IEEE-CNSV

Consultants' Network of Silicon Valley

Cloud Software: The Benefits of Going Native

January 14, 2020

Ron Herardian

IEEE CNSV

Speaker Bio

Ron Herardian has worked for the past 20+ years in the software industry in Silicon Valley, California, in the areas of enterprise software, mobile and web applications, and Software as a Service (SaaS). He has served in multiple engineering, management, and consulting roles at leading software companies, and has been a founder, investor, board member, and advisor for several software startups.

Mr. Herardian previously worked for IBM Lotus, Cisco, and Oracle. In a previous role, Mr. Herardian co-founded a cybersecurity software startup that developed a security policy enforcement system for cloud operations. Prior to that, he was responsible for cloud infrastructure, operations and security for a financial technology startup serving the banking industry, including DevOps and CI/CD, as well as regulatory compliance, including SOC 2 and GDPR.

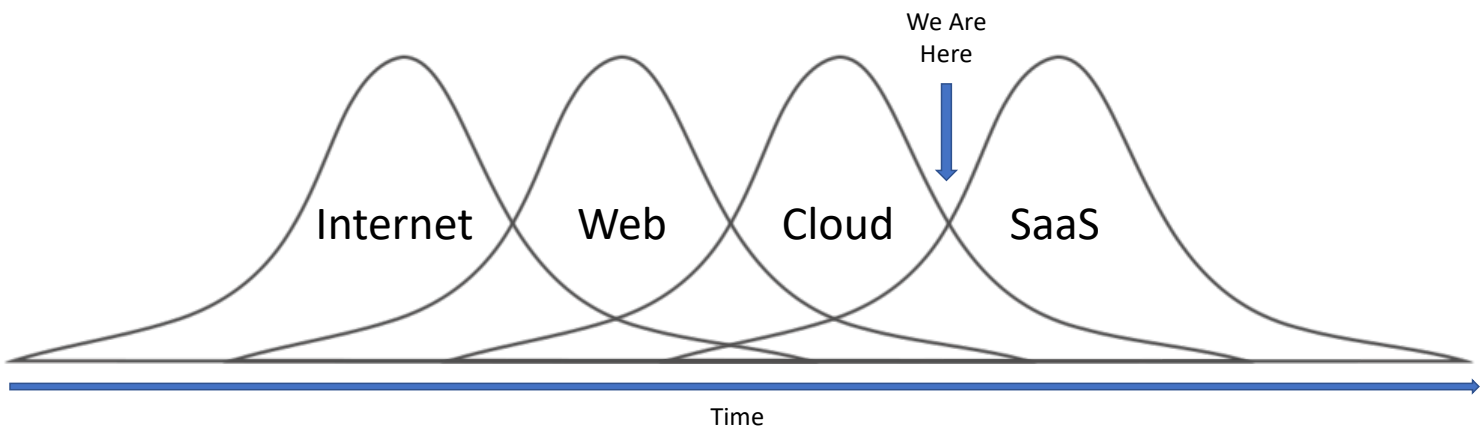
Mr. Herardian, who holds a Certified Information Systems Security Professional (CISSP) certification, is a graduate of Stanford University, a Senior Member of the Institute for Electrical and Electronic Engineers (IEEE) and a Senior Member of the Association of Computing Machinery (ACM). He is a speaker at conferences and events, as well as a published author who has coauthored a book and written many whitepapers and articles.

