Microservice and Docker

Prepared for

Faculty of Mathematics and Informatics (FMI)

Building Microservices

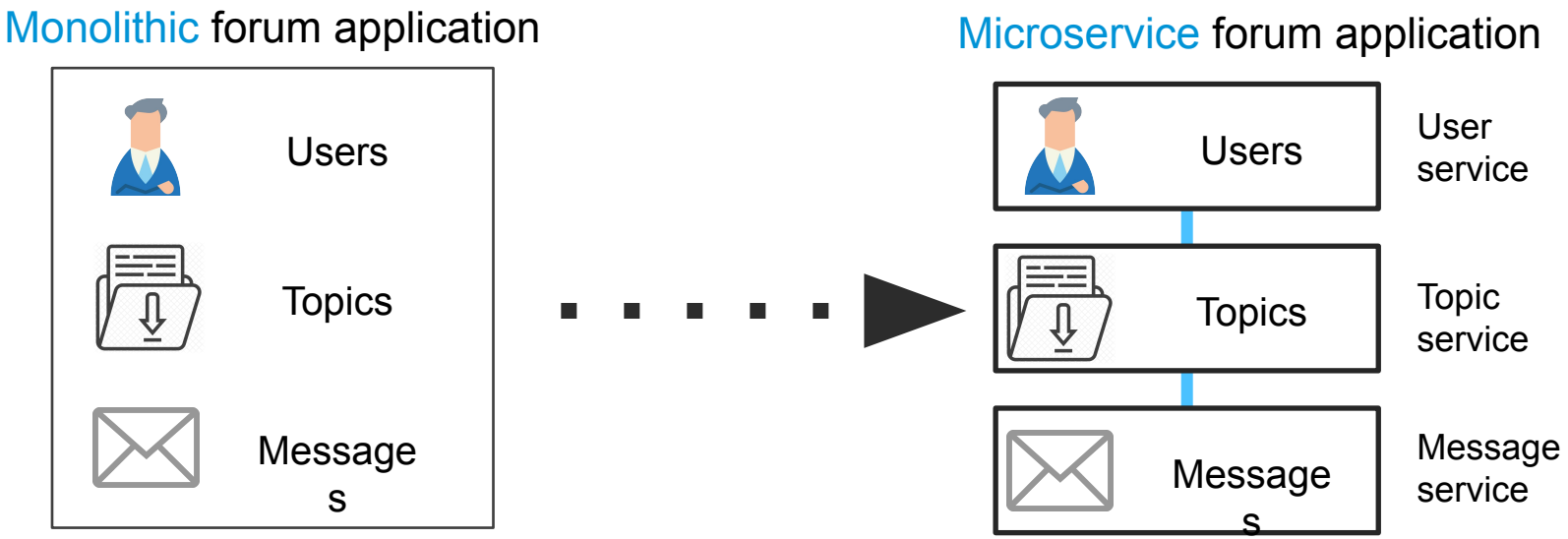
What are microservices?

Applications composed of independent services that communicate over well-defined APIs

© 2021 Daniel Rankov. All rights reserved.

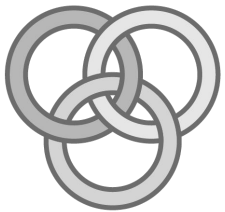
What are microservices?

Applications composed of independent services that communicate over well-defined APIs

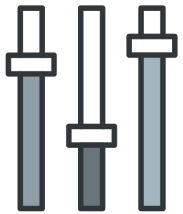


© 2021 Daniel Rankov. All rights reserved.

Why use microservices?



Your system can withstand individual failures



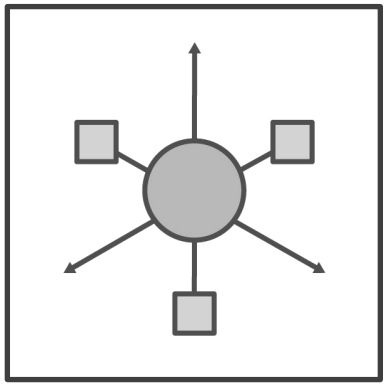
Allows parallel production and faster iteration

Spend on scaling only where needed

© 2021 Daniel Rankov. All rights reserved.

