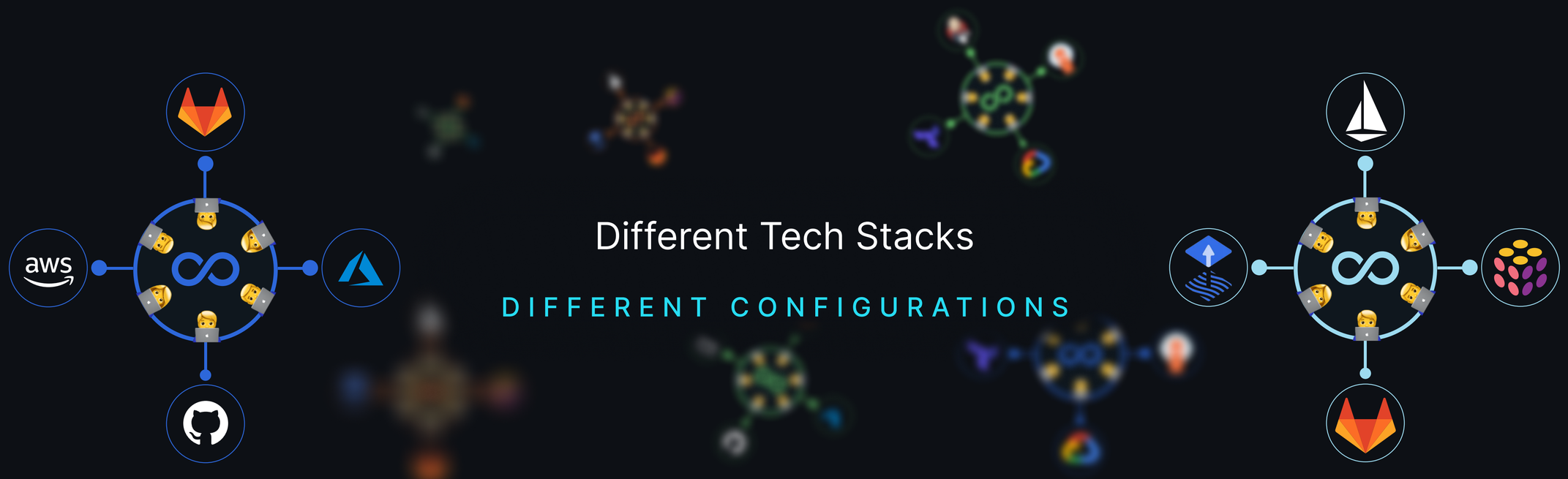
Platform Engineering Guide

KodeKloud

In the landscape of modern cloud adoption, organizations often encounter a scenario where numerous DevOps teams, each utilizing distinct tools and technology stacks, are engaged in application development. This diversity, while offering flexibility, introduces a complex array of challenges.



Required Tools in a Diverse DevOps Ecosystem

When managing applications across a diverse landscape, certain tools become essential.

Version Control Systems (VCS):

Essential for tracking and managing changes in the codebase.

Continuous Integration (CI):

Automates the integration of code changes, ensuring early detection of issues.

Continuous Deployment (CD):

Automates the delivery of applications to selected infrastructure.

Runtime Environment:

Ensures the application runs smoothly in a stable and scalable environment.

Infrastructure Provisioning:

Involves automated setup of required infrastructure, aligning closely with Infrastructure as Code (IaC) practices.

Logging and Monitoring:

Critical for observing application performance and identifying issues.

Security:

Encompasses measures to protect applications from threats and vulnerabilities.

Challenges in Traditional App Deployment

Scaling Difficulties:

As new teams form, replicating expertise and maintaining standards becomes challenging.

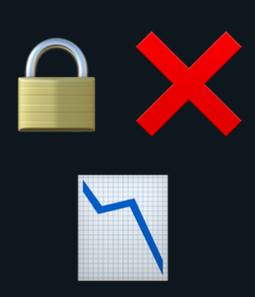
Organizational Inconsistency:

Varied practices across teams can result in a lack of uniformity in processes and standards.

In light of these challenges, the traditional mantra of "You build it, you run it" becomes less feasible.

