# API e Microservizi

FullStackConf - 10/10/2019

# Walter Dal Mut

aws certified Solutions Architect Professional

aws certified Solutions Architect Associate

aws certified SysOps Administrator Associate

docker CERTIFIED ASSOCIATE

@walterdalmut
walter.dalmut@corley.it
corley.it

aws partner network
Standard
Consulting Partner

corley

# SUMMARY

# API?

APIs immediately creates a new building block for any application

# I want to add filesystem feature to my application?

## Create a filesystem API layer

- Manage files and folders?
- Enable collaboration?
- Detect changes and revisions?

# Filesystem as a service

```
$fileMetadata = new Google_Service_Drive_DriveFile([
  'name' => 'photo.jpg'
]);

$file = $driveService->files->create($fileMetadata, [
  'data' => file_get_contents("/tmp/photo.jpg"),
  'mimeType' => 'image/jpeg',
  'uploadType' => 'multipart',
  'fields' => 'id'
]);
```

# Think about any AWS service!

- S3 - Simple Storage Service
- Lambda - Function as a Service
- SQS - Simple Queue Service
- SNS - Simple Notification Service
- ….

Everything is an API in AWS

# Service or Microservice?

- Granularity
  - microservices are typically single-purpose
- Component Sharing
  - SOA enhances component sharing than microservices tries to minimize them

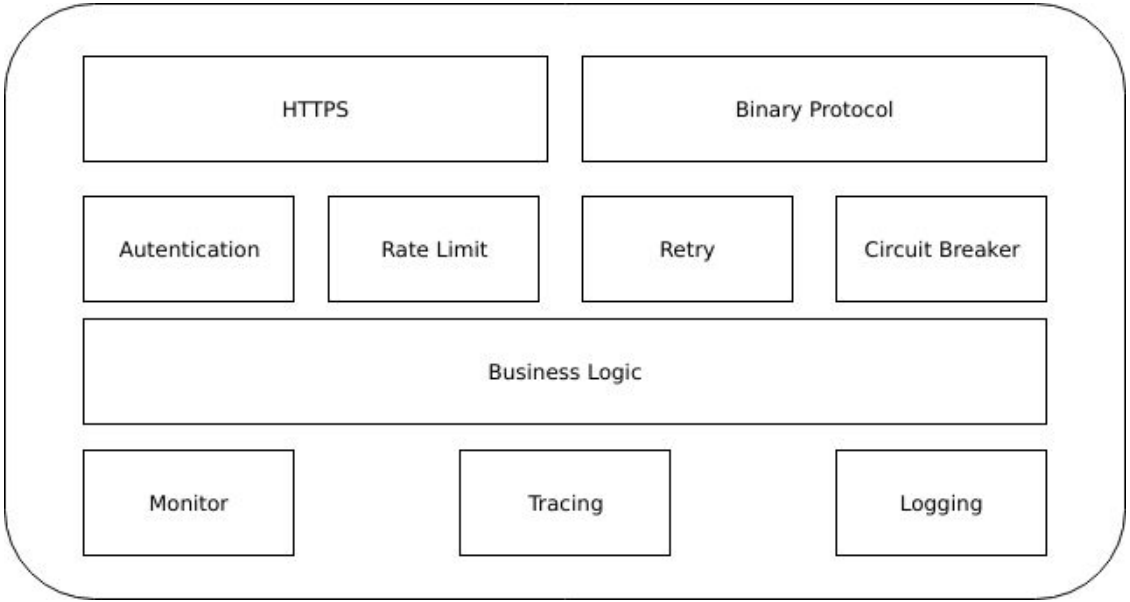
# How to think about microservice with a reference (from OOP)?

S.O.L.I.D.

Single Responsibility Principle
Open/Closed
Liskov substitution principle
Interface segregation principle
Dependency inversion principle

Use it as a starting point not a rule!

