par-tec
beyond the IT domain

PYCON OTTO
FIRENZE
6-9 APRILE 2017

# Just one Shade of OpenStack

## Cloud applications are simple!

**headquarters / operating unit**
via campanini 6
20124 milano
tel: +39 02/66.732.1 – fax: +39 02/66.732.300

**operating unit**
p.zza san benedetto da norcia 33
00040 pomezia (rm)
tel: +39 06/9826.9600 – fax: +39 06/9826.9680

vat number: 12938200156
c.c.i.a.a. milano n.1599095
incorporation number 12938200156
capital stock € 2.418.433,00

# Who? What? Why?

Roberto Polli - Solutions Architect @ par-tec.it. Loves writing in C, Java and Python. RHC{E,VA}, MySQL|MongoDB Certified DBA.

Par-Tec – Proud sponsor of this talk ;)  provides expertise in IT Infrastructure & Services and Business Intelligence solutions + Vertical Applications for the financial market. Contributes to various FLOSS.

Manage OpenStack with python Shade library to simplify automation.

# Agenda

- Intro to OpenStack
- The day I met Shade (Quickstart)
- Building the cloud
- Shade in Ansible Modules
- A contribution story

# OpenStack is Programmable Infrastructure

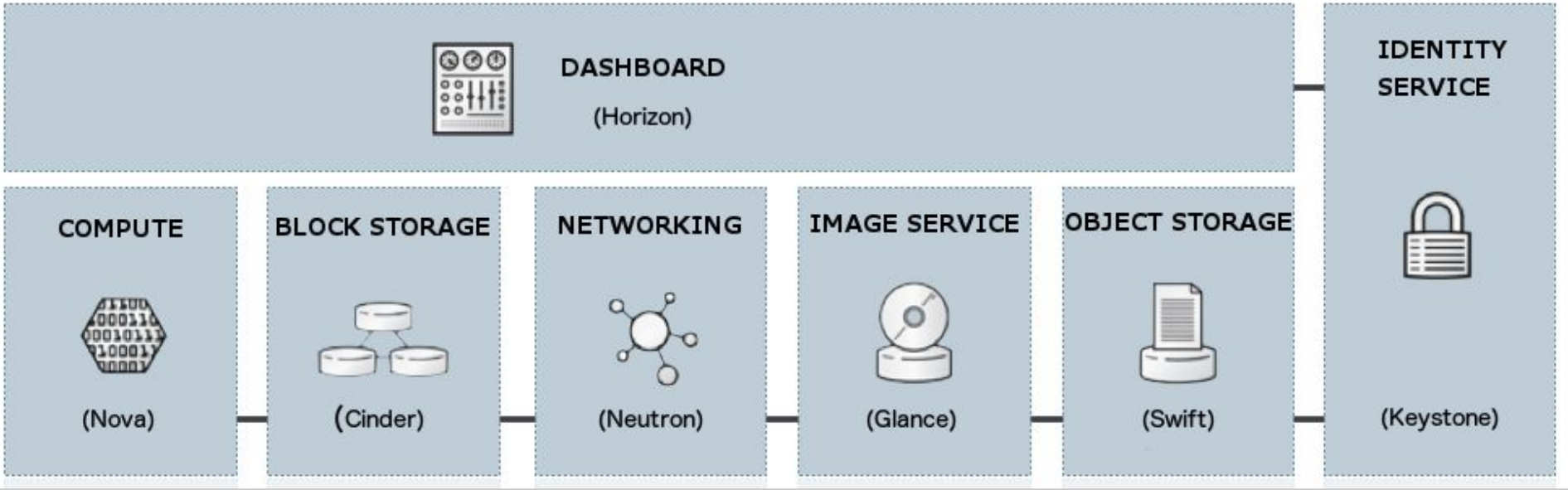
Programmable Infrastructure made open

- Amazon Web Services like (IaaS)
- Deploy Virtual {Machines,Network,...} on the internet
- REST interface

API Specs + python implementation

Industry backed (Red Hat, Rackspace, HP, Cisco, Oracle, ..)

