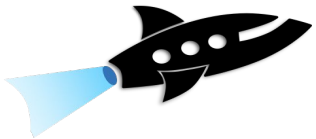
Can We Operate and Use an Edge Computing Infrastructure with OpenStack?

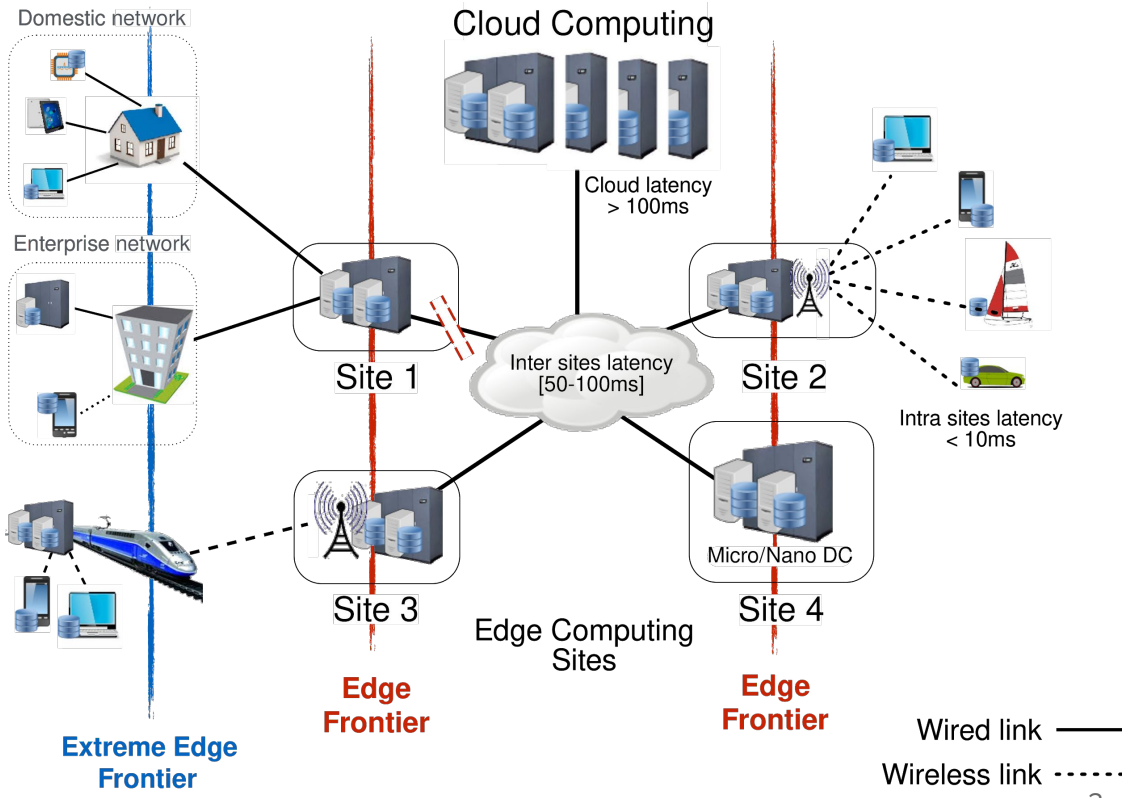
Ronan-Alexandre Cherrueau, Adrien Lebre,
Dimitri Pertin, João Monteiro Soares, Fetahi Wuhib



Edge Computing Infrastructure and its Requirements

What is an Edge Infrastructure?

- New from of Cloud infra.
- Properties
  - Hundreds of DCs
  - Dozen of servers per DC
  - WAN links (from 10 to 300 ms of RTT)
  - Intermittent connection and partition
  - Unmanned



Why using an Edge Infrastructure?

