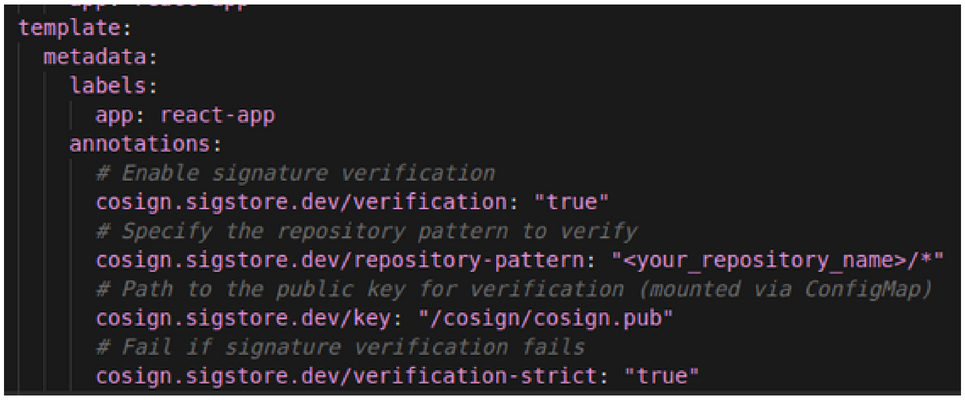
```
    - name: Update Kubernetes deployment file
      env:
        IMAGE_TAG: sha-${{ github.sha }}
        DOCKERHUB_USERNAME: ${{ secrets.DOCKERHUB_USERNAME }}
        IMAGE_NAME: <your_image_name>
      run: |
        # Define the new image with tag
        NEW_IMAGE="${DOCKERHUB_USERNAME}/${IMAGE_NAME}:${IMAGE_TAG}"

        # Update the deployment file directly
        sed -i "s|image: .*/${IMAGE_NAME}:.*|image: ${NEW_IMAGE}|g"

        # Verify the change
        echo "Updated deployment to use image: ${NEW_IMAGE}"
```

Go to **kubernetes/argocd/applications/base/react-app.yaml** and modify with your GitHub name in <your_github_username>



```
metadata:
  name: devsecopsreact
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/<your_github_username>/DevSecOpsReact.git
    targetRevision: main
    path: kubernetes
    directory:
      exclude: "{argocd/**,monitoring/**}"  # Exclude argocd and monitori
  destination:
    server: https://kubernetes.default.svc
    namespace: default
```

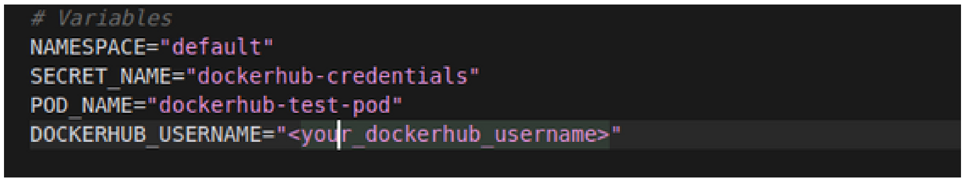
Go to **kubernetes/deployment.yaml** and modify with your DockerHub repository name in <your_repository_name>, mine is robclusterdev



```
template:
  metadata:
    labels:
      app: react-app
    annotations:
      # Enable signature verification
      cosign.sigstore.dev/verification: "true"
      # Specify the repository pattern to verify
      cosign.sigstore.dev/repository-pattern: "<your_repository_name>/*"
      # Path to the public key for verification (mounted via ConfigMap)
      cosign.sigstore.dev/key: "/cosign/cosign.pub"
      # Fail if signature verification fails
      cosign.sigstore.dev/verification-strict: "true"
```

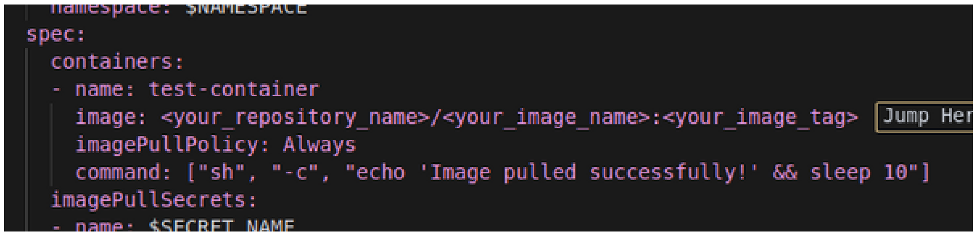
4) Testing DockerHub secret and Cosign - Not mandatory but will verify if your keys and secret work

Go to **scripts/test-dockerhub-secret.sh** and paste your dockerhub username, my is robclusterdev



```
# Variables
NAMESPACE="default"
SECRET_NAME="dockerhub-credentials"
POD_NAME="dockerhub-test-pod"
DOCKERHUB_USERNAME="<your_dockerhub_username>"
```

In the test container, specify any container you have in DockerHub. If you don't have one, you won't be able to verify whether your secret works or confirm that you can connect to DockerHub and pull images using it.

