




# The SUSE ALP Dolomite Guide

## The SUSE ALP Dolomite Guide

This guide introduces the SUSE ALP Dolomite (ALP Dolomite)—its deployment, system management and software installation as well as running of containerized workloads. To enhance this ALP Dolomite documentation, find its sources at <https://github.com/SUSE/doc-modular/edit/main/xml/> .

### TOPIC

ALP Dolomite is a minimal operating system that allows customers to run their workloads in a containerized or virtualized form. Typically, ALP Dolomite is deployed on bare-metal servers and runs data center-oriented workloads.

### INTENTION

This guide introduces an overview of ALP Dolomite. It describes steps required to deploy and administer ALP Dolomite as well as install and manage SUSE workloads.

### EFFORT

Although ALP Dolomite is a complex platform, its basic deployment takes less than 30 minutes. More advanced topics, such as using the **transactional-update** command, or installing and configuring containerized SUSE workloads, take considerably more time.

### REQUIREMENTS

To understand the concepts and perform tasks described in this guide, you need to have good knowledge and practice with the SUSE Linux operating system.

Publication Date: 19 Oct 2023

<https://documentation.suse.com> 

# Contents

<b>1</b>	<b>General description</b>	<b>1</b>
1.1	What is ALP Dolomite?	1
<b>2</b>	<b>Deployment</b>	<b>2</b>
2.1	Considerations before deployment	2
	root SSH login	2
2.2	Deployment methods	3
	The Agama installer	3
	• Raw disk image	3
2.3	Hardware requirements	3
2.4	Preparing the ALP Dolomite virtual machine	4
	Requirements	4
	• Configuring the virtual machine for ALP Dolomite deployment	5
2.5	Interactive deployment using the Agama installer	8
2.6	Automated deployment using the Agama installer	13
	What is an automated deployment?	13
	• How does an automated deployment work?	13
	• Benefits of an automated deployment	15
	• Profile-based installation	15
	• Script-based installation	19
2.7	Deploying ALP Dolomite on an IBM Z host	19
	Requirements	20
	• Download and prepare the installation image	20
	• Example deployment procedure	20
2.8	Deployment using a raw disk image	23
	Introduction	23
	• Deploying ALP Dolomite with JeOS Firstboot	25
	• Next steps	26
	• Configuring with Ignition	27
	• Configuring with Combustion	38

- 2.9 Post-deployment considerations 43
  - Full disk encryption 43 • SELinux 44 • Enabling root login via SSH 45
- 2.10 For more information 46
- 3 Transactional updates 47**
  - 3.1 What are transactional updates? 47
  - 3.2 How do transactional updates work? 47
  - 3.3 Benefits of transactional updates 47
    - Environment within transactional-update command 48
  - 3.4 /etc on a read-only file system 48
  - 3.5 Directories excluded from snapshots 49
  - 3.6 Applying multiple changes without reboot 50
    - The **transactional-update --continue** option 50 • The **transactional-update run** command 50 • The **transactional-update shell** 51
  - 3.7 Usage of the **transactional-update** command 52
    - The **register** command 55
- 4 Containers and Podman 57**
  - 4.1 What are containers and Podman? 57
  - 4.2 How does Podman work? 57
    - Pods architecture 57
  - 4.3 Benefits of containers 59
  - 4.4 Enabling Podman 59
    - Requirements 59 • Installing Podman 59 • Enabling rootless mode 60 • Next steps 60
  - 4.5 Autostarting containers 60

- 4.6 Podman usage 62
  - Getting container images 62 • Working with containers 63 • Working with pods 68

## 5 SUSE Workloads 72

- 5.1 Common requirements 72
- 5.2 Running the YaST workload using Podman 72
  - Introduction 72 • Starting YaST in text mode 73 • Starting graphical YaST 73
- 5.3 Running the KVM virtualization workload using Podman 75
  - Deploying the KVM Server workload 75 • Usage of the **kvm-server-manage** script 76 • Using the KVM Client workload 77
- 5.4 Running the Cockpit Web server using Podman 78
  - Software updates 80
- 5.5 Running the GNOME Display Manager workload using Podman 80
  - Starting the GDM workload 80
- 5.6 Running firewalld using Podman 83
  - Running the firewalld workload 83
- 5.7 Running the Grafana workload using Podman 85
  - Starting the Grafana workload 85 • Setting up a Grafana client 86 • Configuring the Grafana Web application 87 • Usage of the **grafana-container-manage.sh** script 89
- 5.8 Running the NeuVector workload using Podman 90
  - Starting NeuVector 90 • Uninstalling NeuVector 91
- 5.9 Running the Ansible workload using Podman 91
  - Installing Ansible commands 91 • Uninstalling Ansible 93 • Operation via SSH 93 • Examples of Ansible playbooks 95
- 5.10 Running the Kea DHCP server using Podman 97
  - Deploying and running the Kea workload 97 • Configuration files 99 • The **keactrl** wrapper 100
- 5.11 For more information 100

## 6 Creating customized VMs using **virt-scenario** 101

- 6.1 How does **virt-scenario** work? 101
- 6.2 Benefits of using **virt-scenario** 101
- 6.3 Creating VMs 102
  - Requirements 103 • Overriding default scenario settings 103 • Specifying **virt-scenario** mode 104 • Interactive commands 105
- 6.4 Predefined scenarios 106
- 6.5 Managing VMs 109

## 7 Remote attestation using Keylime 111

- 7.1 Terminology 111
- 7.2 What is Keylime? 112
- 7.3 Architecture 113
  - Keylime agent 113 • Keylime registrar 113 • Keylime verifier 113
- 7.4 Setting up the verifier, registrar and tenant 114
  - Monitoring Keylime services 115 • Executing the tenant CLI 115 • Extracting the Keylime certificate 115
- 7.5 Installing the agent 116
- 7.6 Registering the agent 117
- 7.7 Secure payloads 118
  - What is a secure payload? 118 • How does a secure payload work? 118
- 7.8 Enabling IMA tracking 119
- 7.9 For more information 120

## 8 Scheduling jobs using systemd timers 121

- 8.1 Creating a timer 121
  - Hello World* example 121 • The example explained 122
- 8.2 Managing timers 124

8.3	Timer types	126
8.4	Testing calendar entries	129
8.5	Getting e-mail notifications when a timer fails	130
8.6	Using timers as a regular user	132
8.7	Migrating from cron to systemd timers	134
8.8	Troubleshooting and FAQs	135
	Avoiding errors	135
	• Event is not triggered	136
	• Checking the system journal for errors	136
	• systemd timer: catching up on missed runs	137
	• How to migrate from cron to systemd timers?	137
8.9	For more information	139
<b>A</b>	<b>Legal Notice</b>	<b>140</b>
<b>B</b>	<b>GNU Free Documentation License</b>	<b>141</b>



# 1 General description

## 1.1 What is ALP Dolomite?

ALP Dolomite is a minimal operating system that allows customers to run their workloads in a containerized or virtualized form. Typically, ALP Dolomite is deployed on bare-metal servers and runs data center-oriented workloads. ALP Dolomite is best described by the following characteristics:

### Minimalistic

ALP Dolomite provides the bare minimum to run workloads and services as containers or virtual machines. Its default software collection is carefully preselected to include only packages required to support its mission.

### Secure

ALP Dolomite is an immutable operating system with a read-only file system. Changes to the file system are performed in *transactions* and can easily be rolled back. Moreover, the SELinux access control makes it a truly secure platform.

### Container-ready

In ALP Dolomite, containerized workloads replace traditional applications. They contain all software dependencies required to run a specific application or tool. ALP Dolomite uses Podman as the default container engine. Podman makes it easy to find, run, build, share and deploy containerized workloads.

## 2 Deployment

### 2.1 Considerations before deployment

This section introduces tips and suggestions that need to be considered before or during the deployment.

#### 2.1.1 `root` SSH login

By default, `root` SSH login in ALP Dolomite is permitted only by using the SSH key. `root` SSH login with password is prohibited. To enable `root` SSH login with password, you have several options:

- **Create an unprivileged user.** Depending on the installation method, you can either create a new user in the Agama installer, or use Combustion/Ignition tools. Refer to [Section 2.5, “Interactive deployment using the Agama installer”](#) and [Section 2.8.4, “Configuring with Ignition”](#) for more details.



#### Tip

Creating an unprivileged user during system installation is useful for logging in to the Cockpit Web interface. Find more details in <https://documentation.suse.com/alp/dolomite/html/cockpit-alp-dolomite/index.html> [↗](#).

- **Install the `openssh-server-config-rootlogin` package.** Use Combustion/Ignition tools to install the package during the deployment process.



#### Tip

See [Section 2.9.3, “Enabling root login via SSH”](#) for details on installing `openssh-server-config-rootlogin` manually after the system is deployed.

## 2.2 Deployment methods

SUSE ALP Dolomite is distributed either as a disk image of the Agama installer, or as a pre-built raw disk image.

### 2.2.1 The Agama installer

While the Agama installer handles both bare-metal and virtualized/cloud deployments, it is a preferred method for bare-metal deployments. ALP Dolomite deployment using the Agama installer is similar to a traditional operating system setup. After booting the Agama installer image, the installer uses a graphical user-friendly interface to walk you through the system configuration and deployment.



#### Note

The installer live images do not contain `linuxrc`, `wicked` and `installation-images` packages. Therefore, the device activation and configuration need adjusting, and there is no interactive menu for configuring certain parameters of the installation.

### 2.2.2 Raw disk image

This method handles both bare-metal and virtualized/cloud deployment. It is different from the installer-based deployment in that you do not boot an installer but the actual ALP Dolomite image itself. On first boot, you can configure basic system options using an *ncurses* user interface. Using a raw disk image, you can fine-tune the deployment setup with Combustion and Ignition tools.

## 2.3 Hardware requirements

The minimum supported hardware requirements for deploying ALP Dolomite follow:

#### BIOS

Installing ALP Dolomite is supported on hosts with UEFI BIOS only. Hosts with the legacy BIOS are not supported.

## CPU

AMD64/Intel 64, AArch64 and IBM Z CPU architectures are supported.

## Maximum number of CPUs

The maximum number of CPUs supported by software design is 8192.

## Memory

ALP Dolomite requires at least 1 GB RAM. Bear in mind that this is a minimal value for the operating system, the actual memory size depends on the workload.

## Hard disk

The minimum hard disk space is 12 GB, while the recommended value is 20 GB of hard disk space. Adjust the value according to the workloads of your containers.

# 2.4 Preparing the ALP Dolomite virtual machine

This article describes how to configure a new virtual machine suitable for the ALP Dolomite deployment by using the Virtual Machine Manager.

## 2.4.1 Requirements

- A VM Host Server with KVM hypervisor.
- Download either the ALP Dolomite raw disk or the Agama installer image from <https://susealp.io/downloads/> on the VM Host Server where you intend to run virtualized ALP Dolomite.



### Note

For the raw disk image deployment, there are two types of images, depending on whether you intend to run ALP Dolomite on an encrypted disk or an unencrypted disk.

## ! Important: Encrypted image does not expand to the full disk capacity

As of now, the encrypted raw disk image does not expand to the full disk capacity automatically. As a workaround, the following steps are required:

1. Use the `qemu-img` command to increase the disk image to the desired size.
2. Set up the virtual machine and boot it. When the JeOS Firstboot wizard asks you which method to use for encryption, select *passphrase*.
3. When the system is ready, use the `parted` command to resize the partition where the LUKS device resides (for example, partition number 3) to the desired size.
4. Run the `cryptsetup resize luks` command. When asked, enter the passphrase to resize the encrypted device.
5. Run the `transactional-update shell` command to open a read-write shell in the current disk snapshot. Then resize the Btrfs file system to the desired size, for example:

```
# btrfs fi resize max /
```

6. Leave the shell with `exit` and reboot the system with `reboot`.

### 2.4.2 Configuring the virtual machine for ALP Dolomite deployment

1. Start Virtual Machine Manager and select *File > New Virtual Machine*.
  - a. For deployment using the Agama installer, select *Local install media*.
  - b. For the raw disk deployment, select *Import existing disk image*.
2. Confirm with *Forward*.
3. Specify the path to the ALP Dolomite disk image that you previously downloaded and the type of linux OS you are deploying, for example, Generic Linux 2020. Confirm with *Forward*.

4. Specify the amount of memory and number of processors that you want to assign to the ALP Dolomite virtual machine and confirm with *Forward*.
5. For deployment using the Agama installer, enable storage for the virtual machine and specify the size of the disk image.
6. Specify the name for the virtual machine and the network to be used.
7. If you are deploying an encrypted ALP Dolomite image, perform these additional steps:
  - a. Enable *Customize configuration before install* and confirm with *Finish*.
  - b. Click *Overview* from the left menu and change the boot method from BIOS to UEFI for secure boot. Confirm with *Apply*.

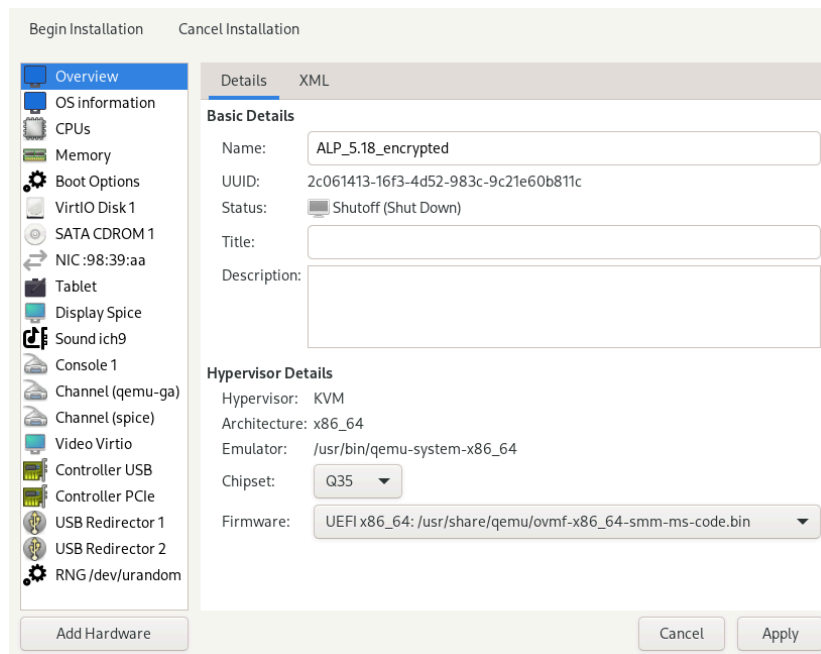


FIGURE 2.1: SET UEFI FIRMWARE FOR THE ENCRYPTED ALP DOLOMITE IMAGE

- c. Add a Trusted Platform Module (TPM) device. Click *Add Hardware*, select *TPM* from the left menu, and select the *Emulated* type.

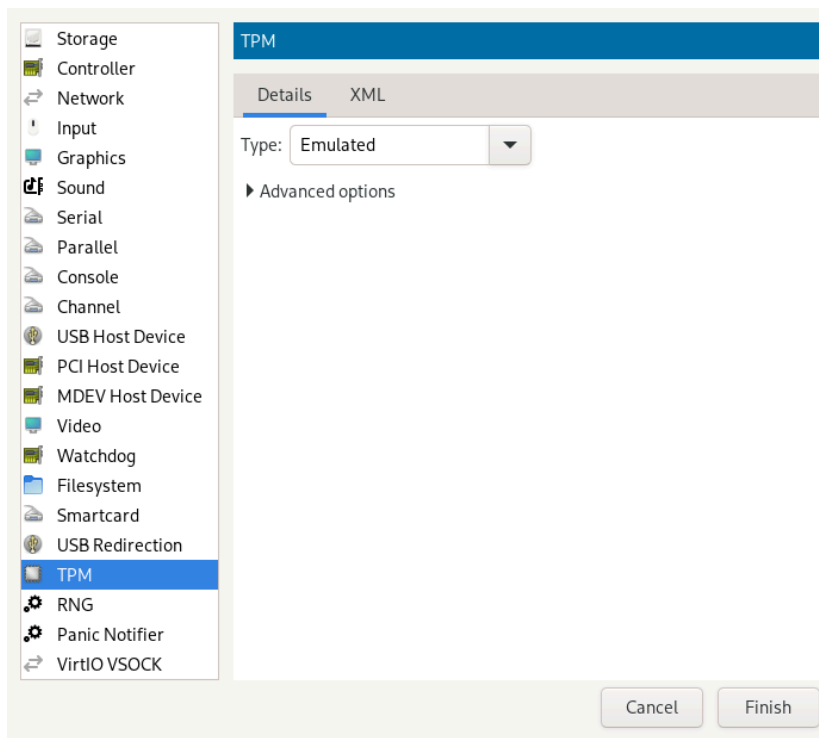


FIGURE 2.2: ADD AN EMULATED TPM DEVICE

Confirm with *Finish* and start the ALP Dolomite deployment by clicking *Begin Installation* from the top menu.

8. a. For the raw disk image deployment, to deploy ALP Dolomite with only minimal setup options, confirm with *Finish*. The disk image will be booted and JeOS Firstboot will take care of the deployment. Refer to [Section 2.8.2, “Deploying ALP Dolomite with JeOS Firstboot”](#) for next steps.




### Tip

You can detail the deployment setup by using the Ignition or Combustion tools. For more details, refer to [Section 2.8.4, “Configuring with Ignition”](#) and [Section 2.8.5, “Configuring with Combustion”](#).

- b. To continue the deployment by using the Agama installer, confirm with *finish* and continue with [Section 2.5, “Interactive deployment using the Agama installer”](#).

## 2.5 Interactive deployment using the Agama installer

This article describes how to deploy ALP Dolomite interactively using the Agama installer. For an automated deployment, refer to [Section 2.6, “Automated deployment using the Agama installer”](#).

1. Download the ALP Dolomite Agama installer image from <https://susealp.io/downloads/> .
2. If you are deploying ALP Dolomite as a VM Guest, you need to first prepare the virtual machine. To do this, follow the steps in [Section 2.4, “Preparing the ALP Dolomite virtual machine”](#).
3. Boot the Agama installer image and select *agama-live* from the boot menu. A screen with the main installation menu is displayed.

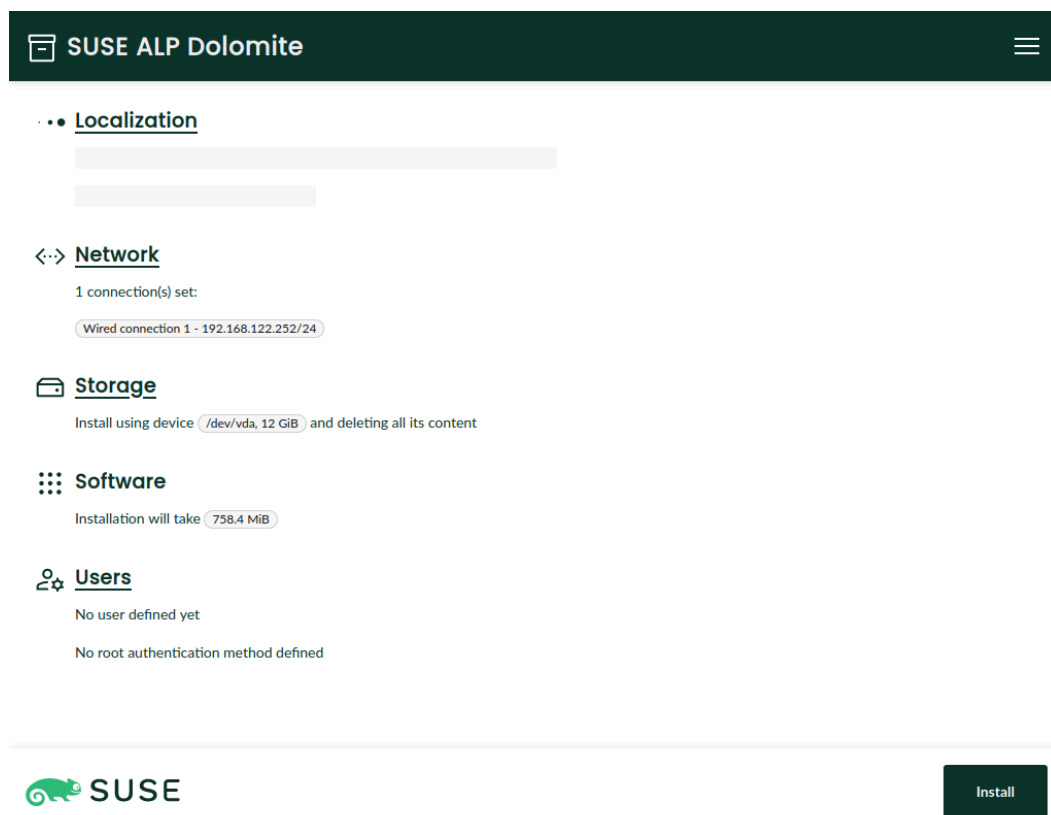


FIGURE 2.3: AGAMA INSTALLER MAIN MENU

4. Click *Localization* and select your preferred language from the drop-down list.
5. Configure *Network* settings by selecting *Edit* from the menu on the right side of the default wired connection. You can, for example, change the networking mode to *Manual*, add IP addresses and related prefixes or netmasks, or add gateway and DNS servers.



**Edit "Wired connection 1" connection**

**Mode \***

Manual ▼

**Addresses \*** Add another address

192.168.1.11	255.255.0.0	<a href="#">Remove</a>
IP Address	Prefix length or netma...	<a href="#">Remove</a>

**Gateway**

192.168.1.1

**DNS**

Add another DNS

192.168.1.2	<a href="#">Remove</a>
-------------	------------------------

Cancel Confirm

FIGURE 2.4: CONFIGURING THE NETWORK

By clicking *Connect to a Wi-Fi network* you can utilize your local wireless network.

6. Configure *Storage*.

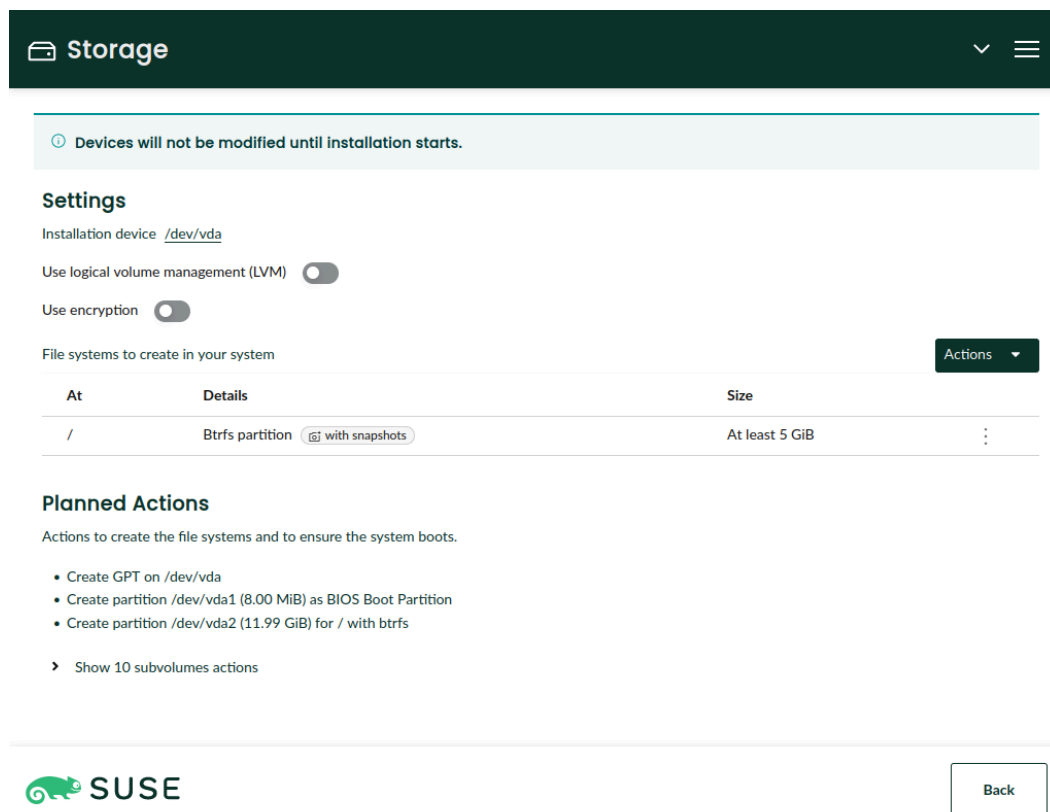


FIGURE 2.5: CONFIGURING STORAGE

Select the device where ALP Dolomite should be installed. Optionally, enable LVM or disk encryption and enter an encryption password.



