

One year of Deploying Applications for Docker, CoreOS, Kubernetes and Co

thomas@endocode.com

LinuxCon+ContainerCon, Berlin, Oct 5, 2016

HI!



Thomas Fricke

thomas@endocode.com

CTO Endocode

- System Automation
- DevOps
- Cloud, Database and Software Architect

ENDOCODE

- high-quality software solutions
- best software engineering practices: test driven
- well known open source projects: <https://github.com/endocode>
- diverse range of technologies
- decades of experience
 - software development,
 - team management
 - 100000s of server years in public and private clouds
- **Be it web, mobile, server or desktop we use:
open source meet any challenge**

F.E. A FEW DAYS AGO: FIXING A BUG

- Bug hunt in fleet
- Found the bug in a Go library:
<https://golang.org/pkg/crypto/>
- Fixed!!!
<https://go-review.googlesource.com/#/c/20687/>

MORE BUGFIX EXAMPLES

- Application breaks
- **systemd problem**
- **NO! journald problem**
- analysis: application writes a log line longer than the kernel buffer used by journald
- **FIX: enlarge the kernel buffer**
- **Push fix to the upstream kernel**

AGENDA

Containers or Virtualization

Kubernetes

CoreOS

Starting point

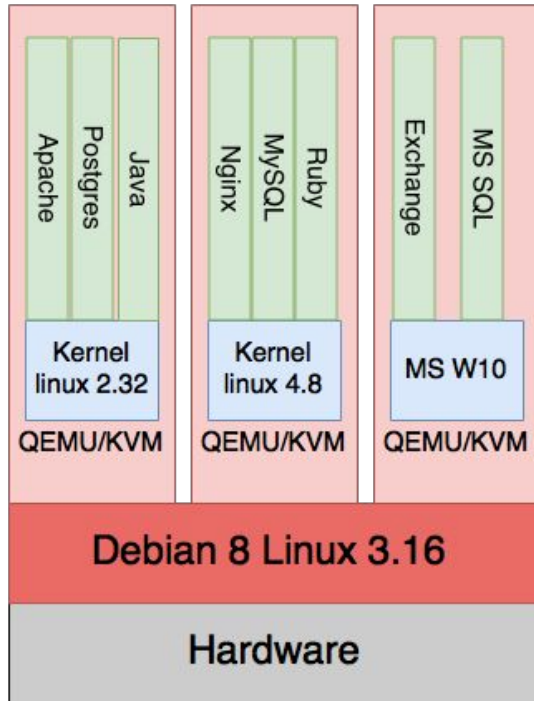
Migration

Case Study: immmr

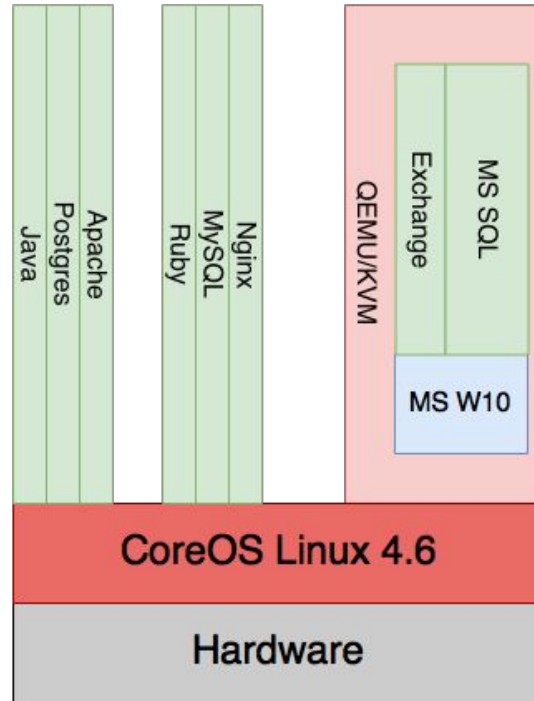
Success, challenges, 'what is missing'

PROCESS LAYOUT

Virtual Machines



Container



CONTAINER OR VIRTUALIZATION

Topic	Container	Virtualisation	
Isolation	OS Level, OS namespaces	CPU Level: Ring 0/Ring 3	
foreign CPU	no	yes, with emulation	
foreign kernels, OS	no	yes	kernel is common
emulated devices	no	yes	security
host devices	direct	virtio driver	security
CPU performance	100%	95%	
IO performance	100%	<<100%	
root isolation	yes	yes	USER directive
CPU cache attacks	easy	possible	PoC ?