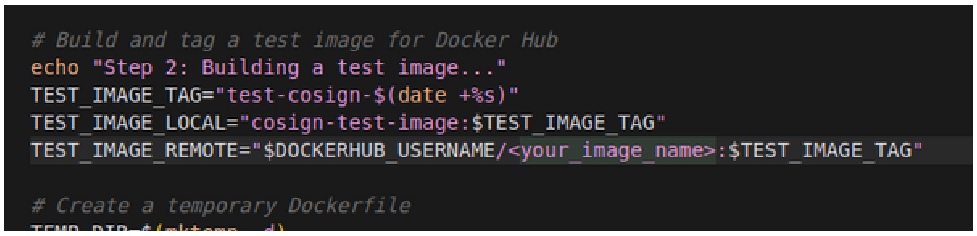


```
  namespace: $NAMESPACE
spec:
  containers:
  - name: test-container
    image: <your_repository_name>/<your_image_name>:<your_image_tag>
    imagePullPolicy: Always
    command: ["sh", "-c", "echo 'Image pulled successfully!' && sleep 10"]
  imagePullSecrets:
  - name: $SECRET_NAME
```

run tests:

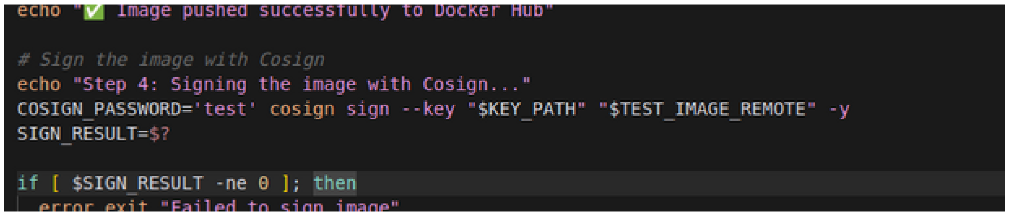
```
./test-dockerhub-secret.sh
```

Now it's time to verify that your Cosign setup is working properly. Navigate to **scripts/test-signing-direct.sh** and insert your DockerHub image name there (mine is clusterimages).



```
# Build and tag a test image for Docker Hub
echo "Step 2: Building a test image..."
TEST_IMAGE_TAG="test-cosign-$(date +%s)"
TEST_IMAGE_LOCAL="cosign-test-image:$TEST_IMAGE_TAG"
TEST_IMAGE_REMOTE="$DOCKERHUB_USERNAME/<your_image_name>:$TEST_IMAGE_TAG"

# Create a temporary Dockerfile
TEMP_DIR=$(mktemp -d)
```

in COSIGN_PASSWORD write password to decrypt cosign private key



```
echo "✅ Image pushed successfully to Docker Hub"

# Sign the image with Cosign
echo "Step 4: Signing the image with Cosign..."
COSIGN_PASSWORD='test' cosign sign --key "$KEY_PATH" "$TEST_IMAGE_REMOTE" -y
SIGN_RESULT=$?

if [ $SIGN_RESULT -ne 0 ]; then
  error_exit "Failed to sign image"
```

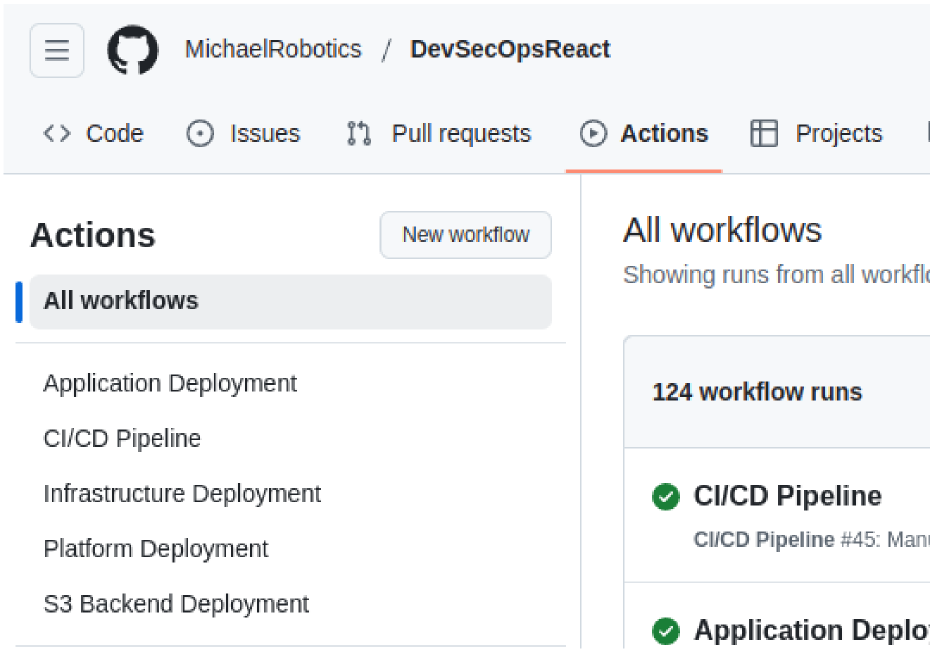
run cosign test:

```
./test-signing-direct.sh <YOUR_DOCKERHUB_PAT> dev
```

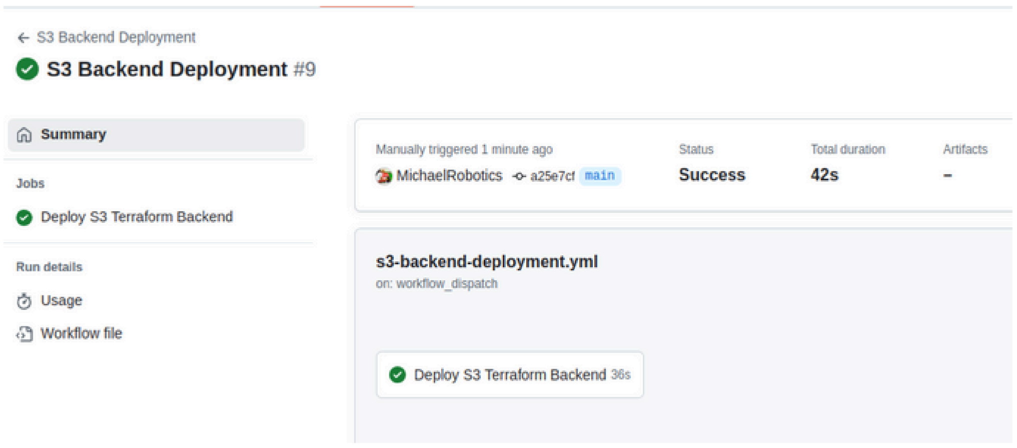
3) Deploy Product

Ok everything is setup correctly. Now lets Deploy Our Product to DEV environment or if your team is really small and you have just created your startup it can be your PROD. First, commit all of your changes:

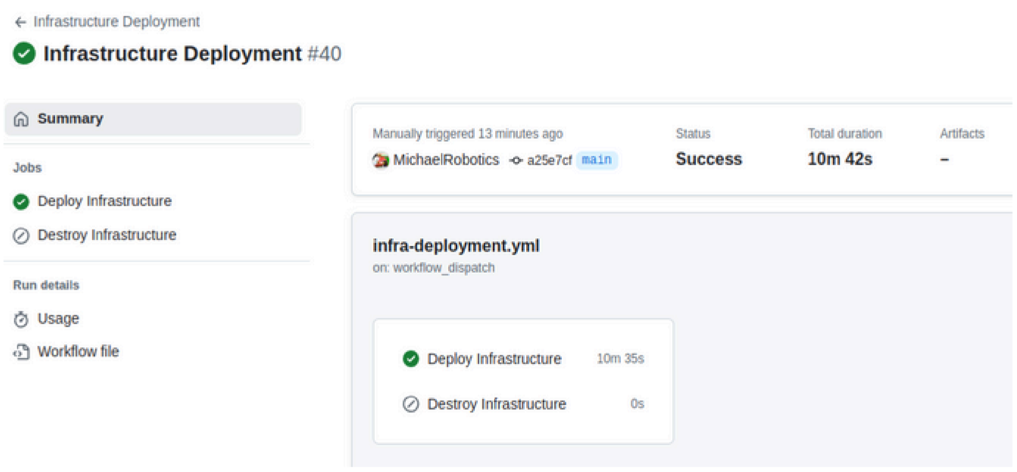
```
git add .
git commit -m “Modified DevSecOpsReact Project”
git push origin main
```



Deploy infra: S3 Backend Deployment



Deploy infra: Infrasctructure Deployment



← Infrastructure Deployment

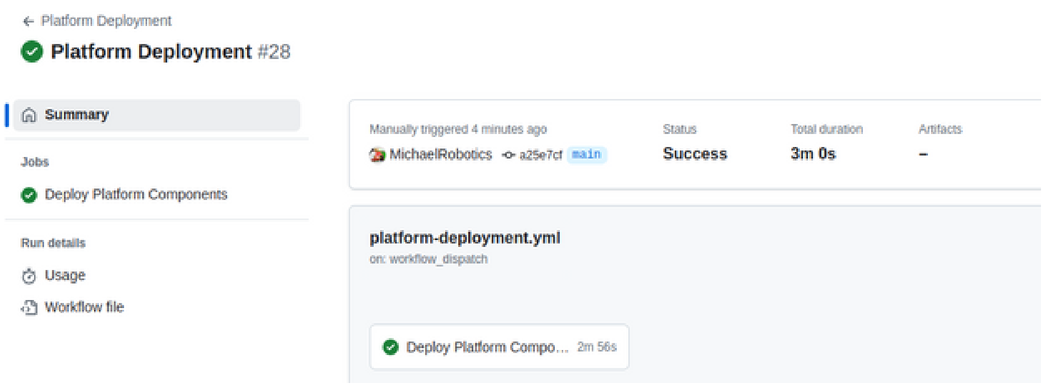
Infrastructure Deployment #40

Manually triggered 13 minutes ago	Status	Total duration	Artifacts
MichaelRobotics a25e7cf main	Success	10m 42s	–

infra-deployment.yml
on: workflow_dispatch

Deploy Infrastructure	10m 35s
Destroy Infrastructure	0s

Deploy platform: Platform Deployment



← Platform Deployment

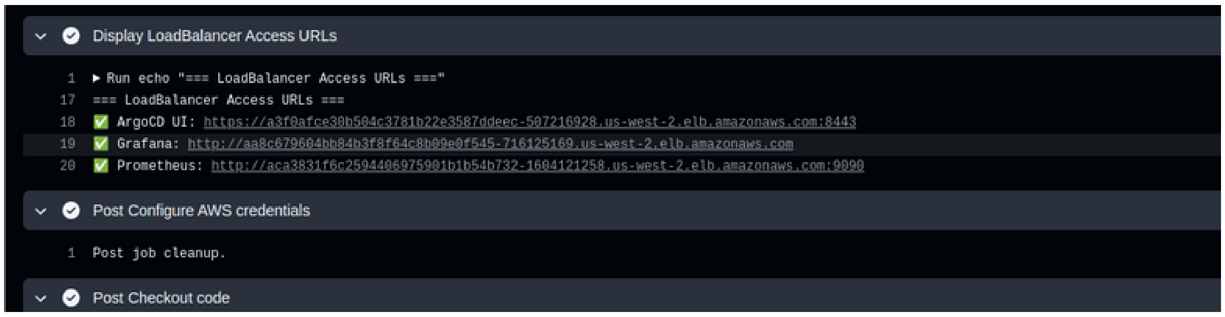
Platform Deployment #28

Manually triggered 4 minutes ago	Status	Total duration	Artifacts
MichaelRobotics a25e7cf main	Success	3m 0s	–

platform-deployment.yml
on: workflow_dispatch

Deploy Platform Compo… 2m 56s

Inside pipeline, you will have ArgoCD, Grafana and Prometheus loadbalancers DNS



Display LoadBalancer Access URLs

```
▶ Run echo "=== LoadBalancer Access URLs ==="
=== LoadBalancer Access URLs ===
✅ ArgoCD UI: https://a3f0afce30b504c3781b22e3587ddeec-507216928.us-west-2.elb.amazonaws.com:8443
✅ Grafana: http://aa8c679604bb84b3f8f64c8b09e0f545-716125169.us-west-2.elb.amazonaws.com
✅ Prometheus: http://aca3831f6c2594406975901b1b54b732-1604121258.us-west-2.elb.amazonaws.com:9090
```

Post Configure AWS credentials

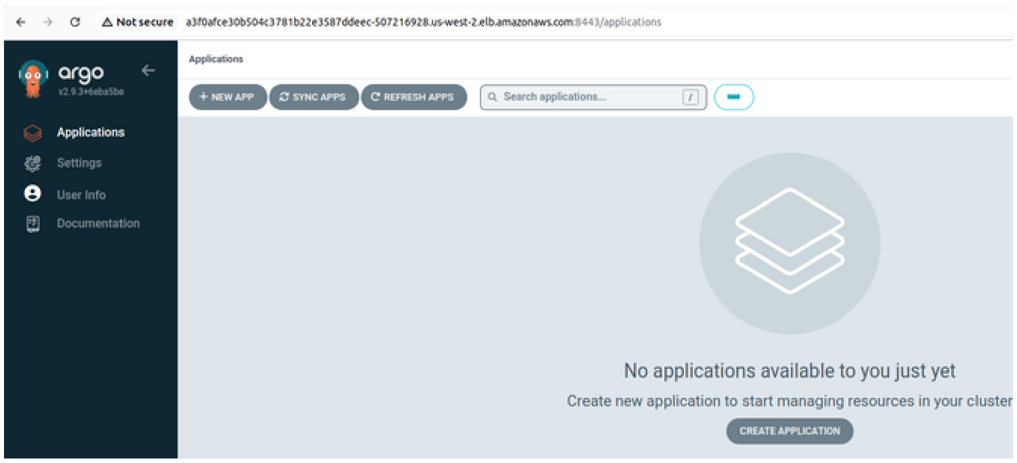
```
Post job cleanup.
```

For grafana and ArgoCD username is: admin. Passwords are stored in secrets.

Locally switch to your cluster context and check for secrets:

```
aws eks update-kubeconfig --region us-west-2 --name DevSecOpsReact
kubectl get secret monitoring-grafana -n monitoring -o jsonpath="{.data.admin-password}" | base64 -d
kubectl get secret argocd-initial-admin-secret -n argocd -o jsonpath="{.data.password}" | base64 -d
```

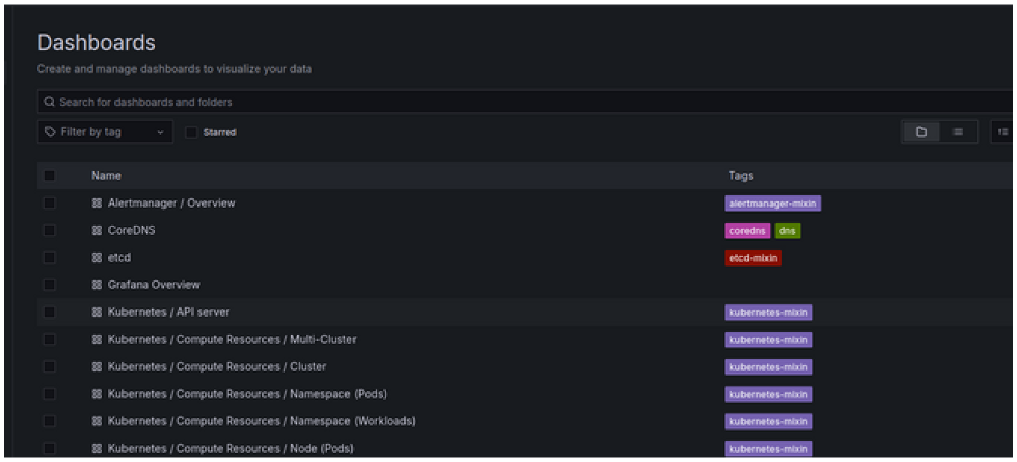
Argo:



No applications available to you just yet

Create new application to start managing resources in your cluster

Grafana with basic Dashboards:

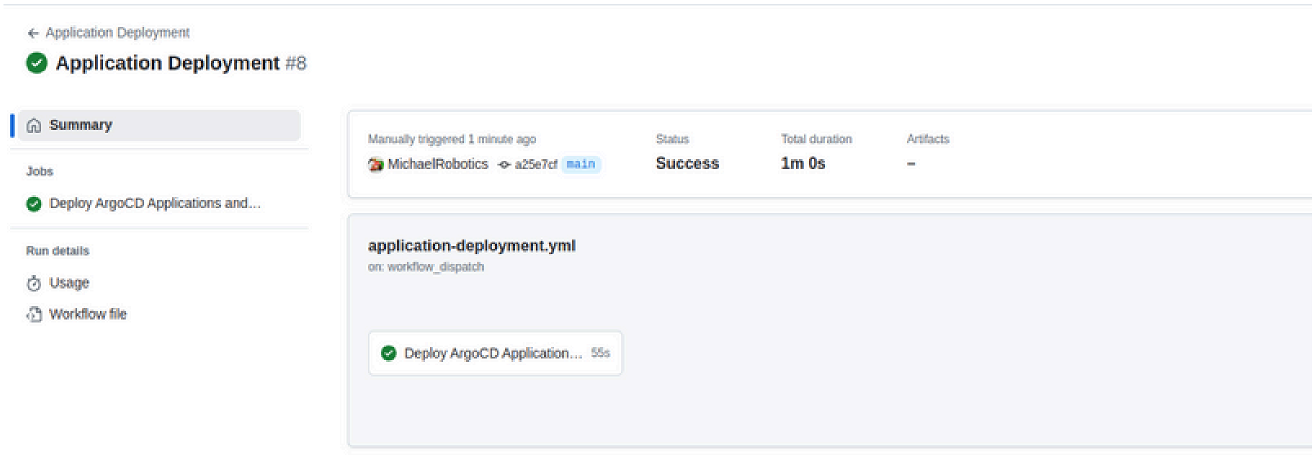


Dashboards

Create and manage dashboards to visualize your data

Name	Tags
Alertmanager / Overview	alertmanager-mixin
CoreDNS	coredns dns
etcd	etcd-mixin
Grafana Overview	
Kubernetes / API server	kubernetes-mixin
Kubernetes / Compute Resources / Multi-Cluster	kubernetes-mixin
Kubernetes / Compute Resources / Cluster	kubernetes-mixin
Kubernetes / Compute Resources / Namespace (Pods)	kubernetes-mixin
Kubernetes / Compute Resources / Namespace (Workloads)	kubernetes-mixin
Kubernetes / Compute Resources / Node (Pods)	kubernetes-mixin

