Microservices

Carlos Avendaño

Agenda

Microservices Architecture

API Gateway

Inter-Process Communication (IPC)

Service Discovery

Microservices Deployment Strategy

Refactoring a Monolith into Microservices

Docker Platform (Samples)

Kubernetes

Note:

“Most of the topics in this presentation is based on Microservices From Design Deployment blog articles by Chris Richardson with Floyd Smith.”

https://www.nginx.com/blog/introduction-to-microservices/

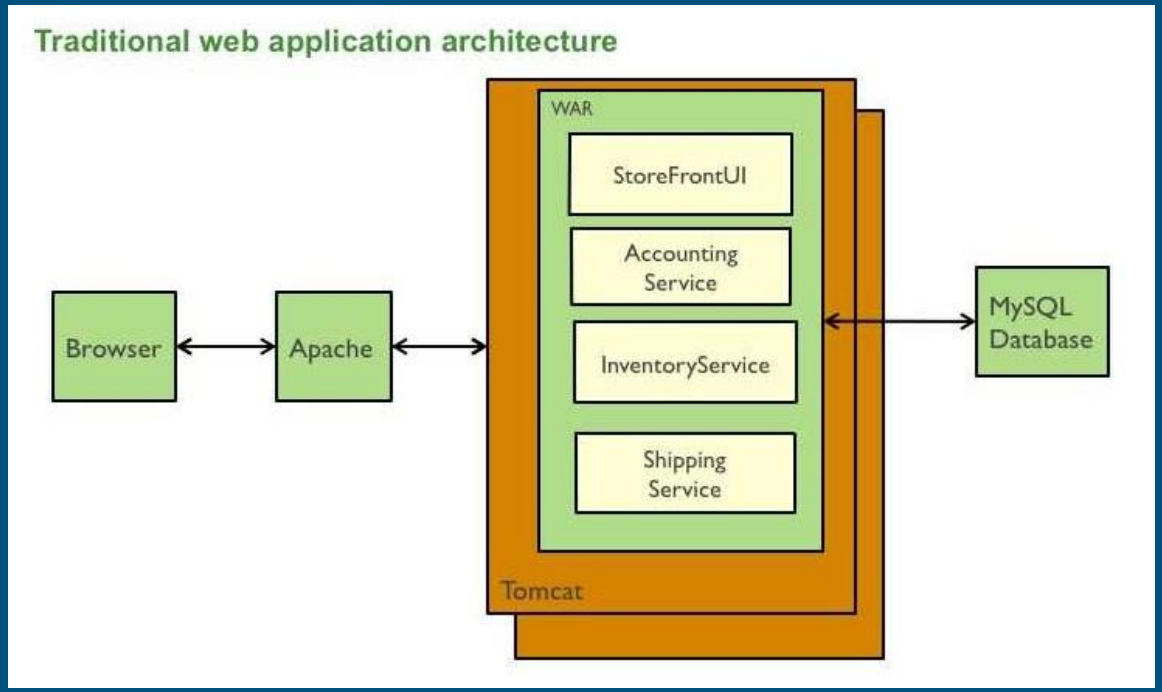
Microservices Architecture

The term “Microservices Architecture” has sprung up over the last few years to describe a particular way of designing software applications as suites of independently deployable services. (*James Lewis, Martin Fowler*)

Microservices are small, autonomous services that work together. (*Building Microservices - Sam Newman*)

https://martinfowler.com/articles/microservices.html - https://www.nginx.com/wp-content/uploads/2015/01/Building_Microservices_Nginx.pdf

Monolithic Applications



Simple to develop, test, deploy, scale.

BUT...

