# GitOps Essential Training

## About DevOpsSchool

DevOpsSchool is a unit of "Cotocus PVT ltd" and a leading platform which helps IT organizations and professionals to learn all the emerging technologies and trend which helps them to learn and embrace all the skills, intelligence, innovation and transformation which requires to achieve the end result, quickly and efficiently. We provide over 40 specialized programs on DevOps, Cloud, Containers, Security, AI, ML and on Big data that are focused on industry requirement and each curriculum is developed and delivered by leading experts in each domain and aligned with the industry standards.

## About Course

GitOps defines a better approach to performing Continuous Delivery in the context of a Kubernetes cluster. It does so by promoting Git as the single source of truth for declarative infrastructure and workloads.

This Learning Path will get you started with GitOps and will bring you quickly up to speed with the basic features and processes involved in a GitOps workflow. First, we introduce you to GitOps and the many benefits that it provides when it comes to automating deployments. The Learning Path then provides you with a validated hands-on lab that will walk you through the process of setting up and using GitOps. Finally, you're provided with a multichoice exam to assess the GitOps knowledge the Learning Path has provided you with

© 2021 Cotocus private limited - All rights Reserved.

Co-coordinator – Akanksha Kumari

Call/WhatsApp: - +91 1800 889 7977

Mail Address: -

contact@DevOpsSchool.com

Secondary contact – Patrick

Call/WhatsApp: - +91 7004 215 841

Mail Address: -contact@DevOpsSchool.com

| | |
|---|---|
| Duration | 40 Hours |
| Mode | Online (Instructor-led, live & Interactive) |
| Projects (Real time scenario based) | 1 |

© 2021 Cotocus private limited - All rights Reserved.

| FEATURES | DEVOPSSCHOOL | OTHERS |
|---|---|---|
| Faculty Profile Check | ✓ | ✗ |
| Lifetime Technical Support | ✓ | ✗ |
| Lifetime LMS access | ✓ | ✗ |
| Top 25 Tools | ✓ | ✗ |
| Interviews Kit | ✓ | ✗ |
| Training Notes | ✓ | ✗ |
| Step by Step Web Based Tutorials | ✓ | ✗ |
| Training Slides | ✓ | ✗ |
| Training + Additional Videos | ✓ | ✗ |

© 2021 Cotocus private limited - All rights Reserved.

## Training

DevOps As part of this course, you would be strong in DevOps technology. You would learn Linux, Python, DevOps, Docker, Jira, Git, SonarQube, Maven, Ansible, Jenkins, Kubernetes, Datadog, Splunk, NewRelic, Terraform and various other stacks related to this methodology.

## Projects

As part of this project, we would help our participant to have first-hand experience of real time software project development planning, coding, deployment, setup and monitoring in production from scratch to end. We would also help participants to visualize a real development environment, testing environment and production environments. Project technology would be based on Java, Python and DOTNET and based on Microservices concept.

## Interview

As part of this, you would be given complete interview preparation support until you clear a interview and get on boarded with organization including demo interview and guidance. More than 50 sets of Interview KIT would be given including various project scenario of the projects.

© 2021 Cotocus private limited - All rights Reserved.

## AGENDA OF THE GITOPS ESSENTIAL TRAINING

### Introduction

- Course Introduction

### GitOps

- GitOps Workflow
- GitOps Architecture

### Demonstrations

- Prepare Local Kubernetes Cluster
- Install Helm and Tiller
- Install Flux Operator
- Review Cloud Academy GitOps Demo GitHub Repo
- Test Container Deployment
- Update Deployment Manifest and Sync
- Update Container Image and Sync
- Configuration Drift and Sync

### Required knowledge

- Git: Committing code and creating pull requests
- Kubernetes: Deploying a service to Kubernetes and basic checks with kubectl
- Docker: Pushing an image to a Docker repository
- CI/CD: GitOps reverses the traditional understanding of continuous integration/continuous development.

