OpenShift 3 Techlab

1. Ziele
2. Container
3. OpenShift 3
4. Workshop

Agenda

1

Techlab

Ziele des Techlabs

- Gemeinsamer Einstieg in neue, moderne Technologie
- Grundkonzepte verstehen
- Erste Applikation deployen

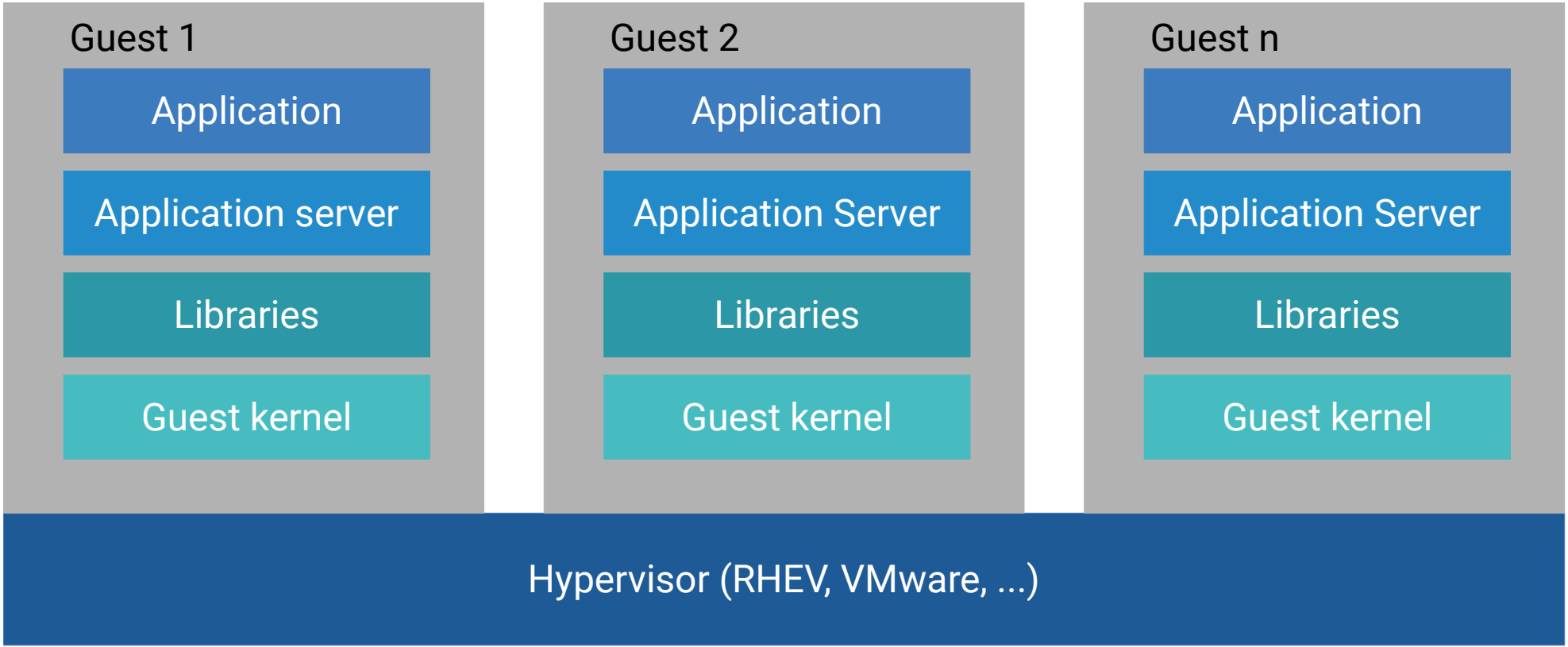
2

Was sind Container?

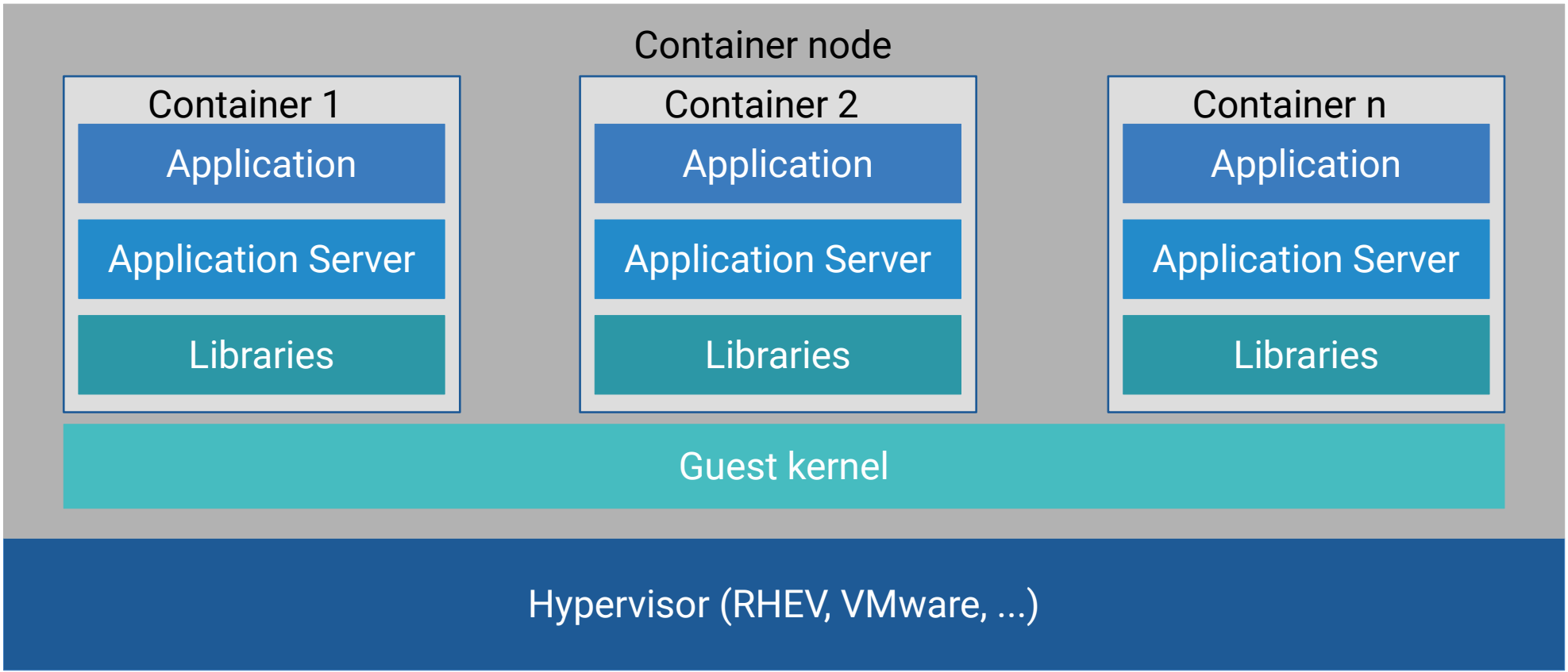
Nichts neues!

