Tenks Documentation

Release 2.1.0.dev18

OpenStack Foundation

Dec 17, 2025

CONTENTS

Tenks is a utility that manages virtual bare metal clusters for development and testing purposes.

CHAPTER

ONE

CONFIGURATION

1.1 Hosts

Tenks uses Ansible inventory to manage hosts. A multi-host setup is therefore supported, although the default hosts configuration in `ansible/inventory/` will deploy an all-in-one setup on the host where the `ansible-playbook` command is executed (*localhost*).

- Configuration management of the Tenks cluster is always performed on *localhost*.
- The `hypervisors` group should not directly contain any hosts. Its sub-groups must contain one or more system. Systems in its sub-groups will host a subset of the nodes deployed by Tenks.
  - The `libvirt` group is a sub-group of `hypervisors`. Systems in this group will act as hypervisors using the Libvirt provider.

1.2 Variable Configuration

A variable override file should be created to configure Tenks. Any variables specified in this file will take precedence over their default settings in Tenks. This will allow you to set options as necessary for your setup, without needing to directly modify Tenks variable files. An example override file can be found in `ansible/override.yml.example`.

Most of the configuration you will need to do relates to variables defined in `ansible/host_vars/localhost`. You can set your own values for these in your override file (mentioned above). In addition to other options, you will need to define the types of node youd like to be able to manage as a dict in `node_types`, as well as the desired deployment specifications in `specs`. Format and guidance for available options will be found within the variable file.

Broadly, most variables in `ansible/group_vars/*` have sensible defaults which may be left as-is unless you have a particular need to configure them. A notable exception to this is the variable `physnet_mappings` in `ansible/group_vars/hypervisors`, which should map physical network names to the device to use for that network: this can be a network interface, or an existing OVS or Linux bridge. If these mappings are the same for all hosts in your `hypervisors` group, you may set a single dict `physnet_mappings` in your overrides file, and this will be used for all hosts. If different mappings are required for different hosts, you will need to individually specify them in an inventory host_vars file: for a host with hostname *myhost*, set `physnet_mappings` within the file `ansible/inventory/host_vars/myhost`.

Another variable that may be useful is `bridge_type`. This may be either `openvswitch` (default) or `linuxbridge`, and defines the type of bridges created by Tenks. This may be different from the type of interfaces or bridges in `physnet_mappings`.

The default boot mode is UEFI. This may be changed to legacy BIOS by setting `default_boot_mode` to `bios` in a variable file. The boot mode for nodes may be set individually via `ironic_config.properties.capabilities.boot_mode` in the `specs` list.

1.3 Standalone Ironic

In standalone ironic environments, the placement service is typically not available. To prevent Tenks from attempting to communicate with placement, set `wait_for_placement` to `false`.

It is likely that a standalone ironic environment will not use authentication to access the ironic API. In this case, it is possible to set the ironic API URL via `clouds.yaml`. For example:

```yaml
---
clouds:
  standalone:
    auth_type: "none"
    endpoint: http://localhost:6385
```

Then set the `OS_CLOUD` environment variable to `standalone`.

CHAPTER

TWO

INSTALLATION

2.1 Assumptions

Some assumptions that are made about the configuration of your system are noted below.

It is assumed that

- you already have an OpenStack cloud deployed, for which
  - the host from which Tenks is executed (*localhost*) has access to the OpenStack APIs. These are used for Ironic node enrolment and Nova flavor registration.
  - the OpenStack *OS_** authentication variables are present in *localhost*s environment. These can typically be sourced from your *openrc* file.
- a distinct network device (interface or bridge) is present for each physical network that a hypervisor is connected to.

2.2 Pre-Requisites

Currently, Tenks supports the following OS distributions on the hypervisor:

- CentOS Stream 9
- CentOS Stream 10
- Debian Bookworm (12)
- Rocky Linux 9
- Ubuntu Noble 24.04

To avoid conflicts with Python packages installed by the system package manager it is recommended to install Tenks in a virtualenv. Ensure that the `virtualenv` Python module is available. For cloning and working with the Tenks source code repository, Git is required. These pre-requisites can be installed with a command such as:

```
$ yum install --assumeyes python-virtualenv git
```

If using Open vSwitch for networking, it must be installed and running. Please see the Open vSwitch docs for more details.

