FRANKFURT UNIVERSITY OF APPLIED SCIENCES

# Cloud computing OpenFaaS-Project

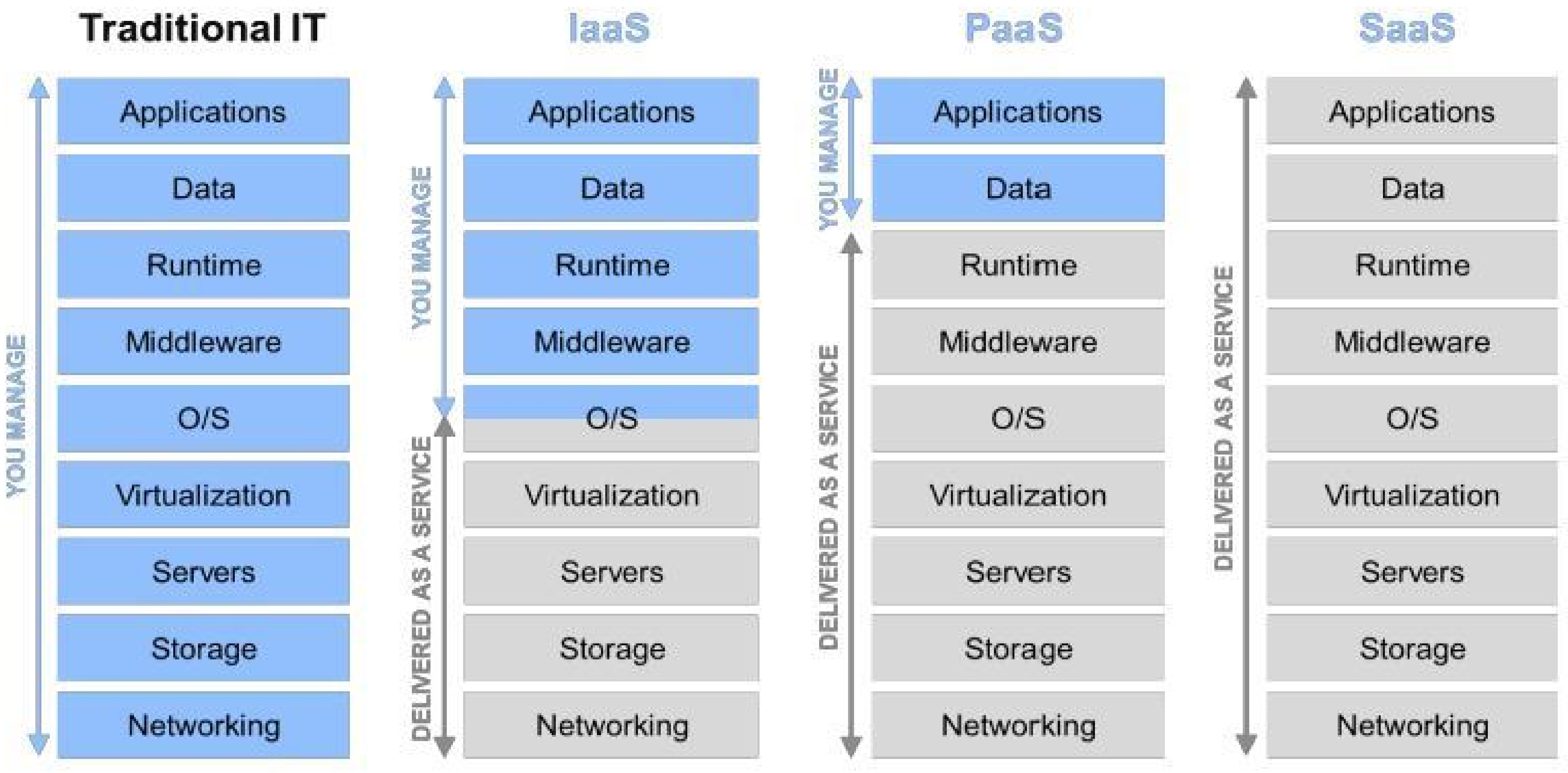
Özlem De Grave

grave@stud.fra-uas.de

# Contents

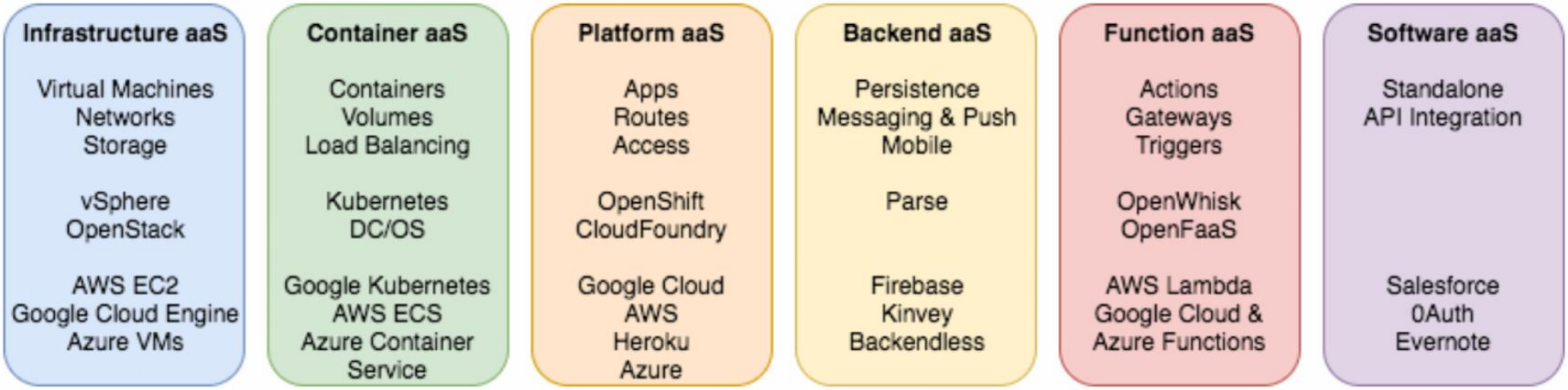
- Characteristics of OpenFaaS
- Function as a Service = Serverless
- X86-Processor vs. ARM-Processor
- Sytem overview
- Description of components
- Scalable Serverless Raspberry Pi cluster

# Cloud Service Models



| Traditional IT | IaaS | PaaS | SaaS |
|---|---|---|---|
| Applications | Applications | Applications | Applications |
| Data | Data | Data | Data |
| Runtime | Runtime | Runtime | Runtime |
| Middleware | Middleware | Middleware | Middleware |
| O/S | O/S | O/S | O/S |
| Virtualization | Virtualization | Virtualization | Virtualization |
| Servers | Servers | Servers | Servers |
| Storage | Storage | Storage | Storage |
| Networking | Networking | Networking | Networking |

# YaaaS: ”Yet another as a Service”



| Infrastructure aaS | Container aaS | Platform aaS | Backend aaS | Function aaS | Software aaS |
|---|---|---|---|---|---|
| Virtual Machines<br>Networks<br>Storage | Containers<br>Volumes<br>Load Balancing | Apps<br>Routes<br>Access | Persistence<br>Messaging & Push<br>Mobile | Actions<br>Gateways<br>Triggers | Standalone<br>API Integration |
| vSphere<br>OpenStack | Kubernetes<br>DC/OS | OpenShift<br>CloudFoundry | Parse | OpenWhisk<br>OpenFaaS | |
| AWS EC2<br>Google Cloud Engine<br>Azure VMs | Google Kubernetes<br>AWS ECS<br>Azure Container Service | Google Cloud<br>AWS<br>Heroku<br>Azure | Firebase<br>Kinvey<br>Backendless | AWS Lambda<br>Google Cloud &<br>Azure Functions | Salesforce<br>0Auth<br>Evernote |

# Function as a Service - FaaS

FaaS forms the lines between Backend and Software as a Service.

FaaS is providing a plattform on which you can run your code without thinking about the underlying structure.

FaaS is one of the components like products/services for implementing serverless .

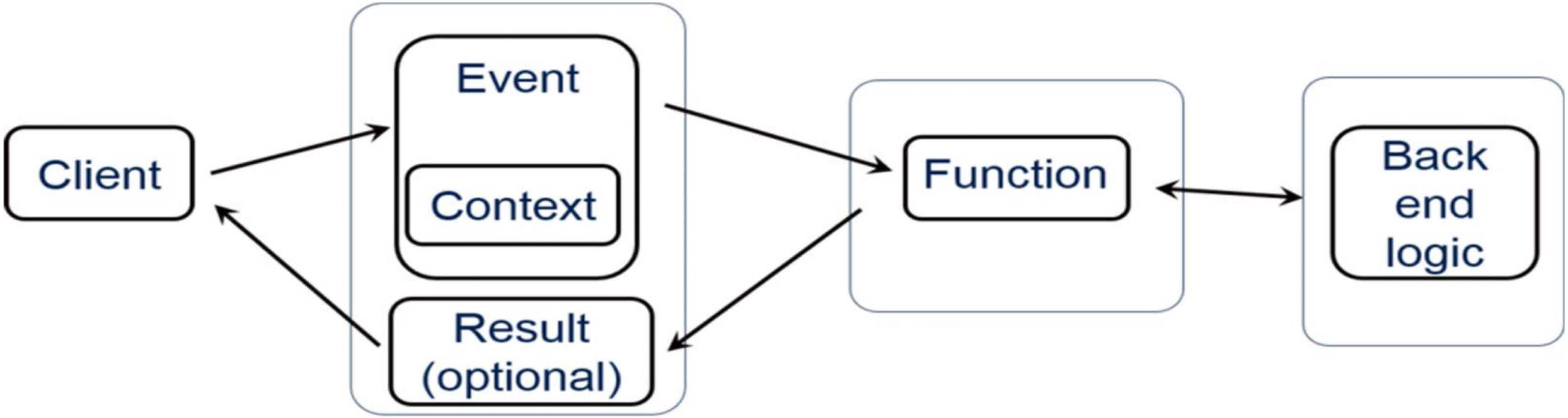
# Serverless „run code, not servers“
pay per use

Serverless is a new architectural pattern for event-driven systems. What it basically means is that for each event or request to the server, a state is created and after the request is served, the state is automatically destroyed.

