Kubernetes: The Arecibo Message for Developers

Who am I?

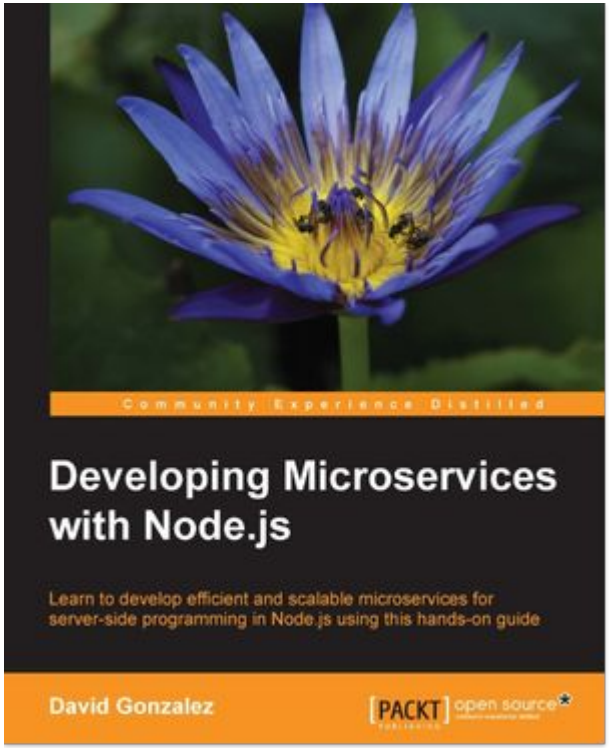
David Gonzalez (@dagonzago)

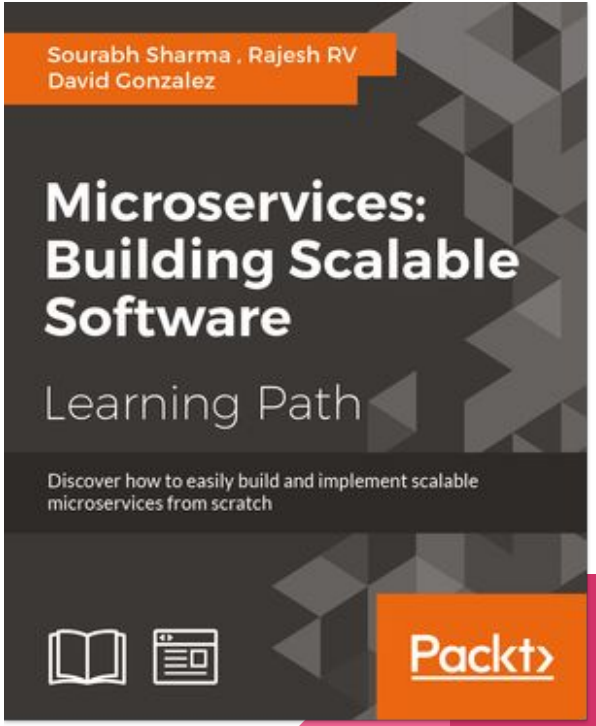
david.gonzalez@nearform.com

What do I do?

Google Developers Experts

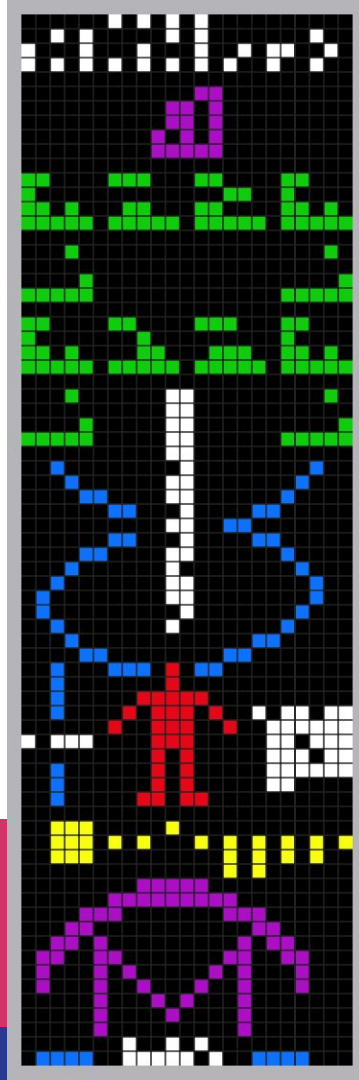
What do I do?


