FRAFOS ABC SBC Installation Guide

Release 5.4.8

FRAFOS GmbH

Nov 07, 2024

Table of Contents

The ABC SBC is distributed in form of a container (systemd or OCI type). It can be run on operating system of customer choice, if the OS supports running of those container types.

FRAFOS also offers an Amazon cloud based solution where the ABC SBC is running as an instance in AWS (EC2). This is by far the fastest installation, the software can be started by several clicks. See amazon.

FRAFOS can also provide a hardware based solution with preinstalled ABC SBC software on a reference hardware, see *Hardware Requirements* for more details.

Chapter 1

Hardware Requirements

FRAFOS ABC SBC is provided as container, internally based on Debian 12 64bit operating system (x86_64 architecture).

Capacity and performance of the system depends mainly on the number and type of processors (CPU), available operating memory (RAM) and the number and performance of network cards (NIC).

There are no specific constraints for vendors of hardware and components, but we do have some suggestions and recommendations for the used hardware and its settings. Generally amount of memory and CPU power increases system resilience against load peaks, Ethernet cards with high packet rate facilitate high media anchoring throughput and fast solid-state drives facilitate WAV and PCAP recording.

Minimum hardware:

- CPU: 1x processor - 64bit architecture
- RAM: 4 GB
- NIC: 1x 1Gb network card
- HDD: 10 GB

Recommended / reference hardware: Fujitsu Primergy RX1330 (M3 or M4)

- CPU: Intel® Xeon® processor E3-1200 family, 4GHz
- RAM: 64 GB (DDR4 2666 MHz, dual channel)
- NIC: 2 x Intel 1Gb adapter
- SSD: 2 x 256 GB

For more details about system capacity and dimensioning, see Sec. Sec-Dimensioning.

Chapter 2

Deployment Modes

According to the system dimensioning and high-availability requirements, ABC SBC can be deployed in different modes:

2.1 Single Node Mode

In single node mode a single ABC SBC container is used. It is necessary to connect this node to a CCM module which provides GUI for administration of system and serves as a configuration master to all connected ABC SBC nodes. The CCM module is distributed as a single container and can be either deployed on the same host together with ABC SBC node or on a different server.

2.2 High Available (HA) Pair Mode

IMPORTANT note: up to ABC SBC release 4.1, it was using HA solution based on Pacemaker. The ABC SBC 4.2 was a transitional release that removed Pacemaker based HA solution. The new HA solution based on keepalived was introduced in ABC SBC 4.3 release.

HA pair ABC SBC installation is formed by two physically identical servers running in an active/hot-standby configuration. Only the active (HA master) server processes signaling and media traffic. In case of any failure, the internal management system performs a failover where the originally standby (HA backup) machine becomes active. Switching the active and standby operation modes is also useful during the upgrades of the system.

Both the HA master and backup servers share virtual IP addresses (VIPs) and communicate with each other over the “Internal Management Interface” - IMI. The HA backup server can check the availability of the HA master server. Once the HA backup server determines that the HA master server is no longer available then the backup server will assume the role of HA master and take over the VIPs used for receiving and sending the media and signaling messages.

Further, the HA master server replicates state information about running sessions to the HA backup machine. Thereby, after a failover the backup server will be able to continue processing already established calls and the failure of the server will not result in dropping of already established calls. Note however that calls are replicated after they are established and calls unanswered yet may be dropped. Also only signaling over connection-less transport protocols is certain to reach the Call Agents as transport protocol context gets lost during failover.

Note: status about non running SEMS process reported by SNMP from nodes in ‘BACKUP’ state should be ignored

2.3 Cluster based solution

For very high traffic and performance requirements, ABC SBC instances (in a single or HA pair mode) can create a cluster. A SIP load balancer is put in front of these cluster nodes so as to distribute the SIP traffic to a particular ABC SBC instances.

Chapter 3

Container Installation

Before actual installation admin should consider enabling coredumps on the host, see coredumpsref. It may be beneficial to have them enabled in advance, to ease later troubleshooting but it needs to be considered that whole host and all containers running there will be influenced.

3.1 Systemd-nspawn

In this section we describe the installation process of the ABC SBC systemd container.

