DevOps Cheatsheet

Essential operations for software deployment and infrastructure management

This cheatsheet provides a quick reference to fundamental DevOps operations, tools, and best practices, ideal for both beginners and experienced engineers for efficient deployment, monitoring, and infrastructure management.

Infrastructure & Configuration
Manage and provision infrastructure

CI/CD Pipelines
Automate build and deployment

Container Management
Deploy and orchestrate containers

Monitoring & Logging
Track system performance

Security & Compliance
Secure development lifecycle

Infrastructure as Code (IaC)

Terraform: Infrastructure Provisioning

Define and provide infrastructure using declarative configuration language.

```
# Initialize Terraform
terraform init
# Plan infrastructure changes
terraform plan
# Apply infrastructure changes
terraform apply
# Destroy infrastructure
terraform destroy
# Format configuration files
terraform fmt
# Validate configuration
terraform validate
```

Ansible: Configuration Management

Automate application deployment and configuration management.

```
# Run playbook
ansible-playbook site.yml
# Run playbook on specific hosts
ansible-playbook -i inventory site.yml
# Check syntax
ansible-playbook --syntax-check site.yml
# Run with specific user
ansible-playbook -u ubuntu site.yml
```

CloudFormation: AWS Native IaC

Provision AWS resources using JSON/YAML templates.

```
# Create stack
aws cloudformation create-stack --stack-name mystack --
template-body file://template.yml
# Update stack
aws cloudformation update-stack --stack-name mystack -
-template-body file://template.yml
# Delete stack
aws cloudformation delete-stack --stack-name mystack
```

Docker: Containerization

Build, ship, and run applications in containers.

```
# Build image
docker build -t myapp:latest .
# Run container
docker run -d -p 8080:80 myapp:latest
# List running containers
docker ps
# Stop container
docker stop container_id
# Remove container
docker rm container_id
```

Kubernetes: Container Orchestration

Deploy and manage containerized applications at scale.

```
# Apply configuration
kubectl apply -f deployment.yml
# Get pods
kubectl get pods
# Scale deployment
kubectl scale deployment myapp --replicas=5
# View logs
kubectl logs pod_name
# Delete resources
kubectl delete -f deployment.yml
```

Helm: Kubernetes Package Manager

Manage Kubernetes applications using charts.

```
# Install chart
helm install myrelease stable/nginx
# Upgrade release
helm upgrade myrelease stable/nginx
# List releases
helm list
# Uninstall release
helm uninstall myrelease
```

CI/CD Pipeline Management

Automate the software delivery process from code commit to production deployment.

01

Jenkins: Build Automation

Set up and manage continuous integration pipelines.

```
# Jenkinsfile example
pipeline {
  agent any
  stages {
    stage('Build') {
      steps {
        sh 'mvn clean compile'
      }
    }
    stage('Test') {
      steps {
        sh 'mvn test'
      }
    }
    stage('Deploy') {
      steps {
        sh './deploy.sh'
      }
    }
  }
}
```

02

GitHub Actions: Cloud CI/CD

Automate workflows directly from GitHub repositories.

```
# .github/workflows/ci.yml
name: CI
on: [push, pull_request]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - name: Setup Node
      uses: actions/setup-node@v2
      with:
        node-version: '14'
    - run: npm install
    - run: npm test
```

03

GitLab CI: Integrated DevOps

Use GitLab's built-in CI/CD capabilities for complete DevOps workflows.

```
# .gitlab-ci.yml
stages:
  - build
  - test
  - deploy

build_job:
  stage: build
  script:
    - echo "Building the app"

test_job:
  stage: test
  script:
    - echo "Running tests"
```

Version Control & Collaboration

Git: Version Control System

Track changes and collaborate on code development.

```
# Clone repository
git clone https://github.com/user/repo.git
# Check status
git status
# Add changes
git add .
# Commit changes
git commit -m "Add feature"
# Push to remote
git push origin main
# Pull latest changes
git pull origin main
# Create branch
git checkout -b feature-branch
# Merge branch
git merge feature-branch
```