LXC, VServer, FreeBSD Jails, Google…

Klassische Virtualisierung

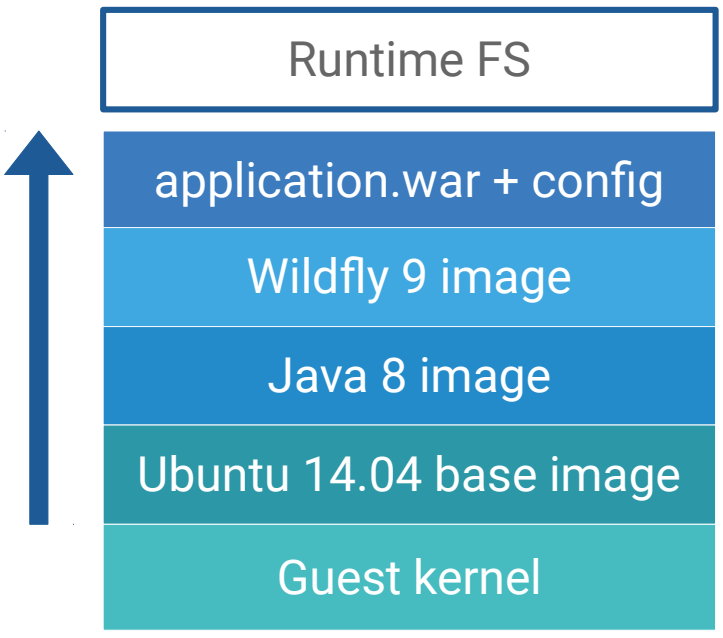


Container Virtualisierung



Docker Container

Hierarchische Dockerfiles



read-write

read-only

```
# buildet das Dockerfile unter ./
$ docker build .
# container von image app starten
$ docker run -p 8080:8080 app
```

Container und deren Infrastruktur

- Container sind unveränderbar
- Aktualisierung eines Containers erfolgt durch austauschen
  - sowohl bei Applikations- als auch System-Updates
- Kein lokales Filesystem für Applikationsdaten
- Persistent Storage

3

OpenShift V3

Was ist OpenShift 3?

“Next generation PaaS” OpenShift Enterprise V3 von Red Hat

Neu-Implementation: V2 wurde verworfen und komplett neu gebaut

V2	**V3**
Namespace/Domain	Project
Gear	Docker Container
Cartridge	Docker Base Image
rhc	oc (OpenShift Client)

OpenShift basiert auf etablierten Open Source-Konzepten

Docker

Kubernetes

OpenShift 3 (1/2)

Container Platform as a Service (PaaS)

Multinode Platform, um Applikationen in Containern zu betreiben

One platform runs it all!

Fancy CLI und GUI (self-service)

OpenShift 3 (2/2)

Base Images für RHEL 6 und 7 patched analog Standard RHEL

Redeployments werden durch Basis Image Update getriggert

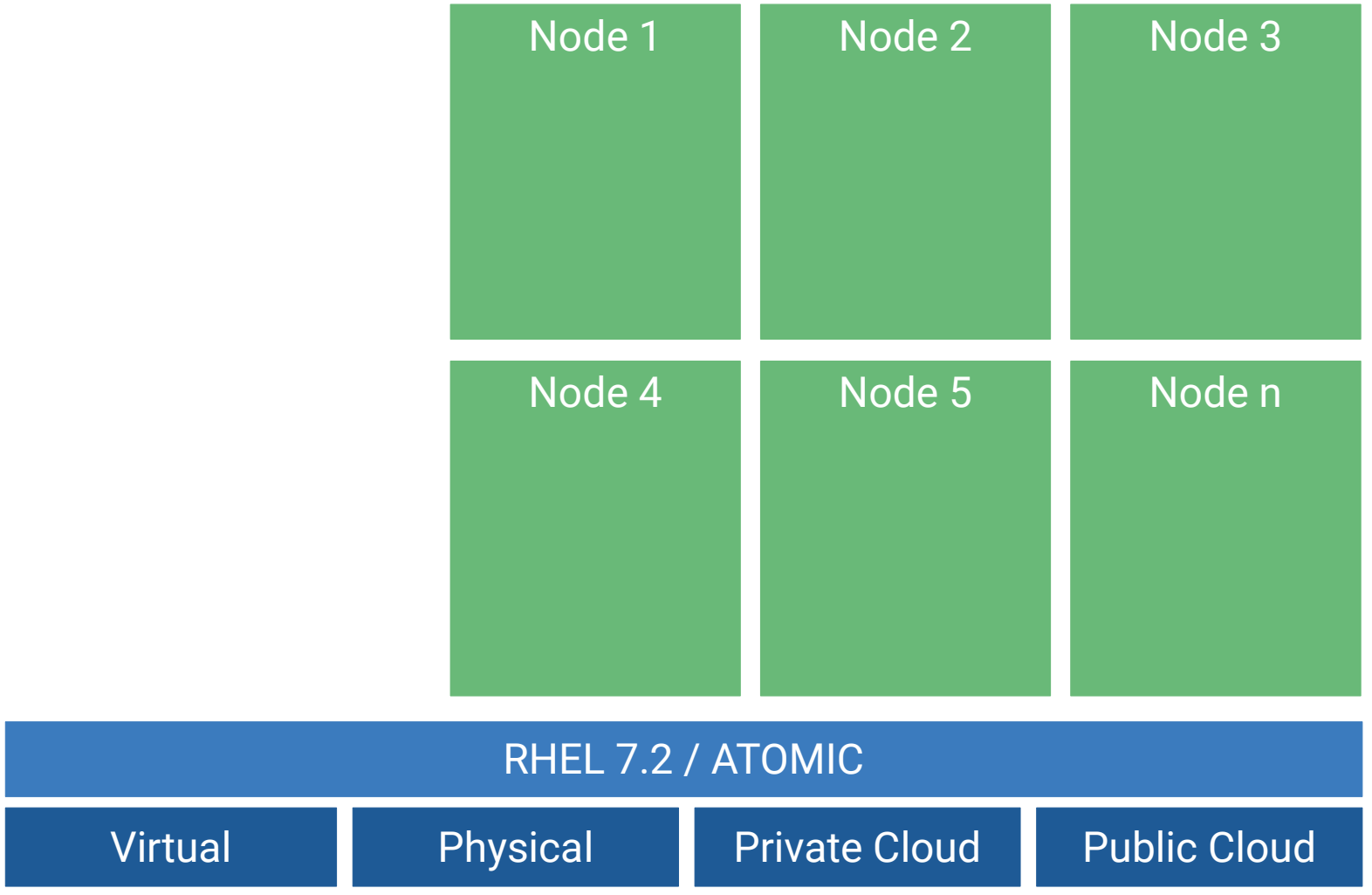
Standard Deployment Mechanismus und Workflow

