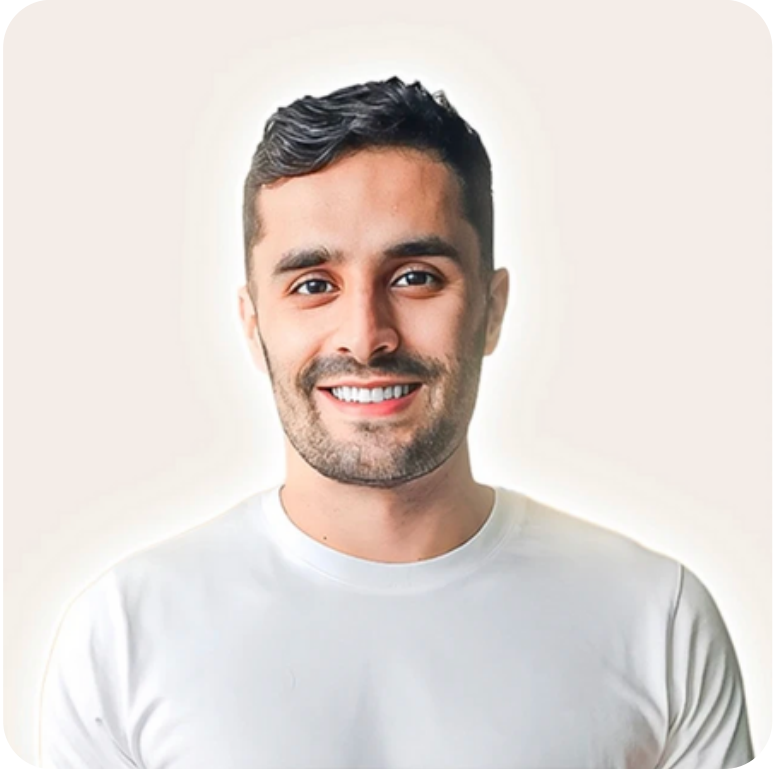
CLOUD ENGINEERING 101: A BEGINNER'S GUIDE TO MASTERING THE BASICS

Welcome to the Cloud Engineer Handbook, your comprehensive guide to mastering the essentials of cloud computing. This resource is tailored for beginners and aspiring cloud professionals, providing a solid foundation in the key areas of modern cloud engineering.



SOLEYMAN SHAHIR

TECH ENTREPRENEUR

With a decade in tech, I've worked in Software and Cloud Engineering for diverse organizations globally. I've architected major AWS solutions and transitioned to expert cloud consulting.

I now run a Specialized Consulting Business, automating processes across the Middle East and EU using AI, Software, and Cloud Technologies.

I also manage Tech with Soleyman, the leading Cloud Engineering YouTube channel.

Cloud Engineering has accelerated my growth through hands-on learning and exposure to diverse tools. With AI's rise driving cloud adoption, demand for Cloud Engineers is soaring.

Cloud Engineers design and manage cloud systems, offering an accessible entry point into tech. In this role, you'll build the infrastructure shaping our future.

TABLE OF CONTENTS

CHAPTER 1

CLOUD FUNDAMENTALS

UNDERSTANDING CLOUD COMPUTING

At it’s core, Cloud Computing is the delivery of computing services over the internet. Instead of running applications or storing data on your local computer or server, you access these resources remotely through the cloud.

This allows for greater flexibility, scalability, and cost-efficiency. Imagine you're using Instagram. When you open the app, you can seamlessly browse through posts, like photos, and upload your own content.

But have you ever wondered where all this data is stored and processed? This is all on the cloud.

Cloud computing is like having a massive, powerful computer that you can access from anywhere with an internet connection.
Instead of running Instagram on your phone's limited resources, it runs on servers in the cloud, allowing millions of users to access it simultaneously.

Just like how Instagram uses different services to function, cloud computing has three main service models:

1. Infrastructure as a Service (IaaS)
2. Platform as a Service (PaaS)
3. Software as a Service (SaaS)

INFRASTRUCTURE AS A SERVICE

Think of IaaS as the foundation of Instagram. It includes the servers, storage, and networks that power the app.

IaaS providers, like Amazon EC2, offer these basic building blocks, allowing companies to rent and configure them as needed.

PLATFORM AS A SERVICE

PaaS is like the tools and frameworks used to build and run Instagram.

It provides a platform for developers to create, test, and deploy the app without worrying about managing the underlying infrastructure.

Services like AWS Elastic Beanstalk fall under this category.