Characteristics of a microservice

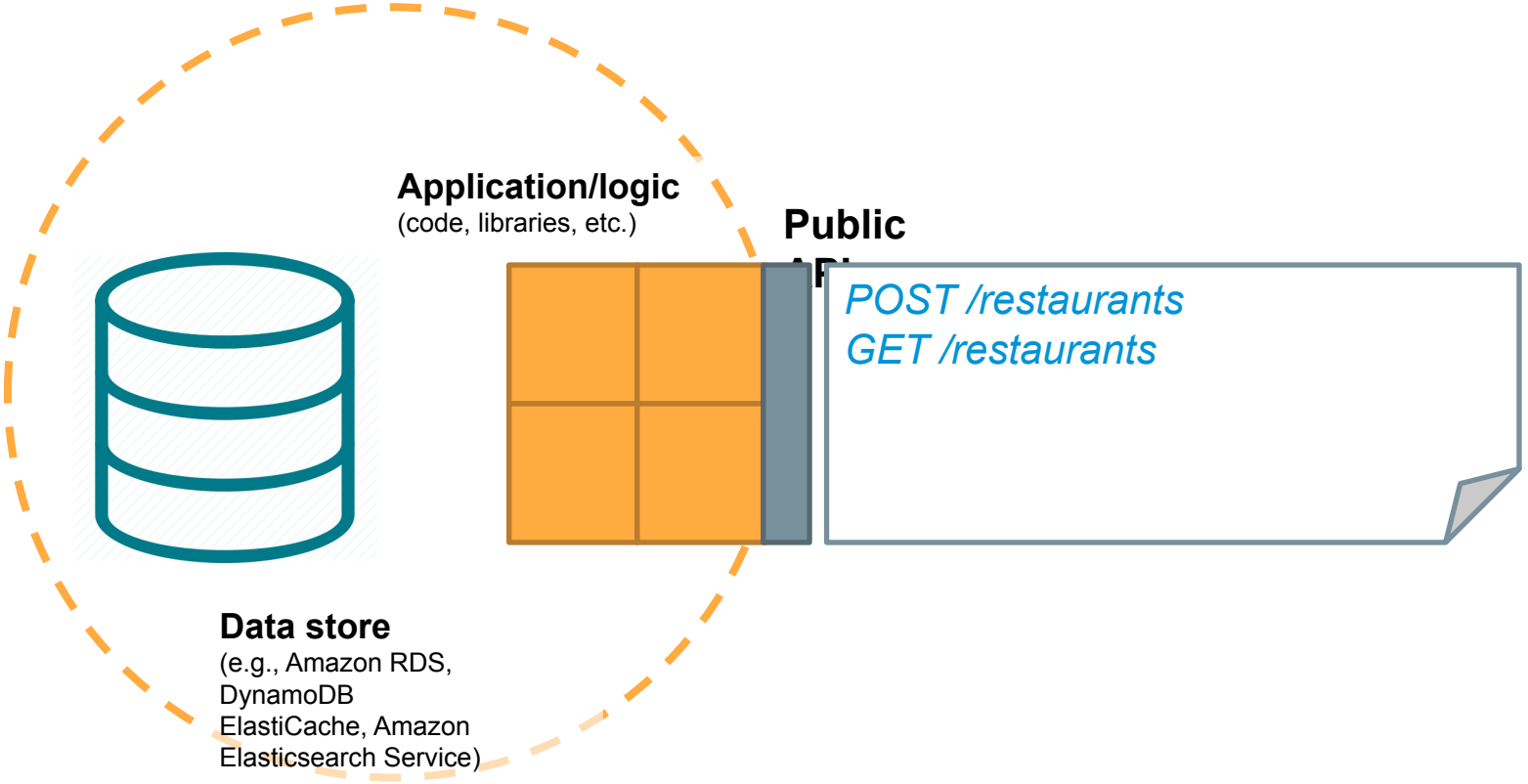
Autonomous



Specialized

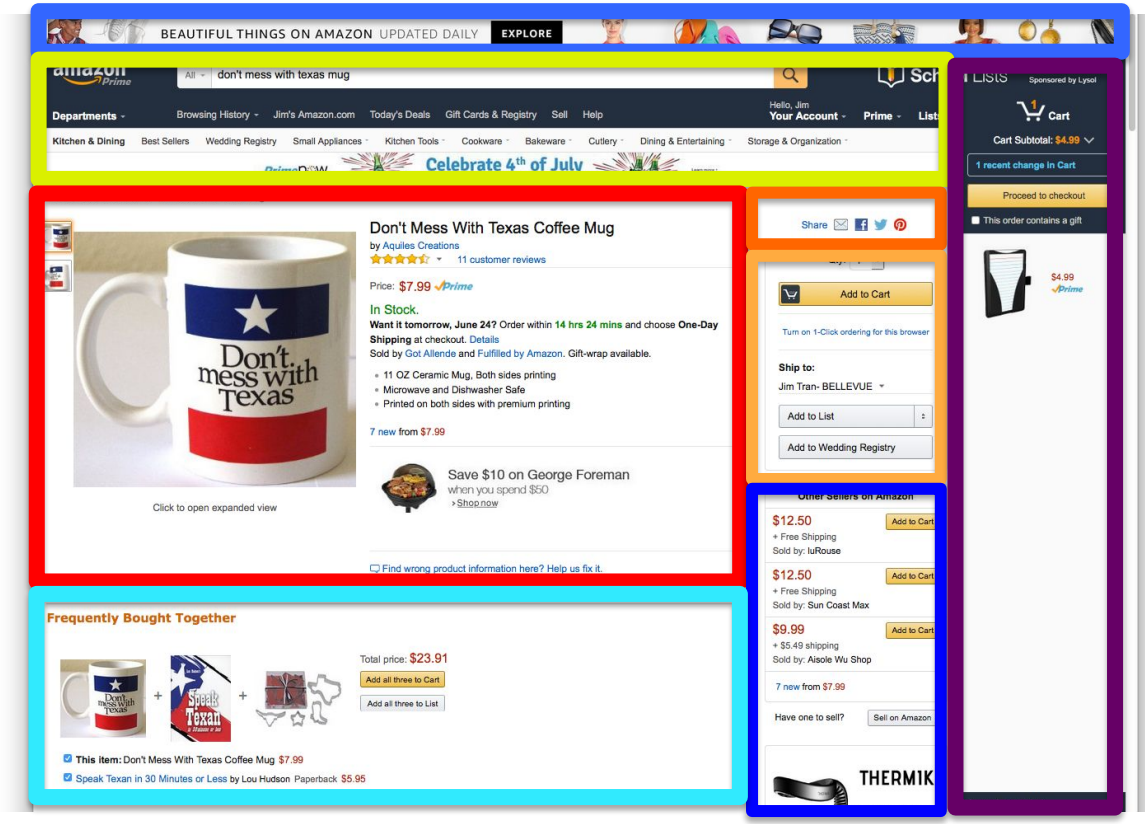
© 2021 Daniel Rankov. All rights reserved.

Anatomy of a microservice



© 2021 Daniel Rankov. All rights reserved.

Microservices at Amazon



© 2021 Daniel Rankov. All rights reserved.

