Cloud Engineer Roadmap

Your Step-by-Step Guide from Beginner to Advanced Skills

by TechWorld with Nana

Provided by TechWorld with Nana

I'm Nana, Co-Founder of TechWorld with Nana.

As a Cloud and DevOps engineer, I'm dedicated to helping engineers build the most valuable and highly-demanded DevOps and Cloud skills.

Through my YouTube channel and my comprehensive DevOps bootcamps, **I've helped 1,000,000s of engineers** master the tools and concepts that drive modern software development.

About this Cloud Engineer Roadmap

This detailed guide outlines a **complete step-by-step roadmap for becoming a Cloud engineer** - from core IT fundamentals to advanced cloud security practices.

Whether you're just starting out or looking to advance your existing skills, this document gives you a structured guide to master cloud technologies, with practical advice for each phase of learning.

Happy learning!

Nana Janashia

Co-Founder TechWorld with Nana

TECHWORLD WITH NANA

Table of Contents

The Power of Cloud Computing

FROM ON-PREMISE...

Imagine working at a company that still manages all their infrastructure on-premises.

✗ Whenever a new application needs to be deployed, it **takes weeks** to order hardware, set up networking, configure servers, and finally deploy the application.

✗ When traffic spikes unexpectedly, the application crashes because you can't scale resources quickly enough.

...TO THE CLOUD

✓ Now contrast this with a cloud-native company where infrastructure is provisioned in minutes, automatically scales based on demand, and developers can focus on building features instead of waiting for infrastructure.

✓ That's the power of cloud engineering - enabling businesses to move faster, scale effortlessly, and reduce operational overhead.

TECHWORLD WITH NANA

CLOUD ENGINEERING

CAREER PATH AND MARKET

TECHWORLD WITH NANA

Cloud Engineering Career Path

1

Junior Cloud Engineer

Entry-level position focused on basic cloud implementations.

Experience:

0-2 years

Avg. Compensation:

$80,000 - $100,000

2

Cloud Engineer

Manages cloud infrastructure and implements solutions independently.

Experience:

2-5 years

Avg. Compensation:

$100,000 - $130,000

3

Senior Cloud Engineer / DevOps

Leads complex cloud initiatives and establish best practices for organizations

Experience:

5-8 years

Avg. Compensation:

$130,000 - $160,000

4

Cloud Architect

Design enterprise-scale cloud architectures and drive technical strategy

Experience:

8+ years

Avg. Compensation:

$150,000 - $200,000+

Note: Compensation figures represent U.S. market averages and may vary based on location, company size, specific technologies, and individual experience levels.

Remember that titles vary between organizations, but the progression typically involves:

- increasing responsibility for system design
- architecture decisions
- and technical leadership.

Cloud Engineering Career Path

Some professionals may choose to specialize in specific areas, such as:

Related Specialized Roles

- **Site Reliability Engineer (SRE)**: Focus on building and maintaining highly reliable and scalable systems
- **Cloud Security Engineer**: Specialize in securing cloud environments and ensuring compliance
- **Cloud Data Engineer**: Design and implement cloud-based data processing and analytics solutions
- **Cloud Network Engineer**: Focus on designing and optimizing cloud networking architectures
- **Cloud Financial Analyst**: Specialize in cloud cost optimization and financial operations (FinOps)

The most successful cloud professionals **combine technical expertise with business acumen**, understanding not just how to implement cloud solutions but why they matter to the organization.

By continuously expanding your knowledge, building practical experience, and adapting to evolving technologies, you can build a rewarding and sustainable career in cloud engineering.

Cloud Computing Market

Growing Demand

The demand for cloud skills is growing year after year.

Lucrative Compensation

According to recent data, cloud computing jobs are among the highest-paying in tech.

Why Cloud Engineers Earn Top Salaries

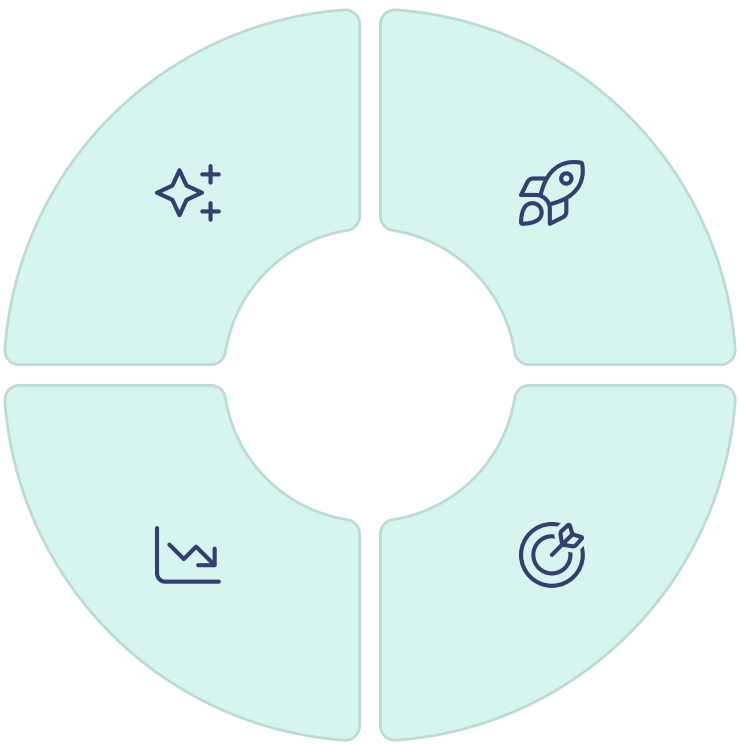
The high salaries in cloud computing come from a few factors:

High business impact

Cloud infrastructure directly affects an organization's operational efficiency, security posture, and ability to innovate

Cost savings

Companies willingly invest in talent that can optimize cloud spending and efficiency



Complex skillset

Professionals must master multiple technologies and continuously adapt to rapid innovation

Shortage of Cloud Engineering skillset

The number of qualified cloud professionals is way less than what market demands

TECHWORLD WITH NANA

CLOUD ENGINEER ROADMAP

Phase 1:

Foundational IT Knowledge

Many aspiring cloud engineers ask: *"Can I just jump straight into learning AWS or Azure without any prior tech experience?"*

Technically, yes, but you would be building a house without a proper foundation.

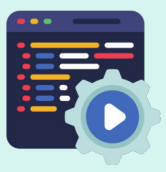
Think about a situation where you're deploying a web application to the cloud and suddenly it's not accessible. Without understanding networking concepts like IP addressing, DNS, load balancing, and firewalls, you'd struggle to troubleshoot the issue.

So before diving into cloud-specific technologies, it's important to **build a solid foundation in core IT skills.**

These fundamentals are the backbone of cloud engineering expertise and will help you understand the underlying infrastructure that powers cloud environments.