Diverse Expertise Required:

Teams need to possess expertise in multiple domains such as Kubernetes, cloud technologies, security, and compliance, which can be demanding and unrealistic.



Inefficient Security and Compliance:

Having one team for security and compliance working with multiple teams, each with different tech stacks, leads to inefficiency and inconsistency.

Role of Platform Engineering

In the role of Platform Engineering, the primary focus is to establish a "paved road" that is easy to use and meets company standards, enabling developers to deploy their applications effortlessly. The objective is to minimize developer friction within the framework of company regulations. This approach aims to eliminate barriers that hinder developers from delivering their work by making the process straightforward and user-friendly.

Essentially, it's as simple as, "Place your application here, and the Platform Engineering Team will take care of the deployment for you. The team guarantees that the deployment complies with company standards and regulations.

Subsequently, they develop a layer of abstraction over these tools, incorporating an intuitive interface, such as a UI, API, CLI-based or simply code-based. This enables software development teams to independently access and utilize any service or tool they require through self-service capabilities using a catalog.

CI

Circle CI · Codefresh · Github · Bitbucket

Gitlab · Gitlab Actions · CodeDeploy

Jenkins · Travis · Azure DevOps

Google Cloud Build

Registry

Docker · Azure Container Registry · JFrog

Harbor · AWS ECR Registry

Google Container Registry

Messaging

RabbitMQ · Kafka · ActiveMQ

Infrastructure Control Plane

Terraform · Atlantis · Spacelift

Service Catalog / Developer Portal / Platform UI

Backstage · LeanIX · Port

Platform Orchestrator

Humanitec

CD / Operators

Jenkins · ArgoCD

Kubernetes Control Plane

Kubermatic · Redhat · Rafay · GiantSwarm

Ambassador · Okteto

Database & Storage

Aiven · PostgresSQL · Redis

Amazon S3 · MariaDB · MySQL

ElasticSearch · MongoDB

Security

Azure Sentinel · Snyk · Gremlin

Armo · Tigera

Logging

Google Stack Driver · Fluentbit · Logzio

Datadog

IoC

Terraform · Pulumi · Cloud Formation

DNS

CloudFlare · Route53 · Google Public DNS

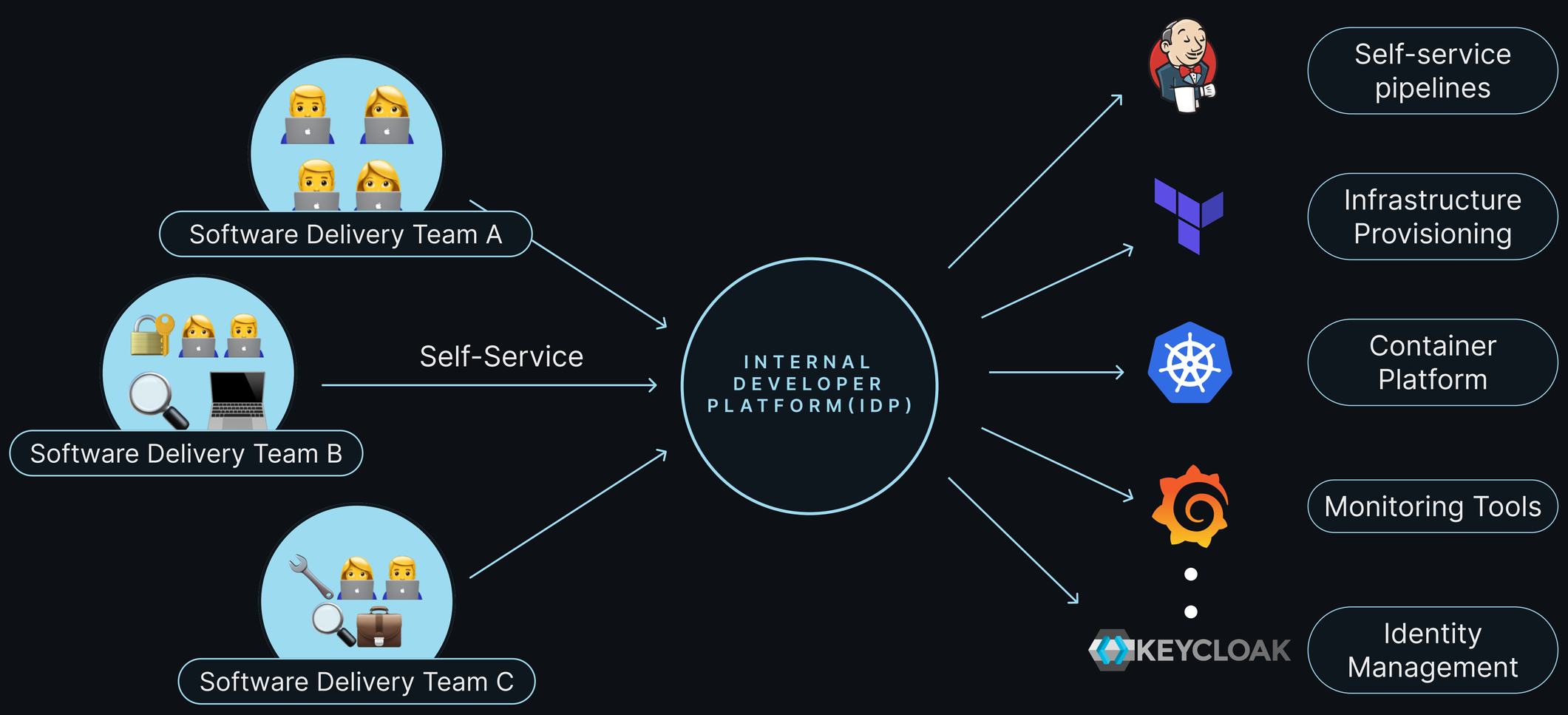
Cloud Providers

Azure · Google Cloud Platform

Amazon Web Services

These services are made accessible through a user-friendly interface, simplifying the interaction and usage for software development teams.

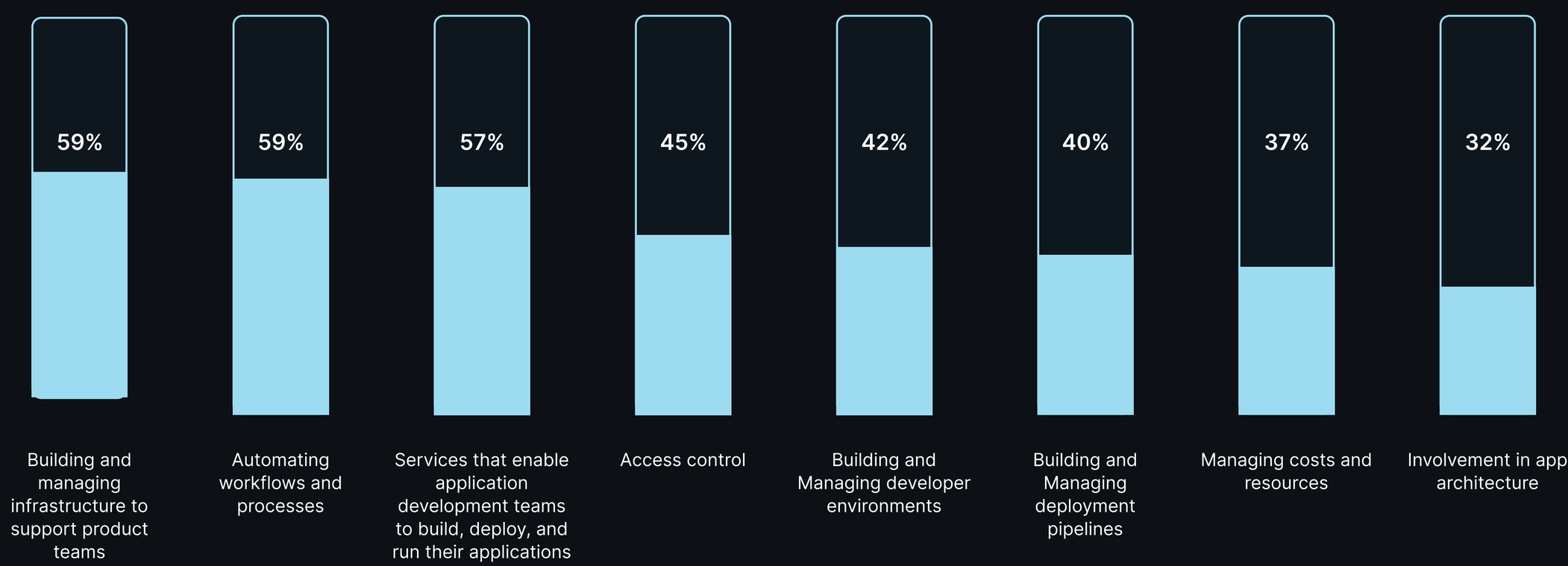
This arrangement forms a platform. Given that developers can log in and independently access resources or services without the necessity of contacting specific individuals or the platform team and waiting for resources, it operates as an **Internal Developer Platform (IDP)**, effectively acting as a Platform as a Service (PaaS) for our internal developers.



