DevOps Deployment Tools Cheatsheet

A comprehensive cheat sheet covering essential DevOps and Cloud deployment tools, their functionalities, and usage, with practical examples.

Ansible

Core Concepts

Ansible: An open-source automation tool used for configuration management, application deployment, task automation, and IT orchestration.

Playbooks: YAML files that define the tasks to be executed on managed nodes.

Inventory: A list of managed nodes (hosts) that Ansible manages, typically defined in a file.

Modules: Reusable, standalone scripts that Ansible uses to perform tasks on managed nodes.

Roles: A way to organize and reuse Ansible playbooks. Roles group together related tasks, variables, and handlers.

Common Commands

`ansible --version`	Check the Ansible version.
`ansible all -m ping -i inventory`	Ping all hosts in the inventory file.
`ansible-playbook playbook.yml -i inventory`	Run an Ansible playbook against the inventory.
`ansible-galaxy install role_name`	Install a role from Ansible Galaxy.
`ansible-vault encrypt file.yml`	Encrypt a file using Ansible Vault.

Example Playbook

```yaml
---
- hosts: webservers
  become: true
  tasks:
    - name: Ensure Apache is installed
      apt:
        name: apache2
        state: present
    - name: Ensure Apache is running
      service:
        name: apache2
        state: started
```

Terraform

Key Concepts

Terraform: An infrastructure as code (IaC) tool that enables you to define and provision infrastructure using a declarative configuration language.

Providers: Plugins that allow Terraform to interact with different infrastructure platforms (e.g., AWS, Azure, GCP).

Resources: Components of your infrastructure, such as virtual machines, networks, and databases.

Modules: Reusable and composable units of Terraform configuration, similar to functions in programming.

State: Terraform uses a state file to track the current configuration of your infrastructure.

Common Commands

`terraform init`	Initialize a Terraform working directory.
`terraform plan`	Show changes required by the current configuration.
`terraform apply`	Apply the changes to the infrastructure.
`terraform destroy`	Destroy the infrastructure managed by Terraform.
`terraform show`	Inspect the current Terraform state.

Example Configuration

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.0"
    }
  }
}

provider "aws" {
  region = "us-west-2"
}

resource "aws_instance" "example" {
  ami           = "ami-0c55b9479a3c8c88c" # Example AMI
  instance_type = "t2.micro"

  tags = {
    Name = "ExampleInstance"
  }
}
```

Kubernetes Deployments

Core Components

Deployment: Manages the desired state of your application by ensuring the specified number of replicas are running.

Pod: The smallest deployable unit in Kubernetes, representing a single instance of a running process.

Service: An abstraction that defines a logical set of Pods and a policy by which to access them.

Namespace: A way to divide cluster resources between multiple users or teams.

Ingress: Manages external access to the services in a cluster, typically via HTTP.

Common kubectl Commands

`kubectl apply -f deployment.yaml`	Apply a configuration file to create or update resources.
`kubectl get deployments`	List all deployments in the current namespace.
`kubectl describe deployment <deployment-name>`	Show detailed information about a deployment.
`kubectl scale deployment <deployment-name> --replicas=<number>`	Scale the number of replicas in a deployment.
`kubectl delete deployment <deployment-name>`	Delete a deployment.

Example Deployment YAML

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Jenkins

Key Features

Jenkins: An open-source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery.

Pipelines: Jenkins pipelines allow you to define your entire build, test, and deployment process as code.

Plugins: Jenkins has a wide variety of plugins available to extend its functionality, such as integrations with source control systems, build tools, and deployment platforms.

Jobs: Automated tasks or series of tasks defined within Jenkins to perform specific actions such as building or deploying applications.

Nodes/Agents: Machines or containers that Jenkins uses to execute build jobs.

Pipeline Syntax

`pipeline { ... }`	Defines the overall pipeline structure.
`agent { ... }`	Specifies where the pipeline will execute (e.g., any node, a specific label).
`stages { ... }`	Defines the different stages of the pipeline.
`steps { ... }`	Contains the actual commands to execute in each stage.
`post { ... }`	Defines actions to be performed after the pipeline, regardless of the outcome.

Example Jenkinsfile

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                echo 'Building...'
            }
        }
        stage('Test') {
            steps {
                echo 'Testing...'
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploying...'
            }
        }
    }
}
```