

Upstream CI/CD tools for your production clouds

That can operate 1,000+ test cycles in
a day

Slide URL: <https://goo.gl/aZ3Tq3>

Masayuki Igawa

Core in Tempest,
openstack-health,
subunit2sql



Ghanshyam Mann

Core in Tempest



Ken'ichi Ohmichi

QA PTL, Core in Nova



Agenda

1. Introduction
2. Overview of QA tools
3. How to use QA tools for productions
4. Future works
5. Demo

Introduction

Why we are here

- We are working for OpenStack quality as upstream QA members
- There are many tools for testing/debugging for the quality

What will you learn

- How to keep/improve the quality at upstream development
- How to use these tools for production clouds

What's the QA in OpenStack?

Mission Statement of QA project

Develop, maintain, and initiate tools and plans to ensure the upstream stability and quality of OpenStack, and its release readiness at any point during the release cycle.



Overview of gate tests

Every proposed code needs to be passed via multiple gate tests

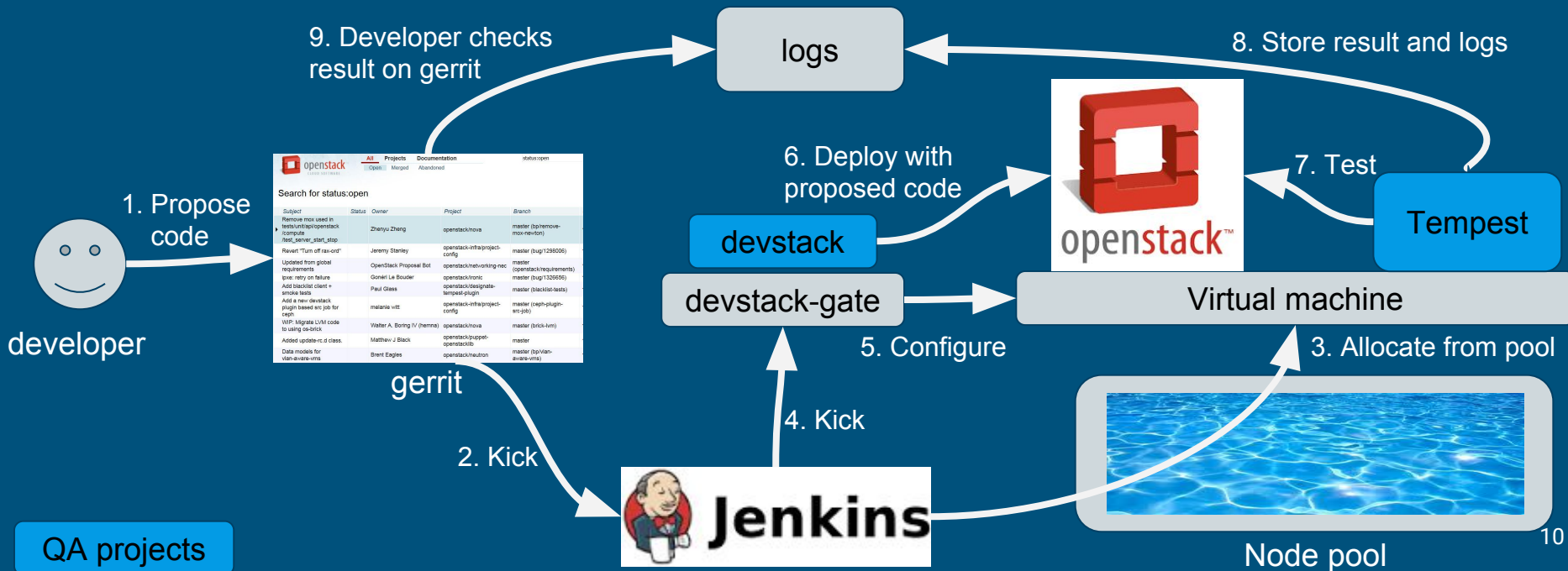
- pep8 (coding style)
- python 27/34 (unit test)
- gate-tempest-dsvm (integration test)