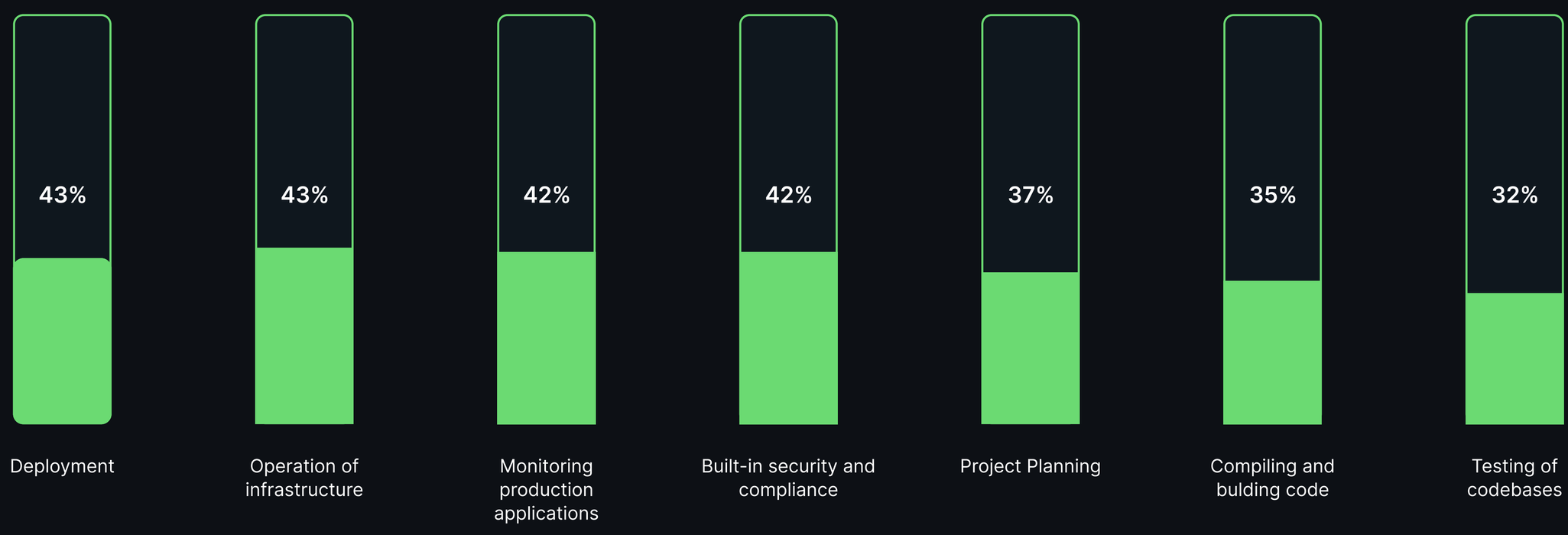
State of DevOps Report:Platform Engineering Edition 2023