Agenda

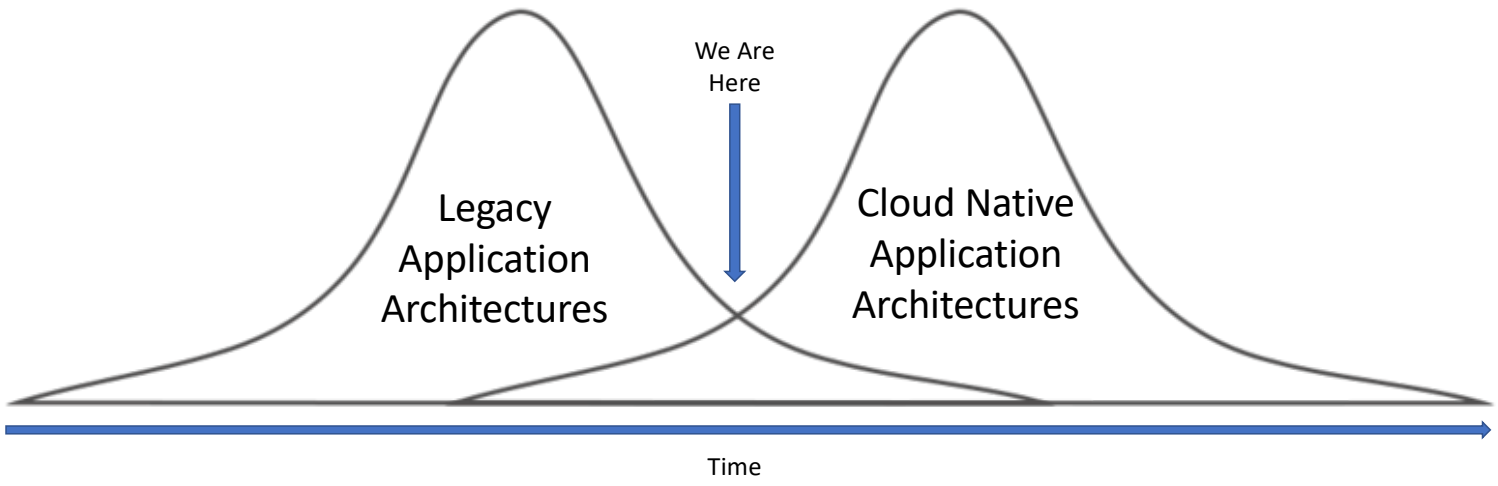
1. Drivers of Change
2. Cloud Native Concepts
3. Legacy Versus Cloud Native
4. Implementing Cloud Native
5. Cloud Native Landscape
6. Benefits of Cloud Native
7. Example Application Walk Through

1. Drivers of Change

Technology Adoption Curves



Software Transformation



2. Cloud Native Concepts

Cloud Native Concepts (1 of 2)

Infrastructure

- Everything is code
- Containerization
- Immutable artifacts

Architecture

- Microservices
- Horizontal scaling
- Stateless protocols

Cloud Native Concepts (2 of 2)

Design Patterns

- OAuth, OpenID Connect (OIDC), JSON Web Tokens (JWT)
- Representational state transfer (REST APIs)
- Event Stream Processing, e.g., Apache Kafka
- Command Query Responsibility Segregation (CQRS)

3. Legacy Versus Cloud Native

Legacy Architectures (1 of 2)

People

- Separate development, QA, and operations teams

Processes

- Waterfall software development lifecycle (SDLC)
- Traditional Release Engineering (customer facing packages)
- Long release cycles, e.g., months
- Production systems integration is a separate post-release process
- Manual orchestration
- Compartmentalized automation

Legacy Architectures (2 of 2)

Technology

- Monolithic code base and executables
- Vertical scaling on highly available, reliable hardware
- Large increments of scale / all services scale together
- Stateful (session oriented) protocols
- Strong consistency models for data (typically SQL)
- Use of internal non public APIs
- Fixed deployment architectures

Cloud Native Architectures (1 of 2)

People

- Integrated development, QA and operations teams (DevOps)

Processes

- Agile
- Continuous integration, delivery and deployment (CI/CD)
- Fast release cycles, e.g., minutes
- Production systems integration is a continuous process
- Automated orchestration
- End-to-end automation