### Core concepts: A quick introduction

- Add your content...Immutable infrastructure
- Infrastructure as code
- Orchestration
- Convergence
- CI/CD

### What GitOps is Not

- GitOps is not infrastructure as code.
- GitOps doesn't replace continuous integration (CI).

© 2021 Cotocus private limited - All rights Reserved.

## The use case: Deploying a highly available microservice

- Deploying a Microservices to Kubernetes, with all the surrounding infrastructure to make it available

## Implementing GitOps for Kubernetes in AWS

Use Weave Flux and Helm to implement GitOps methodologies in an AWS-hosted Kubernetes application by using Git as a single source of truth for Kubernetes deployments.

- Logging in to the Amazon Web Services Console
- Connecting to the Cloud Academy Web based Containers IDE Port 8080
- Reviewing a DevOps Pipeline for Kubernetes in AWS
- Deploying a Kubernetes Application with AWS Code Pipeline
- Implementing GitOps for Kubernetes in AWS
- Validating GitOps for Kubernetes in AWS

## Kubernetes

- In advance: Setting up a cluster from scratch is time-consuming, even if you use a managed solution like EKS. Pre-allocating a cluster per person is something you can do in advance.

## Preparation: Setting up kubectl

- Introduction
- CIDR Classless Inter-Domain Routing
- Subletting Overview
- Subletting Class C Networks and VLSM
- Subletting Practice Questions
- Variable Length Subnet Masking Example Part 1
- Variable Length Subnet Masking Example Part 2
- Subletting Large Networks Part 1
- Subletting Large Networks Part 2
- Subletting on the 4th Octet - Written Example
- Subletting on the 3rd Octet - Written Example
- Private IP Addresses Part 1
- Private IP Addresses Part 2
- Subletting Table and Subnet Calculator
- Where to Get More Subletting Practice
- Additional Subletting Practice Sites

© 2021 Cotocus private limited - All rights Reserved.

## Preparation: Access a cluster through kubectl/k9s

- Check running pods.
    - Check deployments.

## Repository

## Preparation: Infrastructure repository

- An empty repository in GitHub/GitLab to use for deploying infrastructure

## Application repository

- Strictly speaking, you don't need to separate the application and infrastructure, but it's easier to understand what goes where this way.
    - A sample application that serves a web server with a hello world response as the baseline (NodeJS-based, for instance)

## ArgoCD

## Why ArgoCD?

- ArgoCD is tightly integrated with Kubernetes and closely follows the GitOps mindset. Therefore, it's a good tool to showcase GitOps.

## Exercise: Add ArgoCD to the cluster.

- Create namespace.
- Deploy ArgoCD to the cluster.
- Access ArgoCD using the CLI.

## Exercise: Prepare a simple microservice to be deployed in k8s.

- Build sample application as a Docker container (Dockerfile can be provided in advance)
- Push service to a Docker registry (cloud-native, docker.io, or quay.io)

## Exercise: Create a k8s deployment

- Create a Kubernetes deployment definition in code for the application (here's a sample).
- Push code to infrastructure repository.
- Create an application in ArgoCD.
- This time, you'll use the ArgoCD CLI so you can see that part. You'll move to use Git from here on, which is more aligned to GitOps.
- Sync the application.
- Again, use the CLI.
- Test: Use kubectl check to ensure that deployment works..

© 2021 Cotocus private limited - All rights Reserved.

## Automated synchronization

- Pull versus push: How ArgoCD can read from a repository and automatically apply the changes
- Exercise: Activate synchronization so that further changes happen when you push code to the infrastructure repository.

## Exercise: Create a k8s service. (A deployment alone doesn't expose the microservice, so let's build on that.)

- Create service definition.

## Pull request

- This is an opportunity to introduce the pull request aspect of the flow. You can extend pull requests so that extra checks are performed, using something like GitHub Actions.
- Implementing CI with something like GitHub Actions isn't part of the exercise, although it's something that you can complete as an extra exercise. (See the bonus section at the end of this post.)
- Test: Carry out a kubectl check to prove that service was deployed.

