Availability in a Cloud-Native World.

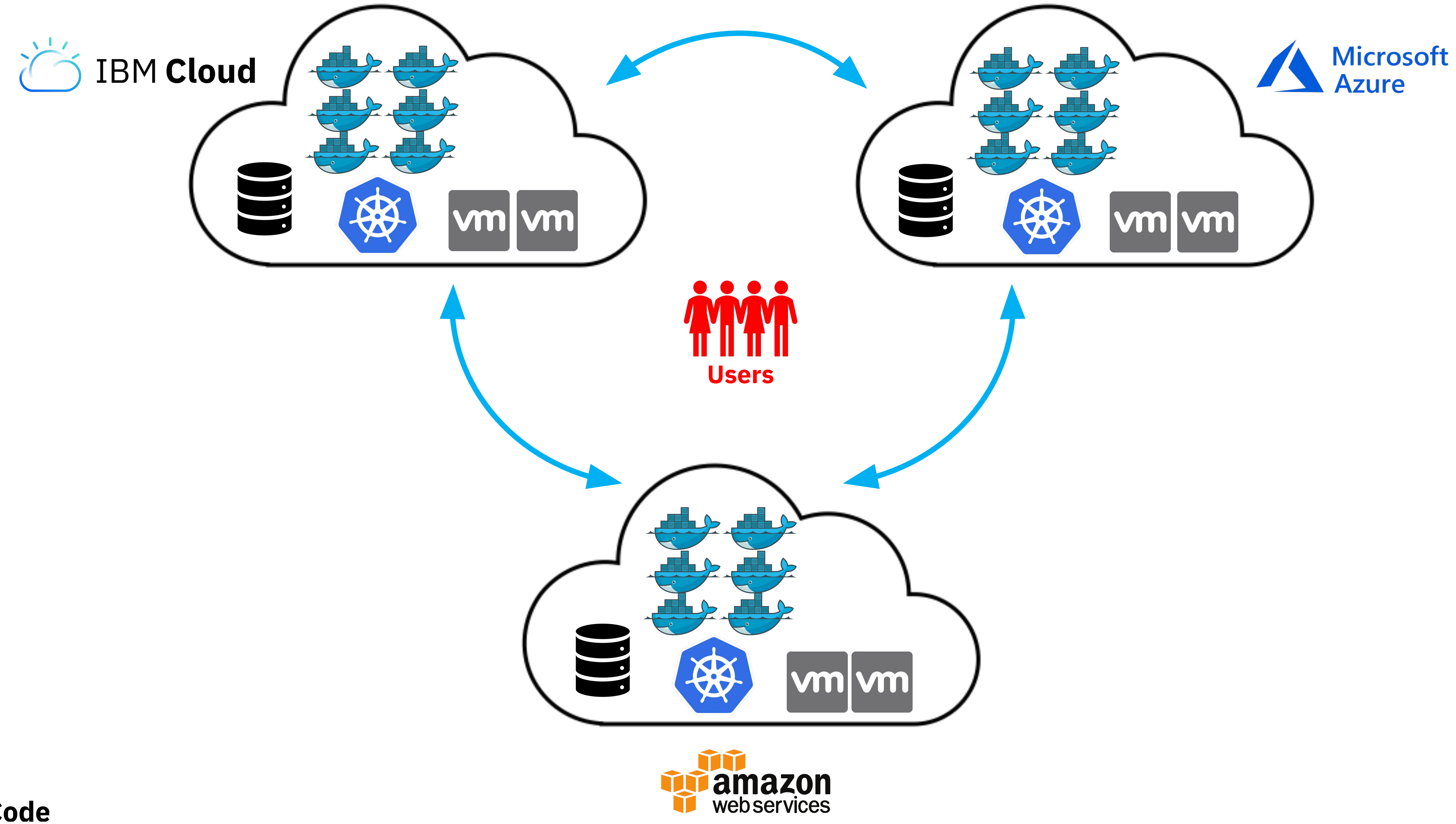
Guidelines for mere mortals.

Haytham Elkhoja
Platform Architect, IBM
haytham.elkhoja@ibm.com
@haythamelkhoja

v1.0 30 June 2018

IBM Code

What you should aim for.


IBM Cloud
Microsoft Azure
Users
amazon web services

Definition

Cloud-Native

born on the cloud,
scales on the cloud,
consumes the cloud,
resilient on the cloud,
performs on the cloud.

Cloud-Native

Why?