# My business logic now is small but typically...

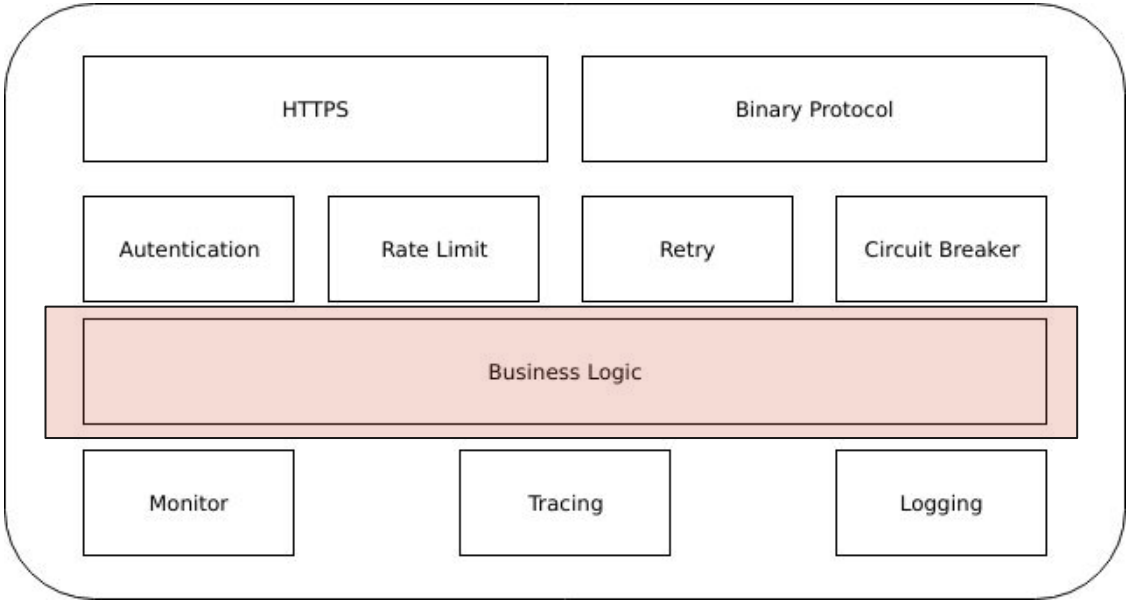



Fat Microservice

# Why fat microservice?

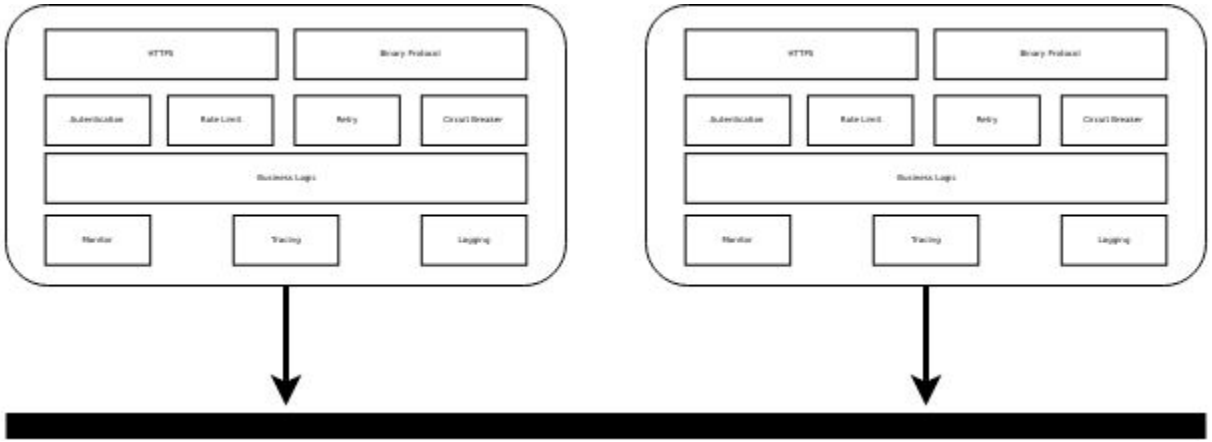
- Authentication
- Authorization
- Monitor
- Tracing
- Logging
- …

# What is a microservice?




My Business Logic is a microservice!

