HPC on OpenStack

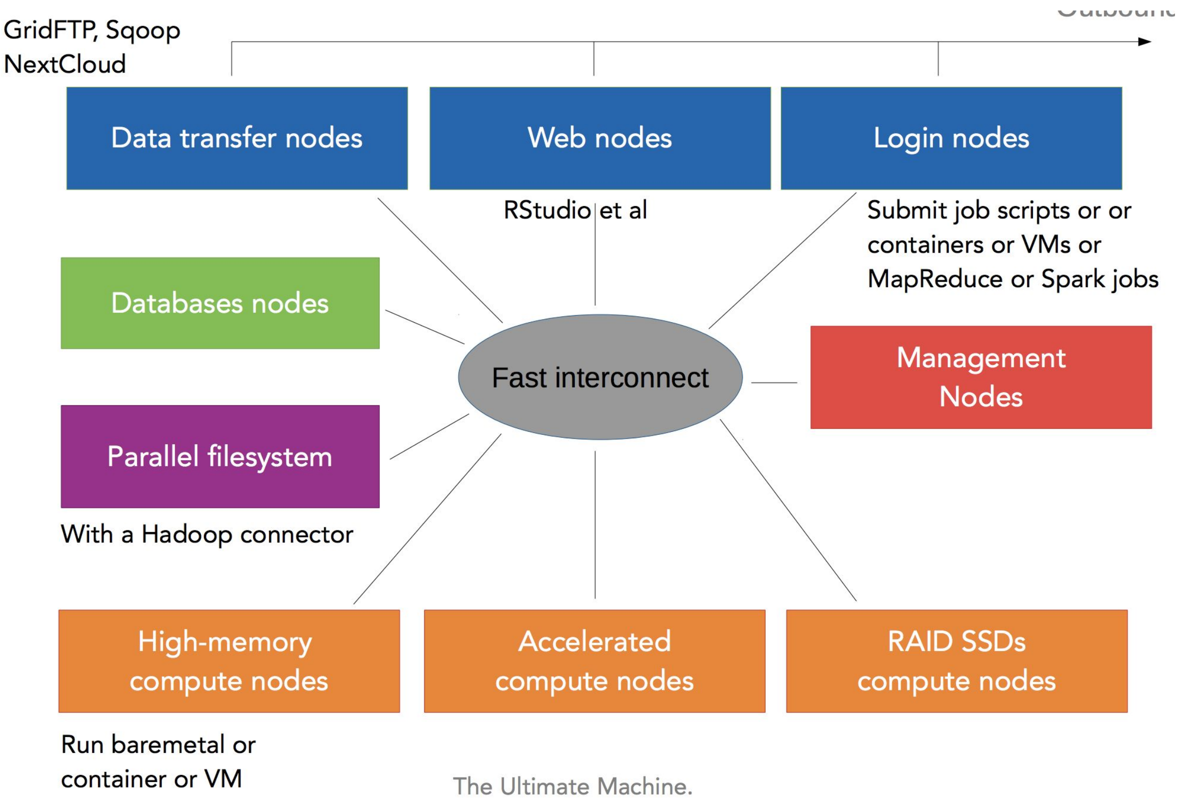
the good, the bad and the ugly

Ümit Seren
HPC Engineer at the Vienna BioCenter

Github: @timeu
Twitter: @timeu_s

FOSDEM 2020 - Feb 02, 2020 - Brussels

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 ?

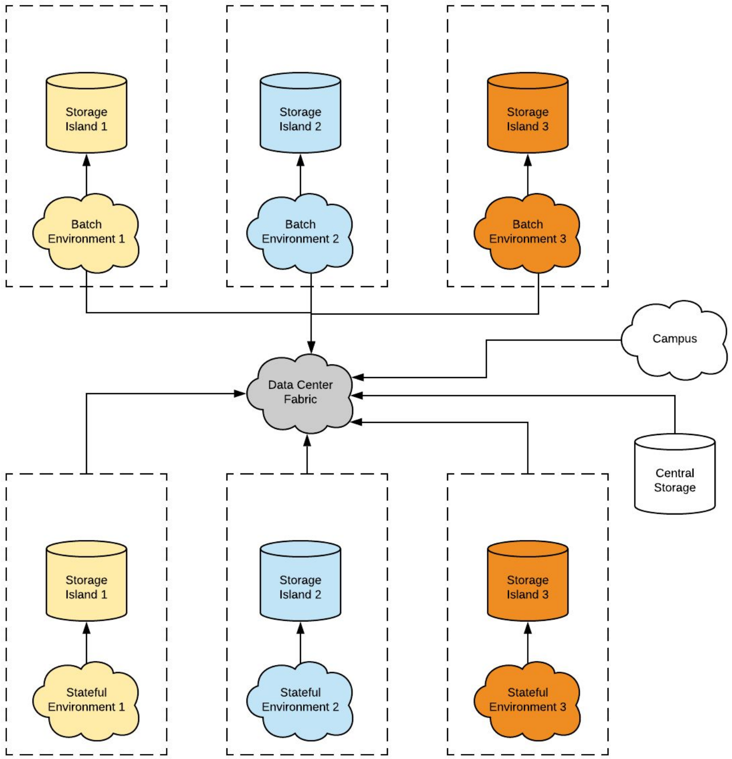
- Part of Cloud Platform Engineering Team at molecular biology research institutes (IMP, IMBA,GMI) located in Vienna, Austria at the Vienna Bio Center.
- Tasked with delivery and operations of IT infrastructure for ~ 40 research groups (~ 500 scientists).
- IT department delivers full stack of services 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

