SAP on **IBM Power Hybrid Cloud**
IBM Research & Development Lab

White Paper

SAP S/4HANA on Red Hat OpenShift Container Platform: Business Perspective from a Cloud Hosting Provider

TUM: Johannes Rank, Simon Fuchs
IBM: Sabine Jaeschke, Dr. Jochen Röhrig, Achim Haaser

IBM Deutschland Research & Development GmbH
Created in December 2022 – Version 1.0

IBM

Acknowledgement

We would like to express our special thanks and gratitude to Dr. Holger Wittges, Managing Director of the UCC Munich; as well as Mr. Thomas Heinlein, Manager SAP on IBM Power Hybrid Cloud at IBM Germany; and Mr. Alexander Loibl, TUM Liaison Manager, SAP on IBM Power Hybrid Cloud – who all greatly benefitted this research by their constant support. Without them, this project would not have been possible at all. Thank you for lots of organization, arrangements, backing and enabling.
Secondly, we want to thank IBM Germany and the Technical University of Munich for enabling and sustaining this research.

Edition Notice and Version Information

IBM Corporation
New Orchard Road
Armonk, NY 10504

Focus: SAP S/4HANA® deployment in Red Hat® OpenShift® containers

Document Version	Changes
01	Initial version covering the containerization of S/4HANA® systems at the SAP® University Competence Center (UCC) of the Technical University Munich, December 2022

Preface

Red Hat® OpenShift® is a powerful platform that allows the orchestration and management of lightweight and resource-efficient containers in a highly available fashion. This document describes the experiences of applying this concept to the on premises hosted SAP® systems at the *SAP® University Competence Center (UCC)* of the *Technical University of Munich (TUM*).

IBM's containerization-for-sap-s4hana tools are used to containerize the UCC's S/4HANA® 2020 systems in an automated fashion. This allows the provisioning of multiple identical copies of an SAP S/4HANA® reference SAP system in an easy way. These containerized SAP systems are highly resource-efficient, quick to deploy and offer high reliability. This document describes the implementation and the management of containerized SAP systems and evaluates the applicability to the UCC's business scenario. This solution is NOT supported by SAP and only usable in non-production scenarios. However, it describes an efficient and elegant solution for the use case from UCC, where production support from SAP is not required.

About this Document

This document is intended for architects and specialists planning to deploy SAP S/4HANA® systems in Red Hat® OpenShift® on IBM Power® in non-productive environments. It describes the design considerations for hardware, networking, and software components of the Red Hat® containerization on IBM Power solution stack.
This guide does not replace existing SAP or Red Hat documentation and sizing guides. It serves as a supplement to them.

Table of Contents

Acknowledgement 2
Edition Notice and Version Information 3
Preface 3
About this Document 3
Table of Contents 4
Figures 6
Tables 6
Disclaimer and Special Notices 7
Copyright License 8
Introduction 9
UCC Business Model 10
Business Requirements and Expectations 11
Hosting Requirements 12
Technical Introduction to OpenShift 12
Kubernetes and OpenShift 13
Terminology 13
Solution Architecture 14
Setting up the OpenShift Cluster on IBM Power 16
Preparing the OpenShift Cluster Installation 16
Sizing the Worker Nodes 16
Setting up the Cluster Node LPARs 16
Installing Additional Packages 17
Configuring SELinux on the Helper Node 17
Downloading the Red Hat OpenShift Pull Secret 18
Exchanging the SSH Key with HMC 19
Cloning the GitHub Repository 19
Adapting the Ansible Configuration Files 19
Running the OpenShift Cluster Installation 20
Monitoring the Installation Progress 20
Troubleshooting 21
Postprocessing 23
Accessing the OpenShift Web Console 23
Disabling SELinux on all Worker Nodes 24
Changing the Runtime Limits 25
Creating Users in OpenShift 25
Changing the Default Route of Image Registry 26
Building and Deploying Process 27
Preparing the Building and Deploying Process 27

Exchanging SSH Keys 27
Setting up the Build Directories 27
Installing the Software Requirements 27
Cloning the GitHub Repository 27
Copying the Compatibility Library for SLES 28
Creating an OpenShift Project 28
Creating the Credentials and Configuration Files 29
Stopping the Reference SAP System 29
Building Process 30
Creating a Snapshot Copy of the Reference HANA Database 30
Building the Container Images 30
Pushing the Images to the Cluster Registry 30
Deploying Process 31
Creating the Overlay Filesystem 31
Generating the Deployment Description File 31
Starting the Deployment 31
Full Automated Building and Deploying 31
Monitoring the Deployment 32
Displaying the State of a Deployment 32
Checking the Status of the Containerized SAP System 32
Checking SAP System Logfiles 33
Logging into a Container 33
Building and Deploying with Ansible 34
Preparing the Ansible Configuration File 34
Ansible Restrictions 35
Providing Access with HAProxy 35
Initial Port Forwarding 35
Changing Port Forwarding during Runtime 36
Managing Deployments 38
Adding Deployments 38
Conclusion and Next Steps 39
Business Requirements 39
Fast Deployment 39
Computing Resources Efficiency 39
High Degree of Automation 39
Fast and Easy Life-Cycle Management 39
Hosting Requirements 39
Strictly Separated Containerized SAP Systems 39
Access via SAP GUI, Fiori Launchpad and SAP HANA Studio 40

Shiftable Workload 40
HANA Databases Connectivity to the UCC's TSM Backup Nodes 40
Easily Stoppable and Restartable SAP systems 40
SAP Solution Manager Integration 40
SAP Cloud Connector and SAP Analytics Cloud Agent 40
Integration into the UCC's Transport Domain 40
Future Work 41
Summary Assessment of the Present Status 41

Figures

Figure 1: The UCC Munich's Business Scenario 10
Figure 2: Solution Architecture 14
Figure 3: Downloading the Pull Secret 18
Figure 4: Output of: *oc get nodes* 20
Figure 5: Output of: *oc get co* 21
Figure 6: OpenShift Frontend and Navigation 23

Tables

Table 1: Configuration of the Cluster Node LPARs at UCC 16
Table 2: Compatibility Libraries related to SAP HANA SP Levels 28
Table 3: Ansible RestrictionsManaging SAP Workloads 35

Disclaimer and Special Notices

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes are incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.
Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those

Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.
Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements are the same on generally available systems.
Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.
Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.
This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products.
All these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Copyright License

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

ANY INFORMATION HEREIN IS PROVIDED “AS IS” WITHOUT WARRANTY OR INDEMNIFICATION OF ANY KIND BY IBM AND DO NOT ANY EXPRESS OR IMPLIED, WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS OR USAGE FOR PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT.

Introduction

The IBM Team of the IBM Research & Development Lab in Böblingen Germany is enabling, verifying, and developing SAP solutions on IBM Power and End2End Automation for the Hybrid Cloud including on premises integration. The automation of entire SAP deployments in the IBM Cloud includes the infrastructure/network over the OS levels to the unattended installation of SAP S/4HANA® systems on IBM Power systems. More info and live demos can be found on IBM Cloud as "free to use" service.
Target is to leverage the IBM Power unique features for enterprise SAP workloads and combine the Red Hat OpenShift Container Platform capabilities to provide ultra-fast SAP deployments in containers with all the advantages like high availability, resiliency, and sustainability of the IBM® Power10 processor-based Power servers in the Cloud.
Renewing the IBM & TUM partnership with the next IBM Power generation intensified our joint-development approach to enable SAP S/4HANA in containers for education purposes.
If you would like to know more about the IBM Team, the solution or if you face any questions during your work with that document, don't hesitate to contact: Thomas Heinlein from the IBM R&D Lab in Böblingen, Germany (thomas_heinlein@de.ibm.com)

The research and development for this document is conducted on a real-world business case at the SAP University Competence Center (UCC) of the Technical University of Munich (TUM) (short: UCC Munich). The UCC Munich is an SAP-certified Education-as-a-Service (EaaS) provider for SAP products (i.e., SAP systems & platforms) for teaching, research, and co-innovation to their academic and educational partners like universities, colleges, and schools. It works as an autonomous cluster at the chair *I17 - Information Systems and Business Process Management* at the TUM.
These educational services include software and platform services, like hosting of SAP on premise systems like SAP S/4HANA or SAP BW/4HANA or hosting the ERP learning simulation game *ERPsim*. Additionally, they include further educational services, such as developing, polishing and distribution of ready-made curricula for SAP software and technologies.

