

**The right way to embrace docker!**

# agenda for next 25 mins!

- about the speaker
- motivation and subject of this chat
- our happy story with Docker
- docker adoption milestones
- recommendations

# About the speaker




linkedin.com/in/luixg



hadesbox



luis.gonzalez (at) beeva (dot) com

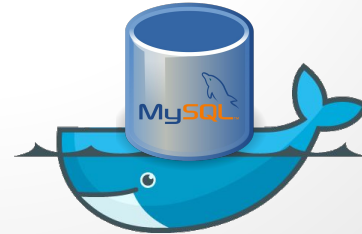
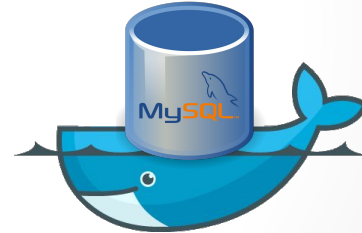
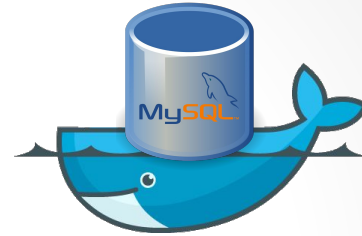
- Full time **engineer** at  B E E V A / **BBVA**
- Focused on **scalability**, **cloud** (AWS), **backend** (REST, Python, Java, Go), **devops** (docker, git, jenkins, puppet), **analytics** (hadoop ecosystem) and **Linux**!

# Disclaimers...

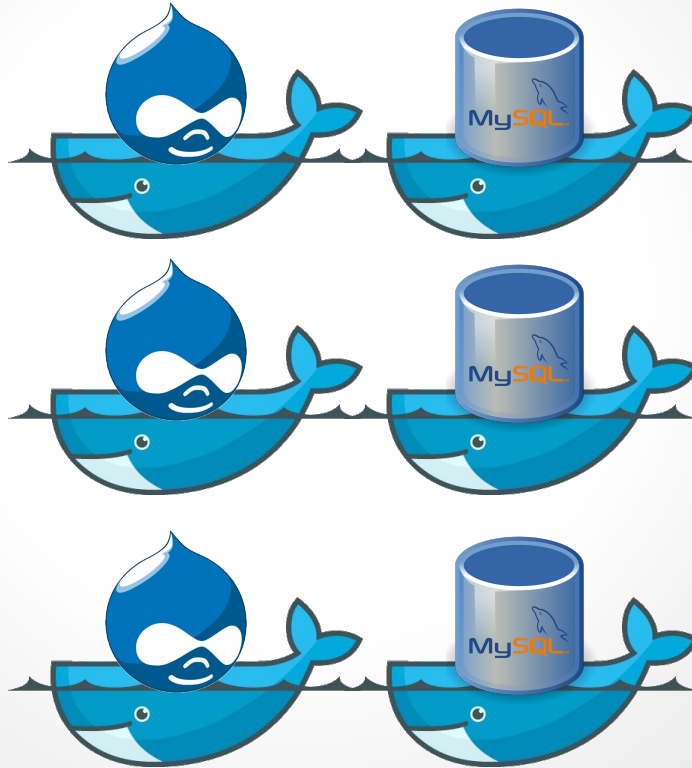
- The prestige is not only mine, many engineers have collaborated.
- This is techno philosophy session!