SOFTWARE AS A SERVICE

SaaS is what you experience as an end-user of Instagram. It's fully developed software accessible through a web browser or app.

You don't need to install anything on your device, you simply log in and start using it.

Examples of SaaS is Microsoft Office 365.

MAIN CLOUD PROVIDERS

Just like there are different social media platforms, there are three main cloud providers in the market.

AWS is like the Facebook of cloud providers - it's the largest and most popular.

Many companies, including Netflix and Airbnb, rely on AWS for their cloud needs.

Azure is like LinkedIn - it's owned by Microsoft and integrates well with their other products.

If you're using Office 365 or Windows, Azure might be a good fit.

GCP is like YouTube - it's known for its strengths in big data and machine learning.

Just as YouTube handles massive amounts of video data, GCP excels in processing and analyzing large datasets.

CLOUD COMPUTING BENEFITS

The key benefits of cloud computing are scalability, cost efficiency, flexibility, reliability and focusing on core business functionality.

Scalability:
Imagine if Instagram couldn't handle a sudden surge in users. With cloud computing, they can easily scale up their resources to accommodate increased demand, just like adding more servers to handle the load.

Cost-efficiency:
Building and maintaining your own servers can be expensive. Cloud computing allows companies to pay only for what they use, like renting a car instead of buying one.

Flexibility:
Just like how you can access Instagram from your phone, tablet, or computer, cloud computing enables you to access resources and services from anywhere with an internet connection.

Reliability:
Cloud providers ensure that Instagram is always available, even if a server fails.
They have backup systems and failover mechanisms in place, like having multiple paths to reach your destination.

Focusing on Core Business:
By using cloud services, Instagram can focus on improving the app and user experience instead of worrying about managing servers and infrastructure.

CHAPTER 2

CORE CLOUD SERVICES

CORE CLOUD SERVICES

As a Cloud Engineer, understanding the core cloud services is important. These services form the building blocks of any cloud infrastructure, and mastering them is essential for designing, implementing, and managing scalable and resilient cloud solutions.

COMPUTE SERVICES

Compute services are the backbone of any cloud infrastructure.
Imagine you're a builder constructing houses.
Compute services are like the power tools you use to get the job done.
Just as you need different tools for different tasks (e.g., a drill for making holes, a saw for cutting wood), compute services provide various options to run your applications and workloads in the cloud.
They provide the processing power needed to run applications, services, and workloads in the cloud.

In AWS, the main compute service is Amazon Elastic Compute Cloud (EC2).
EC2 allows you to rent virtual machines (instances) in the cloud.

You can choose from a wide range of instance types, each with different configurations of CPU, memory, storage, and networking capacity.

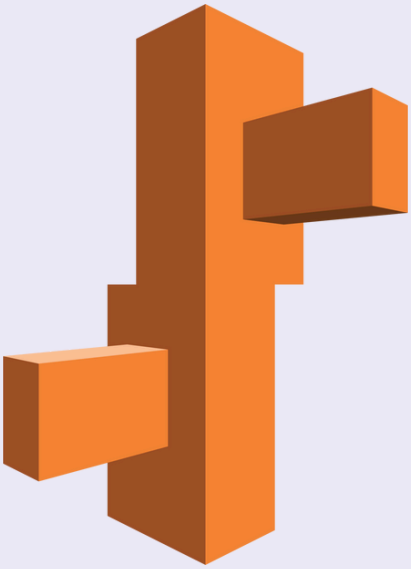
With EC2, you can easily scale your compute resources up or down based on your application's needs.

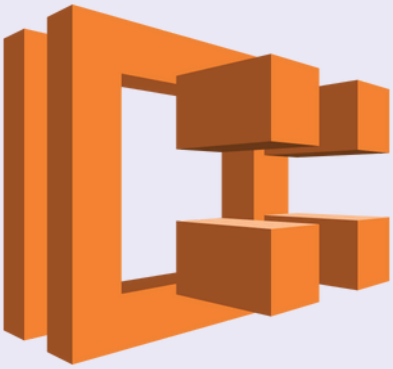
COMPUTE SERVICES

Other compute services in AWS include:

AWS Lambda: Serverless computing platform for running code without provisioning or managing servers

AWS Elastic Beanstalk: Platform for easily deploying and managing web applications





Amazon ECS (Elastic Container Service): Fully managed container orchestration service

STORAGE SERVICES

Think of storage services as the containers where you store all your tools, materials, and finished products.

