Getting started with Trento Premium

SUSE® Linux Enterprise Server for SAP Applications

Trento Premium is an open cloud native Web console that aims to help SAP Basis consultants and administrators to check the configuration, monitor and manage the entire OS stack of their SAP environments, including HA features.

Publication Date: 14 Oct 2024

Contents

Important: Trento is an evolving product

The product continues to be under active development and is available at no extra cost for any SUSE customer with a registered SUSE Linux Enterprise Server for SAP Applications 15 (SP1 or higher). Contact https://www.suse.com/support for further assistance.

1 What is Trento Premium?

Trento Premium is the extended version of the Trento community project containing additional checks. Apart from their naming, both are comprehensive monitoring solutions consisting of two main components, the Trento Server and the Trento Agent. Both Trento variants provide the following features:

- A clean, simple, reactive console targeting SAP admins
- Monitoring of CPU and Memory usage at host level through basic integration with Prometheus
- Discovery of Pacemaker clusters using different fencing mechanisms, including diskless SBD.
- Awareness of maintenance situations in a Pacemaker cluster at cluster, node or resource level.
- Configuration validation for SAP HANA Scale-Up Performance-optimized and ASCS/ERS clusters deployed on Azure, AWS, GCP or on-premise bare metal platforms, including KVM and Nutanix
- Insightful information about the execution of configuration checks.
- Email alerting for critical events in the monitored landscape
- Intregration of saptune into the console and specific configuration checks at host and cluster levels.
- Summary of available patches and upgradable packages for each registered hosts via integration with SUSE Manager.
- Monitoring of CPU and Memory usage at host level through basic integration with Prometheus
- API-based architecture to facilitate integration with other monitoring tools.

- Rotating API key to better protect the communication from the Trento Agent to the Trento Server.
- Housekeeping capabilities.

The **Trento Server** is an independent, distributed system, designed to run on a Kubernetes cluster or as a regular systemd stack. The Trento Server interacts with users via a Web front-end.

The **Trento Agent** is a single background process (`trento agent`) running on each host of the SAP infrastructure that is monitored.

See *Figure 1, "Architectural overview"* for additional details.

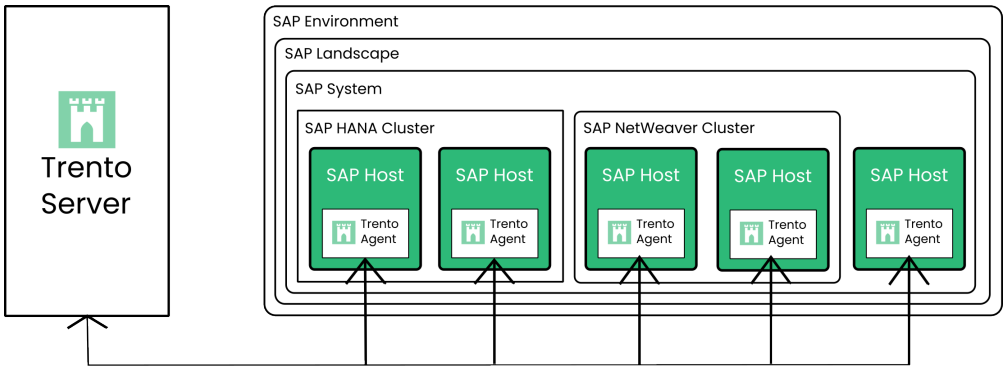


FIGURE 1: ARCHITECTURAL OVERVIEW

2 Product lifecycle and update strategy

Trento Premium is available at no extra cost for SLES for SAP subscribers. Particularly, Trento's two main components have the following product lifecycles:

Trento Agent

Delivery mechanism: RPM package for SUSE Linux Enterprise Server for SAP Applications 15 SP1 and newer.

Supported runtime: Supported in SUSE Linux Enterprise Server for SAP Applications 15 SP1 and newer.

Trento Server

Delivery mechanisms: A set of container images from the SUSE public registry together with a Helm chart that facilitates their installation or a set of RPM packages for SUSE Linux Enterprise Server for SAP Applications 15 SP3 and newer.

Kubernetes deployment: The Trento Server runs on any current Cloud Native Computing Foundation (CNCF)-certified Kubernetes distributions based on a x86_64 architecture. Depending on your background and needs, SUSE supports several usage scenarios:

- If you already use a CNCF-certified Kubernetes cluster, you can run the Trento Server in it.
- If you have no Kubernetes cluster and want enterprise support, SUSE recommends SUSE Rancher Kubernetes Engine (RKE) version 1 or 2.
- If you do not have a Kubernetes enterprise solution but you still want to try Trento Premium, SUSE Rancher's K3s provides you with an easy way to get started. But keep in mind that K3s default installation process deploys a single node Kubernetes cluster, which is not a recommended setup for a stable Trento production instance.

systemd and containerized deployments: Supported in SUSE Linux Enterprise Server for SAP Applications 15 SP3 and newer.

3 Requirements

This section describes requirements for the Trento Server and its Trento Agents.

3.1 Trento Server requirements

Running all the Trento Server components requires a minimum of 4 GB of RAM, two CPU cores and 64 GB of storage. When using K3s, such storage should be provided under `/var/lib/rancher/k3s`.

Trento is powered by event driven technology. Registered events are stored in a PostgreSQL database with a retention period of 10 days. For each host registered with Trento, you need to allocate at least 1.5GB of space in the PostgreSQL database.

Trento Server supports different deployment scenarios: Kubernetes, systemd and containerized. A Kubernetes based deployment of Trento Server is meant to be cloud native and OS agnostic. It can be done on the following services:

- RKE1 (Rancher Kubernetes Engine version 1)
- RKE2

- a Kubernetes service in a cloud provider
- any other CNCF-certified Kubernetes running on x86_64 architecture

A proper, production ready Kubernetes based deployment of Trento Server requires Kubernetes knowledge. The Helm chart is meant to be used by customers lacking such knowledge or who want to get started quickly. However, Helm chart delivers a basic deployment of the Trento Server with all the components running on a single node of the cluster.

3.2 Trento Agent requirements

The resource footprint of the Trento Agent is designed to not impact the performance of the host it runs on.

The Trento Agent component needs to interact with several low-level system components that are part of the SUSE Linux Enterprise Server for SAP Applications distribution.

Trento requires unique machine identifiers (ids) on the hosts to be registered. Therefore, if any host in your environment is built as a clone of another one, ensure that you change the machine identifier as part of the cloning process before starting the Trento Agent on it.

3.3 Network requirements

- From any Trento Agent host, the Web component of the Trento Server must be reachable via HTTP (port TCP/80) or via HTTPS (port TCP/443) if SSL is enabled.
- From any Trento Agent host, the checks engine component of the Trento Server, called Wanda, must be reachable via Advanced Message Queuing Protocol or AMQP (port TCP/5672).
- The Prometheus component of the Trento Server must be able to reach the Node Exporter in the Trento Agent hosts (port TCP/9100).
- The SAP Basis administrator needs access to the Web component of the Trento Server via HTTP (port TCP/80) or via HTTPS (port TCP/443) if SSL is enabled.

3.4 Installation prerequisites

- **Trento Server.** Access to SUSE public registry for the deployment of Trento Server premium containers, if we choose a Kubernetes based deployment. A registered SUSE Linux Enterprise Server for SAP Applications 15 (SP3 or higher) distribution, if we choose a systemd deployment. Both in the case of a containerized deployment.
- **Trento Agents.** A registered SUSE Linux Enterprise Server for SAP Applications 15 (SP1 or higher) distribution.

4 Installing Trento Server

Important: Expect changes in the installation procedure

The product is under active development. Expect changes in the described installation procedure.

4.1 Kubernetes deployment

The subsection uses the following placeholders:

- *`TRENTO_SERVER_HOSTNAME`*: the host name that will be used by the end user to access the console.
- *`ADMIN_PASSWORD`*: the password that the SAP administrator will use to access the Web console. It should have at least 8 characters.

4.1.1 Installing Trento Server on an existing Kubernetes cluster

Trento Server consists of a few components which are delivered as container images and meant to be deployed on a Kubernetes cluster. A manual deployment of these components in a production ready fashion requires Kubernetes knowledge. Customers lacking such knowledge or who want to get started quickly with Trento, can use the Trento Helm chart. This approach

automates the deployment of all the required components on a single Kubernetes cluster node. You can use the Trento Helm chart in order to install Trento Server on a existing Kubernetes cluster as follows:

1. Install Helm:

   ```
   curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 | bash
   ```

2. Connect Helm to the existing Kubernetes cluster.

3. Use Helm to install Trento Server with the Trento Helm chart:

   ```
   helm upgrade \
      --install trento-server oci://registry.suse.com/trento/trento-server \
      --set trento-web.trentoWebOrigin=TRENTO_SERVER_HOSTNAME \
      --set trento-web.adminUser.password=ADMIN_PASSWORD
   ```

   When using a Helm version lower than 3.8.0, a experimental flag must be set before the helm command:

   ```
   HELM_EXPERIMENTAL_OCI=1 helm upgrade \
      --install trento-server oci://registry.suse.com/trento/trento-server \
      --set trento-web.trentoWebOrigin=TRENTO_SERVER_HOSTNAME \
      --set trento-web.adminUser.password=ADMIN_PASSWORD
   ```

4. To verify that the Trento Server installation was successful, open the URL of the Trento Web console (`http://TRENTO_SERVER_HOSTNAME`) from a workstation located in the SAP administrator's LAN.

4.1.2 Installing Trento Server on K3s

If you do not have a Kubernetes cluster or have one but do not want to use it for Trento, SUSE Rancher's K3s provides you with an easy way to get started. All you need is a small server or VM (see *Section 3.1, "Trento Server requirements"* for minimum requirements) and follow steps in *Procedure 1, "Manually installing Trento on a Trento Server host"* to get Trento Server up and running.

Important: Deploying Trento on K3s

The following procedure deploys Trento Server on a single-node K3s cluster. Such set up is not recommended for production purposes.

PROCEDURE 1: MANUALLY INSTALLING TRENTO ON A TRENTO SERVER HOST

1. Log in to the Trento Server host.
2. Install K3s:
   - Installing as user `root`

     ```
     # curl -sfL https://get.k3s.io | INSTALL_K3S_SKIP_SELINUX_RPM=true sh
     ```

   - Installing as non-`root` user:

     ```
     > curl -sfL https://get.k3s.io | INSTALL_K3S_SKIP_SELINUX_RPM=true sh -s - --
     write-kubeconfig-mode 644
     ```

3. Install Helm as `root`:

   ```
   # curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 | bash
   ```

4. Set the `KUBECONFIG` environment variable for the same user that installed K3s:

   ```
   export KUBECONFIG=/etc/rancher/k3s/k3s.yaml
   ```

5. With the same user that installed K3s, use Helm to install Trento Server with the Helm chart:

   ```
   helm upgrade \
      --install trento-server oci://registry.suse.com/trento/trento-server \
      --set trento-web.trentoWebOrigin=TRENTO_SERVER_HOSTNAME \
      --set trento-web.adminUser.password=ADMIN_PASSWORD
   ```

   When using a Helm version lower than 3.8.0, a experimental flag must be set before the helm command:

   ```
   HELM_EXPERIMENTAL_OCI=1 helm upgrade \
      --install trento-server oci://registry.suse.com/trento/trento-server \
      --set trento-web.trentoWebOrigin=TRENTO_SERVER_HOSTNAME \
      --set trento-web.adminUser.password=ADMIN_PASSWORD
   ```

6. Monitor the creation and start-up of the Trento Server pods and wait until they are all ready and running:

   ```
   watch kubectl get pods
   ```

   All pods should be in ready and running status.

7. Log out of the Trento Server host.

8. To verify that the Trento Server installation was successful, open the URL of the Trento Web console (`http://TRENTO_SERVER_HOSTNAME`) from a workstation located in the SAP administrator's LAN.

4.1.3 Installing Trento Server on K3s running on SUSE Linux Enterprise Server for SAP Applications 15

If you choose to deploy K3s on a VM with a registered SUSE Linux Enterprise Server for SAP Applications 15 distro, you can install the package `trento-server-installer` and then execute script `install-trento-server`: it will run *Step 2* to *Step 5* in *Procedure 1, "Manually installing Trento on a Trento Server host"* for you automatically.

4.1.4 Deploying Trento Server on selected nodes

If you use a multi node Kubernetes cluster, it is possible to deploy Trento Server images on selected nodes by specifying the field `nodeSelector` in the helm upgrade command, with the following syntax:

```
HELM_EXPERIMENTAL_OCI=1 helm upgrade \
   --install trento-server oci://registry.suse.com/trento/trento-server \
   --set trento-web.trentoWebOrigin=TRENTO_SERVER_HOSTNAME \
   --set trento-web.adminUser.password=ADMIN_PASSWORD \
   --set prometheus.server.nodeSelector.LABEL=VALUE \
   --set postgresql.primary.nodeSelector.LABEL=VALUE \
   --set trento-web.nodeSelector.LABEL=VALUE \
   --set trento-runner.nodeSelector.LABEL=VALUE
```

4.1.5 Enabling email alerts

Email alerting feature notifies the SAP administrator about important changes in the SAP Landscape being monitored/observed by Trento.

Some of the reported events are:

- Host heartbeat failed
- Cluster health detected critical

- Database health detected critical
- SAP System health detected critical

This feature is disabled by default. It can be enabled at installation time or anytime at a later stage. In both cases, the procedure is the same and uses the following placeholders:

SMTP_SERVER
: The SMTP server designated to send alerting emails.

SMTP_PORT
: The port on the SMTP server.

SMTP_USER
: User name to access SMTP server.

SMTP_PASSWORD
: Password to access SMTP server.

ALERTING_SENDER
: Sender email for alerting notifications.

ALERTING_RECIPIENT
: Recipient email for alerting notifications.

The command to enable email alerting is the following:

```
HELM_EXPERIMENTAL_OCI=1 helm upgrade \
   --install trento-server oci://registry.suse.com/trento/trento-server \
   --set trento-web.trentoWebOrigin=TRENTO_SERVER_HOSTNAME \
   --set trento-web.adminUser.password=ADMIN_PASSWORD \
   --set trento-web.alerting.enabled=true \
   --set trento-web.alerting.smtpServer=SMTP_SERVER \
   --set trento-web.alerting.smtpPort=SMTP_PORT \
   --set trento-web.alerting.smtpUser=SMTP_USER \
   --set trento-web.alerting.smtpPassword=SMTP_PASSWORD \
   --set trento-web.alerting.sender=ALERTING_SENDER \
   --set trento-web.alerting.recipient=ALERTING_RECIPIENT
```

4.1.6 Enabling SSL

Ingress may be used to provide SSL termination for the Web component of Trento Server. This would allow to encrypt the communication from the agent to the server, which is already secured by the corresponding API key. It would also allow HTTPS access to the Web console with trusted certificates.

Configuration must be done in the tls section of the `values.yaml` file of the chart of the Trento Server Web component.

For details on the required Ingress setup and configuration, refer to: https://kubernetes.io/docs/concepts/services-networking/ingress/. Particularly, refer to section https://kubernetes.io/docs/concepts/services-networking/ingress/#tls for details on the secret format in the YAML configuration file.