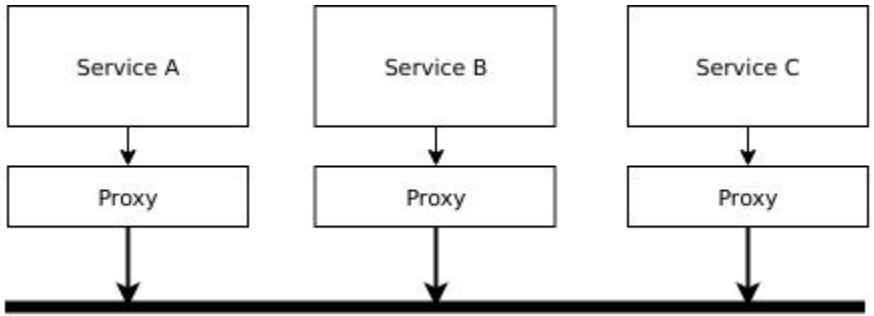
# Many services - Many connections



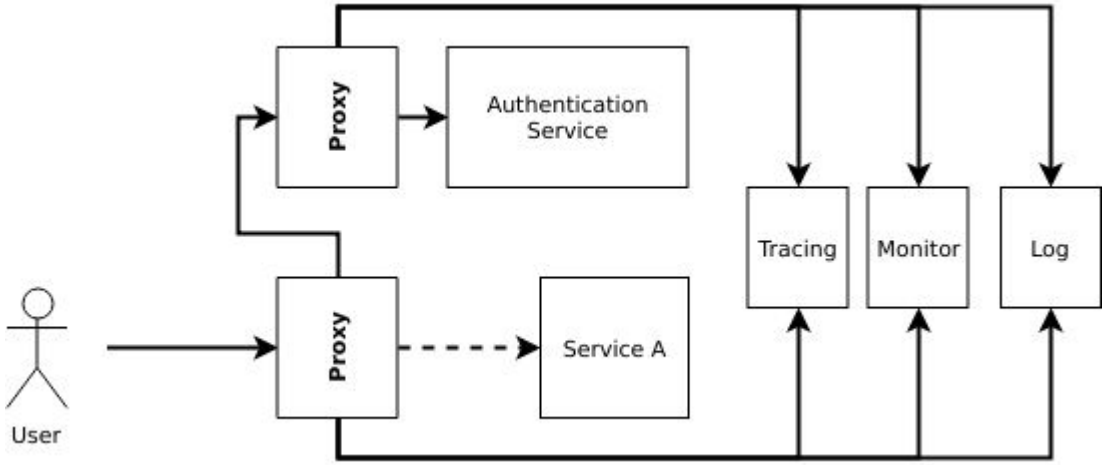
# Shift features to a dedicated component (sidecar containers)

- Authentication
- Authorization
- Monitor
- Tracing
- Logging
- Retry
- Rate Limit
- Circuit Braker
- ...