Deploy Application: Application Deployment


← Application Deployment

Application Deployment #8

Summary

Jobs

Deploy ArgoCD Applications and...

Run details

Usage

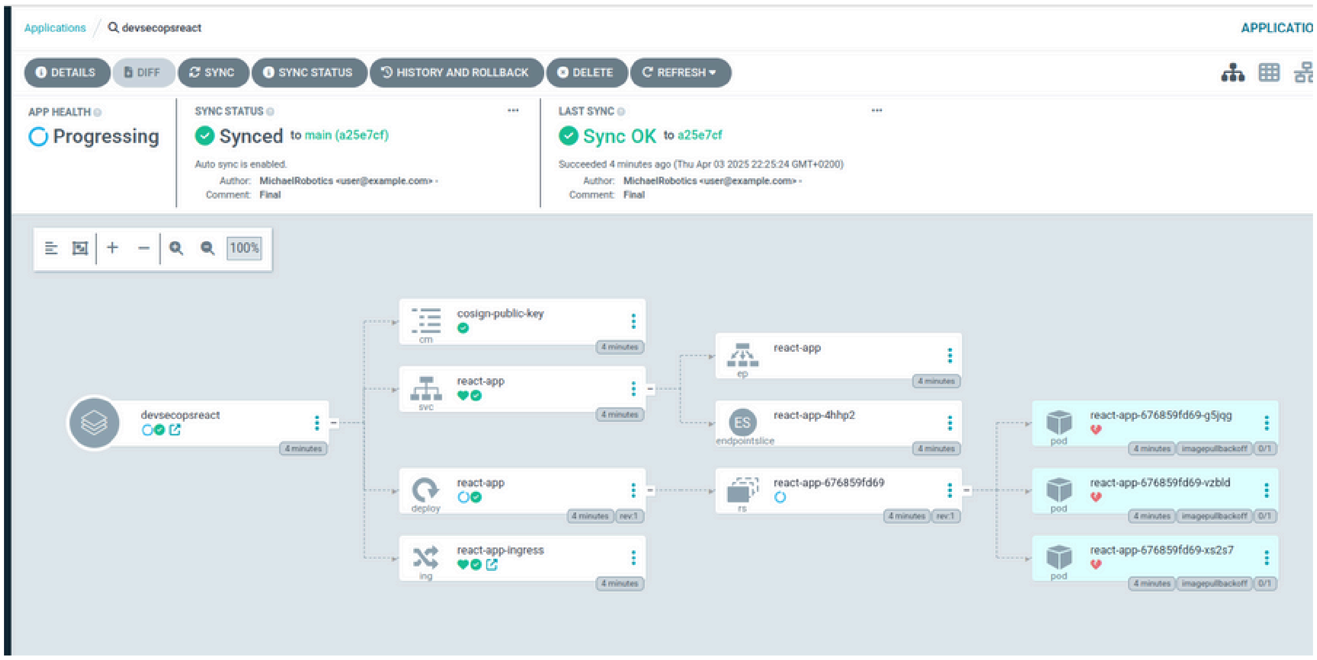
Workflow file

Manually triggered 1 minute ago	Status	Total duration	Artifacts
MichaelRobotics a25e7cf main	Success	1m 0s	–

application-deployment.yml
on: workflow_dispatch

Deploy ArgoCD Application... 55s

Check ArgoCD. It should throw error with wrong images, beacause we didnt created images and pushed changes into kubernetes manifests files:

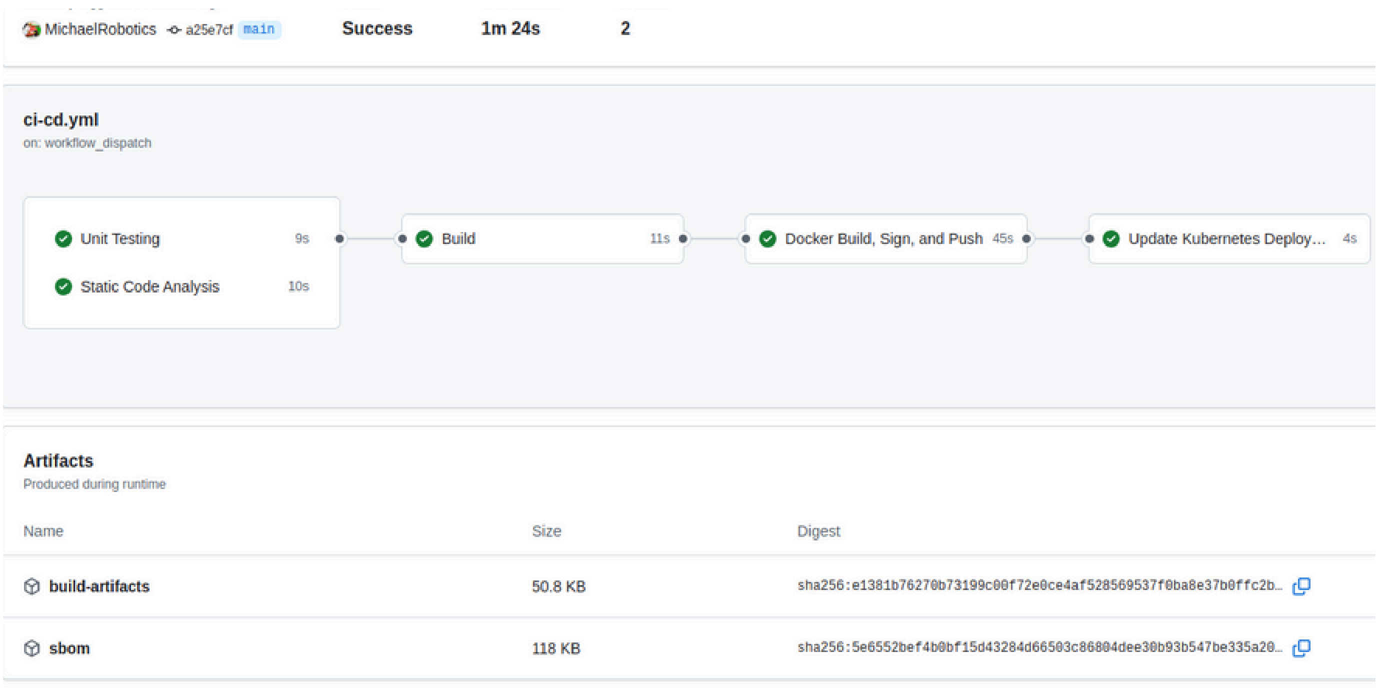


APP HEALTH
Progressing

SYNC STATUS
Synced to main (a25e7cf)
Auto sync is enabled.
Author: MichaelRobotics <user@example.com> -
Comment: Final

LAST SYNC
Sync OK to a25e7cf
Succeeded 4 minutes ago (Thu Apr 03 2025 22:25:24 GMT+0200)
Author: MichaelRobotics <user@example.com> -
Comment: Final

Launch CI/CD: CI/CD Pipeline



MichaelRobotics a25e7cf main Success 1m 24s 2

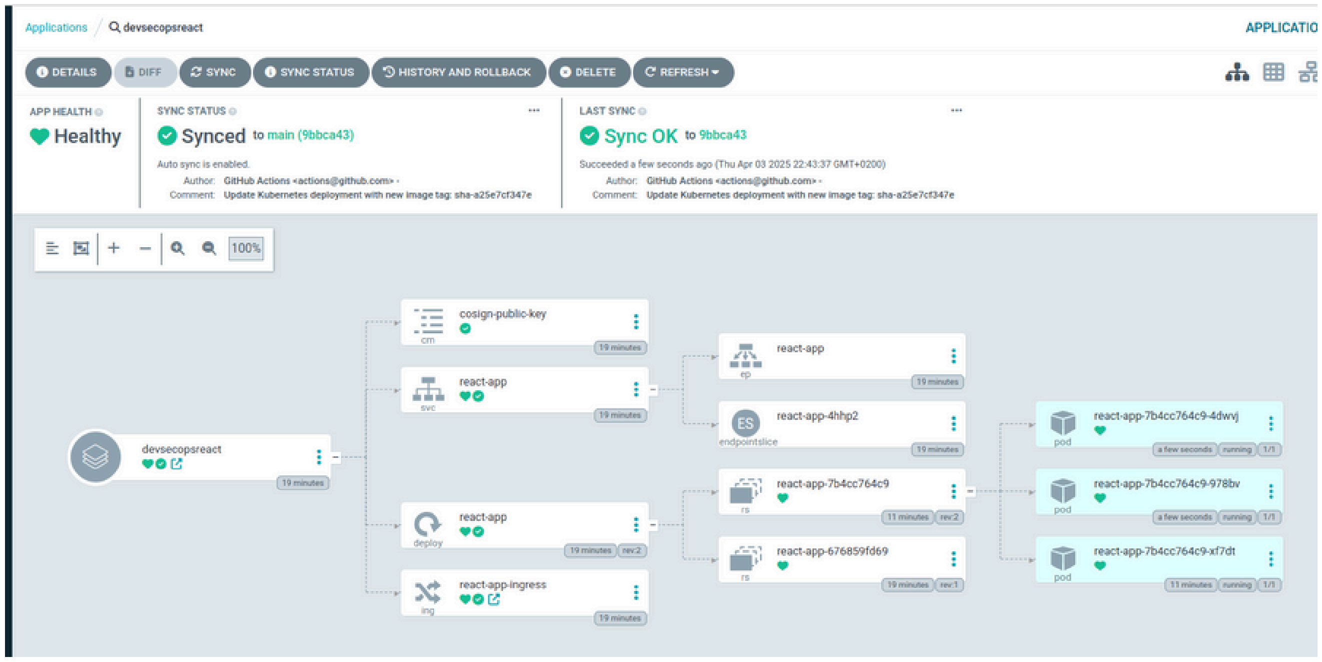
ci-cd.yml
on: workflow_dispatch

Unit Testing	9s
Static Code Analysis	10s
Build	11s
Docker Build, Sign, and Push	45s
Update Kubernetes Deploy…	4s

Artifacts
Produced during runtime

Name	Size	Digest
build-artifacts	50.8 KB	sha256:e1381b76270b73199c00f72e0ce4af528569537f0ba8e37b0ffc2b…
sbom	118 KB	sha256:5e6552bef4b0bf15d43284d66503c86804dee30b93b547be335a20…

ArgoCD should catch new images and download those:



