Page No.
Date

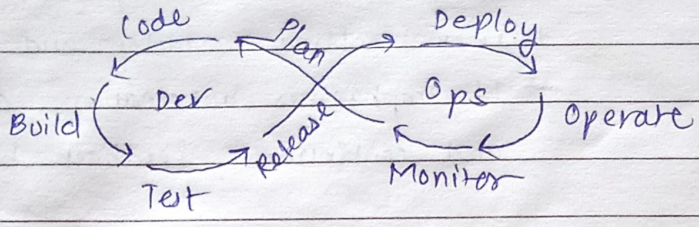
DevOps : A set of practices that combines software development (Dev) and IT Operations (Ops) to shorten the development lifecycle and deliver software faster and more reliably.
→ It emphasizes automation, collaboration, and continuous feedback between teams.

Containerisation
- It is the process of packaging an application and its dependencies into a lightweight, portable unit called a container.
- It ensures that the application runs consistently across different environments.
- Docker is the most commonly used containerization tool.
- Container orchestration automates the deployment, scaling, and management of containers in production environments.
- Kubernetes is the most popular orchestration tool, handling load balancing, self healing, and resource management of containers.

DevOps Lifecycle : Includes continuous development, integration, testing, deployment, monitoring, and feedback.
- It's an iterative process that ensures countinuous improvement and automation throughout the software delivery pipeline.

Page No.
Date

Plan → Code → Build → Test → Release → Deploy → Operate → Monitor → Feedback




CI/CD Pipeline

It automate the process of building, testing, and deploying code.

- Continous Integration (CI)
  - Developers frequently merge code changes into a shared repository where automated builds and tests run
- Continous Deployment / Delivery (CD)
  - Automatically delivers or deploys tested code to production or staging environments.

EX: Jenkins, Github Actions, Gitlab CI/CD

Infrastructure as Code (IaC)

- It is the practice of managing and provisioning infrastructure using code, instead of manual configuration.
- It allows you to version, test, and automate your infrastructure setup just like application code
- Tools: Terraform, Ansible, Puppet, . . .

Page No.
Date

## Configuration Management

- It ensures that systems and software maintain a consistent, desired state across all servers
- It automates tasks like installing packages, managing updates, and maintaining configurations.
- IaC provisions infrastructure (creates resources), while configuration management maintains and configures it
- Tools : Ansible, Puppet, Chef

## Version Control

- It is a system that tracks and manages changes to code over time.
- It allows multiple developers to collaborate, revert changes, and maintain code history.
- Types : 1) Centralized : SVN
  2) Distributed : Git

## Blue - Green Deployment

- Blue environment represents the current production version of the application that users are currently using.
- Green environment represents the new version of the application. It runs parallel to Blue for testing.
- Once validated, all user traffic is redirected to Green, making it the new production environment
- Key Benefit: Zero downtime, easy rollback

Page No.
Date

Docker :

- It is a platform that lets you build, ship, and run application inside containers.
- It is a lightweight, portable environments that includes everything the app needs to run.
- It eliminates environment inconsistencies between development, testing, and production.
- Uses dockerfile to define the environment and dependencies
- Images are stored in Docker Hub or private registries

Docker Hub: cloud based registry for sharing and managinging Docker images. You can push, pull, and store images for public or private use.

Docker Components

- Docker Engine – core service that runs containers on a host machine
- Docker Image – immutable template used to create containers
- Docker Containers – running instance of a docker image
- Dockerfile – script defining how to build a docker image
- Docker Compose – tool to define and run multi container applications
- Docker Hub / Registry – storage and distribution platform for images.

Page No.
Date

# Docker Commands

1) [docker --version] — Check Docker version
   Eg: docker --version
2) [docker ps] — List running containers
   Eg: docker ps
3) [docker stop] — Stop a container
   Eg: docker stop <container_id>
4) [docker rm] — Remove a container
   Eg: docker rm <container_id>
5) [docker logs] — View container logs
   Eg: docker logs <container_id>
6) [docker build] — Build image from Dockerfile
   Eg: docker build -t myapp:1.0
7) [docker run] — Run a container from image
   Eg: docker run -d -p 8080:80 myapp:1.0
8) [docker tag] — Label a specific version of image
   Eg: docker tag myapp:1.0 myusername/myapp:1.0
9) [docker push] — Push image to Docker Hub
   Eg: docker push myusername/myapp:1.0
10) [docker pull] — Pull image from Docker Hub
    Eg: docker pull ubuntu:latest
11) [docker rmi] — Remove image
    Eg: docker rmi myapp:1.0

