Containers Everywhere!

Using Docker from Development to Production

aboutbits.it alex.lanz@aboutbits.it @alex__lanz

About Me

- Alex Lanz - About Bits
- Software Development
- Consulting/Training

Story 1: Deployment

- A company started developing an application/feature
- They tested it locally => Tests passed ✓
- They pushed the changes to the repository
- The CI pipeline tested the application => Tests passed ✓
- They deployed the application to the stating/production environment
- And …

it failed! ✗

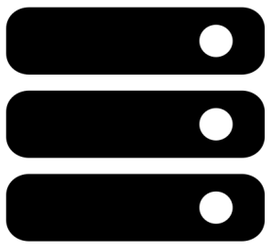
Story 1: Deployment

- „But it worked on my machine“
- Possible reasons:
  - Closed firewall
  - Other operating systems
  - Other software dependencies
  - Wrong configuration
- In other words:

There were differences in the environments

Story 2: Microservices

Monolith Application
PHP 7.1



aboutbits.it alex.lanz@aboutbits.it @alex__lanz

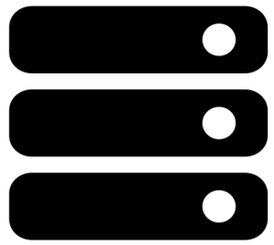
Story 2: Microservices

Microservice 1
PHP 7.1

Microservice 2
PHP 7.1

Microservice 3
PHP 7.1

Microservice 4
PHP 7.1



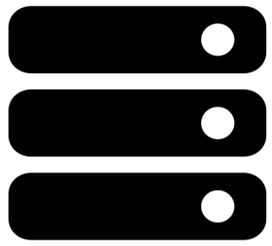
Story 2: Microservices

Microservice 1
PHP 7.1

Microservice 2
PHP 7.1

Microservice 3
PHP 7.1

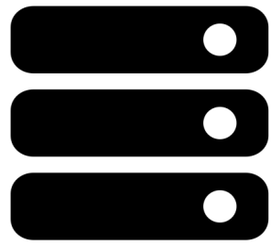
Microservice 4
PHP 7.1



Microservice 5
Python 3.6

Microservice 6
Python 3.6

Microservice 7
Go 1.10



aboutbits.it alex.lanz@aboutbits.it @alex__lanz

Story 2: Microservices

Microservice 5
Python 3.6

Microservice 1
PHP 7.1

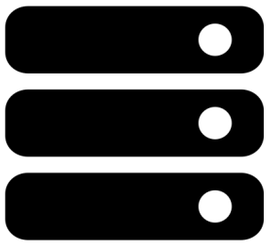
Microservice 3
PHP 7.1

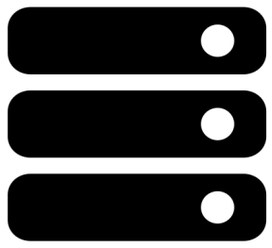
Microservice 6
Python 3.6

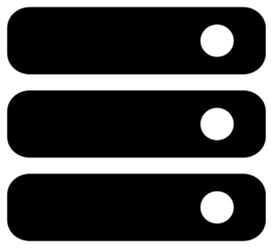
Microservice 2
PHP 7.1

Microservice 4
PHP 7.1

Microservice 7
Go 1.10







Story 2: Microservices

Microservice 5
Python 3.6

Microservice 1
PHP 7.1

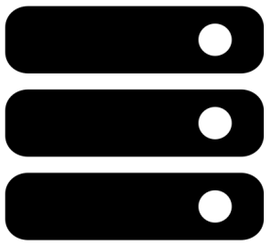
Microservice 3
PHP 7.1

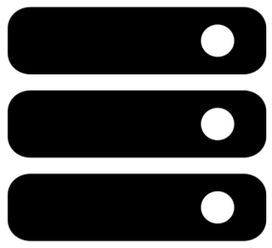
Microservice 6
Python 3.6

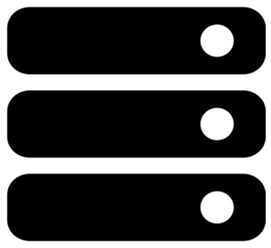
Microservice 2
PHP 7.1

Microservice 4
PHP 7.1

Microservice 7
Go 1.10







Story 2: Microservices

Microservice 5
Python 3.7

Microservice 1
PHP 7.2

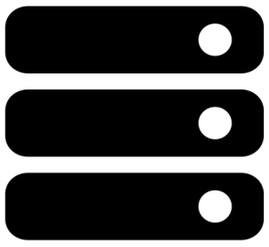
Microservice 3
PHP 7.1

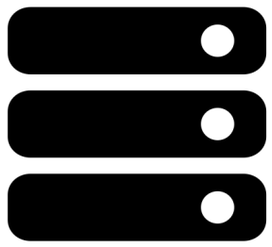
Microservice 6
Python 3.6

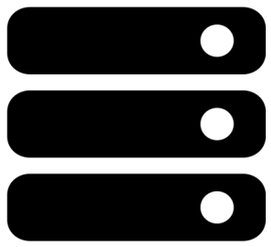
Microservice 2
PHP 7.1

Microservice 4
PHP 7.1

Microservice 7
Go 1.10







Story 2: Microservices

Microservice 5
Python 3.7

Microservice 1
PHP 7.2

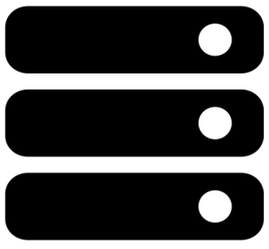
Microservice 3
PHP 7.1

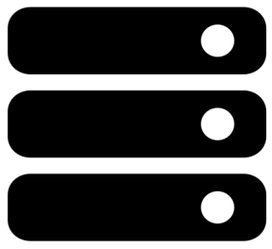
Microservice 6
Python 3.6

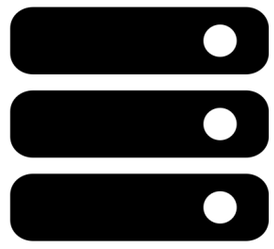
Microservice 2
PHP 7.1

Microservice 4
PHP 7.1

Microservice 7
Go 1.10







aboutbits.it alex.lanz@aboutbits.it @alex__lanz

Story 2: Microservices

- Questions:
  - Should we install two versions of a programming language on the same node?
  - Can we run two different programming languages on the same node?
  - How can we easily scale up one service if it receives more requests?
  - How can we reduce the setup costs?

Problems

- Keep **Setup/Maintenance** low
- Run applications in **Isolation**
- Provide easy **Scalability**
- Use the **Same Environment** everywhere
- **Development Speed** should be high

Solution

How can we solve these problems?

Docker

Docker

- Docker is a software platform for virtualized containers
- Containers are an abstraction at the app layer that packages code and dependencies together
- Multiple containers can run on the same machine
- Each running as isolated processes in user space
- Containers share the OS kernel with other containers

