A Kubernetes cloud provider for VISA

Rodrigo Cabezas Quirós

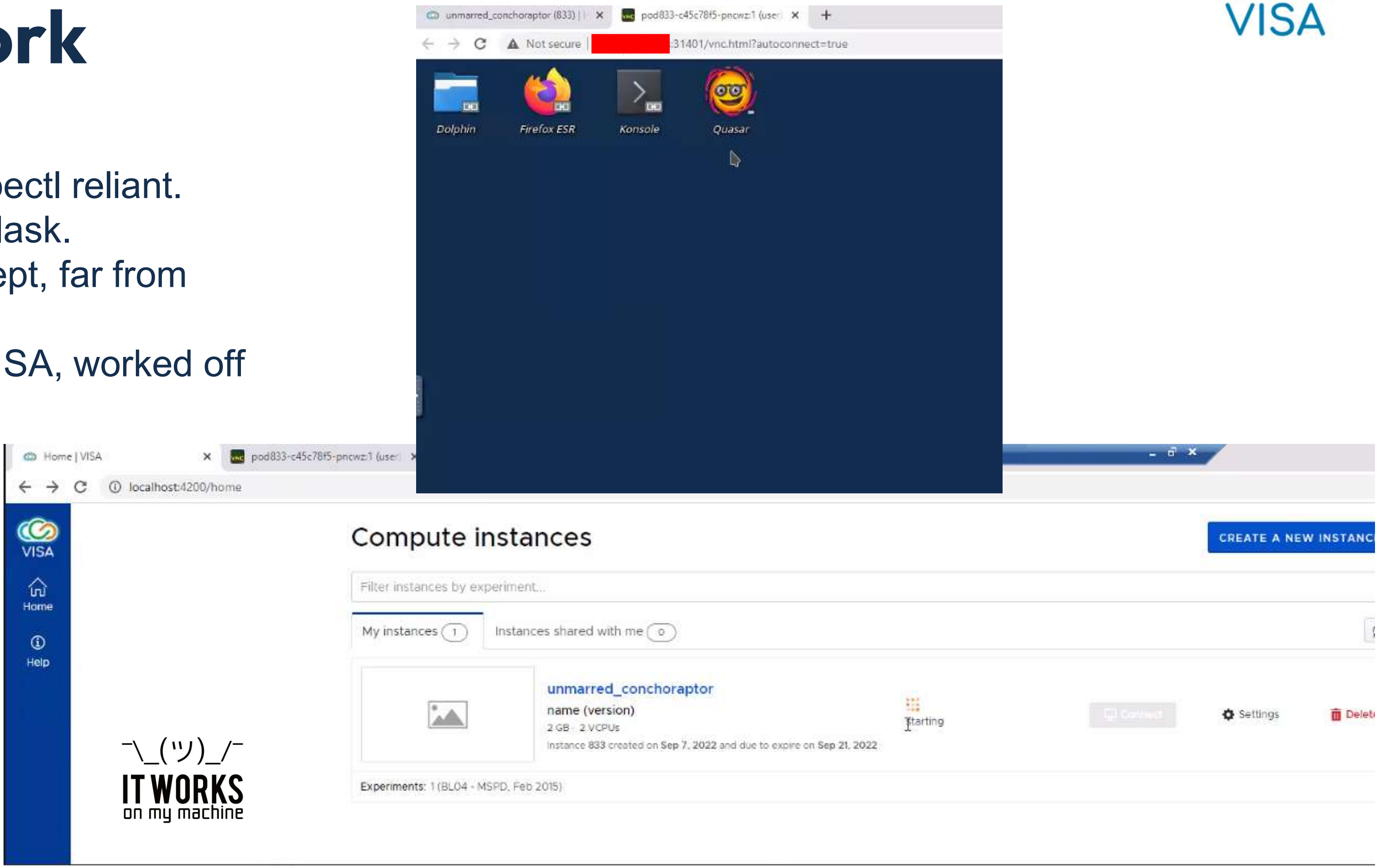
June 2nd, 2025 — PaN Facilities meeting 2025 @ ALBA Synchrotron

Table of contents

- Previous work.
- New solution.
- Implementation.
- Demo.
- Future work.

Previous work

- Circa ~2021.
- YAML based and kubectl reliant.
- Built in Python with Flask.
- Simple proof of concept, far from production grade.
- Not integrated with VISA, worked off noVNC.
- Abandoned in 2023.



unmarred_conchoraptor (833) | pod833-c45c7bf5-pncwz:1 (user

Not secure | [redacted]:31401/vnc.html?autoconnect=true

Home | VISA pod833-c45c78f5-pncwz:1 (user

localhost:4200/home

Compute instances

CREATE A NEW INSTANC

Filter instances by experiment...

My instances (1) Instances shared with me (0)

unmarred_conchoraptor

name (version)