Autoscaling

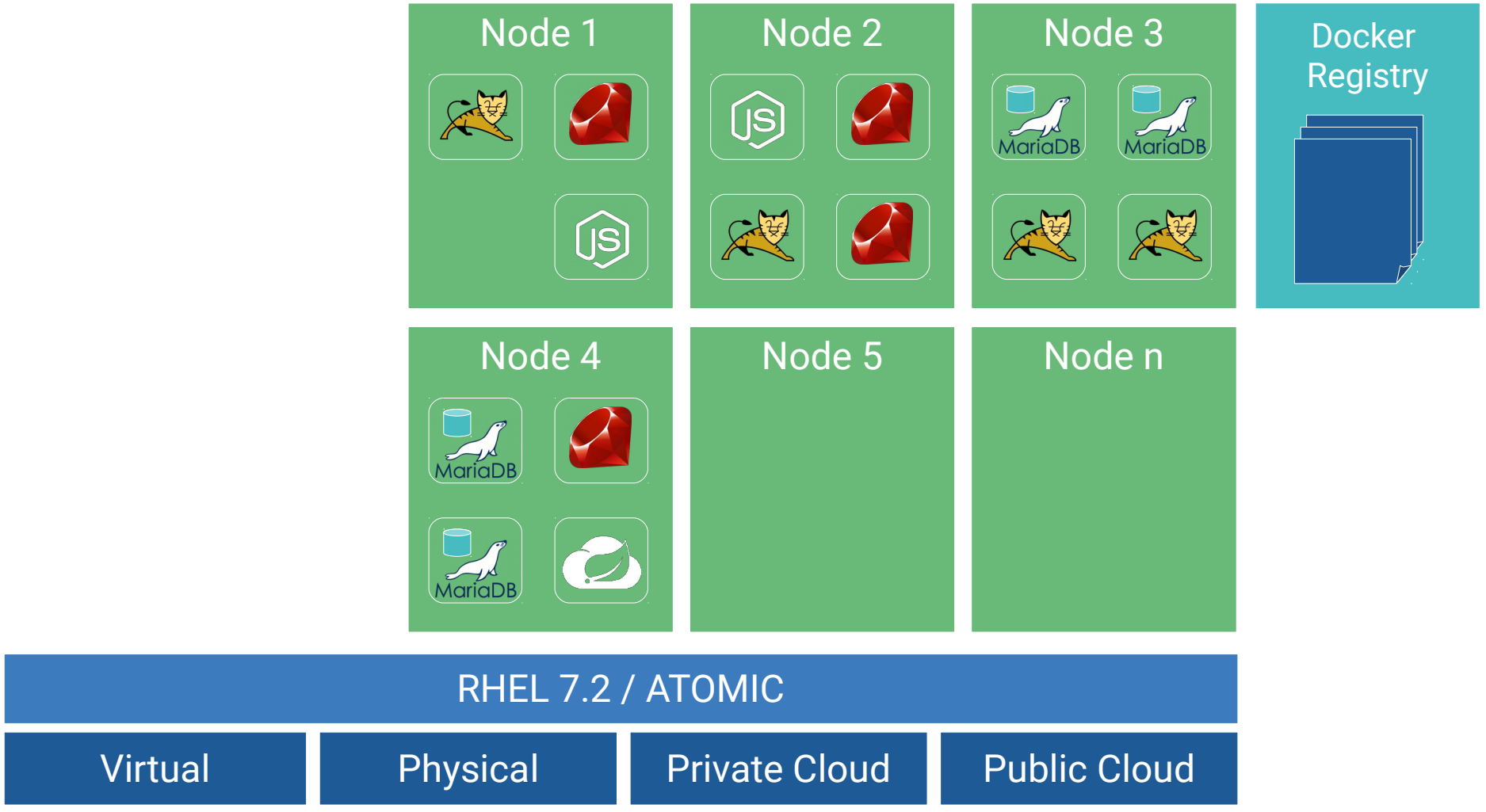
Container Security

RHEL 7.2 als Basis

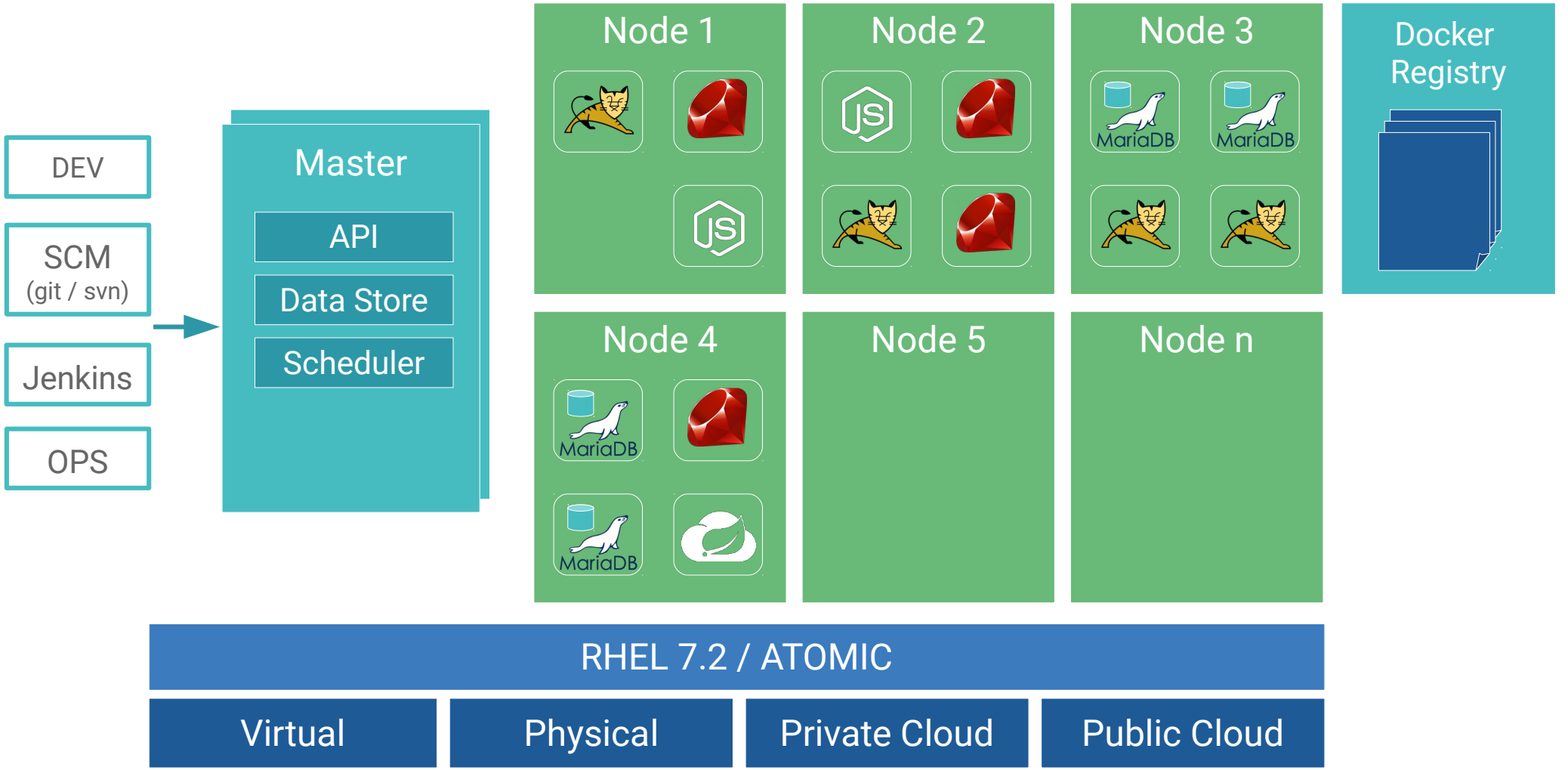
Compute Nodes



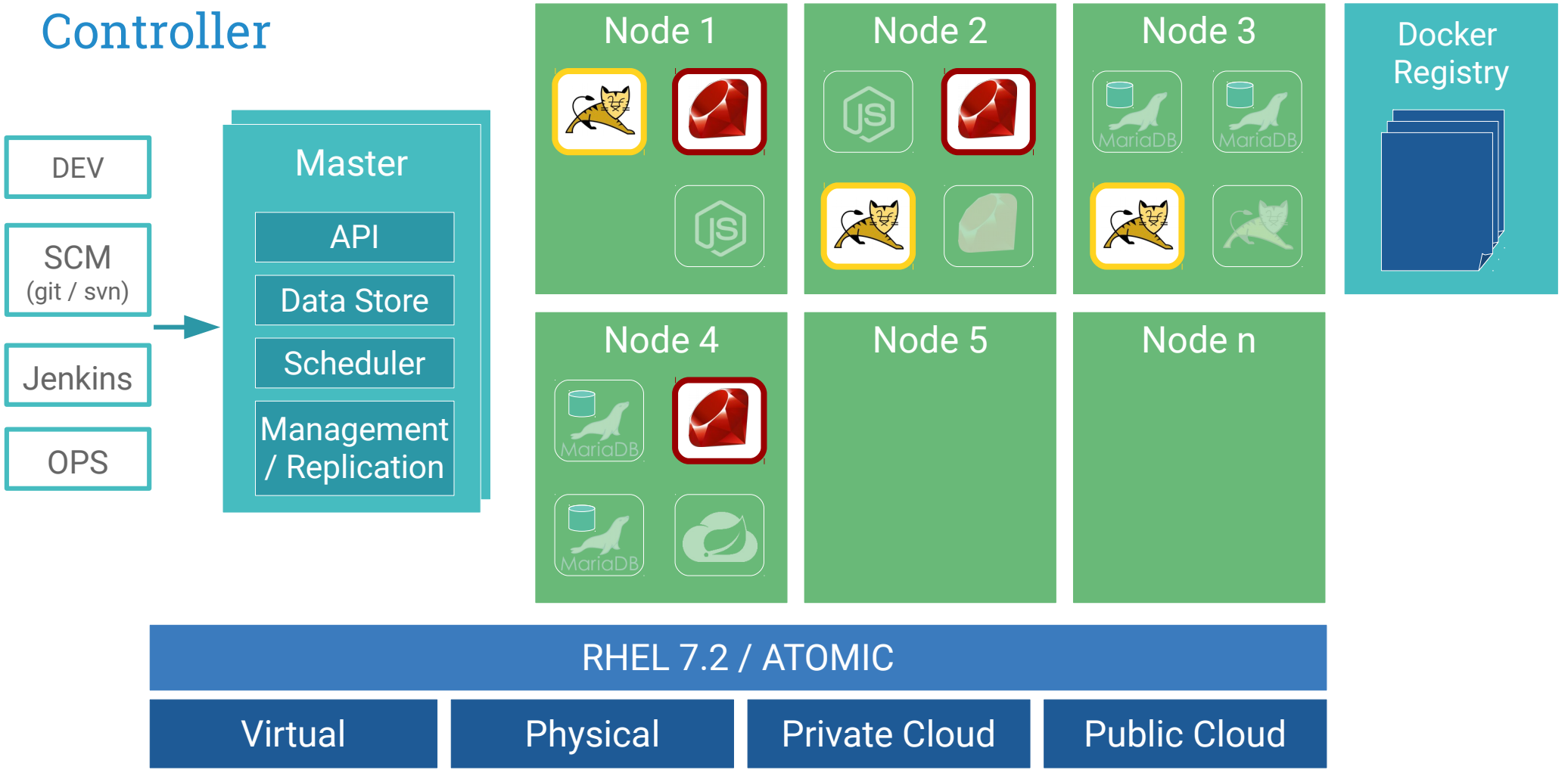
Pods


Node 1
Node 2
Node 3
Docker Registry
MariaDB
MariaDB
Node 4
Node 5
Node n
MariaDB
MariaDB
RHEL 7.2 / ATOMIC