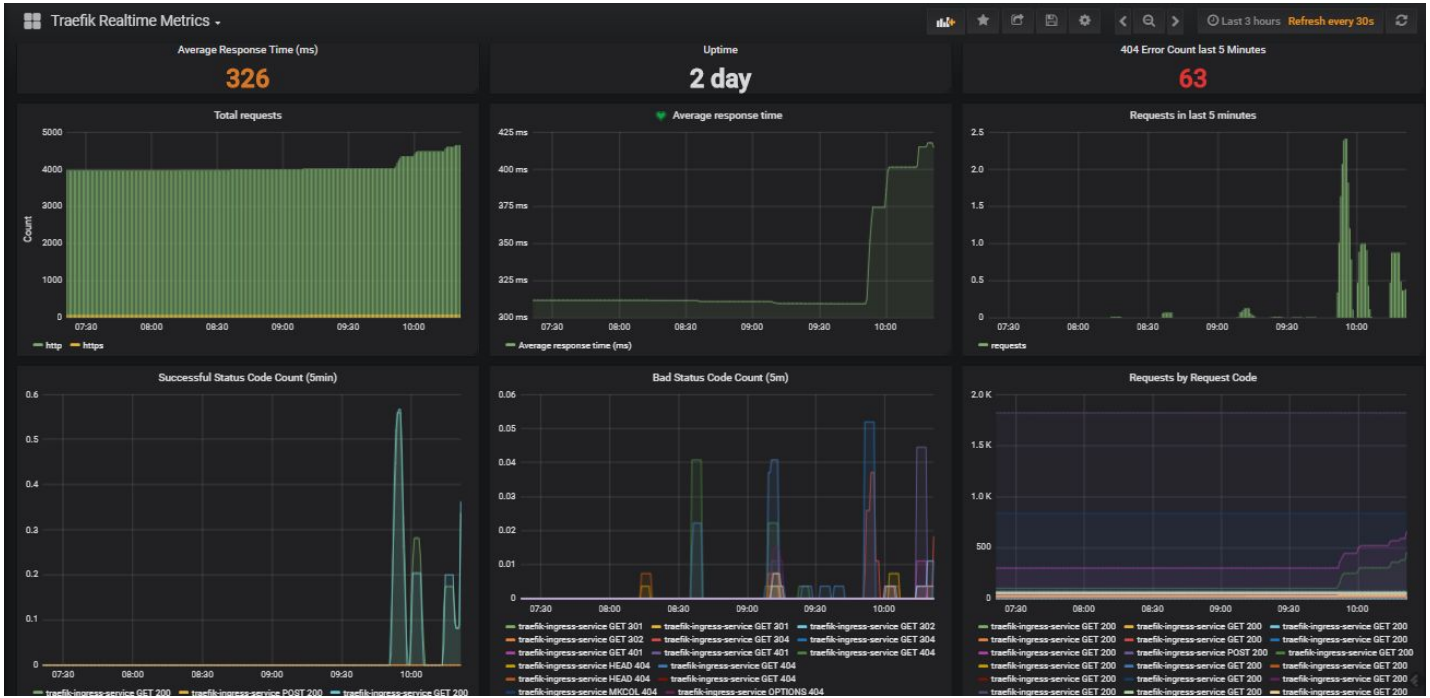
# Service Mesh



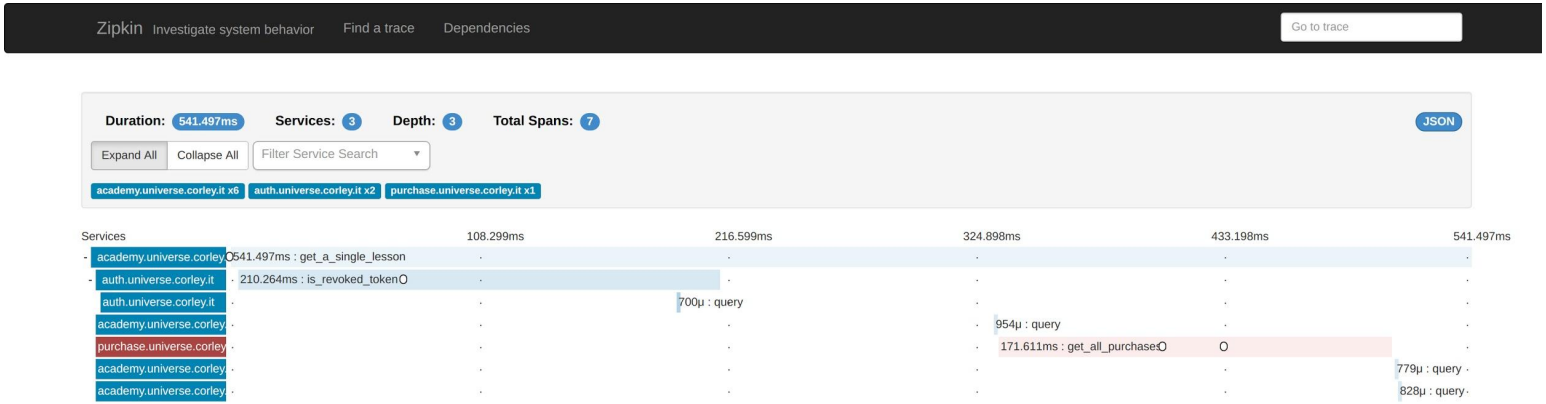
# Authentication in a Service Mesh



# Every proxy sends data to the monitor (example)



# So all data traces



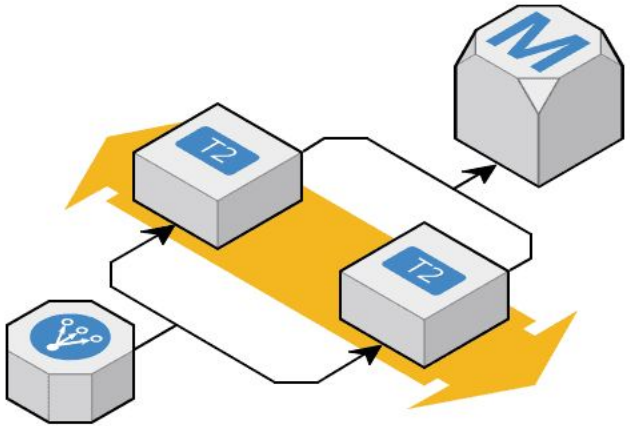