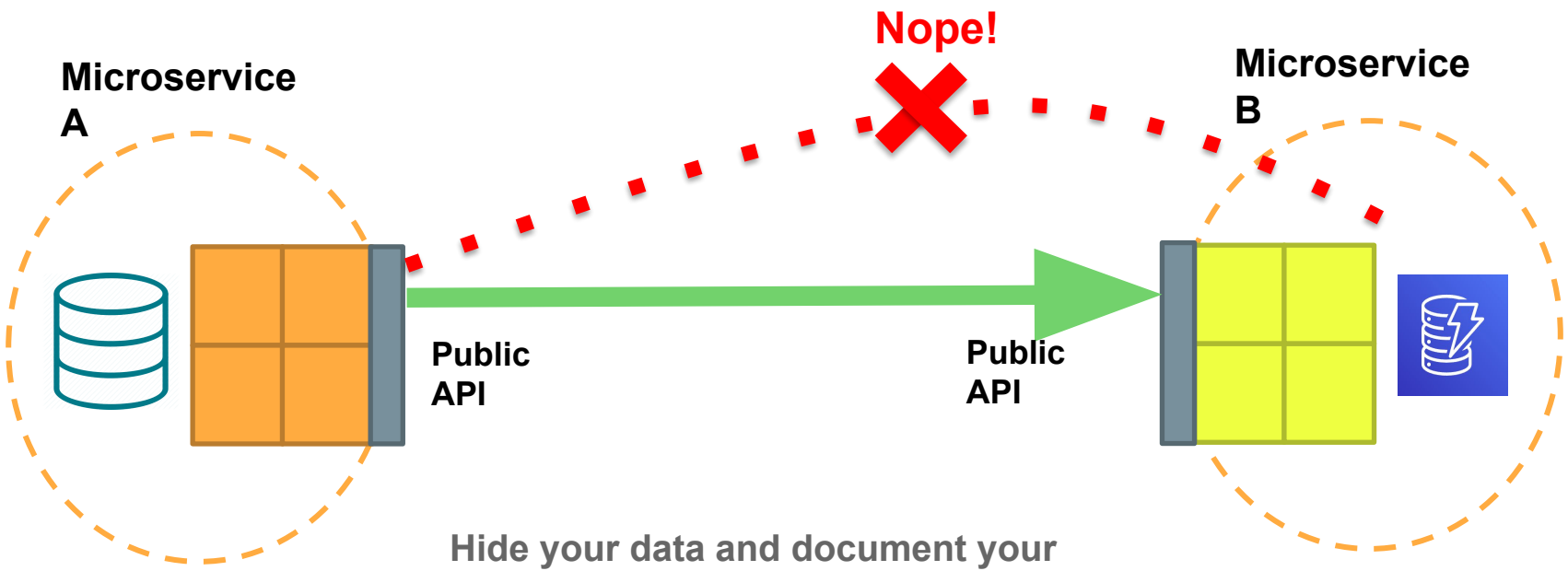
Six Principles of Microservices

Six principles of microservices

1. Rely only on the public API.
2. Use the right tool for the job.
3. Secure your services.
4. Be a good citizen within the ecosystem.
5. It's more than just technology transformation.
6. Automate everything.