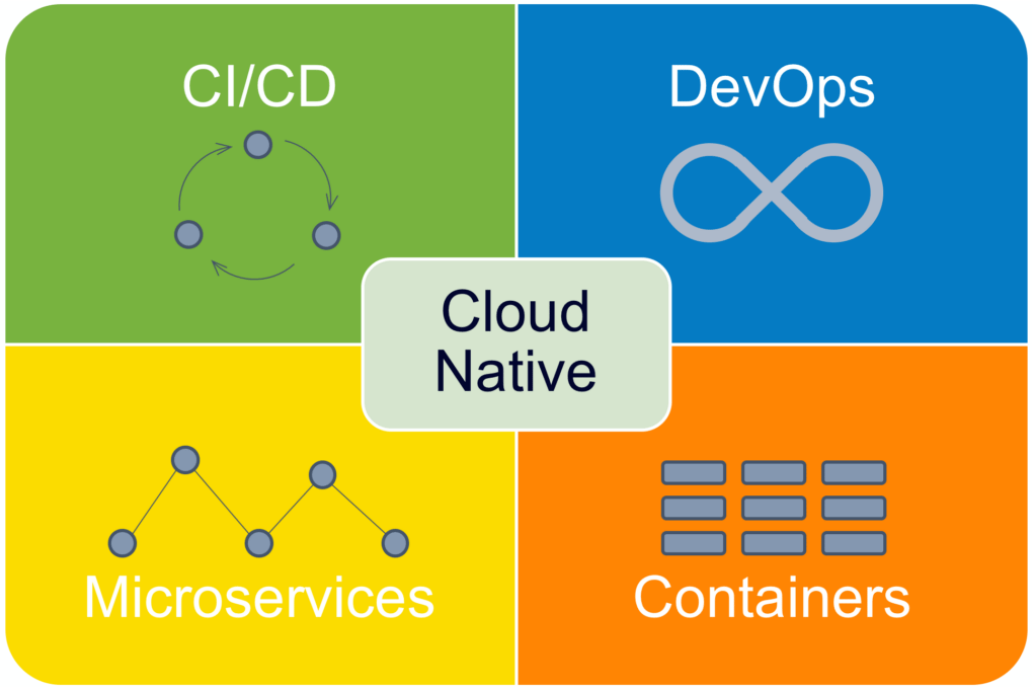
Cloud Native Architectures (2 of 2)

Technology

- Individual microservices
- Horizontal scaling on commodity hardware (everything fails)
- Small increments of scale / each service scales independently
- Stateless protocols, e.g., RESTful APIs
- Eventual consistency and NoSQL for data
- API-centric approach and public APIs
- Flexible, evolutionary deployment architectures

