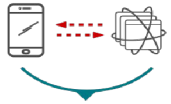
APPLICATION MODERNIZATION WITH CONTAINER READY MIDDLEWARE

Eran Mansour
Senior JBoss Consultant
Matrix
eranman@matrix.co.il

IT Must Evolve to Stay Ahead of These Trends

Application Architecture

Monolithic

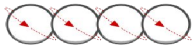


N-Tier

Microservices

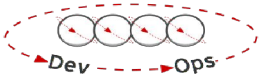
Development Process

Waterfall



Agile

DevOps



Application Infrastructure

Datacenter

Hosted

Cloud

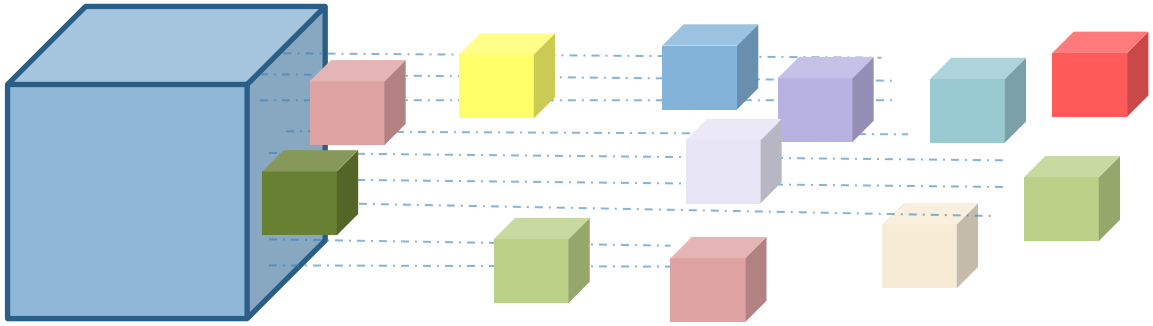
Modern Applications

Microservices are an expression of modern application

Microservices:
Architectural style that structures an application as a collection of loosely coupled services.

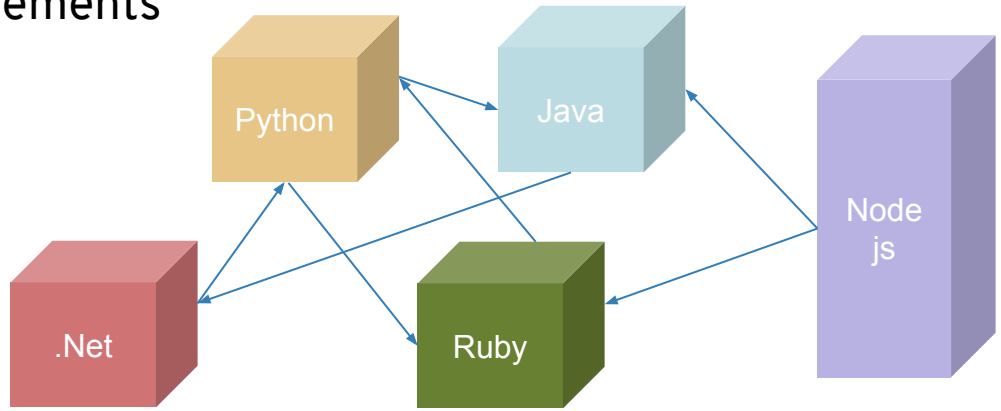
Microservices advantages

Moving from Monolithic to Microservice can allowing us utilize the new infrastructure in a way that couldn't be in the past



Microservices advantages

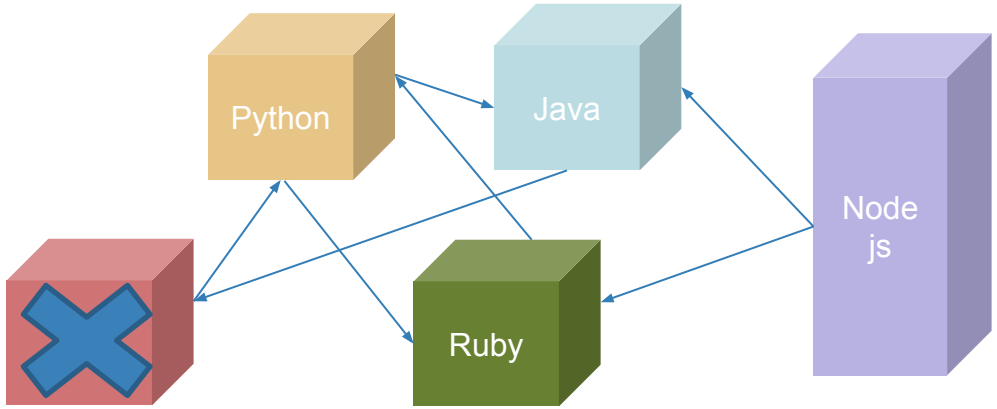
Various Technologies:
Each service is written in the most suitable technology to implements its requirements



