DevOps Final cheatsheet

# Linux commands and bash scripting

**Linux Commands**

|  |  |  |
| --- | --- | --- |
| **Command** | **Description** | **Example** |
| pwd | Print current directory | pwd |
| ls | List directory contents | ls -l |
| cd | Change directory | cd /etc |
| mkdir | Create a new directory | mkdir logs |
| touch | Create an empty file | touch file.txt |
| cp | Copy files or directories | cp a.txt b.txt |
| mv | Move or rename files | mv a.txt folder/ |
| rm | Remove files or directories | rm -r old/ |
| cat | Display file contents | cat notes.txt |
| less | View file content page by page | less bigfile.txt |
| head | Show first lines of a file | head file.txt |
| tail | Show last lines of a file | tail -n 5 file.txt |
| grep | Search for patterns in files | grep "error" log.txt |
| chmod | Change file permissions | chmod +x script.sh |
| chown | Change file ownership | chown user:user file |
| sudo | Execute command as superuser | sudo apt update |
| apt install | Install packages (Debian/Ubuntu) | sudo apt install nginx |

**Bash Scripting Basics**

**Echo**

Print text to the terminal

*echo "Hello, World!"*

**Variables**

*name="Faraz"  
echo "Hello, $name"*

**User Input**

*read name  
echo "Welcome $name"*

**Script Arguments**

Access command-line arguments

*echo "First argument: $1"  
echo "Second argument: $2"*

**If Statement**

*if [ $age -ge 18 ]; then  
 echo "Adult"  
else  
 echo "Minor"  
fi*

**For Loop**

Iterate over a list

*for i in 1 2 3; do  
 echo "Count $i"  
done*

**While Loop**

Loop while a condition is true

*i=1  
while [ $i -le 3 ]; do  
 echo $i  
 i=$((i + 1))  
done*

**Functions**

Define and call functions

*greet() {  
 echo "Hello $1"  
}  
greet Faraz*

**Check File Existence**

Verify if a file exists

*if [ -f file.txt ]; then  
 echo "File exists"  
fi*

**Loop Through Files**

Process multiple files

*for file in \*.sh; do  
 chmod +x "$file"  
done*

**Cron Job Example**

Schedule a script to run every minute

*\* \* \* \* \* /path/to/script.sh*

**Practical Script Example**

A basic script to count lines and show top 3 lines if the file exists:

*#!/bin/bash  
file=$1*if [ -f "$file" ]; then  
 echo "Lines: $(wc -l < "$file")"  
 echo "--- First 3 lines ---"  
 head -n 3 "$file"  
else  
 echo "File not found!"  
fi

# Containers, Docker and Docker compose

**Frontend/Dockerfile**

# Use official Node.js image

FROM node:18

# Set working directory inside container

WORKDIR /app

# Copy package files and install dependencies

COPY package\*.json ./

RUN npm install

# Copy the rest of the code

COPY . .

# Expose frontend port

EXPOSE 3000

# Start the frontend app

CMD ["npm", "start"]

**Backend/Dockerfile**

FROM node:18

WORKDIR /app

# Copy package files and install dependencies

COPY package\*.json ./

RUN npm install

# Copy the rest of the backend code

COPY . .

# Expose backend port

EXPOSE 5000

CMD ["node", "server.js"]

**docker-compose.yml**

version: '3.8'

services:

frontend:

build: ./frontend

ports:

- "3000:3000"

volumes:

- ./frontend:/app

depends\_on:

- backend

backend:

build: ./backend

ports:

- "5000:5000"

volumes:

- ./backend:/app

environment:

- DB\_URL=mongodb://mongo:27017/mydb

depends\_on: - mongo

mongo:

image: mongo

ports:

- "27017:27017"

volumes:

- mongo-data:/data/db

volumes:

mongo-data:

# Kubernetes, kubectl, helm, Service Mesh (Istio)

k8s-app/  
├── namespace.yaml  
├── configmap.yaml  
├── deployment.yaml  
├── service.yaml

**namespace.yml:**

apiVersion: v1

kind: Namespace

metadata:

name: demo-app

**configMap.yml:**

apiVersion: v1

kind: ConfigMap

metadata:

name: app-config

namespace: demo-app

data:

APP\_ENV: "production"

APP\_VERSION: "v1.0.0"

**deployment.yaml:**

apiVersion: apps/v1

kind: Deployment

metadata:

name: demo-deployment

namespace: demo-app

spec:

replicas: 2

selector:

matchLabels:

app: demo

template:

metadata:

labels:

app: demo

spec:

containers:

- name: demo-container

image: nginx:latest

ports:

- containerPort: 80

envFrom:

- configMapRef:

name: app-config

resources:

requests:

cpu: "200m"

memory: "100Mi"

limits:

cpu: "500m"

memory: "200Mi"

**service.yaml:**

apiVersion: v1

kind: Service

metadata:

name: demo-service

namespace: demo-app

spec:

selector:

app: demo

ports:

- port: 80

targetPort: 80

protocol: TCP

type: NodePort

**Create All:**

kubectl apply -f namespace.yaml | configmap.yaml | deployment.yaml | service.yaml

**View Resources in Namespace:**

kubectl get all -n demo-app

kubectl get configmap -n demo-app

kubectl describe deployment demo-deployment -n demo-app

kubectl describe svc demo-service -n demo-app

**Delete All:**

Kubectl delete -f namespace.yaml | configmap.yaml | deployment.yaml | service.yaml

Terraform and Cloud Deployments (Digital Ocean)

terraform-do/  
├── main.tf # Main logic (Droplets, Spaces, etc.)  
├── variables.tf # All variables used in the project  
├── outputs.tf # Outputs to display after apply  
├── terraform.tfvars # Actual values for variables