Additional steps are required on the Agent side.

4.2 systemd deployment

A systemd installation of Trento Server, based on RPM packages, can be done manually. The latest available version of SUSE Linux Enterprise Server for SAP Applications is used as the base operating system, which is SLE 15 SP5 (https://www.suse.com/download/sles/) at the time of writing. For installations on Service Packs other than SP5, update the repository address as indicated in the respective notes provided throughout this guide.

Supported Service Packs:

- SP3
- SP4
- SP5

4.2.1 List of dependencies

- PostgreSQL (https://www.postgresql.org/)
- RabbitMQ (https://rabbitmq.com/)
- NGINX (https://nginx.org/en/)
- Prometheus (https://prometheus.io/) (optional)

4.2.2 Install Trento dependencies

4.2.2.1 Install Prometheus (Optional)

Prometheus (https://prometheus.io/) is not required to run Trento, but it is recommended, as it allows Trento to display charts for each host with useful information about the CPU load, memory, and other important metrics.

Note

If you choose not to install Prometheus, or you choose to use an existing installation, make sure that `CHARTS_ENABLED` is set to `false` in the Trento web RPM configuration file stored at `/etc/trento/trento-web`, or when it is provided to the Trento web container.

4.2.2.1.1 Option 1: Use existing installation

Minimal required Prometheus version is **2.28.0**.

If you have a existing Prometheus server (https://prometheus.io/docs/prometheus/latest/installation/), ensure to set the PROMETHEUS_URL environment variable to your Prometheus server's URL as part of the Docker command when creating the `trento-web` container or configuring the RPM packages. Use *Section 4.2.2.1.2, "Option 2: Install Prometheus using the **unsupported** PackageHub repository"* as a reference to adjust the Prometheus configuration.

4.2.2.1.2 Option 2: Install Prometheus using the **unsupported** PackageHub repository

PackageHub (https://packagehub.suse.com/) packages are tested by SUSE, but they do not come with the same level of support as the core SLES packages. Users should assess the suitability of these packages based on their own risk tolerance and support needs.

1. Enable PackageHub repository:

```
SUSEConnect --product PackageHub/15.5/x86_64
```

Note

SLE15 SP3 requires a provided Prometheus server. The version available through `SUSEConnect --product PackageHub/15.3/x86_64` is outdated and is not compatible with Trento's Prometheus configuration. Refer to *Section 4.2.2.1.1, "Option 1: Use existing installation"* for SLE 15 SP3.

Note

For SLE15 SP4 change the repository to `SUSEConnect --product PackageHub/15.4/x86_64`

2. Add the Prometheus user/group:

```
groupadd --system prometheus
useradd -s /sbin/nologin --system -g prometheus prometheus
```

3. Install Prometheus using Zypper:

```
zypper in golang-github-prometheus-prometheus
```

Note

In case the missing dependency cannot be satisfied, we have already added the Prometheus user/group. This makes it safe to proceed with the installation by choosing Solution 2: break golang-github-prometheus-prometheus

4. Change Prometheus configuration by replacing the configuration at `/etc/prometheus/prometheus.yml` with:

```
global:
  scrape_interval: 30s
  evaluation_interval: 10s

scrape_configs:
  - job_name: "http_sd_hosts"
    honor_timestamps: true
    scrape_interval: 30s
    scrape_timeout: 30s
    scheme: http
```

```
    follow_redirects: true
    http_sd_configs:
      - follow_redirects: true
        refresh_interval: 1m
        url: http://localhost:4000/api/prometheus/targets
```

Note

localhost:4000 in **url:** http://localhost:4000/api/prometheus/targets refers to the location where Trento web service is running.

5. Enable and start the Prometheus service:

```
systemctl enable --now prometheus
```

6. If firewalld is running, allow Prometheus to be accessible and add an exception to firewalld:

```
firewall-cmd --zone=public --add-port=9090/tcp --permanent
firewall-cmd --reload
```

4.2.2.2 Install PostgreSQL

The current instructions are tested with the following PostgreSQL version:

- **13.9 for SP3**
- **14.10 for SP4**
- **15.5 for SP5**

Using a different version of PostgreSQL may require different steps or configurations, especially when changing the major number. For more details, refer to the official PostgreSQL documentation (https://www.postgresql.org/docs/).

1. Install PostgreSQL server:

```
zypper in postgresql-server
```

2. Enable and start PostgreSQL server:

```
systemctl enable --now postgresql
```

4.2.2.2.1 Configure PostgreSQL

1. Start `psql` with the `postgres` user to open a connection to the database:

```
su - postgres
psql
```

2. Initialize the databases in the `psql` console:

```
CREATE DATABASE wanda;
CREATE DATABASE trento;
CREATE DATABASE trento_event_store;
```

3. Create the users:

```
CREATE USER wanda_user WITH PASSWORD 'wanda_password';
CREATE USER trento_user WITH PASSWORD 'web_password';
```

4. Grant required privileges to the users and close the connection:

```
\c wanda
GRANT ALL ON SCHEMA public TO wanda_user;
\c trento
GRANT ALL ON SCHEMA public TO trento_user;
\c trento_event_store;
GRANT ALL ON SCHEMA public TO trento_user;
\q
```

   You can exit from the `psql` console and `postgres` user.

5. Allow the PostgreSQL database to receive connections to the respective databases and users. To do this, add the following to `/var/lib/pgsql/data/pg_hba.conf`:

```
host   wanda                      wanda_user    0.0.0.0/0   md5
host   trento,trento_event_store  trento_user   0.0.0.0/0   md5
```

Note

The `pg_hba.conf` file works sequentially. This means that the rules on the top have preference over the ones below. The example above shows a permissive address range. So for this to work, the entires must be written at the top of the `host` entries. For further information, refer to the pg_hba.conf (https://www.postgresql.org/docs/current/auth-pg-hba-conf.html) documentation.

6. Allow PostgreSQL to bind on all network interfaces in `/var/lib/pgsql/data/postgresql.conf` by changing the following line:

```
listen_addresses = '*'
```

7. Restart PostgreSQL to apply the changes:

```
systemctl restart postgresql
```

4.2.2.3 Install RabbitMQ

1. Install RabbitMQ server:

```
zypper install rabbitmq-server
```

2. Allow connections from external hosts by modifying `/etc/rabbitmq/rabbitmq.conf`, so the Trento-agent can reach RabbitMQ:

```
listeners.tcp.default = 5672
```

3. If firewalld is running, add a rule to firewalld:

```
firewall-cmd --zone=public --add-port=5672/tcp --permanent;
firewall-cmd --reload
```

4. Enable the RabbitMQ service:

```
systemctl enable --now rabbitmq-server
```

4.2.2.3.1 Configure RabbitMQ

To configure RabbitMQ for a production system, follow the official suggestions in the RabbitMQ guide (https://www.rabbitmq.com/production-checklist.html).

1. Create a new RabbitMQ user:

```
rabbitmqctl add_user trento_user trento_user_password
```

2. Create a virtual host:

```
rabbitmqctl add_vhost vhost
```

3. Set permissions for the user on the virtual host:

```
rabbitmqctl set_permissions -p vhost trento_user ".*" ".*" ".*"
```

4.2.3 Install Trento using RPM packages

The `trento-web` and `trento-wanda` packages come in the supported SLES4SAP distributions by default.

Install Trento web and wanda:

```
zypper install trento-web trento-wanda
```

4.2.3.1 Create the configuration files

Both services depend on respective configuration files. They must be placed in `/etc/trento/trento-web` and `/etc/trento/trento-wanda` respectively, and examples of how to modify them are available in `/etc/trento/trento-web.example` and `/etc/trento/trento-wanda.example`.

Important: The content of `SECRET_KEY_BASE` and `ACCESS_TOKEN_ENC_SECRET` in both `trento-web` and `trento-wanda` must be the same.

Note

You can create the content of the secret variables like `SECRET_KEY_BASE`, `ACCESS_TOKEN_ENC_SECRET` and `REFRESH_TOKEN_ENC_SECRET` with `openssl` running `openssl rand -out /dev/stdout 48 | base64`

Note

Depending on how you intend to connect to the console, a working hostname, FQDN, or an IP is required in `TRENTO_WEB_ORIGIN` for HTTPS. Otherwise websockets fail to connect, causing no real-time updates in the UI.

4.2.3.2 trento-web configuration

```
# /etc/trento/trento-web
```

```
AMQP_URL=amqp://trento_user:trento_user_password@localhost:5672/vhost
DATABASE_URL=ecto://trento_user:web_password@localhost/trento
EVENTSTORE_URL=ecto://trento_user:web_password@localhost/trento_event_store
ENABLE_ALERTING=false
CHARTS_ENABLED=true
PROMETHEUS_URL=http://localhost:9090
ADMIN_USER=admin
ADMIN_PASSWORD=test1234
ENABLE_API_KEY=true
PORT=4000
TRENTO_WEB_ORIGIN=trento.example.com
SECRET_KEY_BASE=some-secret
ACCESS_TOKEN_ENC_SECRET=some-secret
REFRESH_TOKEN_ENC_SECRET=some-secret
```

Note

Note: Add `CHARTS_ENABLED=false` in Trento web configuration file if Prometheus is not installed or you do not want to use Trento's charts functionality.

The alerting system to receive email notifications (https://github.com/trento-project/web/blob/main/guides/alerting/alerting.md) can be enabled by setting `ENABLE_ALERTING` to `true` and adding the following entries:

```
# /etc/trento/trento-web
ENABLE_ALERTING=true
ALERT_SENDER=<<SENDER_EMAIL_ADDRESS>>
ALERT_RECIPIENT=<<RECIPIENT_EMAIL_ADDRESS>>
SMTP_SERVER=<<SMTP_SERVER_ADDRESS>>
SMTP_PORT=<<SMTP_PORT>>
SMTP_USER=<<SMTP_USER>>
SMTP_PASSWORD=<<SMTP_PASSWORD>>
```

4.2.3.3 trento-wanda configuration

```
# /etc/trento/trento-wanda
CORS_ORIGIN=http://localhost
AMQP_URL=amqp://trento_user:trento_user_password@localhost:5672/vhost
DATABASE_URL=ecto://wanda_user:wanda_password@localhost/wanda
PORT=4001
SECRET_KEY_BASE=some-secret
ACCESS_TOKEN_ENC_SECRET=some-secret
```

4.2.3.4 Start the services

Enable and start the services:

```
systemctl enable --now trento-web trento-wanda
```

4.2.3.5 Monitor the services

Use `journalctl` to check if the services are up and running correctly. For example:

```
journalctl -fu trento-web
```

4.2.4 Check the health status of trento web and wanda

You can check if Trento web and wanda services function correctly by accessing accessing the `healthz` and `readyz` API.

1. Check Trento web health status using `curl`:

   ```
   curl http://localhost:4000/api/readyz
   ```

   ```
   curl http://localhost:4000/api/healthz
   ```

2. Check Trento wanda health status using `curl`:

   ```
   curl http://localhost:4001/api/readyz
   ```

   ```
   curl http://localhost:4001/api/healthz
   ```

If Trento web and wanda are ready, and the database connection is set up correctly, the output should be as follows:

```
{"ready":true}{"database":"pass"}
```

4.2.5 Prepare SSL certificate for NGINX

Create or provide a certificate for NGINX (https://nginx.org/en/) to enable SSL for Trento. For detailed instructions, refer to the OpenSSL documentation (https://www.openssl.org/docs/man1.0.2/man5/x509v3_config.html).

4.2.5.1 Create a self-signed certificate

1. Generate a self-signed certificate:

Note

Adjust `subjectAltName = DNS:trento.example.com` by replacing `trento.example.com` with your domain and change the value `5` to the number of days for which you need the certificate to be valid. For example, `-days 365` for one year.

```
openssl req -newkey rsa:2048 --nodes -keyout trento.key -x509 -days 5 -out
 trento.crt -addext "subjectAltName = DNS:trento.example.com"
```

2. Move the generated `trento.key` to a location accessible by NGINX:

```
mv trento.key /etc/ssl/private/trento.key
```

3. Move the generated `trento.crt` to a location accessible by NGINX:

```
mv trento.crt /etc/ssl/certs/trento.crt
```

4.2.5.2 Create a signed certificate with Let's Encrypt using PackageHub repository

Note

Change repository if you use a Service Pack other than SP5. For example: `SUSEConnect --product PackageHub/15.3/x86_64` for SLE15 SP3, `SUSEConnect --product PackageHub/15.4/x86_64` for SLE15 SP4. Users should assess the suitability of these packages based on their own risk tolerance and support needs.

1. Add PackageHub, if it is not already added:

```
SUSEConnect --product PackageHub/15.5/x86_64
zypper refresh
```

2. Install Certbot and its NGINX plugin:

```
zypper install certbot python3-certbot-nginx
```

3. Obtain a certificate and configure NGINX with Certbot:

Note

Replace `example.com` with your domain. For more information, refer to Certbot instructions for NGINX (https://certbot.eff.org/instructions?ws=nginx&os=leap)

```
certbot --nginx -d example.com -d www.example.com
```

Note

Certbot certificates are valid for 90 days. Refer to the above link for details on how to renew certificates.

4.2.6 Install and configure NGINX

1. Install NGINX package:

```
zypper install nginx
```

2. If firewalld is running, add firewalld rules for HTTP and HTTPS:

```
firewall-cmd --zone=public --add-service=https --permanent
firewall-cmd --zone=public --add-service=http --permanent
firewall-cmd --reload
```

3. Start and enable NGINX:

```
systemctl enable --now nginx
```

4. Create a configuration file for Trento:

```
vim /etc/nginx/conf.d/trento.conf
```

5. Add the following configuration to `/etc/nginx/conf.d/trento.conf`:

```
server {
    # Redirect HTTP to HTTPS
    listen 80;
    server_name trento.example.com;
    return 301 https://$host$request_uri;
```

```
}

server {
    # SSL configuration
    listen 443 ssl;
    server_name trento.example.com;

    ssl_certificate /etc/ssl/certs/trento.crt;
    ssl_certificate_key /etc/ssl/private/trento.key;

    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-
ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-
RSA-AES256-GCM-SHA384';
    ssl_prefer_server_ciphers on;
    ssl_session_cache shared:SSL:10m;

    # Wanda rule
    location ~ ^/(api/checks|api/v1/checks|api/v2/checks|api/v3/checks)/  {
        allow all;

        # Proxy Headers
        proxy_http_version 1.1;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_set_header X-Cluster-Client-Ip $remote_addr;

        # Important Websocket Bits!
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";

        proxy_pass http://localhost:4001;
    }

    # Web rule
    location / {
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;

        # The Important Websocket Bits!
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";

        proxy_pass http://localhost:4000;
```

```
    }
}
```

6. Check the NGINX configuration:

```
nginx -t
```

   If the configuration is correct, the output should be as follows:

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

   If there are issues with the configuration, the output indicates what needs to be adjusted.

7. Reload NGINX to apply changes:

```
systemctl reload nginx
```

4.2.7 Accessing the trento-web UI

Pin the browser to `https://trento.example.com`. You should be able to login using the credentials specified in the `ADMIN_USERNAME` and `ADMIN_PASSWORD` environment variables.