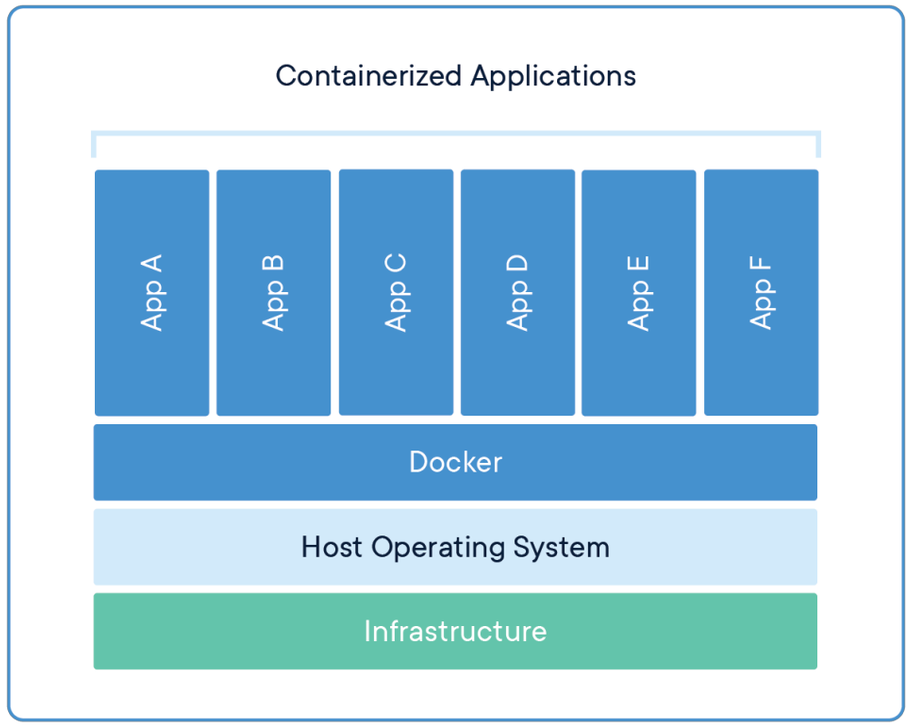


Image: https://www.docker.com/resources/what-container

Docker

Microservice 5
Python 3.7

Microservice 1
PHP 7.2

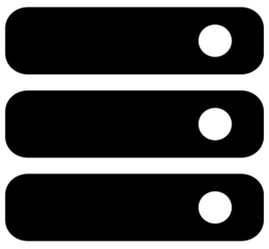
Microservice 3
PHP 7.1

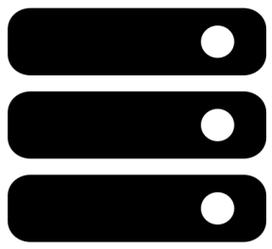
Microservice 6
Python 3.6

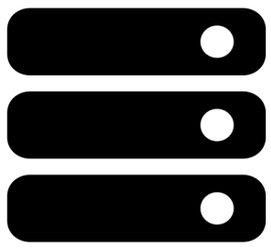
Microservice 2
PHP 7.1

Microservice 4
PHP 7.1

Microservice 7
Go 1.10

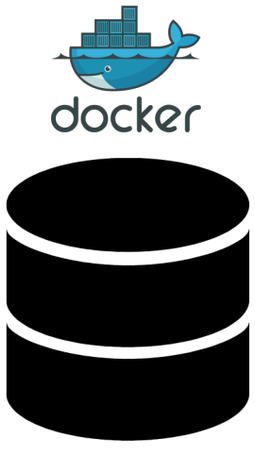






aboutbits.it alex.lanz@aboutbits.it @alex__lanz

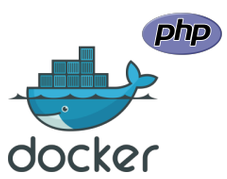
Docker



Docker Registry

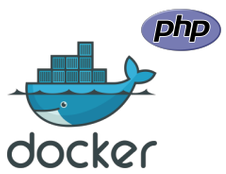
Microservice 5
Python 3.7

Microservice 1
PHP 7.2

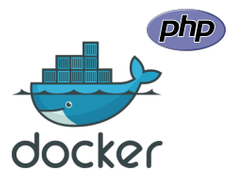


Microservice 3
PHP 7.1

Microservice 6
Python 3.6

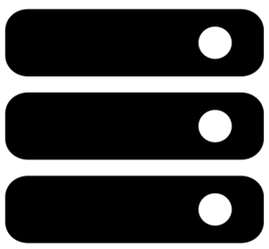


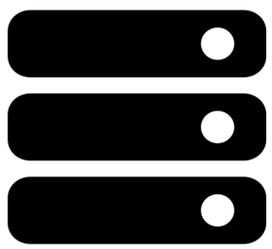
Microservice 2
PHP 7.1

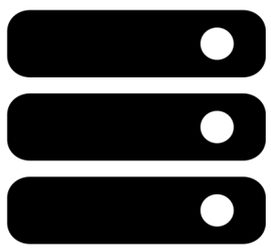


Microservice 4
PHP 7.1

Microservice 7
Go 1.10

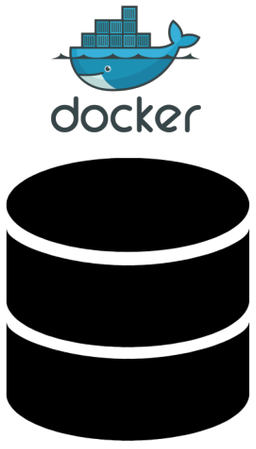




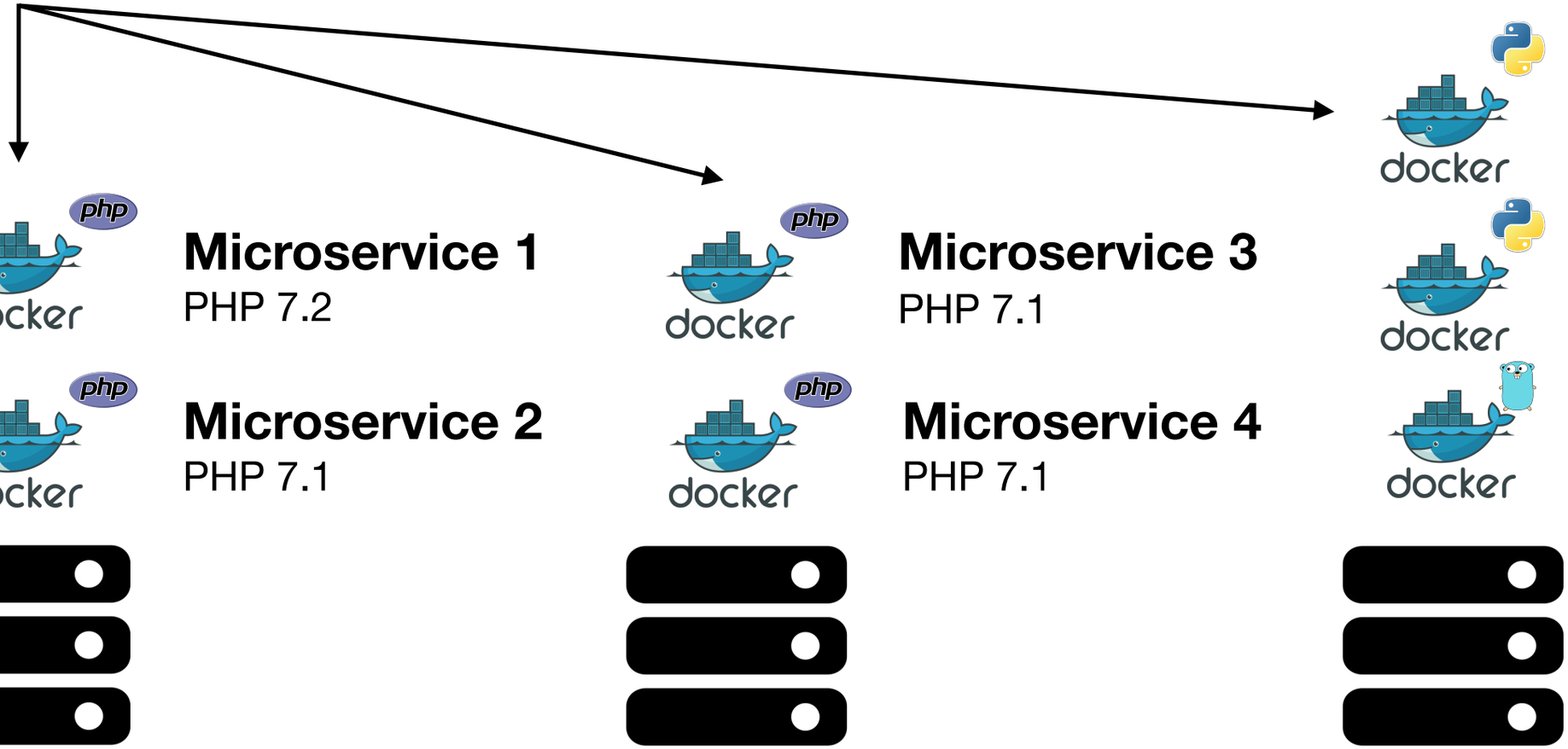


aboutbits.it alex.lanz@aboutbits.it @alex__lanz

Docker



Docker Registry