## Tip

If you enable disk encryption, you may be asked for a decryption password on each reboot. Because the GRUB 2 boot loader does not enable switching keyboard layouts, select a password made of alphanumeric characters and be aware of national keyboard layout differences. For extended post-deployment information about disk encryption, refer to [Section 2.9.1, “Full disk encryption”](#).

By default, the partitioning scheme includes one root file system. To change its size, click the corresponding three dots on the right side and select *Edit*.

### Edit file system

Mount point \*

/

File system type \*

Btrfs

Size \*

☐ Fixed ☒ Range

Limits for the file system size. The final size will be a value between the given minimum and maximum sizes. If no maximum is given, then the file system will be as big as possible.

Minimum \* Maximum

5 GiB GiB

Cancel Accept

FIGURE 2.6: EDITING FILE SYSTEM PROPERTIES

To create additional file systems, click *Actions* on the *Settings* page and select *Add file system*.

To configure iSCSI targets or DASD disks (for IBM Z) for the installation, click the down arrow on the right of the top bar and select *iSCSI* or *DASD*.

### Storage iSCSI

Initiator

Name	iBFT	Offload card
iqn.1996-04.de.suse:01:351e6d6249	No	None

Targets

Discover

Name	Portal	Interface	iBFT	Status
iqn.2023-03.com.example:a7b6213e1835593b2359aaa	192.168.100.215:3260	default	No	Disconnected
iqn.2023-01.com.suse:12eee9e754de2664ab96	192.168.100.215:3260	default	No	Disconnected

SUSE

Back

FIGURE 2.7: DISCOVER ISCSI TARGETS

Click *Discover* to add new iSCSI targets.

## Discover iSCSI Targets

IP address \*

Port \*

Authentication by target

Username

Password

Authentication by initiator

Username

Password

FIGURE 2.8: ADDING A NEW ISCSI TARGET

Storage DASD

Filter by min channel

Filter by max channel

Perform an action ▼

<input checked="" type="checkbox"/> Channel ... ↑	Status ↓	Device ↓	Type ↓	Diag ↓	Format
<input checked="" type="checkbox"/> 0.0.0160	active	dasda	ECKD	No	Yes

Activate

Deactivate

Set DIAG On

Set DIAG Off

Format

Back

FIGURE 2.9: DASD STORAGE CONTEXTUAL MENU (IBM Z)

7. In the *Users* section, specify a root password, upload a *Root SSH public key*, or create an additional user account and optionally enable auto login for it.

### Create user account

Full name

Tux User

Username \*

tux

Password

•••••

Password confirmation

•••••

☒ Auto-login

Cancel

Confirm

FIGURE 2.10: ADDING NEW USERS

8. To begin the installation, click *Install* and confirm with *Continue*.  
After the installation is finished, click *Reboot* and select *ALP Dolomite* from the boot menu after reboot.

## 2.6 Automated deployment using the Agama installer

### 2.6.1 What is an automated deployment?

As an addition to the interactive deployment described in [Section 2.5, “Interactive deployment using the Agama installer”](#), the Agama installer supports an unattended automated deployment.

### 2.6.2 How does an automated deployment work?

The Agama installer supports the following types of unattended deployment:

#### Profile-based deployment

This type of deployment uses a customized file called *profile* that includes a description of the system to install.

## Script-based deployment

This type of deployment uses a plain shell script that enables custom pre-deployment workflows.

The actual automated deployment is started by passing the following parameter on the kernel command line during the Agama installer boot process:

```
agama.auto=PROFILE_OR_SCRIPT_URL
```



### Tip

When booting from the Agama installer live media ISO, you need to edit and modify the GRUB 2 boot loader command line and specify the path to the profile configuration file. When booting via PXE, the file can be reached via HTTP, for example:

```
agama.auto=http://example.net/profile1.jsonnet
```



### Important: Use correct file suffix

Using the correct suffix of the file name is important:

**.jsonnet**

Enables dynamic content through Jsonnet.

**.json**

Assumes that the profile is just a JSON file with no dynamic content.

**.sh**

Is interpreted as a shell script.

## 2.6.3 Benefits of an automated deployment

- You can prepare the deployment setup in advance and modify it easily for future deployments.
- By running an unattended deployment, you can save the time that you would normally spend in the interactive deployment process.
- After fine-tuning the deployment profile, you can deploy multiple hosts at the same time to meet datacenter requirements.

## 2.6.4 Profile-based installation

Select the profile-based installation if you prefer fine-tuning installation scenarios, using dynamic modifications, or validating the installation profile.

### 2.6.4.1 Deployment profile

A *profile* defines which options to use during the deployment process. For example, which product to install, localization settings, or partitioning layout. Profiles are written in [Jsonnet](https://jsonnet.org/) (<https://jsonnet.org/>). Jsonnet is a superset of JSON that provides additional features, such as creating dynamic and more concise profiles.

#### EXAMPLE 2.1: EXAMPLE PROFILE

```
{
  "localization": {
    "language": "en_US"
  },
  "software": {
    "product": "ALP-Bedrock"
  },
  "storage": {
    "devices": [
      {
        "name": "/dev/sda"
      }
    ]
  },
  "user": {
    "fullName": "Jane Doe",
```

```
"password": "123456",
"userName": "jane.doe"
}
}
```

#### 2.6.4.2 Dynamic profiles

You can adapt the deployment profile at runtime depending on the system where the automated deployment is running. The Agama installer injects the hardware information into the profile to be processed using Jsonnet.

The following example profile is adapted to install ALP Dolomite on the biggest disk discovered. The hardware information provided by the `lshw` command is available as a `hw.libsonnet` JSON object.



### Important

Only the storage information is injected for now. You can inspect the available data by installing the `lshw` package and running the following command:

```
lshw -json -class disk
```

```
local agama = import 'hw.libsonnet';
local findBiggestDisk(disks) =
  local sizedDisks = std.filter(function(d) std.objectHas(d, 'size'), disks);
  local sorted = std.sort(sizedDisks, function(x) x.size);
  sorted[0].logicalname;

{
  software: {
    product: 'ALP-Bedrock',
  },
  root: {
    password: 'nots3cr3t',
  },
  localization: {
    language: 'en_US',
  },
  storage: {
    devices: [
      {
        name: findBiggestDisk(agama.disks),
      }
    ]
  }
}
```



```
    },  
    1,  
  },  
}
```

#### 2.6.4.3 Evaluating and validating profiles

The Agama installer includes a command-line interface available from the `agama-cli` package. It handles multiple tasks, such as downloading, validating and evaluating profiles. For example, the following command checks the result of running the previous profile:

```
> sudo agama profile evaluate example-profile.jsonnet
```

To validate a JSON profile, run the following command:

```
> agama profile validate my-profile.json
```



### Important

You can only validate JSON profiles. Jsonnet profiles must be evaluated first.

#### 2.6.4.4 Generating profiles

Although writing deployment profiles manually in JSON format is easy, the Agama installer can export the current setup as a profile:

1. Boot the Agama image.
2. Configure all deployment options in the Agama user interface as described in [Section 2.5, “Interactive deployment using the Agama installer”](#).
3. Open the Agama built-in terminal and enter the following command to dump the current deployment profile:

```
> sudo agama config show
```

#### 2.6.4.5 Supported options

The following options are supported when creating installation profiles:



## Tip

Although profiled-based configuration files use JSON syntax, this reference uses *PARENT.CHILD* expressions for convenience purposes. Therefore,

```
localization.language
```

translates to

```
"localization": {  
  "language":  
}
```

Refer to [Section 2.6.4.1, “Deployment profile”](#) for the description and syntax of the profile-based configuration files.

### software

Specifies which software or product to install.

### software.product

Mandatory product identifier. For example, ALP-Dolomite.

### localization

Localization settings

### localization.language

System language ID. For example, en\_US.

### storage

Storage settings

### storage.devices

Array of devices where ALP Dolomite will be installed. For example, [ "/dev/sda" ].

### user

First user account settings

### user.fullName

Full user name

### user.userName

User login name

`user.password`

User password

`root`

Authentication of the root

`root.password`

root password

`root.sshPublicKey`

root SSH public key

## 2.6.5 Script-based installation

Select the script-based installation to obtain full control of the installation process.

The script that you specify at the boot command line

```
agama.auto=PROFILE_OR_SCRIPT_URL
```

is a Linux shell script and you can include any commands available on the deployment media. To specify ALP Dolomite deployment option, use the `/usr/bin/agama` command.

The following is a minimal working example to install ALP Dolomite:

```
set -ex

/usr/bin/agama config set software.product=ALP-Dolomite
/usr/bin/agama config set user.userName=EXAMPLE_USER user.password=PASSWORD
/usr/bin/agama install
```

## 2.7 Deploying ALP Dolomite on an IBM Z host

This article describes how to deploy ALP Dolomite on an IBM Z host using the Agama installer.

## 2.7.1 Requirements

Before installing ALP Dolomite on IBM Z, you need to fulfill the following requirements:

- The deployment on an IBM Z architecture is specific and requires that you review its basic concepts. Although ALP Dolomite is different from SUSE Linux Enterprise systems, study the information provided by <https://documentation.suse.com/sles/html/SLES-all/cha-zseries.html#sec-zseries-prep> before starting the deployment because most of it is valid for the ALP Dolomite deployment as well.
- Refer to <https://documentation.suse.com/sles/html/SLES-all/cha-zseries.html#sec-zseries-requirements> for generic system requirements.

## 2.7.2 Download and prepare the installation image

Download the Agama installer image for the IBM Z architecture from <https://susealp.io/downloads/>

Prepare the ISO image to be served by the FTP server. Extract its content so that you can modify it:

```
> sudo mv agama-live.s390x-1.0-ALP-M2.iso /srv/ftp/agama.iso
> sudo cd /srv/ftp/
> sudo isoinfo -R -X -i agama.iso
> sudo chmod a+u boot s390x/initrd
```



### Tip

Setting up an installation server (NFS or FTP) is out of the scope of this article. For further information, refer to <https://documentation.suse.com/sles/html/SLES-all/cha-zseries.html#sec-zseries-prep>.

## 2.7.3 Example deployment procedure

The following procedure describes steps to deploy ALP Dolomite on an IBM Z machine via a z/VM console.

1. Install the x3270 package that provides the 3270-type terminal emulator.

2. Connect to the LPAR server using the x3270 console. When asked, provide your login credentials.

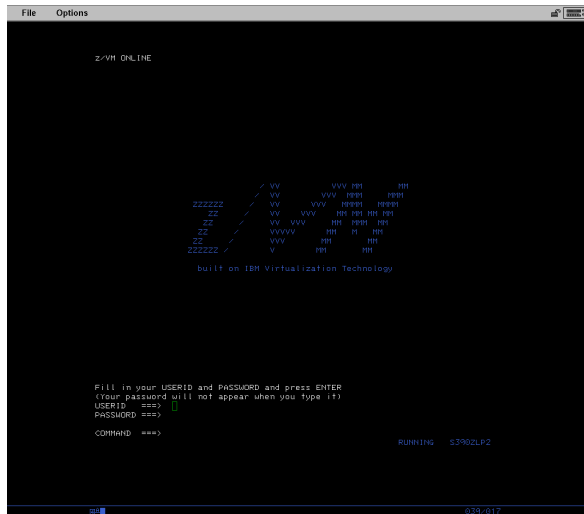


FIGURE 2.11: LOGIN PROMPT INSIDE THE Z/VM CONSOLE

3. Enter the Conversational Monitoring System (CMS):

```
#CP IPL CMS
```

4. Link the TCPMAINT disk to have the FTP command available:

```
VMLINK TCPMAINT 592
```

5. Connect to the FTP server and download the required files for IPLing installation. In our case, the anonymous user is allowed:

```
FTP example.org (addr ipv4
anonymous

cd boot/s390x
locsite fix 80
ascii
get parmfile sles.parmfile.a (repl
get sles.exec sles.exec.a (repl
locsite fix 80
binary
get linux sles.linux.a (repl
get initrd sles.initrd.a (repl
quit
```



## Note

The **locs site fix 80** command sets the VM file format to a fixed length of 80. This file format is necessary for *punching* the binary files to a virtual machine reader.

6. Optionally, you can use the **FILELIST** command to list the files and edit the **parmfile** with XEDIT. Our example parmfile has the following content:

```
cio_ignore=all,!condev,!0.0.0160 ❶  
rd.zdev=qeth,0.0.0800:0.0.0801:0.0.0802,layer2=1,portno=0 ❷  
ip=192.168.0.111::192.168.0.1:24:zvmtest.example.org:enc800:none ❸  
nameserver=192.168.0.1  
root=live:ftp://example.org/agama.iso ❹
```

- ❶ Although the `cio_ignore` parameter is optional, it is used to list only the relevant installation devices and accept the devices that are used for the installation.
- ❷ ❸ Because we do not have an interactive dialog for enabling and configuring our network device, we need to provide the settings through the kernel command line. The `rd.zdev` option activates the `qeth` device and the `ip` option configures network settings for the `enc800` Linux network interface.
- ❹ The system boots from a live image retrieved from the specified URL. Our example uses FTP protocol, but it can be HTTP as well.

7. The following is the content of the `sles.exec` file:

```
/* REXX LOAD EXEC FOR SUSE LINUX S/390 VM GUESTS      */  
/* LOADS SUSE LINUX S/390 FILES INTO READER          */  
SAY ''  
SAY 'LOADING SLES FILES INTO READER...'  
'CP CLOSE RDR'  
'PURGE RDR ALL'  
'SPOOL PUNCH * RDR'  
'PUNCH SLES LINUX A (NOH'  
'PUNCH SLES PARMFILE A (NOH'  
'PUNCH SLES INITRD A (NOH'  
'IPL 00C'
```

Boot the installation image by running the `sles.exec` REXX file:

```
sles
```

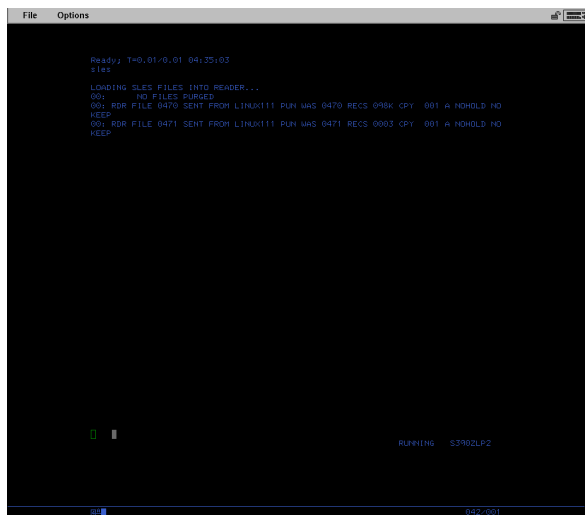


FIGURE 2.12: BOOTING THE INSTALLATION IMAGE

8. After the installation system finishes the booting process, connect to the machine either with the Web browser (for example, <https://example.host.org:9090>) or via SSH using the default credentials (user name: `root`, password: `linux`). The rest of the installation process is identical to *Section 2.5, “Interactive deployment using the Agama installer”*.

## 2.8 Deployment using a raw disk image

### 2.8.1 Introduction

This article describes how to deploy the SUSE ALP Dolomite (ALP Dolomite) raw disk image. It applies to ALP Dolomite running both on encrypted and unencrypted disk.

#### 2.8.1.1 First boot detection

The deployment configuration runs on the first boot only. To distinguish between the first and subsequent boots, the flag file `/boot/writable/firstboot_happened` is created after the first boot finishes. If the file is not present in the file system, the attribute `ignition.firstboot` is passed to the kernel command line and thus both Ignition and Combustion are triggered to run (in the `initrd`). After completing the first boot, the `/boot/writable/firstboot_happened` flag file is created.



### Note: The flag file is always created

Even though the configuration may not be successful due to improper or missing configuration files, the `/boot/writable/firstboot_happened` flag file is created.



### Tip

You may force the first boot configuration on subsequent boot by passing the `ignition.firstboot` attribute to the kernel command line or by deleting the `/boot/writable/firstboot_happened` flag file.

## 2.8.1.2 Default partitioning

The pre-built images are delivered with a default partitioning scheme. You can change it during the first boot by using Ignition or Combustion.



### Important: Btrfs is mandatory for the root file system

If you intend to perform any changes to the default partitioning scheme, the root file system must be Btrfs.

Each image has the following subvolumes:

```
/home
/root
/opt
/srv
/usr/local
/var
```

The `/etc` directory is mounted as overlayFS, where the upper directory is mounted to `/var/lib/overlay/1/etc/`.

You can recognize the subvolumes mounted by default by the option `x-initrd.mount` in `/etc/fstab`. Other subvolumes or partitions must be configured either by Ignition or Combustion.



## 2.8.2 Deploying ALP Dolomite with JeOS Firstboot



### Tip

When booting the ALP Dolomite raw image for the first time, *JeOS Firstboot* enables you to perform a minimal configuration of your system. If you need more control over the deployment process, find more information in [Section 2.8.4, “Configuring with Ignition”](#) and [Section 2.8.5, “Configuring with Combustion”](#).



### Tip

If you wish to inspect the installation image before installation, the default LUKS password 1234 is required to successfully map the image on a local Linux system.

1. Download the ALP Dolomite raw disk image from <https://susealp.io/downloads/>. There are two types of raw images, depending on whether you intend to run ALP Dolomite on an encrypted disk or an unencrypted disk.
2. If you are deploying ALP Dolomite as a VM Guest, you need to first prepare the virtual machine by following [Section 2.4, “Preparing the ALP Dolomite virtual machine”](#).
3. After booting the ALP Dolomite disk image, you are presented with a boot loader screen. Select *ALP Dolomite* and confirm with **Enter**.
4. *JeOS Firstboot* displays a welcome screen. Confirm with **Enter**.



FIGURE 2.13: INSTALLATION WELCOME SCREEN

5. On the next screens, select keyboard, confirm the license agreement and select the time zone.
6. In the *Enter root password* dialog window, enter a password for the root and confirm it.

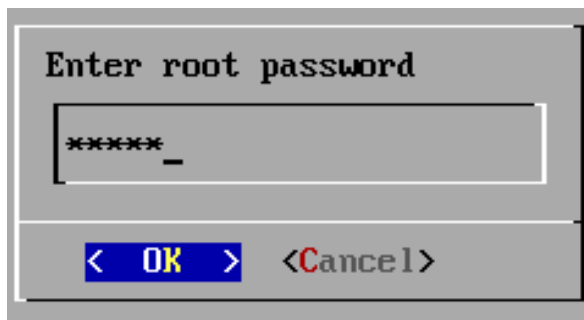


FIGURE 2.14: ENTER ROOT PASSWORD

7. For encrypted deployments, JeOS Firstboot does the following:

- Asks for a new passphrase that replaces the default passphrase.
- Generates a new LUKS key and re-encrypts the partition.
- Adds a secondary key slot to the LUKS header and seals it against the TPM device.

If you are deploying an encrypted image, follow these steps:

- a. Select the desired protection method and confirm with *OK*.
- b. Enter a recovery password for LUKS encryption and retype it. The root file system re-encryption begins.

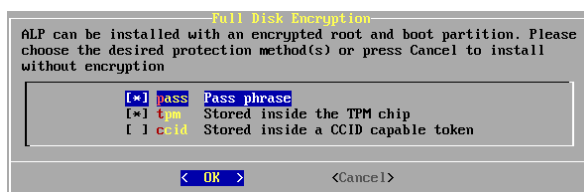


FIGURE 2.15: SELECT METHOD FOR ENCRYPTION


8. ALP Dolomite is successfully deployed using a minimal initial configuration.

### 2.8.3 Next steps

- Install additional software with **transactional-update**. Refer to *Chapter 3, Transactional updates* for more details.
- Install and run additional workloads. Refer to *Chapter 5, SUSE Workloads* for more details.


## 2.8.4 Configuring with Ignition

### 2.8.4.1 What is Ignition?

Ignition (<https://coreos.github.io/ignition/>)  is a provisioning tool that enables you to configure a system according to your specification on the first boot.

### 2.8.4.2 How does Ignition work?

When the system is booted for the first time, Ignition is loaded as part of an `initramfs` and searches for a configuration file within a specific directory (on a USB flash disk, or you can provide a URL). All changes are performed before the kernel switches from the temporary file system to the real root file system (before the `switch_root` command is issued).

Ignition uses a configuration file in the JSON format named `config.ign`. You can either write the configuration manually or use the Fuel Ignition Web application at <https://ignite.opensuse.org>  to generate it.



#### Important

Fuel Ignition does not cover the complete Ignition vocabulary yet, and the resulting JSON file may need additional manual tweaking.



#### Tip

If you decide to write the Ignition configuration manually and prefer the YAML format over JSON, you can create a YAML file and convert this file to JSON using a `Butane` tool. For details, refer to [Section 2.8.4.3, “Converting YAML formatted files into JSON”](#).

#### 2.8.4.2.1 `config.ign`

When installing on bare metal, the configuration file `config.ign` must reside in the `ignition` subdirectory on the configuration media, for example, a USB stick labeled `ignition`. The directory structure must look as follows:

```
<root directory>
```

```
└─ ignition
   └─ config.ign
```



## Tip

To create a disk image with the Ignition configuration, you can use the Fuel Ignition Web application at <https://ignite.opensuse.org>.

If you intend to configure a virtual machine with Virtual Machine Manager ([libvirt](#)), provide the path to the `config.ign` file in its XML definition, for example:

```
<domain ... >
  <sysinfo type="fwcfg">
    <entry name="opt/com.coreos/config" file="/location/to/config.ign"/>
  </sysinfo>
</domain>
```

The `config.ign` contains various data types: objects, strings, integers, booleans and lists of objects. For a complete specification, refer to [Ignition specification v3.3.0 \(https://coreos.github.io/ignition/configuration-v3\\_3/\)](https://coreos.github.io/ignition/configuration-v3_3/).

The `version` attribute is mandatory and in case of ALP Dolomite, its value must be set either to `3.3.0` or to any lower version. Otherwise, Ignition will fail.

If you want to log in to your system as `root`, you must at least include a password for `root`. However, it is recommended to establish access via SSH keys. To configure a password, make sure to use a secure one. If you use a randomly generated password, use at least 10 characters. If you create your password manually, use even more than 10 characters and combine uppercase and lowercase letters and numbers.

### 2.8.4.3 Converting YAML formatted files into JSON

JSON is a universal file format for storing structured data. Applications, for example, Ignition, use it to store and retrieve their configuration. Because JSON's syntax is complex and hard to read for human beings, you can write the configuration in a more friendly format called YAML and then convert it into JSON.

### 2.8.4.3.1 Converting YAML files into JSON format

The tool that converts Ignition-specific vocabularies in YAML files into JSON format is `butane`. It also verifies the syntax of the YAML file to catch potential errors in the structure. For the latest version of `butane`, add the following repository:

```
> sudo zypper ar -f \
  https://download.opensuse.org/repositories/devel:/kubic:/ignition/opensuse_Tumbleweed/
  \
  devel_kubic_ignition
```

Replace `opensuse_Tumbleweed` with one of the following (depending on your distribution):

- `'opensuse_Leap_$releasever'`
- `15.3`

Now you can install the `butane` tool:

```
> sudo zypper ref && zypper in butane
```

After the installation is complete, you can invoke `butane` by running:

```
> butane -p -o config.ign config.fcc
```

- `config.fcc` is the path to the YAML configuration file.
- `config.ign` is the path to the output JSON configuration file.
- The `-p` command option adds line breaks to the output file and thus makes it more readable.

## 2.8.4.4 Ignition configuration examples

### 2.8.4.4.1 Configuration examples

This section provides several examples of the Ignition configuration in both the native JSON format and the YAML format in addition. Note that Ignition does not accept configuration in the YAML format, and you need to convert it to the JSON format. To do so, you can use the `butane` tool as described in [Section 2.8.4.3, “Converting YAML formatted files into JSON”](#).

## Important

[Section 2.8.1.2, “Default partitioning”](#) lists subvolumes that are mounted by default when running the pre-built image. If you want to add a new user or modify any of the files on a subvolume that is not mounted by default, you need to declare such subvolume first so that it is mounted as well. Find more details about mounting file systems in [Section 2.8.4.4.1.1.3, “The filesystems attribute”](#).