## Exercise: Create a load balancer. (You still can't access service from the outside.)

- Create loadbalancer k8s definition for cloud provider.
- Pull request
- Test: Curl to load balancer address to ensure that service is actually online.

## Exercise: Update the application.

- Change something in the application, such as the body of the response of a route in the application.
- Rebuild container with a new tag and push it to Docker registry.
- Update k8s deployment to use new tag.
- Pull request
- Test: New version of the app should be deployed.

## Exercise: Update the infrastructure. (Why do this? So you can demonstrate that changing the application and the infrastructure results in blurry boundaries.)

- Update k8s deployment to be highly available (more than one replica).
- Pull request
- Test: kubectl shows that there are multiple pods running.

## Wrap-up: This covers the workflow of deploying an application and then performing updates and changes on it.

- This is the core of GitOps!
- There are also other, more advanced use cases to cover.

© 2021 Cotocus private limited - All rights Reserved.

## In advance: Prepare a second cluster.

- As with the first cluster, this is something to have prepared in advance.

## In advance: Prepare a Preparation: Register the cluster in ArgoCD to allow deployments to it.

## From development to production

- Which options are there to represent different stages?
- This is an open discussion, as there's no set recipe to do environment promotion, with different options:
- Use different infrastructure repositories.
- Use different folders in the same infrastructure repository.
- Use branches.

## Exercise: Promotion of a version

- Set up a second cluster (production) to read from a different folder.
- Copy the infrastructure created for the first folder into this one.
- Pull request
- Test: Second cluster should have the service available as well.

## More advanced deployment scenarios (Controlled release is an important part of releasing traffic, especially to production. It's worth talking about the options that you have that can be based on the exact same building blocks as explained before.)

## Exercise: Blue/Green

- Enable Argo Rollouts in cluster.
    - Test: Observe rolling deployment with kubectl.
- Install argo-rollouts plugin for kubectl.
- Create rollout to apply to existing Microservices.

## Canary release

- Theory only (This can be a good lead-in to a discussion of the merits and tradeoffs of different deployment strategies.)

© 2021 Cotocus private limited - All rights Reserved.

## Exercise: Error handling (This exercise shows that failure in infra deployment is expected and is handled through code changes—not panicked actions!)

- Introduce an error in the hello world application (this results in a thrown exception instead of starting the webserver).
- Rebuild the container with a new tag and push it to Docker registry.
- Update k8s deployment to use new tag.
- Pull request
- Test: Confirm with kubectl that deployment is failing.
- Revert a failed change through code.

## Accessing resources

- kubectl shouldn't replace observability, such as logging and monitoring (similar to secure shell—SSH—into a production server)

## Secrets

- No plaintext secrets should ever be stored in Git.

## Vault

- This is theory only because it's probably too much to do for a practical exercise.

## Exercise: Sealed secrets

- Depending on time, this can be treated as theory or as an exercise. Furthermore, you can split it in two depending on how much time you have.
- Modify Microservices to read the secret and make it available through a request.
- Provision secrets in the infrastructure repository.
- Use secrets from either the cluster or the application.
- Install the sealed secrets controller.
- Inject an encrypted secret in the infrastructure repository.
- Modify Kubernetes deployment to inject a secret into the Microservices.

## Core concepts: Infra as code, Git as the source of truth, pull model, converging changes

- In advance: Setting up a cluster from scratch is time-consuming, even if you use a managed solution like EKS. Pre-allocating a cluster per person is something you can do in advance.

## Next steps

- Automated promotion (If a deployment to a staging environment succeeds, then trigger a deployment to production.)
- Observability (Microservices that export metrics, logging aggregator, and monitoring).

## Preparation: Access a cluster through kubectl/k9s.

- Check deployments.

© 2021 Cotocus private limited - All rights Reserved.

# Thank you!

Connect with us for more info

Call/WhatsApp: - +91 968 682 9970

Mail: contact@DevOpsSchool.com

www.DevOpsSchool.com

© 2021 Cotocus private limited - All rights Reserved.