Serverless architectural pattern is resorting to the components of backend, FaaS and SaaS.

Virtual machines (EC2) are not components of serverless because you have to worry about the maintenance of OS or other system components (webserver, database engine etc.).

# Serverless is a new architectural pattern for event-driven systems.



You have differents sources  for events: changing the data stage, request to your endpoints, change in the ressource state etc.

# Who hosts FaaS?

To date, there are not many companies that sell machine time:

Amazon® with its “AWS Lambdas®”

Microsoft® with its”Azure Functions®”

IBM® with its”OpenWhisk actions®”.

Google® with its”Cloud Functions”

# OpenFaaS - Open Source Software

- We need to make a distinction between *Serverless products* from IaaS providers and Open Source Software projects.

- On one hand we have *Serverless products* from IaaS providers such as Lambda, Google Cloud Functions and Azure functions and on the other frameworks such as OpenFaaS which let an orchestration platform such as Docker Swarm or Kubernetes do the heavy lifting. No vendor lock in!!
- OpenFaaS is a framework for building Serverless functions on top of containers.

# OpenFaas Alternatives

**Fission**

Fission is a framework for serverless functions on Kubernetes.

Fission is an open source project maintained by Platform9 and many contributors.

**Kubeless**

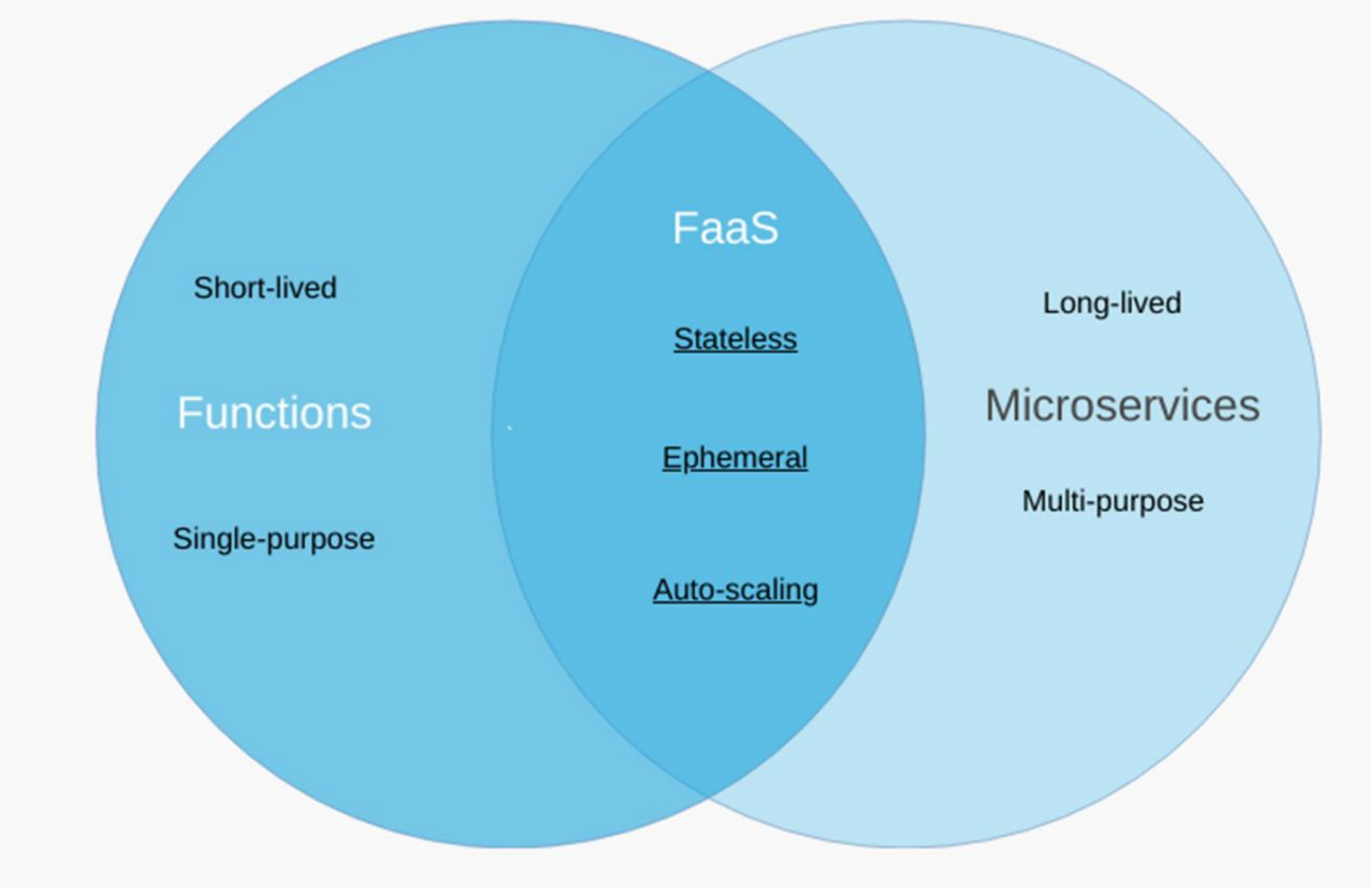
The Kubernetes Native Serverless Framework

**Knative**

Kubernetes-based platform to build, deploy, and manage modern serverless workloads.

# Faas Functions

- **FaaS** are the short programs (functions) that we make with a certain programming language, they are monolithic and we host them with a provider and they can be called by us, by other applications or even events.
- FaaS are functions that are loaded, run and when finished *all your data is destroyed and we will only be charged for the running time*.
- FaaS also comes with a convenient gateway tester that allows you to try out each of your functions directly in the browser.


Short-lived
Functions
Single-purpose
FaaS
Stateless
Ephemeral
Auto-scaling
Long-lived
Microservices
Multi-purpose

# Functions

- Functions are the unit of deployment and scale
- This scales per request
- Functions communicate through stdin/stdout.

You can attache the functions to each other; one web request is writing the request result into a queue and one another function is reading this queue and gives out an result.

# The OpenFaaS stack

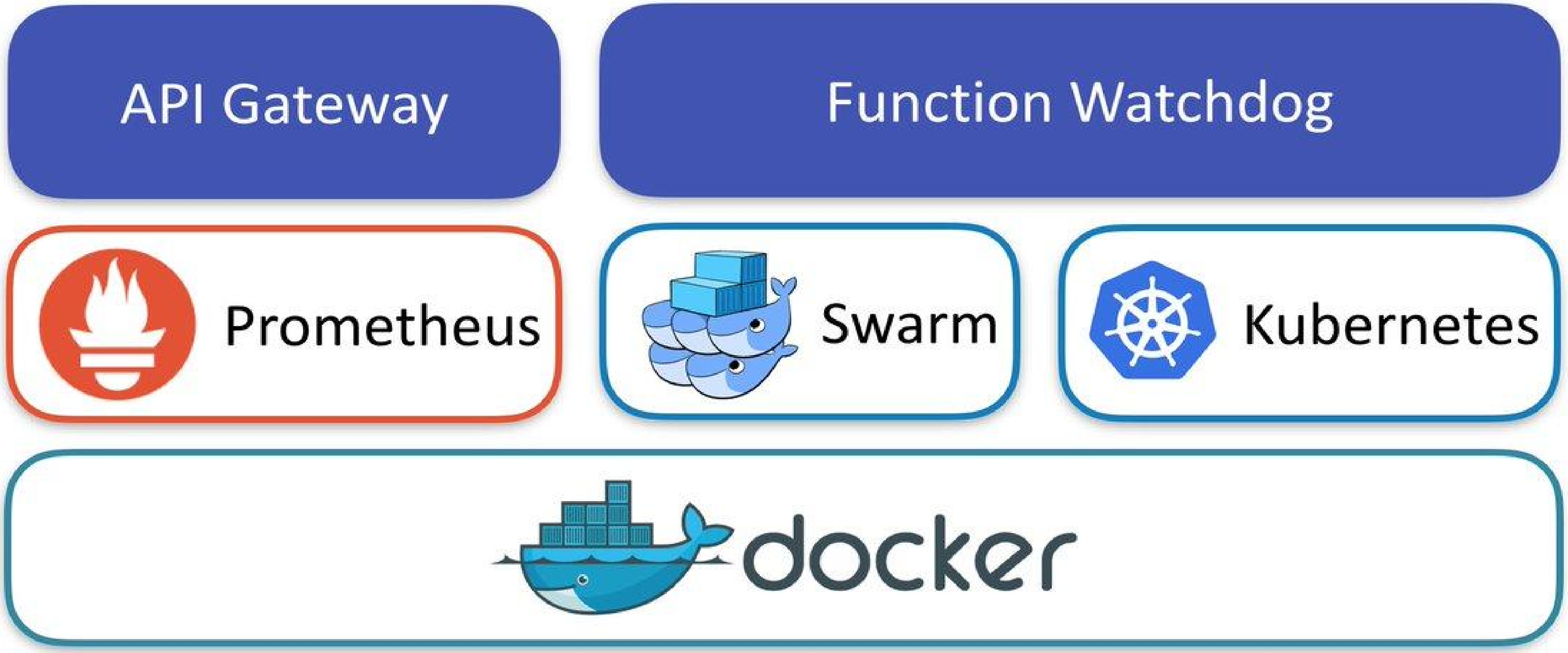
The difference with the OpenFaaS project is that any process can become a serverless function through the watchdog component and a Docker container.

