# Testing at scale

Leveraging py.test to test over multiple clouds

Hello!

# Carine-Belle Feder

@cbelle1234

Software Developer @ Oracle-Ravello

# Agenda

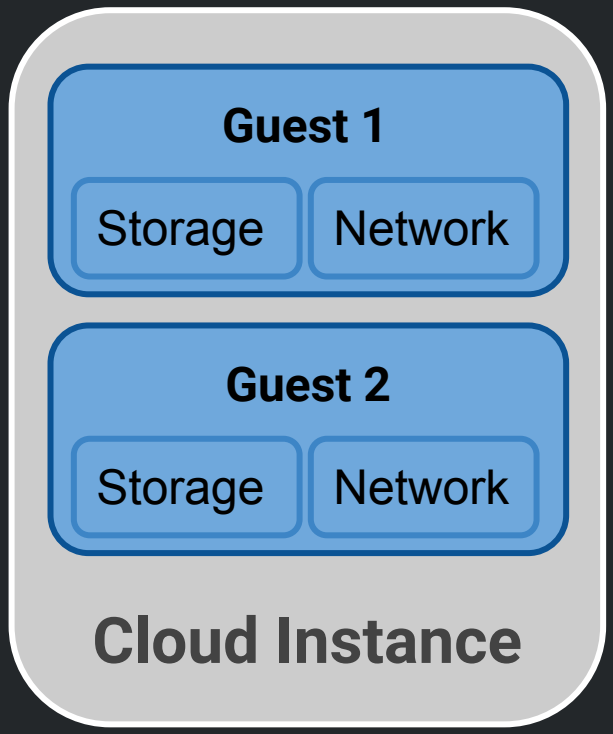
- Oracle-Ravello Product Introduction
- Our Testing Requirements
  - Why Py.Test is magical
  - Running on multiple clouds
- Thoughts and conclusions

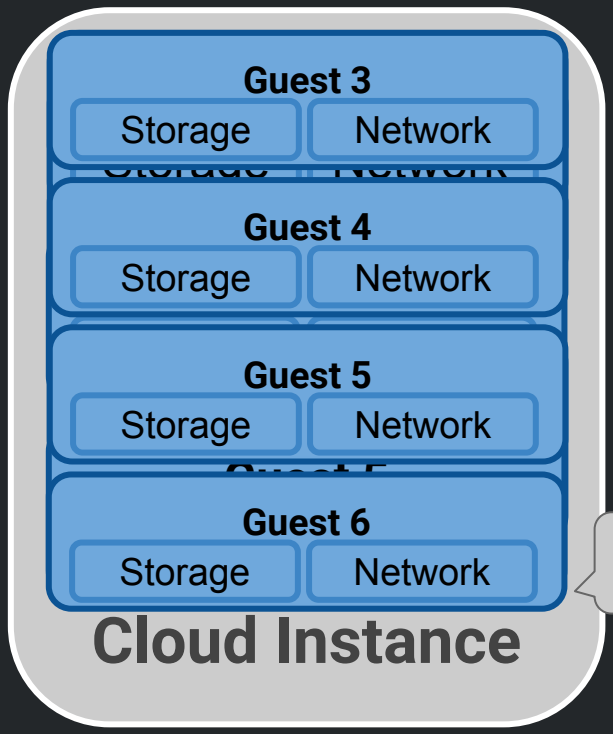

# Oracle-Ravello

- Founded in 2011 by Qumranet founders, Acquired by Oracle in 2016
- Virtual Cloud Provider
  - We run on top of other clouds, such as OCI (Oracle Cloud Infrastructure), AWS, Google
- Allows data-centers “Lift and Shift”
  - Migrate on-premise data-center workloads to the public cloud without changing machines\network configuration