In the cloud, storage services allow you to store and retrieve data.

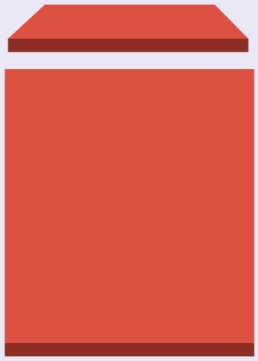
AWS offers a variety of storage services tailored for different use cases and data types.

The most widely used storage service in AWS is Amazon Simple Storage Service (S3).
S3 is an object storage service that allows you to store and retrieve any amount of data from anywhere on the web.

It's highly scalable, durable, and secure, making it ideal for storing static website files, media files, backups, and more.

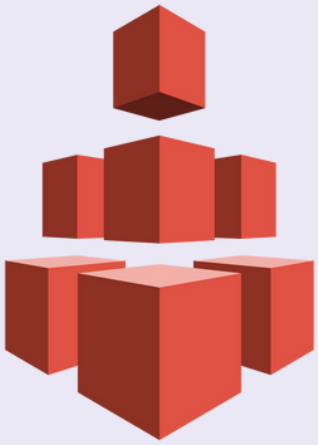
STORAGE SERVICES

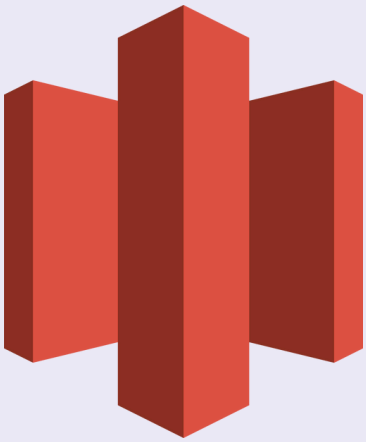
Other storage services in AWS include:



Amazon EBS (Elastic Block Store): Persistent block-level storage volumes for use with EC2 instances

Amazon EFS (Elastic File System): Scalable file storage for use with EC2 instances and on-premises resources





Amazon Glacier: Low-cost storage service for data archiving and long-term backup

NETWORKING SERVICES

Networking services are like the roads and highways that connect your construction sites (cloud resources) and allow traffic (data) to flow between them.

They enable communication and connectivity within your cloud environment.

In AWS, the primary networking service is Amazon Virtual Private Cloud (VPC).

A VPC is a virtual network dedicated to your AWS account. It allows you to launch AWS resources, such as EC2 instances, in a logically isolated virtual network that you define.

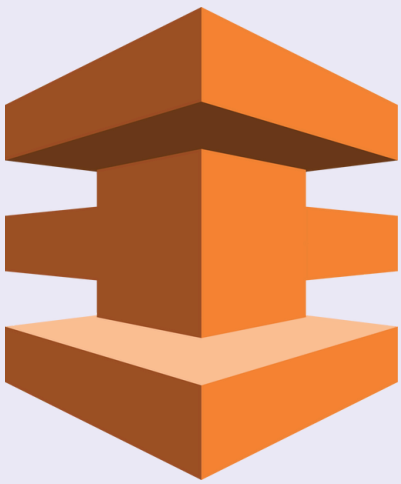
You have complete control over your virtual networking environment, including IP address ranges, subnets, route tables, and network gateways.

NETWORKING SERVICES

Other networking services in AWS include:

Amazon Route 53: Scalable domain name system (DNS) web service

Elastic Load Balancing: Automatically distributes incoming application traffic across multiple targets



AWS Direct Connect: Dedicated network connection from your premises to AWS

DATABASE SERVICES

Database services are like organized filing cabinets where you store and manage important information about your construction projects.

They provide structured and efficient ways to store, retrieve, and manage data in the cloud.

AWS offers a wide range of database services to cater to different data models and use cases.

The most popular database service in AWS is Amazon Relational Database Service (RDS).

RDS is a managed service that makes it easy to set up, operate, and scale a relational database in the cloud. It supports popular database engines like MySQL, PostgreSQL, Oracle, and Microsoft SQL Server.

DATABASE SERVICES

Other database services in AWS include:

Amazon DynamoDB: Fully managed NoSQL database service

Amazon Aurora: MySQL and PostgreSQL-compatible relational database built for the cloud

Amazon Redshift: Fast, simple, and cost-effective data warehousing service

CHAPTER 3

GIT & VERSION CONTROL

GIT & VERSION CONTROL

As a Cloud Engineer, you'll work on complex projects with multiple moving parts and collaborators.

