SUSE
We adapt. You succeed.

Rethinking the OS

A travel journal

Simona Arsene

As almost all stories, also mine starts with a dream...

Engineers wished

- OS focused only on containers
- Stripped down system designed for one use case
- Transactional Updates
- Focused on large deployments
- Reduced end-user interactions
- An always up-to-date Operating System

Customers & Community wanted

- A small and easy to manage OS
- A fast way to setup a cluster and to manage multiple nodes
- An always up-to-date Operating System
- Safe way to update the system
- Ready to run
- Kubernetes

Let’s start by challenging everything

What do we currently have:

- Multipurpose operating system
- Flexibility over preconfiguration
- RPMs
- Pets, lots of them
- Community
- Skilled engineers with a “vision”

… and I mean everything!

What is the current in trend:

- Atomic Updates, no RPMs
- Single use case
- Fully containerized solution
- Focus on size
- Several orchestration solutions

Draw the line

Why “wanting it all” if you only have ONE use case to cover?

Dependencies: Rule the whole stack

Make base and upper layer work hand in hand

Enterprise consumable

Mix and match induced instability

Immutable infrastructure

Real cattles, not fake pets

Understanding the scale

Think of clusters since day one

Know *exactly* what runs on you machine

Less surface for attacks

Auto updates suggested but not enforced

Real Persona

Not only “DevOps” dreams

Community focus

Kubic, the very same but based on openSUSE

In medio stat virtus [0]

[0] Virtue stands in the middle

Turn dreams into reality:

- **Ready to run out of the box**
  - One page installer
  - Customization possible, not necessary
  - Sane defaults and supported deployments out of the box

- **Don’t drop the user in a bash prompt!**
  - Having an up and running cluster is why the user is deploying in the first place.
  - Not everyone is a K8S expert

- **Administration Node with dashboard to manage the cluster**

Leverage your strengths:

Btrfs with snapshots and rollback for transactional updates

A “transactional update” is a kind of update that:

- **Is atomic**
  - Either fully applied or not at all
  - The update does not influence your running system

- **Can be rolled back**

Leverage your strengths:

- **Read-only filesystem with overlayfs for /etc**
  - Base OS and snapshots are read-only
  - Subvolumes to store data are read-write
  - Example: /var/log, /var/cache, /var/crash and similar directories
  - Use overlayfs for /etc (for cloud-init and salt)
  - Introduce /var/lib/overlay/{work,etc} for overlayfs
- **Cloud-init for initial configuration (Network, Accounts, Salt)**
- **SALT for full system configuration**

Introducing SUSE MicroOS

What is SUSE MicroOS

A purpose built Operating System designed for **microservices** & **containers** and optimized for **large deployments**.

Term “Micro” in MicroOS signifies Microservices.

Key Features

An always up-to-date Operating System

An easy to manage/upgrade OS

Easily setup/manage a cluster of nodes

Scalable — up to 1000s of nodes

Upstream First!

openSUSE and beyond

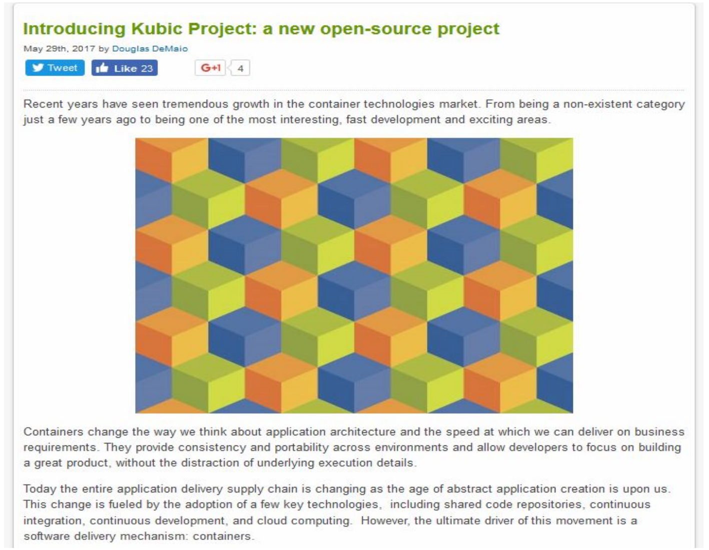
- Stay as close as possible to upstream
- openSUSE as first class citizen
- openSUSE is not enough
- Portus [0]
- Kubic [1]
- Transparent direction