2 GB · 2 VCPUs

Instance 833 created on Sep 7, 2022 and due to expire on Sep 21, 2022

Starting

Connect Settings Delet

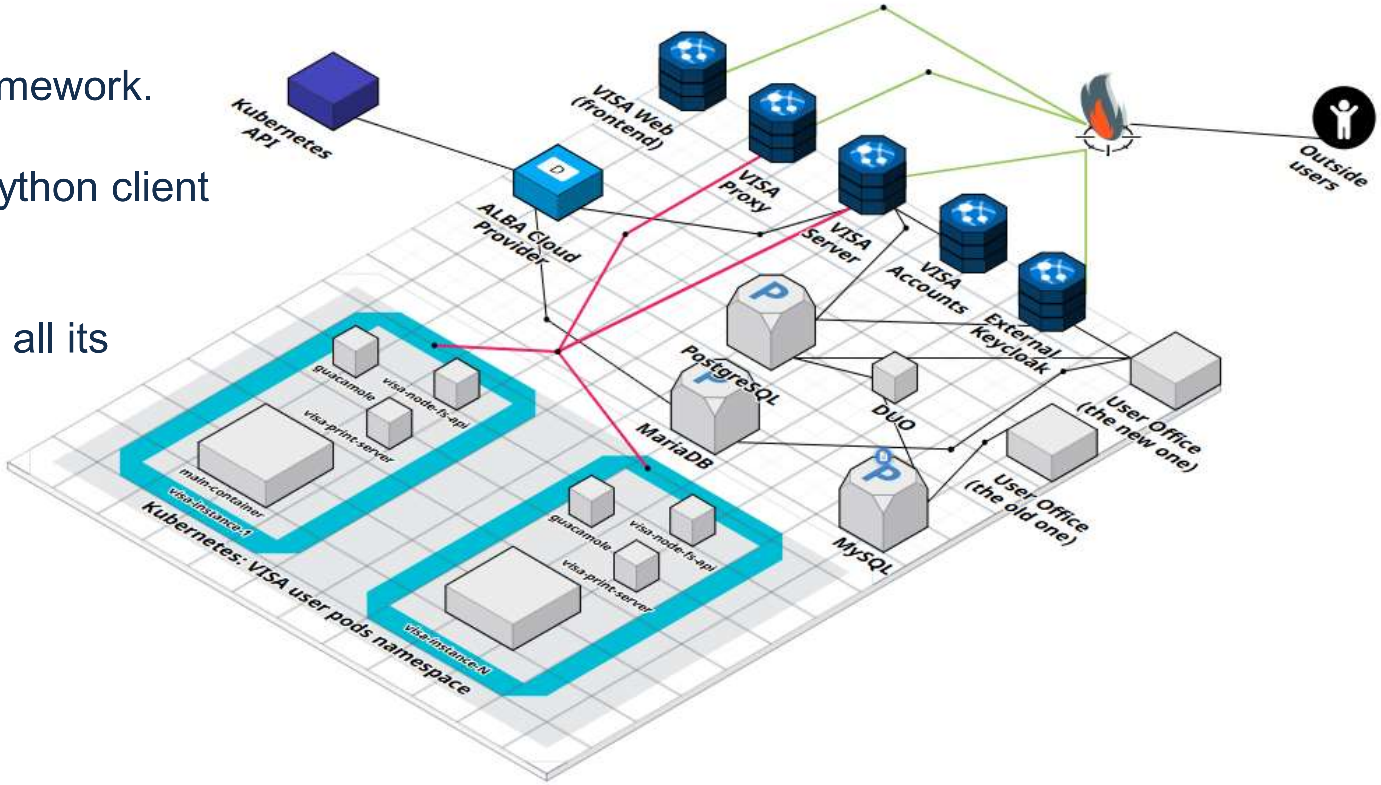
Experiments: 1 (BL04 - MSPD, Feb 2015)

¯_(ツ)_/¯

IT WORKS on my machine

New solution*

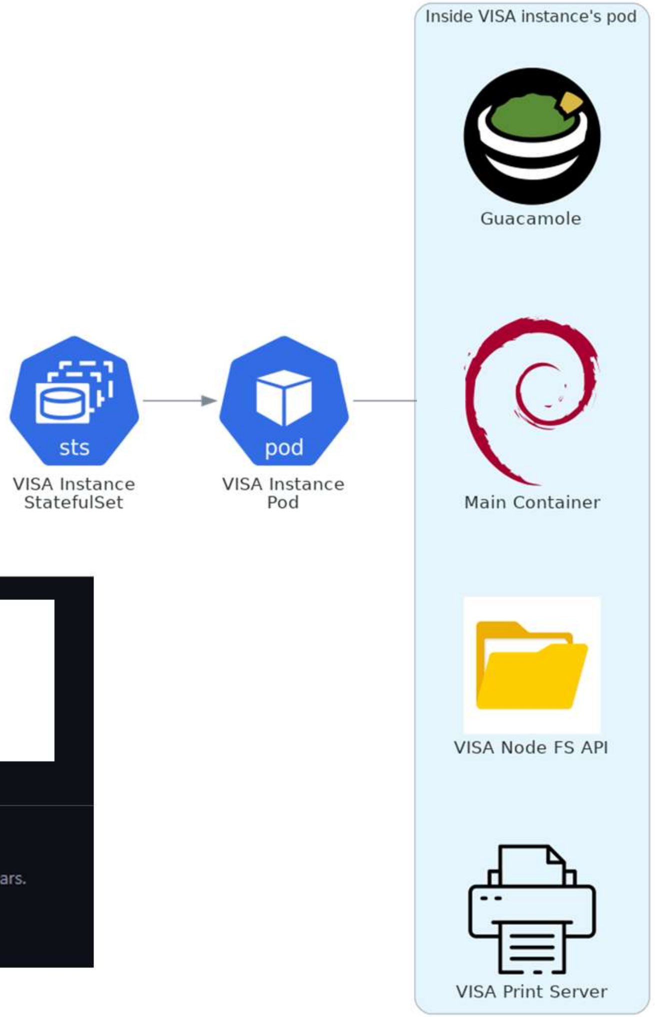
- *Solution still the same, but re-implemented from scratch.
- From ~2024 until now.
- Python and Django REST Framework.
- MariaDB database.
- 100% reliant on Kubernetes Python client (no more YAMLs).
- Leverage k8s features.
- Fully integrated with VISA and all its services.

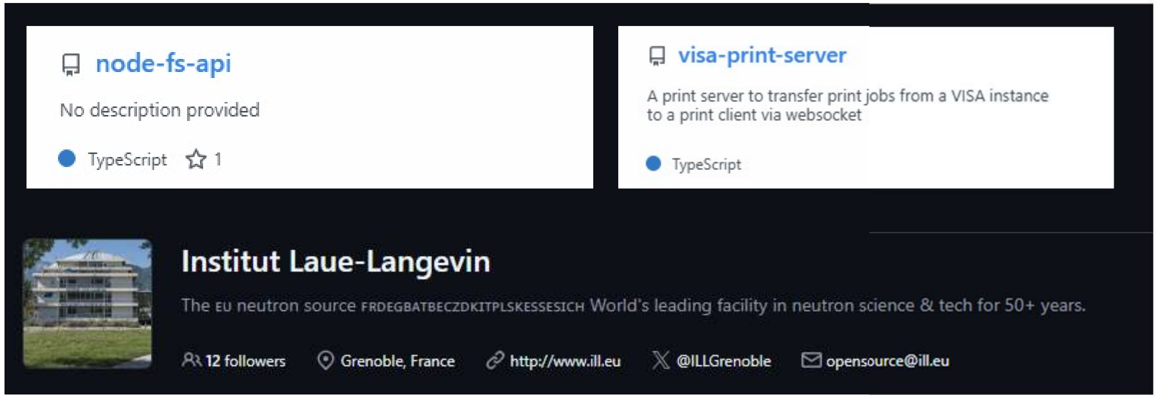


Instance architecture

Implementation

- VISA instances as ~~Deployments~~ StatefulSets.
- Dedicated namespace for visa instances.
- Avoid monolithic instances, leverage multi-containers pods.
- Ease image's complexity, achieve faster startups and easier fixable breakups.





node-fs-api

No description provided

● TypeScript ☆ 1

visa-print-server

A print server to transfer print jobs from a VISA instance to a print client via websocket