# OpenStack Architecture



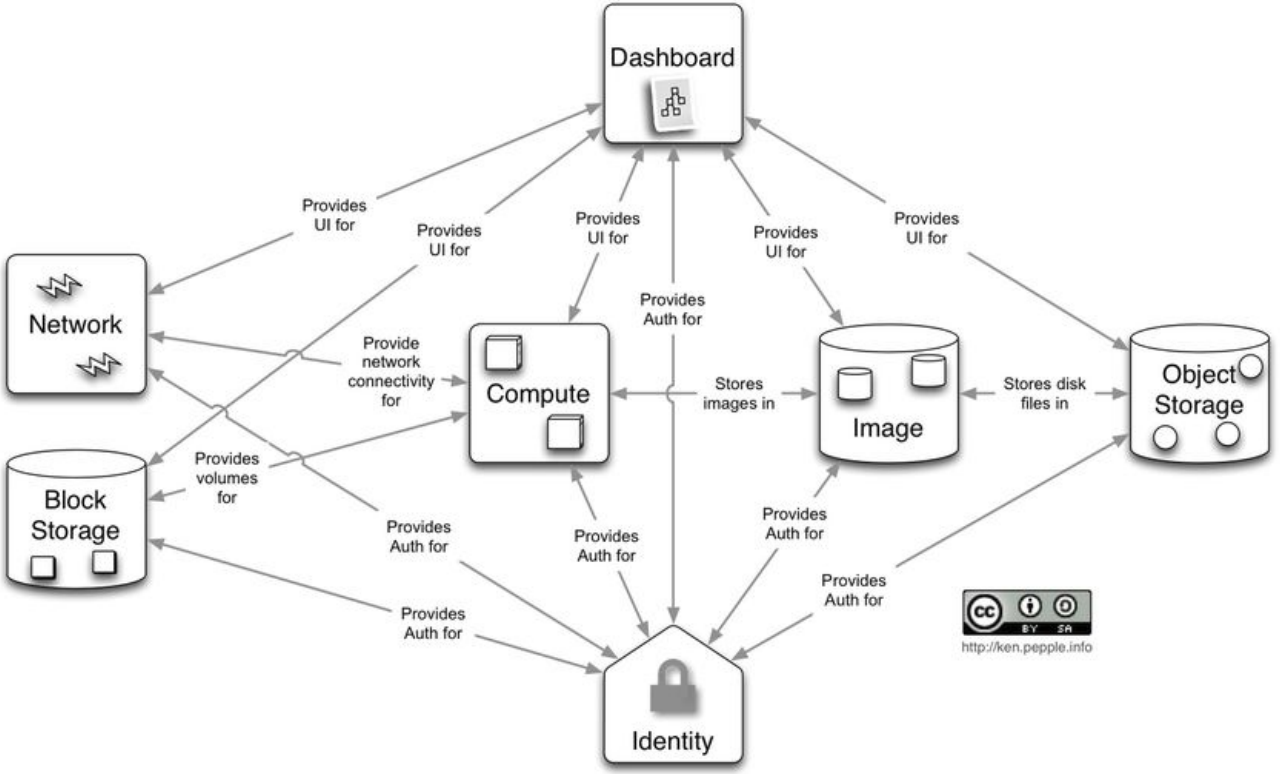
# OpenStack Workflows

Web Interface

API Endpoint

Queues

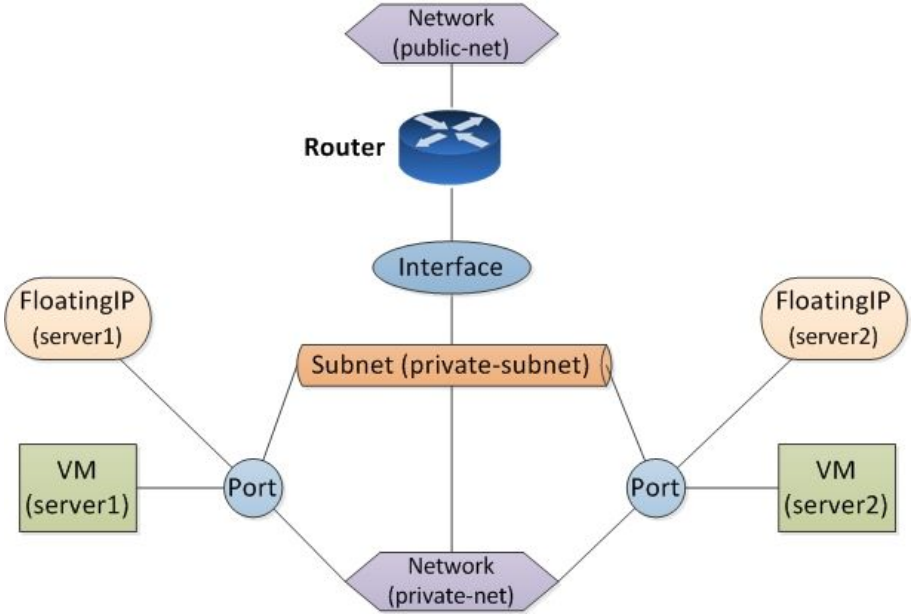
Datastores



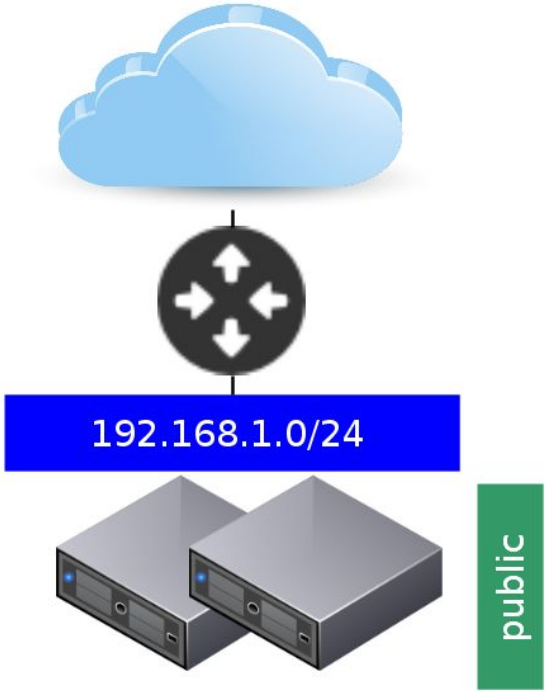
# OpenStack but... what's a Stack?

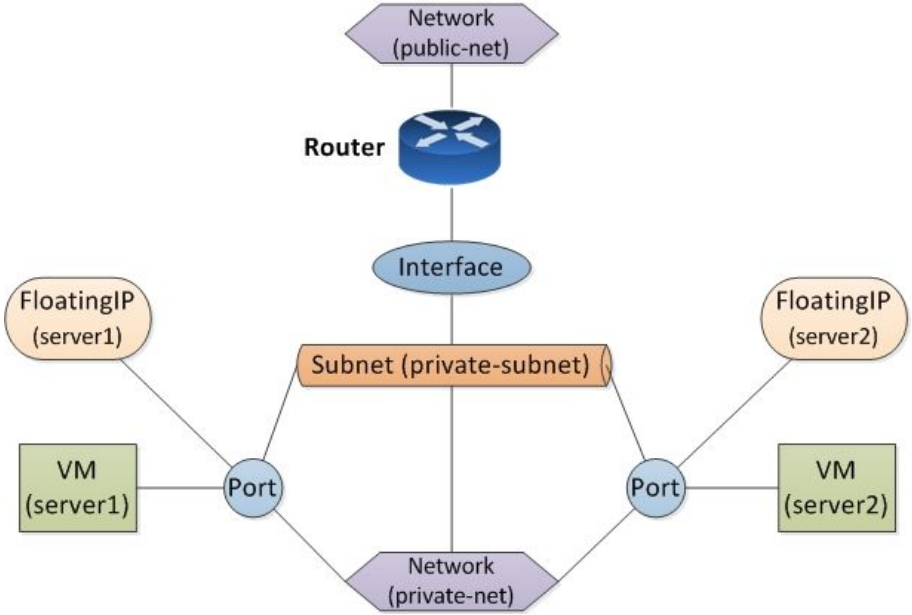
Virtualized infrastructure made of multiple components:

- load balancers
- web servers
- networks and routers
- datastores
- ...



# OpenStack but... what's a Stack?



# Deploy OpenStack Applications - CLIent API

```
$ openstack server create --image centos-7 --flavor m1.tiny web-1

$ openstack network create my-dmz

$ openstack subnet create --network my-dmz ...

$ openstack volume create --size 50 my-disk

$ openstack server add volume web-1 my-disk
```

https://docs.openstack.org/user-guide/cli-cheat-sheet.html

# Deploy OpenStack Applications: Heat Templates

Heat is the Orchestration component.

- group resources into a HOT
- Heat Orchestration Template
- reuse them

Stacks can be updated and deleted, Heat takes care of them.