1,000+ gate test cycles run in most working days

- Each gate test cycle consists of 1,000+ tests
- That means 1,000,000+ tests run in a day

Overview of upstream CI/CD

Testing the proposed code on temp env with the CI/CD



Overview of QA tools

QA tools

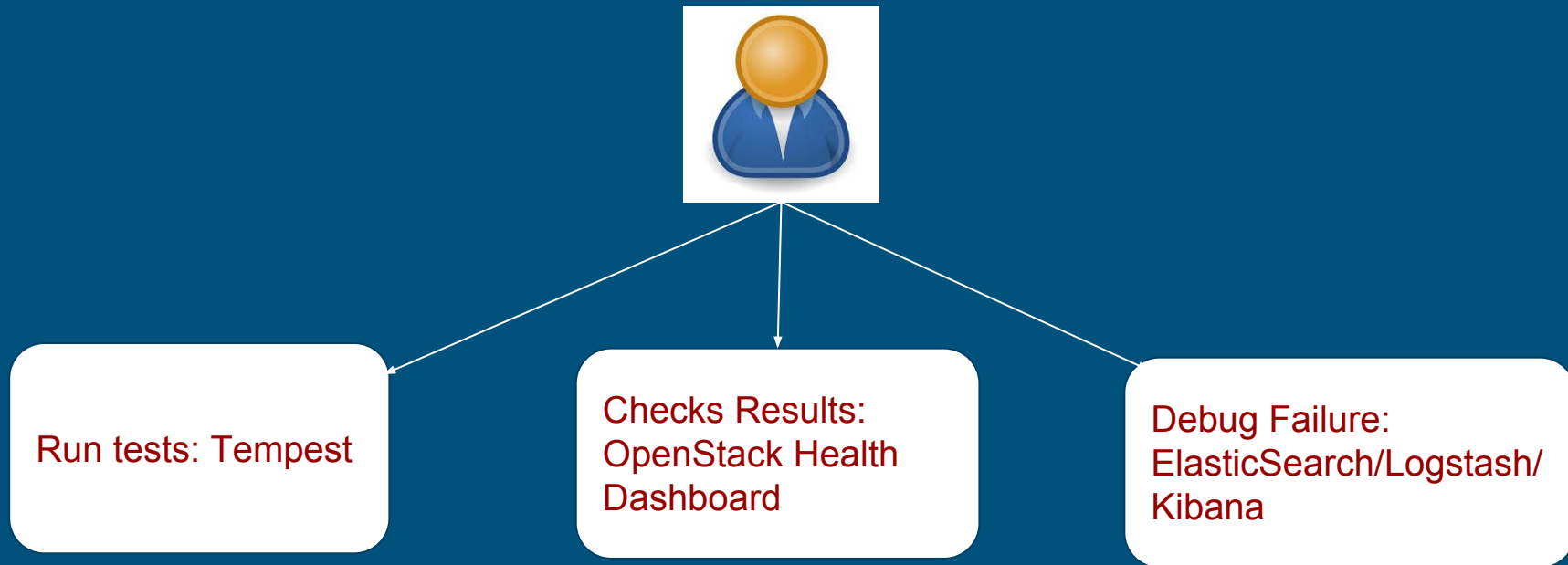
Testing: **Tempest**, os-testr, Grenade, Bashate, Hacking

OpenStack specific: **OpenStack-health**, **subunit2sql**, elastic-recheck, DevStack, stackviz

Generic: **ElasticSearch/Logstash/Kibana**, Gerrit, Jenkins, Graphite

<https://wiki.openstack.org/wiki/QA>

QA tools



Tempest - OpenStack Integration Test Suite

- This is a set of integration tests to be run against a live OpenStack cluster.
- Tempest is designed to be useful for a large number of different environments. This includes being useful for gating commits to OpenStack core projects and used for testing the production Cloud.