# Oracle-Ravello's Product

- What exactly do we need to test?
  - Let's be more specific



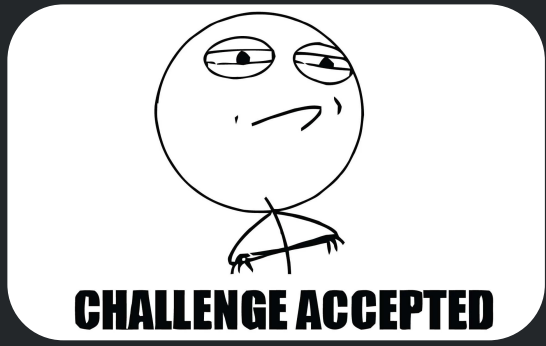



# Oracle-Ravello's Product

- What exactly do we need to test?
  - Let's be more specific


# Testing Our Product

- Requirements
  - ☐ We have A LOT of scenarios\use cases
  - ☐ Each test needs a different cloud instance-type
  - ☐ Each test has to run over multiple clouds
  - ☐ Automated - By groups\suits



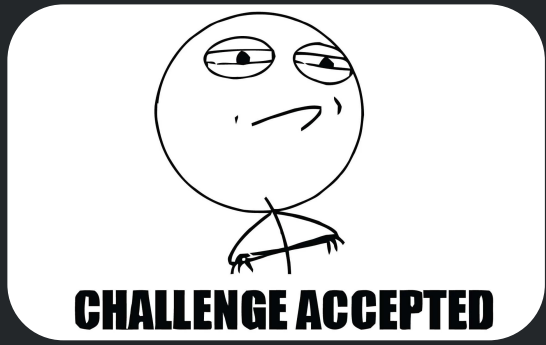
# pytest

```
cbelle@73839c654269:~/hvx-host-75/rtest$ VERSION=2.15.8 py.test -sv tests/matrix/test_operation_systems.py --collect-only
============================================================================================ test session starts ============
platform linux2 -- Python 2.7.6, pytest-3.5.1, py-1.5.3, pluggy-0.6.0 -- /usr/bin/python
cachedir: .pytest_cache
metadata: {'Python': '2.7.6', 'Platform': 'Linux-3.19.0-68-generic-x86_64-with-Ubuntu-14.04-trusty', 'Packages': {'py': '1.5
: '2.5.1', 'xdist': '1.22.2', 'celery': '4.1.0', 'flaky': '3.4.0', 'html': '1.17.0', 'timeout': '1.2.1', 'forked': '0.2', 'p
rootdir: /home/cbelle/hvx-host-75/rtest, inifile: pytest.ini
plugins: xdist-1.22.2, timeout-1.2.1, repeat-0.4.1, progress-1.2.1, metadata-1.7.0, html-1.17.0, forked-0.2, cov-2.5.1, flak
collected 22 items
<Module 'tests/matrix/test_operation_systems.py'>
  <Function 'test_operating_systems_boot[checkpoint_gateway-google_hvx]'>
  <Function 'test_operating_systems_boot[ubuntu-google_hvx]'>
  <Function 'test_operating_systems_boot[windows10-google_hvx]'>
  <Function 'test_operating_systems_boot[esx_6_default-google_hvx]'>
  <Function 'test_operating_systems_boot[centos_65_32bit-google_hvx]'>
  <Function 'test_operating_systems_boot[windows_vista-google_hvx]'>
  <Function 'test_operating_systems_boot[oracle_linux_7-google_hvx]'>
  <Function 'test_operating_systems_boot[centos7-google_hvx]'>
  <Function 'test_operating_systems_boot[windows7-google_hvx]'>
  <Function 'test_operating_systems_boot[juniper_firefly-google_hvx]'>
  <Function 'test_operating_systems_boot[ubuntu16-google_hvx]'>
  <Function 'test_operating_systems_boot[windows_server_2012-google_hvx]'>
  <Function 'test_operating_systems_boot[windows_server_2016-google_hvx]'>
  <Function 'test_operating_systems_boot[windows7_32bit-google_hvx]'>
  <Function 'test_operating_systems_boot[esx_55_64bit-google_hvx]'>
  <Function 'test_operating_systems_boot[windows_server_2008_64bit-google_hvx]'>
  <Function 'test_operating_systems_boot[checkpoint_sms-google_hvx]'>
  <Function 'test_operating_systems_boot[esx_65_64bit-google_hvx]'>
  <Function 'test_operating_systems_boot[esx_6_64bit-google_hvx]'>
  <Function 'test_operating_systems_boot[windows_81_32bit-google_hvx]'>
  <Function 'test_operating_systems_boot[windows_81_64bit-google_hvx]'>
  <Function 'test_operating_systems_boot[windows_server_2003-google_hvx]'>

===Flaky Test Report===
```