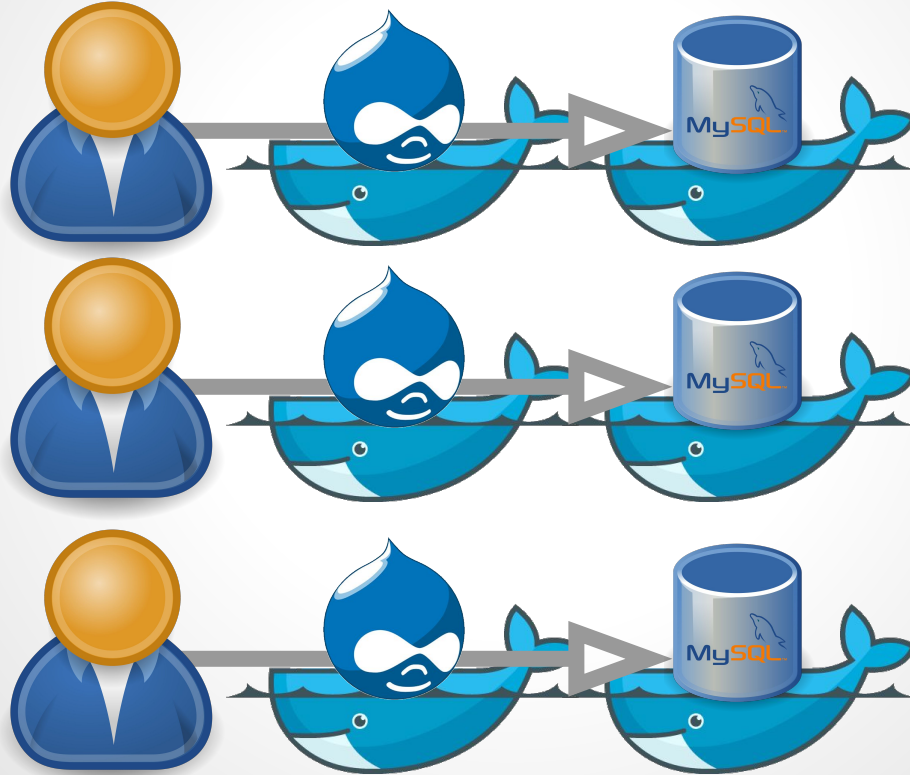
# Motivation ...



