Administration Guide

SUSE CaaS Platform 3

Liam Proven, Christoph Wickert, Markus Napp, Sven Seeberg-Elverfeldt, Jana Halačková

Version 1.0

Contents

About This Guide 2

Required Background 2

Available Documentation 3

Feedback 4

Documentation Conventions 5

About the Making of This Documentation 7

1. Security 8

2. Security 9

2.1. Access Control Overview 9

2.2. Administrator Accounts 10

2.3. Preparing LDAP Authentication 12

2.4. Managing Users and Groups 12

2.5. Role Management 16

2.6. Certificates 21

2.7. Pod Security Policies 23

2.8. Security Audit Log 29

2.9. Configuring External LDAP Server 29

3. Cluster Management 38

3.1. Interacting With Kubernetes 38

3.2. Interacting with Salt 39

3.3. Node Management 40

3.4. Graceful Shutdown and Startup 44

3.5. Scaling the Cluster 48

3.6. Configuring Remote Container Registry 49

3.7. Configuring A Registry Mirror 50

3.8. Reserving Compute Resources 52

4. Software Management 53

4.1. Transactional Updates 53

4.2. Program Temporary Fixes 59

4.3. Upgrading From SUSE CaaS Platform 2 60

4.4. Additional Software Installation 64

4.5. Installing Kubernetes Dashboard 68

5. Monitoring 70

6. Monitoring 71

6.1. Monitoring Stack On Kubernetes 71

6.2. Health Checking 86

7. Logging 94

8. Logging 95

8.1. About SUSE CaaS Platform Logging 95

8.2. Admin Node Logs . . . 96
8.3. Salt Logging . . . 96
8.4. Transactional Update Log . . . 99
8.5. Kubernetes Audit Log . . . 100
8.6. External log collection . . . 102
9. Miscellaneous Configuration 103
10. Miscellaneous Configuration 104
10.1. Configuring Timezone . . . 104
10.2. The `issue-generator` Command . . . 104
11. SUSE Enterprise Storage Integration 106
11.1. Prerequisites . . . 106
11.2. Using RBD in a Pod . . . 106
11.3. Using RBD in Persistent Volume . . . 108
12. Troubleshooting 112
13. Troubleshooting 113
13.1. Overview . . . 113
13.2. New Nodes Not Showing In Velum . . . 113
13.3. Debugging Failed Bootstrap . . . 115
13.4. Recovering From Failed Update . . . 117
13.5. Checking `etcd` Health . . . 117
13.6. Locking Installed Program Temporary Fixes . . . 117
13.7. Network Planning And Reconfiguration . . . 118
13.8. Using A Proxy Server With Authentication . . . 118
13.9. Replacing TLS/SSL Certificates . . . 118
13.10. Fixing “x509: certificate signed by unknown authority” . . . 120
13.11. Replacing a Lost Node . . . 121
Appendix A: GNU Licenses 122
GNU Free Documentation License 123
Appendix B: Documentation Changelog 132
January 2019 . . . 132
December 2018 . . . 135
November 2018 . . . 136
October 2018 . . . 137
September 2018 . . . 139
August 2018 . . . 141

Information on administration of a SUSE® CaaS Platform cluster. Discusses authorization, cluster and node management, software management, monitoring, logging, use of Helm and Tiller, integration with SUSE Enterprise Storage, and troubleshooting advice.

About This Guide

Required Background

To keep the scope of these guidelines manageable, certain technical assumptions have been made:

- You have some computer experience and are familiar with common technical terms.
- You are familiar with the documentation for your system and the network on which it runs.
- You have a basic understanding of Linux systems.

Available Documentation

We provide HTML and PDF versions of our books in different languages. Documentation for our products is available at http://www.suse.com/documentation/, where you can also find the latest updates and browse or download the documentation in various formats.

The following documentation is available for this product:

[_book.caasp.deployment]

The SUSE CaaS Platform deployment guide gives you details about installation and configuration of SUSE CaaS Platform along with a description of architecture and minimum system requirements.

[_book.caasp.installquick]

The SUSE CaaS Platform quick start guides you through installation of a minimum cluster in a fastest way as possible.

Administration Guide: SUSE CaaS Platform 3

The SUSE CaaS Platform Admin Guide discusses authorization, updating clusters and individual nodes, monitoring, use of Helm and Tiller, the Kubernetes dashboard, and integration with SUSE Enterprise Storage.

Feedback

Several feedback channels are available:

Bugs and Enhancement Requests

For services and support options available for your product, refer to http://www.suse.com/support/.

To report bugs for a product component, go to https://scc.suse.com/support/requests, log in, and click **Create New** .

User Comments

We want to hear your comments about and suggestions for this manual and the other documentation included with this product. Use the User Comments feature at the bottom of each page in the online documentation or go to http://www.suse.com/documentation/feedback.html and enter your comments there.

Mail

For feedback on the documentation of this product, you can also send a mail to `doc-team@suse.com`. Make sure to include the document title, the product version and the publication date of the documentation. To report errors or suggest enhancements, provide a concise description of the problem and refer to the respective section number and page (or URL).

Documentation Conventions

The following notices and typographical conventions are used in this documentation:

- `/etc/passwd` : directory names and file names
- `PLACEHOLDER`: replace `PLACEHOLDER` with the actual value
- `PATH`: the environment variable PATH
- `ls`, `--help`: commands, options, and parameters
- `user` : users or groups
- package name : name of a package
- `Alt` , `+F1` : a key to press or a key combination; keys are shown in uppercase as on a keyboard
- **File ›] › menu:File[Save As** : menu items, buttons
- This paragraph is only relevant for the AMD64 /Intel 64 architecture. The arrows mark the beginning and the end of the text block.

  This paragraph is only relevant for the architectures `z Systems` and `POWER`. The arrows mark the beginning and the end of the text block.
- *Dancing Penguins* (Chapter *Penguins*, {uarr} Another Manual): This is a reference to a chapter in another manual.
- Commands that must be run with <systemitem xmlns='http://docbook.org/ns/docbook' class='username'>root</systemitem> privileges. Often you can also prefix these commands with the `sudo` command to run them as non-privileged user.

  ```
  {prompt.root}``command`` {prompt.user}``sudo command``
  ```
- Commands that can be run by non-privileged users.

  ```
  {prompt.user}``command``
  ```
- Notices

Warning Notice

Vital information you must be aware of before proceeding. Warns you about security issues, potential loss of data, damage to hardware, or physical hazards.

+ .Important Notice IMPORTANT: Important information you should be aware of before proceeding.

+ .Note Notice NOTE: Additional information, for example about differences in software versions.

Tip Notice

Helpful information, like a guideline or a piece of practical advice.

About the Making of This Documentation

This documentation is written in SUSEDoc, a subset of DocBook 5. The XML source files were validated by `jing` (see https://code.google.com/p/jing-trang/), processed by `xsltproc`, and converted into XSL-FO using a customized version of Norman Walsh's stylesheets. The final PDF is formatted through FOP from Apache Software Foundation. The open source tools and the environment used to build this documentation are provided by the DocBook Authoring and Publishing Suite (DAPS). The project's home page can be found at https://github.com/openSUSE/daps.

The XML source code of this documentation can be found at https://github.com/SUSE/doc-caasp.

Chapter 1. Security

Chapter 2. Security

***This section introduces the security features of SUSE CaaS** Platform . This includes authentication, management of users and groups, and the configuration of audit log files.*

2.1. Access Control Overview

SUSE CaaS Platform provides role-based access control (RBAC) to perform authentication and authorization of activities performed against a Kubernetes cluster. Kubernetes uses several steps for access control. The steps in processing order are:

Authentication

The authentication confirms the identity of an user. The *OpenID Connect* (*OIDC*) implementation `dex` is used as an authentication provider by Kubernetes

1. `dex` connects to an LDAP server to look up user information. To authenticate against the cluster, the `kubeconfig` file is used. For details, refer to Interacting With Kubernetes.

Authorization

SUSE CaaS Platform uses *role-based access control* (*RBAC*). For Kubernetes , RBAC defines which subjects can do which operations on API resources. Groups and users, called *subjects* in Kubernetes , are managed with LDAP. For details, refer to Managing Users and Groups. To define which operations, called *verbs* in Kubernetes , are allowed for subjects, refer to Role Management.

Admission

Admission controllers in Kubernetes can mutate and validate requests. For details, refer to https://v1-11.docs.kubernetes.io/docs/reference/access-authn-authz/admission-controllers/.

For details about access control in Kubernetes , refer to https://v1-11.docs.kubernetes.io/docs/reference/access-authn-authz/controlling-access/.

2.2. Administrator Accounts

By default there are two administrator accounts added to openLDAP:

openLDAP admin user

`cn=admin,dc=infra,dc=caasp,dc=local`

This is the "root" user for openLDAP and has full permissions to change credentials.

When the instructions in this section ask for `LDAP_ADMIN_PASSWORD` they require the openLDAP admin password.

You can retrieve the current password from the admin node.

```
{prompt.user}``cat /var/lib/misc/infra-secrets/openldap-password``
c88a9c67056a74e0357befdff93f87bbe0904214
```

Velum admin user

`uid=test,ou=People,dc=infra,dc=caasp,dc=local`

This account does not have privileges to change administrator passwords in openLDAP.

The account is created by the user on first login to Velum and consequently for configuring SUSE CaaS Platform .

2.2.1. Changing openLDAP Admin User Password

Procedure: Changing openLDAP Admin User Password

1. Log in to the admin node via SSH.
2. Change the openLDAP admin password in the file `/var/lib/misc/infra-secrets/openldap-password` .

```
{prompt.user}``echo new_password > /var/lib/misc/infra-
secrets/openldap-password``
```

3. Restart openLDAP to activate the changes.

```
{prompt.user}``docker stop $(docker ps -q -f name=ldap)``
```

After about 20 seconds, the OpenLDAP container will be automatically restarted with the new root password. This will only replace the credentials for the openLDAP admin user and not affect any other user configurations.

2.2.2. Changing Velum Admin Password

Changing the Velum admin password must be done in the openLDAP container running on the admin node. You will need the openLDAP admin password that is stored on the admin node itself.

Procedure: Changing VelumAdmin Password

1. Log in to the admin node via SSH.
2. Open a shell session on the openLDAP container.

```
{prompt.user}``docker exec -it $(docker ps -q -f name=ldap)
/bin/bash``
```

3. Use slappasswd to generate a hashed and salted password string.

```
{prompt.user}``slappasswd -n -s password``
{SSHA}mU7vDqF+cyNQlnQ2bZyvY4oFfjX9uDm3
```

By ommitting the -s <password> parameter, you will be prompted to enter a new secret instead of providing it through the input.

4. Find the distinguished name (DN) user string for the administrator user.

```
{prompt.user}``UNAME=$(slapcat -n1 | grep uniqueMember | cut -d': '
-f2)``
```

5. Set the new password. Replace new_password with the string you generated in the previous step. Replace $UNAME will replace it with the DN from the previous step (e.g. uid=user,ou=People,dc=infra,dc=caasp,dc=local).

```
{prompt.user}``ldappasswd -H ldaps:// -D
"cn=admin,dc=infra,dc=caasp,dc=local" \
-w $(cat /var/lib/misc/infra-secrets/openldap-password) \
$UNAME -s new_password``
```

2.3. Preparing LDAP Authentication

To perform administrative tasks from a `WORKSTATION` on the LDAP directory, retrieve the OpenLDAP administrator password and install the LDAP certificate.

1. Retrieve the LDAP admin password. Note the password for later use.

```
{prompt.root.admin}``cat /var/lib/misc/infra-secrets/openldap-
password``
```

2. Import the LDAP certificate to your local trusted certificate storage. On the Administration Node , run:

```
{prompt.root.admin}``docker exec -it $(docker ps -q -f name=ldap) \
cat /etc/openldap/pki/ca.crt > ~/ca.pem`` {prompt.root.admin}``scp
~/ca.pem root@WORKSTATION:/usr/share/pki/trust/anchors/ca-
caasp.crt.pem``
```

Replace `WORKSTATION` with the appropriate hostname for the workstation where you wish to run the LDAP queries.

3. Then, on that workstation, run:

```
{prompt.root}``update-ca-certificates``
```

2.4. Managing Users and Groups

User information is stored in OpenLDAP running in a container on your SUSE CaaS PlatformAdministration Node . You can use standard LDAP administration tools for managing these users remotely. To do so, install the openldap2 package on a computer in your network and make sure that computer can connect to the Administration Node on port `389`.

2.4.1. Adding New User

By default, when you create the first user in Velum during bootstrap of your cluster, that user is granted `Cluster Administrator` privileges within Kubernetes . You can add additional users with these rights by adding new entries into the LDAP directory.

1. To add a new user, create a LDIF file like this:

   Example 1. LDIF File For a New User

   ```
   dn: uid=`USERID` <1>,ou=People,dc=infra,dc=caasp,dc=local
   objectClass: person
   objectClass: inetOrgPerson
   objectClass: top
   uid:`USERID`<<_co.admin.security.users.add.uid>>userPassword:`PA
   SSWORD_HASH` <2>givenname:`FIRST_NAME` <3>sn:`SURNAME`
   <4>cn:`FULL_NAME` <5>mail:`E-MAIL_ADDRESS` ⑥
   ```

   <1> User ID (UID) of the new user. Needs to be unique.

   <2> The user's hashed password. Use `/usr/sbin/slappasswd`

   ```
   to generate the hash.
   ```

   <3> The user's first name

   <4> The user's last name

   <5> The user's full name

   <6> The user's e-mail address. It is used as the login name to Velum

   ```
             and {kube}
   .
   ```

2. Populate your OpenLDAP server with this LDIF file:

   ```
   {prompt.user}``ldapadd -H ldap://ADMINISTRATION_NODE_FQDN:389 -ZZ \
   -D cn=admin,dc=infra,dc=caasp,dc=local -w LDAP_ADMIN_PASSWORD -f
   LDIF_FILE``
   ```

2.4.2. Showing User Attributes

To show the attributes of a user, use the `ldapsearch` command.

```
{prompt.user}``ldapsearch -H ldap://ADMINISTRATION_NODE_FQDN:389 -ZZ \
    -D cn=admin,dc=infra,dc=caasp,dc=local -w LDAP_ADMIN_PASSWORD \
    -b uid=USERID,ou=People,dc=infra,dc=caasp,dc=local``
```

2.4.3. Changing User

The following procedure details how to modify a user in the LDAP directory. The example LDIF files detail how to change a user password and add a user to the `Administrators` group. To modify other fields, use the the password example and replace `userPassword` with other field names.

1. Create a LDIF file that contains the change to the LDAP directory.

   Example 2. Change User Password

   ```
   dn: uid=`USERID` <7>,ou=People,dc=infra,dc=caasp,dc=local
   changetype: modify
   replace: userPassword
   userPassword:`PASSWORD` ⑧
   ```

   <7> `USERID` with the user's ID.

   <8> `PASSWORD` with the user's new hashed password. Use `/usr/sbin/slappasswd` to generate the hash.

   Example 3. Add User to Administrators Group

   ```
   dn: cn=Administrators,ou=Groups,dc=infra,dc=caasp,dc=local
   changetype: modify
   add: uniqueMember
   uniqueMember: uid=`USERID`
   <9>,ou=People,dc=infra,dc=caasp,dc=local
   ```

   <9> `USERID` with the user's ID.

2. Execute `ldapmodify`.

```
{prompt.user}``ldapmodify -H ldap://ADMIN_NODE:389 -ZZ -D
cn=admin,dc=infra,dc=caasp,dc=local \
-w LDAP_ADMIN_PASSWORD -f LDIF_FILE``
```

2.4.4. Deleting User

The following procedure details how to delete a user from the LDAP database.

1. Create an LDIF file that contains that specifies the distinguished name of the entry and a deletion command.

```
dn: uid=`USER_ID`,ou=People,dc=infra,dc=caasp,dc=local
changetype: delete
```

2. Execute `ldapmodify`.

```
{prompt.user}``ldapmodify -H ldap://ADMIN_NODE:389 -ZZ -D
uid=USER_ID,ou=People,dc=infra,dc=caasp,dc=local \
-w LDAP_ADMIN_PASSWORD -f LDIF_DELETE``
```

2.4.5. Adding New Group

To grant users access to manage a single namespace in Kubernetes , first create your users as mentioned in Adding New User. Then execute the following procedure.

1. Create a LDIF file for a new group:

Example 4. LDIF File to Add a New Group

```
dn: cn=`group name` <10>,ou=Groups,dc=infra,dc=caasp,dc=local
objectclass: top
objectclass: groupOfUniqueNames
cn:`group name`<<_co.admin.security.groups.cn>>uniqueMember:
uid=`member1`, <11>ou=People,dc=infra,dc=caasp,dc=local
uniqueMember:
uid=`member2`,<<_co.admin.security.groups.member>>ou=People,dc=i
nfra,dc=caasp,dc=local
uniqueMember:
uid=`member3`,<<_co.admin.security.groups.member>>ou=People,dc=i
nfra,dc=caasp,dc=local
```

<10> The group's name.

<11> Members of the group. Repeat the `uniqueMember`

```
attribute for every member of this group.
```

2. Populate your OpenLDAP server with the LDIF file:

```
{prompt.user}``ldapadd -H ldap://ADMINISTRATION_NODE_FQDN:389 -ZZ \
-D cn=admin,dc=infra,dc=caasp,dc=local -w LDAP_ADMIN_PASSWORD -f
LDIF_FILE``
```

2.5. Role Management

SUSE CaaS Platform uses *role-based access control* authorization for Kubernetes . Roles define, which *subjects* (users or groups) can use which *verbs* (operations) on *resources*. The following sections provide an overview of resources, verbs and how to create roles. Roles can then be assigned to users and groups.

2.5.1. List of Verbs

This section provides an overview of the most common *verbs* (operations) used for defining roles. Verbs correspond to sub-commands of `kubectl`.

create

Create a resource.

delete

Delete resources.

deletecollection

Delete a collection of CronJob.

get

Display individual resource.

list

Display collections.

patch

Update an API object in place.

proxy

Allows running <command xmlns='http://docbook.org/ns/docbook'>kubectl</command> in a mode where it acts as a reverse proxy.

update

Update fields of a resource, for example annotations or labels.

watch

Watch resource.

2.5.2. List of Resources

This section provides an overview of the most common *resources* used for defining roles.

Autoscaler

https://v1-11.docs.kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/

ConfigMaps

https://v1-11.docs.kubernetes.io/docs/tasks/configure-pod-container/configure-pod-configmap/

Cronjob

https://v1-11.docs.kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/

DaemonSet

https://v1-11.docs.kubernetes.io/docs/concepts/workloads/controllers/daemonset/

Deployment

https://v1-11.docs.kubernetes.io/docs/concepts/workloads/controllers/deployment/

Ingress

https://v1-11.docs.kubernetes.io/docs/concepts/services-networking/ingress/

Job

https://v1-11.docs.kubernetes.io/docs/concepts/workloads/controllers/jobs-run-to-completion/

Namespace

https://v1-11.docs.kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/

Node

https://v1-11.docs.kubernetes.io/docs/concepts/architecture/nodes/

Pod

https://v1-11.docs.kubernetes.io/docs/concepts/workloads/pods/pod-overview/

PV

https://v1-11.docs.kubernetes.io/docs/concepts/storage/persistent-volumes/

Secrets

https://v1-11.docs.kubernetes.io/docs/concepts/configuration/secret/

Service

https://v1-11.docs.kubernetes.io/docs/concepts/services-networking/service/

ReplicaSets

https://v1-11.docs.kubernetes.io/docs/concepts/workloads/controllers/replicaset/

2.5.3. Create Role

Roles are defined in YAML files. To apply role definitions to Kubernetes , use `kubectl apply -f [replaceable]`YAML_FILE````. The following examples provide an overview about different use cases of roles.

Example 5. Simple Role for Core Resource

This example allows to `get`, `watch` and `list` all `pods` in the namespace `default`.