Linux Administration (OS)

Networking Fundamentals



Programming Skills

Database Knowledge

Phase 1:

Linux Administration

Develop a solid understanding of operating systems.

Most cloud workloads run on Linux, making it a critical skill for anyone in this field.

Key Linux Skills for Cloud Engineers

- ☐ **Command Line Proficiency:** Become comfortable navigating and managing systems using the terminal. Learn essential commands for file management, process control, and system monitoring.
- ☐ **File System Structure:** Understand the Linux file system hierarchy, important directories, and how to navigate between them.
- ☐ **File Permissions:** Master the concepts of ownership, permissions (read, write, execute), and how to modify them using chmod and chown commands.
- ☐ **Process Management:** Learn how to start, stop, and monitor processes. Understand concepts like foreground vs. background processes, job control, and process priorities.
- ☐ **Shell Scripting:** Develop the ability to automate repetitive tasks by writing bash scripts. This includes variables, conditionals, loops, and functions.
- ☐ **Package Management:** Become familiar with package managers like apt, yum, or dnf for installing, updating, and removing software.
- ☐ **System Logging:** Know where logs are stored and how to analyze them for troubleshooting.

ⓘ In cloud environments, you'll frequently need to SSH into virtual machines, configure services, troubleshoot issues, and automate tasks - all of which require strong Linux skills.

Even when working with managed services, understanding the underlying operating system helps you better grasp how these services work and interact.

Phase 1:

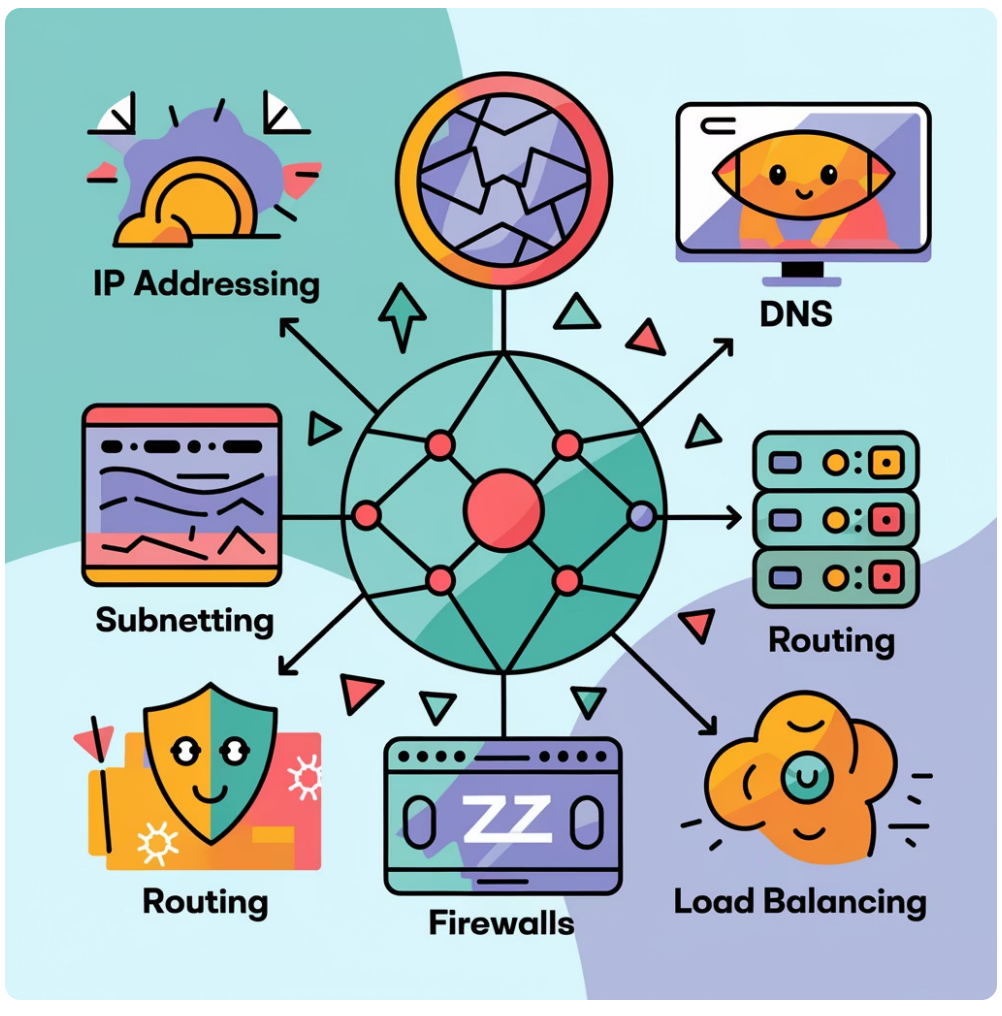
Networking

Networking knowledge is important for cloud engineering as it forms the backbone of all cloud services.

When applications in the cloud aren't working correctly, networking issues are often the culprit.

Essential Networking Concepts

- [ ] **IP Addressing:** Understand IPv4 and IPv6 addressing, subnetting, and CIDR notation.
- [ ] **DNS:** Learn how domain names are resolved to IP addresses and the role of DNS servers.
- [ ] **Subnetting:** Basic idea of breaking a network into smaller parts. Understand CIDR notation
- [ ] **Routing:** Understand how data packets find their way across networks.
- [ ] **Firewalls:** Learn how to control traffic flow and secure network boundaries.
- [ ] **Load Balancing:** Understand how traffic is distributed across multiple servers.
- [ ] **VPN:** Learn how virtual private networks provide secure connections.



Strong networking knowledge allows you to design secure, efficient cloud architectures and **troubleshoot connectivity issues effectively.**

When a customer complains they can't access your application, you'll know exactly where to look - whether it's a DNS issue, a routing problem, or a misconfigured security group.

Phase 1:

Programming Basics

While you don't need to be a software engineer to work in cloud engineering, programming skills are increasingly important for automation, infrastructure as code, and interacting with cloud services programmatically.

Why Programming Matters for Cloud Engineers

- **Automation:** Script repetitive tasks to save time and reduce human error.
- **Infrastructure as Code:** Define and provision infrastructure using code rather than manual processes.
- **API Interaction:** Programmatically interact with cloud services through their APIs.
- **Custom Solutions:** Build tools and utilities to fill gaps in existing cloud services.
- **Data Processing:** Extract, transform, and analyze data from cloud resources.

Recommended Languages

Python

Python is particularly valuable in the cloud space due to its readability, extensive libraries, and strong support from cloud providers. AWS, Azure, and Google Cloud all offer Python SDKs for interacting with their services.

JavaScript/Node.js

Useful for serverless functions and cloud-native applications.

AWS Lambda, Azure Functions, and Google Cloud Functions all support JavaScript/Node.js.

Go (Golang)

Increasingly popular for cloud-native applications and tools.

Many modern cloud tools like Terraform, Docker, and Kubernetes are written in Go.