CAN I MIX CONTAINERS AND HYPERVISORS?

YES!

```
# rkt run --stage1-name=coreos.com/rkt/stage1-kvm:1.14.0 coreos.com/etcd:v2.0.9
```

Docker

- Mac OS: xhyve
- Windows: Hyper-V

Kubernetes

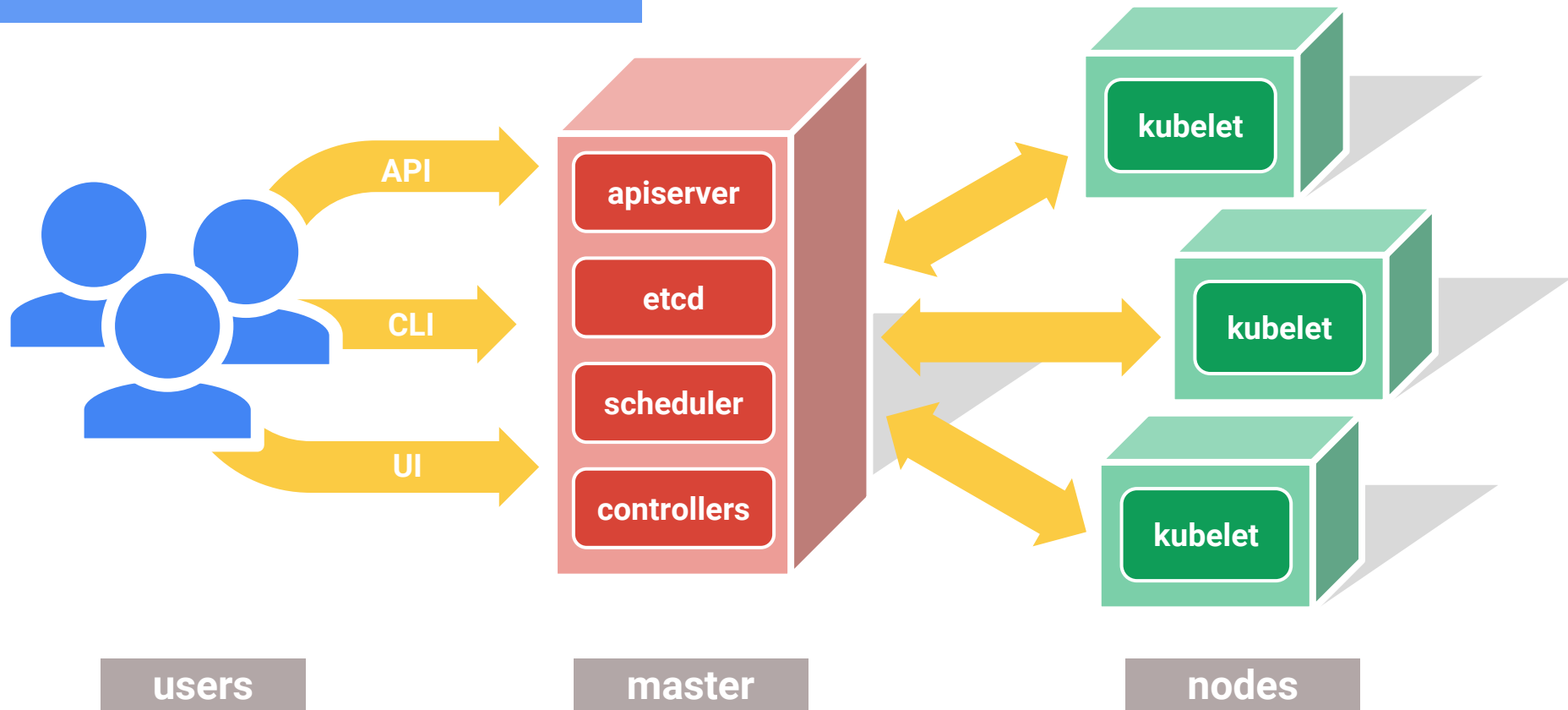
Greek for *“Helmsman”*; also the root of the words *“governor”* and *“cybernetic”*

- Runs and manages containers
- Inspired and informed by Google’s experiences and internal systems
- Supports multiple cloud and bare-metal environments
- Supports multiple container runtimes
- **100% Open source**, written in Go