The provided ABC SBC container image is a systemd-nspawn container type. It can be unpacked into separate directory and started from there on operating system of customer choice, which has to be able to run the systemd type of containers, like CentOS or Debian Linux (64bit "x86_64" architecture). The recommended OS to use is Debian 12 stable.

Note: if the host OS supports "selinux", it has to be disabled, because selinux policy is applied also to containers running on the host, and ABC SBC is not able to work with the selinux policy set to enforcing. Also, if the host OS uses "AppArmor", it may need to be either tuned or disabled, to not limit some processes inside the ABC SBC container (like "tcpdump" process).

The exact way of how to start and manage the container depends on the operating system choice and the tools provided by it, like the "machinectl" command. This section lists just general examples and recommendations, based on Debian 12 OS.

The package "systemd-container" has to be installed first:

```
% apt install systemd-container
```

Different network modes can be used on the host:

- So called "host network": the network interfaces are configured on the host server and are shared with the container, and the container itself cannot change any network system settings. Also the proper firewall rules have to be set on the host server, or firewall disabled on it (if firewall is either not needed, or there is separate firewall in the customer network).
- IPvlan or Macvlan network modes: a separate sub-interface is created on the host for the container. The container gets it's own separate IP address, which has to be configured inside the container (either static or dynamic one). Using this mode, the container can also set own nftables based firewall rules. Also, as the container network is not shared with the host network, it is possible to deploy more containers on the same host without possible port conflicts troubles.

The recommended network mode to use is Macvlan. The IPvlan mode can be also used, but has some limitations like it cannot be used if common DHCP server is used, because the DHCP server would need a unique MAC address which IPvlan does not have.

It is recommended to use different hosts for SBC container and CCM container. An ABC Monitor container can be deployed on the same host as CCM container. The SBC and CCM, or SBC and Monitor containers cannot share the same host, unless network model is used which provides some form of networking / loopback isolation, to prevent port conflicts.

For the ABC SBC container to run properly, it is necessary to disable SELinux on the container host machine, if SELinux is enabled. Under Debian, it is usually disabled by default. On CentOS 7 it is usually enabled and can be disabled by editing "/etc/selinux/config" file and setting "SELINUX=disabled" (takes effect on boot), plus running the following command to set it on the running system:

```
% setenforce 0
```

We assume example container name "testsbc" and image file "frafos-sbc-5.0.0.tgz" used in the following steps. Note that the container name corresponds to the directory name, where container is unpacked.

All the commands should be executed under "root" (or equivalent) user.

3.1.1 Unpack the container image

The ABC SBC container is provided in form of gzip tar file. After getting it from Frafos repository or file server, copy it to the target hosting server.

Create a directory for the container, on a partition with enough space. The recommended default path is "/var/lib/machines/<name>":

```
% mkdir -p /var/lib/machines/testsbc
```

Unpack the container image into that directory. Make sure the correct tar options are used to keep all permissions, ownership and attributes:

```
% tar --xattrs -p --numeric-owner -C /var/lib/machines/testsbc \
  -xzf frafos-sbc-5-4-0.tgz
```

3.1.2 Prepare directory for persistent data

This step is optional, but highly recommended. A separate directory, different from the base one containing the unpacked container, from the host server can be "mounted" to the container on startup and used for data that is expected to be persistent in case of container replacement (e.g. when replacing with newer version). This directory is seen as "/data" path inside of the container, and is used for some basic configuration files, traffic pcaps, recordings, backups etc. If separate directory from the host server is not used, the "/data" path is created just under the basic directory holding the container, and is lost if whole container is replaced.

Create directory for ABC SBC data under some host server partition with enough disk space, like:

```
% mkdir -p /var/data/testsbc
```

Note: if more containers are expected to be run on the same host server, make sure each of them uses own separate directory for "/data". Do not use one common path for more containers.

3.1.3 Create container systemd config file

These steps are based on Debian used as host OS, which is the recommended and tested one. If using CentOS 7, please skip to later *Managing the containers under CentOS 7* section.

If some specific settings are needed for the container, different from defaults, a systemd nspawn config has to be created for the container:

First create a directory:

```
% mkdir -p /etc/systemd/nspawn
```

Then edit a new file like "/etc/systemd/nspawn/testsbc.nspawn" with content similar to:

```
[Network]
MACVLAN=ens3
[Exec]
PrivateUsers=off
[Files]
Bind=/var/data/testsbc:/data
```

Adapt the settings according to networking mode used, system interface name(s) and the persistent /data path location.