Key Programming Concepts

- [ ] **Variables and Data Types:** Understand how to store and manipulate different types of data.
- [ ] **Control Structures:** Learn about conditionals (if/else) and loops (for, while) to control program flow.
- [ ] **Functions:** Know how to organize code into reusable blocks.
- [ ] **Error Handling:** Understand how to gracefully handle exceptions and errors.
- [ ] **File I/O:** Learn to read from and write to files.
- [ ] **Libraries and Packages:** Become familiar with using external libraries to extend functionality.
- [ ] **API Requests:** Know how to make HTTP requests to interact with web services.

Phase 1:

Databases

Database knowledge is essential for cloud engineers as data storage is a core component of any cloud application.

Understanding different database types, their use cases, and how to manage them in the cloud is crucial for designing effective solutions.

Relational Databases (SQL)

- **Concepts:** Tables, rows, columns, primary keys, foreign keys, normalization
- **SQL Basics:** SELECT, INSERT, UPDATE, DELETE statements
- **Joins:** Combining data from multiple tables
- **Indexes:** Improving query performance
- **Transactions:** Ensuring data integrity

Cloud Services: Amazon RDS, Azure SQL Database, Google Cloud SQL

NoSQL Databases

- **Document Stores:** MongoDB, Amazon DocumentDB
- **Key-Value Stores:** Redis, Amazon DynamoDB
- **Column Stores:** Cassandra, Google Bigtable
- **Graph Databases:** Neo4j, Amazon Neptune

When to Use: High scalability needs, flexible schemas, large volumes of data

Cloud Database Considerations

- **Managed vs. Self-Managed:** Understand the trade-offs between using managed database services (like RDS) versus running your own database on virtual machines.
- **Scaling:** Learn about vertical scaling (increasing instance size) and horizontal scaling (adding more instances).
- **High Availability:** Understand concepts like read replicas, multi-AZ deployments, and failover mechanisms.
- **Backup and Recovery:** Know how to implement proper backup strategies and perform point-in-time recovery.
- **Security:** Learn about encryption at rest and in transit, network isolation, and access control.
- **Performance Optimization:** Understand how to monitor and optimize database performance.

As a cloud engineer, you'll need to make informed decisions about which database service to use for different workloads, how to configure them for optimal performance and reliability, and how to integrate them with your applications.

TECHWORLD WITH NANA

CLOUD ENGINEER ROADMAP

Phase 2: Cloud Fundamentals

Phase 2:

Cloud Service Models Explained

Each model offers a different level of control and responsibility:

Infrastructure as a Service (IaaS)

What It Is: Provides virtualized computing resources over the internet.

What You Manage: Operating systems, middleware, applications, data.

What Provider Manages: Physical servers, storage, networking hardware.

Examples: Amazon EC2, Azure Virtual Machines, Google Compute Engine.

Use Case: When you need maximum control over your infrastructure, have specific compliance requirements, or are migrating existing applications with minimal changes.

Platform as a Service (PaaS)

What It Is: Provides a platform allowing customers to develop, run, and manage applications without dealing with the complexity of building and maintaining the infrastructure.

What You Manage: Applications and data.

What Provider Manages: Operating systems, middleware, runtime.

Examples: AWS Elastic Beanstalk, Azure App Service, Google App Engine.

Use Case: When you want to focus on application development without worrying about infrastructure management, patching, or scaling.

Software as a Service (SaaS)

What It Is: Delivers software applications over the internet, on a subscription basis.

What You Manage: Configuration and data.

What Provider Manages: Everything else.

Examples: Microsoft 365, Google Workspace, Salesforce.

Use Case: When you want to use an application without any responsibility for maintenance, updates, or infrastructure.

Phase 2:

Deployment Models

Public Cloud:

Services offered over the public internet and available to anyone who wants to purchase them.

Multi-Cloud:

Using services from multiple cloud providers to avoid vendor lock-in or leverage specific services from different providers.

Private Cloud:

Cloud infrastructure operated solely for a single organization, either managed internally or by a third party.

Hybrid Cloud:

Combination of public and private clouds, allowing data and applications to be shared between them.

Understanding these models helps you determine the **right balance of control, responsibility, and cost for your specific needs.**

Phase 2:

Choosing Your Cloud Platform

Now, choose a cloud provider to focus on.

The cloud computing market is dominated by three major players:

Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP).

AWS currently has the largest market share, so it's often a good starting point.

> Selecting the right cloud platform is a critical decision that will shape your learning journey and potentially your career path.
>
> While the fundamental concepts remain similar across providers, each platform has its own services, terminology, and management interfaces that require specific knowledge and skills.

Amazon - AWS

As the pioneer and market leader with approx. 33% market share, AWS offers the broadest set of services. The platform is widely adopted across industries, which means abundant job opportunities for AWS specialists.

Microsoft Azure

Azure has established itself as a strong 2nd player in the market. Azure offers seamless integration with Microsoft products, making it an attractive choice for organizations with existing Microsoft infrastructure.

Google - GCP

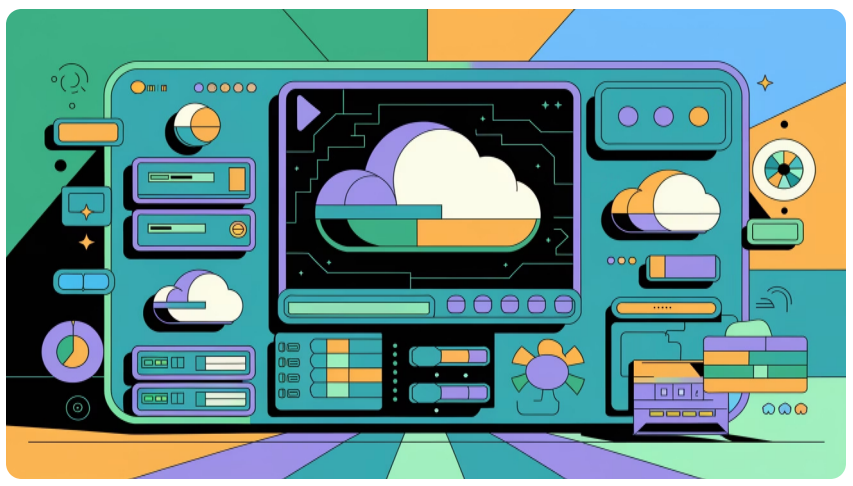
Google Cloud Platform has distinguished itself through innovation in data analytics, artificial intelligence, and machine learning capabilities.

Regardless of which platform you choose initially, the recommendation is to **focus deeply on one provider to build comprehensive knowledge before expanding to others.**

Once you've mastered one cloud platform, it's easy to learn new cloud platforms. Many cloud professionals eventually become proficient in multiple platforms, but starting with one allows you to build depth before breadth.

Phase 2:

Platform-Specific Skills: Deep Dive into Chosen Cloud



Develop expertise in the core services and capabilities specific to that provider.

