
Enos Documentation

Release 4.4.0

discovery

Feb 13, 2019

Contents:

1	Enos workflow	3
1.1	Getting Started	3
1.2	Provider	5
1.3	Enos command line	15
1.4	Benchmarks	15
1.5	Customizations	17
1.6	Network Emulation	19
1.7	Analysis	21
1.8	Contribute	22
1.9	Jenkins	23
1.10	EnOS Tutorial on top of Grid'5000	27
2	Why Enos ?	47
3	License	49
4	Indices and tables	51

Hint: The source code is available at <https://github.com/BeyondTheClouds/enos>

Enos deploys OpenStack and targets reproducible experiments. It allows easy:

- deployment of the system
- customization of the system
- benchmarking of the system
- visualization of various metrics

Enos is developed in the context of the [Discovery](#) initiative.

A typical experiment using Enos is the sequence of several phases:

- `enos up` : Enos will read the configuration file, get machines from the resource provider and will prepare the next phase
- `enos os` : Enos will deploy OpenStack on the machines. This phase rely highly on Kolla deployment.
- `enos init-os` : Enos will bootstrap the OpenStack installation (default quotas, security rules, ...)
- `enos bench` : Enos will run a list of benchmarks. Enos support Rally and Shaker benchmarks.
- `enos backup` : Enos will backup metrics gathered, logs and configuration files from the experiment.

1.1 Getting Started

1.1.1 Installation

```
$ pip install -U enos
```

You may prefer to go with a virtualenv. Please refer to the [virtualenv](#) documentation and the rest of this section for further information.

Then install enos inside a virtualenv (python3.5+ required):

```
$ mkdir my-experiment && cd my-experiment
$ virtualenv -p python3 venv
$ source venv/bin/activate
(venv) $ pip install -U pip
(venv) $ pip install enos
```

Note: The latest *packaged* version of enos will install the latest *stable* version of OpenStack. If you want to install the development version of OpenStack, you should install enos from sources (see [Contribute](#)).

1.1.2 Configuration

To get started you can get the sample configuration file and edit it:

```
$ enos new > reservation.yaml
$ <editor> reservation.yaml
```

The configuration may vary from one provider to another, please refer to the dedicated *Provider* configuration

Note: If a key is defined several times in the configuration file, only the last occurrence will be taken into account. In particular to switch from one provider to another, you can move down the key `provider` and its associated `resources` key.

1.1.3 Deployment

Once your configuration is done, you can launch the deployment :

```
(venv) $ enos deploy
```

The deployment is the combination of the following three phases:

1. Acquire the raw resources that are necessary for the deployment of OpenStack. Enos acquires resources according to the `provider` and `resources` information in the reservation file. One can perform this phase by calling `enos up`.
2. Deploy OpenStack to the resources acquired during the previous phase. Enos uses the resource list provided by the previous phase and combines it with the information specified in the file targeted by the `inventory` key to produce a file that gives a mapping of which OpenStack services have to be deployed to which resources. Enos then calls the Kolla-Ansible tool with this file to deploy the containerized OpenStack services to the right resources. One can perform this phase by calling `enos os`.

Note: If you don't provide an `inventory` in your current working directory, then Enos uses a default one. You can view it on GitHub at [enos/inventories/inventory.sample](#). Note that the produced file is available at `cwd/current/multinode` (with `cwd` referencing to your current working directory).

Warning: If you run Enos on macOS, chances are that the BSD version of *docopt* has been installed. Since it is not compatible with Kolla-Ansible, it leads to failures during the second phase of Enos. macOS users should first install the GNU version of *docopt*, and call `enos deploy` or `enos os` with an appropriate `PATH` environment variable:

```
(venv) $ brew install gnu-docopt
(venv) $ PATH="/usr/local/opt/gnu-getopt/bin:$PATH" enos deploy
```

3. Initialize the freshly deployed OpenStack. Enos initializes OpenStack with the bare necessities, i.e., install a `member` role, download and install a cirros image, install default flavors (`m1.tiny`, ..., `m1.xlarge`) and setup a network (one public/one private). One can perform this phase by calling `enos init`.

1.2 Provider

Enos offers to test different OpenStack deployments over some resources. In the context of Enos, a resource is anything Enos can SSH on and start a Docker daemon. Thus, a resource could be a bare-metal machine, a virtual machine, or a container resource depending on the testbed used for conduction the experiments. To get these resources, Enos relies on a notion of *provider* and already implements the followings:

1.2.1 Static

The static provider reuses already available resources (machines, network) to deploy OpenStack on.

Installation

Refer to the *Getting Started* section to install Enos.

Configuration

The static provider requires already running resources to deploy OpenStack on. Information in the provider description tells Enos where these resources are and how to access to them.

The following shows an example of possible description. It can serve as basis to build your own configuration that will fit your environment.

```

1  ---
2  # Example of using yaml anchors
3  hosts:
4    1: &h1
5      address: 192.168.142.245
6      alias: enos-0
7      user: root
8      keyfile: .vagrant/machines/enos-0/libvirt/private_key
9    2: &h2
10     address: 192.168.142.244
11     alias: enos-1
12     user: root
13     keyfile: .vagrant/machines/enos-1/libvirt/private_key
14    3: &h3
15     address: 192.168.142.243
16     alias: enos-2
17     user: root
18     keyfile: .vagrant/machines/enos-2/libvirt/private_key
19
20  resources:
21    control:
22      - *h1
23    network:
24      - *h2
25    compute:
26      - *h3
27
28  provider:
29    type: static
30    networks:

```

(continues on next page)

(continued from previous page)

```
31 - start: 192.168.142.3
32   end: 192.168.142.119
33   cidr: 192.168.142.0/24
34   gateway: 192.168.142.1
35   dns: 8.8.8.8
36   roles: ["network_interface"]
37 - start: 192.168.143.3
38   end: 192.168.143.119
39   cidr: 192.168.143.0/24
40   gateway: 192.168.143.1
41   dns: 8.8.8.8
42   roles: ["neutron_external_interface"]
43
44 inventory: inventories/inventory.sample
45 enable_monitoring: no
46 registry:
47   type: none
48 kolla_repo: https://git.openstack.org/openstack/kolla-ansible
49 kolla_ref: stable/queens
50 kolla:
51   kolla_base_distro: centos
52   kolla_install_type: source
53   nova_compute_virt_type: "qemu"
```

Note that the above example is based on running machines given by vagrant and the libvirt provider following this Vagrantfile

In the `resources` there must be at least one host entry for each of the following names:

- control
- compute
- network

In the `networks` section of the provider, a network with role `network_interface` must be define. For more information on network roles please refer to the [kolla documentation](#).

1.2.2 Grid'5000

Installation

Connect to a Grid'5000 frontend of your choice. Then, refer to the *Getting Started* section to install Enos.

If the `virtualenv` module is missing:

```
$ pip install virtualenv --user      # Install virtualenv
$ export PATH=~/.local/bin/:${PATH} # Put it into your path
```

Basic configuration

The provider relies on cluster names to group the wanted resources. For example the following is a valid resources description:

```

1  ---
2  provider:
3      type: g5k
4      job_name: 'enos-jenkins'
5      walltime: '02:00:00'
6
7  resources:
8      paravance:
9          compute: 1
10         network: 1
11         control: 1
12
13     # This will describe the topology of your services
14     inventory: inventories/inventory.sample
15
16     registry:
17         type: internal
18
19     # Enos Customizations
20     enable_monitoring: no
21
22     # Kolla parameters
23     kolla_repo: https://git.openstack.org/openstack/kolla-ansible
24     kolla_ref: stable/queens
25
26     # Vars : globals.yml
27     kolla:
28         kolla_base_distro: centos
29         kolla_install_type: source
30         enable_heat: no

```

Deployment

We suggest you to run the deployment from a dedicated node (especially for large deployment). For now the recommended way to do so is to reserve one node prior of your reservation. In the case of an interactive deployment:

```

frontend$ oarsub -I -l 'walltime=2:00:00'
node$ source venv/bin/activate
node$ <edit configuration>
node$ enos deploy

```

Note:

You'll need to configure the access to the Grid'5000 API by setting the following in your `~/ .execo.conf.py`:

```

# ~/.execo.conf.py
g5k_configuration = {
    'api_username': "your username",
    'api_verify_ssl_cert': False
}

```

Default provider configuration

The provider comes with the following default options:

```
DEFAULT_CONFIG = {
  'job_name': 'Enos',           # Job name in oarstat/gant
  'walltime': '02:00:00',     # Reservation duration time
  'env_name': 'debian9-x64-nfs', # Environment to deploy
  'reservation': '',          # Reservation date
  'job_type': 'deploy',       # deploy, besteffort, ...
  'queue': 'default'         # default, production, testing
}
```

They can be overridden in the configuration file.