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: view-pods <12>namespace: default <13>rules:
- apiGroups: [""] <14>resources: ["pods"] <15>verbs: ["get",
"watch", "list"] ⑯
```

<12> Name of the role. This is required to associate the rule with a group or user. For details, refer to Create Role Bindings .

<13> Namespace the new group should be allowed to access. Use `default` for Kubernetes ' default namespace.

<14> Kubernetes API groups. Use ` ""` for the core group `rbac.authorization.k8s.io` .

<15> Kubernetes resources. For a list of available resources, refer to List of Resources .

<16> Kubernetes verbs. For a list of available verbs, refer to List of Verbs .

Example 6. Clusterwide Creation of Pods

This example allows to `create`pods clusterwide. Note the `ClusterRole` value for `kind`.

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: admin-create-pods <17>rules:
- apiGroups: [""] <18>resources: ["pods"] <19>verbs: ["create"] ⑳
```

<17> Name of the role. This is required to associate the rule with a group or user. For details, refer to Create Role Bindings .

<18> Kubernetes API groups. Use ` ""` for the core group `rbac.authorization.k8s.io` .

<19> Kubernetes resources. For a list of available resources, refer to List of Resources .

<20> Kubernetes verbs. For a list of available verbs, refer to List of Verbs .

2.5.4. Create Role Bindings

To bind a group or user to a rule, create a YAML file that contains the role binding description. Then apply the binding with `kubectl apply -f [replaceable]`YAML_FILE````. The following examples provide an overview about different use cases of role bindings.

Example 7. Binding a Group to a Role

This example shows how to bind a group to a defined role.

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name:`ROLE_BINDING_NAME` <21>namespace:`NAMESPACE` <22>subjects:
- kind: Group
  name:`LDAP_GROUP_NAME` <23>apiGroup: rbac.authorization.k8s.io
roleRef:
- kind: Role
  name:`ROLE_NAME` <24>apiGroup: rbac.authorization.k8s.io
```

<21> Defines a name for this new role binding.

<22> Name of the namespace for which the binding applies.

<24> Name of the role used. For defining rules, refer to Create Role .

<23> Name of the LDAP group to which this binding applies. For creating groups, refer to Adding New Group .

2.6. Certificates

During the installation of SUSE CaaS Platform , a CA (Certificate Authority) certificate is generated; that is then used to authenticate and verify all communications. The process also creates and distributes client certificates for the components.

Communication is secured with TLS v1.2 using the `AES 128 CBC` cipher.

All client certificates are 4096 Bit RSA encrypted.

Certificates are located in `/etc/pki` on each cluster node.

2.6.1. Certificate Renewal

The CA certificate is valid for `3650` days (10 years) by default.

The client certificates are valid for `365` days (1 year) by default.

All certificates have a renewal period of `90` days before expiration. If orchestration of the cluster is run during that period, the certificates which are about to expire are renewed automatically.

To manually renew certificates, refer to Replacing TLS/SSL Certificates.

Renewing Expired Certificates

If for whatever reason any of the certificates have failed to renew, please log in to Velum and navigate to **Settings ›] . Click the menu:Apply changes[** button. This will force a refresh of the cluster settings and any expired certificates will be renewed.

If this still fails, you can replace the certificates manually. Refer to: Replacing TLS/SSL Certificates.

2.6.2. Obtaining and Installing Root CA Certificate

1. Obtain the root CA certificate from any node in your cluster with `scp`.

```
{prompt.user}``scp NODE:/etc/pki/trust/anchors/SUSE_CaaSP_CA.crt .``
```

2. Copy the Root CA certificate file into the trust anchors directory `/etc/pki/trust/anchors/` .

```
{prompt.sudo}cp`SUSE_CaaSP_CA`.crt /etc/pki/trust/anchors/
```

3. Update the cache for known CA certificates.

```
{prompt.sudo}``update-ca-certificates``
```

Operating System Specific Instructions

The location of the trust store anchors directory or the command to refresh the CA certificates cache might vary depending on your operating system.

Please consult the official documentation for your operating system to find the respective alternatives.

2.7. Pod Security Policies

This section provides an overview of policy settings for pod security. By default, pod security policies are already enabled on SUSE CaaS Platform .

SUSE CaaS Platform comes with 2 pre-defined policies. These policies are detailed in the examples below, including the required role definitions. All authenticated users and service accounts are given the role `suse:caasp:psp:unprivileged`. Other role bindings have to be created manually. For details about roles and role bindings, refer to Role Management.

Unprivileged Pod Security Policy

This is the default policy. It is a compromise between security and daily needs. This policy is bound to the role `suse:caasp:psp:unprivileged`.

Privileged Pod Security Policy

This policy has few restrictions and should only be given to highly trusted users. This policy is bound to the role `suse:caasp:psp:privileged`.

Privileged DaemonSet

This example details how to define a privileged DaemonSet with a new default service account.

To create new policies, you can adapt the provided example policies to your needs. Then copy them into a YAML file and apply the definition by executing `kubectl apply -f [replaceable]`YAML_FILE````.

Detailed information is available at https://v1-11.docs.kubernetes.io/docs/concepts/policy/pod-security-policy/.

Example 8. Unprivileged Pod Security Policy

The unprivileged Pod Security Policy is intended to be a reasonable compromise between the reality of Kubernetes workloads and the role `suse:caasp:psp:privileged`. By default, SUSE CaaS Platform grants this policy to all users and service accounts.

```
---
apiVersion: extensions/v1beta1
kind: PodSecurityPolicy
metadata:
  name: suse.caasp.psp.unprivileged <25>annotations:
```

```
    seccomp.security.alpha.kubernetes.io/allowedProfileNames:
docker/default
    seccomp.security.alpha.kubernetes.io/defaultProfileName:
docker/default
    apparmor.security.beta.kubernetes.io/allowedProfileNames:
runtime/default
    apparmor.security.beta.kubernetes.io/defaultProfileName:
runtime/default
spec:
  # Privileged
  privileged: false
  # Volumes and File Systems
  volumes:
    # Kubernetes Pseudo Volume Types
    - configMap
    - secret
    - emptyDir
    - downwardAPI
    - projected
    - persistentVolumeClaim
    # Networked Storage
    - nfs
    - rbd
    - cephFS
    - glusterfs
    - fc
    - iscsi
    # Cloud Volumes
    - cinder
    - gcePersistentDisk
    - awsElasticBlockStore
    - azureDisk
    - azureFile
    - vsphereVolume
  allowedHostPaths:
    # Note: We don't allow hostPath volumes above, but set this to a
path we
    # control anyway as a belt+braces protection. /dev/null may be a
better
    # option, but the implications of pointing this towards a device
are
    # unclear.
    - pathPrefix: /opt/kubernetes-hostpath-volumes
  readOnlyRootFilesystem: false
  # Users and groups
  runAsUser:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
  # Privilege Escalation
  allowPrivilegeEscalation: false
  defaultAllowPrivilegeEscalation: false
```

```
    # Capabilities
    allowedCapabilities: []
    defaultAddCapabilities: []
    requiredDropCapabilities: []
    # Host namespaces
    hostPID: false
    hostIPC: false
    hostNetwork: false
    hostPorts:
    - min: 0
      max: 65535
    # SELinux
    seLinux:
      # SELinux is unsed in CaaSP
      rule: 'RunAsAny'
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name:
suse:caasp:psp:unprivileged<<_co.admin.security.pod_policies.unprivi
leged.name>>rules:
  - apiGroups: ['extensions']
    resources: ['podsecuritypolicies']
    verbs: ['use']
resourceNames:
['suse.caasp.psp.unprivileged']<<_co.admin.security.pod_policies.unp
rivileged.name>>
```

<25> Make sure to change the policy and role name when adapting the example for your own policies.

Example 9. Privileged Pod Security Policy

The privileged Pod Security Policy is intended to be given only to trusted workloads. It provides for as few restrictions as possible and should only be assigned to highly trusted users.

```
---
apiVersion: extensions/v1beta1
kind: PodSecurityPolicy
metadata:
  name: suse.caasp.psp.privileged <26>annotations:
    seccomp.security.alpha.kubernetes.io/defaultProfileName:
docker/default
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: '*'
    apparmor.security.beta.kubernetes.io/defaultProfileName:
runtime/default
spec:
  # Privileged
  privileged: true
  # Volumes and File Systems
  volumes:
    # Kubernetes Pseudo Volume Types
    - configMap
    - secret
    - emptyDir
    - downwardAPI
    - projected
    - persistentVolumeClaim
    # Kubernetes Host Volume Types
    - hostPath
    # Networked Storage
    - nfs
    - rbd
    - cephFS
    - glusterfs
    - fc
    - iscsi
    # Cloud Volumes
    - cinder
    - gcePersistentDisk
    - awsElasticBlockStore
    - azureDisk
    - azureFile
    - vsphereVolume
  #allowedHostPaths: []
  readOnlyRootFilesystem: false
  # Users and groups
  runAsUser:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
  # Privilege Escalation
  allowPrivilegeEscalation: true
  defaultAllowPrivilegeEscalation: true
  # Capabilities
  allowedCapabilities:
    - '*'
  defaultAddCapabilities: []
  requiredDropCapabilities: []
  # Host namespaces
  hostPID: true
  hostIPC: true
  hostNetwork: true
  hostPorts:
  - min: 0
    max: 65535
  seLinux:
    # SELinux is unsed in CaaSP
    rule: 'RunAsAny'
```

```
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name:
suse:caasp:psp:privileged<<_co.admin.security.pod_policies.privilege
d.name>>rules:
  - apiGroups: ['extensions']
    resources: ['podsecuritypolicies']
    verbs: ['use']
resourceNames:
['suse.caasp.psp.privileged']<<_co.admin.security.pod_policies.privi
leged.name>>
```

<26> Make sure to change the policy and role name when adapting the example for your own policies.

Example 10. Privileged DaemonSet

This example details how to create a privileged DaemonSet which uses the role `suse:caasp:psp:privileged`.

```
---
apiVersion: v1
kind: Namespace
metadata:
  name:`NAMESPACE`---
apiVersion: v1
kind: ServiceAccount
metadata:
  name:`SERVICE_ACCOUNT_NAME`namespace:`NAMESPACE`---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name:`ROLE_BINDING_NAME`namespace:`NAMESPACE`roleRef:
  kind: ClusterRole
  name: suse:caasp:psp:privileged
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: ServiceAccount
  name:`SERVICE_ACCOUNT_NAME`namespace:`NAMESPACE`---
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name:`DAEMONSET_NAME`namespace:`NAMESPACE`spec:
  selector:
    matchLabels:
      name:`DAEMONSET_NAME`template:
    metadata:
      labels:
        name:`DAEMONSET_NAME`spec:
      serviceAccountName:`SERVICE_ACCOUNT_NAME`hostPID: true
      hostIPC: true
      hostNetwork: true
      nodeSelector:
        beta.kubernetes.io/arch: amd64
      containers:
      - name:`CONTAINER_NAME`image:`IMAGE_NAME`volumeMounts:
        - name: examplemount
          mountPath: /something
        securityContext:
          privileged: true
      volumes:
      - name: examplemount
        hostPath:
          path: /var/log
```

2.8. Security Audit Log

To enable the Kubernetes security audit log please see: Kubernetes Audit Log

2.9. Configuring External LDAP Server

You can configure the cluster to authenticate Velum and Kubernetes users against a pre-existing Lightweight Directory Access Protocol (LDAP) server and use LDAP Filters to select the scope of users that will be permitted access.

Automatic Attributes from LDAP

Please note that users that belong to the `administrators` group in LDAP will automatically be assigned the role of `cluster-admin`.

Procedure: Configuring External LDAP connector

1. Log in to Velum
2. Access the LDAP configuration settings under **Settings → EXTERNAL AUTHENTICATION → LDAP Connectors** .
3. Click on **Add LDAP connector** to add a new connector.
4. Configure the connector.
5. Test the connector.
6. Save your connector settings.

2.9.1. LDAP Connector Settings

LDAP Anonymous Binding

Anonymous binding is available, if allowed by the LDAP server.

[velum settings ldap] | *velum_settings_ldap.png*

Name
: Name shown to user when selecting a connector

Server

Basic settings for the LDAP server host

Host

Host name of LDAP server reachable from the cluster

> *Provide the hostname as FQDN*
>
> The **Host** field must use a Fully Qualified Domain Name, as IP address is not allowed with TLS.

Port

The port on which to connect to the host (e.g. `StartTLS: 389, TLS: 646`)

StartTLS

When enabled use StartTLS otherwise TLS will be used

Certificate

The **Certificate** field must be a Base64-encoded PEM key.

Authentication

Anonymous

Use anonymous authentication to do initial user search.

Selects if you wish to perform an anonymous bind with the LDAP server. If set to **False ›] you must provide a menu:DN[** and a **Password ›] . The latter two are hidden when the slider is set to menu:True[** .

DN

Bind DN of user that can do user searches

Password

Password of the user

User Search

Definition of the user search parameters

Username Prompt

Label of LDAP attribute users will enter to identify themselves (e.g. `username`)

Base DN

BaseDN where users are located (e.g. `cn=users,dc=example,dc=com`)

Filter
: Filter to specify type of user objects (e.g. `"(objectClass=person)"`)

User Attribute Map

Definition of the user attribute map

Username
: Attribute users will enter to identify themselves

ID
: Attribute used to identify user within the system (e.g. `uid`)

Email
: Attribute containing email of users

Name
: Attribute used as username used within OIDC tokens

Group Search

Definition of group search parameters

Base DN
: BaseDN where groups are located (e.g. `cn=users,dc=example,dc=com`)

Filter
: Filter to specify type of user objects (e.g. `"(objectClass=group)"`)

Group Attribute Map

Definition of group attribute map

User
: Attribute to map as user (e.g. `uid`)

Group
: Attribute identifying membership (e.g. `member`)

Name
: Attribute to map as name (e.g. `name`)

2.9.2. Examples

In both directories, `user-regular1` and `user-regular2` are members of the `k8s-users` group, `user-admin` is a member of the `k8s-admins` group.

For Active Directory, `user-bind` is a simple user which is member of the default `Domain Users` group. Hence, we can use it to authenticate because has read-only access to Active Directory.

The mail attribute is used to create the RBAC rules.

Active Directory

Example 11. Active Directory Content LDIF

```
# user-regular1, Users, example.com
dn: CN=user-regular1,CN=Users,DC=example,DC=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: user-regular1
sn: Regular1
givenName: User
distinguishedName: CN=user-regular1,CN=Users,DC=example,DC=com
displayName: User Regular1
memberOf: CN=Domain Users,CN=Users,DC=example,DC=com
memberOf: CN=k8s-users,CN=Groups,DC=example,DC=com
name: user-regular1
sAMAccountName: user-regular1
objectCategory:
CN=Person,CN=Schema,CN=Configuration,DC=example,DC=com
mail: user-regular1@example.com

# user-regular2, Users, example.com
dn: CN=user-regular2,CN=Users,DC=example,DC=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: user-regular2
sn: Regular2
givenName: User
distinguishedName: CN=user-regular2,CN=Users,DC=example,DC=com
displayName: User Regular2
memberOf: CN=Domain Users,CN=Users,DC=example,DC=com
memberOf: CN=k8s-users,CN=Groups,DC=example,DC=com
name: user-regular2
sAMAccountName: user-regular2
```

```
objectCategory:
CN=Person,CN=Schema,CN=Configuration,DC=example,DC=com
mail: user-regular2@example.com

# user-bind, Users, example.com
dn: CN=user-bind,CN=Users,DC=example,DC=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: user-bind
sn: Bind
givenName: User
distinguishedName: CN=user-bind,CN=Users,DC=example,DC=com
displayName: User Bind
memberOf: CN=Domain Users,CN=Users,DC=example,DC=com
name: user-bind
sAMAccountName: user-bind
objectCategory:
CN=Person,CN=Schema,CN=Configuration,DC=example,DC=com
mail: user-bind@example.com

# user-admin, Users, example.com
dn: CN=user-admin,CN=Users,DC=example,DC=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: user-admin
sn:: Admin
givenName: User
distinguishedName: CN=user-admin,CN=Users,DC=example,DC=com
displayName: User Admin
memberOf: CN=Domain Users,CN=Users,DC=example,DC=com
memberOf: CN=k8s-admins,CN=Groups,DC=example,DC=com
name: user-admin
sAMAccountName: user-admin
objectCategory:
CN=Person,CN=Schema,CN=Configuration,DC=example,DC=com
mail: user-admin@example.com

# k8s-users, Groups, example.com
dn: CN=k8s-users,CN=Groups,DC=example,DC=com
objectClass: top
objectClass: group
cn: k8s-users
member: CN=user-regular1,CN=Users,DC=example,DC=com
member: CN=user-regular2,CN=Users,DC=example,DC=com
distinguishedName: CN=k8s-users,CN=Groups,DC=example,DC=com
name: k8s-users
sAMAccountName: k8s-users
objectCategory:
CN=Group,CN=Schema,CN=Configuration,DC=example,DC=com
```

```
# k8s-admins, Groups, example.com
dn: CN=k8s-admins,CN=Groups,DC=example,DC=com
objectClass: top
objectClass: group
cn: k8s-admins
member: CN=user-admin,CN=Users,DC=example,DC=com
distinguishedName: CN=k8s-admins,CN=Groups,DC=example,DC=com
name: k8s-admins
sAMAccountName: k8s-admins
objectCategory:
CN=Group,CN=Schema,CN=Configuration,DC=example,DC=com
```

Example 12. Active Directory LDAP Connector (YAML)

```
# Server
Host: domain-controler.example.com
Port: 636
StartTLS: Off

Certificate: DC_Trust_Root.crt

# Authentication
Anonymous: False
DN: user-bind@example.com
Password: <password>

# User search
Identifying User Attribute: sAMAccountName
Base DN: CN=Users,DC=example,DC=com
Filter: (objectClass=person)

# User Attribute Map
Username: sAMAccountName
ID: distinguishedName
Email: mail
Name: sAMAccountName

# Group Search
Base DN: CN=Groups,DC=example,DC=com
Filter: (objectClass=group)

# Group Attribute Map
User: distinguishedName
Group: member
Name: sAMAccountName
```

openLDAP

Example 13. openLDAP Content LDIF

```
# user-regular1, accounts, example.com
dn: CN=user-regular1,OU=accounts,DC=example,DC=com
cn: User Regular1
uidNumber: 1200
gidNumber: 500
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
uid: user-regular1
mail: user-regular1@example.com
sn: Regular1
givenName: User

# user-regular2, accounts, example.com
dn: CN=user-regular2,OU=accounts,DC=example,DC=com
cn: User Regular2
uidNumber: 1300
gidNumber: 500
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
uid: user-regular2
mail: user-regular2@example.com
sn: Regular2
givenName: User

# user-admin, accounts, example.com
dn: CN=user-admin,OU=accounts,DC=example,DC=com
cn: User Admin
uidNumber: 1000
gidNumber: 100
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
uid: user-admin
mail: user-admin@example.com
sn: Admin
givenName: User

# k8s-users, accounts, example.com
dn: CN=k8s-users,OU=accounts,DC=example,DC=com
gidNumber: 500
objectClass: posixGroup
cn: k8s-users
memberUid: user-regular1
memberUid: user-regular2

# k8s-admins, accounts, example.com
dn: CN=k8s-admins,OU=accounts,DC=example,DC=com
gidNumber: 100
objectClass: posixGroup
cn: k8s-admins
memberUid: user-admin
```

Example 14. External LDAP Connector without Authentication