# Microservices

Now we can focus on a real microservices and not a little monolith

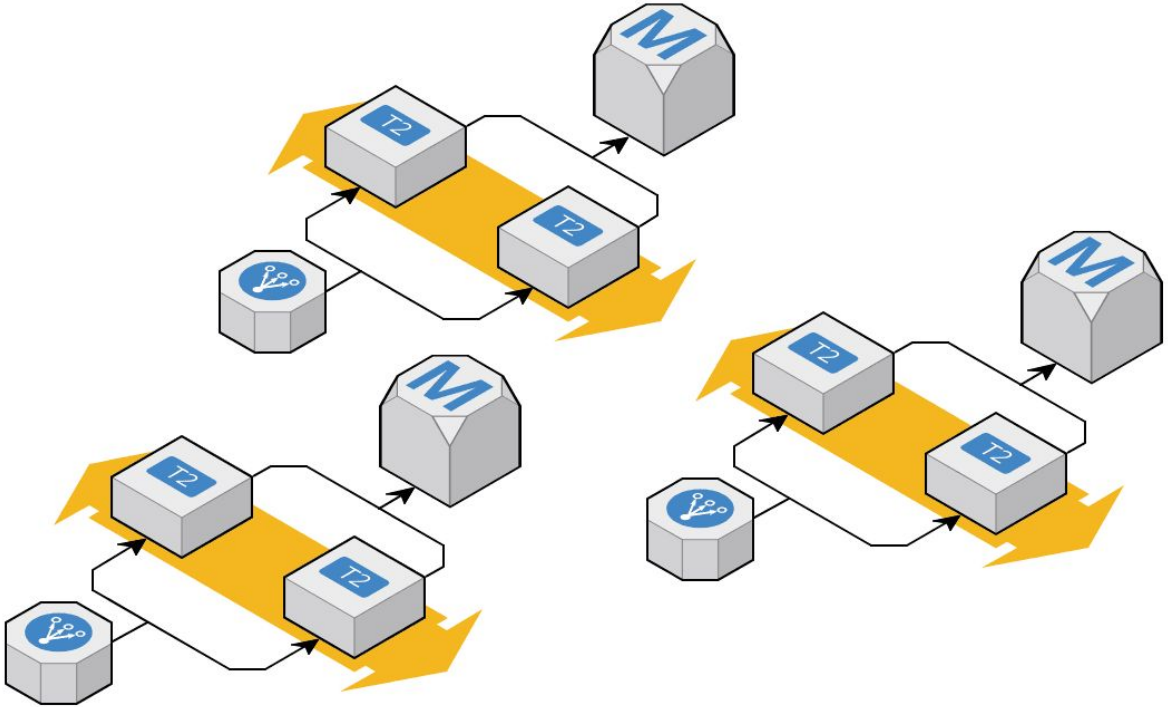
Few dependencies, less maintenance problem, can be replaced easily, etc...