## Note: The version attribute is mandatory

Each `config.fcc` must include version 1.4.0 or lower that is then converted to the corresponding Ignition specification.

### 2.8.4.4.1.1 Storage configuration

The `storage` attribute is used to configure partitions, RAID, define file systems, create files, etc. To define partitions, use the `disks` attribute. The `filesystems` attribute is used to format partitions and define mount points of particular partitions. The `files` attribute can be used to create files in the file system. Each of the mentioned attributes is described in the following sections.

#### 2.8.4.4.1.1.1 The disks attribute

The `disks` attribute is a list of devices that enables you to define partitions on these devices. The `disks` attribute must contain at least one `device`, other attributes are optional. The following example will use a single virtual device and divide the disk into four partitions:

JSON:

```
{
  "ignition": {
    "version": "3.0.0"
  },
  "storage": {
    "disks": [
      {
        "device": "/dev/vda",
        "partitions": [
          {
```

```

        "label": "root",
        "number": 1,
        "typeGuid": "4F68BCE3-E8CD-4DB1-96E7-FBCAF984B709"
    },
    {
        "label": "boot",
        "number": 2,
        "typeGuid": "BC13C2FF-59E6-4262-A352-B275FD6F7172"
    },
    {
        "label": "swap",
        "number": 3,
        "typeGuid": "0657FD6D-A4AB-43C4-84E5-0933C84B4F4F"
    },
    {
        "label": "home",
        "number": 4,
        "typeGuid": "933AC7E1-2EB4-4F13-B844-0E14E2AEF915"
    }
],
"wipeTable": true
}
]
}
}

```

## YAML:

```

variant: fcos
version: 1.0.0
storage:
  disks:
    - device: "/dev/vda"
      wipe_table: true
      partitions:
        - label: root
          number: 1
          type_guid: 4F68BCE3-E8CD-4DB1-96E7-FBCAF984B709
        - label: boot
          number: 2
          type_guid: BC13C2FF-59E6-4262-A352-B275FD6F7172
        - label: swap
          number: 3
          type_guid: 0657FD6D-A4AB-43C4-84E5-0933C84B4F4F
        - label: home
          number: 4
          type_guid: 933AC7E1-2EB4-4F13-B844-0E14E2AEF915

```

#### 2.8.4.4.1.1.2 The raid attribute

The `raid` is a list of RAID arrays. The following attributes of `raid` are mandatory:

**level**

a level of the particular RAID array (linear, raid0, raid1, raid2, raid3, raid4, raid5, raid6)

**devices**

a list of devices in the array referenced by their absolute paths

**name**

a name that will be used for the md device

JSON:

```
{
  "ignition": {
    "version": "3.0.0"
  },
  "storage": {
    "raid": [
      {
        "devices": [
          "/dev/sda",
          "/dev/sdb"
        ],
        "level": "raid1",
        "name": "system"
      }
    ]
  }
}
```

YAML:

```
variant: fcos
version: 1.0.0
storage:
  raid:
    - name: system
      level: raid1
      devices:
        - "/dev/sda"
        - "/dev/sdb"
```



#### 2.8.4.4.1.1.3 The `filesystems` attribute

`filesystems` must contain the following attributes:

##### device

the absolute path to the device, typically `/dev/sda` in case of physical disk

##### format

the file system format (btrfs, ext4, xfs, vfat or swap)



### Note

In case of ALP Dolomite, the `root` file system must be formatted to btrfs.

The following example demonstrates using the `filesystems` attribute. The `/opt` directory will be mounted to the `/dev/sda1` partition, which is formatted to btrfs. The device will not be erased.

##### JSON

```
{
  "ignition": {
    "version": "3.0.0"
  },
  "storage": {
    "filesystems": [
      {
        "device": "/dev/sda1",
        "format": "btrfs",
        "path": "/opt",
        "wipeFilesystem": false
      }
    ]
  }
}
```

##### YAML:

```
variant: fcos
version: 1.0.0
storage:
  filesystems:
    - path: /opt
      device: "/dev/sda1"
```

```
format: btrfs
wipe_filesystem: false
```

Normally, a regular user's home directory is located in the `/home/USER_NAME` directory. Since `/home` is not mounted by default in the `initrd`, the mount has to be explicitly defined for the user creation to succeed:

JSON:

```
{
  "ignition": {
    "version": "3.1.0"
  },
  "passwd": {
    "users": [
      {
        "name": "root",
        "passwordHash": "$6$Dxkc092R4JdlFeLE$bf03TPV1n3a4I1to1/2EkfvU2GiSKpR...",
        "sshAuthorizedKeys": [
          "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDCpNOU+nwWRnZYoMV3biUgCC..."
        ]
      }
    ]
  },
  "storage": {
    "filesystems": [
      {
        "device": "/dev/sda3",
        "format": "btrfs",
        "mountOptions": [
          "subvol=/@/home"
        ],
        "path": "/home",
        "wipeFilesystem": false
      }
    ]
  }
}
```

YAML:

```
variant: fcos
version: 1.1.0
storage:
  filesystems:
    - path: /home
      device: /dev/sda3
```

```

    format: btrfs
    wipe_filesystem: false
    mount_options:
      - "subvol=/@/home"
passwd:
  users:
    - name: root
      password_hash: $6$Dxkc092R4JdlFeLE$bf03TPV1n3a4I1to1/2EkfvU2GiSKpR...
      ssh_authorized_keys:
        - ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDCpN0U+nwWRnZYoMV3biUgCC...

```

#### 2.8.4.4.1.1.4 The files attribute

You can use the `files` attribute to create any files on your machine. Bear in mind that if you want to create files outside the default partitioning schema, you need to define the directories by using the `filesystems` attribute.

In the following example, a host name is created by using the `files` attribute. The file `/etc/hostname` will be created with the `alp-1` host name:



### Important

Note that the file mode specification is different for JSON and YAML. While JSON accepts file modes in decimal numbers, for example, `420`, YAML accepts octal numbers (`0644`).

JSON:

```

{
  "ignition": {
    "version": "3.0.0"
  },
  "storage": {
    "files": [
      {
        "overwrite": true,
        "path": "/etc/hostname",
        "contents": {
          "source": "data:,alp-1"
        },
        "mode": 420
      }
    ]
  }
}

```

```
}
```

YAML:

```
variant: fcos
version: 1.0.0
storage:
  files:
    - path: /etc/hostname
      mode: 0644
      overwrite: true
      contents:
        inline: "alp-1"
```

#### 2.8.4.4.1.1.5 The `directories` attribute

The `directories` attribute is a list of directories that will be created in the file system. The `directories` attribute must contain at least one `path` attribute.

JSON:

```
{
  "ignition": {
    "version": "3.0.0"
  },
  "storage": {
    "directories": [
      {
        "path": "/home/tux",
        "user": {
          "name": "tux"
        }
      }
    ]
  }
}
```

YAML:

```
variant: fcos
version: 1.0.0
storage:
  directories:
    - path: /home/tux
      user:
```

```
name: tux
```

#### 2.8.4.4.1.2 Users administration

The `passwd` attribute is used to add users. If you intend to log in to your system, create `root` and set the `root`'s password and/or add the SSH key to the Ignition configuration. You need to hash the `root` password, for example by using the `openssl` command:

```
openssl passwd -6
```

The command creates a hash of the password you chose. Use this hash as the value of the `password_hash` attribute.

JSON:

```
{
  "ignition": {
    "version": "3.0.0"
  },
  "passwd": {
    "users": [
      {
        "name": "root",
        "passwordHash": "$6$PfKm6Fv5Wbq0vZ0C
$g4kByYM.D2B5GCsgluuqDNL87oeXiHqctr6INNNmF75WPGgkLn909uVx4iEe3UdbbhaHbTJ1vpZymKWuDIrWI1",
        "sshAuthorizedKeys": [
          "ssh-rsa long...key user@host"
        ]
      }
    ]
  }
}
```

YAML:

```
variant: fcos
version: 1.0.0
passwd:
  users:
    - name: root
      password_hash: "$6$PfKm6Fv5Wbq0vZ0C
$g4kByYM.D2B5GCsgluuqDNL87oeXiHqctr6INNNmF75WPGgkLn909uVx4iEe3UdbbhaHbTJ1vpZymKWuDIrWI1"
      ssh_authorized_keys:
        - ssh-rsa long...key user@host
```

The `users` attribute must contain at least one `name` attribute. `ssh_authorized_keys` is a list of ssh keys for the user.

#### 2.8.4.4.1.3 Enabling systemd services

You can enable `systemd` services by specifying them in the `systemd` attribute.

JSON:

```
{
  "ignition": {
    "version": "3.0.0"
  },
  "systemd": {
    "units": [
      {
        "enabled": true,
        "name": "sshd.service"
      }
    ]
  }
}
```

YAML:

```
variant: fcos
version: 1.0.0
systemd:
  units:
    - name: sshd.service
      enabled: true
```

The `name` must be the exact name of a service to be enabled (including the suffix).

## 2.8.5 Configuring with Combustion

### 2.8.5.1 What is Combustion?

Combustion is a dracut module that enables you to configure your system on the first boot. You can use Combustion, for example, to change the default partitions, set user passwords, create files, or install packages.

### 2.8.5.2 How does Combustion work?

Combustion is invoked after the `ignition.firstboot` argument is passed to the kernel command line. Combustion reads a provided file named `script`, executes included commands, and thus performs changes to the file system. If `script` includes the network flag, Combustion tries to configure the network. After `/sysroot` is mounted, Combustion tries to activate all mount points in `/etc/fstab` and then calls **transactional-update** to apply other changes, for example, setting `root` password or installing packages.

#### 2.8.5.2.1 The script file

When installing on bare metal, the configuration file `script` must reside in the `combustion` subdirectory on the configuration media labeled `combustion`. The directory structure must look as follows:

```
<root directory>
├─ combustion
│   └─ script
│   └─ other files
```

If you intend to configure a virtual machine with Virtual Machine Manager (`libvirt`), provide the path to the `script` file in its XML definition, for example:

```
<domain ... >
  <sysinfo type="fwcfg">
    <entry name="opt/org.opensuse.combustion/script" file="/location/to/script"/>
  </sysinfo>
</domain>
```



#### Tip: Using Combustion together with Ignition

Combustion can be used along with Ignition. If you intend to do so, label your configuration medium `ignition` and include the `ignition` directory with the `config.ign` to your directory structure as shown below:

```
<root directory>
├─ combustion
│   └─ script
│   └─ other files
├─ ignition
│   └─ config.ign
```

In this scenario, Ignition runs before Combustion.

### 2.8.5.3 Combustion configuration examples

#### 2.8.5.3.1 The script configuration file

The `script` configuration file is a set of commands that are parsed and executed by Combustion in a **transactional-update** shell. This article provides examples of configuration tasks performed by Combustion.



#### Important: Include interpreter declaration

As the `script` file is interpreted by Bash, always start the file with the interpreter declaration at its first line:

```
#!/usr/bin/bash
```

To log in to your system, include at least the `root` password. However, it is recommended to establish the authentication using SSH keys. If you need to use a `root` password, make sure to configure a secure password. If you use a randomly generated password, use at least 10 characters. If you create your password manually, use even more than 10 characters and combine uppercase and lowercase letters and numbers.

#### 2.8.5.3.1.1 Network configuration

To configure and use the network connection during the first boot, add the following statement to `script`:

```
# combustion: network
```

Using this statement will pass the `rd.neednet=1` argument to dracut. If you do not use the statement, the system will be configured without any network connection.



### 2.8.5.3.1.2 Partitioning

ALP Dolomite raw images are delivered with a default partitioning scheme as described in [Section 2.8.1.2, “Default partitioning”](#). You might want to use a different partitioning. The following set of example snippets moves the /home to a different partition.



#### Note: Performing changes outside of directories included in snapshots

The following script performs changes that are not included in snapshots. If the script fails and the snapshot is discarded, some changes remain visible and cannot be reverted, for example, the changes to the /dev/vdb device.

The following snippet creates a GPT partitioning schema with a single partition on the /dev/vdb device:

```
sfdisk /dev/vdb <<EOF
label: gpt
type=linux
EOF

partition=/dev/vdb1
```

The partition is formatted to BTRFS:

```
wipefs --all ${partition}
mkfs.btrfs ${partition}
```

Possible content of /home is moved to the new /home folder location by the following snippet:

```
mount /home
mount ${partition} /mnt
rsync -aAXP /home/ /mnt/
umount /home /mnt
```

The snippet below removes an old entry in /etc/fstab and creates a new entry:

```
awk -i inplace '$2 != "/home"' /etc/fstab
echo "${(blkid -o export ${partition} | grep ^UUID=) /home btrfs defaults 0 0}" >>/etc/
fstab
```

#### 2.8.5.3.1.3 Setting a password for root

Before you set the `root` password, generate a hash of the password, for example, by using the `openssl passwd -6`. To set the password, add the following to the `script`:

```
echo 'root:$5$.wn2BZHLEJ5R3B1C$TAHEchlU.h2tvf0p0ki54NaHpGYKwdNhjaBuSpDotD7' | chpasswd -e
```

#### 2.8.5.3.1.4 Adding SSH keys

The following snippet creates a directory to store the `root`'s SSH key and then copies the public SSH key located on the configuration device to the `authorized_keys` file.

```
mkdir -pm700 /root/.ssh/  
cat id_rsa_new.pub >> /root/.ssh/authorized_keys
```



### Note

The SSH service must be enabled in case you need to use remote login via SSH. For details, refer to [Section 2.8.5.3.1.5, “Enabling services”](#).

#### 2.8.5.3.1.5 Enabling services

To enable system services, for example, the SSH service, add the following line to `script`:

```
systemctl enable sshd.service
```

#### 2.8.5.3.1.6 Installing packages



### Important: Network connection and registering your system may be necessary

As some packages may require additional subscription, you may need to register your system beforehand. An available network connection may also be needed to install additional packages.

During the first boot configuration, you can install additional packages to your system. For example, you can install the `vim` editor by adding:

```
zypper --non-interactive install vim-small
```



## Note

Bear in mind that you will not be able to use `zypper` after the configuration is complete and you boot to the configured system. To perform changes later, you must use the `transactional-update` command to create a changed snapshot.

## 2.9 Post-deployment considerations

This article includes important information and tasks that you need to consider after you successfully deploy SUSE ALP Dolomite (ALP Dolomite).

### 2.9.1 Full disk encryption

#### 2.9.1.1 Change encryption password

During the ALP Dolomite deployment, you entered a password that is used for disk encryption. To change the password, run the following command:

```
# fdectl passwd
```

#### 2.9.1.2 TPM device

Without a TPM chip, you need to enter the encryption password to decrypt the disk on each ALP Dolomite boot. On systems that have a TPM 2.0 chip, ALP Dolomite deployed with the Agama installer supports the automatic protection of the LUKS volume with a TPM device. The requirement is that the machine must use the UEFI Secure Boot enabled.

If the Agama installer detects a TPM 2.0 chip and UEFI Secure Boot, it creates a secondary LUKS key. On the first boot, ALP Dolomite uses the TPM to protect this key and configure the GRUB 2 boot loader to unwrap the key automatically. Be aware that you must remove the ISO after the

installer has finished and before the system boots for the first time. This is because we use the TPM to ensure that the system comes up with exactly the same configuration before unlocking the LUKS partition.

This allows you to use the full disk encryption without having to type the disk password on each reboot. However, the disk password is still there and can be used for recovery. For example, after updating the GRUB 2 boot loader or the SHIM loader, the TPM can no longer unseal the secondary key correctly, and GRUB 2 has to fall back to the password.

## 2.9.2 SELinux

Security-Enhanced Linux (SELinux) is a security framework that increases system security by defining access controls for applications, processes and files on the file system.

ALP Dolomite ships with SELinux enabled and set to the restrictive *enforce* mode for increased security. The enforce mode can lead to processes or workloads not behaving correctly because the default policy may be too strict. If you observe such unexpected issues, set SELinux to the *permissive* mode that does not enforce SELinux policies but still logs offenses against them called *Access Vector Rules* (AVCs).

To set SELinux to the permissive mode temporarily, run:

```
# setenforce 0
```



### Tip

To set SELinux to the permissive mode permanently, edit `/etc/selinux/config` and update it to include the following line:

```
SELINUX=permissive
```



### Important

If you entered an SELinux permissive mode, you need to relabel your system until it is back in a good state. The reason is that the permissive mode allows you to reach states that are not reachable otherwise. To relabel the system, run the following command and reboot the system:

```
# touch /etc/selinux/.autorelabel
```

```
# reboot
```

To monitor AVCs, search the Audit log and `systemd` journal for log messages similar to the following one:

```
type=AVC msg=audit(1669971354.731:25): avc: denied { create } \
for pid=1264 comm="ModemManager" scontext=system_u:system_r:modemmanager_t:s0 \
tcontext=system_u:system_r:modemmanager_t:s0 tclass=qipcrtr_socket permissive=0
```

To filter such messages, you can use the following commands:

```
# tail -f /var/log/audit/audit.log | grep -i AVC
```

and

```
# journalctl -f | grep -i AVC
```

For more advanced search, use the following command:

```
# ausearch -m avc,user_avc,selinux_err -i
```

If such messages appear while using the application that did not behave correctly when SELinux was set to the enforce mode, the policies are too restrictive and need updating. You can help to fine-tune SELinux policies by creating a bug report at [https://bugzilla.suse.com/enter\\_bug.cgi?classification=SUSE%20ALP%20-%20SUSE%20Adaptable%20Linux%20Platform](https://bugzilla.suse.com/enter_bug.cgi?classification=SUSE%20ALP%20-%20SUSE%20Adaptable%20Linux%20Platform). Specify `Basesystem` as a component, include the word `SELinux` in the bug subject, and attach the gathered unique lines that include AVCs together with reproduction steps.

### 2.9.3 Enabling root login via SSH

`root` login via SSH is not permitted in ALP Dolomite by default for security reasons. To enable it, you have the following options:

- install the `openssh-server-config-rootlogin` package and reboot the system.

```
# transactional-update pkg in openssh-server-config-rootlogin
# reboot
```

- Add a file containing the snippet `PermitRootLogin yes` in the `/etc/ssh/sshd_config.d/` directory and reboot, for example:

```
# echo 'PermitRootLogin yes' >> /etc/ssh/sshd_config.d/root_login_config
```

## 2.10 For more information

- Find detailed information about using the Virtual Machine Manager in <https://documentation.suse.com/sles/html/SLES-all/cha-kvm-inst.html> and <https://documentation.suse.com/sles/15-SP4/html/SLES-all/cha-libvirt-config-gui.html>.
- Transactional updates are described in *Chapter 3, Transactional updates*.
- Installing software packages and patterns is detailed in <https://documentation.suse.com/sles/html/SLES-all/cha-sw-cl.html>.
- The SELinux framework is detailed in <https://documentation.suse.com/sles/html/SLES-all/cha-selinux.html>.

## 3 Transactional updates

### 3.1 What are transactional updates?

To keep the base operating system stable and consistent, the SUSE ALP Dolomite uses a read-only root file system. Therefore, you cannot perform direct changes to the root file system, for example, by using the **zypper** command. Instead, ALP Dolomite introduces *transactional updates* that allow you to apply one or more changes to the root file system.

### 3.2 How do transactional updates work?

Each time you call the **transactional-update** command to change your system—either to install a package, perform an update, or apply a patch—the following actions take place:

#### PROCEDURE 3.1: MODIFYING THE ROOT FILE SYSTEM

1. A new read-write snapshot is created from your current root file system, or from a snapshot that you specified.
2. All changes are applied (updates, patches or package installation).
3. The snapshot is switched back to read-only mode.
4. If the changes were applied successfully, the new root file system snapshot is set as default.
5. After rebooting, the system boots into the new snapshot.

### 3.3 Benefits of transactional updates

- They are atomic—the update is applied only if it completes successfully.
- Changes are applied in a separate snapshot and so do not influence the running system.
- Changes can easily be rolled back.

### 3.3.1 Environment within transactional-update command

Each time you run the **transactional-update** command, the changes are performed in a new snapshot. The environment in the snapshot may differ from the one in the shell you run the **transactional-update** command from. For example, the current working directory (`$PWD`) is not set to the directory from which you run the **transactional-update**, but is set to `/`.

From within the snapshot, you cannot access the `/var` directory. This directory is also not included in the snapshot as described in [Section 3.5, “Directories excluded from snapshots”](#). However, some directories are not included in the snapshot but are accessible inside the **transactional-update** environment, for example, the `/root` directory.

## 3.4 `/etc` on a read-only file system

Even though `/etc` is part of the read-only file system, using an `OverlayFS` layer on this directory enables you to write to this directory. All modifications that you performed on the content of `/etc` are written to the `/var/lib/overlay/SNAPSHOT_NUMBER/etc`. Each snapshot has one associated `OverlayFS` directory.

Whenever a new snapshot is created (for example, as a result of a system update), the content of `/etc` is synchronized and used as a base in the new snapshot. In the `OverlayFS` terminology, the current snapshot's `/etc` is mounted as `lowerdir`. The new snapshot's `/etc` is mounted as `upperdir`. If there were no changes in the `upperdir` `/etc`, any changes performed to the `lowerdir` are visible to the `upperdir`. Therefore, the new snapshot also contains the changes from the current snapshot's `/etc`.



### Important: Concurrent modification of `lowerdir` and `upperdir`

If `/etc` in both snapshots is modified, only the changes in the new snapshot (`upperdir`) persist. Changes made to the current snapshot (`lowerdir`) are not synchronized to the new snapshot. Therefore, we do not recommend changing `/etc` after a new snapshot has been created and the system has not been rebooted. However, you can still find the changes in the `/var/lib/overlay/` directory for the snapshot in which the changes were performed.





Note: Using the `--continue` option of the **transactional-update** command

When using the `--continue` option and the new snapshot is a descendant of the current snapshot, then the `/etc` overlays of all the snapshots in between will be added as additional directories to the `lowerdir` (the `lowerdir` can have several mount points).

## 3.5 Directories excluded from snapshots

As certain directories store user-specific or volatile data, these directories are excluded from snapshots:

### /home

Contains users' data. Excluded so that the data is not included in snapshots and thus potentially overwritten by a rollback operation.

### /root

Contains `root` data. Excluded so that the data is not included in snapshots and thus potentially overwritten by a rollback operation.

### /opt

Third-party products are usually installed to `/opt`. Excluded so that these applications are not uninstalled during rollbacks.

### /srv

Contains data for Web and FTP servers. Excluded to avoid data loss on rollbacks.

### /usr/local

This directory is used when manually installing software. It is excluded to avoid uninstalling these installations on rollbacks.

### /var

This directory contains many variable files, including logs, temporary caches, third-party products in `/var/opt`, and is the default location for virtual machine images and databases. Therefore, a separate subvolume is created with Copy-On-Write disabled to exclude all such variable data from snapshots.

### /tmp

The directory contains temporary data.

the architecture-specific `/boot/grub2` directory

Rollback of the boot loader binaries is not supported.

## 3.6 Applying multiple changes without reboot

The default **`transactional-update`** behavior is to create a new snapshot from the current root file system after each change. To apply the changes, you need to reboot the host. You cannot run the **`transactional-update`** command multiple times without rebooting to add more changes to the snapshot, because this creates separate independent snapshots that do not include changes from the previous snapshots.

To make multiple changes to the root file system, you have several options, which are described in the following sections:

### 3.6.1 The **`transactional-update --continue`** option

Use the **`transactional-update`** command together with the `--continue` option to make multiple changes without rebooting. A separate snapshot is created on each run that contains all changes from the previous snapshot, plus your new changes. The final snapshot includes all changes. To apply them, reboot the system and your final snapshot becomes the new root file system.

### 3.6.2 The **`transactional-update run`** command

The **`transactional-update run`** command normally runs only a single command. However, you can use it to run multiple commands in one transactional session by concatenating them within a command shell such as **`bash`**, for example:

```
# transactional-update run bash -c 'ls && date; if [ true ]; then echo -n "Hello "; echo  
'\''world'\''; fi'
```



## Note

The **transactional-update run** command has the same limitations as the **transactional-update shell** command described in [Section 3.6.3, “The transactional-update shell”](#) except that the entered commands are logged in the `/transactional-update.log` file.