Microservice 1
PHP 7.2

Microservice 2
PHP 7.1

Microservice 3
PHP 7.1

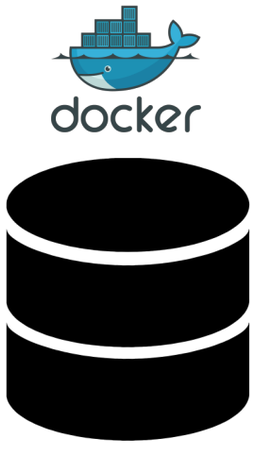
Microservice 4
PHP 7.1

Microservice 5
Python 3.7

Microservice 6
Python 3.6

Microservice 7
Go 1.10

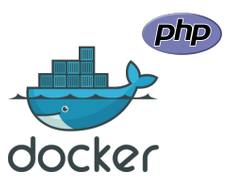
Docker



Docker Registry

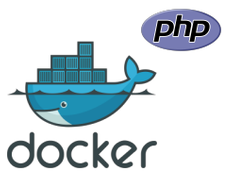
Microservice 5
Python 3.7

Microservice 1
PHP 7.2

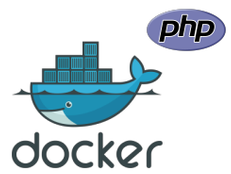


Microservice 3
PHP 7.1

Microservice 6
Python 3.6

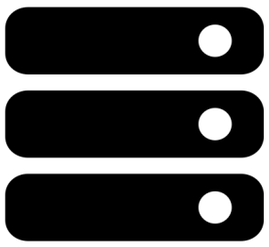


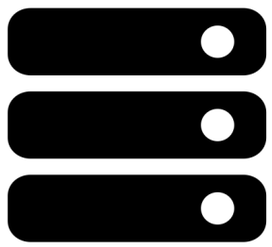
Microservice 2
PHP 7.1

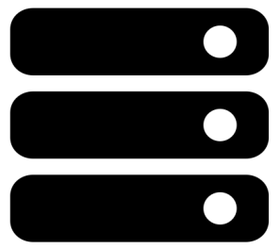


Microservice 4
PHP 7.1

Microservice 7
Go 1.10

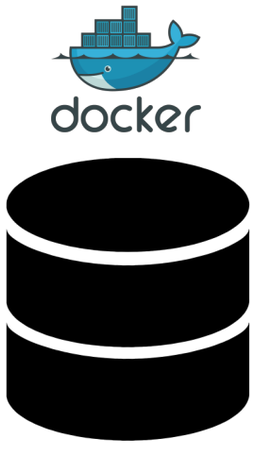






aboutbits.it alex.lanz@aboutbits.it @alex__lanz

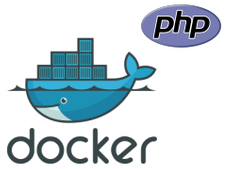
Docker



Docker Registry

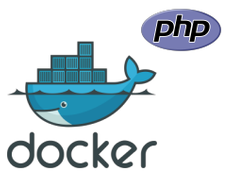
Microservice 4
PHP 7.1

Microservice 5
Python 3.7

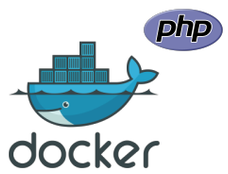


Microservice 1
PHP 7.2

Microservice 6
Python 3.6

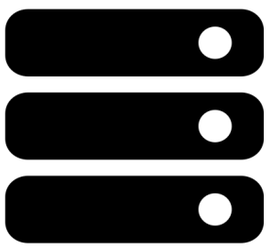


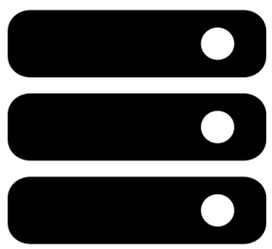
Microservice 2
PHP 7.1

