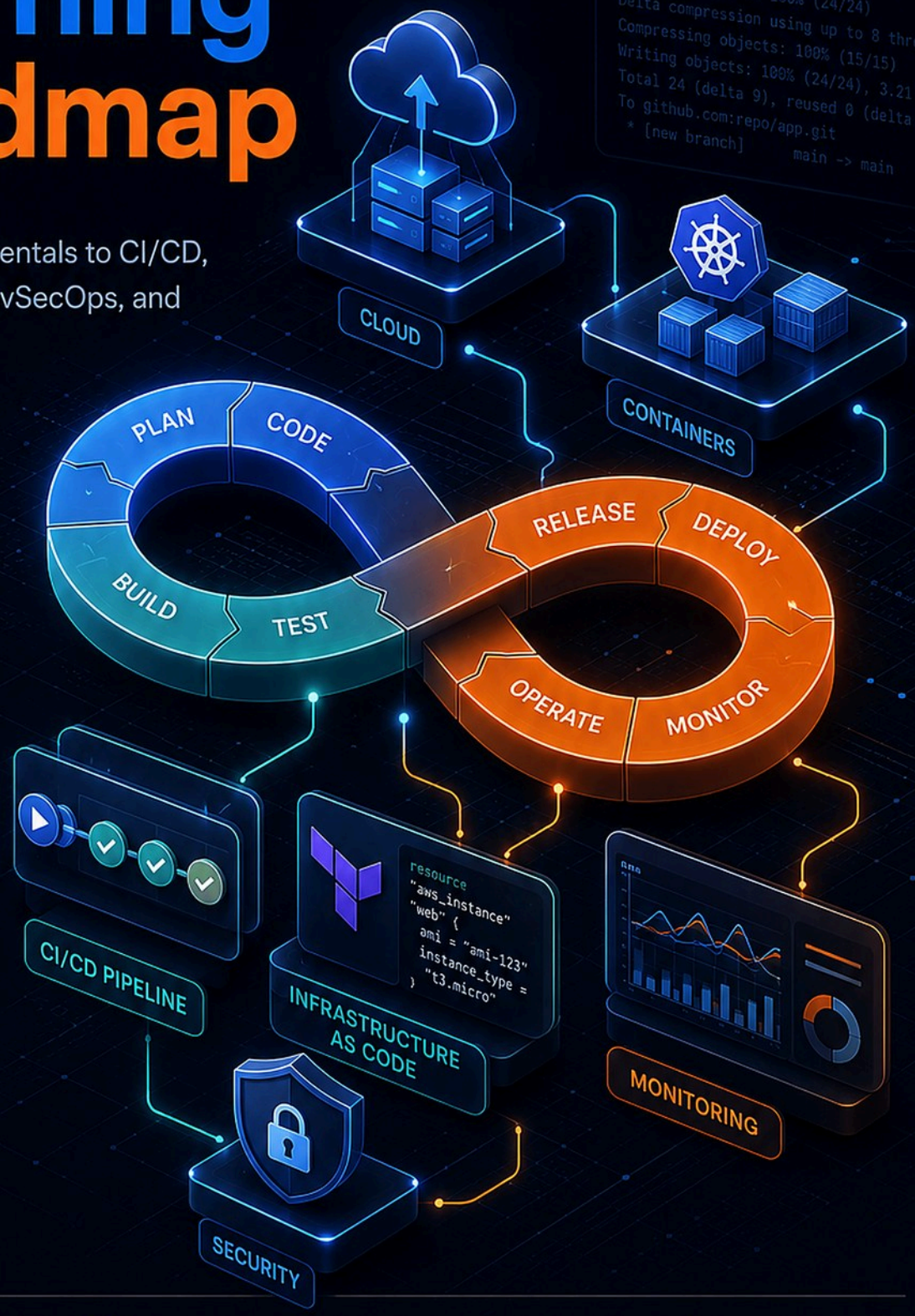
DevOps Learning Roadmap

FOR ENGINEERS

From cloud fundamentals to CI/CD, Kubernetes, IaC, DevSecOps, and certification.



TOOLS COVERED

- AWS
- Docker
- Kubernetes
- Terraform
- CI/CD
- Prometheus

Cloud & CI/CD

Docker & Kubernetes

IaC & Config

Monitoring & Security

TECHNOVIZE PUBLISHING

djangozen.com

DevOps Learning Roadmap

From cloud fundamentals to CI/CD, Kubernetes, IaC, DevSecOps, and certification

KC Ramo

Technovize Publishing
technovize.com

First Edition, 2026
Published by Technovize Publishing · technovize.com

Contents

Preface

DevOps is not a single tool or a job title you can pick up in a weekend — it is a way of building and running software that spans the cloud, automation, containers, delivery pipelines, infrastructure as code, monitoring, and security. The field is broad enough that newcomers often do not know where to start, and experienced engineers often have deep knowledge in one area and gaps in others. This book is a **roadmap**: a structured, hands-on path through the whole landscape, one topic at a time, from cloud fundamentals to the certifications that prove your skills.

