ORACLE
Cloud Infrastructure

OKE – Kubernetes in OCI

Your First Journey in Oracle Cloud World

Brazil Infrastructure and Security Team
LAD Knowledge Team

May/2020

Safe Harbor

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Statements in this presentation relating to Oracle's future plans, expectations, beliefs, intentions and prospects are "forward-looking statements" and are subject to material risks and uncertainties. A detailed discussion of these factors and other risks that affect our business is contained in Oracle's Securities and Exchange Commission (SEC) filings, including our most recent reports on Form 10-K and Form 10-Q under the heading "Risk Factors." These filings are available on the SEC's website or on Oracle's website at http://www.oracle.com/investor. All information in this presentation is current as of September 2019 and Oracle undertakes no duty to update any statement in light of new information or future events.

$> whoami

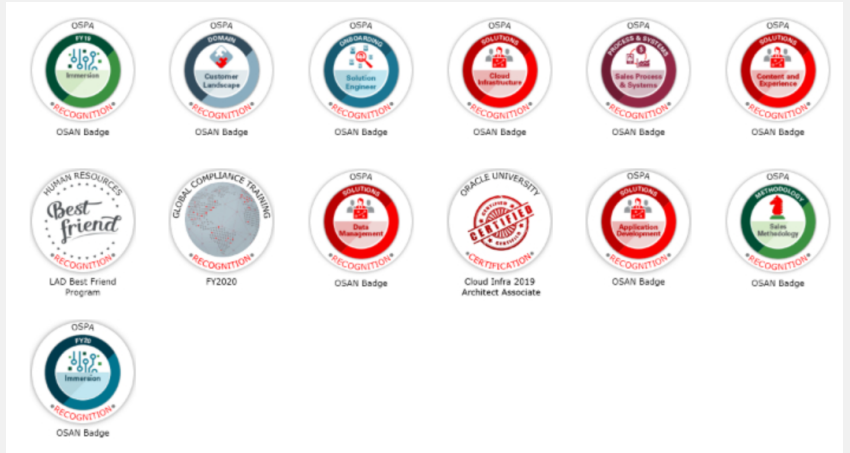
+25 years on IT Market

2.5 years

Working in Oracle Brazil

Solution Eng. Sales Consultant
LAD Team

DevOps, Cloud Native, Integration, Mobile and CRM specialist



Cristiano Hoshikawa

Cloud Solution Engineering
Oracle Cloud Team

cristiano.hoshikawa@oracle.com
https://www.linkedin.com/in/cristianohoshikawa/
https://github.com/hoshikawa2

Agenda

- Abertura
- Cloud Native Computing Foundation
- Kubernetes
- Oracle Cloud Infrastructure Registry & Container Engine for Kubernetes
- Labs
    - Oracle Cloud Infrastructure
    - Containers Docker
    - Kubernetes
    - Repositório de Imagens Docker
    - Deployments
    - Load-Balancer

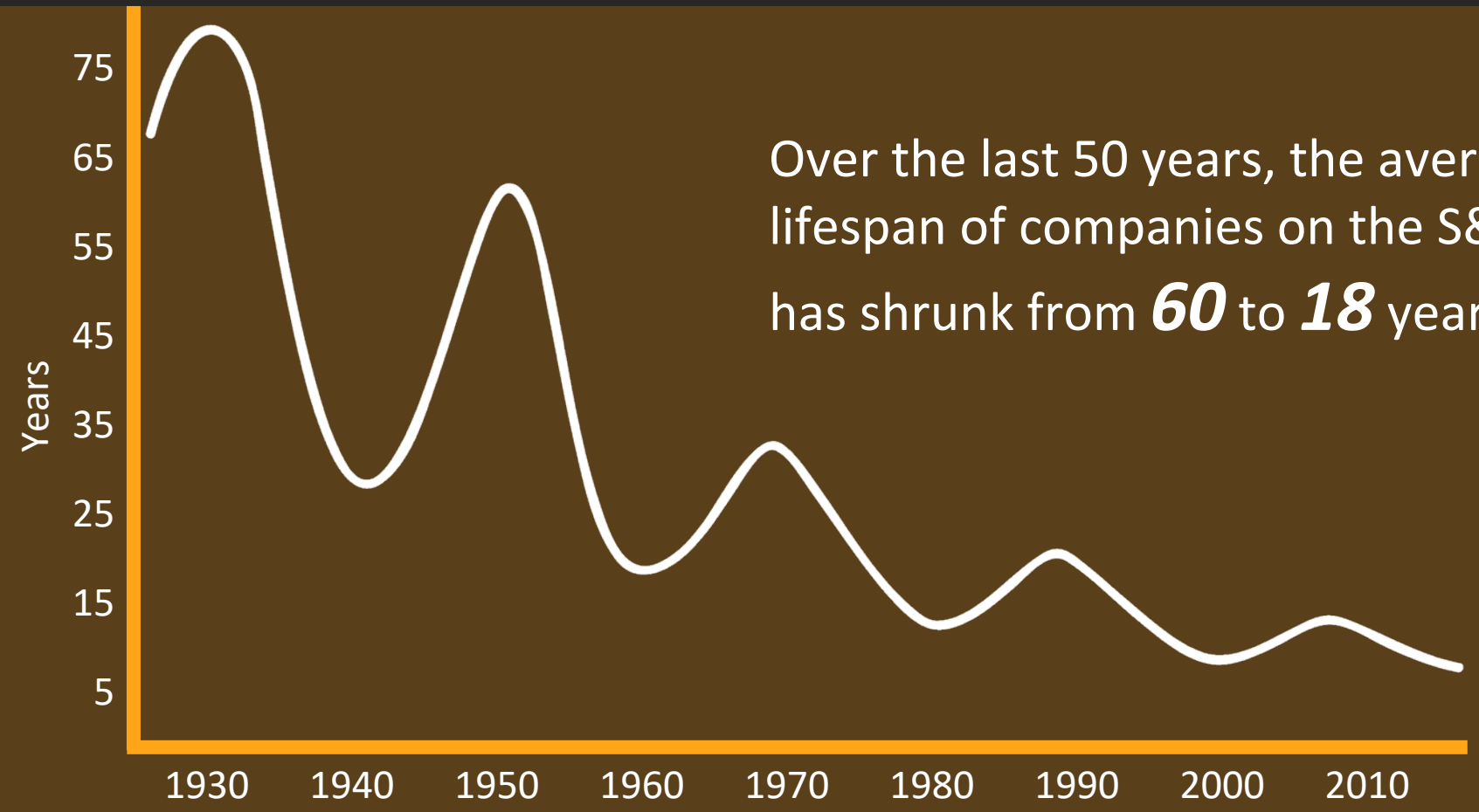
ORACLE
Cloud Platform

"The eye sees only what the mind is prepared to comprehend"

by French philosopher Henri Bergson

Market Insights

Business Cycles Are Faster Than Ever Before



Over the last 50 years, the average lifespan of companies on the S&P 500 has shrunk from ***60*** to ***18*** years

Software is How Companies Differentiate Themselves

*"**More and more major businesses and industries are being run on software and delivered as online services**—from movies to agriculture to national defense. Many of the winners are Silicon Valley-style entrepreneurial technology companies that are invading and overturning established industry structures"*

*"**As our global economy increasingly comes to run on technology-enabled rails and every company becomes a tech company**, demand for high-quality software engineers is at an all-time high. A recent study from Stripe and Harris Poll found that 61 percent of C-suite executives believe access to developer talent is a threat to the success of their business. Perhaps more surprisingly — as we mark a decade after the financial crisis — this threat was even ranked above capital constraints."*

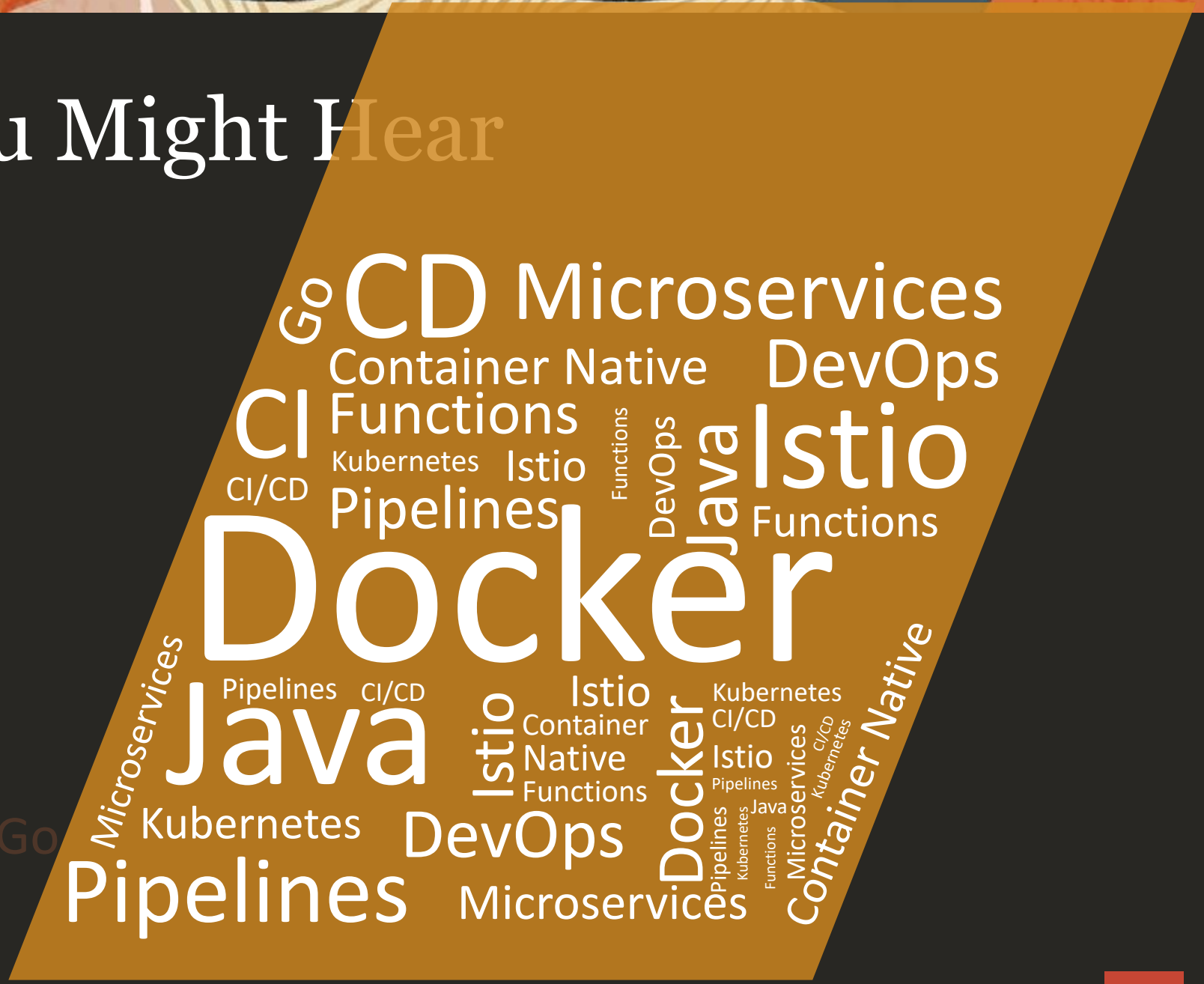
Source: Software developers are now more valuable to companies than money: Survey

Software Is Overtaking the World

Time-to-market is key to success

Buzzwords You Might Hear

- Docker
- Kubernetes
- Microservices
- Serverless / Functions
- Pipelines
- CI / CD
- DevOps
- Containers
- Container Native
- Istio
- Java / NodeJs / Python / Go

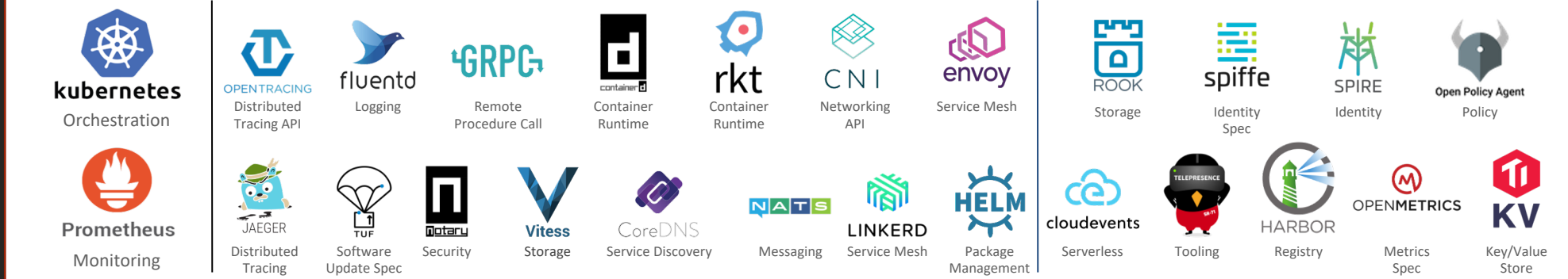


Cloud Native Computing Foundation

Overview

Cloud Native Computing Foundation

Non-profit, part of the Linux Foundation; founded Dec 2015



Platinum members:

Alibaba Cloud, aws, Azure, cisco, DELL Technologies, docker, FUJITSU, Google Cloud, HUAWEI, IBM Cloud, intel, JD.COM, MESOSPHERE, ORACLE, Pivotal, redhat, SAMSUNG SDS, SAP, vmware

Today the Linux Foundation is much more than Linux

Security

We are helping global privacy and security through a program to encrypt the entire internet.