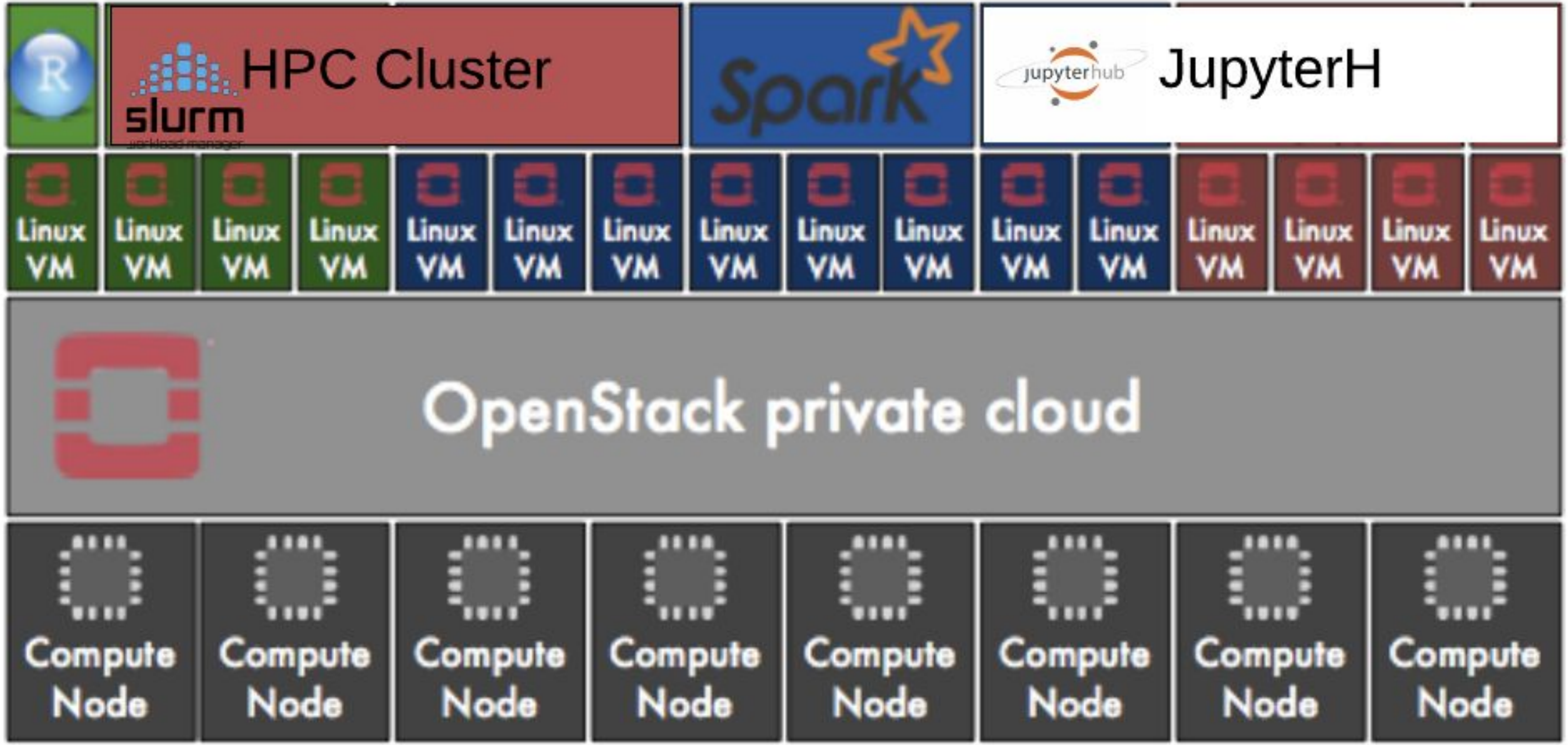
- Siloed islands of infrastructure
- Cant talk to other islands, can't access data from other island (or difficult logistics for users)
- Nightmare to manage
- No central automation across all resources easily possible