```
# Server
Host: ldap.example.com
Port: 636
StartTLS: Off

Certificate: LDAP_Trust_Root.crt

# Authentication
Anonymous: True

# User search
Identifying User Attribute: uid
Base DN: OU=accounts,DC=example,DC=com
Filter: (objectClass=person)

# User Attribute Map
Username: uid
ID: uid
Email: mail
Name: uid

# Group Search
Base DN: OU=accounts,DC=example,DC=com
Filter: (objectClass=posixGroup)

# Group Attribute Map
User: uid
Group: memberUid
Name: cn
```

Chapter 3. Cluster Management

3.1. Interacting With Kubernetes

Kubernetes requires the use of `kubectl` for many tasks. You can perform most of these actions while logged in to an SSH session on the master node of your SUSE CaaS Platform cluster. `kubectl` is a pre-installed component of SUSE CaaS Platform .

The proxy functionality requires `kubectl` to be installed on your local machine to act as a proxy between the local workstation and the remote cluster.

SUSE Linux EnterpriseDesktop 12 SP3 / 15.0 - Installation from Packagehub

The use of PackageHub is exempt from commercial support.

If you are using SUSE Linux Enterprise 12 SP3 or 15.0, you must enable the PackageHub Extension.

The instructions are identical for both versions.

Installing `kubectl` on Non-SUSE OS or Old Release

If you are using an operating system other than the current SUSE Linux Enterprise 12 SP3/15.0 or openSUSE Tumbleweed/Leap please consult the installation instructions from the Kubernetes project.

The KUBECONFIG Variable

<command xmlns='http://docbook.org/ns/docbook'>kubectl</command> uses an environment variable named `KUBECONFIG` to locate your <filename xmlns='http://docbook.org/ns/docbook'>kubeconfig</filename > file. If this variable is not specified, it defaults to `$HOME/.kube/config` . To use a different location, run

```
{prompt.user}``export
KUBECONFIG=/PATH/TO/KUBE/CONFIG/FILE``
```

Procedure: Install the `kubectl` package

1. Install the `kubectl` package:

   ```
   {prompt.sudo}``zypper in kubectl``
   ```

2. To use kubectl to connect to a local machine you must perform Access Control Overview against the Kubernetes master node. Download the `.kubeconfig` file from Velum and place it in `~/.kube/config` .

[velum status] | *velum_status.png*

1. Verify that `kubectl` was installed and is configured correctly:

   ```
   {prompt.user}``kubectl get nodes`` NAME                  STATUS
   ROLES     AGE       VERSION
   caasp3-master     Ready     master    1d        v1.9.8
   caasp3-worker-1   Ready     <none>    1d        v1.9.8
   caasp3-worker-2   Ready     <none>    1d        v1.9.8
   caasp3-worker-3   Ready     <none>    1d        v1.9.8
   caasp3-worker-4   Ready     <none>    1d        v1.9.8
   ```

   You should see the list of nodes known to SUSE CaaS Platform .

3.2. Interacting with Salt

You can run commands across all nodes in the cluster by running them via `salt`.

Log in to the admin node and run:

```
{prompt.user}``docker exec -it $(docker ps -q -f name="salt-master") \
salt -P 'roles:(admin|kube-(master|minion)' \
cmd.run "df -h"``
```

This command tells `docker` to find the `salt-master` container and execute the command on all nodes that match the roles `admin`, `kube-master`, and `kube-minion` (which is all nodes).

Replace the example `df -h` with a command of your choice. The output will be produced in your current terminal session.

3.2.1. Adjusting The Number Of Salt Worker Threads

It will sometimes be necessary to resize the Kubernetes cluster to adjust for workloads or other factors. Salt will run into problems, if the number of nodes to handle becomes too large without adjusting the number of available Salt worker threads.

For the correct value, refer to [_sec.deploy.requirements.system.cluster.salt_cluster_size].

Procedure: Adjust The Salt Worker Count

1. Log in to your admin node via SSH.
2. Run the following command to adjust the configured number of workers (here: `20`).

```
{prompt.root.admin}``echo "worker_threads:20" > /etc/salt/salt-
master-custom.conf``
```

3. Find the ID of the Salt master container.

```
{prompt.root.admin}``saltid=$(docker ps -q -f salt-master)``
```

4. And restart the Salt master .

```
{prompt.root.admin}``docker kill $saltid``
```

Now, Salt will restart and adjust the number of workers in the cluster.

3.3. Node Management

After you complete the deployment and you bootstrap the cluster, you may need to perform additional changes to the cluster. By using Velum you can add additional nodes to the cluster. You can also delete some nodes, but in that case make sure that you do not break the cluster.

3.3.1. Adding Nodes

You may need to add additional Worker Node s to your cluster. The following steps guides you through that procedure:

Procedure: Adding Nodes to Existing Cluster

1. Prepare the node as described in [_sec.deploy.nodes.worker_install]
2. Open Velum in your browser and login.
3. You should see the newly added node as a node to be accepted in **Pending Nodes ›] . Click on menu:Accept Node[** .

[velum pending nodes] | *velum_pending_nodes.png*

1. In the **Summary ›] you can see the menu:New[** that appears next to **New nodes ›] . Click the menu:New[** button.

[velum unassigned nodes] | *velum_unassigned_nodes.png*

1. Select the node to be added and click **Add nodes** .
2. The node has been added to your cluster.

The `create_autoyast_profile` Command

The `create_autoyast_profile` command creates an autoyast profile for fully automatic installation of SUSE CaaS Platform . You can use the following options when invoking the command:

`-o|--output`

Specify to which file the command should save the created profile.

```
{prompt.root}``create_autoyast_profile -o FILENAME``
```

`--salt-master`

Specify the host name of the Salt master .

```
{prompt.root}``create_autoyast_profile --salt-master SALTMASTER``
```

`--smt-url`

Specify the URL of the SMT server.

```
{prompt.root}``create_autoyast_profile --smt-url SALTMASTER``
```

`--regcode`

Specify the registration code for SUSE CaaS Platform .

```
{prompt.root}``create_autoyast_profile --regcode REGISTRATION_CODE``
```

`--reg-email`
Specify an e-mail address for registration.

```
{prompt.root}``create_autoyast_profile --reg-email E-MAIL_ADRESS``
```

3.3.2. Removing Nodes

If you attempt to remove more nodes than are required for the minimum cluster size (3 nodes: 1 master, 2 workers) Velum will display a warning. Your cluster will be disfunctional until you add the minimum amount of nodes again.

As each node in the cluster runs also an instance of `etcd`, SUSE CaaS Platform has to ensure that removing of several nodes does not break the `etcd` cluster. In case you have, for example, three nodes in the `etcd` and you delete two of them, SUSE CaaS Platform deletes one node, recovers the cluster and only if the recovery is successful, allows the next node to be removed. Refer to: [_sec.deploy.requirements.system.cluster.etcd_cluster_size].

If a node runs just an `etcd-proxy`, there is nothing special that has to be done, as deleting any amount of `etcd-proxy` cannot break the `etcd` cluster.

If you have only one master node configured, Velum will not allow you to remove it. You must first add a second master node as a replacement.

1. Log-in to Velum on your SUSE CaaS Platform Admin node. Then, click **Remove** next to the node you wish to remove. A dialog will ask you to confirm the removal.

[velum status] | *velum_status.png*

1. The cluster will then attempt to remove the node in a controlled manner. Progress is indicated by a spinning icon and the words `Pending removal` in the location where the **Remove** -button was displayed before.

[velum pending removal] | *velum_pending_removal.png*

+ This should conclude the regular removal process. If the node is successfully removed, it will disappear from the list after a few moments. . In some cases nodes cannot be removed in a controlled manner and must be forced out of the cluster. A typical scenario is a machine instance was removed externally or has no connectivity. In such cases, the removal will fail. You then get the option to **Force remove** . A dialog will ask you to confirm the removal.

+

[velum failed removal] | *velum_failed_removal.png*

+ Additionally, a large warning dialog will ask you to confirm the forced removal. Click **Proceed with forcible removal** if you are sure you wish to force the node out of the cluster.

+

[velum force removal] | *velum_force_removal.png*

3.3.3. Removing Unassigned nodes

You might run into the situation where you have (accidentally) added new nodes to a cluster but did not wish to bootstrap them. They are now registered against the cluster and show up in "Unassigned nodes". You might have already configured the machine to register with another cluster and want to clean out this entry from the "Unassigned Nodes" view. You must perform the following steps:

1. Find the "Unassigned nodes" line in the overview and click on menu:(new)[] next to the count number. You will be shown the "Unassigned Nodes" view where all the unassigned nodes are listed. Make sure that you first assign all roles to nodes that you wish to keep and proceed with bootstrapping. Once the list only show the nodes you are sure to remove copy the ID of the node you wish to drop.

[velum unassigned nodes] | *velum_unassigned_nodes.png*

1. Log into the Admin node of you cluster via SSH.
2. Run the following command and replace `$ID_FROM_UNASSIGNED_QUEUE` with the node ID that you copied from the "Unassigned nodes" view in Velum .

Make absolutely sure that the node ID you have copied is the one of the node you wish to drop. This command is `irreversible` and will remove the specified node from the cluster without confirmation.

```
{prompt.root}``docker exec -it $(docker ps -q -f name="velum-
dashboard") \
entrypoint.sh bundle exec rails runner 'puts
Minion.find_by(minion_id: "$ID_FROM_UNASSIGNED_QUEUE").destroy'``
```

3.4. Graceful Shutdown and Startup

3.4.1. Overview

Kubernetes , being a self-healing solution, tries to keep all pods and services available. In general, this is of its core features and desired functions. But it is important to take this into account if you are doing a complete shutdown of the infrastructure.

There are two ways of shutting down the whole cluster: Shut down and start all nodes at once or restart them sequentially in segments. In both cases, SUSE CaaS Platform expects that IP addresses do not change after the restart, even when using dynamic IP addresses.

When restarting segments of nodes, it is possible to avoid downtime.

Deviating from Shutdown and Startup Procedures

The procedures described in this section are recommended to reduce logged errors. However, it is possible to not follow this order as long as all nodes are stopped in a graceful way.

3.4.2. Node Types

For shutting down and starting nodes, three different types of nodes are relevant:

- The Administration Node contains state and needs to be shut down in a graceful way to ensure that all state has been synced to disk in a clean way.
- Nodes with `etcd` contain state and also need to be shut down in a graceful

way. They will usually be a subset of the master nodes. But it can happen that some workers run `etcd` members.

- The rest (masters and workers not running `etcd` members): These nodes contain local state possibly created by applications running on top of the cluster. They need to be shut down in a graceful way too, when possible.

3.4.3. Complete Shutdown

Shutting Down

All commands are executed on the admin node.

1. Disable scheduling on the whole cluster. This will avoid Kubernetes rescheduling jobs while you are shutting down nodes.

```
{prompt.root.admin}``kubectl get nodes -o name | xargs -I{} kubectl
cordon {}``
```

2. Gracefully shut down all worker nodes.

```
{prompt.root.admin}``docker exec -it $(docker ps -q -f name="salt-
master") \
salt --async -G 'roles:kube-minion' cmd.run 'systemctl poweroff'``
```

3. Gracefully shut down all master nodes.

```
{prompt.root.admin}``docker exec -it $(docker ps -q -f name="salt-
master") \
salt --async -G 'roles:kube-master' cmd.run 'systemctl poweroff'``
```

4. Shut down the Administration Node :

```
{prompt.root.admin}``systemctl poweroff``
```

Starting Up

`kubectl` *Needs Master Nodes To Function*

`kubectl` requires use of the Kubernetes API hosted on the master nodes. Therefore, until at least some of the master nodes have started successfully, you will see error messages of the type `HTTP 503`.

```
Error from server (InternalError): an error on the server
("<html><body><h1>503 Service Unavailable</h1>\nNo server
is available
to handle this request.\n</body></html>") has prevented
the request
from succeeding (get nodes)
```

1. Start the Administration Node up. All commands are executed on the Administration Node .
2. Once that the admin node is up, start the master nodes. Keep checking the status of the master nodes. Continue as soon as all master nodes are `Ready`.

```
{prompt.root.admin}``kubectl get nodes`` NAME       STATUS
ROLES     AGE       VERSION
master-0   Ready,SchedulingDisabled      master    21h       v1.9.8
master-1   Ready,SchedulingDisabled      master    21h       v1.9.8
master-2   Ready,SchedulingDisabled      master    21h       v1.9.8
worker-0   NotReady,SchedulingDisabled   <none>    21h       v1.9.8
worker-1   NotReady,SchedulingDisabled   <none>    21h       v1.9.8
worker-2   NotReady,SchedulingDisabled   <none>    21h       v1.9.8
worker-3   NotReady,SchedulingDisabled   <none>    21h       v1.9.8
worker-4   NotReady,SchedulingDisabled   <none>    21h       v1.9.8
```

3. Continue by starting all the worker nodes. Keep checking the status of the nodes. Continue when all nodes are `Ready`.

```
{prompt.root.admin}``kubectl get nodes`` NAME       STATUS
ROLES     AGE       VERSION
master-0   Ready,SchedulingDisabled   master    21h       v1.9.8
master-1   Ready,SchedulingDisabled   master    21h       v1.9.8
master-2   Ready,SchedulingDisabled   master    21h       v1.9.8
worker-0   Ready,SchedulingDisabled   <none>    21h       v1.9.8
worker-1   Ready,SchedulingDisabled   <none>    21h       v1.9.8
worker-2   Ready,SchedulingDisabled   <none>    21h       v1.9.8
worker-3   Ready,SchedulingDisabled   <none>    21h       v1.9.8
worker-4   Ready,SchedulingDisabled   <none>    21h       v1.9.8
```

4. Uncordon all nodes so they can receive new workloads:

```
{prompt.root.admin}``kubectl get nodes -o name | xargs -I{} kubectl
uncordon {}``
```

3.4.4. Segmented Reboots

A sequential reboot of cluster segments is a way to completely avoid the downtime of services or at least reduce it as much as possible. However, downtime of services occurs if all pods of a service are forced on one node.

Rebooting Worker Nodes

The number of worker nodes to reboot at once depends on the number of total worker nodes and their labels.

For example: If there are 5 worker nodes with 2 of them having the label `diskType: ssd`, then the two nodes with SSDs must not be shut down at the same time.

The size of segments for simultaneous reboots depends on the topology of the cluster and the workload. We recommend to use small segment sizes. This makes it less likely that all nodes running replicas of the same pod are shut down at the same time.

During this migration time, the worker nodes need to be able to reach the master nodes at all times. This includes master nodes that are already or not yet updated.

Rebooting Master Nodes

Master nodes should not run user workloads. This means that the decision to batch the reboots of master nodes depends on whether you want to keep control of the cluster while the reboot is taking place.

If all the master nodes disappear at the same time, the worker nodes continue serving the services they are running. No further operation will take place on the worker nodes, since they cannot contact an `apiserver` to discover new workloads or perform any other operations.

It is safe to choose batches as desired. Rebooting one by one is the safest, two by two is generally safe too. For larger batches than two, certain cluster services, for example `dex`, could be completely shut down.

3.4.5. Behavior of `etcd`

`etcd` requires special considerations for maintaining cluster health and integrtiy. Refer to: [_sec.deploy.requirements.system.cluster.etcd_cluster_size].

3.5. Scaling the Cluster

The default maximum number of nodes in a cluster is 40. The Salt Master configuration needs to be adjusted to handle installation and updating a of larger cluster:

Table 1. Node Count and Salt Worker Threads

Nodes	Salt Worker Threads
>40	20
>60	30
>75	40
>85	50
>95	60

To change the variable in the Salt master configuration, run the following on the Administration Node :

```
{prompt.root}``echo "worker_threads: 20" > /etc/caasp/salt-master-
custom.conf`` {prompt.root}``docker restart $(docker ps -q -f name="salt-
master")``
```

Salt master will be automatically restarted by kubelet.

Following bootstrapping failure, you can check if Salt worker_threads is too low.

```
{prompt.root}``docker logs $(docker ps -q -f name="salt-master") \
    | grep -i worker_threads``
```

3.6. Configuring Remote Container Registry

A remote registry allows you to access container images locally. This is commonly used in cases where a SUSE CaaS Platform cluster is not allowed to have direct access to the internet. You can create a local registry with the images that you will need and add the information for that registry here. If the registry is using a self-signed certificate, it can be added here to create trust between Kubernetes and the registry.

By default, Docker Hub and the SUSE container registry are available sources for container images.

[velum settings registry overview] | *velum_settings_registry_overview.png*

3.6.1. Adding A Remote Registry

1. Log in to Velum and navigate to **Settings → Remote Registries** .
2. Click on **Add Remote Registry** to add a new remote registry configuration.

[velum settings remote registry] | *velum_settings_remote_registry.png*

1. Fill in the options for the new registry.

[velum settings new registry] | *velum_settings_new_registry.png*

+

Name
: Define a name for the registry.

URL
: Enter the URL for the registry in the format `http(s)://<hostname>:<port>`.

Certificate
: Will only be shown if the `URL` field contains `https:`.

  Provide the body of the (self-signed) SSL certificate for the registry. . You will be shown a summary of the details of the registry you have just created.

  + If you have to adjust the registry click **Edit** to return to the editing dialog.

  + Click **Delete** if you made a mistake and wish to remove the registry. You can always remove the registry from the overview later.

+ If you wish to define a mirror for this registry you can click on **Add Mirror** to do so. For details, refer to Configuring A Registry Mirror

[velum settings registry details] | *velum_settings_registry_details.png*

1. If you have further registries to add, repeat the previous steps.
2. Finally, click the **Apply Changes** button on the top of the page. This will update the registry settings across the cluster.

3.6.2. Modifying A Registry

1. Log in to Velum and navigate to **Settings → Remote Registries** .
2. Click on the pencil icon in the row of the registry you wish to modify. Perform the changes you wish to make and click "Save".
3. If you have further registries to modify, repeat the previous steps.
4. Finally, click the **Apply Changes** button on the top of the page. This will update the registry settings across the cluster.

3.6.3. Removing A Registry

1. Log in to Velum and navigate to **Settings → Remote Registries** .
2. Click on the red trashcan icon in the row of the registry you wish to delete and confirm the popup dialog by clicking **OK** .
3. If you have further registries to remove, repeat the previous steps.
4. Finally, click the **Apply Changes** button on the top of the page. This will update the registry settings across the cluster.

3.7. Configuring A Registry Mirror

Similar to the **Remote Registries ›] page › the menu:Mirrors[** page allows you to add redundant image mirrors to existing registries. The internal container engine will use this information to reroute requests from the cluster nodes to the defined mirror address.

[velum settings mirror overview] | *velum_settings_mirror_overview.png*

3.7.1. Adding A Mirror

1. Log in to Velum and navigate to **Settings → Mirrors** .
2. Click on **Add Mirror** to add a new registry mirror configuration.

[velum settings mirror] | *velum_settings_mirror.png*

1. Fill in the options for the new mirror.

[velum settings new mirror] | *velum_settings_new_mirror.png*

+

Mirror of
: Select one of the configured registries from the menu.

Name
: Define a name for the mirror.

URL
: Enter the URL for the mirror in the format `http(s)://<hostname>:<port>`.

Certificate
: Will only be shown if the `URL` field contains `https:`.

  Provide the body of the (self-signed) SSL certificate for the registry. .

[velum settings mirror details] | *velum_settings_mirror_details.png*

3.7.2. Modifying A Mirror

1. Log in to Velum and navigate to **Settings → Mirrors** .
2. Click on the pencil icon in the row of the mirror you wish to modify. Perform the changes you wish to make and click "Save".
3. If you have further mirrors to modify, repeat the previous steps.
4. Finally, click the **Apply Changes** button on the top of the page. This will update the mirror settings across the cluster.

3.7.3. Removing A Mirror

1. Log in to Velum and navigate to **Settings → Mirrors** .

2. Click on the trashcan icon in the row of the mirror you wish to remove and confirm the popup dialog with **OK** .
3. If you have further mirrors to remove, repeat the previous steps.
4. Finally, click the **Apply Changes** button on the top of the page. This will update the mirror settings across the cluster.

3.8. Reserving Compute Resources

By default, Kubernetes will allocate all available hardware resources of a node to pods. This can starve core services of needed resources, which are, for example, required for managing single nodes or the cluster. To prevent core services from running out of resources, you can reserve CPU, memory, and disk resources for them.

Carefully Check Entered Values

Entering invalid values into the input fields may break nodes. Carefully check the entered values before selecting the **Save** button.

[velum settings compute resource] | *velum_settings_compute_resource.png*

To reserve hardware resources, go to the Velum dashboard and then proceed to *Settings* and *Compute Resources Reservation*.

You can reserve resources for Kubernetes services in the box `Kubernetes core services` and for services running on a single node in `Host system services`.

In the box `Eviction threshold`, you can set rules for killing pods when the usage of RAM or storage reaches a defined level. This prevents nodes from actually running out of resources, which would then trigger the default out-of-resource-handling.

To activate entered settings, use the **Save** button at the bottom of the page.

Chapter 4. Software Management

4.1. Transactional Updates

For security and stability reasons, the operating system and application should always be up-to-date. While with a single machine you can keep the system up-to-date quite easily by running several commands, in a large-scale cluster the update process can become a real burden. Thus transactional automatic updates have been introduced. Transactional updates can be characterized as follows:

- They are atomic.
- They do not influence the running system.
- They can be rolled back.
- The system needs to be rebooted to activate the changes.

Transactional updates are managed by the `transactional-update` script, which is called once a day. The script checks if any updates are available. If there are any updates to be applied, a new snapshot of the root file system is created in the background and is updated from the release channels. All updates released to this point are applied. The running file system/machine state is left untouched.

The new snapshot, once completely updated, is then marked as active and will be used as the new default after the next reboot of the system.

Snapshot Activation

For each "transaction" performed by `transactional-updates` a new snapshot is generated and requires a reboot to incorporate the changes. If another transaction is run before the reboot, only the latest snapshot is used and changes might be lost. If you perform manual updates or installation on your nodes, please make sure to reboot after each transaction. Refer to: Applying Updates Manually.

Velum will show a list of nodes that have new updates available for use. The cluster administrator then uses Velum to reboot the nodes to the new snapshots to ensure the health of services and configuration. Velum uses `salt` to safely disable services on the nodes, apply new snapshots, rewrite configurations and then bring the services and nodes back up.

4.1.1. The `transactional-update` Command

Only use when requested by SUSE Support

This reference for `transactional-update` should only be used when requested by SUSE Support. Updates are handled by an automated process and only require user interaction for rebooting of nodes.

The `transactional-update` enables you to install or remove updates of your system in an atomic way. The updates are applied all or none of them if any package cannot be installed. Before the update is applied, a snapshot of the system is created in order to restore the previous state in case of a failure.

If the current root file system is identical to the active root file system (after applying updates and reboot), run cleanup of all old snapshots:

```
{prompt.root}``transactional-update cleanup``
```

Other options of the command are the following:

`pkg in/install`

Installs individual packages from the available channels using the `zypper install` command. This command can also be used to install PTF RPM files. Please note that the changes to the base file system only become permanent after a reboot.

