Cloud Native

Infrastructure, Patterns, and Technology

Andreas Jaeger
Product Manager
aj@suse.com
Agenda

- Cloud Native
- Infrastructure, Patterns, Technology
- SUSE CaaS Platform
Changing Business Demands are Transforming IT

**IT Infrastructure**
- Datacenter
- Hosted / Managed
- Cloud

**Application Deployment**
- Physical Servers
- Virtual Servers
- Containers

**Application Architecture**
- Monolithic
- N-Tier
- Microservices

**Development Process**
- Waterfall
- Agile
- DevOps
Scaling

More developers
More and larger applications
Availability, Rapid Change at Scale
Availability, Rapid Change at Scale

Shifting the Curve...

https://www.slideshare.net/AmazonWebServices/dmg206
Cloud Native
Cloud Native Elevator Pitches

Container packaged, dynamically managed, micro service oriented.

Declarative, dynamic, resilient, and scalable.

Cloud Native is structuring teams, culture and technology to utilize automation and architectures to manage complexity and unlock velocity.

Justin Garrison [1]

Joe Beda [2]

[1] https://lists.cncf.io/g/cncf-toc/message/1554
[2] https://blog.heptio.com/cloud-native-part-1-definition-
Cloud Native?

Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil.

CNCF Cloud Native Definition v1.0, see https://github.com/cncf/foundation/blob/master/charter.md
Cloud Native

- Container packaged
- Dynamically managed
- Micro service oriented
Microservices

Assemble small components

• Loosely coupled
• More agile systems
• Use of third-party services

Independent deployable
Independent scalable
Micro Services for Scaling

More developers
More and larger applications
Container Packaged

Standardized unit of software
Application with all dependencies
Isolated execution
Universal Deployment
Portable
Lift & Shift vs Cloud Native

**Lift & shift:**
- Moving to cloud, not taking advantage of it

**Cloud ready:**
- Working with cloud

**Cloud native:**
- Embracing cloud infrastructure
Why should I use it?

- Reduce operating costs
- Deliver faster
- Handling scale, resiliency and security in a superior way
Cost of using it?

Continuously learn new paradigms, skills
Adopt new technology
Accept constraints of platforms
Patterns, Technology, Infrastructure
Kubernetes and Container for Cloud Native

**Kubernetes:**
- declarative
- efficient scheduling
- extensible API

**Container:**
- portable
- immutable
- reproducible
Kubernetes is a new container technology leader
Container orchestration and management

Orchestration
• Scheduling
• Service discovery

Performance and availability
• Scaling
• Load balancing
• Self-healing
• Monitoring

Maintenance
• Rollout
• Rollback
Reconciler Pattern

1. Get current state
2. Get expected state
3. Reconcile – ensure current state is expected
Service Mesh

Pod

Pod

Pod

Pod
Service Mesh – A networking model

Examples:
- Linkerd
- Envoy
- Istio
Istio – Securing Services with a Service Mesh

- Automatic load balancing for HTTP, gRPC, WebSocket, and TCP traffic.
- Fine-grained control of traffic behavior with rich routing rules, retries, failovers, and fault injection.
- A pluggable policy layer and configuration API supporting access controls, rate limits and quotas.
- Automatic metrics, logs, and traces for all traffic within a cluster, including cluster ingress and egress.
- Secure service-to-service communication in a cluster with strong identity-based authentication and authorization.

Homepage https://istio.io/
Istio Architecture

https://istio.io/docs/concepts/what-is-istio/
Istio: Envoy as Proxy

High-performance proxy, deployed as side car (no need to change applications!):

- Dynamic service discovery
- Load balancing
- TLS termination
- HTTP/2 and gRPC proxies
- Circuit breakers
- Health checks
- Staged rollouts with %-based traffic split
- Fault injection
- Rich metrics
Jaeger - Tracing

Distributed tracing system
Open source by Uber Technologies
Monitoring and troubleshooting microservices-based distributed systems, including:
- Distributed context propagation
- Distributed transaction monitoring
- Root cause analysis
- Service dependency analysis
- Performance / latency optimization
Needs instrumentation of applications
Homepage: https://www.jaegertracing.io
Function as a Service (FaaS) - Serverless

Running backend code without managing your own server systems or your own long-lived server applications. Implemented using containers!

Steps for functions:
• Write function
• Deploy to FaaS infrastructure
• Use it from application – and FaaS infrastructure triggers start, stop, scale of functions

Steps for applications:
• Use FaaS framework
• Call a remote function in FaaS
Example of FaaS Frameworks

AWS Lambda
OpenFaaS
OpenWhisk
...

Caveat: Security

Different mindset – developer deliver applications with dependencies

Take care of:

- Vulnerable images
- Inter-container communication
- Separate users
Advertisement Time
SUSE CaaS Platform
Speed application delivery to improve business agility

SUSE CaaS Platform is Kubernetes-based container management solution used by application development and DevOps teams to deploy, manage, and scale container-based applications and services.
SUSE CaaS Platform simplifies and extends Kubernetes Container management for the enterprise
### Running on Kubernetes

#### Infrastructure & Lifecycle Management
- SUSE Manager
- SUSE OpenStack
- Cloud Monitoring

#### Application Delivery
- **Container Management**
  - SUSE CaaS Platform
- **Platform as a Service**
  - SUSE Cloud Application Platform

#### Software-Defined Infrastructure
- **Private Cloud / IaaS**
  - SUSE OpenStack Cloud
- **Compute**
  - Virtual Machine & Container
- **Storage**
  - SUSE Enterprise Storage
- **Networking**
  - SDN and NFV
- **Multimodal Operating System**
  - SUSE Linux Enterprise Server

#### Physical Infrastructure: Multi-platform Servers, Switches, Storage
Some Related SUSECON Sessions

Continuous Application Delivery on SUSE CaaS Platform, HO1023
SUSE CaaS Platform Hands-on, HO1209
Enabling Business Agility with SUSE CaaS Platform, BOV1078
Roadmap SUSE CaaS Platform, FUT1431
Questions
General Disclaimer
This document is not to be construed as a promise by any participating company to develop, deliver, or market a product. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. SUSE makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. The development, release, and timing of features or functionality described for SUSE products remains at the sole discretion of SUSE. Further, SUSE reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All SUSE marks referenced in this presentation are trademarks or registered trademarks of Novell, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners.