TUT1131 - Best Practices in Deploying SUSE CaaS Platform

Martin Weiss
Senior Architect Infrastructure Solutions
Martin.Weiss@SUSE.com

Juan Utande Herrera
Senior Architect Infrastructure Solutions
Juan.Herrera@suse.com
### AGENDA

1. **What is SUSE CaaS Platform**
2. **Requirements**
3. **Planning and Sizing**
4. **Deployment Best Practices**
5. **Testing**
6. **Operations**
What is SUSE CaaS Platform 3
SUSE: Underpinning Digital Transformation

**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

**Services**

- SUSE Global Services
- Consulting Services
- Select Services
- Premium Support Services

**Open, Secure, Proven**
What is SUSE CaaS Platform 3?

- Kubernetes
- MicroOS with Transactional Updates
- Simple deployment
- SUSE supported
- LDAP / Active Directory Integration
- Caching Registry Integration
- Air Gapped Implementation Support
- Registry.suse.com
- Helm
- Docker or Cri-o (tech preview), Flannel
- Multiple deployment methods
Requirements
## General requirements

<table>
<thead>
<tr>
<th>Where to deploy</th>
<th>What do I need</th>
<th>Who can help me</th>
<th>Support options</th>
</tr>
</thead>
</table>
| • Deploy on physical Hardware or on your Virtualization infrastructure  
• Ready to Run on Public and Private Clouds | • SUSE CaaS Platform subscriptions  
• SLES for infrastructure nodes | • Sales and Pre/Post Sales Consulting:  
- Help choosing the right Hardware  
- Architect the solution  
- Initial implementation | • Included 24/7 priority support in case of issues  
• Consulting for maintenance and proactive support to scale, upgrade, review and fix |
Use Case Specific Requirements

Application Requirements (Sizing)
- Number of Pods
- Memory, CPU
- Storage requirements (file, block, object, single or multi-writer, capacity, static or dynamic provisioning)
- Specific Hardware / CPU / GPU requirements
- Network Entry points / Services / Bandwidth

Security Requirements
- Images (source and size)
- Isolation
- Integration into existing Identity Sources

Availability Requirements
- Single or multi data-center
- Distance / Latency

$$$ BUDGET $$$
Politics, Religion, Philosophy, Processes ;(-)
Planning and Sizing
Planning and Sizing

SUSE CaaS Platform - CLUSTER 1

- Admin
- LDAP, Salt, Velum, SQL

Kubernetes

- Master (3)
- Fault tolerance
- ETCD cluster
- Based on number of pods

Workers as VM or physical

Second cluster:
- Fault tolerance
- Disaster recovery

SUSE CaaS Platform - CLUSTER 2

Kubernetes

- Admin
- Master
- Worker

Disk Space for each Worker:
- 50 GB for OS (BTRFS minimum for OS)
- 100 GB for /var/lib/docker (BTRFS for Images and Containers)
- Space really depends on image sizes, versions and changes
Deployment Best Practices
Deployment - Processes and People

Prepare the Team (DevOps?)
– Server
– Storage
– Network
– Application
– Security
– User

Other
Review the Design
• Depending on the requirements adjust before implementation

Hardware Installation
• Ensure that hardware installation and cabling is correct
• Update Firmware
• Adjust Firmware / BIOS settings

Disable everything not required (i.e. serial ports, network boot, power saving)

Configure HW date/time

VM Preparation
• Use paravirtual SCSI

Deployment

Preparation of Time Synchronization
• Have a fault tolerant time provider group

Name Resolution
• Ensure that all addresses of the servers have different names
• Add all addresses to DNS with forward and reverse lookup
• Ensure DNS is fault tolerant
• /etc/HOSTNAME must be the name in the public network
• Define and create DNS Entries for internal and external Velum and API targets (Cname, Load Balancer, no round robin)
Deploy On-Premise Registry (docker-distribution-registry)

- Implement Portus to Secure the On-Premise Registry
- Create DNS entry for Registry
- Create Namespaces and Users on Registry
- Optional: Integrate Portus into existing LDAP or Active-Directory

Put all required images into registry into the right namespace

- Dashboard, Prometheus, Grafana, etc.