```
{prompt.root}``transactional-update pkg install PACKAGE_NAME``
```

or

```
{prompt.root}``transactional-update pkg install RPM1 RPM2``
```

`pkg rm/remove`

Removes individual packages from the active snapshot using the `zypper remove` command. This command can also be used to remove PTF RPM files. Please note that the changes to the base file system only become permanent after a reboot.

```
{prompt.root}``transactional-update pkg remove PACKAGE_NAME``
```

`pkg up/update`

Updates individual packages from the active snapshot using the `zypper update` command. This command can also be used to update PTF RPM files. Please note that only packages that are part of the snapshot of the base file system can be updated and changes only become permanent after a reboot.

```
{prompt.root}``transactional-update pkg remove PACKAGE_NAME``
```

`up/update`

If there are new updates available, a new snapshot is created and `zypper up/update` is used to update the snapshot. The snapshot is activated afterwards and is used as the new root file system after reboot.

```
{prompt.root}``transactional-update up``
```

`dup`

If there are new updates available, a new snapshot is created and `zypper dup –no-allow-vendor-change` is used to update the snapshot. The snapshot is activated afterwards and is used as the new root file system after reboot.

```
{prompt.root}``transactional-update dup``
```

`patch`

If there are new updates available, a new snapshot is created and `zypper patch` is used to update the snapshot. The snapshot is activated afterwards and is used as the new root file system after reboot.

```
{prompt.root}``transactional-update patch``
```

`rollback`

The command sets the default sub volume. On systems with read-write file system `snapper rollback` is called. On a read-only file system and without any argument, the current system is set to a new default root file system. If you specify a number, that snapshot is used as the default root file system. On a read-only file system, no additional snapshots are created.

```
{prompt.root}``transactional-update rollback SNAPSHOT_NUMBER``
```

`grub.cfg`

The command creates a new grub2 config. Sometimes it is necessary to adjust the boot configuration, e.g. by adding additional kernel parameters. This can be done by editing `/etc/default/grub`, calling `transactional-update grub.cfg` and then rebooting the machine to activate the change. Please note that without rebooting the machine, the new grub config will be overwritten with the default by any transactional-update that takes place.

```
{prompt.root}``transactional-update grub.cfg``
```

`reboot`

This parameter triggers a reboot after the action is completed.

How the reboot is done depends on how `transactional-update` is configured. For cluster nodes this will set a Salt grain to show the updated node in Velum as requiring reboot.

```
{prompt.root}``transactional-update dup reboot``
```

`--help`

The option outputs possible options and subcommands.

```
{prompt.root}``transactional-update --help``
```

4.1.2. Disabling Transactional Updates

Even though it is not recommended, you can disable transactional updates by issuing the command:

```
{prompt.root}``systemctl --now disable transactional-update.timer``
```

Disabling transaction update timer is required during upgrade

You must disable transactional updates during the upgrade procedure from one version of SUSE CaaS Platform to the next.

4.1.3. Applying Updates

It is paramount that you never "hard reboot" nodes in the cluster after transactional updates. This will omit reconfiguring services and applications and will leave nodes in unhealthy, if not unsusable, states.

Updates are typically applied to nodes automatically and will be flagged in Velum for reboot. If you have nodes with pending transactional updates follow the steps below.

General Notes to the Updates Installation

Only packages that are part of the snapshot of the root file system can be updated. If packages contain files that are not part of the snapshot, the update could fail or break the system.

RPMs that require a license to be accepted cannot be updated.

After the `transactional-update` script has run on all nodes, Velum displays any nodes in your cluster running outdated software. The updates are only applied after a reboot. For this purpose, Velum enables you to update your cluster directly. Follow the next procedure to update your cluster.

Procedure: Updating the Cluster with Velum

1. Login to Velum .
2. If required, click **UPDATE ADMIN NODE** to start the update.

[velum updating] | *velum_updating.png*

1. Confirm the update by clicking **Reboot to update** .

[velum reboot and update] | *velum_reboot_and_update.png*

1. Now you have to wait until the Administration Node reboots and Velum is available again.
2. Click **update all nodes** to update Master Node and Worker Node s.

[velum update nodes] | *velum_update_nodes.png*

Applying Updates Manually

You can use `transactional-update` to apply updates or install PTF files manually.

```
{prompt.root}``transactional-update pkg install PACKAGE_NAME reboot``
```

If your node is accepted to the cluster, it will have been configured to use Salt orchestration to reboot. The updated node will show in Velum requiring a reboot.

If your node is not (yet) accepted into the cluster it will reboot after the transactional-update has finished.

4.1.4. Recovering From Failed Updates

Velum notifies you about failed updates. If the update failed, there are several things that can be the cause. The following list provides an overview of things to check. For general information about troubleshooting, read Overview.

Do Not Interfere with Transactional Updates

Do not manually interfere with transactional updates. Do so only if you are requested to do so by SUSE support.

For details, see The `transactional-update` Command.

Stopping Services and Reboot

Velum uses Salt to stop all services and reboot the node. Salt also takes care of adjusting configuration. Check the logs of the Salt master and minions for error messages. For details, see Salt Master Log and Salt Minion Logs.

Installing Updates

Updates are installed once a day but only applied after a reboot is manually triggered. If the installation of updates fails, Velum shows the message `Update Failed` as the node's status. In this case, log in on the node and check `/var/log/transactional-update.log` for problems.

Starting Services

Finally, all services of the node are being restarted. Look which services have failed by executing `systemctl list-units --failed`. Then check the logs of failed services.

The following procedure can help in some situations.

1. Reboot all nodes.

```
{prompt.root.admin}``docker exec -it $(docker ps -q -f name="salt-
master") \
salt -P "roles:(admin|kube-(master|minion))" system.reboot``
```

2. On the Administration Node run

```
{prompt.root.admin}``docker exec -it $(docker ps -q -f name="salt-
master") \
 salt -P "roles:(admin|kube-(master|minion))" cmd.run "transactional-
update cleanup reboot dup"``
```

3. Reboot all nodes again.

```
{prompt.root.admin}``docker exec -it $(docker ps -q -f name="salt-
master") \
salt -P "roles:(admin|kube-(master|minion))" system.reboot``
```

4. Start the update with debug output.

```
{prompt.root.admin}``docker exec -it $(docker ps -q -f name="salt-
master") \
salt-run -l debug state.orchestrate orch.update``
```

5. If there is any ongoing problem, look at all the Salt grains of all nodes in `/etc/salt/grains` . This file contains the status if the update is ongoing, and is therefore providing the "Update Retry" in Velum.

4.2. Program Temporary Fixes

Program temporary fixes (PTFs) are available in the SUSE CaaS Platform environment. You install them by using the `transactional-update` script. Typically you invoke the installation of PTFs by running:

```
{prompt.root}``transactional-update reboot ptf install RPM1 RPM2 …``
```

The command installs PTF RPMs. The `reboot` option then schedules a reboot after the installation. PTFs are activate only after rebooting of your system.

Reboot Required

If you install or remove PTFs and you call the `transactional-update` to update the system before reboot, the applied changes by PTFs are lost and need to be done again after reboot.

In case you need to remove the installed PTFs, use the following command:

```
{prompt.root}``transactional-update reboot ptf remove RPM1 RPM2 …``
```

4.3. Upgrading From SUSE CaaS Platform 2

Read This Section Carefully

Before executing the single steps of the upgrade procedure, carefully read all information in this overview section.

As SUSE CaaS Platform is constantly developed and improved, new versions get released. You are strongly advised to upgrade to a supported release. These upgrades may involve manual intervention.

Procedure: Overview of Upgrade Procedure

1. Plan a maintenance window. Upgrades may take some time, during which services may be degraded in performance or completely unavailable.
2. If you are using *Repository Mirroring Tool* or *Subscription Management Tool*, enable the SUSE CaaS Platform 3 repositories and mirror the packages.
3. Install all updates for SUSE CaaS Platform 2. For details, see Install SUSE CaaS Platform 2 Updates
4. Disable automatic updates during the upgrade procedure. For details, see Disable Automatic Updates.
5. Upgrade the nodes. For details, refer to Upgrading to SUSE CaaS Platform

3.

6. Reboot all nodes. For details, refer to Reboot Cluster Nodes.

4.3.1. Install SUSE CaaS Platform 2 Updates

Before you start the upgrade procedure to SUSE CaaS Platform v3, you must ensure that all your nodes are running on the latest v2 updates. You can check the `SUSEConnect` package version to see if you are up to date. To do so you will run a `salt` command to display the package version installed on each node.

```
{prompt.user}``docker exec -i  $(docker ps -q -f name="salt-master") \
salt --batch 10 -P "roles:(admin|kube-(master|minion))" \
cmd.run "rpm -q SUSEConnect"`` Executing run on
['12cda3c374144d74804298bdee4d686c',
                  '9b6d8d28393045c0914c959d0a5c0e33',
                  '73b92dd7816147058c3d0fbb67fb18f9',
                  'admin']
admin:
    SUSEConnect-0.3.11-3.15.1.x86_64
jid:
    20180809103558881056
retcode:
    0
73b92dd7816147058c3d0fbb67fb18f9:
    SUSEConnect-0.3.11-3.15.1.x86_64
jid:
    20180809103558881056
retcode:
    0
9b6d8d28393045c0914c959d0a5c0e33:
    SUSEConnect-0.3.11-3.15.1.x86_64
jid:
    20180809103558881056
retcode:
    0
12cda3c374144d74804298bdee4d686c:
    SUSEConnect-0.3.11-3.15.1.x86_64
jid:
    20180809103558881056
retcode:
    0
```

If the package version is `0.3.11-3.15.1` (or higher) you have the latest updates from the v2 channel installed.

4.3.2. Disable Automatic Updates

To begin with the upgrade procedure, you first must disable the automatic transactional update mechanism to avoid conflicts. To do so you must run a `salt` command across the nodes to disable the `transactional-update.timer`.

The automatic update timer will be re-enabled automatically after the migration procedure.

```
{prompt.user}``docker exec -i $(docker ps -q -f name="salt-master") \
salt --batch 10 -P "roles:(admin|kube-(master|minion))" \
cmd.run "systemctl disable --now transactional-update.timer"`` Executing
run on ['5f6688bbeac94d2ab5c4330dc7043fb2',
                  'c3afd049edbe43afb4e2e5913a88291b',
                  '5bf346291a18406290886c2e2f7c3e3f',
                  'admin']

5bf346291a18406290886c2e2f7c3e3f:
    Removed symlink
/etc/systemd/system/timers.target.wants/transactional-update.timer.
jid:
    20180807122220543037
retcode:
    0
admin:
    Removed symlink
/etc/systemd/system/timers.target.wants/transactional-update.timer.
jid:
    20180807122220543037
retcode:
    0
c3afd049edbe43afb4e2e5913a88291b:
    Removed symlink
/etc/systemd/system/timers.target.wants/transactional-update.timer.
jid:
    20180807122220543037
retcode:
    0
5f6688bbeac94d2ab5c4330dc7043fb2:
    Removed symlink
/etc/systemd/system/timers.target.wants/transactional-update.timer.
jid:
    20180807122220543037
retcode:
    0
```

4.3.3. Upgrading to SUSE CaaS Platform 3

Run the update command across your nodes.

Batch size for upgrade

In this example we have limited the number of nodes this step will be performed on to `10 nodes` at a time.

This is a precaution to avoid problems on slower network connections. If you are performing this step on a high bandwidth connection (for example from within the same datacenter as the cluster), you can raise the number of nodes by replacing the value for the (`--batch`) parameter. It is highly recommended not to change this setting.

```
{prompt.user}``docker exec -i $(docker ps -q -f name="salt-master") \
salt --batch 10 -P "roles:(admin|kube-(master|minion))" \
cmd.run "transactional-update salt migration -n" \
| tee transactional-update-migration.log`` Executing run on
['5f6688bbeac94d2ab5c4330dc7043fb2',
                  'c3afd049edbe43afb4e2e5913a88291b',
                  '5bf346291a18406290886c2e2f7c3e3f',
                  'admin']

5bf346291a18406290886c2e2f7c3e3f:


    Executing 'zypper --root /tmp/tmp.vbaqUwrLIh --non-interactive
refresh'

    Retrieving repository 'SUSE-CAASP-ALL-Pool' metadata [...done]
    Building repository 'SUSE-CAASP-ALL-Pool' cache [....done]
    Retrieving repository 'SUSE-CAASP-ALL-Updates' metadata [....done]
    Building repository 'SUSE-CAASP-ALL-Updates' cache [....done]
    All repositories have been refreshed.
    Upgrading product SUSE CaaS Platform 3.0 x86_64.

[ SNIP ... ]

    done
jid:
    20180807122253512832
retcode:
    0
```

During the procedure the nodes will be switched to the new release channel for v3, available updates are downloaded and installed, services and applications are reconfigured and brought up in a orderly fashion.

This operation will produce a lot of output for each node. The entire output is mirrored to a log file `transactional-update-migration.log` to the current working directory. This log file can be very helpful should any of the update operations fail.

4.3.4. Reboot Cluster Nodes

To complete the procedure, you must reboot the cluster nodes. To do this properly, use Velum to restart the nodes.

1. Log in to Velum .
2. Update the Admin node as described in Applying Updates.
3. Update the remaining nodes as described in Applying Updates.

4.3.5. Troubleshooting

In case the upgrade fails, please perform the support data collection by running `supportconfig` on the affected nodes. Provide the resulting files including the `transactional-update-migration.log` to SUSE Support.

4.4. Additional Software Installation

Once your cluster is ready, you may want to deploy additional software that is not installed on SUSE CaaS Platform by default. This chapter provides instructions on how to install and configure Helm , the Kubernetes package manager.

4.4.1. Building Kernel Modules

Some vendors will only provide certain kernel drivers or modules as source. In order to use these modules you must build them on the machine they are required on. We provide a `caasp-toolchain` module that includes all necessary tools to **build** kernel modules.

A full list of tools and packages available through the module can be found in the SUSE Customer Center.

Reboot Required For Toolchain

The toolchain module must be enabled through `transactional-update`. Due to the nature of transactional updates, the machine must reboot at least twice. First to activate the module and a second time to start the machine from the new snapshot that incorporates the installed tools, packages, and libraries.

Please plan for maintenance windows when setting up toolchain usage.

Procedure: Enabling `caasp-toolchain` Module

1. Log in to the machine where you wish to use the toolchain
2. Register the `caasp-toolchain` module

   ```
   {prompt.root}``transactional-update reboot register -p caasp-
   toolchain/3.0/x86_64``
   ```

   The machine will reboot to incorporate the module into the read-only file system and start from the new snapshot.

3. *Avoid Reboots By Installing Multiple Packages*

   If you wish to install multiple packages, you should install them all in a single operation. Each time `transactional-update` is run, it creates a new snapshot and discards all previous changes. The changes can only be persisted by starting from the new snapshot through reboot.

   Use `transactional-update` to install the desired packages from the toolchain module

   ```
   {prompt.root}``transactional-update reboot pkg in binutils kernel-
   devel kernel-default-devel kernel-syms kernel-macros``
   ```

   After the operation is finished the machine will reboot and start from the new snapshots with the packages installed.

Procedure: Disabling `caasp-toolchain` Module

After you are done using the toolchain module, you can free up space by uninstalling the tools you no longer need and disable the toolchain module. . Uninstall the packages you no longer need

+

```
{prompt.root}``transactional-update reboot pkg rm binutils kernel-devel
kernel-default-devel kernel-syms kernel-macros``
```

+ The machine will reboot and start from the new snapshot without these packages. . Disable the toolchain module

+

```
{prompt.root}``transactional-update reboot register -d -p caasp-
toolchain/3.0/x86_64``
```

+ The machine will reboot and start from the new snapshot without the module registered.

4.4.2. Deploying Helm and Tiller

Helm has two parts: Helm is the client and Tiller is the server component. Helm runs on your remote workstation that has access to your cluster, and Tiller is installed as a container on SUSE CaaS Platform when you run Velum for the first time. (See [_sec.deploy.nodes.admin_configuration].)

You should match the Helm version with the version of Tiller that is running on your cluster. The Tiller binary cannot report its version, and you need the version that is packaged inside the Tiller container. Run the following command from your workstation to query the logs:

```
{prompt.root}``kubectl logs -l name=tiller --namespace=kube-system | grep
"Starting Tiller"`` [main] 2018/04/04 16:48:27 Starting{tiller}v2.6.1
(tls=false)
```

If the log gets overwritten and loses this information, the following command queries the `rpm` package manager inside the container. This works only on SUSE CaaS Platform /SUSE Cloud Foundry installations:

```
{prompt.root}``kubectl exec -it $(kubectl get pods -n kube-system | awk
'/tiller/{print$1}') \
-n kube-system -- rpm -q helm`` helm-2.6.1-1.6.x86_64
```

If the Linux distribution on your workstation doesn't provide the correct Helm version, or you are using some other platform, see the Helm Quickstart Guide for installation instructions and basic usage examples. Download the matching Helm binary into any directory that is in your PATH on your workstation, such as your `~/bin` directory. Then initialize just the client part:

```
{prompt.user}``helm init --client-only``
```

The Tiller version that ships with SUSE CaaS Platform is supported by SUSE . While SUSE does not provide support for third-party Helm charts, you can easily use them if necessary.

4.4.3. Example: Installing heapster

By default, `tiller` will be installed and you only need to initialize data for it. Use the `--client-only` parameter.

Procedure: Installation of heapster

By default, the chart repository for helm will not be known to the system. You must perform `helm init` to initialize the necessary repository files and then refresh the information using `helm repo update`. After that, you can install `heapster` from the Kubernetes helm charts repository. . (On CaaSP Admin Node) Initialize helm repo data.

