

# HPC on OpenStack

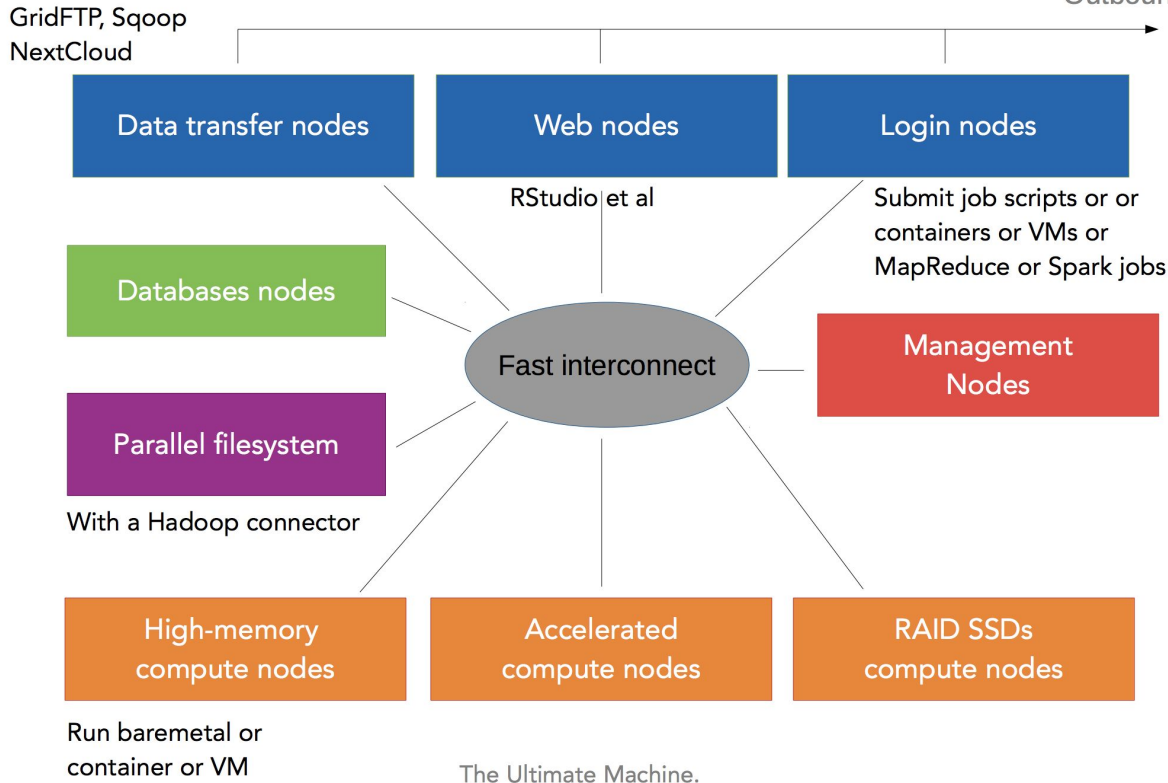
the good, the bad and the ugly

*Ümit Seren*  
*HPC Engineer at the Vienna BioCenter*

Github: @timeu  
Twitter: @timeu\_s

*5th EasyBuild User Meeting - Jan 30th, 2020 - Barcelona*

# The “Cloudster” and How we’re Building it!



Shamelessly stolen from  
Damien François Talk --  
“*The convergence of HPC  
and BigData*  
What does it mean for  
HPC sysadmins?” -  
FOSDEM 2019

# Who Are We ?

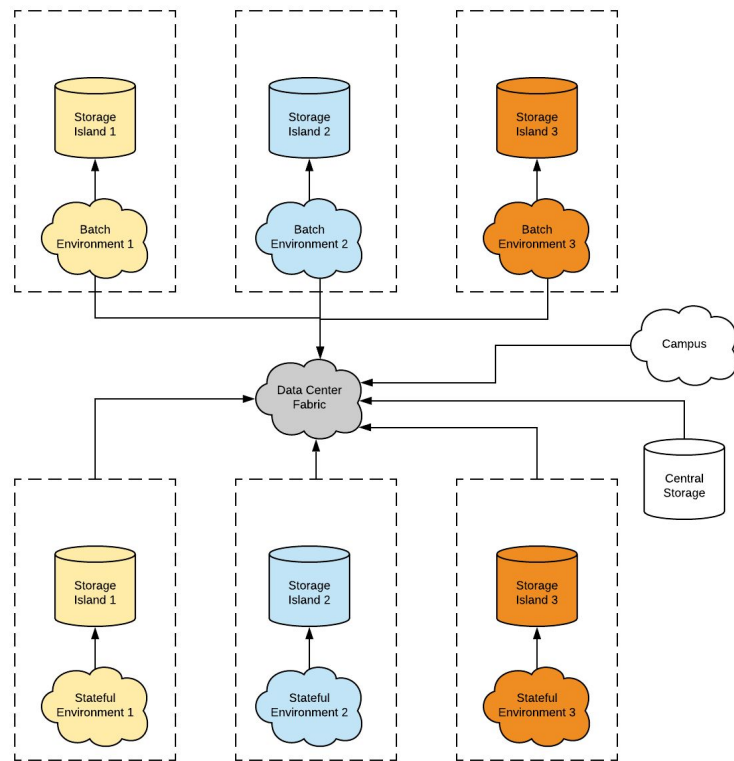
- Part of Cloud Platform Engineering Team at molecular biology research institutes (IMP, IMBA, GMI) located in Vienna, Austria at the Vienna Bio Center.
- Tasked with delivery and operations of IT infrastructure for ~ 40 research groups (~ 500 scientists).
- IT department delivers full stack of services from workstations, networking, application hosting and development (among many others).
- Part of IT infrastructure is delivery of HPC services for our campus
- 14 People in total for everything.

# Vienna BioCenter Computing Profile

- Computing infrastructure almost exclusively dedicated to bioinformatics (genomics, image processing, cryo electron microscopy, etc.)
- Almost all applications are data exploration, analysis and data processing, no simulation workloads
- Have all machinery for data acquisition on site (sequencers, microscopes, etc.)
- Operating and running several compute clusters for batch computing and several compute clusters for stateful applications (web apps, databases, etc.)

# What We Had Before

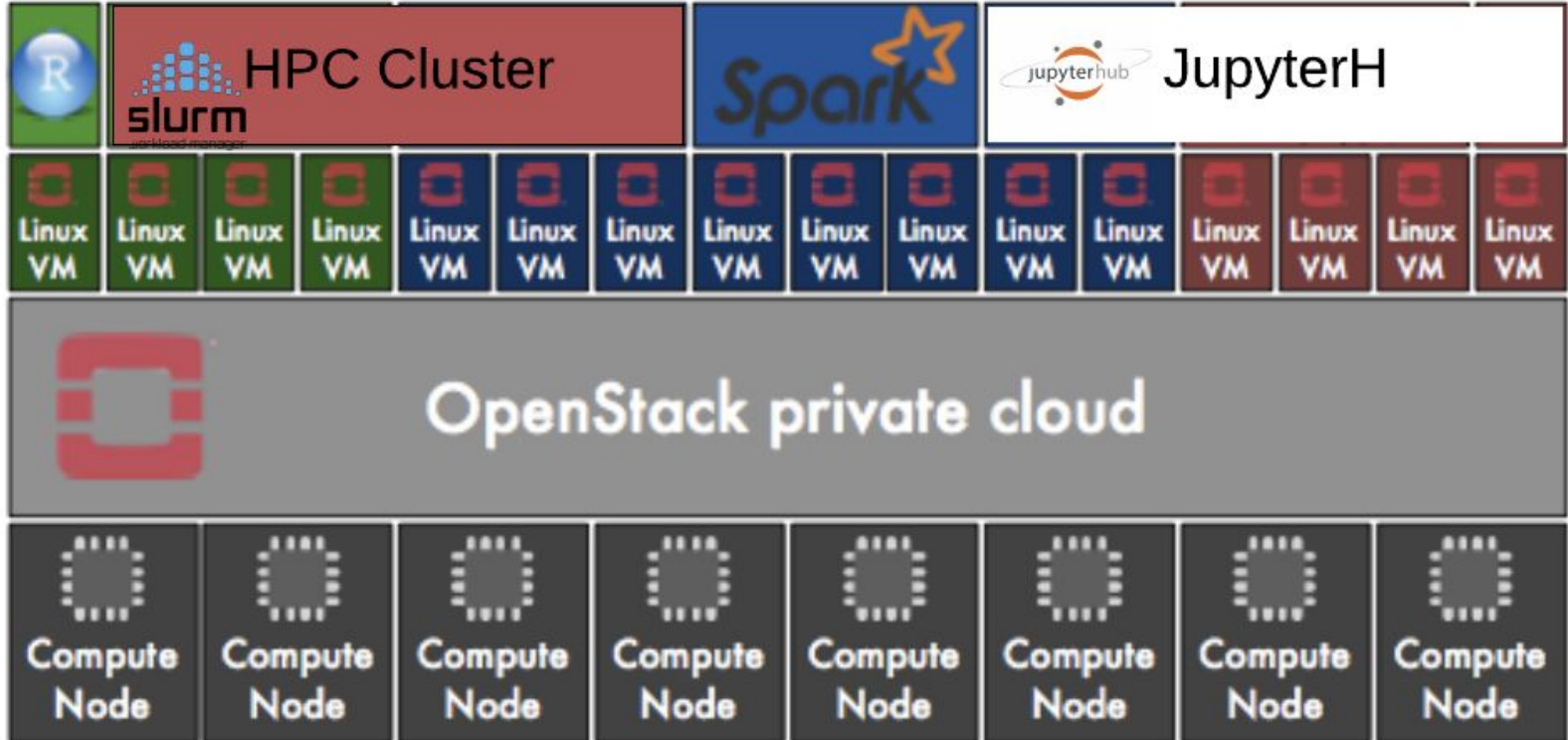
- Siloed islands of infrastructure
- Cant talk to other islands, can't access data from other island (or difficult logistics for users)
- Nightmare to manage
- No central automation across all resources easily possible



