Introducción a la Orquestación de Contenedores con SUSE CaaS Platform y Kubernetes

Juan Herrera Utande

Sales Engineer Cloud Solutions

juan.herrera@suse.com

@jufherrera

@SUSELinux_ESP

Agenda

- Introducción
- Visión del mercado de los contenedores
- SUSE Containers as a Service Platform
- La apuesta de SUSE por los contenedores
- Nuevas características de SUSE CaaS Platform version 2
- Introducción a Kubernetes
- Despliegue de aplicaciones en Kubernetes con Helm
- Casos de uso para SUSE Containers as a Service Platform
- Recursos
- Q & A

Introducción

Presentación

Duración 40 - 45 minutos con bloque final para preguntas
Utilizar chat para cualquier pregunta
Este webinar será grabado y publicado para su fácil acceso

Ciertos conocimientos de contenedores podrían ser necesarios

Empecemos!

Why Conatiners as a Service?

Container Applications are Growing

$0.5B

2015

$2.7B

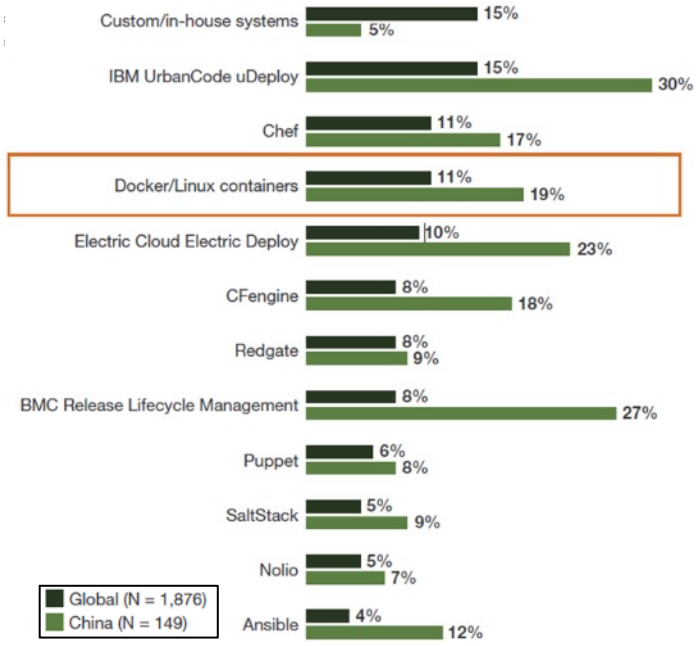
2020

Containers Market 5x in 5 years

Source: http://www.informationweek.com/cloud/infrastructure-as-a-service/451-research-containers-a-$27-billion-market-by-2020/d/d-id/1327868

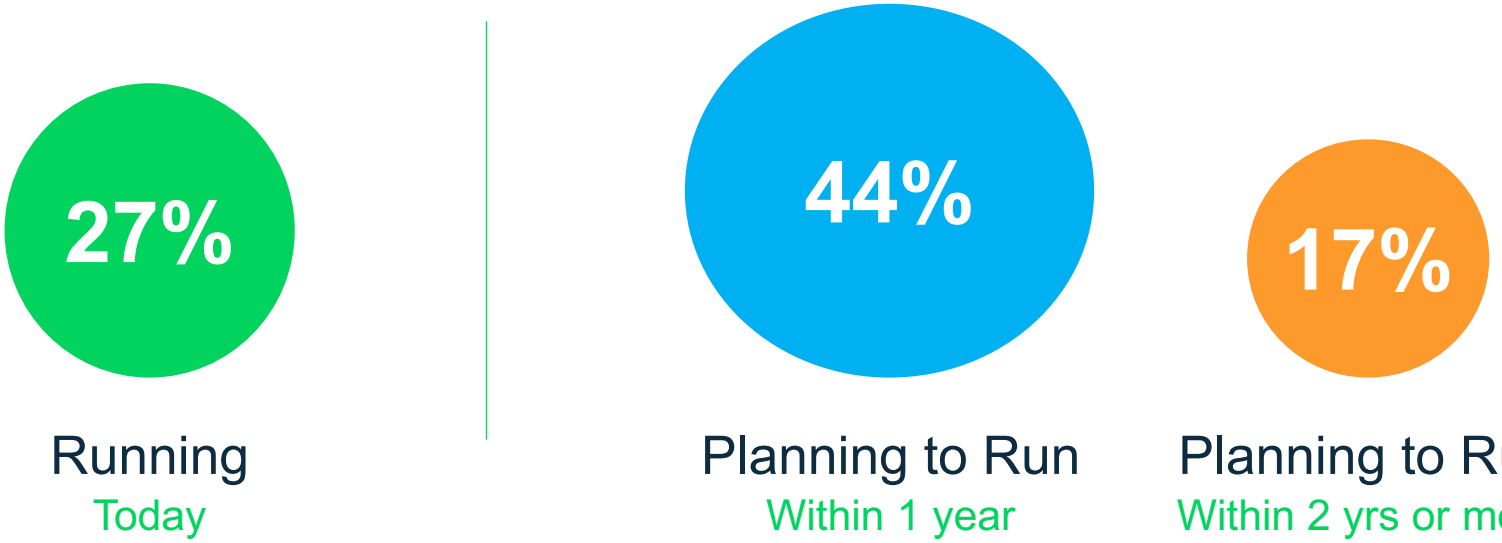
1 in 10 Companies
Use Containers for Application Deployment

“What deployment tools or frameworks do you typically use to release software?”



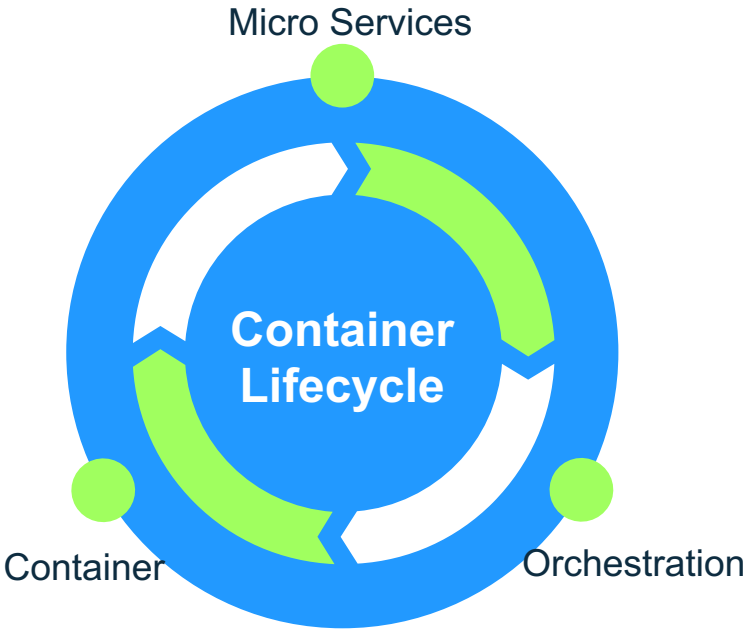
Source: Forrester Research - Vendor Landscape: Container Solutions For Cloud-Native Applications

Enterprises are running container workloads in production

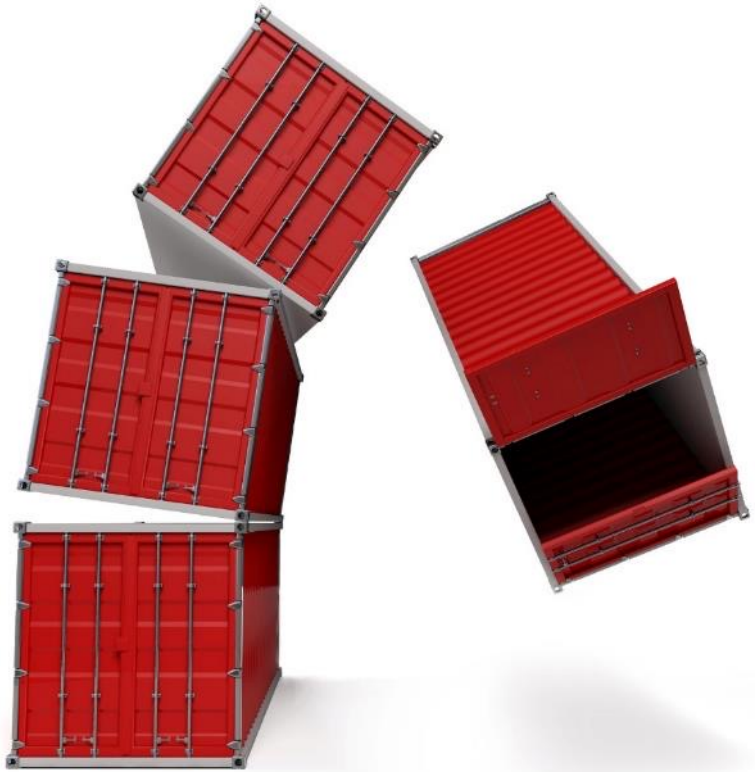


Cloud Adoption Trends Driving IT Transformation Research Report, Insight Avenue, 2017
1412 IT decision makers in companies with 250+ employees, across all sectors, interviewed in 2017
(55% VP / C-level / Director level, 45% Senior Manager level)

Container Engine is Not Sufficient



Provision	Manage	Automate	Host Services



Building a container stack from the ground up is not for everyone.

Container Services in the Public Cloud. Gartner 2017

Organizations Need Containers to be More Consumable

- Achieve faster time to value
- Simplify management of container platform
- Maximize return on investment

Platform for containers enables easy deployment of containerized applications

Who's using Micro Services, Containers, Kubernetes, …?

Some background data:

- **2007 – Velocity Conf: *10 Deployments per Day: Dev and Ops cooperation at Flickr***
- **2011 – Velocity Conf: Amazom -> production deployment every 11,6 seconds. +7000 per day!**
- **2017 – Facebook: continuous push from master (+1000 per day)**

Companies using Containers: ??

Companies using Kubernetes container orchestration

Amadeus, Disney, Google, SAP, Veritas, Yahoo, Zalando, … and we should sum up those using other orchestrators like Mesos or Swarm.