A new paradigm: *the Edge computing*

- Locality aware
  - Reduces end-user to compute node latency for latency-sensitive apps:
    - Internet of Things
    - Smart cars
    - Tactile Internet
    - NFV (telco)
  - Placement enforcement (jurisdiction concern – ask for a compute node in France)

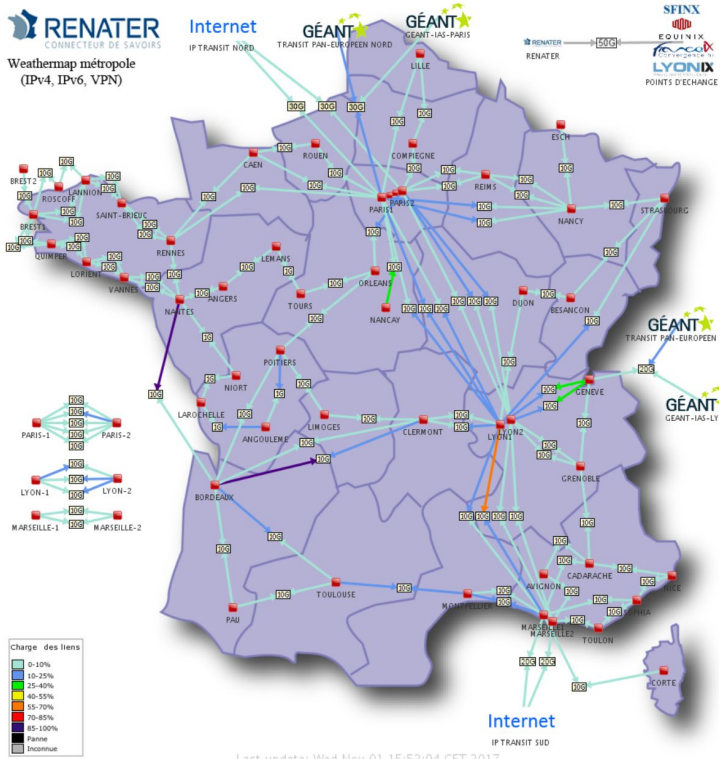
- Reliable – no single point of failure, Split-Brain

An example of an Edge Infrastructure

Renater backbone:

- A red point is a Point of Presence (PoP)
- A micro data-center in each PoP

Requirements to manage such a massively distributed Cloud infrastructure?



Requirements

- We have listed the features expected by both admins and DevOps
- Features + properties drive requirements to *operate and use* an edge infra.
- Requirements divided into 5 levels (from L1: easy to fulfill, to L5: complex)

Easiest aspects

More complex aspects

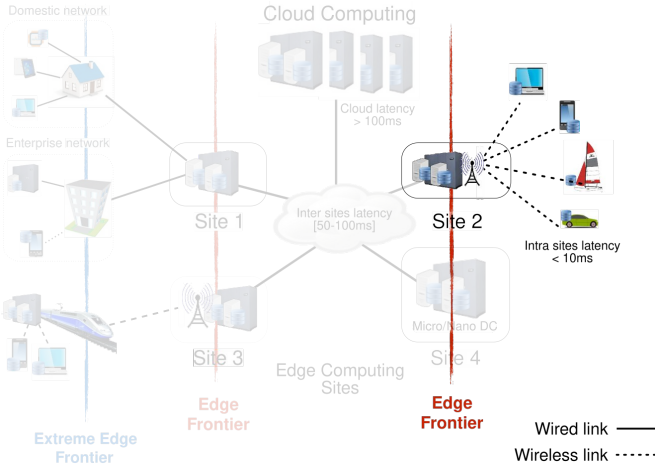
Levels	*Admin*	*DevOps*	*Both*
L1: Operate/use any site			
L2: Operate/use several sites			
L3: Robustness *w.r.t.* split brains			
L4: Multiple VIM environments			
L5: Multiple operators			

Level 1: Administer/use any Site

L1 contains the actions admins and DevOps expect to perform regarding any single reachable site:

- Examples:
  - Admins: manage (install/upgrade) manager services, users, quotas or images on a site
  - DevOps: provision on-demand resources on a single site
  - Both: collect metrics for supervision
- Ability to perform operations remotely (unmanned)
- Here, each site can be considered as an independent Cloud

  ⇒ OpenStack fulfill that level

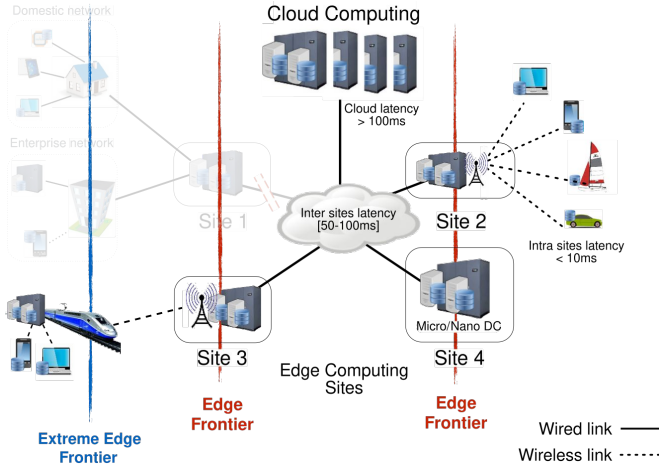


Levels	*Admin*	*DevOps*	*Both*
L1: Operate/use any site	Manage any site: install, upgrade site's services; manage users, flavors, quotas	Provision compute, storage, network resources on-demand on any site	Collect metrics and ensure security, integrity and resiliency for any site

Level 2: Administer/use several Sites

L2 = L1 on many sites (cross-sites collaboration)

- Implies ≠ kinds of collaboration between services:
  - **Intra-service** operations: same service on different sites
    - e.g. configuring users' access on a per-site basis (K2K)
    - e.g. list VMs on *Site 2* and *Site 3* (N2N)
  - **Inter-service** operations: different services on different sites
    - e.g. boot VM on *Site 2* with image from *Site 3* (N2G)
- Different ways to request collaboration:
  - **L2.1 Explicit manner**: "*openstack server list --site 2 --site 3*"
  - **L2.2 Implicit manner**: autonomous management/provisioning

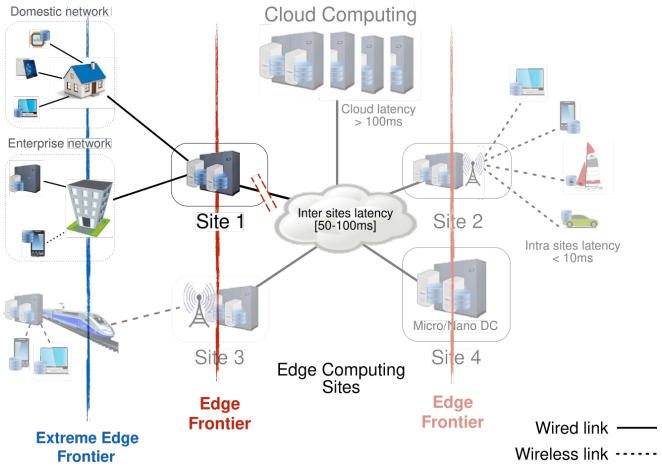


Levels	*Admin*	*DevOps*	*Both*
L1: Operate/use any site	Manage any site: install, upgrade site's services; manage users, flavors, quotas	Provision compute, storage, network resources on-demand on any site	Collect metrics and ensure security, integrity and resiliency for any site
→ L2: Operate/use several sites - L2.1 Explicit manner: - L2.2 Implicit manner:	Manage multiple sites simultaneously Manage a specific set of sites Cross-site autonomous management	Cross-site collaborative resources Provision on a specific set of sites Cross-site autonomous provisioning	L1 but over a set of sites Aggregated metrics from multiple sites and collaborative security mechanisms

