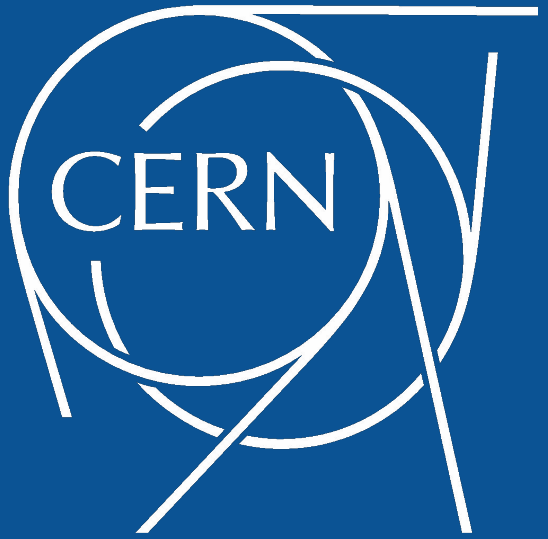
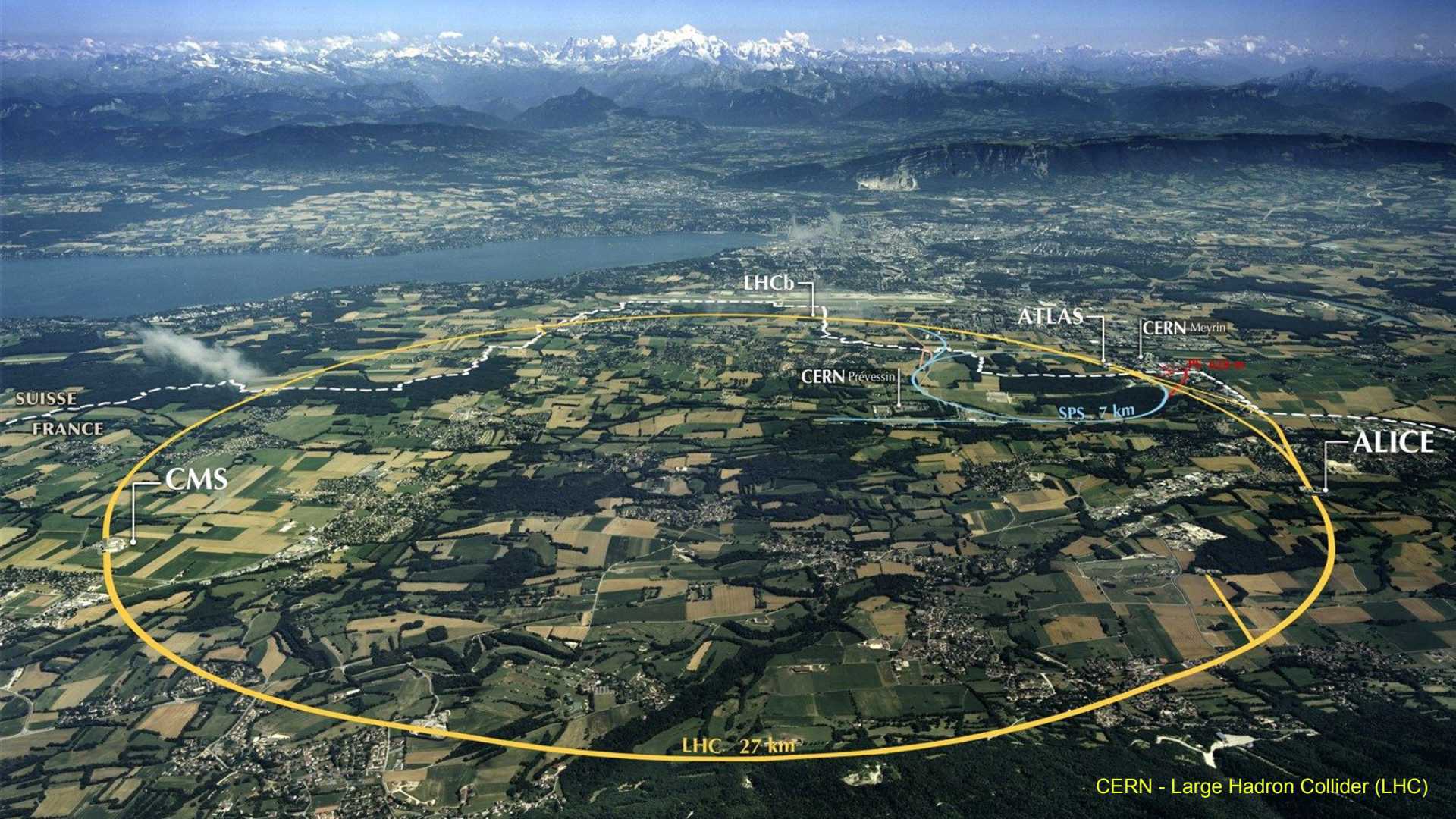

CERN

CERN OpenStack Cloud Control Plane

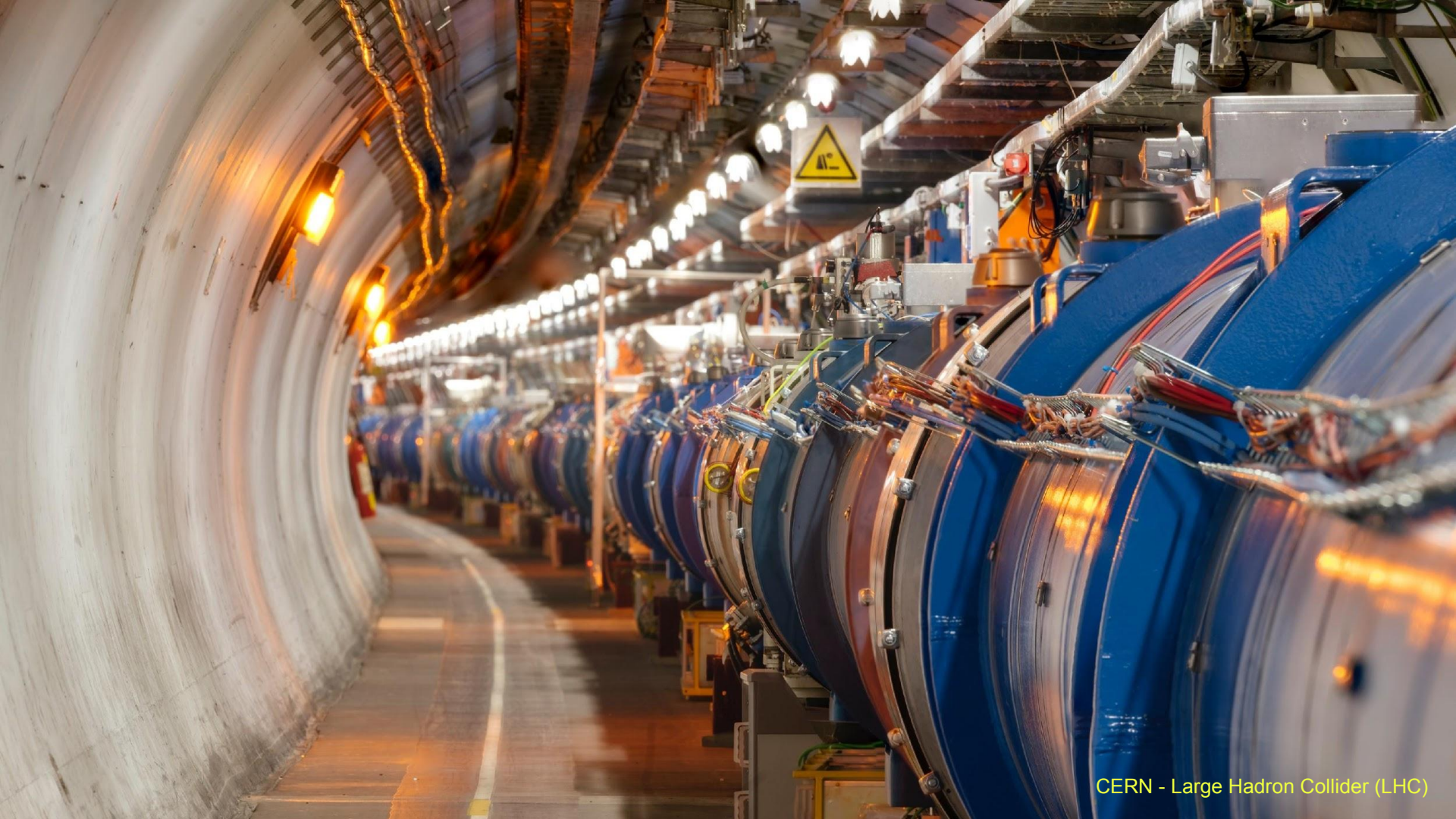
From VMs to K8s

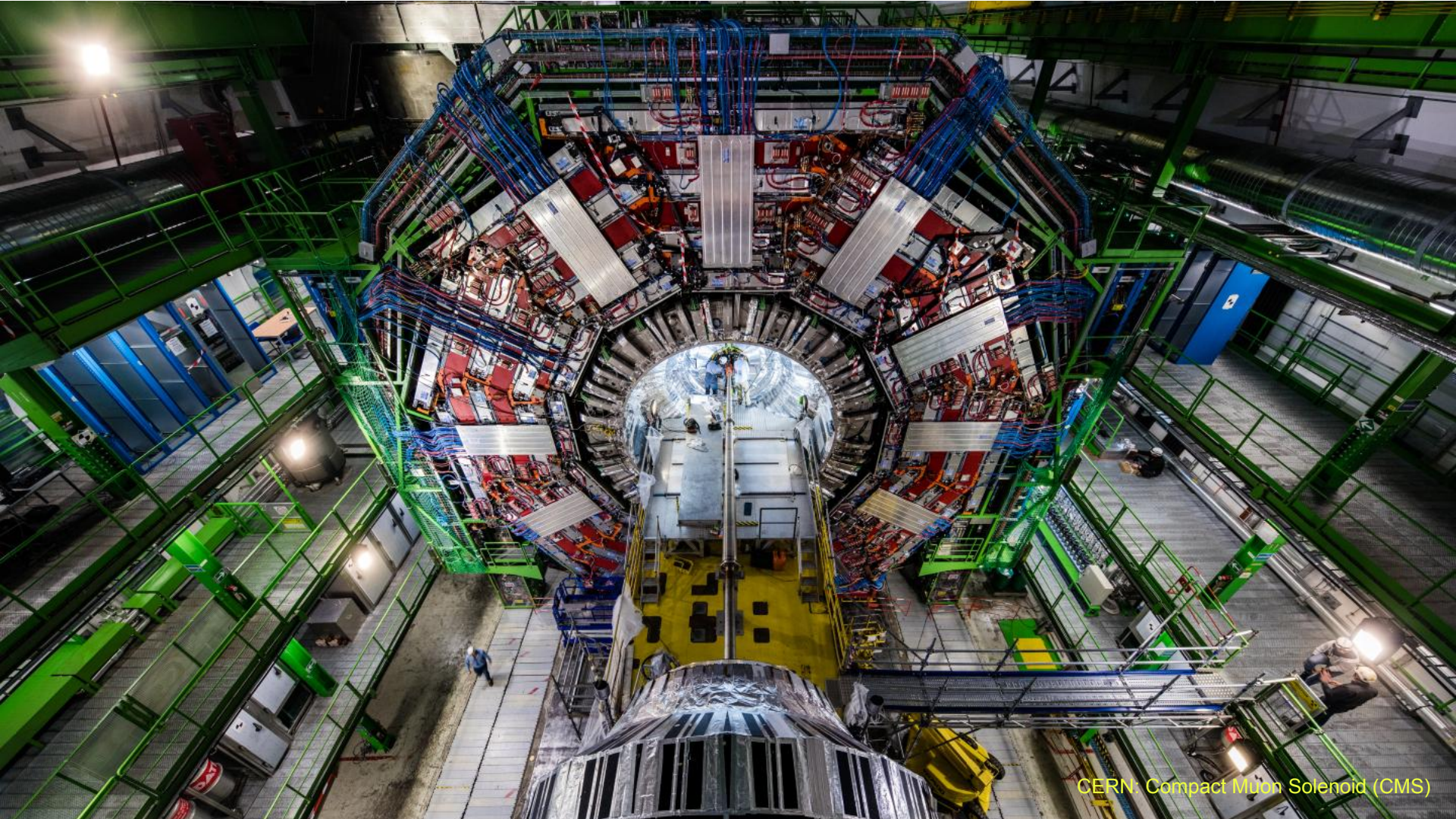
OpenStack Summit - Shanghai 2019

Belmiro Moreira - @belmiromoreira
Spyridon Trigazis - @strigazi



CERN - Large Hadron Collider (LHC)