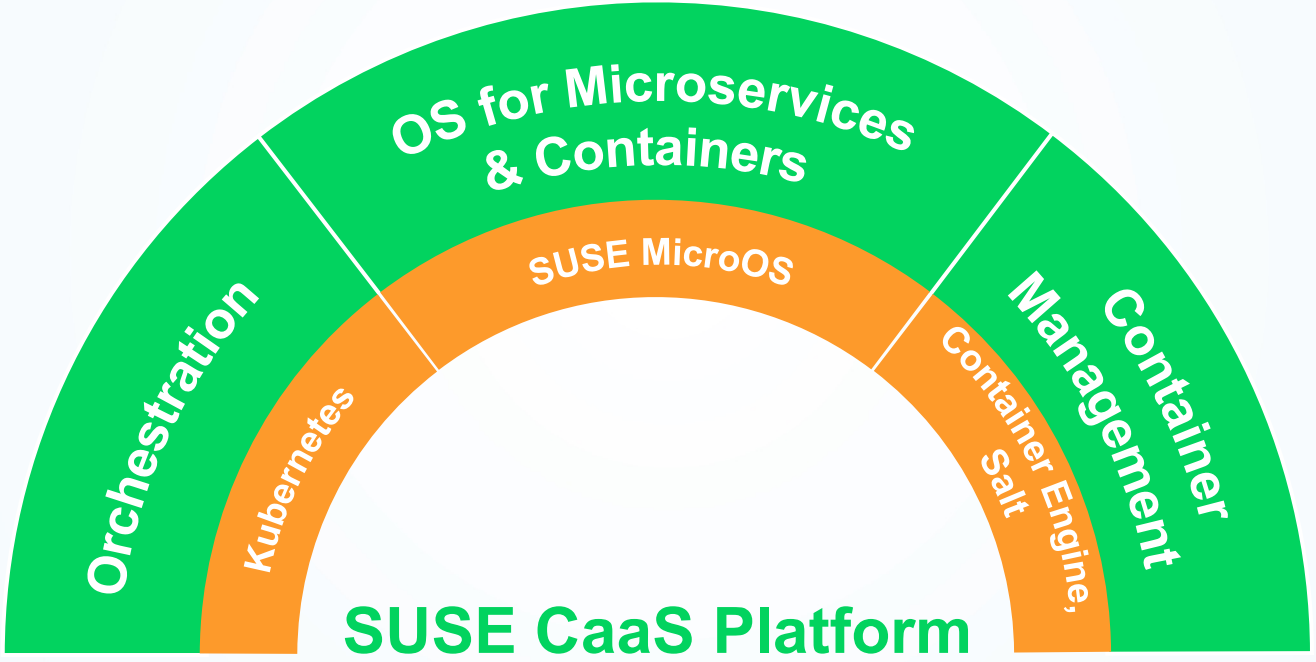
Recommended reading: The Phoenix Project

Great description of how methodologies, microservices and new deployment approaches may solve many of the traditional IT related ineficiencies.

What is SUSE CaaS Platform?

SUSE CaaS Platform is an enterprise class **container management solution** that enables IT and DevOps professionals to more **easily deploy, manage, and scale** container-based applications and services.

3 Key Technology Components
One Enterprise Platform

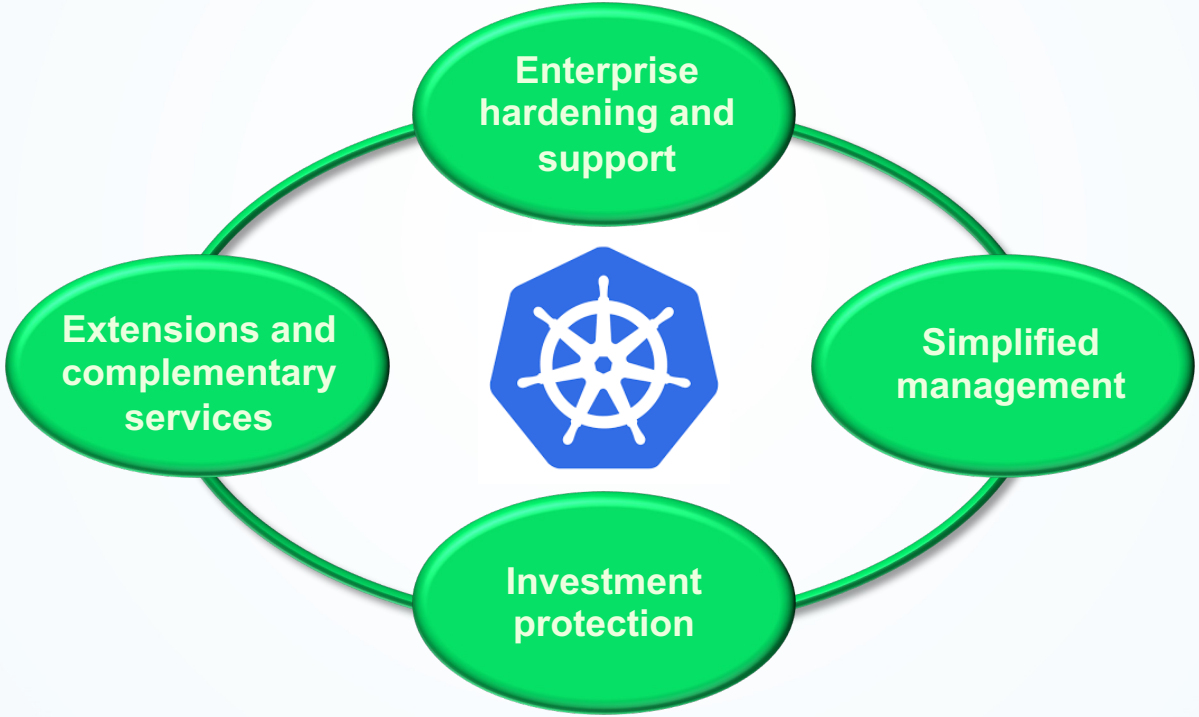


Best of breed

The best orchestration engine: Kubernetes
The best container OS: MicroOS based on SLES
The best configuration tools: Salt and containers engines

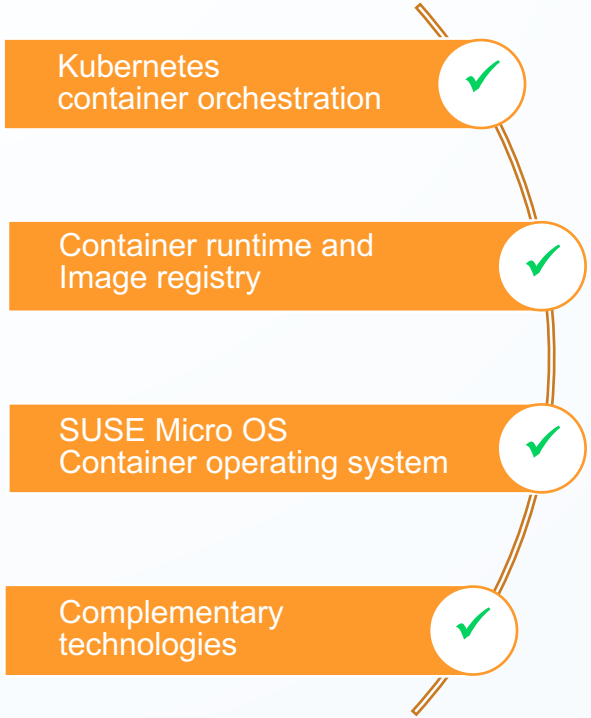
SUSE CaaS Platform simplifies and extends Kubernetes

Container management for the enterprise



Achieve Faster Time-to-Value

With everything you need to quickly offer container services



Curated Package

- Maintained holistically
- Tested to ensure interoperability

SUSE Enterprise Hardening & Support

- Meet internal compliance standards
- Reliable, scalable, and robust

Simplify Management of Your Container Platform

Efficient Installation

Easy setup of Kubernetes

Mass deploy in minutes

Easy Scaling

Manage using CLI or dashboard

Simplify with deployment profiles

Always up-to-date

Complete control with auto & on-command updates

Unique rollback capability by SUSE MicroOS

Maximize ROI

Flexible Platform

Run generic Linux containers, built on any Linux

Deliver using enterprise grade SUSE MicroOS

Designed for Today & Tomorrow

Bank on technology standards (Kubernetes)

