



DevOps & CI/CD Essentials Cheatsheet



Overview

A concise yet comprehensive cheatsheet covering the **core DevOps ecosystem** — from containerization to continuous delivery. Designed to help learners, engineers, and architects quickly recall the **key concepts, commands, and workflows** that make modern infrastructure tick.



Structure

Organized into **six essential sections**:

1. Core DevOps Concepts
 2. Containers & Docker
 3. Kubernetes Basics
 4. CI/CD Pipelines
 5. GitHub Actions Overview
 6. Monitoring & Observability
-



Core DevOps Concepts

- **DevOps Philosophy:** collaboration between dev & ops for faster, reliable delivery.
- **Key Pillars:** automation, CI/CD, observability, IaC, scalability, feedback loops.
- **CI vs CD:**
 - *Continuous Integration* → merge & test code frequently.

- *Continuous Delivery/Deployment* → automatically push tested builds to production.
 - **Infrastructure as Code (IaC):** automate infra provisioning using code (Terraform, CloudFormation, Ansible).
 - **Blue-Green & Canary Deployments:** strategies for safe rollouts and testing in production.
-

◆ Containers & Docker

- **Why Containers?** Portable, lightweight, isolated environments.
- **Dockerfile Essentials:**

```
FROM python:3.10
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
CMD ["python", "main.py"]
```

- **Common Commands:**
 - `docker build -t app .` → build image
 - `docker run -p 8000:8000 app` → run container
 - `docker ps` → list running containers
 - **Volumes & Networks:** for persistent storage and inter-container communication.
 - **Docker Compose:** multi-container orchestration with a single YAML.
-

◆ Kubernetes Basics

- **Core Objects:** Pod, Deployment, Service, Namespace, ConfigMap, Secret.
- **Architecture:** Master Node (API Server, Scheduler, Controller) + Worker Nodes (Kubelet, Kube Proxy).
- **Declarative Management:** define desired state in YAML files.
- **Common Commands:** `kubectl get pods` , `kubectl apply -f deployment.yaml` , `kubectl logs pod-name`

- **Scaling:** `kubectl scale deployment app --replicas=3`
 - **Helm Charts:** package Kubernetes apps for reusability and versioning.
-

◆ CI/CD Pipeline Flow

Goal: automate build → test → deploy → monitor.

Typical Stages:

1. **Source:** code pushed to repo triggers pipeline.
2. **Build:** compile/test artifacts (e.g., Docker image).
3. **Test:** run unit/integration tests automatically.
4. **Deploy:** push to staging or production.
5. **Monitor:** track errors, latency, and user impact.

Example Pipeline (Conceptual):

Code Commit → CI Trigger → Build → Test → Deploy → Notify

- **Rollback Mechanism:** automated reversion to previous stable version on failure.
-

◆ GitHub Actions Overview

- **Purpose:** automate workflows directly in GitHub repos.
- **Key Components:**
 - `workflow.yml`: defines triggers (push, PR, schedule)
 - **jobs & steps:** run tasks sequentially or in parallel
 - **runners:** hosted or self-hosted compute agents
- **Sample workflow:**

```
name: CI Pipeline
on: [push]
jobs:
  build:
    runs-on: ubuntu-latest
```

steps:

- uses: actions/checkout@v3
- name: Install dependencies
- run: pip install -r requirements.txt
- name: Run tests
- run: pytest

- **Secrets & Environments:** store tokens and credentials securely.
-

◆ Monitoring & Observability

- **Monitoring vs Observability:**
 - *Monitoring* = collect metrics (CPU, latency, errors).
 - *Observability* = ability to understand system state via logs, metrics, traces.
 - **Common Tools:**
 - **Prometheus + Grafana** → metrics visualization
 - **ELK Stack** → centralized logging (Elasticsearch, Logstash, Kibana)
 - **Jaeger / OpenTelemetry** → distributed tracing
 - **Alerting:** integrate with Slack, PagerDuty for proactive responses.
-

🎯 Takeaway

A compact guide to help you **grasp, recall, and apply** the essential DevOps and CI/CD practices — ideal for learners, engineers, and interview prep alike.