by puppet

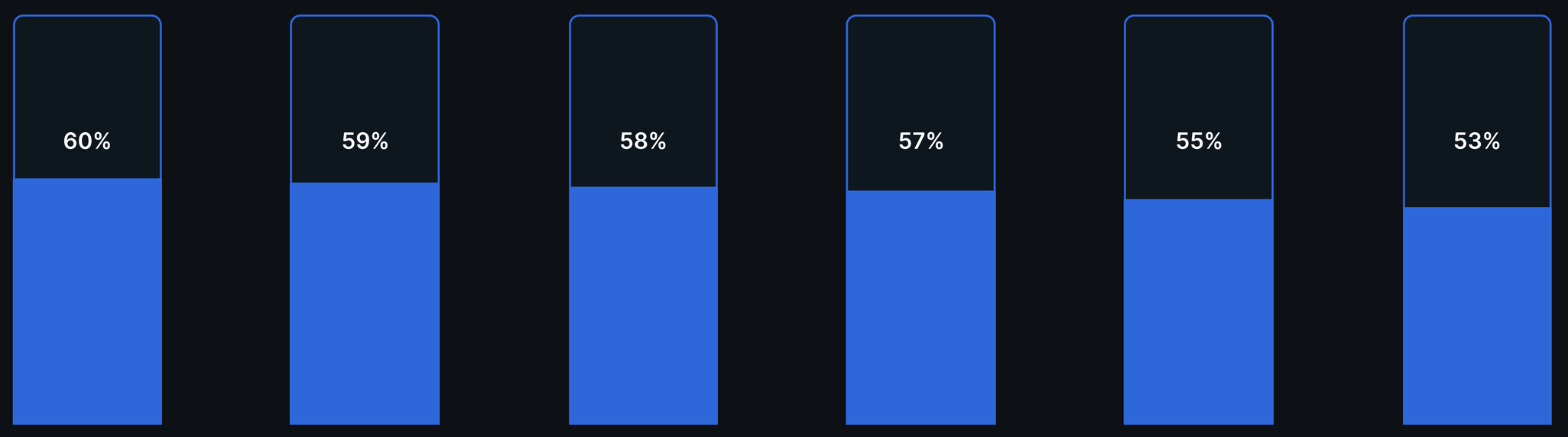
What services should fall within the platform team's scope of work?



What capabilities does the self-service platform offer?



What are the benefits of platform engineering?



Download the Full Report

Daily Tasks of Platform Engineering Role

Members of platform engineering teams have the primary responsibility of constructing, deploying, and managing the Internal Developer Platform or IDP that equips software developers with essential tools, APIs, services, and data storage solutions, enabling them to create new products.

IN THEIR DAILY ACTIVITIES,

🎨🏗️🤖

Design, implement and maintain a company's Technology Infrastructure such as servers, network resources, storage solutions, and cloud services.

🔄🔍🏗️🚀

Engage with the product development, architecture, governance and compliance, management teams and other stakeholders to ensure that the platform team constructs the platform correctly.

👀📊🔒

Monitoring the overall system to assess its performance, security, and reliability.

🔒🚀📋

Embedding security measures and compliance protocols in the platform, including encryption and access control.

👨‍🔬🔄🚀

Investigating and integrating cutting-edge technologies into the platform, while evaluating their potential implications.

📄📘🔄

Creating comprehensive documentation of configurations and procedures to facilitate different teams' understanding and use of the platform.

🐞🔍🚧

Debugging and Resolving infrastructure and application issues

Roadmap to become a Platform Engineer

1

Basics of Computing and Networking

(1-2 MONTHS)

WHAT TO LEARN:

Linux Fundamentals: Understand scripting, filesystem, memory, processes, package management, and basic security.

Networking Basics: Learn about TCP/IP, UDP, HTTP/HTTPS, DNS, SSH, and the ISO-OSI model. Understanding network protocols is essential for troubleshooting and designing scalable systems. Also, Proxy, Firewall, Reverse-Proxy and Network topologies knowledge is a must.

WHY:

Knowledge of Linux and networking is fundamental for troubleshooting, system administration, and understanding how applications communicate over the network.

TECHNOLOGIES:

- Linux (Ubuntu or CentOS)
- Basic networking tools (ping, traceroute, netstat)

Linux Ubuntu CentOS

2

Programming and Version Control

(2 MONTHS)

WHAT TO LEARN:

Programming Languages: Python and Go are widely used for their simplicity and efficiency in automation, respectively.