### 3.6.3 The transactional-update shell

The **transactional-update shell** command opens a shell in the transactional-update environment. In the shell, you can enter almost any Linux command to make changes to the file system, for example, install multiple packages with the **zypper** command or perform changes to files that are part of the read-only file system. You can also verify that the changes you previously made with the **transactional-update** command are correct.



## Important

The transactional shell has several limitations. For example, you cannot operate start or stop services using `systemd` commands, or modify the `/var` partition because it is not mounted. Also, commands entered during a shell session are not logged in the `/transactional-update.log` file.

All changes that you make to the file system are part of a single snapshot. After you finish making changes to the file system and leave the shell with the **exit** command, you need to reboot the host to apply the changes.



## Tip

Another approach to making multiple changes to the file system without rebooting the host is to use the `--continue` option. For more details, refer to [--continue \[number\], -c](#).

## 3.7 Usage of the **transactional-update** command

The **transactional-update** command enables the atomic installation or removal of updates. Updates are applied only if all can be successfully installed. **transactional-update** creates a snapshot of your system and uses it to update the system. Later you can restore this snapshot. All changes become active only after reboot.

The **transactional-update** command syntax is as follows:

```
transactional-update [option] [general_command] [package_command] standalone_command
```



### Note: Running **transactional-update** without arguments

If you do not specify any command or option while running the **transactional-update** command, the system updates itself.

Possible command parameters are described further.

#### **transactional-update** OPTIONS

--interactive, -i

Can be used along with a package command to turn on interactive mode.

--non-interactive, -n

Can be used along with a package command to turn on non-interactive mode.

--continue [number], -c

The --continue option is for making multiple changes to the root file system without rebooting. Refer to [Section 3.6, “Applying multiple changes without reboot”](#) for more details.

Another useful feature of the --continue option is that you may select any existing snapshot as the base for your new snapshot. The following example demonstrates running **transactional-update** to install a new package in a snapshot based on snapshot 13, and then running it again to install another package:

```
# transactional-update pkg install package_1
```

```
# transactional-update --continue 13 pkg install package_2
```

--no-selfupdate

Disables self-updating of **transactional-update**.

--drop-if-no-change, -d

Discards the snapshot created by transactional-update if there were no changes to the root file system. If there are changes to the /etc directory, those changes merged back to the current file system.

--quiet

The transactional-update command does not output to stdout .

--help, -h

Prints help for the transactional-update command.

--version

Displays the version of the transactional-update command.

The general commands are the following:

## GENERAL COMMANDS

### cleanup-snapshots

The command marks all unused snapshots that are intended to be removed.

### cleanup-overlays

The command removes all unused overlay layers of /etc .

### cleanup

The command combines the cleanup-snapshots and cleanup-overlays commands.

### grub.cfg

Use this command to rebuild the GRUB boot loader configuration file.

### bootloader

The command reinstalls the boot loader.

### initrd

Use the command to rebuild initrd .

### kdump

In case you perform changes to your hardware or storage, you may need to rebuild the Kdump initrd.

### shell

Opens a read-write shell in the new snapshot before exiting. The command is typically used for debugging purposes.

## **reboot**

The system reboots after the **transactional-update** command is complete.

## **run <command>**

Runs the provided command in a new snapshot.

## **setup-selinux**

Installs and enables targeted SELinux policy.

The package commands are the following:

### **PACKAGE COMMANDS**

## **dup**

Performs upgrade of your system. The default option for this command is **--non-interactive**.

## **migration**

The command migrates your system to a selected target. Typically, it is used to upgrade your system if it has been registered via SUSE Customer Center.

## **patch**

Checks for available patches and installs them. The default option for this command is **--non-interactive**.

## **pkg install**

Installs individual packages from the available channels using the **zypper install** command. This command can also be used to install Program Temporary Fix (PTF) RPM files. The default option for this command is **--interactive**.

```
# transactional-update pkg install package_name
```

or

```
# transactional-update pkg install rpm1 rpm2
```

Or, to install a software pattern:

```
# transactional-update pkg install -t pattern pattern_name
```

## **pkg remove**

Removes individual packages from the active snapshot using the **zypper remove** command. This command can also be used to remove PTF RPM files. The default option for this command is **--interactive**.

```
# transactional-update pkg remove package_name
```

### **pkg update**

Updates individual packages from the active snapshot using the **zypper update** command. Only packages that are part of the snapshot of the base file system can be updated. The default option for this command is `--interactive`.

```
# transactional-update pkg update package_name
```

### **register**

Registers or deregisters your system. For a complete usage description, refer to [Section 3.7.1, “The register command”](#).

### **up**

Updates installed packages to newer versions. The default option for this command is `--non-interactive`.

The stand-alone commands are the following:

#### STAND-ALONE COMMANDS

### **rollback <snapshot number>**

This sets the default subvolume. The current system is set as the new default root file system. If you specify a number, that snapshot is used as the default root file system. On a read-only file system, it does not create any additional snapshots.

```
# transactional-update rollback snapshot_number
```

### **rollback last**

This command sets the last known to be working snapshot as the default.

### **status**

This prints a list of available snapshots. The currently booted one is marked with an asterisk, the default snapshot is marked with a plus sign.

## 3.7.1 The **register** command

The **register** command enables you to handle all tasks regarding registration and subscription management. You can supply the following options:

### **--list-extensions**

With this option, the command lists available extensions for your system. You can use the output to find a product identifier for product activation.

#### -p, --product

Use this option to specify a product for activation. The product identifier has the following format: `<name>/<version>/<architecture>`, for example, `sle-module-live-patching/15.3/x86_64`. The corresponding command has the following form:

```
# transactional-update register -p sle-module-live-patching/15.3/x86_64
```

#### -r, --regcode

Register your system with the registration code provided. The command registers the subscription and enables software repositories.

#### -d, --de-register

The option deregisters the system, or when used along with the `-p` option, deregisters an extension.

#### -e, --email

Specify an email address that is used in SUSE Customer Center for registration.

#### --url

Specify the URL of your registration server. The URL is stored in the configuration and is used in subsequent command invocations. For example:

```
# transactional-update register --url https://scc.suse.com
```

#### -s, --status

Displays the current registration status in JSON format.

#### --write-config

Writes the provided options value to the `/etc/SUSEConnect` configuration file.

#### --cleanup

Removes old system credentials.

#### --version

Prints the version.

#### --help

Displays the usage of the command.



## 4 Containers and Podman

### 4.1 What are containers and Podman?

Containers offer a lightweight virtualization method to run multiple virtual environments (containers) simultaneously on a single host. Unlike technologies such as Xen or KVM, where the processor simulates a complete hardware environment and a hypervisor controls virtual machines, containers provide virtualization on the operating system level, where the kernel controls the isolated containers.

*Podman* is a short name for Pod Manager Tool. It is a daemonless container engine that enables you to run and deploy applications using containers and container images. Podman provides a command line interface to manage containers.

### 4.2 How does Podman work?

Podman provides integration with systemd. This way you can control containers via systemd units. You can create these units for existing containers as well as generate units that can start containers if they do not exist in the system. Moreover, Podman can run systemd inside containers.

Podman enables you to organize your containers into pods. Pods share the same network interface and resources. A typical use case for organizing a group of containers into a pod is a container that runs a database and a container with a client that accesses the database.

#### 4.2.1 Pods architecture

A pod is a group of containers that share the same namespace, ports and network connection. Usually, containers within one pod can communicate directly with each other. Each pod contains an infrastructure container ( INFRA ), whose purpose is to hold the namespace. INFRA also enables Podman to add other containers to the pod. Port bindings, cgroup-parent values, and kernel namespaces are all assigned to the infrastructure container. Therefore, later changes of these values are not possible.

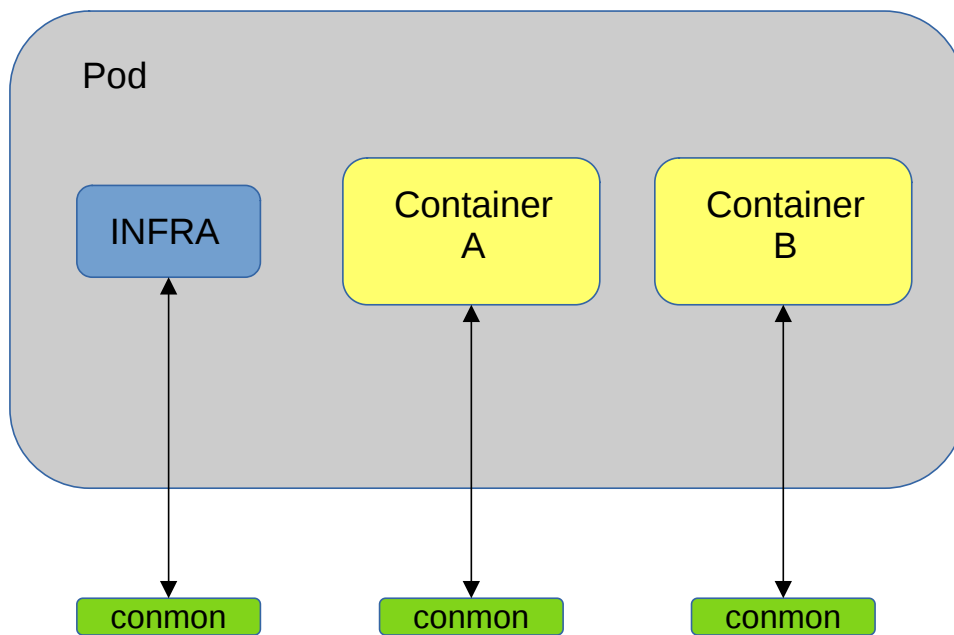


FIGURE 4.1: PODS ARCHITECTURE

Each container in a pod has its own instance of a monitoring program. The monitoring program watches the container's process and if the container dies, the monitoring program saves its exit code. The program also holds open the tty interface for the particular container. The monitoring program enables you to run containers in the detached mode when Podman exits, because this program continues to run and enables you to attach tty later.

## 4.3 Benefits of containers

- Containers make it possible to isolate applications in self-contained units.
- Containers provide near-native performance. Depending on the runtime, a container can use the host kernel directly, thus minimizing overhead.
- It is possible to control network interfaces and apply resources inside containers through kernel control groups.

## 4.4 Enabling Podman

This article helps you verify that Podman is installed on the ALP Dolomite system and provides guidelines to enable its systemd service when Cockpit requires it.

### 4.4.1 Requirements

- Deployed ALP Dolomite base OS.

### 4.4.2 Installing Podman

1. Verify that Podman is installed on your system by running the following command:

```
# zypper se -i podman
```

2. If Podman is not listed in the output, install it by running:

```
# transactional-update pkg install podman*
```

3. Reboot the ALP Dolomite host for the changes to take effect.
4. Optionally, enable and start the podman.service service for applications that require it, such as Cockpit. You can enable it either in Cockpit by clicking *Podman containers* > *Start podman*, or by running the following command:

```
# systemctl enable --now podman.service
```

### 4.4.3 Enabling rootless mode

By default, Podman requires root privileges. To enable rootless mode for the current user, run the following command:

```
> sudo usermod --add-subuids 100000-165535 \  
--add-subgids 100000-165535 USER
```

Reboot the machine to enable the change. The command above defines a range of local UIDs to which the UIDs allocated to users inside the container are mapped on the host. Note that the ranges defined for different users must not overlap. It is also important that the ranges do not reuse the UID of an existing local user or group. By default, adding a user with the useradd command automatically allocates subUID and subGID ranges.



#### Note: Limitations of rootless containers

Running a container with Podman in rootless mode on SLE Micro may fail, because the container might need access to directories or files that require root privileges.

### 4.4.4 Next steps

- Run containerized workloads. For details, refer to *Chapter 5, SUSE Workloads*.

## 4.5 Autostarting containers

Podman does not have a command-line option to enable autostarting a specific container after the system boot. Podman can, however, create a systemd service for an existing container. After you enable such service, the container starts on every system boot.

#### PROCEDURE 4.1: CREATING AND ENABLING A CONTAINER SERVICE AS A NON-root USER

1. Identify the container that you want to start on system boot.

```
> podman ps  
CONTAINER ID   IMAGE [...]  
9a0fdeee9320   registry.opensuse.org/.../cockpit-ws:latest   cockpit-ws
```

2. Create a systemd unit file for the service related to the cockpit-ws container and save it as container-cockpit-ws.service in the current directory.

```
> podman generate systemd --new --name cockpit-ws \
> container-cockpit-ws.service
```

3. Move the `systemd` unit file to `~/.config/systemd/user/` in the user's home directory.

```
> mv container-cockpit-ws.service \
~/.config/systemd/user/
```

4. Make `systemd` aware of the previously created unit file.

```
> sudo systemctl --user daemon-reload
```

5. Enable the `systemd` service.

```
> systemctl --user enable container-cockpit-ws.service
```

6. Because the service is run by a non-`root` user, you need to make sure that the user is logged in at boot and stays active even after they are logged out of a display manager or a terminal session. This mechanism is called *lingering* and is achieved by the following command:

```
> sudo loginctl enable-linger USER_NAME
```

7. Reboot the host and check if the service is running.

```
> systemctl status --user
container-cockpit-ws.service
```

#### PROCEDURE 4.2: CREATING AND ENABLING A CONTAINER SERVICE AS A `root` USER

1. Identify the container that you want to start on system boot.

```
> podman ps
CONTAINER ID   IMAGE [...]          NAMES
9a0fdeee9320   registry.opensuse.org/.../cockpit-ws:latest   cockpit-ws
```

2. Create a `systemd` unit file for the service related to the `cockpit-ws` container and save it as `container-cockpit-ws.service` in the current directory.

```
> podman generate systemd --new --name cockpit-ws \
> container-cockpit-ws.service
```

3. Move the `systemd` unit file to `/etc/systemd/system/`.

```
> sudo mv container-cockpit-ws.service \
```

```
/etc/systemd/system/
```

4. Make `systemd` aware of the previously created unit file.

```
> sudo systemctl daemon-reload
```

5. Enable the `systemd` service.

```
> sudo systemctl enable container-cockpit-ws.service
```

6. Reboot the host and check if the service is running.

```
> systemctl status container-cockpit-ws.service
```

## 4.6 Podman usage

This article introduces basic Podman usage that you may need when running containerized workloads.

### 4.6.1 Getting container images

To run a container, you need an image. An image includes all dependencies needed to run an application. You can obtain images from an image registry. Available registries are defined in the `/etc/containers/registries.conf` configuration file. If you have a local image registry or want to use other registries, add the registries into the configuration file.



#### Important: No tools for building images in ALP Dolomite

ALP Dolomite does not provide tools for building custom images. Therefore, the only way to get an image is to pull it from an image registry.

The `podman pull` command pulls an image from an image registry. The syntax is as follows:

```
# podman pull [OPTIONS] SOURCE
```

The `source` can be an image without the registry name. In that case, Podman tries to pull the image from all registries configured in the `/etc/containers/registries.conf` file. The default image tag is `latest`. The default location of pulled images is `/var/lib/containers/storage/overlay-images/`.

To view all possible options of the `podman pull` command, run:

```
# podman pull --help
```



### Note: Getting images using Cockpit

If you are using Cockpit, you can also pull images from an image registry in the *Podman containers* menu by clicking `+ Get new image`.

Podman enables you to search for images in an image registry or a list of registries using the command:

```
# podman search IMAGE_NAME
```

## 4.6.2 Working with containers

The following section covers common container management tasks. This includes creating, starting, and modifying containers.



### Warning

The current version of ALP Dolomite does not provide tools for building custom images. Therefore, the only way to get a container image is to pull it from an image registry.

### 4.6.2.1 Running containers



### Tip

For specific details on running ALP Dolomite containers, refer to links in the [Chapter 5, SUSE Workloads](#) article.

After you have pulled your container image, you can create containers based on it. You can run an instance of the image using the `podman run` command. The command syntax is as follows:

```
# podman run [OPTIONS] IMAGE [CONTAINER_NAME]
```

*IMAGE* is specified in format *transport:path*. If *transport* is omitted, the default *docker* is used. The *path* can reference to a specific image registry. If omitted, Podman searches for the image in registries defined in the `/etc/containers/registries.conf` file. An example that runs a container called `sles15` based on the `sle15` image follows:

```
# podman run registry.opensuse.org/suse/templates/images/sle-15-sp3/base/images/suse/sle15 sle15
```

Below is a list of frequently used options. For a complete list of available options, run the command: **podman run --help**.

**--detach, -d**

The container will run in the background.

**--env, -e=env**

This option allows arbitrary environment variables that are available for the process to be launched inside of the container. If an environment variable is specified without a value, Podman will check the host environment for a value and set the variable only if it is set on the host.

**--help**

Prints help for the **podman run** command.

**--hostname=*name*, -h**

Sets the container host name that is available inside the container.

**--pod=*name***

Runs the container in an existing pod. To create a pod, prefix the pod name with **new: .**

**--read-only**

Mounts the container's root file system as read-only.

**--systemd=true|false|always**

Runs the container in systemd mode. The default is true.

#### 4.6.2.2 Stopping containers

If the **podman run** command finished successfully, a new container has been started. You can stop the container by running:

```
# podman stop [OPTIONS] CONTAINER
```



You can specify a single container name or ID or a space-separated list of containers. The command takes the following options:

`--all, -a`

Stops all running containers.

`--latest, -l`

Instead of providing a container name, the last created container will be stopped.

`--time, -t=seconds`

Seconds to wait before forcibly stopping the container.

To view all possible options of the **podman stop** command, run the following:

```
# podman stop --help
```

#### 4.6.2.3 Starting containers

To start already created but stopped containers, use the **podman start** command. The command syntax is as follows:

```
# podman start [OPTIONS] CONTAINER
```

*CONTAINER* can be a container name or a container ID.

For a complete list of possible options of **podman start**, run the command:

```
# podman start --help
```

#### 4.6.2.4 Updating containers

To update an existing container, follow these steps:

1. Identify the image of the container that you want to update, for example, `yast-mgmt-qt`:

```
> podman image ls
REPOSITORY                                TAG          IMAGE ID      CREATED      SIZE
[...]
registry.opensuse.org/suse/alp/workloads/publish/tumbleweed_containerfiles/suse/alp/
workloads/yast-mgmt-qt latest       f349194a439d 13 days ago  674 MB
```

2. Pull the image from the registry to find out if there is a newer version. If you do not specify a version tag, the `latest` tag is used:

```
# podman pull registry.opensuse.org/suse/alp/workloads/publish/
tumbleweed_containerfiles/suse/alp/workloads/yast-mgmt-qt
Trying to pull registry.opensuse.org/suse/alp/workloads/publish/
tumbleweed_containerfiles/suse/alp/workloads/yast-mgmt-qt:latest...
Getting image source signatures
Copying blob 6bfbcdeee2ec done
[...]
Writing manifest to image destination
Storing signatures
f349194a439da249587fbd8baffc5659b390aa14c8db1d597e95be703490ffb1
```

3. If the container is running, identify its ID and stop it:

```
# podman ps
CONTAINER ID   IMAGE
               COMMAND          CREATED          STATUS
[...]
28fef404417b   /workloads/tumbleweed_containerfiles/suse/alp/workloads/yast-mgmt-
ncurses:latest 2 weeks ago      Up 24 seconds ago
# podman stop 28fef404417b
```

4. Run the container following specific instructions at [Chapter 5, SUSE Workloads](#), for example:

```
# podman container runlabel run \
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/
workloads/yast-mgmt-ncurses:latest
```

#### 4.6.2.5 Committing modified containers

You can run a new container with specific attributes that are not part of the original image. To save the container with these attributes as a new image, you can use the `podman commit` command:

```
# podman commit [OPTIONS] CONTAINER IMAGE
```

`CONTAINER` is a container name or a container ID. `IMAGE` is the new image name. If the image name does not start with a registry name, the value `localhost` is used.

When using Cockpit, you can perform the **commit** operation directly from a container's *Details*, by clicking *Commit*. A dialog box opens. Specify all required details as shown below and click *Commit*:

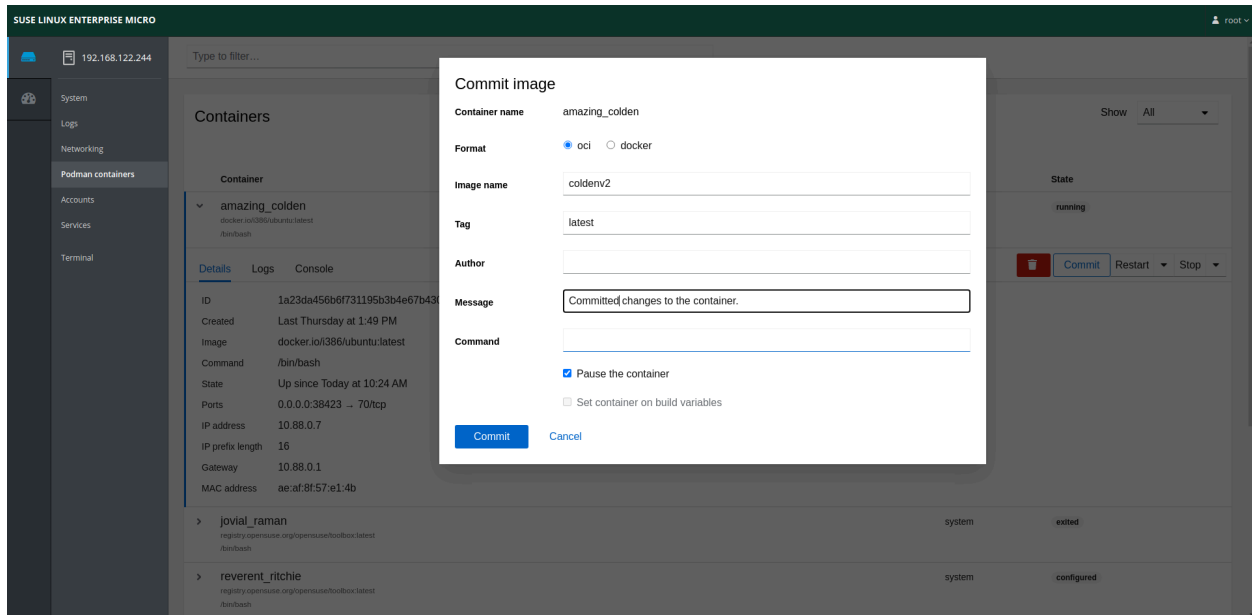


FIGURE 4.2: COMMITTING A CONTAINER IN COCKPIT

#### 4.6.2.6 Listing containers

Podman enables you to list all running containers using the **podman ps** command. The generic syntax of the command is as follows:

```
# podman ps [OPTIONS]
```

Command options can change the displayed information. For example, using the **--all** option will output all containers created by Podman (not only the running containers).

For a complete list of **podman ps** options, run:

```
# podman ps --help
```

#### 4.6.2.7 Removing containers

To remove one or more unused containers from the host, use the **podman rm** command as follows:

```
# podman rm [OPTIONS] CONTAINER
```

CONTAINER can be a container name or a container ID.

The command does not remove the specified container if the container is running. To remove a running container, use the `-f` option.

For a complete list of **podman rm** options, run:

```
# podman rm --help
```



### Note: Deleting all stopped containers

You can delete all stopped containers from your host with a single command:

```
# podman container prune
```

Make sure that each stopped container is intended to be removed before you run the command, otherwise you might remove containers that are still in use and were stopped only temporarily.

## 4.6.3 Working with pods

Containers can be grouped into a pod. The containers in the pod then share network, pid, and IPC namespace. Pods can be managed by **podman pod** commands. This section provides an overview of the commands for managing pods.

### 4.6.3.1 Creating pods

The command **podman pod create** is used to create a pod. The syntax of the command is as follows:

```
# podman pod create [OPTIONS]
```

The command outputs the pod ID. By default, the pods are created without being started. You can start a pod by running a container in the pod, or by starting the pod as described in [Section 4.6.3.3, "Starting/stopping/restarting pods"](#).



### Note: Default pod names

If you do not specify a pod name with the `--name` option, Podman will assign a default name for the pod.

For a complete list of possible options, run the following command:

```
# podman pod create --help
```

#### 4.6.3.2 Listing pods

You can list all pods by running the command:

```
# podman pod list
```

The output looks as follows:

POD ID	NAME	STATUS	CREATED	# OF CONTAINERS	INFRA ID
30fba506fecb	upbeat_mcclintock	Created	19 hours ago	1	4324f40c9651
976a83b4d88b	nervous_feynman	Running	19 hours ago	2	daa5732ecd02

As each pod includes the INFRA container, the number of containers in a pod is always larger than zero.