Virtual
Physical
Private Cloud
Public Cloud

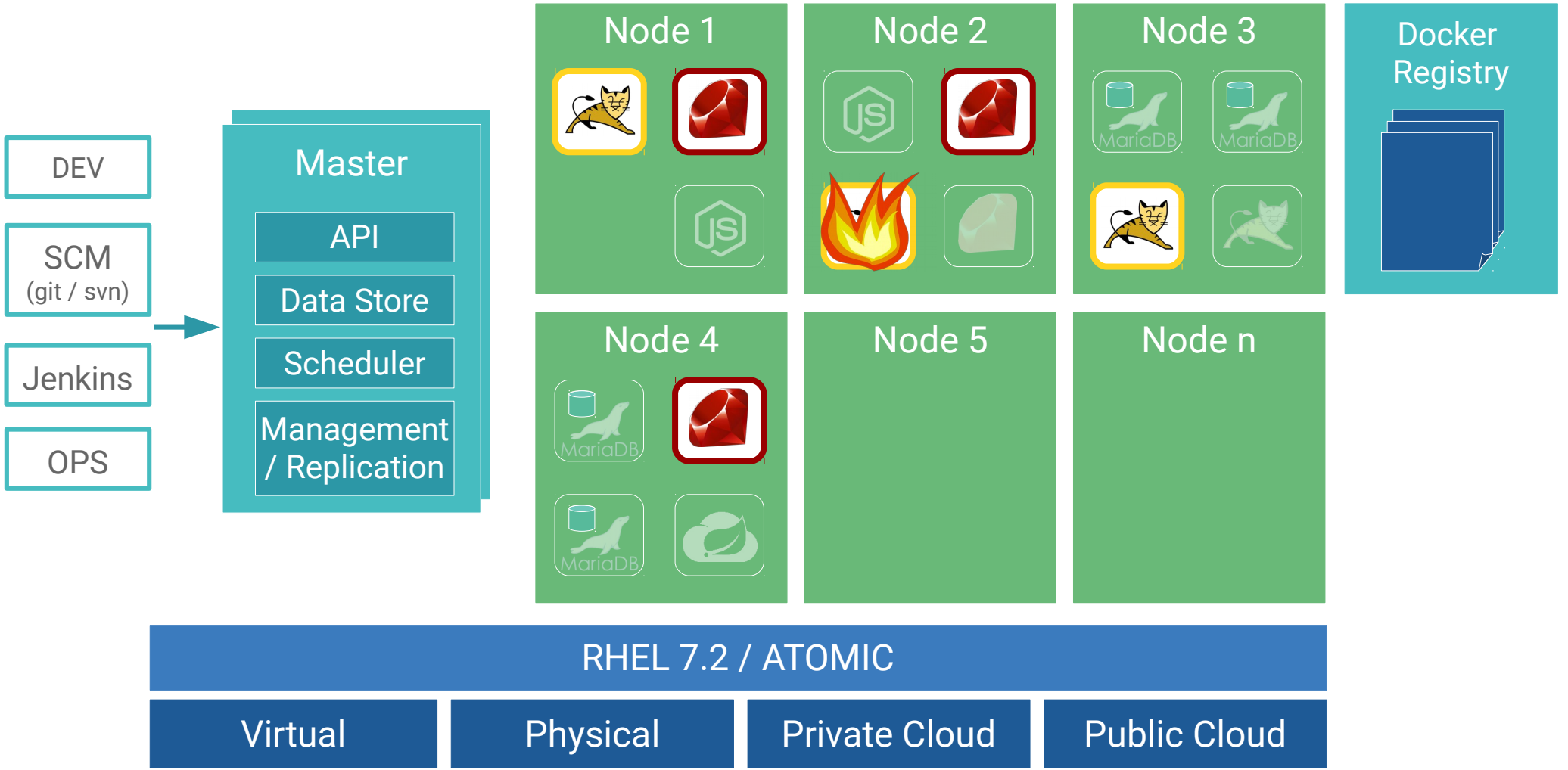
OSE 3 Master


DEV
SCM
(git / svn)
Jenkins
OPS
Master
API
Data Store
Scheduler
Node 1
Node 2
Node 3
Node 4
Node 5
Node n
MariaDB
MariaDB
MariaDB
Docker
Registry
RHEL 7.2 / ATOMIC
Virtual
Physical
Private Cloud
Public Cloud