● TypeScript

Institut Laue-Langevin

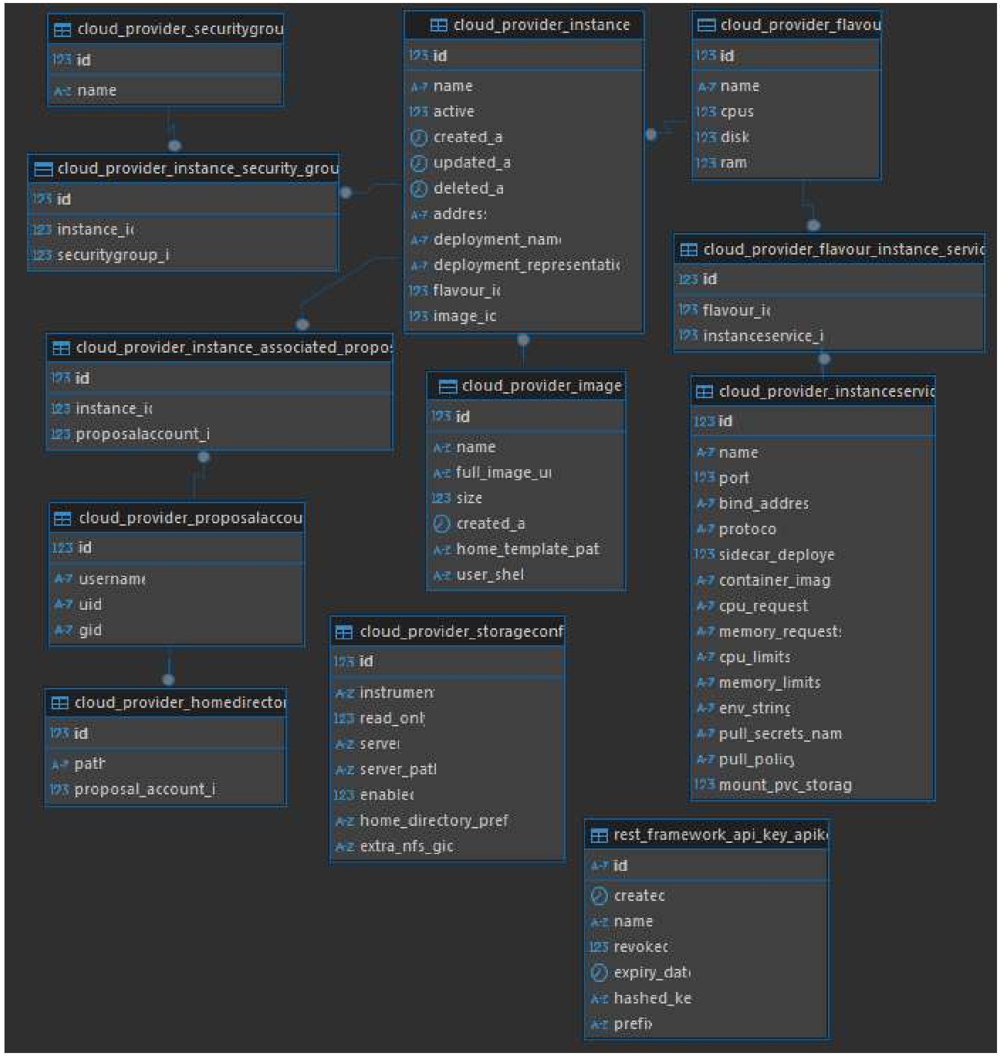
The EU neutron source FRDEGBATBECZDKITPLSKESSESICH World's leading facility in neutron science & tech for 50+ years.

12 followers · Grenoble, France · http://www.ill.eu · @ILLGrenoble · opensource@ill.eu

Database

Implementation

- Developed Cloud Provider API (extracted exact spec from Open Stack's cloud provider).
- API key authorization.
- Database schema for local storage (Instances, Flavours, Security Groups, etc).



cloud_provider_securitygrou
id
name

cloud_provider_instance_security_grou
id
instance_i
securitygroup_i

cloud_provider_instance_associated_propo:
id
instance_i
proposalaccount_i

cloud_provider_proposalaccou
id
usernam
uid
gid

cloud_provider_homedirecto
id
path
proposal_account_i

cloud_provider_instance
id
name
active
created_a
updated_a
deleted_a
addres:
deployment_nam
deployment_representatio
flavour_i
image_ic

cloud_provider_image
id
name
full_image_u
size
created_a
home_template_pat
user_shel

cloud_provider_storageconf
id
instrumen
read_onl
serve
server_patl
enable
home_directory_pref
extra_nfs_gic

cloud_provider_flavou
id
name
cpus
disk
ram

cloud_provider_flavour_instance_servic
id
flavour_i
instanceservice_i

cloud_provider_instanceservic
id
name
port
bind_addres
protoco
sidecar_deploye
container_imag
cpu_request
memory_request:
cpu_limits
memory_limits
env_string
pull_secrets_nam
pull_polic
mount_pvc_storag