[0] https://github.com/SUSE/Portus

[1] https://github.com/kubic-project

openSUSE and beyond

- Stay as close as possible to upstream
- openSUSE as first class citizen
- openSUSE is not enough
- Portus [0]
- Kubic [1]
- Transparent direction



Introducing Kubic Project: a new open-source project

May 29th, 2017 by Douglas DeMaio

Recent years have seen tremendous growth in the container technologies market. From being a non-existent category just a few years ago to being one of the most interesting, fast development and exciting areas.

Containers change the way we think about application architecture and the speed at which we can deliver on business requirements. They provide consistency and portability across environments and allow developers to focus on building a great product, without the distraction of underlying execution details.

Today the entire application delivery supply chain is changing as the age of abstract application creation is upon us. This change is fueled by the adoption of a few key technologies, including shared code repositories, continuous integration, continuous development, and cloud computing. However, the ultimate driver of this movement is a software delivery mechanism: containers.

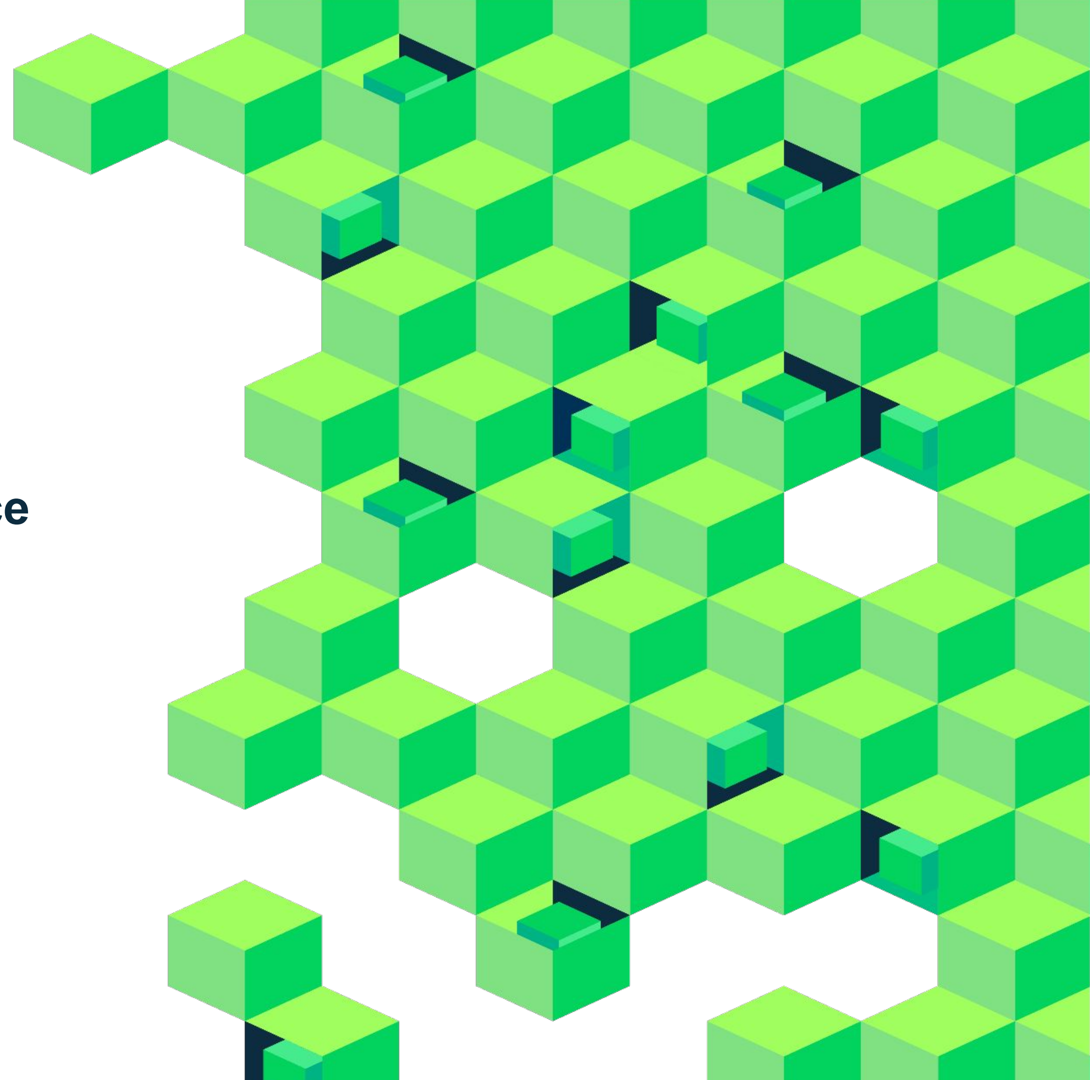
[0] https://github.com/SUSE/Portus

[1] https://github.com/kubic-project

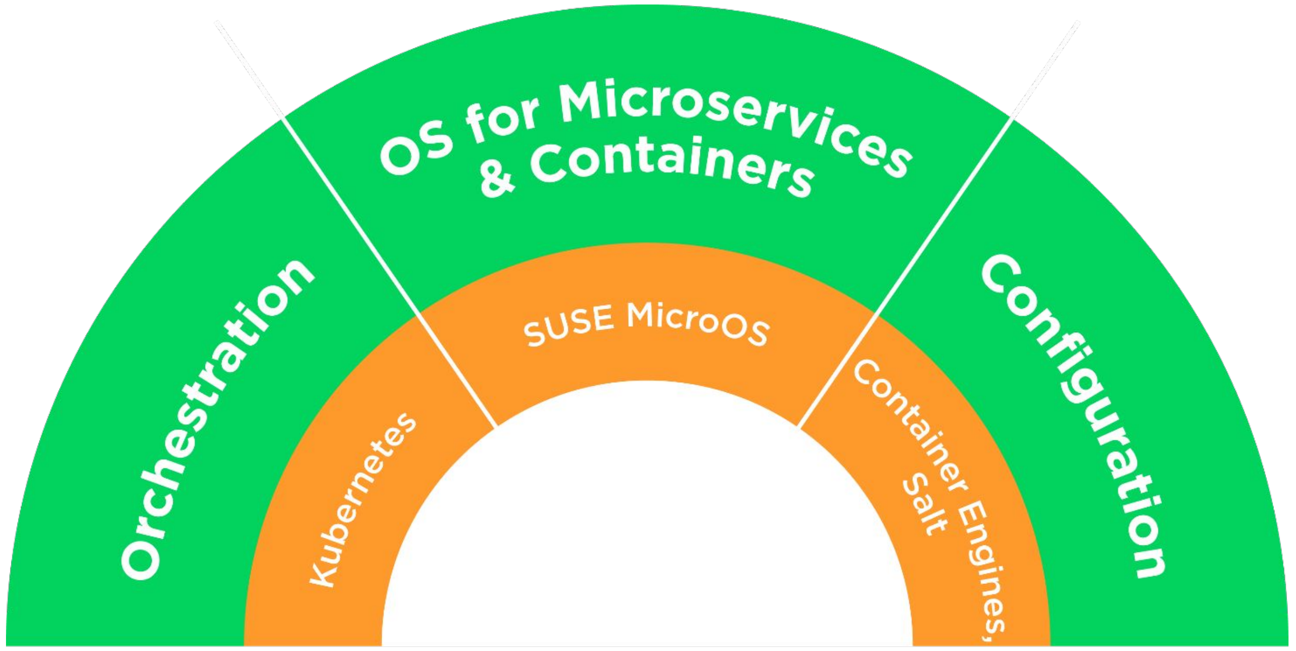
Introducing

SUSE Container as a Service Platform

Launch: today
General Availability: Aug 3



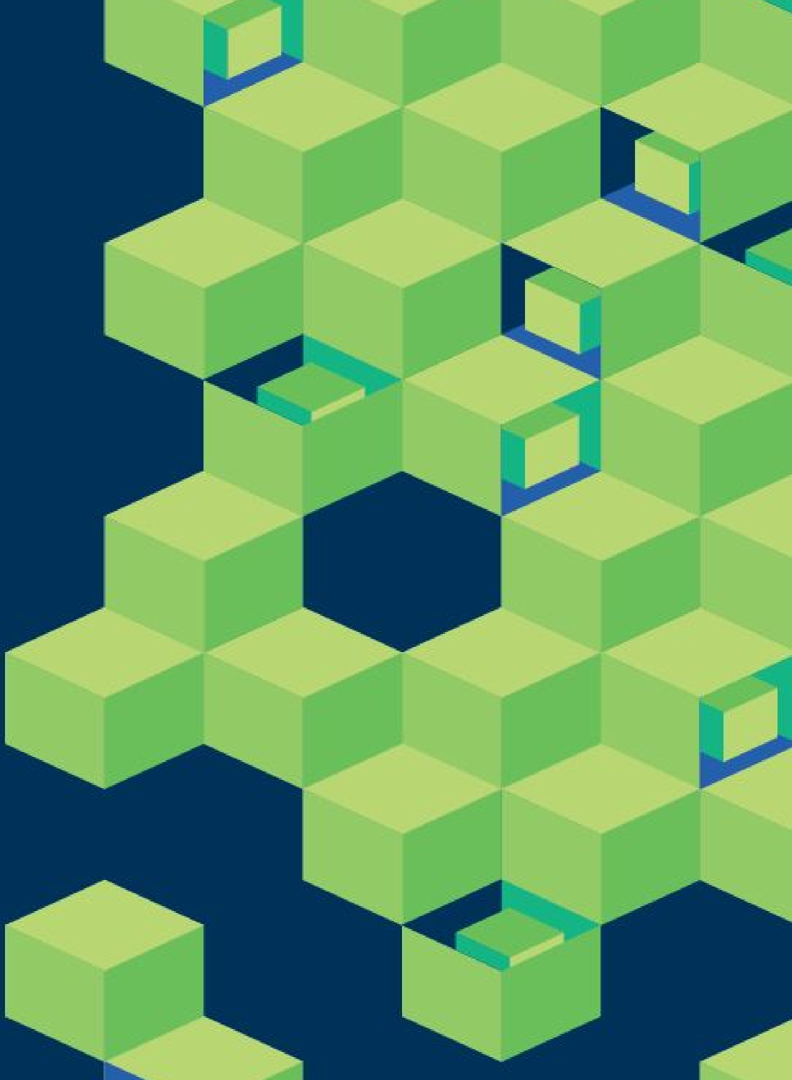
3 Key Technology Components



SUSE CaaS Platform

利用SUSE CaaS平台 轻松构建和运行容器应用

Build and Run Container Apps with Ease Using SUSE CaaS Platform



- 支持DevOps
  Enable DevOps
- 创建基于微服务的应用
  Create microservices-based apps
- 实现容器管理的自动化
  Automate container management

Thank you!

SUSE®

We adapt. You succeed.