Who this roadmap is for

This roadmap is for anyone who wants to become a capable DevOps engineer — developers moving toward operations, system administrators moving toward automation, students building a first portfolio, and working engineers filling gaps in their knowledge. It assumes basic comfort with the command line and a willingness to try things rather than only read about them. You do not need prior DevOps experience; each tutorial builds on the last, and every concept is introduced from the ground up before it is put to work.

How the roadmap is organized

The book is eleven tutorials, grouped into five parts that follow a natural learning progression. **Part I, Foundations**, covers cloud computing fundamentals and the programming and scripting skills every DevOps engineer needs. **Part II, Configuration & Containers**, covers configuration management and containerization with Docker and Kubernetes. **Part III, Delivery & Infrastructure**, covers building CI/CD pipelines and infrastructure as code. **Part IV, Operations & Security**, covers monitoring and observability and DevSecOps. **Part V, Advanced Practice & Career**, covers emerging trends, building a portfolio of real projects, and preparing for the certifications that open doors.

Every tutorial is hands-on

Reading about DevOps will only take you so far — the skills live in your hands. Every tutorial ends with a supplement section containing **lab exercises** you can run yourself (on free-tier cloud accounts), a **quiz** to check your understanding, and a **troubleshooting guide** for the errors you are most likely to hit. Do the labs. Break things, fix them, and delete your resources afterward to avoid charges. The muscle memory you build doing the work is what turns roadmap knowledge into real capability — and into the portfolio and certifications that get you hired.

How to use this book

You can read the tutorials in order for a complete path from beginner to job-ready, or jump to a specific topic to fill a gap. Either way, treat each tutorial as a unit: read the concepts, then do the labs, then take the quiz. The appendices gather a quick reference to the DevOps toolchain and a consolidated glossary you will return to. The field changes quickly — new tools appear, providers rename services, best practices evolve — but the fundamentals in this roadmap are durable, and they are the foundation on which you keep learning. Let us begin where every DevOps journey begins: in the cloud.

PART I

Foundations

CHAPTER 1

Cloud Computing Fundamentals for DevOps

IaaS, PaaS, SaaS — and AWS vs. Azure vs. GCP

Introduction

Before you can master CI/CD pipelines, container orchestration, or Infrastructure as Code, you need a solid foundation in cloud computing. Cloud platforms are the environment where virtually every modern DevOps workflow lives — from automated deployments to monitoring dashboards. Without understanding how the cloud is structured and what each provider offers, you'll constantly be fighting your tools instead of leveraging them.

This tutorial covers:

- What cloud computing is and why it matters for DevOps
- The three service models: IaaS, PaaS, and SaaS
- The three major cloud providers: AWS, Azure, and GCP
- How to choose the right platform for your DevOps projects
- Practical next steps to get hands-on experience

By the end, you'll be able to confidently discuss cloud architectures, understand which service model fits a given problem, and make informed decisions about which cloud provider to work with or recommend.

---

1. What Is Cloud Computing?

Cloud computing is the delivery of computing resources — servers, storage, databases, networking, software, and more — over the internet ("the cloud"), on a pay-as-you-go basis. Instead of buying and maintaining physical hardware in a data center, you rent the capacity you need from a cloud provider and scale up or down instantly.

For DevOps engineers, cloud computing isn't just a hosting convenience — it's the foundation of the entire modern software delivery workflow. Consider what DevOps depends on:

- **Automated build environments** — spun up on demand in the cloud
- **Scalable testing infrastructure** — dozens of parallel test runners
- **Staging and production environments** — mirroring each other consistently
- **Container registries and orchestration** — Kubernetes runs in the cloud
- **Monitoring and logging systems** — collecting data from cloud-hosted services

All of these either live in the cloud or interact with it. Understanding cloud fundamentals isn't optional — it's table stakes.

Key Characteristics of Cloud Computing

The National Institute of Standards and Technology (NIST) defines five essential characteristics of cloud computing that you should internalize:

1. **On-demand self-service** — You provision compute resources yourself, automatically, without needing to involve a human on the provider's side.
2. **Broad network access** — Resources are available over the internet and accessible from any device (laptop, phone, CI/CD runner, etc.).

3. **Resource pooling** — The provider's resources serve multiple customers simultaneously (multi-tenancy), with resources dynamically assigned based on demand.
4. **Rapid elasticity** — Resources can scale up or down quickly — sometimes automatically — to match demand. From the user's perspective, capacity appears unlimited.
5. **Measured service** — Cloud systems automatically monitor, control, and report resource usage, giving you full transparency on what you're consuming and being billed for.

These characteristics directly empower DevOps practices. Rapid elasticity enables autoscaling in production. On-demand self-service enables Infrastructure as Code. Measured service powers cost-aware engineering.