Without proper version control, it's like trying to build a house without a blueprint.

This is where Git and Version Control come into play, helping you manage your codebase, collaborate effectively, and maintain a smooth software development lifecycle.

Picture this: you're working on a critical cloud project with a tight deadline. Your team is spread across different time zones, each working on their piece of the puzzle.

In the old days, you'd be emailing code snippets back and forth, never quite sure who had the latest version.

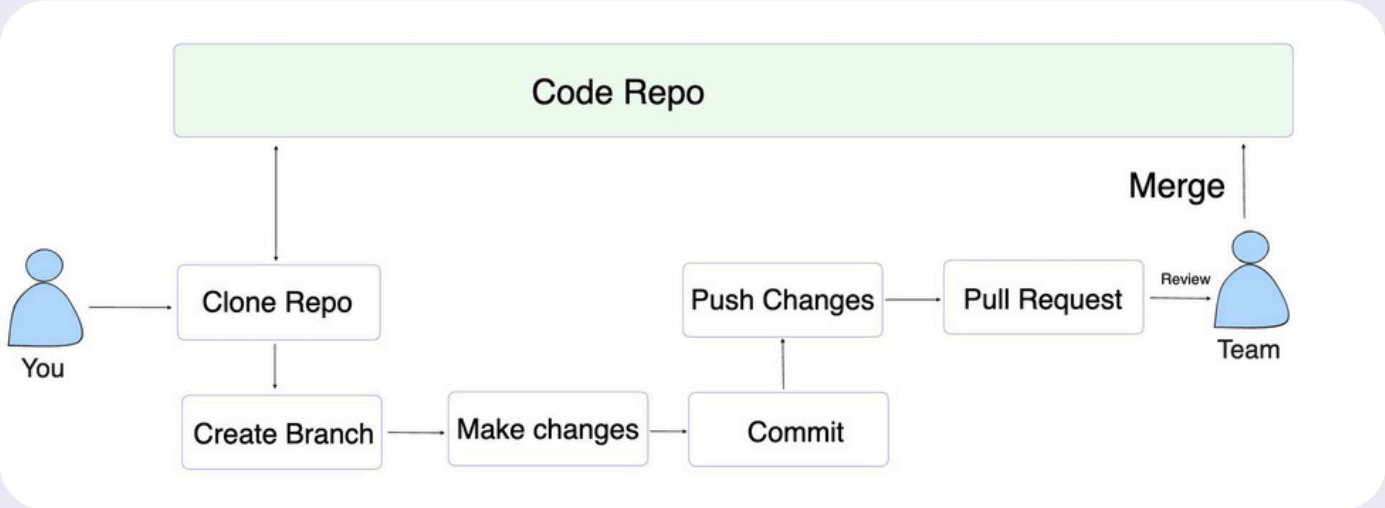
But with Git, it's like having a magic wand to keep everyone in sync.
Git is a distributed version control system that allows you to track changes in your codebase over time.

It's like having a time machine for your code, enabling you to go back to previous versions, experiment with new features, and collaborate with others without the fear of losing your work.

Now, imagine you're working on a cloud project with a team of engineers. Each engineer is responsible for different parts of the codebase, such as infrastructure scripts, application code, or configuration files.

With Git, each team member can work independently on their tasks, making changes to their local copy of the codebase.

TYPICAL GIT WORKFLOW



1. **Clone the repository**: You create a local copy of the central codebase on your machine.

2. **Create a branch**: You create a separate branch to work on a specific feature or bug fix, isolating your changes from the main codebase.

3. **Make changes**: You modify the code, add new files, or delete existing ones in your local branch.

4. **Commit changes**: You save your changes along with a descriptive message, creating a snapshot of your work at a specific point in time.

5. **Push changes**: You upload your local branch and its commits to the central repository, making your work available to others.

6. **Create a pull request**: You propose your changes to be merged into the main codebase, requesting a review from your team members.