- Polyglot Development.
- Parallel, agile and siloed Development.
- Choose the right tool for the job.
- Microservices and Loosely-Coupled Components.
- Dependencies Authoring and Tracing.
- Scale out vs Scale up.
- The Pet vs the Cattle.

Definition

Availability

The vast majority of software services and systems should aim for *almost-perfect* reliability rather than perfect reliability.

Why?

- First impression, last impression.
- Cost of downtime.
- There are 8,760 hours in a year.
- Business and service continuity.
- Availability, resilience and scalability go hand in hand.

Cloud Native + Availability.

Cloud Native + Availability

Why?

- Resiliency and highest SLA/Continuous Availability.
- Scalability and Performance.
- Redirect users to their closest region/cloud.
- Green/Blue deployments per region/cloud.
- Right cloud for the right job.
- 3 regions/clouds always cheaper than 2.

Some Guidelines we picked up in the field.

Guideline

Architect your Application to be cloud and infrastructure **agnostic**.

Guideline

Understand Service **Levels**. Calculate **Availability**. Formalize Error **Budgets**.

- 99% availability signals over 7 hours of downtime a month
- 99.9% availability signals over 43 minutes of downtime a month
- 99.99% availability signals under 5 minutes of downtime a month
- So on and so forth…

Guideline

Welcome and embrace **Asynchronous** events and data replication. Timestamp every breath you make.

Guideline

Share-nothing.
Cluster-nothing.

Guideline

Design for Failure.

KISS (Keep It Simple Stupid).

Fail small.

Guideline

Religiously steer clear from IP addresses. **DNS** and **Service Discovery** are your best friends.

Guideline

12 Factors applications development and design methods help you achieve application and cloud mobility.

Guideline

Aim for **Stateless**, but maintain session states, if you must.

Guideline

Delegate responsibilities. **Whatever as a Service**. Somebody, somewhere has done it better.

Guideline

GitOps. Everything should be **Versioned** and **Reproducible**, this includes configuration files and **Infrastructure as Code**.

Guideline

Automation is a way of life.

Guideline

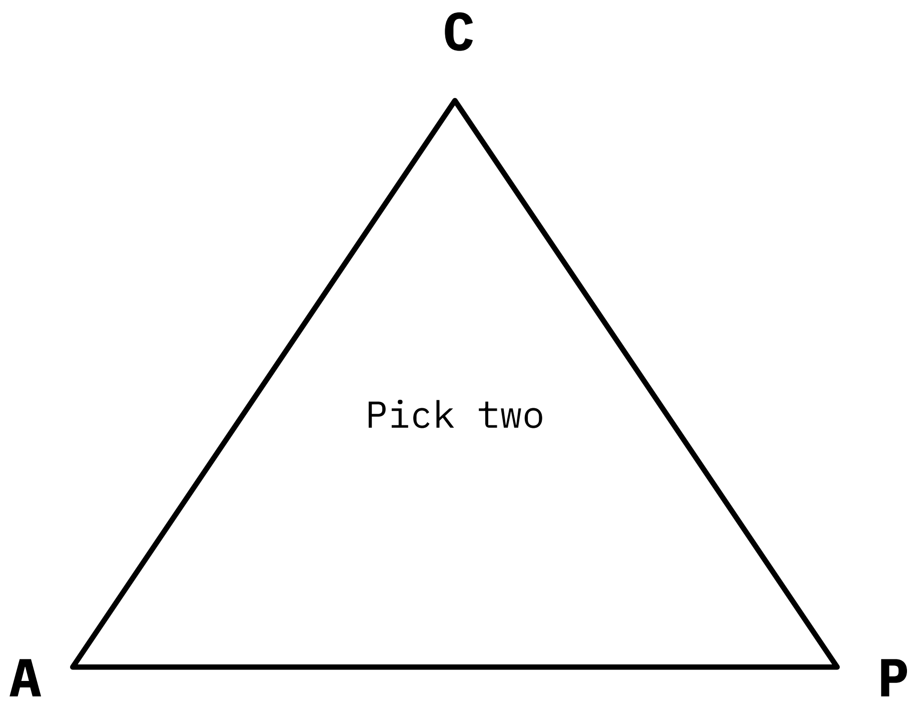
Write anywhere and everywhere. **Peer to Peer** data and session replication.

Guideline

CAP Theorem

decisions early on.