Tempest - Test types

- API tests
 - API tests are validation tests for the OpenStack APIs.
 - Make sure stability and backward compatibility of APIs.
- Scenario tests
 - Scenario tests are complex “through path” tests for OpenStack functionality.
 - Scenario tests interact with many services and tests cross projects features.
- Stress tests - Deprecated, will be removed in Newton
 - Stress tests are designed to stress an OpenStack environment by running a high workload against it and seeing what breaks.

Tempest - External Plugin

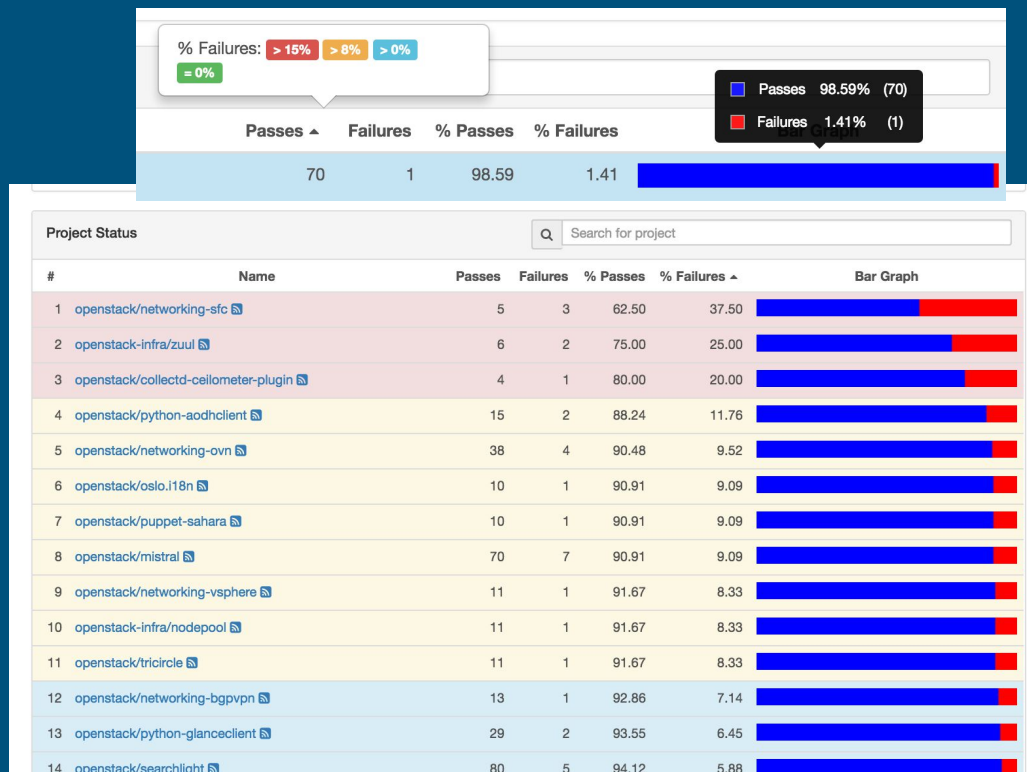
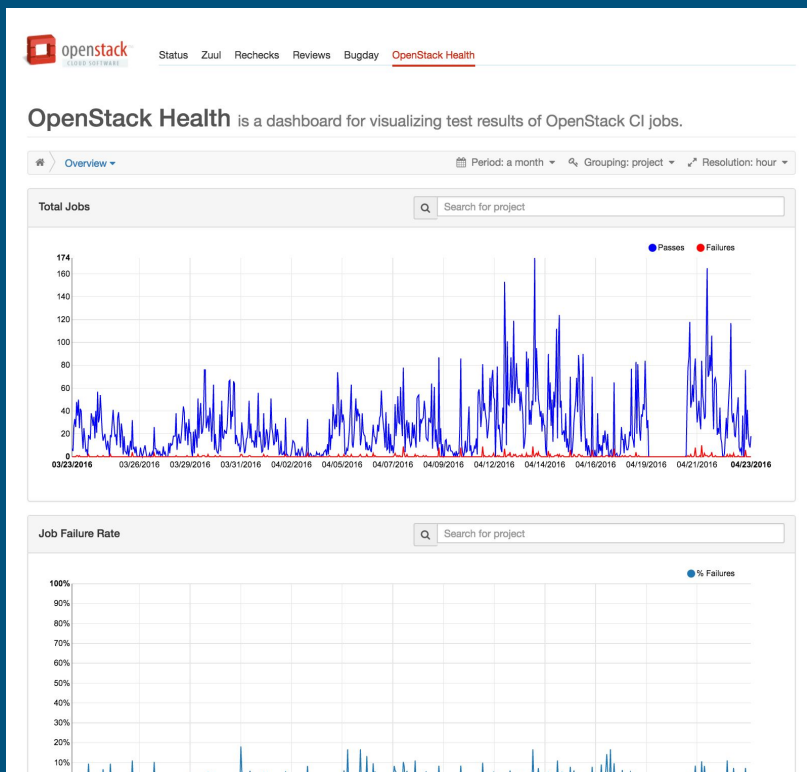
- Tempest provides an external test plugin interface to enables projects to implement tempest like tests and run those as part of tempest run.