Large monolithic code base

Overloaded IDE

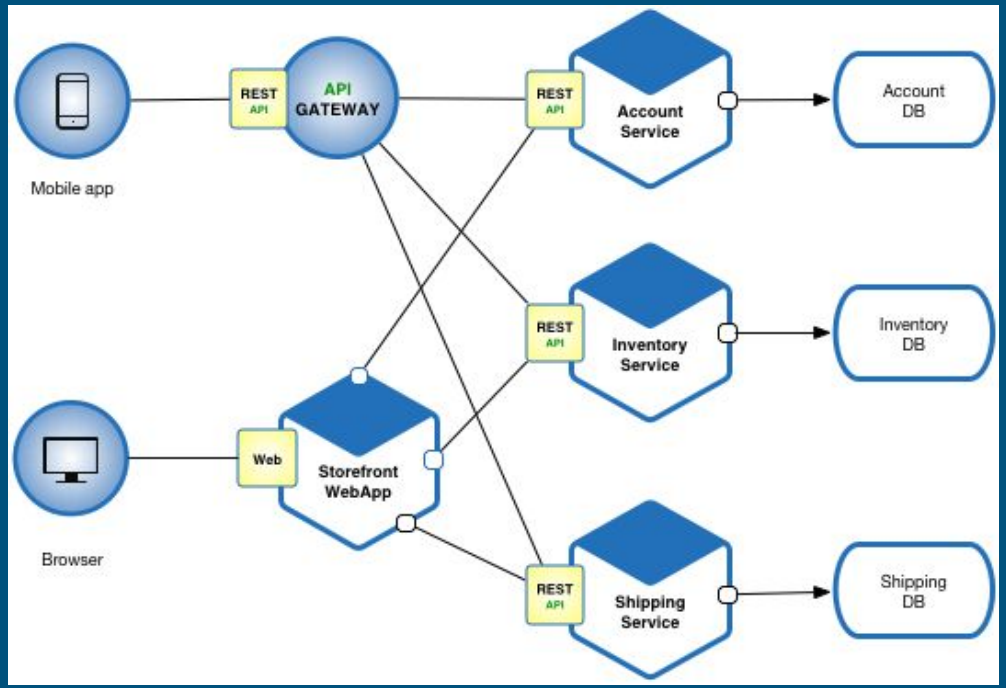
Overloaded web container

Continuous deployment

Scaling the application

http://microservices.io/patterns/monolithic.html

Microservices Architecture



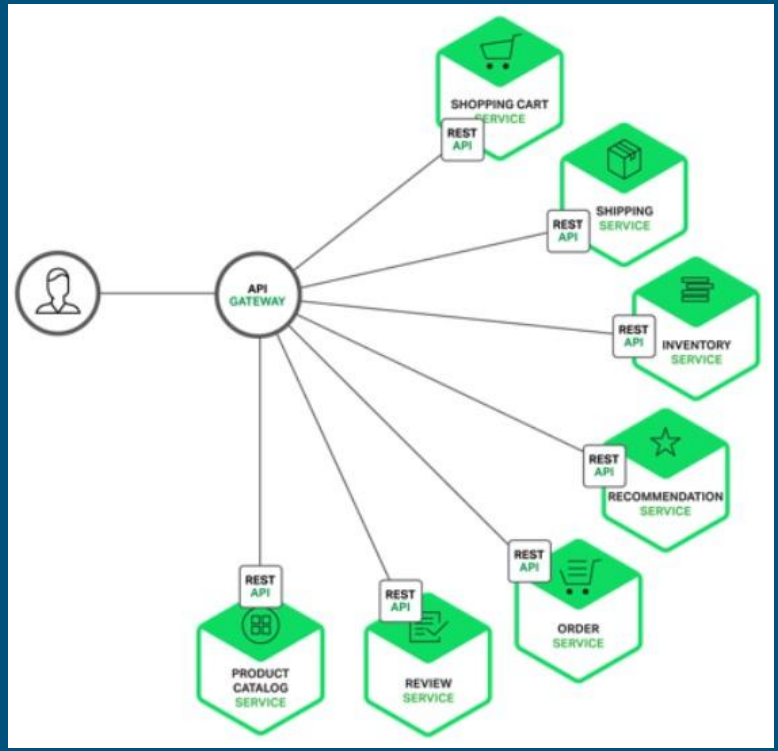
- Team of developers working on each service
- New team members must quickly become productive
- The application must be easy to understand and modify
- Continuous deployment
- Multiple instances of the application

http://microservices.io/patterns/microservices.html

API Gateway

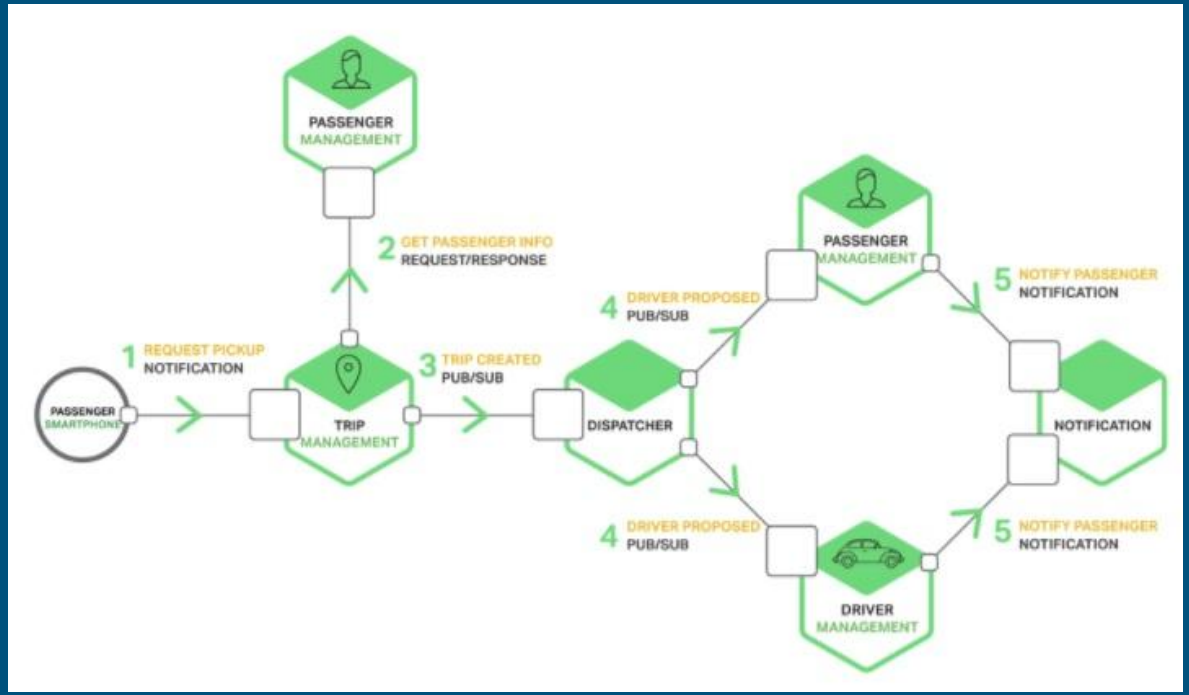
The API Gateway is responsible for request routing, composition, and protocol translation.