CERN - Large Hadron Collider (LHC)


CERN: Compact Muon Solenoid (CMS)

Used	Available	Used	Available	Used	Available
291.1 K cores	270.6 K cores	828.4 TiB RAM	918.0 TiB RAM	9.1 PiB disk	16.3 PiB disk

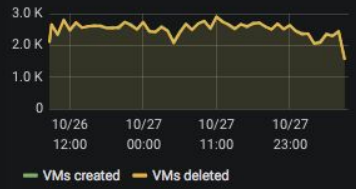
Openstack services stats

Users	Projects	VMs	Magnum clusters	Hypervisors	Images	Baremetal nodes
60758	4359	34509	524	8452	3554	3592

Volumes	Volume size	Fileshares	Fileshares size
6488	1.91 PiB	710	306 TiB

Resource overview by time

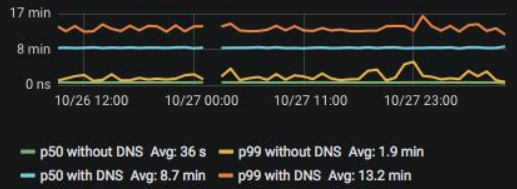
VMs created/deleted



Shared cells availability

Total VMs

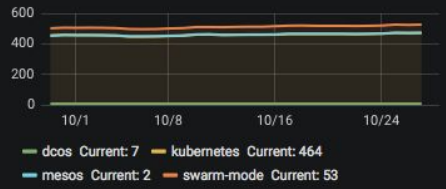
Average VM boot time



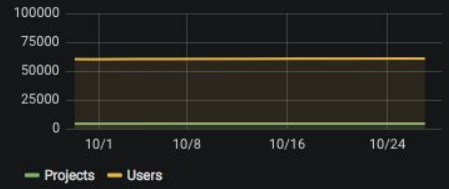
VM changes

Hypervisors

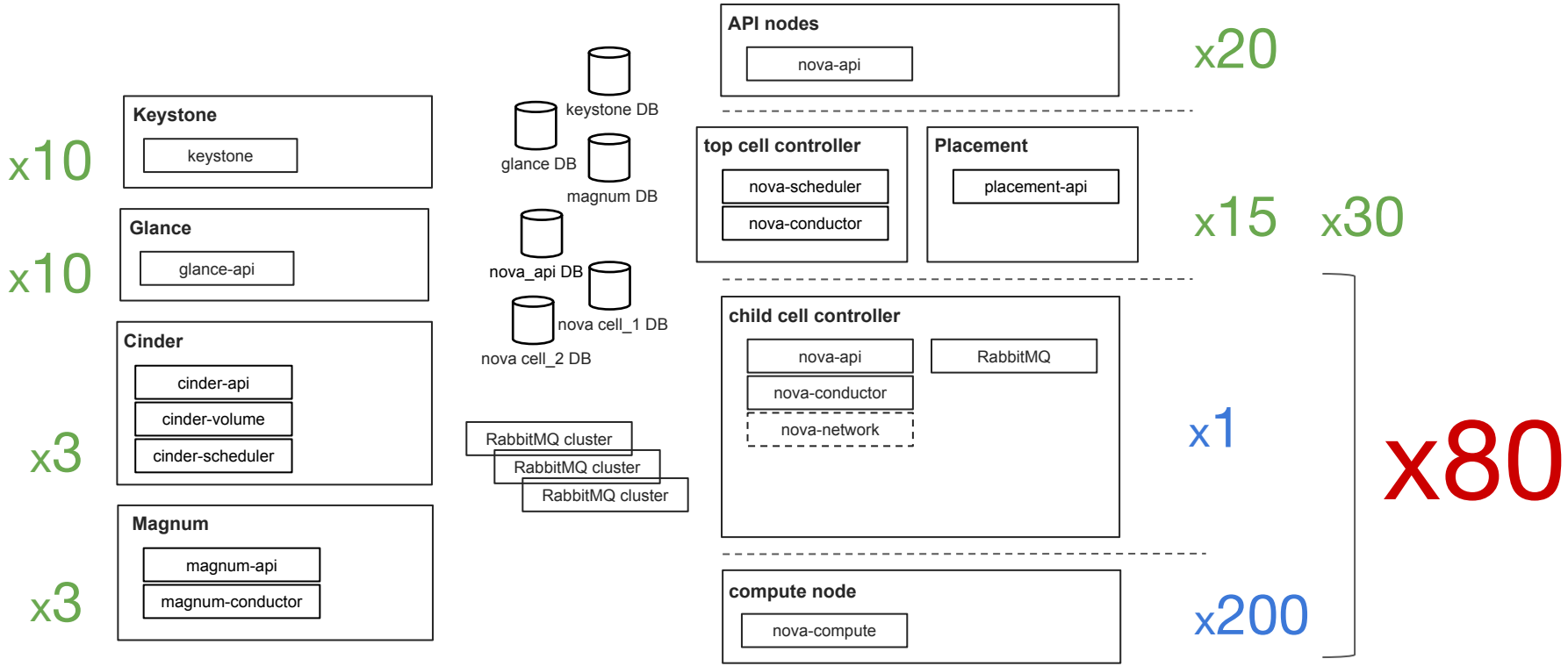
Magnum clusters