~30 plugins



OpenStack-Health

is a dashboard for visualizing test results of OpenStack CI jobs.



What is the OpenStack-Health?

Why do we need this?

BigTents, numerous projects and test jobs, notify to ML doesn't work

Features

Provide a dashboard :Graphs, Colorful table for analyzing our "BIG-DATA"

Technically

OpenStack-Health

AngularJS 1(FrontEnd) + Flask(API server)



+

subunit2sql API

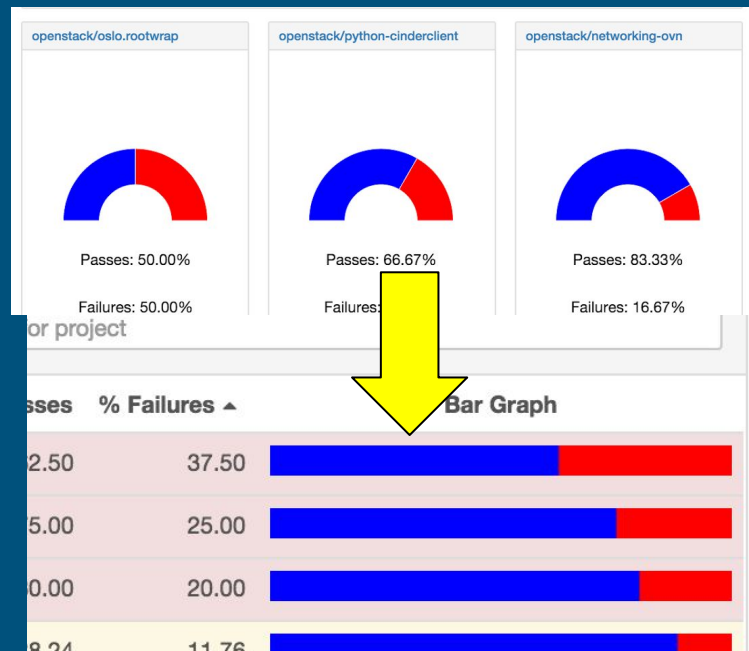
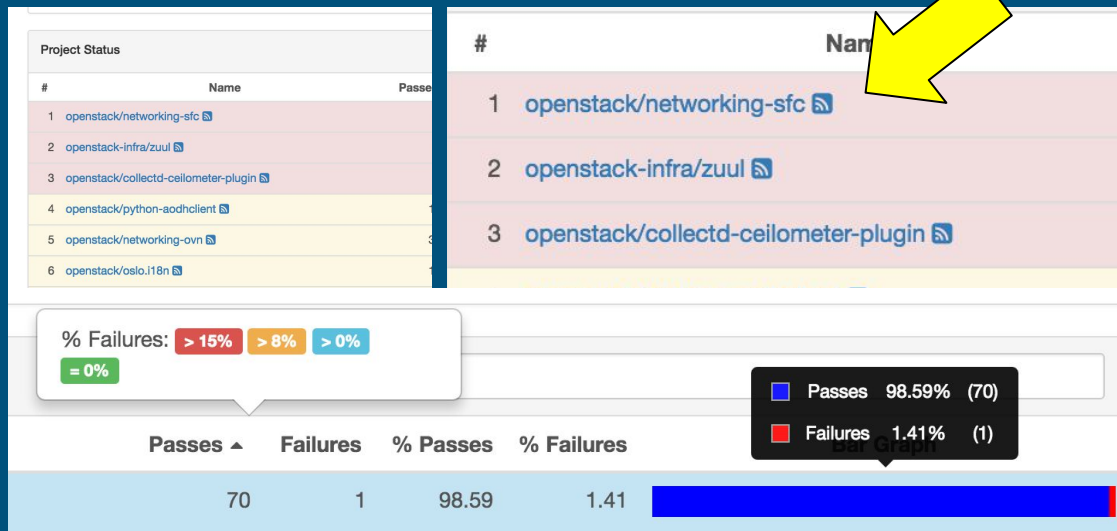
+