**variables.tf:**

variable "do\_token" {

type = string

description = "DigitalOcean API Token"

sensitive = true

}

variable "region" {

type = string

description = "Region for resources"

default = "nyc3"}

variable "droplet\_size" {

type = string

default = "s-1vcpu-1gb"

}

variable "droplet\_name" {

type = string

default = "my-droplet"

}

**terraform.tfvars:**

do\_token = "your\_digitalocean\_api\_token"

droplet\_name = "exam-droplet"

**main.tf:**

provider "digitalocean" {

token = var.do\_token

}

resource "digitalocean\_droplet" "web" {

name = var.droplet\_name

region = var.region

size = var.droplet\_size

image = "ubuntu-22-04-x64"

tags = ["terraform", "exam"]

}

resource "digitalocean\_spaces\_bucket" "bucket" {

name = "${var.droplet\_name}-bucket"

region = var.region

}

resource "digitalocean\_spaces\_bucket\_policy" "read\_policy" {

bucket = digitalocean\_spaces\_bucket.bucket.name

policy = jsonencode({

Version = "2012-10-17",

Statement = [{

Sid = "ReadAccess",

Effect = "Allow",

Principal = "\*",

Action = ["s3:GetObject"],

Resource = "arn:aws:s3:::${digitalocean\_spaces\_bucket.bucket.name}/\*"

}]

})

}

**outputs.tf:**

output "droplet\_ip" {

description = "Public IP of the Droplet"

value = digitalocean\_droplet.web.ipv4\_address

}

output "space\_name" {

description = "DigitalOcean Space (S3 bucket name)"

value = digitalocean\_spaces\_bucket.bucket.name

}

# Preview what will be created

terraform plan -var-file="terraform.tfvars"

# Apply the infrastructure

terraform apply -var-file="terraform.tfvars"

# Destroy everything

terraform destroy -var-file="terraform.tfvars”

CI/CD pipelines and Github Actions

your-project/  
├── .github/  
│ └── workflows/  
│ └── ci.yml  
├── app/  
│ └── index.js  
├── Dockerfile  
├── package.json

**ci.yml:**

name: CI Pipeline

on:

push:

branches: [ main ]

pull\_request:

branches: [ main ]

jobs:

build-and-test:

runs-on: ubuntu-latest

steps:

- name: ⬇️ Checkout code

uses: actions/checkout@v3

- name: 🟢 Set up Node.js

uses: actions/setup-node@v4

with:

node-version: '18'

- name: 📦 Install dependencies

run: npm install

- name: ✅ Run tests

run: npm test

**Full CI/CD Pipeline Example with Docker:**

name: CI/CD Pipeline

on:

push:

branches: [ main ]

jobs:

deploy:

runs-on: ubuntu-latest

steps:

- name: Checkout Code

uses: actions/checkout@v3

- name: Set up Docker

uses: docker/setup-buildx-action@v3

- name: Login to DockerHub

uses: docker/login-action@v2

with:

username: ${{ secrets.DOCKER\_USERNAME }}

password: ${{ secrets.DOCKER\_PASSWORD }}

- name: Build and Push Docker Image

uses: docker/build-push-action@v5

with:

context: .

push: true

tags: ${{ secrets.DOCKER\_USERNAME }}/my-app:latest

**DigitalOcean Droplet via SSH:**

name: CI/CD to DigitalOcean Droplet

on:

push:

branches: [ main ]

jobs:

deploy:

runs-on: ubuntu-latest

steps:

- name: ⬇️ Checkout code

uses: actions/checkout@v3

- name: 🔐 Set up SSH

uses: webfactory/ssh-agent@v0.9.0

with:

ssh-private-key: ${{ secrets.DO\_SSH\_PRIVATE\_KEY }}

- name: 📡 Deploy to Droplet via SSH

run: |

ssh -o StrictHostKeyChecking=no ${{ secrets.DO\_SSH\_USER }}@${{ secrets.DO\_DROPLET\_IP }} << 'EOF'

cd /root/my-app

git pull origin main

docker build -t my-app .

docker stop my-app || true && docker rm my-app || true

docker run -d -p 80:3000 --name my-app my-app

EOF

DevOps Principles (CALMS and The Three Ways)

**CALMS Framework:**

C : culture -> Collaboration > silos. Emphasis on shared responsibility and open feedback.

A: Automation -> Automate testing, builds, deployments, infra. Improves speed and consistency.

L : Lean -> Deliver small, incremental changes. Eliminate waste and rework.

M : Measurement -> Metrics-driven decisions (lead time, failure rate, MTTR, etc.)

S : Sharing -> Share knowledge, tools, and responsibilities across teams.

**The Three Ways of DevOps:**

**The First Way – Flow (Left to Right)**

- Focus: Deliver work fast from Dev to Ops.

- Optimize continuous delivery, pipelines, reduce handoffs.

- Use tools like GitHub Actions, Jenkins, Docker, etc.

Example: Automating builds & tests so code moves quickly to production.

**The Second Way – Feedback (Right to Left)**

- Focus: Create fast, constant feedback loops from Ops to Dev.

- Detect issues early, improve product quality.

- Use monitoring, logging, alerts, incident reports.

Example: Using Prometheus + Grafana to monitor services and alert Devs on failures.

**The Third Way – Continual Learning & Experimentation**

- Focus: A culture of innovation, learning from failures, and constant improvement.

- Encourage safe experimentation, blameless postmortems, and Kaizen (small improvements).

Example: After incident, team holds a retrospective and writes postmortem doc.

1st: Flow -> CI/CD, pipelines, automation, small batch sizes.

2nd: Feedback -> Monitoring, observability, error tracking, SLOs.

3rd: Learning -> Retrospectives, postmortems, innovation days.