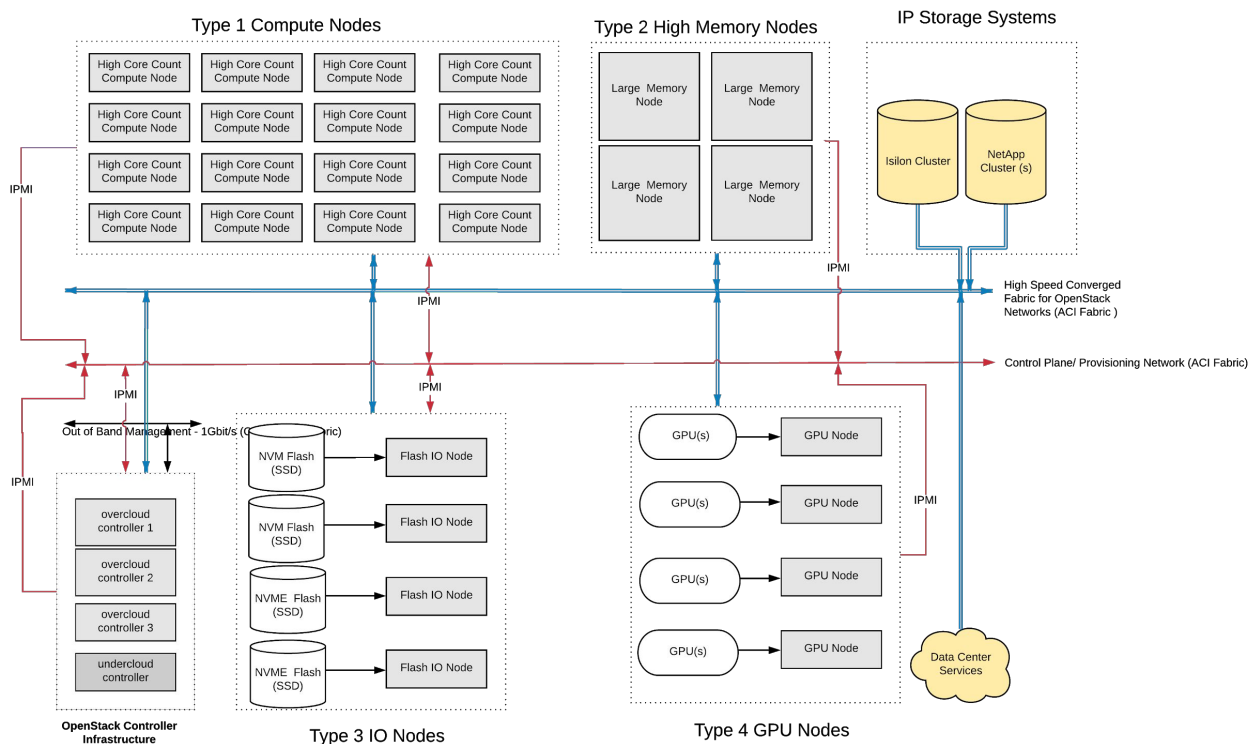
# Meet the CLIP Project

- OpenStack was chosen to be evaluated further as platform for this
- Setup a project “CLIP” (Cloud Infrastructure Project) and formed project team (4.0 FTE) with a multi phase approach to delivery of the project.
- Goal is to implement not only a new HPC platform but a software defined datacenter strategy based on OpenStack and deliver HPC services on top of this platform
- Delivered in multiple phases