Branch Management

Manage different development streams and releases.

```
# List branches
git branch -a
# Switch branch
git checkout main
# Delete branch
git branch -d feature-branch
# Reset to previous commit
git reset --hard HEAD~1
# View commit history
git log --oneline
```

GitHub: Code Hosting & Collaboration

Host repositories and manage collaborative development.

```
# GitHub CLI commands
gh repo create myrepo
gh repo clone user/repo
gh pr create --title "New feature"
gh pr list
gh pr merge 123
gh issue create --title "Bug report"
gh release create v1.0.0
```

Code Review & Quality

Ensure code quality through peer review and automated checks.

```
# Create pull request
git push -u origin feature-branch
# Then create PR on GitHub/GitLab

# Pre-commit hooks example
#!/bin/sh
# Run tests before commit
npm test
if [ $? -ne 0 ]; then
  echo "Tests failed"
  exit 1
fi
```

Monitoring & Observability

Prometheus: Metrics Collection

Monitor system and application metrics with time-series data.

```
# PromQL queries
# CPU usage
cpu_usage_percent{instance="server1"}
# Memory usage
(node_memory_MemTotal_bytes -
node_memory_MemAvailable_bytes) /
node_memory_MemTotal_bytes * 100
# HTTP request rate
rate(http_requests_total[5m])
# Alert rules example
ALERT HighCPUUsage
  IF cpu_usage_percent > 80
  FOR 5m
```

Grafana: Visualization Dashboard

Create dashboards and visualizations for monitoring data.

```
# Grafana API examples
# Create dashboard
curl -X POST
http://admin:admin@localhost:3000/api/dashboards/db \
  -H "Content-Type: application/json" \
  -d @dashboard.json
# Get dashboard
curl
http://admin:admin@localhost:3000/api/dashboards/uid/
dashboard-uid
```

ELK Stack: Log Management

Collect, search, and analyze log data across infrastructure.

```
# Elasticsearch queries
# Search logs
GET /logs/_search
{
  "query": {
    "match": {
      "message": "error"
    }
  }
}

# Logstash configuration
input {
  file {
    path => "/var/log/app.log"
  }
}
filter {
  grok {
    match => { "message" => "%
{TIMESTAMP_ISO8601:timestamp}" }
  }
}
output {
  elasticsearch {
    hosts => ["localhost:9200"]
  }
}
```

Application Performance Monitoring

Track application performance and user experience metrics.

```
# New Relic agent setup
# Add to application
require 'newrelic_rpm'

# Datadog metrics
from datadog import DogStatsdClient
statsd = DogStatsdClient('localhost', 8125)
statsd.increment('web.requests')
statsd.histogram('web.response_time', 0.75)
```

Cloud Platform Management

Deploy and manage applications across major cloud providers.

AWS CLI: Amazon Web Services

Interact with AWS services from command line.

```
# Configure AWS CLI
aws configure
# List EC2 instances
aws ec2 describe-instances
# Create S3 bucket
aws s3 mb s3://my-bucket-name
# Deploy Lambda function
aws lambda create-function --function-name myfunction
--runtime python3.8 --role
arn:aws:iam::123456789:role/lambda-role --handler
lambda_function.lambda_handler --zip-file
fileb://function.zip
# List running services
aws ecs list-services --cluster my-cluster
```

Azure CLI: Microsoft Azure

Manage Azure resources and services.

```
# Login to Azure
az login
# Create resource group
az group create --name myResourceGroup --location
eastus
# Create virtual machine
az vm create --resource-group myResourceGroup --name
myVM --image Ubuntu2204 --admin-username azureuser
--generate-ssh-keys
# List web apps
az webapp list
```

Google Cloud: GCP

Deploy and manage applications on Google Cloud Platform.

```
# Authenticate with GCP
gcloud auth login
# Set project
gcloud config set project my-project-id
# Deploy App Engine application
gcloud app deploy
# Create Compute Engine instance
gcloud compute instances create my-instance --zone=us-
central1-a
# Manage Kubernetes cluster
gcloud container clusters create my-cluster --num-
nodes=3
```

Multi-Cloud Management

Tools for managing resources across multiple cloud providers.

```
# Pulumi (multi-cloud IaC)
import pulumi_aws as aws
import pulumi_gcp as gcp

# Create AWS S3 bucket
bucket = aws.s3.Bucket("my-bucket")
# Create GCP storage bucket
gcp_bucket = gcp.storage.Bucket("my-gcp-bucket")
```

Security & Secrets Management

Implement security best practices and manage sensitive data securely.