Advanced Configuration

The following is equivalent to the basic configuration but allows for a finer grained definition of the resources and associated roles.

```
1 provider:
2   type: g5k
3   job_name: 'enos-jenkins'
4   walltime: '02:00:00'
5   role_distribution: debug
6   resources:
7     machines:
8       - role: control
9         nodes: 1
10        cluster: paravance
11        primary_network: int-net
12        secondary_networks: [ext-net]
13       - role: compute
14         nodes: 1
15        cluster: paravance
16        primary_network: int-net
17        secondary_networks: [ext-net]
18       - role: network
19         nodes: 1
20        cluster: paravance
21        primary_network: int-net
22        secondary_networks: [ext-net]
23     networks:
24       - id: int-net
25         site: renes
26         type: kavlan
27         role: network_interface
28       - id: ext-net
29         site: renes
30         type: kavlan
31         role: neutron_external_interface
32
33 inventory: inventories/inventory.sample
34
35 registry:
36   type: internal
37
```

(continues on next page)

(continued from previous page)

```

38 kolla_repo: https://git.openstack.org/openstack/kolla-ansible
39 kolla_ref: stable/queens
40 kolla:
41   kolla_base_distro: centos
42   kolla_install_type: source
43   enable_heat: no

```

Reservation

In order to reserve in advance the resources needed by your deployment you can set the `reservation` key to the desired start date. And launch `enos deploy`.

```

# Reserve the 29. February 2020 at 19:00:00 for 14 hours
provider:
  type: g5k
  ...
  reservation: 2020-02-29 19:00:00
  walltime: 14:00:00
  ...

```

EnOS will wait for the job to start. You can keep the process running in the background (e.g using a screen). Alternatively you can stop it (once the reservation is done) and relaunch it later with the exact same configuration file. For this purpose you can leverage the `-f` options of EnOS.

1.2.3 Vagrant

Installation

To get started with the vagrant provider, you need to install

- [Vagrant](#)

You'll also need a virtualization backend. EnOS supports both Virtualbox and Libvirt as shown below.

Then, refer to the [Getting Started](#) section to install Enos.

Basic Configuration

The provider relies on virtual machine sizes to group the wanted resources. For example the following is a valid configuration

```

1 ---
2 provider:
3   type: vagrant
4   backend: virtualbox
5   box: generic/debian9
6 resources:
7   extra-large:
8     control: 1
9   medium:
10    network: 1
11    compute: 1
12 inventory: inventories/inventory.sample

```

(continues on next page)

(continued from previous page)

```
13 registry:
14   type: internal
15 enable_monitoring: no
16 kolla_repo: "https://git.openstack.org/openstack/kolla-ansible"
17 kolla_ref: "stable/queens"
18 kolla:
19   kolla_base_distro: "centos"
20   kolla_install_type: "source"
21   enable_heat: "no"
22   nova_compute_virt_type: "qemu"
```

The list of the sizes may be found [here](#).

By default virtualbox will be used. See below to learn how to change the default virtualbox backend.

Use libvirt as the backend for Vagrant

Declaring your provider options as the following will spin up virtual machines using libvirt. Note that `vagrant libvirt` must be installed on your system.

```
1 ---
2 provider:
3   type: vagrant
4   backend: libvirt
5   box: generic/debian9
6 resources:
7   extra-large:
8     control: 1
9   medium:
10    network: 1
11    compute: 1
12 inventory: inventories/inventory.sample
13 registry:
14   type: internal
15 enable_monitoring: no
16 kolla_repo: "https://git.openstack.org/openstack/kolla-ansible"
17 kolla_ref: "stable/queens"
18 kolla:
19   kolla_base_distro: "centos"
20   kolla_install_type: "source"
21   enable_heat: "no"
22   nova_compute_virt_type: "qemu"
```

Use the advanced syntax

The following is equivalent to the basic configuration but allows for a finer grained definition of the resources and associated roles.

```
1 ---
2 provider:
3   type: vagrant
4   backend: libvirt
5   box: generic/debian9
6   resources:
```

(continues on next page)

(continued from previous page)

```

7   machines:
8     - role: control
9       flavor: extra-large
10      number: 1
11      networks: ['network_interface', 'neutron_external_interface']
12     - role: compute
13       flavor: medium
14       number: 1
15       networks: ['network_interface', 'neutron_external_interface']
16     - role: network
17       flavor: medium
18       number: 1
19       networks: ['network_interface', 'neutron_external_interface']
20 inventory: inventories/inventory.sample
21 registry:
22   type: internal
23 enable_monitoring: no
24 kolla_repo: "https://git.openstack.org/openstack/kolla-ansible"
25 kolla_ref: "stable/queens"
26 kolla:
27   kolla_base_distro: "centos"
28   kolla_install_type: "source"
29   enable_heat: "no"

```

Default Configuration

```

DEFAULT_CONFIG = {
    'backend': 'virtualbox',
    'box': 'generic/debian9',
    'user': 'root',
}

```

1.2.4 Openstack

The OpenStack provider allows you to use Enos on an OpenStack cloud. In other words this lets you run OpenStack on OpenStack. In the following, the under-cloud is the underlying OpenStack infrastructure, the over-cloud is the OpenStack configured by Enos.

The over-cloud configured by Enos needs a set of resources to be present on the under-cloud. The first step in the deployment workflow consists in checking or creating such resources. Some resources are mandatory and must be present before the deployment (base image, keypairs, ...), some others can be created or reused during the deployment (private networks). For the latter, you can use the default values set by the provider.

For specific under-clouds (e.g Chameleon), specific providers deriving from the OpenStack provider may be used. They will enforce more default values that fit the under-cloud specificities (e.g specific DNS, base image, ...)

Installation

In addition refer to *Getting Started*, extra dependencies are required. You can install them with

```
pip install enos[openstack]
```

Basic Configuration

The provider relies on flavor names to group the wanted resources. The following gives an idea of the resource description available.

```
provider:
  type: openstack
  <options see below>

resources:
  m1.medium:
    control: 1
    network: 1
  m1.small:
    compute: 10
```

Default provider configuration

The OpenStack provider is shipped with the following default options. These options will be set automatically and thus may be omitted in the configuration file.

```
DEFAULT_CONFIG = {
  "type": "openstack",

  # True if Enos needs to create a dedicated network to work with
  # False means you already have a network, subnet and router to
  # your ext_net configured
  "configure_network": True,

  # Name of the network to use or to create
  # It will be use as external network for the upper-cloud
  "network": {'name': 'enos-network'},

  # Name of the subnet to use or to create
  "subnet": {'name': 'enos-subnet', 'cidr': '10.87.23.0/24'},

  # DNS server to use when creating the network
  "dns_nameservers": ['8.8.8.8', '8.8.4.4'],

  # Floating ips pool
  "allocation_pool": {'start': '10.87.23.10', 'end': '10.87.23.100'},

  # Whether one machine must act as gateway
  # - False means that you can connect directly to all the machines
  # started by Enos
  # - True means that one machine will be assigned a floating ip and used
  # as gateway to the others
  "gateway": True,

  # MANDATORY OPTIONS
  'key_name': None,
  'image': None,
  'user': None
}
```

These options can be overridden in the provider config.

Deployment

Enos will interact with the remote OpenStack APIs. In order to get authenticated you must source your rc file. To use Enos on Openstack there are two distinct cases :

- You have direct access to all your machines. You can set `gateway: False`

Hint: In this case, prior to the Enos deployment, you have probably started a machine to act as a frontend. This machine is in the same network as those used by Enos

- You don't have direct access to all your machines. You have to set `gateway: True` in the configuration. EnOS will use a freshly started server as a gateway to access the other nodes.

1.2.5 Chameleon Cloud (KVM)

This provider is an OpenStack based provider where some options are set to fit the following platform :

- <https://openstack.tacc.chameleoncloud.org>

Basic Configuration

As more default values can be enforced automatically, the following is a valid resources description.

```

1  ---
2  provider:
3    type: chameleonkvm
4    key_name: enos_matt
5    gateway: true
6  resources:
7    ml.medium:
8      compute: 1
9      network: 1
10     control: 1
11
12  # This will describe the topology of your services
13  inventory: inventories/inventory.sample
14
15  registry:
16    type: internal
17
18  # Enos Customizations
19  enable_monitoring: no
20
21  # Kolla parameters
22  kolla_repo: https://git.openstack.org/openstack/kolla-ansible
23  kolla_ref: stable/queens
24
25  # Vars : globals.yml
26  kolla:
27    kolla_base_distro: centos
28    kolla_install_type: source
29    enable_heat: no

```

Default provider configuration

The following options will be set automatically and thus may be omitted in the configuration file :

```
DEFAULT_CONFIG = {
  'type': 'chameleonkvm',
  'image': 'CC-Ubuntu16.04',
  'user': 'cc',
  'dns_nameservers': ['129.114.97.1',
                     '129.114.97.2',
                     '129.116.84.203']
}
```

These options can be overridden in the provider config.