Projects and users



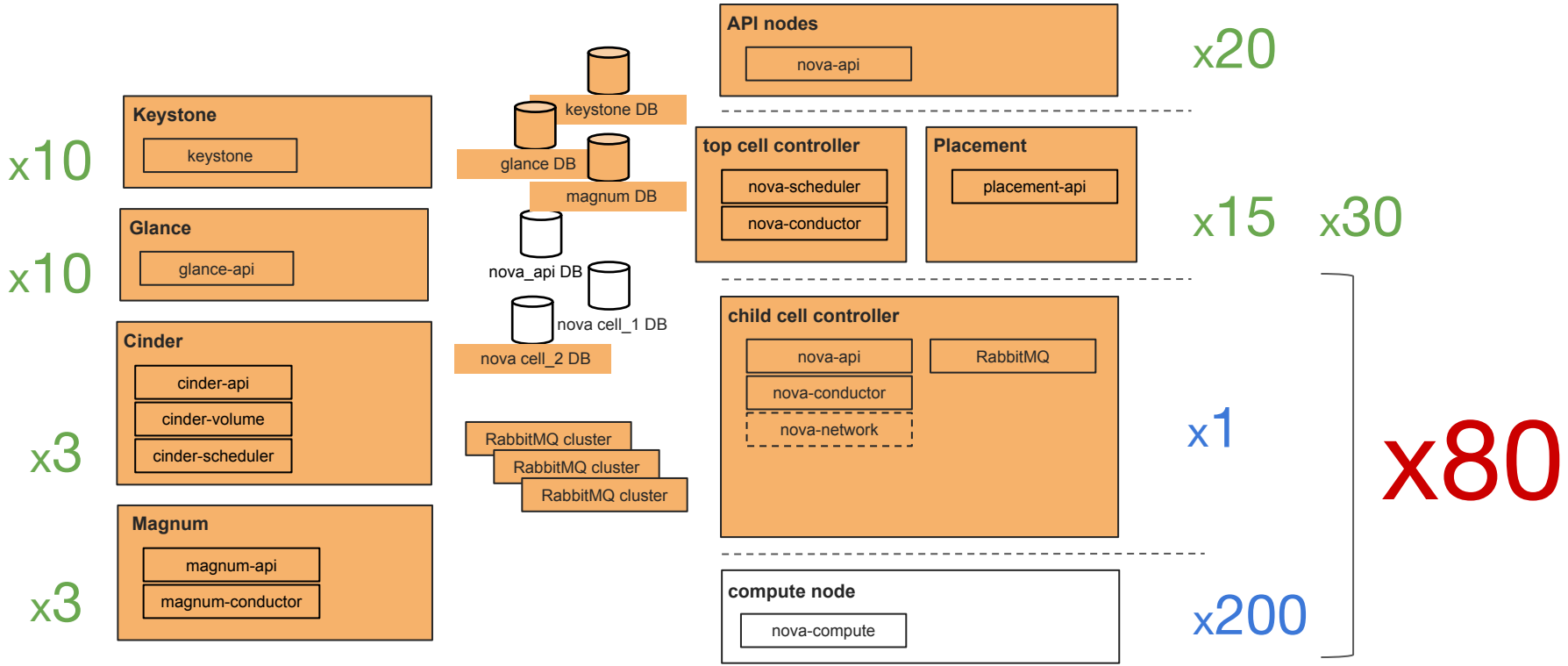
CERN Cloud Architecture (High level view)



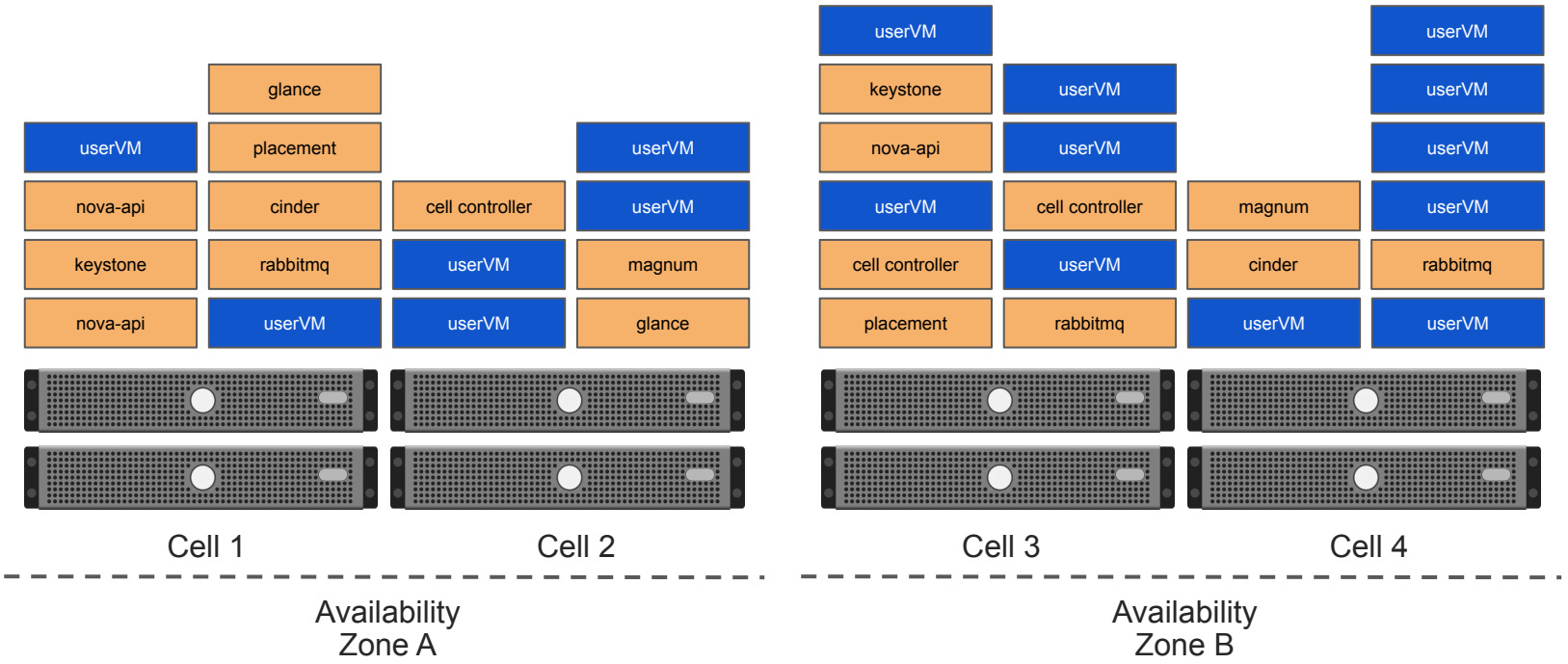
CERN Cloud Control Plane - VMs

- Cloud “inception”
  - The CERN Cloud Control Plane runs in the Cloud that it provisions!
- Advantages
  - Each OpenStack component runs in a different VM
    - keystone; nova-api; nova-conductor; glance-api; rabbitmq …
  - Isolation between components
    - Scale individual components (Add more VMs)
    - Upgrade individual components
  - Use the same configuration management tool (Puppet) as in physical nodes
- Disadvantages
  - Large number of VMs
    - Difficult to manage
    - VM overhead creates unused resources
  - Configuration changes need to propagate into all service VMs