Details:

- The "MACVLAN=<system interface name>" option enables the Macvlan networking mode, and results in a sub-interface "mv-<interface name>" to be visible inside the container.
- The "Bind=..." option takes two directories separated by colon. The first is the host directory prepared for persistent data, which will be mapped under the second directory ("/data") path inside the container.

3.1.4 Optional: configure container network interface(s)

If the "host network" mode is used, where container shares the interface with host, this section can be skipped.

If the Macvlan or IPvlan network mode is used, which provide separate network sub-interface for the container, then the network has to be configured inside the container.

Create one or more system interface(s) configuration files in "/data/interfaces.d" directory of the container. That directory can be accessed usually from the host using path like "/var/lib/machines/testsbc/data/interfaces.d", if persistent "/data" path is not used, or under the host path where the persistent "/data" mount is available for the container, even before starting the container.

Create a new file in that directory, with content similar to this, if DHCP is used:

```
auto mv-ens3
iface mv-ens3 inet dhcp
```

Or similar to this, if static IP address is used:

```
auto mv-ens3
iface mv-ens3 inet static
      address 192.168.0.123
      netmask 255.255.255.0
      gateway 192.168.0.1
```

Notes:

- The interface name has to correspond to sub-interface name which the host OS creates for the container. Usually, it is named like "mv-XXX" where the "XXX" is the original host side network interface name.

Refer to "man interfaces" man page for more details about the network interface config file options.

3.1.5 Manage the containers

To list running containers, use:

```
% machinectl list
```

To start a container, use:

```
% machinectl start testsbc
```

To stop running container: use either "poweroff" command inside the container, or use the following command on host:

```
% machinectl poweroff testsbc
```

To connect to the container console from the host server, use the following command:

```
% machinectl shell testsbc
```

For more details, refer to "man machinectl" man page.

3.1.6 Managing the containers under CentOS 7

This section applies only to CentOS 7, where the native method of managing containers using "machinectl" command has some limitations, so we recommend to create a new separate systemd service for each container.

Create systemd service file

The container can be started directly from command line using "systemd-nspawn -bD <dir>" command, but usually it is desirable to use a separate systemd service for it, which allows automatic start and better management of the running container.

Create a new systemd service file for the container, by creating file like "/etc/systemd/system/testsbc.service" with content like:

```
[Unit]
Description=Test Sbc Container

[Service]
ExecStart=/usr/bin/systemd-nspawn --machine=testsbc \
--directory=/var/lib/machines/testsbc/ -b \
--bind /var/data/testsbc:/data
Restart=always

[Install]
WantedBy=multi-user.target
```

Notes:

- If the persistent directory for "/data" was not created in the previous step, remove the "`--bind /data/testsbc:/data`" option.
- Adapt the "/var/data/testsbc" path to reflect the actual host directory used for persistent data.
- The "Restart=always" option makes the container to be restarted automatically in case it stops for whatever reason. It can be changed to "Restart=no" if needed.

Call command to update systemd services:

```
% systemctl daemon-reload
```

Start and manage the container

Use the following command to start the container:

```
% systemctl start testsbc
```

If it should be started automatically on host server boot, use:

```
% systemctl enable testsbc
```

To check status, the following command can be used:

```
% systemctl status testsbc
```

To list all running containers:

```
% machinectl list
```

To connect to the container console from the host server, use the following command:

```
% machinectl shell testsbc
```

Please refer to the machinectl man page for more information on how to interact with systemd-nspawn containers.

3.2 Podman

In this section it is described the OCI container installation process under podman for the ABC SBC and the Cluster Config Manager. The procedure is based on podman 4.3.1 used on Debian 12.

Please refer to the official documentation for more detailed information.