Optional: Setup caching registries
Prepare Load Balancer Endpoints for API and DEX

- Port 6443 and 32000

Storage Network setup and connectivity

Prepare on-premise helm chart repository

Prepare docker host to pull from internet, scan images, push to on-premise registry

Prepare GIT for storing all manifests / yaml files
Deployment

Software Staging
- Subscription Management Toolkit, SUSE Manager, RMT (limited)
- Ensure staging of patches to guarantee same patch level on existing servers and newly installed servers

General
- Use BTRFS for the OS
- Disable Firewall / AppArmor / IPv6

AutoYaST
- Ensure that all servers are installed 100% identical
- Consulting solution available (see https://github.com/Martin-Weiss/cif)

Configuration Management
- Templates
- Salt
Deployment

1. ONLY USE STATIC IP Configs
2. Verify Time Synchronization
3. Verify Name Resolution
4. Test all Network Connections
   - Bandwidth
   - Latency
Deployment

- Install all Servers (Admin, Master, Worker) via AutoYaST
- Ensure that all the patches available are installed at this point in time
- AutoYaST configures Salt to ensure all Master/Worker connect to Salt-Master on the Admin host
- Access Velum web-interface and create admin user
- Specify Internal Dashboard FQDN (CNAME)
- Enable Tiller (for later Helm usage)
- Configure the overlay network
- Add the SSL certificate of the CA signing the registry and external LDAP certificates
- Accept Nodes, Assign Roles
- Specify External API FQDN (load balancer for API and DEX)
- Specify External Velum FQDN (CNAME)
- Run Bootstrap (and now have a cup of coffee ;-))
Deployment

Create required Namespaces
Create required Users / Groups in LDAP or Connect to Active Directory
Create Roles and Role-Assignments

Deploy Basic Services
  • K8s Dashboard
  • Persistent Storage / Storage Classes
  • Ingress
  • Monitoring
  • Logging

Deploy Application
  • Application based scripts
  • CI/CD
  • Helm
Testing
Testing - Preparation

Create a test plan

For every test describe
• Starting point
• Test details
• Expected result

When executing the test
• Prepare and verify starting point
• Execute test
• Document the test execution
• Document the test results
• Compare test results with expectation
• Repeat the test several times
Ensure all fault tolerance tests are done with load on the system

**Network failure**
- Single / Multiple NIC
- Single / Multiple Switches
- Cluster / Public Network

**Node failure**
- Admin
- Master
- Worker
Operations
Life Cycle

- New Patches
- Create new Stage on Staging System
- Assign new Stage to Admin and Nodes
- Wait until next day or “transactional-update dup reboot”
- Access Velum - reboot admin
- Ensure NO Single Pod application runs in the cluster*
- Access Velum - reboot all
Monitoring and Logging

- Old: cAdvisor, Heapster, InfluxDB, Grafana
- New: cAdvisor with Prometheus and Grafana
- Alertmanager
- Logfile collection and cleanup
- Disk space usage
- Application Specific Monitoring?
Don’t do backup and recovery

- Everything that is deployed to the cluster must be 100% reproducible
- Use a second cluster for disaster recovery and deploy the application twice
- Have proper staging for the application
- For persistent data - the application MUST support consistent backup and restore and this cannot be done on the k8s side of things
- Recommendation: use a GIT or similar source code management system
- Disaster Recovery: delete the whole cluster, de-deploy and re-configure the cluster, re-deploy the application and restore the applications data via application functionality
Backup and Recovery (2)

- Backup ETCD
- LDIF export of openLDAP
- Snapshot of Admin VM
- Power off everything and snapshot
- Kubectl export
- GIT / Helm / Yaml File backup and versioning
- Backup of Persistent Volumes
- Single object restore?
- Create an alias for kubectl - -record
Questions?
We adapt. You succeed.
Backup slides
General Requirements ->

- Hardware / Virtualization Infrastructure
  - Where to deploy?
  - On premise or public / private cloud?
- Software
  - CaaS Platform Subscriptions (plus SLES for Infrastructure Service)
- Sales and Pre-/Post-Sales Consulting
  - For architecture and to buy the right hardware
  - For the initial implementation
- Support
  - 24/7 in case of issues
- Maintenance and pro-active support
  - Scale, Upgrade, Review and Fix
BENEFIT SAVINGS

- Where to deploy?
- On premise or public / private cloud?
  - Improve operational efficiency, reduce costs, keep developers focused on development