# Delivery so many microservices?

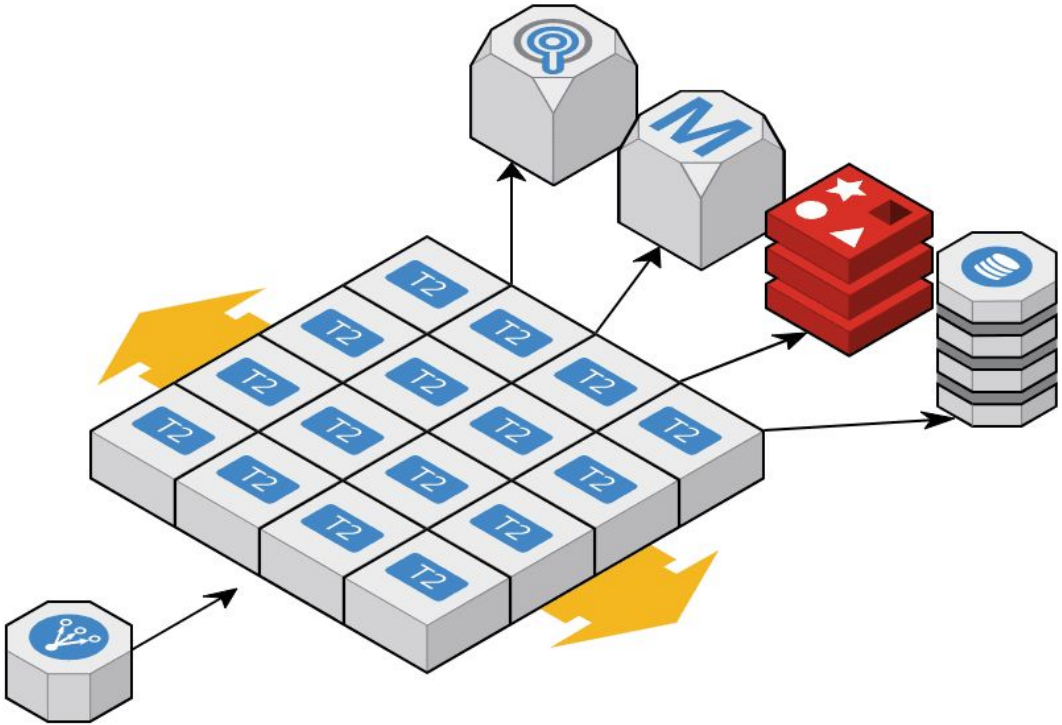
# Single Service



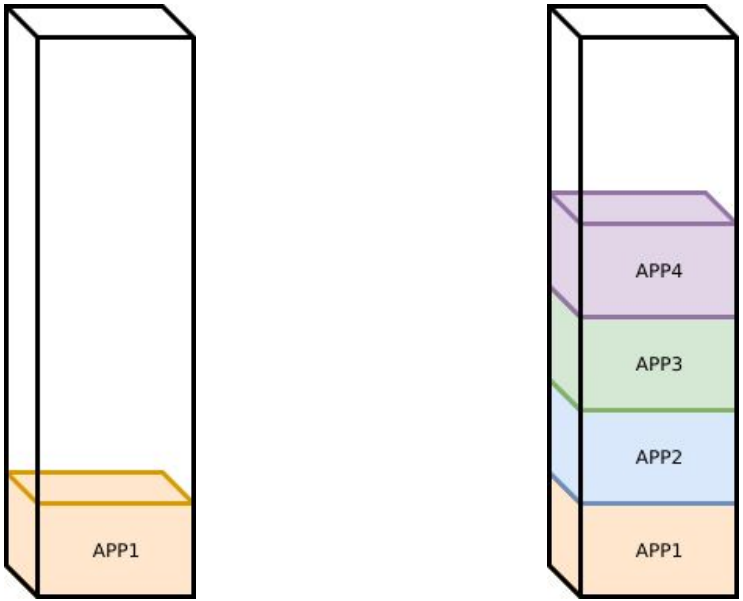
# Create a lot of tiny services (microservices)