**That means three things:**

- You can run code in whatever language you want
- For however long you need
- Wherever you want to

**FaaS has Prometheus metrics baked-in which means it can automatically scale your functions up and down for demand.**

# Functions as a Service



API Gateway

Function Watchdog

Prometheus

Swarm

Kubernetes

docker

# Container

- A container provides an isolated context in which an app, together with its environment, can run.
- But isolated containers often need to be managed and connected to the external world.
- Shared file systems, networking, scheduling, load balancing, and distribution are all challenges

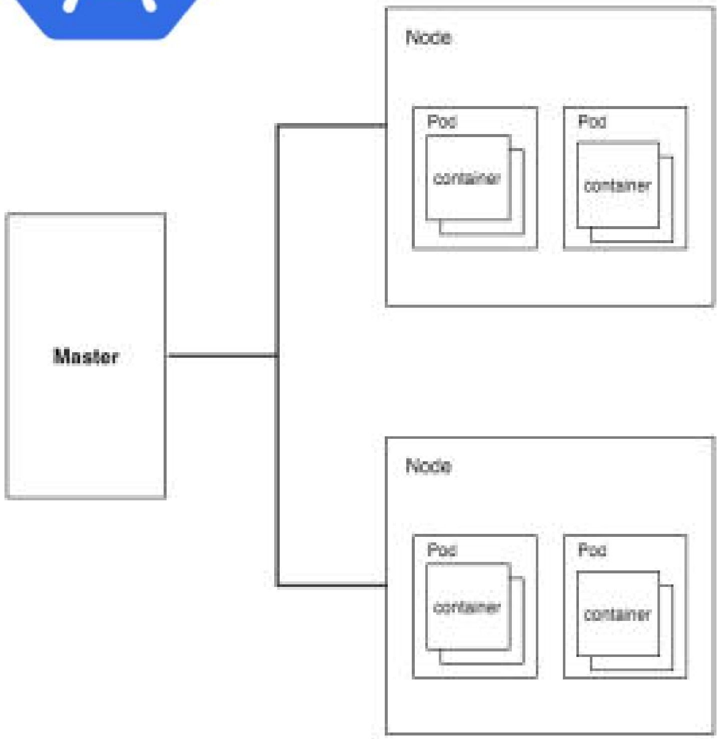
# “Build, Ship, and Run any App, Anywhere”.

- **Docker features** the Docker Engine, which is a runtime and allows you to build and run containers, and includes Docker Hub, a service for storing and sharing images. help you “Build, Ship, and Run any App, Anywhere”.
- **Docker Swarm provides** a way to administer a large number of containers spread across clusters of servers. Its filtering and scheduling system enables the selection of optimal nodes in a cluster to deploy containers.

# K8s - Phi-Beta-Kappa
# Philsophia biou kubernetes
# Love of Wisdom Pilots Life

Kubernetes” is the Greek word for a ship’ captain. The words Cybernetic and Gubernatorial are derived from“Kubernetes”.

Led by Google, the Kubernetes project focuses on building a robust platform for running thousands of containers in production.



# Kubernetes high level

A Pod is the basic smallest deployable object in the Kubernetes object model.

A Pod represents processes running on your Cluster. A single container runs inside of a pod.

Pods can hold any number of containers but usually only holds two

# Kubernetes high level

A job creates one or more Pods and ensures that a specified number of them successfully terminate. As pods successfully complete, the Job tracks the successful completions. When a specified number of successful completions is reached, the task (ie, Job) is complete. Deleting a Job will clean up the Pods it created.

A simple case is to create one Job object in order to reliably run one Pod to completion. The Job object will start a new Pod if the first Pod fails or is deleted (for example due to a node hardware failure or a node reboot).

You can also use a Job to run multiple Pods in parallel.

# Create a new function via the Faas CLI

The easiest way to create a function is to use a template and the FaaS CLI. The CLI allows you to abstract all Docker knowledge away, you just have to write a handler file in one of the supported programming languages.

The **faas-cli is** for use with OpenFaaS - a serverless functions framework for Docker & Kubernetes.

The CLI can be used to build and deploy functions to OpenFaaS.

You can build OpenFaaS functions from a set of supported language templates (such as Node.js, Python, CSharp and Ruby).

Just write a handler file such as (handler.py/handler.js) and the CLI does the rest to create a Docker image.

# Minikube

- Local development of k8s – runs a single node Kubernetes cluster in a Virtual Machine on your laptop/PC.
- All about making things easy for local development, it is not a production solution, or even close to it.

# Software Setup on VM with 10 Steps

1 - Install Hypervisor, OS- Linux on VirtualBox, KVM

2 - Install Minikube and Helm

3 - Wiring up Helm and tiller Helm's server component

Helm is the package manager for Kubernetes and client tool.

use the `helm` cli to do all of your commands. The other part is Tiller.

Tiller is the service to communicate with the Kubernetes API to manage the Helm packages.

4 - check if you can point at your local minikube instance

send *`kubectl`* to kubernetes

5- After the confirmation *`kubectl`* is pointed at minikube, start Helm `$ helm init`

6 - Install OpenFaas with Helm on kubernetes

7- Install faas CLI *`curl -sL cli.openfaas.com | sudo sh`*

8- Don't forget the *secret* for dockerhub

Using secrets is a two step process. First you need to define a new secret in your cluster and then you need to 'use' the secret to your function by adding it the deployment request or stack YAML file.

9- Install docker on DEV-VM

10- Start a local cluster: *`minikube start`*

# Minikube commands

```
$ kubectl get pod - - all namespaces
$ kubectl get nodes
```

See on which node ist running

```
$ kubectl get pod – all namespaces –o wide
```

A namespace functions as a grouping mechanism inside of Kubernetes. Services, pods, replication controllers,

and volumes can easily cooperate within a namespace, and the namespace provides a degree of isolation from

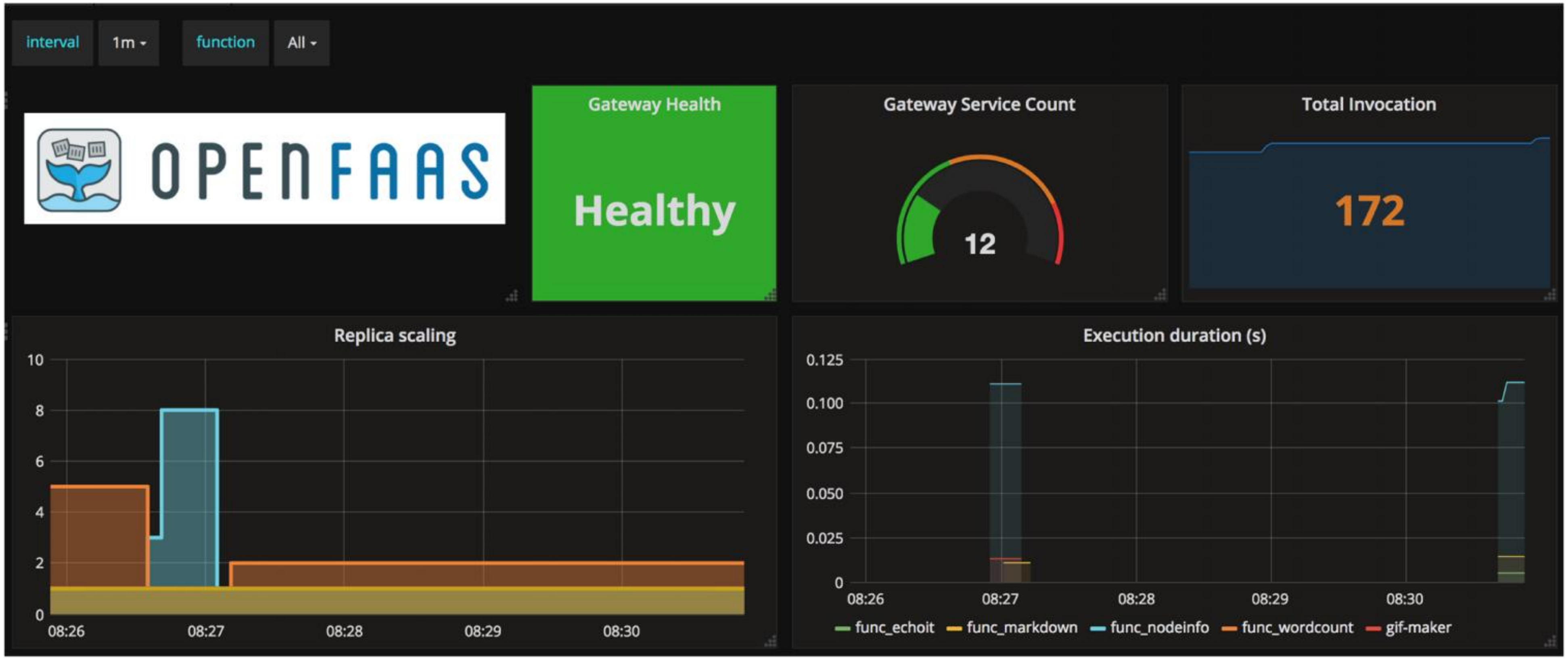
other parts of the cluster.