Key services you should master for each of the major cloud platforms:

Compute Services

Begin with understanding compute services like EC2 (AWS) or Virtual Machines (Azure).

Learn how to launch instances, connect to them, and understand the different instance types and pricing models.

Storage Services

Move on to storage services like S3 (AWS) or Blob Storage (Azure).

Understand how object storage works, how to create buckets/containers, upload files, and set permissions.

Networking

Explore networking in the cloud - Virtual Private Clouds (VPCs), subnets, route tables, internet gateways, and security groups.

This is where your networking fundamentals will really help you.

ⓘ For example, a team experiencing slow application performance might discover they had placed their database in a different region than their web servers, causing high latency.

Without understanding cloud networking concepts, this issue would be much harder to diagnose and fix.

CLOUD ENGINEER ROADMAP

Phase 3: Infrastructure as Code (IaC)

Phase 3:

IaC - Automating Cloud Deployments

As you become comfortable with manually creating resources in the cloud, you'll quickly realize this isn't scalable. Imagine having to click through the UI console to create hundreds of resources for a production environment - not only is it time-consuming, but you may make tons of mistakes or forget some important steps, as we humans usually do.

This is where Infrastructure as Code (IaC) comes in. Instead of manually creating resources, you **define your infrastructure in code**, with tools like Terraform, and let it provision the whole infrastructure automatically.

IaC Tools

- **Terraform:** The most widely used IaC tool that works across all major cloud providers. Learn the basic syntax, how to create resources, use variables, modules, and manage state.
- **AWS CloudFormation:** AWS's native IaC service, which only works on their cloud platform.
- **Ansible:** While Terraform is great for provisioning infrastructure, Ansible helps you install software packages and configure settings on existing servers.

Some of its Benefits

Version Control: Infrastructure definitions can be stored in version control systems like Git, providing history, rollback capabilities, and collaboration features.

Repeatability: The same infrastructure can be deployed multiple times consistently across different environments (development, staging, production).

Automation: Infrastructure deployment can be integrated into CI/CD pipelines, reducing manual work and errors.

Documentation: The code itself serves as documentation of the infrastructure, making it easier to understand and modify.

A practical IaC use case is deploying the same application across multiple environments (development, staging, and production).

By using Terraform and Ansible, you can ensure each environment is almost identical in configuration, providing confidence that when you test your application on development and staging environments, it will work the same way in production.

Phase 3:

Infrastructure as Code Deep Dive

Infrastructure as Code (IaC) is a key practice in cloud engineering that allows you to **manage and provision infrastructure through code instead of manual processes.**

Terraform

Terraform is the most widely used IaC tool that works across all major cloud providers.

It uses a **declarative approach where you specify the desired end state of your infrastructure.**

Key Concepts to learn

- **Providers:** Plugins that allow Terraform to interact with cloud platforms
- **Resources:** Infrastructure components you want to create
- **Variables:** Parameters that can be passed to your configuration
- **Modules:** Reusable components for organizing and encapsulating resources
- **State:** Terraform's record of what infrastructure it manages

Terraform Workflow

Write

Define infrastructure in configuration files

Plan

Preview changes before applying

Apply

Create or modify infrastructure

Destroy

Remove infrastructure when no longer needed

TECHWORLD WITH NANA

CLOUD ENGINEER ROADMAP

Phase 4: Containers and Container Orchestration

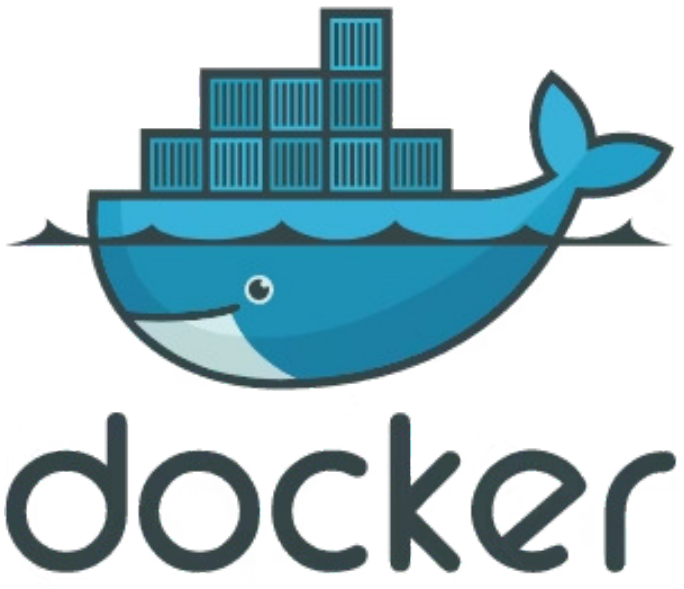
Phase 4:

Containers

As you progress in your cloud engineering journey, you'll encounter containerization - a technology that has revolutionized **how applications are packaged and deployed.**

Think about this scenario: Your development team has built an application that works perfectly on their laptops, but when deployed to the cloud, it fails with dependency issues or because it expects certain environment variables to be defined in its OS environment, which are not set or expects latest version of Python to be installed.

This is the "it works on my machine" problem that containers help solve.



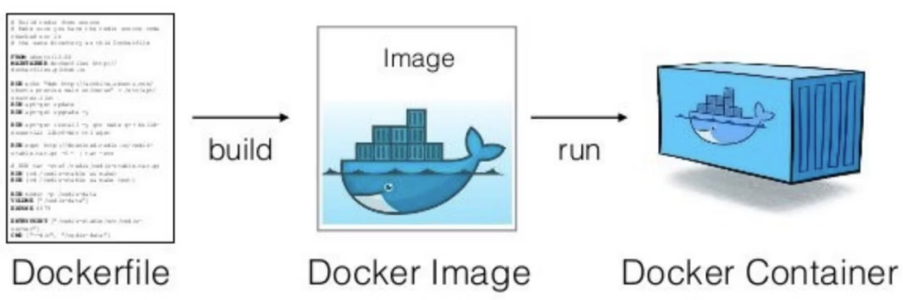
Docker

Standard tool for containerization. Docker packages the whole OS configuration and dependencies together with the application code in an artifact, which you can deploy on any server without any pre-configuration.

Start by learning what containers are, how they differ from virtual machines, what Docker images are, and the basic commands for building and running containers.

Key Docker concepts to master include:

- [ ] Dockerfile syntax for defining container images
- [ ] Image building, tagging, and versioning
- [ ] Container lifecycle management
- [ ] Multi-stage builds for efficient images
- [ ] Docker Compose for multi-container applications
- [ ] Container networking and storage volumes
- [ ] Security best practices for container images



Phase 4:

Container Orchestration

Running individual containers isn't enough for production workloads, where you may have 1000s Docker containers of various microservices, databases, and other services.

So while Docker solves the problem of containerization, **Kubernetes addresses the challenges of deploying, scaling, and managing containerized applications at scale.**

> As an open-source platform, Kubernetes (also called "K8s") has become the de facto standard for container orchestration.