Page No.
Date

Dockerfile Example :-

base image — FROM python:3.10-slim
set working directory — WORKDIR /app
copy requirements & install dependencies — COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
copy application code — COPY . .
Expose port for container — EXPOSE 5000
command to run app — CMD ["python", "app.py"]

Kubernetes :

- An open source container orchestration tool that automates deployment, scaling, and management of containerized applications.
- It groups containers into pods, manages load balancing, and automatically replaces failed containers.
- Features :
  - Self healing → restarts failed pods automatically
  - Scaling → scales up/down based on demand
  - Rolling updates → deploys new versions without downtime

Kuberenetes (K8s) Components

- Cluster - group of nodes managed by Kubernetes
- Master Node - controls the cluster, manages API server, scheduler, controller manager
- Worker Node - runs containers (via Pods). Containe

Page No.
Date

Kubelet, kube-proxy, container runtime.

- Pod - smallest deployable unit, can contain one or more containers
- Deployment - manages a set of identical Pods, ensures desired state
- Service - exposes pods to network, load balances traffic
- ConfigMap / Secret - stores configuration and sensitive data for Pods
- Namespace - virtual cluster within a cluster for isolation

Kubernetes Commands

1) [kubectl version] Show kubernetes version
   Eg: kubectl version --short
2) [kubectl get nodes] List cluster nodes
   Eg: kubectl get nodes
3) [kubectl get pods] list pods
   Eg: kubectl get pods
4) [kubectl describe pod <pod-name>] Detailed pod info
   Eg: kubectl describe pod myapp-pod
5) [kubectl create -f <file.yaml>] Create resources from YAML
   Eg: kubectl create -f deployment.yaml
6) [kubectl apply -f <file.yaml>] Update or create resources
   Eg: kubectl apply -f deployment.yaml
7) [kubectl delete pod <pod-name>] Delete pod
   Eg: kubectl delete pod myapp-pod
8) [kubectl logs <pod-name>] View container logs
   Eg: kubectl logs myapp-pod

9) [kubectl exec -it <pod-name> -- bash] Access container shell

Eg: kubectl exec -it myapp-pod --bash

10) [kubectl expose deployment <name> -- type = LoadBalancer
--port = 80] Expose deployment

Eg: kubectl expose deployment myapp-deploy -type = LoadBalancer --port = 80

Sample Files

Deployment YAML :-

```
apiVersion : apps/v1
kind : Deployment
metadata :
    name : myapp-deployment          # Deployment name
spec :
    replicas : 3                     # no. of pod replicas
    selector :
        matchLabels :
            app : myapp
    template :
        metadata :
            labels :
                app : myapp
        spec :
            containers :
                - name : myapp-container
                  image : myusername/myapp : 1.0
                                     # Docker image
                  ports :
                    - containerPort : 5000
```

Page No.
Date

Service YAML :-

```
apiVersion : v1
kind : Service
metadata :
    name : myapp-service
spec :
    selector :
        app : myapp          # connect to Pods with this label
    type : LoadBalancer      # Expose externally
    ports :
        - protocol : TCP
          port : 80
          targetPort : 5000
```

Explanation of Sample

- Deployment → ensures 3 Pods are always running
- Service → exposes Pods via a load balancer on port 80
- Labels / Selectors → link service to pods
- Replicas → enables scaling and fault tolerance.

Page No.
Date

Jenkins

- It is an open source automation server used for implementing Continuous Integration (CI) and Continuous Delivery (CD).
- It automates the building, testing, and development of code whenever a developer pushes changes.
- When code is pushed to Github, Jenkins automatically builds it, runs tests, and deploys containers to Kubernetes.
- Written in Java, Uses plugins to integrate with tools like Git, Maven, Docker, and Kubernetes, supports pipelines defined using Jenkinsfile (Declarative or Scripted)

Ansible

- An open source automation tool used for configuration management, application deployment, and task automation.
- It's agentless, meaning it doesn't require software to be installed on target machines. It communicates over SSH.
- Uses YAML playbooks to define automation tasks
- Idempotent: running the same playbook multiple times results in the same state.
- Integrates well with CI/CD pipelines for automated deployments.

Page No.
Date

## Selenium

- An open source tool for automated testing of web applications
- It allows you to simulate user interactions like clicks, form submissions, and navigation across different browsers.
- Supports multiple languages : Java, Python, C#, JS
- Components : Selenium WebDriver, Selenium IDE, Selenium Grid
- Used mainly for functional and regression testing

## Puppet

- A configuration management tool that automates system administration tasks and enforces a desired state on servers.
- It uses a declarative language to define system configurations.
- Requires Puppet agent on target machines
- Ideal for large scale server management
- Manages packages, services, and files across multiple servers automatically.

