

REMOTE CLAZZ

by Zenika Nantes

21 AVRIL 2020
18h00-19h00

CLOUD PATTERNS

Durant cette RemoteClazz, nous expliquerons les différents patterns pour réaliser une application moderne 'Cloud Ready' en 2020.

Nous exposons les différences entre IaaS, CaaS, PaaS, FaaS. Toute une partie sur les concepts de 12-factor ainsi que d'autres patterns seront présentés pour amener le développeur à utiliser des services managés et se concentrer uniquement sur la valeur ajoutée : le code qu'il produit.

Retrouvez-nous sur



zenika

<animés par la passion>

Julien Landuré

CTO Zenika Nantes
GDE Cloud

meetup

Cloud Patterns

Writing Applications that Scale





Cloud Insider

Julien Landuré



CTO Zenika Nantes

Organisateur
GDG Nantes, GDG Cloud
Nantes, DevFest Nantes

GDE Cloud



@jlandure



@jlandure

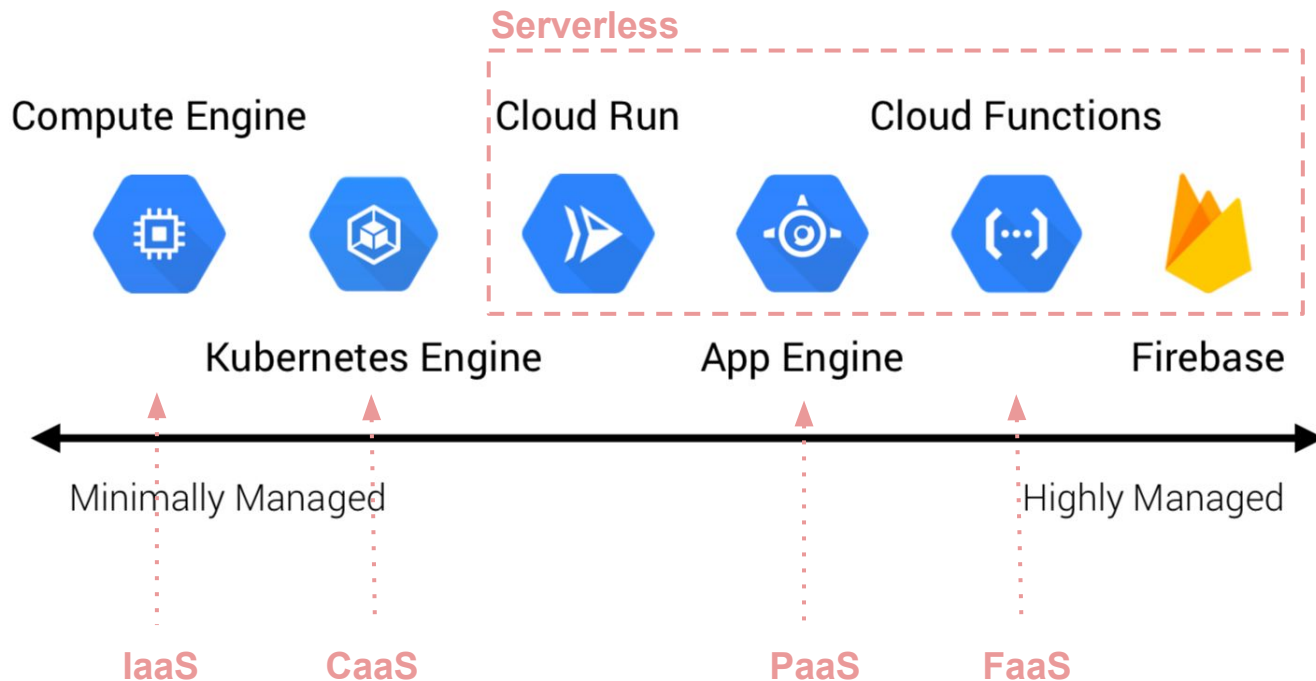
Cloud?