All request from clients first go through the API Gateway. It then routes requests to the appropriate microservice.



https://www.nginx.com/blog/building-microservices-using-an-api-gateway/

Inter-Process Communication



Interaction Styles

How services interact.

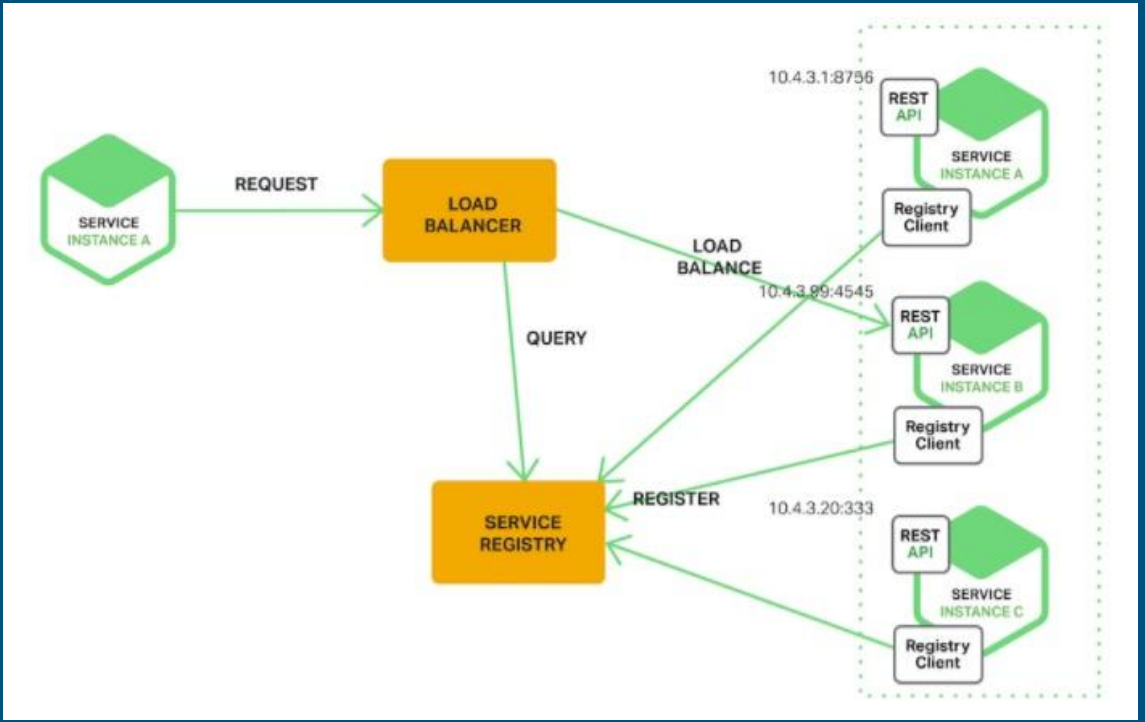
One to One
One to Many

Synchronous
Asynchronous

https://www.nginx.com/blog/building-microservices-inter-process-communication/

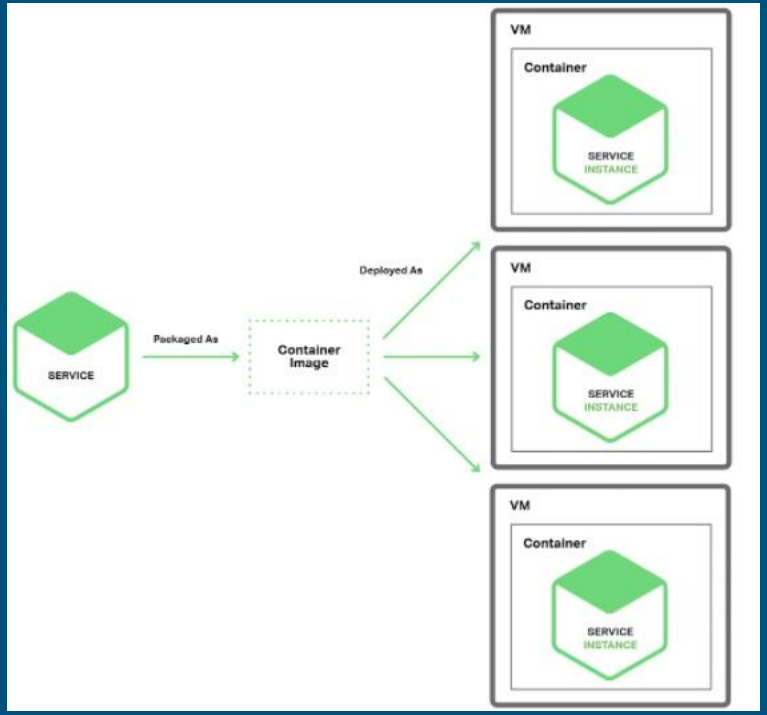
Service Discovery

Service instances have dynamically assigned network locations. Moreover, the set of service instances changes dynamically because of autoscaling, failures, and upgrades.