Microservices advantages

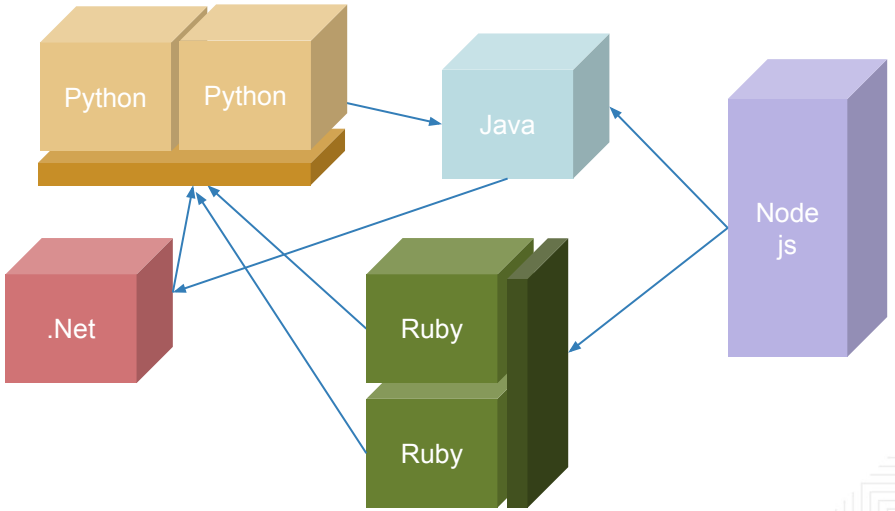
Various Technologies:

Microservice failure shall impact only on part of the application



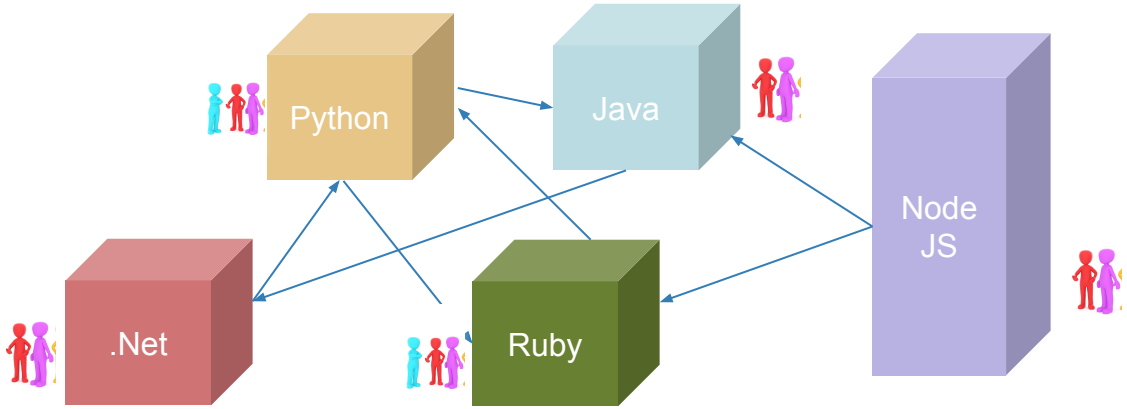
Microservices advantages

Scale only what you need to scale



Microservices advantages

Developers focus on one microservice with small team, updates and new versions are easier



