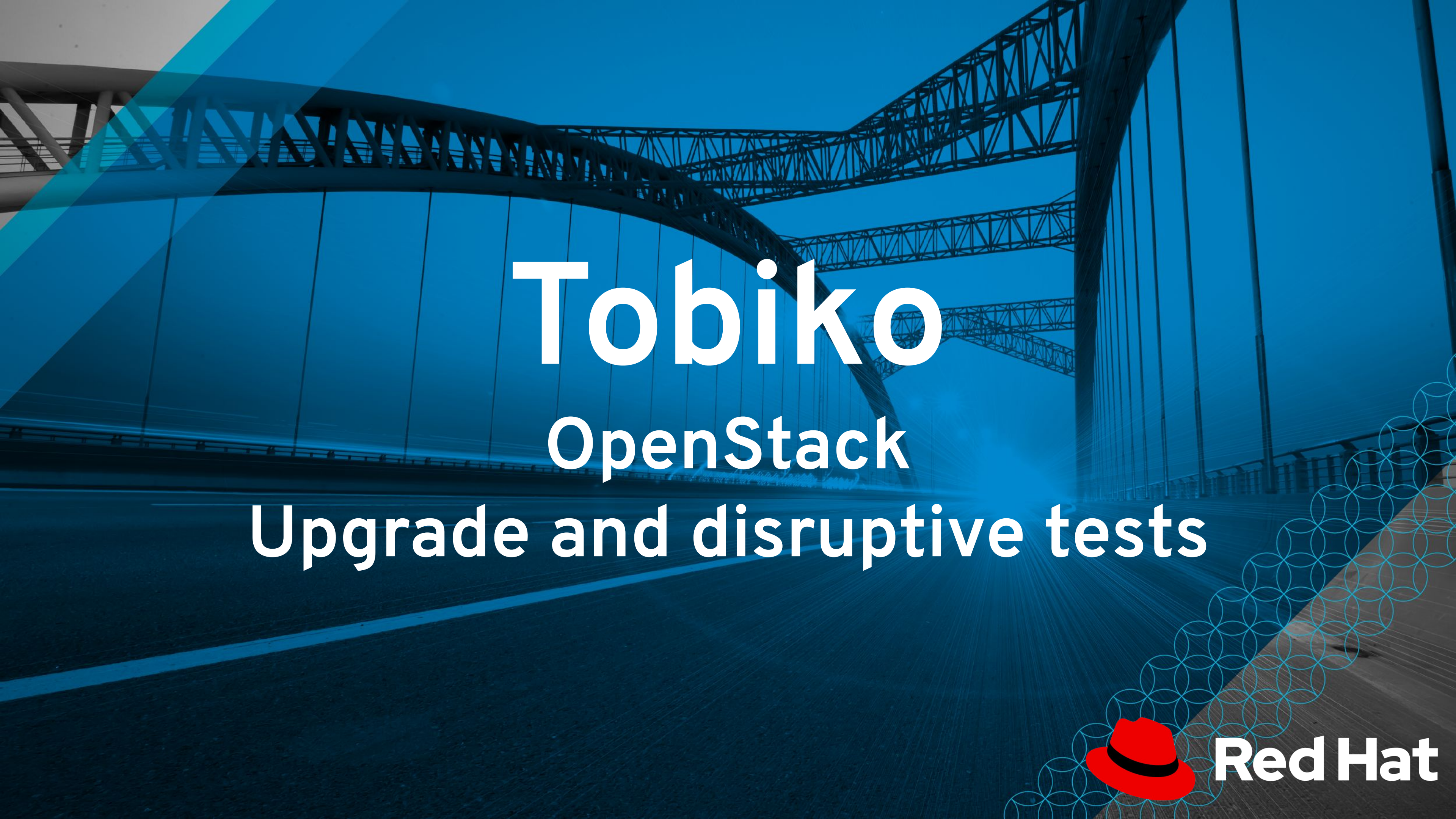

Tobiko

OpenStack

Upgrade and disruptive tests

Red Hat

Tobiko project

Test disruptive operations on OpenStack nodes

Hosted @ **https://opendev.org/x/tobiko**

- Python testing framework
- Python test cases
- Ansible roles to run CI testing workflow