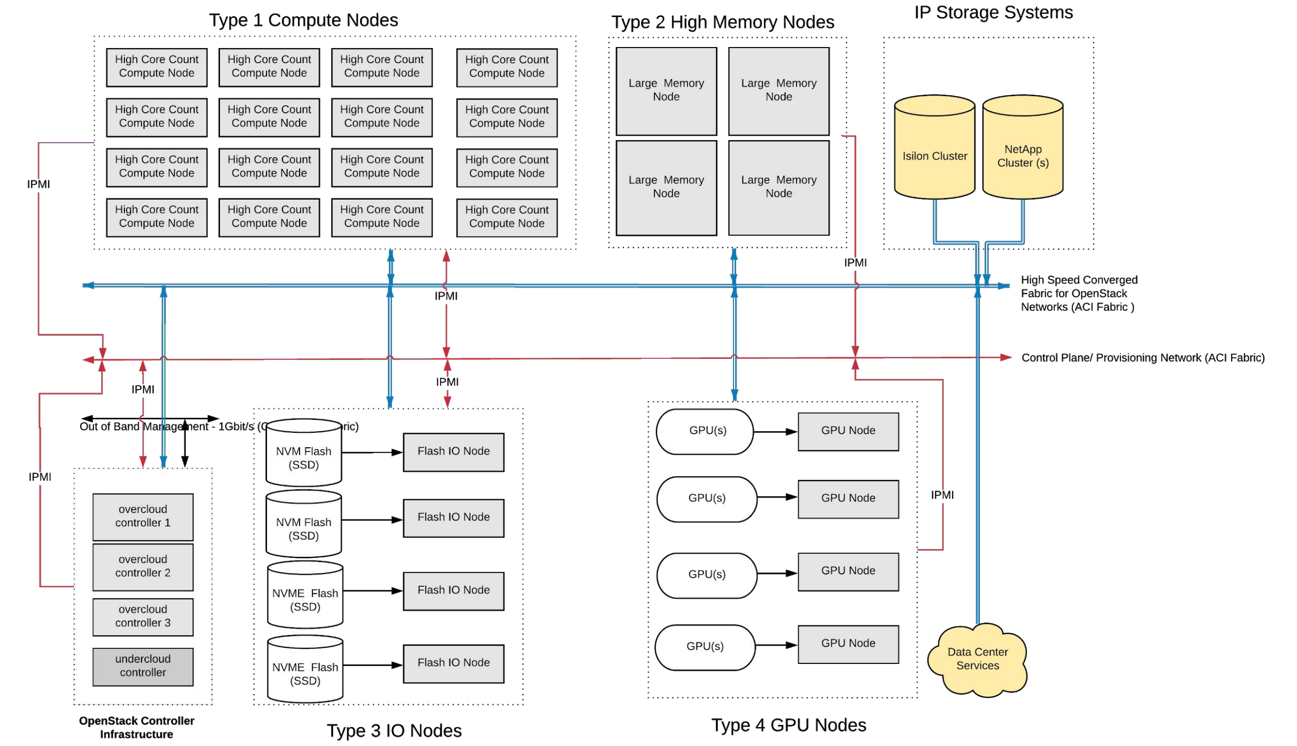
Meet the CLIP Project

- 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



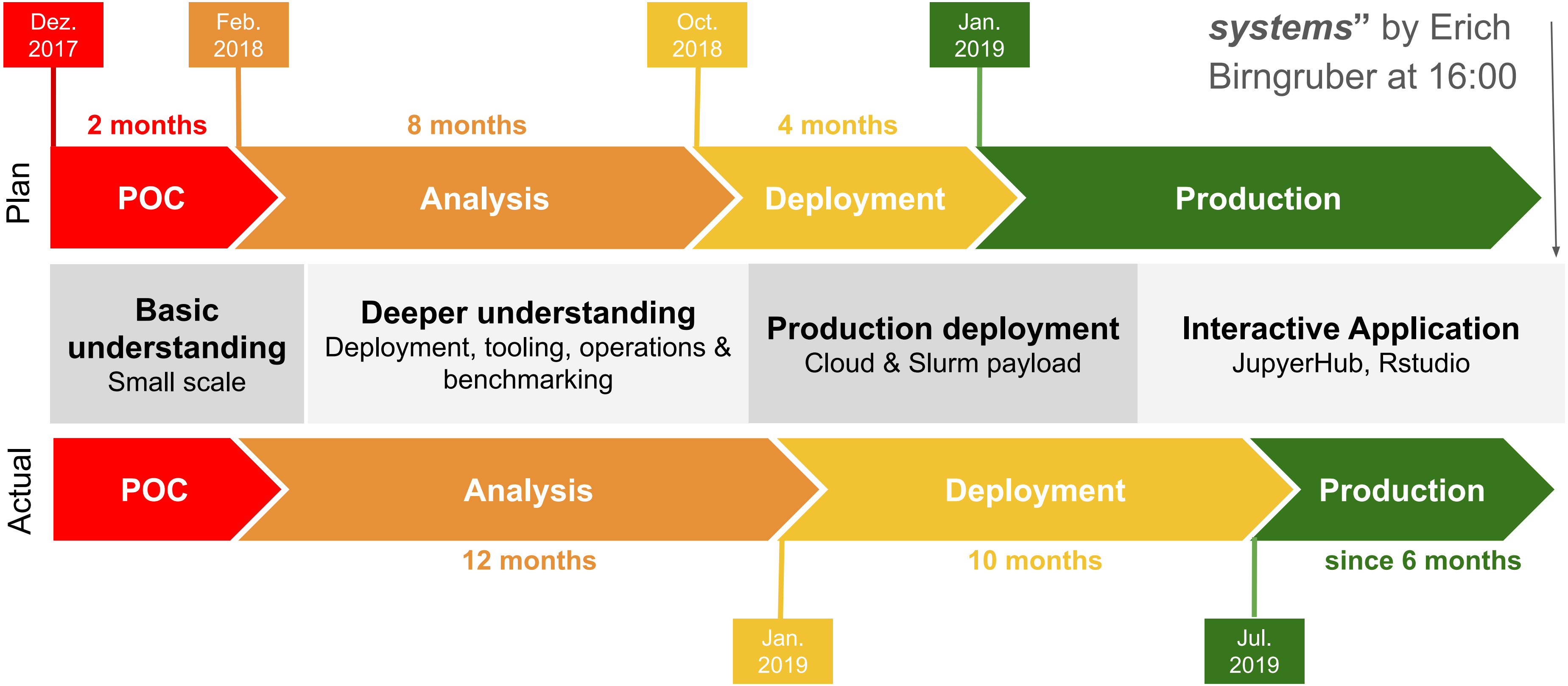
CLIP Cloud Architecture Hardware



- 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”

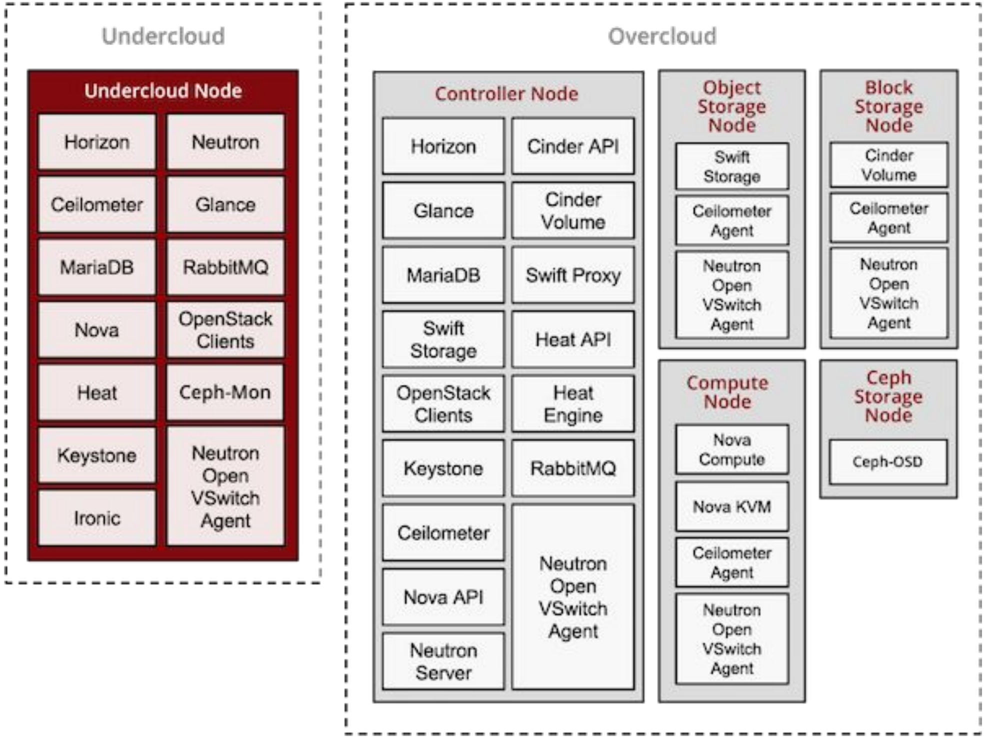
***Interactive applications on HPC systems*”** by Erich Birngruber at 16:00