Community Experience Distilled
Developing Microservices with Node.js
Learn to develop efficient and scalable microservices for server-side programming in Node.js using this hands-on guide
David Gonzalez
PACKT open source


Sourabh Sharma , Rajesh RV
David Gonzalez
Microservices: Building Scalable Software
Learning Path
Discover how to easily build and implement scalable microservices from scratch
Packt

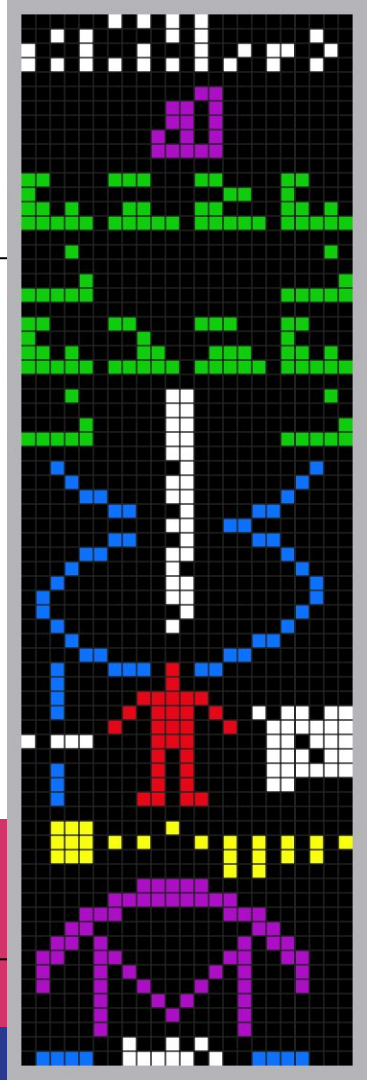
Arecibo Message

- Radiosignal sent to the deep space in 1974 carrying information about us:
  - Numbers
  - DNA information
  - Solar system
  - Human shape
  - …
- It was designed to be **culture and language agnostic** so that It can be understood but an alien intelligence.



Arecibo Message II (Curiosities)

- Will take **25000 years** to reach its destination
- It was sent on a single burst (about 3 minutes of duration) at 10 Hz per second
- It was designed to be **understood**
- **Simplicity** was always in mind



Coming back to 2017: Apocalypse

- **Software Engineering** has been around for 60 odd years but we don't have a clear defined standard for **measuring quality**
- Teams tend to be created from **horizontal slices of your company** limiting the holistic view of the system
- Developers are seen as **"brick stackers"** and decisions about the software are made by people with a very limited amount of information (and sometimes very limited amount of knowledge)
- **No one has the full picture**