Level 3: Robustness w.r.t. Network Partitioning

L3 = L1/L2 + ability to deal with network partitioning:

- Different considerations regarding the partition:
  - L1 for partitions with a single site
  - L1 + L2 for partitions with multiple sites
- Split-brains might have different impacts:
  - **L3.1 Application robustness**
  - **L3.2 Management service robustness**



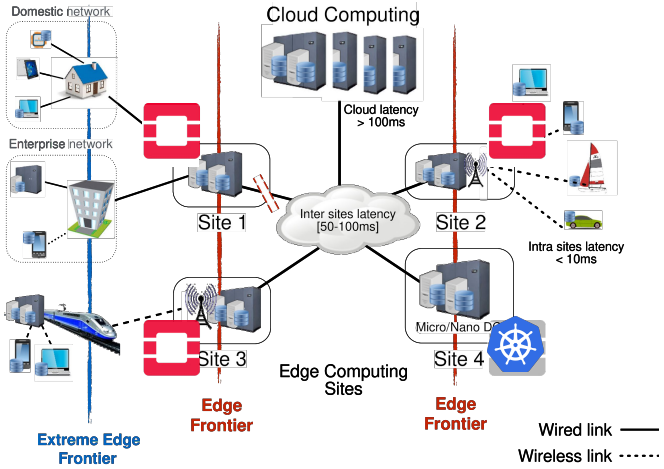
Levels	*Admin*	*DevOps*	*Both*
L2: Operate/use several sites	Manage multiple sites simultaneously	Cross-site collaborative resources	L1 but over a set of sites
- L2.1 Explicit manner:	Manage a specific set of sites	Provision on a specific set of sites	Aggregated metrics from multiple sites
- L2.2 Implicit manner:	Cross-site autonomous management	Cross-site autonomous provisioning	and collaborative security mechanisms
→ L3: Robustness *w.r.t.* split brains			L1 for an isolated site; L1 and L2 for
- L3.1 Application robustness:	——	Access reachable applications	isolated sets of sites
- L3.2 Management service robustness:	Manage reachable site(s)	Provision on reacheable site(s)	Support intermittent connectivity

Level 4: Multiple Cloud Environments

L4 = L3 with multiple VIM environments:

- **L.4.1 Different VIMs (APIs) versions:**
  - e.g. Collaboration between OpenStack Queen and Mitaka
- **L.4.2 Different VIM technologies:**
  - Might involve VIMs with different concepts
  - e.g. Collaboration OpenStack and Kubernetes

Requires the ability to discover sites' capabilities

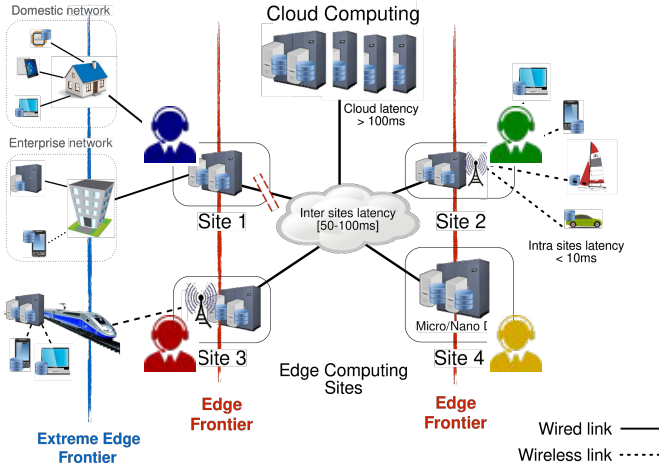


Levels	*Admin*	*DevOps*	*Both*
L3: Robustness *w.r.t.* split brains			L1 for an isolated site; L1 and L2 for isolated sets of sites
- L3.1 Application robustness:	——	Access reachable applications	
- L3.2 Management service robustness:	Manage reachable site(s)	Provision on reacheable site(s)	Support intermittent connectivity
→ L4: Multiple Cloud environments			L3 with different IaaS environments
- L4.1 Different IaaS versions:	Manage different IaaS versions	Provision on different IaaS versions	Discover sites' capabilities and compatibility
- L4.2 Different IaaS technologies:	Manage different IaaS technos	Provision on different IaaS technos	