4.3 Containerized deployment

A containerized deployment of Trento Server is identical to a systemd deployment. However, the web and check engine components are deployed as Docker containers.

Follow the steps in *Section 4.2, "systemd deployment"*, but skip the **Install Trento using RPM packages** step and follow the procedures in *Section 4.3.1, "Install Trento using Docker"*.

4.3.1 Install Trento using Docker

4.3.1.1 Install Docker container runtime

1. Enable the Container module:

```
SUSEConnect --product sle-module-containers/15.5/x86_64
```

Note

To use a different Service Pack than SP5, you have to choose the appropriate repository. For example, **`SUSEConnect --product sle-module-containers/15.3/x86_64`** for SLE15 SP3, **`SUSEConnect --product sle-module-containers/15.4/x86_64`** for SLE15 SP4.

2. Install Docker:

```
zypper install docker
```

3. Enable and start Docker:

```
systemctl enable --now docker
```

4.3.1.2 Create a dedicated Docker network for Trento

1. Create the Trento Docker network:

```
docker network create trento-net
```

Note

When creating the trento-net network, Docker assigns a default subnet: `172.17.0.0/16`. Ensure that this subnet matches the one specified in your PostgreSQL configuration file (refer to `/var/lib/pgsql/data/pg_hba.conf`). If the subnet of `trento-net` differs from `172.17.0.0/16` then adjust `pg_hba.conf` and restart PostgreSQL.

2. Verify the subnet of trento-net:

```
docker network inspect trento-net  --format '{{range .IPAM.Config}}{{.Subnet}}
{{end}}'
```

Expected output is as follows:

```
172.17.0.0/16
```

4.3.1.3 Install Trento on Docker

1. Create secret environment variables:

Note

Consider using an environment variable file (see official Docker documentation (https://docs.docker.com/engine/reference/commandline/run/#env)). Adjust upcoming commands for env file usage.

```
WANDA_SECRET_KEY_BASE=$(openssl rand -out /dev/stdout 48 | base64)
TRENTO_SECRET_KEY_BASE=$(openssl rand -out /dev/stdout 48 | base64)
ACCESS_TOKEN_ENC_SECRET=$(openssl rand -out /dev/stdout 48 | base64)
REFRESH_TOKEN_ENC_SECRET=$(openssl rand -out /dev/stdout 48 | base64)
```

2. Deploy trento-wanda:

```
docker run -d --name wanda \
    -p 4001:4000 \
    --network trento-net \
    --add-host "host.docker.internal:host-gateway" \
    -e CORS_ORIGIN=localhost \
    -e SECRET_KEY_BASE=$WANDA_SECRET_KEY_BASE \
    -e ACCESS_TOKEN_ENC_SECRET=$ACCESS_TOKEN_ENC_SECRET \
    -e AMQP_URL=amqp://trento_user:trento_user_password@host.docker.internal/vhost \
    -e DATABASE_URL=ecto://wanda_user:wanda_password@host.docker.internal/wanda \
    --restart always \
    --entrypoint /bin/sh \
    registry.suse.com/trento/trento-wanda:1.2.0 \
    -c "/app/bin/wanda eval 'Wanda.Release.init()' && /app/bin/wanda start"
```

3. Deploy trento-web.

   Make sure to change the `ADMIN_USERNAME` and `ADMIN_PASSWORD`, these are the credentials that are required to login to the trento-web UI. Depending on how you intend to connect to the console, a working hostname, FQDN, or an IP is required in `TRENTO_WEB_ORIGIN` for HTTPS. Otherwise websockets fail to connect, causing no real-time updates on the UI.

Note

Add `CHARTS_ENABLED=false` if Prometheus is not installed, or you do not want to use Trento's charts functionality.

```
docker run -d \
-p 4000:4000 \
--name trento-web \
--network trento-net \
--add-host "host.docker.internal:host-gateway" \
-e AMQP_URL=amqp://trento_user:trento_user_password@host.docker.internal/vhost \
-e ENABLE_ALERTING=false \
-e DATABASE_URL=ecto://trento_user:web_password@host.docker.internal/trento \
-e EVENTSTORE_URL=ecto://trento_user:web_password@host.docker.internal/
trento_event_store \
-e PROMETHEUS_URL='http://host.docker.internal:9090' \
-e SECRET_KEY_BASE=$TRENTO_SECRET_KEY_BASE \
-e ACCESS_TOKEN_ENC_SECRET=$ACCESS_TOKEN_ENC_SECRET \
-e REFRESH_TOKEN_ENC_SECRET=$REFRESH_TOKEN_ENC_SECRET \
-e ADMIN_USERNAME='admin' \
-e ADMIN_PASSWORD='test1234' \
-e ENABLE_API_KEY='true' \
-e TRENTO_WEB_ORIGIN='trento.example.com' \
--restart always \
--entrypoint /bin/sh \
registry.suse.com/trento/trento-web:2.2.0 \
-c "/app/bin/trento eval 'Trento.Release.init()' && /app/bin/trento start"
```

Email alerting are disabled by default, as described in enabling alerting (https://github.com/trento-project/web/blob/main/guides/alerting/alerting.md#enabling-alerting) guide. Enable alerting by setting `ENABLE_ALERTING` env to `true`. Additional required variables are: `[ALERT_SENDER,ALERT_RECIPIENT,SMTP_SERVER,SMTP_PORT,SMTP_USER,SMTP_PASSWORD]` All other settings should remain at their default. Example:

```
docker run -d \

...[other settings]...

-e ENABLE_ALERTING=true \
-e ALERT_SENDER=<<SENDER_EMAIL_ADDRESS>> \
-e ALERT_RECIPIENT=<<RECIPIENT_EMAIL_ADDRESS>> \
```

```
-e SMTP_SERVER=<<SMTP_SERVER_ADDRESS>> \
-e SMTP_PORT=<<SMTP_PORT>> \
-e SMTP_USER=<<SMTP_USER>> \
-e SMTP_PASSWORD=<<SMTP_PASSWORD>> \

...[other settings]...
```

4. Check that everything is running as expected:

```
docker ps
```

Expected output:

```
CONTAINER ID   IMAGE                                          COMMAND
  CREATED          STATUS          PORTS                                          NAMES
8b44333aec39   registry.suse.com/trento/trento-web:2.2.0    "/bin/sh -c '/app/bi…"
   6 seconds ago    Up 5 seconds    0.0.0.0:4000->4000/tcp, :::4000->4000/tcp
 trento-web
e859c07888ca   registry.suse.com/trento/trento-wanda:1.2.0   "/bin/sh -c '/app/bi…"
  18 seconds ago   Up 16 seconds   0.0.0.0:4001->4000/tcp, :::4001->4000/tcp   wanda
```

Both containers should be running and listening on the specified ports.

4.4 Automated deployment with Ansible

An automated installation of Trento Server using on RPM packages or Docker images can be performed with a Ansible playbook. For further information, refer to the official Trento Ansible project (https://github.com/trento-project/ansible).

4.5 Integrating Single Sign-On

Trento can be integrated with an identity provider (IDP) that uses different Single sign-on (SSO) protocols like OpenID Connect (OIDC) and Open Authorization 2.0 (OAuth 2).

Note

Trento cannot start with multiple SSO options together, so only one can be chosen.

Available protocols are:

- OpenID Connect (OIDC)
- Open Authorization 2.0 (OAuth 2)
- Security Assertion Markup Language (SAML)

4.5.1 User Roles and Authentication

User authentication is entirely managed by the IDP, which is responsible for maintaining user accounts. A user, who does not exist on the IDP, is unable to access the Trento web console.

During the installation process, a default admin user is defined using the `ADMIN_USER` variable, which defaults to `admin`. If the authenticated user's IDP username matches this admin user's username, that user is automatically granted `all:all` permissions within Trento.

User permissions are entirely managed by Trento, they are not imported from the IDP. The abilities must be granted by some user with `all:all` or `all:users` abilities (admin user initially). This means that only basic user information is retrieved from the external IDP.

4.5.2 Using OpenID Connect

Trento integrates with an IDP that uses the OIDC protocol to authenticate users accessing the Trento web console.

By default, OIDC is disabled. You can enable OIDC when using RPM packages or using Docker images.

4.5.2.1 Enabling OpenID Connect when using kubernetes deployment

To enable OIDC when using kubernetes deployment with helm, proceed as follows:

1. Add the following variables to the previously used helm installation command:

```
HELM_EXPERIMENTAL_OCI=1 helm ... --set trento-web.oidc.enabled=true
 --set trento-web.oidc.cliendId=<OIDC_CLIENT_ID> --set trento-
```

```
web.oidc.clientSecret=<OIDC_CLIENT_SECRET> --set trento-
web.oidc.baseUrl=<OIDC_BASE_URL>
```

4.5.2.2 Enabling OpenID Connect when using RPM packages

To enable OIDC when using RPM packages, proceed as follows:

1. Open the file `/etc/trento/trento-web`.
2. Add the following environment variables to this file. Required variables are:

```
ENABLE_OIDC=true
OIDC_CLIENT_ID=<OIDC_CLIENT_ID>
OIDC_CLIENT_SECRET=<OIDC_CLIENT_SECRET>
OIDC_BASE_URL=<OIDC_BASE_URL>
```

3. Optionally, add the OIDC callback URL to the configuration. This can be useful if for some reason the default callback URL cannot be used, for example, if `http` is used instead of `https`. Use the next variable for that:

```
OIDC_CALLBACK_URL=<OIDC_CALLBACK_URL>
```

4. Restart the application.

4.5.2.3 Enabling OpenID Connect when using Docker images

To enable OIDC when using Docker images, proceed as follows:

1. Provide the following environment variables to the Docker container via the `-e` option:

```
docker run -d \
-p 4000:4000 \
--name trento-web \
--network trento-net \
--add-host "host.docker.internal:host-gateway" \

...[other settings]...

# Required:
-e ENABLE_OIDC=true \
-e OIDC_CLIENT_ID=<OIDC_CLIENT_ID> \
-e OIDC_CLIENT_SECRET=<OIDC_CLIENT_SECRET> \
-e OIDC_BASE_URL=<OIDC_BASE_URL> \
```

```
# Optional:
-e OIDC_CALLBACK_URL=<OIDC_CALLBACK_URL> \

...[other settings]...
```

2. Restart the application.

4.5.2.4 Available variables for OpenID Connect

OIDC_CLIENT_ID

OIDC client id

OIDC_CLIENT_SECRET

OIDC client secret

OIDC_BASE_URL

OIDC base url

OIDC_CALLBACK_URL

OIDC callback url where the IDP is redirecting once the authentication is completed (default value: `https://#{TRENTO_WEB_ORIGIN}/auth/oidc_callback`)

4.5.3 Using OAuth 2.0

Trento integrates with an IDP that uses the OAuth 2 protocol to authenticate users accessing the Trento web console.

By default, OAuth 2.0 is disabled. You can enable OIDC when using RPM packages or using Docker images.

4.5.3.1 Enabling OAuth 2.0 when using kubernetes deployment

To enable OAuth 2.0 when using kubernetes deployment with helm, proceed as follows:

1. Add the following variables to the previously used helm installation command:

```
HELM_EXPERIMENTAL_OCI=1 helm ... --set trento-web.oauth2.enabled=true
 --set trento-web.oauth2.cliendId=<OAUTH2_CLIENT_ID> --set
 trento-web.ouath2.clientSecret=<OAUTH2_CLIENT_SECRET> --set
 trento-web.oauth2.baseUrl=<OAUTH2_BASE_URL> --set trento-
web.oauth2.authorizeUrl=<OAUTH2_AUTHORIZE_URL> --set trento-
```

```
web.oauth2.tokenUrl=<OAUTH2_TOKEN_URL> --set trento-web.oauth2.userUrl= --set
 trento-web.oauth2.scopes=<OAUTH2_USER_URL>
```

4.5.3.2 Enabling OAuth 2.0 when using RPM packages

To enable OAuth 2.0 when using RPM packages, proceed as follows:

1. Open the file `/etc/trento/trento-web`.
2. Add the following environment variables to this file. Required variables are:

```
# Required:
ENABLE_OAUTH2=true
OAUTH2_CLIENT_ID=<OAUTH2_CLIENT_ID>
OAUTH2_CLIENT_SECRET=<OAUTH2_CLIENT_SECRET>
OAUTH2_BASE_URL=<OAUTH2_BASE_URL>
OAUTH2_AUTHORIZE_URL=<OAUTH2_AUTHORIZE_URL>
OAUTH2_TOKEN_URL=<OAUTH2_TOKEN_URL>
OAUTH2_USER_URL=<OAUTH2_USER_URL>

# Optional:
OAUTH2_SCOPES=<OAUTH2_SCOPES>
OAUTH2_CALLBACK_URL=<OAUTH2_CALLBACK_URL>
```

3. Restart the application.

4.5.3.3 Enabling OAuth 2.0 when using Docker images

To enable OAuth 2.0 when using Docker images, proceed as follows:

1. Use the following environment variables to the Docker container via the `-e` option:

```
docker run -d \
-p 4000:4000 \
--name trento-web \
--network trento-net \
--add-host "host.docker.internal:host-gateway" \

...[other settings]...

-e ENABLE_OAUTH2=true \
-e OAUTH2_CLIENT_ID=<OAUTH2_CLIENT_ID> \
-e OAUTH2_CLIENT_SECRET=<OAUTH2_CLIENT_SECRET> \
-e OAUTH2_BASE_URL=<OAUTH2_BASE_URL> \
```

```
-e OAUTH2_AUTHORIZE_URL=<OAUTH2_AUTHORIZE_URL> \
-e OAUTH2_TOKEN_URL=<OAUTH2_TOKEN_URL> \
-e OAUTH2_USER_URL=<OAUTH2_USER_URL> \

# Optional:
-e OAUTH2_SCOPES=<OAUTH2_SCOPES> \
-e OAUTH2_CALLBACK_URL=<OAUTH2_CALLBACK_URL> \

...[other settings]...
```

4.5.3.4 Available variables for OAuth 2.0

OAUTH2_CLIENT_ID

OAUTH2 client id

OAUTH2_CLIENT_SECRET

OAUTH2 client secret

OAUTH2_BASE_URL

OAUTH2 base url

OAUTH2_AUTHORIZE_URL

OAUTH2 authorization url

OAUTH2_TOKEN_URL

OAUTH2 token url

OAUTH2_USER_URL

OAUTH2 token url

OAUTH2_SCOPES

OAUTH2 scopes, used to define the user values sent to the SP. It must be adjusted depending on IDP provider requirements (default value: `openid profile email`)

OAUTH2_CALLBACK_URL

OAUTH2 callback url where the IDP is redirecting once the authentication is completed (default value: `https://#{TRENTO_WEB_ORIGIN}/auth/oauth2_callback`)

4.5.4 Using SAML

Trento integrates with an IDP that uses the SAML protocol to authenticate users accessing the Trento web console. Trento will behave as a Service Provider (SP) in this case.

Commonly, SAML protocol messages are signed with SSL. This is optional using Trento, and the signing is not required (even though it is recommended). If the IDP signs the messages, and expect signed messages back, certificates used by the SP (Trento in this case) must be provided to the IDP, the public certificate file in this case.