Version Control: Git, to manage code changes and collaborate with others.

WHY:

Programming skills are necessary for automating tasks, writing infrastructure as code, and developing internal tools. Version control is essential for collaboration and managing codebases.

TECHNOLOGIES:

- Python, Go
- Git, GitHub/GitLab

Python Go Git Github

3

Database and Storage Concepts

(1 MONTH)

WHAT TO LEARN:

Types of Databases: Differentiate between SQL and NoSQL databases; learn usage scenarios for PostgreSQL, MySQL, MongoDB, and Cassandra.

Database Operations: Understand CRUD operations, transactions, and basic performance optimization techniques.

Cloud-based Database Services: Explore AWS RDS, Google Cloud SQL, Azure SQL Database, and NoSQL services like DynamoDB, Firestore, and Cosmos DB.

Caching, CDN

WHY:

Mastery of databases is crucial for choosing effective data storage solutions, administration, and troubleshooting in platform egineering, directly impacting application performance and scalability.

Cloud-based database services offer managed, scalable, and highly available data storage solutions, reducing the overhead of manual database administration.

TECHNOLOGIES:

- **SQL Databases:** PostgreSQL, MySQL
- **NoSQL Databases:** MongoDB, Cassandra
- **Cloud Services:** AWS RDS, Google Cloud SQL, Azure SQL Database, DynamoDB, Firestore, Cosmos DB

PostgreSQL MongoDB AzureSQL

4

Cloud Fundamentals

(1-2 MONTHS)

WHAT TO LEARN:

Cloud Service Models: Understand SaaS, PaaS, IaaS, and serverless computing.

Major Cloud Providers: Basics of AWS, GCP, and Azure, focusing on compute, storage, and networking services.

WHY:

Cloud computing is the backbone of modern platform engineering, offering scalability, reliability, and a wide array of services for building and running applications.

TECHNOLOGIES:

- AWS, GCP, Azure
- Serverless frameworks (AWS Lambda, Google Cloud Functions)

AWS GCP Azure

5

Infrastructure as Code (IaC) and Configuration Management

(2 MONTHS)

WHAT TO LEARN:

IaC Concepts: Automate infrastructure provisioning using code.

Configuration Management: Automate the configuration of software and systems.

WHY:

IaC and configuration management enable scalable, reproducible, and manageable infrastructure provisioning and maintenance.

TECHNOLOGIES:

- Terraform, CloudFormation
- Ansible, Puppet, Chef, Pulumi

Terraform CloudFormation Ansible Pulumi

6

Continuous Integration and Continuous Deployment (CI/CD)

(3 MONTHS)

WHAT TO LEARN:

CI/CD Concepts: Automate the building, testing, and deployment of applications.

CI/CD Tools: Implement pipelines that integrate code changes efficiently and reliably.

WHY:

CI/CD practices are essential for fast, reliable software releases and are central to DevOps practices.

TECHNOLOGIES:

- Jenkins, GitLab CI, GitHub Actions
- ArgoCD, FluxCD for GitOps

Jenkins GitLab CI ArgoCD FluxCD

7

Containers and Orchestration

(3 MONTHS)

WHAT TO LEARN:

Container Basics: Docker, container registries, and container runtime.

Container Orchestration: Kubernetes fundamentals, including pods, deployments, services, and ingress.

WHY:

Containers and orchestration tools are key to developing, deploying, and managing applications consistently and efficiently across different environments.

TECHNOLOGIES:

- Docker
- Kubernetes, Helm

Docker Kubernetes Helm

8

Observability

(1-2 MONTHS)

WHAT TO LEARN:

Monitoring and Logging: Collect and analyze metrics, logs, and traces to understand system behavior.

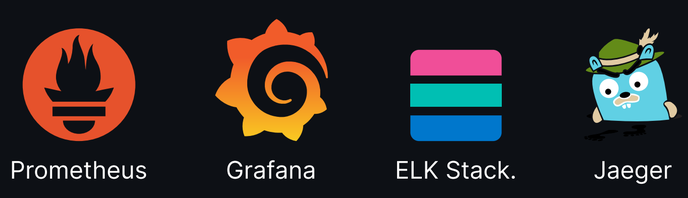
Alerting and APM: Implement alerting strategies and application performance monitoring.