Level 5: Multiple Operators

L5 = L4 with sites owned by multiple operators, similarly to 3GPP for mobile telecommunications

- No administration expectations
- Collaboration between operators to offer their resources to DevOps
- Collect relevant metrics to perform charging/billing



	Levels	*Admin*	*DevOps*	*Both*
	L3: Robustness *w.r.t.* split brains			L1 for an isolated site; L1 and L2 for isolated sets of sites
	- L3.1 Application robustness:	——	Access reachable applications	
	- L3.2 Management service robustness:	Manage reachable site(s)	Provision on reacheable site(s)	Support intermittent connectivity
→	L5: Multiple operators	——	Provision on one or many sites	L4 with multiple operators

Classification of Requirements for the Edge

Levels	*Admin*	*DevOps*	*Both*
L1: Operate/use any site	Manage any site: install, upgrade site's services; manage users, flavors, quotas	Provision compute, storage, network resources on-demand on any site	Collect metrics and ensure security, integrity and resiliency for any site
L2: Operate/use several sites	Manage multiple sites simultaneously	Cross-site collaborative resources	L1 but over a set of sites
- L2.1 Explicit manner:	Manage a specific set of sites	Provision on a specific set of sites	Aggregated metrics from multiple sites
- L2.2 Implicit manner:	Cross-site autonomous management	Cross-site autonomous provisioning	and collaborative security mechanisms
L3: Robustness *w.r.t.* split brains			L1 for an isolated site; L1 and L2 for
- L3.1 Application robustness:	——	Access reachable applications	isolated sets of sites
- L3.2 Management service robustness:	Manage reachable site(s)	Provision on reacheable site(s)	Support intermittent connectivity
L4: Multiple Cloud environments			L3 with different IaaS environments
- L4.1 Different IaaS versions:	Manage different IaaS versions	Provision on different IaaS versions	Discover sites' capabilities and
- L4.2 Different IaaS technologies:	Manage different IaaS technos	Provision on different IaaS technos	compatibility
L5: Multiple operators	——	Provision on one or many sites	L4 with multiple operators