3.2.1 Installing podman

For proper Frafos SBC installation, podman 4 version must be installed. Since Debian 11 the podman package is available in official repositories and can be installed on host with Debian OS via:

```
% apt install podman
```

Make sure that the network_backend parameter for podman container is `cni`, it can be updated in this way:

```
 % vi /usr/share/containers/containers.conf

network_backend = "cni"
```

3.2.2 OCI images download

Start by downloading the OCI images directly from Frafos's docker registry (registry.frafos.net) using the following:

```
% podman pull registry.frafos.net/abc/sbc:5.4
% podman pull registry.frafos.net/abc/ccm:5.4
```

Frafos's OCI tagging strategy is the following:

- *5.0.[0,100]*: the tag matches the image exact version (*5.0.1*, *5.0.2*)
- *5.0*: alias to the latest *5.0.X* image for the major release
- *5.0-XX*: alias to an exact *5.0.X* image (*5.0-rc1*, *5.0-dev*)

One may also run the following to pull the desired exact images:

```
% # exact minor release
% podman pull registry.frafos.net/abc/sbc:5.0.42
% # test the release candidate
% podman pull registry.frafos.net/abc/ccm:5.0-rc1
% # test the "latest" 5.0
% podman pull registry.frafos.net/abc/ccm:5.0
```

The access to container registry requires authentication using username and password. The account can be self-created at https://registry.frafos.net/, but the account has to be then assigned to proper project by Frafos. Please contact Frafos support, if your account is not approved yet for access to the ABC Sbc repository project.

The podman command to pull container will prompt for username and password, or this commandline option can be added to pass that:

```
% --creds=[username[:password]]
```

OCI images signature verification

Starting 5.4.0, OCI images available on the public registry (registry.frafos.net) are signed by using the `cosign` utility. Image may be verified using the following:

```
$ cosign verify --key frafos.pub registry.frafos.net/abc/sbc:5.4.0
```

The Frafos certificate can be found at https://doc.frafos.com/keys/cosign.pub

For installation of the `cosign` tool, please refer to https://github.com/sigstore/cosign

3.2.3 Persistent data

It is highly recommended to use persistent `/data` directory for the container, where all configuration, that needs to be preserved across upgrades, is stored.

This can be reached either by mounting an external directory from the host into the container's `/data` directory:

```
% mkdir -p /var/data/ccm
% podman run ... -v /var/data/ccm:/data ...
```

or by using a *podman volume* for it:

```
% podman volume create ccm-data
% podman run ... --mount type=volume,src=ccm-data,target=/data,rw=true ...
```

In the first case, the storage is fully under administrator's control and can be accessed not only from containers, but directly by other processes as well. This can be beneficial for downloading CDRs from ABC SBC, for backups generated from host or for mounting a specific partition if huge data volumes may be expected (ABC Monitor).

In the second case, the storage is fully managed by podman and can be accessed only from containers or via podman commands. This may be sometimes limiting and thus in general rather not the recommended way.

3.2.4 Podman Container Installation

There are many ways how to configure networking with podman which may vary use case by use case. For simplification we will focus on the most common scenario only: separate IPVLAN networks for management and VoIP traffic. This configuration allows multiple containers (Cluster Config Manager, ABC Monitor, ABC SBC) running on the same host or on different hosts, depending on desired performance and other requirements.

In networking matter, if the platform is AWS, Azure or GCP, it is not needed to have network configuration as these platforms are not supporting IPVLAN. Only the "host" network type is supported.

Podman installation for AWS - Azure - GCP

After installing Cluster Config Manager container to the server, it can be created by the following command:

```
% podman create --name ccm --hostname ccm --tz=local --tty \
 --mount type=volume,src=ccm-data,target=/data,rw=true \
 --cap-add=AUDIT_CONTROL --cap-add=NET_RAW --cap-add=AUDIT_WRITE \
 --network host \
 registry.frafos.net/abc/ccm:5.4
```

For ABC SBC container creation:

```
% podman create --name sbc --hostname sbc --tz=local --tty \
 --mount type=volume,src=sbc-data,target=/data,rw=true \
 --cap-add=AUDIT_CONTROL --cap-add=NET_RAW --cap-add=AUDIT_WRITE \
 --pids-limit 65536 \
 --network host \
 registry.frafos.net/abc/sbc:5.4
```

For ABC Monitor container creation:

```
% podman create --name mon --hostname mon --tz=local --tty \
 --mount type=volume,src=mon-data,target=/data,rw=true \
 --cap-add=AUDIT_CONTROL --cap-add=NET_RAW --cap-add=AUDIT_WRITE \
 --network host \
 registry.frafos.net/abc/mon:5.2
```

Notes about parameters:

- The `--pids-limit` can be updated according to the threads number that is used by sems. By default it is 2048, however with higher number of threads (used by sems) sbc requires higher limit.
- The `--tz=local` enables container to use the same timezone with host.