You can compose templates specifying a filename or an URI

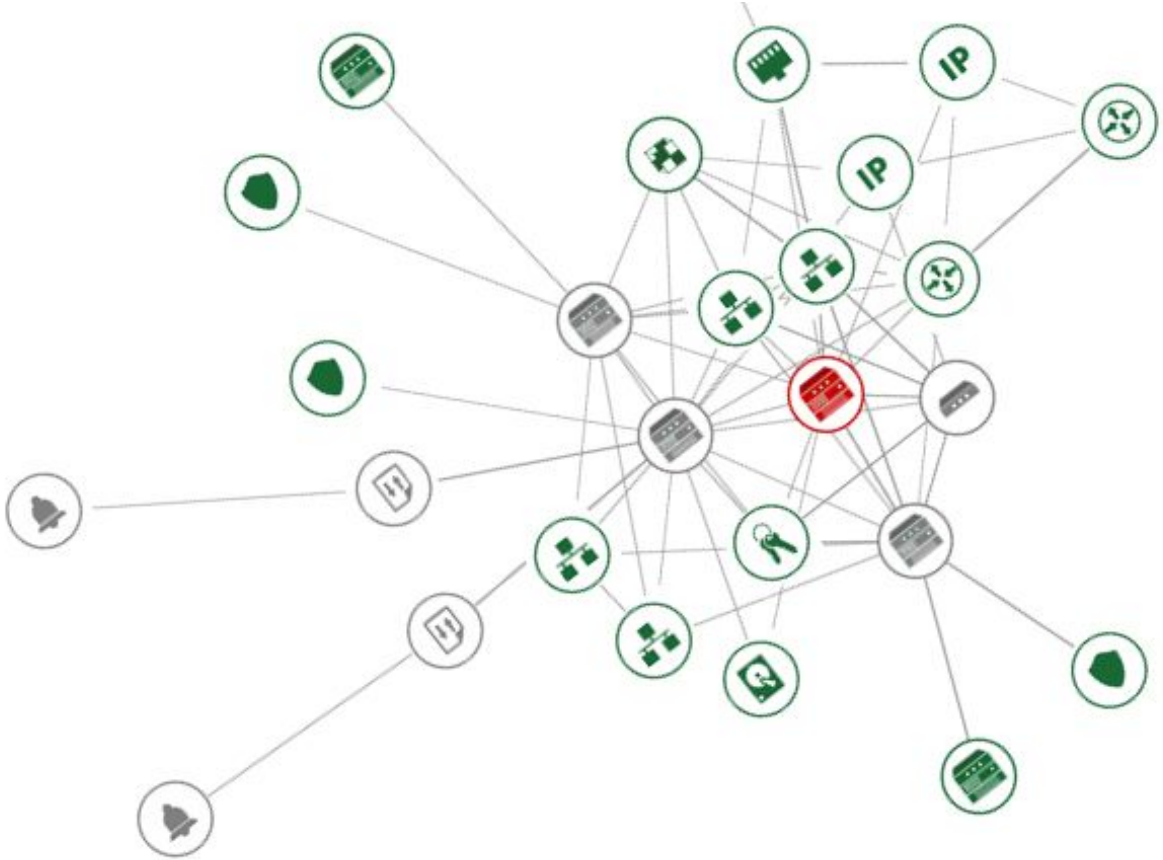
**Stacks are *persistent***

```
heat_template_version: 2016-04-08

description: >
  A Template is a yaml file describing resources
  and getting parameters in input.

parameters:
 service_name:
   type: string
 ...
resources:
 web-1:
   type: OS::Nova::Server
   properties:
     name: { get_param: service_name }
     ...
 network:
   type: https://git.it/network_template.yaml
   properties:
     ...
```

# Deploy OpenStack Applications: Heat Templates



# Deploy OpenStack Applications: Ansible

OpenStack Modules

- more flexible than HOT
- removing a stack won't remove created resources
- requires an external orchestrator
- can invoke heat (eg. for testing HOTs)

HOT Quota is planned only in Ocata.
Older versions can't do it

```
- name: Yes, you can
  hosts: localhost
  ...
  tasks:
  - name: Create and configure
    os_project:
       state: present
       name: a_project
       description: with Ansible
       domain: default
       enabled: true
       wait: yes
  -name: Create a user
   os_user: name="joe" default_project="a_project"

  -name: Manage ACL
   os_user_role:
     user: joe
     role: admin
     project: a_project
```