CERN Cloud Architecture (High level view)



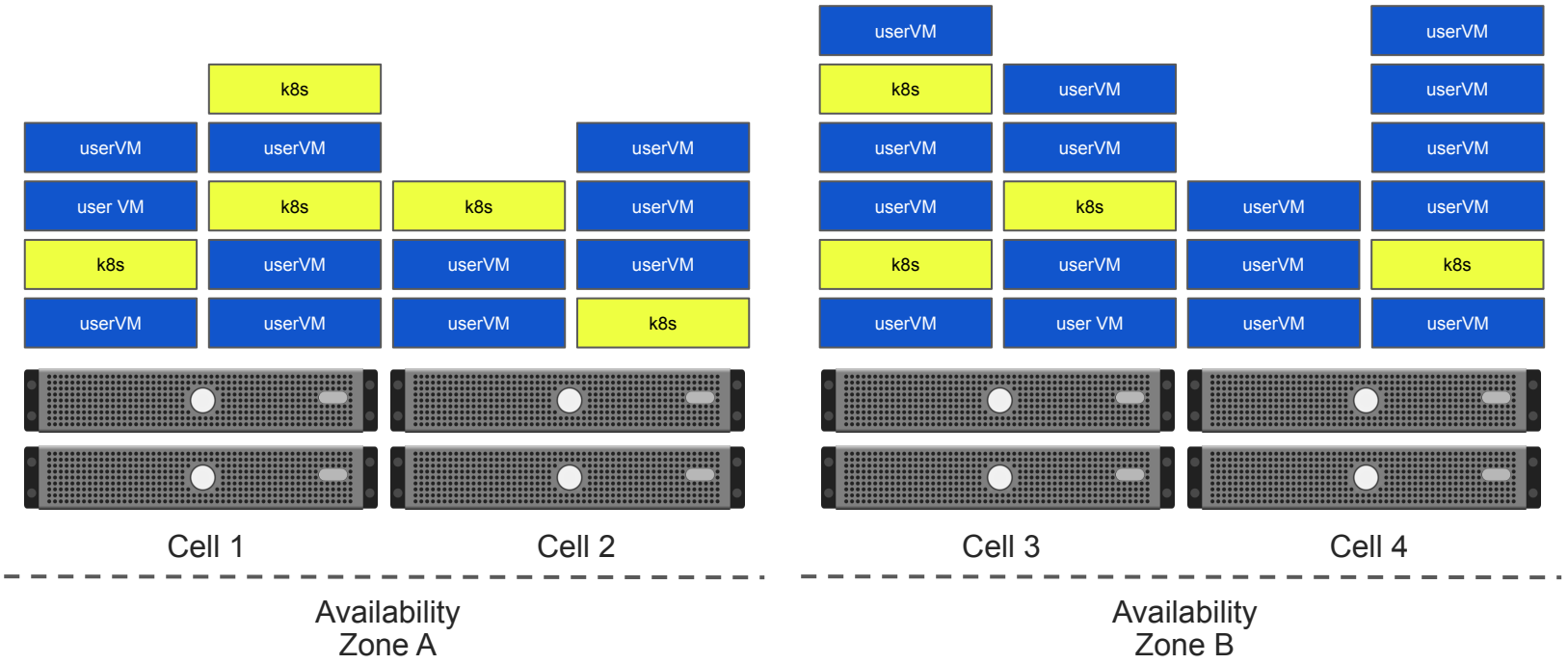
CERN Cloud Architecture - Control Plane



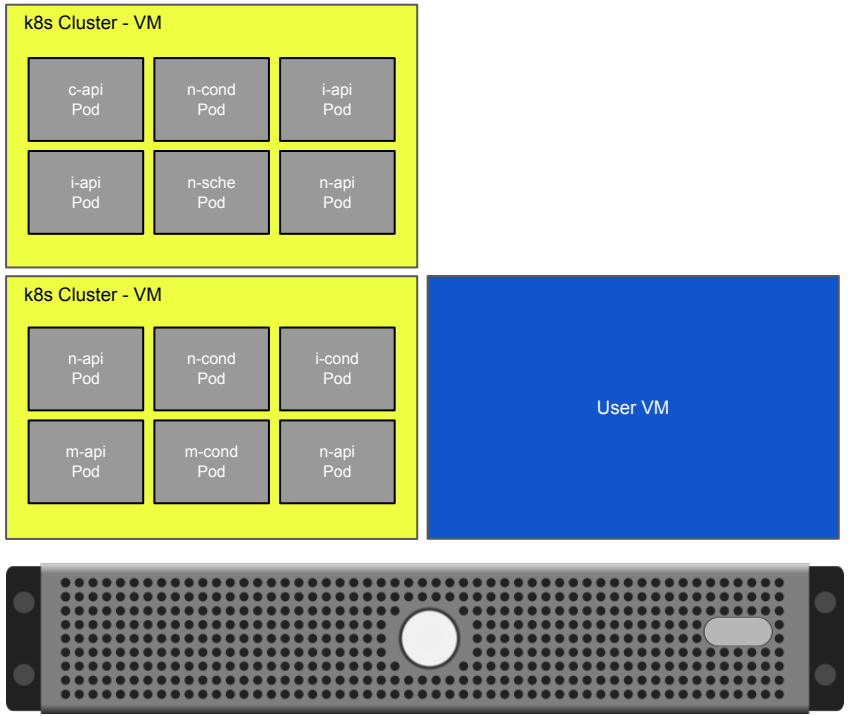
CERN Cloud Control Plane - K8s

- Even more... Cloud “inception”!
- Advantages
  - Strong resource consolidation
  - Service replication and resilience native to the K8s orchestration
  - Accelerate deployment/development iterations (and rollback)
    - Handle faster configuration changes/upgrades when comparing with puppet
  - Cluster footprint scale up/down
  - Native autoscaling
- Disadvantages
  - One more “Inception” layer!
  - All support infrastructure (monitoring, alarming, ...) is still not ready for K8s
  - All staff needs to be trained for K8s