2. The Three Cloud Service Models

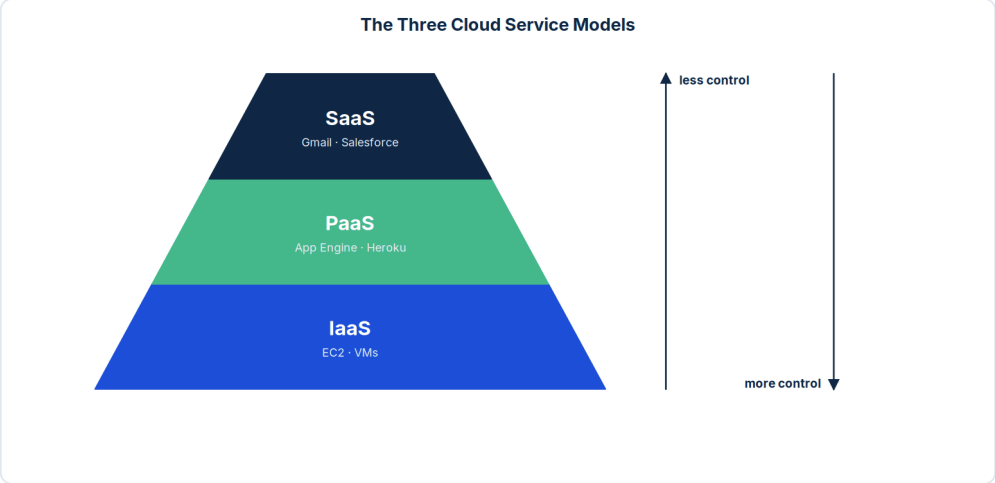


Figure 1.1 The three cloud service models and the control trade-off.

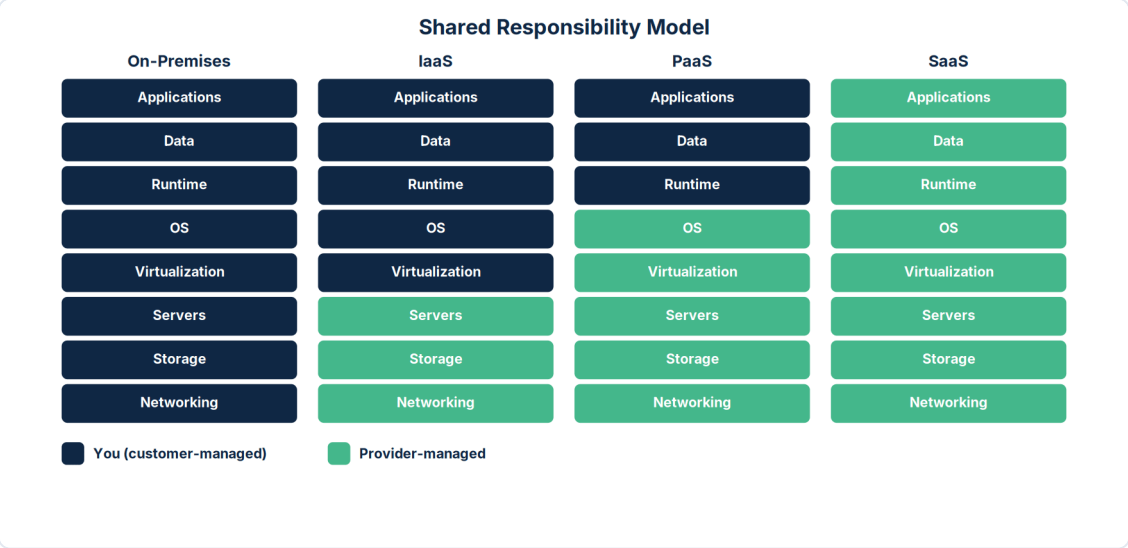


Figure 1.2 The shared responsibility model across On-Premises, IaaS, PaaS, and SaaS.

Cloud services are grouped into three models based on how much of the infrastructure stack the provider manages versus how much you manage yourself. Understanding this distinction is

critical because it determines your responsibilities, your flexibility, and your operational overhead.

2.1 Infrastructure as a Service (IaaS)

Definition: IaaS provides virtualized computing infrastructure — servers (virtual machines), networking, storage, and firewalls — over the internet. You get raw compute power and manage everything above the hypervisor yourself.

You manage: Operating system, runtime, middleware, applications, data
Provider manages: Physical hardware, hypervisor, networking fabric, storage hardware

Real-world examples:

- AWS EC2 (Elastic Compute Cloud)
- Azure Virtual Machines
- Google Compute Engine (GCE)
- DigitalOcean Droplets

When to use IaaS:

- You need full control over the OS and software stack
- You're migrating legacy applications from on-premises servers
- You have custom compliance or security requirements
- You're running databases that need specific OS-level tuning

DevOps relevance: IaaS is the backbone of manual and automated infrastructure provisioning. Tools like Terraform and Ansible were built to manage IaaS resources. When you define an EC2 instance in a Terraform configuration file, you're working at the IaaS layer.