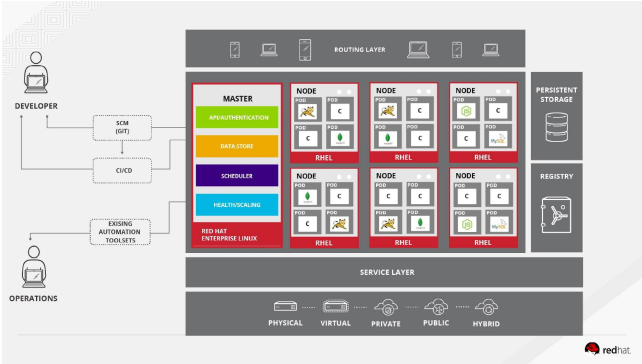
Microservices Ecosystem

With

Red-Hat Openshift

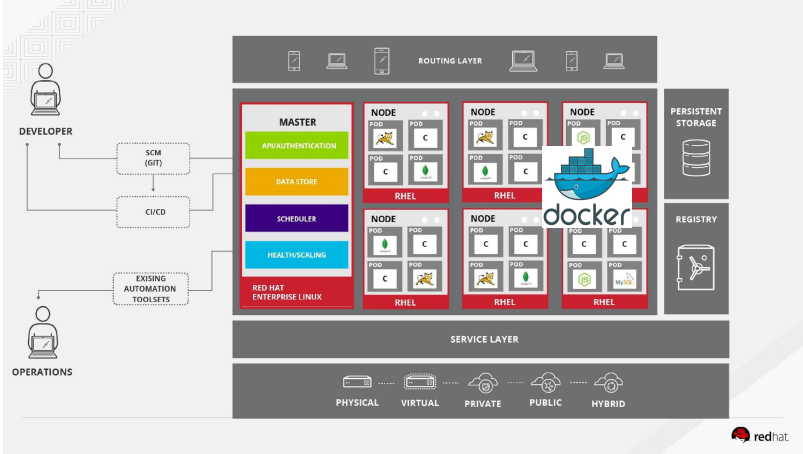
Managing Microservices with Openshift

- Microservices required supporting infrastructure which simplifies the creation and running of Microservices
- Openshift as a Pass technology functions as an echosystem for running Microservices



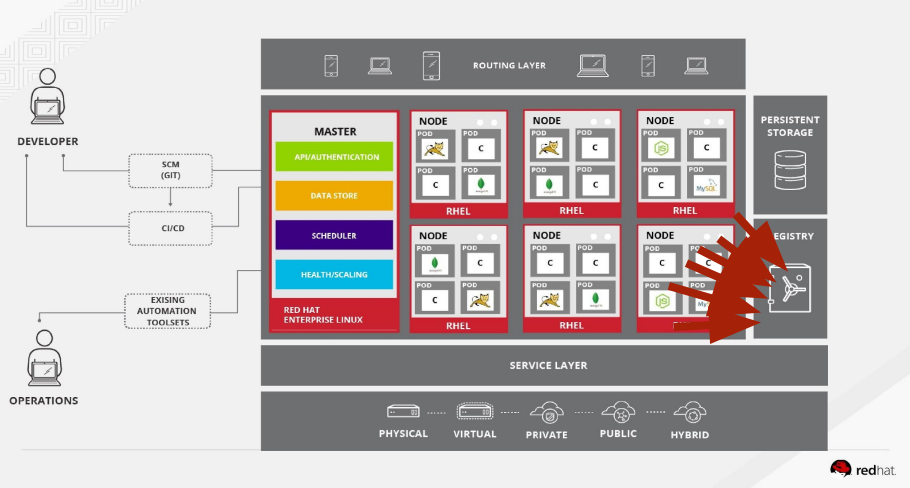
Managing Microservices with Openshift

- Each service is packed in a Docker Image
- Dockerized images enables running your application services everywhere



Managing Microservices with Openshift

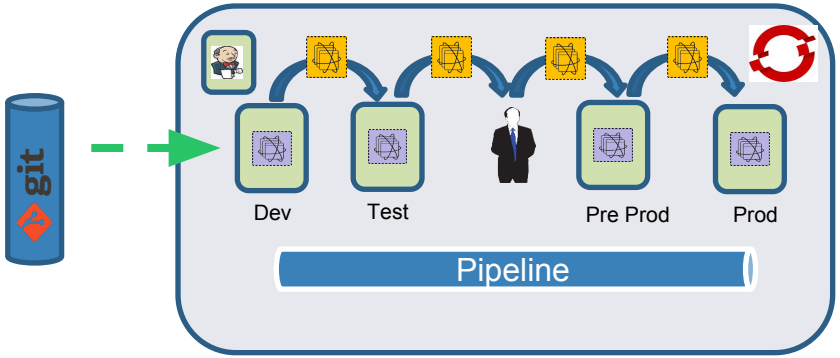
- All Docker images are stored in a registry