4. Implementing Cloud Native

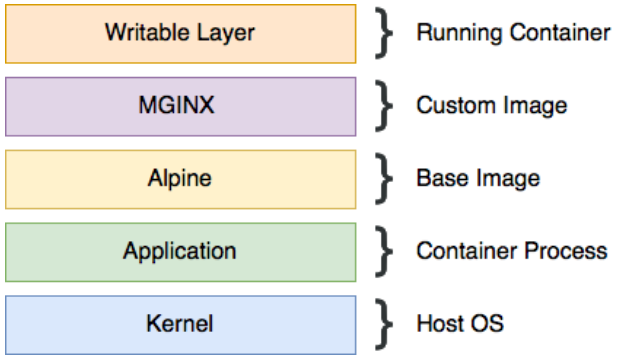
Building Blocks

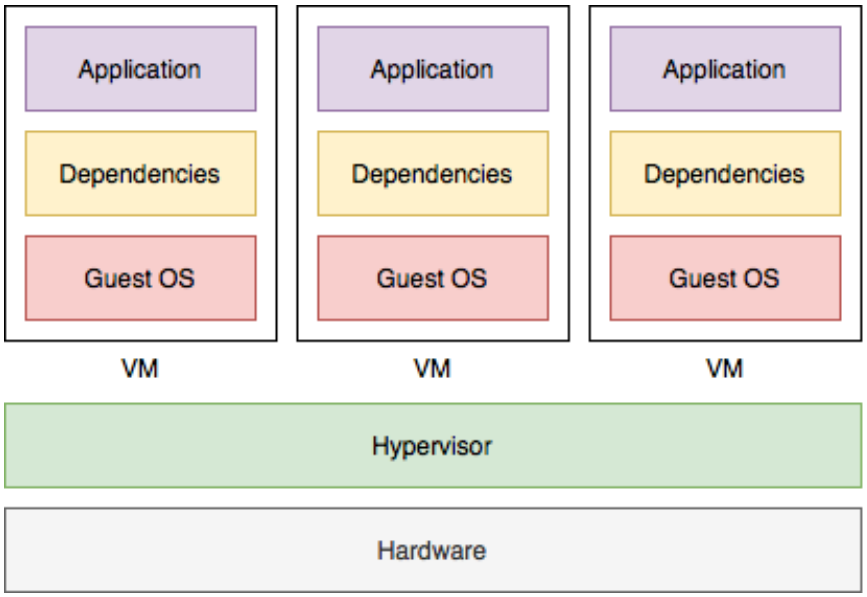


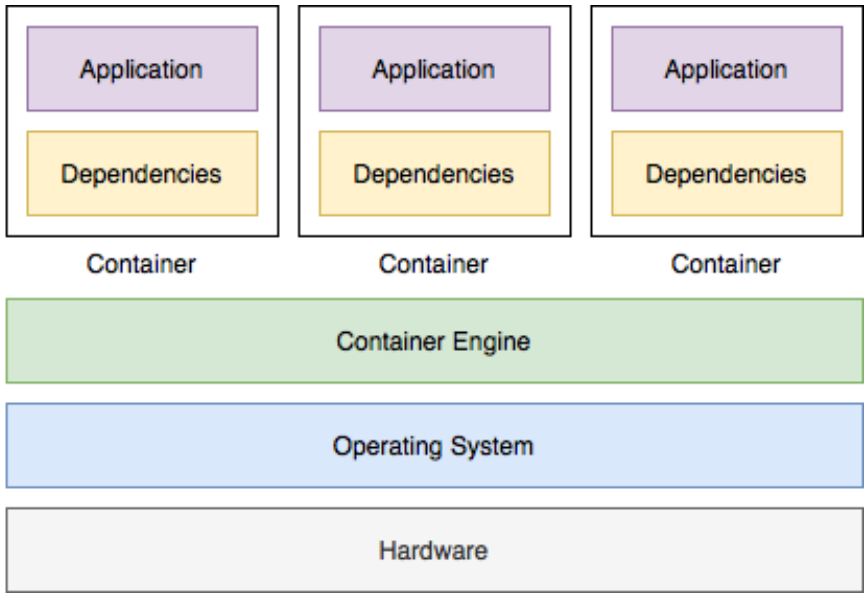
https://www.instana.com/blog/cloud-native-seeing-through-hype/