# Testing Our Product

- Requirements
  - ☒ We have A LOT of scenarios\use cases
  - ☐ Each test needs a different cloud instance-type
  - ☐ Each test has to run over multiple clouds
  - ☒ Automated - By groups\suits



# A Single Test Flow

Specify Test Scenario

```
 1 from rtest.defaults.guests import ubuntu
 2 from rtest.setup.guest_utils import sanity
 3
 4 import pytest
 5
 6 @pytest.mark.commit_tests
 7 def test_guest_started(setup, cfg):
 8     cfg.guests[0] = ubuntu()
 9     setup.update(cfg)
10     sanity.verify_guests_alive(setup)
~
~
~
~
~
```

# A Single Test Flow

Specify Test Scenario

Fill the Missing Data

```
cfg.guests[0].toggles[1].value = 'ubuntu'
cfg.guests[0].username = 'ubuntu'
cfg.guests[0].verification_method = 'network'
cfg.guests[0].verification_port = 22
cfg.guests[0].vnc_port = 0
cfg.hosts[0].allowing_overcommit = True
cfg.hosts[0].cloud = 'google'
cfg.hosts[0].host_volumes[0].datasize = 4958925
cfg.hosts[0].host_volumes[0].id = 0
cfg.hosts[0].hypervisor = 'hvx'
cfg.hosts[0].id = 0
cfg.hosts[0].ips = 1
cfg.hosts[0].min_ram = 3072M
cfg.hosts[0].min_vcpus = 1
cfg.hosts[0].region = 'us-central1-b'
```

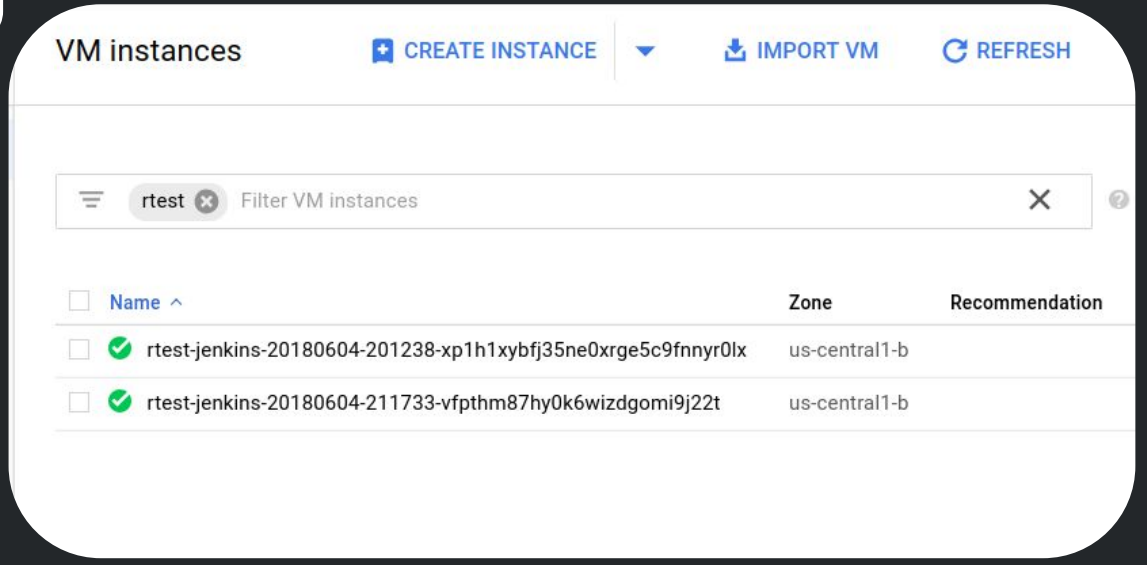
```
cfg.guests[0].mem = 2G
cfg.guests[0].name = 'ubuntu'
cfg.guests[0].network_compiler.dns_entries[0].id = 0
cfg.guests[0].network_compiler.dns_entries[0].name = 'ubuntu'
cfg.guests[0].network_compiler.dns_entries[0].ptr = False
cfg.guests[0].network_compiler.dns_entries[1].id = 1
cfg.guests[0].network_compiler.dns_entries[1].name = 'ubuntu'
cfg.guests[0].network_compiler.dns_entries[1].ptr = True
cfg.guests[0].network_compiler.nics[0].device = 'virtio'
cfg.guests[0].network_compiler.nics[0].ips[0].addr = '10.0.0.3'
cfg.guests[0].network_compiler.nics[0].ips[0].id = 0
cfg.guests[0].network_compiler.nics[0].ips[0].netmask = '255.255.0.0'
cfg.guests[0].network_compiler.nics[0].ips[0].public = 'port_fwd'
```