Networking

We are creating ecosystems around networking to improve agility in the evolving software-defined datacenter.

Cloud

We are creating a portability layer for the cloud, driving de facto standards and developing the orchestration layer for all clouds.

Automotive

We are creating the platform for infotainment in the auto industry that can be expanded into instrument clusters and telematics systems.

Blockchain

We are creating a permanent, secure distributed ledger that makes it easier to create cost-efficient, decentralized business networks.

Web

We are providing the application development framework for next generation web, mobile, serverless, and IoT applications.

Let's Encrypt

LF NETWORKING

CLOUD NATIVE COMPUTING FOUNDATION

AUTOMOTIVE GRADE LINUX

HYPERLEDGER

node JS

We are regularly adding projects; for the most up-to-date listing of all projects visit tlfprojects.org

CLOUD NATIVE COMPUTING FOUNDATION

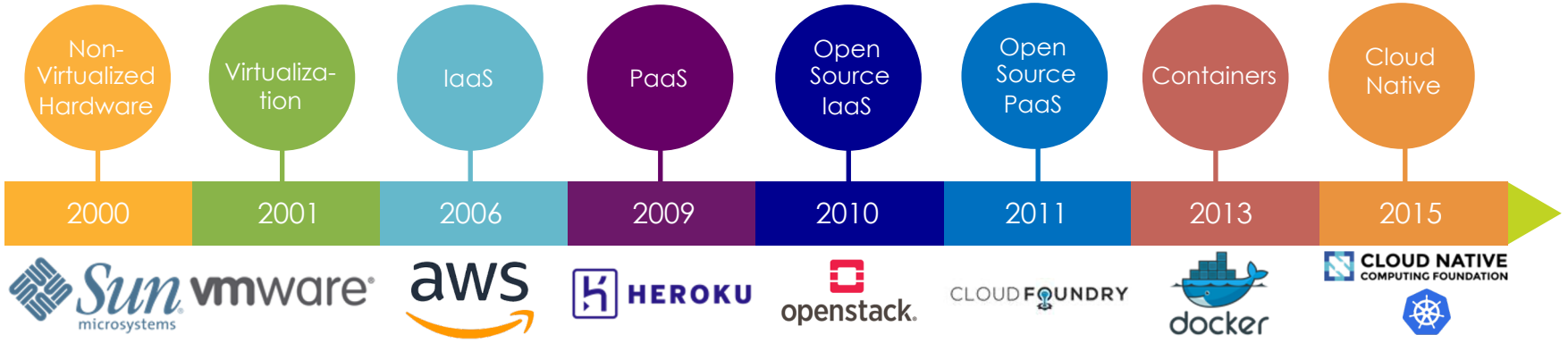
From Virtualization to Cloud Native

CLOUD NATIVE COMPUTING FOUNDATION

kubernetes

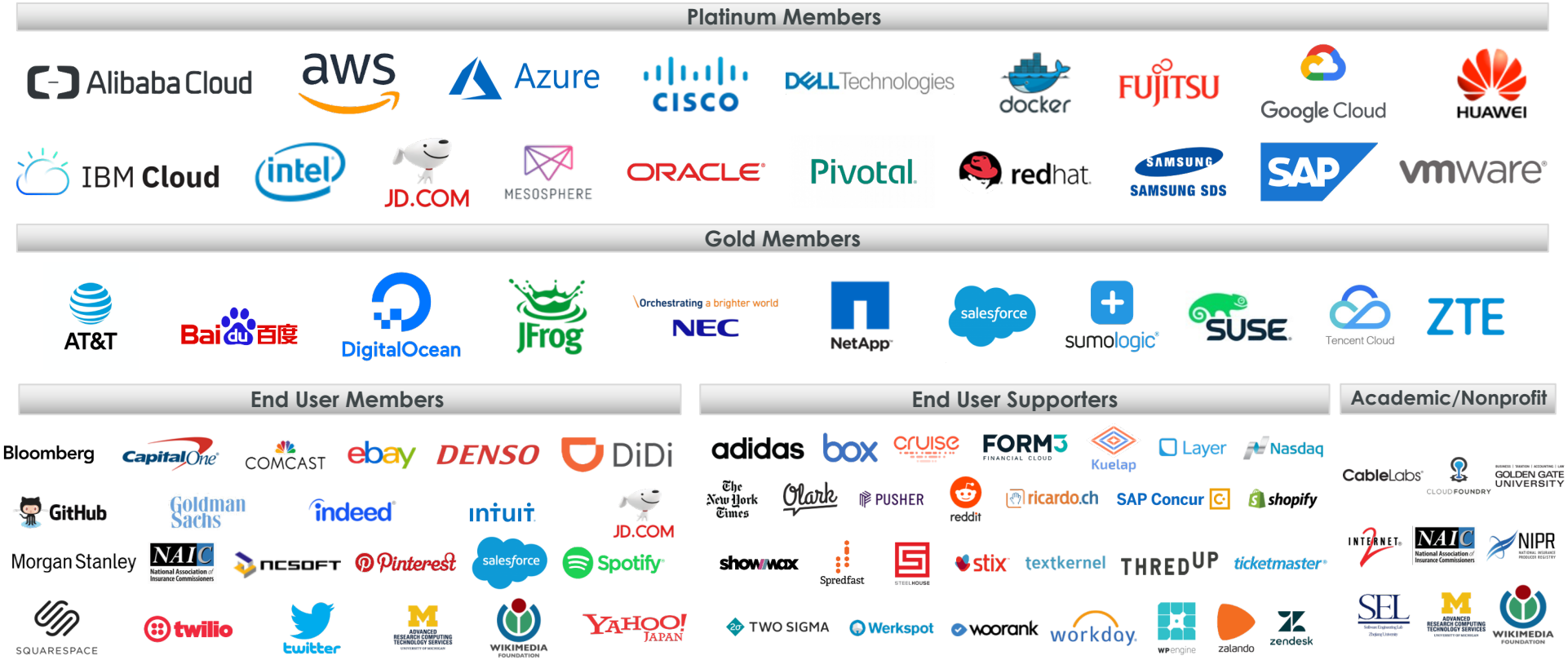
Cloud native computing uses an ***open source software*** stack to:

- **segment** applications **into *microservices***,
- **package** each part **into its own *container***
- and **dynamically *orchestrate*** those containers **to optimize resource utilization**



290+ Members and Growing

CLOUD NATIVE COMPUTING FOUNDATION



CNCF Worldwide Community

Cloud Native Computing Foundation (CNCF)

Members	Groups	Countries
79,470	155	36

Cloud Native Value Propositions

Avoid Vendor Lock-in

Open source software stack enables deployment on any public, private cloud or hybrid cloud

CLOUD NATIVE COMPUTING FOUNDATION

Enable Unlimited Scalability

Scales from several nodes on your laptop to tens of thousands of self-healing multi-tenant nodes

CLOUD NATIVE COMPUTING FOUNDATION

Increase Agility and Maintainability

By splitting applications into microservices
with explicitly described dependencies

CLOUD NATIVE
COMPUTING FOUNDATION

Achieve Resiliency

To failures of individual containers, machines, and even data centers and to varying levels of demand

CLOUD NATIVE COMPUTING FOUNDATION

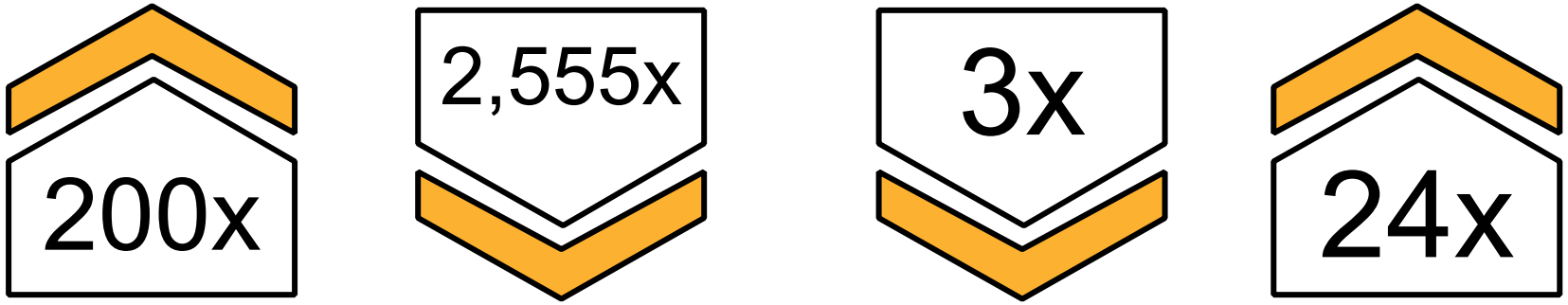
Improve Efficiency and Resource Utilization

Via a central orchestrating process that dynamically manages and schedules microservices

CLOUD NATIVE COMPUTING FOUNDATION

Enables High Performance

High-performing teams deploy more frequently and have much faster lead times.



2016 State of DevOps Report Infographic from Puppet


kubernetes
The “Captain” of yours Containers
CLOUD NATIVE
COMPUTING FOUNDATION

What is Kubernetes?

kubernetes

Kubernetes (κυβερνήτης, Greek for "governor", "helmsman" or "captain")

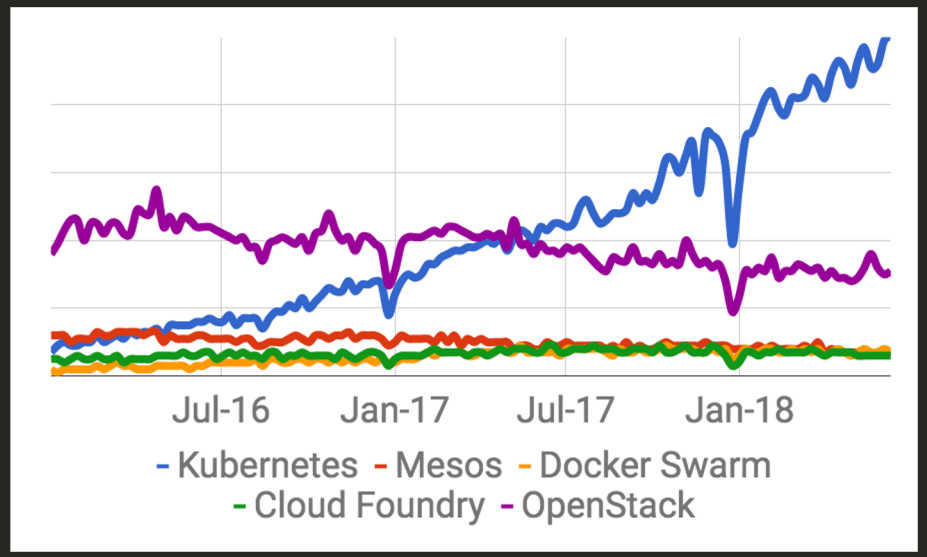
Kubernetes is a "Container Orchestrator" or "Cluster Manager".