# What We're Aiming At



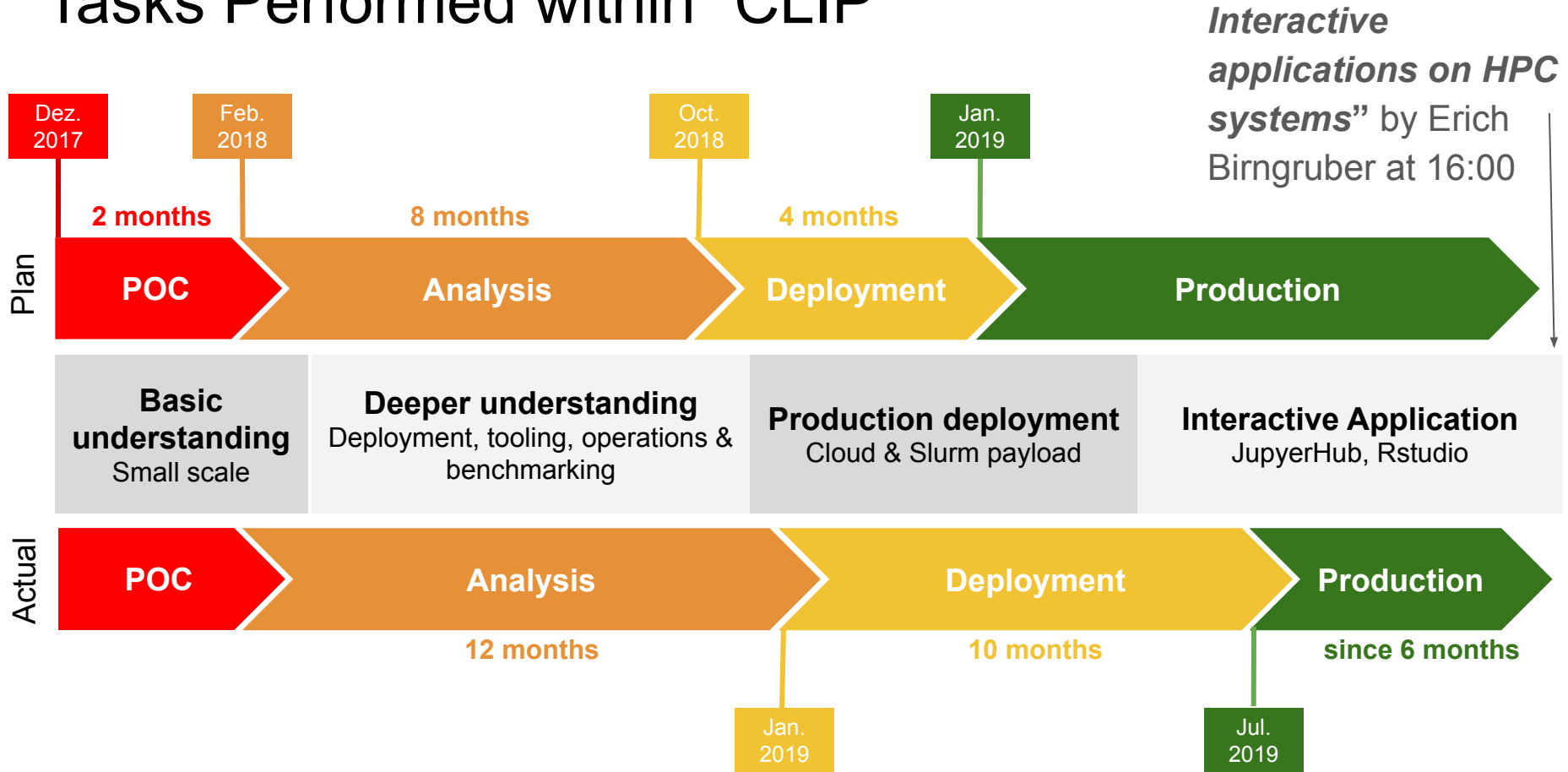
# CLIP Cloud Architecture Hardware



- Heterogeneous nodes (**high core count, high clock, large memory, GPU accelerated, NVME**)
- ~ **200** compute nodes and ~ **7700** Intel SkyLake cores
- **100GbE** SDN RDMA capable Ethernet and some nodes with 2x or 4x ports
- ~ **250TB** NVMe IO Nodes ~ **200Gbyte/s**



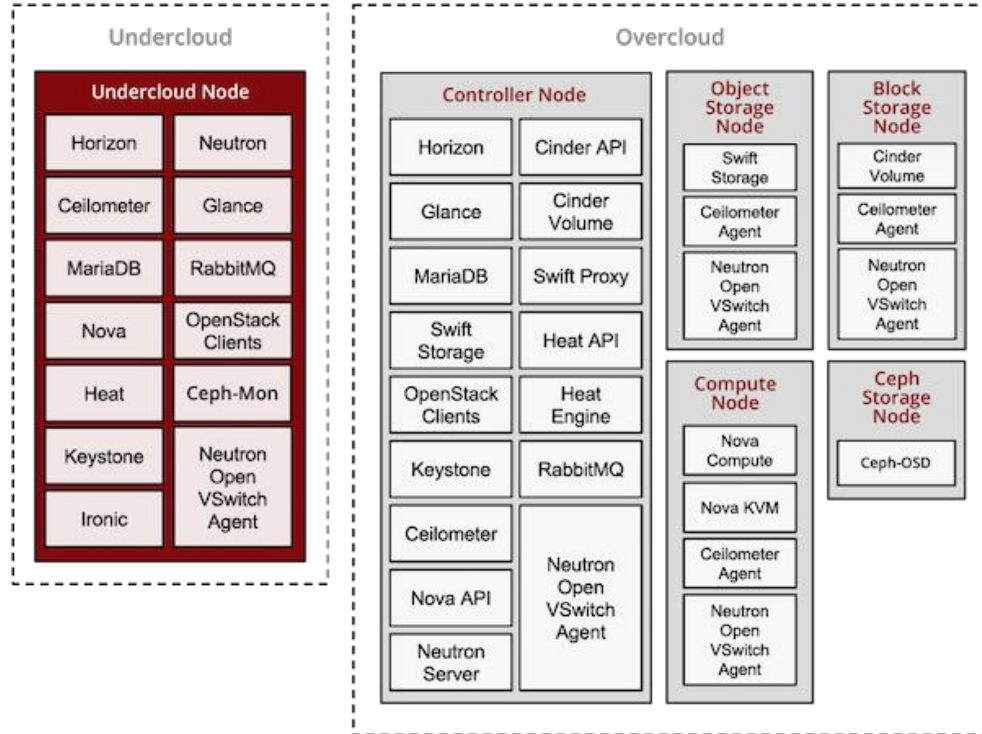
# Tasks Performed within “CLIP”



# Deploying and Operating the Cloud

# Deploying the Cloud - TripleO (OoO)

- TripleO (OoO): Openstack on OpenStack
- **Undercloud**: single node deployment of OpenStack.
  - Deploys the **Overcloud**
- **Overcloud**: HA deployment of OpenStack.
  - Cloud for **Payload**
- Installation with **GUI** or **CLI** ?



# Deploying the Cloud - Should we use the GUI ?

The screenshot displays the Red Hat OpenStack Platform Director interface. At the top, the header reads "RED HAT OPENSTACK PLATFORM DIRECTOR" with a user profile for "admin" and a "Logout" link. Below the header, there are tabs for "Deployment Plan" and "Nodes". The main content area is titled "overcloud" with a sub-link "Manage Deployments".

The deployment process is shown in four steps:

- 1 Prepare Hardware**: Includes a "+ Register Nodes" button.
- 2 Specify Deployment Configuration**: Includes a link to "Edit Configuration" and the text "Base resources configuration".
- 3 Configure Roles and Assign Nodes**: Shows "1 Nodes available to assign". Below this, five role cards are displayed: "Block Storage", "Controller", "Compute", "Object Storage", and "Ceph Storage". Each card shows "0 Nodes assigned" and an "Assign Nodes" button.
- 4 Deploy**: A green banner indicates "Deployment succeeded" with the message "Stack CREATE completed successfully".

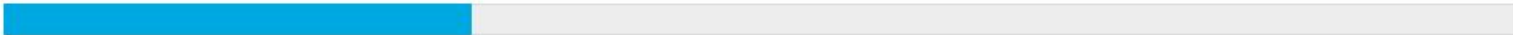
At the bottom, "Overcloud Information:" lists "Overcloud IP address: 10.12.148.155".

On the right side, a "Validations" panel is visible, featuring a "Refresh" button and a list of validation checks with their status (pass/fail) and the phase they occurred in (e.g., pre-deployment, post-deployment).

# Deploying the Cloud - Should we use the GUI ?

Plan overcloud deployment ✕

Deployment in progress 31%



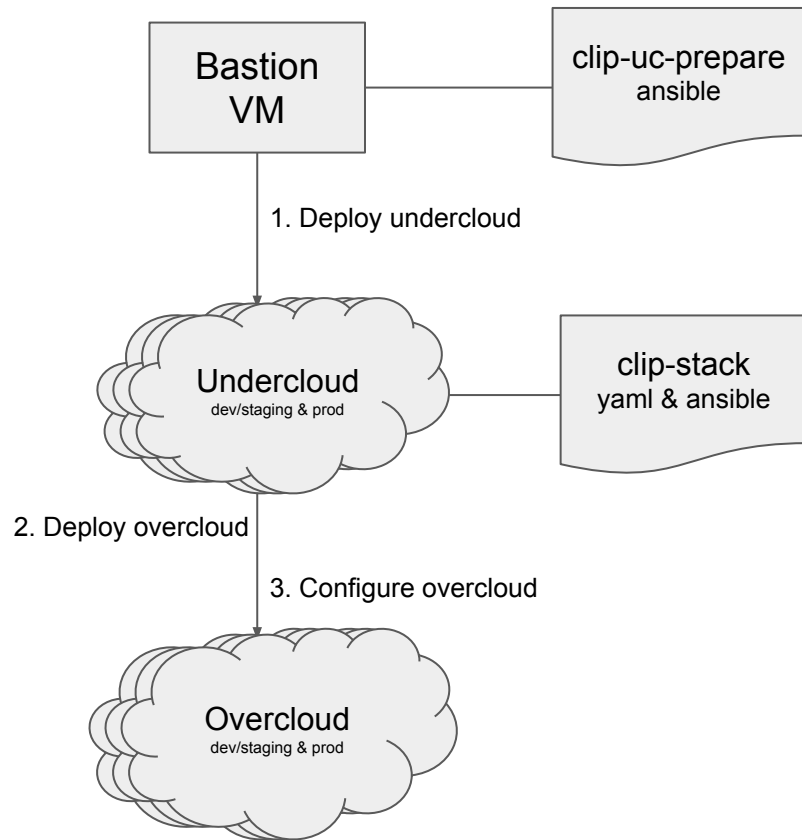
### Resources

Filter Showing 52 of 52 items

| Name                     | Status             | Updated Time         |
|--------------------------|--------------------|----------------------|
| MysqlRootPassword        | CREATE_COMPLETE    | 2016-11-24T07:00:08Z |
| PcsdPassword             | CREATE_COMPLETE    | 2016-11-24T07:00:08Z |
| VipMap                   | CREATE_COMPLETE    | 2016-11-24T07:00:08Z |
| RabbitCookie             | CREATE_COMPLETE    | 2016-11-24T07:00:08Z |
| Controller               | INIT_COMPLETE      | 2016-11-24T07:00:08Z |
| ObjectStorage            | INIT_COMPLETE      | 2016-11-24T07:00:08Z |
| ObjectStorageIpListMap   | INIT_COMPLETE      | 2016-11-24T07:00:08Z |
| ControllerIpListMap      | INIT_COMPLETE      | 2016-11-24T07:00:08Z |
| BlockStorageServiceChain | CREATE_IN_PROGRESS | 2016-11-24T07:00:08Z |
| ComputeHostsDeployment   | INIT_COMPLETE      | 2016-11-24T07:00:08Z |
| RedisVirtualIP           | CREATE_COMPLETE    | 2016-11-24T07:00:08Z |
| StorageVirtualIP         | CREATE_COMPLETE    | 2016-11-24T07:00:08Z |