WHY:

Observability is crucial for maintaining system reliability, performance, and troubleshooting issues proactively.

TECHNOLOGIES:

- Prometheus, Grafana
- Elasticsearch, Logstash, Kibana (ELK Stack)
- Datadog, New Relic



9

Security and Compliance

(2 MONTHS)

WHAT TO LEARN:

DevSecOps: Integrate security practices throughout the DevOps lifecycle.

Security Fundamentals: Understand IAM, encryption, network security, and compliance standards.

WHY:

Security is paramount in protecting data, ensuring privacy, and maintaining trust in platform services.

TECHNOLOGIES:

- Vault, OPA
- SAST/DAST tools, container scanning

10

Advanced Cloud-Native Technologies

(2-3 MONTHS)

WHAT TO LEARN:

Service Mesh: Istio, Linkerd for advanced network traffic control, and security.

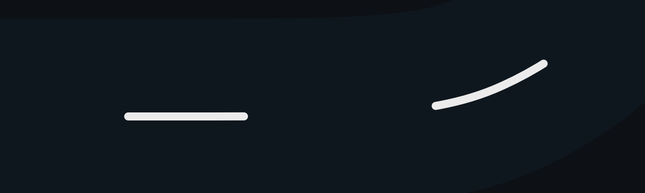
Serverless and Microservices: Best practices, patterns, and pitfalls.

WHY:

Advanced cloud-native technologies offer sophisticated patterns for scaling, managing, and securing microservices and serverless applications.

TECHNOLOGIES:

- Istio, Linkerd
- Knative, Open FaaS, Kubeless



11

Soft Skills

(CONTINUOUS)

WHAT TO LEARN:

Communication: Effective writing, speaking, Team working and collaboration skills.

Problem-Solving: Critical thinking and the ability to navigate ambiguity.

WHY:

Effective communication, teamwork and collaboration is crucial for platform engineers, who manage the infrastructure essential for product development teams

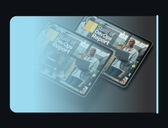
Puppet's 2023 State of Platform Engineering Report highlights:

61% prioritize communication skills.

54% value collaboration skills.

45% emphasize translating user requests into requirements.

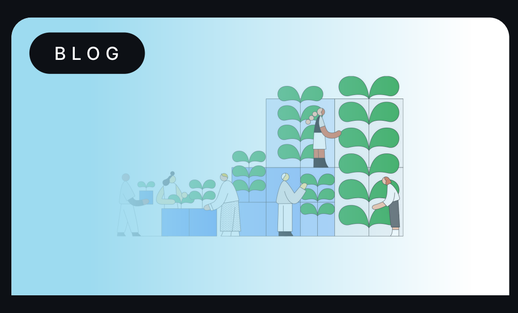
37% appreciate questioning established practices



Download the Full Report

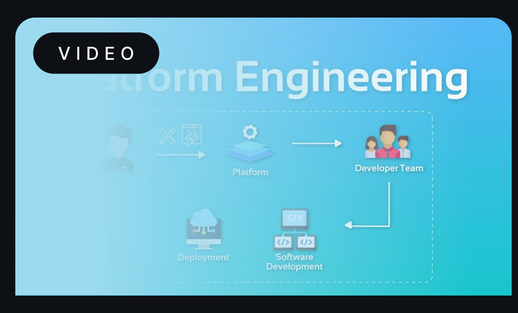
Resources

What Is a Platform Team and What Problems Do They Solve?



Laying the foundation for a career in platform engineering

Platform Engineering Tools: 12 Types of Tools to Use in Your Platform



What is Platform Engineering? | KodeKloud

Platform Engineering 101: Get Started with Platforms

Platform Engineering 101: Get Started with Platforms

Platform Engineer Learning Path

Join the #1 platform engineering virtual conference

Embark on your journey to master Platform Engineering

Platform Engineer Learning Path

★★★★½ 4.8

https://kode.wiki/490bRp3