Kubernetes

Kubernetes is like your automated operations manager, automating the deployment, scaling, and management of containerized applications.

Begin with understanding K8s concepts like pods, deployments, services, and ingress. Then learn how to deploy applications to Kubernetes, scale them, handle updates and rollbacks.

Essential Kubernetes concepts include:

- ☐ Pods, deployments, and services
- ☐ ConfigMaps and Secrets for configuration
- ☐ Namespace organization and resource quotas
- ☐ Horizontal and vertical pod autoscaling
- ☐ StatefulSets for stateful applications
- ☐ Persistent storage with PersistentVolumes
- ☐ Ingress controllers for routing external traffic
- ☐ RBAC for access control

Cloud platforms like AWS, Azure, and Google Cloud all offer **managed Kubernetes services (EKS, AKS, GKE)**, which handle the provisioning of K8s cluster and control plane management for you.

Start with these rather than setting up your own Kubernetes cluster from scratch.

Phase 4:

Cloud-Native Architecture

For cloud engineers, mastering containers and orchestration tools is essential for building scalable, resilient cloud-native applications.

Understanding containerization leads naturally to cloud-native architecture principles, which emphasize:

Microservices

Breaking applications into small, independently deployable services that can be developed, scaled, and maintained separately. This approach enables teams to work in parallel and deploy changes more frequently with less risk.

Immutable Infrastructure

Treating infrastructure components as disposable and replacing them entirely rather than modifying them in place. This principle ensures consistency and reliability across environments.

Resilience

Designing systems to withstand failures through redundancy, load balancing, circuit breakers, and graceful degradation, ensuring high availability even when individual components fail.

Observability

Implementing comprehensive monitoring, logging, and tracing to gain insights into application behavior and performance in distributed environments.

For Hands-on Experience:

1. **Deploy a containerized application on Kubernetes** using either a local environment like Minikube or a managed Kubernetes service such as Amazon EKS, Azure AKS, or Google GKE.

2. **Start with a simple multi-tier application** that includes a web frontend, an API service, and a database. This project will help you understand how container orchestration handles deployment, scaling, networking, and persistence in a real-world scenario.

3. As your expertise grows, **explore advanced Kubernetes topics** like custom resource definitions (CRDs), operators for automating application management, service meshes for advanced networking capabilities, and GitOps workflows for declarative application deployment.

TECHWORLD WITH NANA

CLOUD ENGINEER ROADMAP

Phase 5: CI/CD and DevOps Practices

Phase 5:

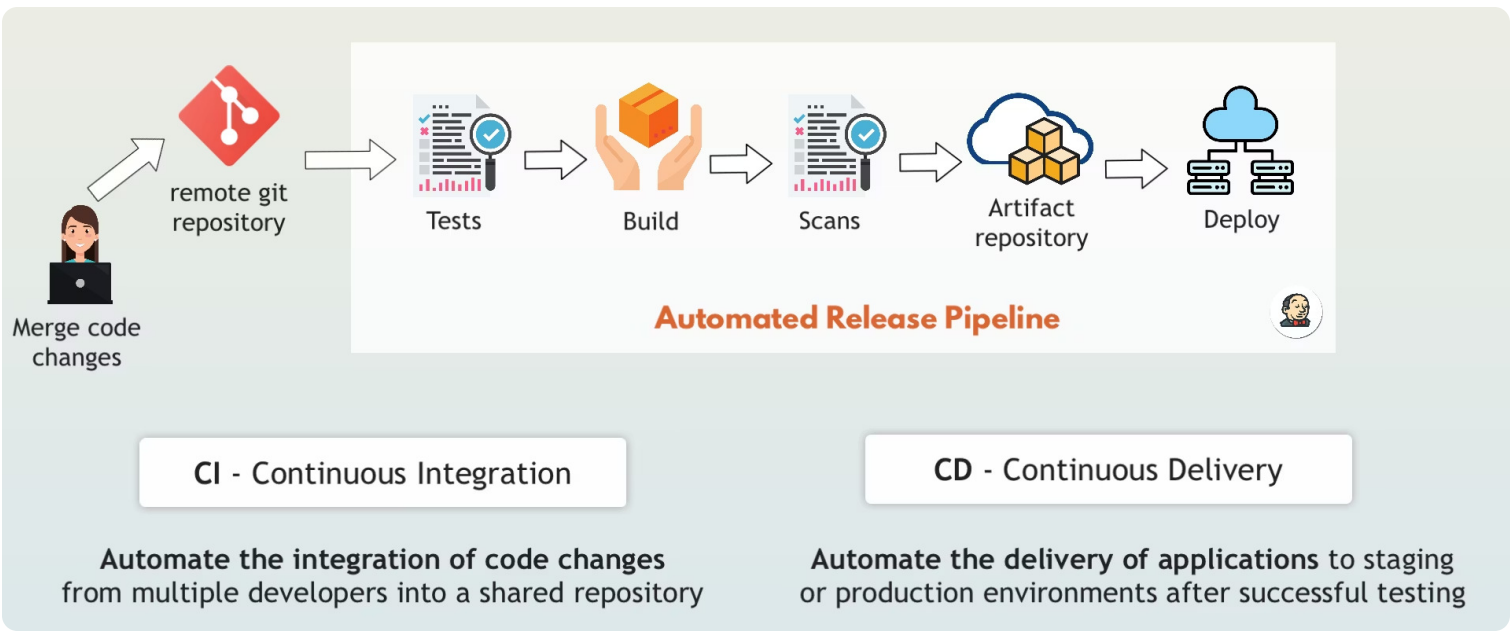
CI/CD and DevOps Practices

Now you've mastered containerization and Kubernetes, but there's still a critical piece missing.

> Think about this scenario: Your team has 10 microservices running in Kubernetes. Each service is being actively developed by different teams.
>
> How do you get all these changes from code repositories to your Kubernetes cluster without a slow manual deployment process, without causing application downtime, or without the security risk of all 10 teams having direct access to the cluster?

This is exactly where CI/CD pipelines come in.