#### 4.6.3.3 Starting/stopping/restarting pods

After a pod is created, you must start it, as it is not in the state running by default. In the commands below, POD can be a pod name or a pod ID.

To start a pod, run the command:

```
# podman pod start [OPTIONS] POD
```

For a complete list of possible options, run:

```
# podman pod start --help
```

To stop a pod, use the podman pod stop as follows:

```
# podman pod stop POD
```

To restart a pod, use the podman pod restart command as follows:

```
# podman pod restart POD
```

#### 4.6.3.4 Managing containers in a pod

To add a new container to a pod, use the `podman run` command with the option `--pod`. A general syntax of the command follows:

```
# podman run [OPTIONS] --pod POD_NAME IMAGE
```

For details about the `podman run` command, refer to [Section 4.6.2.1, “Running containers”](#).



#### Note: Only new containers can be added to a pod

The `podman start` command does not allow for starting a container in a pod if the container was not added to the pod during the container's initial running.

You cannot remove a container from a pod and keep the container running, because the container itself is removed from the host.

Other actions like start, restart and stop can be performed on specific containers without affecting the status of the pod.

#### 4.6.3.5 Removing pods

There are two ways to remove pods. You can use the `podman pod rm` command to remove one or more pods. Alternatively, you can remove all stopped pods using the `podman pod prune` command.

To remove a pod or several pods, run the `podman pod rm` command as follows:

```
# podman pod rm POD
```

`POD` can be a pod name or a pod ID.

To remove all currently stopped pods, use the `podman pod prune` command. Make sure that all stopped pods are intended to be removed before you run the `podman pod prune` command, otherwise you might remove pods that are still in use.

#### 4.6.3.6 Monitoring processes in pods

To view all containers in all pods, use the following command:


```
# podman ps -a --pod
```

The output of the command will be similar to the following one:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	[...]
4324f40c9651	k8s.gcr.io/pause:3.2		21 hours ago	Created	
daa5732ecd02	k8s.gcr.io/pause:3.2		22 hours ago	Up 3 hours ago	
e5c8e360c54b	localhost/test:latest	/bin/bash	3 days ago	Exited (137) 3 days ago	
82dad15828f7	localhost/opensuse/toolbox	/bin/bash	3 days ago	Exited (137) 3 days ago	
1a23da456b6f	docker.io/i386/ubuntu	/bin/bash	4 days ago	Exited (0) 6 hours ago	
df890193f651	localhost/opensuse/toolbox	/bin/bash	4 days ago	Created	

The first two records are the INFRA containers of each pod, based on the k8s.gcr.io/pause:3.2 image. Other containers in the output are stand-alone containers that do not belong to any pod.

## 5 SUSE Workloads

SUSE ALP Dolomite runs containerized workloads instead of traditional applications. Images of these containers are stored in image registries online. ALP Dolomite can run any containerized workload that is supported by the default container manager Podman. This article lists and describes workloads securely distributed and supported by SUSE. You can find the source files of the workloads at <https://build.opensuse.org/project/show/SUSE:ALP:Workloads> .

### 5.1 Common requirements

To run workloads on ALP Dolomite using Podman, you generally need to have:

- Deployed ALP Dolomite.
- Installed and enabled Podman.

### 5.2 Running the YaST workload using Podman

#### 5.2.1 Introduction

This article describes how to start the YaST workload on SUSE ALP Dolomite. The following YaST container images are available:

##### **yast-mgmt-ncurses**

The base YaST workload. It contains the text version of YaST (ncurses).

##### **yast-mgmt-qt**

This workload adds the Qt-based graphical user interface.

##### **yast-mgmt-web**

This workload exposes the standard graphical interface via a VNC server and uses a JavaScript VNC client to render the screen in a Web browser.



## 5.2.2 Starting YaST in text mode

To start the text version (ncurses) of YaST as a workload, follow these steps:

1. Identify the full URL address in a registry of container images, for example:

```
> podman search yast-mgmt-ncurses
[...]  
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/  
workloads/yast-mgmt-ncurses
```

2. To start the container, run the following command:

```
# podman container runlabel run \  
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/  
workloads/yast-mgmt-ncurses:latest
```

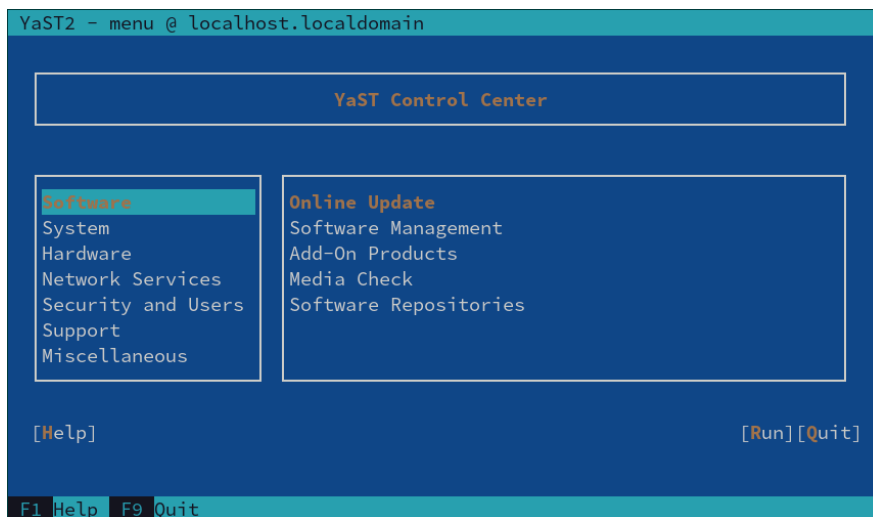


FIGURE 5.1: YAST RUNNING IN TEXT MODE ON ALP DOLOMITE

## 5.2.3 Starting graphical YaST

To start the graphical Qt version of YaST as a workload, follow these steps:

1. To view the graphical YaST on your local X server, you need to use SSH X forwarding. It requires the `xauth` package installed, applied by the host reboot:

```
# transactional-update pkg install xauth && reboot
```

2. Connect to the ALP Dolomite host using `ssh` with the X forwarding enabled:

```
> ssh -X ALP_HOST
```

3. Identify the full URL address in a registry of container images, for example:

```
> podman search yast-mgmt-qt
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/
workloads/yast-mgmt-qt
[...]
```

4. To start the container, run the following command:

```
# podman container runlabel run \
  registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/
workloads/yast-mgmt-qt:latest
```

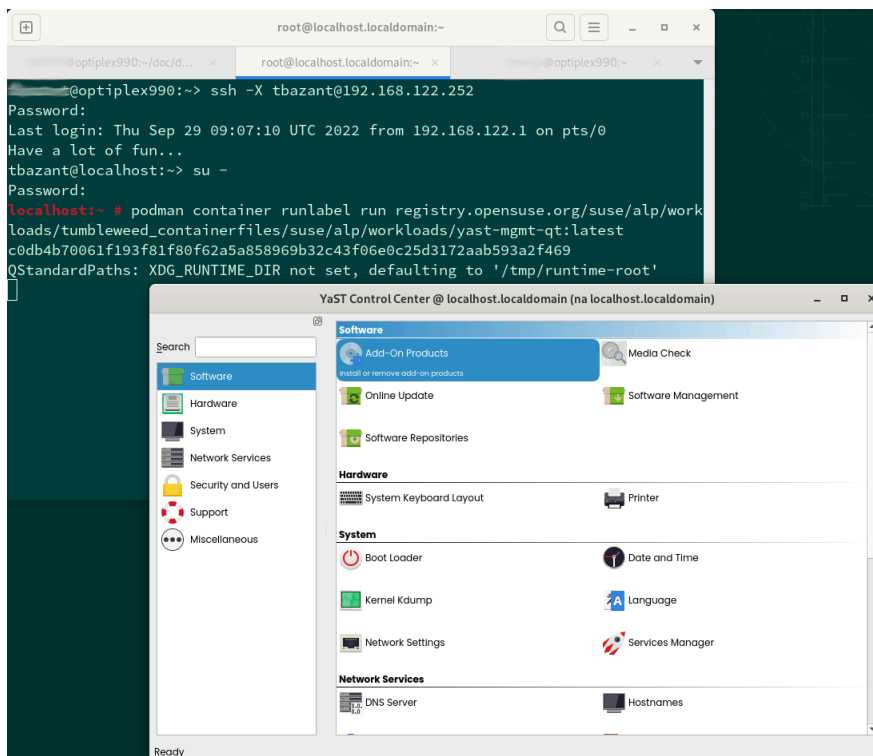


FIGURE 5.2: RUNNING GRAPHICAL YAST ON TOP OF ALP DOLOMITE

## 5.3 Running the KVM virtualization workload using Podman

This article describes how to run a KVM VM Host Server on SUSE ALP Dolomite (ALP Dolomite). This workload adds virtualization capability to ALP Dolomite so that you can use it as a VM Host Server. It uses the KVM hypervisor supported by the `libvirt` toolkit.

### Important

When running ALP Dolomite in a virtualized environment, you need to enable the nested KVM virtualization on the bare-metal host operating system and use `kernel-default` kernel instead of the default `kernel-default-base` in ALP Dolomite.

### 5.3.1 Deploying the KVM Server workload

ALP Dolomite can serve as a virtualization host for running virtual machines. The following procedure describes steps to prepare the ALP Dolomite host to run the containerized KVM Server workload.

1. Identify the KVM Server container image.

```
# podman search kvm-server
[...]  
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/  
workloads/kvm-server
```

2. Install the KVM Server workload. This pulls the container image from the registry and installs all `systemd` services, helper scripts, and configuration files on the host.

```
# podman container runlabel install registry.opensuse.org/suse/alp/workloads/  
tumbleweed_containerfiles/suse/alp/workloads/kvm-server:latest
```

3. Deploy the KVM Server workload.

```
# kvm-server-manage enable
```

Refer to [Section 5.3.2, “Usage of the `kvm-server-manage` script”](#) for more info on managing the deployment of the KVM Server workload.

4. Verify successful deployment of the KVM Server workload.

```
# kvm-server-manage verify
✓ All required services are currently active
```



### Important

The KVM Server container does not include client tooling by default. Refer to [Section 5.3.3, “Using the KVM Client workload”](#) for instructions on containerized client tooling.

Alternatively, client tooling can be installed on a separate host for remote VM management.

## 5.3.2 Usage of the `kvm-server-manage` script

The `kvm-server-manage` script is used to manage the KVM Server container and its dependent services on ALP Dolomite. This article describes each subcommand of the script. `kvm-server-manage` is installed together with the KVM Server container.

### `kvm-server-manage enable`

Disables the monolithic `libvirtd` daemon (if present).

Starts the KVM Server container if it is not currently running.

Enables and (re)starts the modular `libvirt` daemons that are run inside the KVM Server container.

### `kvm-server-manage restart`

Performs the same actions as `kvm-server-manage enable` and also restarts the KVM Server container.

### `kvm-server-manage disable`

Disables and stops the KVM Server container and all `libvirt` daemons.

### **kvm-server-manage stop**

Stops the running KVM Server container and all `libvirt` daemons.

The services are stopped but are still enabled to run on the next host boot. Use **kvm-server-manage disable** to disable the KVM Server container on the next host boot.

### **kvm-server-manage verify**

Verifies that all required services are active. Any inactive services are shown and can be addressed by the administrator.

## 5.3.3 Using the KVM Client workload

The KVM Client workload can be installed on ALP Dolomite to provide client tooling to interact with the KVM Server. The KVM Client workload provides wrappers for many virtualization tools to be run inside of a container.

1. Identify the KVM Client workload image:

```
# podman search kvm-client
[...]  
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/  
workloads/kvm-client
```

2. Install the KVM Client workload. This pulls the container image from the registry and installs the included wrappers on the host.

```
# podman container runlabel install registry.opensuse.org/suse/alp/workloads/  
tumbleweed_containerfiles/suse/alp/workloads/kvm-client:latest
```

3. Optionally, print the list of installed wrappers.

```
# find -L /usr -xtype l -samefile $(command -v kvm-client-wrapper)
```

4. Run the desired tool from the list of installed wrappers. The wrapper transparently handles passing through the given executable, flags, and arguments into the container.

## 5.4 Running the Cockpit Web server using Podman

Cockpit is a tool to administer one or more hosts from one place via a Web user interface. Its default functionality is extended by plug-ins that you can install additionally. You do not need the Cockpit Web user interface installed on every ALP Dolomite host. One instance of the Web interface can connect to multiple hosts if they have the `alp_cockpit` pattern installed.

This article describes how to run a containerized Cockpit Web server on SUSE ALP Dolomite using Podman. This workload adds the Cockpit Web server to ALP Dolomite so that you can administer the system and container via a user-friendly interface in your Web browser.



### Tip

Find more comprehensive information about managing Cockpit in <https://documentation.suse.com/alp/dolomite/html/cockpit-alp-dolomite/index.html>.



### Important

You need to have the `alp_cockpit` software pattern installed before deploying the Cockpit container.



### Note

An alternative way of installing and enabling the Cockpit Web server is described in [https://en.opensuse.org/openSUSE:ALP/Workgroups/SysMngmnt/Cockpit#Installing\\_the\\_Web\\_Server\\_Via\\_Packages](https://en.opensuse.org/openSUSE:ALP/Workgroups/SysMngmnt/Cockpit#Installing_the_Web_Server_Via_Packages).

ALP Dolomite has the base part of the Cockpit component installed by default. It is included in the `alp_cockpit` pattern. To install and run Cockpit's Web interface, follow these steps:

1. Identify the Cockpit Web server workload image:

```
# podman search cockpit-ws  
[...]
```

```
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/
workloads/cockpit-ws
```

2. Pull the image from the registry:

```
# podman container runlabel install \
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/
workloads/cockpit-ws:latest
```

3. Run the Cockpit's containerized Web server:

```
# podman container runlabel --name cockpit-ws run \
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/
workloads/cockpit-ws:latest
```

4. To run the Cockpit's Web server on each ALP Dolomite boot, enable its service:

```
# systemctl enable cockpit.service
```

5. To view the Cockpit Web user interface, point your Web browser to the following address and accept the self-signed certificate:

```
https://HOSTNAME_OR_IP_OF_ALP_HOST:9090
```

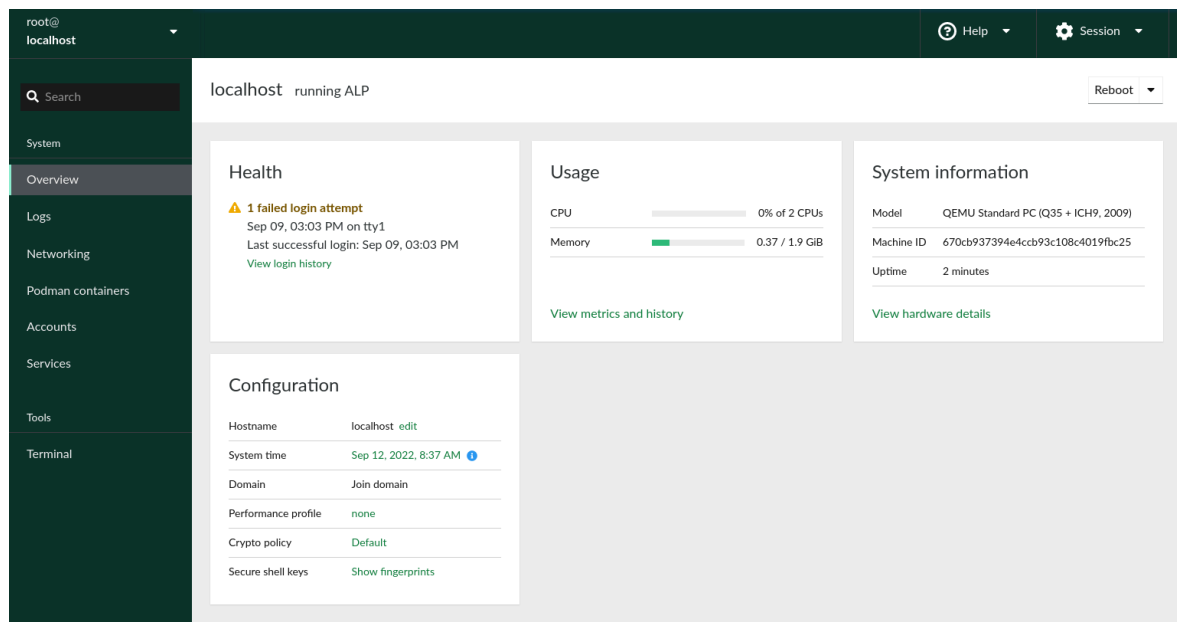


FIGURE 5.3: COCKPIT RUNNING ON ALP DOLOMITE

## 5.4.1 Software updates

To be able to perform transactional software updates from Cockpit, install the `cockpit-tukit` package:

```
# transactional-update pkg install cockpit-tukit
# reboot
```

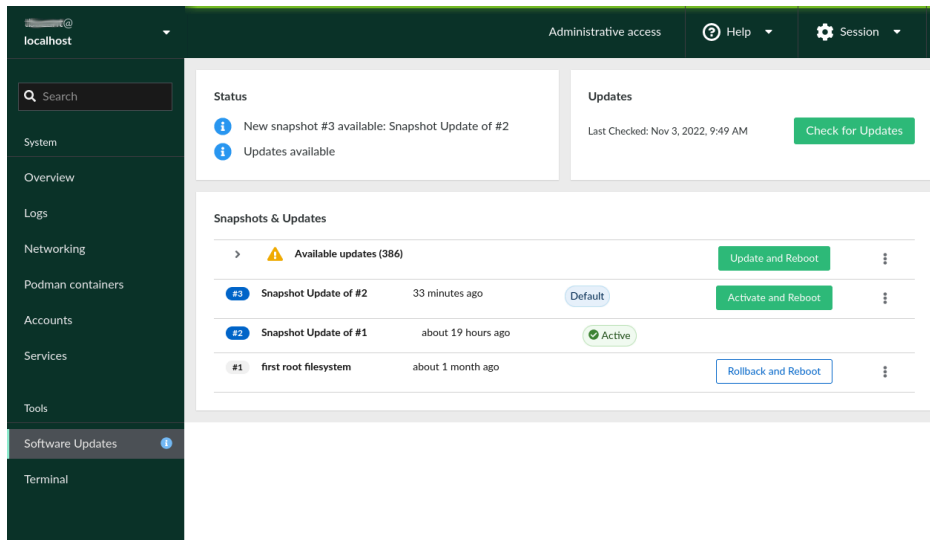


FIGURE 5.4: SOFTWARE UPDATES IN COCKPIT

## 5.5 Running the GNOME Display Manager workload using Podman

This article describes how to deploy and run the GNOME Display Manager (GDM) on SUSE ALP Dolomite. This workload runs GDM and basic GNOME environment. For more details, refer to [Section 5.5, “Running the GNOME Display Manager workload using Podman”](#).

### 5.5.1 Starting the GDM workload

1. On the ALP Dolomite host system, install `accountsservice` and `systemd-experimental` packages:

```
# transactional-update pkg install accountsservice systemd-experimental
```



```
# reboot
```

2. Verify that SELinux is configured in the *permissive* mode and enable the *permissive* mode if required:

```
# setenforce 0
```

3. Identify the GDM container:

```
> podman search gdm
[...]  
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/  
workloads/gdm
```

4. Download and recreate the GDM container locally:

```
# podman container runlabel install \  
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/  
workloads/gdm:latest
```

5. Reload the affected systemd services:

```
# systemctl daemon-reload  
# systemctl reload dbus  
# systemctl restart accounts-daemon
```

6. Run the GDM container.

- a. For a standalone process in a container, run:

```
# systemctl start gdm.service
```

Alternatively, run the command manually:

```
# podman container runlabel --name gdm run \  
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/  
workloads/gdm:latest
```

- b. For systems with systemd running in a container, run:

```
# systemctl start gdm-systemd.service
```

Alternatively, run the command manually:

```
# podman container runlabel run-systemd --name gdm \  

```

```
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/
workloads/gdm:latest
```

7. The GDM starts. After you log in, a basic GNOME environment opens.

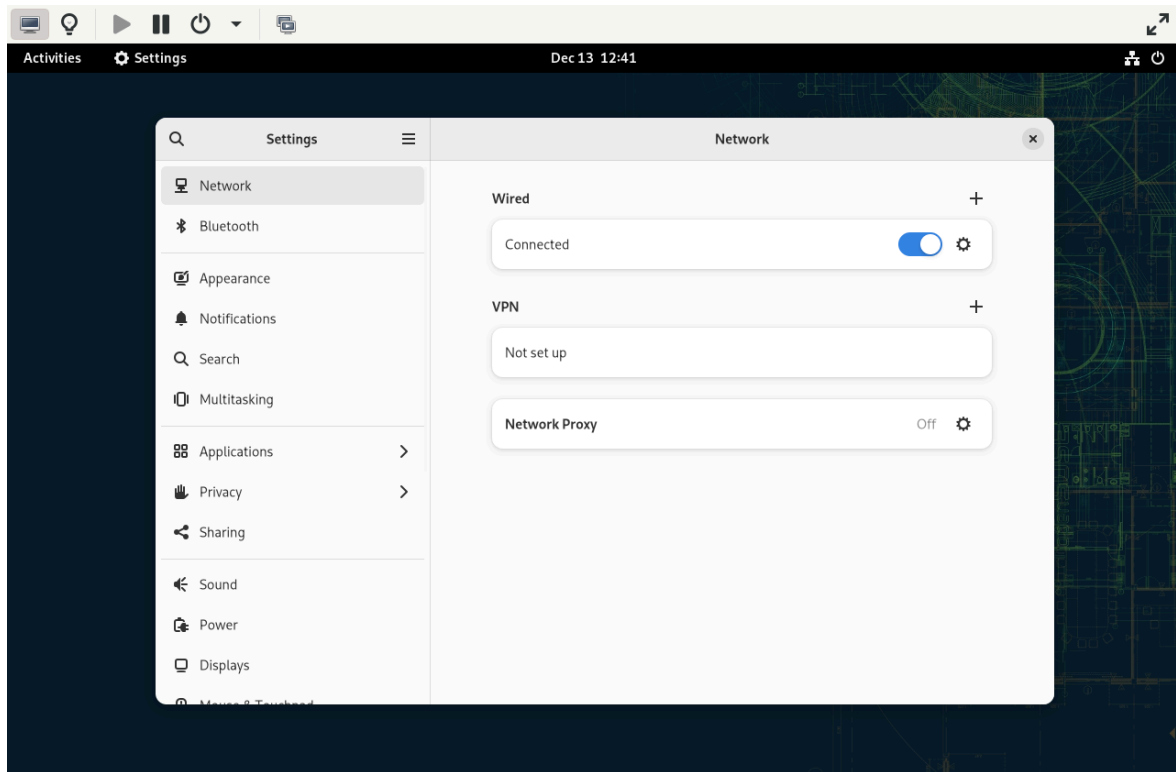


FIGURE 5.5: GNOME SETTINGS ON TOP OF ALP DOLOMITE



### Tip: Uninstalling deployed files

If you need to clean the environment from all deployed files, run the following command:

```
# podman container runlabel uninstall \
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/
workloads/gdm:latest
```

## 5.6 Running firewalld using Podman

This article describes how to run a containerized `firewalld` on SUSE ALP Dolomite using Podman. This workload adds firewall capability to ALP Dolomite to define the trust level of network connections or interfaces.

### Important

The `firewalld` container needs access to the host network and needs to run as a privileged container. The container image uses the system dbus instance. Therefore, you need to install `dbus` and `polkit` configuration files first.