- DevOps
- CI/CD
- Containers
- Microservices

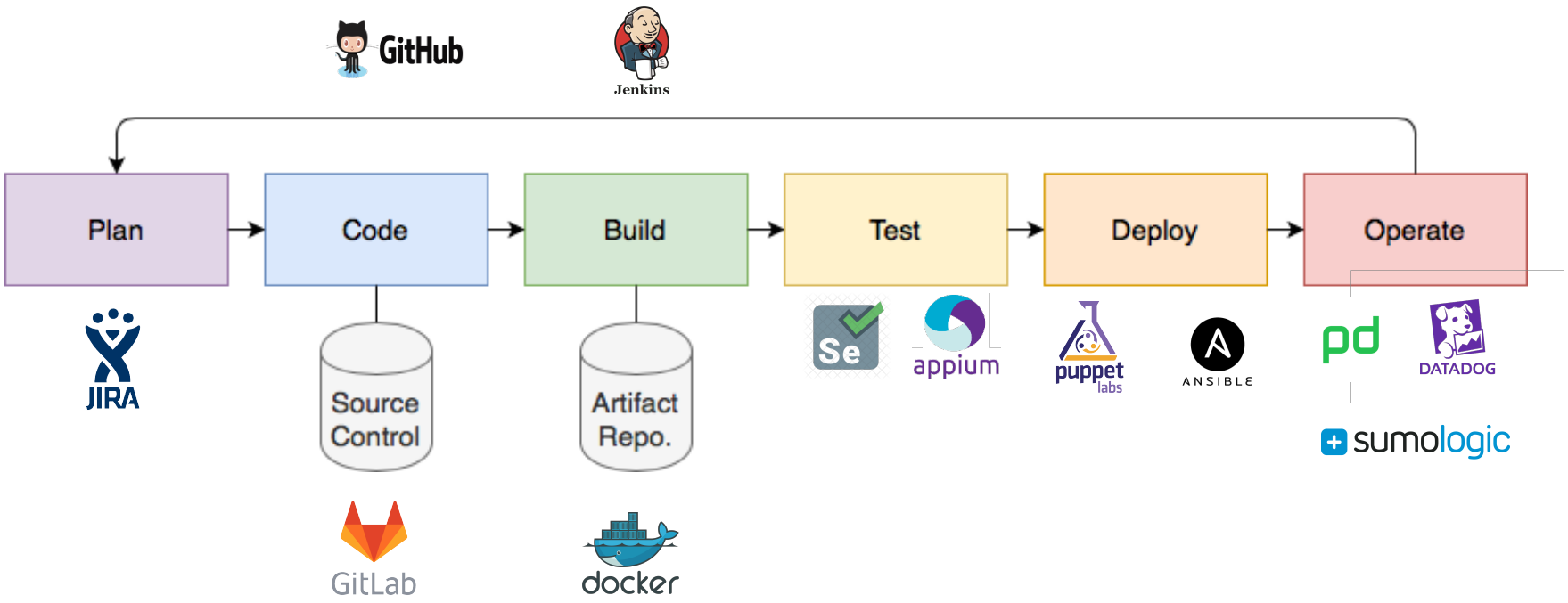
Containers


Writable Layer
Running Container
MGINX
Custom Image
Alpine
Base Image
Application
Container Process
Kernel
Host OS