They serve as the bridge between your application code and your Kubernetes infrastructure:



How it works

Code Changes

Developers commit code changes to the repository

Build and Test

The code is automatically built and tested

Create Image

A new Docker image is created, tagged, and pushed to a container registry

Deploy

Kubernetes manifests are updated and the application is deployed

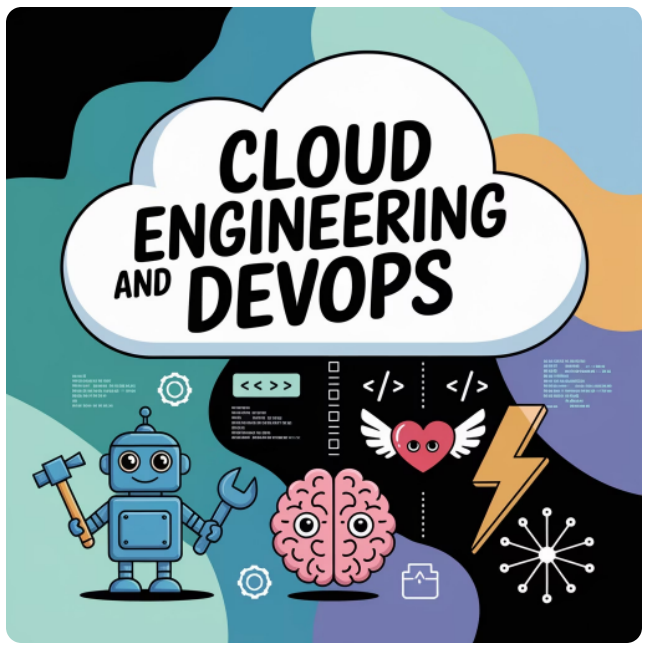
Validate

The deployment is validated with automated tests and monitoring

Phase 5:

CI/CD and DevOps Practices

Without this automation, you'd be manually building images, updating manifests, and applying changes to your cluster - a process that's not only time-consuming but prone to human error.



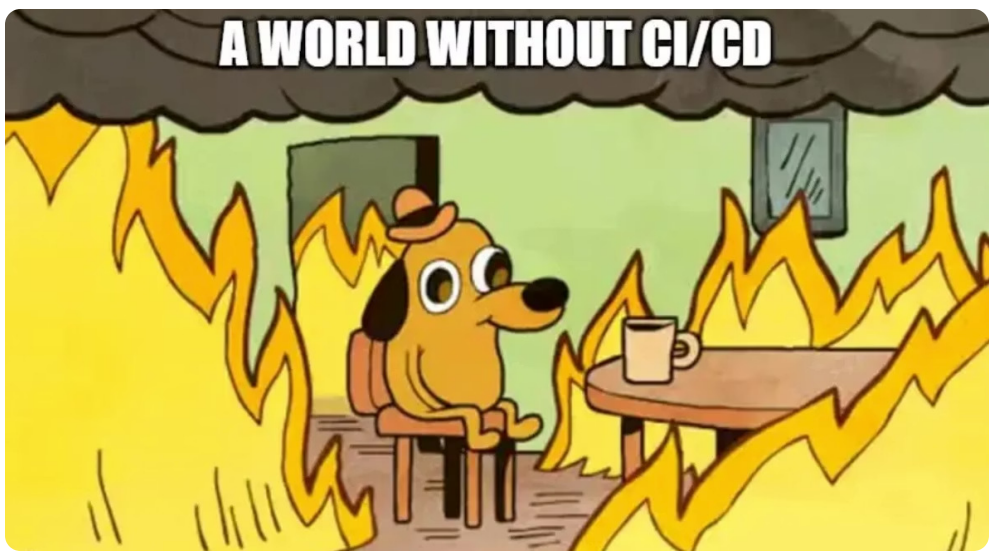
Cloud engineering and DevOps practices are deeply intertwined, with both focusing on

- Automation
- Collaboration
- Continuous Improvement.

Mastering DevOps methodologies will enable you to build and maintain cloud systems more efficiently while ensuring high reliability and rapid delivery of new features.

How to get started

1. Start by understanding the concepts of CI/CD and the overall software delivery lifecycle.
2. Then learn a specific CI/CD tool like GitHub Actions, Jenkins, or GitLab CI.
3. Create a basic pipeline that automatically builds, tests, and deploys an application when changes are pushed to the repository



CLOUD ENGINEER ROADMAP

Phase 6: Monitoring, Logging, and Observability

Phase 6:

Monitoring, Logging, and Observability

As you deploy more complex applications to the cloud, you need to **ensure they're running correctly and troubleshoot issues when they arise.**

This is where monitoring, logging, and observability come in.

> Imagine getting a call at 2 AM because the application is down, but you have no monitoring in place. You'd have to manually check each component to find the issue, under pressure, because every minute counts when your users are affected.

Monitoring

Think of monitoring as the alarm system in a building.

It alerts you when something goes wrong based on predefined thresholds.

Explore solutions like Prometheus or cloud-specific options like AWS CloudWatch.

Logging

Logging is like having cameras installed everywhere to record all activities.

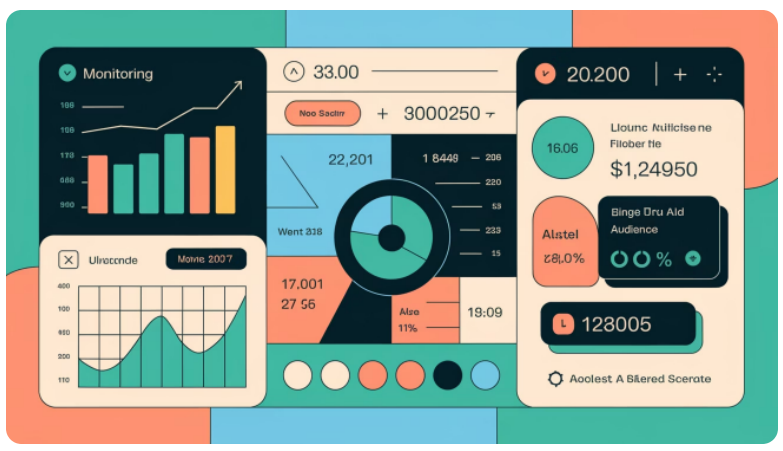
It provides detailed information about what's happening in your system.

Learn about logging solutions like EFK or ELK stack (Elasticsearch, Fluentd, Kibana) or cloud-specific options like AWS CloudTrace.

Observability

Observability is the entire system including monitoring, logging, and the ability to understand the internal state of your system based on its outputs.

It helps you answer the question *"Why is this happening?"* rather than just *"What is happening?"*



A proper observability setup might include:

- **Metrics** showing system performance and request rates
- **Alerts** that notify you when something goes wrong,
- **Dashboards** providing a visual overview of system health
- **Logs** with detailed information about specific events, and traces showing the path of requests through your distributed system.

TECHWORLD WITH NANA

CLOUD ENGINEER ROADMAP

Phase 7: Cloud Security

Phase 7:

Cloud Security

Security should not be an afterthought - it should be integrated into every step of your cloud engineering journey.

As you build more complex systems in the cloud, and as your attack surface increases, security becomes increasingly important.

Consider this scenario: Your company has just experienced a data breach because an S3 bucket was accidentally made public.

Without security checks in place, among hundreds of buckets and a complex infrastructure setup, one misconfiguration slipped through to production.

Now your company is facing potential regulatory fines, lost customer trust, and countless hours of remediation work. This is why cloud security matters.

As a cloud engineer, you need to implement robust security measures to protect data and applications while ensuring compliance with regulatory requirements.