### 5.6.1 Running the firewalld workload

1. Identify the `firewalld` workload image:

```
# podman search firewalld
[...]  
registry.opensuse.org/suse/alp/workloads/tumbleweed_images/suse/alp/workloads/  
firewalld
```

2. Verify that `firewalld` is not installed in the host system. Remove it, if necessary, and reboot the ALP Dolomite host:

```
# transactional-update pkg remove firewalld  
reboot
```

3. Initialize the environment:

```
# podman container runlabel install \  
registry.opensuse.org/suse/alp/workloads/tumbleweed_images/suse/alp/workloads/  
firewalld
```

The command prepares the system and creates the following files on the host system:

```
/etc/dbus-1/system.d/FirewallD.conf  
/etc/polkit-1/actions/org.fedoraproject.FirewallD1.policy ❶
```

```
/etc/systemd/system/firewalld.service ❷  
/etc/default/container-firewalld  
/usr/local/bin/firewall-cmd ❸
```

- ❶ The polkit policy file will only be installed if polkit itself is installed. It may be necessary to restart the dbus and polkit daemon afterwards.
- ❷ The systemd service and the corresponding configuration file /etc/default/container-firewalld allow to start and stop the container using systemd if Podman is used as a container manager.
- ❸ /usr/local/bin/firewall-cmd is a wrapper to call the firewall-cmd inside the container. Docker and Podman are supported.

#### 4. Run the container:

```
# podman container runlabel run \  
registry.opensuse.org/suse/alp/workloads/tumbleweed_images/suse/alp/workloads/  
firewalld
```

The command will run the container as a privileged container with the host network. Additionally, /etc/firewalld and the dbus socket are mounted into the container.



#### Tip

If your container manager is Podman, you can operate firewalld by using its systemd unit files, for example:

```
# systemctl start firewalld
```

#### 5. Optionally, you can remove the firewalld workload and clean the environment from all related files. Configuration files are left on the system.

```
# podman container runlabel uninstall \  
registry.opensuse.org/suse/alp/workloads/tumbleweed_images/suse/alp/workloads/  
firewalld
```

### 5.6.1.1 Managing the `firewalld` instance

After the `firewalld` container is started, you can manage its instance in two ways. You can manually call its client application via the `podman exec` command, for example:

```
podman exec firewalld firewall-cmd OPTIONS
```

Alternatively, you can use a shorter syntax by running the `firewall-cmd` wrapper script.

### 5.6.1.2 `firewalld` manual pages

To read the `firewalld` manual page, run the following command:

```
> podman run -i --rm \
  registry.opensuse.org/suse/alp/workloads/tumbleweed_images/suse/alp/workloads/firewalld
  \
  man firewalld
```

To read the `firewall-cmd` manual page, run the following command:

```
> podman run -i --rm \
  registry.opensuse.org/suse/alp/workloads/tumbleweed_images/suse/alp/workloads/firewalld
  \
  man firewall-cmd
```

## 5.7 Running the Grafana workload using Podman

This article describes how to run the Grafana visualization tool on SUSE ALP Dolomite. This workload adds a Web-based dashboard to the ALP Dolomite host that lets you query, monitor, visualize and better understand existing data residing on any client host.

### 5.7.1 Starting the Grafana workload

This section describes how to start the Grafana workload, set up a client so that we can test it with real data, and configure the Grafana Web application to visualize the client's data.

1. Identify the Grafana workload image:

```
# podman search grafana
```

```
[...]
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/
workloads/grafana
```

2. Pull the image from the registry and prepare the environment:

```
# podman container runlabel install \
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/
workloads/grafana:latest
```

3. Create the grafana container from the downloaded image:

```
# grafana-container-manage.sh create
```

4. Start the container with the Grafana server:

```
# grafana-container-manage.sh start
```

## 5.7.2 Setting up a Grafana client

To test Grafana, you need to set up a client that will provide real data to the Grafana server.

1. Log in to the client host and install the golang-github-prometheus-node\_exporter and golang-github-prometheus-prometheus packages:

```
# zypper in golang-github-prometheus-node_exporter golang-github-prometheus-
prometheus
```



### Note

If your Grafana server and client hosts are virtualized by a KVM containerized workload, use the `--network` option while creating the POD because the `--publish` option does not work in this scenario. To get the IP of the VM Host Server default network, run the following command on the VM Host Server:

```
> virsh net-dhcp-leases default
```

2. Restart the Prometheus services on the client host:

```
# systemctl restart prometheus-node_exporter.service
# systemctl restart prometheus
```

### 5.7.3 Configuring the Grafana Web application

To configure a data source for the Grafana Web dashboard, follow these steps:

1. Open the Grafana Web page that is running on port 3000 on the ALP Dolomite host where the Grafana workload is running, for example:

```
> firefox http://ALP_HOST_IP_ADDRESS:3000
```

2. Log in to Grafana. The default user name and password are both set to `admin`. After logging in, enter a new password.
3. Add the Prometheus data source provided by the client. In the left panel, hover your mouse over the gear icon and select *Data sources*.

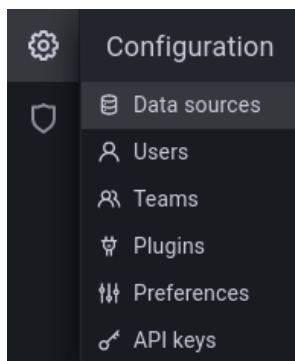


FIGURE 5.6: GRAFANA DATA SOURCES

4. Click *Add data source* and select *Prometheus*. Fill the *URL* field with the URL of the client where the Prometheus service runs on port 9090, for example:

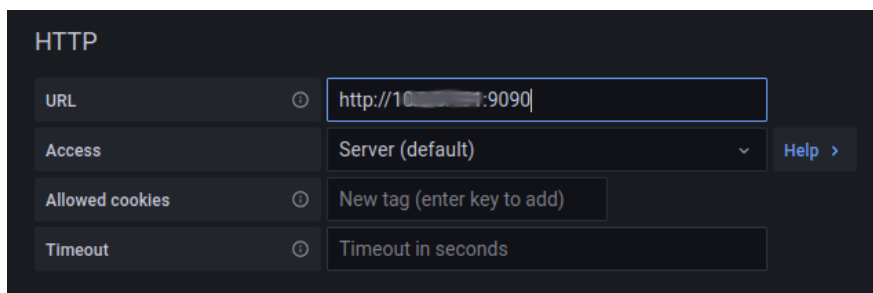


FIGURE 5.7: PROMETHEUS URL CONFIGURATION IN GRAFANA

Confirm with *Save & test*

5. Create a dashboard based on Prometheus data. Hover your mouse over the plus sign in the left panel and select *Import*.

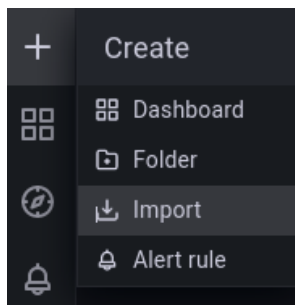


FIGURE 5.8: CREATING A GRAFANA DASHBOARD

6. Enter 405 as the dashboard ID and confirm with *Load*.
7. From the *Prometheus* drop-down list at the bottom, select the data source you have already created. Confirm with *Import*.
8. Grafana shows your newly created dashboard.



FIGURE 5.9: NEW GRAFANA DASHBOARD



## 5.7.4 Usage of the **grafana-container-manage.sh** script

The **grafana-container-manage.sh** script is used to manage the Grafana container on SUSE ALP Dolomite. This article lists each subcommand of the script and describes its purpose.

### **grafana-container-manage.sh create**

Pulls the Grafana image and creates the corresponding container.

### **grafana-container-manage.sh install**

Installs additional files that are required to manage the grafana container.

### **grafana-container-manage.sh start**

Starts the container called grafana.

### **grafana-container-manage.sh uninstall**

Uninstalls all files on the host that were required to manage the grafana container.

### **grafana-container-manage.sh stop**

Stops the grafana container.

### **grafana-container-manage.sh rm**

Deletes the grafana container.

### **grafana-container-manage.sh rmcache**

Removes the container image in cache.

### **grafana-container-manage.sh**

Runs the grafana container.

### **grafana-container-manage.sh bash**

Runs the **bash** shell inside the grafana container.

### **grafana-container-manage.sh logs**

Displays log messages of the grafana container.

## 5.8 Running the NeuVector workload using Podman

NeuVector is a powerful container security platform that includes end-to-end vulnerability scanning and complete runtime protection for containers, pods and hosts. This article describes how to run NeuVector on SUSE ALP Dolomite.

### Important

NeuVector requires SELinux to be set into the *permissive* mode by running the following command:

```
# setenforce 0
```

You can find more details about SELinux in [Section 2.9.2, "SELinux"](#).

### 5.8.1 Starting NeuVector

1. Identify the NeuVector workload image:

```
# podman search neuvector
[...]  
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/  
workloads/neuvector
```

2. Pull the image from the registry and install `systemd` services to handle NeuVector container start-up and shutdown:

```
# podman container runlabel install \  
  registry.opensuse.org/suse/alp/workloads/bci_containerfiles/suse/alp/workloads/  
  neuvector-demo:latest
```

3. Start the NeuVector service:

```
# systemctl start neuvector.service
```

4. Connect to NeuVector in the Web browser by entering the following URL:

```
https://HOST_RUNNING_NEUVECTOR_SERVICE:8443
```

You need to accept the warning about the self-signed SSL certificate and log in with the following default credentials: `admin` / `admin`.

## 5.8.2 Uninstalling NeuVector

To uninstall NeuVector, run the following command:

```
# podman container runlabel uninstall \
  registry.opensuse.org/suse/alp/workloads/bci_containerfiles/suse/alp/workloads/
  neuvector-demo:latest
```

## 5.9 Running the Ansible workload using Podman

Ansible is a suite of tools for managing and provisioning data centers via definition files. This article describes how to run Ansible on SUSE ALP Dolomite.

### Important

`python3-lxml` and `python3-rpm` packages are required for Ansible to interact with `libvirt` and gather package facts. The `kernel-default-base` package does not contain the required drivers for multiple NetworkManager or `nmcli` operations, such as creating bonded interfaces. Replace it with `kernel-default`:

```
# transactional-update pkg install python3-rpm python3-lxml kernel-default -kernel-
  default-base
# shutdown -r now
```

### 5.9.1 Installing Ansible commands

1. Identify the Ansible workload image:

```
# podman search ansible
```

```
[...]
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/
workloads/ansible
```

2. Pull the image from the registry and install Ansible commands.

- a. For root, the Ansible commands are placed in the /usr/local/bin directory. Run the following command to install Ansible commands for root:

```
# podman container runlabel install \
  registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/
workloads/ansible:latest
```



### Tip: Example Ansible playbooks

If you installed the Ansible commands as root, you can find example playbooks in the /usr/local/share/ansible-container/examples directory.

- b. For non-root, the Ansible commands are placed in the bin/ subdirectory of the current working directory. When installing them in your home directory, verify that the bin/ subdirectory exists. Run the following commands to install Ansible commands in your home directory:

```
> cd && podman container runlabel user-install \
  registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/
workloads/ansible:latest
```

After the successful installation of Ansible, the following commands are available:

- ansible
- ansible-community
- ansible-config
- ansible-connection
- ansible-console
- ansible-doc
- ansible-galaxy
- ansible-inventory

- `ansible-lint`
- `ansible-playbook`
- `ansible-pull`
- `ansible-test`
- `ansible-vault`

## 5.9.2 Uninstalling Ansible

To uninstall Ansible as root, run the following command:

```
# podman container runlabel uninstall \  
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/workloads/  
ansible:latest
```

To uninstall Ansible as non-root, run the following commands:

```
> cd && podman container runlabel user-uninstall \  
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/workloads/  
ansible:latest
```

## 5.9.3 Operation via SSH

Because Ansible is running inside a container, the default localhost environment is the container itself and not the system hosting the container instance. Therefore, any changes made to the localhost environment are made to the container and are lost when the container exits.

Instead, Ansible can be targeted via an SSH connection at the host that is running the container, namely host.containers.internal, using an Ansible inventory similar to the following example:

```
alhost_group:  
  hosts:  
    alphost:  
      ansible_host: host.containers.internal  
      ansible_python_interpreter: /usr/bin/python3
```

An equivalent `alphost` default inventory item has also been added to the container's `/etc/ansible/hosts` inventory, which can be used by the **ansible** command-line tool. For example, to run the `setup` module to collect and show system facts from the `alphost`, run a command similar to the following:

```
# ansible alphost -m setup
alphost | SUCCESS => {
  "ansible_facts": {
    [...]
  },
  "changed": false
}
```



### Tip

The inventory record may also contain other hosts to be managed.



### Important: Set up SSH keys

The container must be able to connect to the system being managed. The following conditions must be fulfilled:

- The system supports SSH access.
- SSH keys are created using `ssh-keygen`.
- The public SSH key is included in the `.ssh/authorized_keys` file for the target user.

The preferred method is to use a non-`root` account that has passwordless `sudo` privileges. Any operations in Ansible playbooks that require system privileges need to use the `become: true` setting.

Note that the SSH access can be validated with the `ssh localhost` command.

## 5.9.4 Examples of Ansible playbooks

### 5.9.4.1 Introduction

On the ALP Dolomite system where the Ansible workload container has been installed using the `install` runlabel (refer to [Section 5.9.1, “Installing Ansible commands”](#) for more details), the examples are available in `/usr/local/share/ansible-container/examples/ansible`.

### 5.9.4.2 Simple Ansible test

The `playbook.yml` playbook tests several common Ansible operations, such as gathering facts and testing for installed packages. To invoke the play, change to the directory `/usr/local/share/ansible-container/examples/ansible` and enter the following command:

```
> ansible-playbook playbook.yml
...
PLAY RECAP *****
alphost      : ok=8 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
```

### 5.9.4.3 Drive `nmcli` to change system networking

The `network.yml` playbook uses the `community.general.nmcli` plugin to test common network operations, such as assigning static IP addresses to NICs or creating bonded interfaces.

The NICs, IP addresses, bond names, and bonded NICs are defined in the `vars` section of the `network.yml` file. Update it to reflect the current environment. To invoke the play, change to the directory `/usr/local/share/ansible-container/examples/ansible` and enter the following command:

```
> ansible-playbook network.yml
...
ASK [Ping test Bond IPs]
*****
ok: [alphost] => (item={'name': 'bondcon0', 'ifname': 'bond0', 'ip4':
'192.168.181.10/24', 'gw4': '192.168.181.2', 'mode': 'active-backup'})
ok: [alphost] => (item={'name': 'bondcon1', 'ifname': 'bond1', 'ip4':
'192.168.181.11/24', 'gw4': '192.168.181.2', 'mode': 'balance-alb'})

TASK [Ping test static nics IPs]
*****
```

```
ok: [alphost] => (item={'name': 'enp2s0', 'ifname': 'enp2s0', 'ip4': '192.168.181.3/24',
'gw4': '192.168.181.2', 'dns4': ['8.8.8.8']})
ok: [alphost] => (item={'name': 'enp3s0', 'ifname': 'enp3s0', 'ip4': '192.168.181.4/24',
'gw4': '192.168.181.2', 'dns4': ['8.8.8.8']})
```

#### PLAY RECAP

```
*****
alphost      : ok=9    changed=3    unreachable=0    failed=0    skipped=0
               rescued=0    ignored=0
```

### 5.9.4.4 Set up ALP Dolomite as a libvirt host

The `setup_libvirt_host.yml` playbook installs the `kvm-container` workload and enables the `libvirtd` `systemd` service. To invoke the play, change to the directory `/usr/local/share/ansible-container/examples/ansible` and enter the following command:

```
> ansible-playbook setup_libvirt_host.yml
...
PLAY RECAP *****
alphost      : ok=9 changed=2 unreachable=0 failed=0 skipped=4 rescued=0 ignored=0

> sudo /usr/local/bin/virsh list --all
using /etc/kvm-container.conf as configuration file
+ podman exec -ti libvirtd virsh list --all
Authorization not available.
Check if polkit service is running or see debug message for more information.
```



#### Note

If the required kernel and supporting packages are not already installed, a reboot is required to complete the installation of missing packages. Follow the directions generated by the playbook. After the reboot has completed successfully, rerun the playbook to finish the setup of the `libvirtd` service.

### 5.9.4.5 Create an openSUSE Tumbleweed appliance VM

The playbook creates and starts a `libvirt` managed VM called `tumbleweed` that is based on the latest available openSUSE Tumbleweed appliance VM image.



It uses the `setup_libvirt_host.yml` playbook (see [Section 5.9.4.4, “Set up ALP Dolomite as a libvirt host”](#)) to ensure that the ALP Dolomite host is ready to manage VMs before creating the new one. It may fail prompting you to reboot before running the playbook again to finish setting up `libvirt` and creating the VM.

To invoke the play, change to the directory `/usr/local/share/ansible-container/examples/ansible` and enter the following command:

```
> ansible-playbook create_tumbleweed_vm.yml
...
TASK [Query list of libvirt VMs] *****
ok: [alphost]

TASK [Show that Tumbleweed appliance has been created] *****
ok: [alphost] => {
  "msg": "Running VMs: tumbleweed"
}

PLAY RECAP *****
```

## 5.10 Running the Kea DHCP server using Podman

Kea is an open-source DHCP server that supports both DHCPv4 and DHCPv6 protocols. It provides IPv6 prefix delegation, host reservations optionally stored in a database, PXE boot, high-availability setup and other features.

### 5.10.1 Deploying and running the Kea workload

1. Identify the Kea DHCP server container image:

```
# podman search kea
[...]
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/
workloads/kea
```

2. Pull the image from the registry:

```
# podman pull \
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/
workloads/kea:latest
```

3. Install all required parts of the Kea workload:

```
# podman container runlabel install \
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/
workloads/kea:latest
```

The previous command installs:

- Default configuration files in the `/etc/kea` directory
- The `keactrl` wrapper in the `/usr/local/bin` directory
- `systemd` service files for the `dhcp4` and `dhcp6` containers in the `/etc/systemd/system/` directory

4. Run the Kea DHCP server. You can run it either using `systemd` unit files, or manually.



### Tip

To run DHCP server with `firewalld` active, you need to add exception rules based on the version of DHCP you're using.

For DHCPv4:

```
> sudo firewall-cmd --add-service=dhcp
```

For DHCPv6:

```
> sudo firewall-cmd --add-service=dhcpv6
```

a. To run Kea as a `systemd` service, use one of the following commands:

```
# systemctl start kea-dhcp4.service
```

Or, for DHCPv6:

```
# systemctl start kea-dhcp6.service
```

b. To run Kea manually, use one of the following commands:

```
# podman container runlabel run \
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/
workloads/kea:latest
```

Or, for DHCPv6:

```
# podman container runlabel run_dhcp6 \  
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/  
workloads/kea:latest
```

5. Optionally, you can uninstall the Kea workload. The following command removes all Kea-related files except for the configuration directory and its content:

```
# podman container runlabel uninstall \  
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/  
workloads/kea:latest
```



### Tip

The `purge` `runlabel` removes the Kea configuration directory `/etc/kea` but leaves the rest of Kea deployment in place:

```
# podman container runlabel purge \  
registry.opensuse.org/suse/alp/workloads/tumbleweed_containerfiles/suse/alp/  
workloads/kea:latest
```

## 5.10.2 Configuration files

The Kea configuration files—`kea-dhcp4.conf` and `kea-dhcp6.conf`—are located in the `/etc/kea` directory. They include the default configuration. You can find detailed information about configuring the DHCP server in the official documentation at <https://kea.readthedocs.io/>.



### Tip

If you modify configuration files, run `keactrl reload` to apply them to running servers.

### 5.10.3 The **keactrl** wrapper

The installed **keactrl** wrapper uses the original **keactrl** tool to send commands to deployed containers. It uses the same options as the original tool with one exception: the `-s` option is adjusted to send commands to the DHCPv4 (`-s dhcp4`) or DHCPv6 (`-s dhcp6`). If `-s` is not specified, **keactrl** sends commands to both servers if they are started.

## 5.11 For more information

- The general concept of Podman is described in *Chapter 4, Containers and Podman*.
- Podman usage is explained in *Section 4.6, "Podman usage"*.
- YaST is generally described in <https://documentation.suse.com/sles/html/SLES-all/cha-yast-gui.html>.
- **libvirt** virtualization is described in <https://documentation.suse.com/sles/html/SLES-all/part-virt-libvirt.html>.
- Installing software packages and patterns is detailed in <https://documentation.suse.com/sles/html/SLES-all/cha-sw-cl.html>.
- Details about the usage of the **kvm-server-manage** script are described in *Section 5.3.2, "Usage of the **kvm-server-manage** script"*.
- Enabling KVM nested virtualization is described in <https://documentation.suse.com/sles/html/SLES-all/cha-vt-installation.html#sec-vt-installation-nested-vms>.
- Managing **systemd** services is described in <https://documentation.suse.com/smart/linux/html/reference-systemctl-enable-disable-services/reference-systemctl-enable-disable-services.html>.
- Find more details about Grafana at its home page <https://grafana.com/grafana/>.
- The NeuVector documentation is at <http://open-docs.neuvector.com/>.
- The Ansible documentation is at [https://docs.ansible.com/?extIdCarryOver=true&sc\\_cid=701f2000001OH7YAAW](https://docs.ansible.com/?extIdCarryOver=true&sc_cid=701f2000001OH7YAAW).
- On ALP Dolomite with the Ansible workload using the `install` runlabel, the examples are available in the `/usr/local/share/ansible-container/examples/ansible` directory.

## 6 Creating customized VMs using **virt-scenario**

**virt-scenario** is a tool that helps you create virtual machines (VM) suitable for a specific scenario. It provides predefined *profiles* that include optimal settings for each scenario. You can override settings that are common to all profiles.

### Important

Although **virt-scenario** generally provides the best possible configuration for a specific scenario, this cannot be guaranteed because each environment may have specific requirements.

### 6.1 How does **virt-scenario** work?

Interactive **virt-scenario** script creates a `libvirt` XML configuration file for a VM based on the following:

- selected basic profile
- overridden values for common parameters
- parameters that you specify interactively

After the configuration is validated, **virt-scenario** adjusts the VM Host Server system and creates the image file for the VM Guest. You can then operate the VM using standard `libvirt` commands.

### 6.2 Benefits of using **virt-scenario**

- Creating virtual machines is fast and simple. **virt-scenario** leaves all the virtualization complexity aside. You can focus on basic features only.
- Fine-tuned profiles already offer optimal settings for specified scenarios. You do not have to search and copy them to each virtual machine of the same type, just use the same profile.
- The override mechanism lets you specify other values than the profile's default for selected options. This way, you can customize virtual machines to your needs.

## 6.3 Creating VMs

The **virt-scenario** command presents an interactive shell where you optionally specify configuration parameters and finally select the target scenario. Its welcome screen shows categories with available commands.

```
virt-scenario Interactive Terminal!

Setting the virt-scenario Configuration: conf
Guest/Host/Both mode could be selected using: mode
Force overwrite previous setting: overwrite

Prepare a Libvirt XML guest config and the host to run a customized guest:
computation | desktop | securevm

Possible User Settings For VM are:
name | vcpu | memory | machine | bootdev | diskpath | cdrom

Hypervisors parameters:
hconf | hv_select | hvlist

You can overwrite some recommended VM settings editing: /etc/virt-scenario/virtscenario.yaml

Please read the manpage and the README.md file:
https://github.com/aginies/virt-scenario/blob/main/README.md

-----
Main Configuration: /etc/virt-scenario/virtscenario.yaml
Hypervisor Configuration: /etc/virt-scenario/virthosts.yaml
> 
```

FIGURE 6.1: **virt-scenario** WELCOME SCREEN



### Tip

Each command has a built-in description of its usage. Enter **help COMMAND\_NAME** to view it on the screen.




### Warning: Unsafe force\_sev option

The **force\_sev** option forces the extraction of the Platform Diffie-Hellman key (PDH) on the current AMD SEV system. The PDH file is used to negotiate a master secret between the SEV firmware and the external entities. This file must be stored in a secure place, and this option is only provided for *testing* purposes.

After you finish optional configuration steps, enter the name of one of the scenarios—**computation**, **desktop** or **securevm**. **virt-scenario** then compiles all the configuration, prepares the VM Host Server, and saves the **libvirt** XML of the new VM.

### 6.3.1 Requirements

- Running ALP Dolomite with the KVM workload deployed. Refer to [Section 5.3, “Running the KVM virtualization workload using Podman”](#) for detailed steps.
- Secure virtual machines are supported only on an AMD processor that supports AMD SEV or SEV-ES technology. For more information about SUSE and AMD SEV, refer to <https://documentation.suse.com/sles/single-html/SLES-amd-sev/> .

### 6.3.2 Overriding default scenario settings

On each run, **virt-scenario** reads settings for a new VM from the `/etc/virt-scenario/virtscenario.yaml` file. Change settings in this file to affect *all* VMs created by **virt-scenario**.