Key trade-off: Maximum control, maximum responsibility. You must patch the OS, manage security groups, handle backups, and plan for failure.

2.2 Platform as a Service (PaaS)

Definition: PaaS provides a complete development and deployment environment in the cloud. The provider manages the underlying infrastructure (servers, OS, runtime), and you focus on writing and deploying application code.

You manage: Applications, data, configuration
Provider manages: OS, runtime, middleware, servers, networking, storage

Real-world examples:

- AWS Elastic Beanstalk
- Google App Engine
- Azure App Service
- Heroku (runs on AWS)
- AWS RDS (Relational Database Service) — managed database as PaaS

When to use PaaS:

- You want to focus on application development, not infrastructure operations

- You're building web apps, APIs, or microservices that follow standard patterns
- You want faster deployment cycles without managing servers
- Your team doesn't have dedicated infrastructure expertise

DevOps relevance: PaaS shrinks the operational surface area considerably. A CI/CD pipeline that deploys to AWS Elastic Beanstalk or Google App Engine needs far less infrastructure configuration than one deploying to raw EC2 instances. PaaS platforms handle scaling, load balancing, and health checks natively.

Key trade-off: Less operational overhead, but less control. You're constrained by what the platform supports. Unusual runtime requirements, custom kernel settings, or specific software versions may not be supported.

2.3 Software as a Service (SaaS)

Definition: SaaS delivers fully functional applications over the internet, accessible through a web browser or API. The provider manages everything — infrastructure, platform, and application code.

You manage: User configuration, data input, integrations
Provider manages: Everything else — hardware, OS, application code, updates, scaling

Real-world examples:

- GitHub / GitLab (version control, CI/CD)
- Jira (project management)
- Slack (team communication)
- Datadog (monitoring and observability)
- PagerDuty (incident management)
- Snyk (security scanning)

When to use SaaS:

- You need a tool or service without wanting to operate it yourself
- The functionality is commodity (email, chat, ticketing, monitoring)
- You want guaranteed uptime and automatic updates managed by the vendor

DevOps relevance: As a DevOps engineer, you'll constantly integrate SaaS tools into your pipelines and workflows. Your CI/CD system (GitHub Actions, GitLab CI) is a SaaS. Your monitoring dashboard (Datadog, Grafana Cloud) may be a SaaS. Your secret management system (HashiCorp Vault Cloud, AWS Secrets Manager) is a SaaS or managed service. Understanding SaaS helps you make build-vs-buy decisions.

Key trade-off: Zero operational overhead, zero control. You're entirely dependent on the vendor's reliability, pricing, and feature roadmap.

The Shared Responsibility Model Visualized

Here's how the three models compare in terms of what you manage:

Component	On-Premises	IaaS	PaaS	SaaS
Applications	You	You	You	Provider
Data	You	You	You	Provider
Runtime	You	You	Provider	Provider
Middleware	You	You	Provider	Provider
Operating System	You	You	Provider	Provider
Virtualization	You	Provider	Provider	Provider
Servers	You	Provider	Provider	Provider
Storage	You	Provider	Provider	Provider
Networking	You	Provider	Provider	Provider

As you move from left to right, operational responsibility shifts to the provider. As a DevOps engineer, you'll work across all three columns depending on the organization and the task.

3. The Three Major Cloud Providers

AWS, Azure, and Google Cloud Platform (GCP) collectively dominate the global cloud market. Each has distinct strengths, ecosystems, and sweet spots. Let's look at each from a DevOps perspective.

3.1 Amazon Web Services (AWS)

Market position: Largest cloud provider globally. AWS pioneered cloud computing when it launched EC2 and S3 in 2006 and has maintained its leadership position ever since.

Core DevOps services:

Category	AWS Service
Compute	EC2, Lambda, ECS, EKS
CI/CD	CodePipeline, CodeBuild, CodeDeploy
Containers	ECS (Elastic Container Service), ECR
Kubernetes	EKS (Elastic Kubernetes Service)
IaC	CloudFormation, CDK
Monitoring	CloudWatch, X-Ray
Secrets	AWS Secrets Manager, Parameter Store
Networking	VPC, Route 53, ALB, API Gateway

Strengths:

- Largest service catalog of any provider (200+ services)

- Most mature DevOps tooling and deepest third-party ecosystem
- Strongest community: the most Stack Overflow answers, tutorials, and courses reference AWS
- Most widely adopted — knowing AWS makes you the most employable in the job market

Weaknesses:

- Steeper learning curve due to the sheer number of services
- Console UX can be overwhelming for beginners
- Pricing is complex and can surprise teams with unexpected bills

Best suited for: Organizations looking for the broadest service selection, teams building on a well-established ecosystem, and DevOps engineers who want maximum job market coverage.