- Places containers on nodes
- Recovers automatically from failure
- Basic monitoring, logging, health checking
- Enables containers to find each other.

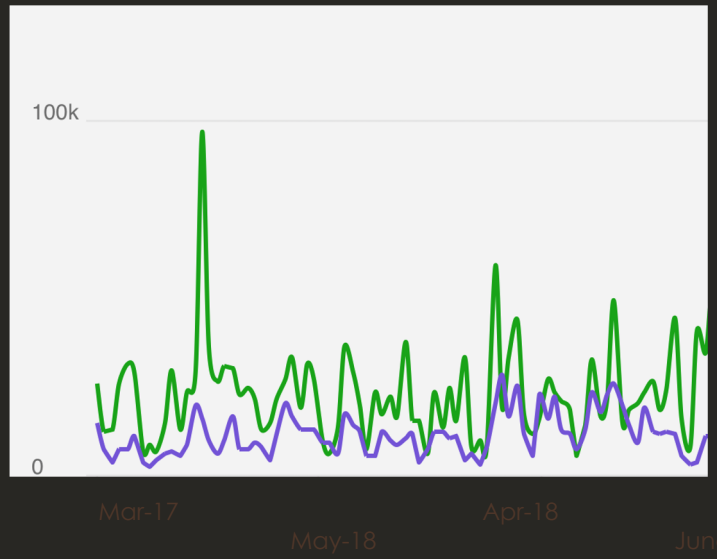
CLOUD NATIVE COMPUTING FOUNDATION

Kubernetes in Search Trends

Google Trends

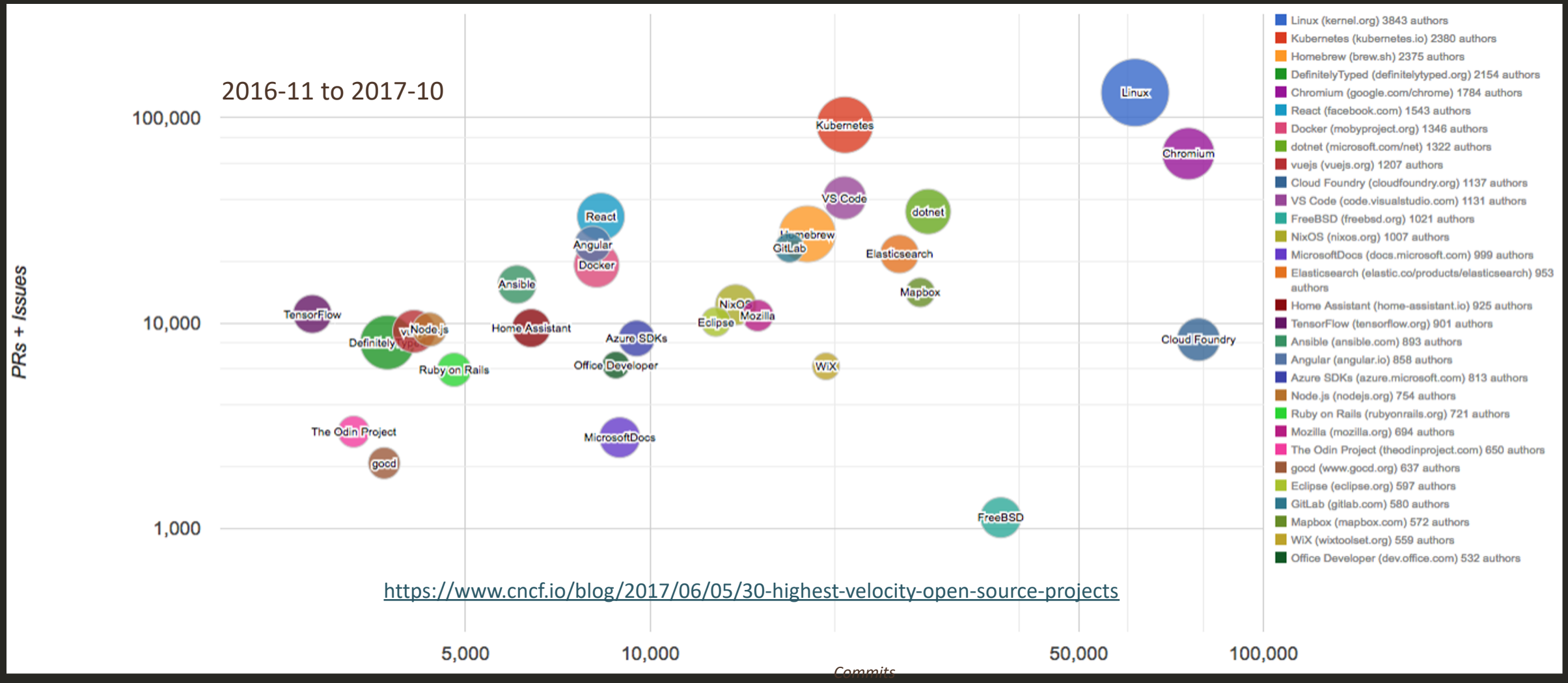


WeChat



kubernetes

30 Highest Velocity Open Source Projects

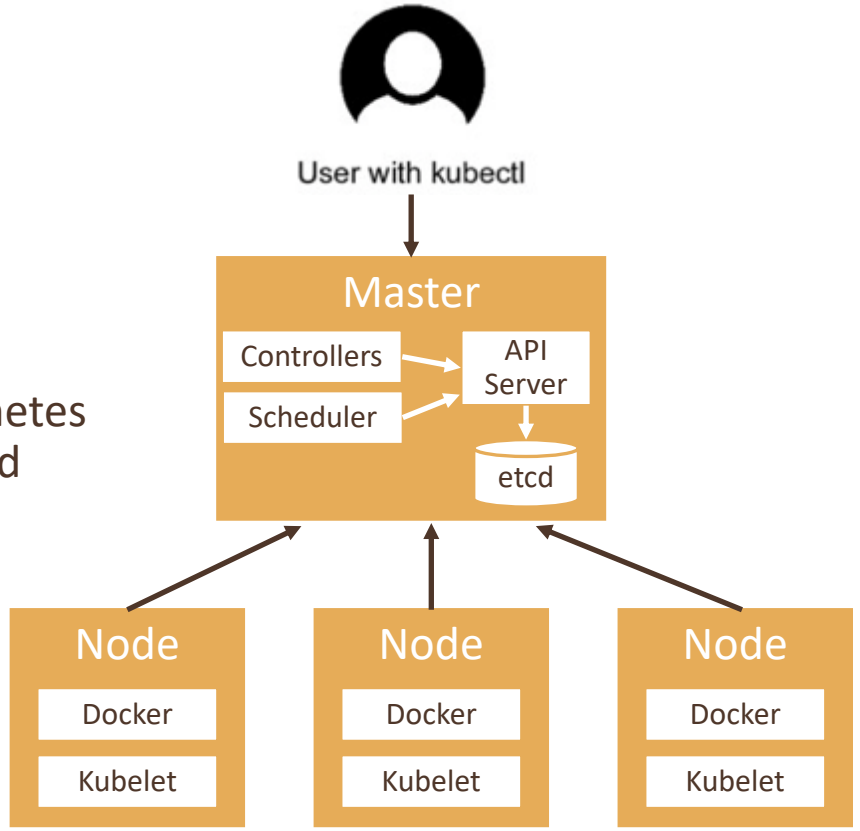


https://www.cncf.io/blog/2017/06/05/30-highest-velocity-open-source-projects

Basics: Cluster, Master and Nodes

- Kubernetes coordinates a **highly available cluster** of computers that are connected to work as a single unit (**Kubernetes cluster**). The abstractions in Kubernetes allow you to deploy Docker containerized applications to a cluster without tying them specifically to individual machines.
- You manage your cluster through web UI or through CLI (**kubectl**)
- A Kubernetes cluster consists of two types of resources:
  - The **Master** coordinates the cluster
  - **Nodes** are workers that run the containers (application)
- A node is a VM or a physical computer that is a worker machine in a Kubernetes cluster. Each node has a **Kubelet**, which is the agent for managing nodes and communicating with the Kubernetes master.
- Nodes are where the Docker containers run.

https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/



Basics: Deployments

- Once you have a running Kubernetes cluster, you can deploy your containerized applications on top of it. To do so, you create a Kubernetes **Deployment** configuration. The Deployment, written in YAML, instructs Kubernetes how to create and update instances of your application.
- Once you've created a Deployment, the Kubernetes master schedules the containers (application instances) onto individual Nodes in the cluster.