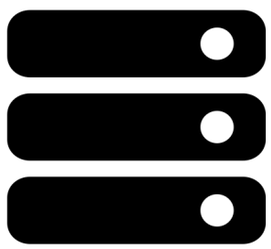


Microservice 3
PHP 7.1

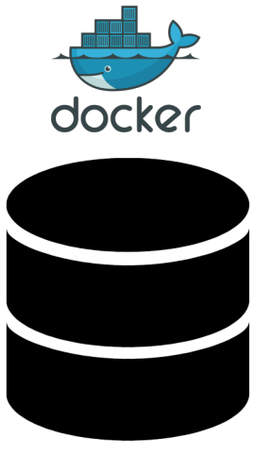
Microservice 7
Go 1.10







Docker



Docker Registry

Microservice 1
PHP 7.2

Microservice 2
PHP 7.1

Microservice 3
PHP 7.1

Microservice 4
PHP 7.1

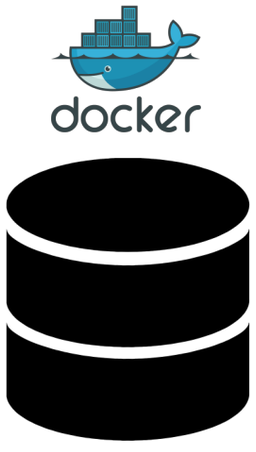
Microservice 5
Python 3.7

Microservice 6
Python 3.6

Microservice 7
Go 1.10

aboutbits.it alex.lanz@aboutbits.it @alex__lanz

Docker



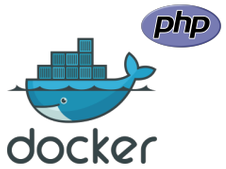
Docker Registry

Microservice 4
PHP 7.1

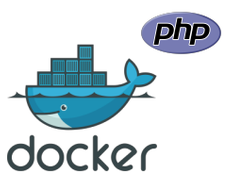
Microservice 5
Python 3.7

Microservice 6
Python 3.6

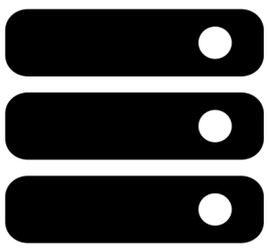
Microservice 7
Go 1.10

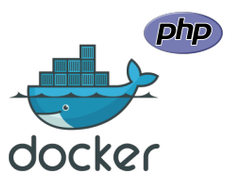


Microservice 1
PHP 7.2

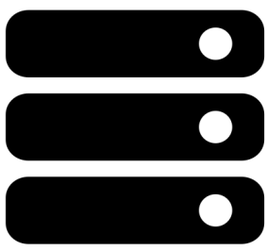


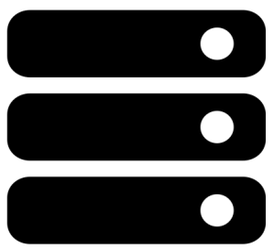
Microservice 2
PHP 7.1



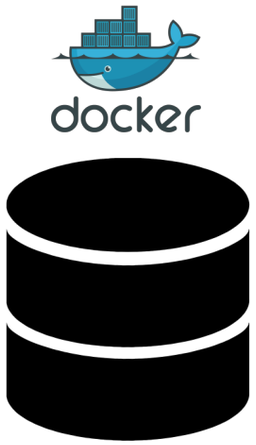


Microservice 3
PHP 7.1





Docker



Docker Registry

Microservice 1
PHP 7.2

Microservice 2
PHP 7.1

Microservice 3
PHP 7.2

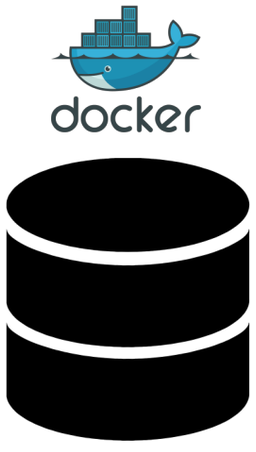
Microservice 4
PHP 7.1

Microservice 5
Python 3.7