UCC Business Model

The added value of containerization of SAP solutions based on Red Hat OpenShift for the UCC is strongly interconnected with the business model of the UCC Munich, which is visualized in *Figure 1: The UCC Munich's Business Scenario*. In detail, any educational institution (university or school) can become a partner of the UCC by signing a partnership contract. With this, the UCC grants access to one of its hosted SAP training systems, grants access to its SAP teaching curricula and provides technical and educational support for these systems and curricula. Teachers and professors working at the institution and the institution's students can then access the SAP training system(s) provided to the institution for training, teaching and lectures. Furthermore, the teachers and professors can use the SAP teaching curricula to teach relevant concepts of information systems and economic sciences as well as hands-on SAP knowledge in their lectures, seminars, and lessons. In summary, the UCC serves as an *Education-as-a-Service* provider for SAP products.

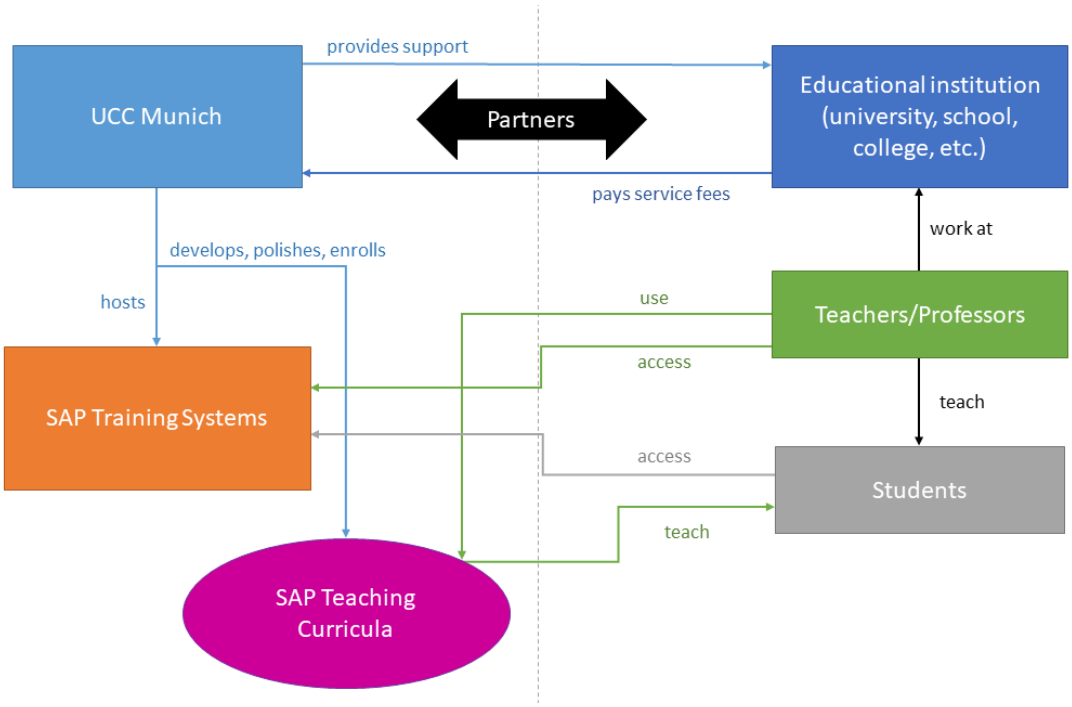


Figure 1: The UCC Munich's Business Scenario

Business Requirements and Expectations

The UCC's system landscape consists of both long running systems and short-lived training and demo systems. While the long running systems are used for long-term teaching in universities, the short-lived systems are used for demo access to potential new customers or train-the-trainer sessions given to teachers and professors at the partner institutions. Currently, setting up a new SAP system can take up to 5 working days, as SAP systems need many configuration steps after Logical Partition (LPAR) setup and installation. These post-installation configurations comprise SGEN-generation of SAP objects, password homogenization, integrating the new system into the SAP landscape for monitoring purposes, port-unblocking, integration into the transport domain, etc. Some of these steps are unneeded in case of short-lived systems, but most are indispensable. Especially, in case of such short-lived systems, an easy way to deploy and dismantle them would come in very handy and save working time that could be allocated to other tasks. Beneficially to this research, most of these short-lived systems are identical to each other.
Additionally, these almost identical systems in the UCC computing center produce a huge data duplication on storage level. Each standardized SAP training LPAR consumes about 350 GB storage volume for operating system, HANA database, SAP primary application server, and the training data set of the UCC's fictional company "*Global Bike Inc*.". The containerization solution described in this document promises to drastically reduce this data redundancy, as all containers use the same read-only HANA image as starting point for their operations.

We therefore formulate four business requirements to the prototype developed in this project:

1. A containerized SAP system should be fast deployable.
2. The containerized systems should be more efficient regarding computing resources than setting up similar systems on classical LPARs.
3. The deployment of new systems should be highly automated, to save time and manual effort of the UCC's technical staff.
4. The life-cycle management, especially deprovisioning and deletion, of a containerized system should be easy and fast.

Hosting Requirements

For the business purpose described in the previous section, the UCC Munich hosts more than 200 SAP systems, each installed on separate LPARs which run on four IBM Power8® and two IBM® Power9® servers. An SAP® Solution Manager 7.20 manages and monitors the complete SAP landscape. For the use of an OpenShift container system at the UCC, the container system must meet some external requirements that are explicated in the following:

1. It is essential that all systems are strictly separated from each other.
2. The systems must be accessible by SAP® GUI, SAP® Fiori launchpad and SAP® HANA Studio.
3. Workloads must be easily shiftable between servers.
4. Each deployed HANA database requires connection to the UCC's TSM backup nodes at the Leibniz Supercomputing Centre (Leibniz-Rechenzentrum, LRZ).
5. The deployed SAP systems must be easily stoppable and restartable.
6. The solution should support SAP® Solution Manager integration of the deployed SAP systems.
7. The containerized SAP systems should support connection to the SAP® Cloud Connector and installation and operability of the SAP® Analytics Cloud Agent.
8. It should be possible that all containerized SAP systems can be integrated into the UCC's transport domain.

Keep in mind that this solution is NOT supported by SAP for production environments.

Technical Introduction to Red Hat OpenShift

OpenShift is one of the most popular distributions of Kubernetes among over 100 competitors. Kubernetes is an open-source multiple-vendor container management platform. Containers describe executable units of software that contain the application code as well as its dependencies and libraries. In contrast to virtual machines, however, containers do not include the full operating system and are therefore considered being much more lightweight. The de facto standard platform for orchestration, management and provisioning of container-based software is Kubernetes, especially in the cloud native computing applications context. Some of the key features of Kubernetes include built-in support for autoscaling, automatic rollouts and rollbacks of updates, health checking, self-healing and a declarative deployment model. A deeper insight into the features of Kubernetes can be found in the official Kubernetes documentation at https://kubernetes.io/docs/home/.

Kubernetes and OpenShift

One of the most important contributors to Kubernetes is Red Hat. It has been part of the Kubernetes ecosystem since the beginning of the open-source project and is now the second largest contributor. With OpenShift, Red Hat provides the most widespread distribution of Kubernetes. This is a 100% conformant Kubernetes platform with further enhancements that are intended to optimize the productivity of developers (image creation, image pushing to registries, and a base image catalog) and IT operations: OpenShift 4 enables reliable, repeatable, and automated full-stack installations of entire infrastructures, which can also include virtual machines or domain name service components. Furthermore, the tight integration of OpenShift with the RHEL CoreOS operating system allows cluster management operations to perform updates of the operating system. OpenShift also allows automatic adjustment of cluster size and version. More insights about OpenShift can be found in the official documentation at https://docs.OpenShift.com/.

Terminology

This section covers a few terms that are essential to the OpenShift architecture:

A **pod** is the smallest compute unit that can be deployed, defined and managed in OpenShift. It always receives its own internal IP address and serves as an execution context for possibly multiple containers. These containers share storage and network resources.

The containers of a pod are instantiated based on an **image**. It comprises all the artefacts required to run an application in a container (binaries, metadata, ...).

Images are stored in the **image registry**. The registry serves as a *content server* to provide images upon request.

Pods are executed on **nodes**. A node is a real or virtual machine which hosts an operating system with all required Kubernetes components.