#### Tip

Always backup `/etc/virt-scenario/virtscenario.yaml` before modifying it as incorrect parameters may lead to an invalid VM configuration.

To override settings for specific VMs only, copy `/etc/virt-scenario/virtscenario.yaml` to a different location and modify the settings that you need to override. On next **virt-scenario** run, specify the path to the new configuration file with the **conf** command, for example:

```
conf /home/tux/virt-scenarios/my-overriden-scenario.yaml
```



#### Important

In the overriding `virtscenario.yaml` file, you need to include *all* available settings from the original `virtscenario.yaml` file, not only the modified settings.



#### Warning

Never change section names in the `/etc/virt-scenario/virtscenario.yaml` file or its overriding copies.

The following is an example of `/etc/virt-scenario/virtscenario.yaml`:

```
config:
  - path: /etc/virt-scenario
  - vm-config-store: ~/.local/virtscenario/
emulator:
  - emulator: /usr/bin/qemu-system-x86_64
input:
  - keyboard: virtio
  - mouse: virtio
architecture:
  - arch: x86_64
STORAGE_DATA:
# some options are only available with qcow2 format and
# will be ignored in case of any other image format
  - disk_type: file
  - disk_cache: none
  - disk_target: vda
  - disk_bus: virtio
  - path: /var/libvirt/images
  - format: qcow2
# host side: qemu-img creation options (-o), qemu-img --help
  - unit: G
  - capacity: 20
  - cluster_size: 1024k
  - lazy_refcounts: on
preallocation: full
  - preallocation: off
  - compression_type: zlib
  - encryption: off
host_filesystem:
  - fmode: 644
  - dmode: 755
  - source_dir: /tmp
  - target_dir: /tmp/host
```

### 6.3.3 Specifying **virt-scenario** mode

By default, **virt-scenario** creates the `libvirt` XML configuration of the new guest and adjusts the VM Host Server. You can instruct **virt-scenario** to perform only part of the task. After entering the **mode** command, you can enter one of the following:

#### **guest**

Creates only the `libvirt` XML configuration of the guest.



**host**

Prepares the VM Host Server system only.

**both**

Creates the guest configuration and prepares the host. This is the default mode.

### 6.3.4 Interactive commands

You can use the following commands when configuring a new VM using the **virt-scenario** interactive shell.



#### Tip

Each command has a built-in description of its usage. Enter **help COMMAND\_NAME** to view it on the screen.

#### HYPERVISOR CONFIGURATION

**hvconf**

Loads hypervisor configuration.

**hvselect**

Sets the hypervisor for which VMs are configured.

**hvlist**

Lists available hypervisors.

**overwrite**

Forces overwriting previous configuration.

#### GUEST CONFIGURATION

**name**

Defines the name of the VM.

**vcpu**

Specifies the number of virtual CPUs.

**memory**

Specifies the memory size (in GiB).

#### machine

Selects the machine type.

#### bootdev

Selects the boot device.

#### diskpath

Specifies the directory where to store the VM disk image.

#### conf

Specifies the path to the custom virtscenario.yaml file.

#### cdrom

Specifies the path to the CD/DVD installation media.

### GENERATE VM CONFIGURATION

#### computation

Creates a libvirt XML configuration and VM Host Server adjustments for the computation scenario.

#### desktop

Creates a libvirt XML configuration and VM Host Server adjustments for the desktop scenario.

#### securevm

Creates a libvirt XML configuration and VM Host Server adjustments for the secure VM scenario.

## 6.4 Predefined scenarios

When creating a VM, you can specify one of the following scenarios:

#### securevm

Selecting this scenario results in an encrypted VM image with a high level of isolation and data security.

#### computation

This scenario puts emphasis on the high performance of the resulting VM.

#### desktop

The result of this scenario is a VM suitable for running desktop applications.

The following tables show default settings for each scenario:

TABLE 6.1: DEFAULT STORAGE SETTINGS

Setting	securevm	computation	desktop
preallocation	metadata	off	metadata
encryption	on	off	off
disk_cache	writethrough	unsafe	none
lazy_refcounts	on	on	off
format	qcow2	raw	qcow2
disk bus	virtio	virtio	virtio
capacity	20G	20G	20G
cluster_size	1024k	N/A	1024k

TABLE 6.2: DEFAULT HOST SETTINGS

Setting	securevm	computation	desktop
Transparent HugePages	on	on	on
KSM	disable	enable	enable
KSM merge across	disable	enable	enable
swappiness	0	0	35
IO Scheduler	bfq	mq-deadline	mq-deadline

TABLE 6.3: DEFAULT GUEST SETTINGS

Setting	securevm	computation	desktop
CPU migratable	off	off	on
machine	pc-q35-6.2	pc-q35-6.2	pc-q35-6.2

Setting	securevm	computation	desktop
watchdog	none	i6300esb poweroff	none
boot UEFI	auto	auto	auto
vTPM	tpm-crb 2.0	none	none
iothreads	disable	4	4
video	qxl	qxl	virtio
network	e1000	virtio	e1000
keyboard	ps2	virtio	virtio
mouse	disable	virtio	virtio
on_poweroff	destroy	restart	destroy
on_reboot	destroy	restart	restart
on_crash	destroy	restart	destroy
suspend_to_mem	off	off	on
suspend_to_disk	off	off	on
features	acpi apic pae	acpi apic pae	acpi apic pae
host fs fmode, dmode, source_dir, target_dir	N/A	N/A	644 755 /tmp/ / tmp/host

TABLE 6.4: **DEFAULT SEV SETTINGS**

Setting	securevm	computation	desktop
kvm SEV	mem_encrypt = on kvm_amd sev = 1 sev_es = 1	N/A	N/A

Setting	securevm	computation	desktop
sec cbitpos	auto	N/A	N/A
sec reducedPhysBits	auto	N/A	N/A
sec policy	auto	N/A	N/A

## 6.5 Managing VMs

After you created a VM using the **virt-scenario** interactive shell, use the **virt-scenario-launch** command to manage it. The command identifies VMs by their name as displayed by the **--list** option.

```
# virt-scenario-launch --list
Version: 2.1.2
Available VMs:
  ALP_OS
  desktop
  testing_vm
  SLE15_HPC
```

When the VM is identified, you can manage it with the following options.

**--help**

Prints short descriptions of available options.

**--start**

Starts the VM and prints security attestation information, for example:

```
# virt-scenario-launch --start ALP_OS
Connected to libvirtd socket; Version: 7001000
SEV(-ES) attestation passed!
Validation successfull for domain ALPOS
```

**--status**

Shows the status of the VM, for example:

```
# virt-scenario-launch --status ALP_OS
Version: 2.1.2
Connecting to libvirt qemu:///system ...
Connected to libvirtd socket; Version: 7001000
```

```
Domain SLE15SP5HPC state: Shutoff
```

### --off

Shuts a VM down.

```
# virt-scenario-launch --off ALP_OS
```

### --force

Forces a VM off.

```
# virt-scenario-launch --force ALP_OS
```

## 7 Remote attestation using Keylime

With the growing demand on securing devices against unauthorized changes, the use of the security mechanism called *remote attestation (RA)* has been experiencing significant growth. Using RA, a host (client) can authenticate its boot chain status and running software on a remote host (verifier). RA is often combined with public-key encryption (using TPM2), thus the sent information can only be read by the services that requested the attestation, and the validity of the data can be verified. Remote attestation on ALP Dolomite is implemented by *Keylime*.

### 7.1 Terminology

Remote attestation technology uses the following terms:

#### Attestation key (AK)

A data signing key that proves that the data comes from a real TPM and has not been tampered with.

#### Core root of trust for measurement

Calculates its own hash and the hash of the next step in the boot process, initiating the chain of measurements.

#### Endorsement key (EK)

An encryption key that is permanently embedded in the TPM when it is manufactured. The public part of the key and the certification stored in the TPM are used to recognize a genuine TPM.

#### Integrity management architecture (IMA)

A kernel integrity subsystem that provides a means of detecting malicious changes to files.

#### Measured boot

A method with which each component in the booting sequence calculates a hash of the next one before delegating the execution of the next component. The hash extends one or several PCRs of the TPM. An event is created with the information about where the measurement took place and what was measured. Such events are collected in an event log, and, along with the extended PCR values, the events can be compared with the expected values representing a healthy system.

### Platform Configuration Register (PCR)

A memory location in TPM that, for example, stores hashes of booting layers. PCR can be updated only by using the non-reversible operation: extend. A signed list of current PCR values can be obtained by the quote command on TPM, and this quote can be verified by a third party during the attestation process.

### Secure boot

Each step of the booting process checks a cryptographic signature on the executable of the next step before launching it.

### Trusted Platform Module (TPM)

A self-contained security cryptographic processor present in the system as hardware or implemented in the firmware that serves as a root of trust. TPM provides a PCR for storing the hashes of booting layers. A typical TPM provides several functions, like a random number generator, counters or a local clock. It also stores 24 PCRs grouped by banks per each supported cryptographic hash function (SHA1, SHA256, SHA384 or SHA512).



#### Note

By default, TPM usage is disabled. Therefore, the measured boot does not take place. To enable the remote attestation, enable TPM in the EFI/BIOS menu.

### Secure payload

A mechanism to deliver encrypted data to healthy agents. Payloads are used to provide keys, passwords, certificates, configurations or scripts that are further used by the agent.

## 7.2 What is Keylime?

Keylime is a remote attestation solution that enables you to monitor the health of remote nodes using a TPM as a root of trust for measurement. With Keylime, you can perform multiple tasks, for example:

- Validate of the PCRs extended during the measured boot.
- Create analysis and make assertions of the event log.
- Make assertion of the value of any PCR in the remote system.



- Monitor the validity of open or executed files.
- Deliver encrypted data to verified nodes via *secure payloads*.
- Execute custom scripts that are triggered when a machine fails the attested measurements.

## 7.3 Architecture

Keylime consists of an agent, a verifier, a registrar and a command-line tool (tenant). Agents are on those systems that need to be attested. The verifier and registrar are on remote systems that perform the registration and attestation of agents. Keep in mind that only the agent role is available on ALP Dolomite. For details about each component, refer to the following sections.

### 7.3.1 Keylime agent

The agent is a service that runs on the system that needs to be attested. The agent sends the event log, IMA hashes, and information about the measured boot to the verifier, using the local TPM as a certifier of the data validity.

When a new agent is started, it needs to register itself in the registrar first. To do so, the agent needs a TLS certificate to establish the connection. The TLS certificate is generated by the registrar, but it needs to be installed manually to the agent. After the registration, the agent sends its attestation key and the public part of the endorsement key to the registrar. The registrar responds to the agent with a challenge in a process called credential activation, which validates the TPM of the agent. Once the agent has been registered, it is ready to be enrolled for attestation.

### 7.3.2 Keylime registrar

The registrar is used to register agents that should be attested. The registrar collects the agent's attestation key, the public part of the endorsement key and the endorsement key certification, and verifies that the agent attestation key belongs to the endorsement key.

### 7.3.3 Keylime verifier

The verifier performs the actual attestation of agents and continuously pulls the required attestation data from agents (among others, the PCR values, IMA logs, and UEFI event logs).

## 7.4 Setting up the verifier, registrar and tenant



### Note

The container described in this article delivers control plane services *verifier* and *registrar* and a *tenant* command-line tool (CLI) that are part of the Keylime project.

Before you start installing and registering agents, prepare the verifier and the registrar on remote hosts, as described in the following procedure.

1. Identify the Keylime workload image.

```
# podman search keylime
[...]  
registry.opensuse.org/devel/microos/containers/containerfile/opensuse/keylime-  
control-plane
```

2. Pull the image from the registry.

```
# podman pull \\\n  registry.opensuse.org/devel/microos/containers/containerfile/opensuse/keylime-  
control-plane:latest
```

3. Create the keylime-control-plane volume to persist the database and certificates required during the attestation process.

```
# podman container runlabel install \\\n  registry.opensuse.org/devel/microos/containers/containerfile/opensuse/keylime-  
control-plane:latest
```

4. Start the container and related services.

```
# podman container runlabel run \\\n  registry.opensuse.org/devel/microos/containers/containerfile/opensuse/keylime-  
control-plane:latest
```

The keylime-control-plane container is created. It includes configured and running registrar and verifier services. Internally, the container exposes ports 8881, 8890 and 8891 to the host using the default values. Validate the firewall configuration to allow access to the ports and to allow communication between containers, because the tenant CLI requires it.



## Tip

If you need to stop Keylime services, run the following command:

```
# podman kill keylime-control-plane-container
```

### 7.4.1 Monitoring Keylime services

To get the status of running containers on the host, run the following command:

```
# podman ps
```

To view the logs of Keylime services, run the following command:

```
# podman logs keylime-control-plane-container
```

### 7.4.2 Executing the tenant CLI

The tenant CLI tool is included in the container, and if the host firewall does not interfere with the ports exposed by Keylime services, you can execute it using the same image, for example:

```
# podman run --rm \
-v keylime-control-plane-volume:/var/lib/keylime/ \
keylime-control-plane:latest \
keylime_tenant -v 10.88.0.1 -r 10.88.0.1 --cert default -c reglist
```

### 7.4.3 Extracting the Keylime certificate

The first time that the Keylime container is executed, its services create a certificate required by several agents. You need to extract the certificate from the container and copy it to the agent's /var/lib/keylime/cv\_ca/ directory.

```
# podman cp \
keylime-control-plane-container:/var/lib/keylime/cv_ca/cacert.crt
.
# scp cacert.crt
AGENT_HOST:/var/lib/keylime/cv_ca/
```



## Tip

Find more details about installing the agent in [Section 7.5, “Installing the agent”](#).

## 7.5 Installing the agent

The Keylime agent is not present on ALP Dolomite by default, and you need to install it manually. To install the agent, proceed as follows:

1. Install the `rust-keylime` package as follows:

```
# transactional-update pkg in rust-keylime
```

Then reboot the system.

2. Adjust the default agent's configuration.

- a. Create a directory to store a new configuration file for your changes in `/etc/keylime/agent.conf.d/`. The default configuration is stored in `/usr/etc/keylime/agent.conf`, but we do not recommend editing this file because it may be overwritten in upcoming system updates.

```
# mkdir -p /etc/keylime/agent.conf.d
```

- b. Create a new file `/etc/keylime/agent.conf.d/agent.conf`:

```
# cat << EOF > /etc/keylime/agent.conf.d/agent.conf
[agent]

uuid = "d111ec46-34d8-41af-ad56-d560bc97b2e8" ❶
registrar_ip = "<REMOTE_IP>" ❷
revocation_notification_ip = "<REMOTE_IP>" ❸
EOF
```

- ❶ The unique identifier is generated each time the agent is run. However, you can define a specific value by this option.
- ❷ IP address of the registrar.
- ❸ IP address of the verifier.

- c. Change the owner of the `/etc/keylime/` directory to `keylime:tss`:

```
# chown -R keylime:tss /etc/keylime
```

- d. Change permissions on the `/etc/keylime/` directory:

```
# chmod -R 600 /etc/keylime
```

3. Copy the certificates generated by the CA to the agent node. On the agent node, do the following:

- a. Prepare a directory for the certificate:

```
# mkdir -p /var/lib/keylime/cv_ca
```

- b. Copy the certificate to the agent:

```
# scp CERT_SERVER_ADDRESS:/var/lib/keylime/cv_ca/cacert.crt /var/lib/keylime/cv_ca
```

- c. Change the owner of the certificate to `keylime:tss`:

```
# chown -R keylime:tss /var/lib/keylime/cv_ca
```

4. Start and enable the `keylime_agent.service`:

```
# systemctl enable --now keylime_agent.service
```

## 7.6 Registering the agent

You can register a new agent either by using the CLI tenant or by editing the configuration of the verifier. Using the tenant on the verifier host, run the following:

```
# keylime_tenant -v 127.0.0.1 \  
-t AGENT \①  
-u UUID \②  
--cert default \  
-c add  
[--include PATH_TO_ZIP_FILE] ③
```

- ① `AGENT` is an IP address of the agent to be registered.

- ② UUID is the agent's unique identifier.
- ③ The file passed by the include option is used to deliver secret payload data to the agent. For details, refer to [Section 7.7, "Secure payloads"](#).

You can list registered agents by using the reglist command on the verifier host as follows:

```
# keylime_tenant -v 127.0.0.1 \  
--cert default \  
-c reglist
```

To remove a registered agent, specify the agent using the -t and -u options and the -c delete command as follows:

```
# keylime_tenant -v 127.0.0.1 \  
-t AGENT \  
-u UUID \  
-c delete
```

## 7.7 Secure payloads

### 7.7.1 What is a secure payload?

A Keylime secure payload enables you to deliver encrypted data to healthy agents. Payloads are used to provide keys, passwords, certificates, configurations or scripts that are used by the Keylime agent at a later stage.

### 7.7.2 How does a secure payload work?

A secure payload is delivered to the agent in a zip file that must contain a shell script named autorun.sh. The script is executed only if the agent has been properly registered and verified. To deliver the zip file, use the --include option of the **keylime\_tenant** command.

For example, the following autorun.sh script creates a directory structure and copies SSH keys there. The related zip archive must include these SSH keys.

```
> cat autorun.sh  
#!/bin/bash
```

```
mkdir -p /root/.ssh/  
cp id_rsa* /root/.ssh/  
chmod 600 /root/.ssh/id_rsa*  
cp /root/.ssh/id_rsa.pub /root/.ssh/authorized_keys
```

## 7.8 Enabling IMA tracking

When using IMA, the kernel calculates a hash of accessed files. The hash is then used to extend the PCR 10 in the TPM and also log a list of accessed files. The verifier can request a signed quote to the agent for PCR 10 to get the logs of all accessed files including the file hashes. Verifiers then compare the accessed files with a local allowlist of approved files. If any of the hashes are not recognized, the system is considered unsafe, and a revocation event is triggered.

Before Keylime can collect information, IMA/EVM needs to be enabled. To enable the process, boot a kernel of the agent with the `ima_appraise=log` and `ima_policy=tcb` parameters:

1. Update the `GRUB_CMDLINE_LINUX_DEFAULT` option with the parameters in `/etc/default/grub`:

```
GRUB_CMDLINE_LINUX_DEFAULT="ima_appraise=log ima_policy=tcb"
```

2. Regenerate `grub.cfg` by running:

```
# transactional-update grub.cfg
```





3. Reboot your system.

The procedure above uses the default kernel IMA policy. To avoid monitoring too many files and therefore creating long logs, create a new custom policy. Find more details in the [Keylime documentation \(https://keylime-docs.readthedocs.io/en/latest/user\\_guide/runtime\\_ima.html\)](https://keylime-docs.readthedocs.io/en/latest/user_guide/runtime_ima.html).

To indicate the expected hashes, use the `--allowlist` option of the `keylime_tenant` command when registering the agent. To view the excluded or ignored files, use the `--exclude` option of the `keylime_tenant` command:

```
# keylime_tenant --allowlist  
-v 127.0.0.1 \  
-u UUID
```

## 7.9 For more information

- Keylime home page is at <https://keylime.dev> .
- Latest Keylime documentation is at <https://keylime.readthedocs.io/en/latest/> .
- For a high-level overview of IMA/EVM, refer to [https://en.opensuse.org/SDB:Ima\\_evm#Introduction](https://en.opensuse.org/SDB:Ima_evm#Introduction) .
- Find more details about creating a new kernel IMA policy in [https://keylime-docs.readthedocs.io/en/latest/user\\_guide/runtime\\_ima.html](https://keylime-docs.readthedocs.io/en/latest/user_guide/runtime_ima.html) .



## 8 Scheduling jobs using systemd timers

systemd timer units provide a mechanism for scheduling jobs on Linux. The execution time of these jobs can be based on the time and date or on events.

systemd timer units are identified by the .timer file name extension. Each timer file requires a corresponding service file it controls. In other words, a timer file activates and manages the corresponding service file. systemd timers support the following features:

- Jobs scheduled using a timer unit can depend on other systemd services. Timer units are treated as regular systemd services, so can be managed with **systemctl**.
- Timers can be real-time (being triggered on calendar events) or monotonic (being triggered at a specified time elapsed from a certain starting point).
- Time units are logged to the system journal, which makes it easier to monitor and troubleshoot them.
- Timers use the centralized systemd management services.
- If the system is off during the expected execution time, the timer is executed once the system is running again.

### 8.1 Creating a timer

The following example shows how to set up a timer that triggers the hello-world.sh shell script after boot time and repeats its execution every 24 hours relative to its activation time. It also runs Monday to Friday at 10 a.m.

#### 8.1.1 *Hello World* example

1. Create the file /etc/systemd/system/helloworld.service with the following content:

```
[Unit]
Description="Hello World script"

[Service]
```

```
ExecStart=/usr/local/bin/helloworld.sh
```

This is a `systemd` service file that tells `systemd` which application to run.

2. Create the file `/etc/systemd/system/helloworld.timer` with the following content:

```
[Unit]
Description="Run helloworld.service 5min after boot and every 24 hours relative to
activation time"

[Timer]
OnBootSec=5min
OnUnitActiveSec=24h
OnCalendar=Mon..Fri *-** 10:00:*
Unit=helloworld.service

[Install]
WantedBy=multi-user.target
```

This is the timer file that controls the activation of the respective service file.

3. Verify that the files you created above contain no errors:

```
> systemd-analyze verify /etc/systemd/system/helloworld.*
```

If the command returns no output, the files have passed the verification successfully.

4. Start the timer:

```
> sudo systemctl start helloworld.timer
```

Activates the timer for the current session only.

5. Enable the timer to make sure that it is activated on boot:

```
> sudo systemctl enable helloworld.timer
```

## 8.1.2 The example explained

### EXAMPLE 8.1: THE SERVICE FILE

```
[Unit]
Description="Hello World script" ❶

[Service]
```

```
ExecStart=/usr/local/bin/helloworld.sh ❷
```

- ❶ A brief description explaining the service file's purpose.
- ❷ The application to execute.

The `[Unit]` and `[Service]` sections are the minimum sections required for a service file to work. `systemd` service files normally contain an `[Install]` section that determines one or more targets for a service to load. This section is not required in service files for timers, since this information is provided with the timer file. For advanced configuration, refer to [Managing `systemd` targets with `systemctl`](https://documentation.suse.com/smart/systems-management/html/reference-managing-systemd-targets-systemctl/reference-systemctl-managing-targets.html) (<https://documentation.suse.com/smart/systems-management/html/reference-managing-systemd-targets-systemctl/reference-systemctl-managing-targets.html>) ↗.

#### EXAMPLE 8.2: THE TIMER FILE

```
[Unit]
Description="Run helloworld.service 5min after boot and every 24 hours relative to
activation time" ❶

[Timer]
OnBootSec=5min ❷
OnUnitActiveSec=24h ❸
OnCalendar=Mon..Fri *-*- * 10:00:* ❹
Unit=helloworld.service ❺

[Install]
WantedBy=multi-user.target ❻
```

- ❶ A brief description explaining the timer file's purpose.
- ❷ Specifies a timer that triggers the service five minutes after the system boot. See *Monotonic timers* for details.
- ❸ Specifies a timer that triggers the service 24 hours after the service has been activated (that is, the timer triggers the service once a day). See *Real-time timer* for details.
- ❹ Specifies a timer that triggers the service at fixed points in time (in this example, Monday to Friday at 10 a.m.). See *Real-time timer* for details.
- ❺ The service file to execute.
- ❻ The `systemd` target in which the timer gets activated. For more information on `systemd` targets, refer to [Managing `systemd` targets with `systemctl`](https://documentation.suse.com/smart/systems-management/html/reference-managing-systemd-targets-systemctl/reference-systemctl-managing-targets.html) (<https://documentation.suse.com/smart/systems-management/html/reference-managing-systemd-targets-systemctl/reference-systemctl-managing-targets.html>) ↗.

## 8.2 Managing timers

You can manage timers using the `systemctl` command.