# Deploying the Cloud - Code as Infra & GitOps !

- Web GUI does not scale
  - → **Disable the Web UI and deploy from the CLI**
- TripleO internally uses *heat* to drive *puppet* that drives *ansible* `~\_(\_)\_/\_`
- Use *ansible* to drive the TripleO installer and rest of infra
- Entire end-2-end deployment from code



# Deploying the Cloud - Pitfalls and Solutions!

- TripleO is slow because **Heat** → **Puppet** → **Ansible** !!
  - Update takes ~ 60 minutes even for simple config change
- Customize using ansible instead ? Unfortunately not robust :-(
  - Stack update (scale down/up) will overwrite our changes
  - → services can be down
- → Let's compromise: Use both
  - Iterate with ansible → Use TripleO for final configuration
- Ansible everywhere else !
  - Network, Moving nodes between environments, etc

# Operating the Cloud - Package Management

- 3 environments & infra as code: reproducibility and testing of upgrades
- What about software versions ? → **Satellite/Foreman** to the rescue !
- Software Lifecycle environments ↔ Openstack environments

## Lifecycle Environment Paths

[+ Create Environment Path](#)

|                |                    |               |                        |                           |                   |                |
|----------------|--------------------|---------------|------------------------|---------------------------|-------------------|----------------|
| <b>Library</b> | Content Views<br>5 | Products<br>7 | Yum Repositories<br>14 | Docker Repositories<br>99 | Packages<br>52289 | Errata<br>5100 |
|----------------|--------------------|---------------|------------------------|---------------------------|-------------------|----------------|

[+ Add New Environment](#)

|               | <b>Dev</b> | <b>Staging</b> | <b>Prod</b> |
|---------------|------------|----------------|-------------|
| Content Views | 1          | 1              | 1           |
| Content Hosts | 8          | 8              | 199         |



# Operating the Cloud - Package Management

1. Create **Content Views** (contains RPM repos and containers)
2. **Publish** new versions of Content Views
3. **Test** in dev/staging and **roll** them **forward** to production

ccv-clip

Publish New Version

Select Action 

[Content Views](#) » [ccv-clip](#) » [Versions](#) 

Details

Versions













Content Views

History

Tasks

Filter...

Search 

| Version                      | Status  | Environments    | Content  | Description  | Actions   |
|------------------------------|---|-----------------|--|--|---|
| <a href="#">Version 17.0</a> | Published (2020-01-10 14:05:04 +0100)           | Library         | 51050 Packages<br>5026 Errata ( 893  3346  787  )       | Updated RHEL OS base packages, OSP13 RPMs and Cisco ACI RPMs (4.2.3) | <a href="#">Promote</a>    |
| <a href="#">Version 16.0</a> | Promoted to Dev (2019-10-21 16:51:53 +0200)     | Dev             | 50336 Packages<br>4958 Errata ( 869  3308  781  )       | Updated RHEL OS base packages, OSP13 RPMs and Cisco ACI RPMs         | <a href="#">Promote</a>    |
| <a href="#">Version 11.0</a> | Promoted to Library (2019-10-21 15:50:38 +0200) | Staging<br>Prod | 46413 Packages<br>4474 Errata ( 766  2983  725  ) | Upgrade OSP13 packages   | <a href="#">Promote</a>  |

# Operating the Cloud - Tracking Bugs in OS

- How to keep track of bugs in OpenStack ?
- → Track bugs, workaround and the status in JIRA project (CRE)

The screenshot displays a JIRA Kanban board for the 'CLIP CRE Kanban' project. The board is organized into columns representing different stages of the bug lifecycle: TO DO, IN PROGRESS, REVIEW, WAITING, WORKAROUND, and DONE. Each ticket card includes a unique ID (e.g., CRE-39), a brief description of the issue, the current status, and the assigned user's profile picture. The 'TO DO' column contains three tickets, 'IN PROGRESS' has three, 'REVIEW' has one, 'WAITING' has three, 'WORKAROUND' has three, and 'DONE' has three. The board also features a sidebar with navigation icons and a top navigation bar with filters and board settings.

| Column      | Ticket ID | Description  | Status      | Assignee |
|-------------|-----------|--|-------------|----------|
| TO DO       | CRE-39    | cinder is broken<br>Backlog  | Open        | J        |
|             | CRE-25    | Controller runs out of disk space                                  | Open        | J        |
|             | CRE-61    | Stack deploy of payload hangs sometimes with VMs in BUILD state    | Open        | J        |
| IN PROGRESS | CRE-13    | collectd turbostat plugin parsing error                            | In Progress | J        |
|             | CRE-59    | VM cannot bring networking up due to DHCP/Opflex issue             | In Progress | J        |
|             | CRE-61    | Stack deploy of payload hangs sometimes with VMs in BUILD state    | In Progress | J        |
| REVIEW      | CRE-70    | SLURM computes are shown as down because they are not              | In Review   | J        |
| WAITING     | CRE-14    | rsyslog fluentd integration  | Waiting     | J        |
|             | CRE-31    | GBP Mapping Driver does not support shared_external                | Waiting     | J        |
|             | CRE-41    | vhost_net kthread pinning  | Waiting     | J        |
| WORKAROUND  | CRE-3     | nova - multiqueue tap device - increase no. of queues              | Workaround  | J        |
|             | CRE-5     | Installing of CiscoACI Puppet RPM via TripleO                      | Workaround  | J        |
|             | CRE-7     | Overcloud nodes are registered in Satellite with shortname instead | Workaround  | J        |
| DONE        | CRE-1     | all the things   | Resolved    | J        |
|             | CRE-2     | make hostnames great again   | Resolved    | J        |
|             | CRE-4     | Typo in heat template for PublicVirtualFixedIPs                    | Resolved    | J        |
|             | CRE-6     | openstack network agent delete not working                         | Resolved    | J        |