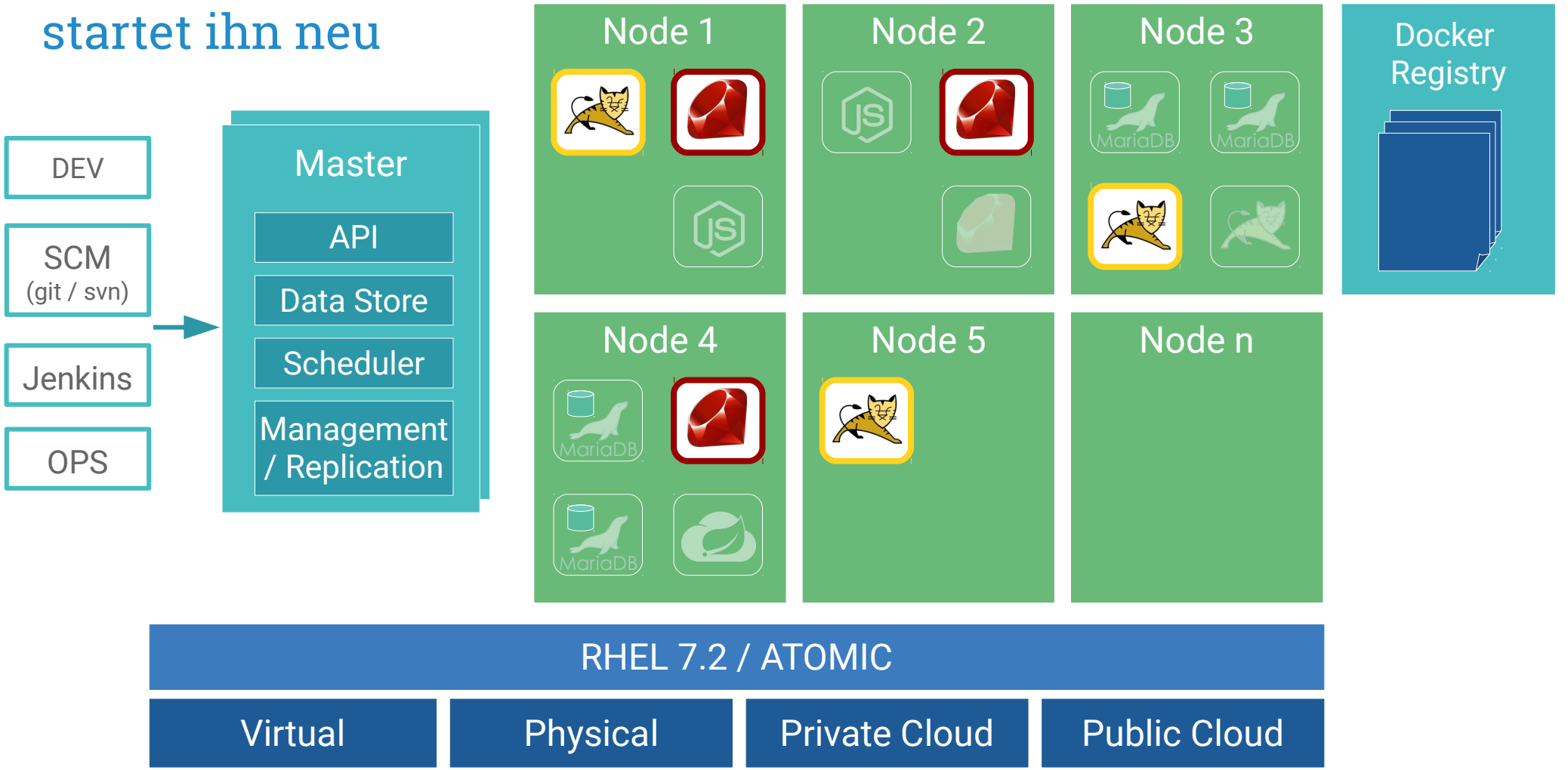
Replication Controller



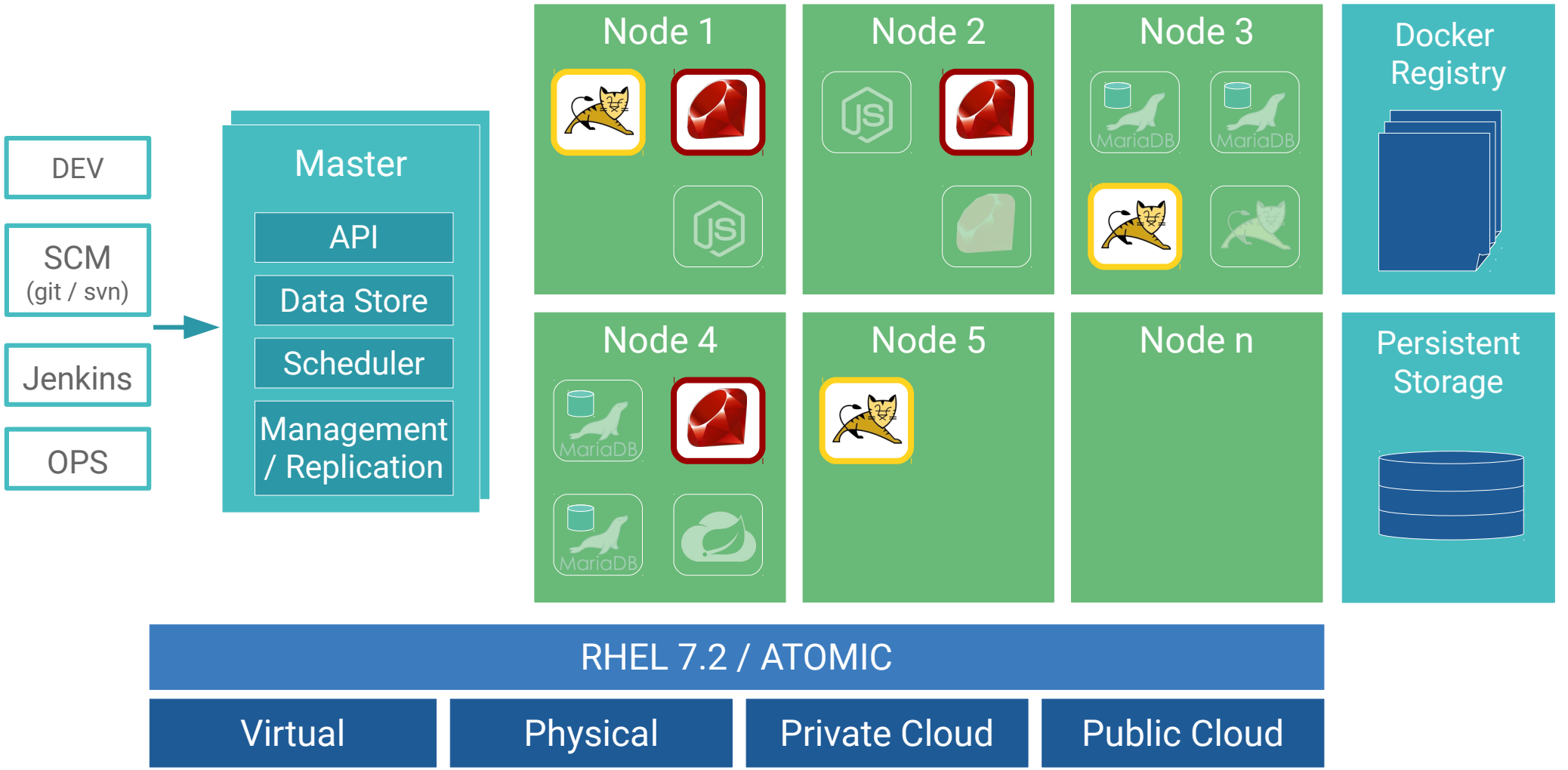
Pod fackelt ab...