Git : distributed version control system used to track changes in code, collaborate with team members, and maintain history.

Page No.
Date

GitHub: Cloud based platform that hosts Git repositories, supports collaboration via pull requests, issues, and branches, and integrates with CI/CD pipelines.

Commands -

1) git init — Initialize a new Git repository
2) git clone — Clone a remote repository
3) git status — Check repository status
4) git add — Stage files for commit
5) git commit — Commit staged changes
6) git push — Push changes to remote
7) git pull — Pull latest changes from remote
8) git branch — List or create branches
9) git checkout — Switch branches
10) git merge — Merge branches

Branching strategy: feature branch → PR → review → merge → main

GitHub Actions

- A CI/CD automation tool built into GitHub that lets you automate workflows like build, test and deployment whenever code changes are pushed
- Uses YAML files in [.github/workflows/]
- Triggers: push, pull-request, schedule
- Supports matrix builds, testing across multiple environments, and deployment automation

Page No.
Date

Git Workflow

- A structured approach for using Git in a team to develop, test, and merge code efficiently.
- It defines how branches are used, how commits are integrated, and how code is reviewed.
- Git Workflows :
  - Feature Branch Workflow
    → Each feature is developed in its own branch
    → merge to main
  - GitFlow Workflow
    → structured workflow with develop, feature, release, hotfix branches
  - Forking Workflow
    → Developers fork repo → work → PR back to main repository

A workflow in GitHub Actions is a YAML file defining automation for CI/CD consisting of :-

- Events : what triggers the workflow (push, pull-request, schedule)
- Jobs : independent tasks that run (can run in parallel or sequentially)
- Steps : individual commands or actions inside a job
- Actions : reusable commands or scripts (actions/checkout, custom actions)
- Runners : machines that execute jobs
- Services : containers needed for the workflow (eg. database)

Page No.
Date

Pipeline : Entire workflow from code commit → build → test → deploy

Task : Individual unit of work inside a pipeline (eg: install dependencies, run tests)

Sample YAML file :—

```
# File : .github/workflows/ci-cd.yml
name : CI/CD                          # workflow name
# Event
on :
  push :
    branches : [main]
  push_request :
    branches : [main]
# Jobs
jobs :
  build :
    runs-on : ubuntu-latest           # Runner
    # Services
    services :
      postgres :
        image : postgres:15
        ports :
          - 5432 : 5432
        env:
          POSTGRES_USER : user
          POSTGRES_PASSWORD : password
        options : >-
```

Page No.
Date

```
--health-cmd "pg_isready"
--health-interval 10s
--health-timeout 5s
--health-retries 5
steps:
  - name: Checkout code          # step
    uses: actions/checkout@v3    # Action
  - name: Set up Python
    uses: actions/setup-python@v4
    with:
      python-version: '3.10'
  - name: Install Dependencies
    run: |
      pip install -r requirements.txt
  - name: Run Tests
    run: |
      pytest
```

## Tekton Trigger Flow

It is a kubernetes-native CI/CD framework

- Pipeline : sequences of tasks (build → test → deploy)
- Task : single unit of execution
- Trigger : event that starts the pipeline (eg: git push)
- TriggerTemplate : defines what pipeline to run and which parameters to pass
- Pipeline Run : executes the pipeline when the event occurs.

Page No.
Date

Tekton YAML sample :-

```
apiVersion : tekton.dev/v1beta1
kind : Pipeline
metadata :
  name : sample-pipeline
spec :
  tasks :
    - name : fetch-repo
      taskRef :
        name : git-clone
      params :
        - name : url
          value : "https://github.com/user/repo.git"
        - name : revision
          value : "main"
    - name : build-app
      taskRef :
        name : build-task
```

TrigerFlow Example :-

```
apiVersion : triggers.tekton.dev/v1beta1
kind : TriggerTemplate
metadata :
  name : pipeline-trigger-template
spec :
  params :
    - name : gitrevision
      description : The git revision
    - name : gitrepositoryurl
```

description : the git repository URL
resourcetemplates :
- apiVersion : tekton.dev/v1beta1
  Kind : PipelineRun
  metadata :
    generateName : sample - pipeline - run -
  spec :
    pipelineRef :
      name : sample - pipeline
    params :
      - name : gitrevision
        value : $(tt.params.gitrevision)
      - name : gitrepositoryurl
        value : $(tt.params.gitrepositoryurl)]