https://www.nginx.com/blog/service-discovery-in-a-microservices-architecture/

Microservices Deployment Strategy



A microservices application consists of tens or even hundreds of services. **Services are written in a variety of languages and frameworks.** Each one is a mini-application with its own specific deployment, resource, scaling, and monitoring requirements.

Multiple Service Instances Per Host Pattern

https://www.nginx.com/blog/deploying-microservices/

Refactoring a Monolith into Microservices

Strategy #1 - Stop Digging

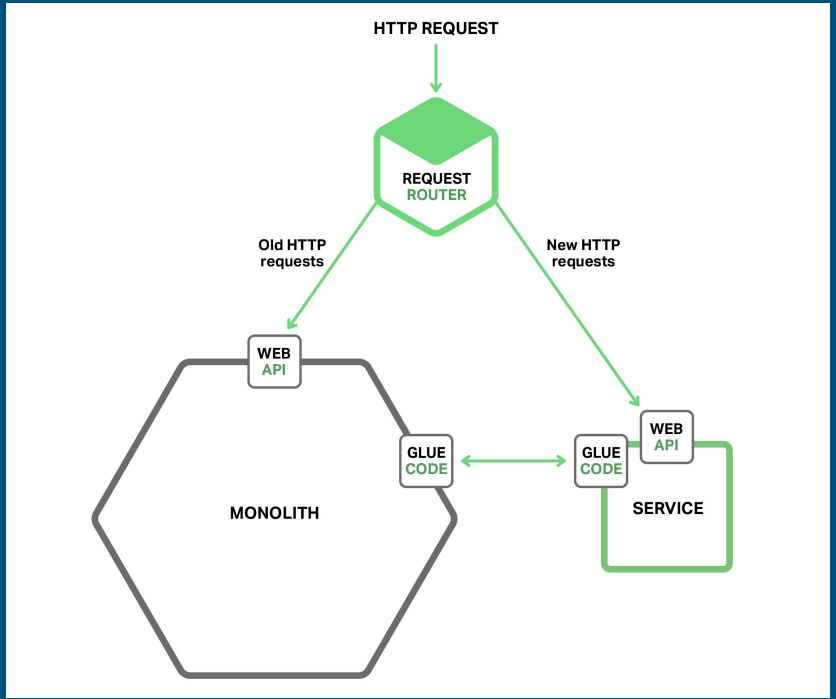
Strategy #2 - Split Frontend and Backend