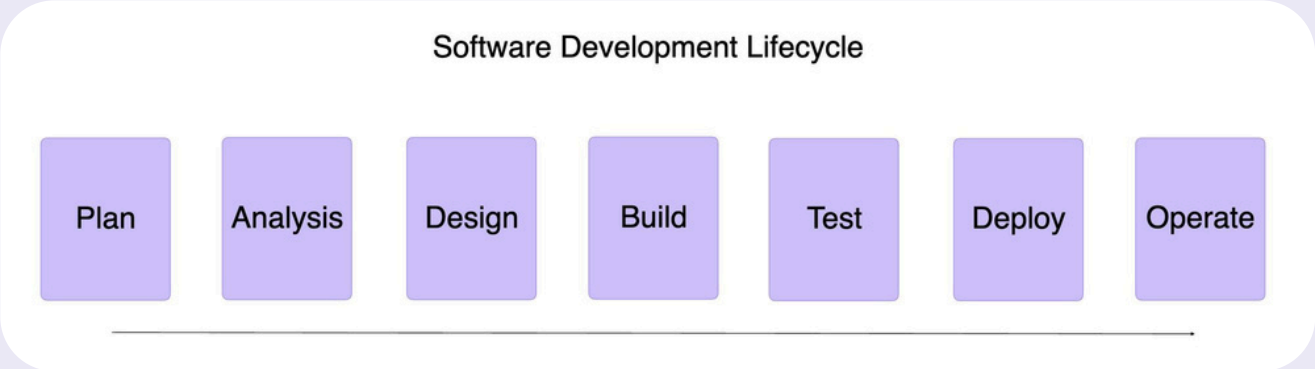
7. **Merge changes**: After reviewing and approving the changes, your branch is merged into the main codebase, becoming part of the official project.

In essence, Git brings order to the chaos of teamwork. And as a Cloud Engineer, that's a game-changer.

SOFTWARE DEVELOPMENT LIFECYCLE

But Git is just the beginning. To truly utilise its power, you need to understand the bigger picture: the Software Development Lifecycle (SDLC).

The Software Development Lifecycle (SDLC) is a structured process that describes the stages involved in developing, deploying, and maintaining software applications.



As a Cloud Engineer, understanding the SDLC is crucial, as it helps you plan, execute, and deliver cloud projects effectively.

A typical SDLC consists of the following stages:
Planning: Defining project goals and requirements
Analysis: Gathering detailed specifications
Design: Creating system architecture and components
Build: Writing code and configuring cloud infrastructure
Testing: Identifying and fixing bugs
Deployment: Releasing the solution to production
Operate: Monitoring and maintaining the live system

GIT & VERSION CONTROL

Git and Version Control integrate seamlessly into each stage of the SDLC. They provide a safety net, allowing you to move fast without breaking things.

By using Git, you can ensure code integrity, maintain a clear audit trail, and easily roll back to previous versions if needed.

Git integrates seamlessly with popular cloud platforms like AWS, enabling you to automate deployments, perform continuous integration and continuous deployment (CI/CD), and streamline your development workflow.

As you progress in your cloud engineering journey, mastering Git and understanding the SDLC will become second nature.

They are essential tools in your arsenal, empowering you to build robust, scalable, and maintainable cloud solutions.

Remember, becoming a great Cloud Engineer isn't just about technical skills. It's about adopting the right mindset and workflows that enable you to build amazing things.

Git and Version Control are your superpowers.
They give you the confidence to experiment, the ability to collaborate, and the power to deliver world-class cloud solutions.

CHAPTER 4

AWS FUNDAMENTALS

AWS FUNDAMENTALS

As a future Cloud Engineer, AWS (Amazon Web Services) will be your playground, your toolkit, and your launchpad to build world-class cloud solutions.
But with so many services and concepts to grasp, it can feel overwhelming at first.

To understand AWS, let's start with the basics.
AWS is a cloud platform offering over 200 fully-featured services from data centers globally.
It provides everything you need to build and run applications in the cloud, from computing power and storage to databases and machine learning.

REGIONS AND AVAILABILITY ZONES (AZS)

An AWS Region is a geographical location with multiple AZs.
Each AZ consists of one or more data centers equipped with independent power, cooling, and networking.
This design ensures high availability and fault tolerance for your applications.

AWS SHARED RESPONSIBILITY MODEL

This model defines the responsibilities of AWS and you, the customer, in securing and managing the cloud environment.
AWS is responsible for the "security of the cloud" (hardware, software, and infrastructure), while you are responsible for the "security in the cloud" (data, applications, and access management).

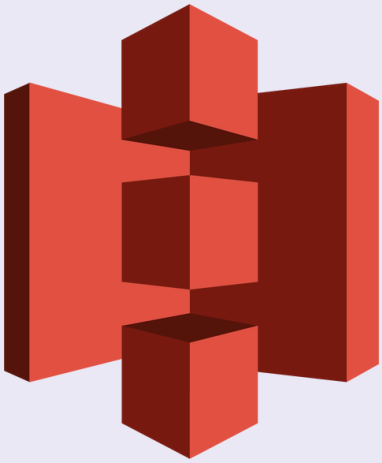
AWS FUNDAMENTALS

Now that you understand the foundational concepts let's dive into the core services that form the building blocks of any AWS solution.