The **control plane nodes** are responsible for managing the cluster. They host several services such as the API-server, which enables external access to the cluster and the scheduler, which is responsible to distribute the workload on the available **worker nodes**.

Solution Architecture

This chapter describes UCC's solution architecture.

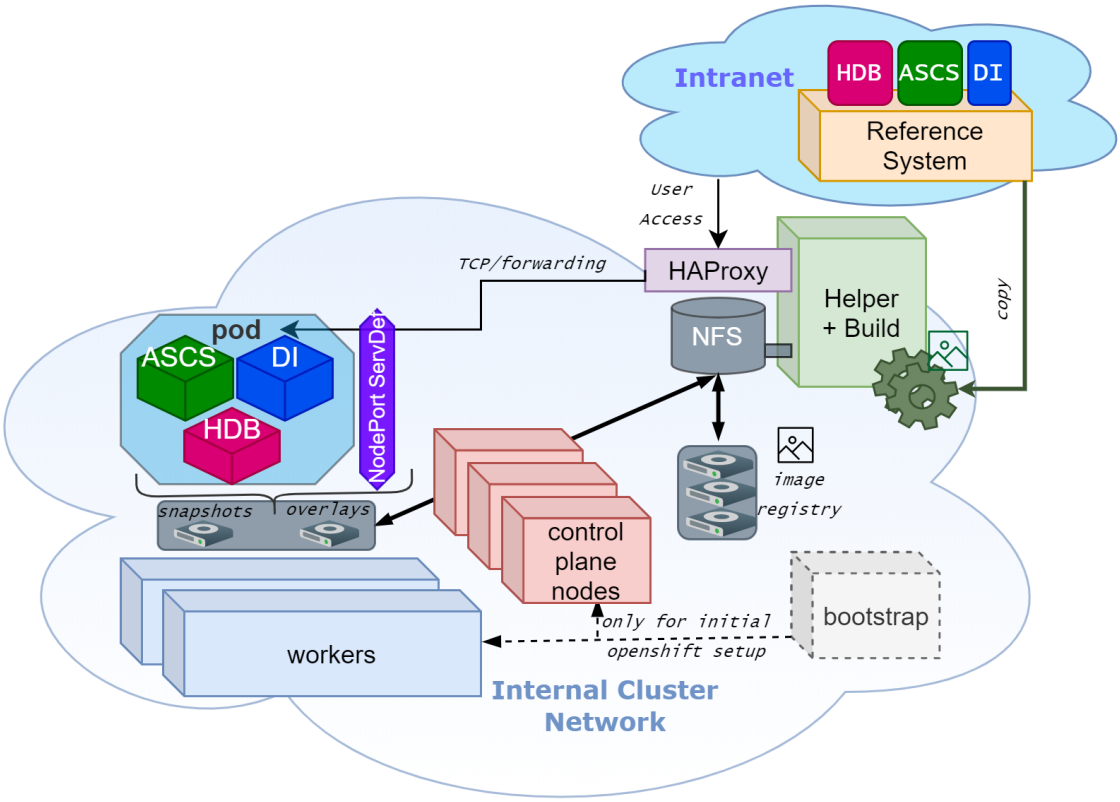


Figure 2: Solution Architecture

As depicted in *Figure 2: Solution Architecture* the UCC OpenShift® cluster is comprised of seven nodes (two worker nodes, three control plane nodes, one helper node and one bootstrap node), where each node runs on an LPAR. All LPARs are set up on one single IBM® Power® E950 server.

Reference SAP System

The reference SAP system is not part of the OpenShift cluster and resides in its own network. Only the build machine requires access to the reference SAP system. In our scenario the reference SAP system is an SAP central system in which all instances (DI: Dialog instance, ASCS: ABAP central services instance, HDB: HANA Database server instance) run on one host.

In addition, SAP distributed systems (instances run on different hosts) are supported as reference SAP system. In this case refer to the project website for further information.

Worker Nodes

The two worker nodes are configured to run the actual SAP workload.

Control Plane Nodes

A full-fledged OpenShift cluster requires at least three control plane nodes for high availability reasons.

Bootstrap Node
The bootstrap node is only required during the installation of the OpenShift cluster. When the cluster is up and running it can be shut down.

Helper Node
The helper node is the central management machine from which the cluster installation is started. It provides the cluster with a central NFS server. In addition, it hosts a *HAProxy*, which allows external access to the containerized SAP systems from outside the cluster network.

Build LPAR
The images which are required to deploy a copy of the reference SAP system are built on the build LPAR. As the build process copies all relevant files from the reference SAP system host to a local directory, the build LPAR requires a considerable amount of disk space. It also needs network access to the application server on which the reference SAP system resides. In our case the helper node serves as the build machine, however the build machine can run on a separate host.

Images and Pods
The build process creates three different images: one for the initialization of the deployment (*init*), one for the dialog instance and the ABAP central services instance (*nws4*) and one for the HANA Database instance (*hdb*). While the *init* image exists only during the initialization, the *nws4* image is used for creating both the DI and the ASCS container. The *hdb* image results in the HDB container in which the HANA database instance is running.
All three containers (HDB, ASCS, and DI) run in one pod.

Snapshots and Overlay Filesystems
Before building the images, a snapshot of the HANA database content is created on the NFS server. The HANA database content of a containerized SAP system exists of two layers:

- The snapshot which serves as the read-only base layer
- An individual read-write layer on which the changes to the database are persisted

Both layers form the overlay filesystem of one containerized SAP system.

Due to this mechanism every containerized copy of a reference SAP system saves only the delta to the original HANA database, which is contained in the snapshot. This reduces data duplication significantly.

Setting up the Red Hat OpenShift Cluster on IBM Power

This chapter describes how to set up the Red Hat OpenShift cluster based on IBM Power. For a full and extensive description of all steps please refer to the project website.

Preparing the **OpenShift** Cluster Installation

Sizing the Worker Nodes

The worker nodes, which run the actual SAP workload, require sufficient sizing, especially with regards to memory. If a worker node is not able to fulfill the memory requirements the deployment is not started completely and OpenShift waits until enough resources are available. The memory requirement of one deployment can be calculated as follows:

$$30GB\ (DI) + 10GB\ (ASCS) + < disk\ usage > GB\ (HANA)$$

The calculated memory requirement must be multiplicated with the intended number of containerized SAP systems per worker node. At UCC, about 190GB RAM per SAP system are assigned, therefore each worker node has 600GB RAM assigned. This amount is sufficient for running three deployments in parallel on one worker node.

In addition, every worker node has 2.0 processing units and 8 virtual CPUs.

Note that although IBM Power supports dynamic logical partitioning (DLPAR) to change resources dynamically during runtime, all workers run on RHCOS (Red Hat® CoreOS), for which dynamic partitioning might not work. Therefore, it is recommended to choose a suitable size in advance.

Setting up the Cluster Node LPARs

The configuration of the cluster node LPARs is shown in the following table:

LPAR	Operating System	Proc/vCPU	Memory	Storage
Bootstrap Node	Initial: RHEL9 Later: RHCOS (CoreOS)	0.3/2	32 GB	120 GB
Helper (Build) Node	Initial: RHEL9	0.5/4	64 GB	120 GB (/) 880 GB NFS (/export) 500 GB build (/data)
3 Control Planes	Initial: RHEL9 Later: RHCOS (CoreOS)	0.3/2	32 GB	120 GB
2 Worker Nodes	Initial: RHEL9 Later: RHCOS (CoreOS)	2.0/8	600 GB	600 GB

Table 1: Configuration of the Cluster Node LPARs at UCC

Recommendation: For performance reasons all six LPARs should be placed on a single IBM Power server. This ensures all network traffic is handled by the virtual switch, which significantly improves the performance of the NFS.

The following command can be used to create an LPAR via HMC:

```
mksyscfg -r lpar -m <managed_system> -i name=<lpar_name>,
profile_name=default_profile, lpar_env=aixlinux,
shared_proc_pool_util_auth=1, min_mem=<min_mem>, desired_mem=<des_mem>,
max_mem=65536, proc_mode=shared, min_proc_units=0.2,
desired_proc_units=0.4, max_proc_units=4.0, min_procs=1,
desired_procs=4, max_procs=16, sharing_mode=uncap, uncap_weight=128,
max_virtual_slots=64, boot_mode=norm, conn_monitoring=1
```

On each LPAR a Red Hat® Linux Server 9 installation is performed (see also here).
RHEL9 can be downloaded from the Red Hat Customer Portal.

