
🚀 DEVOPS MASTER CHEAT SHEET (CDAC-ALIGNED)

0️⃣ DevOps Mental Model (Burn this in)

```
Plan → Code → Build → Test → Package → Deploy → Operate → Monitor
 |    |    |     |      |      |
 Jira  Git   Jenkins  Jenkins  Docker  K8s/AWS
```

If you can't place a tool in this flow, you don't need it yet.

1️⃣ Important Terminal Commands (Ubuntu – NON-NEGOTIABLE)

📁 File & Directory

```
pwd
ls -la
cd folder/
mkdir project
rm -rf folder
cp file1 file2
mv old new
```

📄 File Viewing & Editing

```
cat file
less file
nano file
vim file
touch file.txt
```

🔐 Permissions & Ownership

```
chmod +x script.sh
chmod 755 file
chown user:user file
```

📦 Package Management

```
sudo apt update
sudo apt upgrade
sudo apt install git docker.io
sudo systemctl start docker
sudo systemctl enable docker
```

🧠 Exam Trap

If you can’t use ls, cd, nano, chmod, **you will fail the lab**.

2️⃣ GitHub Operations (Version Control Backbone)

Entity: **GitHub**

🔄 Basic Git Flow

git init

git clone <repo_url>

git status

git add .

git commit -m "message"

git push origin main

git pull origin main

🌿 Branching

git branch feature

git checkout feature

git merge feature

🔑 Authentication (CDAC)

- HTTPS + GitHub token
- SSH optional (advanced)

🧠 MCQ Traps

- git pull = fetch + merge
- .gitignore is checked **before** add

3️⃣ Jenkins (CI Engine)

Entity: **Jenkins**

🔧 Setup Checklist

- Java installed
- Runs on http://localhost:8080
- Admin user created

🏗 Job Types

- Freestyle Project ✅ (CDAC favorite)
- Pipeline (Declarative) ⚠️ advanced

🧪 Freestyle Job Flow

1. Source Code → GitHub
2. Build Step → Shell script
3. Output → Console logs

```
echo "Build successful"
chmod +x app.sh
./app.sh
```

📜 Jenkinsfile (Know this)

```
pipeline {
 agent any
 stages {
  stage('Build') {
   steps {
    echo 'Hello CDAC'
   }
  }
 }
}
```

🧠 Exam Reality

Jenkins = **automation server**, not deployment tool.

4️⃣ Terraform (Infrastructure as Code)

Entity: **Terraform**

📦 Core Files

main.tf

variables.tf

outputs.tf

terraform.tfstate

🛠 Core Commands

```
terraform init
terraform validate
terraform plan
terraform apply
terraform destroy
```

🌍 AWS Example (Mental)

```
provider "aws" {
  region = "ap-south-1"
}
```

🧠 **Trap**

- Terraform **creates infra**, doesn't configure software → that's Ansible's job.

5️⃣ Ansible (Configuration Management)

Entity: **Ansible**

📁 **Key Components**

- Inventory
- Playbook
- Modules

📜 **Inventory**

```
[servers]
192.168.1.10
```

▶ **Playbook**

```
- hosts: servers
  become: yes
  tasks:
    - name: Install nginx
      apt:
        name: nginx
        state: present
```

▶ **Run**

```
ansible-playbook playbook.yml
```

🧠 **Exam Gold**

- Agentless
- Uses SSH
- YAML-based

6️⃣ Docker (Containerization King)

Entity: **Docker**

🐳 **Dockerfile**

```
FROM ubuntu
```

RUN apt update

CMD ["echo", "Hello DevOps"]

🔨 Commands

docker build -t myapp .

docker run myapp

docker ps

docker images

docker stop <id>

🧠 Reality Check

Docker replaces “works on my machine” excuses.

7️⃣ Kubernetes (Container Orchestration)

Entity: **Kubernetes**

📦 Core Objects

- Pod
- Deployment
- Service

🚀 Commands

kubectl get pods

kubectl apply -f deploy.yml

kubectl describe pod podname

kubectl delete pod podname

🧠 CDAC Truth

- Mostly **theory + demo**
- Don’t overbuild unless asked

8️⃣ AWS (Cloud Backbone)

Entity: **Amazon Web Services**

🧱 Core Services You MUST Know

- EC2 → Compute
- S3 → Storage
- IAM → Security
- VPC → Networking

🔑 IAM Golden Rules

- Never use root
- Least privilege
- Roles > access keys

🧠 Exam Focus

IAM + EC2 + S3 = 80% of AWS MCQs.