AMAZON EC2 (ELASTIC COMPUTE CLOUD)

EC2 provides resizable computing capacity in the cloud.
You can launch virtual servers (instances) with various configurations of CPU, memory, storage, and networking.
EC2 allows you to scale your compute resources up or down based on demand.

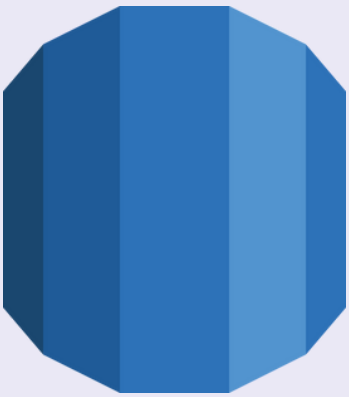
AMAZON S3 (SIMPLE STORAGE SERVICE)



S3 is an object storage service offering industry-leading scalability, data availability, security, and performance.
You can store and retrieve any amount of data from anywhere on the web.

AWS FUNDAMENTALS

AMAZON RDS (RELATIONAL DATABASE SERVICE)



RDS makes it easy to set up, operate, and scale a relational database in the cloud.

It supports popular database engines like MySQL, PostgreSQL, and Oracle, automating time-consuming tasks like hardware provisioning and database setup.

AMAZON VPC (VIRTUAL PRIVATE CLOUD)

VPC lets you provision a logically isolated section of the AWS cloud where you can launch resources in a virtual network you define.

You have complete control over your virtual networking environment, including IP ranges, subnets, and security groups.

AWS FUNDAMENTALS

Learning AWS can seem daunting, but with the right guidance and hands-on practice, you can master the fundamentals and start building impressive cloud solutions in no time.

Remember, mastering AWS is not just about memorizing services and concepts.
It's about developing a cloud-first mindset, understanding how to architect scalable and secure solutions, and continuously expanding your knowledge in this fast-evolving field.

AWS is your launchpad to a thriving career as a Cloud Engineer.
By investing in your AWS skills today, you're setting yourself up for a future of endless opportunities and growth.

So keep learning, keep building, and most importantly, keep challenging yourself to reach new heights in the cloud.

CHAPTER 5

TERRAFORM

TERRAFORM

A tool that will revolutionize the way you manage infrastructure: Terraform. As a Cloud Engineer, you know that manually provisioning and managing resources can be time-consuming, error-prone, and difficult to scale. That's where Terraform comes in – it allows you to define and manage your infrastructure as code, enabling you to build, change, and version your resources with ease.
First, let's understand what Terraform is and why it matters.

WHAT IS TERRAFORM?

Terraform is an open-source Infrastructure as Code (IaC) tool created by HashiCorp.

It enables you to define and provision infrastructure resources across various cloud providers (AWS, Azure, GCP) and services using a declarative language called HashiCorp Configuration Language (HCL).

With Terraform, you write code that describes your desired infrastructure state, and Terraform takes care of creating, modifying, or deleting resources to match that state.

This approach has several benefits:

1. **Consistency**: You can define your infrastructure once and deploy it consistently across different environments (dev, staging, production).
2. **Versioning**: You can version your infrastructure code, track changes, and collaborate with your team using version control systems like Git.
3. **Reusability**: You can create reusable modules and share them across projects, promoting best practices and reducing duplication.

TERRAFORM CONCEPTS

```
# EC2 Resource
resource "aws_instance" "my-ec2" {
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t2.micro"

  tags = {
    Name = "example-instance"
  }
}
  # Data Source
data "aws_ami" "example" {
  most_recent = true
  owners      = ["amazon"]

  filter {
    name   = "name"
    values = ["amzn2-ami-hvm-*"]
  }
}

resource "aws_instance" "example" {
  ami           = data.aws_ami.example.id
  instance_type = "t2.micro"
}

# Module
module "vpc" {
  source = "terraform-aws-modules/vpc/aws"

  name = "example-vpc"
  cidr = "10.0.0.0/16"

  azs             = ["us-west-2a", "us-west-2b", "us-west-2c"]
  private_subnets = ["10.0.1.0/24", "10.0.2.0/24", "10.0.3.0/24"]
  public_subnets  = ["10.0.101.0/24", "10.0.102.0/24", "10.0.103.0/24"]

  enable_nat_gateway = true
  enable_vpn_gateway = true
}
```