Implement these essential security measures:

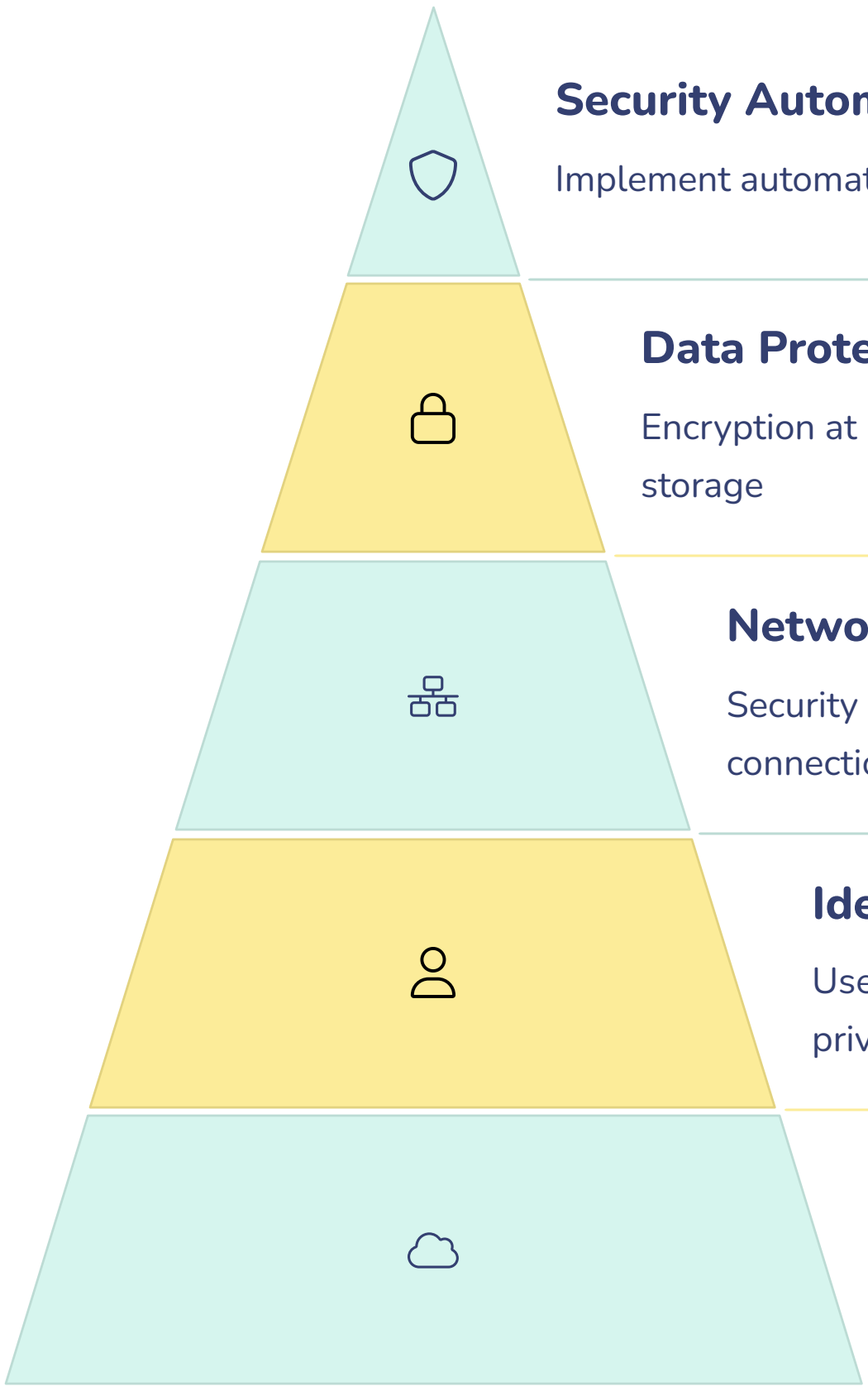
- ☐ **Strong IAM policies**: Follow the principle of least privilege, providing users and services only the permissions they need. Implement role-based access control (RBAC) and regularly audit access rights.
- ☐ **Data encryption**: Encrypt sensitive data both at rest (in storage) and in transit (over networks) using industry-standard encryption protocols and key management services.
- ☐ **Network security**: Configure network security groups, firewalls, and VPCs to control traffic flow and limit exposure. Implement network segmentation to isolate sensitive systems.
- ☐ **Security monitoring**: Set up comprehensive logging and monitoring to detect and respond to suspicious activities. Implement automated alerting for security events.
- ☐ **Vulnerability management**: Regularly scan systems for vulnerabilities and apply security patches promptly. Implement a formal process for security updates.
- ☐ **Secure CI/CD pipelines**: Integrate security testing into development workflows, including static code analysis, dependency scanning, and container image scanning.

TECHWORLD WITH NANA

Phase 7:

Cloud Security

A well-secured cloud environment includes IAM policies that grant minimal necessary permissions, network security that restricts access to resources, encryption for sensitive data, regular security audits, and automated compliance checks:



Security Automation

Implement automated security checks and compliance monitoring

Data Protection

Encryption at rest and in transit, key management, secure storage

Network Security

Security groups, network ACLs, private subnets, VPN connections

Identity & Access Management

Users, roles, and policies following principle of least privilege

Shared Responsibility Model

Understanding what you vs. your cloud provider secures

Security is challenging, because **you need to lock down and secure every single point of access...**

while hackers only need one entry point, one accidentally forgotten configuration, one security loophole.

TECHWORLD WITH NANA

CLOUD ENGINEERING

PRACTICAL LEARNING APPROACH

Practical Learning Approach

Now that we've covered the roadmap, let's talk about how to actually learn these skills effectively.

The most important thing is to **build real projects** - not just follow tutorials passively. Create a personal cloud lab where you can experiment with different services and architectures.

1 Deploy a Simple Static Website

Start with hosting a static website on S3 or Azure Storage

2 Host a Dynamic Web Application

Move to EC2 or Azure VMs with a database like RDS

3 Use Managed Services

Refactor to use managed services like RDS for the database

4 Implement Infrastructure as Code

Automate what you built manually using Terraform

5 Containerize Your Application

Deploy to managed Kubernetes like EKS or AKS

6 Set Up CI/CD Pipeline

Implement automated deployments with GitHub Actions

7 Add Monitoring and Security

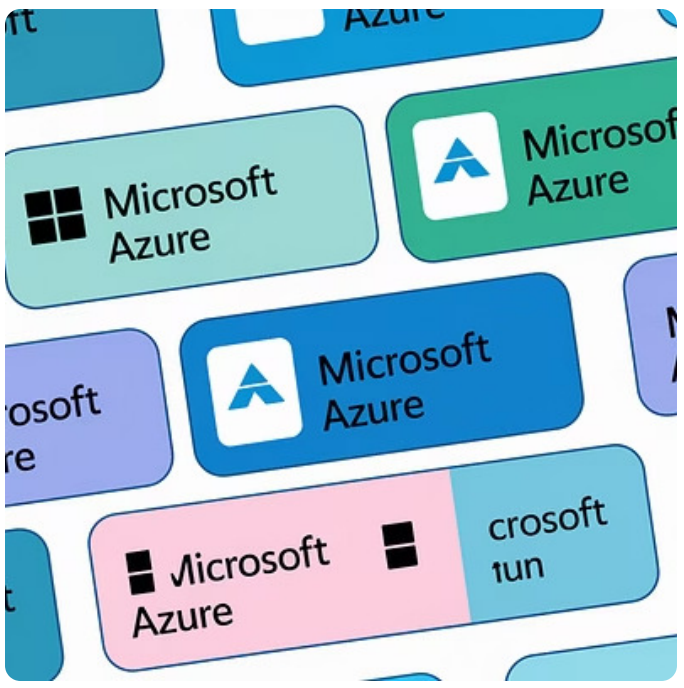
Implement observability and security best practices