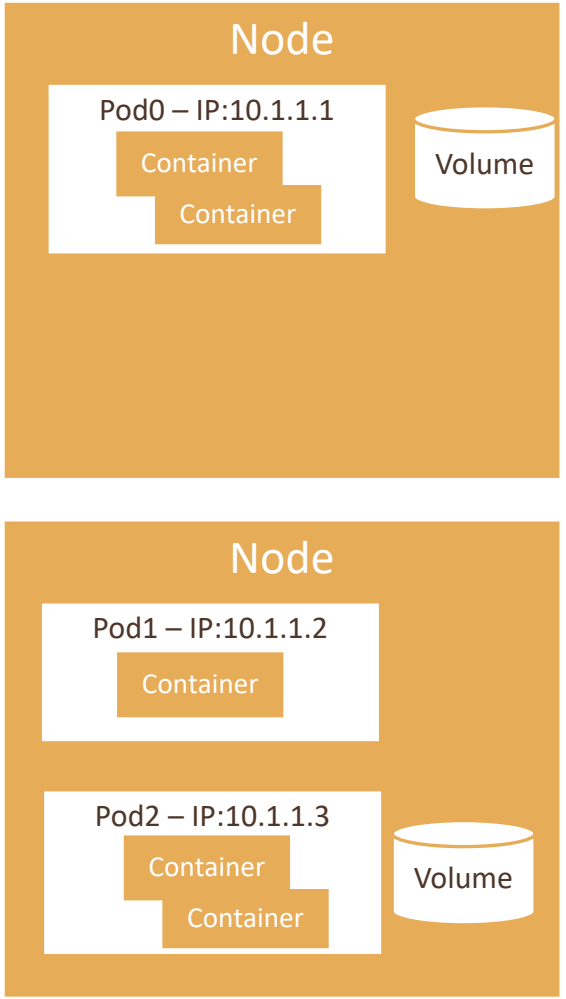
nginx-deployment.yaml

```yaml
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

kubernetes

Basics : Pods

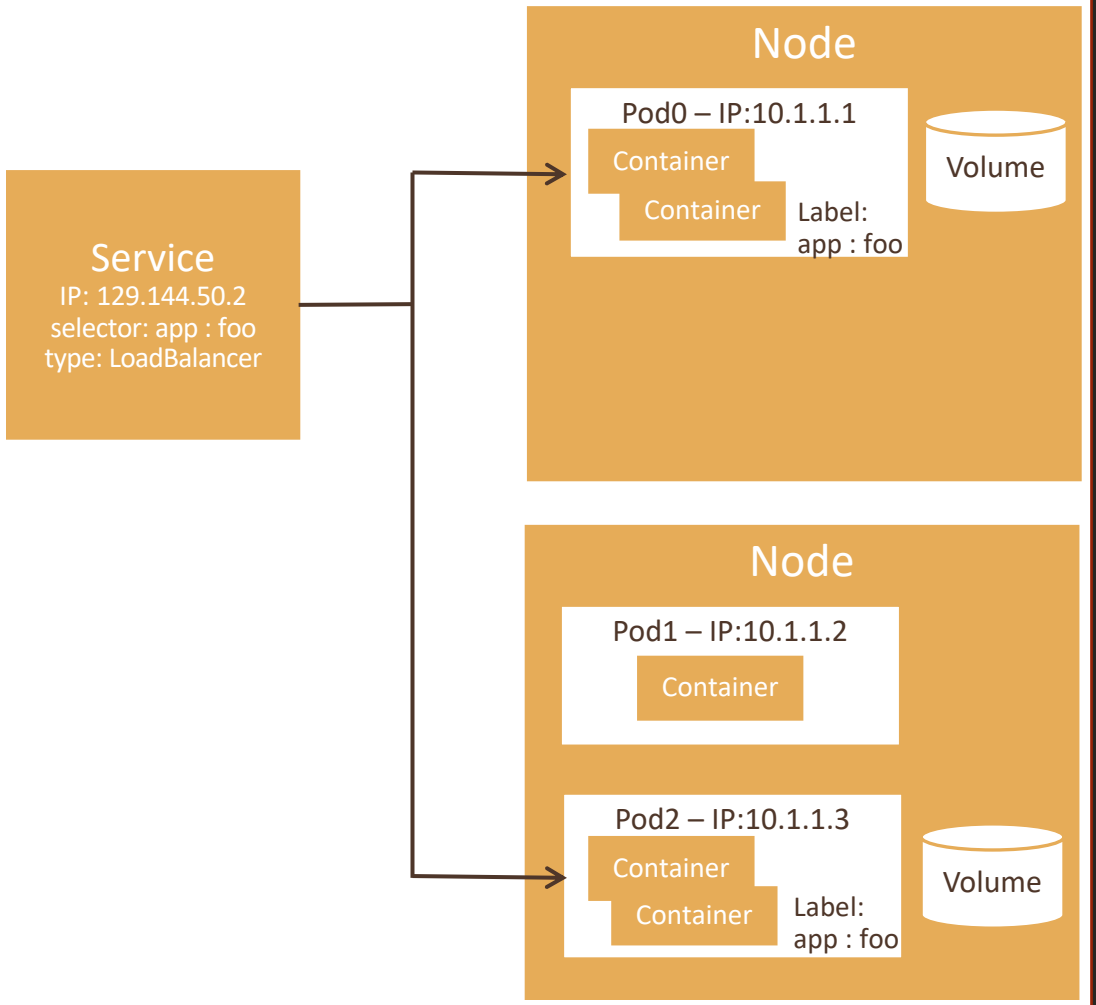
- When you create a Deployment, Kubernetes create **Pods** to host your application instance.
- A **Pod** is a Kubernetes abstraction that represents a group of one or more application containers, and some shared resources for those containers, such as storage resources and a unique network IP.
- Each Pod is tied to the Node where it is scheduled, and remains there until termination (according to restart policy) or deletion. In case of a Node failure, identical Pods are scheduled by the Controller on other available Nodes in the cluster.



kubernetes

Basics: Applications in Kubernetes – how to access them?

- A Kubernetes **Service** is an abstraction which defines a logical set of Pods running somewhere in your cluster. When created, each Service is assigned a unique IP address. Although each Pod has a unique IP address, those IPs are not exposed outside the cluster without a Service.
- Services allow your applications to receive traffic whether from other Pods within the cluster or outside the cluster, for example, from the Internet via a Load Balancer.

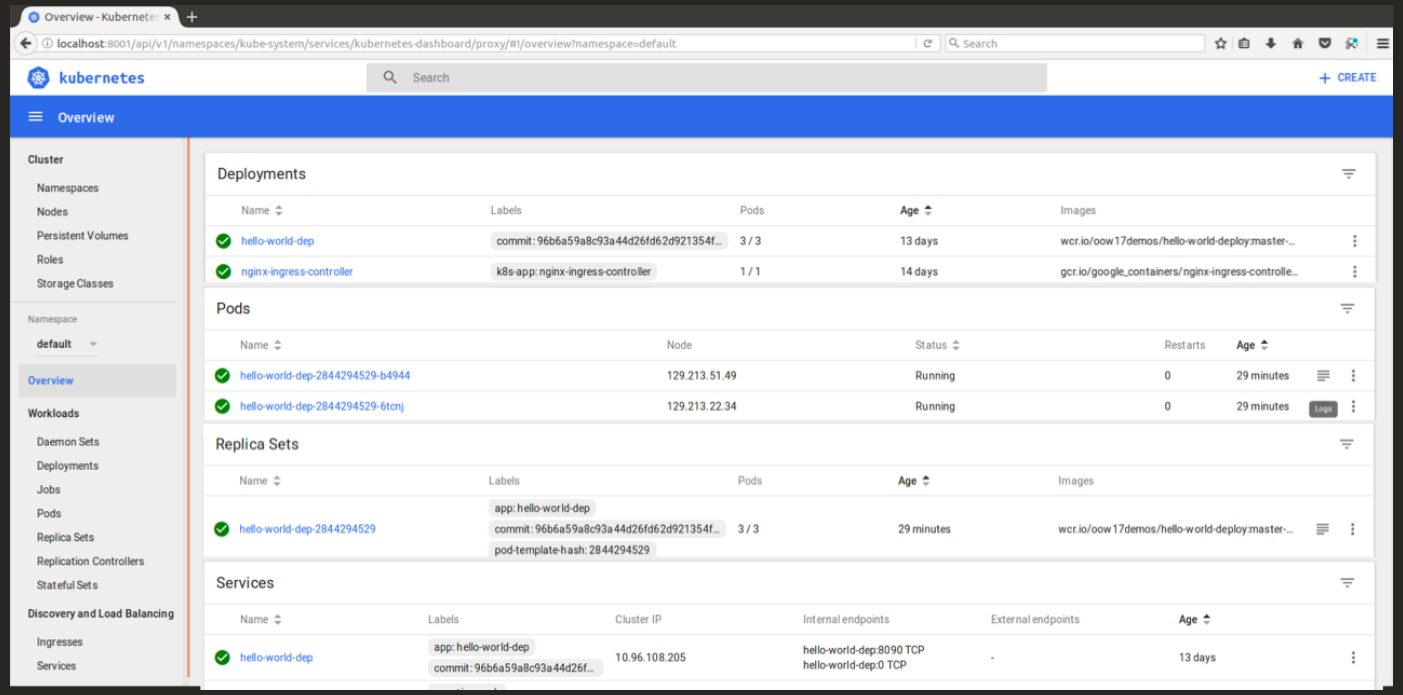


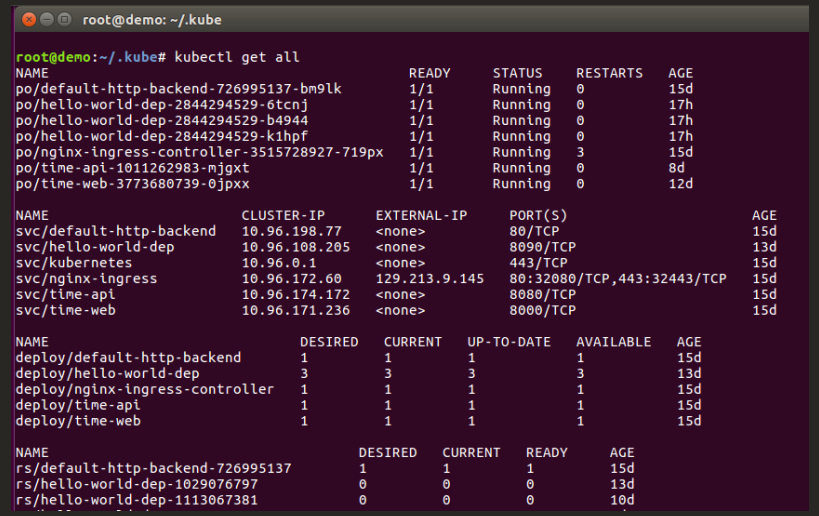
kubernetes

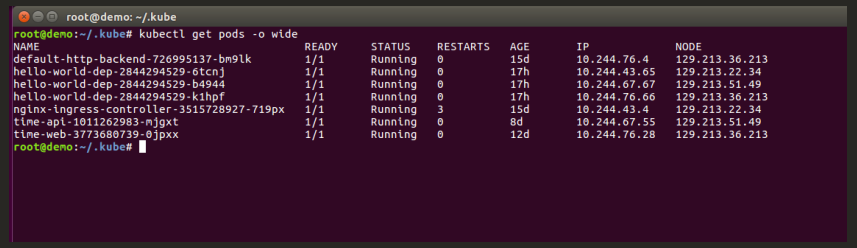
Other Terminology

- Namespace: A logical isolation method. Most resources are associated with a namespace. You can then group similar workloads and enforce different policies.
- Replication Sets: This controls how many identical copies of a Pod should be running somewhere in the cluster. This is a useful feature, when you may want a Pod in each availability domain for high availability or you want to scale the application and create more Pods.
- Persistent Volume: Volumes setup by the administrator that provide persistent storage for applications that require long lived data. The deployment would use a Persistent Volume Claim to access a Persistent Volume

Examples of K8s Dashboard and Kubectl Output

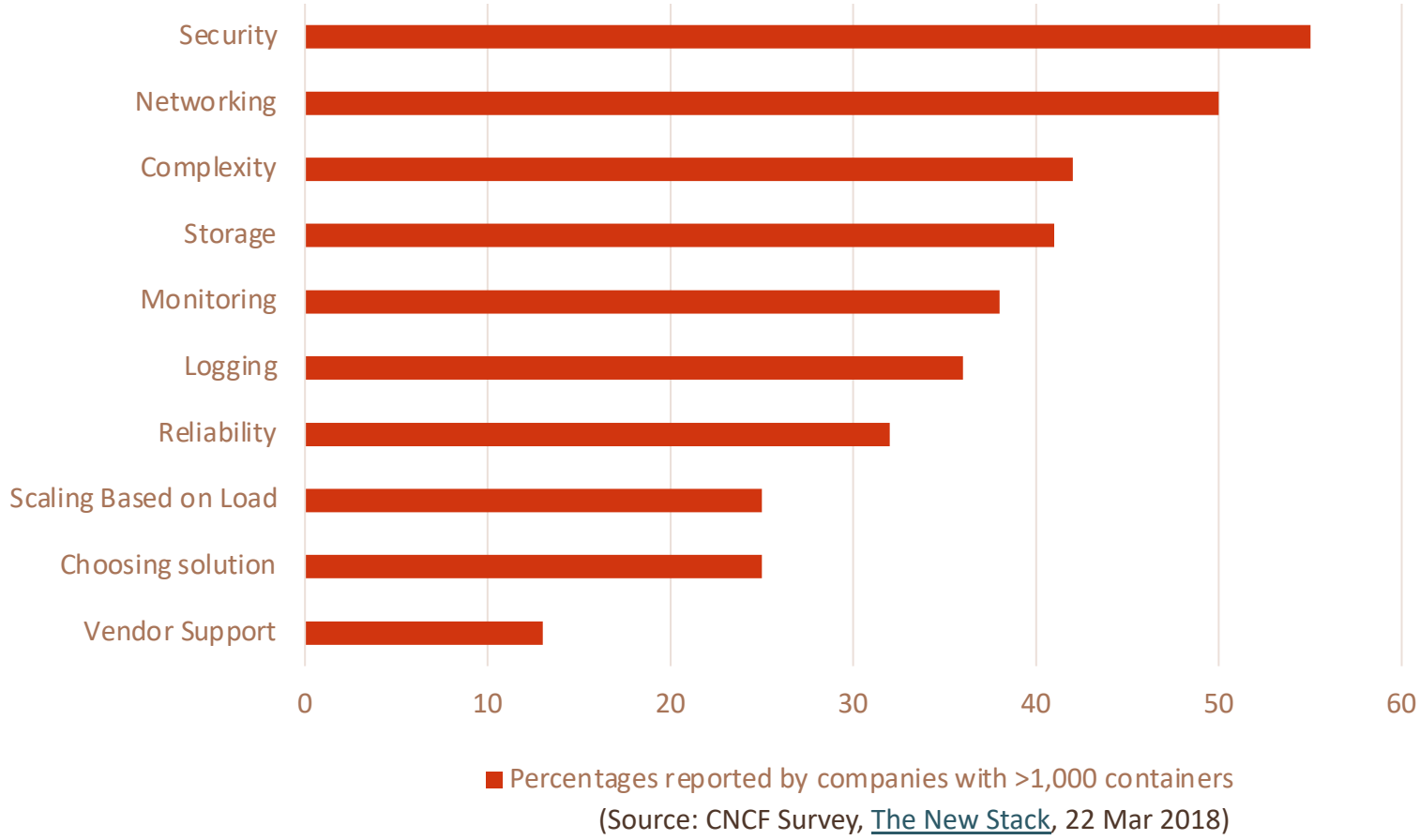






Kubernetes Challenges

- Managing Kubernetes Infrastructure, upgrading, security
- Container networking & persistent storage
- Managing Teams & Access
- CI/CD Integration, automated testing, conditional release



(Source: CNCF Survey, The New Stack, 22 Mar 2018)

Key Containers / Orchestration Use Cases

	Share	Container Use Cases	Orchestration Use Cases
Development	65%	Developer productivity; Consistent appstacks in Dev, Test & Production	Automated deploys to accelerate application release cadence
CI/CD/DevOps	48%	Containerized dependencies; Container registries;	Rolling updates and reversals
Operations	41%	Standardized environments for dev, testing and operations	Resilient, self-healing systems; High Availability; Elastic Scalability
Refactor Legacy Apps	34%	Refactor from N-tier to portable containerized applications	Run distributed, stateful apps on scale-out infrastructure
Migrate to Cloud	33%	Move entire appstacks and see them run identically in the cloud	Cloud bursting; Reduce infrastructure costs by avoiding over-provisioning
New Microservice Apps	32%	Create small purpose-built services that can be assembled to scalable custom applications	Dynamically manage large-scale microservices infrastructure

SOURCE: THE EVOLUTION OF THE MODERN SOFTWARE SUPPLY CHAIN, DOCKER SURVEY 2016

Oracle Cloud Infrastructure Registry & Container Engine for Kubernetes

Oracle's Commitment to Containers and Open Source

- Oracle's participation in open source community
  - Active Participation – Cloud Native Compute Foundation and Kubernetes (see https://www.cncf.io/about/members/)
  - No forked code – straight from the source
  - Continue precedence of Java, MySQL, Linux
- Lead by example
  - Oracle software on Docker Store
  - Kubernetes engineering in CNCF
  - Java SE/EE open sourcing; transparent processes
- Innovate in open source
  - Utilities like K8S terraform install, smith, railcar, crashcart