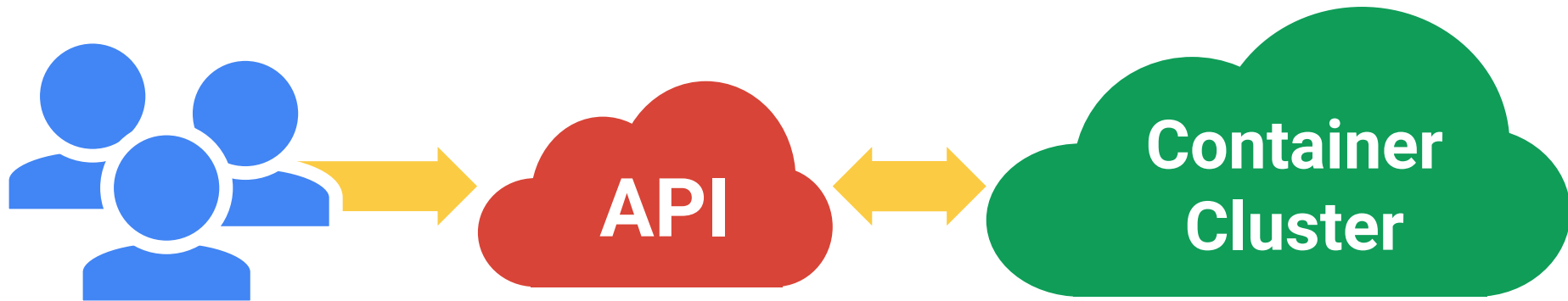
Manage applications, not machines



The 10000 foot view



All you really care about



CoreOS



Core OS

CoreOS trusted computing

Cluster access

Container Integrity

OS Integrity

Hardware

Kubernetes

rkt

CoreOS Linux

Firmware TPM

ECOSYSTEM



Core OS



clair



PROJECT CALICO



flannel



rkt



etcd

Torus



OPENSTACK



TECTONIC



ENDOCODE

STARTING POINT - ARCHITECTURE

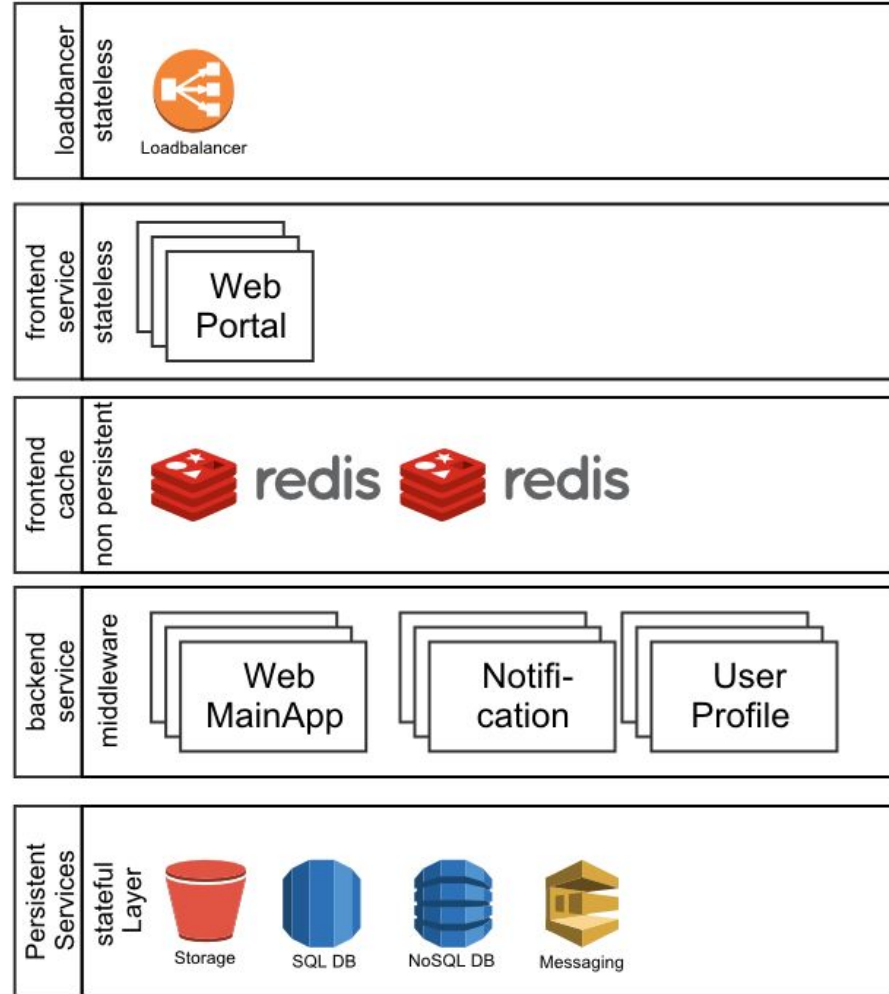
WE NEVER START FROM SCRATCH

- Almost no project starts from a green field
- Technical debt
- Environments not made for microservices