Not If but When



Levels of abstraction

Introduction



The Twelve-Factor App

Getting it right 'By Design'



12-factor app

Introduction

- Checklist for « Cloud Ready » applications
- Stateless App Design
- Distributed systems are hard to build, test, manage, and scale.
- Immutable & ephemeral infrastructure
- Focus on scaling, continuous delivery and portability



Codebase

“ One codebase tracked in revision control,
many deploys

- Everything tracked in version control (git)
- Only one codebase for one app
- Infrastructure as code
- Dockerfile, Procfile, circle.yml, .travis.yml, manifest.yml, .jenkinsfile



Dependencies

“ Explicitly declare and isolate dependencies

- Declared in a tracked file and when possible packaged
- No reliance on specific system tools
- Package.json for Nodejs, pom.xml / build.gradle for Java, requirements.txt for Python, vendor for go lang...



Configuration

“ Store config in the environment

- Configuration separated from code and the app
- Forget `.properties` files for DB credentials, hostnames, passwords...
- `ENV` from Dockerfile, tokens stored as environment variables in CI to avoid to publicly commit them
- Modern architecture: consider Java Spring Cloud Config server or Netflix Archaius



Backing services

“ Treat backing services as attached resources

- Services should be treated as resources: connected and swappable
- DB, Message Queues, SMTP servers may be locally managed or third-party managed
- Very important for testing purpose



Build, Release, Run

“ Strictly separate build and run stages

- Build stage: codebase to build
- Release stage: ready to run built elements with config
- Cannot be mutated once it is created. Any change must create a new release.
- No changes to the code at runtime



Processes

“ Execute the app as one or more stateless processes

- Multiple processes vs Multiple threads
- No shared memory, no shared filesystem
- Processes may intercommunicate via messaging / persistent storage
- Sticky sessions are a violation



Port binding

“ Export services via port binding

- App should not need a “container”, PaaS handle it!
- Use self-contained application
- Url for developers, bound in production
- HTTP is not the only service



Concurrency

“ Scale out via the process model

- Processes are first class citizens
- Define process type: web processing, workers consuming from a queue, scheduled jobs
- Vertical scaling up has it's limits



Disposability

“ Maximize robustness with fast startup and graceful shutdown

- Should be quick to start and stop
- Fast startup times help with elastic scale
- Graceful shutdown doesn't affect the end user experience and cleans up resources correctly
- Design *crash-only* software



Dev/Prod Parity

“ Keep development, staging, and production as similar as possible

- DevOps model: same tools for everyone
- Development, Staging, Production should be similar
- Reduce tool gap, time gap and personnel gap



Logs

“Treat logs as event stream

- Use stdout and stderr
- Apps are not concerned with log management
- Streams of aggregated, time-ordered events
- Sent to a file, to a terminal, to a log management tool, to a hadoop based data system

Admin processes



“ Run admin/management tasks as one-off processes

- DB Migrations, one time scripts are done by a specific admin process
- Should be run in an identical environment as the regular long-running processes
- Admin code must ship with application code

Cloud patterns

Killers have Patterns



Circuit Breaker

“ If you get the opportunity to take a break from life, take it

- VIP : Very Important Pattern
- Keep a fast response to the end user and avoid putting load on a struggling server
- Retry / back-off logic to mitigate the effect of transient failures



Cache

“ Stack Overflow did 157,370,800,409 redis ops in the past 30 days, almost always under 2% CPU

- Complex in a distributed system
- 5 levels: end-user, http, memory, shared memory, datastore
- Optimize stateless calls and fetch personalized items separately



Think async

“ If we don’t create good architecture, then in the end, we’re deceiving our customers, because we’re slowing down their ability to compete.” – Martin Fowler

- Use event-based, asynchronous systems
- Orchestration vs Choreography
- Think CQRS

More info

“we found that while the 12-factor application guidelines are spot-on, the following 7 factors are equally essential for a production environment: **Observable, Schedulable, Upgradable, Least privilege, Auditable, Securable, Measurable.**” – Shikha

- **Factor XIII: Observable**

Apps should provide visibility about current health and metrics

- **Factor XIV : Schedulable**

Applications should provide guidance on expected resource constraints.