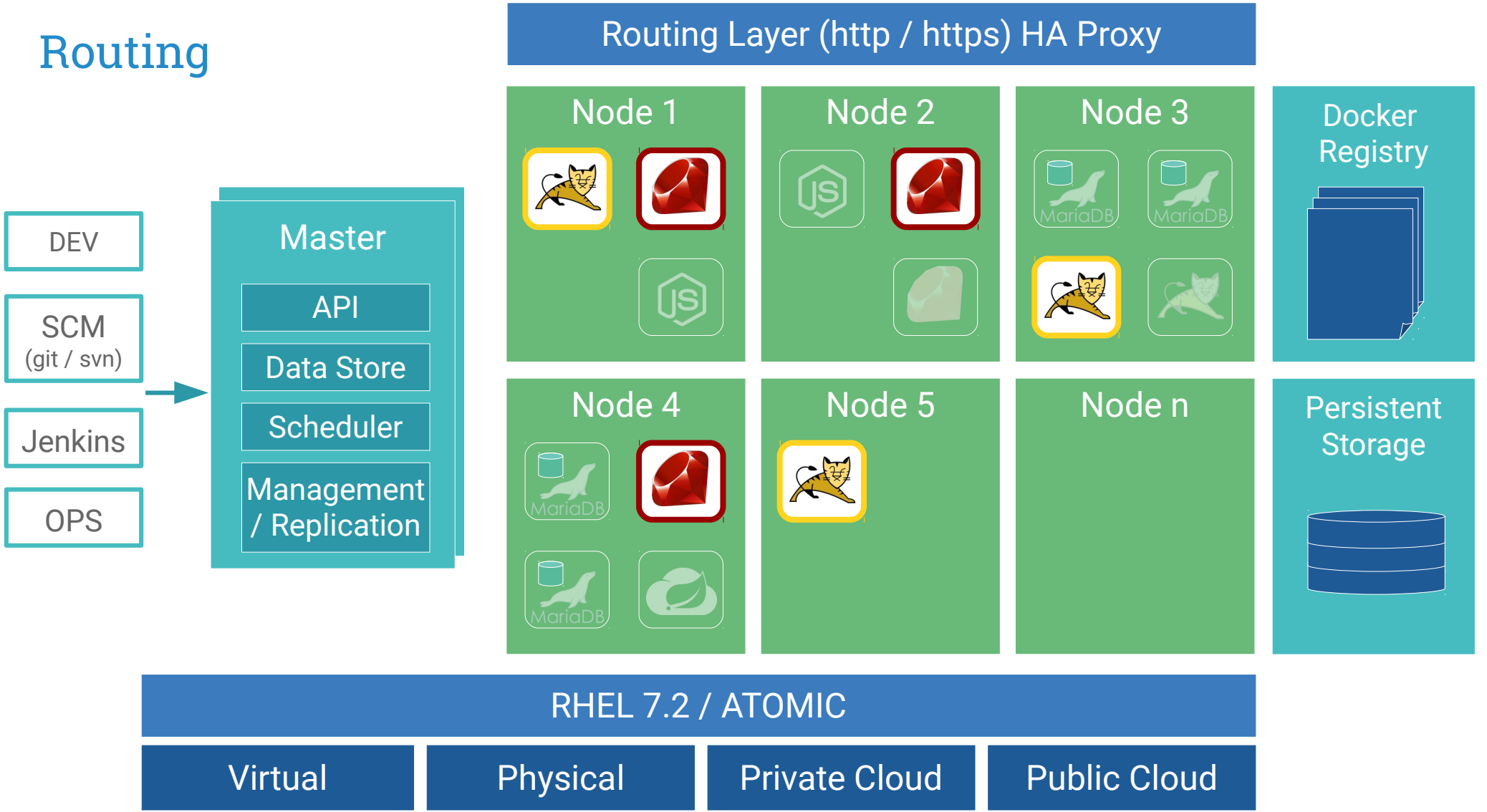
… Kubernetes startet ihn neu



Persistent Storage



Routing


Routing Layer (http / https) HA Proxy
Node 1
Node 2
Node 3
Docker Registry
Node 4
Node 5
Node n
Persistent Storage
MariaDB
MariaDB
MariaDB
DEV
SCM (git / svn)
Jenkins
OPS
Master
API
Data Store
Scheduler
Management / Replication
RHEL 7.2 / ATOMIC
Virtual
Physical
Private Cloud
Public Cloud

Beispiel Java Projekt

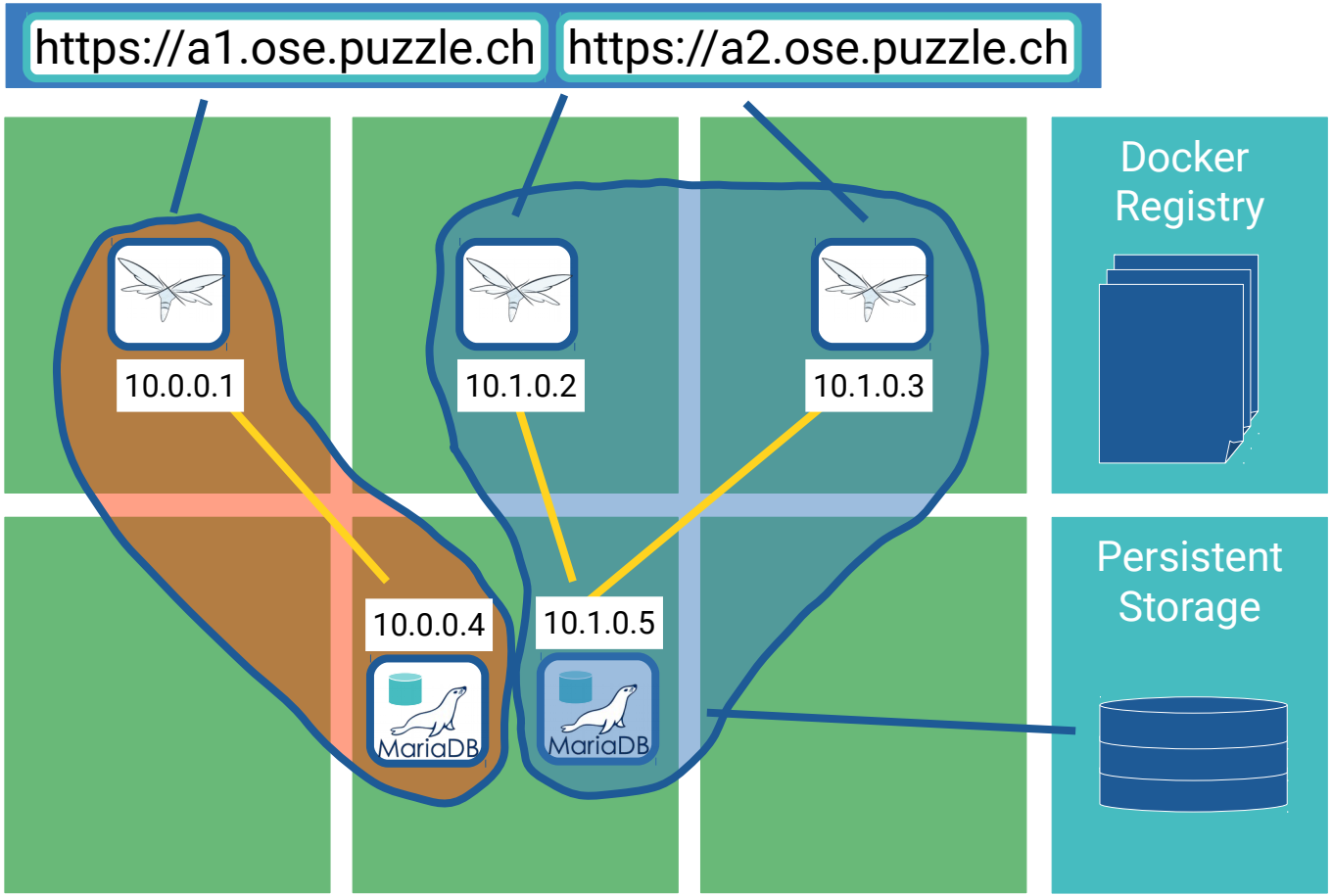
- War deployed in Wildfly 10
- Maria DB (mit Persistent Volume)

Beispiel OSE 3 Projekte

Via Route im Netz verfügbar

Loadbalancing auf Pods

Verbunden über transparentes Software Defined Network (SDN)