Microservice 6
Python 3.6

Microservice 7
Go 1.10

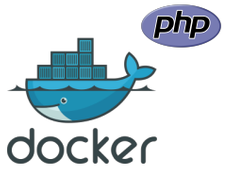
Docker



Docker Registry

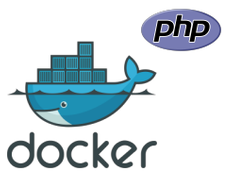
Microservice 4
PHP 7.1

Microservice 5
Python 3.7

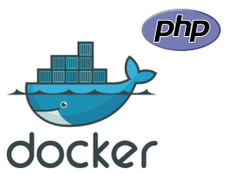


Microservice 1
PHP 7.2

Microservice 6
Python 3.6

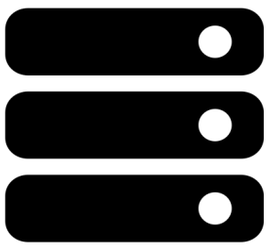


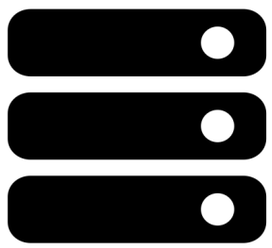
Microservice 2
PHP 7.1

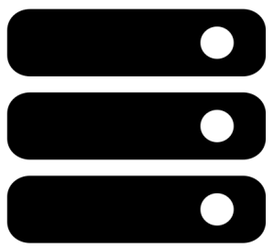


Microservice 3
PHP 7.2

Microservice 7
Go 1.10

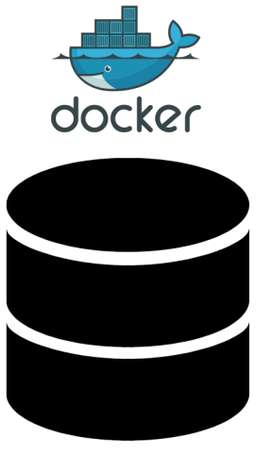






aboutbits.it alex.lanz@aboutbits.it @alex__lanz

Docker



Docker Registry

Microservice 1
PHP 7.2

Microservice 2
PHP 7.1

Microservice 3
PHP 7.2

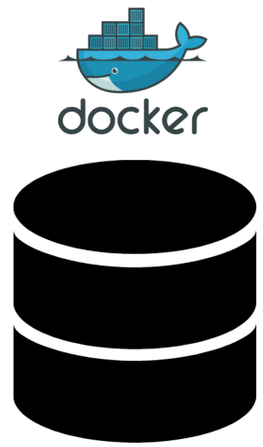
Microservice 4
PHP 7.1

Microservice 5
Python 3.7

Microservice 6
Python 3.6

Microservice 7
Go 1.10

Docker

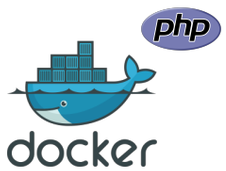


Docker Registry

Microservice 3
PHP 7.2

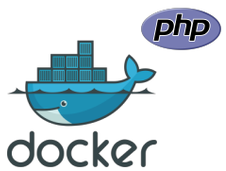
Microservice 4
PHP 7.1

Microservice 5
Python 3.7

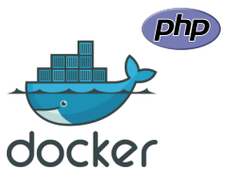


Microservice 1
PHP 7.2

Microservice 6
Python 3.6