- Manage container lifecycles
  - Get to market faster, with fewer disruptions

- Foster agile development and business opportunities
  - Eliminate surprises between development and production environments—innovate faster

- Integrate development and operations
Use Case Specific Requirements

• Application Requirements (Sizing)
  – Number of Pods
  – Memory, CPU
  – Storage requirements (file, block, s3?, single or multi-writer, capacity, static or dynamic provisioning)
  – Specific Hardware / CPU / GPU requirements
  – Network Entry points / Services / Bandwidth

• Security Requirements
  – Images (source and size)
  – Isolation
  – Integration into existing Identity Sources
  – Certificate Authorities

• Availability Requirements
  – Single or multi data-center
  – Distance / Latency

• Budget
• Politics, Religion, Philosophy, Processes ;-)
Planning and Sizing

• 1 Admin VM
  – LDAP, Salt, Velum, SQL
• 3 Master VM (more based on number of PODs)
  – Fault tolerance
  – ETCD cluster
• 3 or more Worker (more based on number of PODs and resource requirements)
  – Workers as VM or physical
• Second Cluster for fault tolerance / disaster recovery
• Disk Space for each Worker
  – 50 GB for OS (BTRFS minimum for OS)
  – 100 GB for /var/lib/docker (BTRFS for Images and Containers)
  – (really depends on image sizes and image versions / image changes)
Use Case Specific Requirements

APP. REQUIREMENTS
- To POD5
- CPU & MEMORY
- Storage:
  - Net entity ports, SENS and read/write
  - SPECIFICATIONS (CPU, GPU, ...)

SECURITY
- Usage score & SEC
- Isolation
  - Integrates w/ identity sources
  - CAS

AVAILABILITY
- Single/ Multi X
- Distance & Latency
- CB2

#$BUDGET#

DOLLS, RECIPE, PROCESSES, ...
Deployment Stages

1. Infrastructure Preparation
2. Base Software Installation
3. Infrastructure Verification
4. CaaS Platform Installation
5. Kubernetes Addons
6. K8s Kubelet Settings
7. CaaS Platform Configuration
8. Kubernetes Addons Installation
Deployment - Infrastructure Preparation

- Deploy On-Premise Registry (docker-distribution-registry)
  - Implement Portus to Secure the On-Premise Registry
  - Create DNS entry for Registry
  - Create Namespaces and Users on Registry
  - Optional: Integrate Portus into existing LDAP or Active-Directory
- Put all required images into registry into the right namespace
  - Dashboard, Prometheus, Grafana, etc.
- Optional: Setup caching registries
- Prepare Load Balancer Enpoints for API and DEX
  - Port 6443 and 32000
- Storage Network setup and connectivity
Deployment - Infrastructure Preparation

• Prepare on-premise helm chart repository
• Prepare docker host to pull from internet, scan images, push to on-premise registry
• Prepare GIT for storing all yaml files

• ToDo: Monitoring/Logging/Backup???
Deployment - Software Installation

- Software Staging
  - Subscription Management Toolkit, SUSE Manager, RMT (limited)
  - Ensure staging of patches to guarantee same patch level on existing servers and newly installed servers
- General
  - Use BTRFS for the OS
  - Disable Firewall / AppArmor / IPv6
- AutoYaST
  - Ensure that all servers are installed 100% identical
  - Consulting solution available (see https://github.com/Martin-Weiss/cif)
- Configuration Management
  - Templates
  - Salt
Deployment – Infrastructure Verification

• ONLY USE STATIC IP Configs
• Verify Time Synchronization
• Verify Name Resolution
• Verify repository sources are ok (staging)
• Test all Network Connections
  – Bandwidth, Latency
Deployment - Step by Step

- Install all Servers (Admin, Master, Worker) via AutoYaST
- Ensure that all the patches available are installed at this point in time
- AutoYaST configures Salt to ensure all Master/Worker connect to Salt-Master on the Admin host
- Access Velum web-interface and create admin user
- Specify Internal Dashboard FQDN (CNAME)
- Enable Tiller (for later Helm usage)
- Configure the overlay network
- Add the SSL certificate of the CA signing the registry and external LDAP certificates
- Accept Nodes, Assign Roles
- Specify External API FQDN (load balancer for API and DEX)
- Specify External Velum FQDN (cname)
- Run Bootstrap (and now hope and pray ;-) )