- Sponsor & contribute to key conferences

Active Community Participation

dockercon 17

CoreOS

Innovate in Open Source

OpenJDK

OpenJDK 4 Containers

CLOUD NATIVE COMPUTING FOUNDATION

Kubernetes

Open Sourcing Docker & K8s Utilities

K8S installer

smith

crashcart

railcar

Oracle Strategy for Container Based Infrastructure

Complete Cloud Native Stack

Deliver tools and services that are complete, integrated and open

- Continuous Integration & Deployment, Container Registry, Orchestration/Scheduling, Management/Operations, Analytics/Introspection
- With an application development platform for serverless and microservices

Open Source

Actively participate in com-munity driven open source container technologies

- Investing in Kubernetes, Docker, Fn, & CNCF, DevTools, and DevOps, with engineering resources, code contributions & sponsorships
- Active support from Oracle's portfolio of open source assets (Java, etc.)

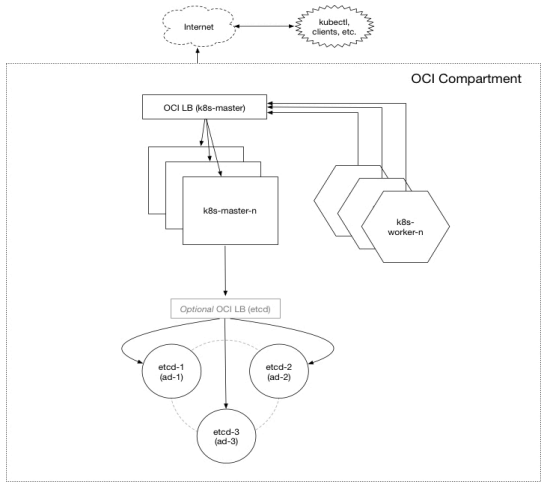
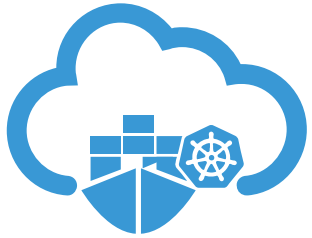
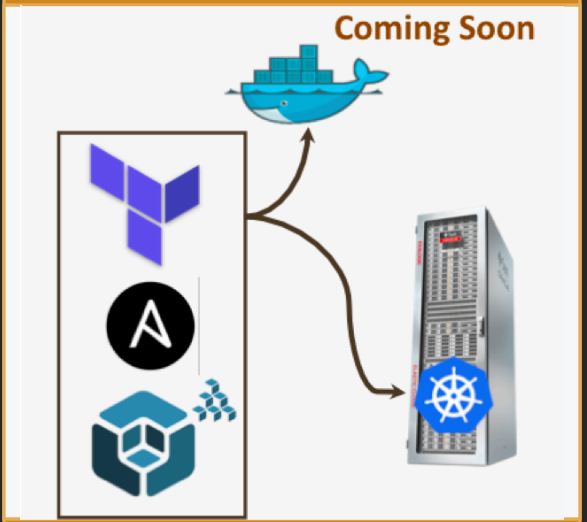
Managed Services

Differentiate on quality of implementation, service and operational excellence

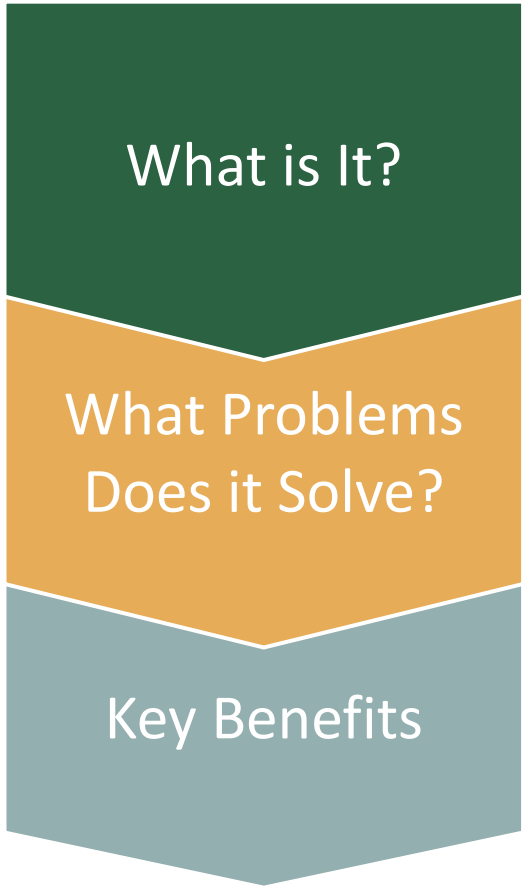
- Full, transparent management
- Open Source Compatible
- Standards compliant
- Deployed to Oracle Cloud Infrastructure
- Enterprise grade performance, security, HA, and governance

Four Ways to Run Kubernetes on Oracle Cloud Infrastructure

Roll Your Own, Pre-Built Installer, Managed Service, Tools for Cloud@Customer

Oracle Cloud Infrastructure	Quickstart Experience (OSS Terraform Installer on Github)	Container Engine for Kubernetes	Manager for Kubernetes on Cloud@Customer
			
Roll-Your-Own Container Management	Pre-Built Kubernetes Installer	Enterprise Class Managed Kubernetes Service	Enterprise Tools for Managing Kubernetes

Introducing Container Engine for Kubernetes - OKE

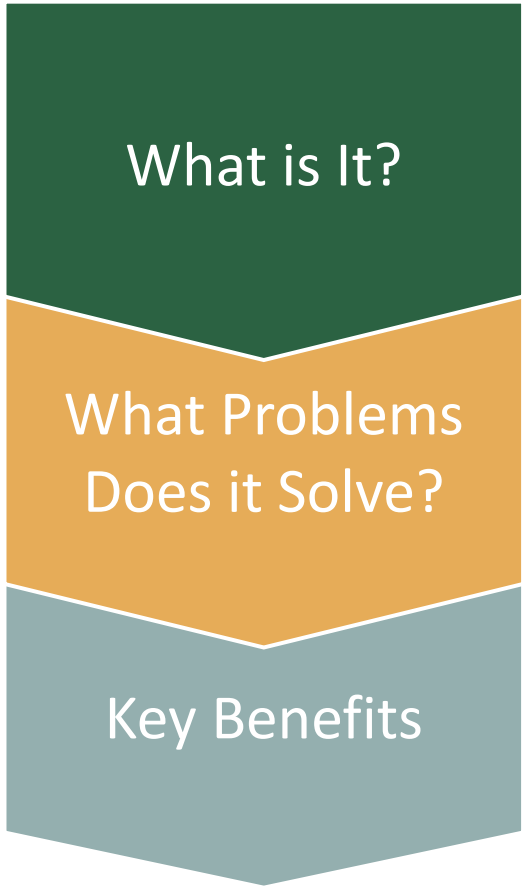


- Managed Kubernetes container service to deploy and run your own container based apps
- Tooling to create, scale, manage & control your own standard Kubernetes clusters instantly

- Too complex, costly and time consuming to build & maintain environments
- Too hard to integrate Kubernetes with a registry and build process for container lifecycle management
- Too difficult to manage and control team access to production clusters

- Enables developers to get started and deploy containers quickly. Gives DevOps teams visibility and control for Kubernetes management.
- Combines production grade container orchestration of open Kubernetes, with control, security, IAM, and high predictable performance of Oracle's next generation cloud infrastructure

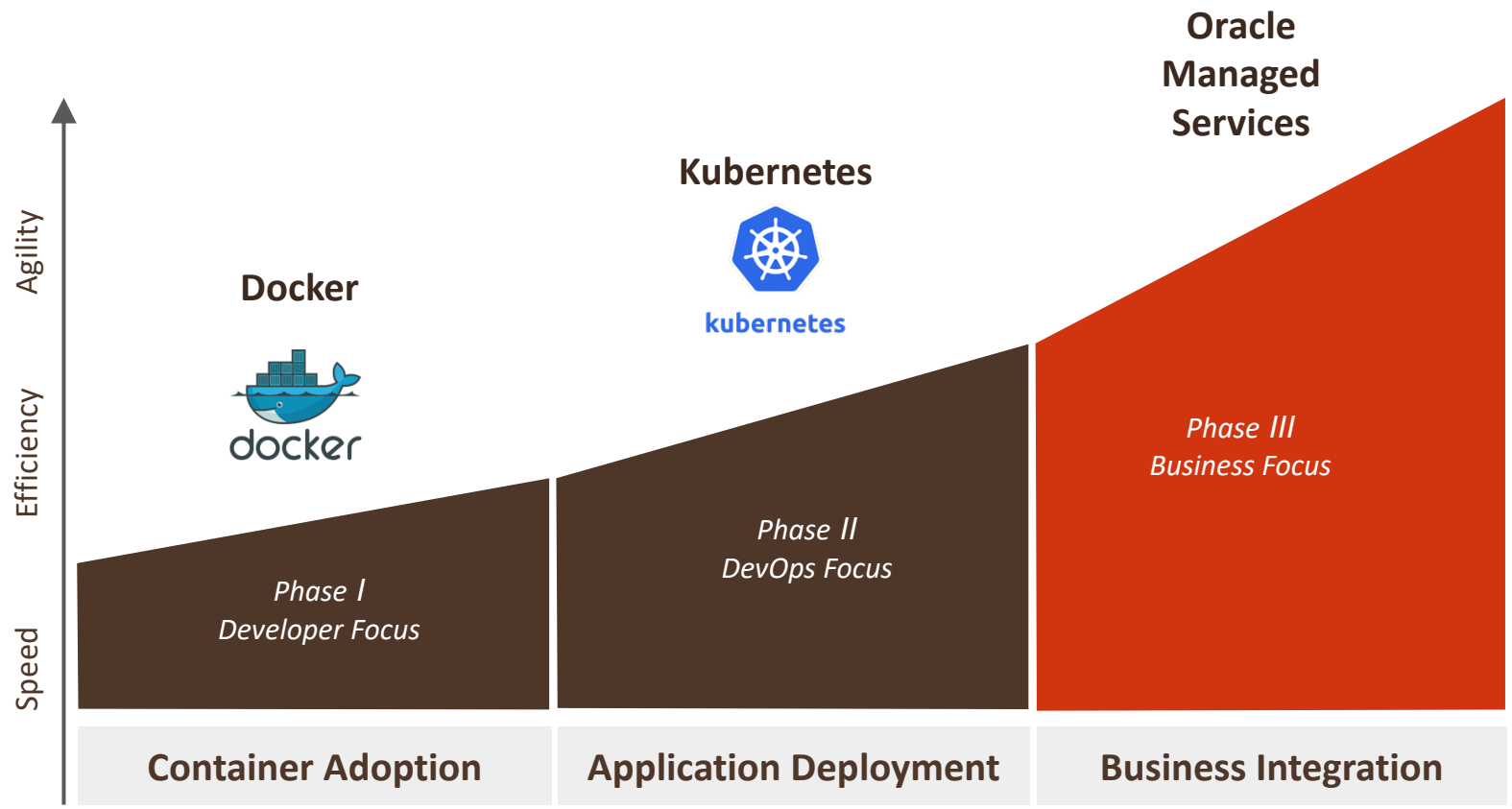
Introducing Oracle Cloud Infrastructure Registry - OCIR



- A high availability Docker v2 container registry service
- Stores Docker Images in Private Repositories
- Runs as a fully managed service on Oracle Cloud Infrastructure

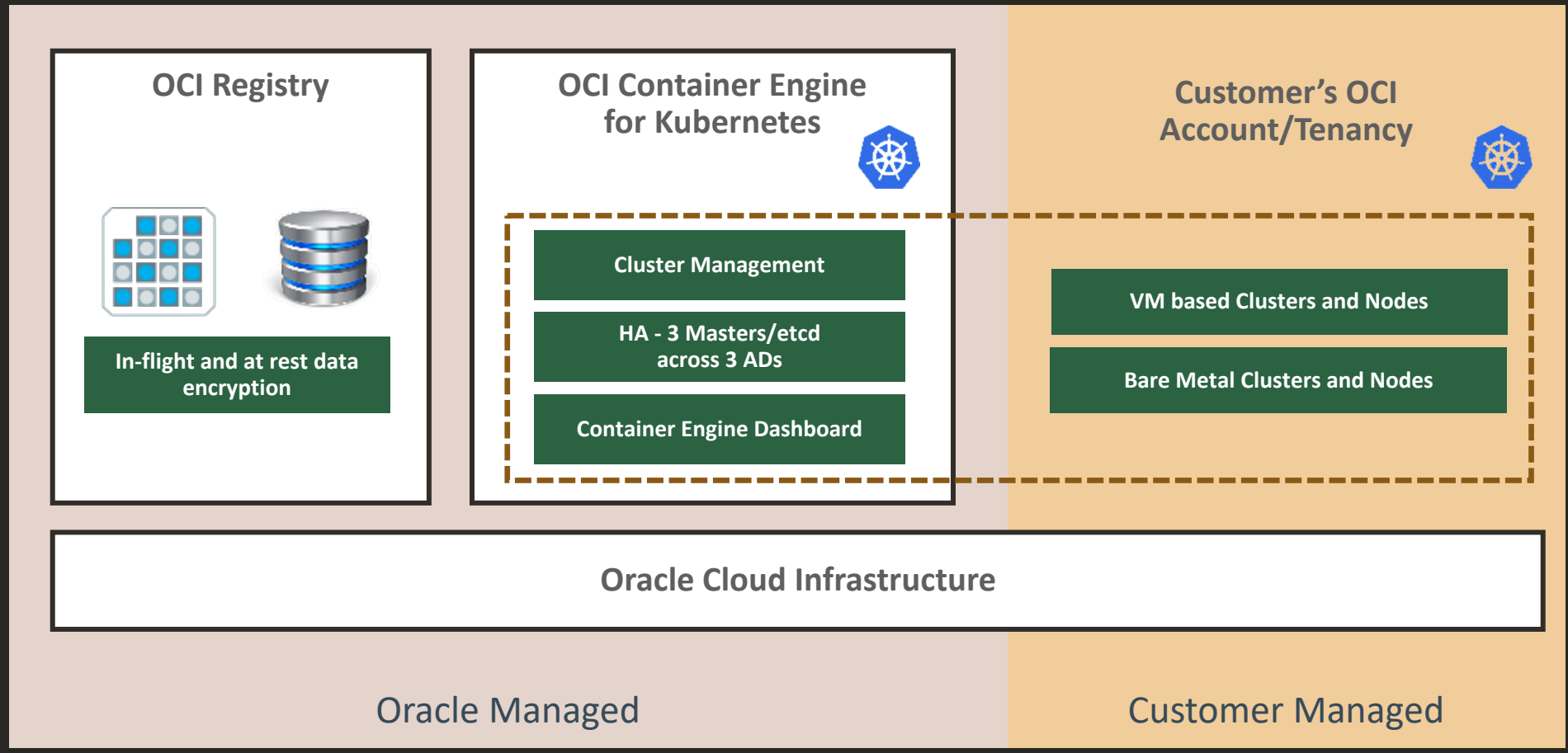
- Without a registry it is hard for Development teams to maintain a consistent set of Docker images for their containerized applications
- Without a registry it is hard to enforce access rights and security policies for images
- It is too hard to find the right images and have them available in the region of deployment

- Full integration with Container Engine for Kubernetes (OKE)
- Registries are private by default, but can be made public by an admin
- Co-located regionally with Container Engine for low latency Docker image deploys
- Leverages OCI for high performance, low latency and high availability

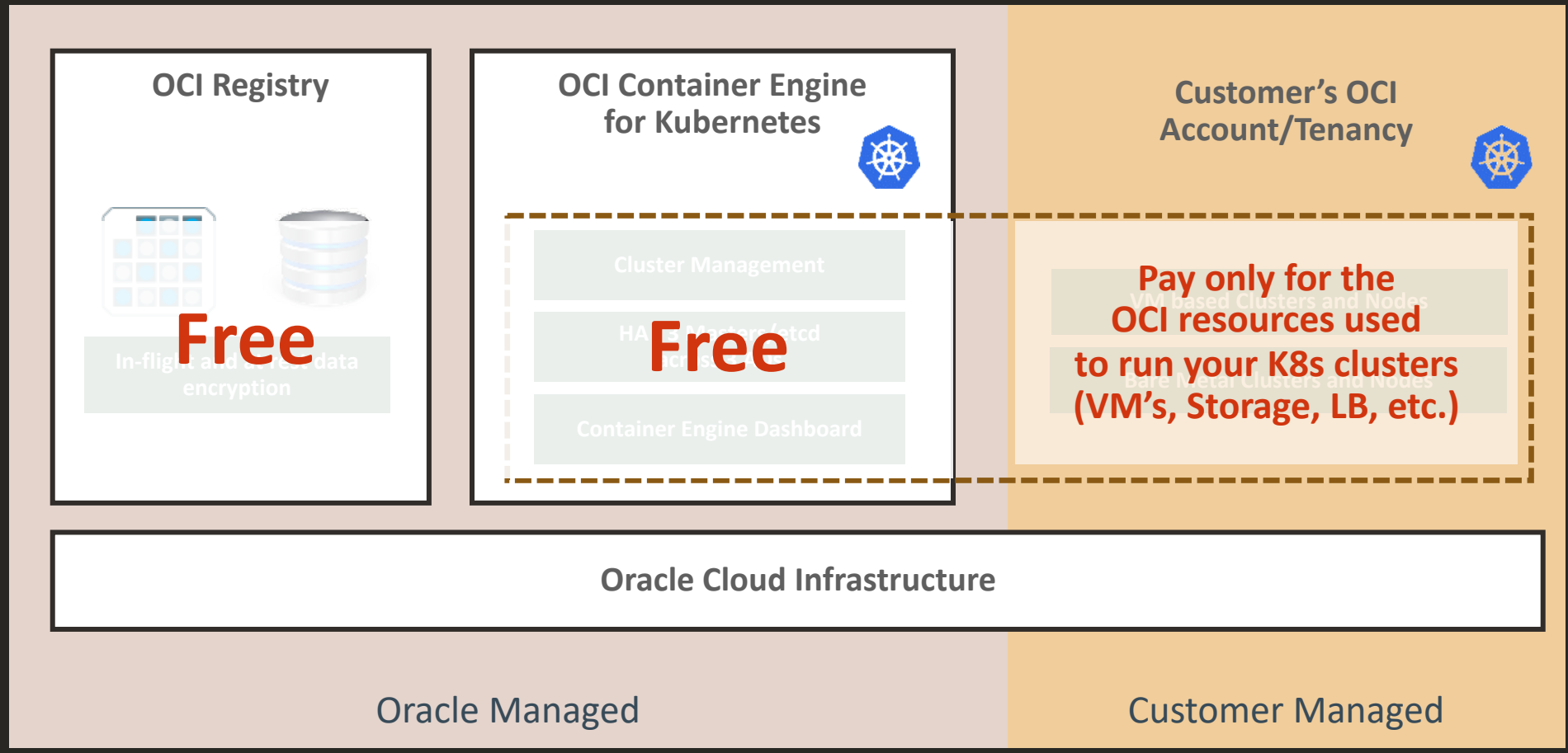


Focus:	Developer adoption	DevOps deployment	Broad integration
Applications:	Dev/Test apps	Production apps	Business apps
Automation:	Simple orchestration	Advanced orchestration	Self management
Community:	Individual developers	Teams & lines of business	Enterprises

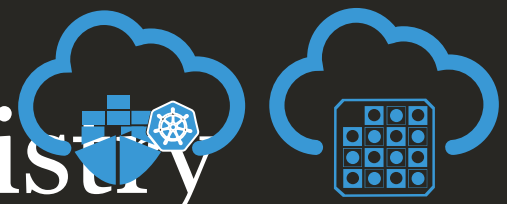
Working with OKE and OCIR on OCI



OKE/OCIR Pricing and Packaging



Summary: Container Engine & Registry