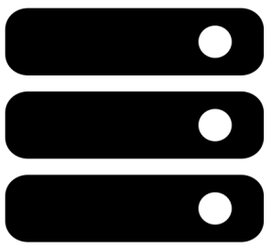


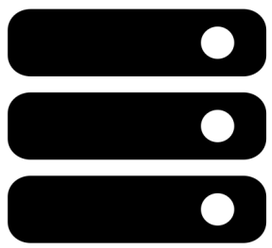
Microservice 2
PHP 7.1

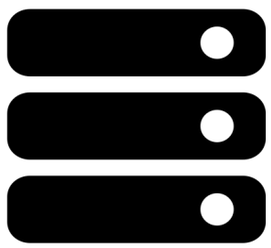


Microservice 3
PHP 7.2

Microservice 7
Go 1.10

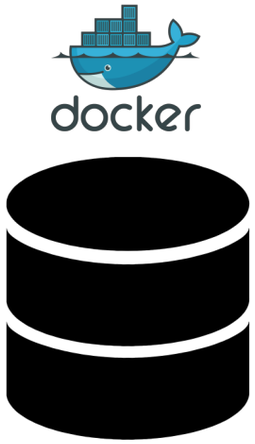




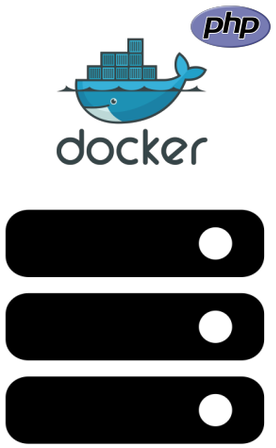


aboutbits.it alex.lanz@aboutbits.it @alex__lanz

Docker



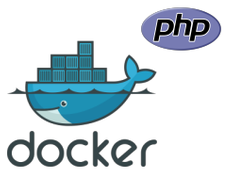
Docker Registry



Microservice 3
PHP 7.2

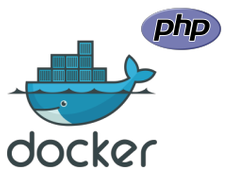
Microservice 4
PHP 7.1

Microservice 5
Python 3.7



Microservice 1
PHP 7.2

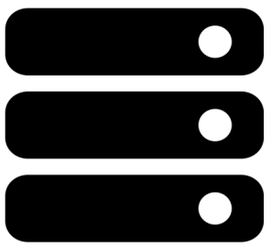
Microservice 6
Python 3.6

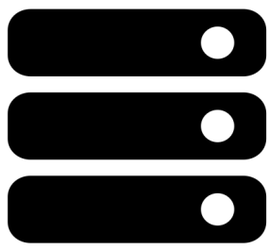


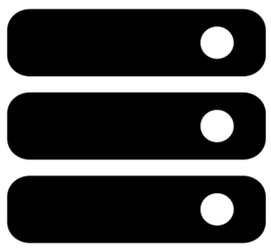
Microservice 2
PHP 7.1

Microservice 3
PHP 7.2

Microservice 7
Go 1.10







aboutbits.it alex.lanz@aboutbits.it @alex__lanz

Problems

- Keep **Setup/Maintenance** low
- Run applications in **Isolation**
- Provide easy **Scalability**
- Use the **Same Environment** everywhere
- **Development Speed** should be high

Problems

- Keep **Setup/Maintenance** low
- Run applications in **Isolation**
- Provide easy **Scalability**
- Use the **Same Environment** everywhere
- **Development Speed** should be high

Problems

- Keep **Setup/Maintenance** low ✓
- Run applications in **Isolation**
- Provide easy **Scalability**
- Use the **Same Environment** everywhere
- **Development Speed** should be high

Problems

- Keep **Setup/Maintenance** low ✓
- Run applications in **Isolation**
- Provide easy **Scalability**
- Use the **Same Environment** everywhere
- **Development Speed** should be high

Problems

- Keep **Setup/Maintenance** low ✓
- Run applications in **Isolation** ✓
- Provide easy **Scalability**
- Use the **Same Environment** everywhere
- **Development Speed** should be high

Problems

- Keep **Setup/Maintenance** low ✓
- Run applications in **Isolation** ✓
- Provide easy **Scalability**
- Use the **Same Environment** everywhere
- **Development Speed** should be high

Problems

- Keep **Setup/Maintenance** low ✓
- Run applications in **Isolation** ✓
- Provide easy **Scalability** ✓
- Use the **Same Environment** everywhere
- **Development Speed** should be high

Problems

Ok, the theory sounds great! But how can we do that?

