Introducing Cloud Native Architecture & Maturity Model

A Modern Approach to Building Resilient, Scalable Applications

Presented by: Dr. Vipul Chudasama

Objectives

- Understand what Cloud Native means
- Explore core components and benefits
- Learn the stages of Cloud Native Maturity
- Discuss real-world applications

Why Cloud Native?

- Shift from monoliths to microservices
- Need for scale, agility, and resilience
- DevOps & CI/CD as enablers

What is Cloud Native?

- CNCF Definition
- Technologies: Containers, Microservices, DevOps, Automation

Evolution of Application Architecture

- Monolith → 3-Tier → Microservices → Serverless

Principles of Cloud Native Architecture

- Stateless applications
- Containerization
- Horizontal scalability
- Immutable infrastructure
- Declarative APIs

Pillars of Cloud Native

- DevOps & CI/CD
- Microservices
- Containers (Docker)
- Orchestration (Kubernetes)
- Observability
- Automation & IaC

Benefits of Cloud Native

- Faster deployment
- Resilient systems
- Efficient resource usage
- Easier to scale and maintain

Challenges with Cloud Native

- Increased complexity
- Need for cultural/skill shift
- Managing distributed systems
- Security at every layer

Introduction to Maturity Models

- Framework to assess and guide adoption
- Why maturity matters:

  Alignment,

  Planning,

  Measurement

4 Stages of Cloud Native Maturity

1. Cloud Aware - Basic cloud use
2. Cloud Enabled - Modular services, basic automation
3. Cloud Native - Full CI/CD, containers
4. Cloud Optimized - AIOps, full scalability

Stage 1 – Cloud Aware

- Lift-and-shift legacy apps
- Minimal changes
- Example: VM-based web hosting

Stage 2 – Cloud Enabled

- Partial decoupling
- Start of DevOps, basic monitoring
- Example: Monolith in container

Stage 3 – Cloud Native

- Microservices, Kubernetes, CI/CD
- Infrastructure as Code (IaC)

Stage 4 – Cloud Optimized

- Serverless, event-driven
- AI/ML for auto-scaling and healing

Example

A university is developing a **student portal system** to manage:

- Student profiles
- Examination results
- Notifications
- Document uploads

Initially, the system was built as a monolithic web application hosted on a single virtual machine. As usage grows and performance requirements increase, the system evolves across the **Cloud Native Maturity Model**.

Activities

Identify Levels

-description

- Tech stack

- challenges/Benefits

Level 1

Lift & Shift — Monolith on EC2
Description:

- Entire student portal is one big monolithic application.
- Hosted on a single **AWS EC2 instance**.
- All components (login, profile, results, notifications) are in one codebase.

Tech Stack:

- EC2 (Ubuntu VM)
- MySQL installed on same instance
- File uploads stored locally

Challenges:

- No auto-scaling
- Downtime if EC2 crashes
- Difficult to maintain and update features independently

Level 2

Cloud-Enabled Architecture

Description:

- Begins using managed cloud services for database and storage.
- Adds **CI/CD** for automated deployments.

Improvements:

- Moves database to **Amazon RDS** (managed MySQL)
- Stores uploaded documents in **Amazon S3**
- Implements **GitHub Actions** for CI/CD to deploy changes to EC2

Benefits:

- Higher availability and durability of data
- Deployment automation
- Still a monolith, but more stable and reliable

Level3 Cloud Native Architecture

Description:

- Application is **refactored into microservices**.
- Each service runs independently and communicates via REST APIs.

Microservices Introduced:

- **Authentication Service**
- **Exam Result Service**
- **Notification Service**
- **Document Service**

Tech Stack:

- **Dockerized microservices**
- Deployed on **Amazon EKS** (Kubernetes) or **Fargate**
- Service discovery and communication via **internal load balancer**
- Centralized logging and monitoring with **CloudWatch**

Benefits:

- Independent deployment and scaling of services
- Easier to develop and debug
- Foundation for observability and DevSecOps

Level 4 Cloud Optimized / Serverless

Description:
•Critical components are now **event-driven** and **serverless**.
•System responds dynamically to student actions.

•**Enhancements:**
•**AWS Lambda** used for sending emails and SMS notifications
•**S3 Events** trigger Lambdas to process uploaded documents (e.g., virus scan, OCR)
•**Amazon Kinesis** streams real-time student login data for **analytics**
•**DynamoDB** used for fast, scalable access to student activity logs

Benefits:
•No infrastructure management
•Cost-effective for burst workloads (e.g., during result days)
•Real-time insights and automated responses

Cloud Native Maturity Model

Level 1 Lift and Shift	Level 2 Cloud-Enabled	Level 3 Cloud-Native	Level 4 Cloud-Optimized
			
Host entire monolith on AWS EC2	Use RDS for the database, S3 for document storage, and a basic CI/CD setup	Break into microservices — authentication, exam results notifications	Use AWS Lambda for notifications, S3 events, and real-time analytics with Kinesis

Designing a **Student Portal System**

Best Practices for Cloud Native Success

- Design for failure – assume components will crash
- Automate everything – from build to deployment
- Monitor everything – use observability tools like Prometheus, Grafana
- Secure everything – zero trust architecture, DevSecOps
- Collaborate across teams – true DevOps culture

How to Assess Maturity

Tools That Help Assess Maturity

- **Open Source Frameworks**:
  - CNCF Cloud Native Maturity Model
  - AWS Well-Architected Tool
  - Microsoft's CAF (Cloud Adoption Framework)
  - Google Cloud’s DevOps Research & Assessment (DORA)
- **Surveys and Questionnaires**:
  - Use Likert-scale assessments (1–5) per domain
  - Score overall and per-dimension maturity

Roadmap to Cloud Native

- Cultural + Tech shift
- Tools: GitHub Actions, Docker, Kubernetes, Terraform

Case Study – Netflix

- Evolution to microservices
- Use of chaos engineering
- Scalability success

Case Study – Spotify

- Microservice + DevOps adoption
- CI/CD pipelines for fast deployment

Summary

- Cloud Native = mindset + architecture
- Maturity model helps structured transformation
- Agile, secure, scalable apps

Quiz / Rapid Recap

- What enables Cloud Native systems?
- Define 'Immutable Infrastructure'
- Give an example of a Cloud Optimized platform

Q&A

Ask your questions!

Further Reading

- https://cncf.io
- https://12factor.net
- Kubernetes Docs, DevOps Handbooks

Thank You

Presented by: Dr. Vipul Chudasama

Contact: [Add Email/QR Code if needed]