Tobiko goals

- To test OpenStack update/upgrade
- To test OpenStack disruptive operations (IE. restart services, faults tests, etc.)
- To test OVN migration

Tobiko requirements

1. Must run 1 complete job in **less than 3 hours**
2. **Poor hardware** (1-3 nodes with 8GB RAM)
3. Disruptive **operations are slow**
4. **Can't run in parallel** disruptive operations
5. **Lot of things to verify** after disruptions
6. **Resources created before** disruptions must be **verified after** them

Simpler solution

For every resource scenario test all disruptive operations

We can delete resources after every test

Computational complexity is O(NxM)

Can't parallelize 'cause of disruptive operations

Tobiko workflow

1. Create **all** OpenStack resources
2. Run **all** disruptive operations
3. Verify **all** OpenStack resources

Keep all resources allocated

Computational complexity is O(N)

Tobiko workflow

1. **Create OpenStack resources**
   a. Run a set of Python tests to create and test OpenStack resources (images, VMs, networks, etc.)
   b. **Resources are "usually" left for later verification**
   c. **Resources are shared between tests**
   d. **Parallel execution is supported**

Tobiko workflow

1. Create OpenStack resources
2. **Run all disruptive operations**
   a. upgrade/update OpenStack services
   b. restart OpenStack services
   c. reboot OpenStack nodes

Parallel execution is not possible

Tobiko workflow

1. Create OpenStack resources
2. Run all disruptive operations
3. **Verify OpenStack resources**
   a. Check services are healthy
   b. Test resources created at first step
   c. **Parallel execution is supported**

Tobiko **workflow**

1. **tobiko-run** Ansible role implements it
2. **tobiko-run** role uses **Tox** to execute each workflow step
3. **Tox** runs Python test cases using **PyTest**

Tobiko resources

1. Pack resources into **Heat stacks** for easier management
2. **Reuse same stacks** between Python test cases
3. **Just in time stacks creation** (only what test cases need)
4. **Parallel stack creation** (handle concurrency issues)
5. Define resources stacks in Python **classes for easier customization**
6. Preconfigured resources **stack classes are part of the python library** (Nova servers, Neutron networks, etc.)

Tobiko resources

1. Download **Glance image files** from configurable URLs
2. **Reuse same images** between Python test cases
3. **Customize image files** using virt-customize
4. **Just in time lazy image creation** (only what test cases need)
5. Pre-configured images (CirrOS, Fedora, CentOS, Ubuntu, etc.)

Tobiko disruptions

1. Provides **OpenStack nodes topology** to tests
2. Tests can **SSH to nodes and VMs** to run commands
3. Run **local and remote commands** with the same API
4. Some common CLI **command wrappers**: (ping, ip, ps, curl, etc.)
5. Tests can **restart services** or reboot nodes

DevStack Tobiko plugin

1. Hosted @ https://opendev.org/x/devstack-plugin-tobiko
2. It sets up Tobiko test suite on any DevStack node
3. Customized Zuul jobs based on it:
   a. devstack-tobiko -> run all Tobiko suite
   b. devstack-tobiko-neutron -> tests Neutron
   c. devstack-tobiko-nova -> tests Nova
   d. ...

Grenade Tobiko plugin

1. **To be implemented!**
2. Hosted @ https://opendev.org/x/devstack-plugin-tobiko
3. **Workflow already implemented by Grenade**
    a. **Create Tobiko resources resources**
    b. Upgrade DevStack services
    c. **Verify Tobiko resources**
4. Tobiko-run ansible role replaced by Grenade hooks
5. New zuul jobs to be developed