Installing Additional Packages

Some additional packages must be installed from the *Extra Packages for Enterprise Linux (EPEL)* repository.
Enabling the *Extra Packages for Enterprise Linux (EPEL)* repository*:*

```
yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-
$(rpm -E %rhel).noarch.rpm
```

Installing the packages:

```
yum -y install ansible git firefox xorg-x11-xauth dbus-x11 grub2-tools-
extra
```

Configuring SELinux on the Helper Node

The SELinux configuration settings can be found in **/etc/selinux/config**.
Both parameters must be set as follows:

```
SELINUX=permissive
SELINUXTYPE=targeted
```

Caution: setting additional parameters may cause problems during the startup process of the OpenShift® cluster.

The SELinux configuration file should look as follows:

```
# NOTE: In earlier Fedora kernel builds, SELINUX=disabled would also
# fully disable SELinux during boot. If you need a system with SELinux
# fully disabled instead of SELinux running with no policy loaded, you
# need to pass selinux=0 to the kernel command line. You can use grubby
# to persistently set the bootloader to boot with selinux=0:
SELINUX=permissive
# SELINUXTYPE= can take one of these three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes
are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

The following command activates the SELinux configuration settings:

```
setenforce Permissive
```

In addition, the parameter *selinux* must be set to 1 in file **/etc/default/grub**

```
selinux=1
```

After updating the boot loader and rebooting:

```
grub2-mkconfig
reboot
```

the output of the command:

```
getenforce
```

should now show *Permissive.*

Downloading the Red Hat OpenShift Pull Secret

The installation process requires a pull secret for authentication when downloading images from the Red Hat image registry.

The pull secret is downloaded from Red Hat Hybrid Cloud Console as shown in *Figure 3*: Downloading the Pull Secret.

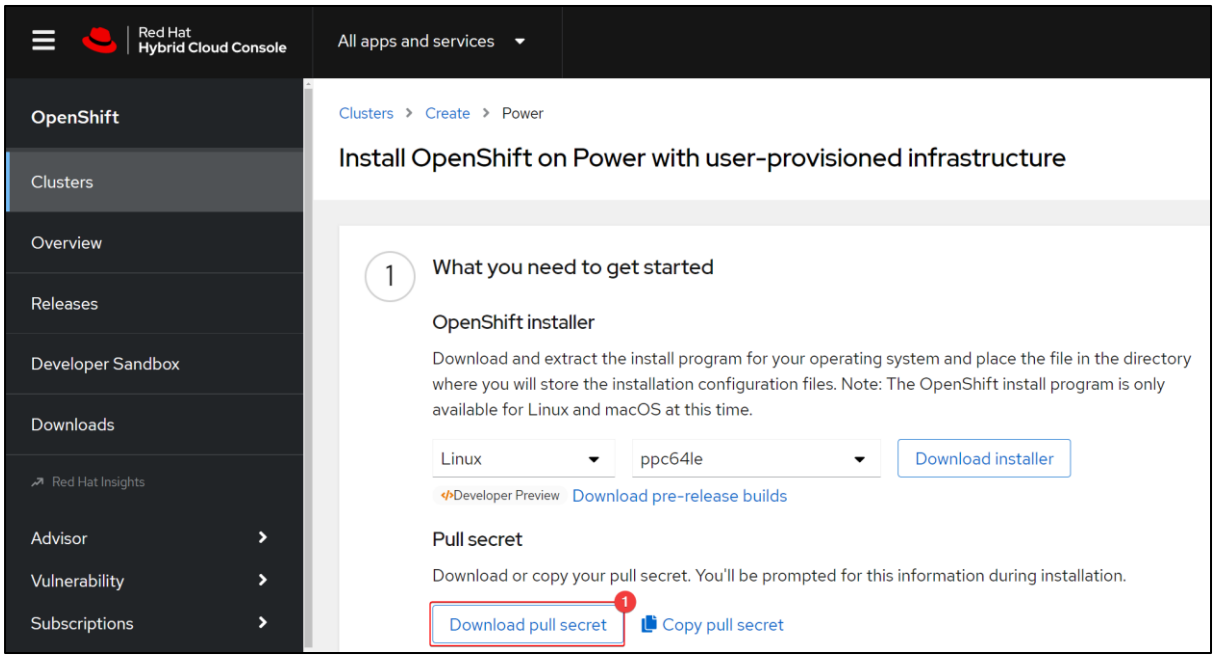


Figure 3: Downloading the Pull Secret

The file must be stored on the helper node as **/root/.OpenShift/pull-secret** without any file extension.

Exchanging the SSH Key with HMC

The automated OpenShift installation requires passwordless access from the helper node to the HMC, e.g., to boot the nodes during the RHCOS installation.

Passwordless authentication requires the creation of an SSH key pair on the helper node:

```
ssh-keygen -t rsa -b 4096 -N '' -C "<user@sample.com>"
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_rsa
ls -1 ~/.ssh/id_rsa
```

On the HMC the public key is added with the command:

```
mkauthkeys -a "ssh-rsa <content of /root/.ssh/id_rsa.pub>
<user@sample.com>"
```

The passwordless access can be checked on the helper node via the following command:

```
ssh hmc_user@hmc_hostname lshwres -m <managed_system> -r virtualio --
rsubtype eth --level lpar -F lpar_name,mac_addr
```

Cloning the GitHub Repository

The OpenShift Cluster is installed with Ansible. The Ansible playbooks are provided by the GitHub repository ocp-upi-powervm-hmc. In the UCC scenario this repository version is used. The repository must be cloned on the helper node:

```
git clone https://github.com/ocp-power-automation/ocp4-upi-powervm-
hmc.git
cd ocp4-upi-powervm-hmc/
git checkout 6cfc78952b3e0114e1ca64066f59f09380e8e676
```

This repository includes external submodules from other repositories. They must be included using:

```
git submodule update --init --recursive –remote
```

Possible errors may be fixed by running:

```
git pull --recurse-submodules
```

Adapting the Ansible Configuration Files

The Ansible playbooks require two files called **inventory** and **vars-powervm.yaml** located in the repository root directory **ocp4-upi-powervm-hmc.** Templates of the files can be copied from the **examples** directory.

The **inventory** file must be adapted as follows:

```
[bastion]
localhost ansible_connection=local
```

All parameters of the **vars-powervm.yaml** must be adapted to the target environment and the option
storage_type: nfs must be set.

```
pvm_hmc: <hmc-user>@<ip-power-server>

…
ocp_release: 4.11
ocp_rhcos_tag: "4.11.2"
ocp_client_tag: "4.11.6"
…

storage_type: nfs
```

The MAC addresses on the HMC can be obtained via the following script:

```
$ for i in <managed_systems>
do
  lshwres -m $i -r virtualio --rsubtype eth --level lpar -F
lpar_name,mac_addr
done
```

Running the OpenShift Cluster Installation

The OpenShift cluster is installed using the following Ansible playbook:

```
ansible-playbook -e @vars-powervm.yaml playbooks/main.yaml -v
```

The installation process is a long-running process and can last up to one hour, it contains of preparation, LPAR booting, and configuration step.

Monitoring the Installation Progress

The installation progress can be monitored via the following URL: http://<helper-ip>:9000

Cluster node monitoring:

```
oc get nodes
```

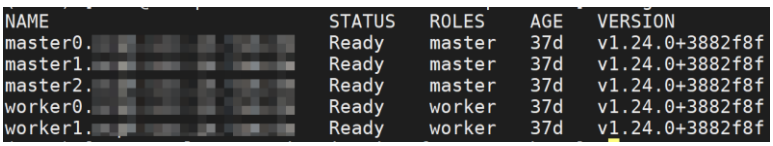
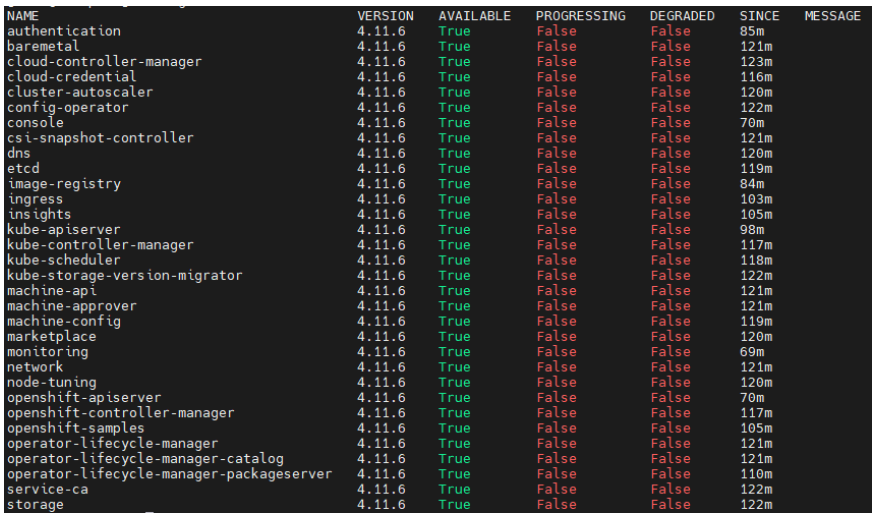


Figure 4: Output of: *oc get nodes*

Also check the status of the cluster operators.