1.2.6 Chameleon Cloud (Bare Metal)

This provider is an OpenStack based provider where some options are set to fit the following platforms :

- <https://chi.uc.chameleoncloud.org/>
- <https://chi.tacc.chameleoncloud.org/>

The provider interacts with blazar to claim automatically a lease.

Basic Configuration

```
1 ---
2 provider:
3   type: chameleonbaremetal
4   key_name: enos_matt
5   gateway: true
6 resources:
7   compute_haswell:
8     compute: 1
9     network: 1
10    control: 1
11
12 # This will describe the topology of your services
13 inventory: inventories/inventory.sample
14
15 registry:
16   type: internal
17
18 # Enos Customizations
19 enable_monitoring: no
20
21 # Kolla parameters
22 kolla_repo: https://git.openstack.org/openstack/kolla-ansible
23 kolla_ref: stable/queens
24
25 # Vars : globals.yml
26 kolla:
27   kolla_base_distro: centos
28   kolla_install_type: source
29   enable_heat: no
```

Note that on Chameleon, they are two groups of machines : compute and storage.

Default provider configuration

The following options will be set automatically and thus may be omitted in the configuration file.

```

DEFAULT_CONFIG = {
  'type': 'chameleonbaremetal',
  # Name of the lease to use
  'lease_name': 'enos-lease',
  # Glance image to use
  'image': 'CC-Ubuntu16.04',
  # User to use to connect to the machines
  # (sudo will be used to configure them)
  'user': 'cc',
  # True iff Enos must configure a network stack for you
  'configure_network': False,
  # Name of the network to use or to create
  'network': {'name': 'sharednet1'},
  # Name of the subnet to use or to create
  'subnet': {'name': 'sharednet1-subnet'},
  # DNS server to use when creating network
  'dns_nameservers': ['130.202.101.6', '130.202.101.37'],
  # Experiment duration
  'walltime': '02:00:00',
}

```

These options can be overridden in the provider config.

Warning: A shared-network is used and may limit the features of the over-cloud (e.g floating ips)

1.2.7 Custom Provider

See *Write a new provider*.

1.3 Enos command line

Once installed Enos give you access to its command line. Please refer to the output of `enos -h`. For a specific command you can use `enos <command> -h`

1.4 Benchmarks

Benchmarks are run by Enos by the mean of a workload description. A workload is a set of scenarios grouped by type. A workload is launched with the following command:

```
(venv) $ enos bench --workload=workload
```

enos will look into the `workload` directory for a file named `run.yml`. This file is the description of the workload to launch. One example is given below:

```
rally:
  enabled: true # default is true
  args:
    concurrency:
      - 1
      - 2
      - 4
    times:
      - 100
  scenarios:
    - name: boot and list servers
      enabled: true # default is true
      file: nova-boot-list-cc.yml
      args:
        sla_max_avg_duration: 30
        times: 50
```

This will launch all the scenarios described under the scenarios keys with all the possible parameters. The parameters are calculated using the cartesian product of the parameters given under the args keys. Locally defined args (scenario level) shadow globally defined args (top level). The same mechanism is applied to the enabled values. The scenario must be parameterized accordingly. The key (rally here) defines the type of benchmark to launch: in the future we may support other type of scenarios.

After running the workload, a backup of the environment can be done through `enos backup`.

1.4.1 Rally

Enos supports natively Rally scenarios. Please refer to the Rally documentation for any further information on this benchmarking tool.

Supported keys :

- `name`: the name of the scenario. Can be any string.
- `file`: must be the path to the scenario file. The path is relative to the workload directory
- `enabled`: Whether to run this scenario
- `args`: Any parameters that can be understood by the rally scenario
- `plugin`: must be the path to the plugin. The path is relative to the workload directory

1.4.2 Shaker

Enos supports natively Shaker scenarios. Please refer to the Shaker documentation for any further information on this benchmarking tool.

Supported keys :

- `name`: the name of the scenario. Can be any string.
- `file`: must be the alias of the scenario. Enos don't support custom scenario yet.
- `enabled`: Whether to run this scenario

1.4.3 Osprofiler

Supporting OSProfiler in Rally benchmarks is planned for Q3 2017.

1.5 Customizations

1.5.1 Changing Kolla / Ansible variables

Custom Kolla / Ansible parameters can be put in the configuration file under the key `kolla`. The complete list of Kolla variables can be found [here](#).

For instance, Kolla uses the `openstack_release` parameter to fix the OpenStack version to deploy. So, Enos tells Kolla to deploy the `4.0.0` version with:

```
kolla:
  openstack_release: "4.0.0"
```

Note that the Kolla code varies from one version of OpenStack to another. You should always target a version of Kolla code that support the deployment of the expected OpenStack. To do so, you can change the git repository/reference of Kolla code with:

```
kolla_repo: "https://git.openstack.org/openstack/kolla-ansible"
kolla_ref: "stable/ocata"
```

You can also your own local clone of `kolla-ansible` with:

```
kolla_repo: "file:///path/to/local/kolla-ansible"
```

Note on the network interfaces:

Providers do their best to configure the network decently. This probably doesn't cover all the possible use cases. But, if you know what interfaces are configured by the provider you can specify a more precise allocation under the `kolla` key. For instance:

```
kolla:
  network_interface: eth1
  neutron_external_interface: eth2
  tunnel_interface: eth3
```

Running from kolla/master

if you want to live on the bleeding edge you can run the latest Kolla code with the latest built kolla images.

```
kolla_repo: "https://git.openstack.org/openstack/kolla-ansible"
kolla_ref: "master"

kolla:
  openstack_release: "master"
```

1.5.2 Changing the topology

Let's assume you want to run the `nova-conductor` in a dedicated node:

1. Add a new node reservation in the configuration file:

```
paravance:
  control: 1
  network: 1
  compute: 1
  conductor-node: 1
```

2. Create a new inventory file in the `inventories` subdirectory (copy paste the sample inventory) and change the group of the conductor service:

```
[nova-conductor:children]
conductor-node
```

3. In the configuration file, points the inventory to use to this new inventory.

4. Launch the deployment as usual, and you'll get the `nova-conductor` on a dedicated node.

1.5.3 Configuration tuning

At some point, Kolla default parameters won't fit your needs. Kolla provides a mechanism to override custom section of configuration files but isn't applicable in our case (at least in the corresponding branch). So we implement a *quick and dirty* way of patching Kolla code to enable custom configuration files to be used (and by extension custom kolla code). See the possible patch declaration in `ansible/group_vars/all.yml`. Patches should be added in the configuration file of the experiment and you can rely on the `{{ cwd }}` key to link patches in your current working directory.

For instance, adding the following in the configuration file tells enos to look into the current working directory for a file called `mariadb_bootstrap.yml` that will replace the `kolla-ansible mariadb start` playbook.

```
patches:
- name: Patch mariadb start
  src: "{{ cwd }}/mariadb_bootstrap.yml
  dst: kolla/ansible/roles/mariadb/tasks/start.yml
  enabled: "yes"
```

1.5.4 Ansible configuration

By default, Enos loads its own `ansible.cfg`. To use another Ansible configuration file, the `ANSIBLE_CONFIG` environment variable can be used. Further information can be found : [see here](#).

1.5.5 Docker registry mirror configuration

EnOS can deploy a docker registry mirror in different ways. This is controlled by the configuration file.

No Registry mirror

```
registry:
  type: none
```

With the above configuration, EnOS won't deploy any registry mirror. Any docker agent in the deployment will use Docker Hub.

Internal Registry mirror

```
registry:
  type: internal
```

With the above configuration, EnOS deploys a fresh registry that acts as a private docker registry mirroring the official one and cache images close to your deployment resources.

This kind of registry can be made persistent by making sure the underlying storage backend is persistent. Historically, it has been provided on Grid5000 by linking a Ceph Rados Block to the registry backend. Thus you can use the following:

```
registry:
  type: internal
  ceph: true
  ceph_keyring: path to your keyring
  ceph_id: your ceph id
  ceph_rbd: rbd in the form "pool/rbd"
  ceph_mon_host: list of ceph monitor addresses
```

Note: The `reservation.yaml.sample` file provides an example of Ceph configuration that relies on the G5k Ceph of Rennes. The [G5k Ceph tutorial](#) will guide you to create your own Rados Block Device.

External Registry mirror

```
registry:
  type: external
  ip: 192.168.142.253
  port: 5000
```

With the above configuration, EnOS will configure all the docker agents to access the registry located at `registry.ip:registry:port`. Note that registry must be an insecure registry.

Note: If you deploy the external registry mirror on the controller node of OpenStack, make sure the port 5000 don't collide with the port of Keystone.

When using EnOS locally, it's a good idea to keep a separated external registry to speed up the deployment.