# A Single Test Flow

Specify Test Scenario

Fill the Missing Data

Setup Test Environment



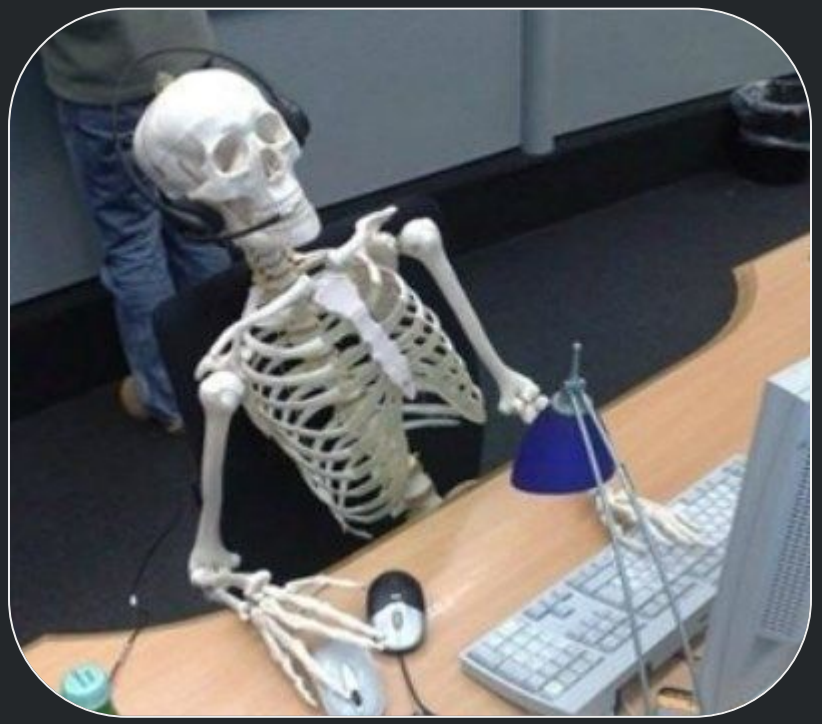
# A Single Test Flow

Specify Test Scenario

Fill the Missing Data

Setup Test Environment

Run Test on Instance

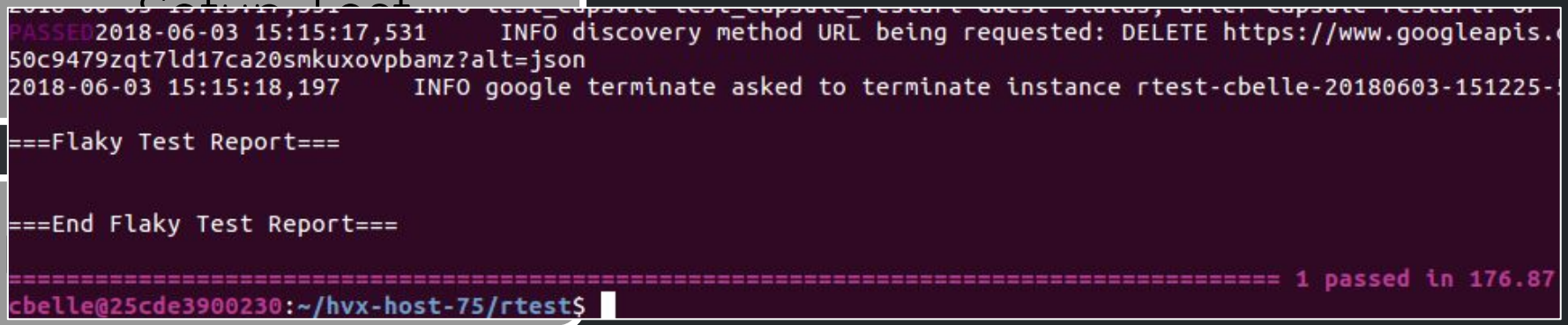


# A Single Test Flow

Specify Test Scenario

Fill the Missing Data

Setup Test



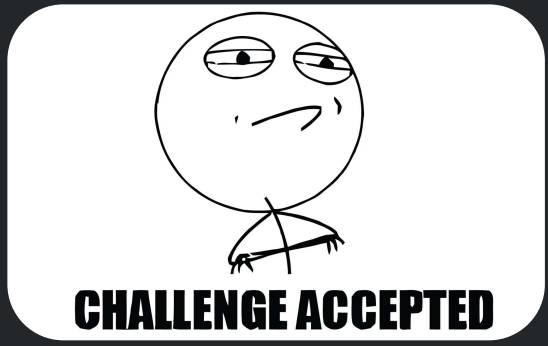

Teardown Test

# Choosing a Cloud Instance

- Every test has different hardware requirements
- Take one massive instance?
- OPTION 2 - Every test gets a fitting instance
  - Way more scalable
  - Easier to debug failed tests
  - Each test gets exactly the resources it needs

# Testing Our Product

- Requirements
  - ☒ We have A LOT of scenarios\use cases
  - ☒ Each test needs a fitting cloud instance-type
  - ☐ Each test has to run over multiple clouds
  - ☒ Automated - By groups\suits