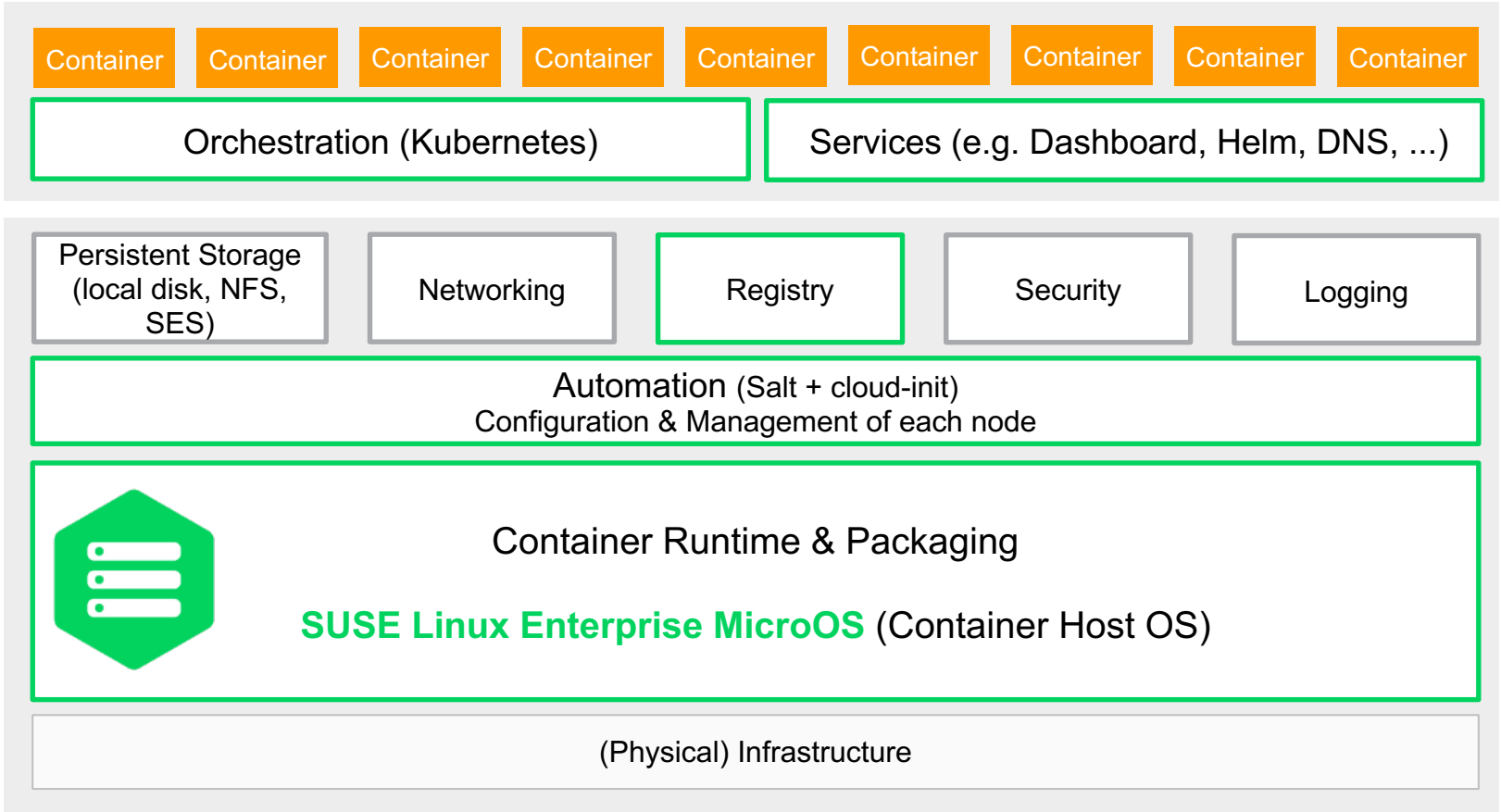
Upgrade to SUSE Cloud Application Platform

Cloud Service Economics

Offer container services

Deploy on private & public clouds

SUSE CaaS Platform Architecture



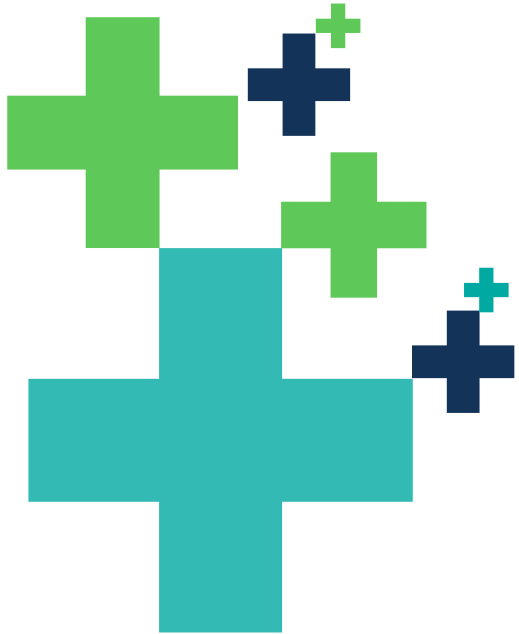
SUSE’s bet on containers management

Not only a tool for our customers!

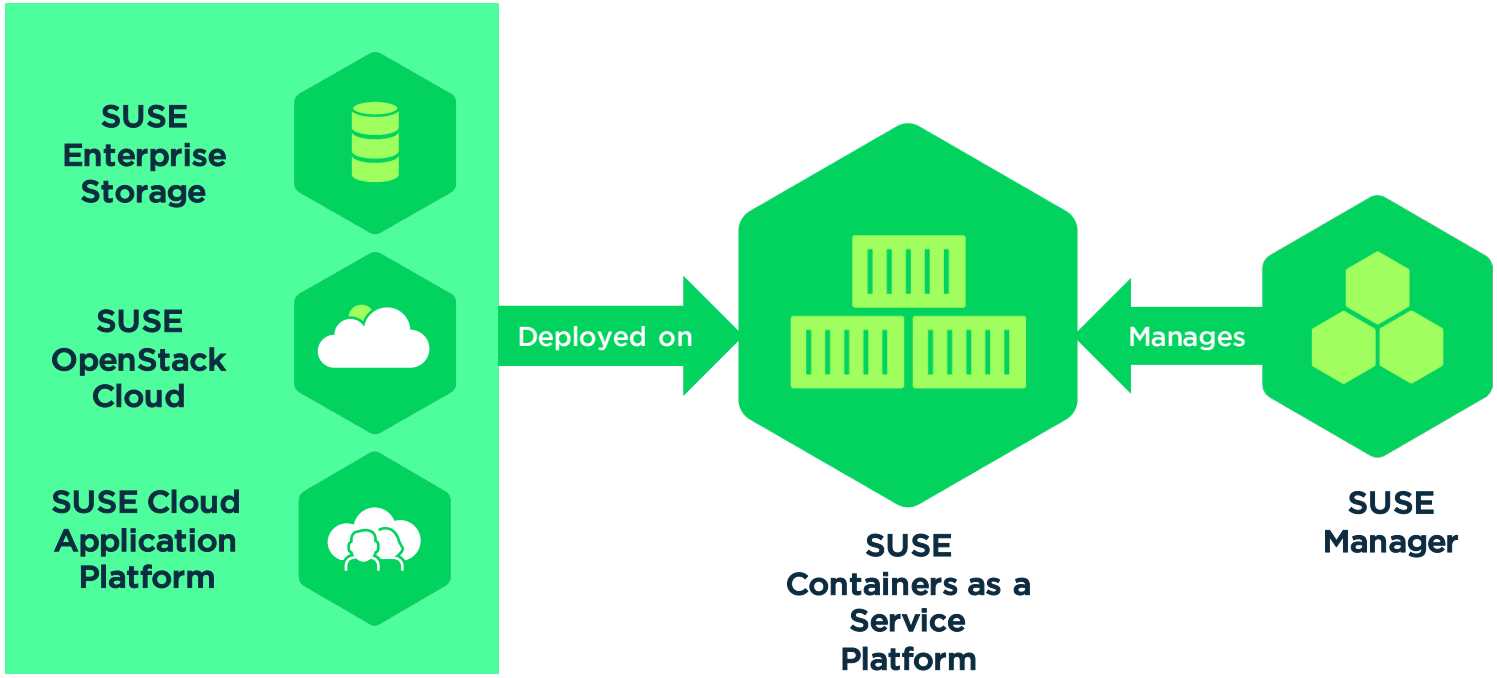
SUSE roadmap will focus on providing containerized versions of its Software Defined Data Center Management Suite

SUSE CaaS Platform will be the core deployment platform

CaaS Platform customer will benefit from broader installed base



CaaS Platform as central deployment tool



Common deployment platform

Simplify your IT ecosystem

The benefits that SUSE CaaS Platform brings:

- Maximum hardware optimization
- Maximum service density
- Hyper convergence
- Flexible deployment scenarios

SUSE CaaS Platform offers:

✓ ***A common deployment and cluster management platform.***

✓ ***Shared knowledge and admin skills***

✓ ***Simplified product lifecycle***

What’s new in SUSE CaaS Platform 2

SUSE CaaS Platform 2

What's new

SUSE CaaS Platfirn is Kuberntes 1.7 Certified

SUSE CaaS Platform 2

What's new

Simplify Kubernetes with Helm

- Create reproducible builds
- Intelligent management
- Easy to share apps
- Easy to search & use popular packages

Kubernetes 1.7

- Better Performance
- Extensible Platform

Updated MicroOS

Based on latest SLE 12 Service Pack 3

Hardware support same as SUSE Linux Enterprise Server

Streamline Public Cloud Access

Pre-defined deployment configurations

Amazon Web Services, Azure, Google

SUSE CaaS Platform 2

Role Base Access Control (RBAC)

- In enterprise settings, access might be based on **job function or role** of the user
- **Users authenticate themselves** to the system
- (Some) Users can activate **one or more roles** for themselves

SUSE CaaS Platform 2

RBAC Examples

Sys Admin	Operate the infrastructure Block access to the infrastructure level Allow developers to interact with Kubernetes
Developer	Full access for my team to manage the application No access to other teams work No access from other team to our work
Manager	Check the usage Have an overview of resources

SUSE CaaS Platform 2

Helm as package manager for Kubernetes

What is Helm?

Tool to manage Kubernetes application
Streamlines installation and management
It's like 'zypper' for Kubernetes

Helm has two parts: a client (helm) and a server (tiller).
Tiller runs inside of Kubernetes cluster, and manages releases (installations) of charts*

During SUSE CaaS Platform set up the server can be installed on the Kubernetes cluster and then Helm can be used to deploy containerized applications.

Why Helm?

Ability to deploy applications from SUSE maintained Helm charts or from 3rd party sources

Official tool to deploy containerized products such as SUSE Cloud Application Platform

Easy to integrate with SUSE CaaS Platform

*Helm will be released on the SUSE Registry by the end of 2017
"Helm Chart" is the Kubernetes equivalent of an RPM file

Kubernetes Introduction

kubernetes

What is Kubernetes (k8s)?

