

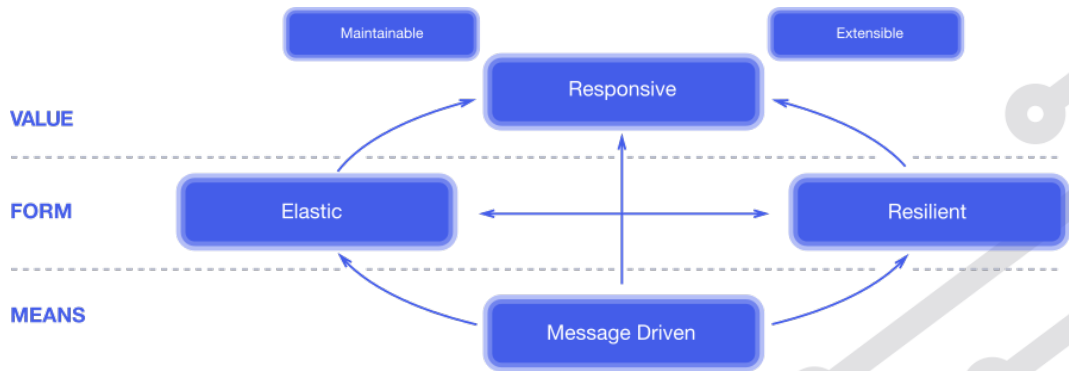
Desarrollo moderno con DevOps y Cloud Native

Víctor Orozco

7 de enero de 2021

Nabenik

Aplicaciones reactivas

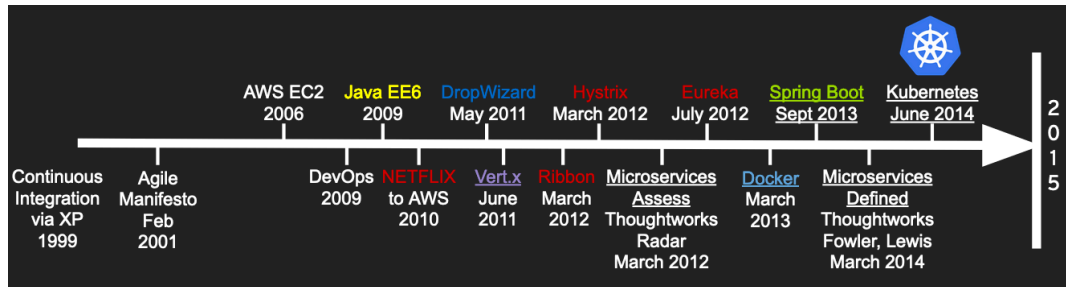


- Sistemas reactivos
- 12 factores cloud native
- Design patterns
- Domain Driven Design
- Microservices chassis e/ou service mesh
- Orquestación de contenedores



- (Nos gustaria tener) **Sistemas reactivos**
- (Es posible con la metodología de) 12 factores Cloud Native
- (Usamos soluciones probadas mediante) design patterns
- (Fragmentamos el sistema mediante) Domain Driven Design
- (Implementamos los servicios con) Microservices chassis y/o service mesh
- (Hacemos deployment) Mediante orquestación de containers

Historia



Créditos: Rafael Benevides

12 factores Cloud Native



12 factores cloud native (Heroku)

Frameworks

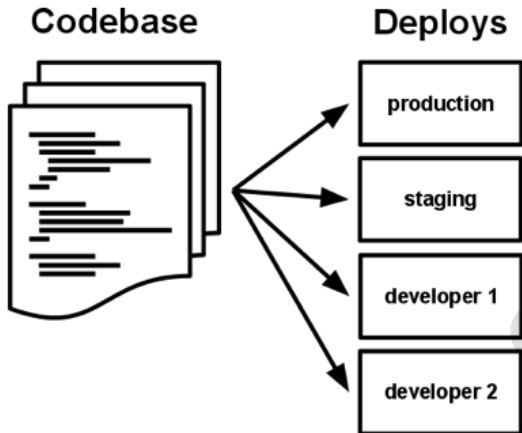
- Config
- Backing service
- Disposability

Cloud

- Codebase (Git-Flow)
- Dependencies (Maven)
- Build, Release, Run
- Processes
- Port binding
- Concurrency (Docker - k8s)
- Dev / Prod parity
- Logs
- Admin process