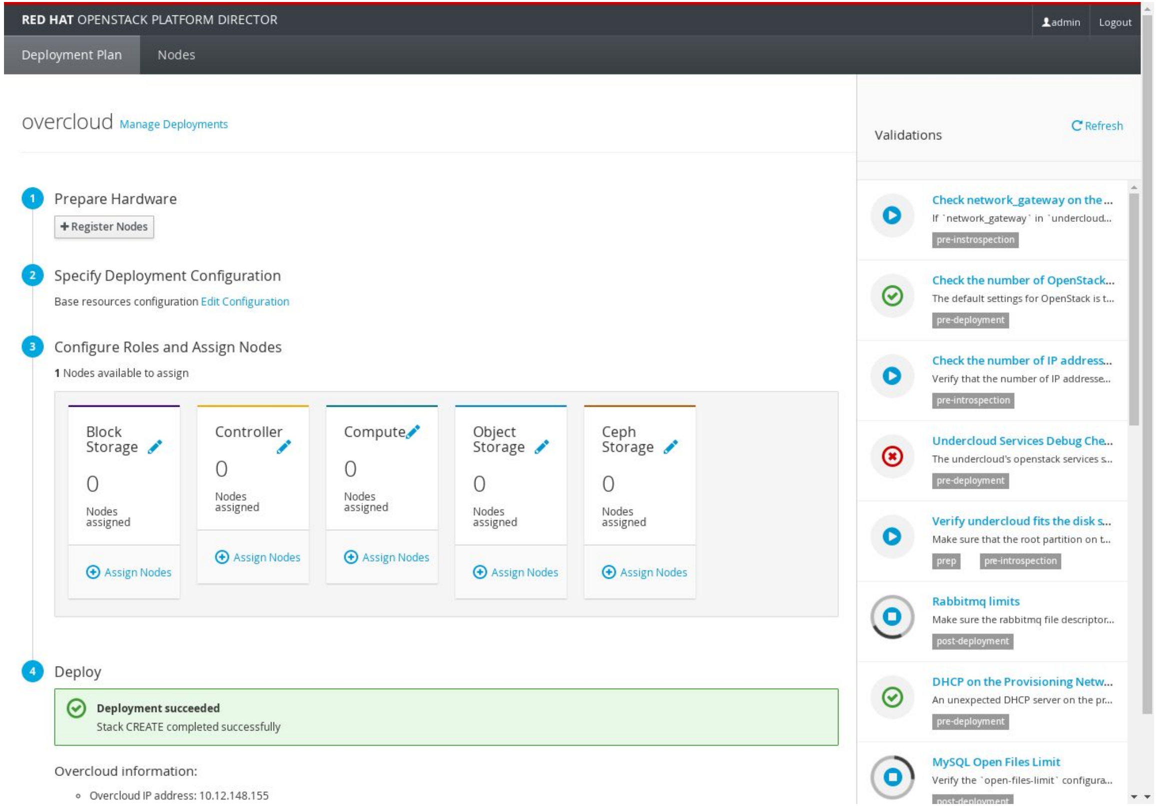
Deploying and Operating the Cloud

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 ?



Deploying the Cloud - Should we use the GUI ?

Plan overcloud deployment

Deployment in progress 31%

Resources

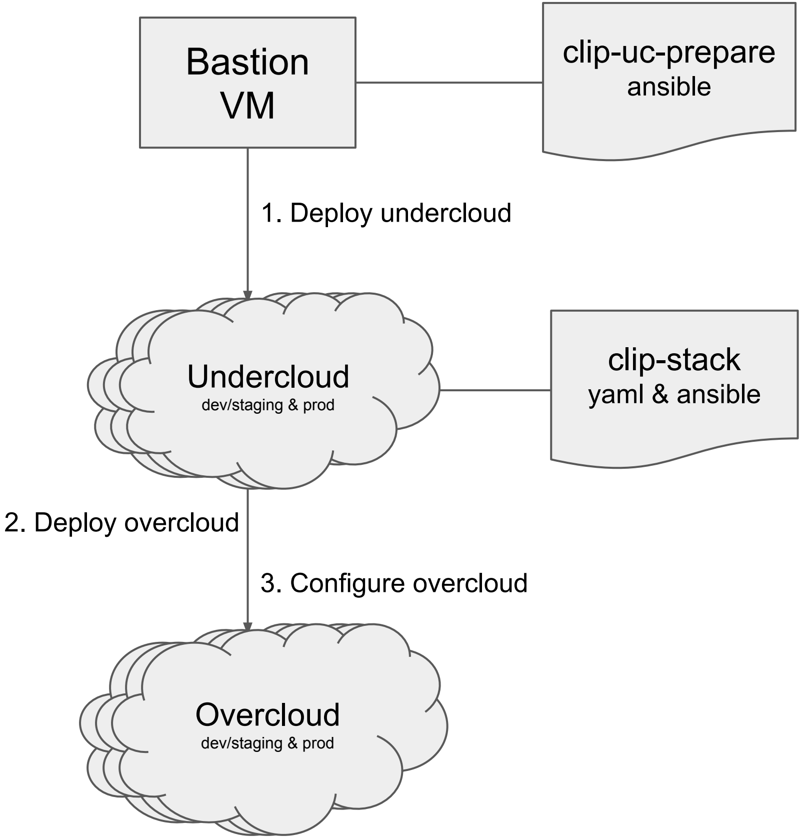
Filter

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

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*** ¯_(ツ)_/¯
- 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