ARCHITECTURE LEVELS

- Distributed Systems
- Container Patterns
- Container Processes
- Cluster Layout
- Resources
 - CPU
 - Memory
 - Network
 - Block IO
 - CUDA
 - ...

- strict layered architecture
 - separation of stateless
 - and persistent data
- inside the pods
 - developers are free to use what they want
 - contract is binding to the outside



EXISTING HETEROGENEOUS ENVIRONMENT

- Programming languages and their runtimes
- Various databases from various generations
 - SQL
 - NoSQL
- Local and sessions storage
- Message queueing

SEMI-AUTOMATED DEPLOYMENT

- Deployment chain automation
- Knowledge about staging and release processes typically implicit and critical

VM CLUSTER BASED ARCHITECTURES

- Assumes complete OS
- Package management
- Configuration management (at runtime)


MIGRATION

FROM VMs TO PODS

OS instances  microservices in Pods

- pods are containers sharing the same fate
 - created together
 - running on same node
 - terminating together
 - one network address
 - shared volumes

PROCESSES IN CONTAINERS

One Process  One Container

Minimize the footprint

- Small base OS
- No OS at all
 - Statically linked processes
 - Go
 - Traveling Ruby
- Separate Debugging
 - Developer Base Image
 - Test and Production Images
 - Inheritance with FROM:

PROCESSES IN CONTAINERS

Paradigm:

ONLY ONE PROCESS PER CONTAINER

No Dogma:

Does not work with your Legacy App?

This is not a shame!

CONTAINER PATTERNS

Brian Burns, David Oppenheimer:

Design patterns for container-based distributed systems

https://www.usenix.org/system/files/conference/hotcloud16/hotcloud16_burns.pdf

- Sidecar : Logging, Monitoring
- Ambassador: Distributed Caching
- Multi Node Application
 - Leader Election
 - Work Queue
 - Scatter/Gather

FROM VMs TO PODS

VM cluster  Pods running on Kubernetes

- cattle: stateless containers
- pets: databases

configuration management  separation of build time
and run time

STATELESS AND STATEFUL SERVICES

- where to keep state? A trade-off
 - provider → lock-in
 - self-managed → overhead
- cattle, no pets
- mindset: ephemeral deployment units

FRONT END AND BUSINESS LOGIC

- Migrate frontend to a stateless, load-balanced Kubernetes service
- Make everything explicit
 - Firewall and load-balancer
 - front-ends
 - web
 - mobile
 - native
 - embedded
 - IoT
 - TV
 - caching
 - business logic
 - persistence