**AUTOSCALING PROBLEMS!!!**

# Prometheus

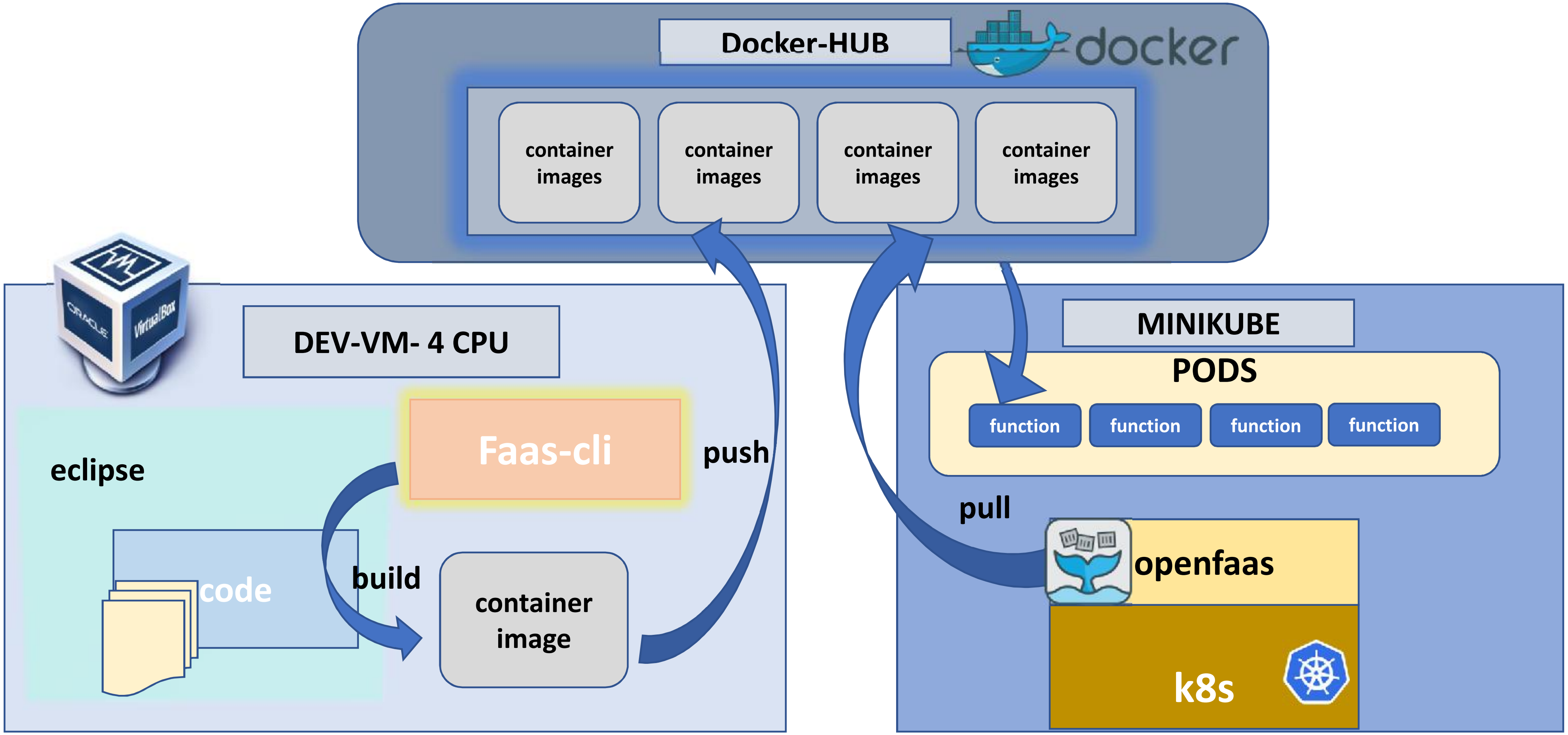
- OpenFaaS (Functions as a Service) is a framework for building serverless functions with Docker which has first class support for metrics. These metrics are scraped by Prometheus and visually represented in Grafana. The entire OpenFaaS framework can be launched in less the 60 seconds which includes the monitoring capability.
- Prometheus scrapes metrics from instrumented jobs, either directly or via an intermediary push gateway for short-lived jobs. It stores all scraped samples locally and runs rules over this data to either aggregate and record new time series from existing data or generate alerts. Grafana or other API consumers can be used to visualize the collected data.

# Prometheus



interval
1m
function
All
OPENFAAS
Gateway Health
Healthy
Gateway Service Count
12
Total Invocation
172
Replica scaling
10
8
6
4
2
0
08:26
08:27
08:28
08:29
08:30
Execution duration (s)
0.125
0.100
0.075
0.050
0.025
0
08:26
08:27
08:28
08:29
08:30
func_echoit
func_markdown
func_nodeinfo
func_wordcount
gif-maker