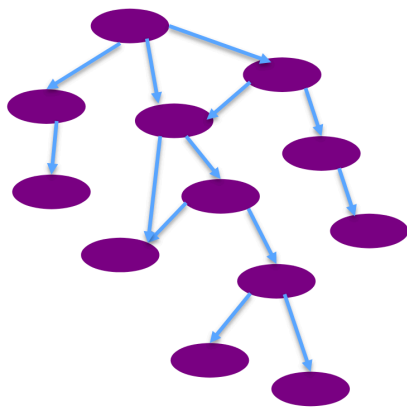
Codebase

- Una base de código con múltiples entornos de despliegue
- Un repositorio por aplicación / microservicio



Dependencias

- Una aplicación cloud native no "depende" de algo en su entorno
- Dependencias aisladas y compilaciones repetibles

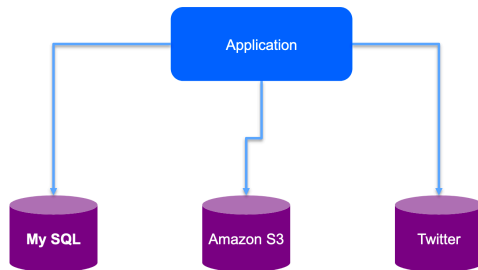


- La configuración de una aplicación debe ser dinámica sin re-compilación/re-empaquete
- Configuraciones inyectables



Backing services

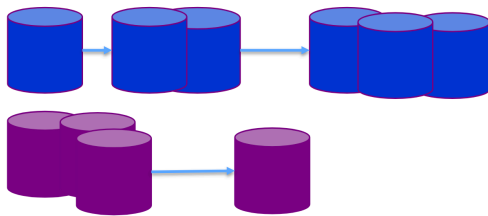
- Acoplamiento debil. Siempre tratar backing services como componentes intercambiables y/o adjuntos



- Separación de etapas de construcción, ejecución y lanzamiento
- CI/CD se hace obligatorio

Procesos

- Ejecutar la aplicación como uno o más procesos sin estado
- REST, Stateless, sesiones portables con JWT

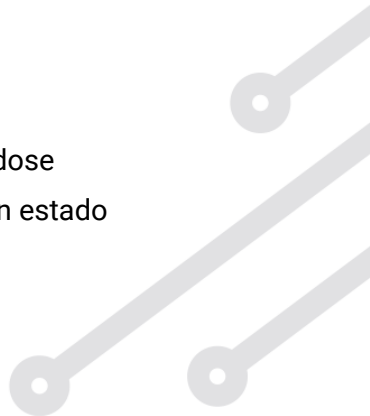


Port binding

- Exponer los servicios con puertos dinámicos
- Kubernetes, Docker, etc.



- Aplicaciones escalan de forma independiente replicándose
- Las nubes escalan mediante copias independientes, sin estado

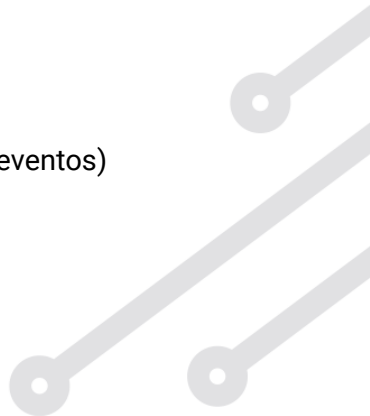


- Procesos arrancar rápido, mueren rápido, reinician rápido
- Procesos son tolerantes a fallas

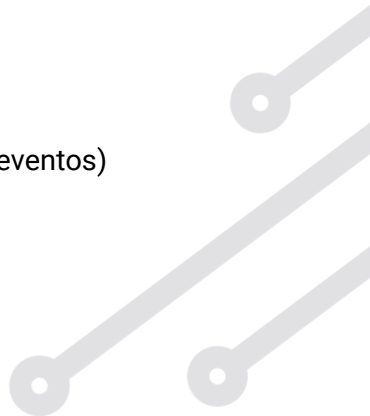


- Entornos de desarrollo, **certificación**, producción lo más homogéneos posible

- Manipular logs de n copias de n servicios (streams de eventos)
- Permitir el análisis posterior



- Manipular logs de n copias de n servicios (streams de eventos)
- Permitir el análisis posterior