© 2021 Daniel Rankov. All rights reserved.

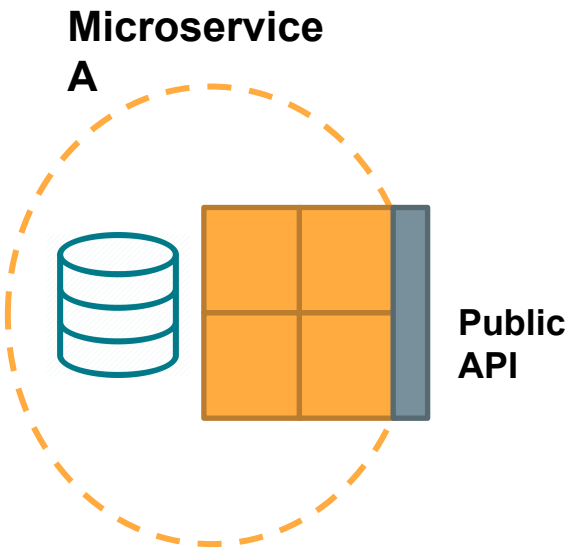
Principle #1: Rely only on the public API



Hide your data and document your APIs

© 2021 Daniel Rankov. All rights reserved.

Principle #1: Rely only on the public API



Define a versioning strategy

Version 1.0.0

storeRestaurant (*id, name, cuisine*)

Version 1.1.0

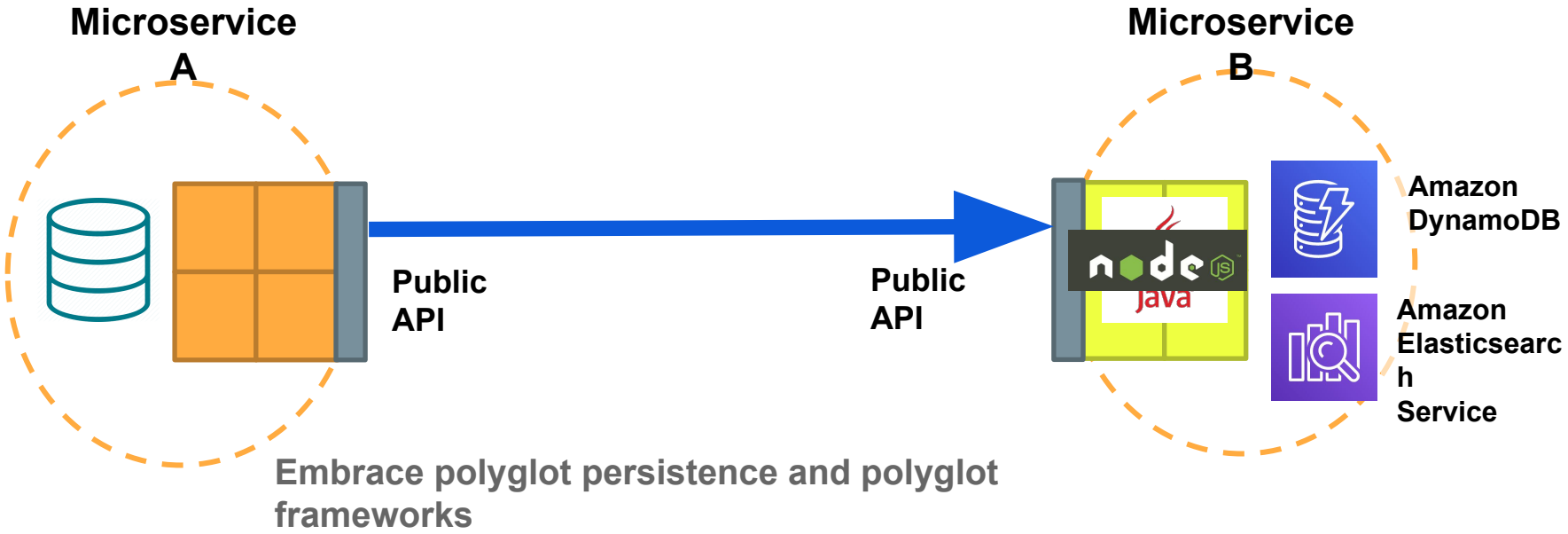
storeRestaurant (*id, name, cuisine*)
storeRestaurant (*id, name, arbitrary_metadata*)
addReview (*restaurantId, rating, comments*)

Version 2.0.0

storeRestaurant (*id, name, arbitrary_metadata*)
addReview (*restaurantId, rating, comments*)

© 2021 Daniel Rankov. All rights reserved.