Complete orchestration solution for container based workloads

- Deploy
- Manage
- Scale
- Clustered

Scheduling of containers

- Critical for Micro Services and distributed applications
- Value added services for storage, network, configuration, …

Provide a way to consume containers in a developer friendly way

- Abstracts infrastructure into consumable APIs
- **Lets users manage applications, not machines**

Features

Multiple cloud and bare-metal environments	Opinionated solution – has an answer to most questions
Supports existing OSS and proprietary apps Not only “cloud-native” ones	Huge active community
Inspired and informed by Google’s experiences (largest container user to date)	Battle-tested with real-world production scenarios
100% Open Source Written in Go	All components loosely coupled Makes it simpler, composable and extensible
Interchangeable parts/modules Avoiding vendor lock-in	Part of the CNCF


Team/Org
Antifragile
DevOps
Self Healing
Auto Scaling (App & Infra)
Stateful Services
(Scheduled) Job Management
API Gateway & Service Security
Resilience & Fault Tolerance
Service Discovery & Load Balancing
Distributed Metrics & Tracing
Centralized Logging & Configuration Management
Declarative Scheduling/Placement
App Deployment (rolling updates & rollbacks)
Health Check & Recovery
Application Runtime and Packaging
Resource, Storage & Capacity Management
Environment & Container Management
CI/CD (Build server, artifact repository)
Operating System & Virtualization
Hardware, Storage, Networking
Kubernetes
IaaS

Kubernetes Vocabulary, Concepts and Architecture

Primary Concepts

Container: A sealed application package

Pod: A small group of tightly coupled Containers

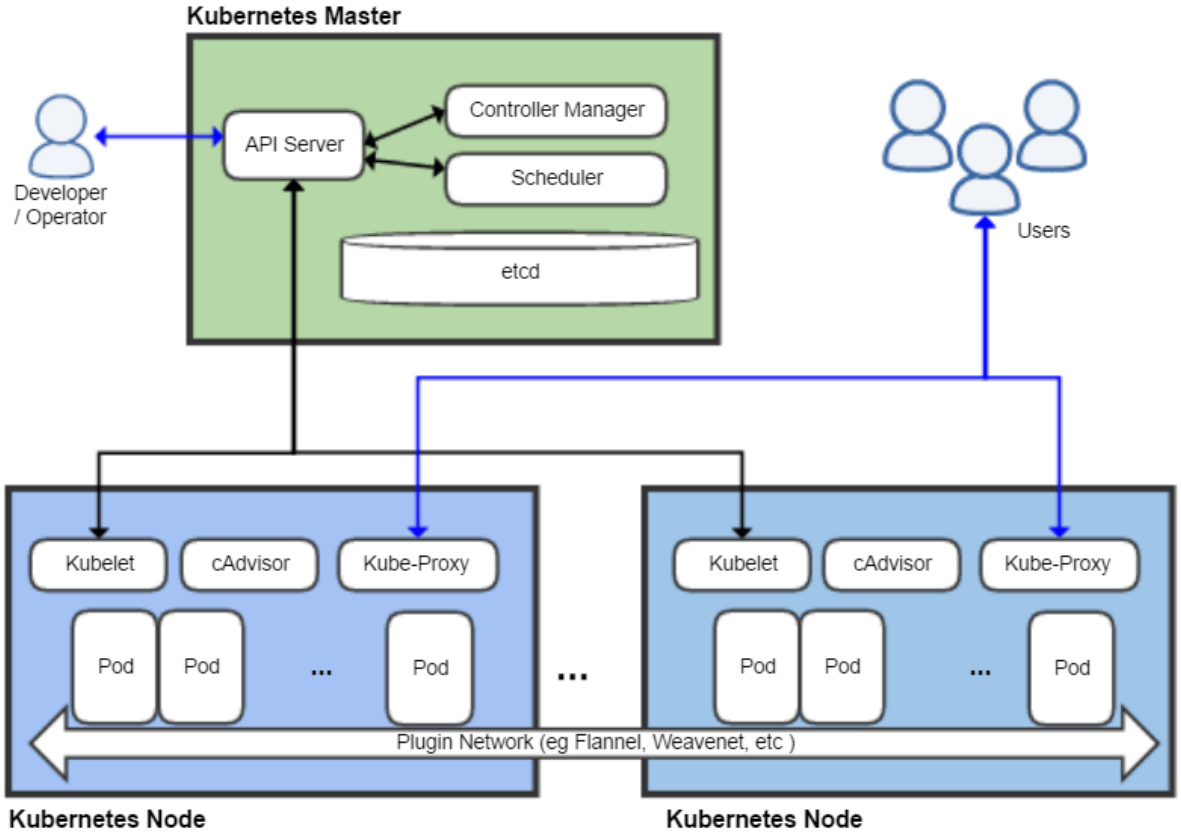
Service: A set of pods that work together and are network reachable

Controller: A reconciliation loop that is always driving the current state towards the desired state

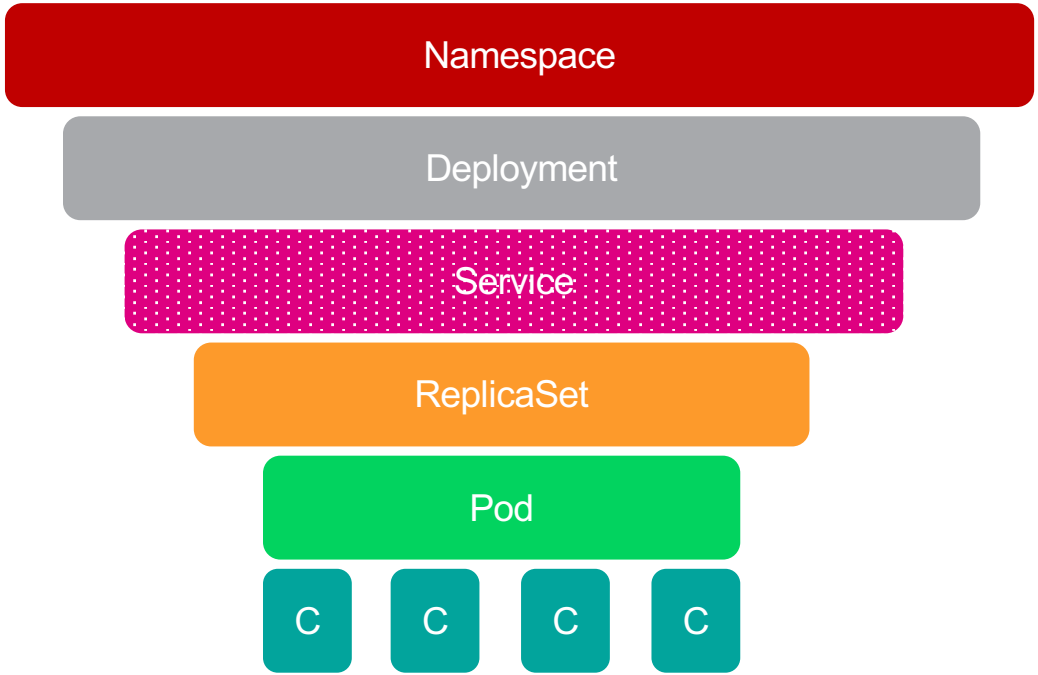
Labels: Identifying metadata attached to objects

Selector: A query against a set of labels, producing result that can have actions applied to it

Kubernetes Architecture - Components



Logical Hierarchy



Namespaces

Used to logically separate groups of cluster resources

Ideal for separating out users, teams or projects

For small clusters use is not necessary

Can be used to divide cluster resources between multiple uses (resource quota)

Names of resources need to be unique within a namespace, but not across namespaces

Use labels to separate slightly different resources (e.g versions)

Controller/Control Loops

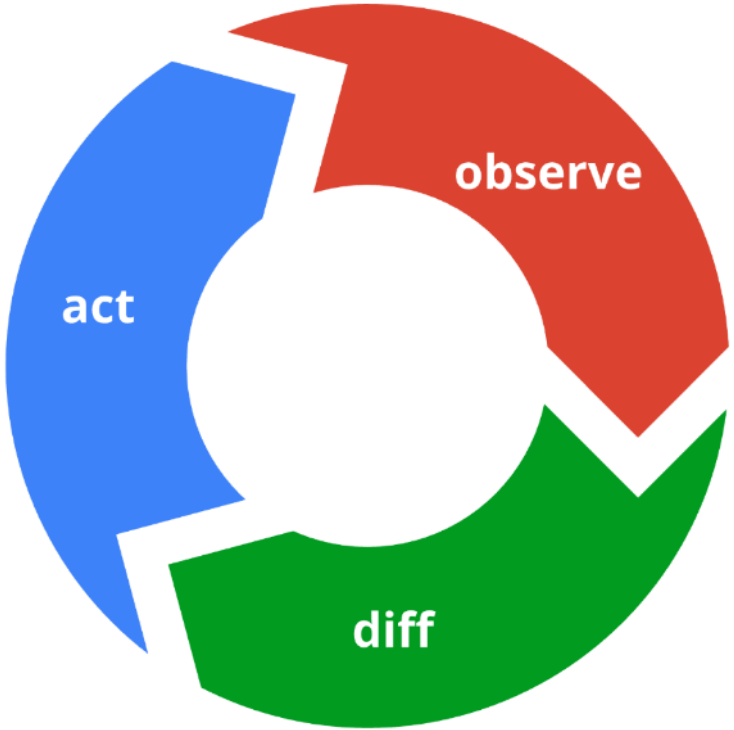
Drive current state -> desired state

Act independently

APIs – no shortcuts, back doors or hacks

Observed state is truth

Recurring pattern in the system

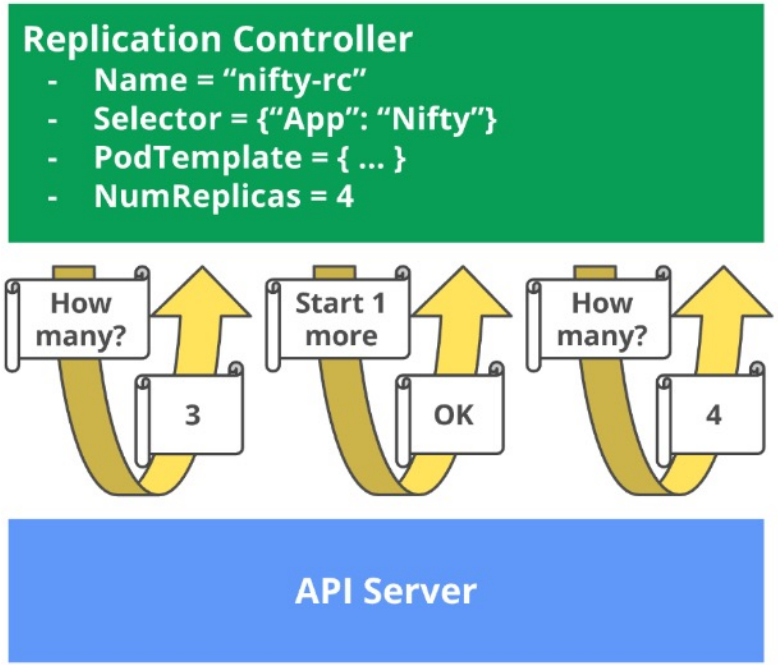


Example: ReplicationController

Replica Set (formally ReplicationController)