# System overview



# New Function openfaas

**1 - build (local DEV-VM)**

Faas-cli build -f <functionname.yml>

**2 - push  ( up to docker hub)**

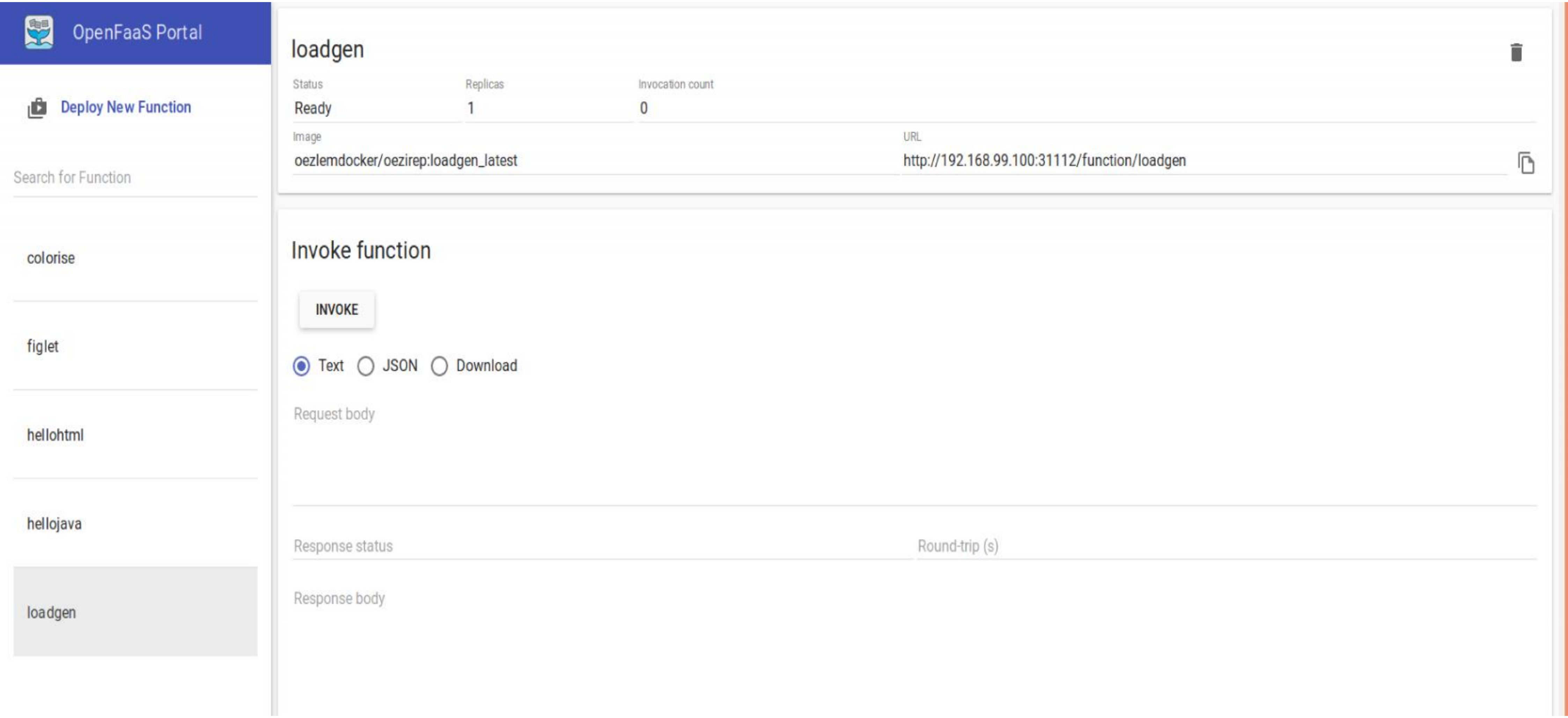
faas-cli push -f <functionname.yml>

**3 - deploy (inform openfaas where to get it)**

faas-cli deploy -f <functionname.yml>

**3*** Run the **up** command which is an alias for build, push and deploy.

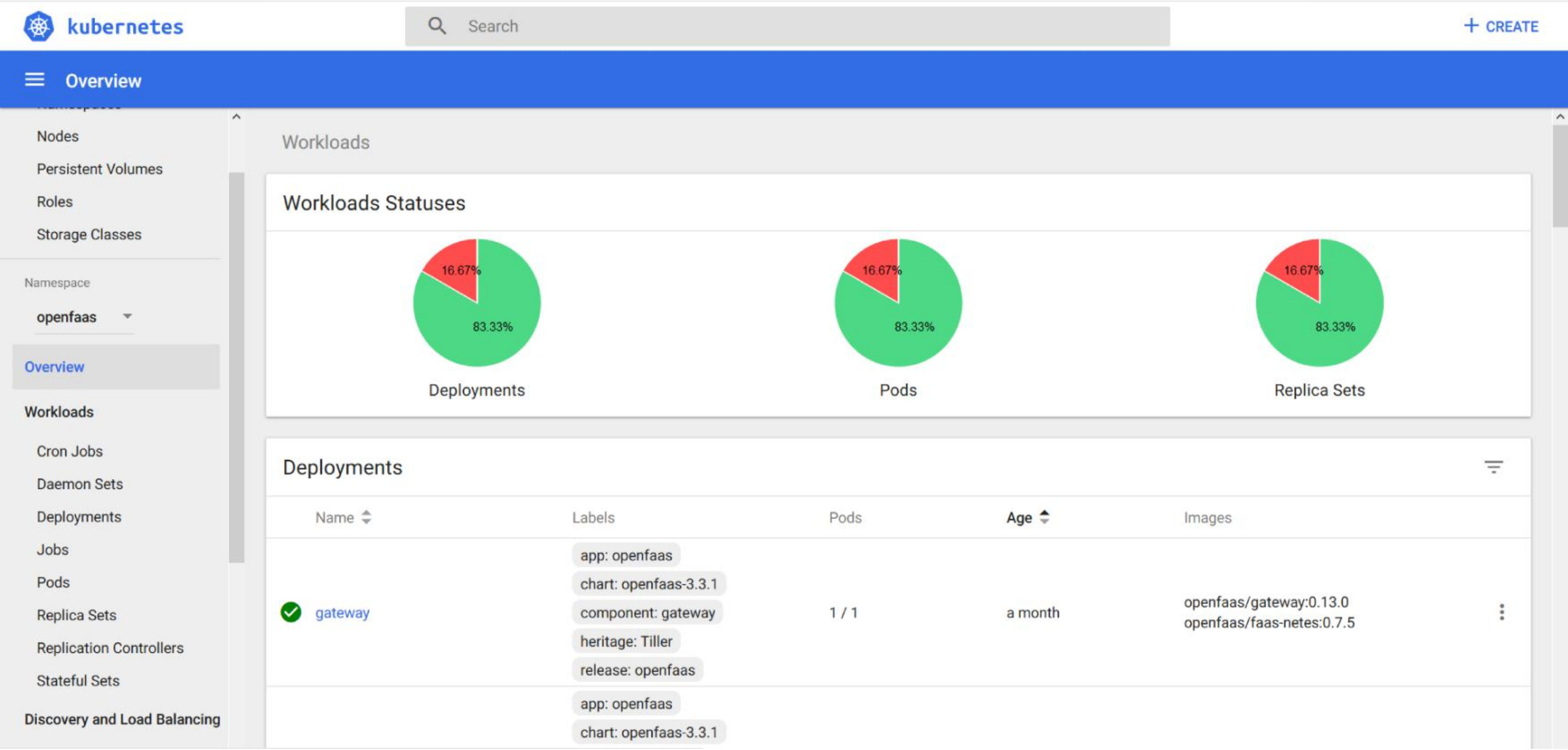
faas-cli up -f <functionname.yml>



loadgen

| Status | Replicas | Invocation count |
|---|---|---|
| Ready | 1 | 0 |

| Image | URL |
|---|---|
| oezlemdocker/oezirep:loadgen_latest | http://192.168.99.100:31112/function/loadgen |



Namespace: openfaas

Deployments

| | Name | Labels | Pods | Age | Images |
|---|---|---|---|---|---|
| ✓ | gateway | app: openfaas<br>chart: openfaas-3.3.1<br>component: gateway<br>heritage: Tiller<br>release: openfaas | 1 / 1 | a month | openfaas/gateway:0.13.0<br>openfaas/faas-netes:0.7.5 |
| | | app: openfaas<br>chart: openfaas-3.3.1 | | | |

# My functions

DEV-VM: /openfaas/functions

*ll* (listlong)

See 3 functions

For a new function

```
$ faas cli new <functionname> - -lang java8
```

# Loadgen function

- Execute Javascript;
- Give html variable back;
- Call two „onclick“ events;
- 2 functions „startload“ and „stopload“ and one counter for the calls
- Variable=ready (true/false)
- Write number of calls into a String  always in new line;
- Count up;
- response from event = request for a function to give backt the html page and generating load!
- Load is generated on the browser side!!

# Quick function deployment from Store

**Quick deployment with**

```
$ openfaas up faas-cli up –f <functionname.yml>
```

- **<u>Deploy „figlet“</u>**

**Insert parameter (TEXT)from GUI**

OR

**Terminal**

```
$ faas-cli invoke figlet
```

# The world’s fastest computer is currently China’s Sunway TaihuLight

- Built at China’s National Supercomputing Center in Wuxi
- For the fourth time in a row, Sunway Taihulight leads the twice-yearly Top500 list of the world’s most powerful supercomputers.
- It has no accelerator chips, relying instead on 40,960 Sunway 26010 processors. Each has 260 cores.
- Together they deliver a maximum sustained performance of 93.01 petaflops and a theoretical peak performance of 125.44 petaflops.

A cluster of Raspberry Pi’s is known as a “Bramble” (and sometimes a “Beowulf” cluster) and the biggest bramble ever built is believed to the made by GCHQ, who used 96 Raspberry Pi devices. This is used by GCHQ to teach big data crunching techniques.

Docker Swarm
Fritz
OpenFaas
Oracle Virtual Box
OpenFaas CLI
Docker
Kubernetes
MiniKube
Eclipse
Raspi
OpenFaas
Kubernetes

# Building a cheap and scalable Serverless Raspberry Pi cluster

Working with ARM hardware is helping me to build a tangible, educational project.

The best-known ARM SoC board is the Raspberry Pi.

The Raspberry Pi is known as a System on Chip (SoC) micro-computer based around a chipset from Broadcom designed for use with mobile phones.

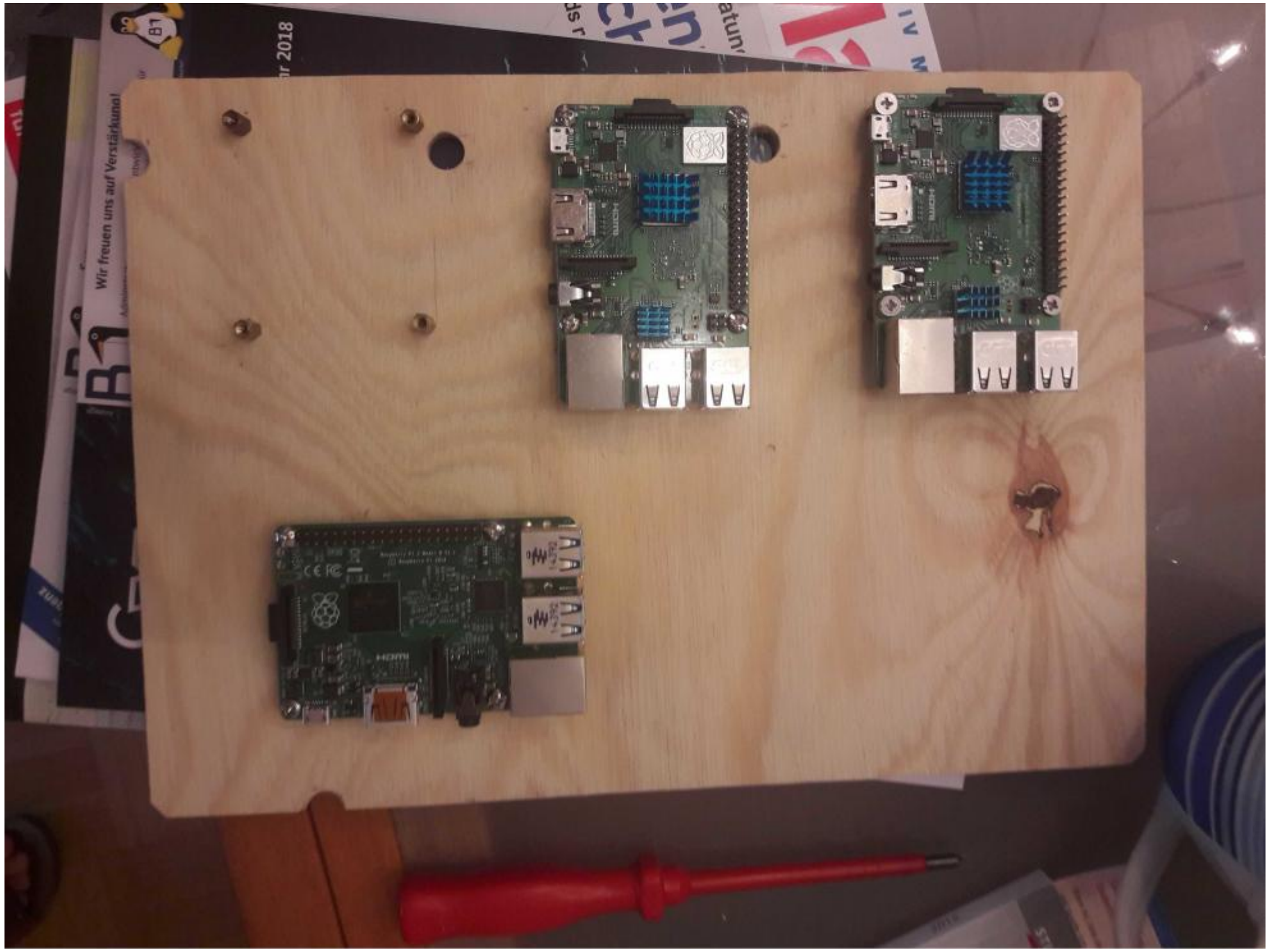
# Build and run a modern cloud infrastructure

