**Objective:**

Experience the Continuous Integration/Continuous Deployment (CI/CD) lifecycle on AWS with Kubernetes (EKS).

**Tasks:**

1. Set up AWS CodeCommit for source code.

2. Create a pipeline in AWS CodePipeline.

3. Automate deployments to EKS.

**Prerequisites:**

1. An AWS Account.
2. AWS CLI installed and configured.
3. EKS Cluster up and running.
4. Docker & Docker Hub Account
5. Git and Github account
6. eksctl
7. Kubernetes CLI (kubectl) installed and configured to interact with your EKS cluster.
8. Basic understanding of Kubernetes, Docker, and AWS services.

**Documentation:**

1. CI/CD essentials.
2. AWS CodeSuite for Kubernetes.
3. Best practices for automated deployments.

**Task 1: Set up AWS CodeCommit for Source Code**

**Create a Repository:**

* Go to AWS Management Console > AWS CodeCommit.
* Choose "Create repository".
* Enter Repository name and Description.
* Choose "Create".

**Clone the Repository:**

* Copy the HTTPS URL of the repository.

| [gitgub-eks-cicd-pipeline](https://github.com/m-muzammil786/gitgub-eks-cicd-pipeline) |
| --- |

* Run git clone <repository-https-url> in your terminal.

**Add your application code to the repository:**

* Navigate to the cloned repository directory.
* Add your application code and Dockerfile.
* Commit and push the changes to CodeCommit.

**Task 2: Create a Pipeline in AWS CodePipeline**

Create the role:

Now that we have the IAM role created, we are going to add the role to the [aws-auth ConfigMap](https://docs.aws.amazon.com/eks/latest/userguide/add-user-role.html) for the EKS cluster.

Once the ConfigMap includes this new role, kubectl in the CodeBuild stage of the pipeline will be able to interact with the EKS cluster via the IAM role.

**Launch template:**

[**CodePipeline & EKS**](https://console.aws.amazon.com/cloudformation/home?#/stacks/create/review?stackName=eksws-codepipeline&templateURL=https://s3.amazonaws.com/eksworkshop.com/templates/main/ci-cd-codepipeline.cfn.yml)

1. codebuild-role

| kubectl **edit** -n kube-**system** configmap/aws-auth |
| --- |

| apiVersion: v1 kind: ConfigMap metadata:  name: aws-auth  namespace: kube-system data:  mapRoles: |  - groups:  - system:bootstrappers  - system:nodes  rolearn: arn:aws:iam::587027614473:role/eksctl-eks-nodegroup-eks-group-NodeInstanceRole-g0bsKhXNi0Cb  username: system:node:{{EC2PrivateDNSName}}  - groups:  - system:masters  rolearn: arn:aws:iam::587027614473:role/eks-pipeline-CodeBuildServiceRole-0eWSEblq6RZ9  username: codebuild  - groups:  - system:masters  rolearn: arn:aws:iam::587027614473:role/eks-pipeline-CodePipelineServiceRole-fJM3q1U5OFl2  username: codepipeline |
| --- |

**Task 3: Automate Deployments to EKS**

To review the status of the deployment, you can run:

| kubectl describe deployment deployment.yml kubectl get services kubectl get svc |
| --- |

**Documentation:**

**CI/CD essentials:** Understanding the principles of Continuous Integration and Continuous Deployment is crucial. It involves automating the build, test, and deployment of applications.

**AWS CodeSuite for Kubernetes:** AWS provides a suite of tools (CodeCommit, CodeBuild, CodeDeploy, and CodePipeline) that can be used to implement CI/CD pipelines for applications running on Kubernetes.

Best practices for automated deployments:

**Immutable Infrastructure:** Once a version of your application is deployed, do not change it. If you need to make changes, deploy a new version.

**Rolling Updates:** Deploy new versions gradually to ensure zero downtime and easy rollback in case of failure.

**Version Control:** Keep track of what is deployed and ensure that everything that goes to production is in version control.

**Automated Testing:** Ensure that your CI/CD pipeline includes steps to run automated tests to catch issues early.

**Monitoring and Logging:** Implement robust monitoring and logging to be aware of what is happening with your application in real-time.