Deploying CloudMan on the NeCTAR Research Cloud

usecloudman.org

Demo

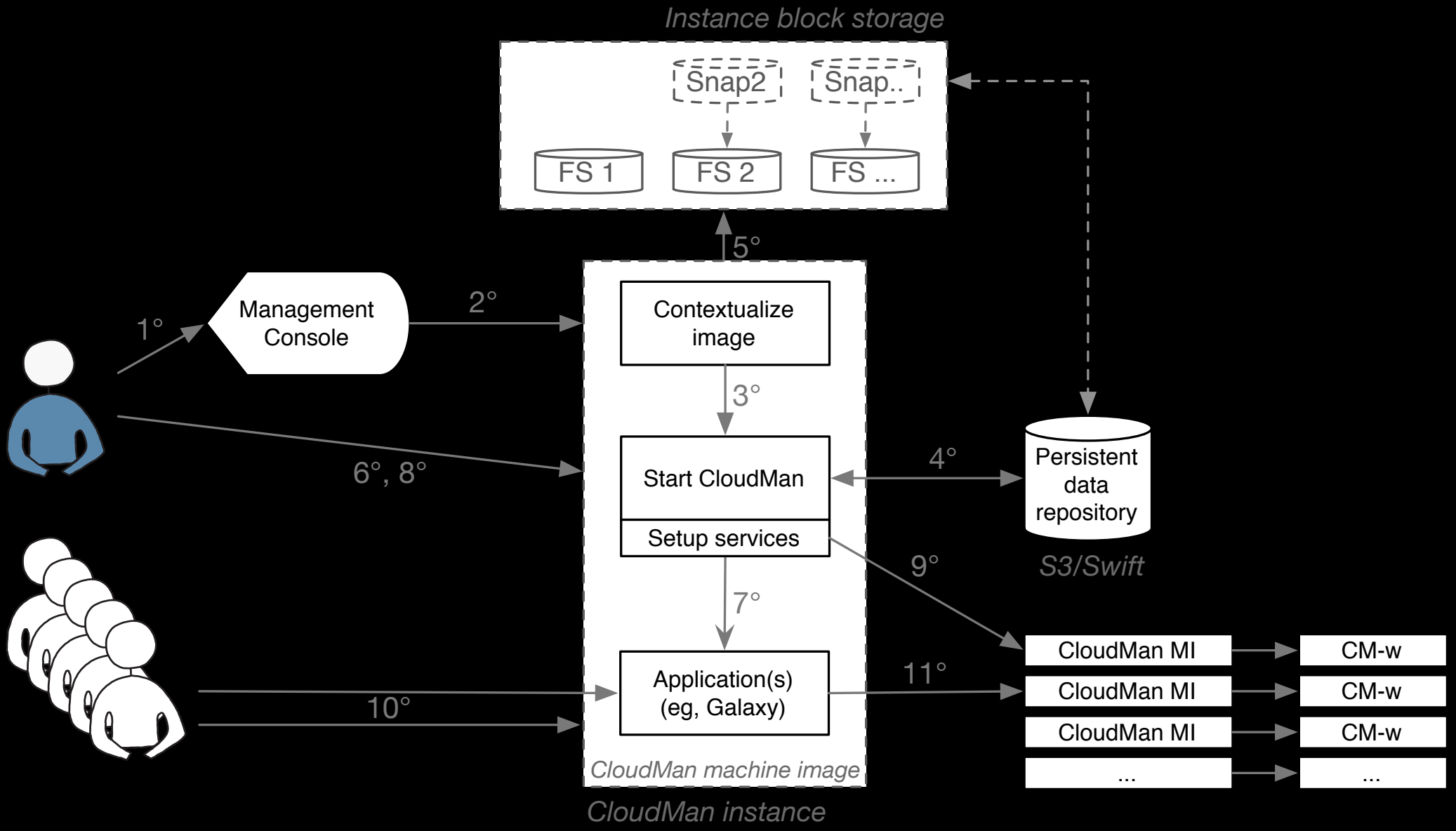
BioCloudCentral.org

CloudMan features

- A complete solution for instantiating, running and scaling cloud resources
  - Get a scalable compute cluster (SGE)
- Get an automatically configured Galaxy
  - Scope of tools and reference datasets exceed Galaxy Main
- Deployment on AWS, OpenStack, and OpenNebula clouds
  - Wizard-guided setup: requires no computational expertise, no infrastructure, no software
- Automated configuration for machine image, tools, and data
- Self-contained deployment
- Elastic resource scaling: manual or automatic
- Dynamic persistent storage
- Share your instance: including all customizations (data, tools & configurations)

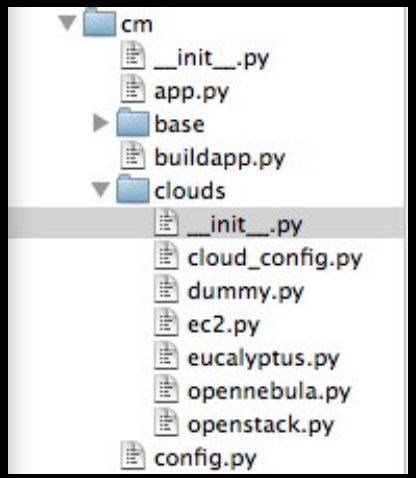
- Deploy a (Galaxy) cluster in minutes!