# Motivation ...

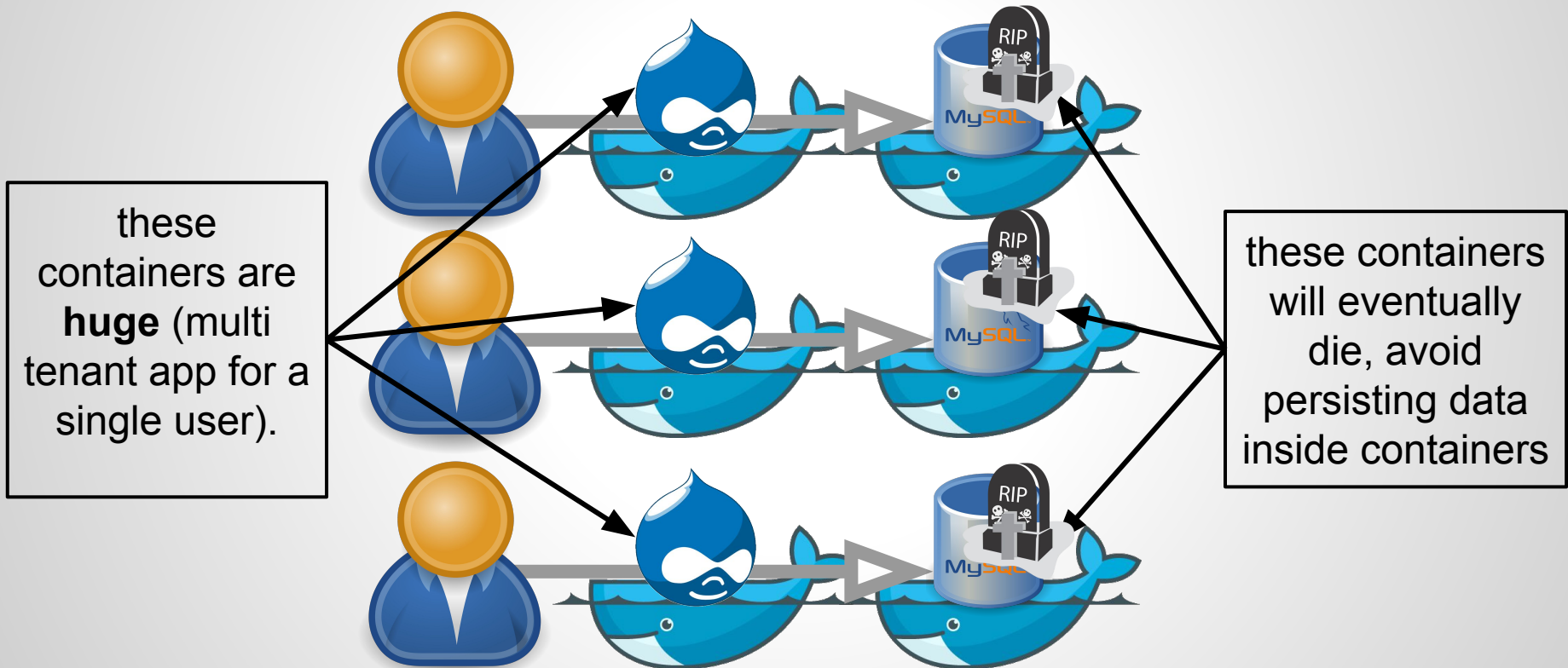


# Motivation ...





# Motivation - don't do it!





# Philosophy 101.

“**Docker** is really simple, but we insist on making it complicated.”

- Confucius



# The misunderstanding

*“Docker’s domain spans a few crowded segments of the industry that include enterprise virtualization platforms (**vmware, xen, kvm**) configuration management tools (**puppet, chef, ansible**) deployment framework (**capistrano, fabric**) cloud platform (**openstack, cloudstack**) and development tool (**vagrant**)”*

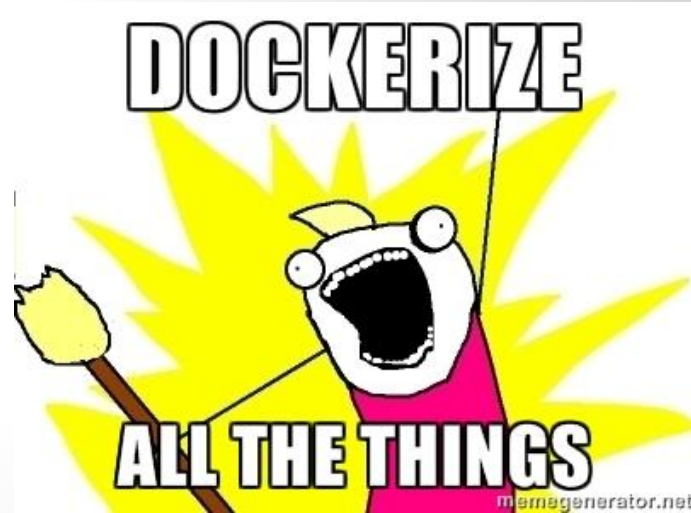
O'Reilly - Docker up & running - Chapter 1, the promise of docker...

# Considerations of docker.

1. docker promotes engineer's independency.
2. Enables automation for daily operations.
3. It's a way (not the only one) to create, deploy and scale applications.
4. Makes whole development process more complex, **automate! automate! automate!**
5. It's **NOT** an all-in bet! Your use case is probably different from the guy next to you.

# Our story started summer 2014.

- 9 Jun 2014 **docker 1.0** is out!
- It's a nice “virtualization” tool.
- Lightweight and fast boots!
- Full replicated environments.
- Proof-of-concept.



