business automation

SDG Project Deployment Process in a Kubernetes Cluster

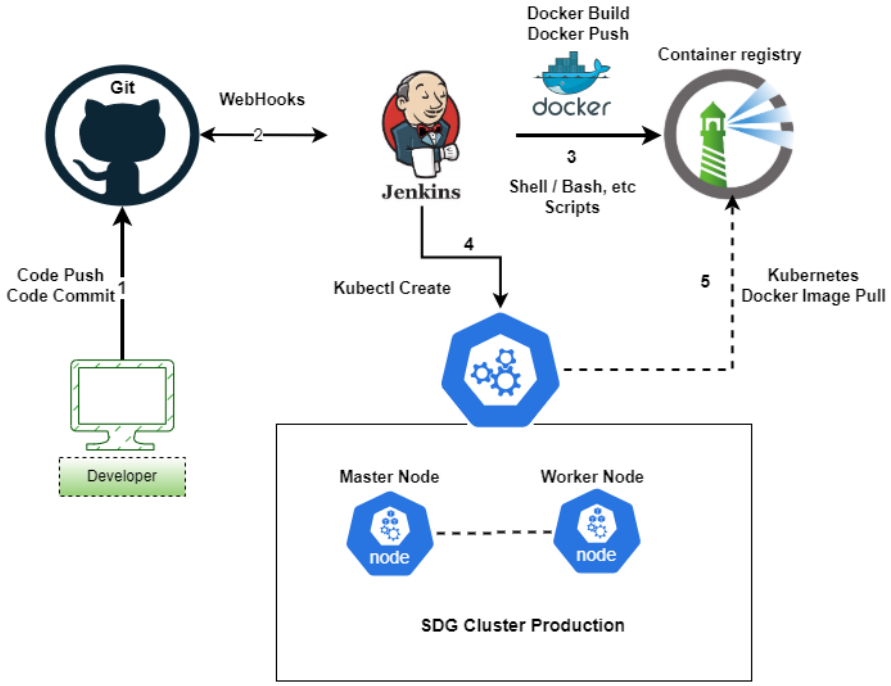
Presented By:

Khandakar Rabbi Ahmed
Associate DevOps Engineer, CIRT & Infra Team
Business Automation Ltd.

Session Outline

- ✓Introduction to CI/CD
- ✓CI/CD Pipeline Workflow
- ✓CI/CD Tools Overview
- ✓Hands-on: Writing a Simple CI/CD Pipeline

Automating CI/CD with Git, Jenkins, Harbor, and Kubernetes



SDG Project Deployment Process in a Kubernetes Cluster

business automation

Pushing Image to Harbor Registry

- Jenkins logs in to Harbor and pushes the image
- Example commands:
  docker login harbor.example.com -u username -p password
  docker push harbor.example.com/my-app:latest

Building the Docker Image

- Jenkins builds a Docker image using a Dockerfile
- Example command:
  docker build -t harbor.example.com/my-app:latest .
- Tags image with version or commit ID

Deploying to Kubernetes Cluster

- Jenkins updates deployment YAML with the new image
- Applies the deployment using kubectl:
  kubectl apply -f deployment.yaml
- Kubernetes pulls the latest image and updates the application

Full Jenkins Pipeline Example

```
pipeline {
    agent any
    environment {
        HARBOR_URL = 'harbor.example.com'
        IMAGE_NAME = 'my-app'
        K8S_NAMESPACE = 'my-namespace'
    }
    stages {
        stage('Checkout Code') {
            steps {
                git 'https://github.com/user/repository.git'
            }
        }
        stage('Build Docker Image') {
            steps {
                sh "docker build -t $HARBOR_URL/$IMAGE_NAME:latest ."
            }
        }
        stage('Push to Harbor') {
            steps {
                sh "docker login $HARBOR_URL -u username -p password"
                sh "docker push $HARBOR_URL/$IMAGE_NAME:latest"
            }
        }
        stage('Deploy to Kubernetes') {
            steps {
                sh "kubectl apply -f k8s/deployment.yaml"
            }
        }
    }
}
```

Conclusion

Jenkins automates deployment from Git to Kubernetes:
- Code changes trigger CI/CD pipeline
- Docker image built and pushed to Harbor
- Kubernetes deployment is updated automatically