2.3 Tenks Installation

Create a virtualenv for Tenks. For example:

```
$ virtualenv tenks
```

Activate the virtualenv and update pip:

```
$ source tenks/bin/activate
(tenks) $ pip install --upgrade pip
```

Obtain the Tenks source code and change into the directory. For example:

```
(tenks) $ git clone https://opendev.org/openstack/tenks.git
(tenks) $ cd tenks
```

Install Tenks and its requirements using the source code checkout:

```
(tenks) $ pip install .
```

Tenks has dependencies on Ansible roles that are hosted by Ansible Galaxy. These can be installed by a command such as:

```
(tenks) $ ansible-galaxy install --role-file=requirements.yml --roles-
↪path=ansible/roles/
```

If you now wish to run Tenks (see *Running Tenks*), keep your virtualenv active. If not, deactivate it:

```
(tenks) $ deactivate
```

CHAPTER

THREE

RUNNING TENKS

3.1 Commands

Tenks has a variable `cmd` which specifies the command to be run. This variable can be set in your override file (see *Configuration*). The possible values it can take are:

- `deploy`: create a virtual cluster to the specification given. This is the default command.
- `teardown`: tear down any existing virtual cluster with the specification given.

3.2 Execution

Currently, Tenks does not have a CLI or wrapper. Before running any of the `ansible-playbook` commands in this section, ensure that your Tenks virtualenv is active (see *Installation*). In this section, `override.yml` represents the path to your override file (see *Configuration*).

The `deploy.yml` playbook will run deployment from start to finish. This can be run by calling:

```
(tenks) $ ansible-playbook --inventory ansible/inventory ansible/deploy.yml --
↪extra-vars=@override.yml
```

`teardown.yml` is `deploy.yml`s mirror image to tear down a cluster. This can be run by calling:

```
(tenks) $ ansible-playbook --inventory ansible/inventory ansible/teardown.yml␣
↪--extra-vars=@override.yml
```

`deploy.yml` and `teardown.yml` automatically set `cmd` appropriately, and they contain various constituent playbooks which perform different parts of the deployment. An individual section of Tenks can be run separately by substituting the path to the playbook(s) you want to run into one of the commands above. The current playbooks can be seen in the Ansible structure diagram in *Architecture*. Bear in mind that you will have to set `cmd` in your override file if you are running any of the sub-playbooks individually.

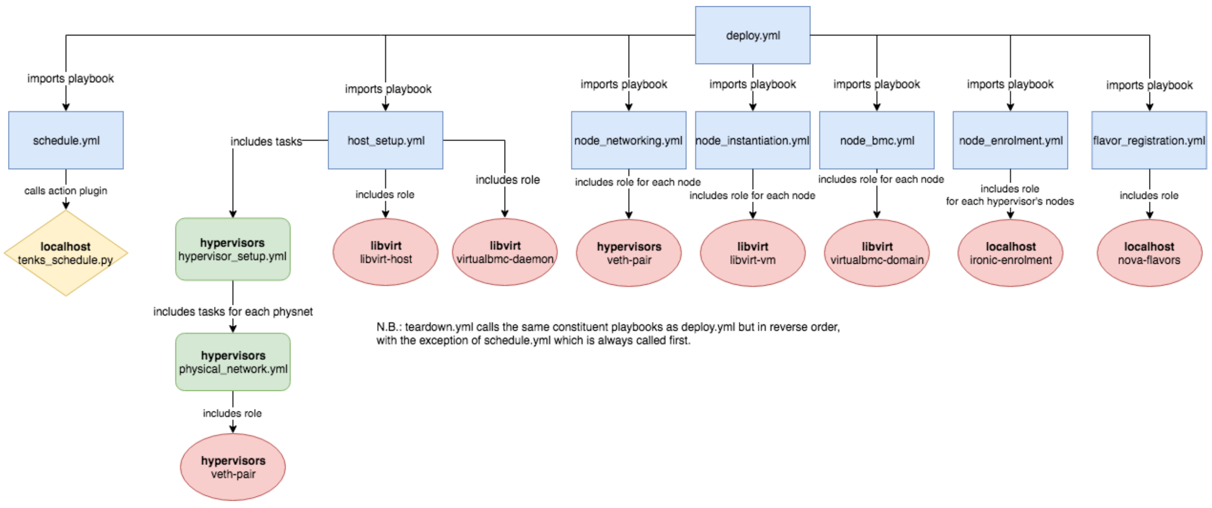
Once a cluster has been deployed, it can be reconfigured by modifying the Tenks configuration and rerunning `deploy.yml`. Node specs can be changed (including increasing/decreasing the number of nodes); node types can also be reconfigured. Existing nodes will be preserved where possible.