CloudMan architecture



CloudMan internals

- Written in Python as a web app
- Uses boto library for cloud communication
- Porting to OpenStack
  - Introduced a new cloud interface
  - Added new connection properties
  - 90% compatible & it works!



NeCTAR RC Experience

CloudMan Requirements

- A preconfigured machine image
- User data
- Persistent object store
- Resource metadata (ie, tags)
- Data volumes and volume snapshots

CloudMan Requirements

- A preconfigured machine image
- User data
- Persistent object store
- Resource metadata (ie, tags)
- Data volumes and volume snapshots

A preconfigured MI

- Contains system-level requirements & contextualization hooks
- Machine image creation is automated via Fabric scripts
  - mi-deployment:

```
fab -f mi_fabfile.py -i private_key_file -H instance_IP configure_MI
```

  - As part of CloudBioLinux

```
fab -f fabfile.py -i private_key_file -H instance_IP -u username install_biolinux
```

- The CloudBioLinux version requires more than 10GB root image size
  - Had to reduce the number of installed tools
  - Offloaded some parts to a shared NFS disk

Resource metadata (ie, tags)

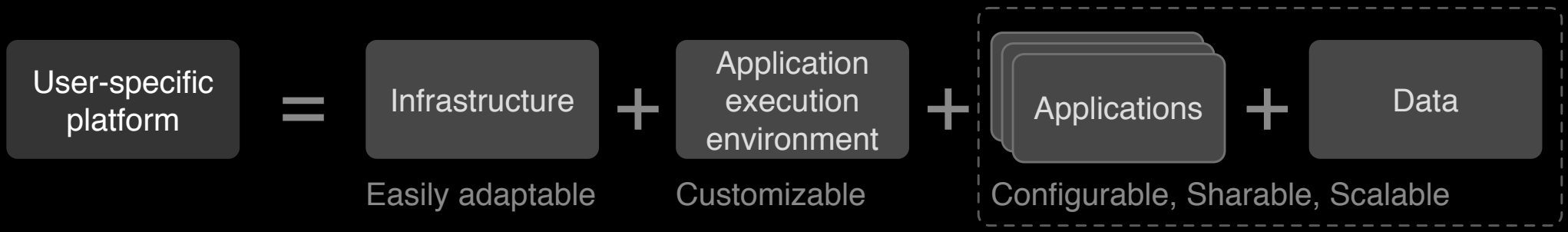
- Used to identify resources
  - Cluster restarts
  - User-identification via the Dashboard
- boto does not work
- Native OpenStack python-novaclient does not work
  - Some code twiddling helps but does not seem to resolve it
- REST API works (but…)
- Filed a bug on launchpad (should be fixed in essex?)

https://bugs.launchpad.net/nova/+bug/972102

```
>>> s = cs.servers.list()
>>> s
<Server: meta test>
>>> cs.servers.set_meta(s, {'py': 'py shell'})
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "novaclient/v1_1/servers.py", line 572, in set_meta
    body, "metadata")
  File "novaclient/base.py", line 157, in _create
    resp, body = self.api.client.post(url, body=body)
  File "novaclient/client.py", line 139, in post
    return self._cs_request(url, 'POST', **kwargs)
  File "novaclient/client.py", line 124, in _cs_request
    **kwargs)
  File "novaclient/client.py", line 107, in request
    raise exceptions.from_response(resp, body)
novaclient.exceptions.NotFound: Server does not exist (HTTP 404)
```

```
>>> cs.servers.set_meta(s, {'py': 'py shell'})
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "novaclient/v1_1/servers.py", line 35, in __repr__
    return "<Server: %s>" % self.name
  File "novaclient/base.py", line 328, in __getattr__
    self.get()
  File "novaclient/base.py", line 347, in get
    new = self.manager.get(self.id)
  File "novaclient/base.py", line 331, in __getattr__
    raise AttributeError(k)
AttributeError: id
```

CloudMan-as-a-Platform



Enable easy creation of user-specific cloud platforms

Couple the infrastructure, complex and functional application execution environments, applications, and data into a single unit that can easily be used and manipulated by a user.

Data volumes (and snapshots)

- Volumes provide
  - Persistence
  - Enable pre-configured data elements
  - Platform sharing
- Waiting…

Overall

The Research Cloud works quite smoothly!

and thank you for that.