1.5.6 Single interface deployment

Please refer to this discussion : <https://github.com/BeyondTheClouds/enos/issues/227>

1.6 Network Emulation

1.6.1 Links description

Enos allows to enforce network emulation in terms of latency and bandwidth limitations.

Network constraints (latency/bandwidth limitations) are enabled by the use of groups of nodes. Resources *must* be described using a topology description instead of a resources description. The following example will define 4 groups named `grp1`, `grp2`, `grp3` and `grp4` respectively:

```
topology:
  grp1:
    paravance:
      control: 1
      network: 1
  grp[2-4]:
    paravance:
      compute: 1
```

Constraints are then described under the `network_constraints` key in the configuration file:

```
network_constraints:
  enable: true
  default_delay: 25ms
  default_rate: 100mbit
  default_loss: 0.1%
  constraints:
    - src: grp1
      dst: grp[2-4]
      delay: 10ms
      rate: 1gbit
      loss: 0%
      symetric: true
```

To enforce the constraints, you can invoke:

```
enos tc
```

Note that The machines must be available, thus the `up` phase must have been called before.

As a result

- the network delay between every machines of `grp1` and the machines of the other groups will be 20ms (2x10ms: symetric)
- the bandwidth between every machines of `grp1` and the machines of the other groups will be 1 Gbit/s
- the packet loss percentage between every machines of `grp1` and the machines of the other groups will be 0%.
- the network delay between every machines of `grp2` and `grp3` (resp. `grp2` and `grp4`) (resp. `grp3` and `grp4`) will be 50ms
- the bandwidth between every machines of `grp2` and `grp3` (resp. `grp2` and `grp4`) (resp. `grp3` and `grp4`) will be 100Mbit/s
- the packet loss percentage between every machines of `grp2` and `grp3` (resp. `grp2` and `grp4`) (resp. `grp3` and `grp4`) will be 0.1%

1.6.2 Checking the constraints

Invoking

```
enos tc --test
```

will generate various reports to validate the constraints. They are based on `fping` and `flent` latency and bandwidth measurements respectively. The reports will be located in the result directory.

1.6.3 Notes

- `default_delay`, `default_rate`, `default_loss` are mandatory
- To disable the network constraints you can specify `enable: false` under the `network_constraints` key and launch again `enos tc`
- To exclude a group from any `tc` rule, you can add an optionnal `except` key to the `network_constraints`:

```
network_constraints:
  enable: true
  default_delay: 25ms
  default_rate: 100mbit
  default_loss: 0%
  constraints:
    - src: grp[1-3]
      dst: grp[4-6]
      delay: 10ms
      rate: 1gbit
      symetric: true
  except:
    - grp1
```

1.7 Analysis

1.7.1 Real-time

Setting `enable_monitoring: true` in the configuration file will deploy a monitoring stack composed of:

- Cadvisor and Collectd agents
- InfluxDB for metrics collection
- Grafana for the visualization / exploration

All these services are accessible on their default ports. For instance you'll be able to access grafana dashboards on port 3000 of the node hosting grafana.

Some dashboards are available [here](#).

1.7.2 Post-mortem

TODO

1.7.3 Annotations

Enos embeds an Ansible plugin to add annotations in Grafana. These annotations are marked points which provide rich information about events when hovered over. Enos uses the `ansible.cfg` file that loads the plugin. The plugin can be disabled by editing the line `callback_whitelist = influxdb_events` in the `ansible.cfg`. Note also that the plugin is automatically disabled when the monitoring tools are not deployed (i.e. when `enable_monitoring = false` is set in the configuration file).

Once the deployment is finished, a compatible dashboard must be used in Grafana to display annotations. An example of such dashboard is available [here](#).

1.8 Contribute

All contributions are welcome on [BeyondTheClouds/enos](#). For any questions, feature requests, issues please use the [GitHub issue tracker](#).

1.8.1 Install from sources and make them editable

```
$ git clone https://github.com/BeyondTheClouds/enos.git
$ cd enos
$ virtualenv venv
$ source venv/bin/activate
(venv) $ pip install -e .
```

1.8.2 Get tox

```
(venv) $ pip install tox
```

1.8.3 Running the tests

```
(venv) $ tox
```

1.8.4 Running syntax checker

```
(venv) $ tox -e pep8
```

1.8.5 Generate the documentation

```
(venv) $ tox -e docs
```

1.8.6 Other Topics

Write a new provider

The actual implementation gives providers for *Static* resources, *Vagrant*, *Grid'5000* and *Openstack* itself. If you want to support another testbed, then implementing a new provider is easy as 500 lines of Python code.

The new provider should follow the [provider.py](#) interface which consists in three methods: `init`, `destroy` and `default_config`. Another good starting point is the simple [static implementation](#).

Init Method

The `init` method provides resources and provisions the environment. To let the provider know what kind and how many resources should be provided, the method is fed with the `config` object that maps the reservations file. So a provider can access the resource description with:

```
rsc = config['resources']

# Use rsc to book resources ...
```

At the end of the `init`, the provider should return a list of `host.py` that Enos can SSH on, together with a pool of available IP for OpenStack Network.

Destroy Method

The `destroy` method destroys resources that have been used for the deployment. The provider can rely on the environment variable to get information related to its deployment.

Default Provider Configuration Methods

The `default_config` specifies keys used to configure the provider with a `dict`. A key could be *optional* and so should be provided with a default value, or *required* and so should be set to `None`. The user then can override these keys in the reservation file, under the `provider` key. Keys marked as `None` in the `default_config` are automatically tested for overriding in the reservation file.

Provider Instantiation

Enos automatically instantiates a provider based on the name specified in the `reservation.yaml`. For instance, based on the following reservation file,

```
provider: "my-provider"
```

Enos seeks for a file called `my-provider.py` into `enos/provider` and instantiates its main class. But sometimes, the provider requires extra information for its initialisation. The good place for this information is to put it under the `provider` key. In this case, the provider name should be accessible throughout the `type` key:

```
provider:
  type: "my-provider"
  extra-var: ...
  ...
```

Then the provider can access `extra-var` with `config['provider']['extra-var']`. Supported extra information is documented into the provider documentation.

1.9 Jenkins

1.9.1 Jobs

We provide here an overview of the jobs that can be runned by our [Jenkins](#).

1.9.2 Grid5000

Workflow:

- Deploys OpenStack in a standalone mode using the Grid'5000 provider.
- Check if a server can be booted and if Rally can be launched.

1.9.3 Grid'5000 environment

Workflow:

- Pull all the docker images on the slave
- Save the environment in /tmp/enos.tar.gz of the slave.

Listing 1: Configuration

```

---
# ##### #
# Enos Configuration File #
# ##### #

# Resources description : this is provider specific
hosts:
  1: &h1
    address: 127.0.0.1
    alias: enos-node
    user: root

resources:
  control:
    - *h1
  network:
    - *h1
  compute:
    - *h1

# Provider description and its options
provider:
  type: static
  network:
    start: 192.168.143.3
    end: 192.168.143.119
    cidr: 192.168.143.0/24
    gateway: 192.168.143.1
    dns: 8.8.8.8
    extra_ips:
      - 192.168.142.100
      - 192.168.142.101
      - 192.168.142.102
      - 192.168.142.103
      - 192.168.142.104
  eths:
    - eth1
    - eth2

# ##### #

```

(continues on next page)

(continued from previous page)

```

# Inventory to use #
# ##### #

# This will describe the topology of your services
inventory: inventories/inventory.sample

# true iff a monitoring stack should be deployed
# cadvisor/collectd - influxdb - grafana
enable_monitoring: true

registry:
  type: none

# ##### #
# Kolla parameters #
# ##### #
# Repository
kolla_repo: https://git.openstack.org/openstack/kolla-ansible
kolla_ref: stable/queens

# globals.yml
kolla:
  kolla_base_distro: centos
  kolla_install_type: source
  # openstack_logging_debug: True
  enable_heat: "no"

  node_custom_config: patch/

```

1.9.4 Packaging

Workflow:

- Deploys OpenStack in a standalone mode using the Static provider in a vagrant box.
- Check if a server can be booted and if Rally can be launched.
- Destroy the deployment (leave the images)
- Package the box

Listing 2: Configuration

```

---
# Example of using yaml anchors
hosts:
  1: &h1
    address: 127.0.0.1
    alias: enos-node
    user: root

resources:
  control:
    - *h1
  network:
    - *h1

```

(continues on next page)

(continued from previous page)

```

compute:
  - *h1

provider:
  type: static
  networks:
    - start: 192.168.143.3
      end: 192.168.143.119
      cidr: 192.168.143.0/24
      gateway: 192.168.143.1
      dns: 8.8.8.8
      roles: ["network_interface"]
    - start: 192.168.142.3
      end: 192.168.142.119
      cidr: 192.168.142.0/24
      gateway: 192.168.142.1
      dns: 8.8.8.8
      roles: ["neutron_external_interface"]

inventory: inventories/inventory.sample
enable_monitoring: no
registry:
  type: none
kolla_repo: https://git.openstack.org/openstack/kolla-ansible
kolla_ref: stable/queens
kolla:
  kolla_base_distro: centos
  kolla_install_type: source
  nova_compute_virt_type: "qemu"