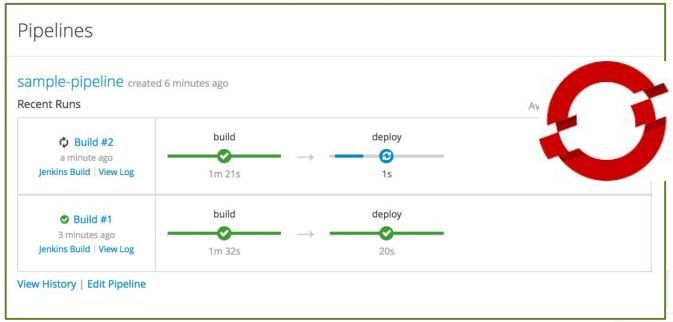


Managing Microservices with Openshift

CI/CD (Continues Integration & Continues Deployment)

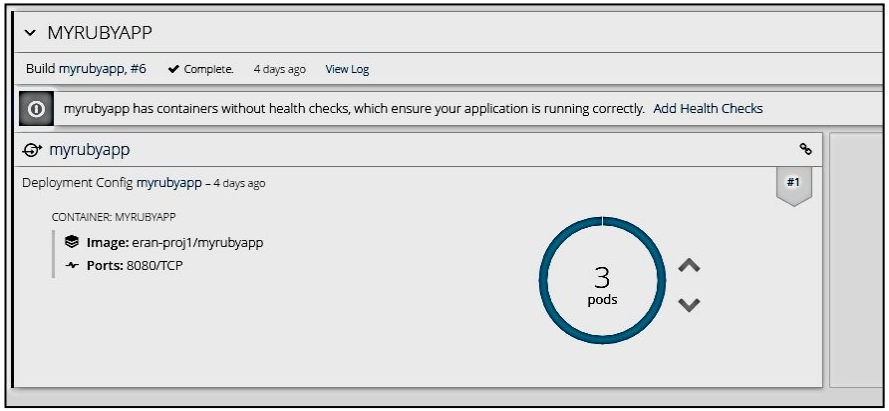
- Short development cycles
- Microservices shouldn't managed manually





Managing Microservices with Openshift

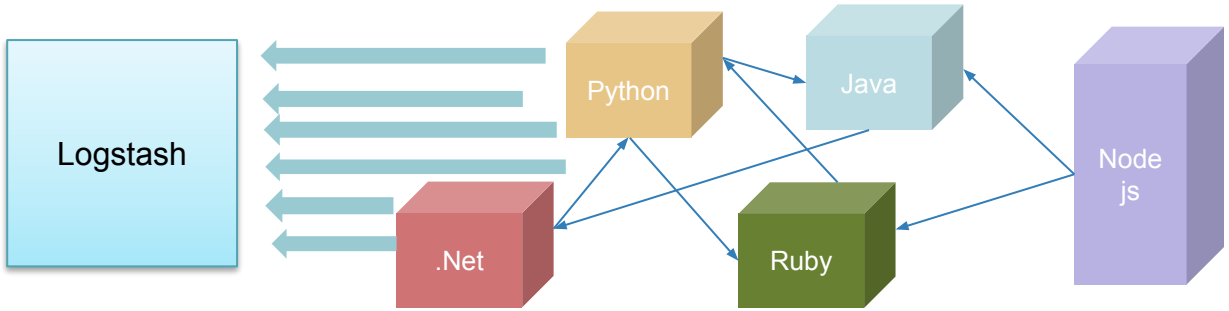
Scaling – the system should allow scaling of your services,
The scale is done on a single microservice