Type of *controller* (control loop)

Ensure N copies of a pod always running



Cleanly layered on top of the core

- All access is through public APIs

Replicated pods are fungible

- No implied order or identity

Other kinds of controllers coming soon

Pod Lifecycle

Once scheduled to a node, pods do not move

- Restart policy means restart **in-place**

Pods can be observed *pending, running, succeeded or failed*

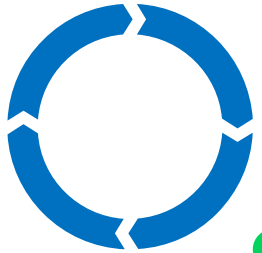
- *Failed* is **really** the end – no more restart attempts
- No complex state machine logic

Pods are restarted by the controllers not the scheduler or apiserver

- Even if a node dies
- Keeps scheduler **simple**

Apps should consider these rules

- Services hide this
- Makes pod-to-pod communication more formal



Persistent Volumes

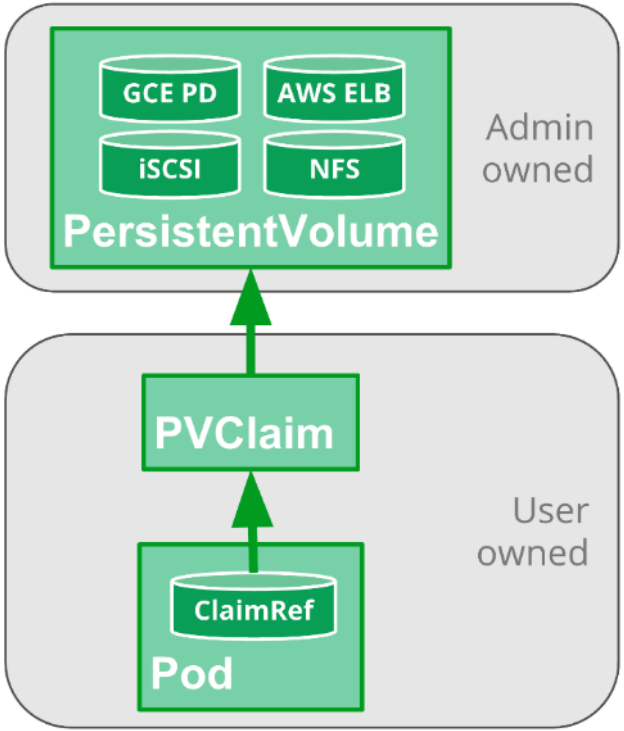
A higher-level abstraction (insulation from any one cloud environment)

Admin provisions them. Users/Pods claim them

Independent lifetime and fate

Can be handed-off between pods and lives until user is done with it

Dynamically “scheduled” and managed (like nodes and pods)



Labels

Arbitrary metadata

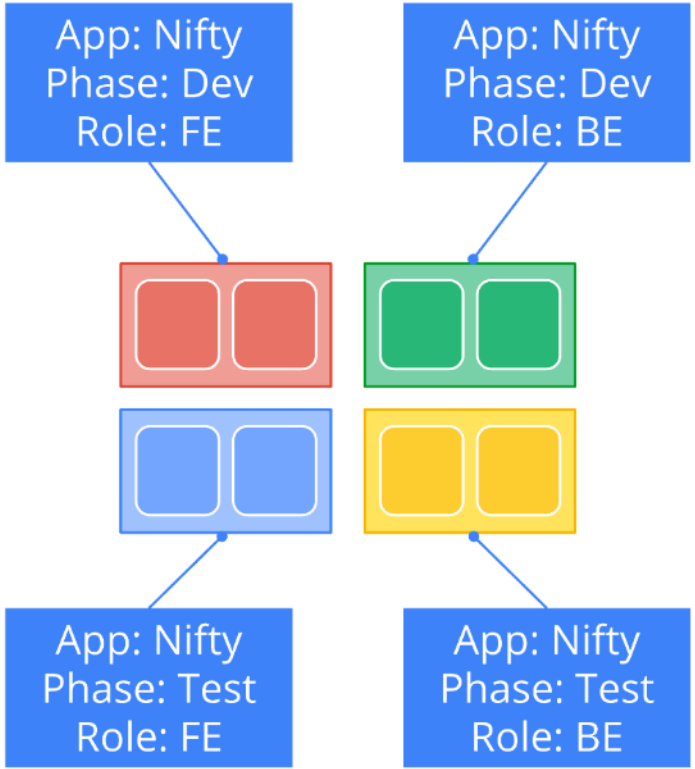
Attached to any API object

Generally represent **identity**

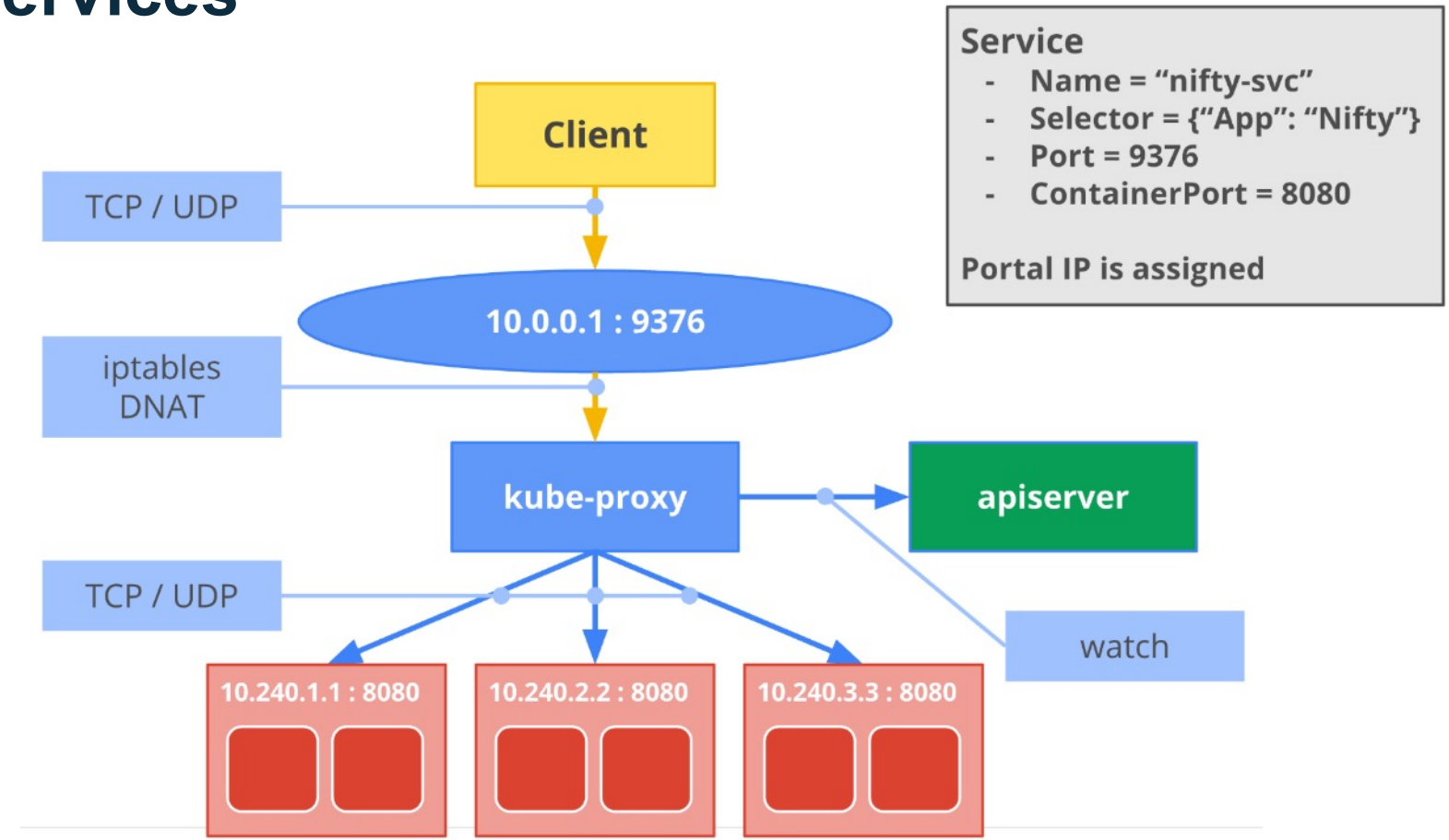
Queryable by **selectors**

The **only** grouping mechanism in k8s

Used to determine which objects to apply an operation to



Services


Service
- Name = “nifty-svc”
- Selector = {“App”: “Nifty”}
- Port = 9376
- ContainerPort = 8080
Portal IP is assigned
Client
TCP / UDP
10.0.0.1 : 9376
iptables DNAT
kube-proxy
apiserver
TCP / UDP
watch
10.240.1.1 : 8080
10.240.2.2 : 8080
10.240.3.3 : 8080

Introducing Helm: a package manager for Kubernetes

Helm Benefits

Efficient Management

Helm makes it easy to start using Kubernetes with real applications

Easy Scaling

Helm helps with the standardization of application deployments

Always up-to-date

Helm helps with application lifecycle management

Helm as the package manager

- Install curated, high quality apps
- Best practices included
- Backed by the community

Kubernetes Manifests

Life without Helm to deploy containerized application:

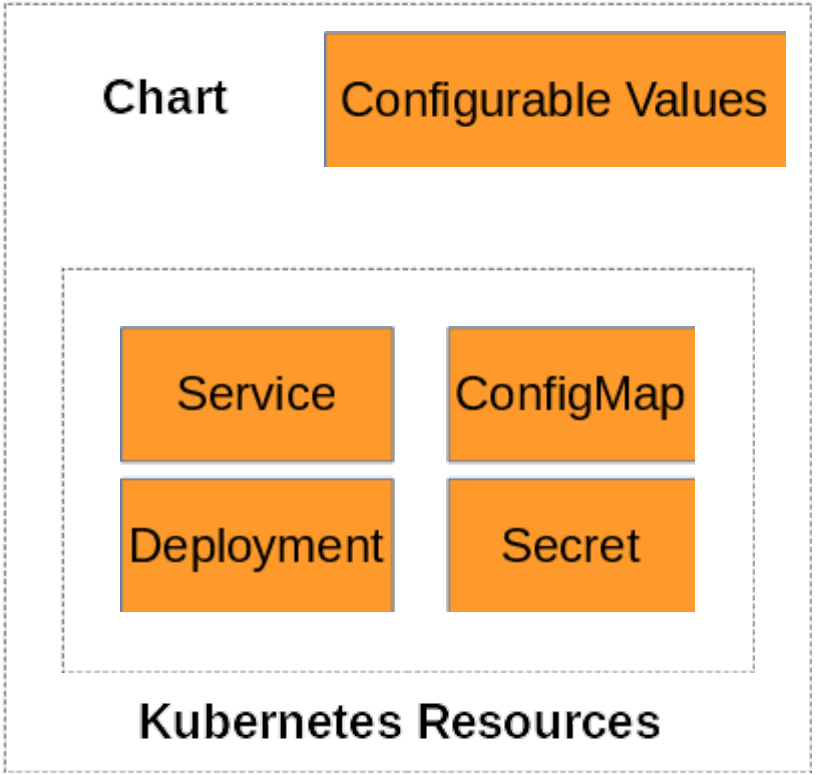
- **Write Kubernetes manifests**
- **Figure out how to share it**
- **Update resource information manually**
- **Use kubectl for managing**

```
apiVersion: v1
kind: Pod
metadata:
  name: mongo
  labels:
    another: value
spec:
  containers:
  - name: master
    image: mongo:3
    resources:
      requests:
        cpu: 100m
        memory: 300Mi
    ports:
    - containerPort: 27017
```

Another look at the chart

A chart is a logical unit of Kubernetes resources

Chats can be installed from a **remote** repository or a **local** path



Let’s take an example

Let’s take an example

Task:
Update
Go/Ruby/Java
Etc.

App

Create a Dev Branch incl. the latest changes

Create/Update the Manifest

Generate the Helm Chart

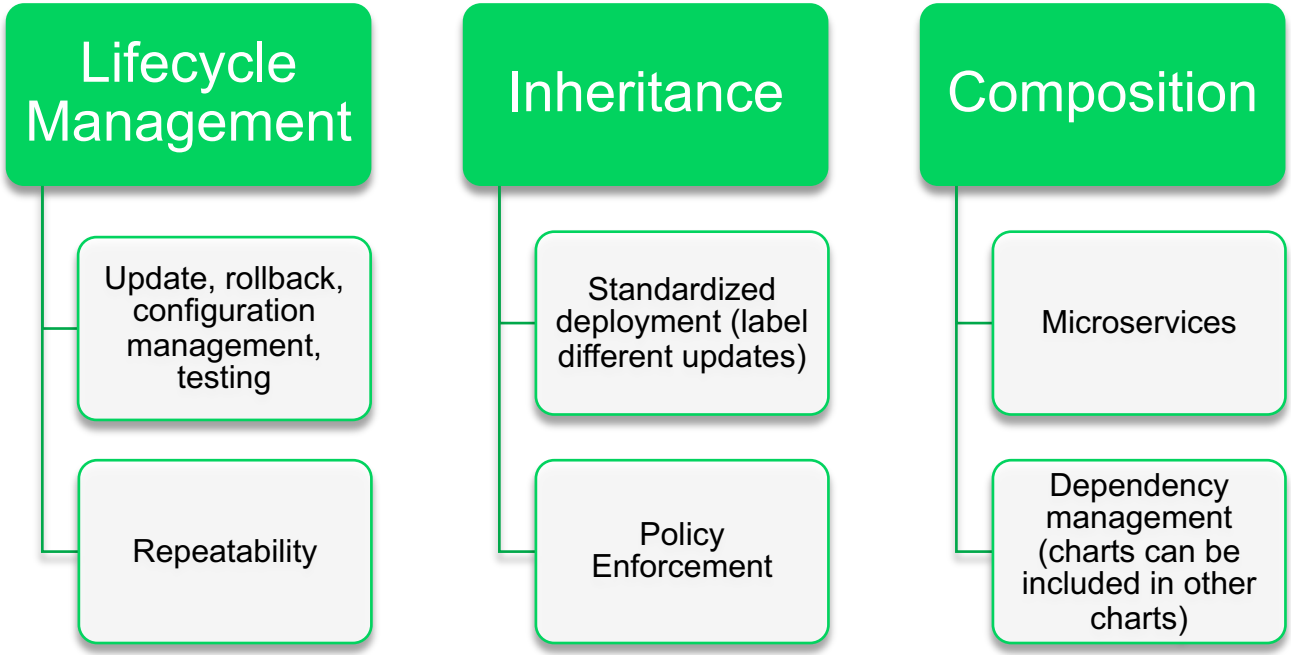
Check-out the changes

Use helm-test to check if the app is deployable on Kubernetes

Once the build is ready, push the changes to production

Multiple deployments with a “repeatable” chart

Other use cases for Helm

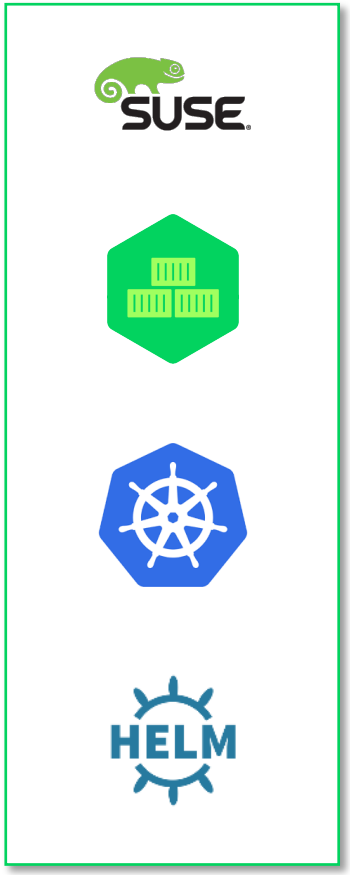


Suse provided Helm charts

All Kubernetes surrounding tooling will be provided by SUSE as Helm charts.

Together with SUSE's upcoming secure images registry, containerized software deployment at scale with Helm charts will use fully supported, optimized and audited library

The same level of quality and support as with tradtional SUSE Linux Enterprise packages.

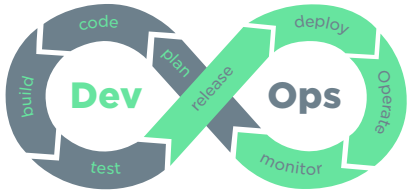


Transforming IT with Containers:

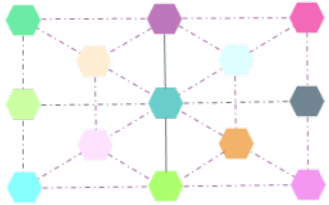
Popular Use Cases for SUSE CaaS Platform

How do containers help transform IT?

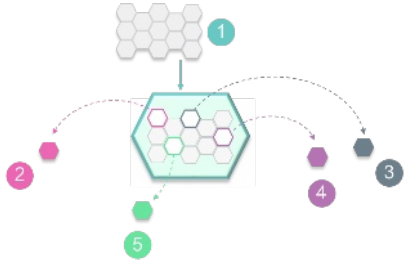
Accelerate application development and delivery



Build and deliver new cloud native applications

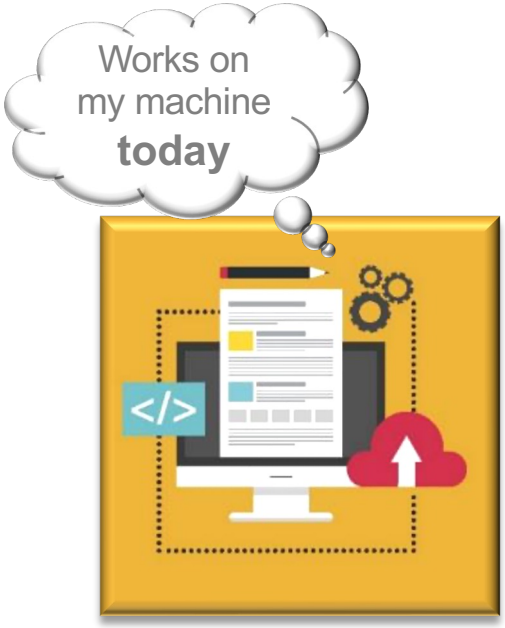


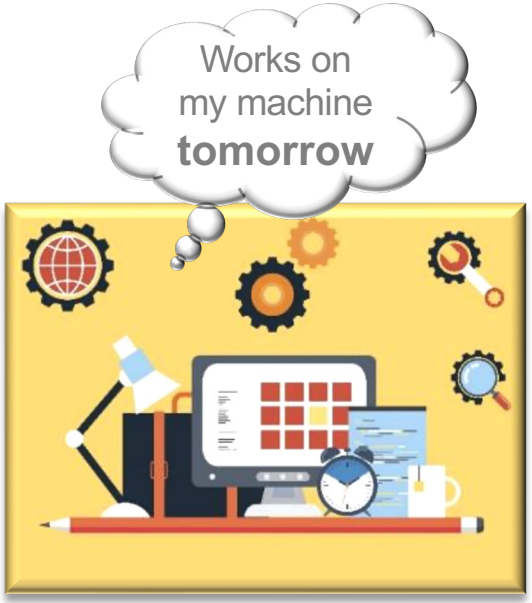
Ease application transformation



Accelerate Application Development & Delivery

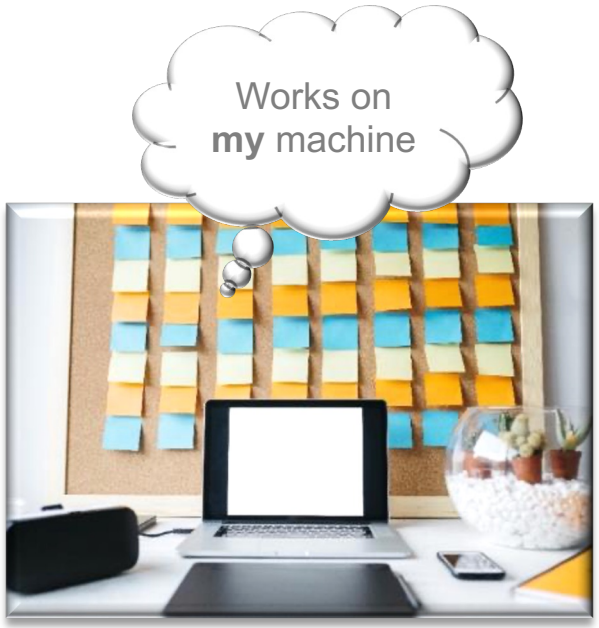
Spend less time deploying, more time developing