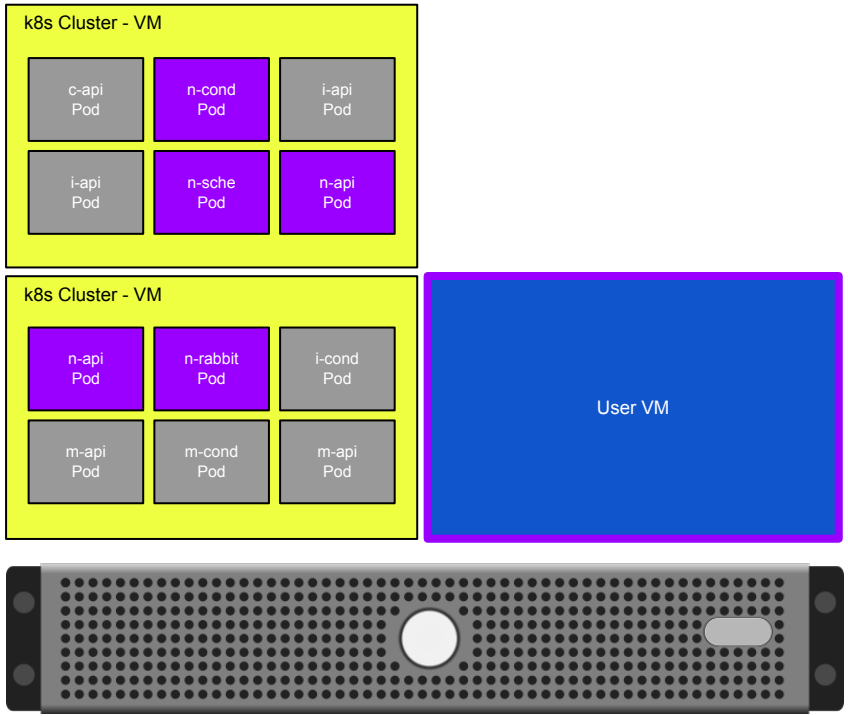
CERN Cloud Architecture - Control Plane



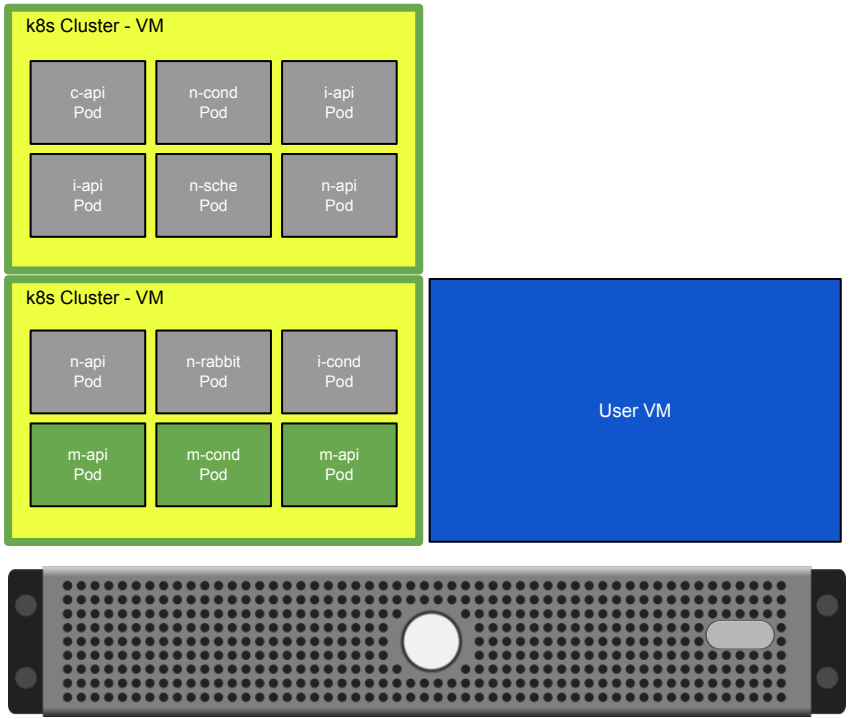
CERN Cloud Architecture - Control Plane



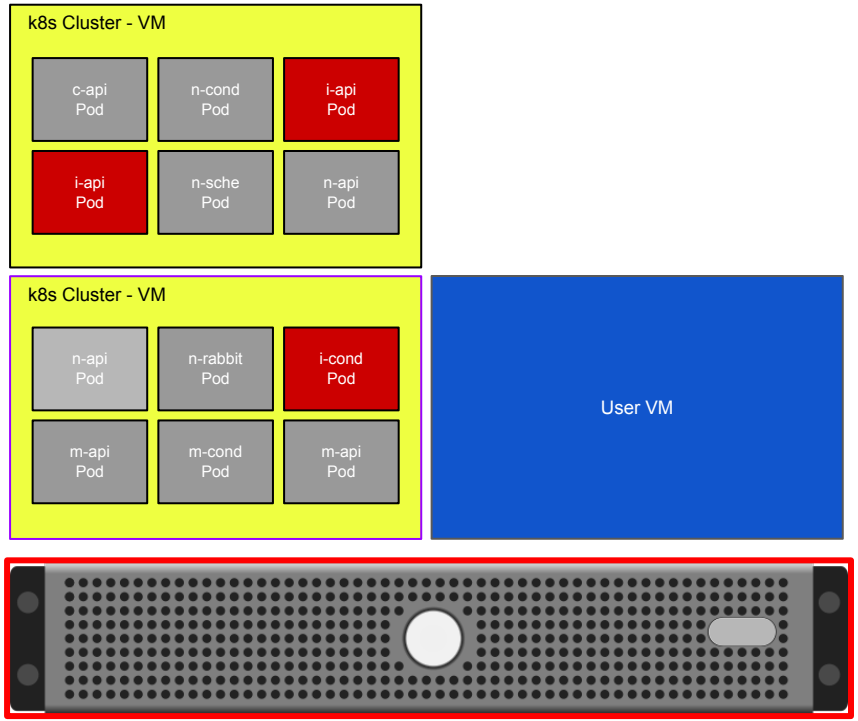
CERN Cloud Architecture - Control Plane



CERN Cloud Architecture - Control Plane



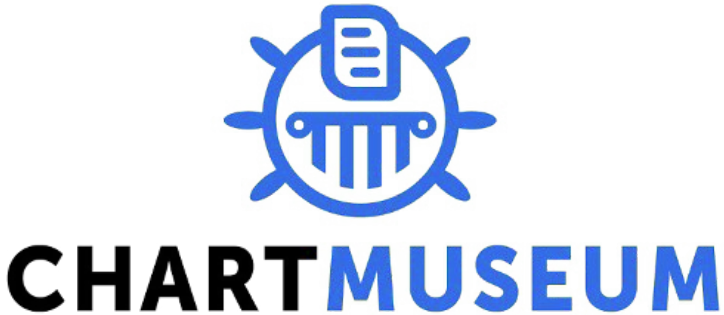
CERN Cloud Architecture - Control Plane



Helm

- The package manager for kubernetes
- Large selection of community managed charts
- Manage only the parameters you need

- Charts stored in s3
- Managed by ChartMuseum



Helm usage (v2)

- Configure client
  - Use secure tiller configuration https://helm.sh/docs/using_helm/#using-ssl-between-helm-and-tiller
- Add chart repositories
- Always inspect the chart contents
- Install charts

```
$ helm init --tiller-tls …
$ helm repo add myrepo https://example.org/
$ helm repo update
$ helm dependency update
$ helm template <path to chart>
$ helm install myrepo/myapp --name myapp_name -f values.yaml
```

OpenStack Helm

- One helm chart per service
- git repos openstack/openstack-helm and openstack/openstack-helm-infra
- 20 repos in openstack-helm
- 46 repos in openstack-helm-infra

Secret Management Requirements

- Offer a gitops style solution, with encrypted secrets version controlled along the rest of the application configuration data
- Allow usage of **unchanged** upstream helm charts
- Provide good integration with existing helm commands install, upgrade, …
- Secure, central store for encryption keys
  - Use existing infrastructure
  - Use existing AuthN/AuthZ

Helm Barbican Plugin

Barbican

- Key Manager OpenStack API service
- types: generic, certificate, RSA
- OpenStack credentials (kerberos for CERN)

Helm plugin

- Written in go
- Wrapper for install, upgrade, lint
- Edit secrets in memory, write to fs encrypted

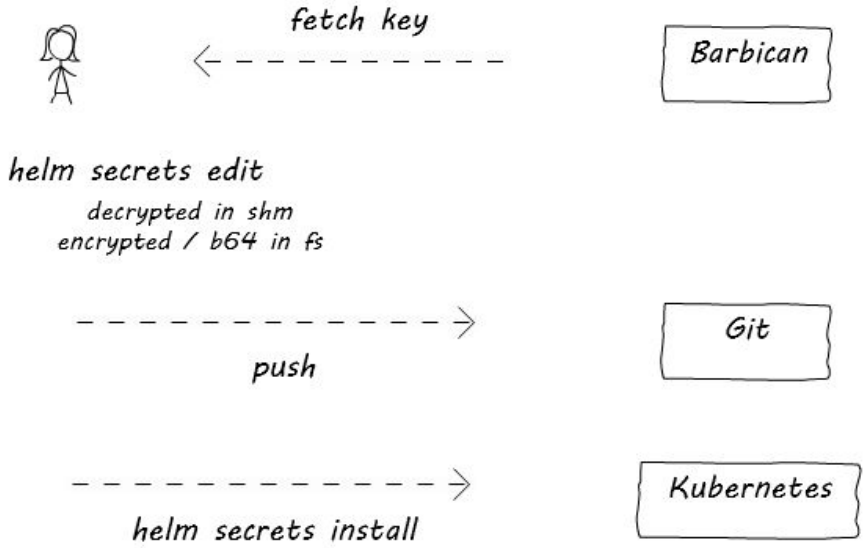


Image Credit: Ricardo Rocha, CERN Cloud

Secrets plugin usage

```
$ helm secrets -h
Secret handling using OpenStack Barbican.