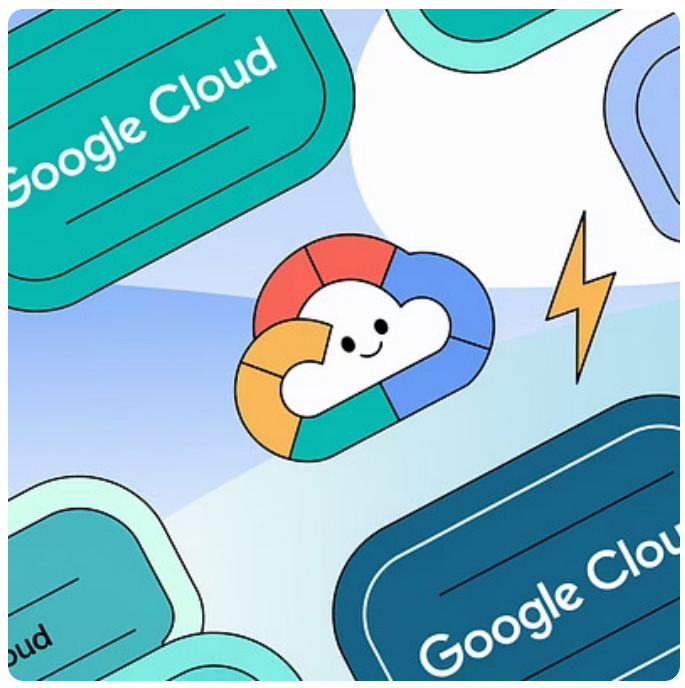
> This progressive approach allows you to **stack knowledge step by step**, learning one important concept after another.
>
> **Document your learning journey** through a blog or GitHub repository, serving as both a learning tool for yourself and a portfolio for potential employers.

Cloud Certifications

While hands-on experience is most important, certifications can validate your knowledge and help you get past HR filters.

Each cloud provider has their own certificates that can demonstrate your expertise in their platform.





AWS Certifications

- AWS Certified Cloud Practitioner
- AWS Certified Solutions Architect - Associate
- AWS Certified Developer - Associate
- AWS Certified SysOps Administrator - Associate

Azure Certifications

- Microsoft Certified: Azure Fundamentals
- Microsoft Certified: Azure Administrator Associate
- Microsoft Certified: Azure Developer Associate
- Microsoft Certified: Azure Solutions Architect Expert

Google Cloud Certifications

- Google Cloud Digital Leader
- Google Cloud Associate Cloud Engineer
- Google Cloud Professional Cloud Architect
- Google Cloud Professional Data Engineer

Very important - don't just memorize answers for the exams - focus on understanding the concepts and implementing them in real projects. The certification should be a validation of skills you actually possess, not just a piece of paper.

Your Cloud Engineering Journey

Becoming a cloud engineer requires a lot of learning and practice, but it's an incredibly rewarding career path with excellent growth opportunities. The cloud continues to evolve rapidly, creating ongoing demand for skilled professionals who can design, implement, and manage cloud solutions.

Your Cloud Engineering Roadmap Summarized:

1

Build a solid foundation in IT fundamentals before diving into cloud-specific technologies

2

Understand the core concepts of cloud computing and choose a cloud provider to focus on initially

3

Learn infrastructure as code to automate and scale your cloud deployments

4

Master containerization and orchestration for modern application deployment

5

Implement CI/CD pipelines to streamline the development and deployment process

6

Set up comprehensive monitoring and observability for your cloud resources

7

Prioritize security at every level of your cloud architecture

The Need for a Structured Program to learn the Complete Profession

When I first started learning cloud technologies, **I felt completely overwhelmed.**

There were so many tools, platforms, and concepts to master - from Linux fundamentals to Kubernetes orchestration to advanced security practices. **I'd spend weeks learning one technology only to discover it was just a tiny piece of a much larger puzzle.**

That's why we created a DevOps Bootcamp to learn DevOps and Cloud in a structured, holistic way. And later developed an advanced DevSecOps bootcamp that focuses on the security aspect of DevOps including cloud security.

You learn a **complete profession, not just technologies.**

Learning Together is Better Than Learning Alone

Many engineers try to piece together their cloud knowledge from random tutorials, documentation, and trial-and-error. While this works for some, it often **leads to knowledge gaps and confusion** about how everything fits together in real-world scenarios.

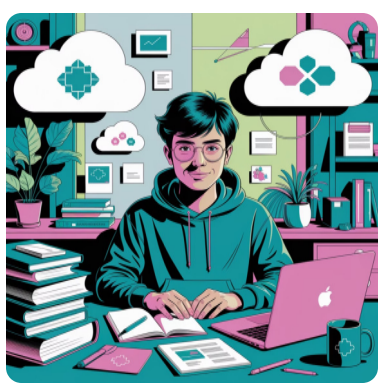
Our bootcamp participants tell us the same story: *"I was learning in circles before this."* What they appreciate most is having a structured path through the complexity.

What Makes Our Approach Different

Rather than teaching isolated technologies, we focus on the entire workflow. You'll understand not just *how* to use Terraform or Kubernetes, but *why* you'd choose them for specific scenarios and **how they integrate with everything else in your toolchain.**

The DevOps and DevSecOps bootcamp connects all the dots from the roadmap you just read.

From Overwhelmed to Confident in Months, Not Years



Most engineers we work with spent years trying to master these skills on their own. Through our program, you can **compress that journey into just a few months.**

Imagine having a **clear path** through all these technologies, someone who tells you what to learn exactly, to what level and with real-world projects.

So if this roadmap feels like a mountain of knowledge to climb alone, and you'd prefer a **guided journey** with expert support, our bootcamp might be the right fit for your learning style.

All the best on your cloud journey! :)

Join 1.3 M+ engineers

Free DevOps and Cloud tutorials

Join 200k+ engineers

Behind-the-scenes and career content