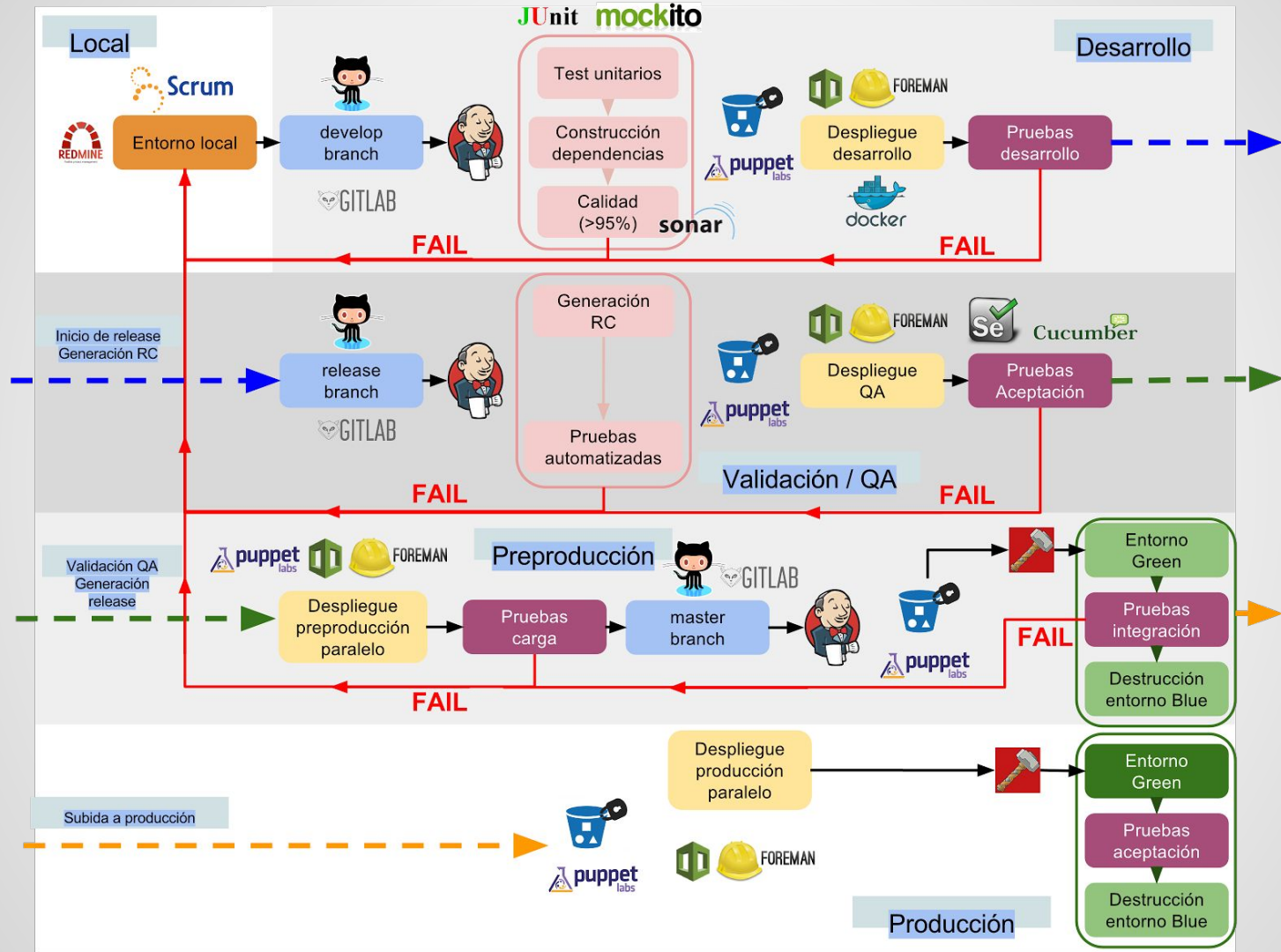
## Q4 2014 ... After POC.

- No integrated **networking**.
- No official **clustering** solution back then,  
available: fleet (coreos), helios (spotify) or Juju  
(canonical) with consul or etcd.
- Sep 2014 **Kubernetes** v0.2 is Open Sourced!
- Oct 2014 Fig 1.0 (now **DockerCompose**).
- Dockerize an application: REST services inside  
containers.