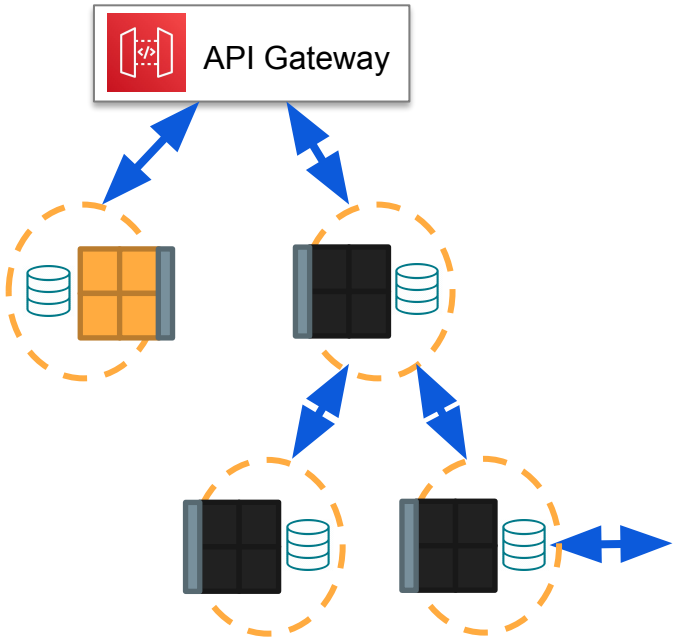
Principle #2:



Embrace polyglot persistence and polyglot frameworks

© 2021 Daniel Rankov. All rights reserved.

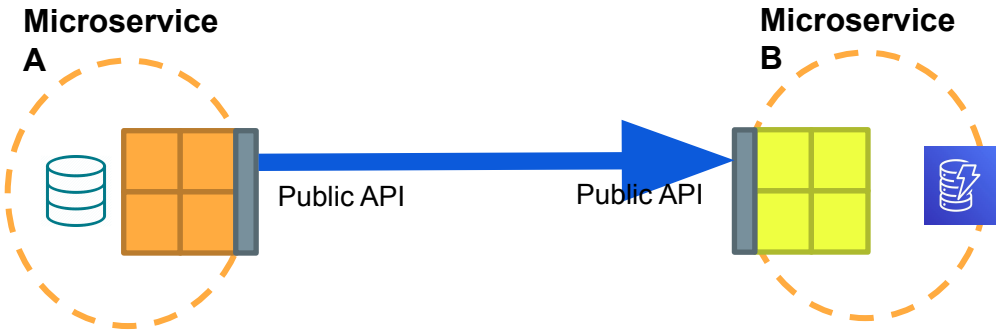
Principle #3: Secure your services



- Defense in depth
- Gateway (“front door”)
- API throttling
- Authentication and authorization
- Secrets management

© 2021 Daniel Rankov. All rights reserved.

Principle #4: Be a good citizen

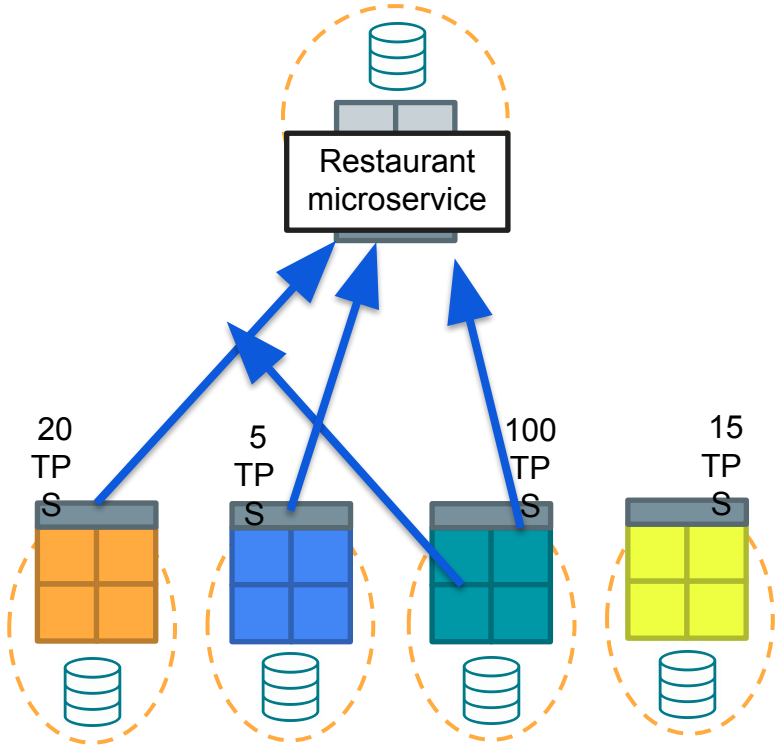


We need to call your microservice to fetch restaurant details.