+

```
{prompt.root}``helm init --client-only`` Creating /root/.helm/repository
Creating /root/.helm/repository/cache
Creating /root/.helm/repository/local
Creating /root/.helm/plugins
Creating /root/.helm/starters
Creating /root/.helm/repository/repositories.yaml
Adding stable repo with URL: https://kubernetes-
charts.storage.googleapis.com
Adding local repo with URL: http://127.0.0.1:8879/charts
$HELM_HOME has been configured at /root/.helm.
Not installing Tiller due to 'client-only' flag having been set
Happy Helming!
```

1. Install `heapster` from stable/heapster Kubernetes charts repository

```
{prompt.root}``helm install --name heapster-default --namespace=kube
-system stable/heapster \
--version=0.2.7 --set rbac.create=true``
```

2. Verify that `heapster` was deployed successfully.

```
{prompt.root}``helm list | grep heapster`` heapster-default  1  Fri
Jun 29 10:48:45 2018  DEPLOYED  heapster-0.2.7  kube-system
```

4.5. Installing Kubernetes Dashboard

Technology Preview

Even though you can install and use the community Kubernetes dashboard, SUSE CaaS Platform currently fully supports only Velum .

Requirements

- Heapster version 1.3.0 or later needs to be installed on the cluster
- Helm version 2.7.2+ and kubectl version 1.8.0+ recommended

Procedure: Installation of KubernetesDashboard

1. If `heapster` is not installed, refer to Example: Installing heapster.
2.

```
``helm install --namespace=kube-system \
--name=kubernetes-dashboard stable/kubernetes-dashboard \
--version=0.6.1``
```

3. Run `kubectl proxy` to expose the cluster on your local workstation.
4. Visit `http://127.0.0.1:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/` in your browser. You will be greeted with by a welcome page containing a dialog to configure authentication.
5. Select **token** authentication. To retrieve your token refer to the value in your kubeconfig file by running the command:

```
``grep "id-token" /path/to/kubeconfig  | awk '{print $2}'``
```

6. On login cluster resources and basic metrics are populated.

Procedure: Exposing the Dashboard

1.
```
``helm upgrade kubernetes-dashboard stable/kubernetes-dashboard --set
service.type=NodePort``
```

2. Now you may visit the dashboard at `https://WORKER_NODE_ADDRESS:NODE_PORT` in your browser from outside of your cluster.

Chapter 5. Monitoring

Chapter 6. Monitoring

6.1. Monitoring Stack On Kubernetes

Monitoring Example

This is not an officially supported recommendation and does not claim complete coverage of any use case in a production environment.

The described monitoring approach in this document is a generalized example of one way of monitoring a SUSE CaaS Platform cluster.

Please apply best practices to develop your own monitoring approach using the described examples and available health checking endpoints.

This document aims to describe monitoring in a Kubernetes environment.

The monitoring stack consists of a metrics server, a visualization platform, and an ingress controller for authentication.

Prometheus Server & Alertmanager

Prometheus is an open-source monitoring system with a dimensional data model, flexible query language, efficient time series database and modern alerting approach.

Prometheus Alertmanager handles client alerts, sanitizes duplicates and noise and routes them to configuratble receivers.

Grafana

Grafana is an open-source system for querying, analysing and visualizing metrics.

NGINX Ingress Controller

Deploying NGINX Ingress Controller allows us to provide TLS termination to our services and to provide basic authentication to the Prometheus Expression browser/API.

6.1.1. Prerequisites

1. Monitoring namespace

   We will deploy our monitoring stack in its own namespace and therefore create one.

   ```
   {prompt.user}``kubectl create namespace monitoring``
   ```

2. Create DNS entries

   In this example, we will use a worker node with IP `192.168.1.113` to expose our services.

   You should configure proper DNS names in any production environment. These values are only for example purposes.

   ```
   monitoring.example.com                        IN  A       192.168.1.113
   prometheus.example.com                        IN  CNAME
   monitoring.example.com
   prometheus-alertmanager.example.com           IN  CNAME
   monitoring.example.com
   grafana.example.com                           IN  CNAME
   monitoring.example.com
   ```

   Or add this entry to /etc/hosts

   ```
   192.168.1.113 prometheus.example.com prometheus-
   alertmanager.example.com grafana.example.com
   ```

3. Create certificates

   You will need SSL certificates for the shared resources. If you are deploying in a pre-defined network environment, please get proper certificates from your network administrator. In this example, the domains are named after the components they represent. `prometheus.example.com`, `prometheus-alertmanager.example.com` and `grafana.example.com`

6.1.2. NGINX Ingress Controller

Procedure: Configure And Deploy NGINX Ingress Controller

1. Choose which networking configuration the Ingress controller should have. Create a file `nginx-ingress-config-values.yaml` with one of the following examples as content.

   - **NodePort**: The services will be publicly exposed on each node of the cluster, including master nodes, at port `30080` for `HTTP` and `30443` for `HTTPS`.

     ```
     # Enable the creation of pod security policy
     podSecurityPolicy:
       enabled: true

     # Create a specific service account
     serviceAccount:
       create: true
       name: nginx-ingress

     # Publish services on port HTTP/30080
     # Publish services on port HTTPS/30443
     # These services are exposed on each node
     controller:
       service:
         type: NodePort
         nodePorts:
           http: 30080
           https: 30443
     ```

   - **ClusterIP with external IP(s)**: The services will be exposed on specific nodes of the cluster, at port `80` for `HTTP` and port `443` for `HTTPS`.

     ```
     # Enable the creation of pod security policy
     podSecurityPolicy:
       enabled: true

     # Create a specific service account
     serviceAccount:
       create: true
       name: nginx-ingress

     # Publish services on port HTTP/80
     # Publish services on port HTTPS/443
     # These services are exposed on the node with IP 192.168.1.113
     controller:
       service:
         externalIPs:
           - 192.168.1.113
     ```

2. Deploy the upstream helm chart and pass along our configuration values

file.

```
{prompt.user}``helm install --name nginx-ingress stable/nginx-ingress
\
--namespace monitoring \
--values nginx-ingress-config-values.yaml``
```

The result should be two running pods:

```
{prompt.user}``kubectl -n monitoring get po`` NAME
READY     STATUS    RESTARTS   AGE
nginx-ingress-controller-74cffccfc-p8xbb         1/1       Running
0          4s
nginx-ingress-default-backend-6b9b546dc8-mfkjk   1/1       Running
0          4s
```

6.1.3. TLS

You must configure your certificates for the components as secrets in Kubernetes . Get certificates from your local certificate authority. In this example we are using a single certificate shared by the components `prometheus.example.com`, `prometheus-alertmanager.example.com` and `grafana.example.com`.

Create Individual Secrets For Components

Should you choose to secure each service with an individual certificate, you must repeat the step below for each component and adjust the name for the individual secret each time.

In this example the name is `monitoring-tls`.

Note Down Secret Names For Configuration

Please note down the names of the secrets you have created. Later configuration steps require secret names to be specified.

Procedure: Create TLS secrets in Kubernetes

1.

```
{prompt.user}``kubectl create -n monitoring secret tls monitoring-tls
\
--key  ./monitoring.key \
--cert ./monitoring.crt``
```

Using Self-signed Certificates (optional)

In some cases you will want to create self-signed certificates for testing of the stack. This is not recommended. If you are using proper CA signed certificates, you must skip this entirely.

Procedure: Create Self-signed Certificates

1. Do not use self-signed certificates in production environments. There is severe risk of Man-in-the-middle attacks. Use proper certificates signed by your CA.

2. Create a file *openssl.conf* with the appropriate values

```
[req]
distinguished_name = req_distinguished_name
req_extensions = v3_req
default_md = sha256
default_bits = 4096
prompt=no

[req_distinguished_name]
C = CZ
ST = CZ
L = Prague
O = example
OU = monitoring
CN = example.com
emailAddress = admin@example.com

[ v3_req ]
basicConstraints = CA:FALSE
keyUsage = keyEncipherment, dataEncipherment
extendedKeyUsage = serverAuth
subjectAltName = @alt_names

[alt_names]
DNS.1 = prometheus.example.com
DNS.2 = prometheus-alertmanager.example.com
DNS.3 = grafana.example.com
```

This certificate uses Subject Alternative Names so it can be used for Prometheus and Grafana.

3. Generate certificate

```
{prompt.user}``openssl req -x509 -nodes -days 365 -newkey rsa:4096 \
-keyout ./monitoring.key -out ./monitoring.crt \
-config ./openssl.conf -extensions 'v3_req'``
```

4. Add TLS secret to Kubernetes

```
{prompt.user}``kubectl create -n monitoring secret tls monitoring-tls
\
--key  ./monitoring.key \
--cert ./monitoring.crt``
```

6.1.4. Prometheus

Prometheus Pushgateway

Deploying Prometheus Pushgateway is out of the scope of this document.

1. Configure Authentication

   We need to create a `basic-auth` secret so the NGINX Ingress Controller can perform authentication.

   Install `htpasswd` on your local workstation

```
{prompt.sudo}``zypper in apache2-utils``
```

   Create the secret file `auth`

   It is very important that the filename is `auth` . During creation, a key in the configuration containing the secret is created that is named after the used filename. The ingress controller will expect a key named `auth`.

```
htpasswd -c auth admin
New password:
Re-type new password:
Adding password for user admin
```

Create secret in Kubernetes

```
{prompt.user}``kubectl create secret generic -n monitoring
prometheus-basic-auth --from-file=auth``
```

2. Create a configuration file `prometheus-config-values.yaml`

   We need to configure the storage for our deployment. Choose among the options and uncomment the line in the config file. In production environments you must configure persistent storage. **Use an existing `PersistentVolumeClaim`** Use a `StorageClass` (preferred)

```
# Alertmanager configuration
alertmanager:
  enabled: true
  ingress:
    enabled: true
    hosts:
    -  prometheus-alertmanager.example.com
    annotations:
      kubernetes.io/ingress.class: nginx
      nginx.ingress.kubernetes.io/auth-type: basic
      nginx.ingress.kubernetes.io/auth-secret: prometheus-basic-auth
      nginx.ingress.kubernetes.io/auth-realm: "Authentication
Required"
    tls:
      - hosts:
        - prometheus-alertmanager.example.com
        secretName: monitoring-tls
  persistentVolume:
    enabled: true
    ## Use a StorageClass
    storageClass: my-storage-class
    ## Create a PersistentVolumeClaim of 2Gi
    size: 2Gi
    ## Use an existing PersistentVolumeClaim (my-pvc)
    #existingClaim: my-pvc

## AlertManager is configured through alertmanager.yml. This file and
any others
## listed in alertmanagerFiles will be mounted into the alertmanager
pod.
```

```
## See configuration options
https://prometheus.io/docs/alerting/configuration/
#alertmanagerFiles:
#  alertmanager.yml:

# Create a specific service account
serviceAccounts:
  nodeExporter:
    name: prometheus-node-exporter

# Allow scheduling of node-exporter on master nodes
nodeExporter:
  hostNetwork: false
  hostPID: false
  podSecurityPolicy:
    enabled: true
    annotations:
      seccomp.security.alpha.kubernetes.io/allowedProfileNames:
'docker/default'
      apparmor.security.beta.kubernetes.io/allowedProfileNames:
'runtime/default'
      seccomp.security.alpha.kubernetes.io/defaultProfileName:
'docker/default'
      apparmor.security.beta.kubernetes.io/defaultProfileName:
'runtime/default'
  tolerations:
    - key: node-role.kubernetes.io/master
      operator: Exists
      effect: NoSchedule

# Disable Pushgateway
pushgateway:
  enabled: false

# Prometheus configuration
server:
  ingress:
    enabled: true
    hosts:
    - prometheus.example.com
    annotations:
      kubernetes.io/ingress.class: nginx
      nginx.ingress.kubernetes.io/auth-type: basic
      nginx.ingress.kubernetes.io/auth-secret: prometheus-basic-auth
      nginx.ingress.kubernetes.io/auth-realm: "Authentication
Required"
    tls:
      - hosts:
        - prometheus.example.com
        secretName: monitoring-tls
  persistentVolume:
    enabled: true
    ## Use a StorageClass
    storageClass: my-storage-class
    ## Create a PersistentVolumeClaim of 8Gi
    size: 8Gi
    ## Use an existing PersistentVolumeClaim (my-pvc)
    #existingClaim: my-pvc

## Prometheus is configured through prometheus.yml. This file and any
others
## listed in serverFiles will be mounted into the server pod.
## See configuration options
##
https://prometheus.io/docs/prometheus/latest/configuration/configurat
ion/
#serverFiles:
#  prometheus.yml:
```

3. Deploy the upstream helm chart and pass our configuration values file.

```
{prompt.user}``helm install --name prometheus stable/prometheus \
--namespace monitoring \
--values prometheus-config-values.yaml``
```

   There need to be 3 pods running (3 node-exporter pods because we have 3 nodes).

```
{prompt.user}kubectl -n monitoring get po | grep prometheus
NAME                                             READY     STATUS
RESTARTS   AGE
prometheus-alertmanager-5487596d54-kcdd6         2/2       Running
0          2m
prometheus-kube-state-metrics-566669df8c-krblx   1/1       Running
0          2m
prometheus-node-exporter-jnc5w                   1/1       Running
0          2m
prometheus-node-exporter-qfwp9                   1/1       Running
0          2m
prometheus-node-exporter-sc4ls                   1/1       Running
0          2m
prometheus-server-6488f6c4cd-5n9w8               2/2       Running
0          2m
```

4. At this stage, the Prometheus Expression browser/API should be accessible, depending on your network configuration at `https://prometheus.example.com` or `https://prometheus.example.com:30443`.

6.1.5. Alertmanager Configuration Example

The configuration sets one "receiver" to get notified by email when a node meets one of these conditions:

- Node is unschedulable
- Node runs out of disk space
- Node has memory pressure
- Node has disk pressure

The first two are critical because the node can not accept new pods, the last two are just warnings.

The Alertmanager configuration can be added to `prometheus-config-values.yaml` by adding the `alertmanagerFiles` section.

For more information on how to configure Alertmanager, refer to Prometheus: Alerting - Configuration.

Procedure: Configuring Alertmanager

1. Add the `alertmanagerFiles` section to your Prometheus configuration.

```
alertmanagerFiles:
  alertmanager.yml:
    global:
      # The smarthost and SMTP sender used for mail notifications.
      smtp_from: alertmanager@example.com
      smtp_smarthost: smtp.example.com:587
      smtp_auth_username: admin@example.com
      smtp_auth_password: <password>
      smtp_require_tls: true

    route:
      # The labels by which incoming alerts are grouped together.
      group_by: ['node']

      # When a new group of alerts is created by an incoming alert,
wait at
      # least 'group_wait' to send the initial notification.
      # This way ensures that you get multiple alerts for the same
group that start
      # firing shortly after another are batched together on the
first
      # notification.
      group_wait: 30s

      # When the first notification was sent, wait 'group_interval'
to send a batch
      # of new alerts that started firing for that group.
      group_interval: 5m

      # If an alert has successfully been sent, wait
'repeat_interval' to
      # resend them.
      repeat_interval: 3h

      # A default receiver
      receiver: admin-example

    receivers:
    - name: 'admin-example'
      email_configs:
      - to: 'admin@example.com'
```

2. Replace the empty set of rules `rules: {}` in the `serverFiles` section of the configuration file.

   For more information on how to configure alerts, refer to: Prometheus: Alerting - Notification Template Examples

```
serverFiles:
  alerts: {}
  rules:
    groups:
    - name: caasp.node.rules
      rules:
      - alert: NodeIsNotReady
        expr:
kube_node_status_condition{condition="Ready",status="false"} == 1
        for: 1m
        labels:
          severity: critical
        annotations:
          description: '{{ $labels.node }} is not ready'
      - alert: NodeIsOutOfDisk
        expr:
kube_node_status_condition{condition="OutOfDisk",status="true"} == 1
        labels:
          severity: critical
        annotations:
          description: '{{ $labels.node }} has insufficient free disk
space'
      - alert: NodeHasDiskPressure
        expr:
kube_node_status_condition{condition="DiskPressure",status="true"} ==
1
        labels:
          severity: warning
        annotations:
          description: '{{ $labels.node }} has insufficient available
disk space'
      - alert: NodeHasInsufficientMemory
        expr:
kube_node_status_condition{condition="MemoryPressure",status="true"}
== 1
        labels:
          severity: warning
        annotations:
          description: '{{ $labels.node }} has insufficient available
memory'
```

3. You should now be able to see you AlertManager at https://prometheus-alertmanager.example.com/.

6.1.6. Grafana

Starting from Grafana 5.0, it is possible to dynamically provision the data sources and dashbords via files. In Kubernetes , these files are provided via the utilization of `ConfigMap`, editing a `ConfigMap` will result by the modification of the configuration without having to delete/recreate the pod.

Procedure: Configuring Grafana

1. Configure provisioning

   Create the default datasource configuration file *grafana-datasources.yaml* which point to our Prometheus server

   ```
   ---
   kind: ConfigMap
   apiVersion: v1
   metadata:
     name: grafana-datasources
     namespace: monitoring
     labels:
        grafana_datasource: "1"
   data:
     datasource.yaml: |-
       apiVersion: 1
       deleteDatasources:
         - name: Prometheus
           orgId: 1
       datasources:
       - name: Prometheus
         type: prometheus
         url: http://prometheus-server.monitoring.svc.cluster.local:80
         access: proxy
         orgId: 1
         isDefault: true
   ```

2. Create the ConfigMap in Kubernetes

   ```
   {prompt.user}``kubectl create -f grafana-datasources.yaml``
   ```

3. Configure storage for the deployment

   Choose among the options and uncomment the line in the config file. In production environments you must configure persistent storage. **Use an existing PersistentVolumeClaim** Use a StorageClass (preferred) ** Create a file *grafana-config-values.yaml* with the appropriate values

```
# Configure admin password
adminPassword: <password>

# Ingress configuration
ingress:
  enabled: true
  annotations:
    kubernetes.io/ingress.class: nginx
  hosts:
    - grafana.example.com
  tls:
    - hosts:
      - grafana.example.com
      secretName: monitoring-tls

# Configure persistent storage
persistence:
  enabled: true
  accessModes:
    - ReadWriteOnce
  ## Use a StorageClass
  storageClassName: my-storage-class
  ## Create a PersistentVolumeClaim of 10Gi
  size: 10Gi
  ## Use an existing PersistentVolumeClaim (my-pvc)
  #existingClaim: my-pvc

# Enable sidecar for provisioning
sidecar:
  datasources:
    enabled: true
    label: grafana_datasource
  dashboards:
    enabled: true
    label: grafana_dashboard
```

4. Deploy the upstream helm chart and pass our configuration values file

```
{prompt.user}``helm install --name grafana stable/grafana \
--namespace monitoring \
--values grafana-config-values.yaml``
```

5. The result should be a running Grafana pod

```
{prompt.user}``kubectl -n monitoring get po | grep grafana`` NAME
READY     STATUS    RESTARTS   AGE
grafana-dbf7ddb7d-fxg6d                           3/3       Running
0          2m
```

At this stage, Grafana should be accessible, depending on your network configuration at `https://grafana.example.com` or `https://grafana.example.com:30443`

6. Now you can deploy an existing Grafana dashboard or build your own.

Adding Grafana Dashboards

There are two ways to add dashboards to Grafana:

- Deploy an existing dashboard from grafana.com
  a. Open the deployed Grafana in your browser and log in.
  b. On the home page of Grafana, hover your mousecursor over the menu:+[] button on the left sidebar and click on the **import** menuitem.
  c. Select an existing dashboard for your purpose from https://grafana.com/dashboards. Copy the URL to the clipboard.
  d. Paste the URL (for example) `https://grafana.com/dashboards/3131` into the first input field to import the "Kubernetes All Nodes" Grafana Dashboard. After pasting in the url, the view will change to another form.
  e. Now select the "Prometheus" datasource in the `prometheus` field and click on the **import** button.
  f. The browser will redirect you to your newly created dashboard.
- Deploy a configuration file containing the dashboard definition.
  a. Create your dashboard defintion file as a `ConfigMap`, for example `grafana-dashboards-caasp-cluster.yaml`.

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: grafana-dashboards-caasp-cluster
  namespace: monitoring
  labels:
     grafana_dashboard: "1"
data:
  caasp-cluster.json: |-
    {
      "__inputs": [
        {
          "name": "DS_PROMETHEUS",
          "label": "Prometheus",
          "description": "",
          "type": "datasource",
          "pluginId": "prometheus",
          "pluginName": "Prometheus"
        }
      ],
      "__requires": [
        {
          "type": "grafana",
[...]
continues with definition of dashboard JSON
[...]
```

b. Apply the `ConfigMap` to the cluster.

```
{prompt.user}``kubectl apply -f grafana-dashboards-caasp-
cluster.yaml``
```

You can find a couple of dashboard examples for SUSE CaaS Platform in the Kubic project on GitHub. This repo provides dashboards to visualize Kubernetes resources.