# Deploying and operating the Cloud - Summary

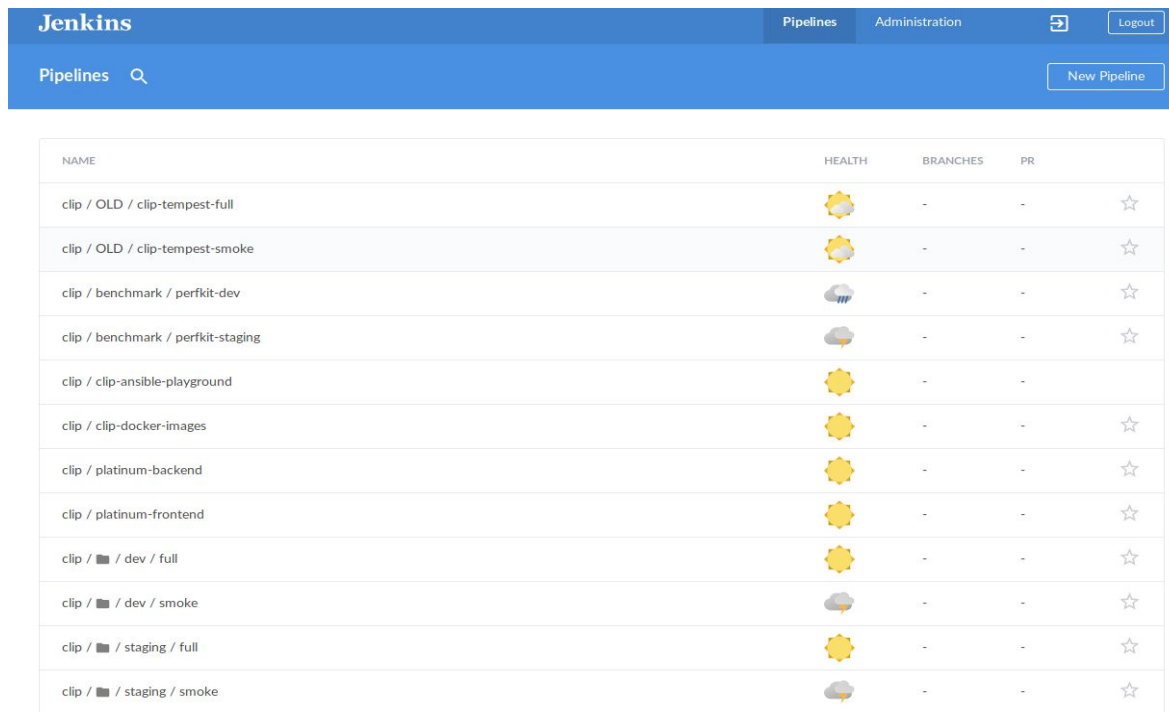
## Lessons learned and pitfalls of OpenStack/TripleO:

- OpenStack and TripleO are complex piece of software
  - **Dev/staging environment & package management**
- Upgrades can break the cloud in unexpected ways.
  - OSP11 (non-containerized) → OSP12 (containerized)
- Containers are no free lunch
  - Container build pipeline for customizations
- TripleO is a supported out of the box installer for common cloud configurations
  - Exotic configurations are challenging
- *“Flying blind through clouds is dangerous”*:
  - Continuous performance and regression testing
- Infra as code (end to end) way to go
  - Requires discipline (proper PR reviews) and release management

















# Cloud Verification & Performance Testing

# Cloud verification & Performance Testing

- How can we make sure and monitor that the cloud works during operations ?
- We leverage OpenStack's own tempest testing suite to run verification against our deployed cloud.
- First smoke test (~ 128 tests) and if this is successful run full test (~ 3000 tests) against the cloud.



The screenshot shows the Jenkins web interface. At the top, there is a blue header with the Jenkins logo, navigation links for 'Pipelines' and 'Administration', a search bar, and a 'Logout' button. Below the header, there is a 'Pipelines' section with a search icon and a 'New Pipeline' button. The main content area displays a table of pipeline jobs.

| NAME   | HEALTH  | BRANCHES | PR |
|--|---|----------|----|
| clip / OLD / clip-tempest-full   |  | -        | -  |
| clip / OLD / clip-tempest-smoke  |  | -        | -  |
| clip / benchmark / perfkit-dev   |  | -        | -  |
| clip / benchmark / perfkit-staging   |  | -        | -  |
| clip / clip-ansible-playground   |  | -        | -  |
| clip / clip-docker-images  |  | -        | -  |
| clip / platinum-backend  |  | -        | -  |
| clip / platinum-frontend   |  | -        | -  |
| clip /  / dev / full      |  | -        | -  |
| clip /  / dev / smoke     |  | -        | -  |
| clip /  / staging / full  |  | -        | -  |
| clip /  / staging / smoke |  | -        | -  |

# Cloud verification & Performance Testing

- How can we make sure and monitor that the cloud works during operations ?
- We leverage OpenStack's own tempest testing suite to run verification against our deployed cloud.
- First smoke test (~ 128 tests) and if this is successful run full test (~ 3000 tests) against the cloud.

```
! 0h / tm / staging / full 19
Branch: --
Commit: --
27m 15s
6 months ago
No changes
Started by upstream job: "15h:testing/staging/node" test #43

X 21 tests have failed
There are 1 new tests failing, 20 existing failing and 204 skipped.

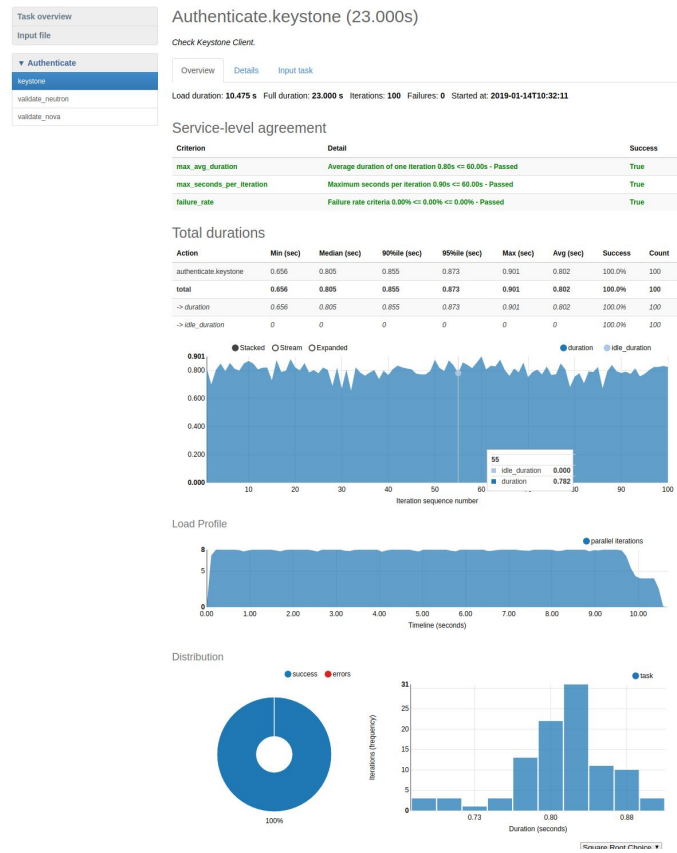
New failing - 1
x test_hotplug_nic[computeId=c5a8f73-e961-41f1-6449-342614f18cfa.network] - tempest.scenario.test_network_basic_ops.TestNetworkBasicOps

Existing failures - 20
x test_hotplug_nic[computeId=c5a8f73-e961-41f1-6449-342614f18cfa.network] - tempest.scenario.test_network_basic_ops.TestNetworkBasicOps
x test_volume_backup_export_import[id=99c541-d680-4724-8a13-13b56840666e] - tempest.api.volume.admin.test_volumes_backup.VolumesBackupAdminTest
x test_volume_backup_restore_status[id=47a25425-a891-4e13-961c-c450ea21e94f] - tempest.api.volume.admin.test_volumes_backup.VolumesBackupAdminTest
x test_create_list_show_delete_interfaces[id=73bf0f02-5990-4b11-8184-v9ca81065051.network] - tempest.api.compute.servers.test_attach_interfaces.AttachInterfacesTestJSON
x test_list_user_projects[id=a831a70c-c35b-430b-92af-81e8bc54378e] - tempest.api.identity.admin.v3.test_users.UserV3TestJSON
x test_enrolled_roles_create_check_show_delete[id=c9c316c-d706-4728-1c8a-eb1912081b69] - tempest.api.identity.admin.v3.test_roles.RoleV3TestJSON
x test_snapshot_backup[id=b6cf285c-a7f1-479b-8c1a-8c34c16543c1] - tempest.api.volume.test_volumes_snapshots.VolumesSnapshotTestJSON
x test_list_endpoints_for_tokens[id=c3a6d77-e008-4a61-a6bd-96906456ad31] - tempest.api.identity.admin.v2.test_tokens.TokensTestJSON
x test_snapshot_create_delete_with_volume_in_use[computeId=8567b54c-4d55-446d-a31f-651d6ea3ff2] - tempest.api.volume.test_volumes_snapshots.VolumesSnapshotTestJSON
x test_snapshot_create_offline_delete_online[computeId=5210a3de-85d0-11e6-b021-641c676a5661] - tempest.api.volume.test_volumes_snapshots.VolumesSnapshotTestJSON
x test_backup_create_and_restore_to_an_existing_volume[id=b5d83760-7064-4556-886c-4a721e899306] - cinder.tests.tempest.api.volume.test_volume_backup.VolumesBackupTest
x test_incremental_backup[id=c810f62c-c840-43ab-96aa-471b74516a98] - cinder.tests.tempest.api.volume.test_volume_backup.VolumesBackupTest
x test_volume_snapshot_backup[id=885410c6-cd1d-452c-a409-7c32b70be151] - cinder.tests.tempest.api.volume.test_volume_backup.VolumesBackupTest
x test_backup_create_attached_volume[computeId=07af8f6d-80af-44c9-a5d-c8427b1b62e6] - tempest.api.volume.test_volumes_backup.VolumesBackupTest
x test_baremetal_volume_backup_and_restore[id=2a8ba340-0ff2-4511-9db7-646607156b15.image] - tempest.api.volume.test_volumes_backup.VolumesBackupTest
x test_baremetal_server_volume[computeId=549173d5-426b-426b-1062-c8bf94d08943.image.network] - ironic.tempest.plugins.tests.scenario.test_baremetal_basic_ops.BaremetalBasicOps
x test_volume_backup_create_get_detailed_list_restore_delete[id=a66b488-80e1-4794-807f-57a095728c6] - tempest.api.volume.test_volumes_backup.VolumesBackupTest
x test_hidden_stack - heat_integrationtests.functional.test_stack_ops.StackOpsTest
x test_create_ebs_image_and_check_block[computeId=36c34c67-7654-4b59-b188-02a2d458a63b.image.volume] - tempest.scenario.test_volume_boot_pattern.TestVolumeBootPattern
x test_volume_boot_pattern[computeId=557c2c2-4e0b-4d0c-9f9e-f86765ff311b.image.volume] - tempest.scenario.test_volume_boot_pattern.TestVolumeBootPattern
x test_baremetal_boot_from_volume[computeId=d665661-0221-44ac-b795-5745f8b0fc1.image.network.volume] - ironic.tempest.plugins.tests.scenario.test_baremetal_boot_from_volume.BaremetalBFW

Skipped - 204
o setUpClass (tempest.api.compute.admin.test_keypairs_v210.KeyPairV210TestJSON) -
o setUpClass (tempest.api.compute.admin.test_security_group_default_rules.SecurityGroupDefaultRuleTest) -
o setUpClass (tempest.api.compute.admin.test_fixed_ips.FixedIPTestJSON) -
o setUpClass (tempest.api.compute.servers.test_servers.ServerShowV247Test) -
o setUpClass (tempest.api.compute.admin.test_server_flagnostics.ServerDiagnosticV248Test) -
o setUpClass (tempest.api.compute.admin.test_volume_swap.TestVolumeSwap) -
o setUpClass (tempest.api.compute.certificates.test_certificates.CertificateV2TestJSON) -
o setUpClass (tempest.api.compute.admin.test_auto_allocate_network.AutoAllocateNetworkTest) -
o setUpClass (tempest.api.compute.admin.test_live_migration.LiveAutoBlockMigrationV225Test) -
o setUpClass (tempest.api.compute.admin.test_server_flagnostics_negative.ServerDiagnosticNegativeV248Test) -
o setUpClass (tempest.api.compute.admin.test_fixed_ips_negative.FixedIPNegativeTestJSON) -
o setUpClass (tempest.api.compute.admin.test_live_migration_negative.LiveMigrationNegativeTest) -
o test_cold_migration[id=48f0b652-3b6f-4746-9a27-3143a366300d] - tempest.api.compute.admin.test_migrations.MigrationsAdminTest
o test_list_migrations_in_flavor_resize_situation[id=1b512062-8093-438a-b47c-37d2597f4641] - tempest.api.compute.admin.test_migrations.MigrationsAdminTest
```