Accelerate Application Development & Delivery

Collaborate more easily

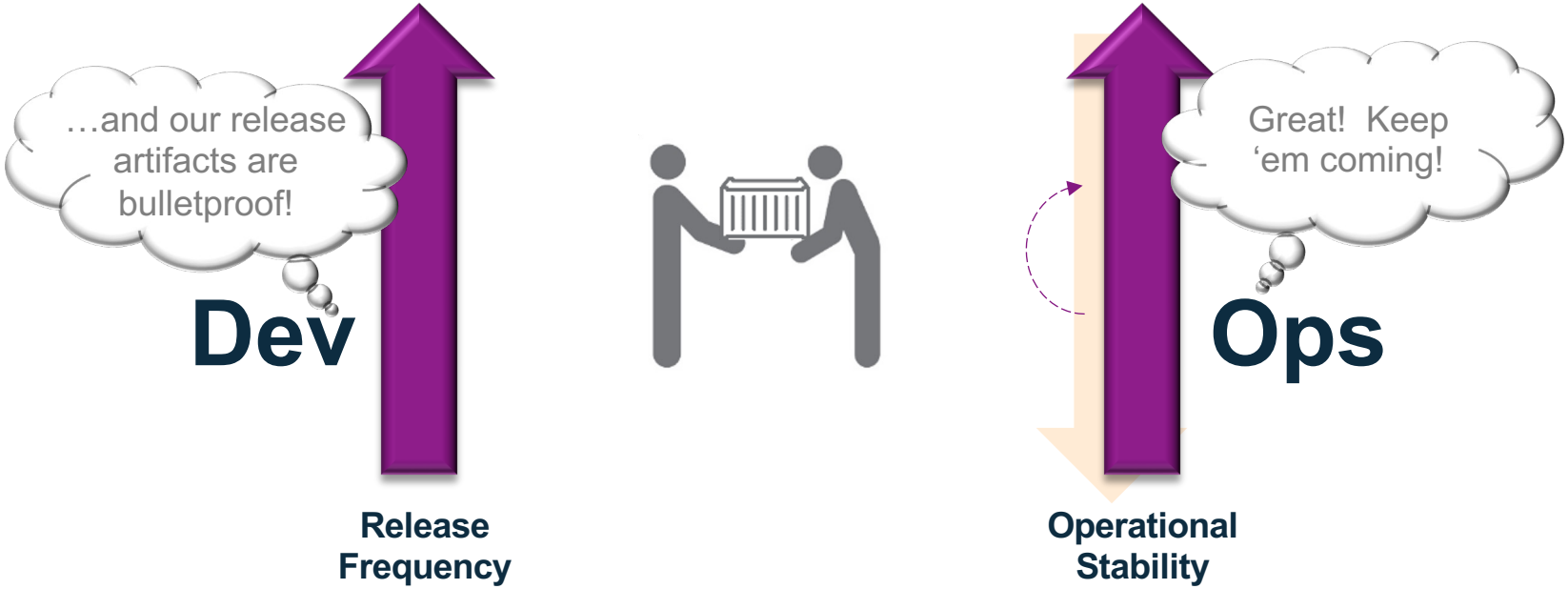


Accelerate Application Development & Delivery

Release more frequently

Accelerate Application Development & Delivery

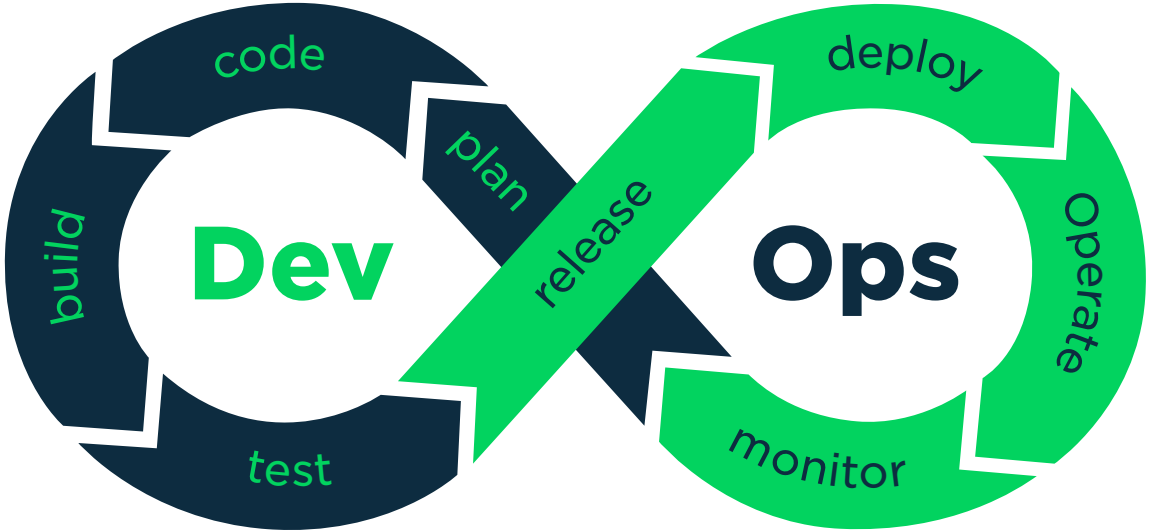
Release more frequently



Accelerate Application Development & Delivery

Speed pipeline execution

Reduce cycle times



Redesign the company website with new technology

Use modern Ruby on Rails technology

Without Containers

Isolated development environment

Small team builds a POC

POC hand over to Dev/Ops teams

One of the following issues is likely:

- POC doesn't run in production/staging
- Development environment not easy to setup (different OS, different tools, missing packages)

POC is not usable, can't handover from coding to production.

With Containers

Team creating POC can also deploy

POC deployed with containers

POC runs in a staging/production environment

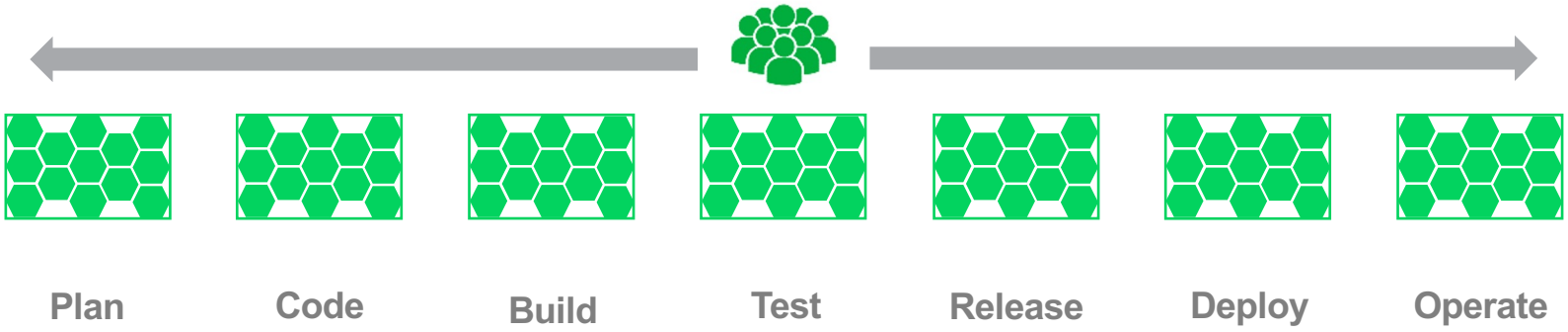
New dev environments are easy to setup

Developers can continue using the preferred tools and OS

Happy Developers! Productive teams!