- **Factor XV: Upgradable**

Apps must upgrade data formats from previous generations.

More info

“we found that while the 12-factor application guidelines are spot-on, the following 7 factors are equally essential for a production environment: **Observable, Schedulable, Upgradable, Least privilege, Auditable, Securable, Measurable.**” – Shikha

- **Factor XVI Least Privilege**

Containers should be running with the least privilege.

- **Factor XVII: Auditable**

Know what, when, who and where for all critical operations.

- **Factor XVIII: Securable (Identity, Network, Scope, Certificates)**

Protect app and resources from the outsiders.

More info

“we found that while the 12-factor application guidelines are spot-on, the following 7 factors are equally essential for a production environment: **Observable, Schedulable, Upgradable, Least privilege, Auditable, Securable, Measurable.**” – Shikha

- <https://medium.com/ibm-cloud/kubernetes-12-factor-apps-555a9a308caf>
- <https://medium.com/ibm-cloud/7-missing-factors-from-12-factor-application-2a3e1169bd9d>

Cloud benefits

Not If, but When



0 downtime deployment

“ Fail fast, Learn Fast, Get results
-- Netflix

- Push to activate, to rollback
- Traffic splitting
- Easy way to measure interaction with beta users
- Work anywhere



Use the right tool

“ By 2020, a Corporate "No-Cloud" Policy Will Be as Rare as a "No-Internet" Policy Is Today --Gartner

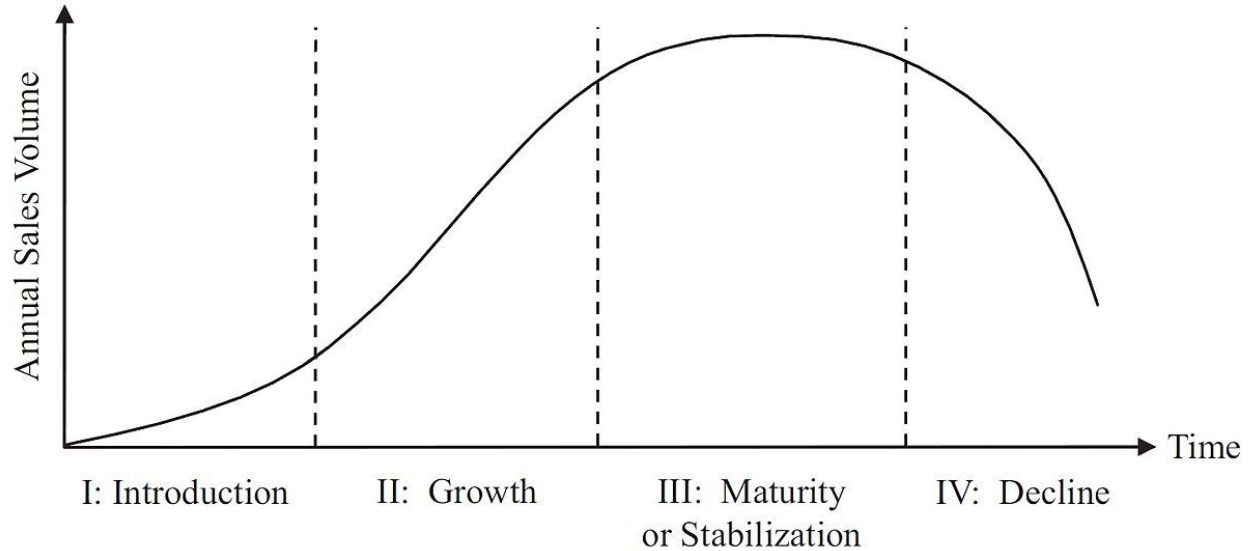
- Quick test and bootstrap
- You put effort into *CODE* and *BUSINESS VALUE*, not the platform
- Your code may be lock... but you can change your provider when you want



Pay only for actual use

“ The profits from A.W.S. represented 56 percent of Amazon’s total operating income --NYTimes