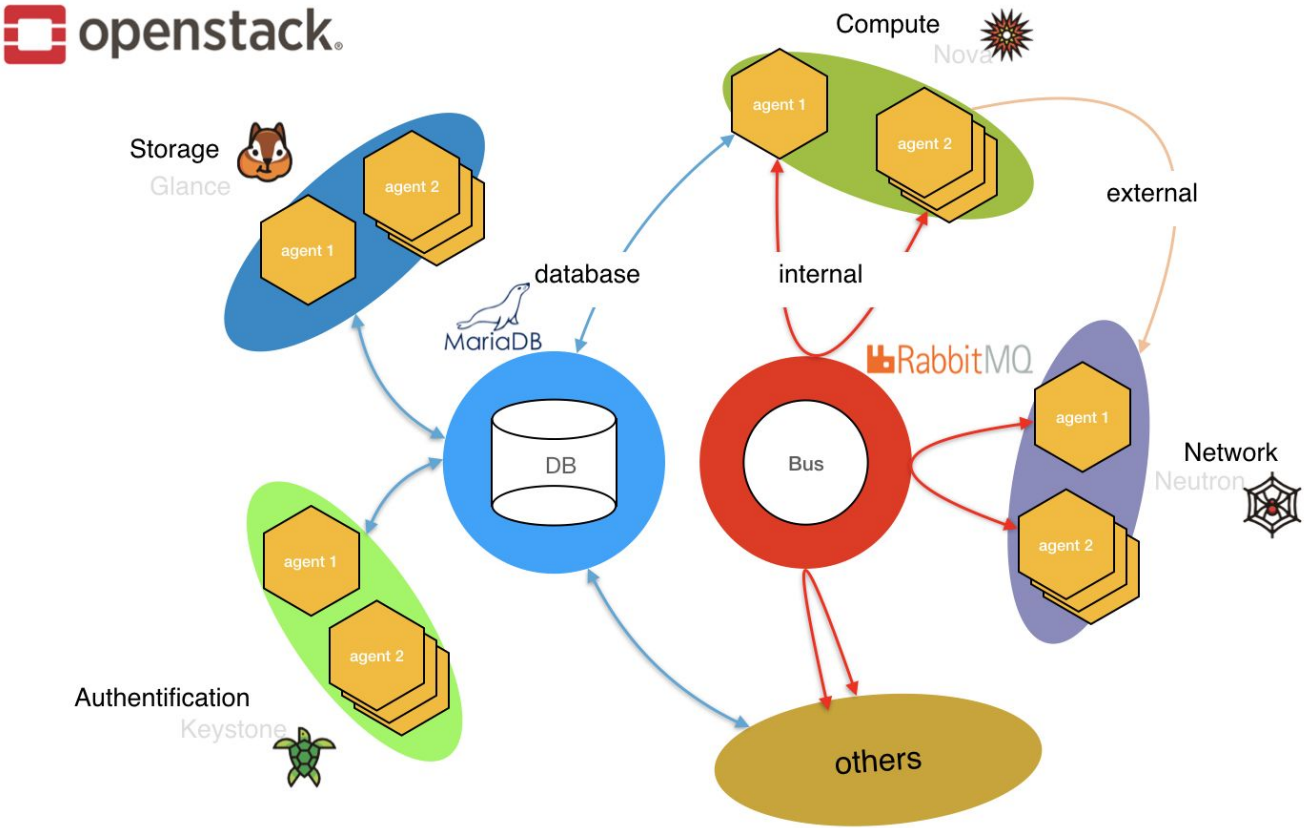
Table 1: Classification of the requirements to administrate and use edge computing infrastructures in 5 levels.

Regarding our use-case, these requirements give significant insights on how to design and implement an edge resource manager

Is OpenStack able to fulfill these requirement levels?

OpenStack at the Edge

OpenStack in one slide



credits: M. Simonin - IPL Discovery

Investigate OpenStack in Edge

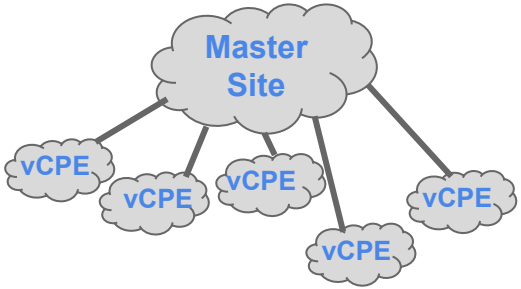
Several deployment possibilities:

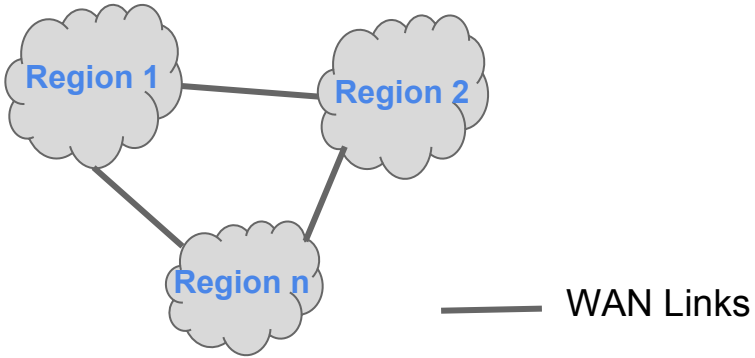
1. Control services deployed at one site, many compute nodes on remote sites