Container Native

- **Standard Docker & Kubernetes**
  - Deploy standard & open upstream Docker and Kubernetes versions for compatibility across environments
- **Registry Integration**
  - Full Docker v2 compatible private registry to store and manage images
- **Container Engine**
  - Deploy and operate containers and clusters
- **Full integration to cloud networking and storage**
  - Leverage the enterprise class networking, load balancing and persistent storage of Oracle Cloud Infrastructure

Developer Friendly

- **Streamlined Workflow**
  - Use your favorite CI to push containers to the registry, then Kubernetes to deploy to clusters and manage operations
- **Full REST API**
  - Automate the workflow, create and scale clusters through full REST API
- **Built In Cluster Add-Ons**
  - Kubernetes Dashboard, DNS & Helm
- **Open Standards**
  - Docker Based Runtime
  - Worker Node SSH Access
  - Standard Kubernetes

Enterprise Ready

- **Simplified Cluster Operations**
  - Use the fully managed, highly available registry, master nodes and control plane
- **Full Bare Metal Performance and Highly Available IaaS**
  - Combine Kubernetes with bare metal shapes for raw performance
  - Deploy Kubernetes clusters across multiple Availability Domains for resilient applications
- **Team Based Access Controls**
  - Control team access and permissions to clusters
- **Autonomous Clusters**
  - Maintain cluster size and performance in face of node failures and load fluctuations

OKE - Labs

“Quanto mais tempo demorar para solucionar um problema, mais simples (e idiota) é a solução”

Cristiano Hoshikawa

Container Application Life Cycle

CI/CD tools deploy container image to OCIR

OKE pulls container images to build cluster

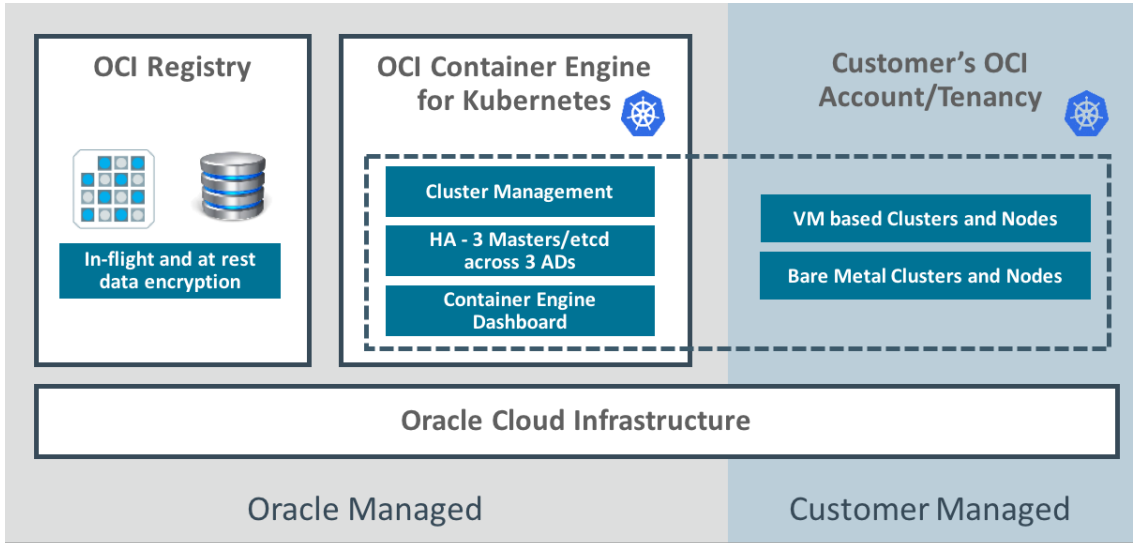
OKE deploys/ manages clusters in user tenancy

Running clusters expose container application

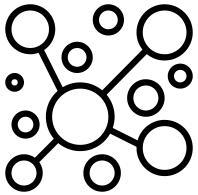
1 2 3 4

CI/CD Tools

Oracle Pipelines, Jenkins, Hudson, TeamCity, Travis, etc.



Exposed Services



Agile, Elastic, Scalable, Resilient, Performant, Secure

Hands-on

https://github.com/hoshikawa2/OCI-DEV

Dashboard Kubernetes

kubectl

O que é: Linha de comando para deployments e gestão dos componentes em kubernetes

É com esta linha de comando que vamos instalar os componentes do workshop hoje!

Dashboard Kubernetes

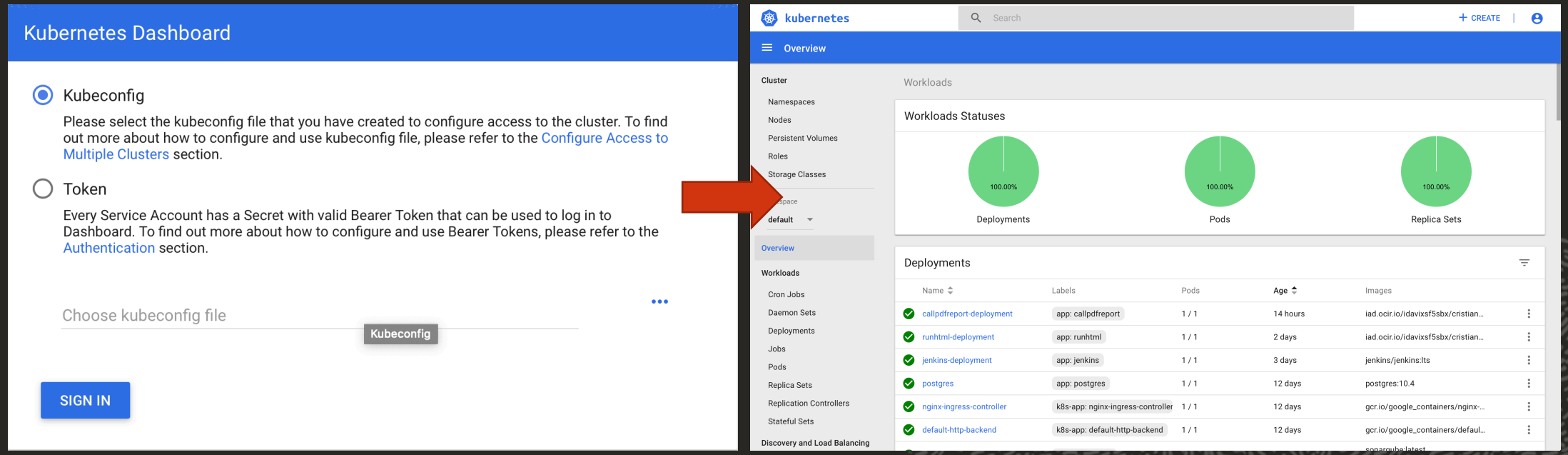
Versão nova de Kubernetes necessita trazer o token:

TOKEN=$(kubectl -n kube-system describe secret default| awk '$1=="token:"{print $2}’)

kubectl config set-credentials kubernetes-admin --token="${TOKEN}"

kubectl proxy

http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/#!/login



YAML

According to www.yaml.org, “YAML is a **human-friendly**, data serialization standard for all programming languages.”

XML	JSON	YAML

<Servers>
<Server>
<name>Server1</name>
<owner>John</owner>
<created>123456</created>
<status>active</status>
</Server>
</Servers>
	{
	Servers: [
	{
	name: Server1,
	owner: John,
	created: 123456,
	status: active
	}
	]
	}
		Servers:
		- name: Server1
		owner: John
		created: 123456
		status: active

YAML - Deployment

Deployment é o componente de software em si

Você define quantas replicas para definir quantos pods precisam entrar em funcionamento para performance

Image define uma imagem de container já previamente criada em um repositório

ContainerPort define a porta que este pod está acessível

VolumeMounts e PersistentVolumeClaim definem a localização física do seu container

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: jenkins-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: jenkins
  template:
    metadata:
      labels:
        app: jenkins
    spec:
      containers:
      - name: jenkins
        image: jenkins/jenkins:lts
        securityContext:
          runAsUser: 0
        ports:
        - containerPort: 8080
        volumeMounts:
          - name: jenkins-home
            mountPath: /var/jenkins_home
      volumes:
        - name: jenkins-home
          persistentVolumeClaim:
            claimName: jenkins-pv-claim
```