```

1.9.5 Topology

Workflow:

- Deploys nodes using vagrant and apply some network constraints
- Validate those constraints (manually)

Listing 3: Configuration

```

---
# ##### #
# Vagrant reservation parameters #
# ##### #
provider:
  type: vagrant
topology:
  grp1:
    extra-large:
      control: 1
  grp2:
    medium:
      network: 1
      compute: 1
network_constraints:

```

(continues on next page)

(continued from previous page)

```

enable: true
default_delay: 10ms
default_rate: 100mbit
default_loss: 0

# ##### #
# Inventory to use #
# ##### #

# This will describe the topology of your services
inventory: inventories/inventory.sample

# ##### #
# docker registry parameters #
# ##### #
registry:
  type: none

# ##### #
# Monitoring stack parameters #
# ##### #
enable_monitoring: no

# ##### #
# Kolla parameters #
# ##### #
# Repository
kolla_repo: "https://git.openstack.org/openstack/kolla-ansible"
kolla_ref: "stable/queens"

# Vars : globals.yml
kolla:
  kolla_base_distro: "centos"
  kolla_install_type: "source"
  enable_heat: "no"

```

1.10 EnOS Tutorial on top of Grid'5000

OpenStack¹ has become the *defacto* solution to operate compute, network and storage resources in public and private Clouds. This lab aims at exploring EnOS^{2,3}, a holistic framework to conduct evaluations of different OpenStack configurations in an easy and reproducible manner. In particular, EnOS helps you in deploying real OpenStack instances on different types of infrastructure (from virtual environments based on VMs like Vagrant, to real large-scale testbeds composed of bare-metal machines like Grid'5000), stressing it and getting feedback. This lab is composed of two part:

The first part is about getting started with EnOS. More precisely we are going to:

- Deploy and configure OpenStack on Grid'5000 using EnOS.
- Operate this OpenStack to manage IaaS resources (*e.g.*, boot VMs).
- Understand the benchmark mechanisms and run some evaluations.
- Visualize the collected metrics through Grafana.

¹ <https://www.openstack.org/>

² <https://hal.inria.fr/hal-01415522v2>

³ <https://enos.readthedocs.io/en/stable/>

For those who desire to go further, we propose to use EnOS to investigate OpenStack in WAN networks. In this investigation we will study the impact of a specific feature used in such context, just like a developer would do. To that end, we will:

- Simulate a WAN-wide topology with EnOS by playing with traffic shaping.
- See how EnOS can be used to customize OpenStack (enable/disable features).
- Benchmark the deployed OpenStack and backup metrics.
- Analyze the collected metrics to highlight the impact of features.

1.10.1 1 Requirements and Setup

To follow the lab you'll need :

- A Web browser (*e.g.*, Firefox)
- A Grid'5000 account
- An SSH client (*e.g.*, OpenSSH on Linux/Mac, Putty on Windows)
 - Follow the [G5K's recommendations](#) and edit your `~/.ssh/config` file to configure SSH for Grid'5000.
 - Be sure your configure works by typing `ssh rennes.g5k` for instance.

1.10.2 2 Presentation

Note: Since OpenStack deployment can be quite long (~20, 30 minutes) you might be interested in starting its deployment before reading the presentation of OpenStack and EnOS (you can [4 Deploy OpenStack using EnOS](#) and come back later).

Adrien Lebre gave a lecture regarding Cloud Computing, OpenStack and EnOS. You can find the slides of this lecture [here](#). In the following, we quickly present some information regarding OpenStack, EnOS and the lab we are going to set today.

2.1 OpenStack

OpenStack is the *defacto* solution to manage infrastructures (*i.e.*, compute, network, storage resources). To that end, it provides management mechanisms as a modular platform composed of several projects, each of which is in charge of an aspect of the infrastructure management. Among the various projects (30+), here is a selection corresponding to the bare necessities to operate infrastructure:

Nova the compute resource manager (*i.e.*, virtual/bare-metal machines and containers)

Glance the image store

Neutron the network manager for compute resources interconnection

Keystone the authentication/authorization manager

Each project are themselves based on multiple modules. Since OpenStack is designed as a distributed software, each module can be deployed on different physical/virtual machines. For instance, here is a set of modules that compose Nova:

- `nova-api`: in charge of managing users' requests
- `nova-scheduler`: in charge of scheduling compute resources on compute nodes

- `nova-compute`: in charge of the life-cycle of compute resources
- ...

To provide all the features expected by an infrastructure manager, OpenStack's modules need cooperation. For instance, when a user asks `nova` to boot a VM, the image is fetched from `glance`, its network interfaces are configured by `neutron`, supposing `keystone` authorized the operation. Such cooperation is possible through three communication channels:

REST APIs used for inter-project communications

Message queue (RabbitMQ) used for intra-project communications

Database (MariaDB) used to store project states

From the user viewpoint, OpenStack can be operated by three ways:

- Horizon: the OpenStack service in charge of providing a Web GUI
- The OpenStack CLI
- REST APIs

2.2 EnOS

EnOS is a holistic framework to conduct evaluations of different OpenStack configurations in an easy and reproducible manner. In particular, EnOS helps you in deploying real OpenStack instances on different types of infrastructure (from virtual environments based on VMs like Vagrant, to real large-scale testbeds composed of bare-metal machines like Grid'5000), stressing it and getting feedback.

Many projects exist to deploy OpenStack (e.g. OpenStack-Ansible⁴, OpenStack-Chef⁵, OpenStack Kolla⁶, Kubernetes⁷, Juju⁸). EnOS relies on the Kolla OpenStack project to deploy OpenStack modules as Docker containers.

EnOS' workflow is the following:

- `enos up`: book, provision and bootstrap testbed resources
 - install dependencies (Docker)
 - install monitoring tools (cAdvisor, collectd, influxdb, grafana)
- `enos deploy`: deploy OpenStack (based on Kolla)
- `enos bench`: benchmark OpenStack
- `enos backup`: backup the collected metrics
- `enos destroy`: release resources

2.3 Topology deployed in this lab

The lab makes use of EnOS to deploy OpenStack on Grid'5000. In particular, we will need four G5K machines for our deployment:

- *enos* node: A machine we will deploy ourselves to run EnOS.
- *control* node: A machine that hosts the control modules, projects' APIs and databases.

⁴ <https://github.com/openstack/openstack-ansible>

⁵ <https://github.com/openstack/openstack-chef-repo>

⁶ <https://docs.openstack.org/developer/kolla-ansible/>

⁷ <https://github.com/kubernetes/kubernetes>

⁸ <https://jujucharms.com/openstack>


```
# fnantes:~$
ssh rennes
# frennes:~$ -- Not mandatory, but recommended
tmux
# frennes:~$ -- Let's connect to the enos node
oarsub -I -l "nodes=1,walltime=4:00:00" -p "cluster='parapide'"
```

Here, we get a new machine in interactive mode (*i.e.*, `-I`) for the next four hours from the parapide cluster. If it succeeds you should be directly connected to this node (check your prompt).

Note: Note that we created a `tmux` session in order to be resilient to any network failure during your `ssh` session. Whenever you want to restore this session, you can connect to the frontend and attach to your `tmux` session, as follows:

```
# laptop:~$
ssh rennes.g5k
# frennes:~$ -- Stands for "tmux attach"
tmux a
```

Make a directory from where you will install EnOS and run your experiments:

```
# enos:~$
mkdir -p ~/enos-myxp
# enos:~$
cd ~/enos-myxp
```

Then, install EnOS in your working directory (python3.5+ is required):

```
# enos:~/enos-myxp$
virtualenv --python=python3 venv
# (venv) enos:~/enos-myxp$
. venv/bin/activate
# (venv) enos:~/enos-myxp$
pip install "enos[openstack]==4.4.0"
```

Note: Note that EnOS is a Python project. We installed it inside a virtual environment, with `virtualenv`, to avoid any conflict regarding the version of its dependencies. Furthermore, it does not install anything outside the virtual environment which keeps your OS clean. Remember that you have to be in the virtual environment to use EnOS. It means that if you open a new terminal, you need to re-enter the `venv`. For instance, now that EnOS is installed, you can come back as follow:

```
# laptop:~$
ssh rennes.g5k
# frennes:~$
cd ~/enos-myxp
# frennes:~/enos-myxp$
source venv/bin/activate
```

Before going further, check EnOS works by typing `enos --help`:

```
Enos: Monitor and test your OpenStack.
[<args> ...] [-e ENV|--env=ENV]
           [-h|--help] [-v|--version] [-s|--silent|--vv]
```

(continues on next page)

(continued from previous page)

```

General options:
  -e ENV --env=ENV  Path to the environment directory. You should
                    use this option when you want to link to a specific
                    experiment. Not specifying this value will
                    discard the loading of the environment (it
                    makes sense for `up`).
  -h --help        Show this help message.
  -s --silent      Quiet mode.
  -v --version     Show version number.
  -vv             Verbose mode.

Commands:
  new              Print a reservation.yaml example
  up              Get resources and install the docker registry.
  os              Run kolla and install OpenStack.
  init           Initialise OpenStack with the bare necessities.
  bench          Run rally on this OpenStack.
  backup         Backup the environment
  ssh-tunnel     Print configuration for port forwarding with horizon.
  tc            Enforce network constraints
  info          Show information of the actual deployment.
  destroy       Destroy the deployment and optionally the related resources.
  deploy        Shortcut for enos up, then enos os and enos config.
  kolla        Runs arbitrary kolla command on nodes
See 'enos <command> --help' for more information on a specific
command.