6.2. Health Checking

Although Kubernetes takes care of a lot of the traditional deployment problems with its self-healing capabilities, it is considered good practice to monitor the availability and health of your services and applications to react to problems should they go beyond these automated measures.

A very basic (visual) health check can be achieved by accessing cAdvisor on the admin node at port `4194`. It will show a basic statistics UI about the cluster resources.

A complete set of instructions on how to monitor and maintain the health of you cluster is, however, beyond the scope of this document.

There are three levels of health checks.

- Cluster
- Node
- Application / Service

6.2.1. Cluster Health Checks

The basic check if a cluster is working correctly is based on a few criteria:

- Are all services running as expected?
- Is there at least one Kubernetes master fully working? Even if the deployment is configured to be highly available, it's useful to know if `kube-controller-manager` is down on one of the machines.

Understanding cluster health

For further information consider reading Kubernetes: Troubleshoot Clusters

Kubernetes master

All components in Kubernetes expose a `/healthz` endpoint. The expected (healthy) response is a `200 HTTP` and a response body containing `ok`.

The minimal services for the master to work properly are:

kube-apiserver

The component that receives your requests from `kubectl` and from the rest of the Kubernetes components.

Endpoint: `https://MASTER NODE FQDN:6444/healthz` (HTTPS)

```
{prompt.user}``curl -i https://localhost:6444/healthz`` ok
```

kube-controller-manager

The component that contains the control loop, driving current state to the desired state.

Endpoint: `http://MASTER NODE FQDN:10252/healthz` (HTTP)

```
{prompt.user}``curl -i http://localhost:10252/healthz`` ok
```

kube-scheduler

The component that schedules workloads to nodes.

Endpoint: `http://MASTER NODE FQDN:10251/healthz` (HTTP)

```
{prompt.user}``curl -i http://localhost:10251/healthz`` ok
```

High-Availability Environments

In a HA environment you can monitor `kube-apiserver` on `https://[replaceable]`MASTER NODE LOADBALANCER`:6443/healthz`.

If any master node is running correctly you will receive a valid response.

This does, however, not mean that all master nodes necessarily work correctly. To ensure that all master nodes work properly, the health checks must be repeated individually for each master node deployed.

This endpoint will return a successful HTTP response if the cluster is operational; otherwise it will fail. It will for example check that it can access `etcd` too. This should not be used to infer that the overall cluster health is ideal. It will return a a successful response even when only minimal operational cluster health exists.

To probe for full cluster health, you must perform individual health checking for all machines individually.

` etcd` Cluster

Check that all machines that have the `etcd` role on the cluster see the etcd cluster as healthy.

```
{prompt.user}``docker exec -it $(docker ps -q -f name="salt-master") salt
-G 'roles:etcd' \
cmd.run 'set -a; source /etc/sysconfig/etcdctl; etcdctl cluster-health'``
f69e7af2880f42d68dca26ca892cb945:
    member af7ffa9bb1cb7c67 is healthy: got healthy result from
https://caasp-master:2379
    member cc40a990d09b4705 is healthy: got healthy result from
https://caasp-worker-1:2379
    member fe9b5ee9e1cc3cf7 is healthy: got healthy result from
https://caasp-worker-2:2379
    cluster is healthy
ab040b25c2584bc8904971c0acbb250f:
    member af7ffa9bb1cb7c67 is healthy: got healthy result from
https://caasp-master:2379
    member cc40a990d09b4705 is healthy: got healthy result from
https://caasp-worker-1:2379
    member fe9b5ee9e1cc3cf7 is healthy: got healthy result from
https://caasp-worker-2:2379
    cluster is healthy
63008aabc75b471b9a1aa2f64e4d30eb:
    member af7ffa9bb1cb7c67 is healthy: got healthy result from
https://caasp-master:2379
    member cc40a990d09b4705 is healthy: got healthy result from
https://caasp-worker-1:2379
    member fe9b5ee9e1cc3cf7 is healthy: got healthy result from
https://caasp-worker-2:2379
    cluster is healthy
```

More information on etcd cluster health can be found in Behavior of `etcd`.

Running Components

Check if the cluster has all required components running:

```
{prompt.user}``kubectl cluster-info`` {kube}master is running at
https://api.infra.caasp.local:6443
Dex is running at
https://api.infra.caasp.local:6443/api/v1/namespaces/kube-
system/services/dex:dex/proxy
KubeDNS is running at
https://api.infra.caasp.local:6443/api/v1/namespaces/kube-
system/services/kube-dns:dns/proxy
Tiller is running at
https://api.infra.caasp.local:6443/api/v1/namespaces/kube-
system/services/tiller:tiller/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info
dump'.
```

You can optionally run `kubectl cluster-info dump` to obtain a much more detailed output

6.2.2. Node Health Checks

The basic check if a node is healthy consists of checking if `kubelet` and the CNI (Container Networking Interface) are working properly.

`kubelet`

Is the `kubelet` up and working in this node?

The `kubelet` has a port exposed `10250` on all machines; it's possible to perform an HTTP request to the endpoint to find out if the kubelet is healthy on that machine. The expected (healthy) response is a `200 HTTP` and a response body containing `ok`.

Endpoint: `https://NODE:10250/healthz` (HTTPS)

```
{prompt.user}``curl -i https://localhost:10250/healthz`` ok
```

` CNI`

Is CNI (Container Networking Interface) working as expected in this node? If not, `kube-dns` can not start. Check if the `kube-dns` service is running.

```
{prompt.user}``kubectl get deployments -n kube-system`` NAME
DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
dex           3         3         3            3           7d
kube-dns      3         3         3            3           7d
tiller-deploy 1         1         1            1           7d
```

If kube-dns is running and you are able to create pods then you can be certain that CNI and your CNI plugin are working correctly.

There's also the Monitor Node Health check. This is a `DaemonSet` that runs on every node, and reports to the `apiserver` back as `NodeCondition` and `Events`.

6.2.3. Service/Application Health Checks

If the deployed services contain a health endpoint, or if they contain an endpoint that can be used to determine if the service is up, you can use `livenessProbes` and/or `readinessProbes`.

> *Health check endpoints vs. functional endpoints*
>
> A proper health check is always preferred if designed correctly.
>
> Despite the fact that any endpoint could potentially be used to infer if your application is up, a specific health endpoint in your application is preferred. Such an endpoint will only respond affirmatively when all your setup code on the server has finished and the application is running in a desired state.

`livenessProbes` and `readinessProbes` share configuration options and probe types.

initialDelaySeconds
: Number of seconds to wait before performing the very first liveness probe.

periodSeconds
: Number of seconds that the kubelet should wait between liveness probes.

successThreshold
: Number of minimum consecutive successes for the probe to be considered successful (Default: 1).

failureThreshold
: Number of times this probe is allowed to fail in order to assume that the

service is not responding (Default: 3).

timeoutSeconds
: Number of seconds after which the probe times out (Default: 1).

There are different options for the liveness probe to check:

Command
: A command executed within a container; a retcode of 0 means success.

  All other return codes mean failure.

TCP
: If a TCP connection can be established is considered success.

HTTP
: Any HTTP response between `200` and `400` indicates success.

livenessProbe

livenessProbes are used to detect running but misbehaving pods/a service that might be running (the process didn't die), but that is not responding as expected.

Probes are executed by each `kubelet` against the pods that define them and that are running in that specific node.

When a `livenessProbe` fails, Kubernetes will automatically restart the pod and increase the `RESTARTS` count for that pod.

These probes will be executed every `periodSeconds` starting from `initialDelaySeconds`.

readinessProbe

readinessProbes are used to wait for processes that take some time to start. Despite the container running, it might be performing some time consuming initializatoin operations. During this time, you don't want Kubernetes to route traffic to that specific pod; also, you don't want that container to be restarted because it will appear unresponsive.

These probes will be executed every `periodSeconds` starting from `initialDelaySeconds` until the service is ready.

They support the same kind of probes as the `livenessProbe`

Both probe types can be used at the same time. The `livenessProbe` will ensure that if a service is running yet misbehaving, it will be restarted, and `readinessProbe` will ensure that Kubernetes won't route traffic to that specific pod until it's considered to be fully functional and running.

6.2.4. General Health Checks

We recommend to apply other best practices from system administration to your monitoring and health checking approach. These steps are not specific to SUSE CaaS Platform and are beyond the scope of this document. To simplify performing tasks like disk usage checks, you can use `salt`. For more information see: Interacting with Salt

Chapter 7. Logging

Chapter 8. Logging

Scope Of This Document

The scope of this document is limited to the SUSE CaaS Platform Infrastructure layer. Logging the activity of deployed applications is beyond the scope of this document.

Please refer to: Kubernetes: Logging Architecture for more information.

The SUSE CaaS Platform components run in indvidual Docker containers and typically you can get a set of log information using docker-logs.

For detailed information on how to use log files for the individual components, please refer to the respective official documentation of the component.

8.1. About SUSE CaaS Platform Logging

Logging across the cluster is done on multiple logical levels. You can think of them as three logical layers (simplified).

- SUSE CaaS Platform Infrastructure (Salt, Velum, LDAP)
- Cluster (etcd, dex, tiller, kubernetes apiserver, kubernetes controller-manager, kubernetes scheduler)
- Pod (kubelet, haproxy, flanneld, systemd, journald, dmesg)

The individual log files allow introspection of activities across the cluster. Due to some technical limitations it is sometimes not possible to directly trace log events from one layer to the next. Most log files would be used for debugging purposes only.

8.1.1. Log levels

There are two main components and their respective sub-components that allow configuration of different loglevels: Salt and Kubernetes .

- Salt
  - `salt-master`

    - `salt-minion` on each machine
- Kubernetes
    - Master nodes
        - `apiserver`
        - `controller-manager`
        - `scheduler`
    - All nodes
        - `kubelet`
        - `kube proxy`

8.2. Admin Node Logs

8.2.1. Velum Logs

The Velum logs will contain more details on error messages displayed in the dashboard.

Logs are generated in the Velum container (`k8s_velum-dashboard`) running on the admin node.

```
{prompt.user}``docker logs $(docker ps -q -f name="k8s_velum-
dashboard")``
```

8.2.2. OpenLDAP Logs

The OpenLDAP logs contain information about the authentication of users in the Velum dashboard.

The OpenLDAP logs are generated in the `k8s_openldap_velum` container running on the admin node.

```
{prompt.user}``docker logs $(docker ps -q -f name="k8s_openldap_velum")``
```

8.3. Salt Logging

`Salt` performs a variety of functions that control behavior and configuration of the Kubernetes cluster. A failure of executing certain `Salt` workflows could lead to an unhealthy cluster, in such a case it can be inspected using `Salt` log information.

Salt orchestration and master log are generated on the `Salt Master` container (`k8s_salt-master`) running on the admin node.

Salt minion logs are generated in the respective salt minion containers on each node. The `salt-master` collects these logs on the admin node and writes them into the MariaDB container `k8s_velum-mariadb` .

8.3.1. Salt Orchestration Log

The `Salt` orchestration logs contains log entries about orchestration events that have changed the cluster.

Orchestration events are:

- Bootstrapping
- Adding new nodes
- Removing nodes
- Updating settings
- Upgrading a cluster

```
{prompt.user}``/var/lib/supportutils-plugin-suse-caasp/debug-salt \
--json_output=events.txt \
--summary_output=events-summarized.txt \
--text-status \
--no-color``
```

Reading the `events-summarized.txt` file should be enough for detecting most (if not all) of the issues caused by `Salt`.

8.3.2. Salt Master Log

Retrieve the `salt-master` logs.

```
{prompt.user}``docker exec -it $(docker ps -q -f name="salt-master") \
cat /var/log/salt/master``
```

8.3.3. Salt Minion Logs

Retrieve the `salt-minion` logs for all nodes. This will show all output for all `salt-minions` at once. Execute the following command on the admin node.

Of course, it’s possible to retrieve this information on any specific node by reading the `/var/log/salt/minion` file.

```
{prompt.user}``docker exec -it $(docker ps -q -f name="salt-master") \
salt '*' cmd.run "cat /var/log/salt/minion"``
```

8.3.4. Salt Log Levels

Salt provides different loglevels that apply both to the master and the minions.

quiet
: Nothing should be logged at this level

critical
: Critical errors

error
: Errors

warning
: Warnings

info
: Normal log information

profile
: Profiling information on salt performance

debug
: Information useful for debugging both salt implementations and salt code

trace
: More detailed code debugging information

garbage
: Even more debugging information

all

Everything

For detailed explanations of the usage of these log levels please see: Salt Log Levels (Upstream)

Setting A Different Log Level

The `salt-master` configuration can be modified on the admin node, at `/etc/caasp/salt-master-custom.conf` . Inside this file you can add: `log_level: debug`.

Note that after any change on this file you need to restart the `salt-master` container, like:

```
``docker rm -f $(docker ps -q -f name="salt-master")`` .
```

After deleting this container, the `kubelet` will bring up a new `salt-master` container automatically with the new configuration applied. Then. you can check the logs with the `debug` loglevel.

```
``docker logs -f $(docker ps -q -f name="salt-master")``
```

8.4. Transactional Update Log

The `transactional-update` method processes updates in the background and generates new machine image snapshots. This process can run into issues. Possible causes are connectivity issues or timeouts against the package repository. In such cases the update fails and the affected node will be marked with a red cross in Velum .

In most cases this situation resolves itself the next time the update process runs automatically. If you have performed manual updates or must debug a failed update, you can read the log for `transactional-update` with the command below.

The `transactional-update` logs are generated in the MicroOS layer on each node respectively.

For more information on transactional-update, see: Transactional Updates

```
{prompt.user}``docker exec -it $(docker ps -q -f name="salt-master") \
salt -P 'roles:(admin|kube-(master|minion)' \
cmd.run "journalctl -u transactional-update"``
```

8.5. Kubernetes Audit Log

To track actions that have been performed on the cluster, you can enable the Kubernetes audit log in Velum .

Navigate to: **Settings → KUBERNETES → Auditing** . This allows the audit logs to be written on the Kubernetes master nodes at `/var/log/kube-apiserver/audit.log` and you can then use an external data collector like `fluentd` to collect all the audit logs.

Kubernetes Audit Log Documentation

For more information on the audit log and its contents, see: Kubernetes Documentation: Auditing

KubernetesAudit Log Limitations

The Kubernetes audit log only collects and stores actions performed on the Kubernetes level of the cluster. This does not include any actions performed by SUSE CaaS Platform administrators in Velum or any of the resulting actions of services.

[velum settings audit] | *velum_settings_audit.png*

Enable Auditing

Enable / Disable the audit logging feature (Default: `Disabled`)

Max size

Maximum size in megabytes of the audit log file before it gets rotated (Default: `10`)

Max age

Maximum number of days to retain old audit log files (Default: `15`)

Max backup

Maximum number of audit log files to retain (Default: `20`)

Policy
: The YAML file defining the auditing policy rules

8.5.1. Kubernetes Log Levels

For Kubernetes our default `loglevel` is `2` (Kubernetes Upstream: Output Verbosity and Debugging).

0
: Generally useful for this to ALWAYS be visible to an operator.

1
: A reasonable default log level if you don't want verbosity.

2
: Useful steady state information about the service and important log messages that may correlate to significant changes in the system. This is the recommended default log level for most systems.

3
: Extended information about changes.

4
: Debug level verbosity.

6
: Display requested resources.

7
: Display HTTP request headers.

8
: Display HTTP request contents.

Setting A Different Log Level

Procedure: Modify KubernetesLog Level Across The Cluster

1. Change the log level value in the Salt pillar.

```
{prompt.user}``docker exec -it $(docker ps -q -f name="dashboard") \
entrypoint.sh bundle exec rake "velum:create_pillar['kube_log_level',
'4']"``
```

2. Then run Salt orchestration to rebuild the configuration across the cluster.

```
{prompt.user}``docker exec -it $(docker ps -q -f name="salt-master")
\
salt-run state.orchestrate orch.kubernetes``
```

If modified, this setting will be applied to all Kubernetes components; there is no way to set a different loglevel per component. Moreover, there is no way to specify different loglevels per machine.

8.6. External log collection

At Kubernetes level, there are different solutions that can be implemented. For example: fluentd can be used to collect all applications log in a central instance.

Then, Elasticsearch and Kibana can be used to provide an intuitive way to visualize and interact with the logs.

Chapter 9. Miscellaneous Configuration

Chapter 10. Miscellaneous Configuration

10.1. Configuring Timezone

To configure the timezone during deployment you can use `cloud-init` or `AutoYaST` during the initial installation procedure. If you wish to change the timezone for an existing cluster, you must update it manually using `Salt`.

10.1.1. New Deployment

Using cloud-init

[_sec.deploy.cloud_init.user_data.timezone]

Using AutoYaST

Use a modified AutoYaST control file as described in [_sec.deploy.autoyast].

10.1.2. Existing Deployment

To change the timezone on an existing cluster you must use `Salt` to remove the `/etc/localtime` symlink and replace it with the desired timezone file from `/usr/share/zoneinfo` in its place. SSH into the Admin node and run:

```
{prompt.user}``docker exec -it $(docker ps -q -f name="salt-master") \
salt -P 'roles:(admin|kube-(master|minion)' \
cmd.run "ln -sf /usr/share/zoneinfo/Europe/Berlin /etc/localtime"``
```

10.2. The `issue-generator` Command

On SUSEMicroOS , `/etc/issue` is generated by `issue-generator`.

This allows MicroOS to display a dynamic message above the login prompt, which is generated from the contents of several different files.

Files in `/etc/issue.d` override files with the same name in `/usr/lib/issue.d` and `/run/issue.d` . Files in `/run/issue.d` override files with the same name in `/usr/lib/issue.d` .

Packages should install their configuration files in `/usr/lib/issue.d` . Files in `/etc/issue.d` are reserved for the local administrator, who may use this logic to override the files installed by vendor packages.

All configuration files are sorted by their filename in lexicographic order, regardless of which of the directories they reside in.

If you run the command without any arguments, all the input files will be applied.

Which network interfaces are shown can be configured in the file `/etc/sysconfig/issue-generator` .

To disable the display of ssh keys, use the following command:

```
{prompt.root}``systemctl disable issue-add-ssh-keys.service``
```

Chapter 11. SUSE Enterprise Storage Integration

SUSE CaaS Platform can use another SUSE product as storage for containers{mdash}SUSE Enterprise Storage (henceforth called SES). This chapter gives details on several ways to integrate these two products.

11.1. Prerequisites

Before you start the integration process, you need to ensure the following:

- The SUSE CaaS Platform cluster can communicate with the SES nodes{mdash} master, monitoring nodes, OSD nodes and the metadata server, in case you need a shared file system. For more details regarding SES refer to the SES documentation, here: https://www.suse.com/documentation/suse-enterprise-storage/.
- The SES cluster has a pool with RADOS Block Device (RBD) enabled.

11.2. Using RBD in a Pod

The procedure below describes steps to take when you need to use a RADOS Block Device in a Pod.

Procedure: Using RBD In A Pod

1. Retrieve the Ceph admin secret. Get the key value from the file `/etc/ceph/ceph.client.admin.keyring` .
2. On the Master Node apply the configuration that includes the Ceph secret by using `kubectl apply`. Replace `CEPH_SECRET` with your Ceph secret.

```
{prompt.user}``kubectl apply -f - << *EOF*
apiVersion: v1
kind: Secret
metadata:
  name: ceph-secret
type: "kubernetes.io/rbd"
data:
  key: "$(echo CEPH_SECRET | base64)"
*EOF*``
```

3. Create an image in the SES cluster. On the Master Node , run the following

command:

```
{prompt.root}``rbd create -s SIZE YOUR_VOLUME``
```

Replace `SIZE` with the size of the image, for example `1G`, and `YOUR_VOLUME` is the name of the image.