YAML - Service

O serviço é a porta de acesso do mundo para o pod

Você consegue definir se o acesso é feito via:

- Load Balancer (mundo externo através de um IP externo)
- Cluster IP (acesso apenas via o cluster de kubernetes)
- NodePort (acesso via as VMs e também via cluster kubernetes)

```yaml
apiVersion: v1
kind: Service
metadata:
  name: jenkins-service
  labels:
    app: jenkins
spec:
  selector:
    app: jenkins
  ports:
    - port: 8080
      targetPort: 8080
  type: ClusterIP
```

YAML - Storage

É necessário definir muitas vezes um storage para persistência de dados, pois caso contrário, seu pod será stateless

A forma como é feita em OCI precisa que storageClassName seja “oci”

Na definição de Deployment, para utilizar o storage definido aqui, é necessário fazer a seleção pelo label

```yaml
kind: PersistentVolume
apiVersion: v1
metadata:
  name: jenkins-pv-volume
  labels:
    type: local
    app: jenkins
spec:
  storageClassName: "oci"
  capacity:
    storage: 10Gi
  accessModes: [ "ReadWriteOnce" ]
  persistentVolumeReclaimPolicy: Retain
  hostPath:
    path: "/mnt/data"
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: jenkins-pv-claim
  labels:
    app: jenkins
spec:
  storageClassName: "oci"
  accessModes: [ "ReadWriteOnce" ]
  resources:
    requests:
      storage: 10Gi
```

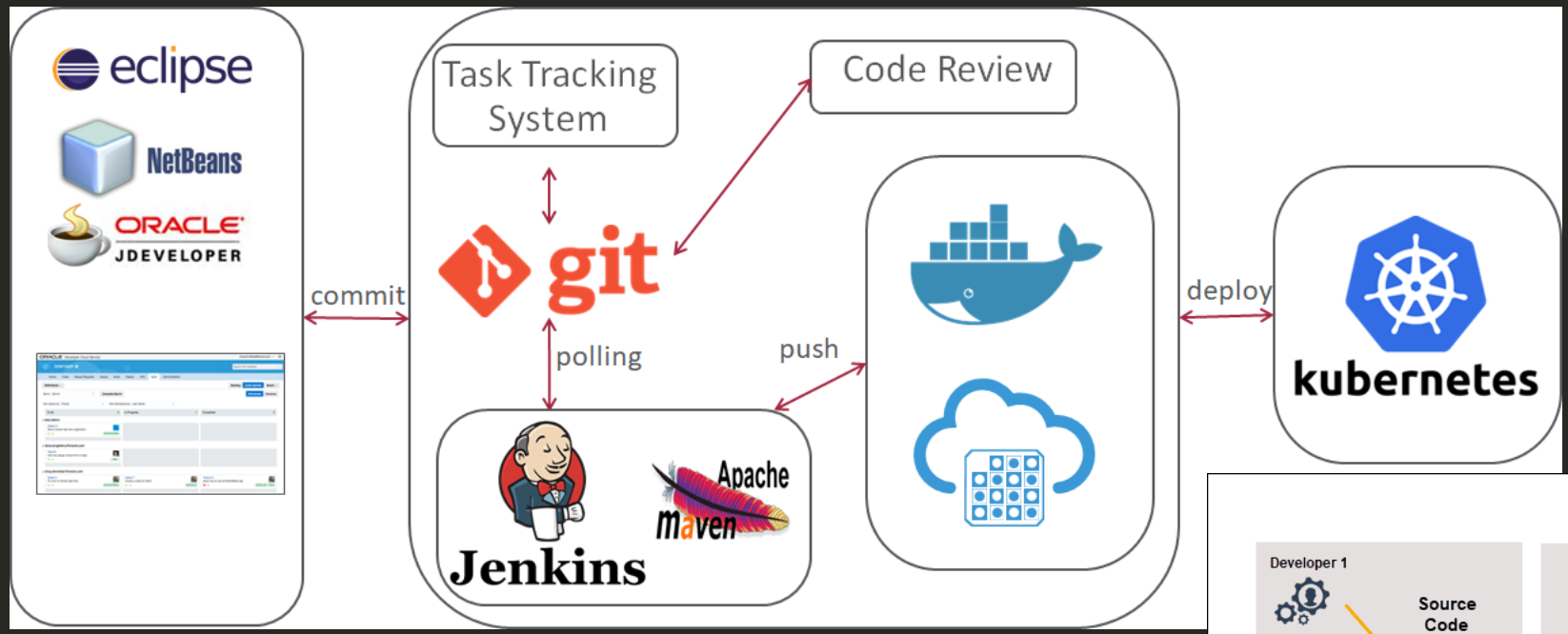
YAML - Ingress

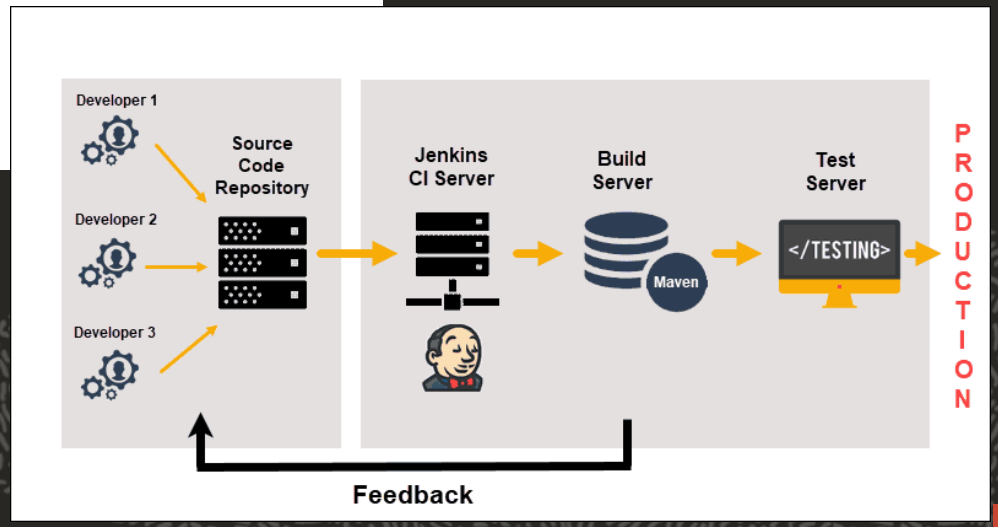
Ingress permite ter controle de direcionamento para os serviços desejados