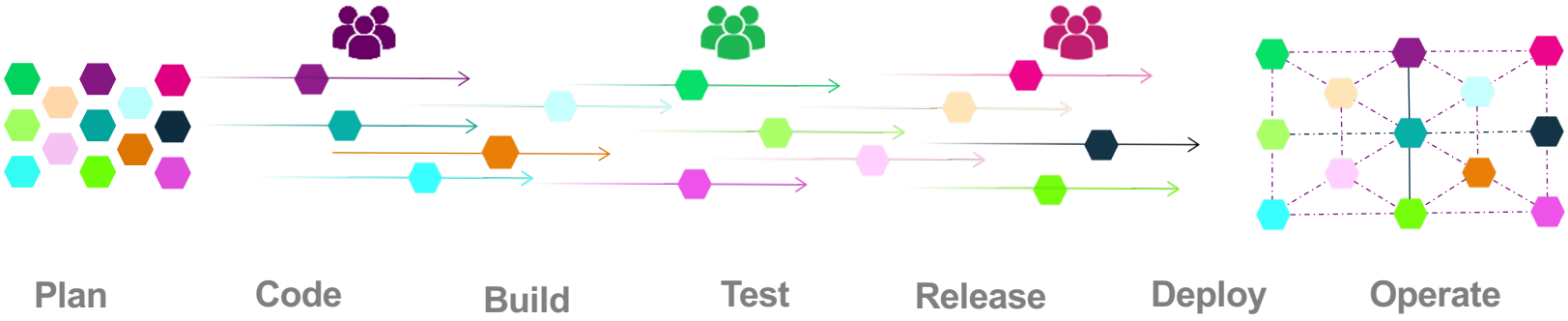
Build and Deliver Cloud Native Applications

Instead of larger, monolithic services …



Build and Deliver Cloud Native Applications

… increase agility with microservices

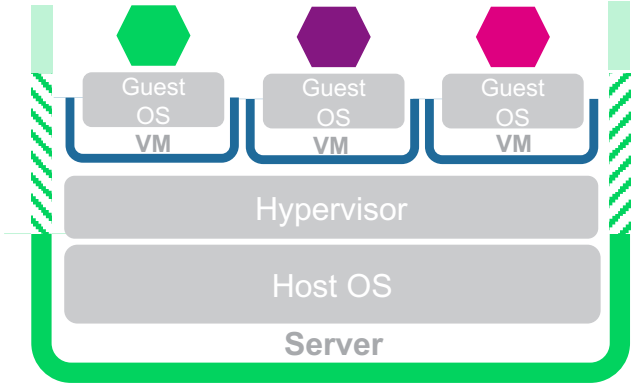


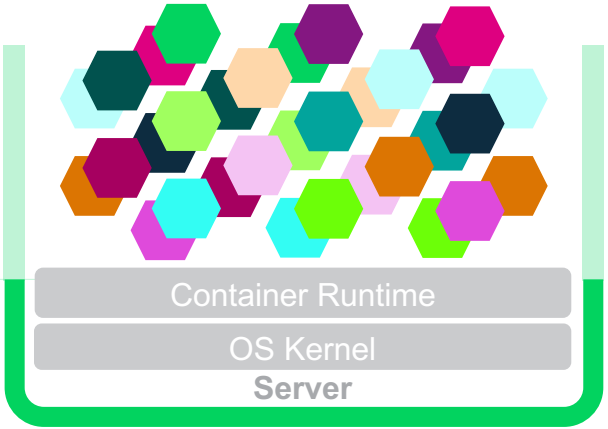
smaller codebase + smaller teams + independent functionality =

Fast IT

Build and Deliver Cloud Native Applications

Containers enable microservices model





Large numbers of small containers require efficiency of shared OS Kernel

Continuous delivery demands fast start and stop capability

Deploy microservices

Microservices help companies move faster. Containers make it work!

Without Containers

Set up development environments

Download, Run all services locally

(Try to) keep all services up to date

Check code against dependencies

Submission of code becomes difficult

With Containers

Containers are designed for microservices

Setup a development namespace

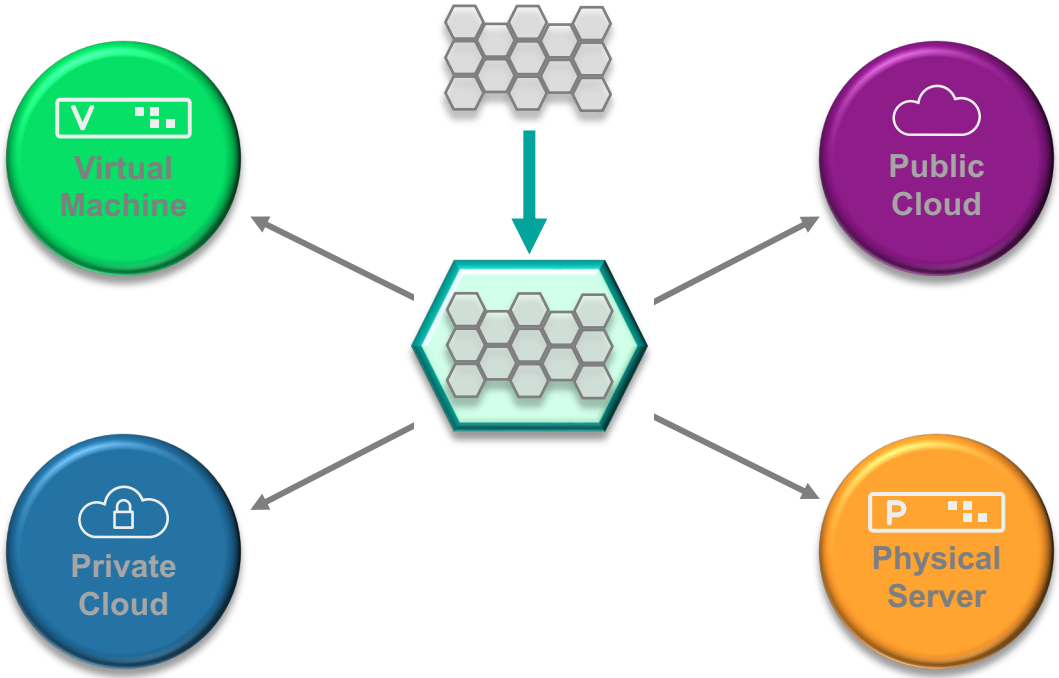
Developers push code into dedicated containers -> saves lot of time

No need to keep code locally

Continuous Integration/Continuous Development (CI/CD) run against all services

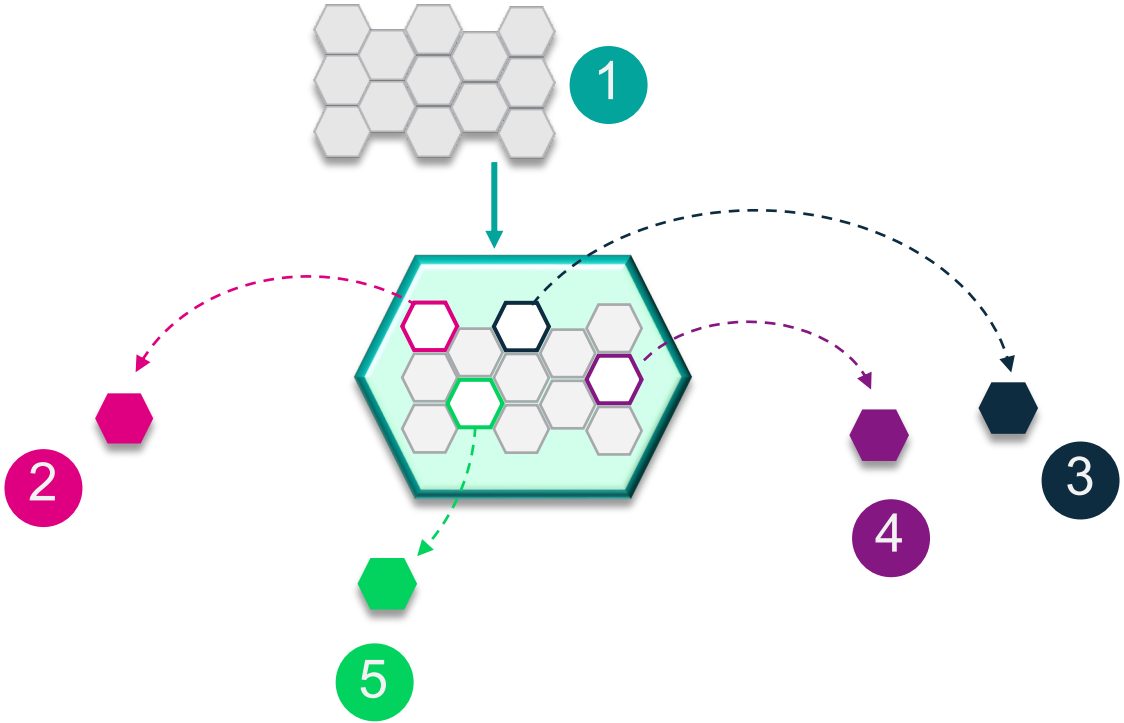
Ease Application Transformation

Re-deploy to virtual or cloud infrastructure



Ease Application Transformation

Modernize application architecture



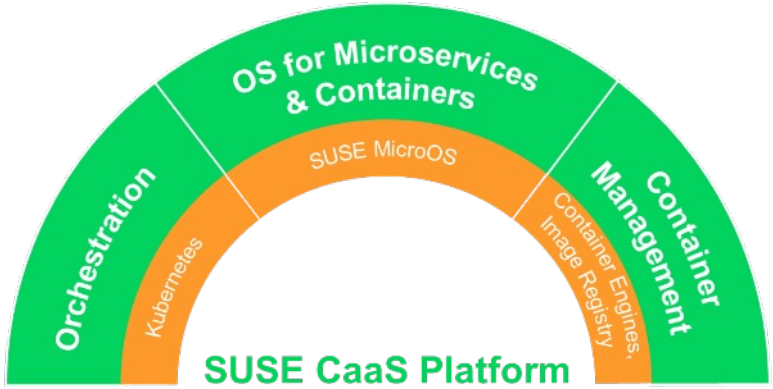
Summary

3 Key Benefits

Achieve faster time to value

Simplify management of your container platform

Maximize return on your investment



Resources

Resources

Webpage suse.com/caas

Product Flyer Link

Analyst Paper

- SUSE slices deeper into container market – 451 Research

Webinars/Videos

- ChalkTalk: Containers 101 - A SUSE Perspective
- ChalkTalk: Overview of SUSE CaaS Platform
- SUSE Container as a Service Platform – An Introduction
- Installing a SUSE CaaS Platform 1.0 Beta 2 Cluster
- Transforming IT with Containers & the SUSE CaaS Platform
- Technical Demo: SUSE MicroOS Demo
- Technical Demo: Deploying SUSE CaaS Platform & Container-based Applications
- Technical Video: SUSE CaaS Platform Setup - Nothing to Everything in One Hour

Blogs

- SUSE CaaS Platform Setup – From Nothing to Everything in 1 Hour
- Rise of the CaaS Platform
- What is SUSE CaaS Platform? What's all the buzz about?
- Container Orchestration with Kubernetes
- Running Containers at Scale
- Introducing Kubic Project: open-source project for SUSE CaaS Platform
- The value of SUSE CaaS Platform in today's world


SUSE
We adapt. You succeed.