Provider: A provider is a plugin that enables Terraform to interact with a specific cloud provider or service (e.g., AWS, Azure, GCP).
Resource: A resource is an infrastructure component you want to manage with Terraform (e.g., EC2 instance, S3 bucket, VPC).
Data Source: A data source allows Terraform to fetch information from a provider and use it in your configuration (e.g., AMI IDs, availability zones).
Module: A module is a self-contained package of Terraform configurations that can be reused across projects.
State: Terraform maintains a state file that keeps track of the current state of your infrastructure. It uses this state to determine what changes need to be made when you modify your configuration.

TERRAFORM COMMANDS

`terraform init`

Initializes a new or existing Terraform working directory, downloading the necessary provider plugins and modules.

`terraform plan`

Creates an execution plan, showing you what changes Terraform will make to your infrastructure based on your configuration.

`terraform apply`

Applies the changes described in the execution plan, provisioning or modifying your infrastructure resources.

`terraform destroy`

Destroys the resources managed by your Terraform configuration, cleaning up your infrastructure.

`terraform validate`

Validates your Terraform configuration files for syntax and consistency.

TERRAFORM

Learning Terraform can seem overwhelming at first, but with the right guidance and hands-on practice, you can master the art of Infrastructure as Code and take your cloud engineering skills to the next level.

Remember, adopting Infrastructure as Code is not just about learning a new tool.

It's about embracing a new way of thinking about infrastructure management – one that emphasizes automation, collaboration, and scalability.

Terraform is your key to unlocking the full potential of IaC.
By mastering Terraform, you'll be able to deliver infrastructure faster, more consistently, and with less risk.

You'll become an indispensable asset to any organization looking to modernize its infrastructure practices.

So keep learning, keep coding, and most importantly, keep challenging yourself to think in terms of infrastructure as code.

CHAPTER 6

DEVOPS

DEVOPS

DevOps is a software development approach that emphasizes collaboration, automation, and integration between development and operations teams. It aims to streamline the entire software development lifecycle, from code creation to deployment and maintenance.

In traditional IT organizations, development and operations teams often work in silos, with different goals and priorities.
Developers focus on creating new features and functionality, while operations teams prioritize stability and reliability.
This disconnect can lead to delays, conflicts, and inefficiencies.

DevOps breaks down these silos by fostering a culture of collaboration and shared responsibility.
It encourages developers and operations teams to work together throughout the entire software development process, leveraging automation and tools to increase efficiency and reduce erro

DEVOPS PILLARS

To understand DevOps, let's explore its key pillars:

Continuous Integration (CI):

CI is a practice where developers frequently merge their code changes into a central repository, such as Git.
Automated build and test processes are triggered each time code is pushed, ensuring that any issues or conflicts are caught early.
CI helps maintain code quality and reduces integration problems.

Continuous Delivery/Deployment (CD):

CD takes CI a step further by automatically deploying code changes to production environments.
With continuous delivery, every change that passes the automated tests is ready to be deployed to production.
Continuous deployment goes even further by automatically deploying every change that passes the tests to production without manual intervention.
CD enables faster and more frequent releases, reducing the risk and effort involved in manual deployments.

Infrastructure as Code (IaC):

IaC is the practice of managing and provisioning infrastructure resources using code and automation tools.
Instead of manually configuring servers, networks, and other infrastructure components, IaC allows you to define your infrastructure as code, which can be version-controlled, tested, and deployed just like any other software.
Tools like Terraform, AWS CloudFormation, and Ansible are commonly used for IaC.

Monitoring and Logging:

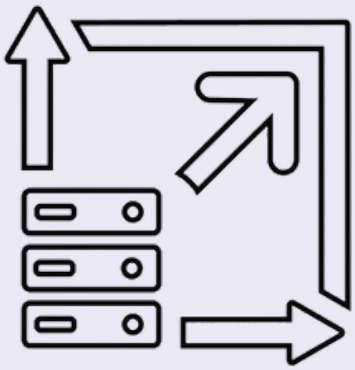
DevOps emphasizes the importance of monitoring and logging to gain visibility into the health and performance of applications and infrastructure.
By collecting and analyzing metrics, logs, and traces, teams can quickly detect and resolve issues, optimize resources, and ensure a smooth user experience.
Tools like AWS CloudWatch, Prometheus, and ELK stack are widely used for monitoring and logging in DevOps environments.