CHAPTER FOUR

ARCHITECTURE

4.1 Ansible

A diagram representing the Ansible structure of Tenks can be seen below. Blue rectangles represent playbooks, green rounded rectangles represent task books, red ellipses represent roles and yellow rhombi represent action plugins.

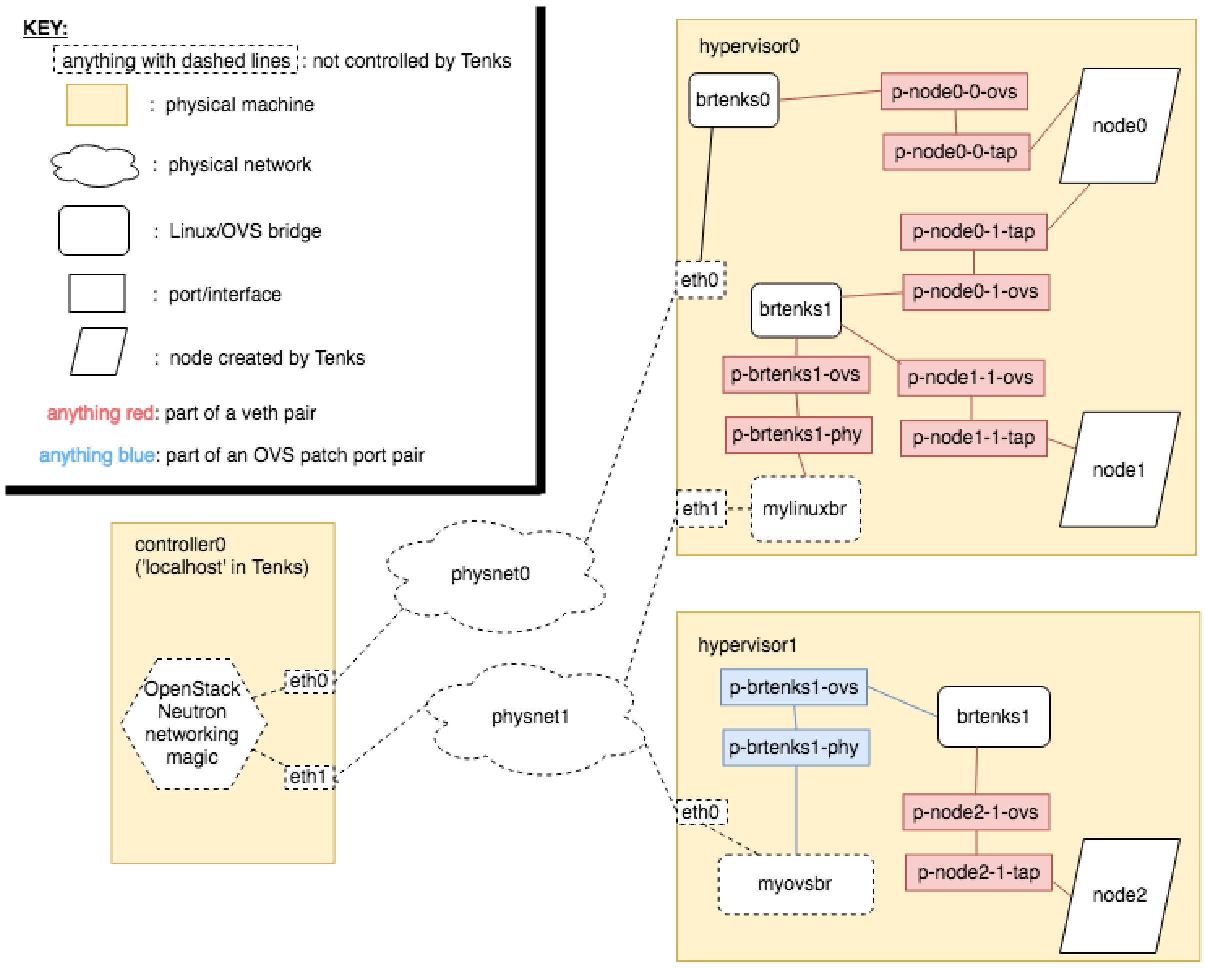


4.2 Networking

Tenks has a concept of physical network which currently must map one-to-one to the hardware networks plugged into the hypervisors. It requires device mappings to be specified on a hypervisor for each physical network that is to be connected to nodes on that hypervisor. This device can be an interface, a Linux bridge or an Open vSwitch bridge. For each physical network that is given a mapping on a hypervisor, a new Tenks-managed Open vSwitch or Linux bridge is created. If the device mapped to this physnet is an interface, it is plugged directly into the new bridge. If the device is an existing Linux bridge, a veth pair is created to connect the existing bridge to the new bridge. If the device is an existing Open vSwitch bridge, an Open vSwitch patch port is created to link the two bridges.