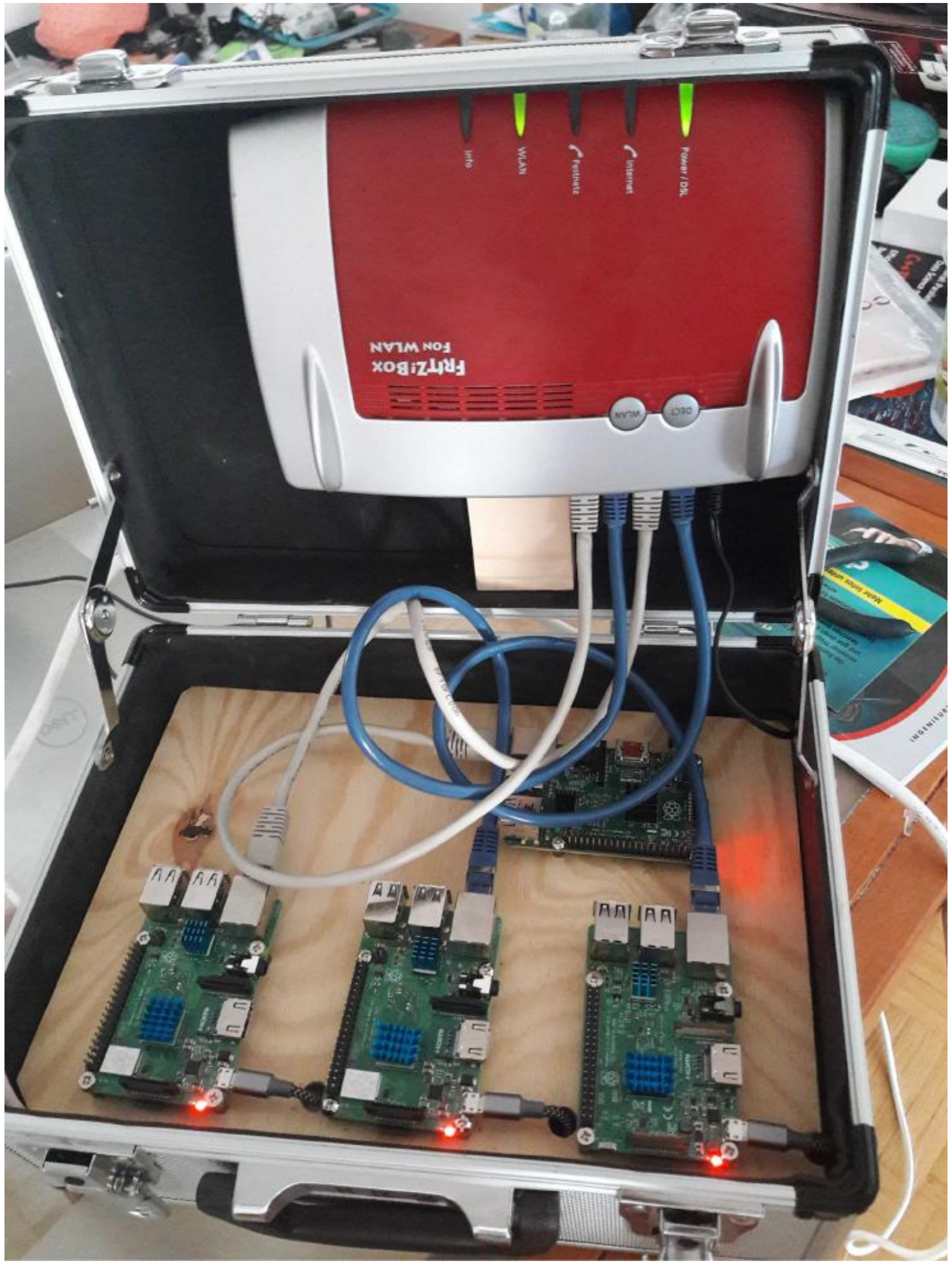
| | RasPi Model | RAM | CPU Freq | SD |
|---|---|---|---|---|
| M1 | 3 B+ | 1 GB | 1,4 GHz | 8 GB |
| w1 | 3 B | 1 GB | 1,2 GHz | 8 GB |
| w2 | 3 B+ | 1 GB | 1,4 GHz | 8 GB |
| w3 | 2 B | 1 GB | 900 MHz | 4 GB |

Enjoy this demonstration on how to quickly turn four Raspberry Pi boards into a cluster computer. Raspberry Pi boards networked and talking to each other.

- FritzBox 7390; Internet with existing WLAN
- ANKER- 6- port USB hub
- Power cable and 2-way junction (for FritzBox and USB charger)
- 4 USB cable
- 4 LAN cable
- 1 Case

DEVELOPER-KONFERENZEN
KOMPAKT
Frühjahr 2018

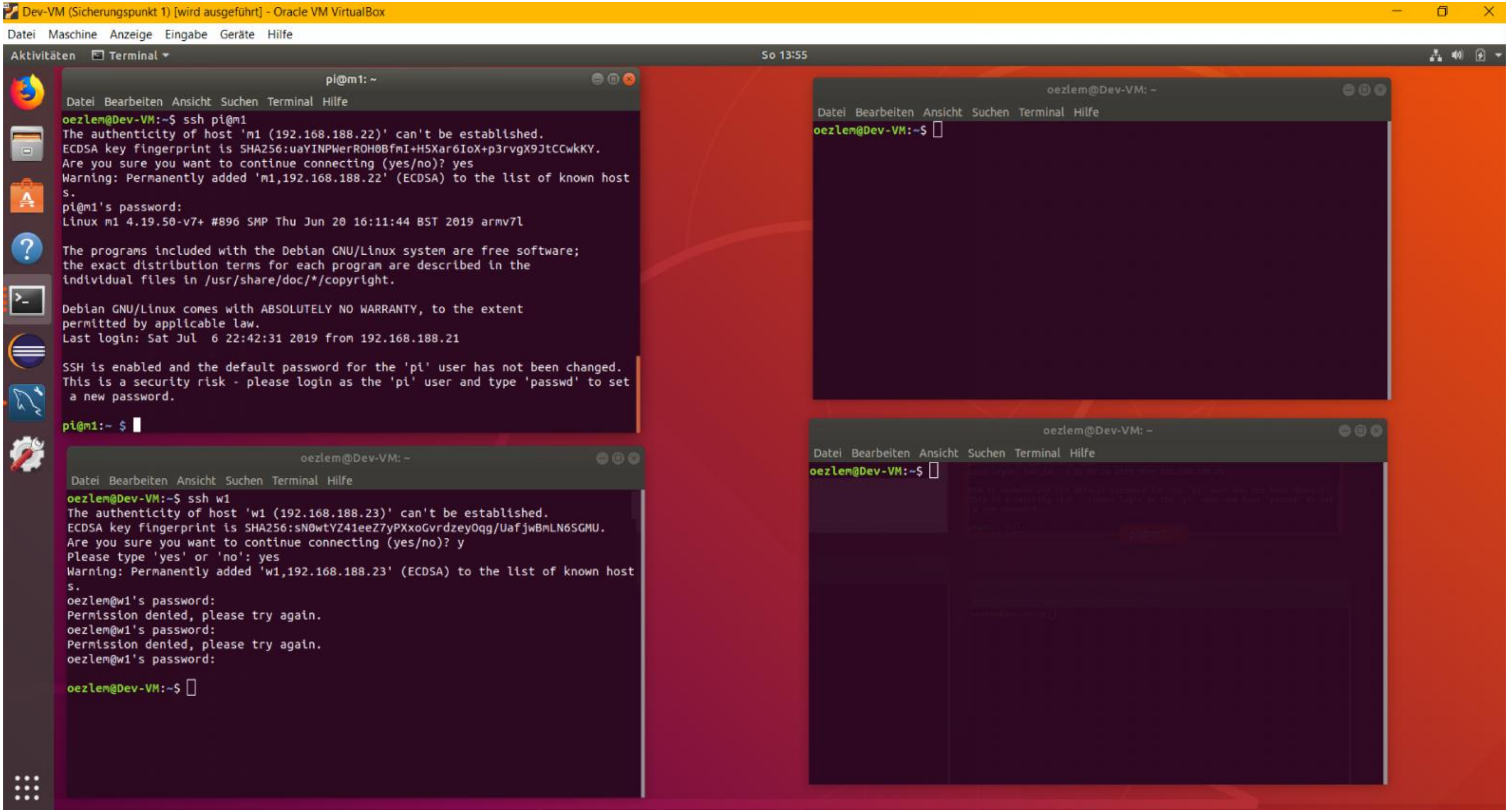

Wir freuen uns auf Verstärkung!
2018


Info
WLAN
Festnetz
Internet
Power / DSL
FRITZ!Box
Fon WLAN
WLAN
DECT

# Set up on Raspi Cluster

- OS- Rasbian Buster Lite (smallest) unpack the zip and you have an image
- Image on SD card with Win32 Disc Imager

You have an image with two partitions; on boot partition set up the `ssh` file so you can reach raspi per `ssh and` use mouse/keyboard.