Understanding Kubernetes Cluster

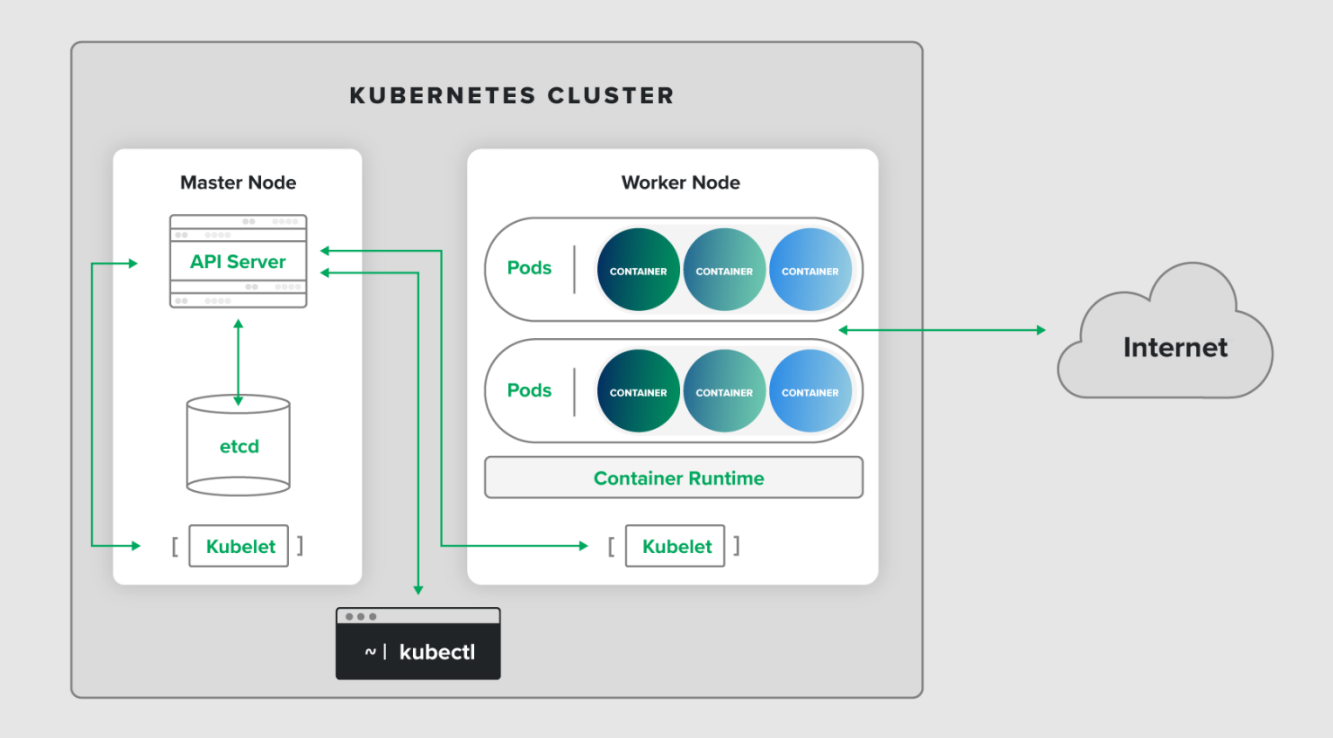
How Kubernetes Manages Pods, Communication, and Node Responsibilities

Kubernetes Cluster Overview

- Kubernetes is a container orchestration system
- Manages application deployment, scaling, and operations

Consists of a Master Node and multiple Worker Nodes

Kubernetes Cluster Diagram



Worker Node Responsibilities

Runs application workloads inside Pods

- Key components:
  - Kubelet: Communicates with Master Node
  - Container Runtime: Runs containers (e.g., Docker, containerd)
  - Kube Proxy: Manages networking and service discovery

How Pods Are Created

- Developer defines a **Deployment** or **Pod manifest**
- Master Node schedules the Pod on a Worker Node
- Kubelet on Worker Node pulls container images and starts the Pod
- Containers inside the Pod share the same network and storage

Internal Communication in Kubernetes

- Pods communicate within a Node using localhost
- Services expose Pods across Nodes
- ClusterIP (default) allows internal-only access
- NodePort & LoadBalancer expose externally

Master Node to Worker Node Communication

- API Server sends instructions to Worker Nodes
- Kubelet on each Worker Node receives tasks from API Server
- etcd stores cluster state, ensuring consistency
- Communication is secured via TLS encryption

Kubernetes Networking Model

- Each Pod gets a unique IP
- CNI (Container Network Interface) manages network plugins
- Common CNI implementations: Flannel, Calico, Cilium
- Services enable stable communication between Pods

Conclusion

Kubernetes simplifies container orchestration by:
- Managing Pods efficiently across Nodes
- Ensuring reliable communication between services
- Automating deployment, scaling, and recovery

Thank You! Questions?