```

1.10.4 4 Deploy OpenStack using EnOS

4.1 The EnOS configuration file

To deploy OpenStack, EnOS reads a *configuration file*. This file states the OpenStack resources you want to deploy/measure together with their topology. A configuration could say, “Deploy a basic OpenStack on a single node”, or “Put OpenStack control services on ClusterA and compute services on ClusterB”, but also “Deploy each OpenStack services on a dedicated node and add WAN network latency between them”. So that EnOS can deploy such OpenStack over your testbed and run performance analysis.

The description of the configuration is done in a `reservation.yaml` file. You may generate a new one with `enos new > reservation.yaml`. The configuration file is pretty fat, with a configuration sample for all testbed supported by EnOS (G5k, Chameleon, Vagrant, ...).

Use your favorite text editor to open the `reservation.yaml` file, for instance: `vim reservation.yaml`, and edit it to fit your situation – *i.e.*, something like listing [lst:reservation.yaml](#). Three parts of this configuration file are interested for a simple use of EnOS:

- `provider` section (l. 5): Defines on which testbed to deploy OpenStack (*i.e.*, G5k, Chameleon, Vagrant, ...).
- `resources` section (l. 10): Defines the number and role of machines to deploy on the testbed (*e.g.*, book 3 nodes on paravance with 1 control node, 1 network node and 1 compute node).
- `kolla` section (l. 36): Defines the OpenStack configuration, for instance:
 - Which OpenStack version to deploy (*e.g.*, `kolla-ref: "stable/queens"`).
 - Which OpenStack project to enable/disable (*e.g.*, `enable_heat: "no"`).

```

1  ---
2  # ##### #
3  # Grid'5000 reservation parameters #
4  # ##### #
5  provider:
6     type: g5k
7     job_name: 'enos'
8     walltime: '04:00:00'
9
10 resources:
11     paravance:
12         compute: 1
13         network: 1
14         control: 1
15
16 # ##### #
17 # Inventory to use #
18 # ##### #
19 inventory: inventories/inventory.sample
20
21 # ##### #
22 # docker registry parameters #
23 # ##### #
24 registry:
25     type: internal
26
27
28 # ##### #
29 # Enos Customizations #
30 # ##### #
31 enable_monitoring: yes
32
33 # ##### #
34 # Kolla parameters #
35 # ##### #
36 kolla_repo: "https://git.openstack.org/openstack/kolla-ansible"
37 kolla_ref: "stable/queens"
38
39 # Vars : kolla_repo/ansible/group_vars/all.yml
40 kolla:
41     kolla_base_distro: "centos"
42     kolla_install_type: "source"
43     enable_heat: "yes"

```

The `provider` section tells on which testbed to deploy OpenStack plus its configuration. The configuration may vary from one testbed to another. For instance, Grid'5000 and Chameleon are research testbed were resources have to be booked, thus the configuration includes a `walltime` to define the time of your reservation. Conversely, the Vagrant provider starts VM with VirtualBox on your local machine, and thus doesn't include such a option. Please, refer to the EnOS provider documentation¹⁰ to find the configuration parameters depending on the testbed. For the sake of this lab we are going to use the Grid'5000 provider (*i.e.*, `type: g5k`). Note that a `walltime` of 3 hours is enough for the first part of this workshop. If you plan to stay for the second part you should set 5 hours

The `resources` key contains the description of the desired resources and their topology. Once again, way you describe your topology may vary a little bit depending on the testbed you target. Please, refer to the EnOS provider documentation¹⁰ to find examples of resources description depending on the testbed. Here we declare the G5K cluster we target (*e.g.*, `paravance`), as well as the resources we want to deploy on: a `control`, a `network` and a

¹⁰ <https://enos.readthedocs.io/en/stable/provider/index.html>

compute node on which will be deployed all the required OpenStack services.

4.2 Deploy OpenStack

EnOS manages all the aspects of an OpenStack deployment by calling `enos deploy`. Concretely, the `deploy` phase first gets resources on your testbed following your configuration description. Then, it provisions these resources with Docker. Finally, it starts each OpenStack services (e.g. Keystone, Nova, Neutron) inside a dedicated Docker container.

Launch the deployment with:

```
# (venv) enos:~/enos-mxyp$
enos deploy -f reservation.yaml
```

EnOS is now provisioning three machines on the cluster targeted by the `reservation.yaml`. Once the machines are provisioned, EnOS deploy OpenStack services on them, and you can display information regarding your deployment by typing:

```
# (venv) enos:~/enos-mxyp$
enos info
```

In particular, you should see the IP address of the deployed nodes.

While EnOS deploys OpenStack (it takes ~20 to 45 minutes – there are way to speed up your deployment¹¹), you can observe EnOS running containers on the control node. For that, you can access to the control node by typing:

```
# (venv) enos:~/enos-mxyp$
ssh -l root $(enos info --out json | jq -r '.rsc.control[0].address')
# control:~# -- List the downloaded Docker images
docker images
# control:~# -- List the running Docker containers
docker ps
# control:~# -- Go back to `(venv) enos:~/enos-mxyp$`
exit
```

Note: Note that at the end of your session, you can release your reservation by typing:

```
# (venv) enos:~/enos-mxyp$
enos destroy --hard
```

It will destroy all your deployment and delete your reservation.

1.10.5 5 Play with OpenStack

The last service deployed is the OpenStack dashboard (Horizon). Once the deployment process is finished, Horizon is reachable from G5k. More precisely, Horizon runs in a Docker container on the control node, and listens on port 80. To access Horizon from your own web browser (from your laptop), you can create an SSH tunnel from your laptop to control node, located in G5K. To that end, you first need to get control node's IP address, and then create the tunnel. Open a new terminal and type the following:

1. Find the control node address using EnOS:

¹¹ <https://enos.readthedocs.io/en/stable/customization/index.html#internal-registry>

```
# (venv) enos:~/enos-myxp$
enos info
# (venv) enos:~/enos-myxp$
enos info --out json | jq -r '.rsc.control[0].address'
```

2. Create the tunnel from your laptop:

```
# laptop:~$ -- `ssh -NL 8000:<g5k-control>:80 <g5k-site>.g5k`, e.g.,
ssh -NL 8000:paravance-14-kavlan-4.nantes.grid5000.fr:80 rennes.g5k
```

Note: This lab has been designed to **run on a cluster where nodes have two network interfaces**. If you plan to run the lab on a cluster with a single network interface, please run the following script on the network node. You can check how many network interfaces are associated to a cluster by consulting the [G5k Cheatsheet](#). If you are concerned, connect to the network node as root with:

```
# (venv) enos:~/enos-myxp$
ssh -l root $(enos info --out json | jq -r '.rsc.network[0].address')
```

And execute the following script:

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

# The network interface
IF=<interface-network-node-(eno|eth) [0-9]>
# This is the list of the vip of $IF
ips=$(ip addr show dev $IF|grep "inet .*/32" | awk '{print $2}')
if [[ ! -z "$ips" ]]
then
  # vip detected
  echo $ips
  docker exec -ti openvswitch_vswitchd ovs-vsctl add-port br-ex $IF && ip addr flush
  ↪$IF && dhclient -nw br-ex
  for ip in $ips
  do
    ip addr add $ip dev br-ex
  done
else
  echo "nothing to do"
fi
```

Once it is done, you can access Horizon from your web browser through <http://localhost:8000> with the following credentials:

- login: admin
- password: demo

From here, you can reach `Project > Compute > Instances > Launch Instance` and boot a virtual machine given the following information:

- a name (e.g., horizon-vm)
- an image (e.g., cirros)
- a flavor to limit the resources of your instance (I recommend `tiny`)
- and a network setting (must be `private`)

You should select options by clicking on the arrow on the right of each possibility. When the configuration is OK, the `Launch Instance` button should be enabled. After clicking on it, you should see the instance in the `Active` state in less than a minute.

Now, you have several options to connect to your freshly deployed VM. For instance, by clicking on its name, Horizon provides a virtual console under the `Console` tab. Use the following credentials to access the VM:

- login: `cirros`
- password: `cubswin:)`

While Horizon is helpful to discover OpenStack features, this is not how a real operator administrates OpenStack. A real operator prefers command line interface .

5.1 Unleash the Operator in You

OpenStack provides a command line interface to operate your Cloud. But before using it, you need to set your environment with the OpenStack credentials, so that the command line won't bother you by requiring credentials each time.

Load the OpenStack credentials:

```
# (venv) enos:~/enos-myxps$  
. current/admin-openrc
```

You can then check that your environment is correctly set executing the following command that should output something similar to the listing `lst:env-os`:

```
# (venv) enos:~/enos-myxps$  
env|fgrep OS_|sort
```

```
OS_AUTH_URL=http://10.24.61.255:35357/v3  
OS_IDENTITY_API_VERSION=3  
OS_PASSWORD=demo  
OS_PROJECT_DOMAIN_ID=default  
OS_PROJECT_DOMAIN_NAME=default  
OS_PROJECT_NAME=admin  
OS_REGION_NAME=RegionOne  
OS_TENANT_NAME=admin  
OS_USER_DOMAIN_ID=default  
OS_USER_DOMAIN_NAME=default  
OS_USERNAME=admin
```

All operations to manage OpenStack are done through one single command line, called `openstack`. Doing an `openstack --help` displays the really long list of possibilities provided by this command. The following gives you a selection of the most often used commands to operate your Cloud:

List OpenStack running services `openstack endpoint list`

List images `openstack image list`

List flavors `openstack flavor list`

List networks `openstack network list`

List computes `openstack hypervisor list`

List VMs (running or not) `openstack server list`

Get details on a specific VM `openstack server show <vm-name>`

Start a new VM `openstack server create --image <image-name> --flavor <flavor-name> --nic net-id=<net-id> <vm-name>`

View VMs logs `openstack console log show <vm-name>`

Based on these commands, you can use the CLI to start a new tiny cirros VM called `cli-vm`:

```
# (venv) enos:~/enos-myxp$
openstack server create --image cirros.uec\
                        --flavor ml.tiny\
                        --network private \
                        cli-vm
```

Then, display the information about your VM with the following command:

```
# (venv) enos:~/enos-myxp$
openstack server show cli-vm
```

Note in particular the status of your VM. This status will go from `BUILD`: OpenStack is looking for the best place to boot the VM, to `ACTIVE`: your VM is running. The status could also be `ERROR` if you are experiencing hard times with your infrastructure.

With the previous `openstack server create` command, the VM boots with a private IP. Private IPs are used for communication between VMs, meaning you cannot ping your VM from the lab machine. Network lovers will find a challenge here: try to ping the VM from the lab machine. For the others, you have to manually affect a floating IP to your machine if you want it pingable from the enos node.

```
# (venv) enos:~/enos-myxp$
openstack server add floating ip\
  cli-vm\
  $(openstack floating ip create public -c floating_ip_address -f value)
```

You can ask for the status of your VM and its IPs with:

```
# (venv) enos:~/enos-myxp$
openstack server show cli-vm -c status -c addresses
```

Wait one minute or two the time for the VM to boot, and when the state is `ACTIVE`, you can ping it on its floating IP and SSH on it:

```
# (venv) enos:~/enos-myxp$
ping <floating-ip>
# (venv) enos:~/enos-myxp$
ssh -l cirros <floating-ip>
```

Note: Waiting for the IP to appear and then ping it could be done with a bunch of bash commands, such as in listing `lst:query-ip`.

```
FIXED_IP=$(openstack server show cli-vm -c addresses -f value | sed -Er 's/
↪private=(10\.0\.0\.[0-9]+)\.*/\1/g')
FLOATING_IP=$(openstack floating ip list --fixed-ip-address "$FIXED_IP" -f value -c
↪"Floating IP Address" | head -n 1)
COUNT=20
while [[ $COUNT -ne 0 ]] ; do
  ping -c 1 "$FLOATING_IP"
  RC=$?
  if [[ $RC -eq 0 ]] ; then
```

(continues on next page)

1.10.6 6 Stress and Visualize OpenStack Behavior using EnOS

EnOS not only deploys OpenStack according to your configuration, but also instruments it with a *monitoring stack*. The monitoring stack polls performance characteristics of the running services and helps you to understand the behavior of your OpenStack.

Activating the monitoring stack is as simple as setting the `enable_monitoring` to `yes` in your `reservation.yaml`. This key tells EnOS to deploy two monitoring systems. First, `cAdvisor`¹², a tool to collect resource usage of running containers. Using `cAdvisor`, EnOS gives information about the CPU/RAM/Network consumption per cluster/node/service. Second, `Collectd`¹³, a tool to collect performance data of specific applications. For instance, `Collectd` enables EnOS to record the number of updates that have been performed on the Nova database.

The rest of this section, first shows how to visualize `cAdvisor` and `Collectd` information. Then, it presents tools to stress OpenStack in order to collect interesting information.

6.1 Visualize OpenStack Behavior

A popular tool to visualize information provided by `cAdvisor` and `Collectd` (and whatever monitoring system you could use) is `Grafana`¹⁴. `Grafana` is a Web metrics dashboard. A Docker container is in charge of providing this service inside the control node. As a consequence, prior being able to be reachable from your browser, you need to set a tunnel to this service, by typing on your laptop:

```
# laptop:~$ -- `ssh -NL 3000:<g5k-control>:3000 <g5k-site>.g5k`, e.g.,
ssh -NL 3000:paravance-14-kavlan-4.nantes.grid5000.fr:3000 nantes.g5k
```

You can then access `Grafana` at `http://localhost:3000` with the following credentials:

- login: admin
- password: admin

The `Grafana` dashboard is highly customizable. For the sake of simplicity, we propose to use our configuration file that you can get with:

```
# laptop:~$
curl -O http://enos.irisa.fr/tp-g5k/grafana_dashboard.json
```

You have then to import this file into `Grafana`. First, click on the `Grafana` logo > + > Import > Upload `.json` file and select the `grafana_dashboard.json` file. Next, make names of the right column matching names of the left column by selecting the good item in the list. And finish by clicking on Save & Open. This opens the dashboard with several measures on Nova, Neutron, Keystone, RabbitMQ, ... services. Keep the dashboard open until the end of the lab, you will see consumption variation as we will perform stress tests.

6.2 Benchmark OpenStack

Stressing a Cloud manager can be done at two levels: at the *control plane* and at the *data plane*, and so it is for OpenStack. The control plane stresses OpenStack API. That is to say, features we used in the previous section to start a VM, get a floating IP, and all the features listed by `openstack --help`. The data plane stresses the usage of resources provided by an OpenStack feature. For instance, a network data plane testing tool will measure how resources provided by Neutron handle networks communications.

¹² <https://github.com/google/cadvisor>

¹³ <https://collectd.org/>

¹⁴ <https://grafana.com/>

OpenStack comes with dedicated tools that provide workload to stress control and data plane. The one for control plane is called Rally¹⁵ and the one for data plane is called Shaker¹⁶. And these two are well integrated into EnOS.

EnOS looks inside the `workload` directory for a file named `run.yml`.

```
# (venv) enos:~/enos-myxp$
mkdir -p workload
# (venv) enos:~/enos-myxp$
touch workload/run.yml
```

Edit the file `run.yml` with your favorite editor. An example of such a file is given in listing *lst:run.yml*. The `rally` (l. 2) key specifies the list of `scenarios` (l. 9) to execute (here, only the *8.1 Nova scenario for Rally* – available at `~/enos-myxp/workload/nova-boot-list-cc.yml` – that asks Nova to boot VMs and list them) and their customization.

The customization could be done by using the top level `args` (l. 4). In such case, it applies to any scenario. For instance here, `concurrency` (l. 5) and `times` (l. 7) tells Rally to launch 5 OpenStack client for a total of 10 execution of every scenario. The customization could also be done on a per-scenario basis with the dedicated `args` (l. 12), and thus could be only applies to the specific scenario. For instance here, the 30 value overrides the `sla_max_avg_duration` default value solely in the `boot` and `list servers` scenario.