After container creation, a service should be created by the following commands to manage container:

```
% cd /etc/systemd/system/
% podman generate systemd --new -f -n ccm
```

This will create a service named "container-ccm" and this service should be started:

```
% systemctl start container-ccm
% systemctl enable container-ccm
```

Please note that each change (container name, adding a network, adding capabilities) must be added in service file (/etc/systemd/system/container-ccm.service).

To check logs, this command can be run:

```
% podman logs -f ccm
```

Regular podman installation

Interface configuration like IP assignment, DNS, etc. can be passed to the container by podman. For this purpose it is necessary to create couple networks under /etc/cni/net.d/.

Deciding interface numbers should be according to the topology. In basic topology Cluster Config Manager and ABC Monitor should have only a management (mgmt) interface. On the other hand, ABC SBC should have these interfaces by default:

mgmt: will be used for communication between Cluster Config Manager, ABC Monitor and other ABC SBC in case HA is used. For mgmt interface, one IP should be assigned on the host, one IP should be assigned for container.

internal: will be used for SIP/MEDIA communication with internal sip servers. For internal interface, there is no need to assign an IP on the host, only for container.

external: will be used for SIP/MEDIA communication with public sip servers (like providers). For external interface, there is no need to assign an IP on the host, only for container.

If more interfaces are needed for cluster, then external2, external3 etc. can be added as well. Obviously, the names of the interfaces can be updated as customers wish.

For Cluster Config Manager and ABC Monitor a specific routing is not needed as it supposed to have only one interface. If additional routing rule is necessary, please check the note related to routing part under ABC SBC network configuration. For Cluster Config Manager container creation this following part should be added in the file (mgmt.conflist) under /etc/cni/net.d/:

```
{
 "cniVersion": "0.4.0",

 "_comment": "##### Set network name. You can use mgmt, internal, external, external2␣
↪etc.  #####",
 "name": "mgmt",

 "plugins": [
   {
     "type": "ipvlan",

     "_comment": "##### Type interface name (on the host) in the line below. #####",
     "master": "enp7s0",

     "ipam": {
       "type": "static",
       "routes": [
         {
           "dst": "0.0.0.0/0"
         }
       ]

       "_comment": "##### Set IP address & gateway in the line below. #####",
       "addresses": [
         {
           "address": "x.x.x.x/x",
           "gateway": "x.x.x.1"
```

(continues on next page)

(continued from previous page)

```
        }
      ]
    }
  },
  {
    "type": "tuning",
    "capabilities": {
      "mac": true
    }
  }
 ]
}
```

Then please set proper parameters while creating the container:

```
% podman create --name ccm --hostname ccm --tz=local --tty \
  --mount type=volume,src=ccm-data,target=/data,rw=true \
  --cap-add=AUDIT_CONTROL --cap-add=NET_RAW --cap-add=AUDIT_WRITE \
  --network mgmt:interface_name=eth0 \
  --dns=172.22.31.2 \
  registry.frafos.net/abc/ccm:5.4
```

Notes about parameters:

- The `--pids-limit` can be updated according to the threads number that is used by sems. By default it is 2048, however with higher number of threads (used by sems) sbc requires higher limit.
- The `--tz=local` enables container to use the same timezone with host.
- The `NET_RAW` is needed for raw sockets usage and additionally for troubleshooting purposes (for example to allow `ping` utility).
- The `AUDIT_WRITE` and `AUDIT_CONTROL` are capabilities that allow writing records into kernel auditing log and control over the kernel auditing. These capabilities are needed for SSH daemon, cron etc.
- The `--dns=none` flag tells podman not to create `/etc/resolv.conf` in the container.

Please note, that to make per-interface DNS configuration working in an ABC SBC container, it is necessary to allow resolvconf when performing the SBC initial config. To make it working in Cluster Config Manager, a CCM configuration option "Enable to use resolvconf for dns" needs to be set.