# Meet Shade

I was working with various OpenStack projects when at EP16 in Bilbao I followed a training on Cloud Applications

Goals:

- simplify openstack usage
- manage multiple openstack clouds
- more interaction

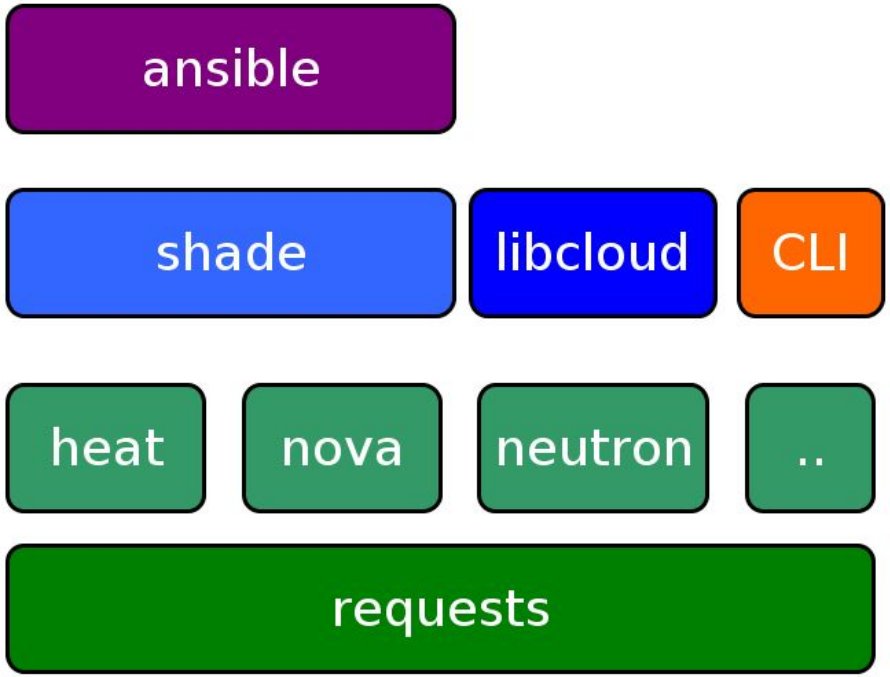
# OpenStack Applications

Use Shade directly:

- flexibility
- custom workflow

Apache libcloud:

- multi-cloud
- more generic



# Shade Quickstart - 1/3

```
$ pip install shade        # Install Shade

$ vim clouds.yml      # Store credentials


$ ipython                  # Run (i)python
import shade

# Connect to the first cloud
mycloud = shade.operator_cloud(cloud='mycloud')

# List virtual servers
servers = mycloud.list_servers()

# List server names
[x.name for x in servers]
[
  u'os3-node1',
  u'os3-master0',
  u'os3-master1',
  ...
]
```

```
# clouds.yml  is the credential file

identity_api_version: '3.0'

clouds:
 # Insert here your credentials.
 mycloud:
   verify: true  # or false
   auth:
     username: admin
     password: secret
     auth_url: 'https://mycloud.com/v3'
     project_name: 'admin'
     domain_id: 'default'
     interface: public
```

# Shade Quickstart - 2/3

```
import shade

# Connect to the first cloud
site_a = shade.operator_cloud(cloud='mycloud')

# Connect to the second cloud
site_b = shade.operator_cloud(cloud=rackspace)

get_servers=lambda site: {x.name: x
                    for x in site.list_servers() }
# List virtual servers
servers_a = get_servers(site_a)
servers_b = get_servers(site_b)

# Check servers not highly available.
set(servers_a.keys()) - set(servers_b.keys())

# Get doppelgangers ip ;)
for k,v in servers_a.items():
    v.doppelganger_ip = servers_b[k].interface_ip
```

```
# Add many clouds to your config
#
# Known cloud providers can be
# referenced via eg. "profile: rackspace"

identity_api_version: '3.0'
clouds:
 mycloud:
     ...

 rackspace:
   auth:
     cloud: rackspace
      project_id: 275610
     username: openstack
     password: xyzpdq!lazydog
     region_name: DFW,ORD,IAD
     interface: internal
```