HashiCorp Vault: Secrets Management

HashiCorp Vault is a tool for securely accessing secrets. A secret is anything that you want to tightly control access to, such as API keys, passwords, or certificates.

```
# Write a secret
vault kv put secret/myapp/config username=myuser
password=mypassword
# Read a secret
vault kv get secret/myapp/config
# Delete a secret
vault kv delete secret/myapp/config
# Enable auth method
vault auth enable kubernetes
# Create policy
vault policy write myapp-policy myapp-policy.hcl
```

Security Scanning: Trivy & SonarQube

Scan containers and code for security vulnerabilities.

```
# Trivy container scanning
trivy image nginx:latest
# Scan filesystem
trivy fs /path/to/project
# SonarQube analysis
sonar-scanner -Dsonar.projectKey=myproject -
Dsonar.sources=. -Dsonar.host.url=http://localhost:9000
```

SSL/TLS Certificate Management

Manage SSL certificates for secure communications.

```
# Let's Encrypt with Certbot
certbot --nginx -d example.com
# Renew certificates
certbot renew
# Check certificate expiry
openssl x509 -in cert.pem -text -noout | grep "Not After"
# Generate self-signed certificate
openssl req -x509 -newkey rsa:4096 -keyout key.pem -
out cert.pem -days 365
```

Container Security

Secure containerized applications and runtime environments.

```
# Run container as non-root user
docker run --user 1000:1000 myapp
# Scan image for vulnerabilities
docker scan myapp:latest
# Use distroless images
FROM gcr.io/distroless/java:11
COPY app.jar /app.jar
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

Performance Optimization

System Performance Monitoring

Whether you're managing servers, setting up deployments, or fixing something that just broke in production, these commands help you move faster and work smarter.

```
# CPU and memory usage
htop
# Disk usage
df -h
# Network connections
netstat -tulpn
# Process monitoring
ps aux | grep process_name
# System load
uptime
# Memory details
free -h
```

Application Performance Tuning

Optimize application performance and resource utilization.

```
# JVM performance monitoring
jstat -gc -t PID 1s
# Node.js performance
node --inspect app.js
# Database query optimization
EXPLAIN ANALYZE SELECT * FROM table WHERE
condition;
# Nginx performance tuning
nginx -t && nginx -s reload
```

Load Testing & Benchmarking

Test application performance under various load conditions.

```
# Apache Bench
ab -n 1000 -c 10 http://example.com/
# wrk HTTP benchmarking
wrk -t12 -c400 -d30s http://example.com/
# Artillery load testing
artillery run load-test.yml
# Kubernetes horizontal pod autoscaler
kubectl autoscale deployment myapp --cpu-percent=70 --
min=1 --max=10
```

Database Performance

Monitor and optimize database performance and queries.

```
# MySQL performance
SHOW PROCESSLIST;
SHOW STATUS LIKE 'Threads_connected';
# PostgreSQL monitoring
SELECT * FROM pg_stat_activity;
# Redis monitoring
redis-cli --latency
redis-cli info memory
```

DevOps Tool Installation

Install and configure essential DevOps tools and environments.

Package Managers

Install tools using system package managers.

```
# Ubuntu/Debian
apt update && apt install -y
docker.io kubectl terraform
# CentOS/RHEL
yum install -y docker
kubernetes-client terraform
# macOS Homebrew
brew install docker kubectl
terraform ansible
```

Container Runtime Installation

Set up Docker and container orchestration tools.

```
# Install Docker
curl -fsSL https://get.docker.com
| sh
systemctl start docker
systemctl enable docker
# Install Docker Compose
curl -L
"https://github.com/docker/com
pose/releases/latest/download/
docker-compose-$(uname -
s)-$(uname -m)" -o
/usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-
compose
```

Cloud CLI Tools

Install command-line interfaces for major cloud providers.