# The story continues...

- BAD IDEA, we added more complexity to our use case:  
**Microservices.**
- Solution we currently use:  
**Netflix Spring Cloud, AWS ECS, Asgard (green-blue), Jenkins and many other things.**







# Important lessons from Project! 1/2

- **Networking** (secure communication multihost).
- **Clustering** (HA, elasticity, autoscaling, multicontainer apps, error handling).
- **Service Discovery** and **configuration**.
- **Lifecycle** of images & containers.
- **μdistros** (alpine, busybox).
- Treat containers as if were **disposable** (they will fail... unexpectedly and eventually).

# Important lessons from Project! 2/2

- Containers **log management**.
- Containers **tagging** (versioning) policy.
- Old container **cleaning/disposal**.
- Infrastructure as code, CI, CD, BlueGreen.
- **Right size** (for our)  $\mu$ services.
- Docker  $\neq$   $\mu$ services.
- Don't dockerize all (databases, stateful components).

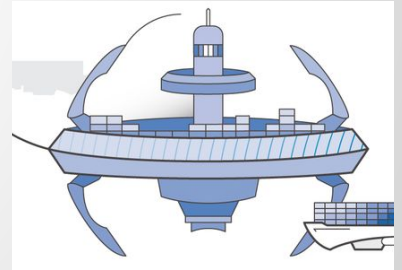
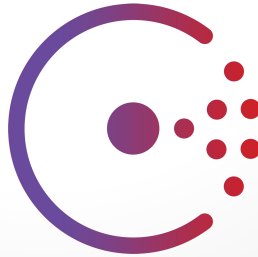
# Docker adoption milestones.

Start small, think big, automate all!

1. Start creating your **local environment**, useful for local development (dockerfile, docker compose).
2. Deploy containers on dev environment and configure **continuous integration/deployment** (single host).
3. Automate all your tests (functional, integration) and use docker for testing.

# Docker adoption milestones

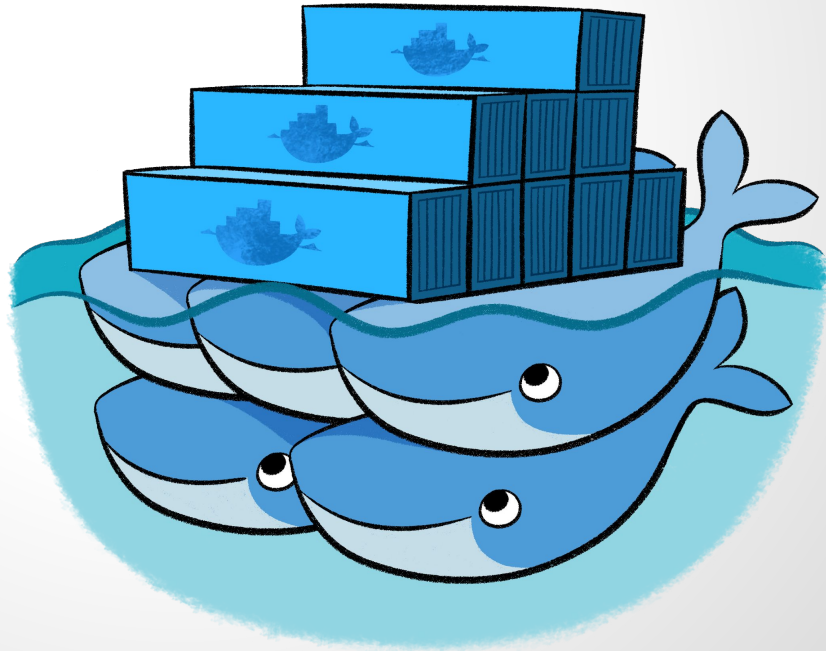
4. Choose your clustering platform (onprem, cloud) and deploy **multi host** and **multi container** applications with simple networking.



# Docker adoption milestones

5. autoscaling, autodiscovery, private networks for containers and perks.

6. to **μmicroservices**?



# Final advice C-levels.

The most valuable benefit that docker provides is that it promotes a **shift of responsibility** of the production operation from **ops to devops**, this means developers, projects and product owners get involved more on the daily operations of the business...

pizza teams!   product oriented business!

**Thanks!**