To use an existing SAML IDP, follow the next instrunctions to met the specific requirements. You need:

1. Start Trento to generate the certificates and get them
2. Provide the generated certificate to the IDP
3. Configure SAML IDP and user profiles
4. Restart Trento

See the following subsections for details.

4.5.4.1 Getting certificates from Trento

Trento provides a certificates set created during the installation. Regardless of the installation mode, when Trento is installed the first time the certificates are created and the public certificate file content is available in the `http://localhost:4000/api/public_keys` route.

Use the following command to get the certificate content:

```
curl http://localhost:4000/api/public_keys
```

Copy the content of the certificate from there and provide it to the IDP. This way, the IDP will sign its messages and verify the messages received from Trento.

4.5.4.2 Configuring SAML IDP setup

Configure the existing IDP with the next minimum options to be able to connect with Trento as a Service Provider (SP).

4.5.4.2.1 Providing certificates

As commented previously, a set of certificates is needed to enable signed communication. Provide the certificate generated by Trento to the IDP (each IDP has a different way to do this). Make sure that the provided certificate is available in the SAML `metadata.xml` file and has preference over other uploaded certificates.

4.5.4.2.2 Configuring SAML user profile

Users provided by the SAML installation must have some few mandatory attributes to login in Trento. The required attributes are: username, email, first name and last name. All of them are mandatory, even though their field names are configurable.

By default, Trento expects the `username`, `email`, `firstName` and `lastName` attribute names. All these 4 attribute names are configurable using the next environment variables, following the same order: `SAML_USERNAME_ATTR_NAME`, `SAML_EMAIL_ATTR_NAME`, `SAML_FIRSTNAME_ATTR_NAME` and `SAML_LASTNAME_ATTR_NAME`.

Both IDP and Trento must know how these 4 fields are mapped. To do this, follow the next instructions:

1. Add the attributes if they don't exist in the IDP user profile. If they already exist, don't change the attributes and keep their original values.
2. Configure Trento to use the IDP attribute field names. To do this, set the `SAML_USERNAME_ATTR_NAME`, `SAML_EMAIL_ATTR_NAME`, `SAML_FIRSTNAME_ATTR_NAME` and `SAML_LASTNAME_ATTR_NAME` environment values with the values configured in the IDP. For example, if the IDP user profile username is defined as `attr:username` use `SAML_USERNAME_ATTR_NAME=attr:username`.

4.5.4.2.3 Checking SAML redirect URI

After a successful login, the IDP redirects the user's session back to Trento and redirected at `https://trento.example.com/sso/sp/consume/saml`. To ensure seamless SSO, this URI must be configured as valid within the IDP.

4.5.4.3 Restarting Trento

Once the certificate is provided to the IDP, the IDP recreates its own `metadata.xml` file. This file defines which certificate is used to sign the messages by both sides. At this point, Trento Web must be restarted to use the new `metadata.xml` content.

If the `SAML_METDATA_CONTENT` option is being used, the content of this variable must be updated with the new metadata as single line string. On the other hand, if `SAML_METADATA_URL` is used, the new metadata is automatically fetched. If neither of these steps are completed, communication will fail because the message signatures will not be recognized.

If the used IDP has the endpoint to provide the `metadata.xml` file content, prefer the variable `SAML_METADATA_URL`. Trento will automatically fetch metadata when restarted.

Note

This restart must be done manually, by the admin.

Follow the next instructions to restart with the configured options:

1. Open the file `/etc/trento/trento-web`.
2. Add the following environment variables to this file. Required variables are:

```
# Other SAML options
SAML_METADATA_URL=<SAML_METADATA_URL>
# Or
SAML_METDATA_CONTENT=<SAML_METADATA_CONTENT>
```

3. Restart the application.

4.5.4.4 Enabling SAML when using kubernetes deployment

To enable SAML when using kubernetes deployment with helm, proceed as follows:

1. Add the following variables to the previously used helm installation command:

```
HELM_EXPERIMENTAL_OCI=1 helm ... --set trento-web.saml.enabled=true --set trento-
web.saml.idpId=<SAML_IDP_ID> --set trento-web.saml.spId=<SAML_SP_ID> --set trento-
web.saml.metadataUrl=<SAML_METADATA_URL>
```

   To use the `SAML_METDATA_CONTENT` option rather than `SAML_METADATA_URL` use:

```
HELM_EXPERIMENTAL_OCI=1 helm ... --set trento-web.saml.enabled=true --set trento-
web.saml.idpId=<SAML_IDP_ID> --set trento-web.saml.spId=<SAML_SP_ID> --set trento-
web.saml.metadataContent=<SAML_METADATA_CONTENT>
```

   Additionally, the following optional values are available:
   HELM_EXPERIMENTAL_OCI=1 helm ... --set trento-web.saml.enabled=true --set trento-web.saml.idpId=<SAML_IDP_ID> --set trento-web.saml.spId=<SAML_SP_ID> --set trento-web.saml.metadataUrl=<SAML_METADATA_URL> --set trento-web.saml.idpNameIdFormat=<SAML_IDP_NAMEID_FORMAT> --set trento-web.saml.spDir=<SAM-

L_SP_DIR> --set trento-web.saml.spEntityId=<SAML_SP_ENTITY_ID> --set trento-web.saml.spContactName=<SAML_SP_CONTACT_NAME> --set trento-web.saml.spContactEmail=<SAML_SP_CONTACT_EMAIL> --set trento-web.saml.spOrgName=<SAML_SP_ORG_NAME> --set trento-web.saml.spOrgDisplayName=<SAML_SP_ORG_DISPLAYNAME> --set trento-web.saml.spOrgUrl=<SAML_SP_ORG_URL> --set trento-web.saml.usernameAttrName=<SAML_USERNAME_ATTR_NAME> --set trento-web.saml.emailAttrName=<SAML_EMAIL_ATTR_NAME> --set trento-web.saml.firstNameAttrName=<SAML_FIRSTNAME_ATTR_NAME> --set trento-web.saml.lastNameAttrName=<SAML_LASTNAME_ATTR_NAME> --set trento-web.saml.signRequests=<SAML_SIGN_REQUESTS> --set trento-web.saml.signMetadata=<SAML_SIGN_METADATA> --set trento-web.saml.signedAssertion=<SAML_SIGNED_ASSERTION> --set trento-web.saml.signedEnvelopes=<SAML_SIGNED_ENVELOPES>

4.5.4.5 Enabling SAML when using RPM packages

To enable SAML when using RPM packages, proceed as follows:

1. Open the file `/etc/trento/trento-web`.
2. Add the following environment variables to this file. Required variables are:

```
# Required:
ENABLE_SAML=true
SAML_IDP_ID=<SAML_IDP_ID>
SAML_SP_ID=<SAML_SP_ID>
# Only SAML_METADATA_URL or SAML_METADATA_CONTENT must by provided
SAML_METADATA_URL=<SAML_METADATA_URL>
SAML_METADATA_CONTENT=<SAML_METADATA_CONTENT>

# Optional:
SAML_IDP_NAMEID_FORMAT=<SAML_IDP_NAMEID_FORMAT>
SAML_SP_DIR=<SAML_SP_DIR>
SAML_SP_ENTITY_ID=<SAML_SP_ENTITY_ID>
SAML_SP_CONTACT_NAME=<SAML_SP_CONTACT_NAME>
SAML_SP_CONTACT_EMAIL=<SAML_SP_CONTACT_EMAIL>
SAML_SP_ORG_NAME=<SAML_SP_ORG_NAME>
SAML_SP_ORG_DISPLAYNAME=<SAML_SP_ORG_DISPLAYNAME>
SAML_SP_ORG_URL=<SAML_SP_ORG_URL>
SAML_USERNAME_ATTR_NAME=<SAML_USERNAME_ATTR_NAME>
SAML_EMAIL_ATTR_NAME=<SAML_EMAIL_ATTR_NAME>
SAML_FIRSTNAME_ATTR_NAME=<SAML_FIRSTNAME_ATTR_NAME>
SAML_LASTNAME_ATTR_NAME=<SAML_LASTNAME_ATTR_NAME>
```

```
SAML_SIGN_REQUESTS=<SAML_SIGN_REQUESTS>
SAML_SIGN_METADATA=<SAML_SIGN_METADATA>
SAML_SIGNED_ASSERTION=<SAML_SIGNED_ASSERTION>
SAML_SIGNED_ENVELOPES=<SAML_SIGNED_ENVELOPES>
```

3. Restart the application.

4.5.4.6 Enabling SAML when using Docker images

To enable SAML when using Docker images, proceed as follows:

1. Use the following environment variables to the Docker container via the `-e` option:

```
docker run -d \
-p 4000:4000 \
--name trento-web \
--network trento-net \
--add-host "host.docker.internal:host-gateway" \

...[other settings]...

-e ENABLE_SAML=true
-e SAML_IDP_ID=<SAML_IDP_ID> \
-e SAML_SP_ID=<SAML_SP_ID> \
# Only SAML_METADATA_URL or SAML_METADATA_CONTENT must by provided
-e SAML_METADATA_URL=<SAML_METADATA_URL> \
-e SAML_METADATA_CONTENT=<SAML_METADATA_CONTENT> \

# Optional:
-e SAML_IDP_NAMEID_FORMAT=<SAML_IDP_NAMEID_FORMAT> \
-e SAML_SP_DIR=<SAML_SP_DIR> \
-e SAML_SP_ENTITY_ID=<SAML_SP_ENTITY_ID> \
-e SAML_SP_CONTACT_NAME=<SAML_SP_CONTACT_NAME> \
-e SAML_SP_CONTACT_EMAIL=<SAML_SP_CONTACT_EMAIL> \
-e SAML_SP_ORG_NAME=<SAML_SP_ORG_NAME> \
-e SAML_SP_ORG_DISPLAYNAME=<SAML_SP_ORG_DISPLAYNAME> \
-e SAML_SP_ORG_URL=<SAML_SP_ORG_URL> \
-e SAML_USERNAME_ATTR_NAME=<SAML_USERNAME_ATTR_NAME> \
-e SAML_EMAIL_ATTR_NAME=<SAML_EMAIL_ATTR_NAME> \
-e SAML_FIRSTNAME_ATTR_NAME=<SAML_FIRSTNAME_ATTR_NAME> \
-e SAML_LASTNAME_ATTR_NAME=<SAML_LASTNAME_ATTR_NAME> \
-e SAML_SIGN_REQUESTS=<SAML_SIGN_REQUESTS> \
-e SAML_SIGN_METADATA=<SAML_SIGN_METADATA> \
-e SAML_SIGNED_ASSERTION=<SAML_SIGNED_ASSERTION> \
-e SAML_SIGNED_ENVELOPES=<SAML_SIGNED_ENVELOPES> \
```

```
...[other settings]...
```

4.5.4.7 Available variables for SAML

SAML_IDP_ID

SAML IDP id

SAML_SP_ID

SAML SP id

SAML_METADATA_URL

URL to retrieve the SAML metadata xml file. One of `SAML_METADATA_URL` or `SAML_METADATA_CONTENT` is required

SAML_METADATA_CONTENT

One line string containing the SAML metadata xml file content (`SAML_METADATA_URL` has precedence over this)

SAML_IDP_NAMEID_FORMAT

SAML IDP name id format, used to interpret the attribute name. Whole urn string must be used (default value: `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified`)

SAML_SP_DIR

SAML SP directory, where SP specific required files (such as certificates and metadata file) are placed (default value: `/etc/trento/trento-web/saml`)

SAML_SP_ENTITY_ID

SAML SP entity id. If it is not given, value from the metadata.xml file is used

SAML_SP_CONTACT_NAME

SAML SP contact name (default value: `Trento SP Admin`)

SAML_SP_CONTACT_EMAIL

SAML SP contact email (default value: `admin@trento.suse.com`)

SAML_SP_ORG_NAME

SAML SP organization name (default value: `Trento SP`)

SAML_SP_ORG_DISPLAYNAME

SAML SP organization display name (default value: `SAML SP build with Trento`)

SAML_SP_ORG_URL

SAML SP organization url (default value: `https://www.trento-project.io/`)

SAML_USERNAME_ATTR_NAME

SAML user profile "username" attribute field name. This attribute must exist in the IDP user (default value: `username`)

SAML_EMAIL_ATTR_NAME

SAML user profile "email" attribute field name. This attribute must exist in the IDP user (default value: `email`)

SAML_FIRSTNAME_ATTR_NAME

SAML user profile "first name" attribute field name. This attribute must exist in the IDP user (default value: `firstName`)

SAML_LASTNAME_ATTR_NAME

SAML user profile "last name" attribute field name. This attribute must exist in the IDP user (default value: `lastName`)

SAML_SIGN_REQUESTS

Sign SAML requests in the SP side (default value: `true`)

SAML_SIGN_METADATA

Sign SAML metadata documents in the SP side (default value: `true`)

SAML_SIGNED_ASSERTION

Require to receive SAML assertion signed from the IDP. Set to false if the IDP doesn't sign the assertion (default value: `true`)

SAML_SIGNED_ENVELOPES

Require to receive SAML envelopes signed from the IDP. Set to false if the IDP doesn't sign the envelopes (default value: `true`)

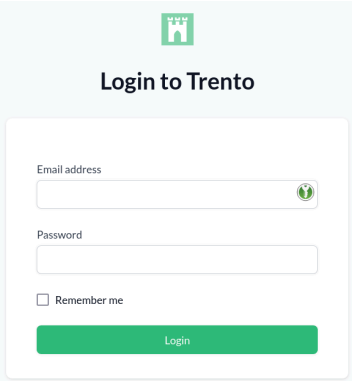
5 Installing Trento Agents

Important: Expect changes in the installation procedure

The product is under active development. Expect changes in the described installation procedure.

Before you can install a Trento Agent, retrieve the API key of your Trento Server. Proceed as follows:

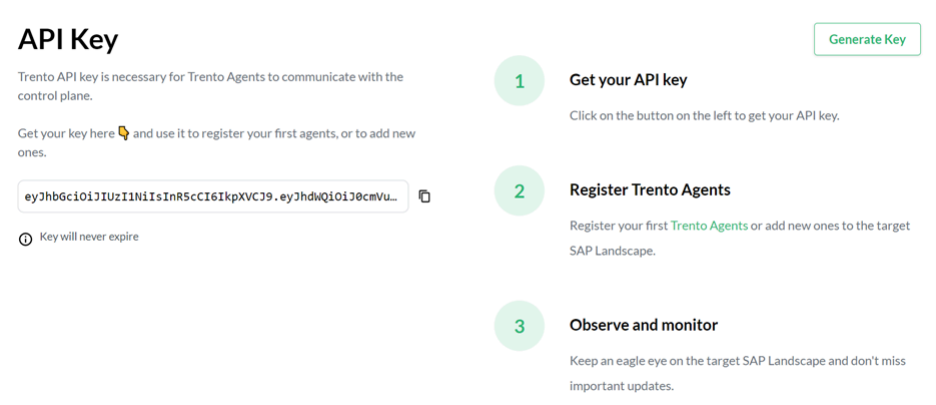
1. Open the URL of the Trento Web console (`http://TRENTO_SERVER_HOSTNAME`). It prompts you for a user name and password:



2. Enter the credentials for the `admin` user (established when installing Trento Server).

3. Click *Login*. As this is the first time you access the console, you will be prompted to accept the license agreement. Click *Accept* to continue. Otherwise, you cannot use Trento.