BENEFITS OF DEVOPS

Faster Time-to-Market

Improved Collaboration

Increased Reliability and Stability



Scalability and Flexibility

GETTING STARTED WITH DEVOPS

To start on your DevOps journey, here are some key steps to consider:

Adopt a DevOps Mindset:
DevOps is more than just tools and processes, it's a cultural shift. Encourage collaboration, communication, and shared responsibility among your development and operations teams.

Automate Everything:
Embrace automation in every aspect of your development and deployment processes.
Automate tasks such as code builds, tests, deployments, and infrastructure provisioning.
Use tools like Jenkins, GitLab CI/CD, or AWS CodePipeline to create automated pipelines.

Implement Infrastructure as Code:
Treat your infrastructure as code.
Use tools like Terraform or AWS CloudFormation to define and manage your cloud resources declaratively.
Version control your infrastructure code and apply the same principles of testing and deployment as you would for application code.

Monitor and Measure:
Implement comprehensive monitoring and logging solutions to gain visibility into your applications and infrastructure.
Use tools like AWS CloudWatch, Prometheus, or Grafana to collect and visualize metrics.
Set up alerts and notifications to proactively identify and resolve issues.

Continuously Improve:
DevOps is an ongoing journey.
Continuously measure and analyze your processes, identify bottlenecks and areas for improvement, and iterate on your practices.

GETTING STARTED WITH DEVOPS

DevOps is a powerful approach that enables organizations to deliver software faster, more reliably, and with higher quality.

By embracing DevOps practices and tools, you can streamline your cloud development process, increase collaboration, and deliver value to your users more efficiently.

CHAPTER 7

CDK WITH TYPESCRIPT

WHAT IS AWS CDK?

AWS CDK is an open-source software development framework that allows you to define your cloud infrastructure using familiar programming languages like TypeScript, JavaScript, Python, and more.
It provides a high-level abstraction over AWS CloudFormation, enabling you to express your infrastructure as code in a more concise and expressive way.

WHY USE AWS CDK WITH TYPESCRIPT?

Familiarity:
If you're already comfortable with TypeScript or JavaScript, AWS CDK allows you to use your existing skills to define your infrastructure.

Type Safety:
TypeScript's static typing helps catch errors early in the development process, improving code quality and reducing bugs.

Modularity:
With AWS CDK and TypeScript, you can create reusable components and libraries for your infrastructure, promoting code reuse and maintainability.

GETTING STARTED WITH AWS CDK AND TYPESCRIPT:

To get started with AWS CDK and TypeScript, follow these steps:

Install AWS CDK on your terminal:

```
npm install -g aws-cdk
```

Create a new directory for your project and navigate to it:

```
mkdir my-cdk-project
cd my-cdk-project
```

Initialize a new AWS CDK app:

```
cdk init app --language typescript
```

Install the required dependencies:

```
npm install
```

GETTING STARTED WITH AWS CDK AND TYPESCRIPT:

Defining Your Infrastructure:

```typescript
import * as cdk from '@aws-cdk/core';
import * as s3 from '@aws-cdk/aws-s3';

export class MyStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    new s3.Bucket(this, 'MyBucket', {
      versioned: true,
      removalPolicy: cdk.RemovalPolicy.DESTROY,
    });
  }
}
```

In this example, we create a new stack called `MyStack` and define an S3 bucket with versioning enabled and a removal policy set to destroy the bucket when the stack is deleted.

Deploying Your Infrastructure:

To deploy your infrastructure, run the following command:

```
cdk deploy
```

AWS CDK will generate the underlying CloudFormation template and perform the necessary provisioning steps to create your infrastructure in the cloud.

Destroying Your Infrastructure:

To clean up and delete your infrastructure when you no longer need it, run:

```
cdk destroy
```

ELEVATE YOUR SKILLS: WHERE TO GO FROM HERE

Ready to take your cloud engineering skills to the next level?

Join our Cloud Engineering Academy and go from Zero to Cloud Engineer Hero in just 12 weeks.
No previous tech experience needed!
Our proven system has already launched 200+ successful cloud careers.
We offer real-world projects to build your portfolio, up-to-date learning that mirrors the evolving cloud industry, personalised mentorship, live expert classes, and a job search success system.
With our academy, you'll gain the skills employers crave and stand out in the job market.

Join Now