Which APIs do you need to call? Once I understand your use cases, I'll give you permission to register your service as a client on our service's directory entry.

© 2021 Daniel Rankov. All rights reserved.

Principle #4: Be a good citizen



Before we let you call our microservice, we need to understand your use case, expected load (TPS), and accepted latency.

Have clear SLAs

© 2021 Daniel Rankov. All rights reserved.

Principle #4: Be a good citizen

Distributed monitoring and tracing

- "Is the service meeting its SLA?”
- "Which services were involved in a request?”
- "How did downstream dependencies perform?”

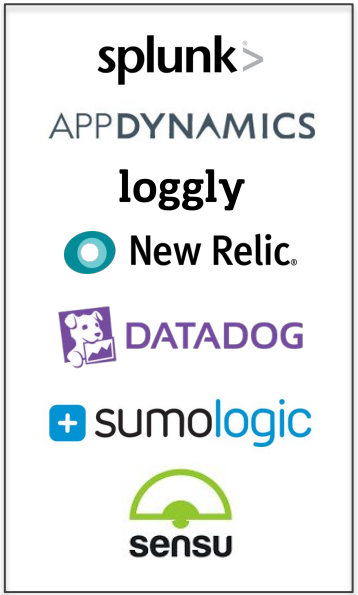
Shared metrics

- For example, request time, time to first byte

Distributed tracing

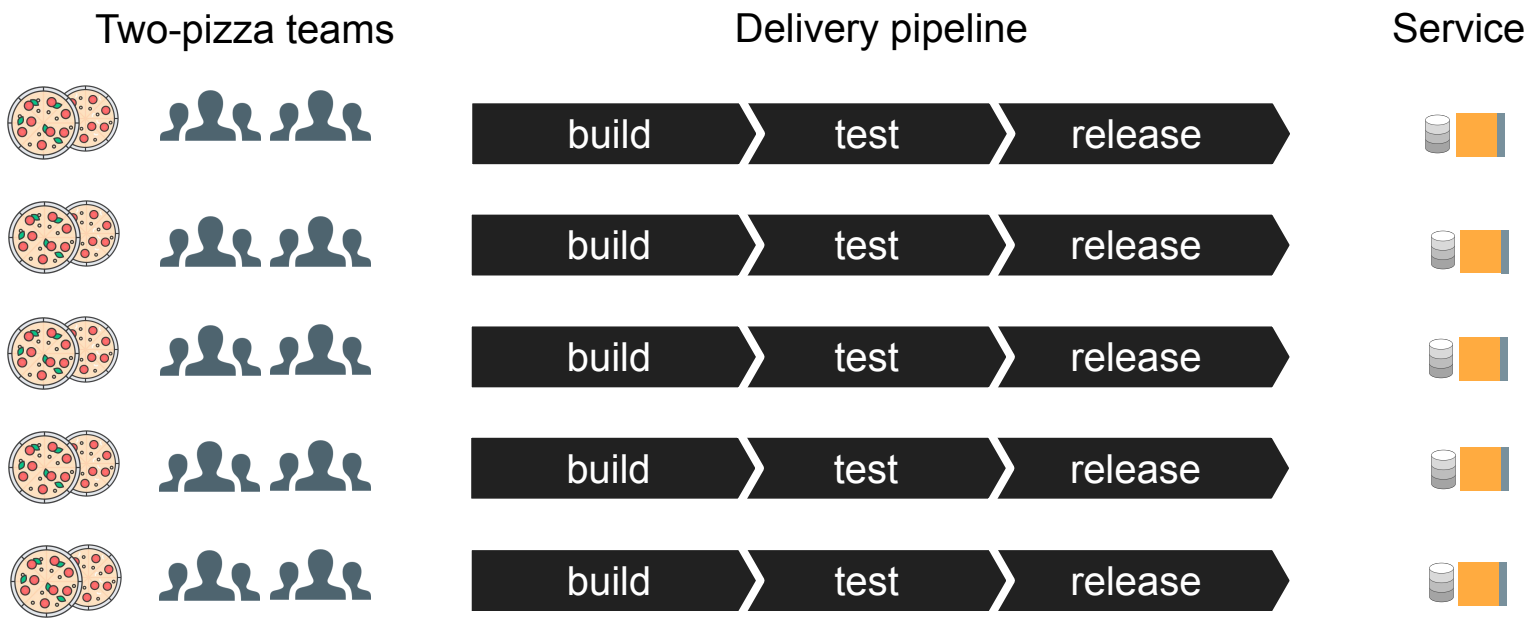
- For example, Zipkin, OpenTracing

User-experience metrics



© 2021 Daniel Rankov. All rights reserved.