AWS Free Tier: AWS offers a generous free tier including 750 hours/month of EC2 t2.micro, 5GB of S3 storage, and free-tier access to Lambda, DynamoDB, and many other services for 12 months. Excellent for learning.

3.2 Microsoft Azure

Market position: Second largest cloud provider. Azure has grown rapidly due to Microsoft's deep enterprise relationships and its tight integration with the Windows and Office 365 ecosystem.

Core DevOps services:

Category	Azure Service
Compute	Azure VMs, Azure Functions, ACI
CI/CD	Azure DevOps (Pipelines), GitHub Actions
Containers	Azure Container Registry (ACR)
Kubernetes	AKS (Azure Kubernetes Service)
IaC	ARM Templates, Bicep
Monitoring	Azure Monitor, Application Insights
Secrets	Azure Key Vault
Networking	Azure Virtual Network, Front Door, API Management

Strengths:

- Seamless integration with Microsoft enterprise products (Active Directory, Office 365, Teams)
- Azure DevOps is a mature, all-in-one DevOps platform (boards, repos, pipelines, artifacts, test plans)
- Strong hybrid cloud story via Azure Arc — connect on-premises infrastructure to Azure
- GitHub Actions integration is best-in-class (Microsoft owns GitHub)
- Dominant in enterprise Windows environments

Weaknesses:

- Historically weaker Linux/open-source ecosystem support (though rapidly improving)
- Some services are less mature compared to AWS equivalents
- Azure portal and naming conventions can be confusing

Best suited for: Organizations already in the Microsoft ecosystem, enterprises running hybrid (on-premises + cloud) environments, and teams using GitHub as their primary SCM.

Azure Free Account: Azure offers $200 in credits for the first 30 days plus 12 months of popular free services including 750 hours of B1S VMs and 5GB of Blob Storage.

3.3 Google Cloud Platform (GCP)

Market position: Third largest provider. GCP is smaller in total market share but punches well above its weight in specific domains — particularly data analytics, machine learning, and Kubernetes (which Google invented).

Core DevOps services:

Category	GCP Service
Compute	Compute Engine, Cloud Run, GKE
CI/CD	Cloud Build, Cloud Deploy
Containers	Artifact Registry, Container Registry
Kubernetes	GKE (Google Kubernetes Engine)
IaC	Deployment Manager, Terraform (preferred)
Monitoring	Cloud Monitoring, Cloud Logging, Cloud Trace
Secrets	Secret Manager
Networking	VPC, Cloud Load Balancing, Cloud Armor

Strengths:

- Invented Kubernetes — GKE is widely considered the most polished managed Kubernetes service
- Outstanding data and analytics services: BigQuery, Dataflow, Pub/Sub
- Best-in-class ML and AI services (Vertex AI, AutoML)
- Often more cost-efficient for compute at scale
- Very strong networking backbone (Google's global private network)

Weaknesses:

- Smaller service catalog than AWS
- Smaller community and ecosystem — fewer third-party integrations and tutorials
- Google has a history of deprecating products (though core cloud infrastructure services are stable)

Best suited for: Data engineering and ML-heavy workloads, Kubernetes-first architectures, organizations that want to leverage Google's global network and AI capabilities.

GCP Free Tier: GCP offers a $300 credit for new accounts (valid 90 days) plus an always-free tier including 1 f1-micro instance per month, 5GB of Cloud Storage, and limited BigQuery usage.

Provider Comparison at a Glance

Dimension	AWS	Azure	GCP
Market Share	~33%	~22%	~11%
Strongest Domain	General / Enterprise	Enterprise / Hybrid	Data / ML / Kubernetes
Kubernetes Service	EKS	AKS	GKE (best-in-class)
CI/CD Native Tool	CodePipeline	Azure Pipelines	Cloud Build
Free Tier	12 months + always-free	$200 credit + 12 months	$300 credit + always-free
Certification Value	Very High	High	Medium-High
Learning Resources	Most abundant	Abundant	Growing
Best Entry Point	For broadest job market	For Microsoft shops	For data/ML/K8s focus

4. Deployment Models

Beyond service models, cloud computing also has deployment models — how and where the infrastructure is hosted.

- **Public Cloud:** Resources are owned and operated by a third-party cloud provider (AWS, Azure, GCP) and delivered over the internet. Shared infrastructure, multi-tenant. This is what most DevOps teams use.
- **Private Cloud:** Cloud infrastructure is operated exclusively for a single organization. Can be on-premises or hosted by a third party. More control, more cost.
- **Hybrid Cloud:** A mix of public and private cloud, often connected via VPN or dedicated network links. Common in enterprises migrating from on-premises data centers.
- **Multi-Cloud:** Using multiple public cloud providers simultaneously. Reduces vendor lock-in, increases complexity. Increasingly common in large enterprises.

Most DevOps engineers begin with public cloud and encounter hybrid or multi-cloud as they advance into senior or architect-level roles.