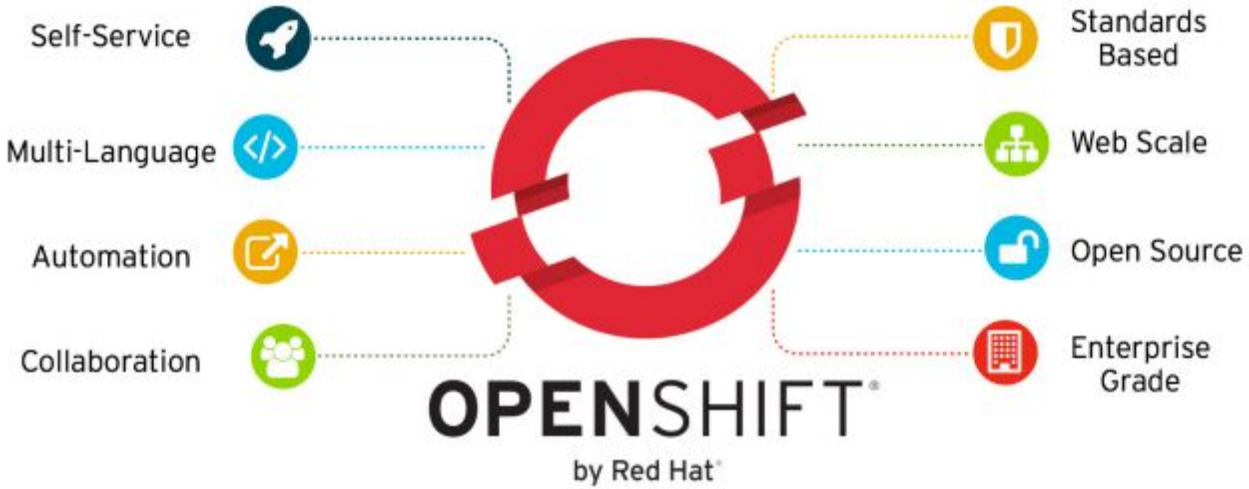
Managing Microservices with Openshift

Logging – Microservices logs are sent into ELK

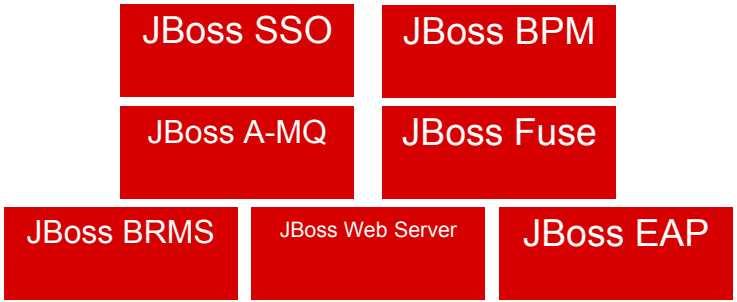
- Logs should be accessible even after microservice instance tears down



Managing Microservices with Openshift

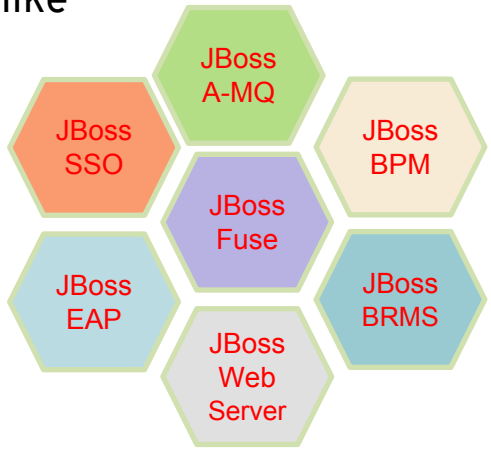


Using Microservices with Red-Hat Middleware products



Red-Hat XPASS middleware

- Red-Hat fits its middleware products in order to functioned as Microservices
- Microservices can be consist of MW product like BPM, Rule –Engine and Messaging Service
- All Products are being packed as a Docker image