Applications / devsecopsreact

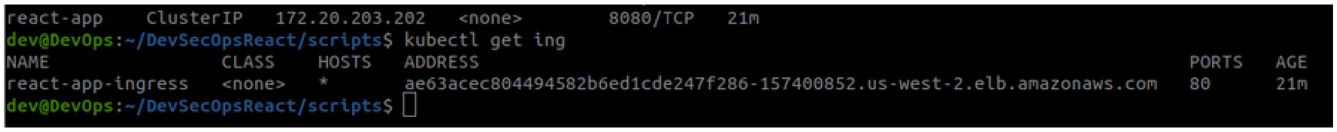
APP HEALTH: Healthy

SYNC STATUS: Synced to main (9bbca43)
Auto sync is enabled.
Author: GitHub Actions <actions@github.com> -
Comment: Update Kubernetes deployment with new image tag: sha-a25e7cf347e

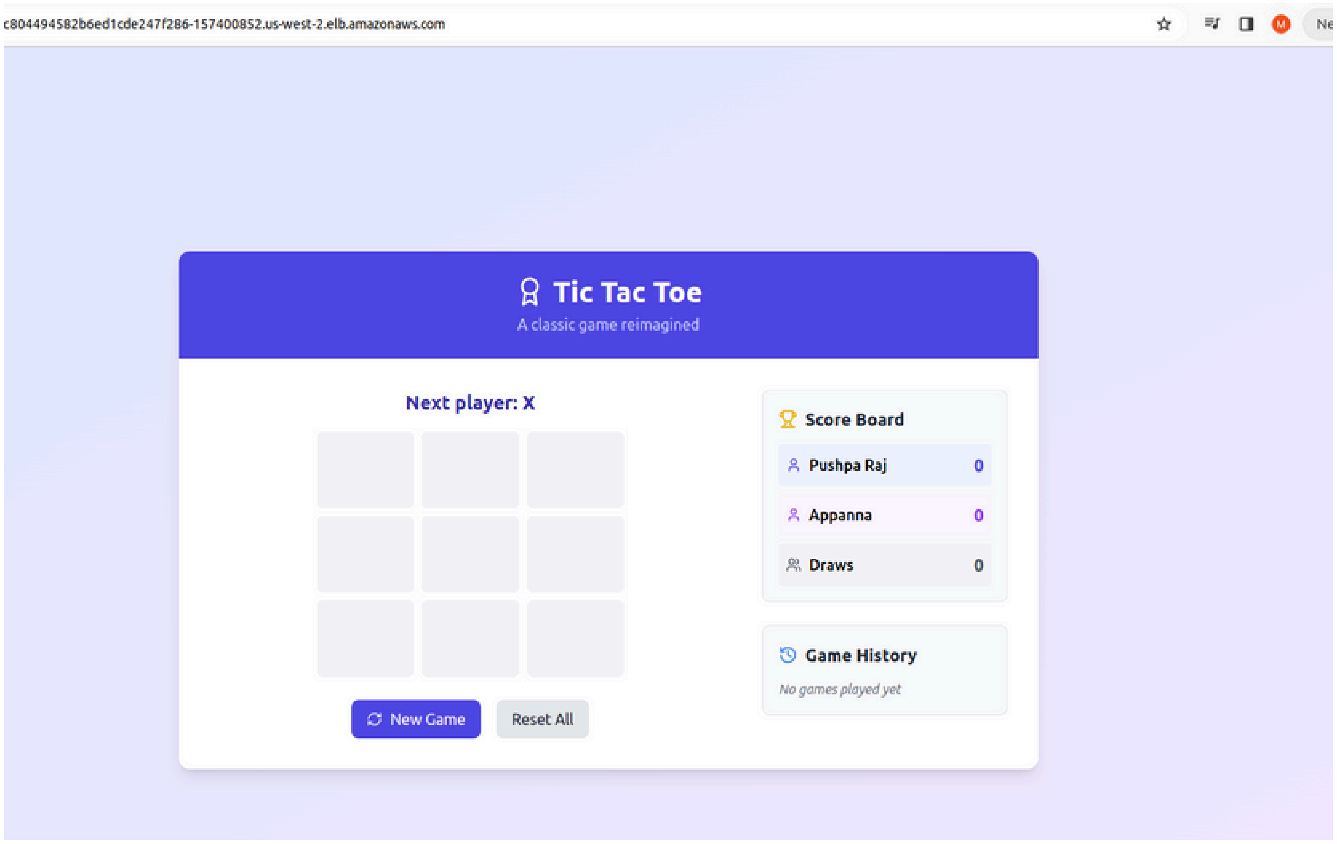
LAST SYNC: Sync OK to 9bbca43
Succeeded a few seconds ago (Thu Apr 03 2025 22:43:37 GMT+0200)
Author: GitHub Actions <actions@github.com> -
Comment: Update Kubernetes deployment with new image tag: sha-a25e7cf347e

Now retrieve application loadbalancer DNS and enjoy Your TicTacToe game!!!!

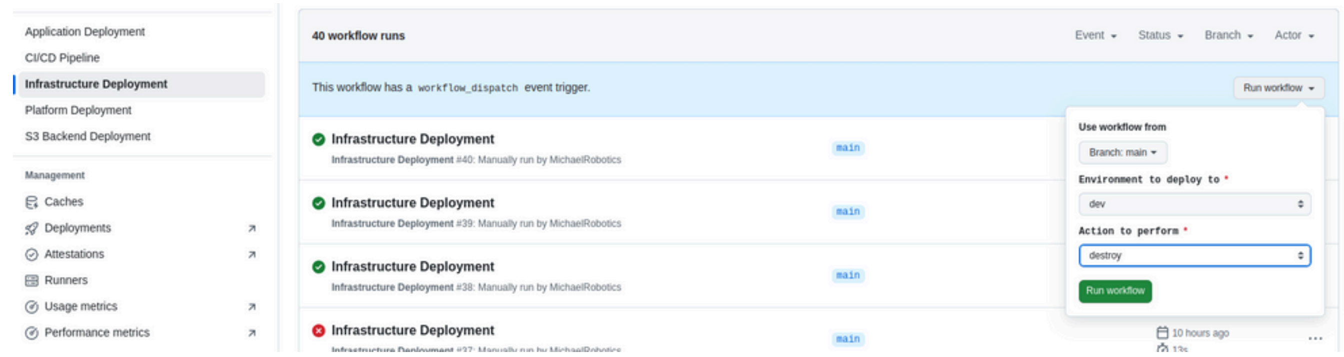
```
kubectl get ing
```



```
react-app     ClusterIP   172.20.203.202   <none>        8080/TCP   21m
dev@DevOps:~/DevSecOpsReact/scripts$ kubectl get ing
NAME                CLASS    HOSTS   ADDRESS                                                                    PORTS   AGE
react-app-ingress   <none>   *       ae63acec804494582b6ed1cde247f286-157400852.us-west-2.elb.amazonaws.com   80      21m
dev@DevOps:~/DevSecOpsReact/scripts$ 
```