4. Create a pod that uses the image by executing the following command on the Master Node . In this example it is a minimal configuration for using a `RADOS` Block Device. Fill in the IP addresses and ports of your monitor nodes. The port number usually is `6789`.

```
{prompt.user}``kubectl apply -f - << *EOF*
apiVersion: v1
kind: Pod
metadata:
  name: POD_NAME
spec:
  containers:
  - name: CONTAINER_NAME
    image: IMAGE_NAME
    volumeMounts:
    - mountPath: /mnt/rbdvol
      name: rbdvol
  volumes:
  - name: rbdvol
    rbd:
      monitors:
      - 'MONITOR1_IP:MONITOR1_PORT'
      - 'MONITOR2_IP:MONITOR2_PORT'
      - 'MONITOR3_IP:MONITOR3_PORT'
      pool: rbd
      image: YOUR_VOLUME
      user: admin
      secretRef:
        name: ceph-secret
      fsType: ext4
      readOnly: false
*EOF*``
```

5. Verify that the pod exists and its status:

```
{prompt.user}``kubectl get po``
```

6. Once the pod is running, check the mounted volume:

```
{prompt.user}``kubectl exec -it CONTAINER_NAME -- df -k`` ...
/dev/rbd1               999320      1284    929224   0% /mnt/rbdvol
...
```

In case you need to delete the pod, run the following command on the Master Node :

```
{prompt.user}``kubectl delete pod POD_NAME``
```

11.3. Using RBD in Persistent Volume

The following procedure describes how to attach a pod to a persistent SES volume.

Procedure: Creating a Pod with RBD in Persistent Volume

1. Retrieve the Ceph admin secret. Get the key value from the file `/etc/ceph/ceph.client.admin.keyring`.
2. On the Master Node , apply the configuration that includes the Ceph secret by using `kubectl apply`. Replace `CEPH_SECRET` with your Ceph secret.

   ```
   {prompt.user}``kubectl apply -f - << *EOF*
   apiVersion: v1
   kind: Secret
   metadata:
     name: ceph-secret
   type: "kubernetes.io/rbd"
   data:
     key: "$(echo CEPH_SECRET | base64)"
   *EOF*``
   ```

3. Create a volume on the SES cluster:

   ```
   {prompt.root}``rbd create -s SIZE YOUR_VOLUME``
   ```

   Replace `SIZE` with the size of the image, for example `1G` (1 Gigabyte), and `YOUR_VOLUME` is the name of the image.

4. Create the persistent volume on the Master Node :

```
{prompt.user}``kubectl apply -f - << *EOF*
apiVersion: v1
kind: PersistentVolume
metadata:
  name: PV_NAME
spec:
  capacity:
    storage: SIZE
  accessModes:
    - ReadWriteOnce
  rbd:
    monitors:
    - 'MONITOR1_IP:MONITOR1_PORT'
    - 'MONITOR2_IP:MONITOR2_PORT'
    - 'MONITOR3_IP:MONITOR3_PORT'
    pool: rbd
    image: YOUR_VOLUME
    user: admin
    secretRef:
      name: ceph-secret
    fsType: ext4
    readOnly: false
*EOF*``
```

Replace `SIZE` with the desired size of the volume. Use the *gibibit* notation, for example `1Gi`.

5. Create a persistent volume claim on the Master Node :

```
{prompt.user}``kubectl apply -f - << *EOF*
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: PV_NAME
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: SIZE
*EOF*``
```

Replace `SIZE` with the desired size of the volume. Use the *gibibit* notation, for example `1Gi`.

Listing Volumes

This persistent volume claim does not explicitly list the volume. Persistent volume claims work by picking any volume that meets the criteria from a pool. In this case we specified any volume with a size of 1G or larger. When the claim is removed the recycling policy will be followed.

6. Create a pod that uses the persistent volume claim. On the Master Node run the following:

```
{prompt.user}``kubectl apply -f - <<*EOF*
apiVersion: v1
kind: Pod
metadata:
  name: POD_NAME
spec:
  containers:
  - name: CONTAINER_NAME
    image: IMAGE_NAME
    volumeMounts:
    - mountPath: /mnt/rbdvol
      name: rbdvol
  volumes:
  - name: rbdvol
    persistentVolumeClaim:
      claimName: PV_NAME
*EOF*``
```

7. Verify that the pod exists and its status. On the Master Node run:

```
{prompt.user}``kubectl get po``
```

8. Once pod is running, check the volume by running on the Master Node :

```
{prompt.user}``kubectl exec -it CONTAINER_NAME -- df -k`` ...
/dev/rbd3               999320      1284    929224   0% /mnt/rbdvol
...
```

In case you need to delete the pod, run the following command on the Master Node :

```
{prompt.user}``kubectl delete pod CONTAINER_NAME``
```

And when the command finishes, run

```
{prompt.user}``kubectl delete persistentvolume PV_NAME``
```

Deleting A Pod

When you delete the pod, the persistent volume claim is deleted as well. The RBD is not deleted.

Chapter 12. Troubleshooting

Chapter 13. Troubleshooting

This chapter summarizes frequent problems that can *occur while using SUSE CaaS Platform and their solutions.*

13.1. Overview

This chapter is a collection of frequent problems that are reported for SUSE CaaS Platform . Additionally, SUSE support collects problems and their solutions online at https://www.suse.com/support/kb/?id=SUSE_CaaS_Platform.

In case of problems, a detailed system report can be created with the `supportconfig` command line tool. It will collect information about the system such as: current kernel version, hardware, installed packages, partition setup, and much more. The result is a TAR archive of files. After opening a Service Request (SR), you can upload the TAR archive to Global Technical Support. It will help to locate the issue you reported and to assist you in solving the problem. For details, see https://www.suse.com/documentation/sles-12/book_sle_admin/data/cha_adm_support.html.

While resolving a technical difficulty with SUSE Support, you may receive a so-called Program Temporary Fix (PTF). PTFs may be issued for various packages as RPMs. For details about handling PTFs, see Program Temporary Fixes.

For details about how to contact SUSE support, refer to https://www.suse.com/media/flyer/suse_customer_support_quick_reference_guide_flyer.pdf and https://www.suse.com/support/.

13.2. New Nodes Not Showing In Velum

All SUSE CaaS Platform nodes must have a `/etc/salt/minion.d/master.conf` configuration file that contains the IP address or the FQDN of (one of) the Salt master(s) in the cluster. During the installation you must provide the IP/FQDN of the Salt master (typically the admin node).

To become part of the cluster, the node must be (manually) accepted. Velum shows a list of Salt minions that are known to the Salt master and can be accepted.

13.2.1. Nodes Not Showing In "Pending Nodes"

You have added a new machine to the cluster but there is no new node key displayed in the **Pending Nodes** list.

- Check that the configured value in the `master.conf` of the machine is correct and it is communicating with the admin node.
- Check if the Administration Node has enough free memory. In some situations Kubernetes will intermittently shut down containers to conserve resources. If any of `salt-master`, `salt-api` or other critical services are affected, this could lead to this kind of issues.

13.2.2. Accepted Nodes Not Showing In "Nodes" List

If the accepted node does not show up in the **Nodes** list, you can log in to the Administration Node via SSH and perform the following checks:

- Check that the minion ID has been actually accepted:

  ```
  {prompt.user}``docker exec -it $(docker ps --filter=name=salt-master
  -q) salt-key`` Accepted Keys:
  admin
  ca
  231bb41553ac4102bb8fda8b2fd9d60d
  Denied Keys:
  Unaccepted Keys:
  bbaf54654d5649f68a0af5f1914ecedf
  db79f30700974971b5e32d62ba0d5d76
  Rejected Keys:
  ```

  The minion ID of the node should appear on the "Accepted Keys" list.

  If the node does not appear on this "Accepted Keys" list, the acceptance request didn't reach the backend and/or the Salt API. Check with `docker ps` that containers on the Administration Node have approximately the same age. If there is a large age differences, it's likely that the Administration Node is under memory pressure and is shutting down containers. This can lead to multiple problems respective of which container is not available. For example: MariaDB could have been down when Salt tried to insert events into the database etc..

  Adjust workloads accordingly to restore available memory and retry accepting the node from Velum .

- Check that the event processor is running on the Administration Node :

  ```
  {prompt.user}``docker ps -q -f name=event-processor``
  ```

  If it's not running or has a significantly different age from the other containers, make sure the admin node is not under memory pressure.

- Check if the event processor is lagging behind processing events:

  If there was a previous problem with the event processor, it's possible that salt continued creating events while it wasn't running for a long time, causing it to lag behind processing events. Check the number of events in the processing queue.

  ```
  {prompt.user}``docker exec -it $(docker ps -q -f name=dashboard) \
  entrypoint.sh bundle exec rails runner \
  'puts "Number of events to process:
  #{SaltEvent.not_processed.count}"'``
  ```

- Check if Salt couldn't introduce an important event on the database

  Make sure there's enough free space on the Administration Node .

13.3. Debugging Failed Bootstrap

If the bootstrapping as described in [_sec.deploy.install.bootstrap] fails, there are several places to look for errors. The following procedure outlines where to start debugging.

1. Check the logs from Velum dashboard. Execute on the Administration Node :

   ```
   {prompt.root.admin}``docker logs $(docker ps -q -f name="velum-
   dashboard")``
   ```

2. Convert the log into a human readable format and open it with `less`.

```
{prompt.root.admin}``/var/lib/supportutils-plugin-suse-caasp/debug-
salt --json_output=salt.json \
    --summary_output=summary.txt --text-status --no-color``
{prompt.root.admin}``less summary.txt``
```

3. Retry bootstrapping from Administration Node console.

```
{prompt.root.admin}``docker exec -it $(docker ps -q -f name="salt-
master") \
    salt-run -l debug state.orchestrate orch.kubernetes > salt-
orchestration.logs``
```

4. If the orchestration failed in late state, you can check if `etcd` and the Kubernetes cluster is up and running. Log in on a Master Node and check if `etcd` cluster is up and running.

```
{prompt.root.master}``set -a`` {prompt.root.master}``source
/etc/sysconfig/etcdctl`` {prompt.root.master}``export ETCDCTL_API=3``
{prompt.root.master}``etcdctl member list``
{prompt.root.master}``etcdctl endpoint health``
{prompt.root.master}``etcdctl endpoint status``
{prompt.root.master}``journalctl -u etcd``
```

Check if Kubernetes cluster is up and running.

```
{prompt.root.master}``kubectl get nodes``
{prompt.root.master}``kubectl cluster-info``
{prompt.root.master}``journalctl -u kubelet``
{prompt.root.master}``journalctl -u kube-apiserver``
{prompt.root.master}``journalctl -u kube-scheduler``
{prompt.root.master}``journalctl -u kube-controller-manager``
```

5. Collect all relevant logs with `supportconfig` on the Master Node and Administration Node .

```
{prompt.root.admin}``supportconfig`` {prompt.root.admin}``tar -xvjf
/var/log/nts_*.tbz`` {prompt.root.admin}``cd nts*``
{prompt.root.admin}``cat etcd.txt kubernetes.txt salt-minion.txt``
```

```
{prompt.root.master}``supportconfig`` {prompt.root.master}``tar -xvjf
/var/log/nts_*.tbz`` {prompt.root.master}``cd nts*``
{prompt.root.master}``cat etcd.txt kubernetes.txt salt-minion.txt``
```

13.4. Recovering From Failed Update

See Recovering From Failed Updates.

13.5. Checking `etcd` Health

To work with `etcdctl`, you have to pass parameters with paths to required certificates.

Use SSH to log into one of the cluster nodes, for example your first master. The following command provides an example for using `etcdctl`.

```
{prompt.root}``set -a`` {prompt.root}``source /etc/sysconfig/etcdctl``
{prompt.root}``export ETCDCTL_API=3`` {prompt.root}``etcdctl cluster-
health``
```

13.6. Locking Installed Program Temporary Fixes

It can happen that `zypper` removes an installed *Program Temporary Fix* (*PTF*) when updates are started with Velum. To prevent this, execute the following procedure.

Locks Disable Updates for Package

If a package is locked, it will no longer receive any updates until the lock is released with `zypper rl`.

1. Install the PTF manually.

```
{prompt.root}``transactional-update reboot pkg install PTF_FILE.rpm``
```

2. As soon as the node has restarted, verify that the RPM is installed.

```
{prompt.root}``rpm -qa | grep PTF``
```

3. Create a `zypper` lock for this RPM.

```
{prompt.root}``zypper al PTF_PACKAGE_NAME``
```

4. Verify that the lock is created.

```
{prompt.root}``zypper ll``
```

13.7. Network Planning And Reconfiguration

It is not possible to reconfigure the used IP ranges of SUSE CaaS Platform once the deployment is complete. Therefore, carefully plan for required IP ranges and future scenarios.

Additionally, keep the network requirements in mind, as stated in [_sec.deploy.requirements.network].

13.8. Using A Proxy Server With Authentication

If you need to register the Administration Node with SUSE Customer Center over a proxy server that requires authentication, it is not possible to specify the configuration in `/etc/sysconfig/proxy` .

For information about setting authentication credentials and connecting to SUSE Customer Center with `SUSEConnect`, see [_sec.configuration.suseconnect.proxy].

13.9. Replacing TLS/SSL Certificates

Sometimes certificates are not updated properly because they are outdated. To replace outdated certificates, execute the following procedure.

1. Use SSH and log in on the Administration Node .
2. Move the expired certs out of the way.

```
{prompt.root.admin}``mv /etc/pki/{velum,ldap,salt-api}.crt /root``
```

3. Generate new certificates.

```
{prompt.root.admin}``cd /etc/pki``
{prompt.root.admin}``/usr/share/caasp-container-manifests/gen-
certs.sh``
```

Generating Additional Certificates

To regenerate additional certificates, for example `/etc/pki/kubectl-client-cert.crt`, add an additional line at the end of the `gen-certs.sh` script:

```
{prompt.root.admin}``transactional-update shell``
transactional update # ``echo "gencert \"kubectl-client-cert\" \"kubectl-
client-cert\" \
    \"\$all_hostnames\" \"\$(ip_addresses)\"" >>/usr/share/caasp-
container-manifests/gen-certs.sh``
transactional update # ``/usr/share/caasp-container-manifests/gen-
certs.sh``
transactional update # ``exit``
```

+

1. Use SSH and log in on a Master Node .
2. Backup and delete the dex-tls secret.

```
{prompt.root.master}``kubectl -n kube-system get secret dex-tls -o
yaml > /root/dex-tls`` {prompt.root.master}``kubectl -n kube-system
delete secret dex-tls``
```

3. On a master node, find and delete the Dex pods.

This Step Breaks Authentication

Executing this step prevents new authentications requests from succeeding. However, the static credentials located on the master nodes will continue to function.

The Dex pods will not restart by themselves until the dex-tls secret is recreated.

+

+

```
{prompt.root.master}``kubectl -n kube-system get pods | grep dex``
{prompt.root.master}``kubectl -n kube-system delete pods DEX_POD1
DEX_POD2 DEX_POD3``
```

1. Manually run the salt orchestration on the Administration Node . This may take some time.

   ```
   {prompt.root.admin}``docker exec -it $(docker ps -q -f name="salt-
   master") \
       bash -c "salt-run state.orchestrate orch.kubernetes" 2&>1 > salt-
   run.log``
   ```

2. Check the tail of `salt-run.log` to see if the orchestration succeeded.

   ```
   {prompt.root.admin}``tail -n 50 salt-run.log``
   ```

3. On a master node, validate the dex pods are running.

   ```
   {prompt.root.master}``kubectl -n kube-system get pods | grep dex``
   ```

4. If you are not able to log in into Velum, reboot the Administration Node . Then test and validate that the cluster is still functional.

13.10. Fixing “x509: certificate signed by unknown authority”

When interacting with Kubernetes you can run into the situation where your existing configuration for the authentication has changed (cluster has been rebuild, certificates have been switched.)

In such a case you might see an error message in the output of your CLI tool.

```
x509: certificate signed by unknown authority
```

This message indicates that your current system does not know the Certificate Authority (CA) that signed the SSL certificates used for encrypting the communication to the cluster. You then need to add or update the Root CA certificate in your local trust store.

For details about installing the root CA certificate, see Obtaining and Installing Root CA Certificate.

13.11. Replacing a Lost Node

If your cluster loses a node, for example due to failed hardware, remove the node as explained in Removing Nodes. Then add a new node as described in Adding Nodes.

Appendix A: GNU Licenses

This appendix contains the GNU Free Documentation License version 1.2.