4. Once inside the console, go to Settings:



5. Click the *Copy* button to copy the key to your clipboard.

To install the Trento Agent on an SAP host and register it with the Trento Server, repeat the steps in *Procedure 2, "Installing Trento Agents"*:

PROCEDURE 2: INSTALLING TRENTO AGENTS

1. Install the package:

```
> sudo zypper ref
> sudo zypper install trento-agent
```

2. Open the configuration file `/etc/trento/agent.yml` and uncomment (use `#`) the entries for `facts-service-url`, `server-url` and `api-key`. Update the values appropriately:

   - `facts-service-url`: the address of the AMQP service shared with the checks engine, where fact gathering requests are received. The right syntax is `amqp://trento:trento@TRENTO_SERVER_HOSTNAME:5672`.
   - `server-url`: URL for the Trento Server (`http://TRENTO_SERVER_HOSTNAME`)
   - `api-key`: the API key retrieved from the Web console

3. If SSL termination has been enabled on the server side (refer to *Section 4.1.6, "Enabling SSL"*), you can encrypt the communication from the agent to the server as follows:

   - Provide an HTTPS URL instead of an HTTP one.
   - Import the certificate from the CA that has issued your Trento Server SSL certificate into the Trento Agent host as follows:

     1. Copy the CA certificate in PEM format to `/etc/pki/trust/anchors/`. If your CA certificate is in CRT format, convert it to PEM using the **`openssl`** command as follows:

        ```
        openssl x509 -in mycert.crt -out mycert.pem -outform PEM
        ```

     2. Run the **`update-ca-certificates`** command.

4. Start the Trento Agent:

```
> sudo systemctl enable --now trento-agent
```

5. Check the status of the Trento Agent:

```
> sudo systemctl status trento-agent
```

```
● trento-agent.service - Trento Agent service
     Loaded: loaded (/usr/lib/systemd/system/trento-agent.service; enabled; vendor
 preset: disabled)
     Active: active (running) since Wed 2021-11-24 17:37:46 UTC; 4s ago
   Main PID: 22055 (trento)
      Tasks: 10
     CGroup: /system.slice/trento-agent.service
             ├─22055 /usr/bin/trento agent start --consul-config-dir=/srv/consul/
consul.d
             └─22220 /usr/bin/ruby.ruby2.5 /usr/sbin/SUSEConnect -s

[...]
```

6. Repeat this procedure in all SAP hosts that you want to monitor.

6 Performing configuration checks

One of Trento's main features is to provide configuration checks that ensure your infrastructure setup adheres to our Best Practices, or other vendor's Best Practices, and does not drift away in time. Configuration checks ara available for HANA clusters, ASCS/ERS clusters and hosts. The following procedure is specific to a HANA cluster. The procedure for an ASCS/ERS cluster or a host would be exactly the same, only starting from the corresponding Details view:

PROCEDURE 3: PERFORMING CONFIGURATION CHECKS ON A HANA CLUSTER

1. Log in to Trento
2. In the left panel, click *Cluster*.
3. In the list, search for an SAP HANA cluster.
4. Click the respective cluster name in the *Name* column. The *Details* view opens.

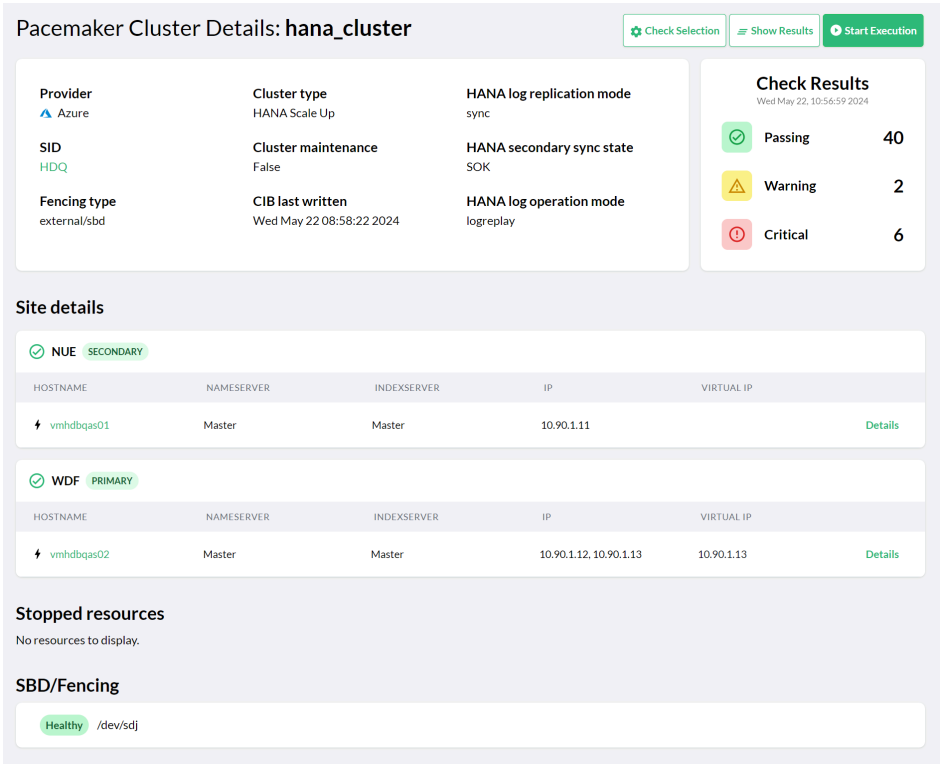


FIGURE 2: PACEMAKER CLUSTER DETAILS

5. Click the *Settings* button to change the cluster settings of the respective cluster. For checks to be executed, a checks selection must be made. Select the checks to be executed and click the button *Select Checks for Execution*. See *Figure 3, "Pacemaker Cluster Settings—Checks Selection"*:

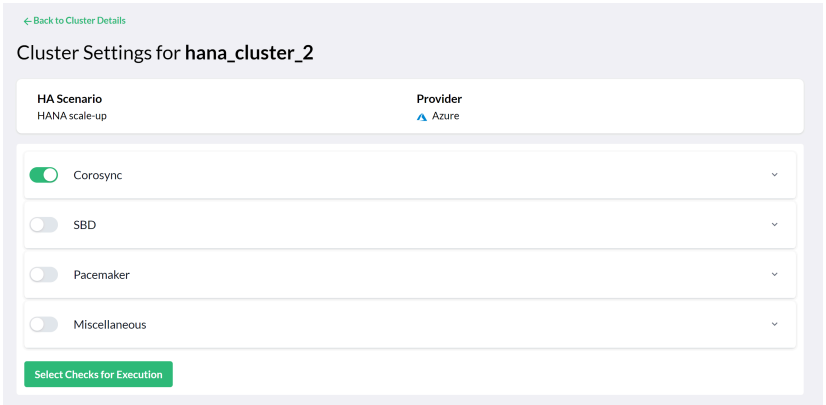


FIGURE 3: PACEMAKER CLUSTER SETTINGS—CHECKS SELECTION

6. At this moment, you can either wait for Trento to execute the selected checks or trigger an execution immediately by clicking the button that has appeared in the Checks Selection tab.

7. Investigate the result in the Checks Results view. Each row in this view shows you a check ID, a short description of the check and the check execution result. By clicking on a row, a section will open providing information about the execution on each node of the cluster. See *Figure 4, "Check results for a cluster"*:

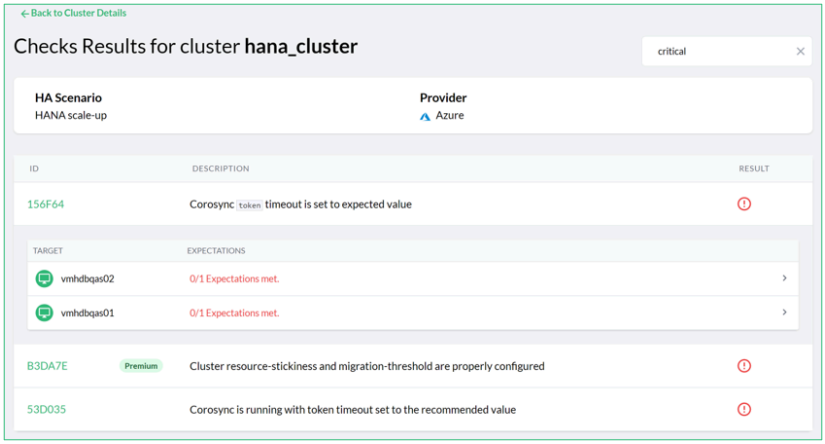


FIGURE 4: CHECK RESULTS FOR A CLUSTER

The result of a check execution can be passing, warning or critical:

- *Passing* means that the checked configuration meets the recommendation.
- *Warning* means that the recommendation is not met but the configuration is not critical for the proper running of the cluster.
- *Critical* means that either the execution itself errored (for example, a timeout) or the recommendation is not met and is critical for the well-being of the cluster.

Use the filter to reduce the list to only show, for example, critical results.

8. Click a check's link to open the *Details* view of this check. This shows you an abstract and how to remedy the problem. The *References* section contains links to the documentation of the different vendors to provide more context when necessary. Close the view with the `Esc` key or click outside of the view.

For each not-met expectation, there will be a detailed view providing information about it: what facts were gathered, what values were expected and what was the result of the evaluation. This will help users understand better why a certain configuration check is failing:

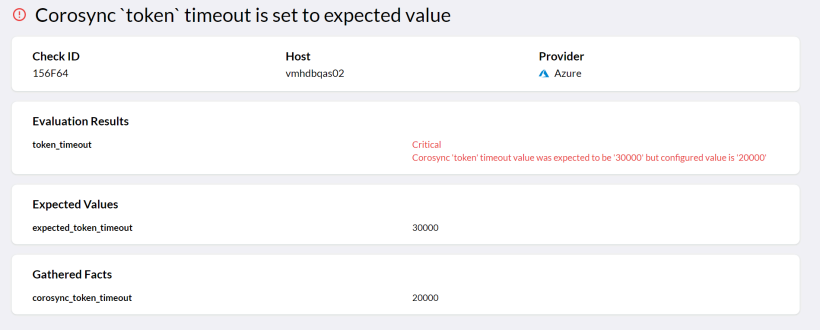


FIGURE 5: NON-MET EXPECTATION DETAIL VIEW

Once a check selection for a given cluster has been made, Trento executes them automatically every five minutes, and results are updated accordingly. A check execution result icon spinning means that an execution is running.

7 Using Trento Web console

When you access the Trento Web console for the first time, it asks to accept the license. Click *Accept* to continue.

After you have accepted the license, Trento can be used. The left sidebar in the Trento Web console contains the following entries:

- ***Dashboard***. Determine at a glance the health status of your SAP environment.
- ***Hosts***. Overview of all registered hosts running the Trento Agent.
- ***Clusters***. Overview of all discovered Pacemaker clusters.
- ***SAP Systems***. Overview of all discovered SAP Systems; identified by the corresponding system IDs.
- ***HANA Databases***. Overview of all discovered SAP HANA databases; identified by the corresponding system IDs.

- ***Checks catalog***. Overview of the catalog of configuration checks that Trento may perform on the different targets (hosts or clusters), cluster types (HANA scale up, HANA scale out or ASCS/ERS) and supported platforms (Azure, AWS, GCP, Nutanix, on-premises/KVM or VMware).
- ***Settings***. Lets you retrieve and operate the API key for this particular installation. This is required for the Trento Agent configuration, as well as managing the connection data for a SUSE Manager instance when one is available.
- ***About***. Shows the Trento flavor, the current server version, a link to the GitHub repository of the Trento Web component, and the number of registered SUSE Linux Enterprise Server for SAP Applications subscriptions that has been discovered.

7.1 Getting the global health state

The dashboard allows you to determine at a glance the health status of your SAP environment. It is the home page of the Trento Web console, and you can go back to it any time by clicking on *Dashboard* in the left sidebar.

The health status of a registered SAP system is the compound of its health status at three different layers representing the SAP architecture:

- ***Hosts***: it reflects the heartbeat of the Trento Agent and, when applicable, the compliance status.
- ***Pacemaker Clusters***: the status here is based on the running status of the cluster and the results of the configuration checks.
- ***Database***: it collects the status of the HANA instances as returned by `sapcontrol`.
- ***Application instances***: it summarizes the status of the application instances as returned by `sapcontrol`.

Complementing the operating system layer, you also have information about the health status of the HA components, whenever they exit:

- **Database cluster**: the status here is based on the running status of the database cluster and the results of the selected configuration checks.
- **Application cluster**: the status here is based on the running status of the ASCS/ERS cluster and, eventually, the results of the selected configuration checks.

The dashboard groups systems in three different health boxes (see *Figure 6, "Dashboard with the global health state"*):

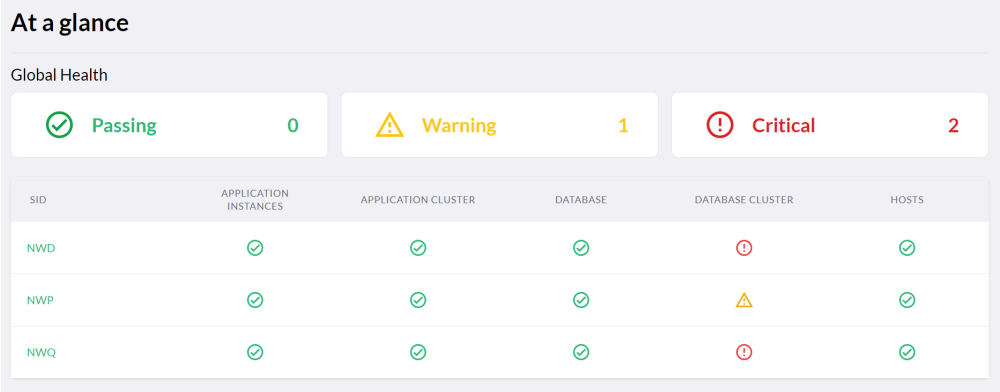


FIGURE 6: DASHBOARD WITH THE GLOBAL HEALTH STATE

THE THREE DIFFERENT HEALTH STATES

Passing

It shows the number of systems with all layers in passing (green) status.

Warning

It shows the number of systems with at least one layer in warning (yellow) status and the rest in passing (green) status.

Critical

It shows the number of systems with at least one layer in critical (red) status.

The health boxes in the dashboard are clickable. By clicking on one particular box, you filter the dashboard by systems with the corresponding health status. In large SAP environments, this feature facilitates the SAP administrator knowing which systems are in a given status.

The icons representing the health summary of a particular layer contain links to the views in the Trento console that can help determine where an issue is coming from:

- ***Hosts health icon***: Link to the Hosts overview, filtered by SID equal to the SAPSID and the DBSID of the corresponding SAP system.
- ***Database cluster health icon***: Link to the corresponding SAP HANA Cluster Details view.
- ***Database health icon***: Link to the corresponding HANA Database Details view.
- ***Application cluster health icon***: Link to the corresponding ASCS/ERS Cluster Details view.
- ***Application Instances health icon***: Link to the corresponding SAP System Details view.