Database



OpenStack-Health: Recent Topics

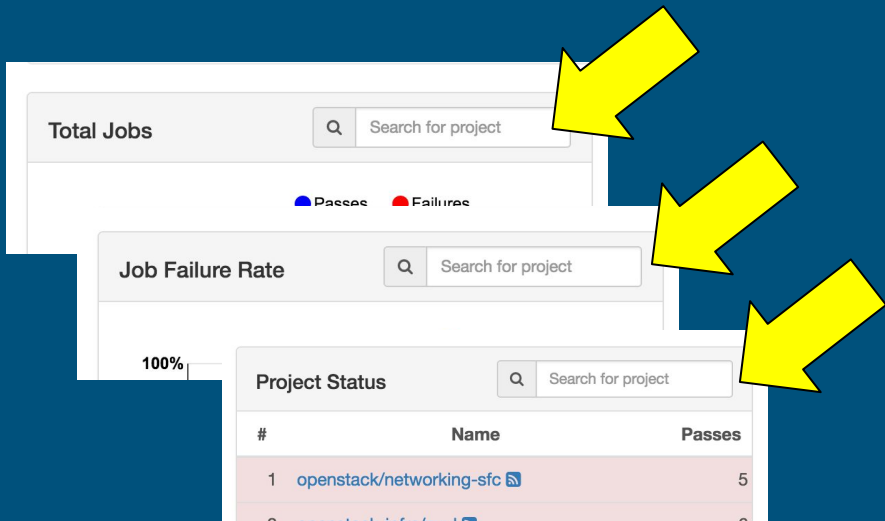
- Context color table: easy to understand the status
- RSS feed: Alternative notification
- Graph: “Gauge -> Bar” : space effective



OpenStack-Health: Recent Topics (cont.)

Recent Topics

- Regex Search filter: specify multiple projects (e.g. nova|glance|cinder...)
- Pagination