# Shade Quickstart - 3/3

```
# Use logging for troubleshooting issues, eventually dumping to a file.
import shade
import logging
logging.basicConfig(level=logging.DEBUG, filename="client.log")

shade.simple_logging(debug=100, http_debug=0)

# Trace python-requests logging here or with http_debug=1
import httplib as http_client
http_client.HTTPConnection.debuglevel = 1
requests_log=logging.getLogger("requests.packages.urllib3")
requests_log.setLevel(logging.DEBUG)
requests_log.propagate = True


# Cloud configs are read with os-client-config
mycloud = shade.operator_cloud(cloud='mycloud')
```

# Some more Shade - 1/2

```
# You can access every entry as an
object
server.image
{u'id': u'42c26768-0edf-...'}

# or as a dict (eg for complex attrsa)
server['OS-SRV-USG:launched_at']
u'2017-02-18T09:42:19.000000'


# Shade retains custom properties
#  in the properties attribute
server.properties  # a dict!
{
  'OS-EXT-AZ:availability_zone':
u'nova',
  'OS-EXT-STS:power_state': 1,
 ...
}
```

```
# The model.rst document,
# describes shade entries

Server = dict(
  location=Location(),
  id=str(),
  name=str(),
  image=dict() or str(),
  flavor=dict(),
  volumes=list(),  # Volume
  interface_ip=str(),
  ...
)
```

# Some more Shade - 2/2

```
# Pandas creates wonderful reports
import pandas as pd

# Get your data
servers_a = cloud_a.list_servers()
servers_b = cloud_c.list_servers()
all_servers = servers_a + servers_b

# dump them into a pandas.DataFrame
df = pd.DataFrame(data=all_servers)
report = df[['name', 'status',
             'cloud', 'interface_ip']]

# then to Excel
writer = ExcelWriter('cloud.xlsx')
report.to_excel(writer,'servers')
writer.save()
```

```
#
# The report outcome!
#
           name  status  cloud      interface_ip
0     os3-node1  ACTIVE  mycloud
10.168.176.124
1   os3-master0  ACTIVE  mycloud
10.168.176.125
...
8   os3-master0  ACTIVE  rackspace
172.23.176.125
9   os3-master2  ACTIVE  rackspace
172.23.176.123
```

# Shade and Ansible

Provisioning projects  on a multi-datacenter OpenStack

Create projects with Heat:

- the project remains as a stack
- accidental removal risk
- can't manage quotas & co with HOT*
- update a quota may trigger a full stack-update

*"Projects represent the base unit of ownership in OpenStack, in that all resources in OpenStack should be owned by a specific project."*

docs.openstack.org

# Implement an Ansible module with Shade

Many policies for creating and updating OpenStack objects

Testing/checking differences between expected and actual settings

Trigger custom actions depending on os_project status

```
# developing the os_project_access module
# grant access on various resources to
# a given project

  -   name: ansible provisioning a project
      os_project_access:
        ...
        resource_type: nova_flavor
        resource_name: x1.medium
        project_id: 0000-ffff-0000
  -   name: with iterations
      os_project_access:
        resource_type: cinder_volume_type
        state: "{{item.state}}"
        resource_name: "{{item.name}}"
        project_id: 000-fff-000
      with_items:
        - {name: "sas", state: present }
        - {name: "ssd", state: absent }
```

# Writing Ansible Modules

PLEASE, DO NOT

write complex modules

accessing directly {nova,cinder,..}-api

# Writing Ansible modules

```
# Implement a general workflow!

resource = _get_resource(resource_name_or_id)

allowed_projects = _get_resource_access(resource_id) # get acls

if state == 'present':
    _add_resource_access(resource_id, target_project_id)

elif state == 'absent' and target_project_id in allowed_projects:
    _remove_resource_access(resource_id, target_project_id)
```

# Contributing to Shade

```
# Implement missing methods in shade
if resource_type == 'nova_flavor':
    _get_resource           = cloud.get_flavor
    _list_resource_access   = cloud.list_flavor_access     # <- write this!
    _add_resource_access    = cloud.add_flavor_access      # <- write this!
    _remove_resource_access = cloud.remove_flavor_access   # <- write this!
    ...
elif resource_type == 'your_resource_type':
    def __not_general_enough_to_be_in_shade(resource_id):  # <- if patch refused ;)
        ...
    _list_resource_access = __not_general_enough_to_be_in_shade

else: raise NotImplementedError("Resource %r not implemented" % resource_type)
```