A grey status is returned when either a component does not exist, or it is stopped (as returned by `sapcontrol`), or its status is unknown (for instance, if a command to determine the status fails). Grey statuses are not yet counted in the calculation of the global health status.

7.2 Viewing the status

Getting to know the status of a system is an important task for every administrator. The status allows you to see if some of your systems need to be examined further.

The following subsection gives you an overview of specific parts of your SAP Landscape to show their state. Each status site shows you an overview of the health states (see details in *The three different health states*).

7.3 Viewing the status of hosts

To display the lists of registered hosts and their details, proceed as follows:

1. Log in to the Trento Web console.
2. Click the *Hosts* entry in the left sidebar to show a summary of the state for all hosts (see *Figure 7, "Hosts entry"*).

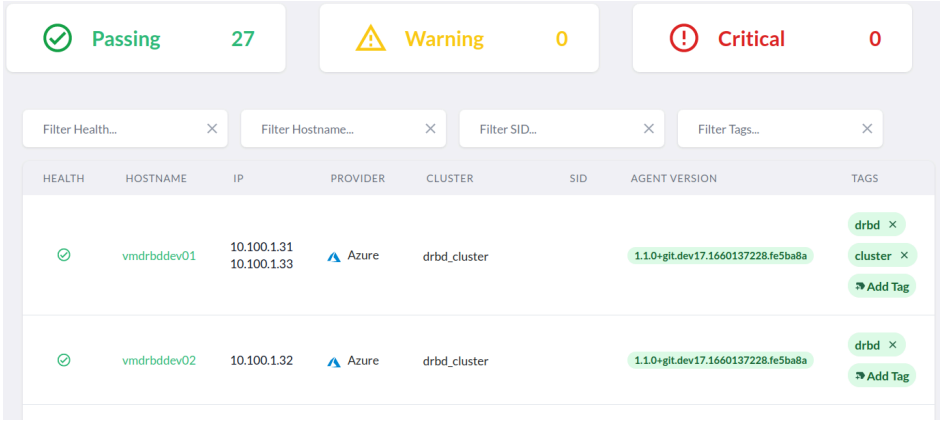


FIGURE 7: HOSTS ENTRY

3. To investigate the specific host details, click the host name in the respective column to open the corresponding *Host details* view. If the list is too big, reduce the list by providing filters.

Clicking on a host name opens the corresponding *Host details* view. This view provides the following:

HOST DETAILS VIEW

- *Hosts details* section: it shows the status of both the Trento Agent and the Node Exporter and provides the host name, the cluster name (when applicable), the Trento Agent version and the host IP addresses.
- saptune summary section: saptune is a solution that comes with SUSE Linux Enterprise Server for SAP Applications and allows SAP administrators to ensure that their SAP hosts are properly configured to run the corresponding SAP workloads. The integration of saptune in the Trento console gives the SAP administrator visibility over the tool even when they are not working at operating system level. The integration supports saptune 3.1.0 and higher, and includes the addition of the host tuning status in the aggregated health status of the host.

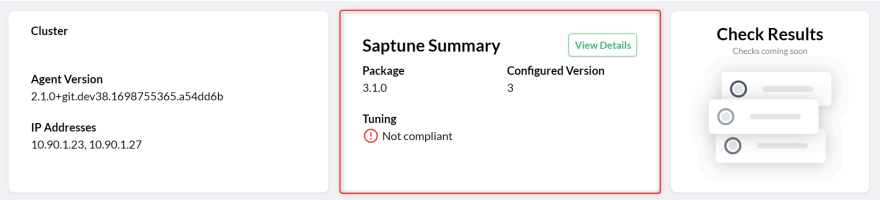


FIGURE 8: SAPTUNE SUMMARY SECTION

If an SAP workload is running on the host but no saptune or a version lower than 3.1.0 is installed, a warning is added to the aggregated health status of the host. When saptune version 3.1.0 or higher, is installed, a details view is also available showing detailed information about the saptune status:

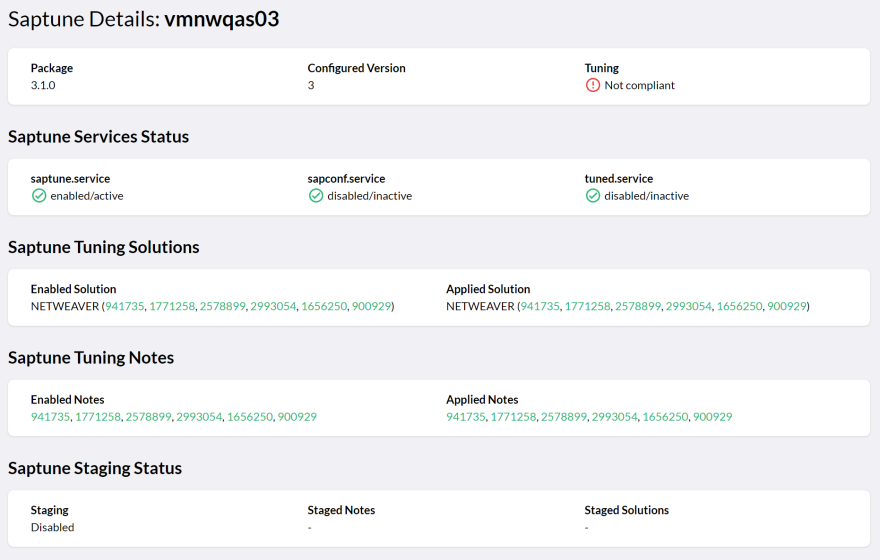
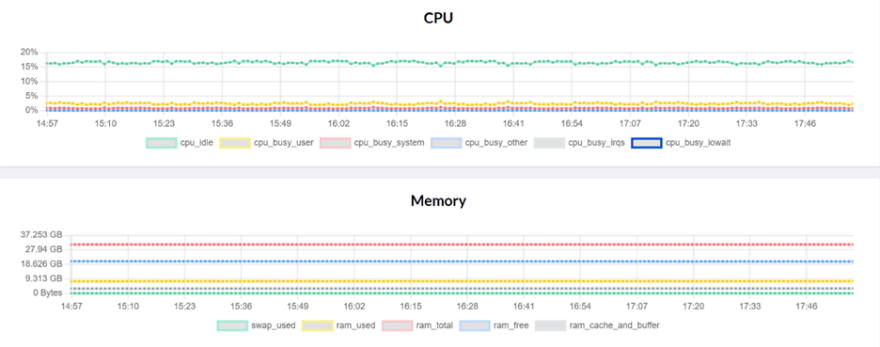


FIGURE 9: SAPTUNE DETAILS VIEW

- Checks results summary section: this section shows a summary of the checks execution results for this particular host.
- Available software updates sections: when settings for SUSE Manager are maintained and the host is managed by the SUMA instance for which connection data has been provided, this section shows a summary of the available patches and upgradable packages for this particular host. Refer to section *Section 10, "Integration with SUSE Manager"* for further details.
- *Monitoring dashboard*: it shows the CPU and Memory usage for the specific hosts.



- *Provider details* section: when applicable, it shows the name of the cloud provider, the name of the virtual machine, the name of the resource group it belongs to, the location, the size of the virtual machine, and other information.
- *SAP instances* section: when applicable, it lists the ID, SID, type, features, and instance number of any SAP instance running on the host (SAP NetWeaver or SAP HANA).
- *SUSE subscription details* section: it lists the different components or modules that are part of the subscription, and for each one of them, it provides the architecture, the version and type, the registration and subscription status as well as the start and end dates of the subscription.

7.4 Viewing the Pacemaker cluster status

To display a list of all available Pacemaker clusters and their details, proceed as follows:

1. Log in to the Trento Web console.
2. Click the *Clusters* entry in the left sidebar to show a summary of the state for all Pacemaker clusters (see *Figure 10, "Pacemaker clusters"*).

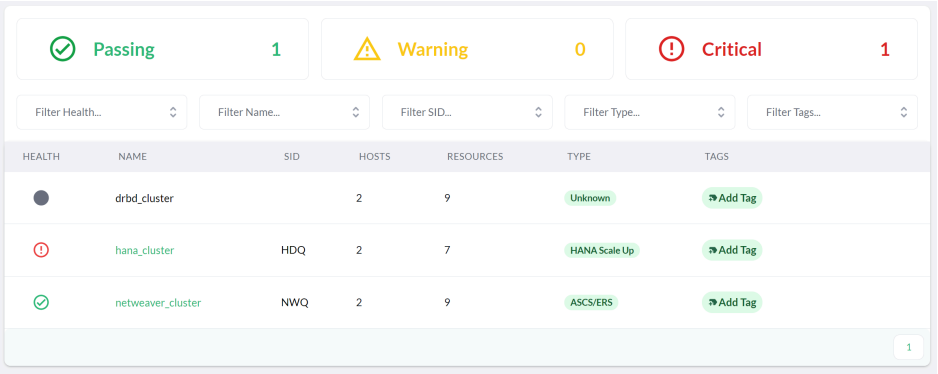


FIGURE 10: PACEMAKER CLUSTERS

3. To investigate the specific Pacemaker cluster details, click the cluster name in the respective column to open the corresponding *Pacemaker cluster details* view. If the list is too big, reduce the list by providing filters.
   The detail views of a HANA cluster and an ASCS/ERS cluster are different:

   HANA CLUSTER DETAILS VIEW

   - The *Settings*, *Show Results*, and *Start Execution* buttons. These buttons are used to enable or disable checks and to start them. If you want to perform specific checks, proceed with *Step 5* of *Procedure 3, "Performing configuration checks on a HANA Cluster"*.
   - Top section: it displayes the cloud provider, the cluster type, the HANA log replication mode, the DBSID, the cluster maintenance status, the HANA secondary sync state, the fencing type, when the CIB was last written, and the HANA log operation mode.
   - *Check Result* section: with a summary of the health of the clusters based on the runtime status and the selected configuration checks for each one of them (passed, warning and critical).
   - *Pacemaker Site details* section: this section is split in three subsections. One for each HANA site and another ones for cluster nodes without a HANA workload. For example, a majority maker in the case of a HANA scale out cluster. Each HANA site subsection will inform about the site role (Primary or Secondary or Failed) and will list the different nodes in the site. Each node entry will display the node status (Online or Maintenance or Other), the roles of the nameserver and indexserver services in

that node, the local IPs and any assigned virtual IP address. If you click the *Details* button, you can view the attributes of that node, the resources running on it and their statuses. Close this view with the `Esc` key.

- *Stopped resources* section: with a summary of resources which have been stopped on the cluster.
- *SBD/Fencing* section: with the status of each SBD device when applicable.

ASCS/ERS CLUSTER DETAILS VIEW

- A top section on the left showing the cloud provider, the cluster type, fencing type, when the CIB was last written and the cluster maintenance status.
- Another top centered section showing the SAP SID, the Enqueue Server version, whether the ASCS and ERS are running on different hosts or not and whether the instance filesystems are resource based or not. This is a multi-tab section. When multiple systems share the same cluster, there will be a tab for each system in the cluster and you can scroll left and right to go through the different systems.
- A *Check Result* section, showing a summary of the results of the last check execution, when applicable.
- A *Node details* section showing the following for each node in the cluster: the node status (Online or Maintenance or Other), the host name, the role of the node in the cluster, the assigned virtual IP address and, in case of resource managed filesystems, the full mounting path. If you click on *Details*, you can view the attributes and resources associated to that particular node. Close this view with the `Esc` key.
  This section is system specific. It will show the information corresponding to the system selected in the multi-tab section above.
- A *Stopped resources* section: with a summary of resources which have been stopped on the cluster.
- *SBD/Fencing* section: with the status of each SBD device when applicable.

7.5 Viewing the SAP Systems status

To display a list of all available SAP Systems and their details, proceed as follows:

1. Log in to the Trento Web console.

2. To investigate the specific SAP Systems details, click the *SAP Systems* entry in the left sidebar to show a summary of the state for all SAP Systems (see *Figure 11, "SAP Systems"*).

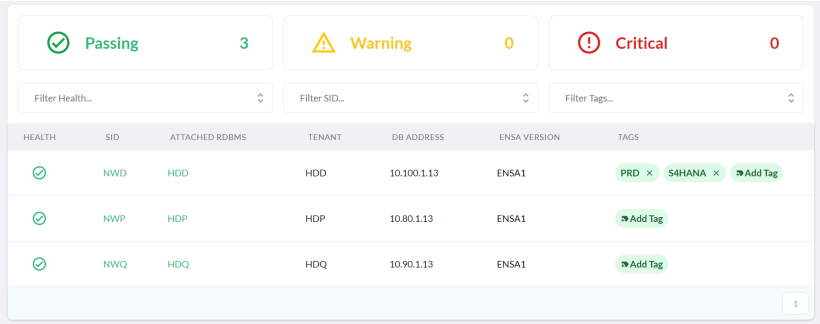


FIGURE 11: SAP SYSTEMS

3. Decide which part you want to examine. You can view the details of the *SID* column of an entry, or you go with *Attached RDBMS* to see the details of the database. If the list is too big, reduce the list by providing filters.
   If you click on an entry in the *SID* column, the *SAP System Details* view opens. This view provides the following:

   SAP SYSTEM DETAILS

   - The *Name* and *Type* of this SAP System.
   - *Layout* section: for each instance, virtual host name, instance number, features (processes), HTTP and HTTPS ports, start priority, and SAPControl status.
   - *Hosts* section: with the host name, the IP address, the cloud provider (when applicable), the cluster name (when applicable), and the Trento Agent version for each listed host. When you click the host name, it takes you to the corresponding *Host details view*.