GNU Free Documentation License

Copyright © 2000, 2001, 2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

2. VERBATIM COPYING

3. COPYING IN QUANTITY

present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

```
Copyright (c) YEAR YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.
A copy of the license is included in the section entitled{ldquo}GNU
Free Documentation License{rdquo}.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “ with...Texts.” line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Appendix B: Documentation Changelog

This appendix contains a curated list of changes to the SUSE CaaS Platform documentation.

January 2019

- 2019-01-15 [a31ea12] Minor wording updates for installation procedure related to ntp and autoyast
- 2019-01-15 [52c6270] Add customizing deployment config with auutoyast
- 2019-01-15 [62f4e23] Describe changing timezone on cluster
- 2019-01-15 [bd372d6] Change requirement for IP addresses to static
- 2019-01-10 [1195d8b] Backport PR#122
- 2019-01-10 [d7f5a52] Fix indentation for command examples
- 2019-01-10 [088189f] xml: admin_software: Merge the 'pkg' and 'ptf' commands
- 2019-01-09 [fa0405e] Some minor wording fixes
- 2019-01-09 [d52d43a] Add instructions for separate charts vhost
- 2019-01-09 [09cc754] Added details about registry via reverse proxy and changed some wording around configuration
- 2019-01-09 [20c9136] Add clarifications for ports, networking reqs, certificates
- 2019-01-09 [0469941] Change wording for helm chart repo webserver
- 2019-01-09 [3c012ff] Add note about rmtool certificate
- 2019-01-09 [12293d9] Fix updated helm-mirror command
- 2019-01-09 [6bf0fed] Add missing path creation in procedure
- 2019-01-09 [a970d65] Changed wording about rmtool install
- 2019-01-09 [cf48045] Fix wording and punctuation after review
- 2019-01-09 [ce94c5b] Correct name in entity definition
- 2019-01-09 [d23e41b] Remove obsolete comment, slightly tweak mirror spec requirements
- 2019-01-09 [a4a48ac] Fix some minor details

- 2019-01-09 [0b9ff32] Remove obsolete navigation command
- 2019-01-09 [a8ad72f] Fix various user prompts and unnecessary operations, fixed directory navigation
- 2019-01-09 [6667d41] Fix some user prompts
- 2019-01-09 [f14280d] Fix urls for helm chart repository
- 2019-01-09 [36ed0a9] Minor changes according to review comments
- 2019-01-09 [70b550f] Clean up registry directory configuration paths
- 2019-01-09 [d7f156f] Add various fixes after review comments
- 2019-01-09 [4938770] Add charts repo root url replacement parameter
- 2019-01-09 [c0d6252] Add various details about procedure, add checklist, fix various wordings and explanations
- 2019-01-09 [39f715c] Load registry container image
- 2019-01-09 [58ebbea] Add various fixes for changed procedure and review comments
- 2019-01-09 [12c9066] Various updates after comments, added details for update procedure
- 2019-01-09 [03d3cff] Add lots of details and descriptions about mirroring process and configuration, reworked structure
- 2019-01-09 [383bf2b] Rework some wording around registry mirroring
- 2019-01-09 [e197d4b] Rework instructions to show methods for SLE12 and SLE15 for rpm repo mirror
- 2019-01-09 [b821e89] Change docker registry port to avoid conflict w/ other services
- 2019-01-09 [3e2024a] Remove duplicated instruction
- 2019-01-09 [448d2e6] Add basic auth for external docker registry instructions
- 2019-01-09 [cc9afd6] Update registry configuration examples
- 2019-01-09 [5ff9a1b] Rework details for airgap environment, Add helm chart mirror instructions, Add new registry mirroring approach
- 2019-01-09 [0fdfa83] Change wording for helm chart transfer
- 2019-01-09 [0aaa71e] Add short form process description for airgap deployment
- 2019-01-09 [cab3746] Remove obsolete notes file

- 2019-01-09 [2f51fcc] Add some clarification, rework order
- 2019-01-09 [b780b28] Add details to container registry mirror
- 2019-01-09 [28a22bd] Add details to helm chart mirror
- 2019-01-09 [4e53578] Fix formatting with xmlformat
- 2019-01-09 [82794d2] Various improvements
- 2019-01-09 [56ec5f9] Update diagram for airgap
- 2019-01-09 [ae67d96] Move note about registry replacement
- 2019-01-09 [68e09a7] Remove obsolete notes file
- 2019-01-09 [6e92f34] Add details to helm chart mirror and registry mirror, Add overall procedure outline
- 2019-01-09 [48bba76] Add details to mirror setup, formatting
- 2019-01-09 [5d58468] Update notes file
- 2019-01-09 [f38766c] Add more details to mirrors, formatting
- 2019-01-09 [3e4fd49] Add details and structure to container registry mirror
- 2019-01-09 [6e8b616] Structure rpm mirror instructions
- 2019-01-09 [14b8fec] Add details to deployment scenarios
- 2019-01-09 [efd5d80] Add diagram for default scenario
- 2019-01-09 [16e7ca1] Update notes file
- 2019-01-09 [07ecde6] Add airgap details to deployment scenarios
- 2019-01-09 [1e984f8] Add airgap diagram
- 2019-01-09 [3edf8e9] Restructure notes
- 2019-01-09 [41f3398] Add deployment scenarios placeholder
- 2019-01-09 [5950a3a] Add notes files for airgap
- 2019-01-09 [7762f79] Fix wording for ay node installation instructions
- 2019-01-09 [b717363] Add static IP autoyast example, change obsolete command to ifcfg
- 2019-01-09 [4ec3b74] Fix image tags, minor formatting
- 2019-01-09 [77b03c1] xml: admin_software: Restore xrefs for master/minion logging

December 2018

- 2018-12-12 [dc36ace] Merge branch 'feature/user-guide' of github.com:SUSE/doc-caasp into feature/user-guide
- 2018-12-12 [576477a] Minor fixes for wording and typos, fixed upstream k8s doc entity
- 2018-12-12 [f809239] Add content to 'Using CronJob'
- 2018-12-12 [150c271] Add content to 'Using Deployment'
- 2018-12-12 [9746c92] Change 'Using kubectl'
- 2018-12-12 [2b5b49d] Change 'Using DaemonSet'
- 2018-12-12 [2045ecb] Change 'Using ReplicaSet'
- 2018-12-12 [fe560f1] Add 'Using Labels'
- 2018-12-12 [4a0d006] Change 'Managing Pods'
- 2018-12-12 [7265de2] Change 'User Guide' Intro
- 2018-12-12 [9e40c84] Change 'User Guide'
- 2018-12-12 [ba10a23] Change 'Managing Services'
- 2018-12-12 [0ad420a] Add Killing and Listing Pods
- 2018-12-12 [d2bba07] Change 'Manage Pods'
- 2018-12-12 [1b32802] Change XML ID 'sec.user.deployment.kubernetes_dashboard'
- 2018-12-12 [36c6fe6] Fix indentation 'Deploying Application'
- 2018-12-12 [41d3b5a] Change 'Gaining Cluster Access'
- 2018-12-12 [e1fa432] Add 'Managing Services' to User Guide
- 2018-12-12 [be121d9] Add User Access
- 2018-12-12 [1e7bb41] Change 'User Guide'
- 2018-12-12 [a25116b] Minor fixes for wording and typos, fixed upstream k8s doc entity
- 2018-12-07 [e89463c] Rework k8s doc links to entity
- 2018-12-06 [006ba6a] Fix details for k8s dashboard instructions
- 2018-12-04 [9329f07] Simplify uniqueMember handling for Velum pw change
- 2018-12-04 [c12e0b3] Add missing step from velum admin password

change

- 2018-12-03 [9383ebd] Add note about update timer enabled after migration

November 2018

- 2018-11-23 [4597df4] Add content to 'Using CronJob'
- 2018-11-23 [9c049b7] Add content to 'Using Deployment'
- 2018-11-23 [5c6ebb1] Change 'Using kubectl'
- 2018-11-23 [0713a4a] Fix typo in troubleshooting
- 2018-11-23 [71232d7] Add 'Replacing a Lost Node' to Troubleshooting
- 2018-11-23 [2fbf166] Change 'Using DaemonSet'
- 2018-11-22 [729e5d9] admin_administration: Fix note about single replica pods
- 2018-11-22 [b8b95e0] Change 'Using ReplicaSet'
- 2018-11-22 [899d528] Add 'Using Labels'
- 2018-11-20 [17b58b3] Add one more TOC level for deployment guide
- 2018-11-20 [d1d01d8] Update incorrect node-exporter config example
- 2018-11-19 [e399c29] Slight rewording for kubic repo
- 2018-11-19 [a34c658] Remove obsolete wording
- 2018-11-19 [b62342b] Add instructions how to add grafana dashboards
- 2018-11-16 [9509d7f] Update docs changelog
- 2018-11-16 [0a69da2] Sanitize IP addresses
- 2018-11-16 [8dcc5de] Add note about grafana dashboard store
- 2018-11-16 [c096208] Add link for testing alertmanager (#1116002)
- 2018-11-16 [1e43c8b] Remove obsolete PSP configuration, rework configure step order
- 2018-11-16 [6d42f93] Rework wording for certificate section (#1115989)
- 2018-11-16 [dca2923] Fix typo (#1115987)
- 2018-11-16 [7d2a6ae] Add back kube dashboard install example
- 2018-11-16 [6bf8d38] Add note about cAdvisor to health checks
- 2018-11-16 [cd1e05b] Remove obsolete wording, Add warning about

example nature of instructions

- 2018-11-16 [e3db364] Updated link to grafana templates
- 2018-11-16 [ec80f23] Remove obsolete sentence
- 2018-11-16 [a8f96b6] Rework grafana section, xmlformat
- 2018-11-16 [c6a7205] Rework prometheus and alertmanager sections
- 2018-11-16 [cd10578] Rework concepts, prereq, ingress and tls sections
- 2018-11-16 [6b2b1bc] Update monitoring example
- 2018-11-16 [c7d0380] Add monitoring instructions from github repo
- 2018-11-16 [d225622] Sanitize IP addresses for examples
- 2018-11-16 [37fbdd4] Replace wrong docker log level command
- 2018-11-16 [21404bf] Switch bugreports to GitHub
- 2018-11-12 [c0ebe56] Add simpler method of setting velum admin pw
- 2018-11-12 [e45b9c1] Rework structure of document
- 2018-11-12 [aa46754] Add information about openldap admin passwords
- 2018-11-12 [d9dd47c] docker command was missing
- 2018-11-12 [bca90e8] Fix typo (bsc#1115553)
- 2018-11-08 [cba6f8b] Rework docker ps commands for consistency
- 2018-11-08 [eba8b3b] Fix command
- 2018-11-07 [607a3a3] Update wording for toolchain module
- 2018-11-07 [aeff1a5] deployment: sysreqs: Fix typo when describing worker threads
- 2018-11-07 [0cda167] Add kernel toolchain installation, add missing options to transactional update description

October 2018

- 2018-10-29 [7df93ec] Change 'Configuring Remote Container Registry'
- 2018-10-29 [c8b00a0] Add 'Reserving Compute Resources'
- 2018-10-18 [b82a0dc] Update AD and openLDAP content examples
- 2018-10-18 [7a7a7fb] Add Active Directory and openLDAP examples
- 2018-10-17 [56919ed] Add note about reboot in grub transactional update
- 2018-10-17 [bbdc0ee] fix bsc#1046128

- 2018-10-16 [762309c] Add required etcdctl API version to troubleshooting examples
- 2018-10-16 [c62789b] Add note about LDAP admin group
- 2018-10-15 [e5c40e3] Remove obsolete certificate handling note for external ldap
- 2018-10-09 [95e5671] Remove unallowed tag in example output
- 2018-10-09 [2d9dee0] Update docs changelog
- 2018-10-08 [290b9fc] Rework wording for LDAP setup
- 2018-10-08 [2bf14f8] Add external LDAP configuration to security section
- 2018-10-08 [148a6fc] Move LDAP options to admin guide
- 2018-10-08 [675d4b2] Added Velum settings with LDAP Connectors to Configuration section
- 2018-10-08 [6f19156] Rewrite links to Kubernetes docs for specific version
- 2018-10-08 [ace031a] Add minor formatting fixes
- 2018-10-08 [833cfc1] Change 'Pod Security Policies'
- 2018-10-08 [cbe7cb4] Change 'Pod Security Policies'
- 2018-10-08 [f2f443e] Add 'Pod Security Policies'
- 2018-10-05 [2cbabd1] Fix typo
- 2018-10-05 [7053849] Add clarification about cert renewal
- 2018-10-05 [c191cd9] Add wording improvement for salt cluster sizing
- 2018-10-05 [4dab836] Fix wording for salt worker threads
- 2018-10-05 [7def5db] Rework wording for salt worker thread info
- 2018-10-05 [686ef12] Remove confusing entity
- 2018-10-05 [96a7425] Add salt cluster sizing reqs and installation notes
- 2018-10-05 [3b5f49f] Add manual adjustment of salt workers
- 2018-10-05 [e59892a] Add entity for salt worker
- 2018-10-04 [22d2243] Add note about master nodes during startup procedure
- 2018-10-04 [55ccfee] Add note about cloud-init network configuration and AutoYast

September 2018

- 2018-09-28 [2f83935] Remove obsolete warning
- 2018-09-28 [64bb59f] Add minor formatting fixes
- 2018-09-28 [01e8648] Update link for CA cert in troubleshooting
- 2018-09-28 [73c0acf] Add section about TLS certificates
- 2018-09-28 [1d3274a] Fix headings to title case
- 2018-09-28 [bd4dcb5] Add details for registry and mirror configuration, Update formatting
- 2018-09-28 [a84170a] Add details to registry and mirror configuration
- 2018-09-28 [f4738e2] Update and Add screenshots for registry and mirrors
- 2018-09-28 [5d0d2b5] Add screenshots for velum registry/mirror configuration
- 2018-09-24 [f7682d9] Fix minor typos and formatting
- 2018-09-24 [5d7ec89] Change 'SES Integreation'
- 2018-09-24 [6cefc89] Change 'SUSE Enterprise Storage Integration'
- 2018-09-21 [dd92566] Deployment: Use entity for OpenStack
- 2018-09-20 [fd03d5b] Change 'Upgrading from CaaSP 2'
- 2018-09-18 [6a98159] Add overlay networking, registry mirror and network proxy settings descriptions
- 2018-09-18 [c656329] Modify screenshot for CPI
- 2018-09-18 [e70a825] Format a port number
- 2018-09-18 [f4f9dce] Change 'Role Management'
- 2018-09-18 [69a0387] Add 'Role Management'
- 2018-09-18 [a5a84b7] Add 'Showing User Attributes'
- 2018-09-18 [f707c55] Change 'Managing Users and Groups'
- 2018-09-18 [e14dcb6] Change ldap client user
- 2018-09-18 [c408d7e] Add 'Deleting User'
- 2018-09-18 [3d34122] Add 'Access Control Overview'
- 2018-09-10 [3aff655] Remove obsolete message from kubectl instructions
- 2018-09-07 [30554fc] Optimize new screenshots

- 2018-09-07 [cd5d21f] Update changelog
- 2018-09-07 [7119c5c] Fix broken tag in troubleshooting guide
- 2018-09-07 [a4d2c9d] Update deployment instructions in quickstart guide
- 2018-09-07 [1b3ec85] Fix deployment instructions in deployment guide
- 2018-09-07 [de75f51] Fix typo in installation instructions
- 2018-09-07 [3526c75] Update instructions for installing local kubectl
- 2018-09-06 [388e3bd] Add note about unknown authority error (bsc#1098409)
- 2018-09-06 [017c1a2] Fix typo in Troubleshooting
- 2018-09-06 [10368ae] Add minor clarification to troubleshooting
- 2018-09-06 [c91beb2] Change Troubleshooting
- 2018-09-06 [66d8be2] Remove Troubleshooting RBAC
- 2018-09-06 [81a5d4a] Add 'Debugging Failed Bootstrap'
- 2018-09-06 [db1f8ea] Change Troubleshooting commands
- 2018-09-06 [9cacadf] Add 'Recovering from Failed Updates'
- 2018-09-06 [a79e053] Add 'Checking etcd Health'
- 2018-09-06 [3bbdc90] Add 'Locking Installed Program Temporary Fixes'
- 2018-09-06 [fa11e88] Change 'Replacing TLS/SSL Certificates'
- 2018-09-06 [0841c07] Change 'Software Management'
- 2018-09-06 [8c6c3ab] Add 'Scaling the Cluster'
- 2018-09-06 [775d9d6] Add 'Replacing TLS/SSL Certificates'
- 2018-09-06 [830da88] Add 'Using a Proxy Server with Authentication'
- 2018-09-06 [227c6f3] Change 'Troubleshooting'
- 2018-09-06 [be33ffd] Change 'Troubleshooting'
- 2018-09-06 [deb0420] Add Troubleshooting section to Admin Guide
- 2018-09-06 [d930f65] Merge branch 'feature/use-docker-native-commands' into develop
- 2018-09-06 [e32d1fa] Add 'Installing VMware Tools'
- 2018-09-06 [828db36] Change cloud-init images to .iso ending
- 2018-09-06 [4b4cb15] Change 'run vmkfstools on ESX/ESXi host'
- 2018-09-06 [376d754] Improve VMware Memory Ballooning important

- 2018-09-06 [6e6c433] Change 'Installing from Virtual Disk Images'
- 2018-09-06 [06537e0] Add 'Converting Images for VMware'
- 2018-09-06 [84d6515] Change Docker-commands
- 2018-09-06 [2521a75] Rework docker ps filter
- 2018-09-06 [dfa2ba1] Minor rewording after reviews
- 2018-09-06 [33507f7] Add details about liveness and readiness probes
- 2018-09-06 [e85d5b6] Add xref to etcd behavior in shutdown section
- 2018-09-06 [f75ae73] Add information about kubernetes master health endpoints
- 2018-09-06 [8771553] Add kubelet health check
- 2018-09-06 [051ef3e] Add more information
- 2018-09-06 [a0be694] Work comments into sections
- 2018-09-06 [ad4e7a4] Add notes for health checks
- 2018-09-04 [22c15b4] Removed obsolete sentence, added clarification of master workloads
- 2018-09-04 [3c778c6] Minor rewording for quorum explanation, removed link, various minor formatting
- 2018-09-04 [b6f46ee] Add 'Graceful Shutdown and Startup'
- 2018-09-04 [ce11207] Add note about salt interaction to admin guide
- 2018-09-04 [4cca70b] fix cloud-init kernel parameter
- 2018-09-03 [02ed542] Add september 18 CL
- 2018-09-03 [6a347e8] Add docs changelog
- 2018-09-03 [d016ab4] Update CPU requirement to 4 cores
- 2018-09-03 [a2117bf] Duplicate hardware requirements in quicktart guide
- 2018-09-03 [5079379] Update minimum node hardware requirements

August 2018

- 2018-08-31 Add logging chapter [f6ceda9]
- 2018-08-29 Reduce TOC level to 2 [5a72260]
- 2018-08-28 Add Sven Seeberg-Elverfeldt to authors [1322927]
- 2018-08-28 Merge pull request #77 from SUSE/feature/deployment-

merge-installation [92cc495]

- 2018-08-28 Change 'Network Requirements' [3f5d4e5]
- 2018-08-28 Change 'Installing from Virtual Disk Images' [632a6d5]
- 2018-08-28 Merge branch 'develop' into feature/deployment-merge-installation [0afc32c]
- 2018-08-27 Add Preface to Admin Guide [536d1d7]
- 2018-08-27 Change 'Network Requirements' [a995254]
- 2018-08-24 Add trademark signs to public cloud names [3bb3a60]
- 2018-08-24 Restructure Installing the Administration Node with CLI [c45f425]
- 2018-08-24 Restructure Preparing the Installation [0ca484d]
- 2018-08-24 Merge sections in System Requirements [8fd1670]
- 2018-08-23 Change title 'Installing in SUSE OpenStack Cloud [6c3c1ee]
- 2018-08-23 Split 'Installing in Public Cloud' [7dad328]
- 2018-08-23 Merge Deployment Guide Chapters [1f613a2]
- 2018-08-22 Fix sentence in 'Preparing Installation on Physical Machines or Private Cloud' [f6fb072]
- 2018-08-22 Merge pull request #76 from SUSE/feature/improve-deployment-guide [c5d2506]
- 2018-08-22 Change 'Preparing Installation on Physical Machines or Private Cloud' [7f7ec99]
- 2018-08-20 Add note about CNAME record for admin/master [d4374e2]
- 2018-08-17 Restructure Deployment Guide [093aa54]
- 2018-08-17 Deployment Guide: proof reading, structural and lingual improvements (#71) [f7225a5]
- 2018-08-16 Fix missing word in admin guide [cc72f85]
- 2018-08-15 Configure admin output to 3 toc levels, fix title comment for DC file [d030f93]
- 2018-08-15 Remove obsolete file [03bb4af]
- 2018-08-14 Fix indentation for LDAP command in security section [8c9b14a]
- 2018-08-14 Fix user password change LDIF replace must be used instead of modify [2ab736c]

- 2018-08-14 Add user guide (#73) [ac73795]
- 2018-08-14 Completely restructure admin guide (#72) [0b949ad]
- 2018-08-13 Add upgrade instructions for CaaSP v3 (#70) [85a5486]
- 2018-08-10 Deployment Guide: Make images in the 'About' chapter consistent [4fbe3a5]
- 2018-08-10 Merge branch 'feature/rewrite-xmlids' into develop [0566da9]
- 2018-08-10 Deployment Guide: Make xml:ids consistent and structurally correct [05888c2]
- 2018-08-10 Deployment Guide: Remove unused skeleton file (installation chapter from the Quickstart was used instead) [0228c22]
- 2018-08-10 Deployment Guide: Add missing xml:ids in the installation chapter [38c59e5]
- 2018-08-10 Admin Guide: Make xml:ids consistent and structurally correct [1952bfa]
- 2018-08-10 Quick Start: Rewrite xml:ids to avoid clashes with the Deployment Guide [22011eb]
- 2018-08-10 Rename 'Services on Nodes' diagram to it's name before the latest update. [af01e25]
- 2018-08-09 Rewrite explanation of transactional update [ca1fbeb]
- 2018-08-09 Fix product name (#69) [476cb21]
- 2018-08-08 Add instructions for cloud-init image generation [add07d8]
- 2018-08-08 Add placeholder for cloud-init iso [5742c52]
- 2018-08-08 Add Velum initial config option descriptions #66 [31eae0e]
- 2018-08-08 Updated "Services on Nodes" diagram #65 [d3a175c]
- 2018-08-08 Reword the text about the issue-generator" command & its use #61 [27376fd]
- 2018-08-08 Run xmlformat on admin_administration [f38ced0]
- 2018-08-08 Harmonise cluster size reqs between QSG & DG, & add link to relnotes for max size (BSC#1092082) (#60) [02fb96c]
- 2018-08-08 Change the intro para of the QS & intro § of the DG (#53) [643a3a3]
- 2018-08-08 First draft of an appendix on installing an admin node with AutoYast. Closes #42 [1c82f37]

- 2018-08-08 Bugfix/bsc#1051838 (#40) [f1a04cd]
- 2018-08-07 VMWare memory ballooning mandates a warning, not just a note (BSC#1095331) [8a741a4]
- 2018-08-07 Remove mention of installing guest additions (BSC#1234567) [7b50515]
- 2018-08-07 Add new Cinder zone option [014960d]
- 2018-08-07 Add node removal instructions [77dae14]
- 2018-08-02 Enable branch for Travis [49b74dd]