rest_framework_api_key_apik
id
createc
name
revokec
expiry_dat
hashed_ke
prefi

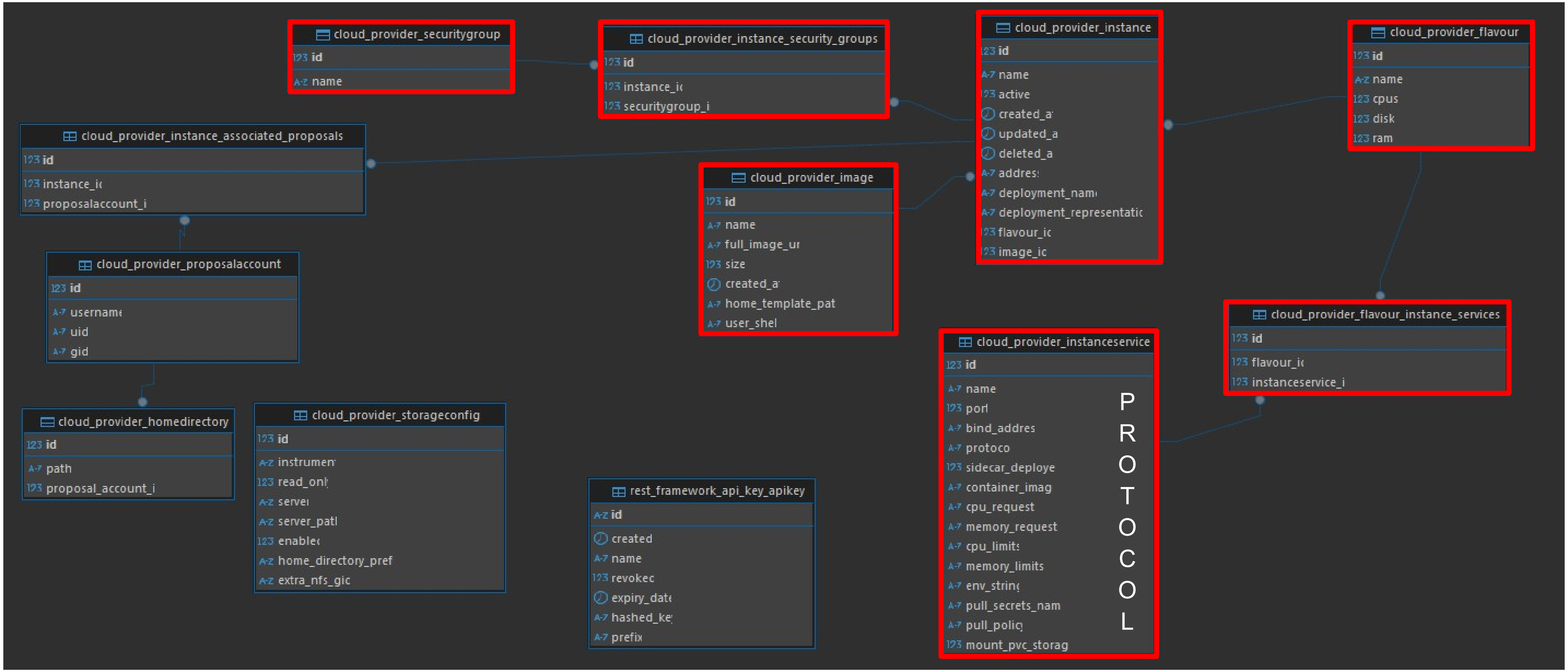
cells.es

Database

Implementation

Highlighted tables correspond to main cloud provider implementation,
the rest of them are ALBA infrastructure focused.

*Django tables not included (migrations, users, etc).



cloud_provider_securitygroup
- 123 id
- A-Z name

cloud_provider_instance_security_groups
- 123 id
- 123 instance_ic
- 123 securitygroup_i

cloud_provider_instance
- 123 id
- A-Z name
- 123 active
- created_a
- updated_a
- deleted_a
- A-Z addres:
- A-Z deployment_nam
- A-Z deployment_representatic
- 123 flavour_ic
- 123 image_ic

cloud_provider_flavour
- 123 id
- A-Z name
- 123 cpus
- 123 disk
- 123 ram

cloud_provider_instance_associated_proposals
- 123 id
- 123 instance_ic
- 123 proposalaccount_i

cloud_provider_image
- 123 id
- A-Z name
- A-Z full_image_ur
- 123 size
- created_a
- A-Z home_template_pat
- A-Z user_shel

cloud_provider_proposalaccount
- 123 id
- A-Z username
- A-Z uid
- A-Z gid

cloud_provider_flavour_instance_services
- 123 id
- 123 flavour_ic
- 123 instanceservice_i

cloud_provider_instanceservice
- 123 id
- A-Z name
- 123 port
- A-Z bind_addres
- A-Z protoco
- 123 sidecar_deploye
- A-Z container_imag
- A-Z cpu_request
- A-Z memory_request
- A-Z cpu_limit:
- A-Z memory_limits
- A-Z env_strin
- A-Z pull_secrets_nam
- A-Z pull_polic
- 123 mount_pvc_storag

PROTOCOL

cloud_provider_homedirectory
- 123 id
- A-Z path
- 123 proposal_account_i

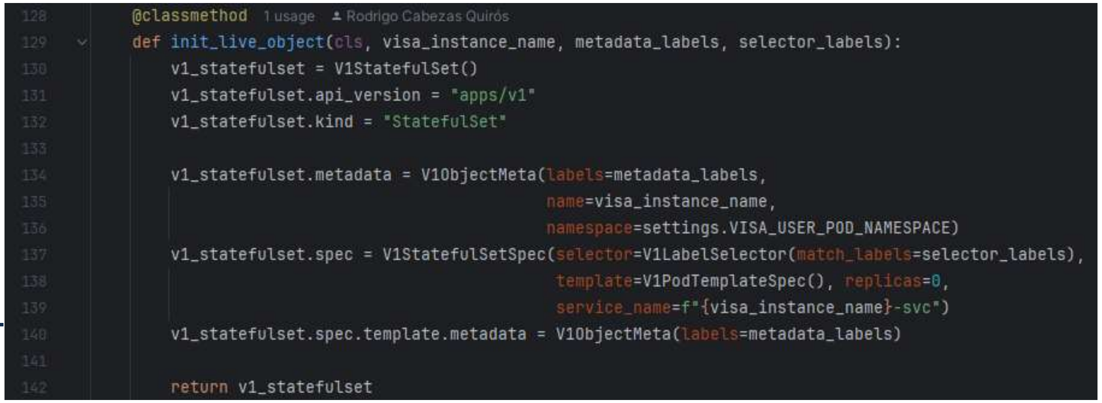
cloud_provider_storageconfig
- 123 id
- A-Z instrumen
- 123 read_onl
- A-Z serve
- A-Z server_patl
- 123 enable
- A-Z home_directory_pref
- A-Z extra_nfs_gic

rest_framework_api_key_apikey
- A-Z id
- created
- A-Z name
- 123 revoke
- expiry_dat
- A-Z hashed_ke
- A-Z prefix