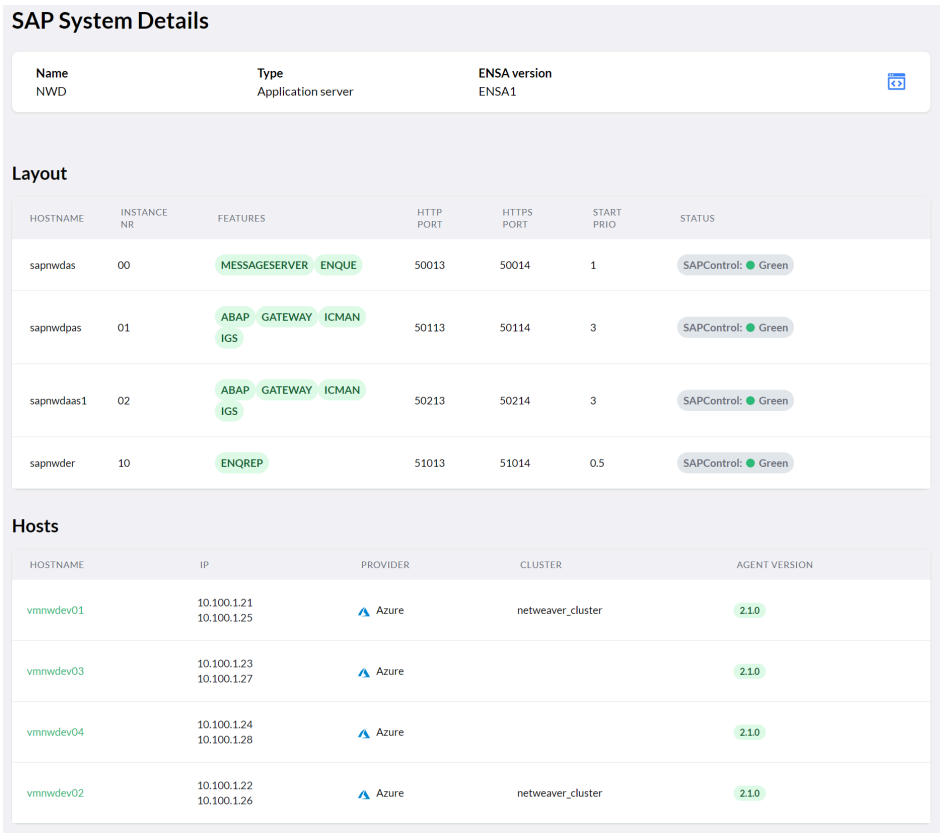


FIGURE 12: SAP SYSTEM DETAILS

7.6 Viewing the SAP HANA database status

To display a list of all available SAP HANA databases and their details, proceed as follows:

1. Log in to the Trento Web console.
2. Click the *HANA databases* entry in the left sidebar to show a summary of the state for all SAP HANA databases (see *Figure 13, "HANA databases"*).

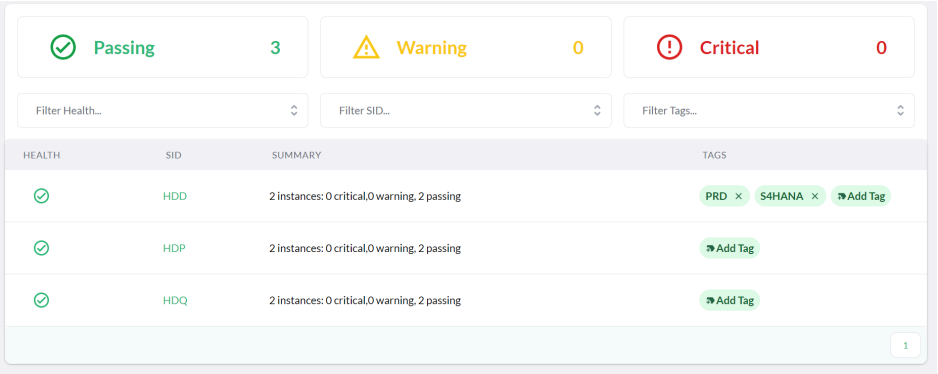


FIGURE 13: HANA DATABASES

3. To investigate the specific SAP HANA database details, click the respective name in the *SID* column to open the corresponding *HANA Database details* view. If the list is too big, reduce the list by providing filters.
   Clicking on one of the *SID*s opens the *HANA Databases* detail view. This view provides the following:

   ### *HANA DATABASE* DETAILS VIEW

   - The *Name* and *Type* of this SAP System.
   - *Layout* section: lists all related SAP HANA instances with their corresponding virtual host name, instance number, features (roles), HTTP/HTTPS ports, start priorities, and SAPControl status.
   - *Hosts* section: lists the hosts where all related instances are running. For each host, it provides the host name, the local IP address(es), the cloud provider (when applicable), the cluster name (when applicable), the system ID, and the Trento Agent version.
     Clicking on a host name takes you to the corresponding *Host details view*.

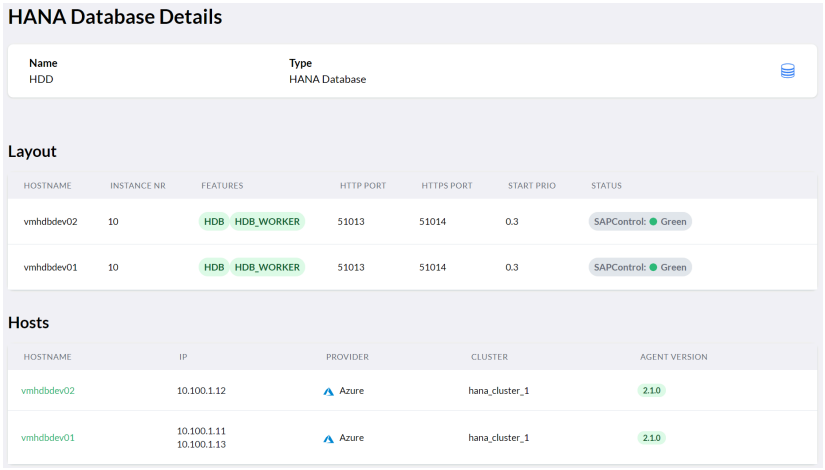


FIGURE 14: HANA DATABASE DETAILS

8 Housekeeping

When the heartbeat of an agent fails, an option to clean-up the corresponding host will show up both in the *Hosts* overview and the corresponding *Host details* view.

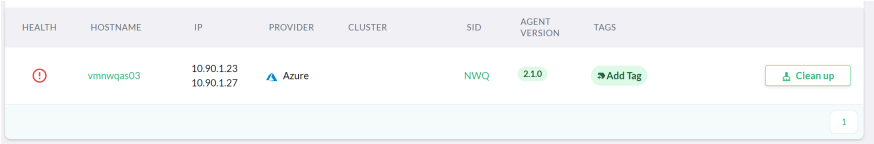


FIGURE 15: CLEAN UP BUTTON IN *HOSTS* OVERVIEW

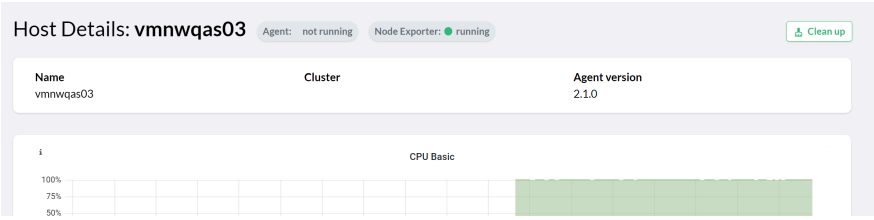


FIGURE 16: CLEAN UP BUTTON IN *HOST DETAILS* VIEW

When the user clicks on the cleanup button, a box will open asking for confirmation. If the user grants it, all the components discovered by the agent in that particular host, including the host itself and other components that might depend on the ones running on the host, will be removed from the console. For example, if the user cleans up the host where the primary application server of an SAP System is registered, the entire SAP System will be removed from the console.

Similarly, when a registered application or SAP HANA instance is no longer discovered, an option to clean it up will show up both in the corresponding overview and the corresponding details view.

FIGURE 17: CLEAN UP BUTTON SAP SYSTEMS OVERVIEW

FIGURE 18: CLEAN UP BUTTON IN SAP SYSTEM DETAILS VIEW

When the user clicks the cleanup button, a box will open asking for confirmation. If the user grants it, the instance will be removed from the console along with any possible dependencies For example, if the user cleans up the ASCS instance of an SAP system, the entire SAP system will be removed from the console.

9 Managing tags

Tags are a means to label specific objects with a purpose, location, owner, or any other property you like. Such objects can be hosts, clusters, databases, or SAP systems. Tags make it easier to distinguish and show all these different objects, making your lists more readable and searchable. You can use any text you like to create your tags except blank spaces and special characters other than + - = . , _ : and @.

The following subsection shows how you can add, remove, and filter objects based on your tags.

9.1 Adding tags to hosts, clusters, databases, and SAP Systems

To add one or more tags to your objects, do the following:

1. Log in to Trento.
2. In the Trento dashboard, go to the overview corresponding to the object for which you want to create tags. For example, the *Hosts* overview.
3. In the *Hosts* overview, search for the host you want to add a tag to.
4. In the *Tags* column, click the *Add Tag* entry.
5. Enter the respective tag and finish with `Enter` .
6. Repeat this procedure to assign as many tags as you want. It can be on the same or on a different host.

After you have finished the above procedure, the changed host contains one or more tags.

The same principle applies to other objects in Trento, be it *Clusters*, *SAP Systems*, or *HANA Databases*.

9.2 Removing tags

If you want to remove existing tags, click the respective part in the dashboard:

1. Log in to Trento.
2. In the Trento dashboard, go to the overview corresponding to the object for which you want to remove tags. For example, the *Hosts* overview.
3. In the *Hosts* overview, search for the host you want to remove a tag from.
4. In the *Tags* column, click the × icon to remove the tag.
5. Repeat this procedure to remove as many tags as you want. It can be on the same or on a different host.

9.3 Filter tags

With tags, you can show only those objects you are interested in. For example, if you have created a `drbd` tag for some hosts, you can show only those hosts that have this `drbd` tag assigned.

1. In the Trento dashboard, go to the respective overview you are interested in.
2. Make sure you have already assigned some tags. If not, refer to *Section 9.1, "Adding tags to hosts, clusters, databases, and SAP Systems"*.
3. In the second row, click the last drop-down list with the name *Filter tags*. The drop-down list opens and shows you all of your defined tags.
4. Select the tag you want to filter from the drop-down list. It is possible to select more than one tag.

After you have finished the above procedure, Trento shows all respective hosts that contain one or more of your selected tags.

To remove the filter, click the × icon from the same drop-down list.

10 Integration with SUSE Manager

Trento integrates with SUSE Manager to provide the SAP admin with a summary of available patches and upgradable packages for any registered host that is also managed by given instance of SUSE Manager.

The user must enter the connection settings for the SUSE Manager instance in the *Settings* view:

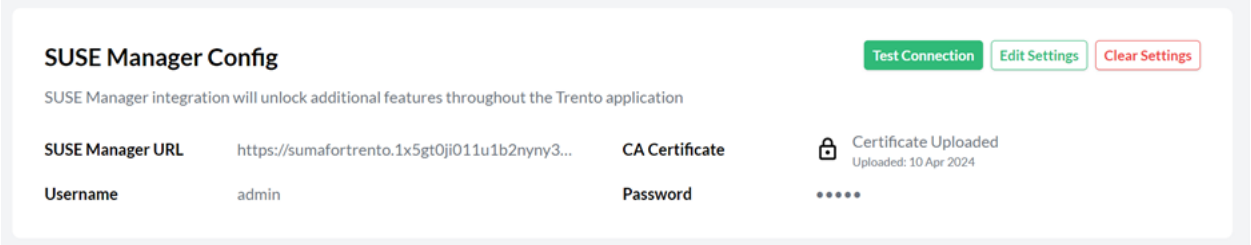


FIGURE 19: SUSE MANAGER SETTINGS

With the SUSE Manager settings configured, the SAP admin can test the connection by clicking *Test Connection*. If the connection is successful, the Host Details view displays a summary of the available patches and the upgradable packages for each host managed by SUSE Manager:

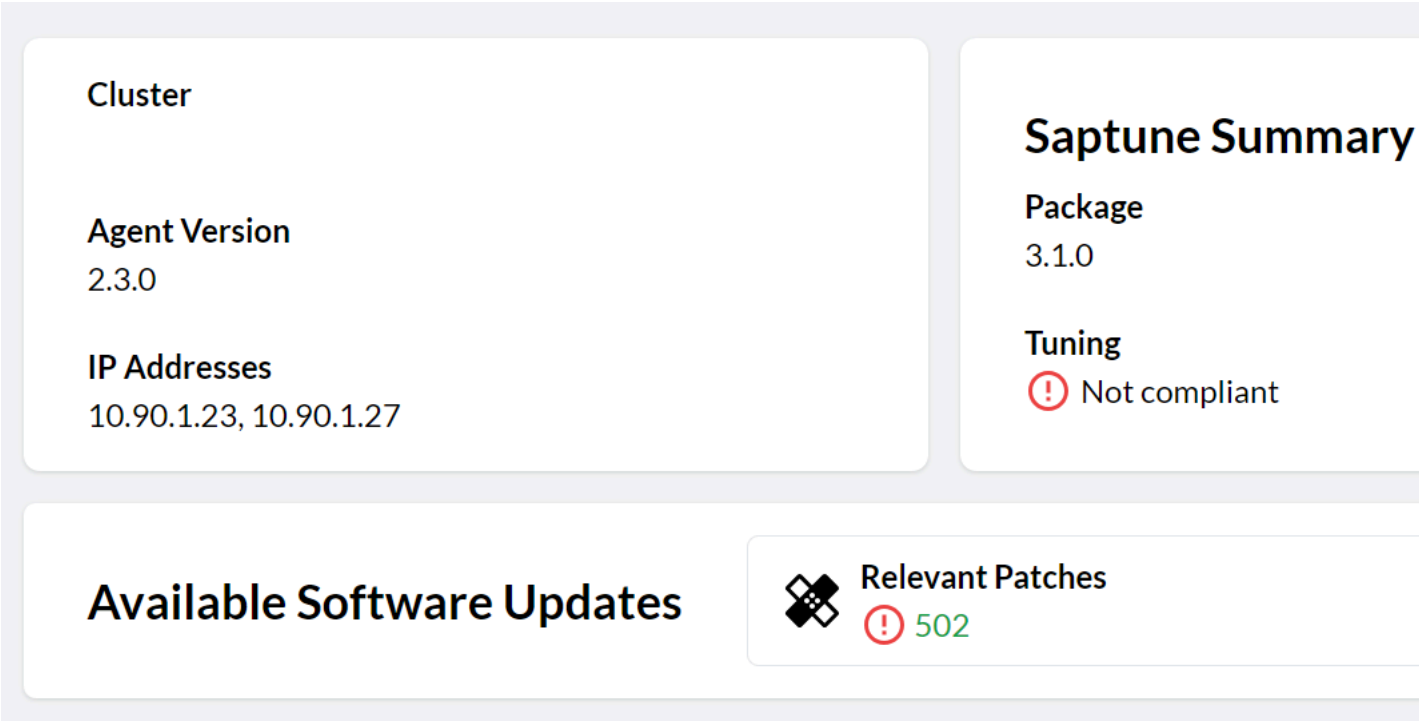


FIGURE 20: *AVAILALE SOFTWARE UPDATES IN THE HOST DETAILS VIEW*

There are three types of available patches: security advisories, bug fixes and feature enhancement. Security advisories are regarded as critical. If one is found, the health of the host will be set to critical too. The others are to be seen as warnings.

At any time, the user can clear the SUSE Manager settings from the Settings view. As a result, all information about available software updates disappears from the console and the status of the hosts is adjusted accordingly.

11 Rotating API keys

The communication from the Trento Agent to the Trento Server is secured by a API key which must be provided in the agent configuration file.

By default, the API key does not have an expiration date. To increase the overall security of the setup, meet internal security requirements, or both, the user can set up a custom expiration date.