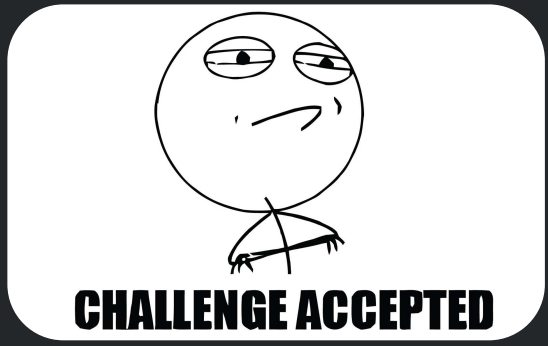


# Run on all the clouds

- The obvious problem:
  - Each cloud has a different SDK, regions, instance-types, credential, etc.
- SO... One API to rule them all?

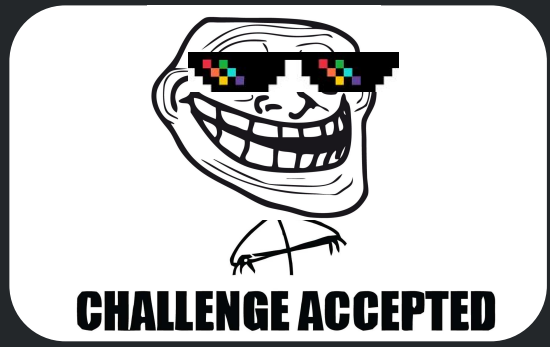
# Testing Our Product

- Requirements
  - ☒ We have A LOT of scenarios\use cases
  - ☒ Each needs a different cloud instance-type
  - ☐ Each has to run over multiple clouds
  - ☒ Automated - By groups\suits



# Testing Our Product

- Requirements
  - ☒ We have A LOT of scenarios\use cases
  - ☒ Each needs a different cloud instance-type
  - ☒ Each has to run over multiple clouds
  - ☒ Automated - By groups\suits



# Conclusions

## Pros

- Huge parameters matrix
- Super easy to create new tests
- Very easy to debug

## Cons

- Big infrastructures require a lot of maintenance
- Could be pretty expensive

# Questions?

# Thank you