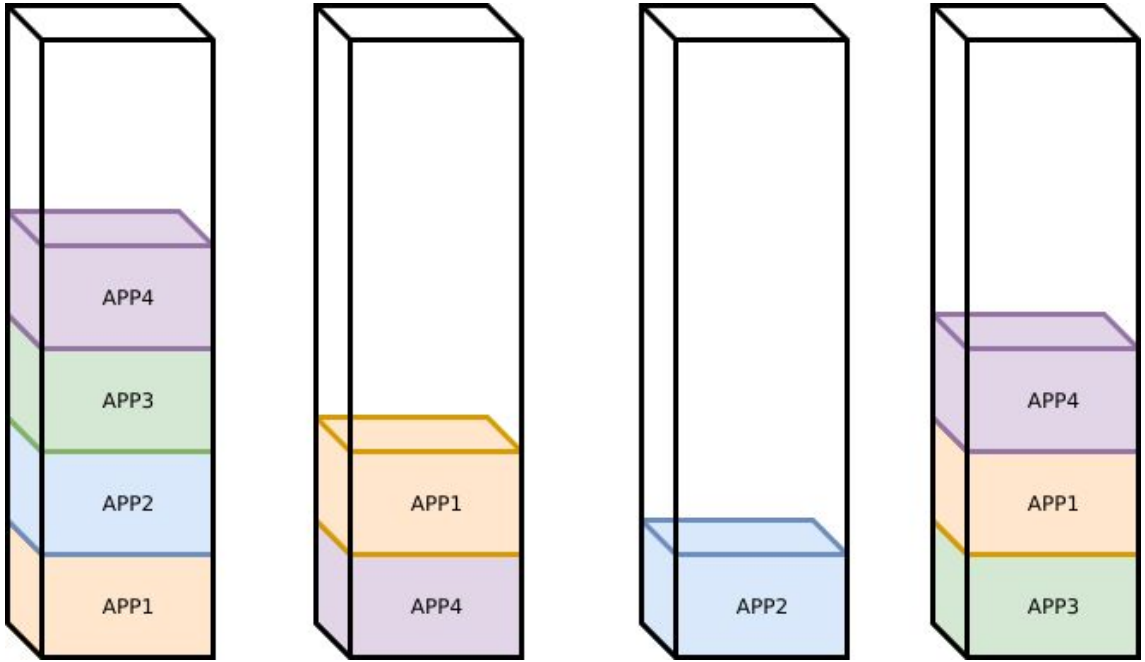
# Cluster for microservices (Kubernetes, etc.)



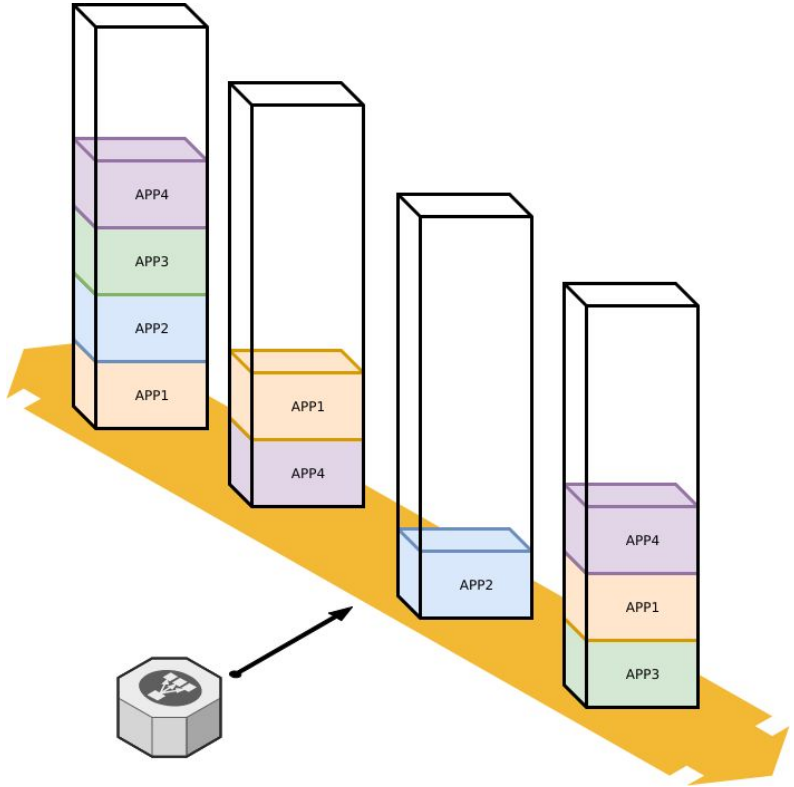
# Containers for applications



# Containers for applications (Cluster)



# Balanced Cluster



# Thanks for listening

corley