To do that, go to the Settings view and click the *Generate Key* button in the API Key section:

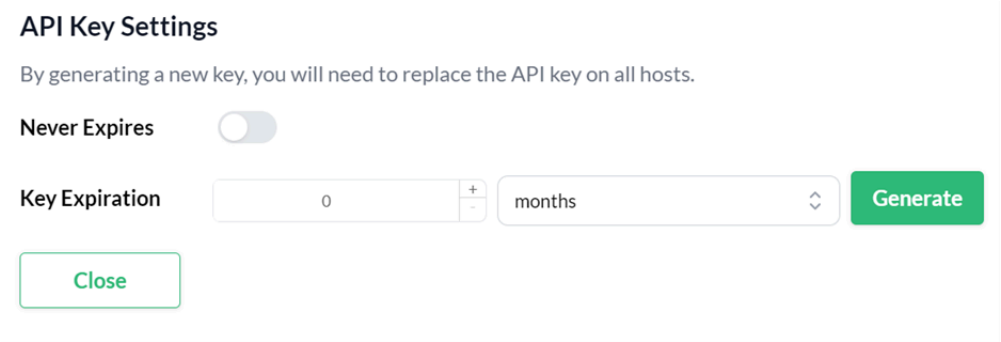


FIGURE 21: *CHECKS CATALOG*

Whenever a new key is generated, the configuration of all the reporting agents must be updated accordingly.

12 Updating Trento Server

The procedure to update the Trento Server depends on the deployment type: Kubernetes, systemd or containerized. The current content of this section is specific for Kubernetes deployments.

If Trento Server was installed manually, then it must be updated manually using the latest versions of the container images available in the SUSE public registry. If it was installed using Helm chart, it can be updated using the same Helm command as for the installation:

```
helm upgrade \
   --install trento-server oci://registry.suse.com/trento/trento-server \
   --set trento-web.trentoWebOrigin=TRENTO_SERVER_HOSTNAME \
   --set trento-web.adminUser.password=ADMIN_PASSWORD
```

A few things to consider:

- Before updating Trento Server ensure that all the Trento Agents in the environment are supported by the target version. See section *Section 18, "Compatibility matrix between Trento Server and Trento Agents"* for details.
- Remember to set the helm experimental flag if you are using a version of Helm lower than 3.8.0.
- When updating from a Trento version lower than 2.0.0 to version 2.0.0 or higher, an additional flag must be set in the Helm command:

  ```
  helm upgrade \
  ```

```
--install trento-server oci://registry.suse.com/trento/trento-server \
--set trento-web.trentoWebOrigin=TRENTO_SERVER_HOSTNAME \
--set trento-web.adminUser.password=ADMIN_PASSWORD \
--set rabbitmq.auth.erlangCookie=$(openssl rand -hex 16)
```

- When updating from an older Trento version to version 2.3.1 or higher, a new API key is generated and the configuration of all registered Trento Agents must be updated accordingly.
- If email alerting has been enabled, then the corresponding `trento-web.alerting` flags should be set in the Helm command as well.

13 Updating a Trento Agent

To update the Trento Agent, do the following:

1. Log in to the Trento Agent host.
2. Stop the Trento Agent:

```
> sudo systemctl stop trento-agent
```

3. Install the new package:

```
> sudo zypper ref
> sudo zypper install trento-agent
```

4. Copy file `/etc/trento/agent.yaml.rpmsave` onto `/etc/trento/agent.yaml` and ensure that entries `facts-service-url`, `server-url`, and `api-key` are maintained properly in the latter:
   - `facts-service-url`: the address of the AMQP service shared with the checks engine where fact gathering requests are received. The right syntax is `amqp://trento:trento@TRENTO_SERVER_HOSTNAME:5672`.
   - `server-url`: HTTP URL for the Trento Server (`http://TRENTO_SERVER_HOSTNAME`)
   - `api-key`: the API key retrieved from the Web console
5. Start the Trento Agent:

```
> sudo systemctl start trento-agent
```

6. Check the status of the Trento Agent:

```
> sudo systemctl status trento-agent
● trento-agent.service - Trento Agent service
     Loaded: loaded (/usr/lib/systemd/system/trento-agent.service; enabled; vendor
 preset: disabled)
     Active: active (running) since Wed 2021-11-24 17:37:46 UTC; 4s ago
   Main PID: 22055 (trento)
      Tasks: 10
     CGroup: /system.slice/trento-agent.service
             ├─22055 /usr/bin/trento agent start --consul-config-dir=/srv/consul/
consul.d
             └─22220 /usr/bin/ruby.ruby2.5 /usr/sbin/SUSEConnect -s

[...]
```

7. Check the version in the *Hosts* overview of the Trento Web console (URL `http://TRENTO_SERVER_HOSTNAME`).

8. Repeat this procedure in all Trento Agent hosts.

14 Uninstalling Trento Server

If Trento Server was deployed manually, then you need to uninstall it manually. If Trento Server was deployed using the Helm chart, you can also use Helm to uninstall it as follows:

```
helm uninstall trento-server
```

15 Uninstalling a Trento Agent

To uninstall a Trento Agent, perform the following steps:

1. Log in to the Trento Agent host.

2. Stop the Trento Agent:

```
> sudo systemctl stop trento-agent
```

3. Remove the package:

```
> sudo zypper remove trento-agent
```

16 Reporting a Problem

Any SUSE customer, with a registered SUSE Linux Enterprise Server for SAP Applications 15 (SP1 or higher) distribution, experiencing an issue with Trento Premium, can report the problem either directly in the SUSE Customer Center or through the corresponding vendor, depending on their licensing model. Please report the problem under SUSE Linux Enterprise Server for SAP Applications 15 and component trento.

When opening a support case for Trento, provide the following information, which can be retrieved as explained in section *Section 17, "Problem Analysis"*:

- The output of the Trento support plugin.
- Your scenario dump.

For issues with a particular Trento Agent, or a component discovered by a particular Trento Agent, also provide the following:

- The status of the Trento Agent.
- The journal of the Trento Agent.
- The output of the command `supportconfig` in the Trento Agent host. See https://documentation.suse.com/sles/html/SLES-all/cha-adm-support.html#sec-admsupport-cli for information on how to run this command from command line.

17 Problem Analysis

This section covers some common problems and how to analyze them.

17.1 Trento Server

To analyze issues within the application, we have two tools at our disposal. They both help us collect information/data that might be useful when troubleshooting/analyzing the issue.

17.1.1 Trento Support Plugin

The Trento support plugin automates the collection of logs and relevant runtime information on the server side. Using the plugin requires a host with the following setup:

- The packages `jq` and `python3-yq` are installed.
- Helm is installed.
- The command `kubectl` is installed and connected to the K8s cluster where Trento Server is running.

To use it, proceed as follows:

1. Download the Trento support plugin script:

```
# wget https://raw.githubusercontent.com/trento-project/support/refs/heads/main/
trento-support.sh
```

Note: Package available in SUSE Linux Enterprise Server for SAP Applications 15 SP3 and higher

Customers of SUSE Linux Enterprise Server for SAP Applications 15 SP3 and higher can install the `trento-supportconfig-plugin` package directly from the official SLES for SAP 15 repos using Zypper.

2. Make the script executable:

```
# chmod +x trento-support.sh
```

3. Ensure `kubectl` is connected to the K8s cluster where Trento Server is running and execute the script:

```
# ./trento-support.sh --output file-tgz --collect all
```

4. Send the generated archive file to support for analysis.

The Trento support plugin admits the following options:

`-o`, `--output`
: Output type. Options: stdout|file|file-tgz

`-c, --collect`
: Collection options: configuration|base|kubernetes|all

`-r, --release-name`
: Release name to use for the chart installation. "trento-server" by default.

`-n, --namespace`
: Kubernetes namespace used when installing the chart. "default" by default.

`--help`
: Shows help messages

17.1.2 Scenario dump

The scenario dump is a dump of the Trento database. It helps the Trento team to recreate the scenario to test it. Using the script that generates the dump requires a host with the following setup:

- The command **`kubectl`** is installed and connected to the K8s cluster where Trento Server is running.

To generate the dump, proceed as follows:

1. Download the latest version of the dump script:

```
> wget https://raw.githubusercontent.com/trento-project/web/main/hack/
dump_scenario_from_k8.sh
```

2. Make the script executable:

```
> chmod +x dump_scenario_from_k8.sh
```

3. Ensure **`kubectl`** is connected to the K8s cluster where Trento Server is running and execute the script as follows:

```
> ./dump_scenario_from_k8.sh -n SCENARIO_NAME -p PATH
```

4. Go to `/scenarios/`, package all the JSON files and send them to support for analysis.

17.1.3 Pods descriptions and logs

In additon to the output of the Trento Support Plugin and the Dump Scenario script, the descriptions and logs of the pods of the Trento Server can also be useful for analysis and troubleshooting purposes. These descriptions and logs can be easily retrieved with the **`kubectl`** command, for which you need a host where command **`kubectl`** is installed and connected to the K8s cluster where Trento Server is running.

1. List the pods running in Kubernetes cluster and their status. Trento Server currently consists of the following six pods:

```
> kubectl get pods
trento-server-prometheus-server-*
trento-server-postgresql-0
trento-server-web-*
trento-server-wanda-*
trento-server-rabbitmq-0
```

2. Retrieve the description of one particular pod as follows:

```
> kubectl describe pod POD_NAME
```

3. Retrieve the log of one particular pod as follows:

```
> kubectl logs POD_NAME
```

4. Monitor the log of one particular pod as follows:

```
> kubectl logs POD_NAME --follow
```

17.2 Trento Agent

The first source to analyze issues with the Trento Agent is the Trento Agent status, which can be retrieved as follows:

```
> sudo systemctl status trento-agent
```

If further analysis is required, it is convenient to activate debug log level for the agent. A detailed log can be then retrieved from the journal:

1. Add parameter `log-level` with value `debug` to the `/etc/trento/agent.yaml` configuration file.

```
> sudo vi /etc/trento/agent.yaml
```

2. Add the following entry:

```
log-level: debug
```

3. Restart the agent:

```
> sudo systemctl restart trento-agent
```

4. Retrieve the log from the journal:

```
> sudo journalctl -u trento-agent
```

18 Compatibility matrix between Trento Server and Trento Agents

TABLE 1: COMPATIBILITY MATRIX BETWEEN TRENTO SERVER AND TRENTO AGENTS

Trento Agent Compatibility matrix		Trento Server versions							
		1.0.0	1.1.0	1.2.0	2.0.0	2.1.0	2.2.0	2.3.1	2.3.2
Trento Agent versions	1.0.0	✓							
	1.1.0	✓	✓						
	1.2.0	✓	✓	✓					
	2.0.0				✓				
	2.1.0				✓	✓			
	2.2.0				✓	✓	✓		
	2.3.0				✓	✓	✓	✓	✓

19 Highlights of Trento versions

The following list shows the most important user-facing features in the different versions of Trento. For a more detailed information about the changes included in each new version, visit the GitHub project at https://github.com/trento-project. Particularly:

- For changes in the Trento Helm Chart, visit https://github.com/trento-project/helm-charts/releases.
- For changes in the Trento Server Web component, visit https://github.com/trento-project/web/releases.
- For changes in the Trento Server old checks engine component (runner), visit https://github.com/trento-project/runner/releases.
- For changes in the Trento Server new checks engine component (wanda), visit https://github.com/trento-project/wanda/releases.
- For changes in the Trento Agent, visit https://github.com/trento-project/agent/releases.

Version 2.3.2 (released on 2024/09/01)

Consisting of Helm chart 2.3.2, Web component 2.3.2, checks engine (wanda) 1.3.0 and agent 2.3.0.

- Fix for bug in the web component causing it to crash upon certain events after update from an earlier version.

Version 2.3.1 (released on 2024/06/18)

Consisting of Helm chart 2.3.1, Web component 2.3.1, checks engine (wanda) 1.3.0 and agent 2.3.0.

- non-K8s installation of Trento Server.
- Enhanced discovery capabilities, including improved support HANA multi-tenant architecture, HANA scale out cluster discovery, discovery of clsluters using diskless SBD and maintenance awareness.
- Summary of available software updates for each registered hosts via integration with SUSE manager.
- Rotating API key.

- Saptune configuration checks.
- Simpler, more secure architecture without Grafana.
- Revamped metric visualization.
- Enhanced Checks Catalog view with dynamic filtering capabilities.

Version 2.2.0 (released on 2023/12/04)

Consisting of Helm chart 2.2.0, Web component 2.2.0, checks engine (wanda) 1.2.0 and agent 2.2.0.

- saptune Web integration.
- Intance clean-up capabilities.
- Ability to run host-level configuration checks.

Version 2.1.0 (released on 2023/08/02)

Consisting of Helm chart 2.1.0, Web component 2.1.0, checks engine (wanda) 1.1.0 and agent 2.1.0.

- ASCS/ERS cluster discovery, from single-sid, two-node scenarios to multi-sid, multi-node setups. The discovery covers both versions of the enqueue server, ENSA1 and ENSA2, and both scenarios with resource managed instance filesystems and simple mount setups.
- Host clean-up capabilities, allowing users to get rid of hosts that are no longer part of their SAP environment.
- New checks results view that leverages the potential of the new checks engine (wanda) and provides the user with insightful information about the check execution.

Version 2.0.0 (released on 2023/04/26)

Consisting of Helm chart 2.0.0, Web component 2.0.0, checks engine (wanda) 1.0.0 and agent 2.0.0.

- A brand-new safer, faster, ligher SSH-less configuration checks engine (wanda) which not only opens the door to configuration checks for other HA scenarios (ASCS/ER, HANA scale-up cost optimized, SAP HANA scale-out) and other targes in the environment (hosts, SAP HANA databases, SAP NetWeaver instances), but also will allow customization of existing checks and addtion of custom checks.
- Addition of VMware to the list of known platforms.
- Versioned external APIs for both the Web and the checks engine (wanda) components. The full list of available APIs can be found at https://www.trento-project.io/web/swaggerui/ and https://www.trento-project.io/wanda/swaggerui/ respectiverly.

Version 1.2.0 (released on 2022/11/04)

Consisting of Helm chart 1.2.0, Web component 1.2.0, checks engine (runner) 1.1.0 and agent 1.1.0.

- Configration checks for HANA scale-up peformance optimized two-node clusters on on-premise bare metal platforms, including KVM and Nutanix.
- A dynamic dashboard that allows you to determine, at a glance, the overall health status of your SAP environment.

Version 1.1.0 (released on 2022/07/14)

Consisting of Helm chart 1.1.0, Web component 1.1.0, checks engine (runner) 1.0.1 and agent 1.1.0.

- Fix for major bug in the checks engine that prevented the Web component to load properly.

Version 1.0.0 (general availability, released on 2022/04/29)

Consisting of Helm chart 1.0.0, Web component 1.0.0, checks engine (runner) 1.0.0 and agent 1.0.0.

- Clean, simple Web console designed with SAP basis in mind. It reacts in real time to changes happening in the backend environment thanks to the event-driven technology behind it.
- Automated discovery of SAP systems, SAP instances and SAP HANA scale-up two-node clusters
- Configuration checks for SAP HANA scale-up performance optimized two-node clusters on Azure, AWS and GCP.

- Basic integration with Grafana and Prometheus to provide graphic dashboards about CPU and memory utilization in each discovered host.
- Basic alert emails for critical events happening in the monitored environment.

20 More information

- Homepage: https://www.trento-project.io/
- Trento project: https://github.com/trento-project/web