Dev-VM (Sicherungspunkt 1) [wird ausgeführt] - Oracle VM VirtualBox
Datei Maschine Anzeige Eingabe Geräte Hilfe
Aktivitäten
Terminal
So 13:55
pi@m1: ~
oezlem@Dev-VM: ~
Datei Bearbeiten Ansicht Suchen Terminal Hilfe

```
oezlem@Dev-VM:~$ ssh pi@m1
The authenticity of host 'm1 (192.168.188.22)' can't be established.
ECDSA key fingerprint is SHA256:uaYINPWerROH0BfmI+H5Xar6IoX+p3rvgX9JtCCwkKY.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'm1,192.168.188.22' (ECDSA) to the list of known host
s.
pi@m1's password:
Linux m1 4.19.50-v7+ #896 SMP Thu Jun 20 16:11:44 BST 2019 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Jul  6 22:42:31 2019 from 192.168.188.21

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
 a new password.

pi@m1:~ $ 
```

```
oezlem@Dev-VM:~$ ssh w1
The authenticity of host 'w1 (192.168.188.23)' can't be established.
ECDSA key fingerprint is SHA256:sN0wtYZ41eeZ7yPXxoGvrdzeyOqg/UafjwBmLN6SGMU.
Are you sure you want to continue connecting (yes/no)? y
Please type 'yes' or 'no': yes
Warning: Permanently added 'w1,192.168.188.23' (ECDSA) to the list of known host
s.
oezlem@w1's password:
Permission denied, please try again.
oezlem@w1's password:
Permission denied, please try again.
oezlem@w1's password:

oezlem@Dev-VM:~$ 
```

```
oezlem@Dev-VM:~$ 
```

```
oezlem@Dev-VM:~$ 
```

# Setup Master

## Master - M1

IP:192.168.188.25

per `ssh pi@` 192.168.188.25 – pswd: „raspberry“

- Change the hostname `$ sudo nano /etc/hosts`

Replace „raspberrypi“ with desired name

- Save + exit
- `$ sudo nano /etc/hostname`

Replace „raspberrypi“ with desired name

- Save + exit
- `$ sudo reboot`

per `ssh pi@` 192.168.188.25 – pswd: „raspberry“

Check hostname

```
$ hostname
```

**EXIT**

# Setup workers

## worker - w1

IP:192.168.188.23

## worker – w2

IP:192.168.188.24

## worker – w3

IP:192.168.188.22

**Repeat all**

..

Check hostname

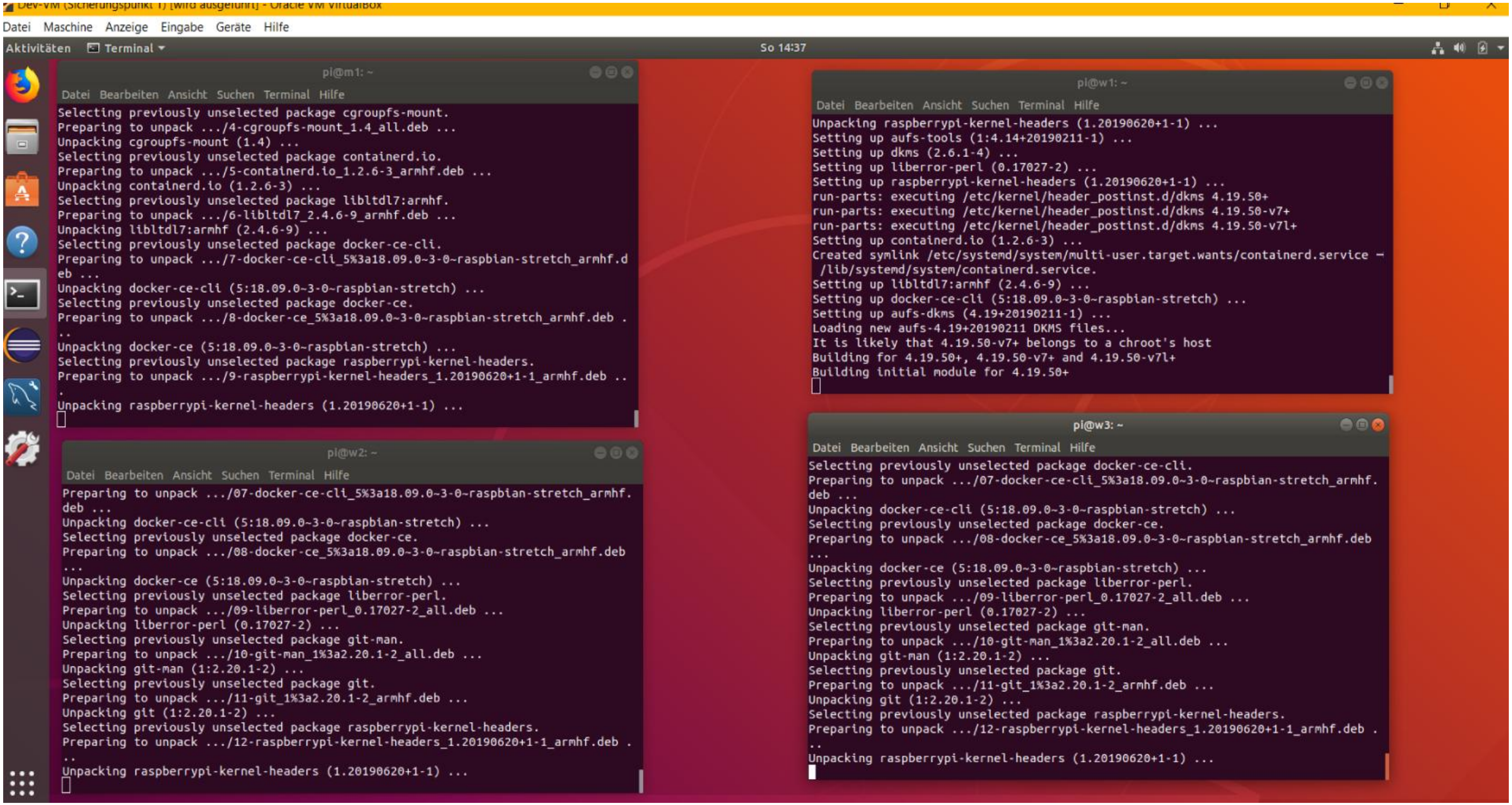
```
$ hostname
```

Run cluster down do for each

```
EXIT
```

Run cluster down do for each

```
$ ssh
$ pi@m1 sudo shutdown –P now
$ pi@w1 sudo shutdown –P now
$ pi@w2 sudo shutdown –P now
$ pi@w3 sudo shutdown –P now
```