     Secrets are stored encrypted in local files, with the key being stored in
     Barbican. These files can be safely committed to version control.

Usage:
  secrets [command]

Available Commands:
  dec         decrypt secrets with barbican key
  edit        edit secrets
  enc         encrypt secrets with barbican key
  help        Help about any command
  install     wrapper for helm install, decrypting secrets
  lint        wrapper for helm lint, decrypting secrets
  upgrade     wrapper for helm upgrade, decrypting secrets
  view        decrypt and display secrets
```

Secrets plugin usage cont’d

```
$ helm secrets view service/secrets.yaml
conf:
  service:
    DEFAULT:
      auth_key: somekey
endpoints:
  identity:
      service:
        password: somepass
$ helm secrets install --name service ./service -f service/secrets.yaml \
  -f service/values.yaml --version 0.0.2
  ...
$ helm secrets edit service/secrets.yaml
$ helm secrets upgrade service ./service -f service/secrets.yaml \
  -f service/values.yaml --version 0.0.2
  ...
```

OpenStack LOCI

- OpenStack LOCI is a project designed to quickly build Lightweight OCI compatible images of OpenStack services
- Several projects supported
  - Nova
  - Glance
  - Heat
  - …
- OpenStack-Helm uses OpenStack-LOCI
- We require custom images because the all the internal patches specific to the CERN Infrastructure
  - Very easy to build local custom images

OpenStack LOCI

- CentOS is supported as base image

```
docker build \
https://opendev.org/openstack/loci.git#master:dockerfiles/centos \
--tag loci-base:centos
```

- Easy to use a custom OpenStack Project repo. Many other options available

```
docker build \
https://opendev.org/openstack/loci.git \
--build-arg PROJECT=nova \
--build-arg PROJECT_REPO=<YOUR_CUSTOM_REPO> \
--build-arg WHEELS="loci/requirements:master-centos" \
--build-arg FROM=loci-base:centos \
--build-arg PROJECT_REF=cern_stein \
--build-arg DIST_PACKAGES="httpd mod_wsgi python2-ldap python2-suds" \
--tag <YOUR_CUSTOM_IMAGE_TAG>
```