Instance operations

Implementation

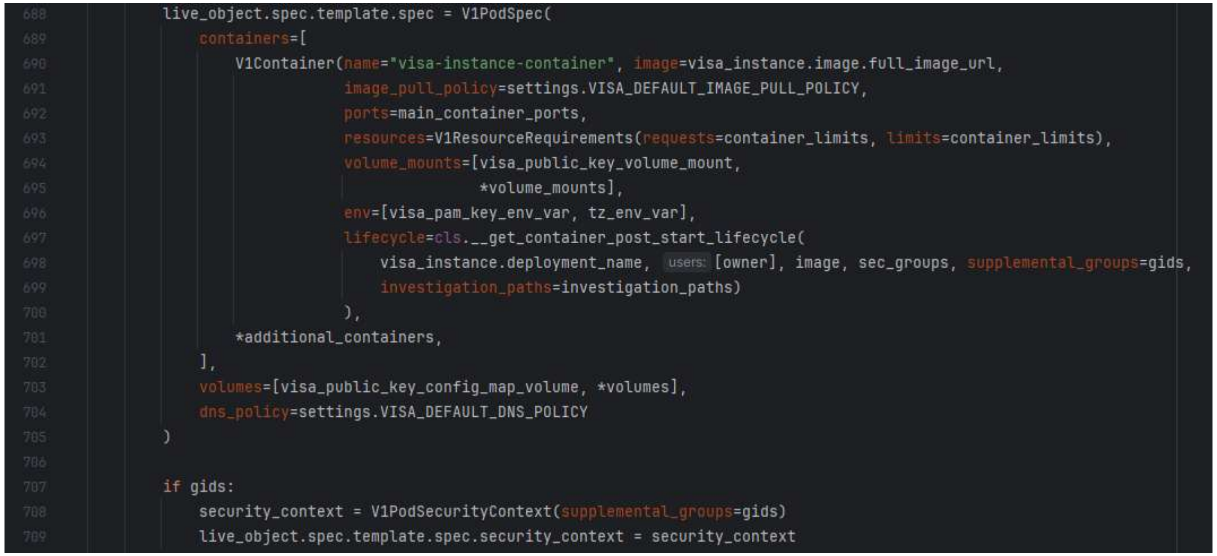
- Creation:
  - Create StatefulSet.
  - Replicas set to 0.
  - Volume attached for persistence.
  - (If applies) Investigation’s storage attached.
  - Instance services “sidecar” containers.
  - User provisioning hook.
  - VISA PAM public key mount.
  - Other config (image pull secrets, etc).
- Start:
  - Set StatefulSet replicas to 1.
- Stop:
  - Scale down StatefulSet to 0.
- Reboot:
  - Trigger re-deployment of StatefulSet.
- Delete:
  - Self-explanatory.
  - Retain attached volume used for persistence.



```
    @classmethod  1 usage  Rodrigo Cabezas Quirós
    def init_live_object(cls, visa_instance_name, metadata_labels, selector_labels):
        v1_statefulset = V1StatefulSet()
        v1_statefulset.api_version = "apps/v1"
        v1_statefulset.kind = "StatefulSet"

        v1_statefulset.metadata = V1ObjectMeta(labels=metadata_labels,
                                               name=visa_instance_name,
                                               namespace=settings.VISA_USER_POD_NAMESPACE)
        v1_statefulset.spec = V1StatefulSetSpec(selector=V1LabelSelector(match_labels=selector_labels),
                                                template=V1PodTemplateSpec(), replicas=0,
                                                service_name=f"{visa_instance_name}-svc")
        v1_statefulset.spec.template.metadata = V1ObjectMeta(labels=metadata_labels)

        return v1_statefulset
```



```
        live_object.spec.template.spec = V1PodSpec(
            containers=[
                V1Container(name="visa-instance-container", image=visa_instance.image.full_image_url,
                            image_pull_policy=settings.VISA_DEFAULT_IMAGE_PULL_POLICY,
                            ports=main_container_ports,
                            resources=V1ResourceRequirements(requests=container_limits, limits=container_limits),
                            volume_mounts=[visa_public_key_volume_mount,
                                           *volume_mounts],
                            env=[visa_pam_key_env_var, tz_env_var],
                            lifecycle=cls.__get_container_post_start_lifecycle(
                                visa_instance.deployment_name,  users: [owner], image, sec_groups, supplemental_groups=gids,
                                investigation_paths=investigation_paths)
                            ),
                *additional_containers,
            ],
            volumes=[visa_public_key_config_map_volume, *volumes],
            dns_policy=settings.VISA_DEFAULT_DNS_POLICY
        )

        if gids:
            security_context = V1PodSecurityContext(supplemental_groups=gids)
            live_object.spec.template.spec.security_context = security_context
```

*StatefulSet initialization and PodSpec definition snippets, whole code does not fit here, there’s more to it.

Instance state retrieval

Implementation

State	Has fault?	StatefulSet exists?	Pods amount	Spec. replicas	Ready replicas
ERROR	Yes	-	-	-	-
DELETED	No	No	-	-	-
STOPPED	No	Yes	0	0	-
ACTIVE	No	Yes	1	1	1
STARTING	No	Yes	-	1	0
STOPPING	No	Yes	-	0	-
REBOOTING	No	Yes	-	1	0
UNKNOWN	Anything else that does not fit in the definition above.				

Instance state retrieval

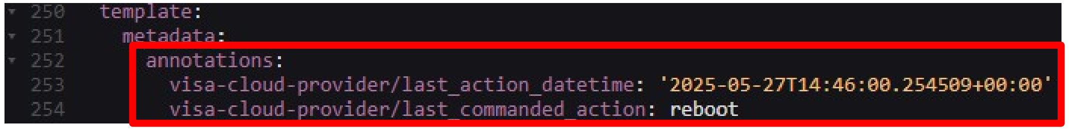
Implementation

State	Has fault?	StatefulSet exists?	Pods amount	Spec. replicas	Ready replicas
ERROR	Yes	-	-	-	-
DELETED	No	No	-	-	-
STOPPED	No	Yes	0	0	-
ACTIVE	No	Yes	1	1	1
STARTING	No	Yes	-	1	0
STOPPING	No	Yes	-	0	-
REBOOTING	No	Yes	-	1	0
UNKNOWN	Anything else that does not fit in the definition above.				

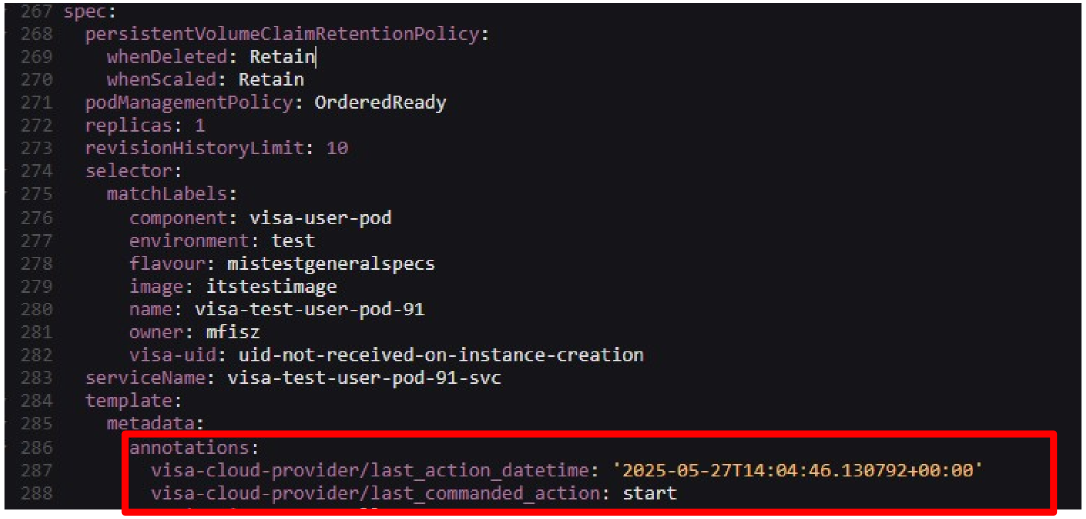
Similar state's definition clash between them: 'Starting' and 'Rebooting'.