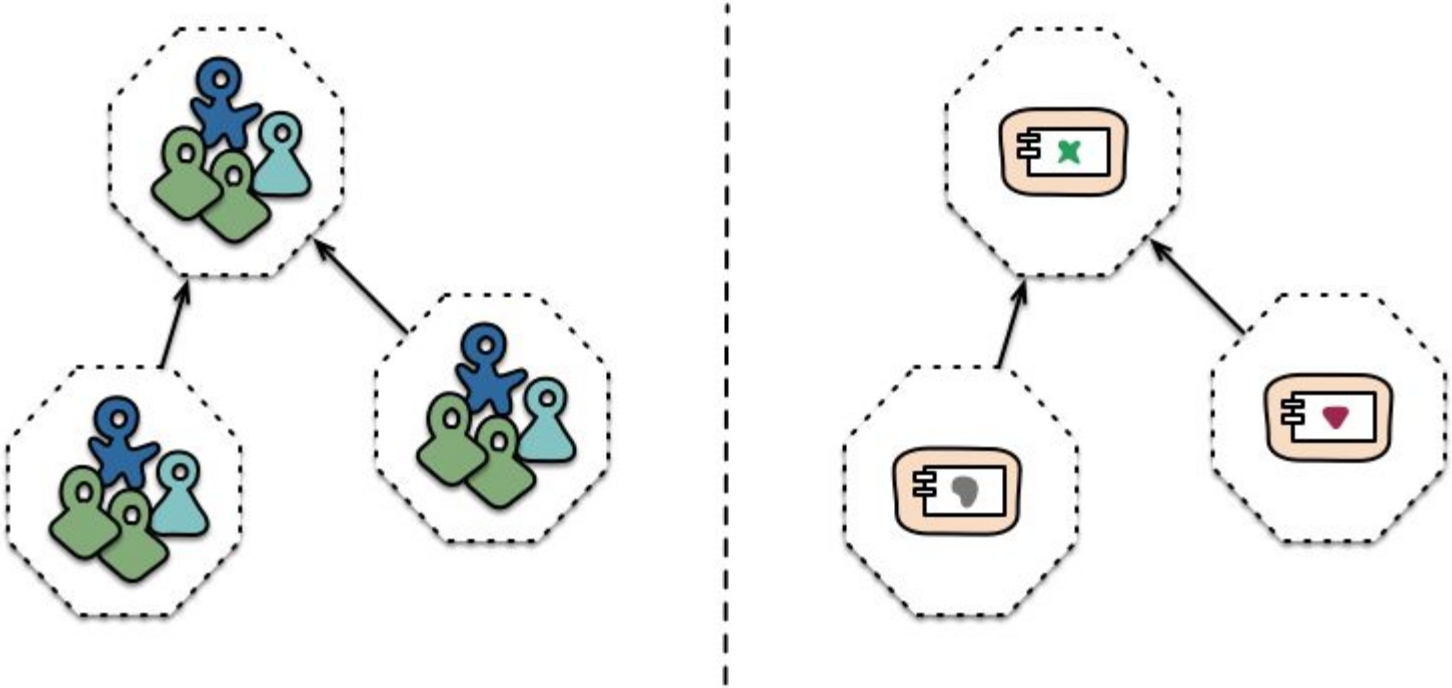
Coming back to 2017: Apocalypse TL; DR

Coming back to 2017: Apocalypse III

AND WE STILL EXPECT OUR PROJECT TO SUCCEED

Microservices came to the rescue

- Microservices are mainstream **(this is dangerous)**
- Microservices do not work without automation (DevOps > Developer)
- Microservices empower cross functional teams: **build, deploy and deliver locally: take ownership of your work**
- Developers + DevOps + Product Owner + QA = The Sweet spot of the successful microservices **continuous delivery** systems.
- Microservices come at a cost:
  - Operations overhead
  - Organizational alignment



Cross-functional teams...

... organised around capabilities
Because Conway's Law

Developers, DevOps, Product Owners and QA

- **Developers and DevOps are a must**: They are the beating heart of the software engineering.
- **Developers** are the **magicians of the Code**
- **DevOps** are the **magicians of the Infrastructure as Code**
- Both can step into each others territory
- **QA and Product Owner** roles can be assumed on a temporary basis by Developers and DevOps
- **Techies need to be business savvy.**

My Opinion ™

Developers and DevOps Arecibo Message

- **There is not a common defined language** for communication between Developers and DevOps
- I've worked in 3 companies with **continuous delivery** and each one of them built a different system in order to **deal with the same problem** and some of them were very complex…
- **Java developers know JSR standards and it is sufficient to hit the ground running in a new company**