Use Case 1 - Glance on K8s

- How OpenStack HELM deploys Glance?

```
helm fetch --untar --untardir . 'openstack/glance'
helm template glance
```

- We would like to integrate the K8s Glance in the current Infrastructure
  - Not build a different deployment from scratch
  - OpenStack HELM is great to build an all in one OpenStack Cloud
  - We would like to have a more controlled initial experience

Use Case 1 - Glance on K8s

- What is needed to deploy Glance on K8s? The basics...
  - Image (LOCI)
  - “ConfigMap” for the configuration file; policy and start the service
  - “Deployment” for the glance-api pod
  - “Service” for port 9292
- How about the secrets?
  - OpenStack can load several configuration files
  - Dedicated configuration file only for the secrets
    - Glance DB password, transport URL for notifications, service accounts
- How about ingress?
  - ngnix Ingress
  - Deployed with HELM

Use Case 1 - Glance on K8s

- What's different from the OpenStack HELM charts?
  - Used the OpenStack HELM template to built it...
  - But… a very simplified version!
  - Configuration/policy is not deployed as a secret
    - Allows to have the config file in git
    - The same configuration file as production
  - Only Glance and CEPH credentials are secrets
- Very easy to understand and deploy!
- How we deploy it?
  - Everything stored on Git but deployed manually
    - No GitOps for now
  - Ingress added into the production HAProxy
- Currently both deployments (VMs and K8s) run in parallel

Use Case 1 - Glance on K8s

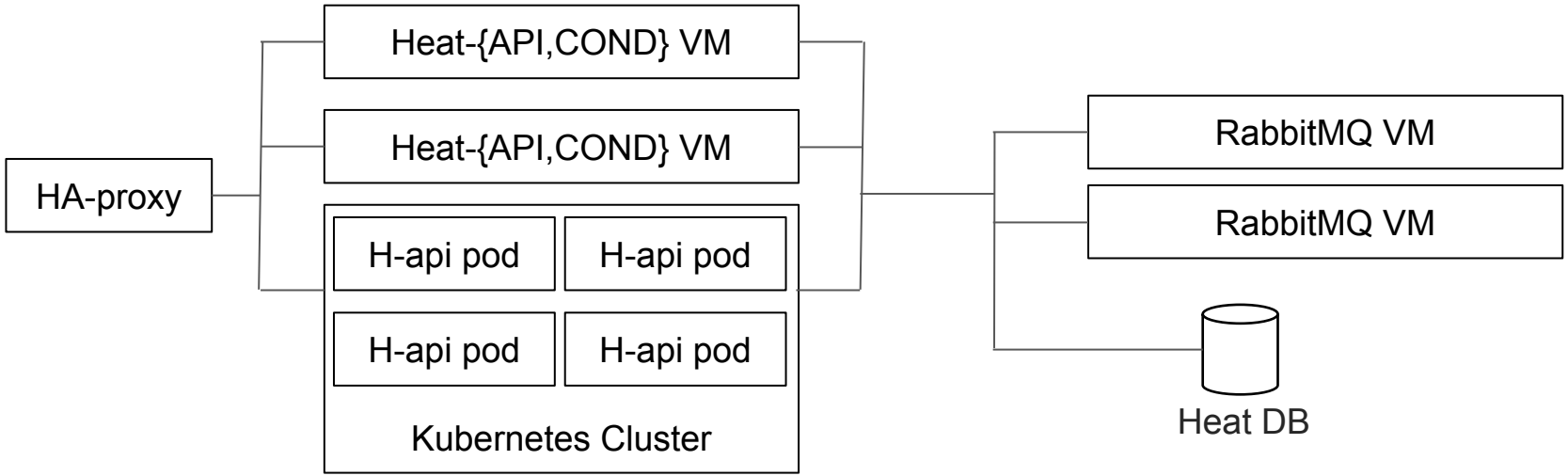
```
$kubectl get pods
NAME                                                     READY   STATUS    RESTARTS   AGE
braided-skunk-nginx-ingress-controller-kglzz             1/1     Running   0          95d
braided-skunk-nginx-ingress-controller-vzgcn             1/1     Running   0          95d
braided-skunk-nginx-ingress-default-backend-68f4755546-rfrr9   1/1     Running   0          95d
glance-api-f686d7cbb-rdw7w                               1/1     Running   0          95d


kubectl get configmaps
NAME                             DATA   AGE
braided-skunk-nginx-ingress-tcp   1      120d
glance-bin                       7      120d
ingress-controller-leader-nginx   0      120d


$kubectl logs -f glance-api-f686d7cbb-rdw7w
(...)
```

Use Case 2 - Heat on K8s

- Deploy in parallel with VMs a la glance
- Stock loci image from docker.io/openstackhelm
- Stock Helm Chart replicated in our ChartMuseum
- External puppet-managed rabbit and DB



Use Case 3 - New Region

- New region requirement
  - Ideal for All-In-One with OpenStack-Helm
- Isolated environment
  - No access to container registry
  - No managed storage
  - No access to puppet
- Small scale
  - Well defined use-case
- Kubernetes on demand for users

Use Case 3 - New Region Architecture

- Single five node kubernetes cluster
  - Manual Deployment with kubeadm
    - Manual import of container images
  - Kubespray has a lot of dependencies on external images
- Self-contained storage
  - Openebs for glance and ‘Registry’
- Self-contained container registry
- External Database
- Requires Glance, Nova, Neutron, Heat, Magnum

Conclusion

- Compact configuration with Helm values
  - Common logging and rabbit configuration
- OpenStack Helm can build a cloud out of the box!
  - Ideal for new deployments
  - Large collection of OS charts available
- OpenStack Helm is challenging for a large deployment
  - No external secret management (eg sealed secrets)
  - Strong dependencies on infra charts
  - Helm 3

Next Steps

- Continue to evaluate different tools
  - Helm3, Kustomize, FluxCD
- GitOps: Automated deployments with FluxCD
- Integrate logging and metrics monitoring
- kustomize with different overlays
- Service Mesh
  - Linkerd vs Istio VS Maesh

@belmiromoreira

@strigazi

www.cern.ch