Implementation

- Annotation is added to the StatefulSet for indicating last action commanded to the instance.
- Allows to distinguish similarly defined states.
- Adding or modifying the annotation also triggers the redeploy / reboot of the instance.



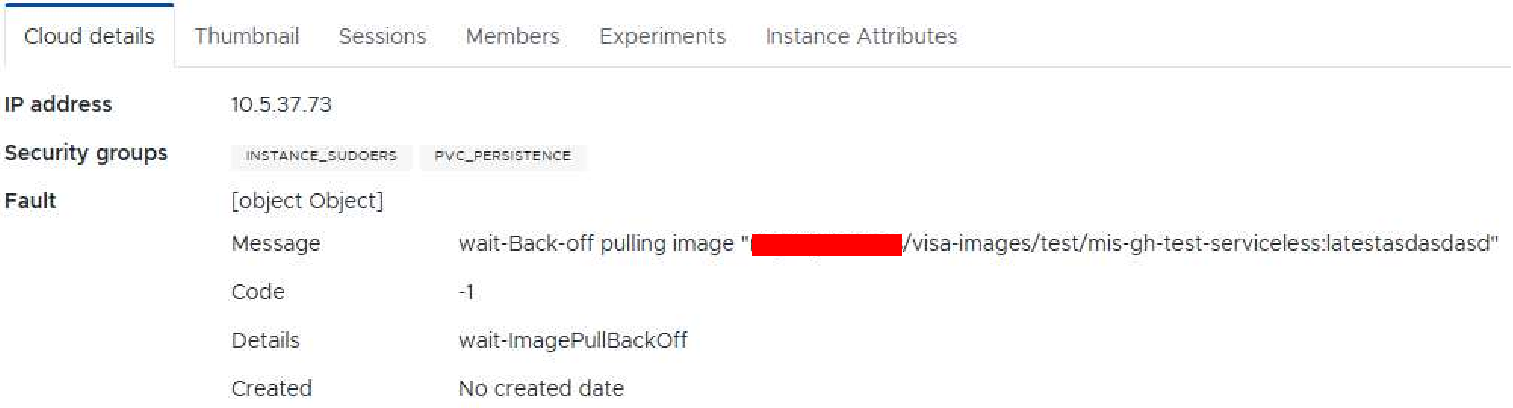
```
  template:
    metadata:
      annotations:
        visa-cloud-provider/last_action_datetime: '2025-05-27T14:46:00.254509+00:00'
        visa-cloud-provider/last_commanded_action: reboot
```



```
spec:
  persistentVolumeClaimRetentionPolicy:
    whenDeleted: Retain
    whenScaled: Retain
  podManagementPolicy: OrderedReady
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      component: visa-user-pod
      environment: test
      flavour: mistestgeneralspecs
      image: itstestimage
      name: visa-test-user-pod-91
      owner: mfisz
      visa-uid: uid-not-received-on-instance-creation
  serviceName: visa-test-user-pod-91-svc
  template:
    metadata:
      annotations:
        visa-cloud-provider/last_action_datetime: '2025-05-27T14:04:46.130792+00:00'
        visa-cloud-provider/last_commanded_action: start
```

Implementation

- An instance has a fault if a container’s state is ‘Waiting’, for any reason excluding these ones:
  - ContainerCreating.
  - PodInitializing.
  - Terminating.
- Fault codes and messages are retrieved from the k8s PodStatus object.



IP address 10.5.37.73

Security groups INSTANCE_SUDOERS PVC_PERSISTENCE

Fault [object Object]

Message wait-Back-off pulling image "[redacted]/visa-images/test/mis-gh-test-serviceless:latestasdasdasd"

Code -1

Details wait-ImagePullBackOff

Created No created date

User provisioning hook

Implementation

- Images are built without any local users in them.
- Instance's owner user is created on-the-fly after the Pod starts.
- A PostStart hook is in charge of:
  - Creating the necessary groups (mainly for accessing NFS storage)
  - Creating user locally with required groups.
  - Setting time zone inside the instance.
  - Creating the user's home.
  - Copy contents of user's home from a predefined home template.
  - Adding user to sudoers group if user belongs to VISA administrators group (sec. group filter: role=admin, then sec. group=INSTANCE_SUDOERS).

```
/bin/sh -c 'ln -sf /usr/share/zoneinfo/Europe/Madrid /etc/localtime && echo Europe/Madrid > /etc/timezone
&& groupadd -g 1001 visa-test-user-pod-94 && mkdir -p /home/rcabezas && useradd -u 1999 -g visa-test-user-pod-94 -s
 /usr/bin/bash -d /home/rcabezas  rcabezas && cp -rf /etc/visa_home_template/. /home/rcabezas/ &&
echo 'export TZ=Europe/Madrid' >> /home/rcabezas/.bashrc && chown -Rf rcabezas:visa-test-user-pod-94 /home/rcabezas &&
 echo 'rcabezas ALL=(ALL) NOPASSWD: ALL' >> /etc/sudoers'
```