2. Segregation technics
   - Regions
   - Cells (nova related)

3. Broker approach: Multiple OpenStack Clouds managed by an external service that builds the global view (Tricircle)

4. Distribute the Database and message bus





Need a Sandbox to Conduct Various Performance Analyses

Experimental eNvironment for OpenStack (EnOS)

EnOS: Experimental Env. for OpenStack

- Motivation: conducting performance analyses
  - **In a scientific and reproducible manner (automation)**
  - At small, large-scale and under different network topologies (traffic shaping)
  - Between different OpenStack releases and configurations
  - Ephemeral perf-oriented deployments: **not for production**
- Built on top of OpenStack/Kolla-Ansible:
  - Ability to highly customize OpenStack deployment and service settings
  - Containerized approach brings flexible deployment capability
  - Real OpenStack deployment for realistic performance evaluation (compared to `devstack`)
- Workflow
  - `$ enos deploy`
  - `$ enos bench`
  - `$ enos backup`
  - `$ enos destroy`

EnOS deploy – Resource/Topology Description

```
$ cat ./basic.yml
resources:
  clusterA:
    control: 1
    network: 1
  clusterB:
    compute: 50

$ enos deploy -f basic.yml
```

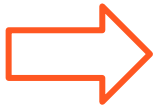
```
$ cat ./advanced.yml
resources:
  clusterA:
    control: 1
    network: 1
    nova-conductor: 5
  clusterB:
    compute: 50

$ enos deploy -f advanced.yml
```

```
$ cat ./network-topo.yml
resources:
  grp1:
    clusterA:
      control: 1
      network: 1
      nova-conductor: 5
  grp2:
    clusterB:
      compute: 50

network_constraints:
  - src: grp1
    dst: grp2
    delay: 100ms
    rate: 10Gbit
    loss: 0%
    symmetric: true

$ enos deploy -f network-topo.yml
```

EnOS deploy: Under the hood

```yaml
resources:
  grp1:
    clusterA:
      control: 1
      network: 1
  grp2:
    clusterB:
      compute: 50

network_constraints:
  delay: 100ms
  rate: 10Gbit
  loss: 0%
```

```
$ enos deploy
```



1. Provider gets 2 nodes on `clusterA`, 50 nodes on `clusterB` and returns node's IP addresses
2. EnOS provisions nodes with Docker daemon (Kolla dependencies)
3. EnOS installs OpenStack using Kolla
4. EnOS sets up bare necessities (flavors, cirros image, router, ...)
5. EnOS applies network constraints between `grp1` and `grp2` using tc

---

- Provider to get testbed resources
  - Resources: anything running a Docker daemon and EnOS can SSH to + some IPs
  - Existing providers: Vagrant (VBox/Libvirt), Grid'5000, Chameleon, OpenStack
  - ~500 LoC each
- Kolla to deploy OpenStack over testbed resources
- `tc` to apply network constraints