RHEL 7.2 / ATOMIC

Virtual | Physical | Private Cloud | Public Cloud

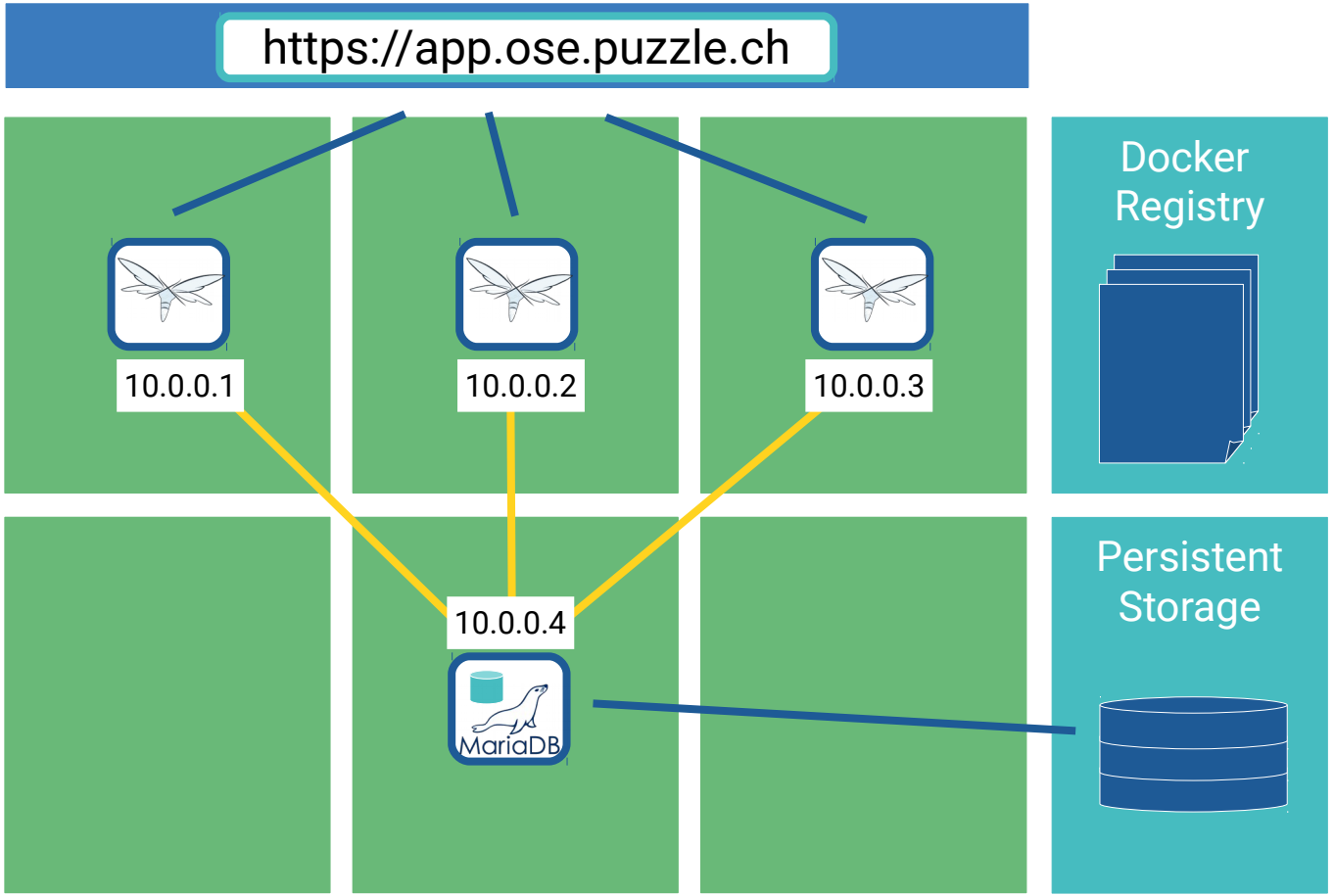
Beispiel Java App
OSE 3 Projekt (1)

Via Route im Netz verfügbar
Loadbalancing auf Pods

oc scale dc app --replicas=3

Verbunden über
transparentes Software
Defined Network (SDN)