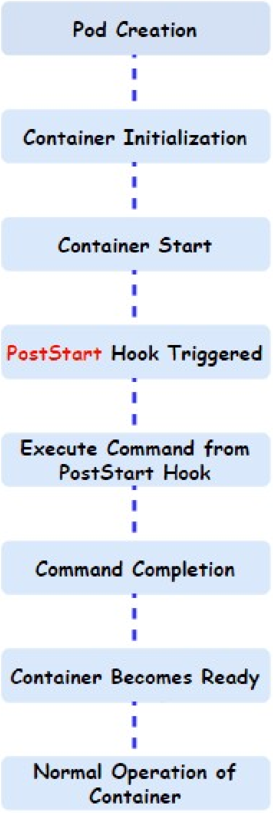
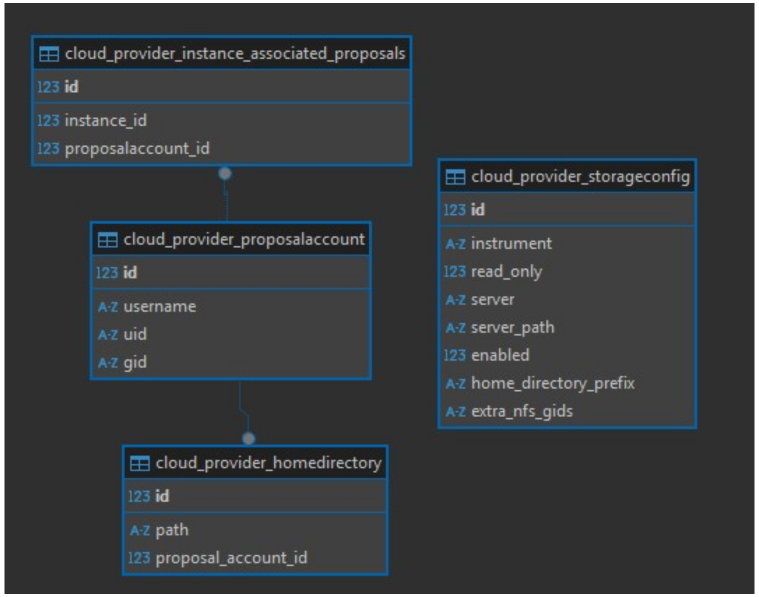


Diagram from Sai Manasa. Kubernetes: Container Lifecycle Hooks.

Implementation

- Instrument storage is mounted via NFS.
- Specific groups must be given to a user in order to grant access to an investigation's data.
- A cron retrieves periodically the following information from ALBA's LDAP:
  - Investigation's full path(s).
  - Specific group ID for each investigation.
- A separate database model keeps additional NFS configuration details.



cloud_provider_instance_associated_proposals
123 id
123 instance_id
123 proposalaccount_id

cloud_provider_proposalaccount
123 id
A-Z username
A-Z uid
A-Z gid

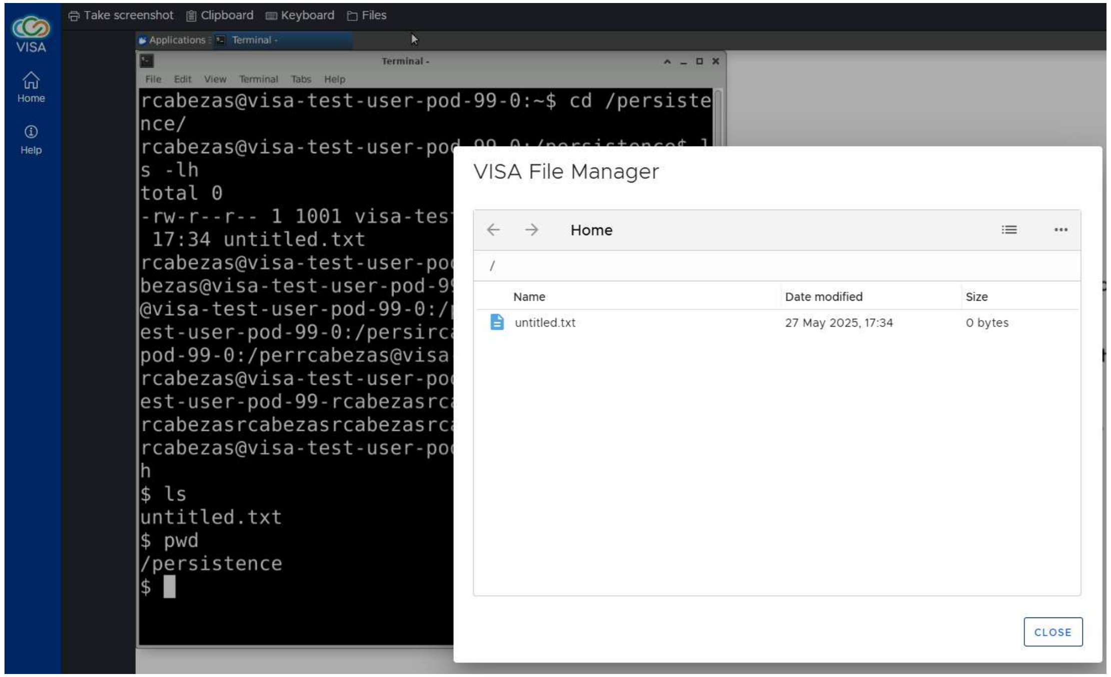
cloud_provider_homedirectory
123 id
A-Z path
123 proposal_account_id

cloud_provider_storageconfig
123 id
A-Z instrument
123 read_only
A-Z server
A-Z server_path
123 enabled
A-Z home_directory_prefix
A-Z extra_nfs_gids

Persistence

Implementation

- Instances write to separate storage from the acquisition folders.
- A PersistentVolumeClaim is created along the instance’s StatefulSet and provides a persistence layer between restarts.
- Storage capacity of the PVC is configurable through the image’s flavour.
- VISA node-fs-api only allows data upload and download from the PVC.
- Still unclear on how to make data stored in the PVC available to users upon instance deletion.
  - Could potentially be ingested into data catalogue.



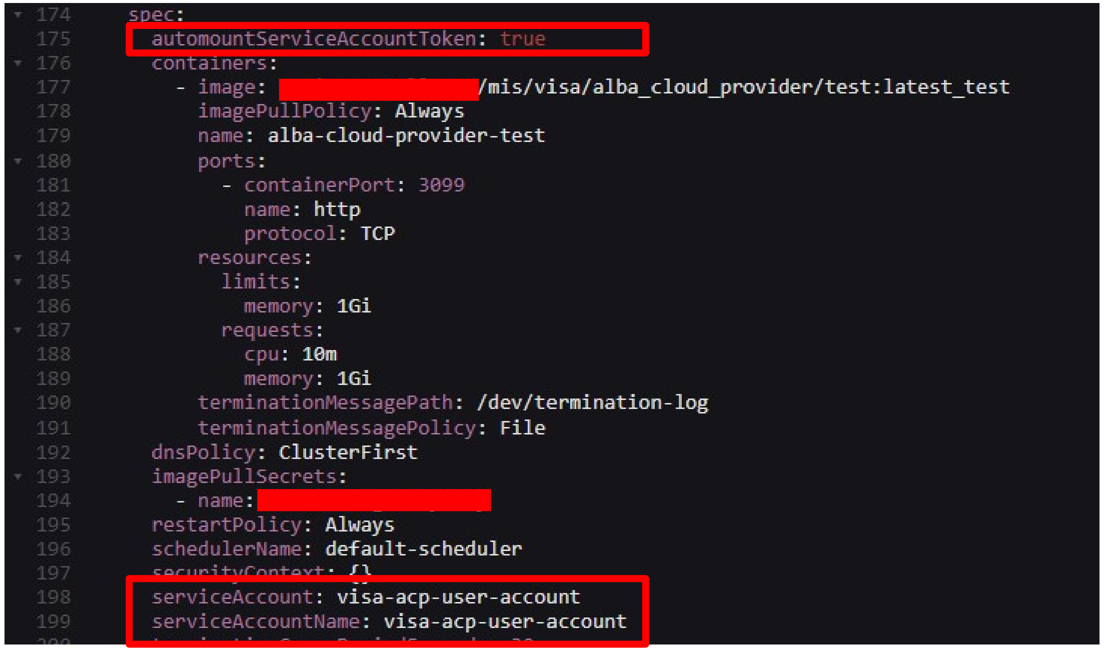
```
rcabezas@visa-test-user-pod-99-0:~$ cd /persiste
nce/
rcabezas@visa-test-user-pod-99-0:/persistence$ l
s -lh
total 0
-rw-r--r-- 1 1001 visa-tes
 17:34 untitled.txt
rcabezas@visa-test-user-poc
bezas@visa-test-user-pod-9
@visa-test-user-pod-99-0:/
est-user-pod-99-0:/persirca
pod-99-0:/perrcabezas@visa
rcabezas@visa-test-user-poc
est-user-pod-99-rcabezasrca
rcabezasrcabezasrcabezasrca
rcabezas@visa-test-user-poc
h
$ ls
untitled.txt
$ pwd
/persistence
$ 
```