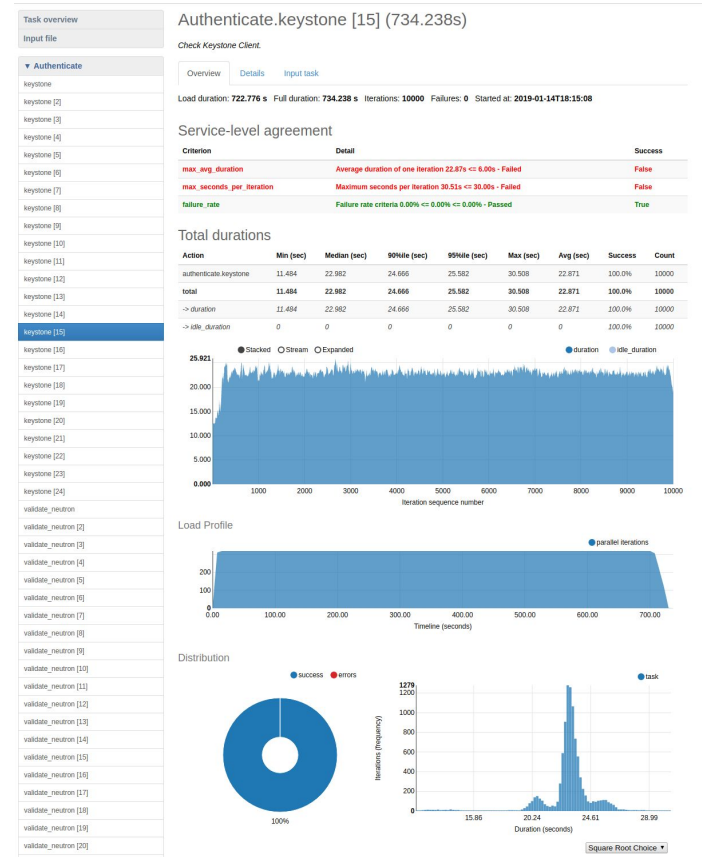
# Cloud verification & Performance Testing

- Ok, the Cloud works but what about performance ? How can we make sure that OS performs when upgrading software packages etc ?
- We plan to use *Browbeat* to run *Rally* (control plane performance/stress testing), *Shaker* (network stress test) and *PerfkitBenchmarker* (payload performance) tests on a regular basis or before and after software upgrades or configuration changes



# Cloud verification & Performance Testing

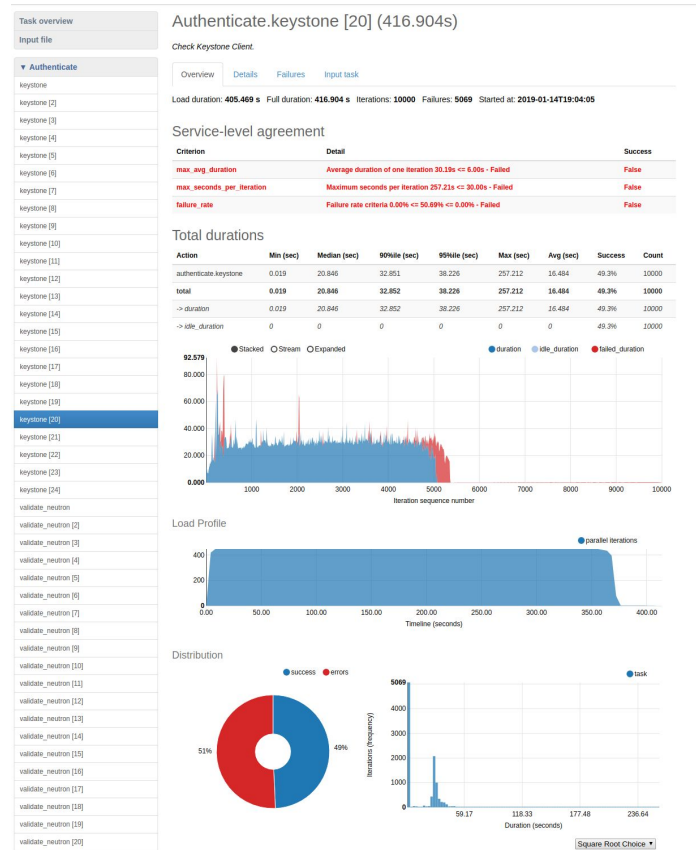
- Ok, the Cloud works but what about performance ? How can we make sure that OS performs when upgrading software packages etc ?
- We plan to use *Browbeat* to run *Rally* (control plane performance/stress testing), *Shaker* (network stress test) and *PerfkitBenchmarker* (payload performance) tests on a regular basis or before and after software upgrades or configuration changes