9️⃣ Jira (Project & Task Tracking)

Entity: **Jira**

📋 Concepts

- Project
- Issue
- Sprint
- Board

🔄 Workflow

To Do → In Progress → Done

🧠 Reality

- Jira ≠ DevOps tool
- Jira = **DevOps enabler**

✅ CDAC DEVOPS – ACTUAL TASK EXECUTION PLAYBOOK

(GitHub → Jenkins → Terraform → Docker → AWS)

1️⃣ GitHub – Version Control Assignment (FOUNDATION)

Entity: **GitHub**

🎯 Objective (What CDAC wanted)

- Create a repository
- Perform basic Git operations from Ubuntu
- Push code successfully to GitHub

🧪 Steps You Performed (Exact Order)

🔹 Step 1: Install Git (Ubuntu VM)

sudo apt update

sudo apt install git -y

git --version

🔹 **Step 2: Configure Git**

git config --global user.name "Renold"

git config --global user.email "your_email@gmail.com"

Verify:

git config --list

🔹 **Step 3: Create GitHub Repository**

- Login → GitHub
- New Repository
- Public
- **NO README** (important for lab)

🔹 **Step 4: Clone Repository**

git clone https://github.com/<username>/<repo>.git

cd <repo>

🔹 **Step 5: Add Files & Commit**

touch app.sh

nano app.sh

#!/bin/bash

echo "Hello CDAC DevOps"

chmod +x app.sh

git add .

git commit -m "Initial commit"

🔹 **Step 6: Push to GitHub**

git branch -M main

git push origin main

✔ Repo populated
✔ Commit visible
✔ GitHub authenticated

🧠 Lab Mistake You Fixed

❌ *“fatal: not a git repository”*
✓ Fixed by running Git commands **inside cloned repo**

2️⃣ Jenkins – CI Automation Assignment

Entity: **Jenkins**

🎯 Objective

- Install Jenkins
- Connect Jenkins to GitHub
- Run automated build

🧪 Steps You Performed

🔹 Step 1: Install Java

sudo apt install openjdk-17-jdk -y

java -version

🔹 Step 2: Install Jenkins

wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -

sudo sh -c 'echo deb http://pkg.jenkins.io/debian binary/ > /etc/apt/sources.list.d/jenkins.list'

sudo apt update

sudo apt install jenkins -y

🔹 Step 3: Start Jenkins

sudo systemctl start jenkins

sudo systemctl enable jenkins

Access:

http://localhost:8080

🔹 Step 4: Initial Setup

- Unlock Jenkins (initialAdminPassword)
- Install suggested plugins
- Create admin user

✓ Dashboard loaded
✓ Jenkins running on 8080

🔹 Step 5: Create Freestyle Job

- New Item → Freestyle Project
- Source Code → Git
- Repo URL → GitHub repo

🔹 Step 6: Build Step

chmod +x app.sh

./app.sh

🔹 Step 7: Build Now

✓ Console output success
✓ Job green
✓ GitHub → Jenkins automation verified

🧠 CDAC Focus

Jenkins **does not deploy**, it **automates tasks**.

3️⃣ Terraform – Infrastructure as Code Assignment

Entity: **Terraform**

🎯 Objective

- Provision AWS resources using code
- Demonstrate IaC concept

🧪 Steps You Performed

🔹 Step 1: Install Terraform

sudo apt install terraform -y

terraform -version

🔹 Step 2: Create Project

mkdir terraform-lab

cd terraform-lab

nano main.tf

🔹 **Step 3: AWS Provider**

provider "aws" {

  region = "ap-south-1"

}

🔹 **Step 4: Initialize Terraform**

terraform init

terraform validate

terraform plan

🔹 **Step 5: Apply Infrastructure**

terraform apply

✓ AWS resource created
✓ State file generated
✓ Infra reproducible

🔹 **Step 6: Cleanup**

terraform destroy

🧠 **Key CDAC Line**

Terraform = **create infra**, not manage software.

4️⃣ **Docker – Containerization Assignment**

Entity: **Docker**

🎯 **Objective**

- Containerize application
- Run container successfully

🧪 **Steps You Performed**

🔹 **Step 1: Install Docker**

sudo apt install docker.io -y

sudo systemctl start docker

sudo systemctl enable docker

🔹 **Step 2: Dockerfile**

nano Dockerfile

FROM ubuntu

RUN apt update

CMD ["echo", "Hello Docker CDAC"]

🔹 **Step 3: Build Image**

docker build -t cdac-docker .

🔹 **Step 4: Run Container**

docker run cdac-docker

✔ Output printed
✔ Container exited successfully

🧠 **Exam Trap**

Container ≠ Image
Image = blueprint

5️⃣ **AWS – IAM + EC2 + Role Assignment**

Entity: **Amazon Web Services**

🎯 **Objective**

- Secure AWS usage
- IAM + EC2 access via roles

🧪 **Steps You Performed**

🔹 **Step 1: IAM User Creation**

- No root usage
- Created devops-engineer-user
- Programmatic + Console access

🔹 **Step 2: Assign Policies**

- EC2FullAccess

- S3FullAccess (if mentioned)
- IAMReadOnlyAccess

🔹 **Step 3: EC2 Instance**

- Instance type: t3.micro
- Region: ap-south-1
- Key pair created

🔹 **Step 4: SSH into EC2**

```
ssh -i devops-lab-key.pem ec2-user@<public-ip>
```

🔹 **Step 5: Role-Based Access**

```
aws sts get-caller-identity
```

✓ Role assumed
✓ No access keys stored
✓ Secure practice confirmed

🧠 **CDAC Emphasis**

IAM Roles > Access Keys
Root user = NEVER