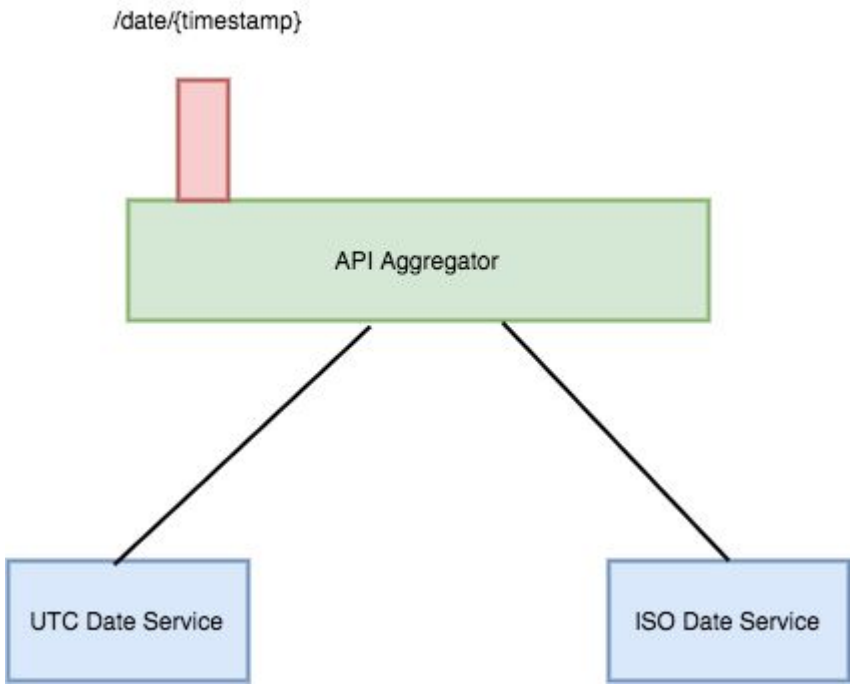
Developers and DevOps Arecibo Message II

Being proud of your system’s complexity is the quickest route to the disaster

Developers DevOps and Arecibo Message III

- We need a **common language** to define our deployments, services, quotas, configuration…
- **Kubernetes** was designed with the idea of providing a common language so that everybody can easily understand the **building blocks** of a complex application