Dev-VM (Sicherungspunkt 1) [wird ausgeführt] - Oracle VM VirtualBox
Datei Maschine Anzeige Eingabe Geräte Hilfe
Aktivitäten
Terminal ▾
So 14:37
pi@m1: ~
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
pi@w2: ~
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
pi@w1: ~
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
pi@w3: ~
Datei Bearbeiten Ansicht Suchen Terminal Hilfe

```
Selecting previously unselected package cgroupfs-mount.
Preparing to unpack .../4-cgroupfs-mount_1.4_all.deb ...
Unpacking cgroupfs-mount (1.4) ...
Selecting previously unselected package containerd.io.
Preparing to unpack .../5-containerd.io_1.2.6-3_armhf.deb ...
Unpacking containerd.io (1.2.6-3) ...
Selecting previously unselected package libltdl7:armhf.
Preparing to unpack .../6-libltdl7_2.4.6-9_armhf.deb ...
Unpacking libltdl7:armhf (2.4.6-9) ...
Selecting previously unselected package docker-ce-cli.
Preparing to unpack .../7-docker-ce-cli_5%3a18.09.0~3-0~raspbian-stretch_armhf.d
eb ...
Unpacking docker-ce-cli (5:18.09.0~3-0~raspbian-stretch) ...
Selecting previously unselected package docker-ce.
Preparing to unpack .../8-docker-ce_5%3a18.09.0~3-0~raspbian-stretch_armhf.deb .
..
Unpacking docker-ce (5:18.09.0~3-0~raspbian-stretch) ...
Selecting previously unselected package raspberrypi-kernel-headers.
Preparing to unpack .../9-raspberrypi-kernel-headers_1.20190620+1-1_armhf.deb ..
.
Unpacking raspberrypi-kernel-headers (1.20190620+1-1) ...
```

```
Preparing to unpack .../07-docker-ce-cli_5%3a18.09.0~3-0~raspbian-stretch_armhf.
deb ...
Unpacking docker-ce-cli (5:18.09.0~3-0~raspbian-stretch) ...
Selecting previously unselected package docker-ce.
Preparing to unpack .../08-docker-ce_5%3a18.09.0~3-0~raspbian-stretch_armhf.deb
...
Unpacking docker-ce (5:18.09.0~3-0~raspbian-stretch) ...
Selecting previously unselected package liberror-perl.
Preparing to unpack .../09-liberror-perl_0.17027-2_all.deb ...
Unpacking liberror-perl (0.17027-2) ...
Selecting previously unselected package git-man.
Preparing to unpack .../10-git-man_1%3a2.20.1-2_all.deb ...
Unpacking git-man (1:2.20.1-2) ...
Selecting previously unselected package git.
Preparing to unpack .../11-git_1%3a2.20.1-2_armhf.deb ...
Unpacking git (1:2.20.1-2) ...
Selecting previously unselected package raspberrypi-kernel-headers.
Preparing to unpack .../12-raspberrypi-kernel-headers_1.20190620+1-1_armhf.deb .
..
Unpacking raspberrypi-kernel-headers (1.20190620+1-1) ...
```

```
Unpacking raspberrypi-kernel-headers (1.20190620+1-1) ...
Setting up aufs-tools (1:4.14+20190211-1) ...
Setting up dkms (2.6.1-4) ...
Setting up liberror-perl (0.17027-2) ...
Setting up raspberrypi-kernel-headers (1.20190620+1-1) ...
run-parts: executing /etc/kernel/header_postinst.d/dkms 4.19.50+
run-parts: executing /etc/kernel/header_postinst.d/dkms 4.19.50-v7+
run-parts: executing /etc/kernel/header_postinst.d/dkms 4.19.50-v7l+
Setting up containerd.io (1.2.6-3) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service →
 /lib/systemd/system/containerd.service.
Setting up libltdl7:armhf (2.4.6-9) ...
Setting up docker-ce-cli (5:18.09.0~3-0~raspbian-stretch) ...
Setting up aufs-dkms (4.19+20190211-1) ...
Loading new aufs-4.19+20190211 DKMS files...
It is likely that 4.19.50-v7+ belongs to a chroot's host
Building for 4.19.50+, 4.19.50-v7+ and 4.19.50-v7l+
Building initial module for 4.19.50+
```

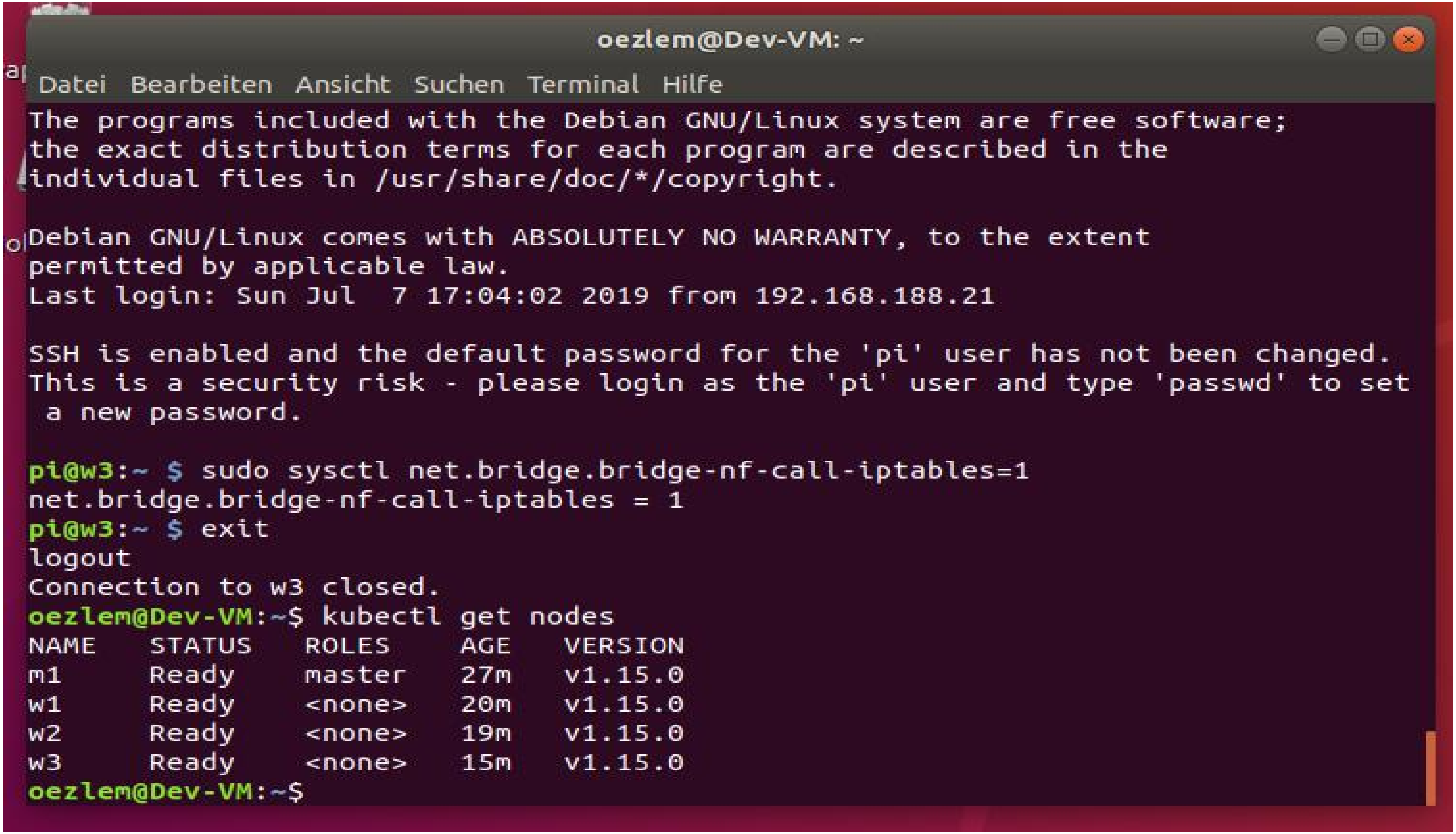
```
Selecting previously unselected package docker-ce-cli.
Preparing to unpack .../07-docker-ce-cli_5%3a18.09.0~3-0~raspbian-stretch_armhf.
deb ...
Unpacking docker-ce-cli (5:18.09.0~3-0~raspbian-stretch) ...
Selecting previously unselected package docker-ce.
Preparing to unpack .../08-docker-ce_5%3a18.09.0~3-0~raspbian-stretch_armhf.deb
...
Unpacking docker-ce (5:18.09.0~3-0~raspbian-stretch) ...
Selecting previously unselected package liberror-perl.
Preparing to unpack .../09-liberror-perl_0.17027-2_all.deb ...
Unpacking liberror-perl (0.17027-2) ...
Selecting previously unselected package git-man.
Preparing to unpack .../10-git-man_1%3a2.20.1-2_all.deb ...
Unpacking git-man (1:2.20.1-2) ...
Selecting previously unselected package git.
Preparing to unpack .../11-git_1%3a2.20.1-2_armhf.deb ...
Unpacking git (1:2.20.1-2) ...
Selecting previously unselected package raspberrypi-kernel-headers.
Preparing to unpack .../12-raspberrypi-kernel-headers_1.20190620+1-1_armhf.deb .
..
Unpacking raspberrypi-kernel-headers (1.20190620+1-1) ...
```

# Running raspi cluster

- I have 4 seperate computers and I run the the software installation parallel.

- So I could also see the performance deferences between

Pi2 and P3 or P3+…..espacially during the docker installation



```
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Jul  7 17:04:02 2019 from 192.168.188.21

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
 a new password.

pi@w3:~ $ sudo sysctl net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-iptables = 1
pi@w3:~ $ exit
logout
Connection to w3 closed.
oezlem@Dev-VM:~$ kubectl get nodes
NAME   STATUS   ROLES    AGE   VERSION
m1     Ready    master   27m   v1.15.0
w1     Ready    <none>   20m   v1.15.0
w2     Ready    <none>   19m   v1.15.0
w3     Ready    <none>   15m   v1.15.0
oezlem@Dev-VM:~$
```

# Sources

- https://blog.alexellis.io/introducing-functions-as-a-service/
- https://kubernetes.io/de/docs/tasks/tools/install-minikube/
- https://de.slideshare.net/AmazonWebServices/getting-started-with-aws-lambda-and-the-serverless-cloud/29
- https://www.slideshare.net/ServerlessConf/sam-kroonenburg-and-pete-sbarski-the-story-of-a-serverless-startup
- https://kubernetes.io/
- https://github.com/openfaas/faas-cli
- https://www.it-talents.de/blog/gastbeitraege/serverless-prinzipien-use-cases-und-mehr-als-nur-function-as-a-service
- https://github.com/teamserverless/k8s-on-raspbian
- https://www.raspberrypi.org/magpi/cluster-computer-raspberry-pi-3/
- https://pandorafms.com/blog/faas-functions-as-a-service-monitoring/