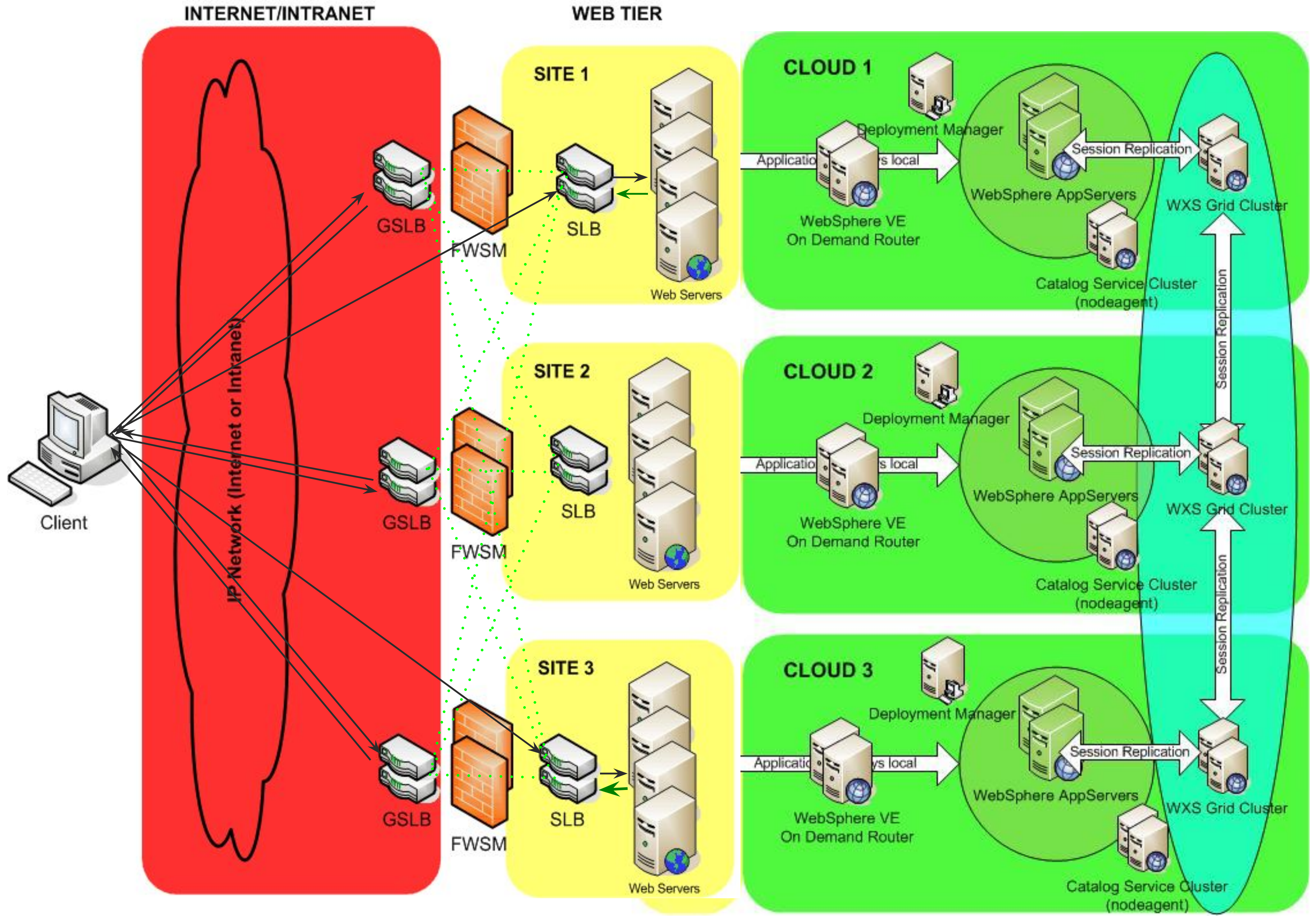
Consistency. **A**vailability. **P**artition-tolerance.



Guideline

Design for **Feedback**. Measure every single detail via KPIs. Capture **Metrics** and **Logs**. There’s no such thing as too much logs.

This should be the end result:


INTERNET/INTRANET
WEB TIER
IP Network (Internet or Intranet)
Client
GSLB
GSLB
GSLB
FWSM
FWSM
FWSM
SITE 1
SITE 2
SITE 3
SLB
SLB
SLB
Web Servers
Web Servers
Web Servers
CLOUD 1
CLOUD 2
CLOUD 3
Deployment Manager
Applicatio s local
WebSphere VE On Demand Router
WebSphere AppServers
Session Replication
WXS Grid Cluster
Catalog Service Cluster (nodeagent)
Session Replication
Session Replication

IBM **Code**

Thanks.