5. Why Cloud Fundamentals Matter for DevOps Specifically

DevOps is about shortening the software delivery lifecycle while maintaining quality and reliability. Cloud computing directly enables every pillar of DevOps:

DevOps Practice	Cloud Enabler
Continuous Integration	Cloud-hosted CI runners (GitHub Actions, AWS CodeBuild)
Continuous Delivery	Managed deployment services (Elastic Beanstalk, App Engine)
Infrastructure as Code	Cloud APIs that IaC tools (Terraform, Pulumi) call
Monitoring & Observability	Cloud-native monitoring (CloudWatch, Azure Monitor)
Containerization	Managed container registries and orchestration (EKS, GKE, AKS)
Scalability & Reliability	Auto Scaling Groups, managed databases, multi-region deployments
Security	IAM, secrets managers, compliance tooling built into each provider

Understanding cloud fundamentals means you can design systems that are observable, scalable, automated, and secure — the four pillars of good DevOps work.

---

6. Practical Next Steps

Reading about cloud computing will only get you so far. Here's how to build hands-on knowledge:

Step 1: Create Free Accounts

Sign up for free tiers on all three providers. You don't need to use all three immediately — start with one — but having accounts set up lets you experiment freely.

- AWS Free Tier
- Azure Free Account
- GCP Free Trial

Step 2: Deploy a Simple Web App

Pick one provider and deploy a simple web application. For AWS, use Elastic Beanstalk. For Azure, use App Service. For GCP, use App Engine or Cloud Run. This gives you immediate experience with IaaS/PaaS workflows.

Step 3: Explore the Console

Spend an hour exploring the provider's management console — EC2, S3, VPCs on AWS; Virtual Machines, Blob Storage, VNets on Azure; Compute Engine, Cloud Storage, VPCs on GCP. Understanding the mental model of each console will pay dividends when you start automating with IaC.

Step 4: Learn the CLI

Every cloud provider has a command-line interface:

- AWS: `aws` (AWS CLI)
- Azure: `az` (Azure CLI)
- GCP: `gcloud` (Google Cloud SDK)

Start running basic commands: listing resources, creating a storage bucket, checking your billing. The CLI is your bridge to automation.

Step 5: Set a Budget Alert

Before experimenting, set a billing alert to avoid surprise charges. All three providers let you set email alerts when your spend exceeds a threshold. On AWS, this is done via Billing > Budgets. On Azure, via Cost Management. On GCP, via Billing > Budgets & Alerts.

---

7. Summary

Let's bring it all together:

- **Cloud computing** delivers compute, storage, and networking over the internet, enabling the on-demand, scalable infrastructure that DevOps depends on.
- **IaaS** gives you raw virtualized infrastructure — maximum control, maximum responsibility. Think EC2 virtual machines.
- **PaaS** abstracts away the OS and runtime — you deploy code, the platform handles the rest. Think Elastic Beanstalk or App Engine.
- **SaaS** delivers fully managed applications — GitHub, Datadog, PagerDuty are all SaaS tools in a typical DevOps toolchain.
- **AWS** leads the market with the deepest service catalog and the largest ecosystem. Best for job market breadth.
- **Azure** excels in enterprise Windows environments and hybrid cloud. Best for Microsoft-heavy organizations.
- **GCP** leads in Kubernetes (via GKE), data analytics, and AI/ML. Best for data-intensive and Kubernetes-first architectures.

As a DevOps engineer, you'll likely touch all three at some point in your career. The fundamentals — service models, deployment models, shared responsibility — transfer across all providers. Master the concepts, then learn the provider-specific implementations.

What's Next

In **Tutorial 2**, we'll cover **Programming & Scripting for DevOps** — the Python and Bash skills you need to automate infrastructure, write deployment scripts, and build monitoring tools. Cloud fundamentals give you the environment; scripting gives you the power to automate it.

Key Terms Glossary

Term	Definition
Cloud Computing	Delivery of computing services over the internet on a pay-as-you-go basis
IaaS	Infrastructure as a Service — virtualized hardware rented from a provider

Term	Definition
PaaS	Platform as a Service — managed platform for deploying application code
SaaS	Software as a Service — fully managed application accessible via browser or API
Elasticity	The ability to scale resources up or down rapidly based on demand
Multi-tenancy	Multiple customers sharing the same underlying physical infrastructure
Hybrid Cloud	Combination of public and private cloud environments
Region	A geographic area containing multiple cloud data centers
Availability Zone	An isolated data center within a region, used for fault tolerance
IAM	Identity and Access Management — controls who can access what cloud resources

Hands-On Labs, Quiz & Troubleshooting

Section A: Lab Exercises

Lab 1.1 — Classify AWS Services by Model

Objective: Correctly classify AWS services as IaaS, PaaS, or SaaS.
Time: 20 minutes | **Cost:** Free

Navigate to the AWS Console and classify each service:

Service	Your Answer	Correct Answer	What You Manage
EC2		IaaS	OS, runtime, app, data
Elastic Beanstalk		PaaS	App code and data only
RDS		PaaS	Schema, queries, data
Lambda		PaaS (serverless)	Function code only
WorkMail		SaaS	User accounts, email content
S3		PaaS	Bucket policies, data
Chime		SaaS	Users and meeting settings

Lab 1.2 — Launch Your First Virtual Machine

Objective: Launch, connect to, and terminate an EC2 instance.
Time: 45 minutes | **Cost:** Free tier

```
# Configure AWS CLI
aws configure

# Create a key pair
aws ec2 create-key-pair \
  --key-name my-first-keypair \
```

```
  --query 'KeyMaterial' \
  --output text > my-first-keypair.pem
chmod 400 my-first-keypair.pem

# Find latest Amazon Linux 2 AMI
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2" \
  --query 'sort_by(Images, &CreationDate)[-1].ImageId' \
  --output text)

# Launch instance
INSTANCE_ID=$(aws ec2 run-instances \
  --image-id $AMI_ID \
  --instance-type t2.micro \
  --key-name my-first-keypair \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=lab-vm}]' \
  --query 'Instances[0].InstanceId' \
  --output text)

echo "Instance ID: $INSTANCE_ID"

# Wait for instance to be running
aws ec2 wait instance-running --instance-ids $INSTANCE_ID

# Get public IP
PUBLIC_IP=$(aws ec2 describe-instances \
  --instance-ids $INSTANCE_ID \
  --query 'Reservations[0].Instances[0].PublicIpAddress' \
  --output text)

echo "Connect with: ssh -i my-first-keypair.pem ec2-user@$PUBLIC_IP"

# SSH in and explore
ssh -i my-first-keypair.pem ec2-user@$PUBLIC_IP
# Inside: uname -a | df -h | free -m | curl ipinfo.io

# TERMINATE when done (avoid charges)
aws ec2 terminate-instances --instance-ids $INSTANCE_ID
```

Reflection: What did you manage that a PaaS would have handled automatically?

Lab 1.3 — Deploy a PaaS Application

Objective: Deploy a Flask app to Elastic Beanstalk and compare effort to Lab 1.2.
Time: 30 minutes | **Cost:** Free tier

```
pip install awsebcli

mkdir paas-demo && cd paas-demo

cat > application.py << 'EOF'
from flask import Flask
application = Flask(__name__)

@application.route('/')
def hello():
    return '<h1>PaaS Demo</h1><p>No OS management required!</p>'
```

```
EOF

echo "flask==3.0.0" > requirements.txt

eb init paas-demo --platform python-3.11 --region us-east-1
eb create paas-demo-env
eb open

# Clean up
eb terminate paas-demo-env
```

Lab 1.4 — Set Up Billing Alerts on All Three Providers

Objective: Prevent unexpected charges during your cloud learning journey.
Time: 20 minutes | **Cost:** Free

```
# AWS: Create a $10 budget alert
aws budgets create-budget \
  --account-id $(aws sts get-caller-identity --query Account --output text) \
  --budget '{
    "BudgetName": "learning-budget",
    "BudgetLimit": {"Amount": "10", "Unit": "USD"},
    "TimeUnit": "MONTHLY",
    "BudgetType": "COST"
  }' \
  --notifications-with-subscribers '[{
    "Notification": {
      "NotificationType": "ACTUAL",
      "ComparisonOperator": "GREATER_THAN",
      "Threshold": 80
    },
    "Subscribers": [{
      "SubscriptionType": "EMAIL",
      "Address": "your@email.com"
    }]
  }]'

# GCP: Set a budget alert in the Console
# Billing -> Budgets & Alerts -> Create Budget -> $10 -> 80% threshold

# Azure: Set a budget in Cost Management
# Cost Management -> Budgets -> Add -> $10 -> Alert at 80%
```

Lab 1.5 — Explore Regions and Availability Zones

Objective: Understand geographic cloud distribution and its DevOps implications.
Time: 20 minutes | **Cost:** Free

```
# List all AWS regions
aws ec2 describe-regions --output table

# List AZs in us-east-1
aws ec2 describe-availability-zones \
  --region us-east-1 \
  --output table
```

```
# Count total regions
aws ec2 describe-regions \
  --query 'length(Regions)' \
  --output text

# Compare latency to different regions (ping the public endpoints)
for region in us-east-1 eu-west-1 ap-southeast-1; do
  echo -n "Latency to $region: "
  ping -c 3 ec2.$region.amazonaws.com 2>/dev/null | \
    grep 'avg' | awk -F'/' '{print $5"ms"}'
done
```

Exercise: Draw a diagram showing 1 region, 3 AZs, and how Multi-AZ RDS protects against data center failure.

Section B: Quiz — 10 Questions

Q1. A company needs kernel-level OS configuration. Which model fits?

- A) SaaS B) PaaS C) **IaaS** (correct) D) FaaS

Rationale: Kernel access requires direct OS control — only IaaS provides this.