Projekt-Setup als JSON
exportier- / importierbar

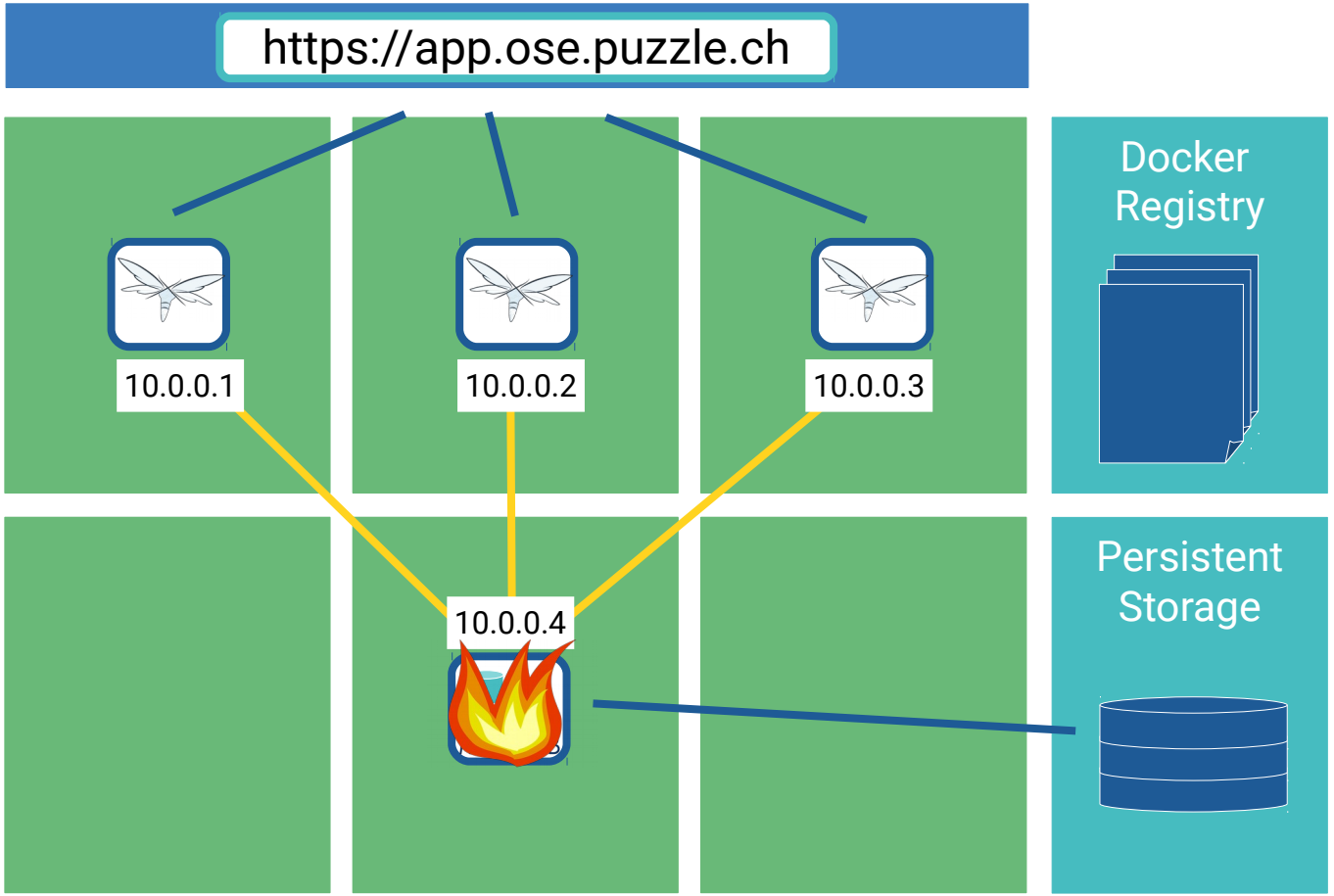


Virtual | Physical | Private Cloud | Public Cloud

RHEL 7.2 / ATOMIC

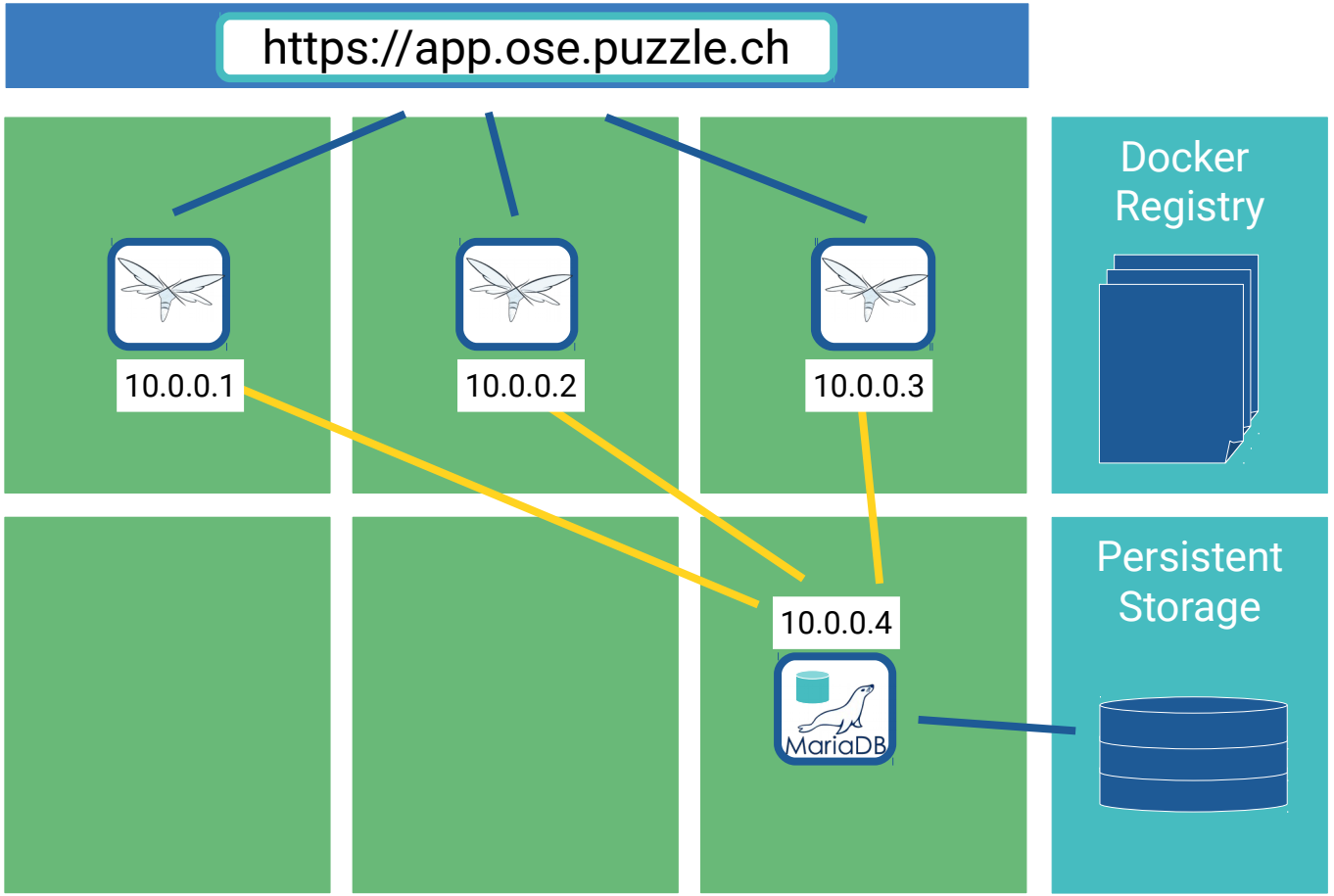
Beispiel Java App
OSE 3 Projekt (2)

Maria DB-Pod failed



Beispiel Java App
OSE 3 Projekt (3)

Pod wird neu gestartet
- Gleiche Adresse
- Persistent Volume wandert mit



Virtual | Physical | Private Cloud | Public Cloud

RHEL 7.2 / ATOMIC

4

Workshop

Resources

DataSheet:

https://www.redhat.com/en/files/resources/cl-openshift-enterprise-3-red-hat-inc0328839mm-201512.pdf

Dokumentation:

https://docs.openshift.com

Resources:

https://www.openshift.com/enterprise/resources.html

Getting Started:

https://docs.openshift.com/enterprise/3.1/cli_reference/index.html

Thank you!

2

Anwendungsbeispiele

Buildinfrastruktur für Applikationen

Java, JavaScript, Ruby on Rails, Node, ...

Installation von Buildtools in verschiedenen Versionen für unterschiedliche Projekte

Exakte Umgebung die explizit für Applikation definiert wird

Wiederverwendbar, schnell, isoliert.

Java EE 7 Applikation Wildfly 9

War und Config ins Image hinzufügen und Go!

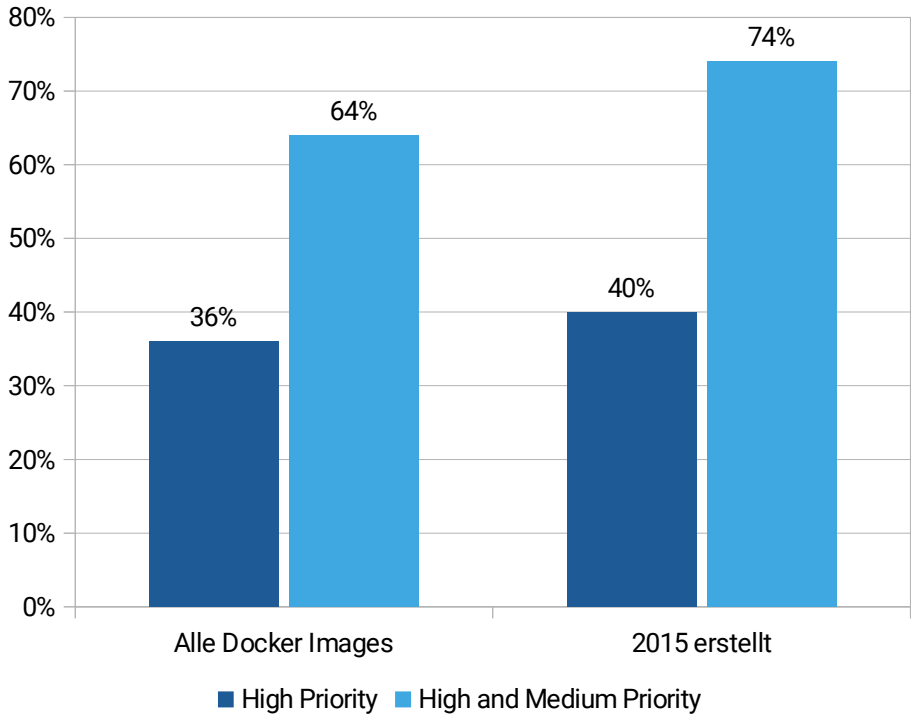
```
FROM jboss/wildfly

ADD app-web.war /opt/jboss/wildfly/standalone/deployments/
ADD standalone.xml /opt/jboss/wildfly/standalone/configuration/
```

Chancen/Herausforderungen

Vorteile von Containern

- Leichtgewichtig und schnell
- Standardisierung
- Einfach zu gebrauchen und zu erweitern
- Grosse Community DockerHub, Tool Ökosystem
- Vielzahl an Docker Images verfügbar
- Kann «Works on my Machine» Probleme ausmerzen

Herausforderungen: Sicherheit (1/2)



64 % aller Docker Images auf Docker Hub der offiziellen Repositories haben Sicherheitslücken:

- ShellShock (bash)
- Heartbleed (OpenSSL)
- Poodle (OpenSSL)
- ...

Quelle: http://www.banyanops.com/blog/analyzing-docker-hub/
Mai 2015: Jayanth Gummaraju, Tarun Desikan and Yoshio Turner

Herausforderungen: Sicherheit (2/2)

- Prozesse behandeln als würden sie auf Host laufen
- Prozesse nicht als root laufen lassen
- Nur notwendige Ports öffnen
- SELinux oder AppArmor im Container einsetzen

Quellen: https://opensource.com/business/14/7/docker-security-selinux
http://www.pro-linux.de/news/1/22315/docker-erh%C3%B6ht-die-sicherheit.html

Herausforderungen: Betrieb