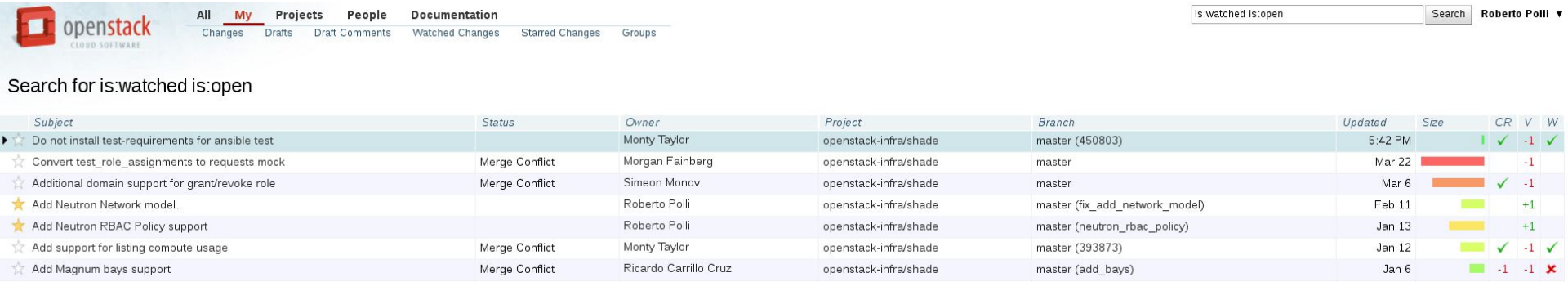
# Contributing to Shade

- Join #openstack-shade on irc
- Sign Up https://launchpad.net/openstack
- Install git-review
- Write functional tests
- Install devstack and test there before submitting for review

# Contributing to Shade

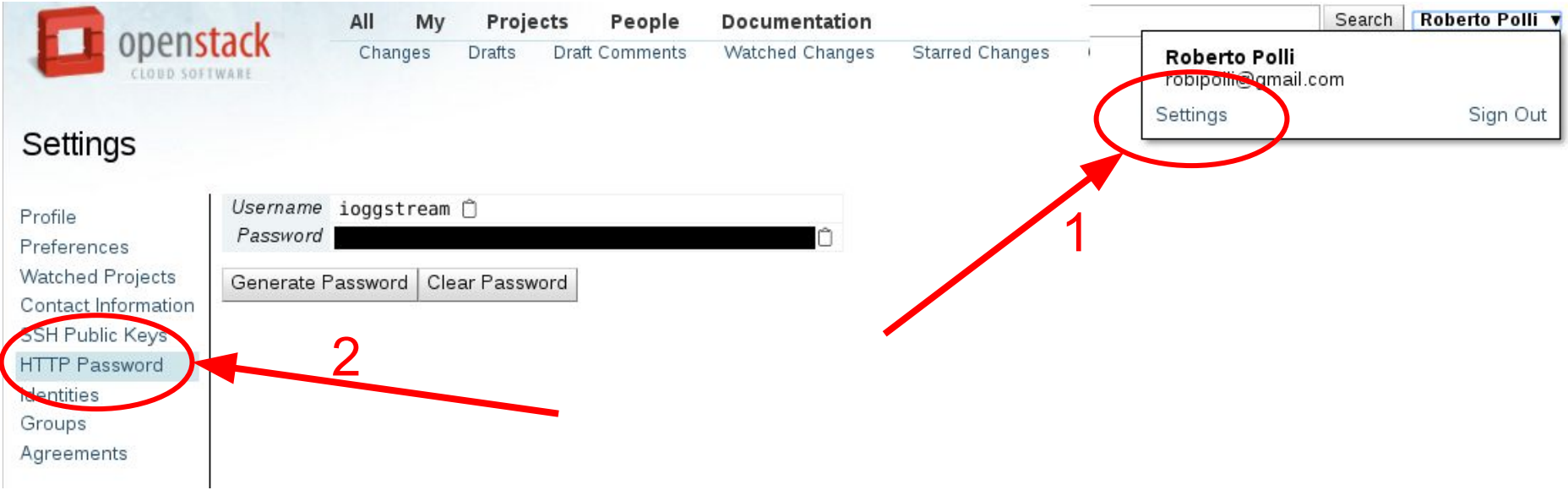
Check your progresses and give feedback on other's patches

https://review.openstack.org/#/q/is:watched is:open



# The Hardest Part

Where is my git-review password? No, it's not your account one ;)



# Questions?

Thank You

roberto.polli@par-tec.it