Tests Detail: nova

Overview ▾ Tests by Prefix: nova

Details for nova

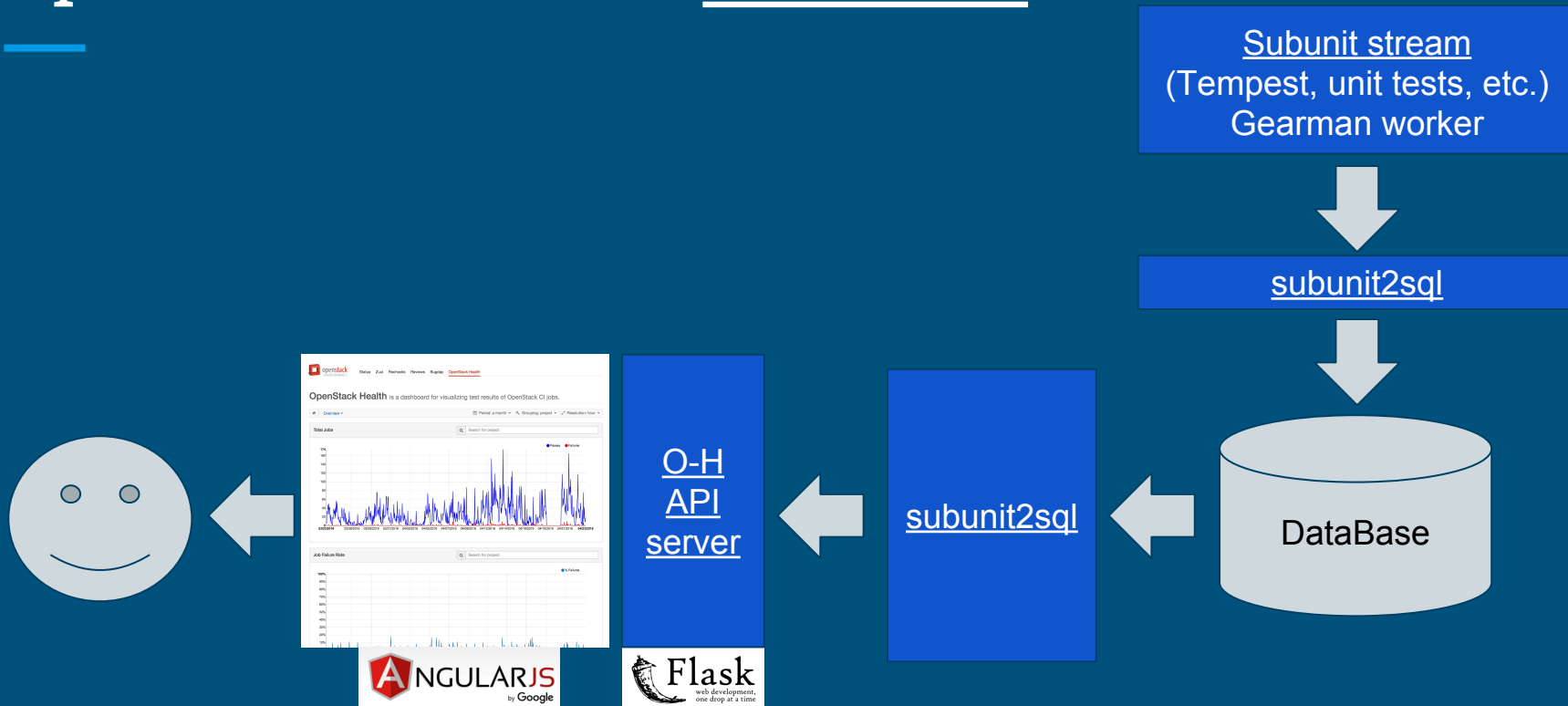
Limit: 1000 ▾ ← Previous Next →

Showing 22001 - 22899 of 22899 tests

Test ID

```
nova.tests.unit.virt.xenapi.test_vm_utils.UnplugVbdTestCase.test_uplug_r
```