Q2. Which is SaaS in a DevOps toolchain?

- A) EC2 B) GCE C) **GitHub** (correct) D) Azure VMs

Rationale: GitHub is a fully managed application. The others are IaaS virtual machines.

Q3. A startup wants the fastest deployment with no server management. Standard Django app. Best fit?

- A) IaaS B) **PaaS** (correct) C) SaaS D) On-premises

Rationale: Standard apps with no special OS requirements are ideal PaaS candidates.

Q4. Which NIST characteristic describes dynamic scaling to match demand?

- A) On-demand self-service B) Broad network access C) Resource pooling D) **Rapid elasticity** (correct)

Rationale: Rapid elasticity = scale up/down quickly, appearing unlimited to the consumer.

Q5. Which provider has the strongest native Active Directory integration?

- A) AWS B) **Azure** (correct) C) GCP D) All equal

Rationale: Azure AD (Entra ID) natively integrates with on-premises Active Directory via Azure AD Connect.

Q6. Which provider invented Kubernetes?

- A) AWS B) Azure C) **GCP** (correct) D) Red Hat

Rationale: Google open-sourced Kubernetes in 2014, derived from their internal Borg system.

Q7. What is ALWAYS the customer's responsibility across all service models?

- A) OS B) Virtualization C) **Customer data** (correct) D) Physical servers

Rationale: Regardless of service model, the customer always owns their data.

Q8. A region goes down entirely. Which architecture keeps the app available?

- A) Multi-AZ in one region B) **Multi-region with Route 53 failover** (correct) C) Larger instance D) More replicas in one region

Rationale: Multi-AZ protects against AZ failure, not regional failure. Only multi-region protects against full regional outage.

Q9. On-premises + public cloud connected via VPN is called:

- A) Public cloud B) Private cloud C) **Hybrid cloud** (correct) D) Multi-cloud

Rationale: Hybrid cloud = on-premises + public cloud, connected via network.

Q10. Primary workloads: BigQuery analytics, TensorFlow training, Kubernetes microservices. Best provider?

- A) AWS B) Azure C) **GCP** (correct) D) All equal

Rationale: GCP invented BigQuery, TensorFlow, and Kubernetes — uniquely strong for all three workloads.

Section C: Troubleshooting Guide

Problem 1: AWS CLI — "Unable to locate credentials"

```
# Fix
aws configure
# Enter: Access Key ID, Secret Access Key, region, output format (json)

# Or via environment variables
export AWS_ACCESS_KEY_ID="your-key"
export AWS_SECRET_ACCESS_KEY="your-secret"
export AWS_DEFAULT_REGION="us-east-1"
```

Problem 2: SSH times out on Lab 1.2

```
# Most likely cause: Security group blocks port 22
MY_IP=$(curl -s ifconfig.me)
aws ec2 authorize-security-group-ingress \
  --group-id sg-xxxxxxxx \
  --protocol tcp --port 22 \
  --cidr "${MY_IP}/32"

# Wrong key permissions fix
```

```
chmod 400 my-first-keypair.pem

# Wrong username per AMI type:
# Amazon Linux 2 -> ec2-user
# Ubuntu         -> ubuntu
# RHEL           -> ec2-user
# Debian         -> admin
```

Problem 3: Elastic Beanstalk Python version error

```
# List available platforms
eb platform list

# Reinitialize with correct platform
eb init paas-demo \
  --platform "Python 3.11 running on 64bit Amazon Linux 2023" \
  --region us-east-1
```

Problem 4: Unexpected AWS charges

```
# Find all running instances immediately
aws ec2 describe-instances \
  --filters "Name=instance-state-name,Values=running" \
  --query 'Reservations[*].Instances[*].[InstanceId,InstanceType,LaunchTime]' \
  --output table

# Terminate all running instances (use with caution)
aws ec2 describe-instances \
  --filters "Name=instance-state-name,Values=running" \
  --query 'Reservations[*].Instances[*].InstanceId' \
  --output text | xargs aws ec2 terminate-instances --instance-ids
```

Section D: Flashcards

Concept	Definition
IaaS	Virtualized infrastructure — you manage OS and above
PaaS	Managed platform — you manage app code and data only
SaaS	Fully managed application accessed via browser/API
Rapid elasticity	Ability to scale resources up/down quickly on demand
Multi-tenancy	Multiple customers sharing underlying physical infrastructure
Cloud Region	Geographic area containing multiple Availability Zones
Availability Zone	Isolated data center within a region
Hybrid cloud	On-premises + public cloud connected via network
Shared responsibility	Division of security duties between provider and customer
Who invented Kubernetes?	Google (open-sourced 2014 from internal Borg system)

End of sample

You have read the opening chapter of

DevOps Learning Roadmap

The complete book continues from here — every chapter, every appendix, in PDF and EPUB.

EUR 39.00

Get the full book

https://djangozen.com/ebooks/book/devops-learning-roadmap/