Strategy #3 - Extract Services

Strategy #1 - Stop Digging

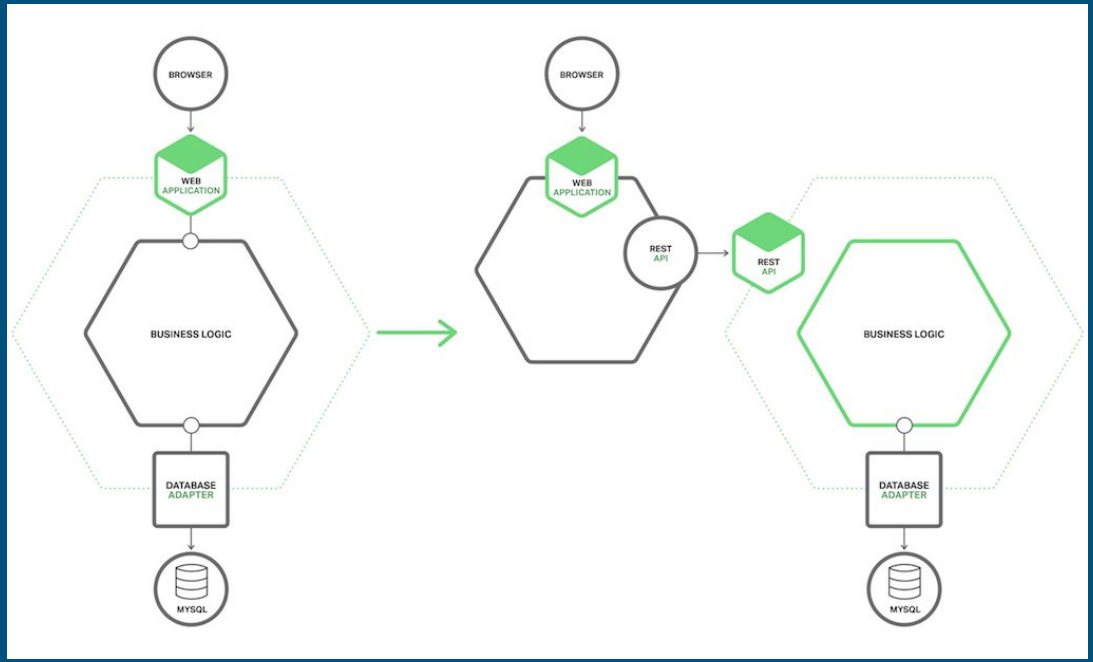
When we are implementing a new functionality we should not add more code to monolith. Instead, the big idea with this strategy is to put new code in a standalone microservice.

This figure shows the system architecture after applying this approach.



https://www.nginx.com/blog/refactoring-a-monolith-into-microservices/

Strategy #2 - Split Frontend and Backend

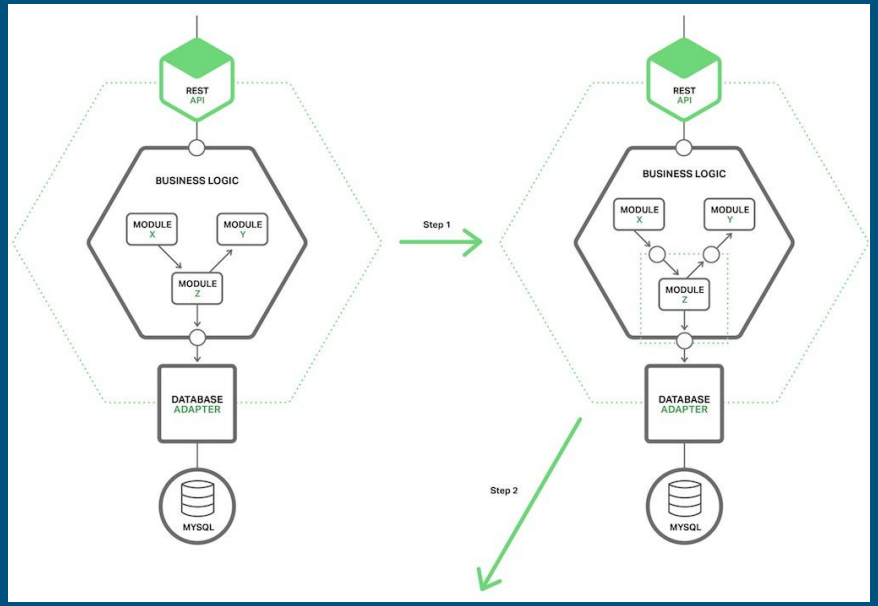


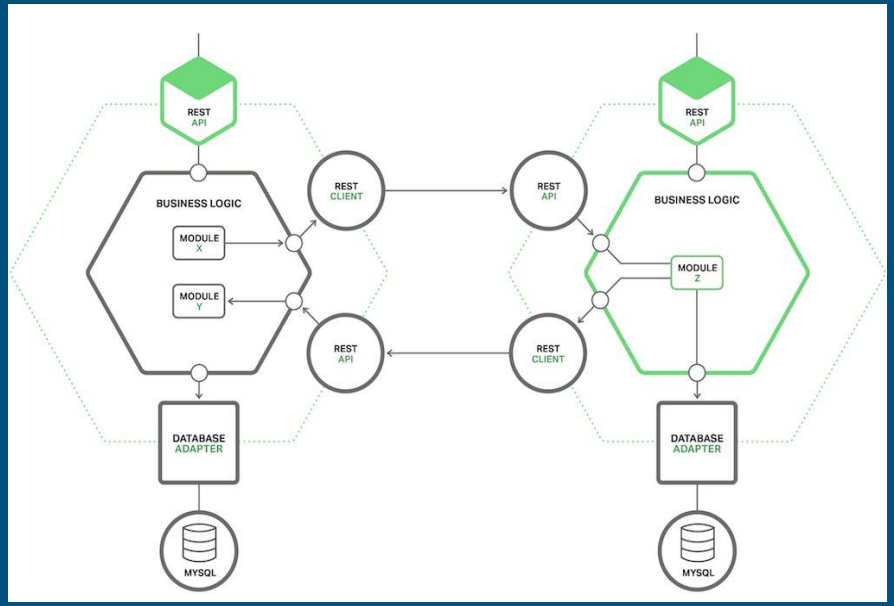
Splitting a monolith in this way has two main benefits. It enables you to develop, deploy, and scale the two applications independently of one another. In particular, it allows the presentation-layer developers to iterate rapidly on the user interface and easily perform A|B testing, for example.