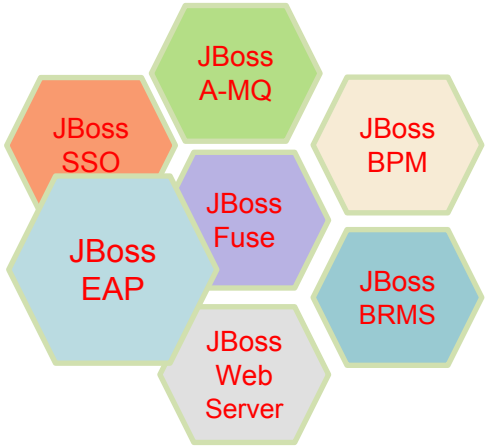


Red-Hat XPASS middleware

Red-Hat fits its middleware products in order to functioned as Microservices

JBoss EAP

- Wildfly Swarm

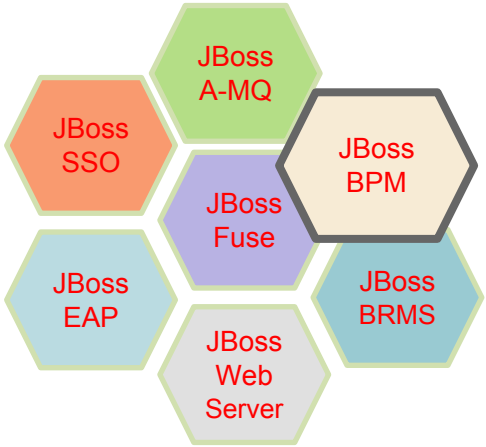


Red-Hat XPASS middleware

Red-Hat fits its middleware products in order to functioned as Microservices

Intelligent Process Server

- Runtime Engine
- Restfull API

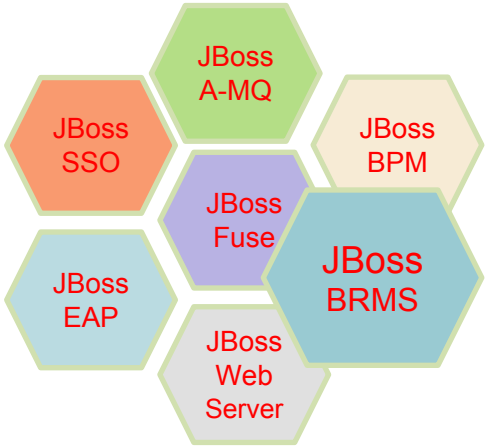


Red-Hat XPASS middleware

Red-Hat fits its middleware products in order to functioned as Microservices

Intelligent Process Server

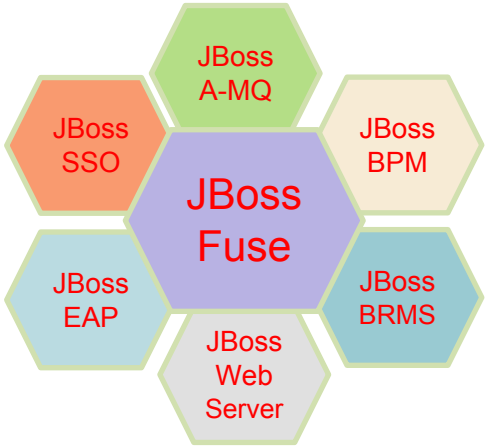
- Runtime Engine
- Restfull API
- Only stateless scenarios



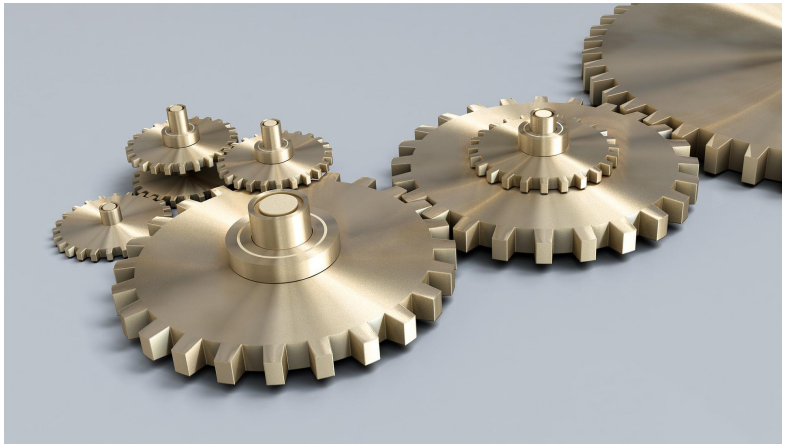
Red-Hat XPASS middleware

Red-Hat fits its middleware products in order to functioned as Microservices

- Fuse Integration Services (FIS)
- Camel implementation only



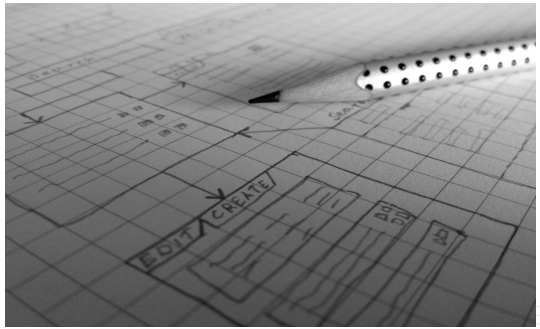
Microservice Design



Microservice Design

When building Microservices it’s important to adopt design concepts

Good design:
Avoid problems which can be in the future



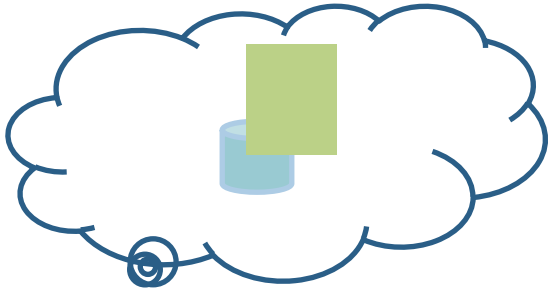
Conway's Law

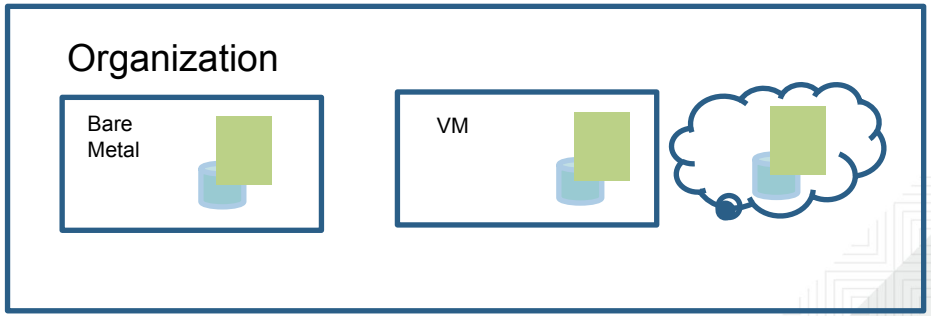
Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.

Microservice Design

Stateless

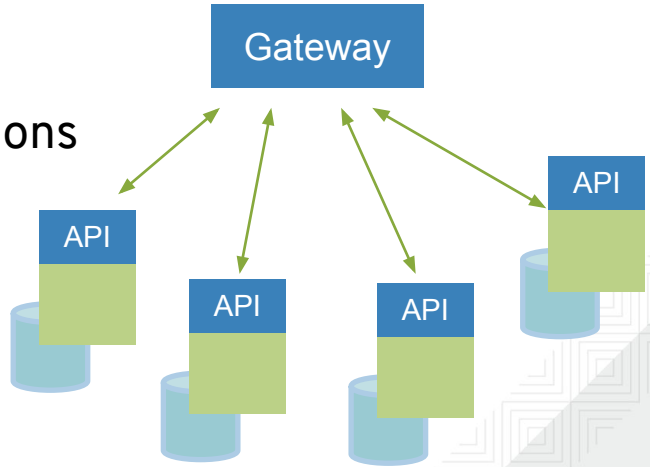
- Microservices should be stateless,
- You should not know where and when your microservice will be set up or tear down,





Microservice Design

- Separate between your business logic and managing your API
- Take care of managing distributed transactions
- Each microservice should deal with its own DB,

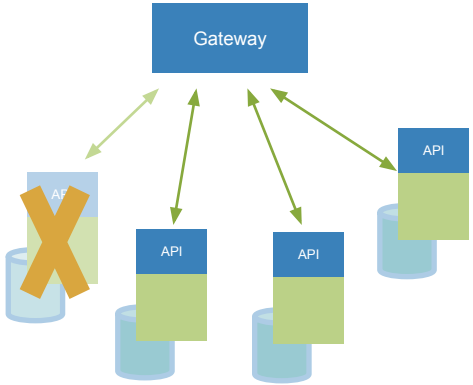


Microservice Design

Resilience

Monolithic systems have more risks to be paralyzed when failure occurs

Plan your Microservices that failure in one microservice doesn't cascade



Application Modernization Summary

Microservices:

- Are implementation of modern applications
- Its architecture has advantages over the monolithic one
- Requires an echo system like Openshift which supports creation and running Microservices
- Architecture has its own characteristics and therefore its should be designed well

redhat®

Thank You

Eran Mansour
Senior JBoss Consultant
Matrix
eranman@matrix.co.il