```
1 ---
2 rally:
3   enabled: yes
4   args:
5     concurrency:
6       - 5
7     times:
8       - 10
9     scenarios:
10      - name: boot and list servers
11        file: nova-boot-list-cc.yml
12        args:
13          sla_max_avg_duration: 30
14 shaker:
15   enabled: yes
16   scenarios:
17     - name: OpenStack L3 East-West Dense
18       file: openstack/dense_l3_east_west
```

Calling Rally and Shaker from EnOS is done with:

```
# (venv) enos:~/enos-myxp$
enos bench --workload=workload
```

Note: At the same time as `enos bench` is running, keep an eye on the Grafana dashboard available at <http://localhost:3000>. At the top left of the page, you can click on the clock icon and tells Grafana to automatically refresh every 5 seconds and only display the last 5 minutes.

Rally and Shaker provide a huge list of scenarios on their respective GitHub¹⁷¹⁸. Before going further, go through the Rally list and try to add the scenario of your choice into the `run.yml`. Note that you have to download the scenario

¹⁵ <https://rally.readthedocs.io/en/latest/>

¹⁶ <https://pyshaker.readthedocs.io/en/latest/>

¹⁷ <https://github.com/openstack/rally/tree/master/rally/plugins/openstack/scenarios>

¹⁸ <https://github.com/openstack/shaker/tree/master/shaker/scenarios/openstack>

file in the `workload` directory and then put a new item under the `scenarios` key (l. 9) . The new item should contain, at least, the name of the scenario and its file path (relative to the `workload` directory).

6.3 Backup your results

Rally and Shaker produce reports on executed scenarios. For instance, Rally produces a report with the full duration, load mean duration, number of iteration and percent of failures, per scenario. These reports, plus data measured by cAdvisor and Collectd, plus logs of every OpenStack services can be backup by EnOS with:

```
# (venv) enos:~/enos-myxp$
enos backup --backup_dir=benchresults
```

The argument `backup_dir` tells where to store backup archives. If you look into this directory, you will see, among others, an archive named `<controler-node>-rally.tar.gz`. Concretely, this archive contains a backup of Rally database with all raw data and the Rally reports. You can extract the Rally report of the *Nova boot and list servers* scenario with the following command and then open it in your favorite browser:

```
# (venv) enos:~/enos-myxp$
tar --file benchresults/*-rally.tar.gz\
  --get $(tar --file benchresults/*-rally.tar.gz\
    --list | grep "root/rally_home/report-nova-boot-list-cc.yml-.*.html")
```

For those interested in playing with deploying applications on top of OpenStack, you can jump to another workshop involving Heat: the OpenStack Orchestration service [here](#).

1.10.7 7 Add Traffic Shaping

EnOS allows to enforce network emulation in terms of latency, bandwidth limitation and packet loss.

7.1 Define Network Constraints

Network constraints (latency/bandwidth limitations) are enabled by the use of groups of nodes. Resources must be described using a `topology` description instead of a `resources` description. For instance, listings *lst:topos-g5k* defines two groups named `grp1` and `grp2`.

```
topology:
  grp1:
    paravance:
      control: 1
      network: 1
  grp2:
    paravance:
      compute: 1
```

Constraints are then described under the `network_constraints` key in the `reservation.yaml` file:

```
network_constraints:
  enable: true
  default_delay: 25ms
  default_rate: 100mbit
  default_loss: 0.1%
  constraints:
    - src: grp1
```

(continues on next page)

(continued from previous page)

```
dst: grp2
delay: 50ms
rate: 1gbit
loss: 0%
symmetric: true
```

Copy your `reservation.yaml` file as `reservation-topo.yaml` with `cp reservation.yaml reservation-topo.yaml` and edit it to include the topology and network constraints definition. An example of such file is given in [8.2 Configuration file with a topology and network constraints](#).

Since our topology is now defined by groups, we need to re-run `enos deploy -f reservation-topo.yaml` (which should be faster than the first time). And then enforce these constraints with `enos tc`, which results in:

- Default network delay is 50ms.
- Default bandwidth is 100Mbit/s.
- Default packet loss percentage is 0.1%.
- Network delay between machines of `grp1` and `grp2` is 100ms (2x50ms: symmetric).
- Bandwidth between machines of `grp1` and `grp2` is 1 Gbit/s.
- Packet loss percentage between machines of `grp1` and `grp2` is 0%.

Note: Invoking `enos tc --test` generates various reports that validate the correct enforcement of the constraints. They are based on `fping` and `flent` latency and bandwidth measurements respectively. The report is located in the `~/enos-myxp/current/_tmp_enos_<g5k-(control|network|compute)>.out`.

7.2 Run Dataplane Benchmarks with and without DVR

Run the Shaker `dense_l3_east_west` scenario with

```
# (venv) enos:~/enos-myxp$
enos bench --workload=workload
```

Note: If you look carefully, you will see that execution of Nova boot and list fails because of a SLA violation. You can try to customize listing `lst:run.yml` to make the test pass.

In this scenario Shaker launches pairs of instances on the same compute node. Instances are connected to different tenant networks connected to one router. The traffic goes from one network to the other (L3 east-west). Get the Shaker report with `enos backup` and analyze it. You will remark that network communications between two VMs co-located on the same compute are 100ms RTT. This is because packet are routed by Neutron service that is inside `grp1` and VMs are inside the `grp2`.

Now, reconfigure Neutron to use DVR¹⁹. DVR will push Neutron agent directly on the compute of `grp2`. With EnOS, you should do so by updating the `reservation.yaml` and add `enable_neutron_dvr: "yes"` under the `kolla` key. Then, tell EnOS to reconfigure Neutron.

```
# (venv) enos:~/enos-myxp$
enos os --tags=neutron --reconfigure
```

And finally, re-execute the `dense_l3_east_west` scenario.

¹⁹ <https://wiki.openstack.org/wiki/Neutron/DVR>

```
# (venv) enos:~/enos-myxps$
enos bench --workload=workload
```

Compare this result with the previous one. You see that you no more pay the cost of WAN latency.

This experiment shows the importance of activating DVR in a WAN context, and how you can easily see that using EnOS. Do not hesitate to take a look at the complete list of Shaker scenarios on their GitHub¹⁸ and continue to have fun with EnOS.

1.10.8 8 Appendix

8.1 Nova scenario for Rally

```
{% set image_name = image_name or "cirros.uec" %}
{% set flavor_name = flavor_name or "m1.tiny" %}
{% set sla_max_avg_duration = sla_max_avg_duration or 60 %}
{% set sla_max_failure = sla_max_failure or 0 %}
{% set sla_max_seconds = sla_max_seconds or 60 %}
---
NovaServers.boot_and_list_server:
-
  args:
    flavor:
      name: {{flavor_name}}
    image:
      name: {{image_name}}
    detailed: true
    auto_assign_nic: true
  runner:
    concurrency: {{concurrency}}
    times: {{times}}
    type: "constant"
  context:
    users:
      tenants: 1
      users_per_tenant: 1
    network:
      start_cidr: "10.2.0.0/24"
      networks_per_tenant: 1
    quotas:
      neutron:
        network: -1
        port: -1
      nova:
        instances: -1
        cores: -1
        ram: -1
  sla:
    max_avg_duration: {{sla_max_avg_duration}}
    max_seconds_per_iteration: {{sla_max_seconds}}
    failure_rate:
      max: {{sla_max_failure}}
```

8.2 Configuration file with a topology and network constraints

```

---
# ##### #
# Grid'5000 reservation parameters #
# ##### #
provider:
  type: g5k
  job_name: 'enos'
  walltime: '04:00:00'

topology:
  grp1:
    paravance:
      control: 1
      network: 1
  grp2:
    paravance:
      compute: 1

network_constraints:
  enable: true
  default_delay: 25ms
  default_rate: 100mbit
  default_loss: 0.1%
  constraints:
    - src: grp1
      dst: grp2
      delay: 50ms
      rate: 1gbit
      loss: 0%
      symmetric: true

# ##### #
# Inventory to use #
# ##### #
inventory: inventories/inventory.sample

# ##### #
# docker registry parameters #
# ##### #
registry:
  type: internal
  ceph: true
  ceph_keyring: /home/discovery/.ceph/ceph.client.discovery.keyring
  ceph_id: discovery
  ceph_rbd: discovery_kolla_registry/datas
  ceph_mon_host:
    - ceph0.rennes.grid5000.fr
    - ceph1.rennes.grid5000.fr
    - ceph2.rennes.grid5000.fr

# ##### #
# Enos Customizations #
# ##### #
enable_monitoring: yes

```

(continues on next page)

(continued from previous page)

```
# ##### #
# Kolla parameters #
# ##### #
kolla_repo: "https://git.openstack.org/openstack/kolla-ansible"
kolla_ref: "stable/queens"

# Vars : kolla_repo/ansible/group_vars/all.yml
kolla:
  kolla_base_distro: "centos"
  kolla_install_type: "source"
  enable_heat: "yes"
```


CHAPTER 2

Why Enos ?

[https://en.wikipedia.org/wiki/Enos_\(chimpanzee\)](https://en.wikipedia.org/wiki/Enos_(chimpanzee))

CHAPTER 3

License

Enos runs performance stress workloads on OpenStack for postmortem analysis. Copyright (C) 2016 Didier Iscovery

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

CHAPTER 4

Indices and tables

- `genindex`
- `modindex`
- `search`