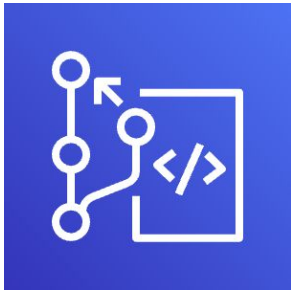
Principle #5:



Favor small, focused dev teams

© 2021 Daniel Rankov. All rights reserved.

Principle #6: Automate everything



AWS
CodeCommit

AWS
CodePipeline

AWS
CodeDeploy

Adopt
DevOps

© 2021 Daniel Rankov. All rights reserved.

Six principles of microservices

1. Rely only on the public API
   - Hide your data
   - Document your APIs
   - Define a versioning strategy
2. Use the right tool for the job
   - Polyglot persistence (data layer)
   - Polyglot frameworks (app layer)
3. Secure your services
   - Defense in depth
   - Authentication/authorization
4. Be a good citizen
   - Have SLAs
   - Distributed monitoring, logging, tracing
5. It's more than just technology transformation
   - Favor small, focused dev teams
6. Automate everything
   - Adopt DevOps

© 2021 Daniel Rankov. All rights reserved.

Using Container Services

What is a container?

Your container

Your application

Dependencies

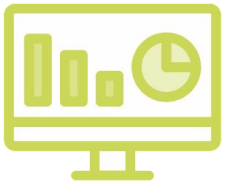
Configurations

Hooks into OS

© 2021 Daniel Rankov. All rights reserved.

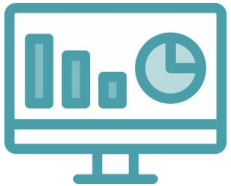
What problems can containers solve?

Getting software to run reliably in different environments



Developer's workstation

Production



Test environment

© 2021 Daniel Rankov. All rights reserved.