# Cloud verification & Performance Testing

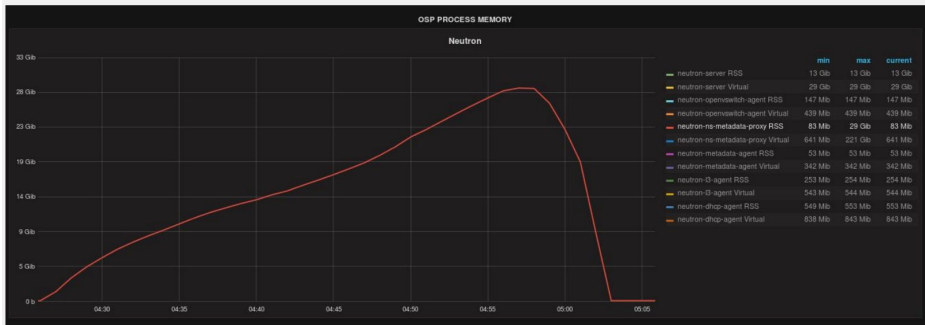
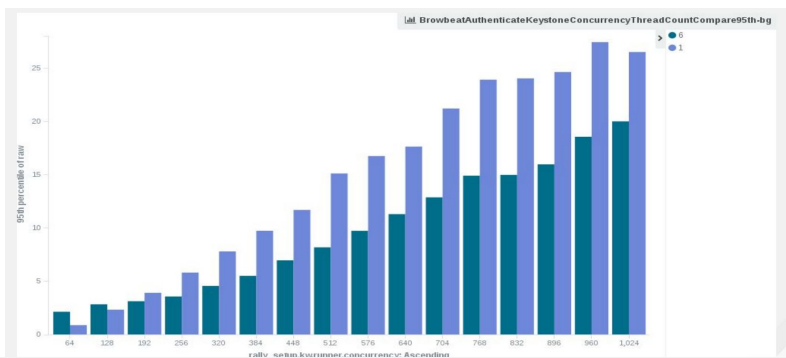
- Ok, the Cloud works but what about performance ? How can we make sure that OS performs when upgrading software packages etc ?
- We plan to use *Browbeat* to run *Rally* (control plane performance/stress testing), *Shaker* (network stress test) and *PerfkitBenchmarker* (payload performance) tests on a regular basis or before and after software upgrades or configuration changes



# Cloud verification & Performance Testing

- Grafana and Kibana dashboard can show more than individual rally graphs:

- Browbeat can show differences between settings or software versions:



Scrolling through Browbeat 22 documents...

| Scenario            | Action                 | conc. | times | 0b5ba58c | 2b177f3b | % Diff  |
|---------------------|------------------------|-------|-------|----------|----------|---------|
| create-list-router  | neutron.create_router  | 500   | 32    | 19.940   | 15.656   | -21.483 |
| create-list-router  | neutron.list_routers   | 500   | 32    | 2.588    | 2.086    | -19.410 |
| create-list-router  | neutron.create_network | 500   | 32    | 3.294    | 2.366    | -28.177 |
| create-list-router  | neutron.create_subnet  | 500   | 32    | 4.282    | 2.866    | -33.075 |
| create-list-port    | neutron.list_ports     | 500   | 32    | 52.627   | 43.448   | -17.442 |
| create-list-port    | neutron.create_network | 500   | 32    | 4.025    | 2.771    | -31.165 |
| create-list-port    | neutron.create_port    | 500   | 32    | 19.458   | 5.412    | -72.189 |
| create-list-subnet  | neutron.create_subnet  | 500   | 32    | 11.366   | 4.809    | -57.689 |
| create-list-subnet  | neutron.create_network | 500   | 32    | 6.432    | 4.286    | -33.368 |
| create-list-subnet  | neutron.list_subnets   | 500   | 32    | 10.627   | 7.522    | -29.221 |
| create-list-network | neutron.list_networks  | 500   | 32    | 15.154   | 13.073   | -13.736 |
| create-list-network | neutron.create_network | 500   | 32    | 10.200   | 6.595    | -35.347 |

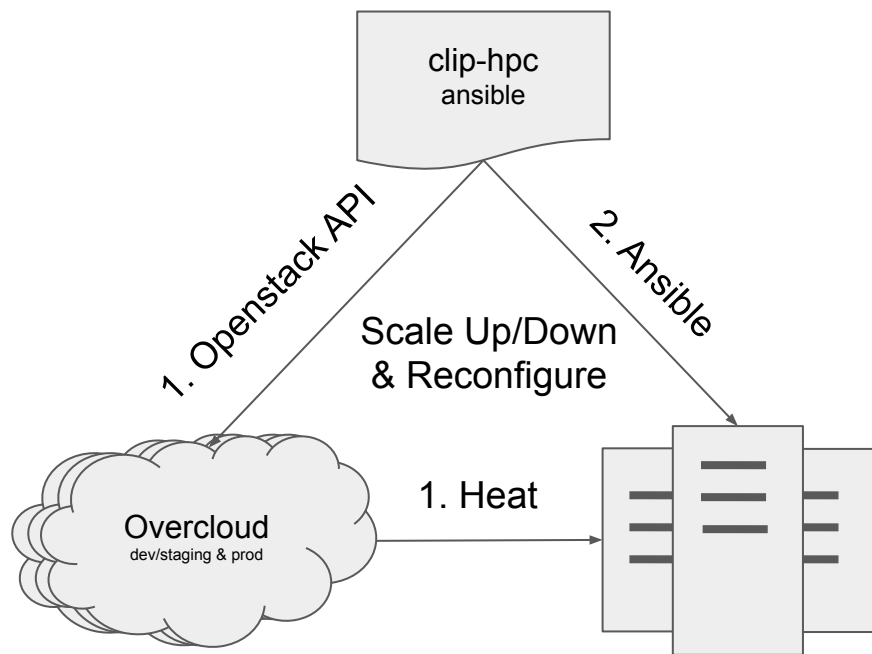
  

| UUID                                 | Version | Build        | Number of runs |
|--------------------------------------|---------|--------------|----------------|
| 938dc451-d881-4f28-a6cb-ad502b177f3b | queens  | 2018-03-20.2 | 1              |
| 6b50b6f7-acae-445a-ac53-78200b5ba58c | ocata   | 2017-XX-XX.X | 3              |

# Deploying the Payload

# Deploying the Cloud - SLURM Cluster

- 2 step process:
  - OpenStack **Heat** to provision → **Ansible inventory**
  - **Ansible** playbook/roles<sup>1</sup> for config -> **SLURM cluster**
- Satellite for package management
- Dev & staging env for testing → roll over to production
- Deploy other complex systems (Spark cluster, k8s, etc)