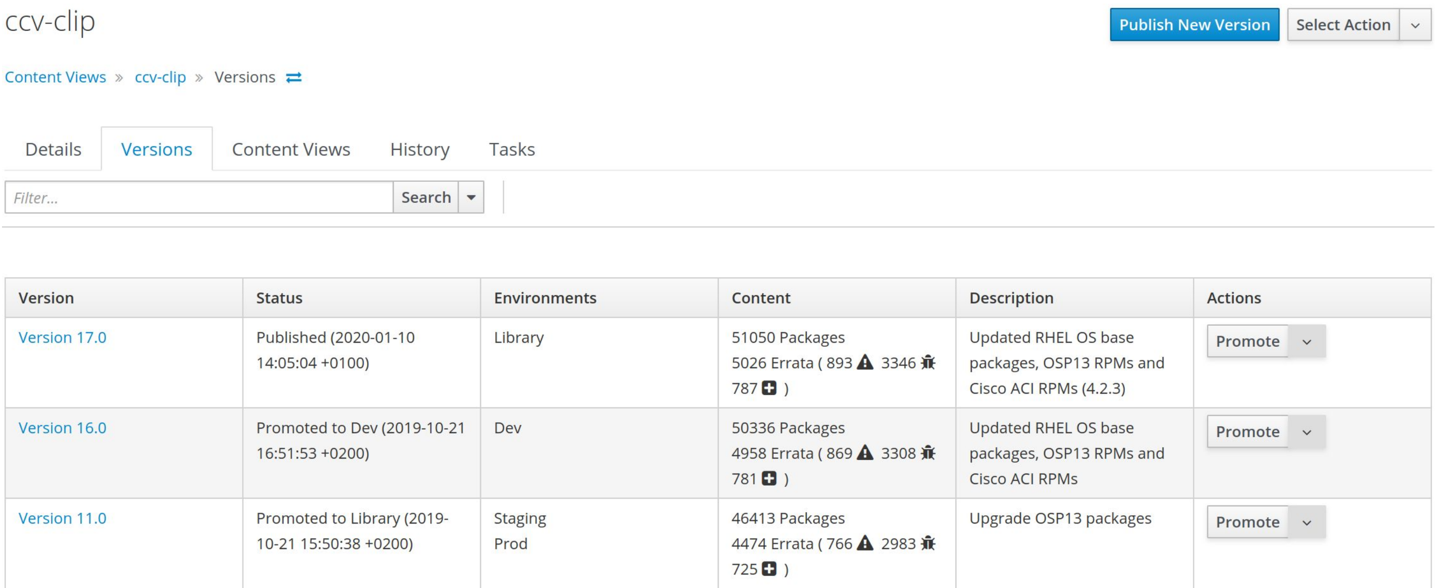
	Content Views	Products	Yum Repositories	Docker Repositories	Packages	Errata
Library	5	7	14	99	52289	5100

Add New Environment

	Dev	Staging	Prod
Content Views	1	1	1
Content Hosts	8	8	199

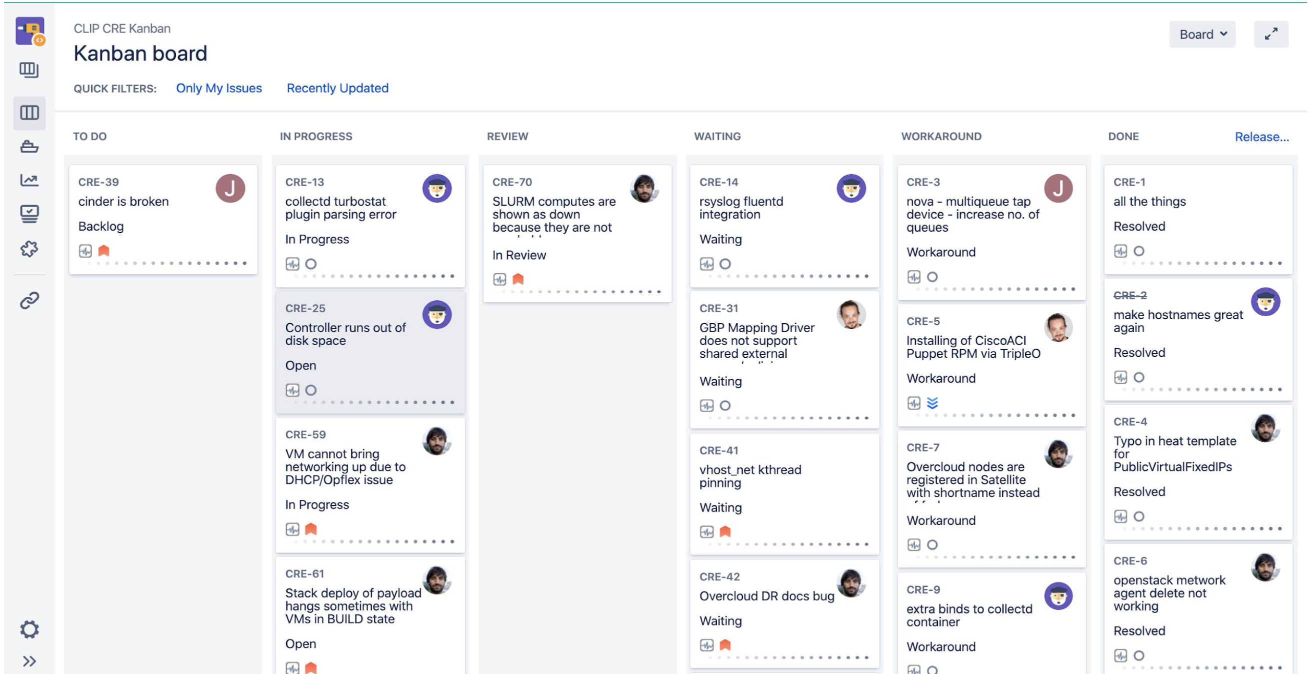
Operating the Cloud - Package Management

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



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)



Deploying and operating the Cloud - Summary

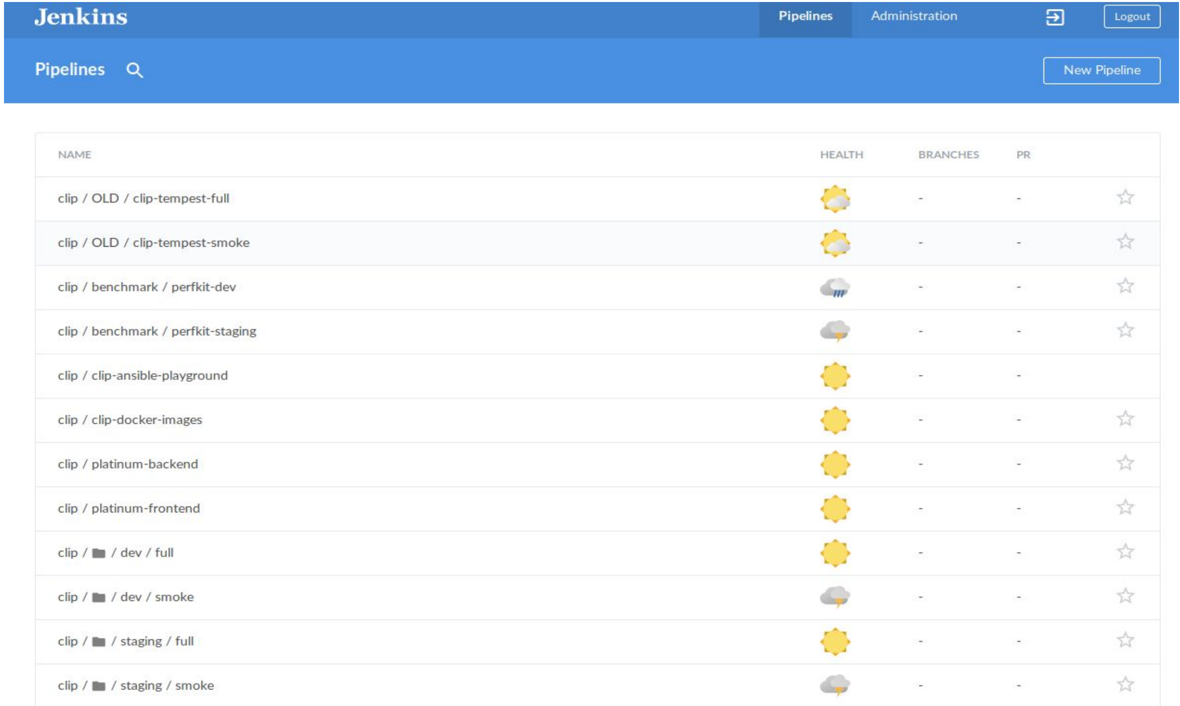
Lessons learned and pitfalls of OpenStack/Tripleo:

- 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

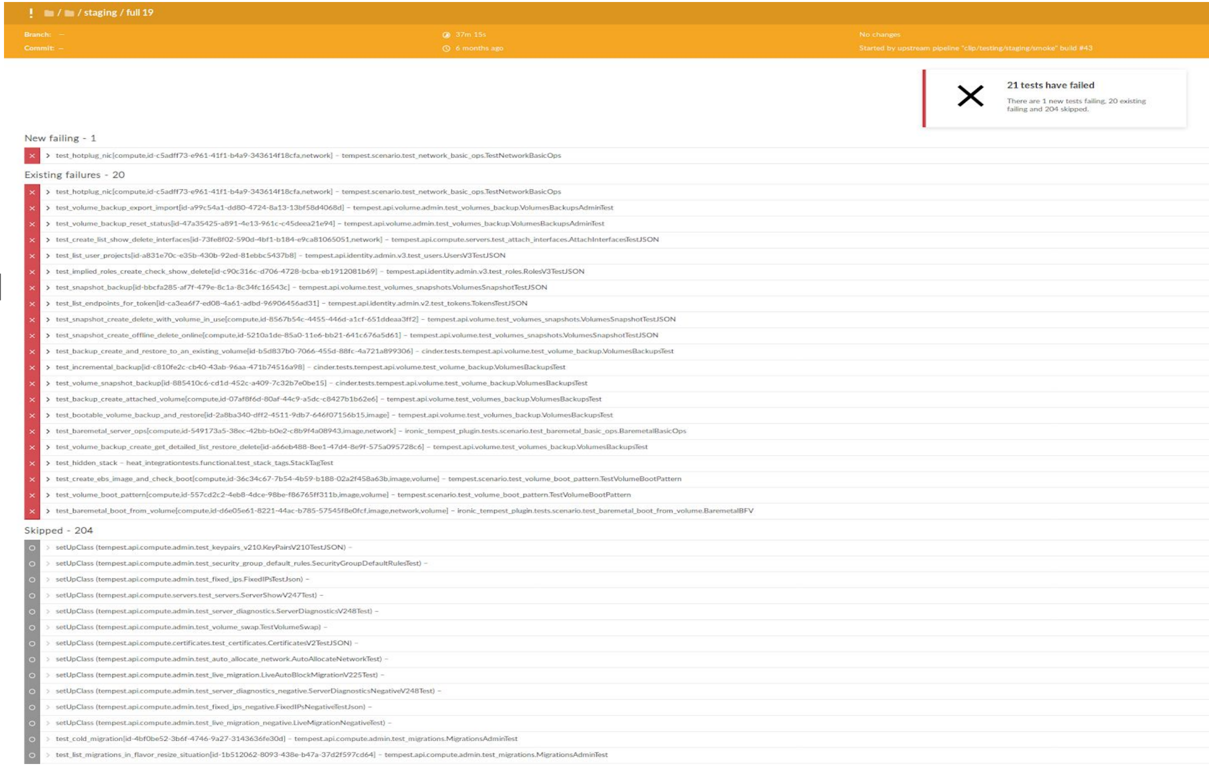
Cloud verification & Performance Testing

- 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.



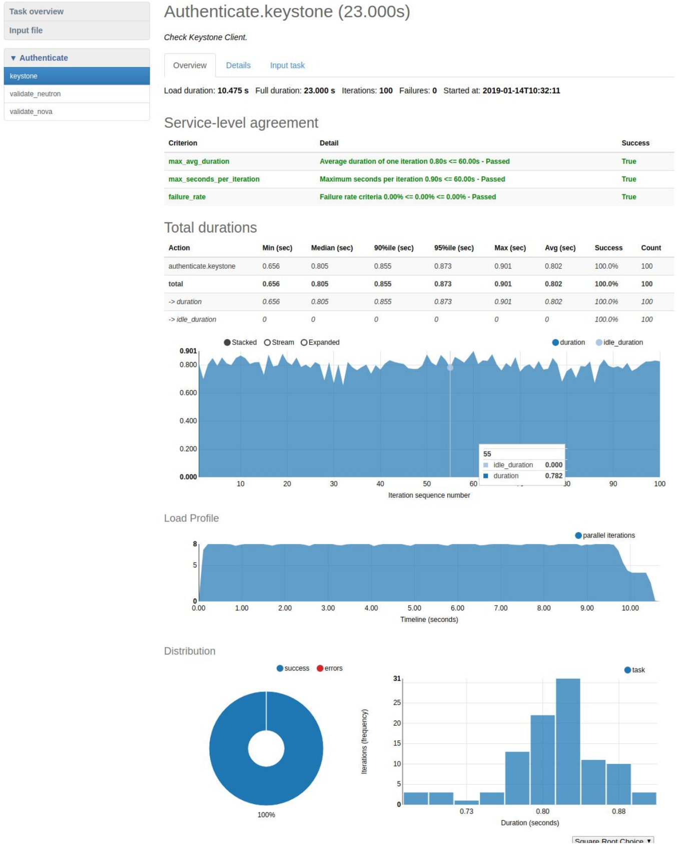
Cloud verification & Performance Testing

- 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.