STEP 3: STANDARDISED DEPLOYMENT PIPELINE

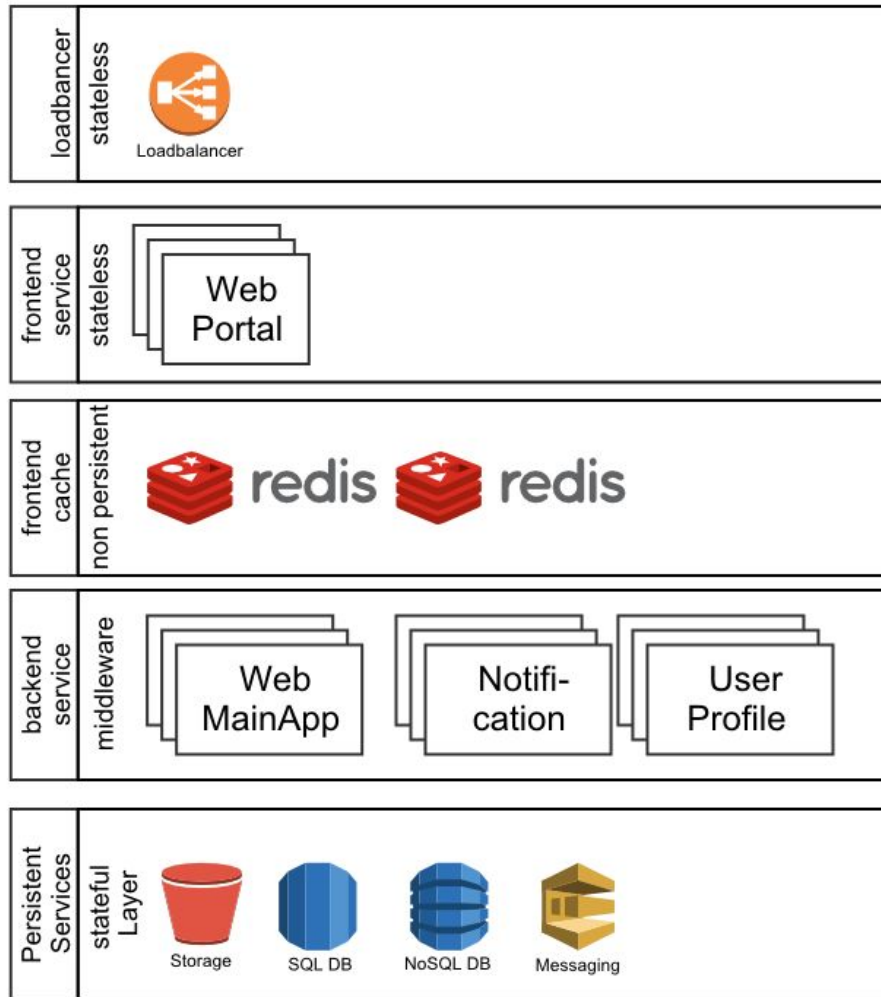
- dev/test/prod, more stages possible (QA, ...)
 - Services, labels
- parametrization
 - etcd
 - environment variables
 - secrets in kubernetes
- logging (rsyslog, ELK, splunk)
 - not every utility needs to be container specific
- measurements
 - f.e. prometheus metrics (easy to integrate in apps and services)

CONTD: FRONT END AND BUSINESS LOGIC

- Avoid privileged 'special' applications
 - application server
 - LAMP stack
- separating concerns
 - web Interface
 - application service
 - scalable through parallelism

ARCHITECTURE WRAP UP

- Desired Architecture
- Cleanups
- Ready to Rock



CASE STUDY

immmr - one number for every need

immmr combines the best of Internet base communication with the advantages of mobile communication

immmr makes it possible to use a single mobile number from any device



immmr - one number for every need

Coming later in 2016:

Launch as an independent, open communications service for voice, messaging and video telephony in the second half of 2016.

The service developed by immmr GmbH, a subsidiary of Deutsche Telekom in Berlin, is currently being tested in selected European countries.

<http://www.immmr.com/>

FROM THE TRENCHES

- Easy:
 - Java with SpringBoot
 - Python
- Hard:
 - Ruby Gems
 - Separation
 - **build**
 - **deployment**
 - no compiler in production
 - change to a static Ruby binary **traveling ruby**
 - adapt to database supported by your cloud provider
 - ruby version hell: rvm

FROM THE TRENCHES

- Lessons learned preparing for a **security audit**:
- this needed to be done anyway
- separation of stateless and persistent services is a good idea anyway and with containers really important
 - Dockerfiles need careful design to be fast
 - private registry for images recommended (same region)
 - quay.io
 - container life cycle monitoring
 - CVE database

RESULTS AND EXPERIENCES

- Scalable, kubified application
 - Service architecture as it always should have been :-)
- Reduced technical debt and implicit knowledge
- Standardised processes and APIs for services management
 - Previously, practises varied between projects
- Pod as deployment unit, single process per container
 - Pods are containers sharint the same fate
- Service as load-balanced entry point
 - external service
 - no LB cluster hassle
- smaller deployments

BUSINESS VALUE

- faster deployments:
 - faster time to market
 - more and faster testing
 - more teams possible
 - faster deployment
 - better quality
- less maintenance in operations
 - less load
 - simpler deployments

RESULTS AND EXPERIENCES

Separation of build-time and run-time

- PODs should require only minimal parametrization for being deployed
 - Secrets
 - Environment variables
- Ongoing debate on role of configuration management, our assumption:
 - Configuration management is a build-time issue
 - It should not be deployed with the container

LESSONS LEARNED

The real world is physical and limited

Do not forget to set the resource limits!

```
resources:  
  limits:  
    cpu: 300m  
    memory: 200Mi  
  requests:  
    cpu: 200m  
    memory: 100Mi
```