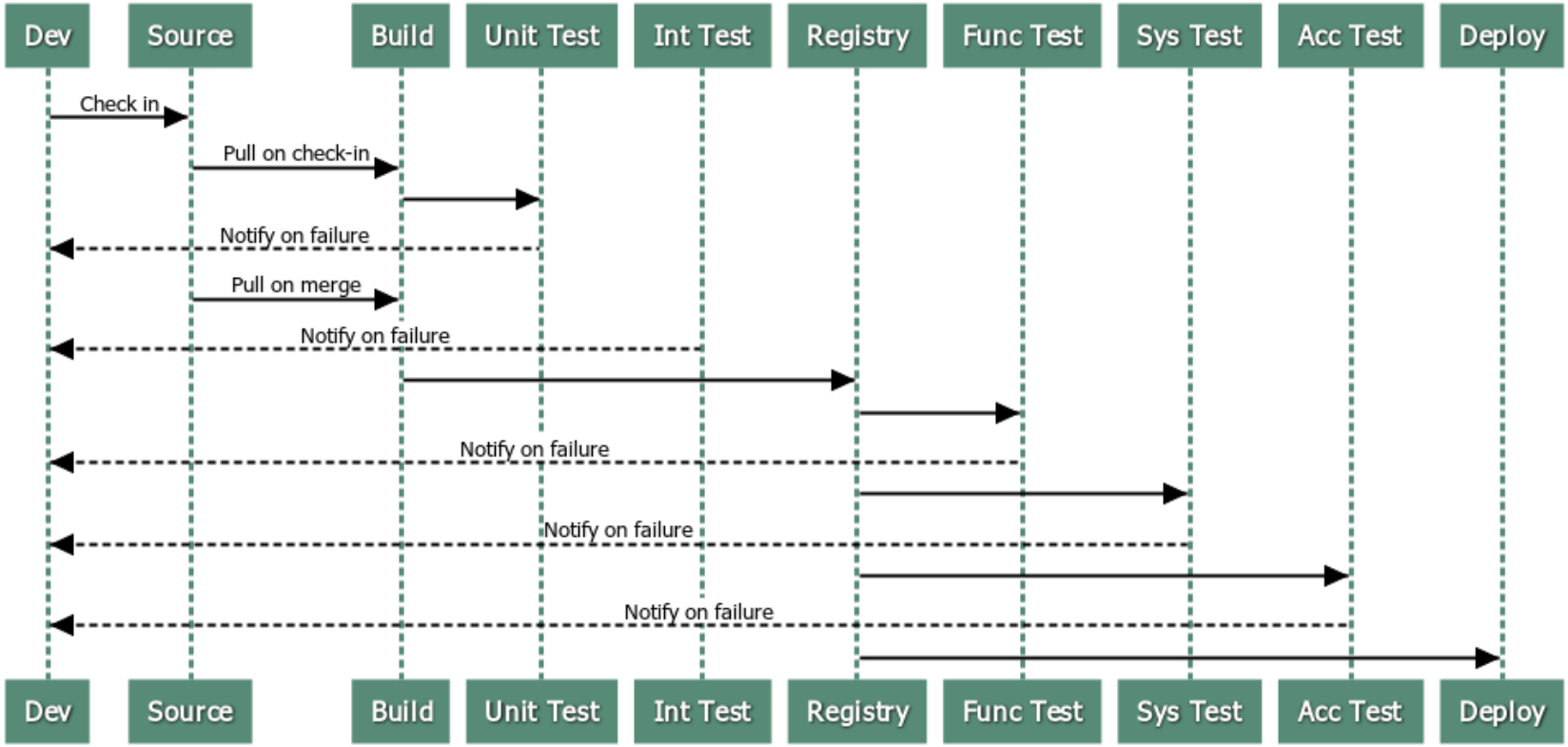

Application
Dependencies
Guest OS
VM
Application
Dependencies
Guest OS
VM
Application
Dependencies
Guest OS
VM
Hypervisor
Hardware


Application
Dependencies
Container
Application
Dependencies
Container
Application
Dependencies
Container
Container Engine
Operating System
Hardware

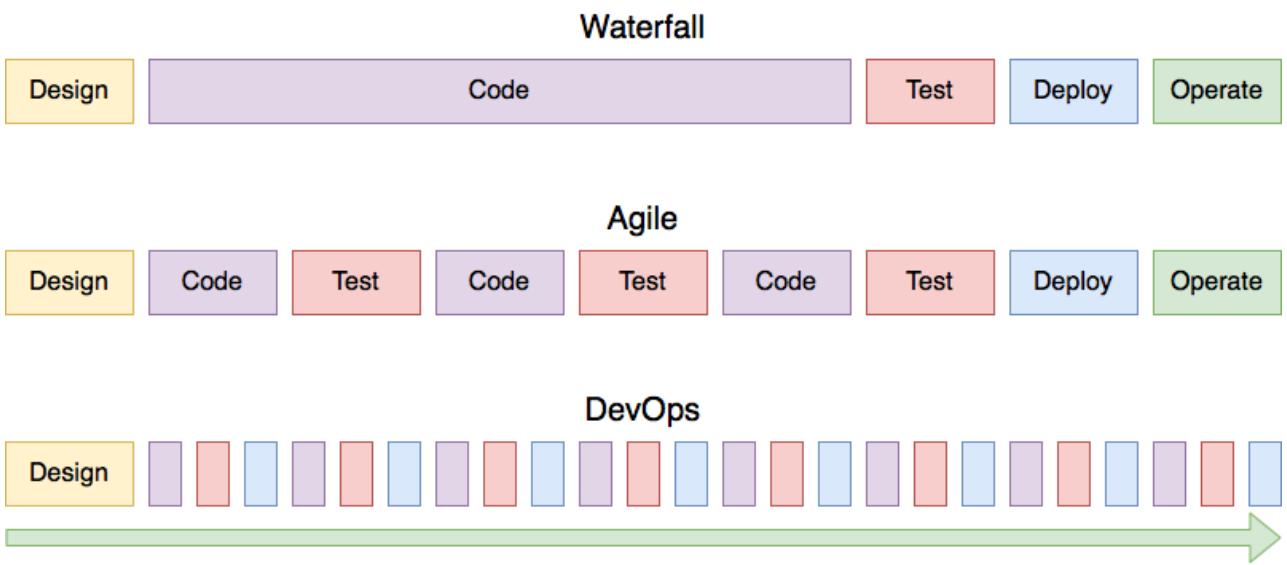
CI/CD


GitHub
Jenkins
Plan
Code
Build
Test
Deploy
Operate
JIRA
Source Control
Artifact Repo.
Se
appium
puppet labs
ANSIBLE
pd
DATADOG
sumologic
GitLab
docker