A new veth pair is created for each physical network that each node on each hypervisor is connected to, and one end of the pair is plugged into the Tenks Open vSwitch or Linux bridge for that physical network; the other end will be plugged into the node itself. Creation of these veth pairs is necessary (at least for the Libvirt provider) to ensure that an interface is present in Open vSwitch even when the node itself is powered off.

An example of the networking structure of Tenks is shown below. In this example, one node was requested to be connected to physnet0 and physnet1, and two nodes were requested to be connected just to physnet1.


KEY:
anything with dashed lines : not controlled by Tenks
: physical machine
: physical network
: Linux/OVS bridge
: port/interface
: node created by Tenks
anything red: part of a veth pair
anything blue: part of an OVS patch port pair
hypervisor0
brtenks0
p-node0-0-ovs
p-node0-0-tap
node0
p-node0-1-tap
eth0
brtenks1
p-node0-1-ovs
p-brtenks1-ovs
p-node1-1-ovs
p-brtenks1-phy
p-node1-1-tap
node1
eth1
mylinuxbr
controller0
('localhost' in Tenks)
physnet0
OpenStack Neutron networking magic
eth0
eth1
physnet1
hypervisor1
p-brtenks1-ovs
brtenks1
p-brtenks1-phy
eth0
p-node2-1-ovs
p-node2-1-tap
myovsbr
node2

CHAPTER

FIVE

DEVELOPMENT

5.1 To-Do List

The following is a non-exhaustive list of features that are on the wishlist to be implemented in future.

- **More providers**. It would be useful to extend Tenks to support providers other than Libvirt/QEMU/KVM - for example, VirtualBox, VMware or OpenStack.
- **More platform management systems**. Redfish is gaining momentum in the Ironic community as an alternative to IPMI-over-LAN, so adding support for this to Tenks would widen its appeal. This would involve adding support for configuration of a Redfish BMC emulator, such as that which sushy-tools offers.
- **Increased networking complexity**. As described in *Assumptions*, making the assumption that each network to which nodes are connected will have a physical counterpart imposes some limitations. For example, if a hypervisor has fewer interfaces than physical networks exist in Tenks, either one or more physical networks will not be usable by nodes on that hypervisor, or multiple networks will have to share the same interface, breaking network isolation.

  It would be useful for Tenks to support more complex software-defined networking. This could allow multiple physical networks to safely share the same physical link on a hypervisor. VLAN tagging is used by certain OpenStack networking drivers (networking-generic-switch, for example) to provide tenant isolation for instance traffic. While this in itself is outside of the scope of Tenks, it would need to be taken into account if VLANs were also used for network separation in Tenks, due to potential gotchas when using nested VLANs.
- **More intelligent scheduling**. The current system used to choose a hypervisor to host each node is rather naïve: it uses a round-robin approach to cycle through the hypervisors. If the next hypervisor in the cycle is not able to host the node, it will check the others as well. However, the incorporation of more advanced scheduling heuristics to inform more optimal placement of nodes would be desirable. All of Ansibles gathered facts about each hypervisor are available to the scheduling plugin, so it would be relatively straightforward to use facts about total/available memory or CPU load to shift the load balance towards more capable hypervisors.
- **Command-line interface**. Currently, Tenks must be called by an `ansible-playbook` invocation with multiple parameters. It would be less clunky to introduce a simple CLI wrapper encapsulating some default commands.

5.2 Contribution

Contribution to Tenks development is welcomed. Tenks uses the OpenStack development processes. Code reviews should be submitted to Gerrit, and bugs and RFEs submitted to StoryBoard.

CHAPTER SIX

LIMITATIONS

The following is a non-exhaustive list of current known limitations of Tenks:

- When using the Libvirt provider (currently the only provider), Tenks hypervisors cannot co-exist with a containerised Libvirt daemon (for example, as deployed by Kolla in the nova-libvirt container). Tenks will configure an uncontainerised Libvirt daemon instance on the hypervisor, and this may conflict with an existing containerised daemon. A workaround is to disable the Nova virtualised compute service on each Tenks hypervisor if it is present (for example, `docker stop nova_libvirt`) before running Tenks.