Estrategia



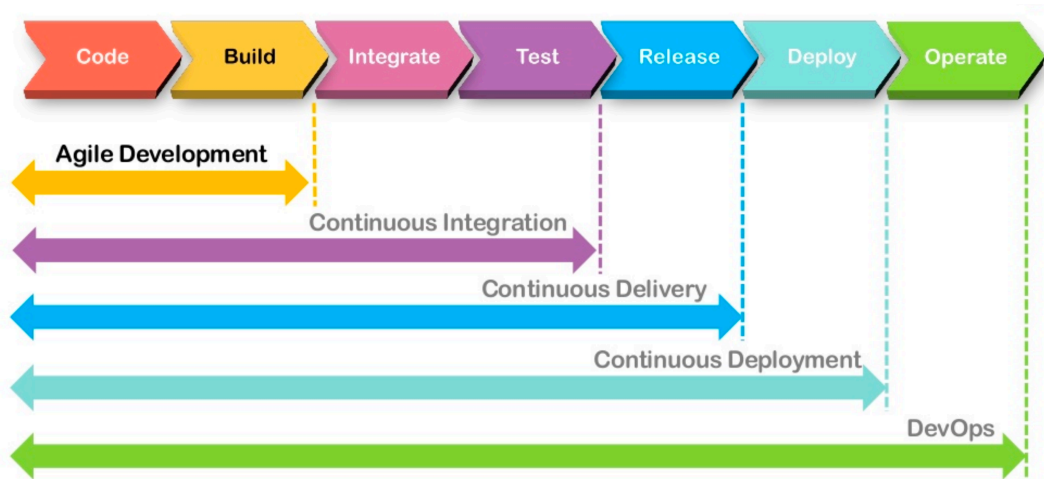
- Implementar DevOps sobre la base de código actual
- Evaluar opciones Cloud Native completas
- Fragmentar en Microservicios solo si es necesario