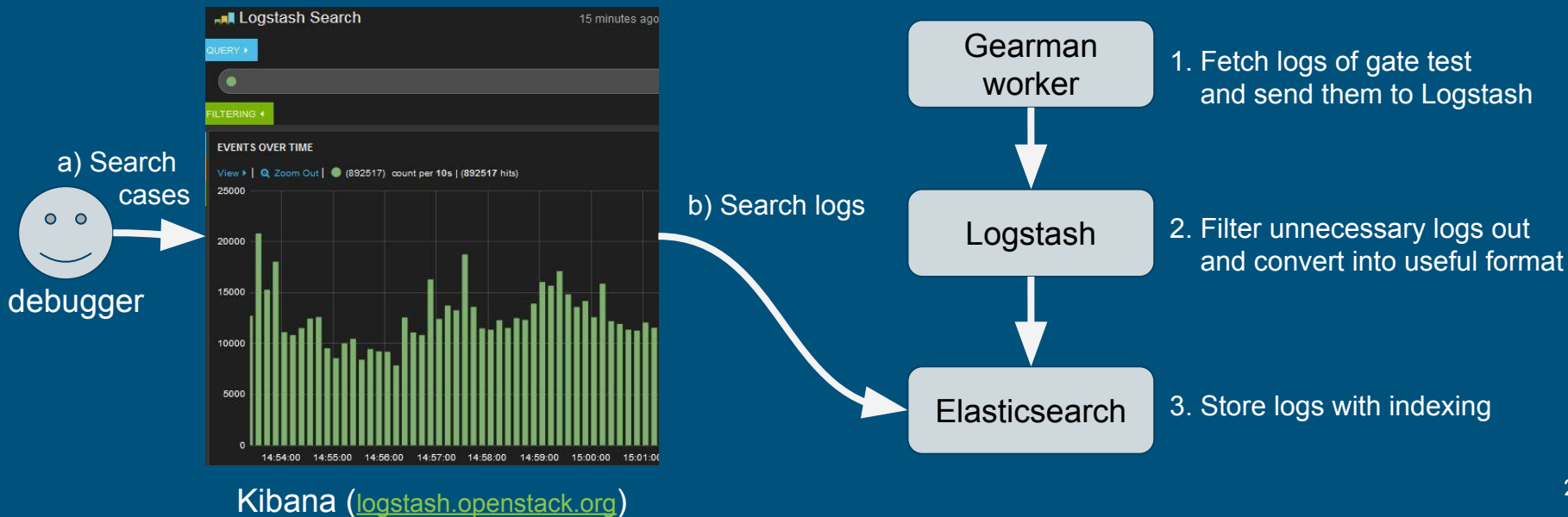
OpenStack-Health - Data flow



Kibana / Elasticsearch / Logstash

Log management system

We are using them for searching similar cases to debug gate problems



How to use QA tools for productions

How to use Tempest

2 Ways to Use Tempest

Tempest as a
System Installed
Program

Tempest Source
Code Repo

How to use Tempest(cont.)

Install Tempest as system program:

```
$ git clone http://git.openstack.org/openstack/tempest  
$ pip install tempest/
```

```
/etc/tempest
```

```
<prefix>/etc/tempest
```

How to use Tempest(cont.)

Set up separate working directory

```
$ tempest init Cloud_01_Test
```

```
etc  tempest_lock .testrepository  
logs .testr.conf
```

```
$ tempest init Cloud_02_Test
```

```
etc  tempest_lock .testrepository  
logs .testr.conf
```

How to use Tempest(cont.)

Tempest Source Code Repo:

```
$ git clone http://git.openstack.org/openstack/tempest  
$ cd tempest
```

```
data          LICENSE          REVIEWING.rst  setup.py      tox.ini  
doc           README.rst        run_tempest.sh tempest  
etc           releasenotes     run_tests.sh  test-requirements.txt  
HACKING.rst   requirements.txt setup.cfg      tools
```

How to use Tempest: Set up Config File

Tempest Configurations:

~80 config options

(Tempest & 6 Core Projects.)

+

Additional for other projects tests

(Tempest tree or in plugin).



How to use Tempest: Set up Config File

Tempest Configuration Sample File.

```
$ tox -e genconfig
```

OR

```
$ oslo-config-generator --config-file etc/config-generator.tempest.conf  
--output-file etc/tempest.conf
```

Best Reference for tempest.conf: Devstack generated one.

How to use Tempest: Set up Config File(cont.)

Auth Configuration:

- test_accounts_file
- Use_dynamic_credentials
 - Admin_username
 - Admin_project_name
 - Admin_password
 - Admin_domain_name
 - Create_isolated_networks
 - default_credentials_domain_name
- ...



How to use Tempest: Set up Config File(cont.)

Identity Configuration:

- uri
- uri_v3
- auth_version
- catalog_type
- region
- v2_admin_endpoint_type
- v2_public_endpoint_type
- v3_endpoint_type
- ...