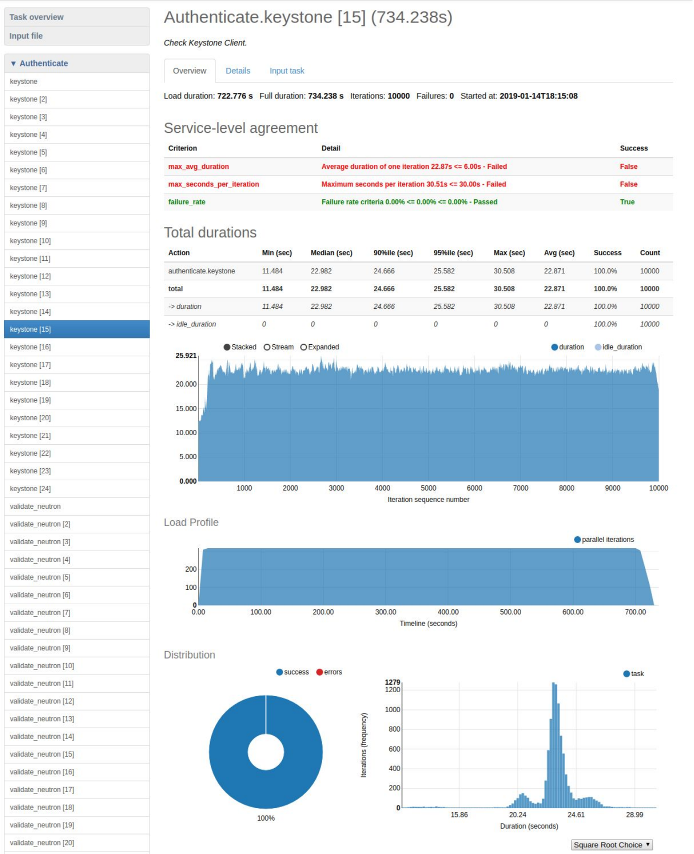
Cloud verification & Performance Testing

- 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



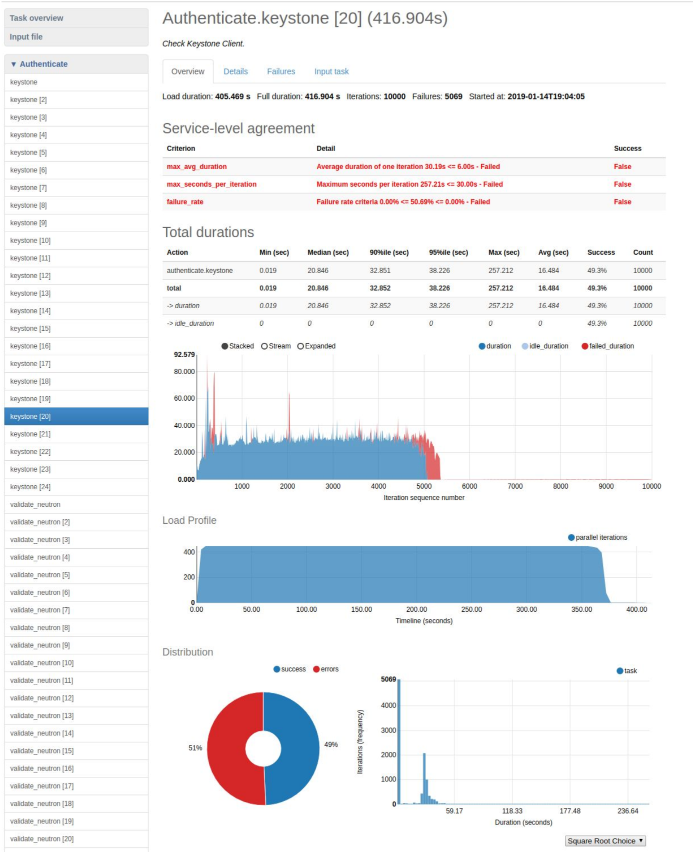
Cloud verification & Performance Testing

- 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



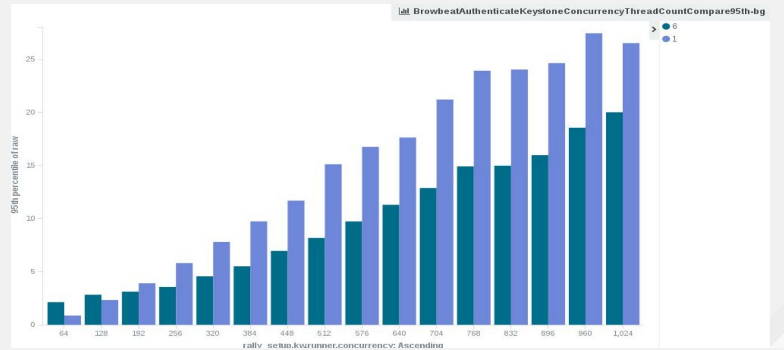
Cloud verification & Performance Testing

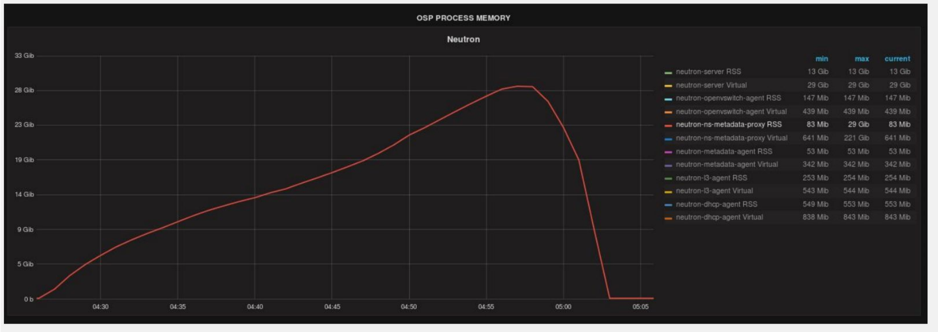
- Ok, the Cloud works but what about performance ? How can we make sure that OS performs when upgrading software packages etc ?
- We plan to use *Browbeat* to run *Rally* (control plane performance/stress testing), *Shaker* (network stress test) and *PerfkitBenchmarker* (payload performance) tests on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