Ao invés de usar LoadBalancer e configurar tudo manualmente

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: jenkins-ingress
  annotations:
    kubernetes.io/ingress.class: "nginx"
spec:
  tls:
  - secretName: tls-secret
  rules:
  - http:
      paths:
      - backend:
          serviceName: jenkins-service
          servicePort: 8080
```

Lab - Jenkins





Lab – Install jenkins on OKE

https://github.com/hoshikawa2/oci-oke-jenkins-kubernetes.git

Executar nesta ordem:

kubectl create -f jenkins-storage.yaml
kubectl create -f jenkins-deployment.yaml
kubectl create -f jenkins.yaml

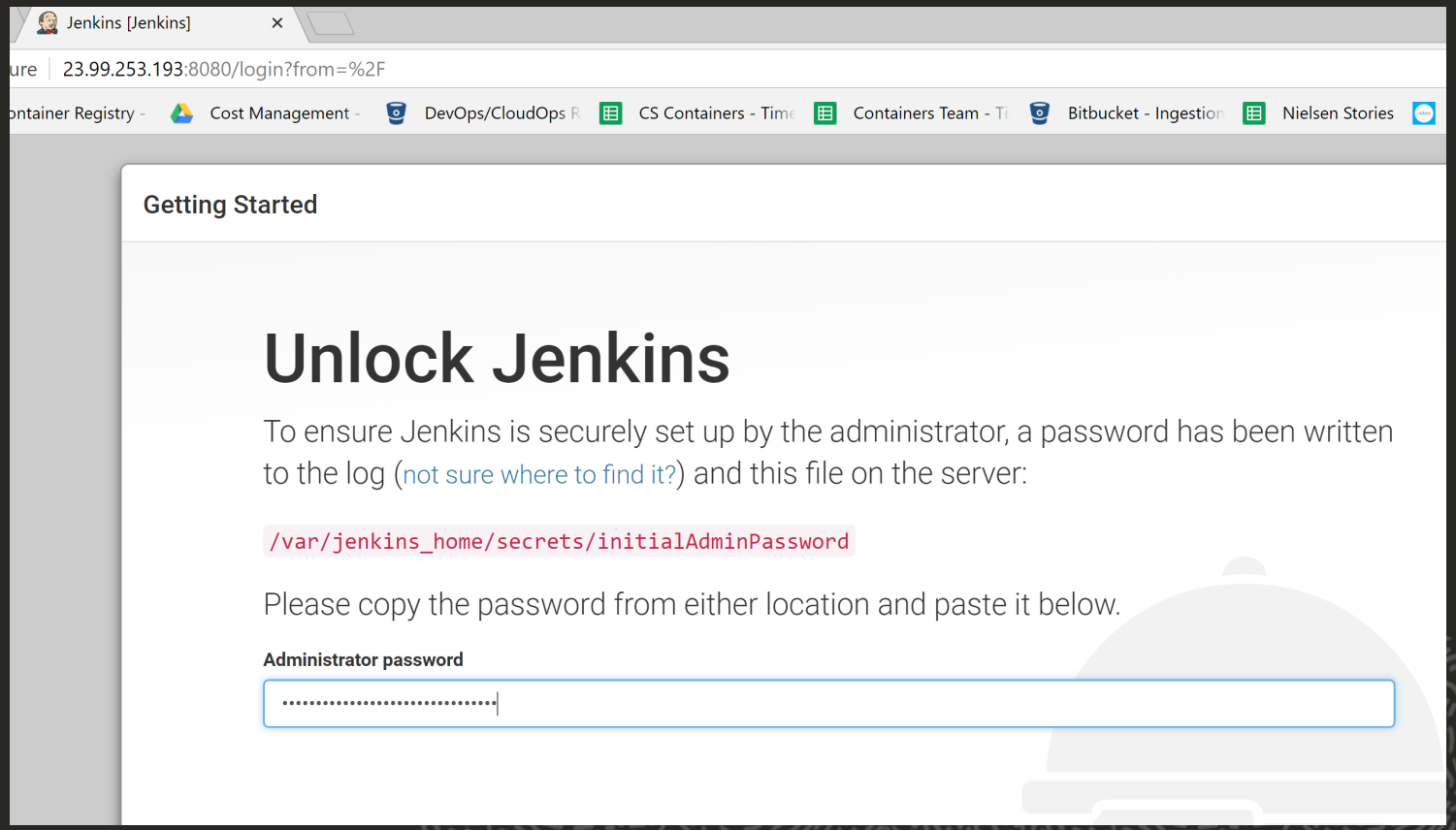
Obs: Não vamos executar o jenkins-ingress.yaml, deixaremos para um próximo evento

kubectl -n logging port-forward $(kubectl -n logging get pod -l app=jenkins -o jsonpath='{.items[0].metadata.name}') 8080:8080

Abrir no browser: http://localhost:8080

Lab – Using Jenkins

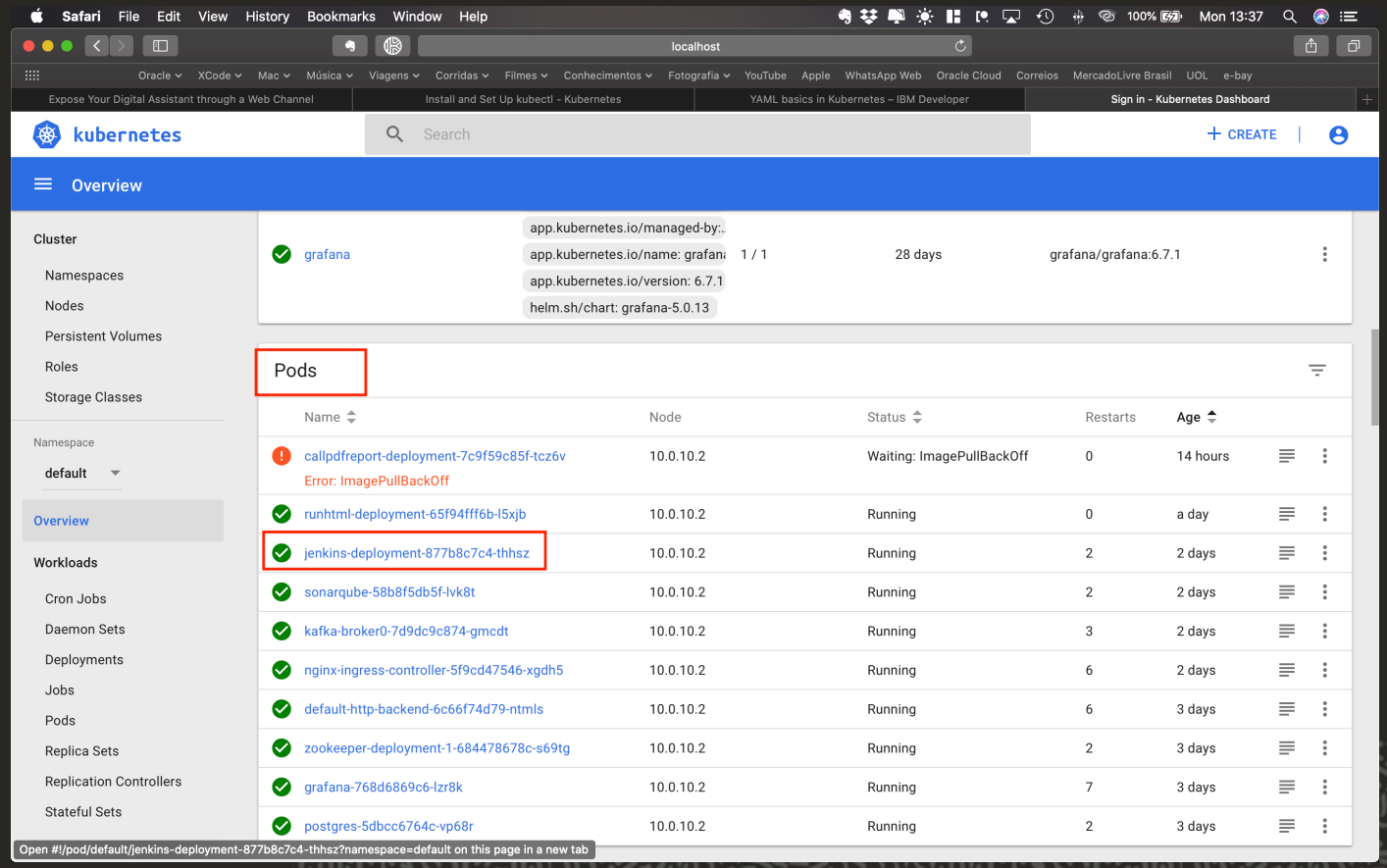
Necessita incluir a senha inicial



Lab – Using Jenkins

Localizar o pod do Jenkins no dashboard Kubernetes

Clicar no POD

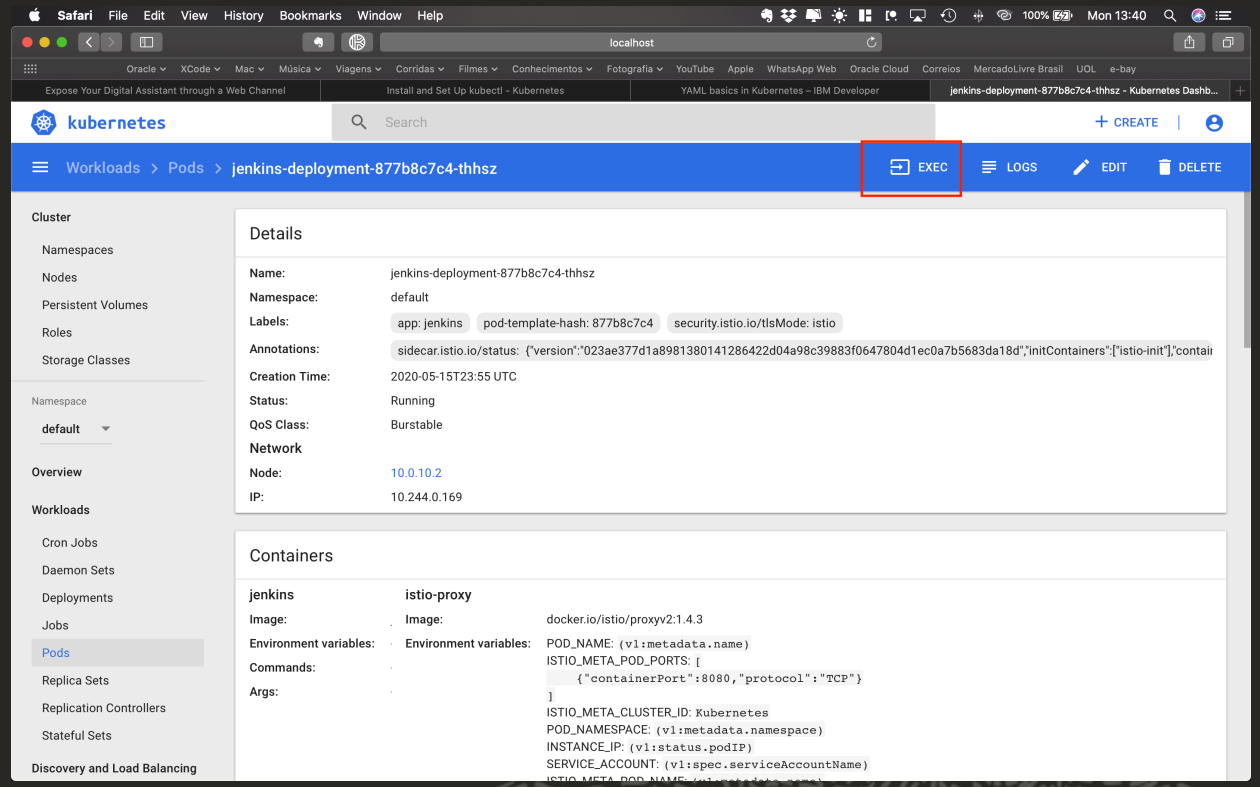


Lab – Using Jenkins

Clicar em EXEC

E no terminal digitar

cat /var/jenkins_home/secrets/initialAdminPassword



Lab – Using Jenkins

Copiar a senha e colar no painel Jenking

E isso vira assunto para um próximo encontro...

ORACLE®
Cloud Platform

Oracle Cloud Platform
Transform Your Business With Innovation

ORACLE