For ABC SBC container creation this following part should be added in the files (mgmt.conflist, internal.conflist and external.conflist) under /etc/cni/net.d/ and update "name", "routes", "master" and "addresses" section for each network. Please note that there is no need to assign IP addresses on the host for the interfaces other than the first interface (ssh interface). Here is the template for ABC SBC network configuration:

```
{
 "cniVersion": "0.4.0",

 "_comment": "##### Set network name. You can use mgmt, internal, external, external2␣
↪etc.  #####",
 "name": "mgmt",

 "plugins": [
   {
     "type": "ipvlan",

     "_comment": "##### Type interface name (on the host) in the line below. #####",
     "master": "enp7s0",
```

(continues on next page)

(continued from previous page)

```
    "ipam": {
      "type": "static",
      "routes": [
        {
           "dst": "0.0.0.0/0"
        }
      ],

      "_comment": "##### Set IP address & gw in the line below. #####",
      "addresses": [
        {
          "address": "x.x.x.x/x",
          "gateway": "x.x.x.1"
        }
      ]
    }
  },
  {
    "type": "tuning",
    "capabilities": {
      "mac": true
    }
  }
 ]
}
```

A note related to routing:

Please make sure only one interface should have default gateway. If default route is not needed, then route field should be removed from route section (above). All other specific routing rules should be added in this way under "routes" field:

```
{ "dst": "x.x.x.x/x", "gw": "x.x.x.x" },
```

Then please set proper parameters while creating the container:

```
% podman create --name sbc --hostname sbc --tz=local --tty \
  --mount type=volume,src=sbc-data,target=/data,rw=true \
  --cap-add=NET_ADMIN --cap-add=AUDIT_CONTROL --cap-add=NET_RAW --cap-add=AUDIT_WRITE␣
↪\
  --pids-limit 65536 \
  --network mgmt:interface_name=eth0 \
  --network internal:interface_name=eth1 \
  .... (for other interfaces)
  --dns=172.22.31.2 \
  registry.frafos.net/abc/sbc:5.4
```

Notes about parameters:

- The `--pids-limit` can be updated according to the threads number that is used by sems. By default it is 2048, however with higher number of threads (used by sems) sbc requires higher limit.
- The `--tz=local` enables container to use the same timezone with host.
- The `NET_RAW` is needed for raw sockets usage and additionally for troubleshooting purposes (for example to allow `ping` utility).
- The `AUDIT_WRITE` and `AUDIT_CONTROL` are capabilities that allow writing records into kernel auditing log and control over the kernel auditing. These capabilities are needed for SSH daemon, cron etc.

- The `--dns=none` flag tells podman not to create `/etc/resolv.conf` in the container.

For ABC Monitor container creation, please follow the same steps with Cluster Config Manager container creation by updating "ccm" word to "mon".

After container creation, a service should be created by the following command to manage container:

```
% cd /etc/systemd/system/
% podman generate systemd --new -f -n ccm
```

This will create a service named "container-ccm" and this service should be started:

```
% systemctl start container-ccm
% systemctl enable container-ccm
```

Please note that each change (container name, adding a network, adding capabilities) must be added in service file (/etc/systemd/system/container-ccm.service).

To check logs, this command can be run:

```
% podman logs -f ccm
```

3.2.5 Container management

To list running containers use:

```
% podman ps
```

To list even stopped containers:

```
% podman ps -a
```

To list volumes:

```
% podman volume ls
```

To list all images:

```
% podman image ls
```

To start, stop or restart container:

```
% systemctl start container-ccm
% systemctl stop container-ccm
% systemctl restart container-ccm
```

Executing commands within the container

To open a shell inside the container, use the following command on the host server:

```
% podman exec -it sbc bash
```

Another commands can be executed directly similar way:

```
% podman exec -it sbc ip route
```

Checking journal

One may check the container's systemd journal either using podman logs:

```
% podman logs -f ccm
```

or via executing `journalctl` command in the container:

```
% podman exec -it ccm journalctl -f
```

or:

```
% journalctl -u container-ccm
```

3.2.6 Upgrade Procedure

For a container upgrade, please start by pulling the newest images:

```
% podman pull registry.frafos.net/abc/sbc:5.4
% podman pull registry.frafos.net/abc/ccm:5.4
```

You may check the latest images metadata using:

```
% podman image ls
REPOSITORY                      TAG      IMAGE ID      CREATED        SIZE
registry.frafos.net/abc/ccm     5.4      bc3c1b61316a  2 hours ago    1.03 GB
registry.frafos.net/abc/sbc     5.4      574b44e60f30  5 hours ago    1.25 GB
<none>                          <none>   655cf4d30cbf  24 hours ago   1.25 GB
<none>                          <none>   8e0eb0df41c0  24 hours ago   1.03 GB
```