Another benefit of this approach is that it exposes a remote API that can be called by the microservices that you develop.

https://www.nginx.com/blog/refactoring-a-monolith-into-microservices/

Strategy #3 - Extract Services





https://www.nginx.com/blog/refactoring-a-monolith-into-microservices/

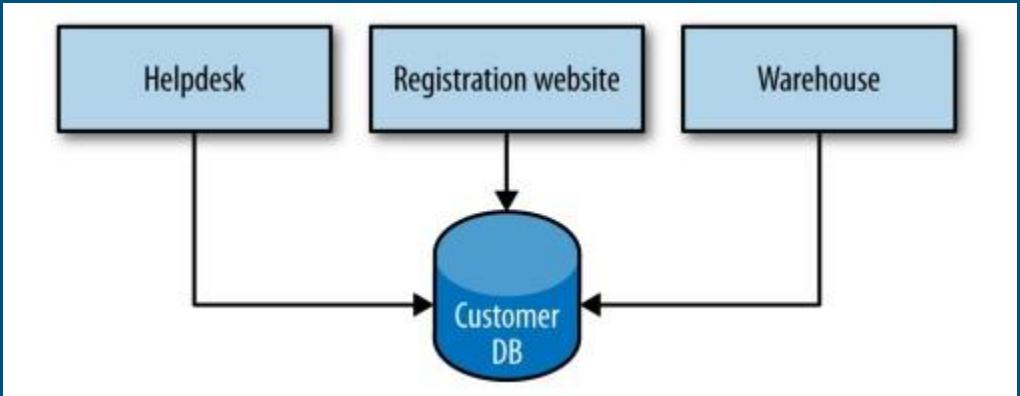
Decomposition patterns

How to decompose an application?

- Decompose by business capability
- Decompose by domain-driven design subdomain
- Decompose by verb or use case
- Decompose by nouns

http://microservices.io/patterns/microservices.html

Shared Database



What if over the time we realize we would be better off storing data in a nonrelational database? (Goodbye, loose coupling)

What if there is going to be logic associated with how a customer has changed and we need to modify the bd? (Goodbye, cohesion)

https://www.nginx.com/wp-content/uploads/2015/01/Building_Microservices_Nginx.pdf

Docker Platform

“Docker is a platform for developers and sysadmins to develop, deploy, and run applications with containers. The use of Linux containers to deploy applications is called containerization. Containers are not new, but their use for easily deploying applications is.”

https://docs.docker.com/get-started/

Docker Platform

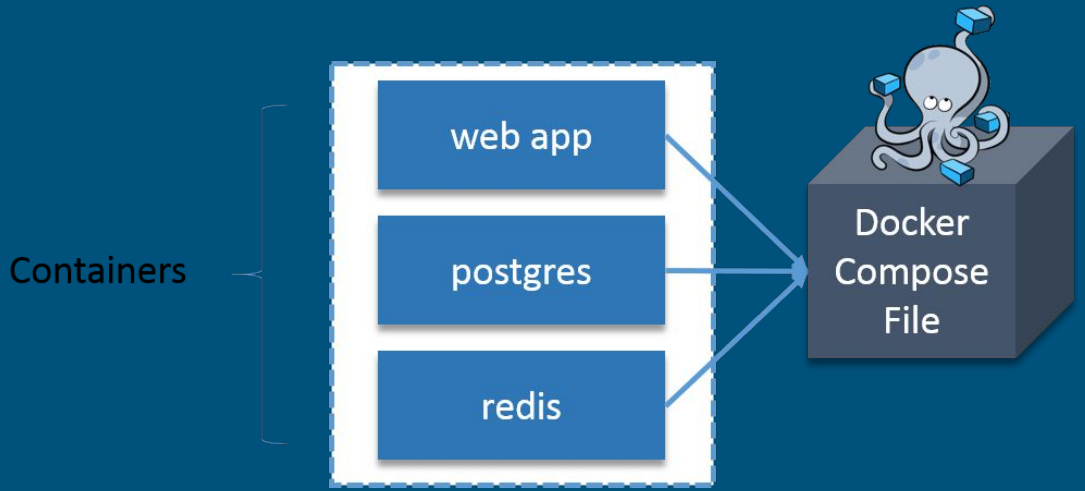
- Containers
- Services
- Swarms Clusters
- Stacks
- Deployment