```
oc get co
```

After the installation completed successfully you should get the following output:



NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE	MESSAGE
authentication	4.11.6	True	False	False	85m	
baremetal	4.11.6	True	False	False	121m	
cloud-controller-manager	4.11.6	True	False	False	123m	
cloud-credential	4.11.6	True	False	False	116m	
cluster-autoscaler	4.11.6	True	False	False	120m	
config-operator	4.11.6	True	False	False	122m	
console	4.11.6	True	False	False	70m	
csi-snapshot-controller	4.11.6	True	False	False	121m	
dns	4.11.6	True	False	False	120m	
etcd	4.11.6	True	False	False	119m	
image-registry	4.11.6	True	False	False	84m	
ingress	4.11.6	True	False	False	103m	
insights	4.11.6	True	False	False	105m	
kube-apiserver	4.11.6	True	False	False	98m	
kube-controller-manager	4.11.6	True	False	False	117m	
kube-scheduler	4.11.6	True	False	False	118m	
kube-storage-version-migrator	4.11.6	True	False	False	122m	
machine-api	4.11.6	True	False	False	121m	
machine-approver	4.11.6	True	False	False	121m	
machine-config	4.11.6	True	False	False	119m	
marketplace	4.11.6	True	False	False	120m	
monitoring	4.11.6	True	False	False	69m	
network	4.11.6	True	False	False	121m	
node-tuning	4.11.6	True	False	False	120m	
openshift-apiserver	4.11.6	True	False	False	70m	
openshift-controller-manager	4.11.6	True	False	False	117m	
openshift-samples	4.11.6	True	False	False	105m	
operator-lifecycle-manager	4.11.6	True	False	False	121m	
operator-lifecycle-manager-catalog	4.11.6	True	False	False	121m	
operator-lifecycle-manager-packageserver	4.11.6	True	False	False	110m	
service-ca	4.11.6	True	False	False	122m	
storage	4.11.6	True	False	False	122m	

Figure 5: Output of: *oc get co*

Troubleshooting

Issue1 - "Please revise your vars.yaml file. Invalid characters found in hostnames"

When receiving the message "*Please revise your vars.yaml file. Invalid characters found in hostnames*" it may be the terms master[0-2] and worker[0-1] are used as actual hostnames. This error can be resolved by deleting the lines containing „*{{ masters }}*" and *"{{ workers | default('') }}*" from the file **playbooks/ocp4-helpernode/tasks/validate_host_names.yaml**

Original:

```
- name: Validate values for DNS compatibility
  fail:
    msg: "Please revise your vars.yaml file. Invalid characters found
in hostnames"
  when: item is search('{{ chars }}')
  with_items:
  - "{{ dns.domain }}"
  - "{{ helper.name }}"
  - "{{ bootstrap.name | default('') }}"
  - "{{ masters }}"
  - "{{ workers | default('') }}"
```

Modified:

```
- name: Validate values for DNS compatibility
  fail:
    msg: "Please revise your vars.yaml file. Invalid characters found
in hostnames"
  when: item is search('{{ chars }}')
  with_items:
  - "{{ dns.domain }}"
  - "{{ helper.name }}"
  - "{{ bootstrap.name | default('') }}"
```

Issue2 – Missing Registration

When receiving the message "*No package bind available"* the helper node may have no valid license. This issue can be fixed by registering the operating system in order to enable the software download.

Issue3 – Network Mask

The installation assumes that the network mask is /24. However, in UCC's case it is /23 which leads to issues during the network boot. For this reason, the *lpar_netboot* command in the file *main.yaml* in directory **playbooks/roles/bootup-nodes/tasks/** has to be changed. The network mask can be changed via the -K option.
Example:

Original:

```
remote_cmd="lpar_netboot -v -i -D -f -t ent -m ${pvm_mac} -s auto -
d auto -S ${server} -C ${ipaddr} -G ${gateway} ${pvmlpar}
${pvm_profile} ${pvmcec}"
```

Modified:

```
remote_cmd="lpar_netboot -v -i -D -f -t ent -m ${pvm_mac} -K
255.255.254.0 -s auto -d auto -S ${server} -C ${ipaddr} -G ${gateway}
${pvmlpar} ${pvm_profile} ${pvmcec}"
```

Postprocessing

After completion of the installation a few postprocessing steps have to be performed.

Accessing the OpenShift Web Console

Access to the web console from an external workstation requires DNS resolution of the cluster domain. This is achieved by adding the following line to the **hosts** file of the workstation operating system.

```
<helper-ip> api.ocp4.<fqdn> oauth-OpenShift.apps.ocp4.<fqdn> console-OpenShift-
console.apps.ocp4.<fqdn>
```

The path of the **hosts** file is as follows:

- Linux: **/etc/hosts**
- Windows: **C:\Windows\System32\drivers\etc\hosts**

Now the web console can be accessed as user *kubeadmin* via the URL:
https://console-openshift-console.apps.ocp4.<FQDN-HELPERNODE>
The password can be found in the file **/root/ocp4-pvm/auth/kubeadmin-password** on the helper node.

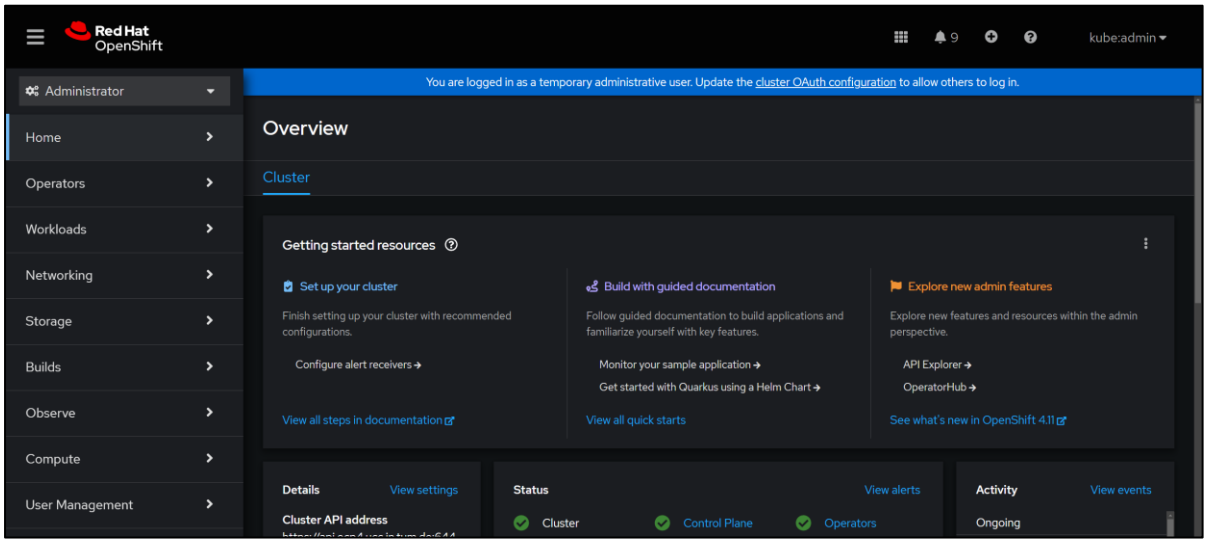


Figure 6: OpenShift Frontend and Navigation

Disabling SELinux on all Worker Nodes

Disabling SELinux is performed on the helper node and achieved by means of a file **selinux-disable.yaml** with the following content:

```
apiVersion: machineconfiguration.OpenShift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.OpenShift.io/role: worker
  name: 05-worker-kernelarg-selinuxoff
spec:
  config:
    ignition:
      version: 2.2.0
  kernelArguments:
    - selinux=0
```

After executing the command:

```
oc create -f selinux-disable.yaml
```

the worker nodes should restart automatically and adopt the settings.

The status of all nodes can be checked with the following command:

```
oc get nodes
```

Whether the settings have been applied successfully can be checked by connecting to the worker node via SSH as user **core**:

```
ssh core@<worker> "getenforce"
```

Changing the Runtime Limits

Adjusting the runtime limits of all worker nodes is performed on the helper node and achieved by means of a file **set-pids.yaml** with the following content:

```
{
      "apiVersion": "machineconfiguration.OpenShift.io/v1",
      "kind": "ContainerRuntimeConfig",
      "metadata": {
            "name": "set-sap-config"
      },
      "spec": {
            "machineConfigPoolSelector": {
                  "matchLabels": {
                        "custom-crio": "set-sap-config"
                  }
            },
            "containerRuntimeConfig": {
                  "pidsLimit": 8192,
                  "overlaySize": "30G"
            }
      }
}
```

After executing the following commands:

```
oc label machineconfigpool worker custom-crio=set-sap-config
oc create -f set-pids.yaml
```

the changes can be checked with:

```
ssh core@workerAddress "crio config 2>/dev/null | grep 'pids_limit'"
```

Creating Users in OpenShift

By default, there exists only the OpenShift administration user *kubeadmin*.
To access the OpenShift cluster during the build and deployment process described in section Building and Deploying Process it is recommended to specify a user other than *kubeadmin*.

Additional users can be created by executing the following commands:

```
htpasswd -c -B -b user.htpasswd <ocp-user-name> <user password>
oc create secret generic htpass-secret --from-file=htpasswd=user.htpasswd
-n OpenShift-config
```

Deploying the user to all worker nodes is performed on the helper node and achieved by means of a file **ht-provider.yaml** with the following content:

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
spec:
  identityProviders:
  - name: htpasswd_provider
    mappingMethod: claim
    type: HTPasswd
    htpasswd:
      fileData:
        name: htpass-secret
```

After executing the following command:

```
oc apply -f ht-provider.yaml
```

the login can be tested by issuing:

```
oc login -u <ocp-user-name>
oc whoami
```

Changing the Default Route of Image Registry

Finally, the default route for the image registry needs to be changed by issuing the following commands on the helper node:

```
oc login -u kubeadmin
oc patch configs.imageregistry.operator.openshift.io/cluster --type merge -p
'{"spec":{"defaultRoute":true}}'
oc get route -n OpenShift-image-registry
```

Building and Deploying Process

This chapter covers all steps required to build all images based on a reference SAP system and to deploy them into the OpenShift cluster. A detailed description of all steps can be found at the project website. If not mentioned otherwise, all steps need to be performed on the build machine. In the UCC the helper node is used as build machine.

Preparing the Building and Deploying Process

Initially, some preparation steps must be performed on the build machine.

Exchanging SSH Keys

During the image build, all SAP-relevant files are copied from the reference SAP system to a local directory on the build machine. Therefore, the build user (in the UCC case *root*), needs passwordless SSH access to the *<SAPsid>adm* user accounts on the reference SAP system. Therefore, the content of **/root/.ssh/id_rsa.pub** must be added to the **authorized_keys** file of both the *<nws4-sid>adm* and the *<hdb-sid>adm* user accounts.

Setting up the Build Directories

By default, all images are stored under **/var/lib/containers**. This directory grows significantly during the build process. Therefore, a dedicated partition (**/data**) with about 500GB is used:

```
mkdir -p /data/var/lib/
mv /var/lib/containers /data/var/lib/containers
ln -s /data/var/lib/containers /var/lib/containers
```

Furthermore, the build process copies all relevant files from the reference SAP system to a local directory, which is **/var/tmp** by default. This directory grows significantly as well.

```
mkdir -p /data/var/
mv /var/tmp /data/var/tmp
ln -s /data/var/tmp /var/tmp
```

Installing the Software Requirements

On the build machine *git, podman,* and *python3* must be installed by issuing the command:

```
yum install -y git podman python3
```

If the build machine is the helper node, this might not be necessary.

Cloning the GitHub Repository

The GitHub repository containerization-for-sap-s4hana provides all the automation scripts required to build and deploy SAP images. The official repository was created and tested based on RHEL8.
Running the build process on RHEL9 raises some issues. A forked version of the original repository containing the fixes can be found here.

```
git clone https://github.com/SAPUCCTUM/containerization-for-sap-s4hana.git
```

The automation tools are written in Python. To execute them, a virtual Python environment needs to be set up:

```
cd containerization-for-sap-s4hana
tools/venv-setup
```

Before executing any of the tools, the virtual environment must be activated:

```
source venv/bin/activate
```

All Python scripts share a set of common parameters. In particular, the parameter *-v debug* provides detailed logging information in the corresponding logfile.

Copying the Compatibility Library for SLES

If the reference SAP system runs on SUSE® Linux Enterprise Server an additional GCC C++ runtime compatibility library may be required in the HANA database container image:

SAP HANA 2.0 SP Level	compatibility library
up to SAP HANA 2.0 SPS 04	none
SAP HANA 2.0 SPS 05	compat-sap-c++-9
SAP HANA 2.0 SPS 06	compat-sap-c++-10

Table 2: Compatibility Libraries related to SAP HANA SP Levels

The compatibility library can be downloaded from the Red Hat Package download (*RHEL for SAP Applications*)
The downloaded *rpm* must be placed in **/tmp/soos/rpm-packages/** on the build machine and is automatically copied to the image during the build process.

Creating an OpenShift Project

On OpenShift, objects such as deployments or pods are managed in *projects*.
A new project can be created by a user issuing the following commands:

```
oc login --insecure-skip-tls-verify=true https://api.<ocp-cluster-domain>:6443 -u <ocp-user-name>
oc new-project <ocp-project-name>
```

The following permissions need to be set:

```
oc login --insecure-skip-tls-verify=true https://api.<ocp-cluster-domain>:6443 -u kubeadmin
oc project <ocp-project-name>
oc adm policy add-scc-to-group anyuid "system:serviceaccounts:<ocp-project-name>"
oc adm policy add-scc-to-user hostmount-anyuid system:serviceaccount:<ocp-project-name>:<ocp-project-name>-sa
```

Creating the Credentials and Configuration Files

The credentials for accessing the reference SAP system, the NFS server and the OpenShift cluster are stored in the credentials file **creds.yaml.gpg.** This file is encrypted via GPG. In addition, the configuration file **config.yaml** contains all other necessary information to build and deploy the images.

When calling:

```
tools/creds -n
```

a new **creds.yaml.gpg** file is created interactively and automatically encrypted with a given encryption key.
This encryption key is needed for executing the Python scripts and is cached for a certain period. If an erroneous key is provided, the Python scripts stop working.
This can be solved by restarting the *gpg-agent*:

```
gpgconf --kill gpg-agent
```

and entering the correct encryption key.

After the credentials file is created, the configuration file can be generated by issuing:

```
tools/config -n
```

To ensure the correctness of the entered data the configuration can be checked using the following Python scripts:

```
tools/verify-config
tools/verify-ocp-settings
```

Stopping the Reference SAP System

Before the building process can be started, the reference SAP system must be stopped to avoid inconsistencies in the HANA database.
The SAP instances can be stopped using the following command:

```
<sidadm>: sapcontrol -nr <instanceNr.> -function StopSystem
```

If a central transport directory (transport group together with other SAP systems) is used, it should be unmounted:

```
umount /usr/sap/trans
```

Building Process

After these preparational steps the image build process can be started on the build machine.