VISA File Manager

Name	Date modified	Size
untitled.txt	27 May 2025, 17:34	0 bytes

Implementation

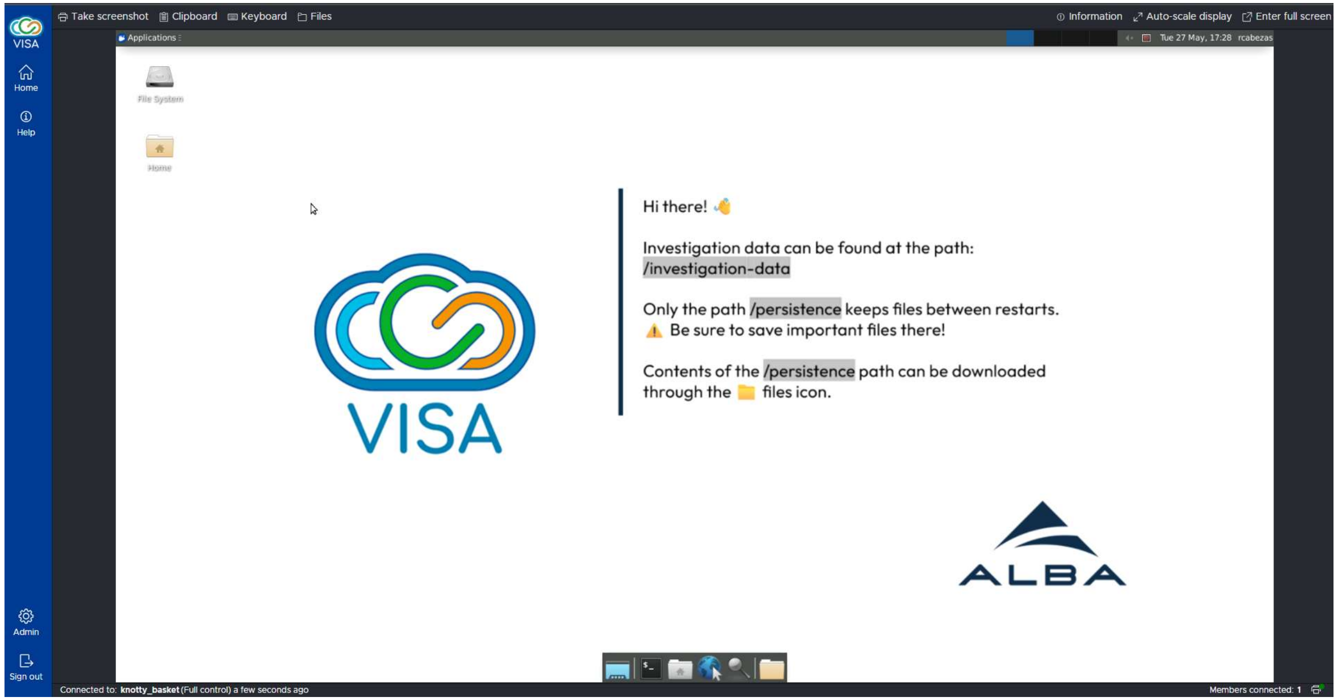
- The cloud provider uses a Service Account as means of k8s API access.
- The Service Account is bound to a cluster role that restricts the actions it can perform over cluster resources.
- SA is also restricted to a specific namespace.
- The deployment running the k8s cloud provider uses this SA and automatically mounts it.



```
spec:
  automountServiceAccountToken: true
  containers:
    - image: [redacted]/mis/visa/alba_cloud_provider/test:latest_test
      imagePullPolicy: Always
      name: alba-cloud-provider-test
      ports:
        - containerPort: 3099
          name: http
          protocol: TCP
      resources:
        limits:
          memory: 1Gi
        requests:
          cpu: 10m
          memory: 1Gi
      terminationMessagePath: /dev/termination-log
      terminationMessagePolicy: File
  dnsPolicy: ClusterFirst
  imagePullSecrets:
    - name: [redacted]
  restartPolicy: Always
  schedulerName: default-scheduler
  securityContext: {}
  serviceAccount: visa-acp-user-account
  serviceAccountName: visa-acp-user-account
```

Demo

- K8s cloud provider administration interface.
- VISA + k8s cloud provider.
- StatefulSets within cluster's management platform.



Hi there! 👋

Investigation data can be found at the path: /investigation-data

Only the path /persistence keeps files between restarts.
⚠️ Be sure to save important files there!

Contents of the /persistence path can be downloaded through the 📁 files icon.

Connected to: **knotty_basket** (Full control) a few seconds ago

Members connected: 1

Future work

- Implement a Kubernetes operator to encapsulate all of the cloud provider's logic into the cluster.
  - Use CRDs for better managing instance's state and faults.
  - Easier management and install.
  - Cloud provider API only to make changes on VISA CRDs.
- Make cloud provider available to collaboration.
  - ~~Improve~~ Create cloud provider documentation.
  - Add integration testing (using kind.sigs.k8s.io).
  - Probably, some refactoring.

ALBA

cells.es

Cerdanyola del Vallès (Barcelona)
Spain
Tel: (+34) 93 592 43 00