Dockerfile

```
FROM openjdk:8-jdk-alpine
MAINTAINER c.avendano10@gmail.com
CMD java ${JAVA_OPTS} -jar test-producer-*.jar
COPY target/test-producer-*.jar .
```

https://docs.docker.com/get-started/

Docker Platform

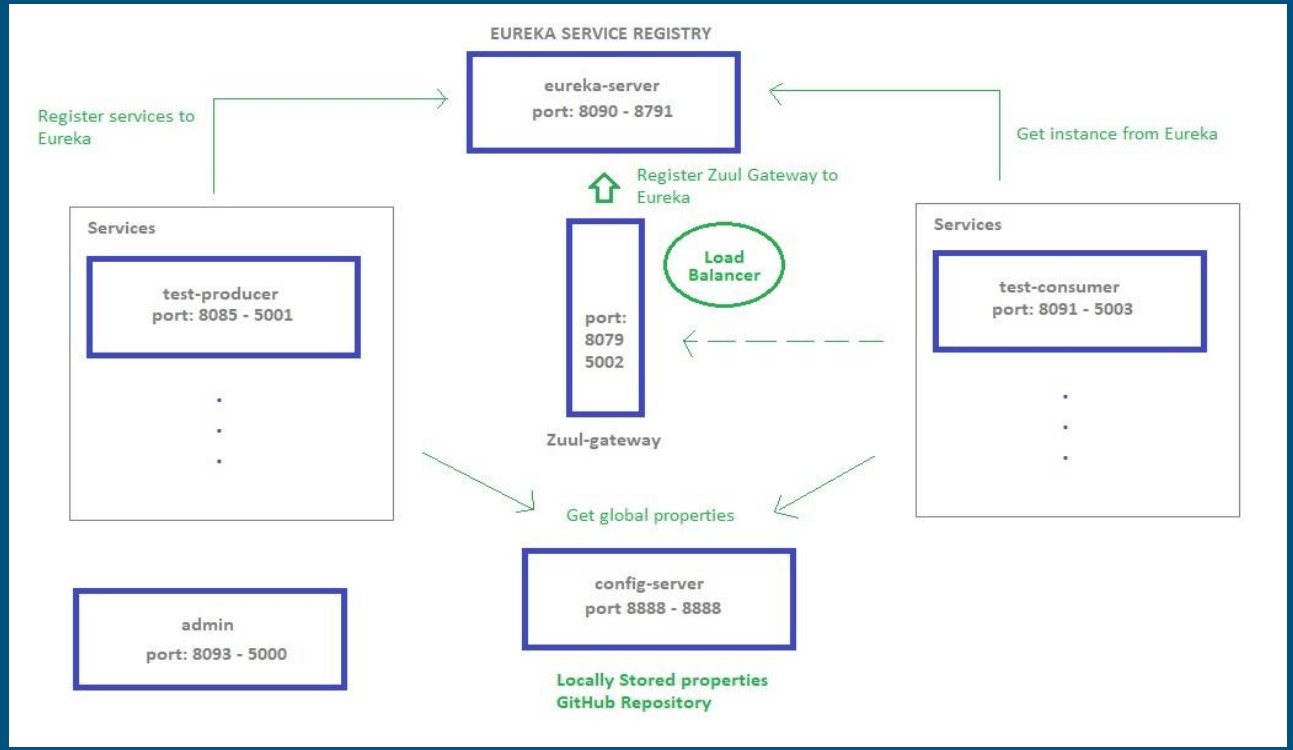


https://www.edureka.co/blog/docker-container/

Samples

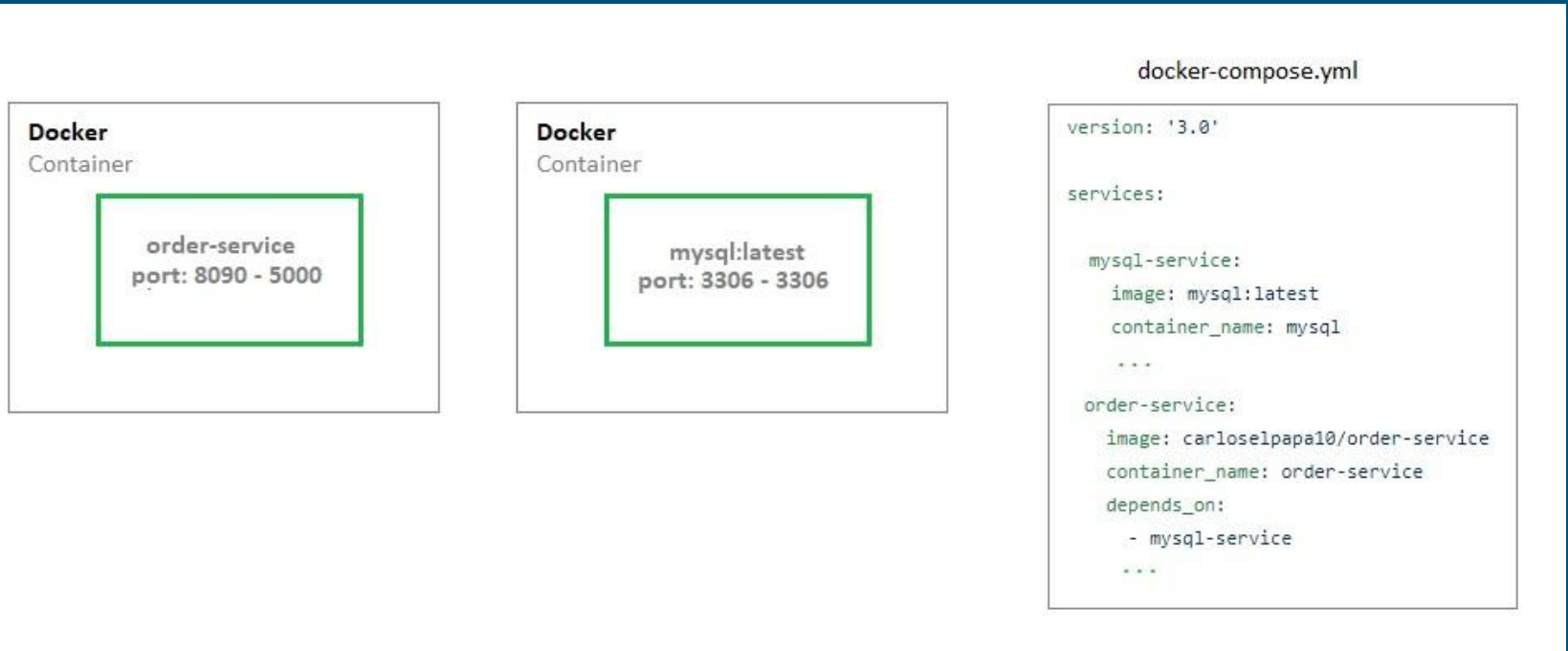
1. Spring Cloud with Netflix Eureka
2. Dockerize first sample
3. Spring Boot + Docker + Mysql
4. JUnit Tests + Docker + Jenkins

Sample 1 & 2



https://github.com/carloselpapa10/mainclass/tree/master/2.%20Docker%20Sample

Sample 3



https://github.com/carloselpapa10/mainclass/tree/master/3.%20Docker%20Sample%202

Sample 4 - Jenkins Blueocean

https://github.com/carloselpapa10/mainclass/tree/master/4.Jenkins

Other Sample - Jenkins Docker Processes

https://github.com/carloselpapa10/admin

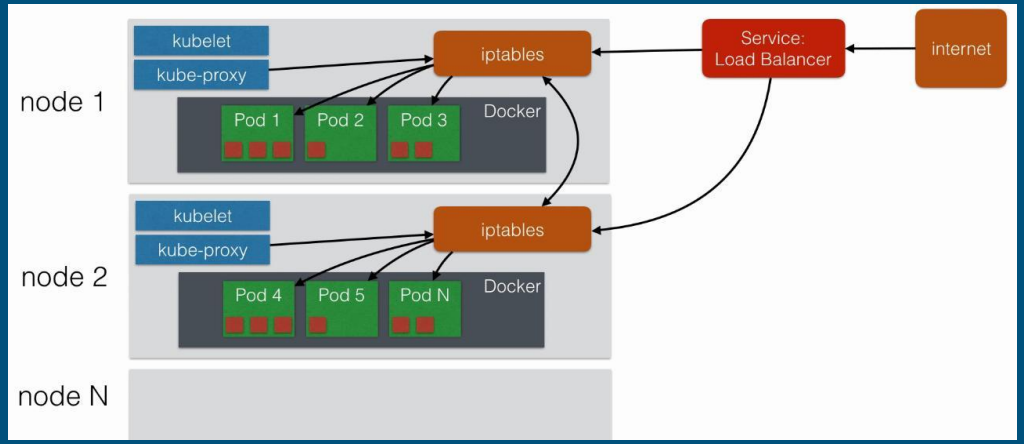
Swagger UI

Swagger UI allows anyone to visualize and interact with the API's resources without having any of the implementation logic in place. It's automatically generated from your Swagger specification, with the visual documentation making it easy for back end implementation and client side consumption.

https://swagger.io/swagger-ui/

Kubernetes

Kubernetes is an open-source system for automating deployment, scaling and management of containerized application.



Kubernetes Architecture

https://kubernetes.io/ - https://www.udemy.com

Conclusion

The concepts and tools presented throughout this presentation show us a different approach to build, test and deploy applications. The Microservices Architecture has a particular way of designing software applications as suites of independently deployable services.

It is time to switch from our monolith applications to microservices architecture applying patterns seen so far and dockerize the world.