To destroy, run destroy option in infrastructure pipeline:



40 workflow runs

This workflow has a `workflow_dispatch` event trigger.

- Infrastructure Deployment — Infrastructure Deployment #40: Manually run by MichaelRobotics — main
- Infrastructure Deployment — Infrastructure Deployment #39: Manually run by MichaelRobotics — main
- Infrastructure Deployment — Infrastructure Deployment #38: Manually run by MichaelRobotics — main
- Infrastructure Deployment — Infrastructure Deployment #37: Manually run by MichaelRobotics — main — 10 hours ago — 13s

Use workflow from: Branch: main

Environment to deploy to *: dev

Action to perform *: destroy

Run workflow

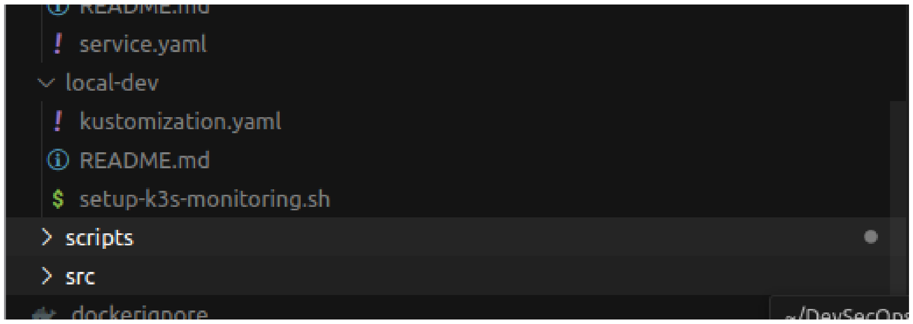
Kubernetes On Vibes: local Environment Setup

1) Intro

We won't push changes directly to Dev/Prod; all experimental work will stay in the local environment.

2) Setup

I created directory for local setup:



```
README.md
! service.yaml
⌄ local-dev
  ! kustomization.yaml
  ⓘ README.md
  $ setup-k3s-monitoring.sh
> scripts
> src
.dockerignore
```

deploy kind cluster with all tools like grafana etc.

```
./setup-k3-monitoring.sh
```

to apply our kubernetes yamls:

```
kubectl apply -k local-dev/ -n your-namespace
```

To push changes into DEV/PROD use pipelines we created!

Kubernetes On Vibes: Final words

In startups or small projects fast-scaling scenarios or simple VPS/EC2 deployments save costs and time. But as complexity rises, a scalable DevOps setup is vital for rapid, reliable delivery.

If you like this project structure, consider these enhancements:

- Dependency scans in DevSecOps for SBOM.
- Stronger AWS security groups for access control.
- EKS with CA/Karpenter for HA.
- ArgoCD backup/rollback.
- Terraform testing to avoid resource recreation.
- Separate VPC for security scanning all traffic.
- Custom dashboards and PagerDuty.
- TLS via Route53 and cert-manager.
- mTLS with Vault for secrets.
- Clean pipelines of unused configs.
- GitHub PR rules.
- Integrate SSM.
- Add tests.
- Neon for DB versioning.
- CI/CD for more microservices.
- Local automation with Makefile/Docker.
- Network policies and Kyverno.
- AWS Control Tower for medium orgs.

Common Troubleshooting

1)Image Signing Failure (Cosign)

Cause: Unsigned or tampered images in Harbour/Docker Registry.
Solution: Verify with cosign verify <image-name>, enforce signed images in admission policies (e.g., Kyverno).

2) Unrestricted Pod Privileges

Cause: Pods with excessive permissions (e.g., privileged mode or root).
Solution: Check with kubectl describe pod <pod-name> for SecurityContext, enforce restricted PodSecurityStandards via kubectl label ns <namespace> pod-security.kubernetes.io/enforce=restricted.

3) EKS Node CrashLoopBackOff

Cause: Misconfigured Helmfiles/Kustomize or resource limits.
Solution: Check logs with kubectl logs <pod-name>, adjust resources in values.yaml or kustomization.yaml.

4) Check my Kubernetes Troubleshooting series:

Michael Robotics
Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats skill information overload by adhering to the set of principles: simplify, prioritize, and execute.

https://github.com/MichaelRobotics

Learn more about Kubernetes

Check Kubernetes and piyushsachdeva - great docs!

Setup a Multi Node Kubernetes Cluster

kubeadm is a tool to bootstrap the Kubernetes cluster

https://github.com/piyushsachdeva/CKA-2024/tree/main/Resources/Day27

Kubernetes Documentation

This section lists the different ways to set up and run Kubernetes

https://kubernetes.io/docs/setup/

Share, comment, DM and check GitHub for scripts & playbooks created to automate process.

Check my GitHub

Michael Robotics

Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats skill information overload by adhering to the set of principles: simplify, prioritize, and execute.

https://github.com/MichaelRobotics

PS.

If you need a playbook or bash script to manage KVM on a specific Linux distribution, feel free to ask me in the comments or send a direct message!