**Objective:**

Deep dive into GitOps principles and their implementation in EKS.

**Tasks:**

1. Set up ArgoCD in EKS.

2. Configure a Git repository.

3. Implement continuous deployment using GitOps.

**Documentation:**

- Introduction to GitOps.

- Benefits of ArgoCD.

- GitOps best practices.

**Prerequisites:**

1. eksctl
2. Kubectl
3. AWS-CLI
4. Argo-CD and Argo-cli
5. Docker

**Tasks:**

**1. Set up ArgoCD in EKS:**

| kubectl create namespace argocd kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/v2.4.7/manifests/install.yaml |
| --- |

**Install Argo CD CLI**

To interact with the API Server we need to deploy the CLI:

| sudo curl --silent --location -o /usr/local/bin/argocd https://github.com/argoproj/argo-cd/releases/download/v2.4.7/argocd-linux-amd64  sudo chmod +x /usr/local/bin/argocd |
| --- |

**Expose argocd-server**

By default argocd-server is not publicaly exposed. For the purpose of this workshop, we will use a Load Balancer to make it usable:

| kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}' |
| --- |

Wait about 2 minutes for the LoadBalancer creation

| **export** ARGOCD\_SERVER=`kubectl get svc argocd-server -n argocd -o json | jq --raw-output '.status.loadBalancer.ingress[0].hostname'` |
| --- |

**Login:**

The initial password is autogenerated with the pod name of the ArgoCD API server:

| export ARGO\_PWD=`kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d` |
| --- |

Using **admin** as login and the autogenerated password:

| argocd login $ARGOCD\_SERVER --username admin --password $ARGO\_PWD --insecure |
| --- |

You should get as an output:

| 'admin' logged **in** successfully |
| --- |

Access the Secret in the Correct Namespace:

You should use the correct namespace, which appears to be argocd:

| kubectl get secret argocd-initial-admin-secret -n argocd -o yaml |
| --- |

To decode the base64 password to a human-readable format, you can use the following command:

| kubectl get secret argocd-initial-admin-secret -n argocd -o jsonpath="{.data.password}" | base64 --decode && echo |
| --- |

| kubectl get **svc** -n argocd |
| --- |

List ArgoCD Applications:

| argocd **app** **list** |
| --- |

Check ArgoCD Version:

| argocd version |
| --- |

Get ArgoCD Status:

| argocd **cluster** **list** |
| --- |

**2. Configure a Git Repository:**

**Create a Repository:** Host your Kubernetes manifests in a Git repository (GitHub, GitLab, Bitbucket, etc.). [**GitHub Link**](https://github.com/m-muzammil786/gitgub-eks-cicd-pipeline)

**Connect the Repository to ArgoCD:** In the ArgoCD UI, go to Settings > Repositories, and add your repository.

**3. Implement continuous deployment using GitOps.**

1. **Create an Application in ArgoCD:** In the ArgoCD UI, click on "New App" and configure it to point to the Git repository and the path where your Kubernetes manifests are stored.
2. **Sync the Application:** Once the application is created, click on "Sync" to deploy the application using the manifests in your Git repository.
3. **Verify the Deployment:** Ensure that the resources defined in your manifests are running in your EKS cluster.

**Create application**

Connect with ArgoCD CLI using our cluster context:

| CONTEXT\_NAME=`kubectl config view -o jsonpath='{.current-context}'` argocd cluster add $CONTEXT\_NAME |
| --- |

Configure the application and link to your fork (replace the GITHUB\_USERNAME):

| kubectl create namespace ecsdemo-nodejs argocd app create ecsdemo-nodejs --repo https://github.com/GITHUB\_USERNAME/ecsdemo-nodejs.git --path kubernetes --dest-server https://kubernetes.default.svc --dest-namespace ecsdemo-nodejs |
| --- |

Application is now setup, let’s have a look at the deployed application state:

| argocd app get ecsdemo-nodejs |
| --- |

You should have this output:

| Health Status: Missing  GROUP KIND NAMESPACE NAME STATUS HEALTH HOOK MESSAGE \_ Service ecsdemo-nodejs ecsdemo-nodejs OutOfSync Missing  apps Deployment default ecsdemo-nodejs OutOfSync Missing |
| --- |

| argocd app **sync** ecsdemo-nodejs |
| --- |

After a couple of minutes our application should be synchronized.

**Documentation:**

Introduction to GitOps:

GitOps is a paradigm for continuous delivery and operations. It leverages Git as a source of truth for declarative infrastructure and applications. In GitOps, everything is codified and versioned, enabling easy rollback, audit trails, and collaboration.

**Benefits of ArgoCD:**

1. Declarative Setup: Everything is defined as code, ensuring consistency and version control.
2. Easy Rollbacks: Quickly rollback to a previous version if there is an issue.
3. Enhanced Security: Use Git’s robust security features to manage access and changes.
4. Visibility and Monitoring: Comprehensive visibility into the state of applications, and their synchronization status.
5. Integration with Multiple Git Repositories: ArgoCD supports integration with various Git providers.

**GitOps Best Practices:**

1. Infrastructure as Code: Store all your infrastructure configurations and application manifests in a Git repository.
2. Automate Everything: Automate deployments, rollbacks, and operations tasks.
3. Keep Everything Versioned: Ensure that all changes are versioned to track changes and rollback if necessary.
4. Use Branching Strategies: Utilize branching strategies like feature branches, develop branch, and main branch for organized and secure development.
5. Monitor and Alert: Implement monitoring and alerting to stay informed about the state of your deployments and take quick actions when needed.
6. Collaborate and Review: Encourage collaboration and conduct code reviews for infrastructure changes, similar to application code.