Our system


/date/{timestamp}
API Aggregator
UTC Date Service
ISO Date Service

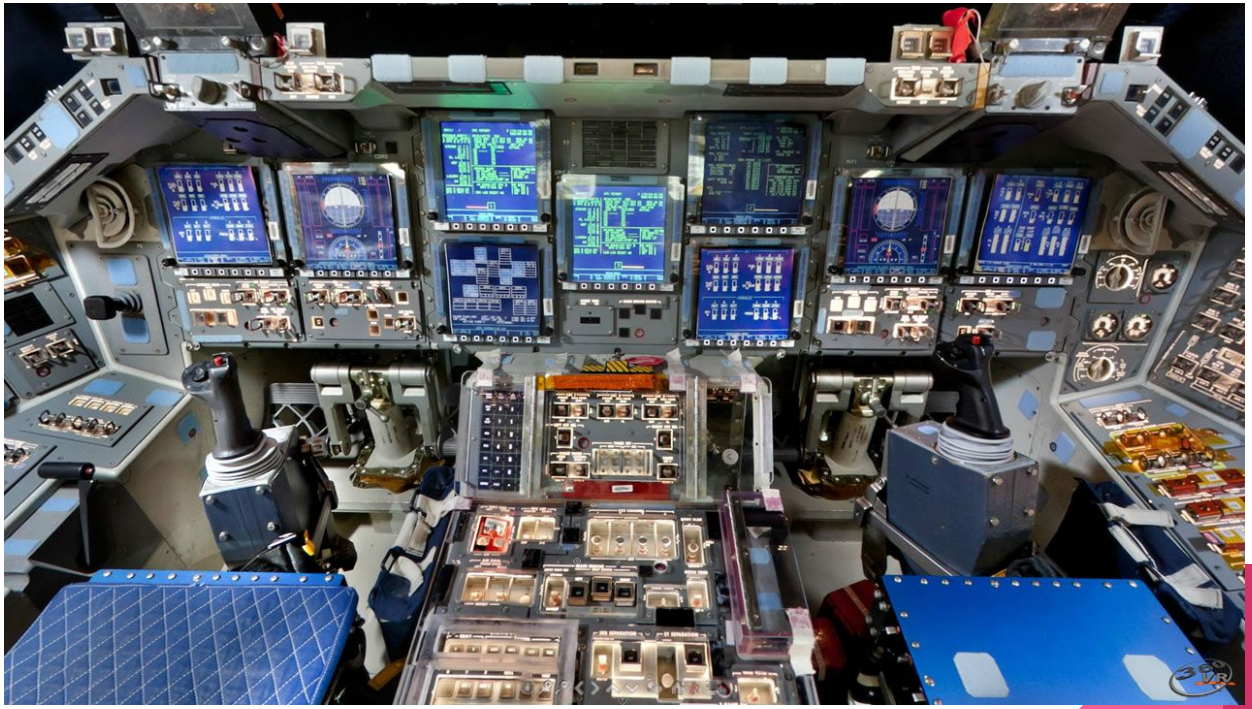
Our system

Sounds simple right?

Our system: Requirements

- High availability
- High observability
- Self healing infrastructure
- Auto-scalability
- Extensibility
- Service discovery
- Continuous delivery

Our system: Requirements



Our system: The solution

- If we try to model the above in bare metal the chances of ending with a **big mess** are huge.
- Solving each of the problems individually requires deep knowledge of many tools: **Consul, DNS, AppDynamics, ELK, Logentries…**

How do you ramp up new engineers?