Automation in the CI/CD Pipeline



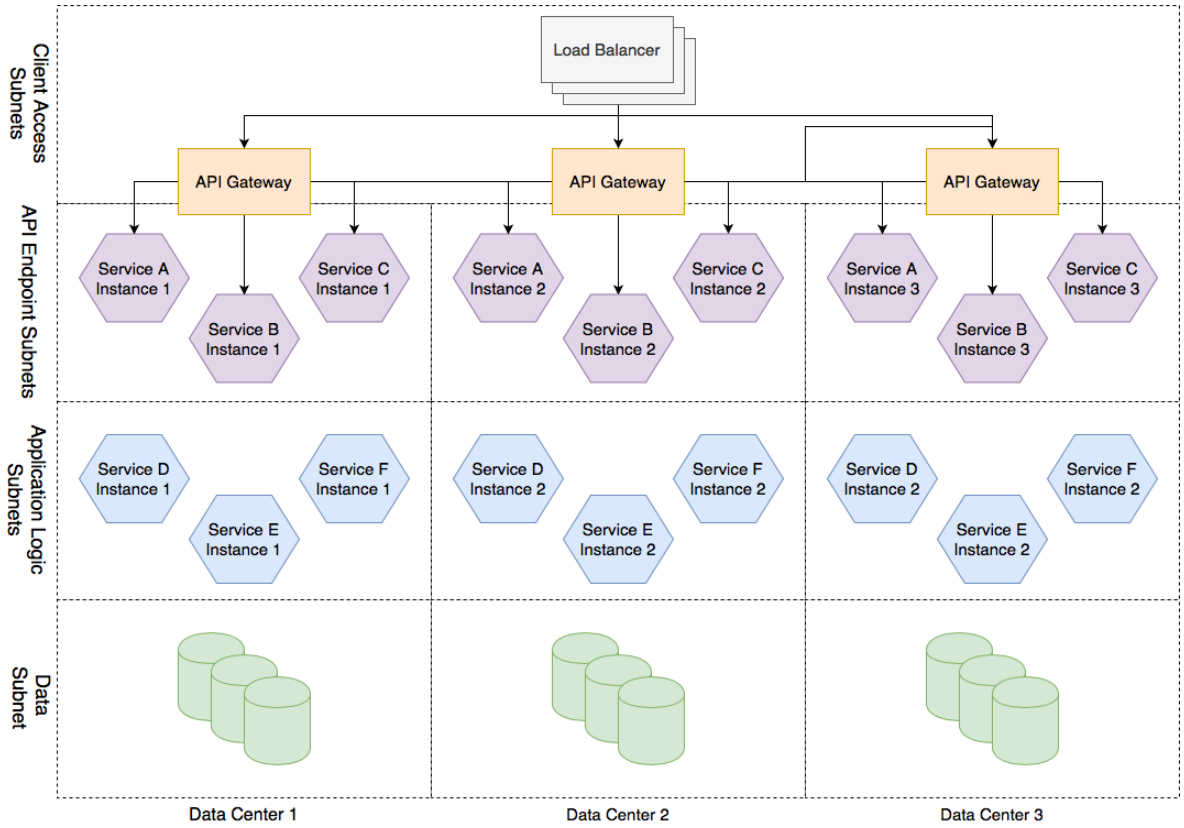
DevOps



DevOps is *"a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality"*

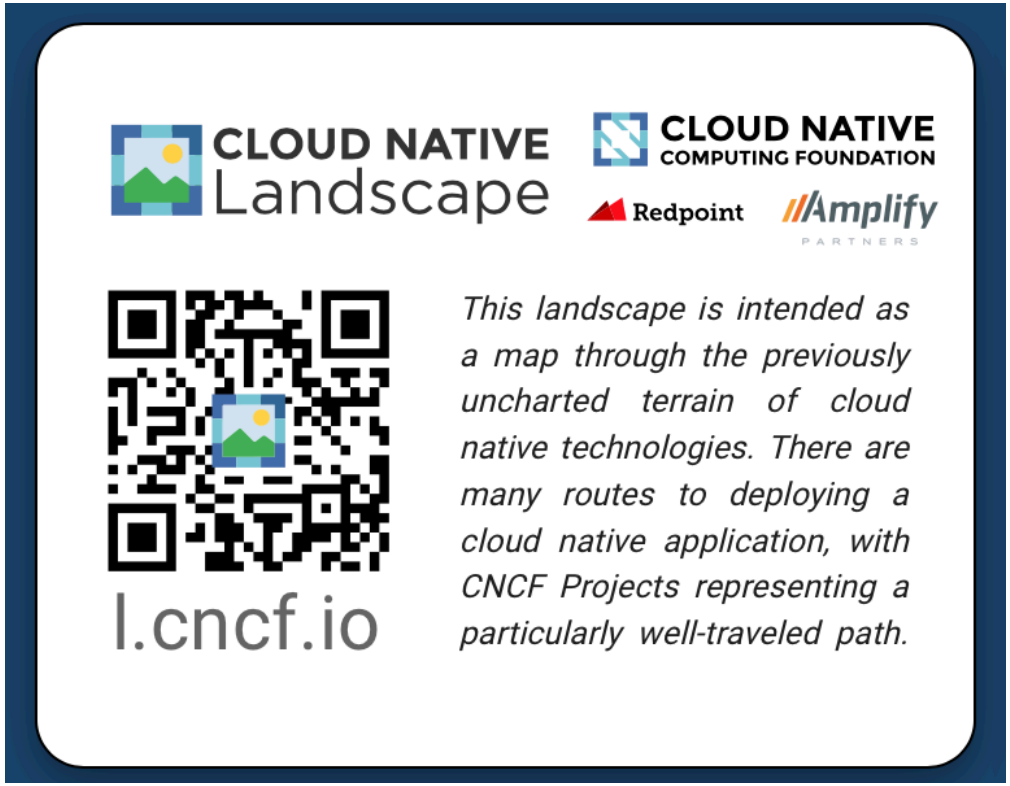
(Bass, Len; Weber, Ingo; Zhu, Liming (2015). DevOps: A Software Architect's Perspective)

Microservices Concepts



5. Cloud Native Landscape

Cloud Native Landscape



https://landscape.cncf.io

Cloud Native Landscape (1 of 3)

- Application Definition and Development
  - Database
  - Streaming and Messaging
  - Application Definition and Image Build
  - Continuous Integration and Delivery

- Observability and Analysis
  - Monitoring
  - Logging
  - Tracing (and Distributed Tracing)
  - Chaos Engineering

Cloud Native Landscape (2 of 3)

- Orchestration and Management
  - Scheduling and Orchestration
  - Coordination and Service Discovery
  - Remote Procedures Call
  - Service Proxy
  - API Gateway
  - Service Mesh

- Kubernetes Services
  - Kubernetes Certified Service Providers
  - Kubernetes Training Partners

Cloud Native Landscape (3 of 3)

Platform

- Certified Kubernetes Distribution
- Certified Kubernetes Hosting
- Certified Kubernetes Installer

Runtime

- Automation and Configuration
- Container Registry
- Security and Compliance
- Key Management

6. Benefits of Cloud Native

Benefits of Cloud Native (1 of 2)

Business

- Cost efficient
- Fast time to market
- Competitive advantage

Technical

- Deployments are elastic
- Architectures can evolve
- Better availability

Benefits of Cloud Native (2 of 2)

Customer

- Better quality
- Better customer experience
- Higher customer value

IEEE-CNSV

Consultants' Network of Silicon Valley

Ron Herardian

https://www.linkedin.com/in/rherardi/

IEEE

CNSV