How to use Tempest: Set up Config File

Example: Compute Configuration

- catalog_type
- region
- endpoint_type

- api_extensions
- image_ref
- flavor_ref
- resize
-

How to use Tempest:

All Set to Hit the Cloud



How to use Tempest: Run Tests

```
$ testr run <test path>
```

```
tempest.api.compute.servers. test_servers.serversTestJSON. test_create_server_with_admin_password
```

```
$ testr run <test idempotent_id>
```

```
b92d5ec7-b1dd-44a2-87e4-45e888c46ef0
```

```
$ testr run <test directory>
```

```
tempest.api.compute
```

```
$ testr run --parallel
```

OpenStack-Health - Basic installation

- Setup subunit2sql & Store your test data to your database

Pre condition: Setup your Database(e.g. dbname:subunit, username:subunit, pass:pass)

```
$ git clone git://git.openstack.org/openstack-infra/subunit2sql
$ subunit2sql-db-manage --database-connection mysql://subunit:pass@127.0.0.1/subunit upgrade head
$ subunit2sql --database-connection mysql://subunit:pass@127.0.0.1/subunit SUBUNIT-V2-FILE
```

Must Read
README.rst

NOTE:

* subunit2sql only takes a subunit v2 stream/file. So if you have only v1, you can convert it with using the subunit-1to2.

```
e.g. cat v1-file | subunit-1to2 | subunit2sql --database-connection mysql://subunit:pass@127.0.0.1/subunit
```

* You may need to integrate subunit2sql with CI tool like Jenkins.

OpenStack-Health - Basic installation

- Setup openstack-health

```
$ git clone git://git.openstack.org/openstack/openstack-health
```

API Server:

```
$ pip install -r requirements.txt (in virtualenv)
```

Frontend:

```
$ sudo apt-get install nodejs npm nodejs-legacy
$ sudo npm -g install npm@2
$ sudo npm -g config set prefix /usr/local
$ sudo npm -g install npm
$ sudo npm -g install gulp
$ npm install
```

- Run (in production)

API Server:

```
$ sudo etc/openstack-health-api.conf /etc/openstack-health.conf
$ uwsgi -s /tmp/uwsgi.sock --module openstack_health.api --callable app --http :5000
```

Frontend:

```
$ gulp prod; (DEPLOY "build/*" to your env such as Apache)
```

Must Read
README.rst

```
[default]
db_uri = mysql+pymysql://query:
query@logstash.openstack.
org/subunit2sql
ignored_run_metadata_keys =
    build_change
    build_master
    build_node
:
```

Future Works

Future work: Tempest

Resource config

Move Common Interfaces to tempest/lib as stable interface

Document for tempest plugin interface

Extend plugin interface

Cover API Microversion tests

...

Future work: OpenStack-Health

Improve Data limitation:

Detect/Store infra failure, check queue (not only gate/periodic)

UI improvement:

Improve messy/confusing UI

“I accessed the o-h site, but didn’t understand it. So I don’t use it anymore” :(

Framework update (AngularJS 1 -> 2)



Demo



Demo

- Tempest
- OpenStack-Health

Demo: How to find necessary patches

1. Run Tempest
2. If failure, find Tempest change log which has added failed test
3. Find “Related-Bug” or something on the commit message
4. Find the bug report on OpenStack LP
5. Find the patch of the component patch which closed the bug

E.g: `test_create_servers_on_different_hosts_with_list_of_servers`

The bug report: <https://bugs.launchpad.net/nova/+bug/1521928>

Useful Links

- [#openstack-qa on Freenode](#)
- openstack-dev@lists.openstack.org

Tempest

- <https://git.openstack.org/openstack/tempest>

OpenStack-Health

- <http://status.openstack.org/openstack-health/>
- <https://git.openstack.org/openstack-infra/subunit2sql>
- <https://git.openstack.org/openstack/openstack-health>
- <https://bugs.launchpad.net/openstack-health>
- OpenStack-Health Dashboard and Dealing with Data from the Gate: https://youtu.be/0__tD88oAlo