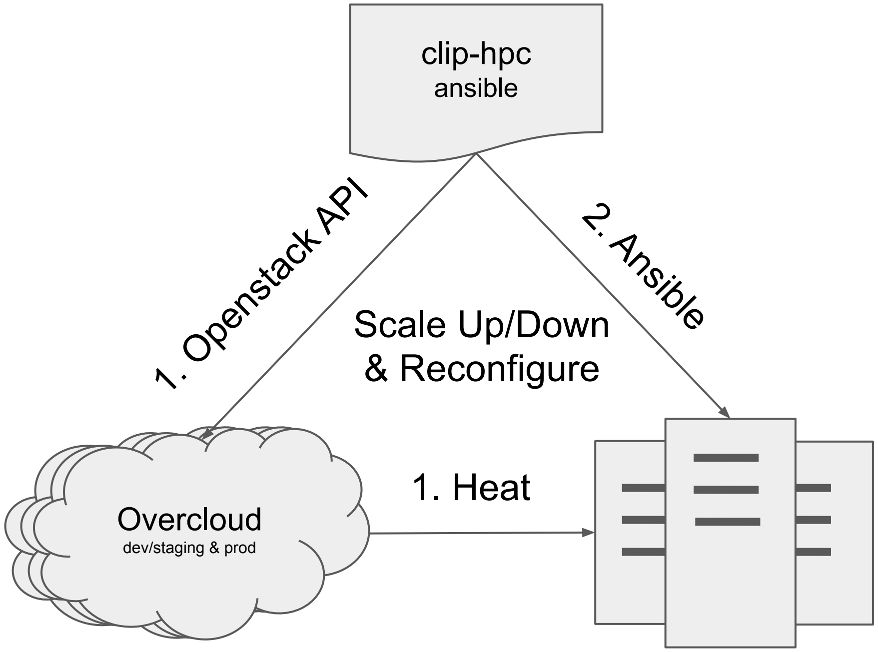
- Browbeat can show differences between settings or software versions:

```
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[1] 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)



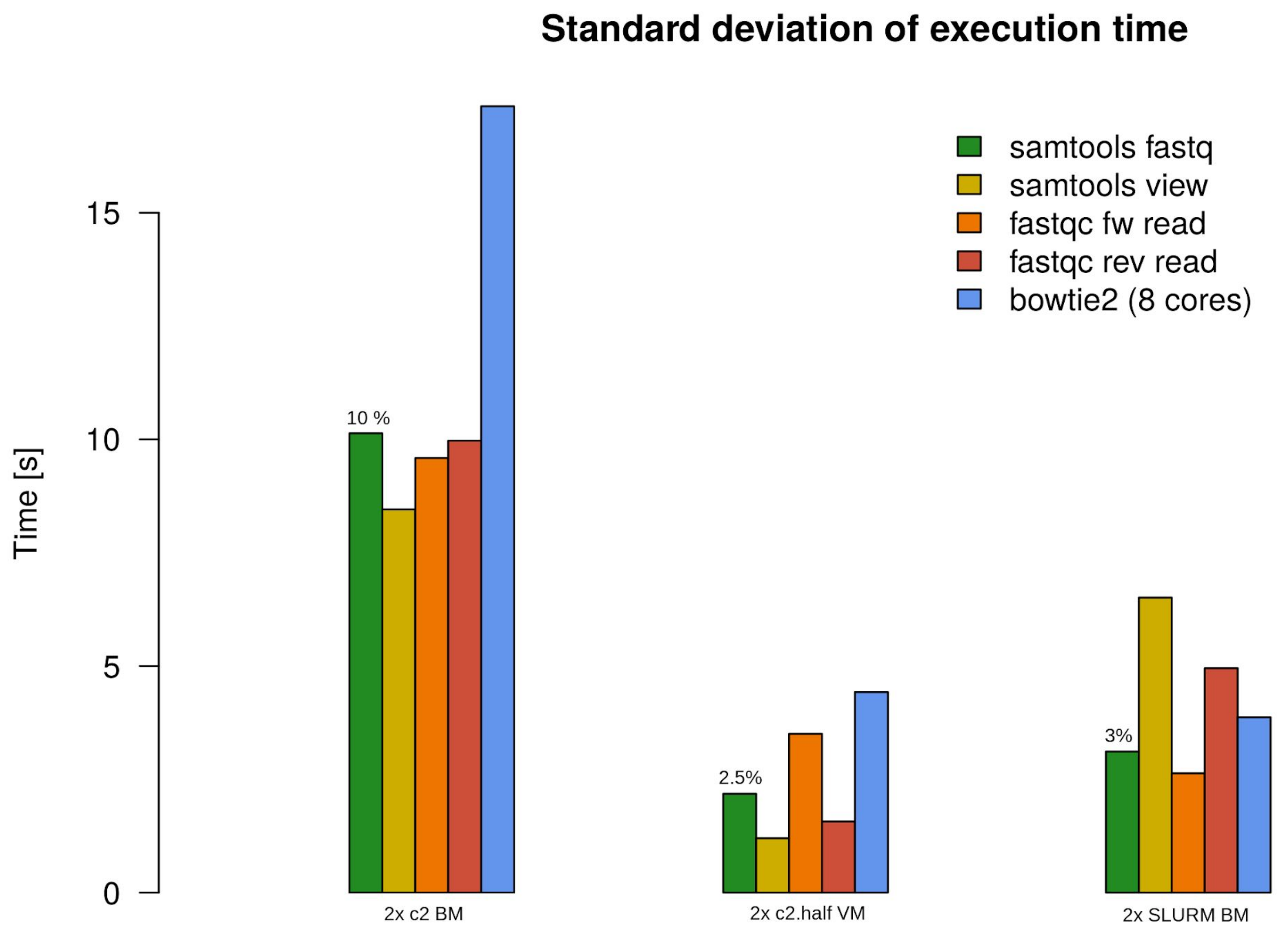
[1] - StackHPC ansible roles: https://github.com/stackhpc

Deploying the Cloud - Tunings for HPC

- Tuning, Tuning, Tuning required for excellent performance

Tuning	Caveats / Downside
NUMA clean instances (KVM process layout)	No live migrations No mixing of different VM flavors
Static huge pages (KSM etc.) setup	If not enough memory is left to hypervisor → swapping or host services get OOM. 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)
Time [s]
400
300
200
100
0
2x c2 BM
2x c2.half VM
2x SLURM BM


Standard deviation of execution time
Time [s]
0
5
10
15
samtools fastq
samtools view
fastqc fw read
fastqc rev read
bowtie2 (8 cores)
10 %
2.5%
3%
2x c2 BM
2x c2.half VM
2x SLURM BM

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