**SUCCESS, CHALLENGES,
'WHAT IS MISSING'**

CONTAINER LIFECYCLE MANAGEMENT

Build-time related

- Audits, scanning of container content in the registry
- Management of ephemeral configuration
(as in regular scheduled updates of keys, ...)
 - Stop-gap: rebuild container often, deploy new versions
- Leaner containers
 - immutable containers on immutable CoreOS
 - incredibly shrinking deployments

CONTAINER LIFECYCLE MANAGEMENT

Runtime related

- Monitoring of pods, containers and apps/processes
- Lifecycle management
- Cleanup of nodes (minions) after POD end-of-live
 - Issue with multi-tenant readiness
 - Clean-up, ... - issue of isolation beyond individual process (in container)

BEST PRACTISES & SIDE EFFECTS

Best practice for deployment pipelines/continuous delivery

- The last thing that is still mostly hand-made for each project
- Often violates **infrastructure is code** paradigm

Side effects of rolling updates

- Database migrations
- Difficult to roll back, structural changes stay behind or require global lock
- Help is on the way (No SQL, Flyway, Liquibase)

DOES IT SCALE IN REAL LIFE?

YES

- Scaling by country
- Or single-tenant and multi-tenant use cases
- Learning by Scaling

WANT TO DO IT YOURSELF?

ENDOCODE

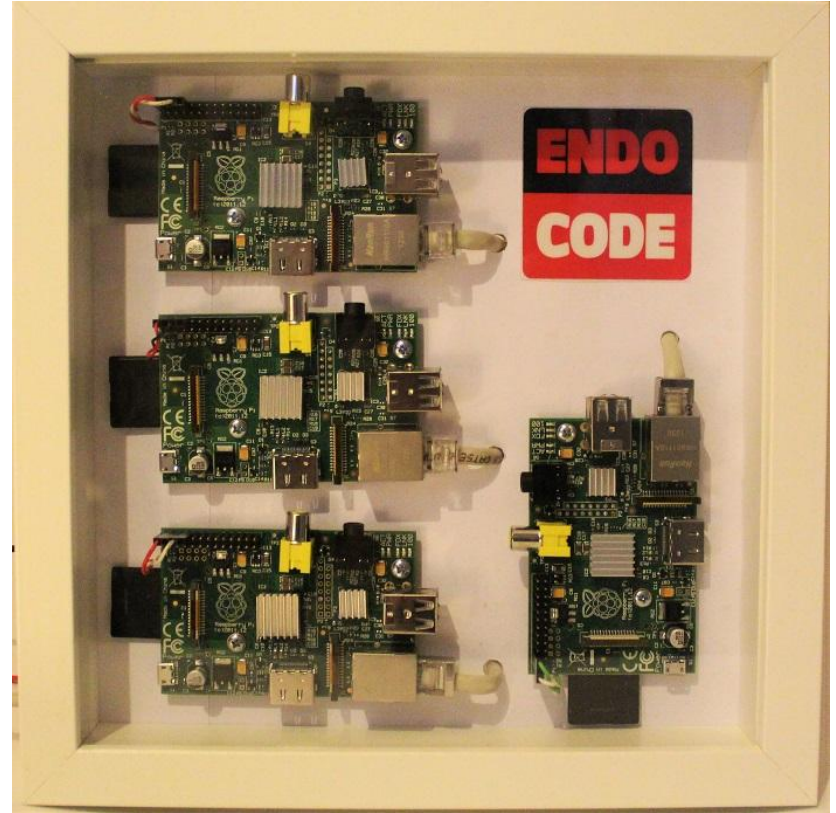
YOUR PRIVATE KUBERNETES DATACENTER

You need providers for:

- Storage
- Network
- Firewalls

<https://endocode.com/blog/2016/01/29/endocodecfgmgtcamp/>

ENDOCODE



MORE FROM ENDOCODE

- <https://endocode.com>
- <https://endocode.com/blog/>
- <https://endocode.com/trainings-overview/>
- Visit us on GitHub
<https://github.com/endocode>



Google Cloud Platform

ENDOCODE

Dive into Kubernetes!

Watch our Webinar 'Dive into Kubernetes' on our YouTube Channel

<https://youtu.be/8694GGJlpZ8>

Register for a free Google Cloud Platform Trial with \$300 Google Cloud Platform Credits

<https://goo.gl/dUzDWi>

Use another \$200 partner credits

<https://goo.gl/eYldnT>



QUESTIONS?