### Starting and stopping timers

```
> sudo systemctl start TIMER.timer  
> sudo systemctl restart TIMER.timer  
> sudo systemctl stop TIMER.timer
```

### Enabling and disabling timers

```
> sudo systemctl enable TIMER.timer  
> sudo systemctl disable TIMER.timer
```

### Showing the timer file contents

```
> sudo systemctl cat TIMER.timer
```

### Checking on a specific timer

```
> sudo systemctl status TIMER.timer
```

#### EXAMPLE 8.3: TIMER STATUS

```
> sudo systemctl status helloworld.timer  
● helloworld.timer - "Run helloworld.service 5min after boot and every 24 hours  
relative to activation time" ❶  
Loaded: loaded (/etc/systemd/system/helloworld.timer; disabled; vendor preset:  
disabled) ❷  
Active: active (waiting) since Tue 2022-10-26 18:35:41 CEST; 6s ago ❸  
Trigger: Wed 2022-10-27 18:35:41 CEST; 23h left ❹  
Triggers: ● helloworld.service ❺  
❻  
Oct 26 18:35:41 neo systemd[1]: Started "Run helloworld.service 5min after boot and  
every 24 hours relative to activation time". ❼
```

- ❶ The timer's file name and description.
- ❷ Lists whether a timer has been successfully parsed and is kept in memory (loaded), shows the full path to the timer file, and shows whether the timer is being started at boot time (enabled) or not (disabled). The first value shows the current system configuration, the second value the vendor preset.

- ③ Indicates whether the timer is active (waiting to trigger events) or inactive. If active, it also shows the time that has passed since the last activation (6 seconds in this example).
- ④ Date and time the timer is triggered next.
- ⑤ Name of the service file the timer triggers.
- ⑥ Optional line pointing to documentation (for example, man pages). If not available, an empty line is shown (as in this example).
- ⑦ Latest journal entry created by the timer.

To list all timers available on the system, use `systemctl list-timers`. The following options are available:

List all active timers:

```
> sudo systemctl list-timers
```

List all timers including inactive ones:

```
> sudo systemctl list-timers --all
```

List all timers matching a pattern:

```
> sudo systemctl list-timers PATTERN
> sudo systemctl list-timers --allPATTERN
```

*PATTERN* must be a name or a shell globbing expression. The operators `*`, `?`, and `[]` may be used. Refer to [man 7 glob](#) for more information on globbing patterns.

List timers matching a certain state:

```
> sudo systemctl list-timers --state=STATE
```

*STATE* takes the following values: `active`, `failed`, `load`, `sub`. See [man systemctl](#) for details.

#### EXAMPLE 8.4: LISTING TIMERS

Running any `systemctl list-timers` results in a table similar to the one below. In this example, all active timers matching the pattern `snapper*` are listed:

```
> sudo systemctl list-timers snapper*
NEXT ①                LEFT ②                LAST ③                PASSED ④
UNIT ⑤                ACTIVATES ⑥
```

```
-----
Tue 2022-10-26 19:00:00 CEST 39min left Tue 2022-10-26 18:00:29 CEST 19min ago
snapper-timeline.timer snapper-timeline.service
Wed 2022-10-27 08:33:04 CEST 14h left Tue 2022-10-26 08:33:04 CEST 9h ago
snapper-cleanup.timer snapper-cleanup.service
```

- ❶ The point in time when the timer runs next.
- ❷ The time left till the next timer run.
- ❸ The point in time when the timer ran last.
- ❹ Time elapsed since the last timer run.
- ❺ The name of the timer unit.
- ❻ The name of the service the timer activates.

## 8.3 Timer types

systemd supports two types of timers: real-time (based on calendar) and monotonic (based on events). Although timers are normally persistent, systemd also allows to set up transient timers that are only valid for the current session.

### Real-time timer

Real-time timers are triggered by calendar events. They are defined using the option OnCalendar.

You can specify when to trigger an event based on date and time. Use the following template:

```
OnCalendar=DayOfWeek❶ Year-Month-Day❷ Hour:Minute:Second❸
```

- ❶ Day of week. Possible values are Sun, Mon, Tue, Wed, Thu, Sat. Leave out to ignore the day of the week.
- ❷ Date. Specify month and day by two digits, year by four digits. Each value can be replaced by the wildcard \* to match every occurrence.
- ❸ Time. Specify each value by two digits. Each value can be replaced by the wildcard \* to match every occurrence.

Applies to all values: Use two dots to define a continuous range (Mon..Fri). Use a comma to delimit a list of separate values (Mon,Wed,Fri).

#### EXAMPLE 8.5: REAL-TIME TIMER EXAMPLES

- 6 p.m. every Friday:

```
OnCalendar=Fri *-*- * 18:00:00
```

- 5 a.m. every day:

```
OnCalendar=Mon..Sun *-*- * 5:00:00
```

- 1 a.m. and 3 a.m. on Sundays and Tuesdays:

```
OnCalendar=Tue,Sun *-*- * 01,03:00:00
```

- Single date:

```
OnCalendar=Mo..Sun 2023-09-23 00:00:01
```

- To specify triggers at different times, you can create more than one OnCalendar entry in a single timer file:

```
OnCalendar=Mon..Fri *-*- * 10:00  
OnCalendar=Sat,Sun *-*- * 22:00
```

For a full list of available features and options, refer to [\*\*man 7 systemd.time\*\*](#) that offers additional information on the following topics:

- shorten the syntax and use abbreviations
- specify repetitions
- find specific days in a month (last day of month, last Sunday, etc.)
- apply time zones

#### Monotonic timers

Monotonic timers are triggered at a specified time elapsed from a certain event, such as a system boot or system unit activation event. Values are defined as time units (minutes, hours, days, months, years, etc.). The following units are supported: usec, msec, seconds, minutes, hours, days, weeks, months, years. There are several options for defining monotonic timers:

- OnActiveSec: time after unit activation

```
OnActiveSec=50minutes
```

- OnBootSec: time after system boot

```
OnBootSec=10hours
```

- OnStartupSec: time after the service manager is started. For system services, this is almost equal to OnActiveSec. Use this for user services where the service manager is started at user login.

```
OnStartupSec=5minutes 20seconds
```

- OnUnitActiveSec: time after the corresponding service was last activated

```
OnUnitActiveSec=10seconds
```

- OnUnitInactiveSec: time after the corresponding service was last deactivated

```
OnUnitInactiveSec=2hours 15minutes 18 seconds
```

## Transient timers

Transient timers are temporary timers that are only valid for the current session. Using these timers, you can either use an existing service file or start a program directly. Transient timers are invoked by running **systemd-run**.

The following example runs the helloworld.service unit every two hours:

```
> sudo systemd-run --on-active="2hours" --unit="helloworld.service"
```

To run a command directly, use the following syntax. In this example, the script /usr/local/bin/helloworld.sh is called directly:

```
> sudo systemd-run --on-active="2hours" /usr/local/bin/helloworld.sh
```

If the command takes parameters, add them separated by space:

```
> sudo systemd-run --on-active="2hours" /usr/local/bin/helloworld.sh --  
language=pt_BR
```

Transient timers can be monotonic or real-time. The following switches are supported and work as described in *Monotonic timers*:

- --on-active
- --on-startup



- [`--on-unit-active`](#)
- [`--on-unit-inactive`](#)
- [`--on-calendar`](#)

For more information, refer to [`man 1 systemd-run`](#).

## 8.4 Testing calendar entries

`systemd` provides a tool for testing and creating calendar timer entries for real-time timers: [`systemd-analyze calendar`](#). It accepts the same argument as the [`OnCalendar`](#) entry required to set up real-time timers.

You can concatenate several arguments separated by space. If the term to test is correct, the output shows you when the timer is triggered next (in local time and UTC). It also shows the string in [`Normalized form`](#), and it is recommended to use that string in the timer file. Consider the following examples:

```
> systemd-analyze calendar "Tue,Sun *-*-* 01,03:00:00"
Normalized form: Tue,Sun *-*-* 01,03:00:00
Next elapse: Sun 2021-10-31 01:00:00 CEST
(in UTC): Sat 2021-10-30 23:00:00 UTC
From now: 3 days left

> systemd-analyze calendar "Mon..Fri *-*-* 10:00" "Sat,Sun *-*-* 22:00"
Original form: Mon..Fri *-*-* 10:00
Normalized form: Mon..Fri *-*-* 10:00:00
Next elapse: Thu 2021-10-28 10:00:00 CEST
(in UTC): Thu 2021-10-28 08:00:00 UTC
From now: 19h left

Original form: Sat,Sun *-*-* 22:00
Normalized form: Sat,Sun *-*-* 22:00:00
Next elapse: Sat 2021-10-30 22:00:00 CEST
(in UTC): Sat 2021-10-30 20:00:00 UTC
From now: 3 days left
```

For recurring timers, use the [`--iterations N`](#) switch to list trigger times, then test whether they work as expected. The argument [`N`](#) specifies the number of iterations you would like to test. The following example string triggers every 8 hours (starting at 00:00:00) on Sundays:

```
> systemd-analyze calendar --iterations 5 "Sun *-*-* 0/08:00:00"
```

```
Original form: Sun *-*- 0/08:00:00
Normalized form: Sun *-*- 00/8:00:00
Next elapse: Sun 2021-10-31 00:00:00 CEST
(in UTC): Sat 2021-10-30 22:00:00 UTC
From now: 3 days left
Iter. #2: Sun 2021-10-31 08:00:00 CET
(in UTC): Sun 2021-10-31 07:00:00 UTC
From now: 3 days left
Iter. #3: Sun 2021-10-31 16:00:00 CET
(in UTC): Sun 2021-10-31 15:00:00 UTC
From now: 4 days left
Iter. #4: Sun 2021-11-07 00:00:00 CET
(in UTC): Sat 2021-11-06 23:00:00 UTC
From now: 1 week 3 days left
Iter. #5: Sun 2021-11-07 08:00:00 CET
(in UTC): Sun 2021-11-07 07:00:00 UTC
From now: 1 week 3 days left
```

## 8.5 Getting e-mail notifications when a timer fails

systemd does not offer a feature similar to cron's MAILTO. The procedure below describes a workaround to enable e-mail notifications when a timer fails.

The procedure consists of the following steps:

1. Create a script that sends an e-mail.
2. Create a systemd service file running the e-mail script.
3. Test the e-mail service file.
4. From the service that the timer controls, call the created e-mail service file via OnFailure.

In the following example, we are using the **mailx** command from package mailx. It requires the Postfix e-mail server to be installed and correctly configured.

1. Create the script /usr/local/bin/send\_systemd\_email.
  - a. The script requires two parameters: \$1, the e-mail address, and \$2, the name of the service file for which the failure notification is received. Both parameters are supplied by the unit file running the mail script.

```
#!/bin/sh
systemctl status --full "$2" | mailx -S sendwait\
-s "Service failure for $2" -r root@$HOSTNAME $1
```

b. Make sure the script is executable:

```
> sudo chmod 755 /usr/local/bin/send_systemd_email
```

2. Create the file `/etc/systemd/system/send_email_to_USER@.service`.

```
[Unit]
Description=Send systemd status information by email for %i to USER

[Service]
Type=oneshot
ExecStart=/usr/local/bin/send_systemd_email EMAIL_ADDRESS %i
User=root
Group=systemd-journal
```

Replace `USER` and `EMAIL_ADDRESS` in the file with the login and e-mail address of the user that should receive the e-mail. `%i` is the name of the service that has failed (it is passed on to the e-mail service by the `%n` parameter).

3. Verify the service file and fix the reported issues:

```
> systemd-analyze verify /etc/systemd/system/send_email_to_USER@.service
```

If the command returns no output, the file has passed the verification successfully.

4. To verify the complete procedure, start the service using the `dbus` instance for testing. (You can use any other service that is currently running. `dbus` is used in this example because the service is guaranteed to run on any installation.)

```
> sudo systemctl start send_email_to_USER@dbus.service
```

If successful, `EMAIL_ADDRESS` receives an e-mail with the subject `Service failure for dbus` containing `dbus` status messages in the body. (This is just a test, there is no problem with the `dbus` service. You can safely delete the e-mail, no action is required).

If the test e-mail has been successfully sent, proceed by integrating it into your service file.

5. To add an e-mail notification to the service, add an `OnFailure` option to the `Unit` section of the service file for which you would like to get notified in the event of failure:

```
[Unit]
Description="Hello World script"
OnFailure❶=send_email_to_USER❷@%n❸.service

[Service]
ExecStart=/usr/local/bin/helloworld.sh
```

- ❶ The `OnFailure` option takes a service as an argument.
- ❷ Replace the part of the service unit file name with the login name.
- ❸ Specifies the name of the service (`helloworld` in this example). This name is available in the e-mail service file as `%i`.

You have successfully set up the failure notification for `systemd` services.



### Tip: Sending e-mail notifications to multiple users

The e-mail service file has the recipient's e-mail address hard-coded. To send notification e-mails to a different user, copy the e-mail service file, and replace the user login in the file name and the e-mail address within the copy.

To send a failure notification to several recipients simultaneously, add the respective service files to the service file (use spaces as a separator):

```
OnFailure=send_email_to_tux@%n.service send_email_to_wilber@%n.service
```

## 8.6 Using timers as a regular user

`systemd` timers can also be used by regular users. It helps you to automate recurring tasks like backups, processing images, or moving data to the cloud.

The same procedures and tasks as for system-wide timers are valid. However, the following differences apply:

- Timer and service files must be placed in `~/.config/systemd/user/`.
- All `systemctl` and `journalctl` commands must be run with the `--user` switch. `systemd-analyze` does *not* require this option.

As a regular user, you must provide the path to the unit files, as in the examples below. Otherwise, if a system-wide timer with the same name exists, it would be executed or listed instead.

```
> systemctl --user start ~/.config/systemd/user/helloworld.timer
> systemctl --user enable ~/.config/systemd/user/helloworld.timer
> systemctl --user list-timers
> journalctl --user -u helloworld.*
> systemctl-analyze ~/.config/systemd/user/helloworld.timer
```

### ! Important: User timers only run during an active session

As with other `systemd` services started as a regular user, user timers only run when the user is logged in. Instead, to start user timers at boot time and keep them running after logout, enable *lingering* for each affected user:

```
sudo loginctl enable-linger USER
```

For more information, refer to `man 1 loginctl`.

### ! Important: Environment variables are not inherited

The `systemd` user instance does not inherit environment variables set by scripts like `~/.profile` or `~/.bashrc`. To check the `systemd` environment, run `systemctl --user show-environment`.

To import any variables missing in the `systemd` environment, specify the following command at the end of your `~/.bashrc`:

```
systemctl --user import-environment VARIABLE1 VARIABLE2
```

## 8.7 Migrating from cron to systemd timers

All cron jobs can be migrated to systemd timers. Find instructions and an example [here](#).

1. Create a service file executing the script. See [Example 8.1, “The service file”](#) for details.
2. Create a timer file executing the service file. See [Example 8.2, “The timer file”](#) for general instructions.

- a. Convert calendar entries. Time is specified differently in cron and systemd. Use the patterns below as a conversion template:

```
Cron:           Minute Hour Day Month DayOfWeek
systemd: OnCalendar=DayOfWeek Year-Month-Day Hour:Minute:Second
```

To test the converted calendar entry, follow the instructions in [Section 8.4, “Testing calendar entries”](#).

- b. Convert cron nicknames (@NICK):

```
Cron      : systemd timer
-----  : -----
@reboot   : OnBootSec=1s
@yearly   : OnCalendar=*-01-01 00:00:00
@annually : OnCalendar=*-01-01 00:00:00
@monthly  : OnCalendar=*-* -01 00:00:00
@weekly   : OnCalendar=Sun *- * - * 00:00:00
@daily    : OnCalendar=*-* - * 00:00:00
@hourly   : OnCalendar=*-* - * *:00:00
```

- c. Convert variable assignments. The systemd variable assignment must go into the [Service] section. You cannot convert MAILTO this way—refer to the next step for this.

```
cron: VARIABLE=VALUE
systemd: Environment="VARIABLE=VALUE"
```

- d. Set up e-mail notifications to replace cron's MAILTO feature by following the instructions in [Section 8.5, “Getting e-mail notifications when a timer fails”](#).

#### EXAMPLE 8.6: CRON TO `systemd` TIMER MIGRATION

Here are the crontab entries which call the script `helloworld.sh` 5 minutes after booting and at 10 o'clock each Monday to Friday:

```
@reboot sleep 300 && /usr/local/bin/helloworld.sh
0 10 * * * 1-5 /usr/local/bin/helloworld.sh
```

The `systemd` service file (`helloworld.service`) calling the script looks like this:

```
[Unit]
Description="Hello World script"
[Service]
ExecStart=/usr/local/bin/helloworld.sh
```

The timer file (`helloworld.timer`) looks like this:

```
[Unit]
Description="Run helloworld.service 5min after boot and at 10am every Mon-Fri"
[Timer]
OnBootSec=5min
OnCalendar=Mon..Fri *-*-* 10:00:*
Unit=helloworld.service
[Install]
WantedBy=multi-user.target
```

## 8.8 Troubleshooting and FAQs

Learn how to debug and troubleshoot `systemd` timers that have failed. Find answers to frequently asked questions on `systemd` timers.

### 8.8.1 Avoiding errors

To avoid errors with `systemd` timers, make sure to follow these best practices:

- Verify that the executable you specify in the service with `ExecStart` runs correctly.
- Check the syntax of the service and timer files by running `systemd-analyze verify FILE`.
- Check execution times of calendar entries by running `systemd-analyze calendar CAL-ENDER_ENTRY`.

## 8.8.2 Event is not triggered

When you activate a timer that contains non-critical errors, `systemd` silently ignores them. For example:

### EXAMPLE 8.7: `systemd` TIMER FILE CUTOUT CONTAINING A NON-FATAL ERROR

```
[Timer]
OnBootSec=5min
OnCalendar=Mon..Fri 10:00
Unit=helloworld.service
```

Line 3 contains a syntax error (`OnCalendar` instead of `OnCalendar`). Since the `[Timer]` section contains a second timer entry (`OnBoot`), the error is not critical and is silently ignored. As a consequence, the Monday to Friday trigger is not executed. The only way to detect the error is to use the command `systemd-analyze verify`:

```
# systemd-analyze verify /etc/systemd/system/helloworld.timer
/etc/systemd/system/helloworld.timer:7: Unknown key name 'OnClendar' in section
'Timer', ignoring.
```

## 8.8.3 Checking the system journal for errors

As with every `systemd` service, events and actions triggered by timers are logged with the system journal. If a trigger does not behave as expected, check the log messages with `journalctl`. To filter the journal for relevant information, use the `-u` switch to specify the `systemd` timers and service files. Use this option to show the log entries for the timer *and* the corresponding service file:

```
sudo journalctl -u helloworld.timer -u helloworld.service
```

or shorter (if applicable):

```
sudo journalctl -u helloworld.*
```

`journalctl` is a tool that supports many options and filters. Please refer to `man 1 journalctl` for in-depth information. The following options are useful for troubleshooting timers:

- `-b`: Only show entries for the current boot.
- `-S today`: Only show entries from today.



- `-x`: Show help texts alongside the log entry.
- `-f`: Start with the most recent entries and continuously print the log as new entries get added. Useful to check triggers that occur in short intervals. Exit with `Ctrl - C`.

## 8.8.4 systemd timer: catching up on missed runs

If a `systemd` timer was inactive or the system was off during the expected execution time, missed events can optionally be triggered immediately when the timer is activated again. To enable this, add the configuration option `Persistent=true` to the `[Timer]` section:

```
[Timer]
OnCalendar=Mon..Fri 10:00
Persistent=true
Unit=helloworld.service
```

## 8.8.5 How to migrate from cron to systemd timers?

All cron jobs can be migrated to `systemd` timers. Here are general instructions on migrating a cron job:

1. Create a service file executing the script. See [Example 8.1, "The service file"](#) for details.
2. Create a timer file executing the service file. See [Example 8.2, "The timer file"](#) for general instructions.
  - a. Convert calendar entries. Time is specified differently in cron and `systemd`. Use the patterns below as a conversion template:

```
Cron:           Minute Hour Day Month DayOfWeek
systemd: OnCalendar=DayOfWeek Year-Month-Day Hour:Minute:Second
```

To test the converted calendar entry, follow the instructions in [Section 8.4, "Testing calendar entries"](#).

- b. Convert cron nicknames (`@NICK`):

```
Cron      : systemd timer
-----  : -----
@reboot   : OnBootSec=1s
```

```
@yearly : OnCalendar=*-01-01 00:00:00
@annually: OnCalendar=*-01-01 00:00:00
@monthly : OnCalendar=*-*-01 00:00:00
@weekly  : OnCalendar=Sun *-*- * 00:00:00
@daily   : OnCalendar=*-*- * 00:00:00
@hourly  : OnCalendar=*-*- *:00:00
```

- c. Convert variable assignments. The `systemd` variable assignment must go into the `[Service]` section. You cannot convert `MAILTO` this way—refer to the next step for this.

```
cron: VARIABLE=VALUE
systemd: Environment="VARIABLE=VALUE"
```

- d. Set up e-mail notifications to replace cron's `MAILTO` feature by following the instructions in [Section 8.5, “Getting e-mail notifications when a timer fails”](#).

#### EXAMPLE 8.8: CRON TO `systemd` TIMER MIGRATION

Here are the crontab entries which call the script `helloworld.sh` 5 minutes after booting and at 10 o'clock each Monday to Friday:

```
@reboot sleep 300 && /usr/local/bin/helloworld.sh
0 10 * * * 1-5 /usr/local/bin/helloworld.sh
```

The `systemd` service file (`helloworld.service`) calling the script looks like this:

```
[Unit]
Description="Hello World script"
[Service]
ExecStart=/usr/local/bin/helloworld.sh
```

The timer file (`helloworld.timer`) looks like this:

```
[Unit]
Description="Run helloworld.service 5min after boot and at 10am every Mon-Fri"
[Timer]
OnBootSec=5min
OnCalendar=Mon..Fri *-*- 10:00:*
Unit=helloworld.service
[Install]
WantedBy=multi-user.target
```


## 8.9 For more information

- For a full reference on systemd timers including advanced configuration options (like delays or handling clock or time zone changes), refer to **man 5 systemd.timer**.
- Basic systemd concepts (<https://documentation.suse.com/smart/systems-management/html/concept-systemd/concept-systemd.html>) ↗
- Starting and stopping systemd services (<https://documentation.suse.com/smart/systems-management/html/reference-systemctl-start-stop-services/reference-systemctl-start-stop-services.html>) ↗
- Enabling and disabling systemd services (<https://documentation.suse.com/smart/systems-management/html/reference-systemctl-enable-disable-services/reference-systemctl-enable-disable-services.html>) ↗
- Debugging failed systemd services (<https://documentation.suse.com/smart/systems-management/html/task-debug-failed-systemd-services/index.html>) ↗
- Sending termination signals to systemd services (<https://documentation.suse.com/smart/systems-management/html/task-send-termination-signals-systemd/task-send-termination-signals-systemd.html>) ↗

## A Legal Notice

Copyright© 2006– 2023 SUSE LLC and contributors. All rights reserved.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or (at your option) version 1.3; with the Invariant Section being this copyright notice and license. A copy of the license version 1.2 is included in the section entitled “GNU Free Documentation License”.

For SUSE trademarks, see <http://www.suse.com/company/legal/> . All other third-party trademarks are the property of their respective owners. Trademark symbols (®, <sup>™</sup> etc.) denote trademarks of SUSE and its affiliates. Asterisks (\*) denote third-party trademarks.

All information found in this book has been compiled with utmost attention to detail. However, this does not guarantee complete accuracy. Neither SUSE LLC, its affiliates, the authors, nor the translators shall be held liable for possible errors or the consequences thereof.

## B GNU Free Documentation License

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

### 0. PREAMBLE

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

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

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

a printed book. We recommend this License principally for works whose purpose is instruction or reference.

### 1. APPLICABILITY AND DEFINITIONS

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

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

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be

a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is

not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or

"History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

## 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non-commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

## 3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

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

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies

in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

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

#### 4. MODIFICATIONS

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

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the

Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document



as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

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

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

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

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

## 5. COMBINING DOCUMENTS

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

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

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

## 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual

copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document

is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

## 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail. If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

## 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automati-

cally terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

## 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

## ADDENDUM: How to use this License for your documents

```
Copyright (c) YEAR YOUR NAME.  
Permission is granted to copy, distribute  
and/or modify this document  
under the terms of the GNU Free  
Documentation License, Version 1.2
```

```
or any later version published by the Free
Software Foundation;
with no Invariant Sections, no Front-Cover
Texts, and no Back-Cover Texts.
A copy of the license is included in the
section entitled "GNU
Free Documentation License".
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with...Texts.” line with this:

```
with the Invariant Sections being LIST
THEIR TITLES, with the
Front-Cover Texts being LIST, and with the
Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.