EnOS bench

- Benchmarks description

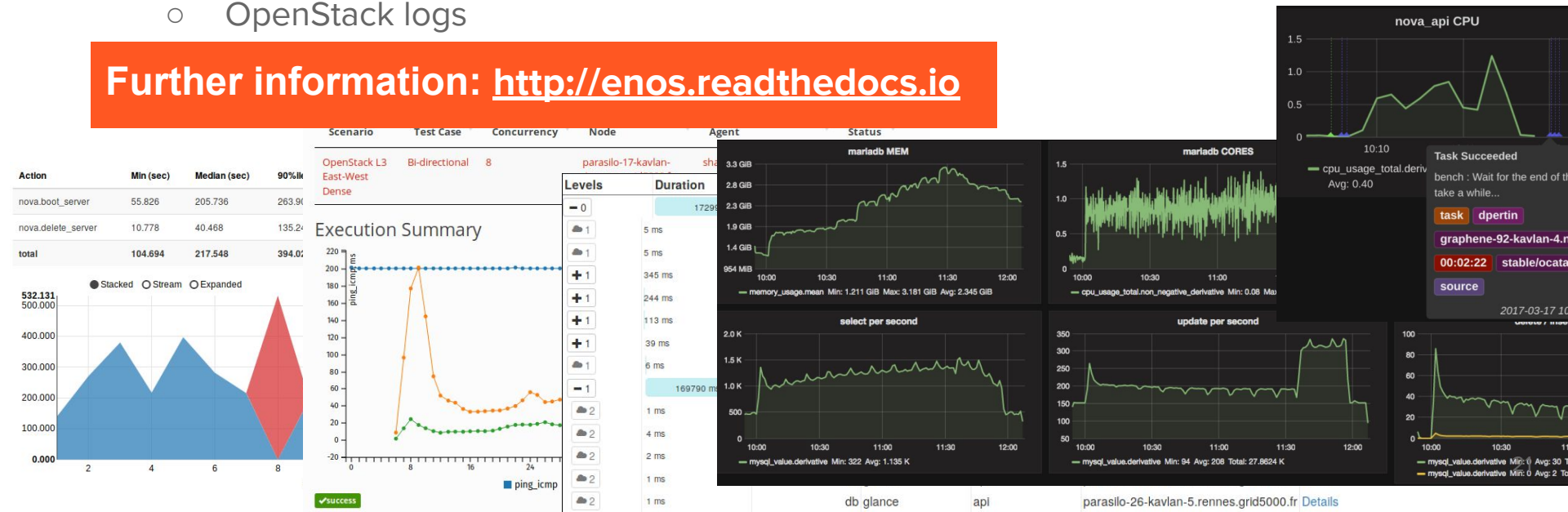
```
$ cat ./run.yml

rally:
  args:
    concurrency: 5
    times: 100
  scenarios:
    - name: boot and list servers
      file: nova-boot-list-cc.yml
      osprofiler: true
    - …
shaker: ...

$ enos bench --workload=run.yml
```

- Under the hood
  - Rally: control plane benchmark
  - Shaker: data plane benchmark
  - OSProfiler: code profiling
  - Monitoring stack: cAdvisor/Collectd to collect CPU/RAM/Network consumption per service/node/cluster

EnOS backup

- enos backup produces a tarball with
  - Rally/Shaker reports
  - OSProfiler traces
  - InfluxDB database with cAdvisor/Collectd measures
  - OpenStack logs

Further information: http://enos.readthedocs.io



Evaluation with EnOS

- Case study
  - Chasing 1000 nodes scale
    - https://docs.openstack.org/developer/performance-docs/test_plans/1000_nodes/plan.html
    - https://www.openstack.org/videos/barcelona-2016/chasing-1000-nodes-scale
  - OpenStack WANwide
    - Study network latency/throughput impacts on functional behavior and performance degradations
    - https://www.openstack.org/videos/boston-2017/toward-fog-edge-and-nfv-deployments-evaluating-openstack-wanwide
  - Massively Distributed RPCs (https://github.com/msimonin/ombt-orchestrator)
    - Study bus decentralization across WAN (focus on scalability and locality)
    - https://docs.openstack.org/performance-docs/latest/test_plans/massively_distribute_rpc/plan.html
    - [partial] https://www.openstack.org/videos/vancouver-2018/openstack-internal-messaging-at-the-edge-in-depth-evaluation
  - Keystone in the context of Massively distributed clouds (https://github.com/Marie-Donnie/juice)
    - Study Geo-Distributed (cockroachDB) database for Keystone
    - [partial] https://www.openstack.org/videos/vancouver-2018/keystone-in-the-context-of-fogedge-massively-distributed-clouds

Thanks!