To re-create and re-start the container (ABC SBC in this case), using the newest image, update the version part (end of the "ExecStart" line) in container service:

```
% vi /etc/systemd/system/container-sbc.service
```

and then reload the daemon and restart the container service:

```
% systemctl daemon-reload
% systemctl restart container-sbc
```

One may then remove the unused images (old, untagged variant), using:

```
% podman image prune
WARNING! This will remove all dangling images.
Are you sure you want to continue? [y/N] y
8e0eb0df41c0c9c8cb6c8154b5fc7f4979869ede92febc7f220b4b6a08ebf133
655cf4d30cbf018198f096bf059283eda27c9f9b0073f92ae748af74316e6be4
```

Chapter 4

Post-installation steps

Note: IMPORTANT: AFTER THE INSTALLATION PROCESS IS COMPLETE AND BEFORE CONFIGURATION AND TESTING BEGINS WE URGE YOU TO WHITELIST THE IP ADDRESS FROM WHICH THE ABC SBC WILL BE ADMINISTERED.

Failure to whitelist the administrator's IP address may – especially during the initial configuration and testing – easily block the administrative access to the machine. Various automated blacklisting techniques can block the whole IP address if they spot unexpected traffic from the IP address. See more details in Section Sec-Abuse.

To whitelist the IP address, visit the administrative GUI under **"Config → Firewall → Exceptions to automatic Blacklists → Add"** as shown in the Figure bellow:

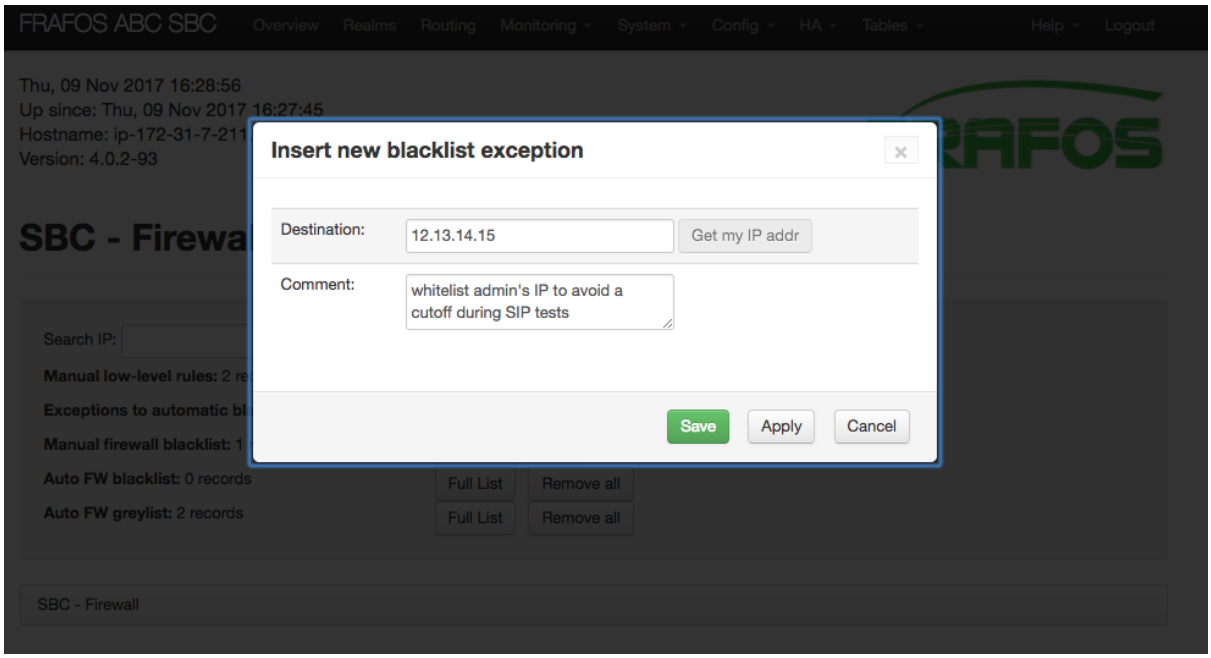

Fig. 1: Warning: Whitelist Administrator's IP Address