Creating a Snapshot Copy of the Reference HANA Database

The first step of the image build process is to create a file-system snapshot of the HANA database. This snapshot contains the content of **/hana/data** and **/hana/log** and is provided via NFS to all deployments that are based on the same reference SAP system.
Depending on the size of the reference SAP system this step may take a while.
The snapshot copy is started by executing the following Python script:

```
tools/nfs-hdb-copy
```

Building the Container Images

The image build process can be started by issuing the following command:

```
tools/containerize -b
```

To have more control over the build process of each image, the image build process can be started individually for each image:

```
tools/image-build -f init
tools/image-build -f nws4
tools/image-build -f hdb
```

Pushing the Images to the Cluster Registry

The created images are only available locally on the build machine. To make them available to the OpenShift cluster, they must be pushed to the OpenShift cluster registry.
All three images can be transferred in one step:

```
containerize -p
```

or one after the other:

```
tools/image-push -f init
tools/image-push -f hdb
tools/image-push -f nws4
```

The result can be checked via the command:

```
oc get imagestream
```

The output should look like:

```
NAME        IMAGE
REPOSITORY
TAGS    UPDATED
soos-init   default-route-OpenShift-image-registry.apps.ocp4-<cluster>/<ocp-
project>/soos-init        latest    9 days ago
soos-s09   default-route-OpenShift-image-registry.apps.ocp4-<cluster>/<ocp-
project>/soos-s09       latest   9 days ago
```

```
soos-w09   default-route-OpenShift-image-registry.apps.ocp4-<cluster>/<ocp-
project>/soos-w09       latest   9 days ago
```

Deploying Process

As soon as the build process has completed, the first copy of the reference SAP system can be deployed in the OpenShift cluster.

Creating the Overlay Filesystem

As mentioned before in chapter Solution Architecture every copy of the reference SAP system works with its own NFS overlay filesystem to ensure data consistency, the overlay filesystem is created by executing the Python script:

```
tools/nfs-overlay-setup
```

This command returns a unique ID which is needed during the generation of the deployment description file.

Generating the Deployment Description File

The deployment description file contains all information that is required to instantiate a working deployment and is created by executing:

```
tools/containerize -d -u <overlay-uuid>
```

This deployment description file has the following naming format:

```
soos-<nws4-sid>-<uuid>-deployment.yaml
```

Each deployment which is based on the same reference SAP system is identified using a unique identifier *<uuid>*.

The command prints the filename of the generated deployment description file, which is needed for starting the deployment.

Starting the Deployment

The deployment can be started by executing:

```
tools/containerize -s -f soos-<nws4-sid>-<uuid>-deployment.yaml
```

Full Automated Building and Deploying

All steps mentioned in sections Building Process and Deploying Process can be fully automated by:

```
tools/containerize -a
```

Monitoring the Deployment

There are several ways to monitor the progress of the deployment. One possibility is the OpenShift Web Console, which is a powerful graphical tool that offers extensive insights.

This section describes the usage of the provided Python scripts for monitoring.

Displaying the State of a Deployment

A list of all deployments is shown via:

```
tools/ocp-deployment --list
```

The Python script distinguishes between three different states. A deployment which is in state *Running* has all containers up and running. It does not necessarily mean that the SAP instances in the containers are running too.
If a deployment is in state *Pending*, the deployment is scheduled, however starting the deployment has not yet finished or there are not enough resources for starting.
For deployments which are in *Prepared* state a deployment description file exists, but the deployment is not instantiated.

Checking the Status of the Containerized SAP System

To check if all SAP instances of the deployed copy are running the following command can be used:

```
tools/sap-system-status --process-list
```

The output looks like:

```
====================================
App-Name:  soos-s09-zhsvvinv6s
------------------------------------
ASCS       running
DI         running
HDB        running
```

Checking SAP System Logfiles

Since all logfiles of the SAP instances are stored on the NFS, they can be easily accessed without logging into the actual containers. Moreover, the NFS serves as a persistence layer, so even in the case of a total container crash this information can still be accessed. In the UCC setup this layer is located on the helper node in:

/export/soos/overlays/<overlay-uuid>/persistence/

This directory looks as follows:

```
├─── <HDB-SID>
│    └─── <SAPDBHOST>
└─── <NWS4-SID>
     ├─── ASCS<instno>
     ├─── D<instno>
     └─── profile -> /data/sap_mnt/<NWS4-SID>/profile
```

Logging into a Container

The processes running in a container can be easily checked by logging into it:

```
tools/ocp-container-login -i <container-flavor> --app-name <application-name>
```

where *<container-flavor>* is one of the following:

di	Dialog instance container, default
ascs	ASCS container
hdb	SAP HANA database container

Building and Deploying with Ansible®

Instead of performing the image build process and deployment manually, an Ansible playbook can be used to perform these steps automatically.

Preparing the Ansible Configuration File

To run the Ansible playbook the configuration file **ansible/vars/ocp-extra-vars.yml** needs to be set up according to the environment. If passwordless authentication is used some of the passwords can be left empty.

```
path_to_ocp_tool: /data/containerization-for-sap-s4hana
debug_level_ocp_tool: warning
tmp_root: /data/tmp
package_state: present
python3x_version: python3
config.j2.template
template_config_file: config.j2.template
creds.j2.template
template_creds_file: creds.j2.template
build_user_sshid: ''
ocp_cluster_domain: ocp4.<fqdn>
ocp_admin_name: kubeadmin
ocp_admin_password: <content of /root/ocp4-pvm/auth/kubeadmin-password>
ocp_user_name: <the non-admin user you have created>
ocp_user_password: <password>
ocp_project_name: <project from previous step e.g. anssos>
ocp_helper_node: <hostname helper>
ocp_helper_node_user_name: root
ocp_helper_node_user_password: ''
ocp_helper_node_user_sshid: ''
nws4_host_name: <hostname of reference system e.g. s09lp1>
nws4_sid: <S/4HANA SID e.g. S09>
nws4_sidadm_name: <e.g s09adm>
nws4_sidadm_password: ''
nws4_hdbconnect_name: ''
nws4_hdbconnect_password: ''
hdb_sid_deployment: <HANADB SID e.g. W09>
hdb_sidadm_name: <e.g. w09adm>
hdb_sidadm_password: ''
containers_di_requests_memory: 20Gi
containers_di_limits_memory: 64Gi
containers_ascs_requests_memory: 10Gi
containers_ascs_limits_memory: 10Gi
distributed_sap_system: <'no' for singlehost, otherwise 'yes'>
containers_di_secret: ''
containers_hdb_requests_memory: ''
containers_hdb_limits_memory: ''
nfs_host_name: ''
nfs_user_name: root
nfs_user_password: ''
nfs_path_to_hdb_copy: /export/soos/snapshots
nfs_path_to_overlay: /export/soos/overlays
github_repo_url:        https://github.com/IBM/containerization-for-sap-
s4hana.git
```

After the configuration file has been adjusted, the Ansible playbook can be started:

```
cd ./ansible
ansible-playbook -i hosts -e @vars/ocp-extra-vars.yml ocp-deployment.yml
```

The playbook installs all prerequisites, creates the three container images, pushes the images to the cluster registry and starts the deployment of the copied SAP system.

Ansible Restrictions

At this point, currently not all SAP on OpenShift management tasks are covered by Ansible. Therefore, some steps must still be performed via the Python scripts. The following table visualizes which steps are supported by Ansible and which have to be performed via the Python scripts.

Task	Ansible	Tools
Image Building	×	×
Image Pushing	×	×
Deploying the SAP system	×	×
Verifying the Deployment		×
Establishing Port Forwarding		×
Cleaning up the Environment		×

Table 3: Ansible RestrictionsManaging SAP Workloads

This chapter covers a set of management procedures to work with the containerized SAP systems. Use-cases include:

- adding and removing of deployments
- configuring access to the containerized SAP system
- starting and stopping of deployments
- etc.

Providing Access with HAProxy

The pods in the OpenShift cluster use an isolated network, meaning that access from outside the cluster is not possible. To allow clients (e.g., SAP GUI, SAP HANA Studio) access to the containerized SAP system deployment, an OpenShift *NodePort* service is required.
In addition, a forwarding has to be enabled to map the client requests to the cluster internal ports. Port forwarding is achieved by either SSH tunneling or a reverse proxy (HAProxy). In the UCC environment, access via SAP® GUI (port 3200), SAP HANA Studio (ports 30813 and 30815), as well as the SAP Fiori® launchpad (port 44300) is required.