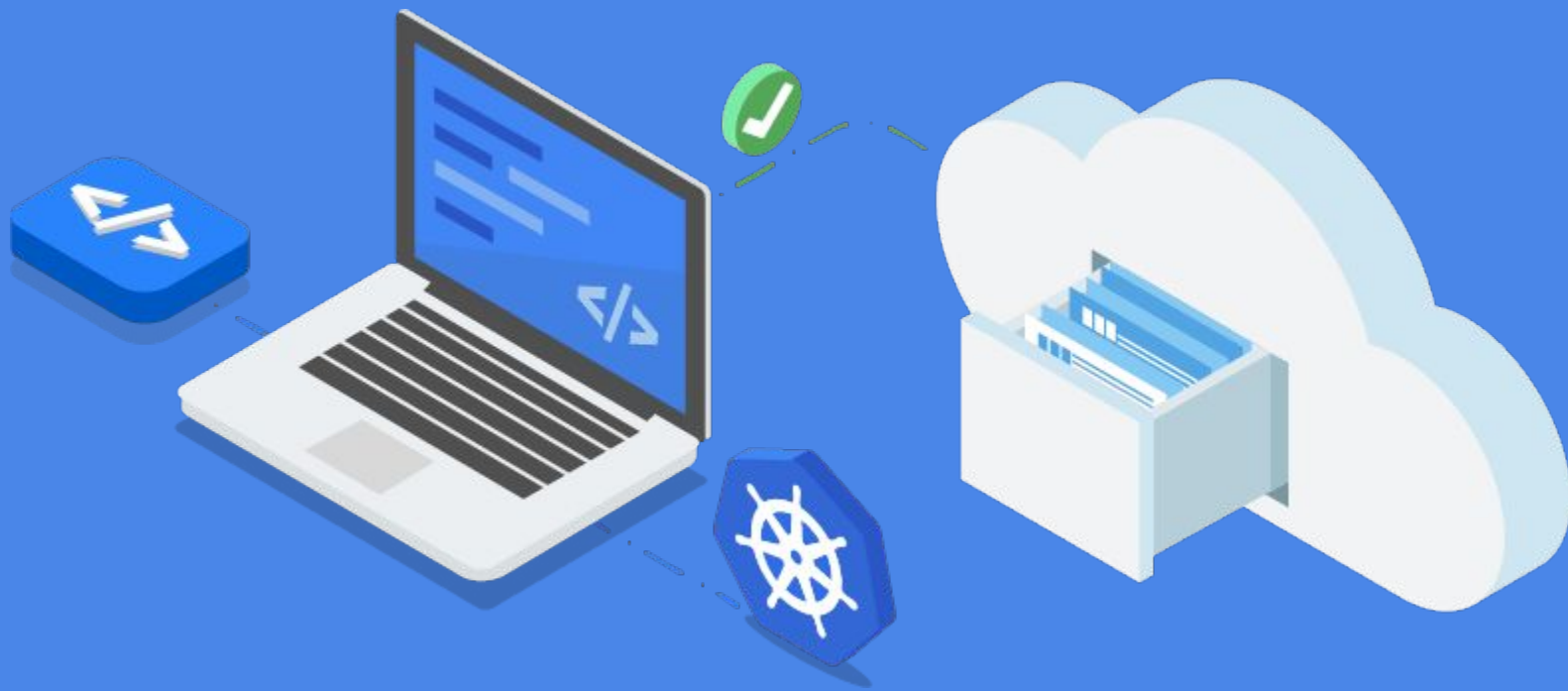
- Perfect for product lifecycle



Source: [https://en.wikipedia.org/wiki/Product_life-cycle_management_\(marketing\)](https://en.wikipedia.org/wiki/Product_life-cycle_management_(marketing))



Démo





Level Up

“ The end of ‘Fashion-IT’ – customers will only pay for value and not technology. --Sunny Ghosh

- IaaS
- Container
- PaaS
- Function



thanks

Questions?

@ZenikaNantes