```
# AWS CLI
curl
"https://awscli.amazonaws.com/
awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
unzip awscliv2.zip &&
./aws/install
# Azure CLI
curl -sL
https://aka.ms/InstallAzureCLID
eb | bash
# Google Cloud SDK
curl
https://sdk.cloud.google.com |
bash
```

Environment Configuration

Configure development, staging, and production environments consistently.

Environment Variables Management

Manage configuration across different environments securely.

```
# .env file example
DATABASE_URL=postgresql://user:pass@localhost/db
API_KEY=your-api-key-here
ENVIRONMENT=production

# Load environment variables
export $(cat .env | xargs)
# Docker environment variables
docker run -e NODE_ENV=production -e API_KEY=secret
myapp
# Kubernetes configmap
kubectl create configmap app-config --from-env-file=.env
```

Service Discovery & Configuration

Manage service discovery and dynamic configuration.

```
# Consul service registration
consul services register myservice.json
# Get service health
consul health service web
# Etcd key-value store
etcdctl put /config/database/host localhost
etcdctl get /config/database/host
```

Development Environment Setup

Set up consistent development environments using containers.

```
# Development Dockerfile
FROM node:16-alpine
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["npm", "run", "dev"]

# Docker Compose for development
version: '3.8'
services:
  app:
    build: .
    ports:
      - "3000:3000"
    volumes:
      - .:/app
      - /app/node_modules
  database:
    image: postgres:13
    environment:
      POSTGRES_DB: myapp
```

Production Environment Hardening

Secure and optimize production environments.

```
# Systemd service configuration
[Unit]
Description=My Application
After=network.target

[Service]
Type=simple
User=myapp
WorkingDirectory=/opt/myapp
ExecStart=/opt/myapp/bin/start
Restart=always
RestartSec=5

[Install]
WantedBy=multi-user.target
```

Automation & Orchestration

Streamline DevOps workflows with automation tools and practices.

Infrastructure Automation with Ansible

Automate infrastructure provisioning and configuration management.

```
# Ansible playbook example
---
- hosts: webservers
  become: yes
  tasks:
    - name: Install nginx
      apt:
        name: nginx
        state: present
    - name: Start nginx
      service:
        name: nginx
        state: started
        enabled: yes
    - name: Deploy application
      copy:
        src: /local/app
        dest: /var/www/html
```

Workflow Orchestration

Orchestrate complex workflows and data pipelines.

```
# Apache Airflow DAG example
from airflow import DAG
from airflow.operators.bash_operator import
BashOperator
from datetime import datetime

dag = DAG('data_pipeline',
          start_date=datetime(2023, 1, 1),
          schedule_interval='@daily')

extract = BashOperator(task_id='extract_data',
                       bash_command='extract.sh',
                       dag=dag)

transform = BashOperator(task_id='transform_data',
                         bash_command='transform.sh',
                         dag=dag)

extract >> transform
```

Event-Driven Automation

Trigger automation based on system events and webhooks.

```
# GitHub webhook handler
#!/bin/bash
if [ "$1" == "push" ]; then
  git pull origin main
  docker build -t myapp .
  docker run -d --name myapp-$(date +%s) myapp
fi

# Prometheus alertmanager webhook
curl -X POST http://webhook-handler/deploy \
  -H "Content-Type: application/json" \
  -d '{"service": "myapp", "action": "restart"}'
```

ChatOps Integration

Integrate DevOps operations with chat platforms for collaborative automation.

```
# Slack bot command example
/deploy myapp to production
/rollback myapp to v1.2.3
/scale myapp replicas=5
/status infrastructure

# Microsoft Teams webhook
curl -H "Content-Type: application/json" \
  -d '{"text": "Deployment completed successfully"}' \
  $TEAMS_WEBHOOK_URL
```

Reference: This cheatsheet covers essential DevOps commands, tools, and modern practices for efficient software deployment, infrastructure management, and automation in DevOps workflows.