Initial Port Forwarding

The *NodePort* service containing the port definitions for SAP GUI and SAP HANA Studio is defined in the generated deployment description file and is created at deployment start.
To configure the *HAProxy* the following command is used:

```
tools/ocp-haproxy-forwarding --add
```

The tool prints out the connection parameters for SAP GUI and SAP HANA Studio:

```
----------------------------------------
   System ID           S09
   Instance Number     04
   Application Server  helper   (xxx.xxx.xxx.xxx)
----------------------------------------


Use the following parameters to create a HANA Studio connection:

----------------------------------------
   Host Name           helper   (xxx.xxx.xxx.xxx)
   Instance Number     11
   System ID           W09
----------------------------------------
```

A full list of all port mappings can be displayed by issuing:

```
tools/ocp-haproxy-forwarding --list
```

Changing Port Forwarding during Runtime

As already noted, the deployment description file defines the ports for SAP GUI and SAP HANA Studio, but not for the SAP Fiori launchpad, which requires port 443*<di-instno>*. Therefore, the NodePort definition in the deployment description file must be extended manually:

```
kind: Service
…
spec:
  ports:
  - name: di-443xx
    port: 44300
    protocol: TCP
    targetPort: 44300
```

The "**di**" specifies the instance for which the port is configured (dialog instance in the UCC case).
To modify the existing OpenShift NodePort object the command:

```
oc apply -f "<deployment.yaml>"
```

must be executed.

OpenShift recognizes that the actual service definition differs from the new service definition. Hence it updates the NodePort object which can be checked using:

```
oc get all
```

In the UCC case the internal port 44300 was mapped to 31530. This port is required for updating the HAProxy configuration. Example output:

```
NAME                              TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)  AGE
service/soos-s09-zjkiadfreu-np    NodePort   172.30.52.227   <none>
3200:31420/TCP,8000:30088/TCP,44300:31530/TCP,30813:32467/TCP,30815:30137/TCP   20d
```

To establish HAProxy access for the SAP Fiori launchpad a free port on the helper node is required.
In the UCC setup port 44304 was used. Its availability was checked with the following command, which must not return any output:

```
lsof -i :<port>
```

A new forwarding rule must be added to the HAProxy configuration file **/etc/haproxy/haproxy.cfg**:

```
listen <deployment-id>-di-<port-to-connect>
    bind *:<port-to-connect>
    option tcplog
    server <worker> <worker-ip>:<mapped-port>  check
```

To enable the configuration HAProxy needs to be reloaded:

```
systemctl reload haproxy.service
```

Managing Deployments

Deployments in *Prepared* state can be started via the following command:

```
tools/ocp-deployment –start --app-name <application-name>
```

or stopped, if in *Running* state:

```
tools/ocp-deployment –stop --app-name <application-name>
```

When a deployment is stopped, it is not available in OpenShift anymore, but the local and NFS resources such as the deployment description file and the NFS overlay filesystem still exist.

The following command removes a deployment and all its resources:

```
tools/ocp-deployment --remove --app-name <application-name>
```

The script stops the deployment if it was running, deletes the NodePort service, tears down the NFS overlay filesystem and deletes the deployment description file. The HANA database snapshot is not removed.

Adding Deployments

Additional deployments can be easily added by using:

```
tools/ocp-deployment --add [-n <number-of-deployments>]
```

The Python script creates *<number-of-deployments>* deployments. For each deployment it creates a new NFS overlay filesystem based on the HANA database snapshot plus a deployment description file and starts the deployment. Technically this is identical to the steps performed in section Deploying Process.
Therefore, for each deployment for which SAP Fiori launchpad access is required the manual steps described section Changing Port Forwarding during Runtime must be performed.

Caution: The number of replicas in the deployment description file must not be increased since these replicas would represent multiple SAP HANA database instances which access the same database content on one overlay filesystem!

Conclusion and Next Steps

As explained in sections Business Requirements and Expectations and Hosting Requirements, the UCC business case enforces some external hosting and business requirements, while also setting particular expectations in the presented project. Especially, the UCC searches a solution for fast and easy scalable deployment of short-lived systems. This section describes which of these requirements are already met by the SAP on OpenShift containerization prototype and which ones need further research.

Business Requirements

Fast Deployment

This requirement is already met by the SAP on OpenShift containerization prototype, as setting up a new container from an already existing image is a matter of seconds. This also provides a high scalability for new systems, because as many containers as wanted can be set up (as long as the worker nodes possess sufficient resources).

Computing Resources Efficiency

This requirement is met by the prototype. The OpenShift solution is very efficient in terms of hardware resources. The containerization itself is more lightweight than a corresponding virtualization based on VMs. In addition, there is great potential for disk savings due to the single HANA database snapshot, which can be reused for multiple deployments.

High Degree of Automation

This requirement is met by the prototype. After setting up the OpenShift cluster, all remaining steps are highly automated. For example, the creation of images, the deployment of containers from an image, the start/stop and deletion of containers, etc. can be achieved with one single command each.

Fast and Easy Life-Cycle Management

This requirement is met by the prototype. Shutting down and dissolving containers is a one-command solution in OpenShift. In addition, the Kubernetes orchestration tools, also available in OpenShift, support advanced life-cycle management.

Hosting Requirements

Strictly Separated Containerized SAP Systems

This requirement is already met by the SAP on OpenShift containerization prototype, as each containerized SAP system is separated from the other systems. The UCC's partners can login to their specific containerized SAP system via SAP GUI or SAP Fiori launchpad due to the corresponding port forwardings. Still, prior to providing user access, the credentials of the SAP users must be set individually for each access as customers can reach other containers by intent or error through guessing or mistyping the port number.

Access via SAP GUI, Fiori Launchpad and SAP HANA Studio

As described in section Providing Access with HAProxy, the port forwarding allows accessing the containerized SAP systems with SAP GUI, SAP Fiori launchpad and SAP HANA Studio, therefore this requirement is met, too.

Shiftable Workload

The OpenShift cluster automatically shifts the workload from one worker node to another, in case of worker node failure. Nevertheless, workload migration between worker nodes must be manually enforced by stopping the deployment and starting the deployment again. Therefore, this requirement is not fully met. There is still room for more research how workload could be distributed, for example through more complex Kubernetes functions supported by OpenShift, or more complex operations on server level like live-partition-mobility.

HANA Databases Connectivity to the UCC's TSM Backup Nodes

The HANA database of one containerized SAP system can be backed up using the file-backup option at HANA studio writing on the persistence layer. Backint backup using the UCC's TSM nodes has not been evaluated, yet.

Easily Stoppable and Restartable SAP systems

Due to the containerized nature of this OpenShift solution, this requirement is met.

SAP Solution Manager Integration

As the containerized copies of one reference SAP system share the same SAP system ID, the deployed SAP systems cannot be easily integrated in the SAP Solution Manager. This constraint affects multiple of the hosting requirements. See also Future Work.

SAP Cloud Connector and SAP Analytics Cloud Agent

Not evaluated.

Integration into the UCC's Transport Domain

The domain controller of an SAP transport domain expects unique SAP system Ids for all SAP systems within the domain. As presently all containerized systems use the SAP system Id of the reference system, multiple containerized systems cannot be integrated into the UCC's transport domain.

Future Work

The requirement of different external SAP system Ids that is essential for SAP Solution Manager monitoring, cloud connection, transport domain, HANA backint backup and landscape management, is not supported by the present prototype. In further research, therefore multiple questions should be addressed:

1. How can a unique SAP system Id be assigned to each containerized SAP system?
2. Can the SAP system Id of a containerized system automatically be changed during container creation?
3. Can additional agents such as SAP Host Agent, SAP Diagnostics Agent or SAP Analytics Cloud Agent be deployed in the containers and work properly in these?
4. Can the containerized SAP systems be connected to SAP Cloud products and to the SAP Cloud Connector?
5. How can HANA backint backup into the UCC's TSM nodes be realized?

Answering these questions would lead to a prototype that is fully operational within the UCC business case.

Summary Assessment of the Present Status

The lack of distinct SAP system Ids limits the usability as previously explained, as the systems cannot be integrated into the existing UCC landscape and monitoring. But, as descriped in section Business Requirements and Expectations, the first and foremost goal of this research was to provide an easy way to start up, shut down, maintain, and manage SAP systems for short-time use, like training or demo systems for two month trials or singular train-the-trainer sessions. This use case is already fully operational.

Furthermore, in this use case of short-lived systems, the restrictions explicated above are negligible, as short-lived systems do not need a transport system implemented or patches installed. Furthermore, short-lived trial or training systems do not need the same monitoring effort as the long-living systems usually used at the UCC Munich. The open question of cloud connection limits the prototype to on-premises only products, but currently these still make up the majority of UCC products. Considering all these factors, it can be concluded that the SAP on OpenShift containerization solution is suitable for the use case described at the beginning of this document.

The OpenShift solution can be used at the UCC and the prototype serves as an excellent starting point for further research projects. Please be aware that this solution is not supported by SAP for productive environments.