Container benefits

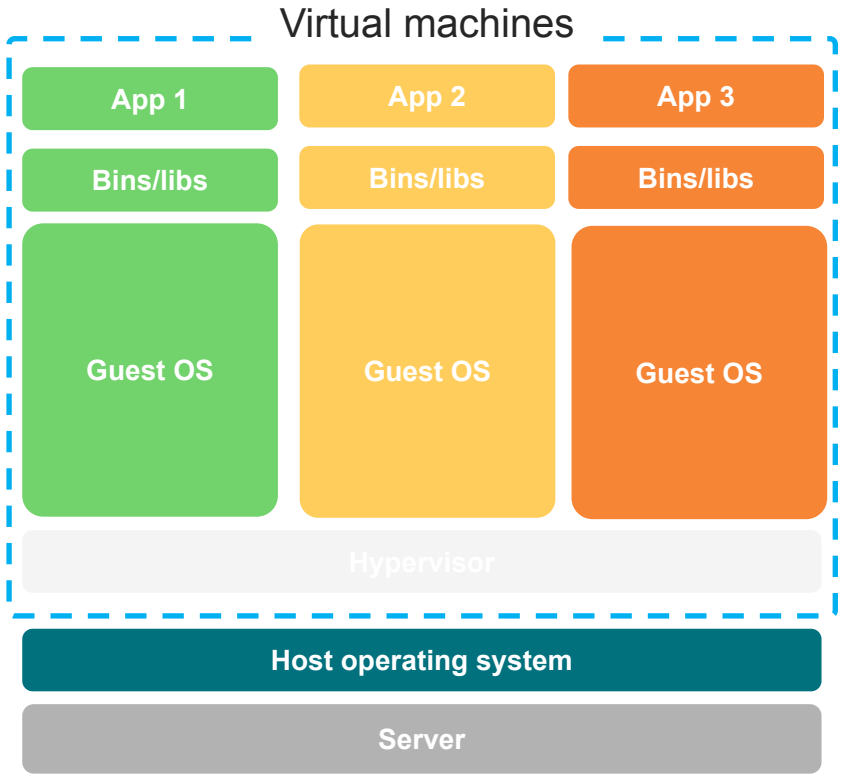
Repeatable

Self-contained running environments

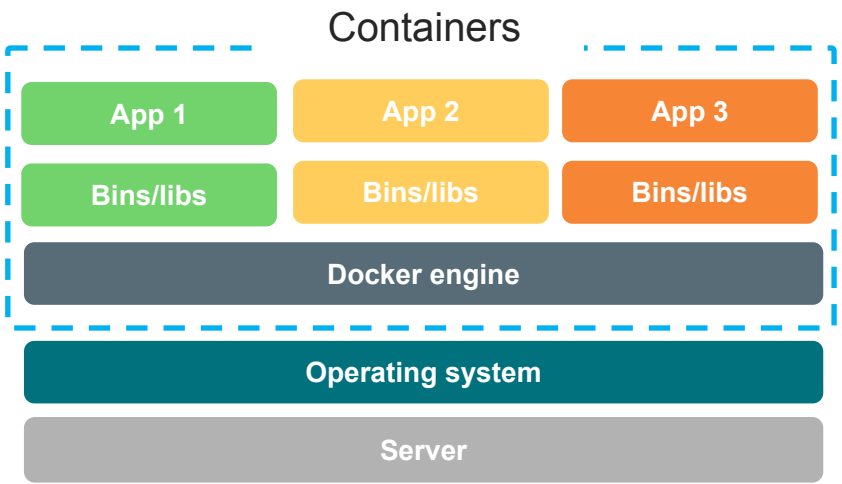
Faster to wind up and down than VMs

© 2021 Daniel Rankov. All rights reserved.

Virtual machines vs. containers



In general: VMs abstract hardware, while containers abstract operating systems



© 2021 Daniel Rankov. All rights reserved.

Docker history

© 2021 Daniel Rankov. All rights reserved.

Docker get started

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

- Lightweight
- Isolated
- Self-sufficient
- Layered filesystem
- Mount local files

© 2021 Daniel Rankov. All rights reserved.

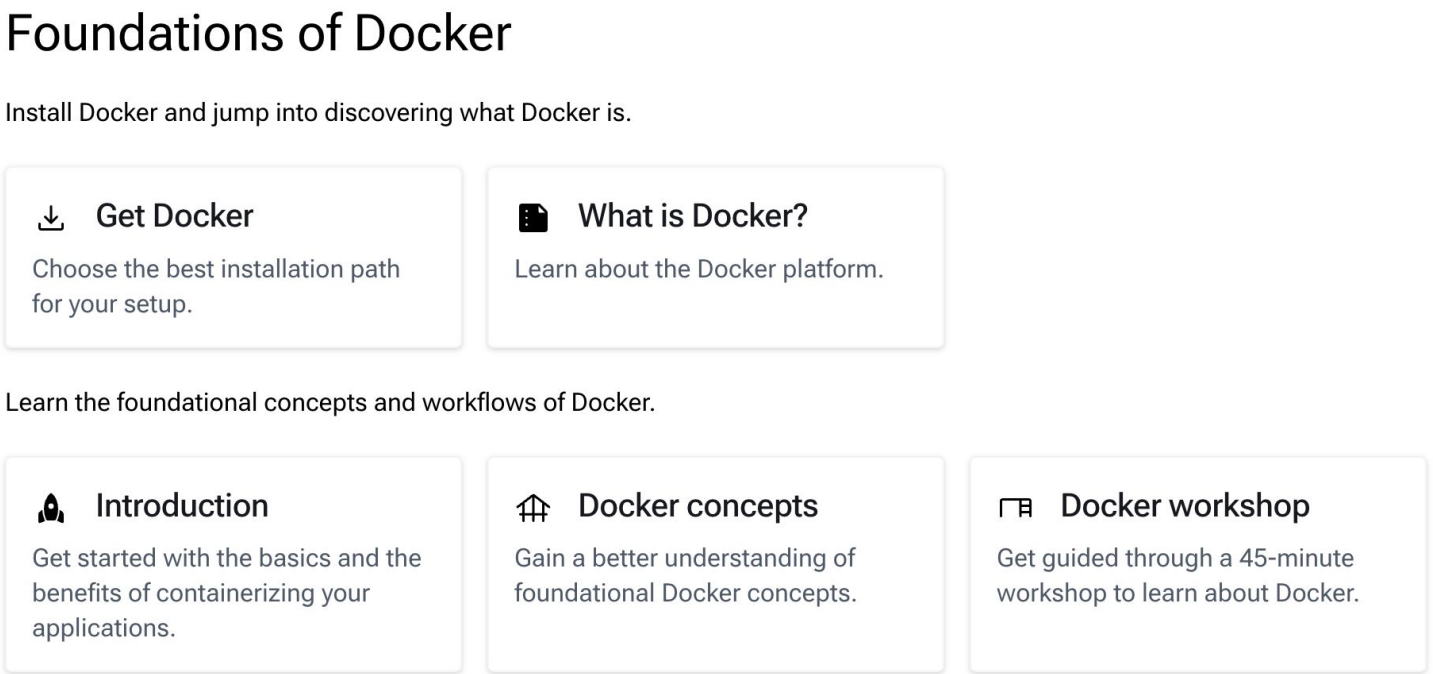
Docker repository

- How to save the artifacts
- DockerHub
- https://gallery.ecr.aws/
- https://ghcr.io
-

© 2021 Daniel Rankov. All rights reserved.

Docker Getting Started

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



© 2021 Daniel Rankov. All rights reserved.

Several Clouds

Thank you!

daniel@severalclouds.com

https://www.linkedin.com/in/danielrankov/