Local Environment

- Prerequisites:
  - Docker (Linux)
  - Docker for Mac/Windows (Mac/Windows)
- Tools:
  - Docker Registry
  - Docker Compose

Local Environment

- Docker Registry
  - Docker Hub (https://hub.docker.com/)
  - Public repository for Docker containers
  - Every well known software has pre-build images there
  - Fast setup/maintenance of environment/containers

Local Environment

- Docker Compose
  - A tool for defining and running multi-container Docker applications
  - Can pull images from Docker Registry or build own images
  - Environment is version controlled

Local Environment

- Problem 1: Running multiple applications
  - Binding multiple applications on same local port (ex. 80, 443) is not possible
  - Microservice architecture requires running multiple applications in parallel

⇒Traefik

Local Environment

- Traefik:
  - Reverse Proxy / Load Balancer
  - Binds on ports 80 and 443
  - Automatically detects new Docker applications
  - Forwards requests appropriate container
  - https://traefik.io/

Local Environment

- Problem 1: Running multiple applications
  - Binding multiple applications on same local port (ex. 80, 443) is not possible
  - Microservice architecture requires running multiple applications in parallel

⇒Traefik

⇒https://github.com/aboutbits/docker-environment

Local Environment

- Problem 2: SSL certificates
  - We access the containers using a domain name (ex. service.aboutbits.local)
  - Some technologies require SSL certificates to work properly when using domain names (ex. service workers)

⇒Certificate Authority

⇒https://github.com/aboutbits/certificate-authority-tools

CI Environment

- Tasks:
  - Running the tests inside the same containers
  - Build images and push them to registry
- Tools:
  - Jenkins (https://jenkins.io/)
  - Drone (https://drone.io/)
  - Gitlab (https://gitlab.com/)

Production Environment

- Tasks:
  - Quick setup and less maintenance
  - Easy Deployment and Scaling
- Tools:
  - Docker (https://docker.com/)
  - Docker Swarm (https://docker.com/)
  - Kubernetes (https://kubernetes.io/)

Production Environment

- Docker:
  - Install Docker on the node
  - SSH into node
  - Pull images from repository
  - Run the container
- Problems:
  - Setup/Maintenance
  - Scaling

Production Environment

- Docker Swarm:
  - A clustering and scheduling tool for Docker containers
  - Connect multiple nodes together to a cluster
  - Control the cluster from your local machine
  - Uses the same concepts and CLI commands as Docker
  - Just tell Docker Swarm on how many nodes a service should run on and it does everything for you

Production Environment

- Kubernetes:

  - A system for automating deployment, scaling and management of containerized applications

  - Use a managed Kubernetes cluster (ex. Digital Ocean, AWS, Google Cloud)

  - They do the setup and maintenance work for you

  - Just tell Kubernetes on how many nodes a service should run on and it does everything for you

Production Environment

- Kubernetes:

⇒https://github.com/aboutbits/docker-environment

⇒Setup for Digital Ocean

⇒Load Balancer and Traefik

⇒Automatic Service Detection

⇒Let’s Encrypt Certificates

Problems

- Keep **Setup/Maintenance** low ✓
- Run applications in **Isolation** ✓
- Provide easy **Scalability** ✓
- Use the **Same Environment** everywhere
- **Development Speed** should be high

Problems

- Keep **Setup/Maintenance** low ✓
- Run applications in **Isolation** ✓
- Provide easy **Scalability** ✓
- Use the **Same Environment** everywhere
- **Development Speed** should be high

Problems

- Keep **Setup/Maintenance** low ✓
- Run applications in **Isolation** ✓
- Provide easy **Scalability** ✓
- Use the **Same Environment** everywhere ✓
- **Development Speed** should be high

Problems

- Keep **Setup/Maintenance** low ✓
- Run applications in **Isolation** ✓
- Provide easy **Scalability** ✓
- Use the **Same Environment** everywhere ✓
- **Development Speed** should be high

Problems

- Keep **Setup/Maintenance** low ✓
- Run applications in **Isolation** ✓
- Provide easy **Scalability** ✓
- Use the **Same Environment** everywhere ✓
- **Development Speed** should be high ✓

Thank you for your attention.

Are there any questions?

https://github.com/aboutbits