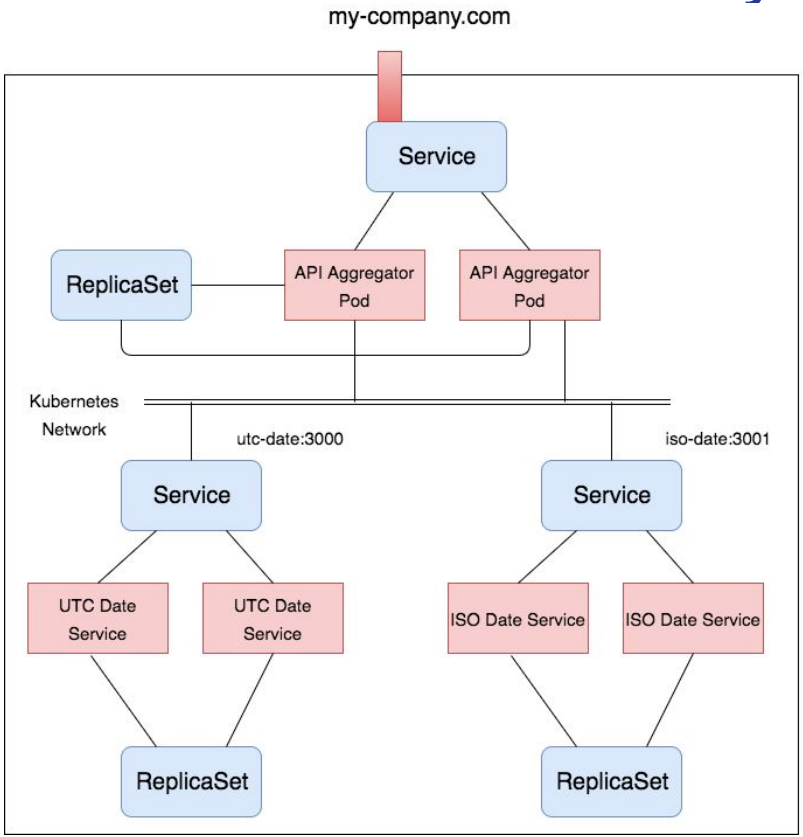
Our system: The Kubernetes Way

- **Kubernetes is the glue of the above requirements:** Service discovery, load balancing, continuous delivery… They are the Kubernetes DNA.
- **Kubernetes offers building blocks described in form of YAML** (or JSON) that are easy to read
- Kubernetes **building blocks** give us all the tools to build any system

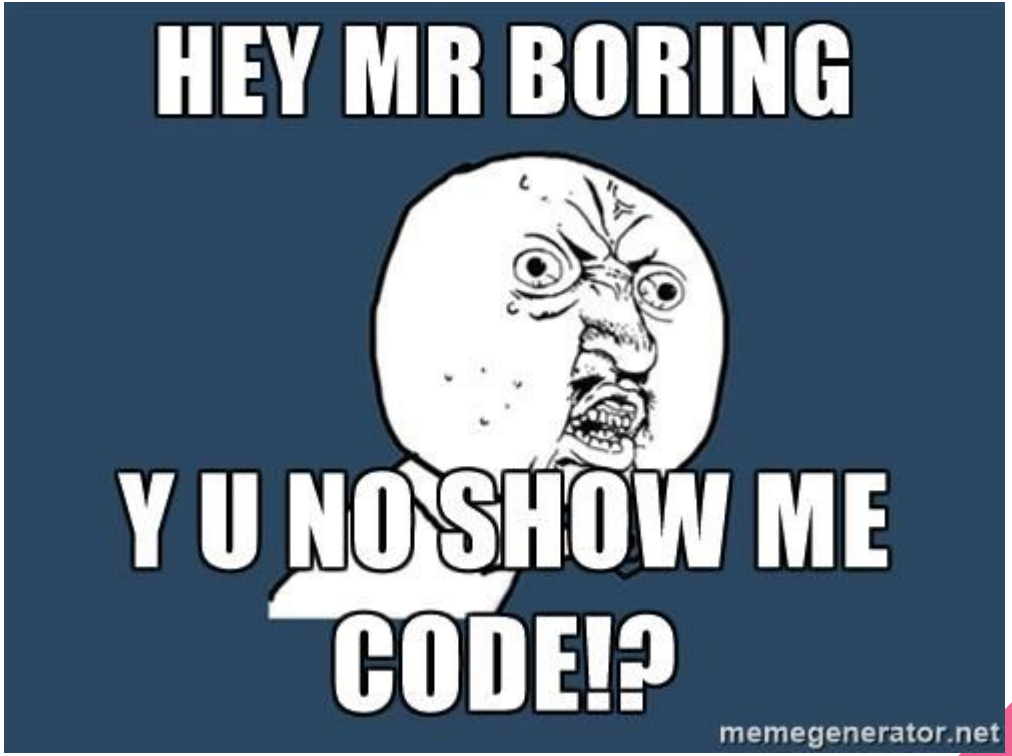
Kubernetes Building Blocks

- **Pod**: A set of containers with high affinity that work together.
- **ReplicaSet**: An abstract element that keeps a set of pods running as per specification
- **Deployment**: A high order element that allows rolling upgrades of a ReplicaSet
- **Service**: An abstract entity that allows us to define how our Pods communicate between themselves or even with world outside Kubernetes
- **HorizontalPodAutoscaler**: Manages the scalability of a ReplicaSet

Our System The Kubernetes Way



Let’s see a demo



Thanks! - Q&A

David Gonzalez (david.gonzalez@nearform.com)

@dagonzago

https://www.linkedin.com/in/david-gonzalez-microservices/

I'll be around this evening so, don't hesitate to chat to me!