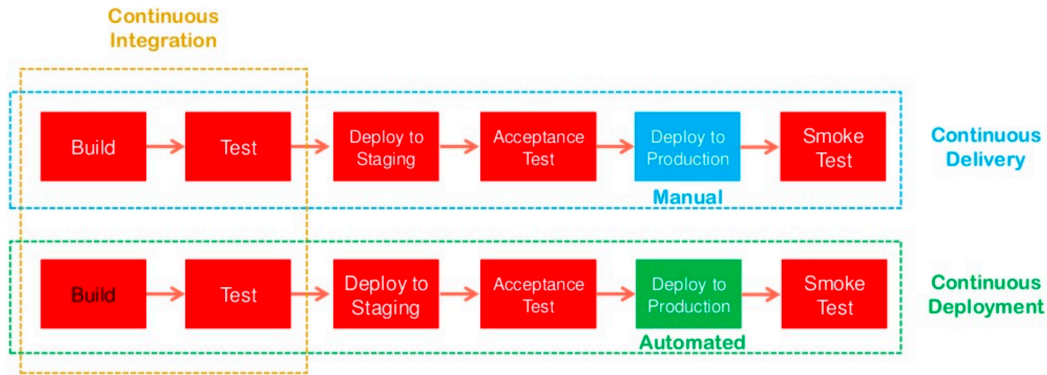
DevOps



Etapas

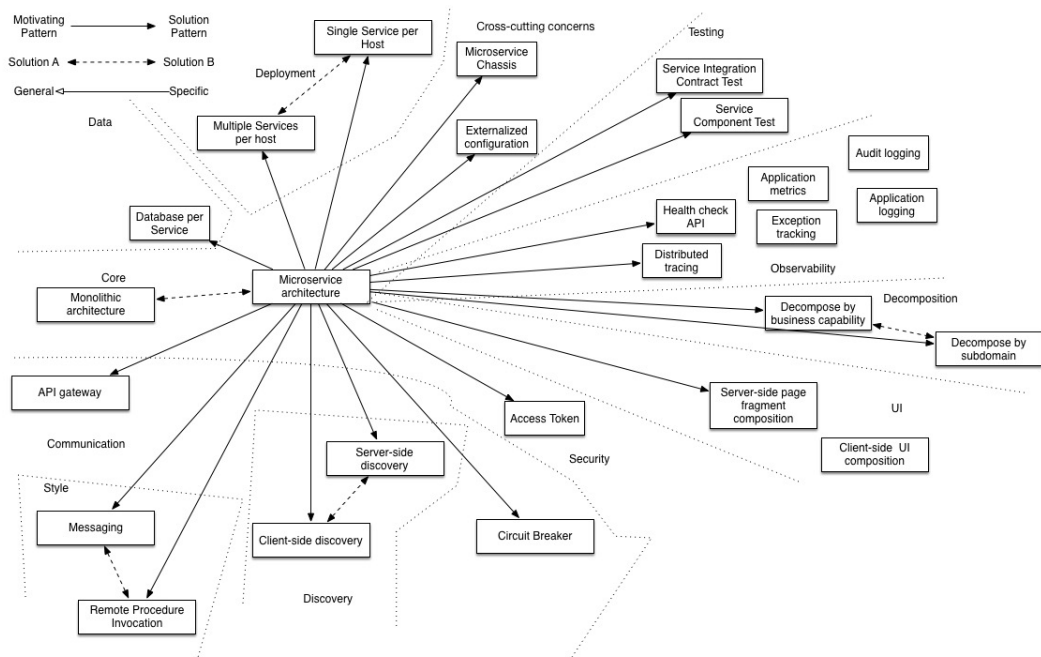


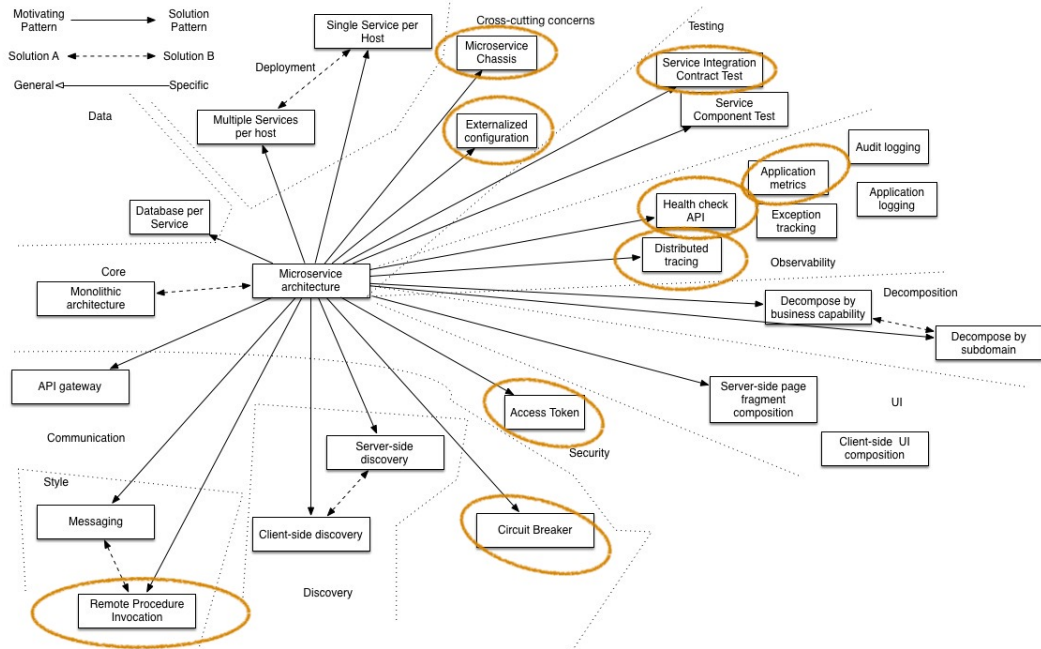
Delivery vs Deployment



Patrones nuevos







Microservice Chassis



Microservice Chassis

Chassis

Los frameworks Cloud Native son soluciones para problemas “cross-cutting concerns”.