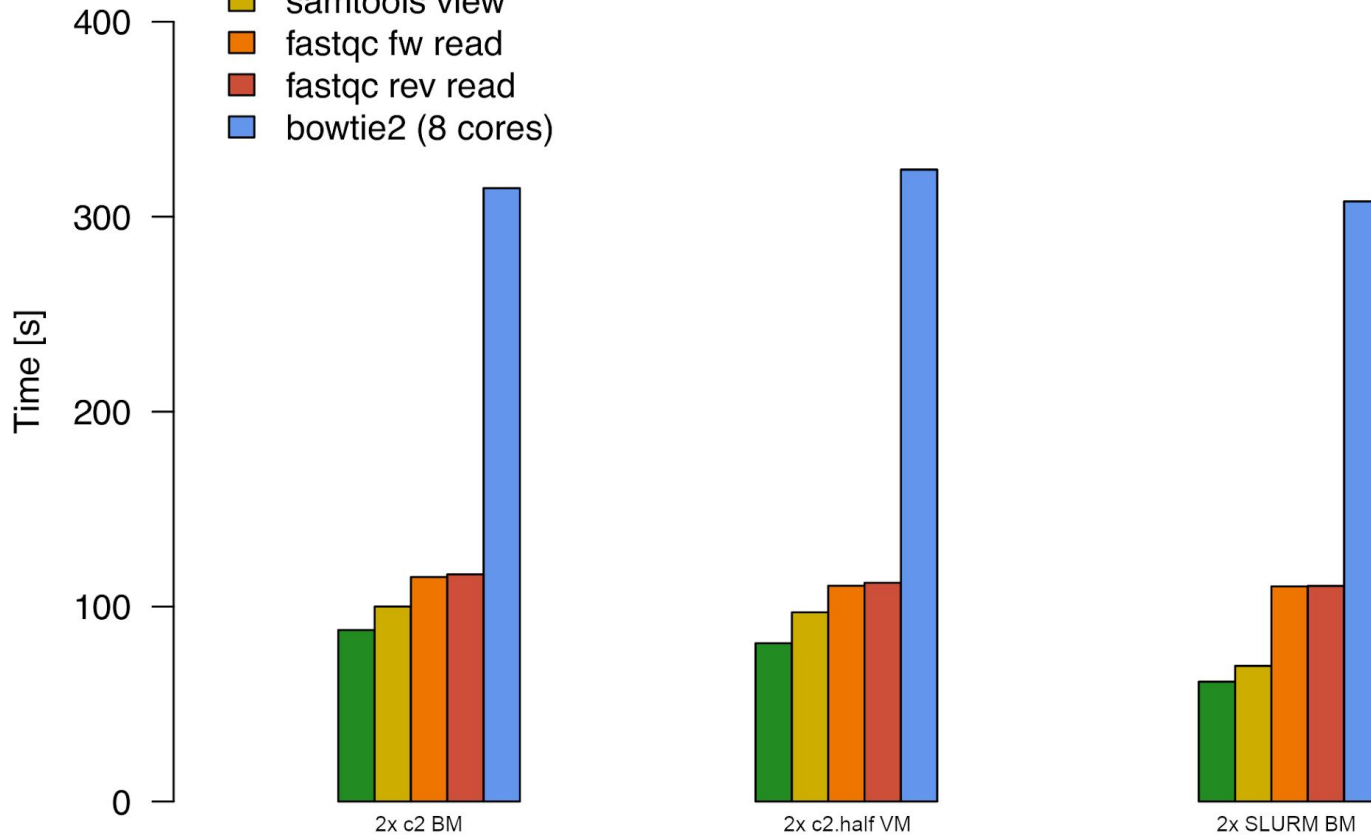
# Deploying the Cloud - Tunings for HPC

- Tuning, Tuning, Tuning required for excellent performance

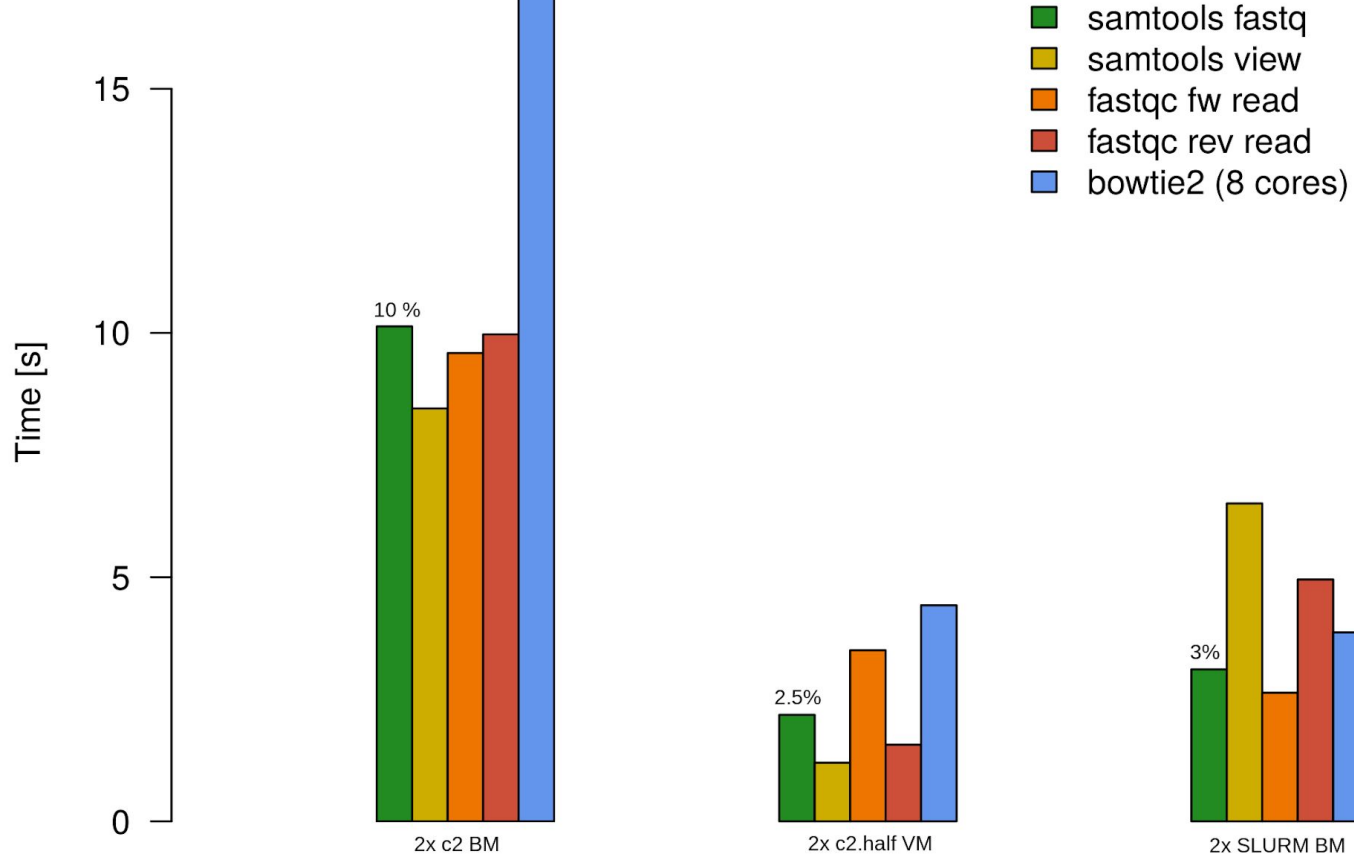
| <b>Tuning</b>                                    | <b>Caveats / Downside</b>   |
|--|---|
| NUMA clean instances (KVM process layout)        | No live migrations<br>No mixing of different VM flavors   |
| Static huge pages (KSM etc.) setup               | If not enough memory is left to hypervisor<br>→ swapping or host services get OOM.<br>No mixing of different VM flavors |
| Core isolation (isolcpus)                        | Performance drop in virtual networking performance → SR-IOV   |
| PCI-E passthrough (GPUs, NVME) and SR-IOV (NICs) | No live migrations and less features compared to fully virtualized networking   |

# Mean execution time

- samtools fastq
- samtools view
- fastqc fw read
- fastqc rev read
- bowtie2 (8 cores)



## Standard deviation of execution time



# Deploying the Cloud - Pitfalls and Issues

- Ansible is slow: Slurm playbook takes ~1 hour (clean 2nd run !)
  - Use tags for recurring day 2 operations (i.e new mount points, change of QOS, etc)
- Satellite 👍 for software versions but remove upstream Centos repos after install
- Some issues only hit under scale:
  - SDN scaling issues when provisioning more than 70 nodes. Workaround: scale in batches
- Isolation of environments ends with shared infra components especially when tightly integrating with OpenStack
  - Update of **DEV** environment caused datacenter wide network outage (bug in SDN)
- Beware of unintended consequences of code changes
  - Triggered accidental re-deploy of payload because of single line change in heat template



# HPC on OpenStack - Lessons Learned

## Bad & Ugly

- OpenStack is *incredibly* complex
- OpenStack is not a product. It is a framework.
- You need 2-3 OpenStack environments (development, staging, prod in our case) to practice and understand upgrades and updates.
- Scaling above certain amount of nodes will be an issue
- Cloud networking is really hard (especially in our case)

## Good

- Open source software with commercial support
- OpenStack integrates well with existing datacenter infrastructure
- API driven software defined datacenter
- Easily deploy multiple payloads side by side like in a Cloud 😊
- Covers a wide range of use cases ranging from virtualized & baremetal HPC clusters to container orchestration engines

# Acknowledgements

## HPC Team

Erich Birngruber

Petar Forai

Petar Jager

Ümit Seren

# Thanks