Chassis Java

Spring Boot, MicroProfile, DropWizard, Akka

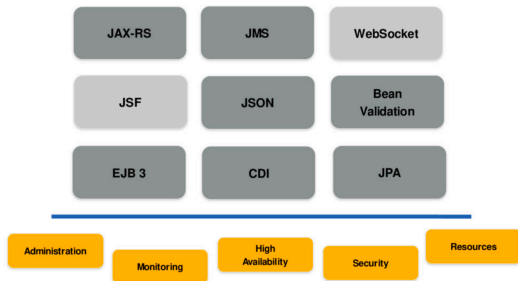
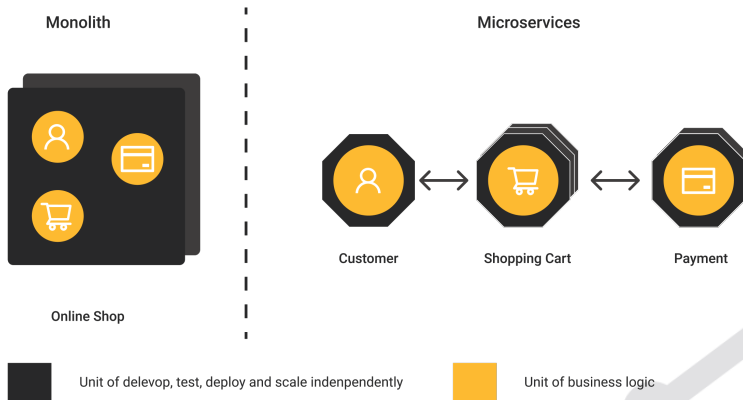
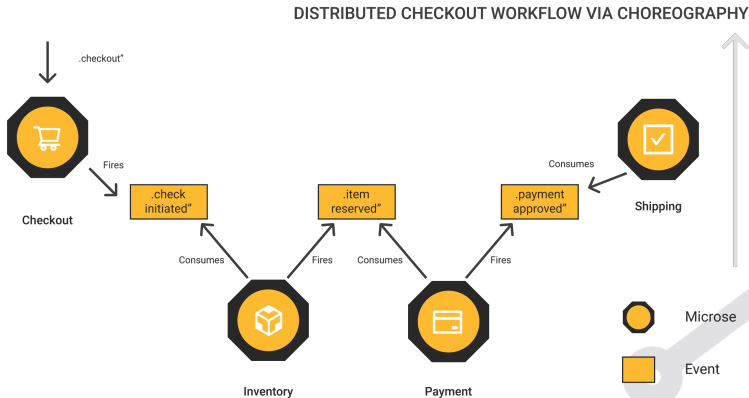


Figura 1: Créditos: Reza Rahman

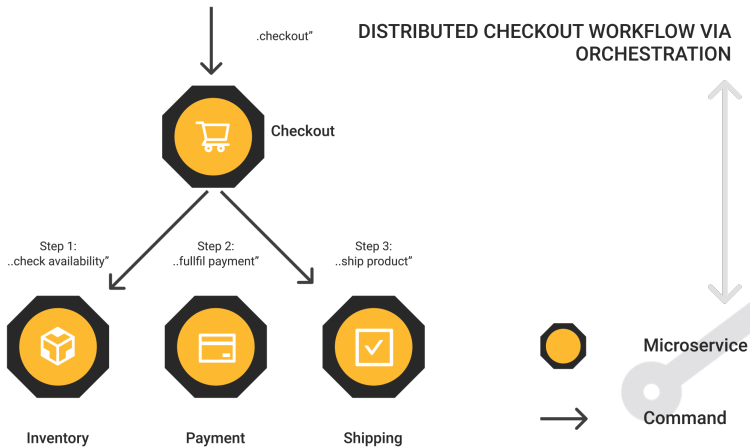
Microservice Chassis



Patterns complementarios - Event Sourcing, CQRS



Patterns complementarios - SAGA



- **Health checks & Metrics** - Recolectar métricas (Prometheus/Grafana) y establecer reglas de despliegue
- **Resilience & Fault Tolerance** - Service mesh -e.g. Likerd, Istio- y Chassis
- **Configuration** - Inyección de configuración en el ambiente
- **Authentication & Authorization** - API Gateway + Chassis
- **Standardized documentation** - OpenAPI + Swagger Server
- **Tracing** - Chassis + Zipkin
- **Remote Procedure & Messaging** - Chassis + K8S service discovery



**Oracle
Groundbreakers**



ORACLE®
Certified Professional
Java SE 8 Programmer

ORACLE®
Certified Associate
Java SE 8 Programmer

- vorozco@nabenik.com
- @tuxtor
- <http://voroazco.com>
- <http://tuxtor.shekalug.org>



This work is licensed under
Creative Commons Attribution-
NonCommercial-ShareAlike 3.0
Guatemala (CC BY-NC-SA 3.0 GT).