University of Basel

Configuration management on a secure OpenStack environment

Jani Heikkinen, sciCORE

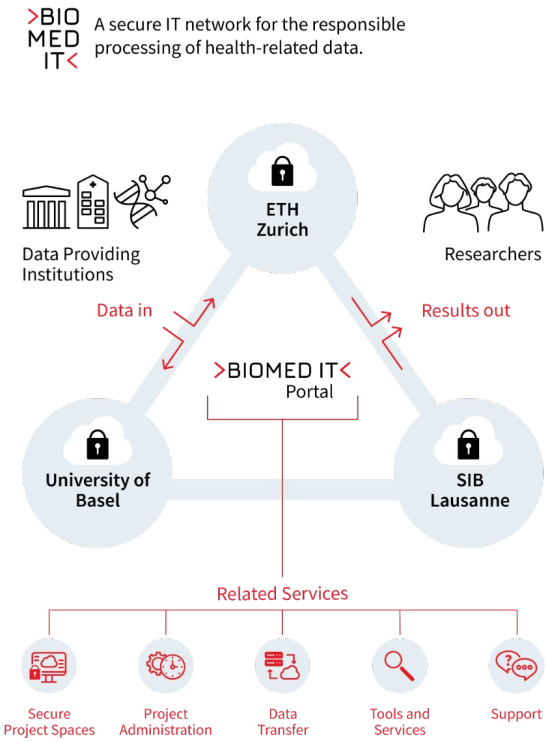
2022-05-22

SIB Swiss Institute of Bioinformatics

Swiss TPH
Swiss Tropical and Public Health Institute
Schweizerisches Tropen- und Public Health-Institut
Institut Tropical et de Santé Publique Suisse

Contents

BioMedIT Network



The BioMedIT Network builds on three legally independent scientific IT competence platforms:

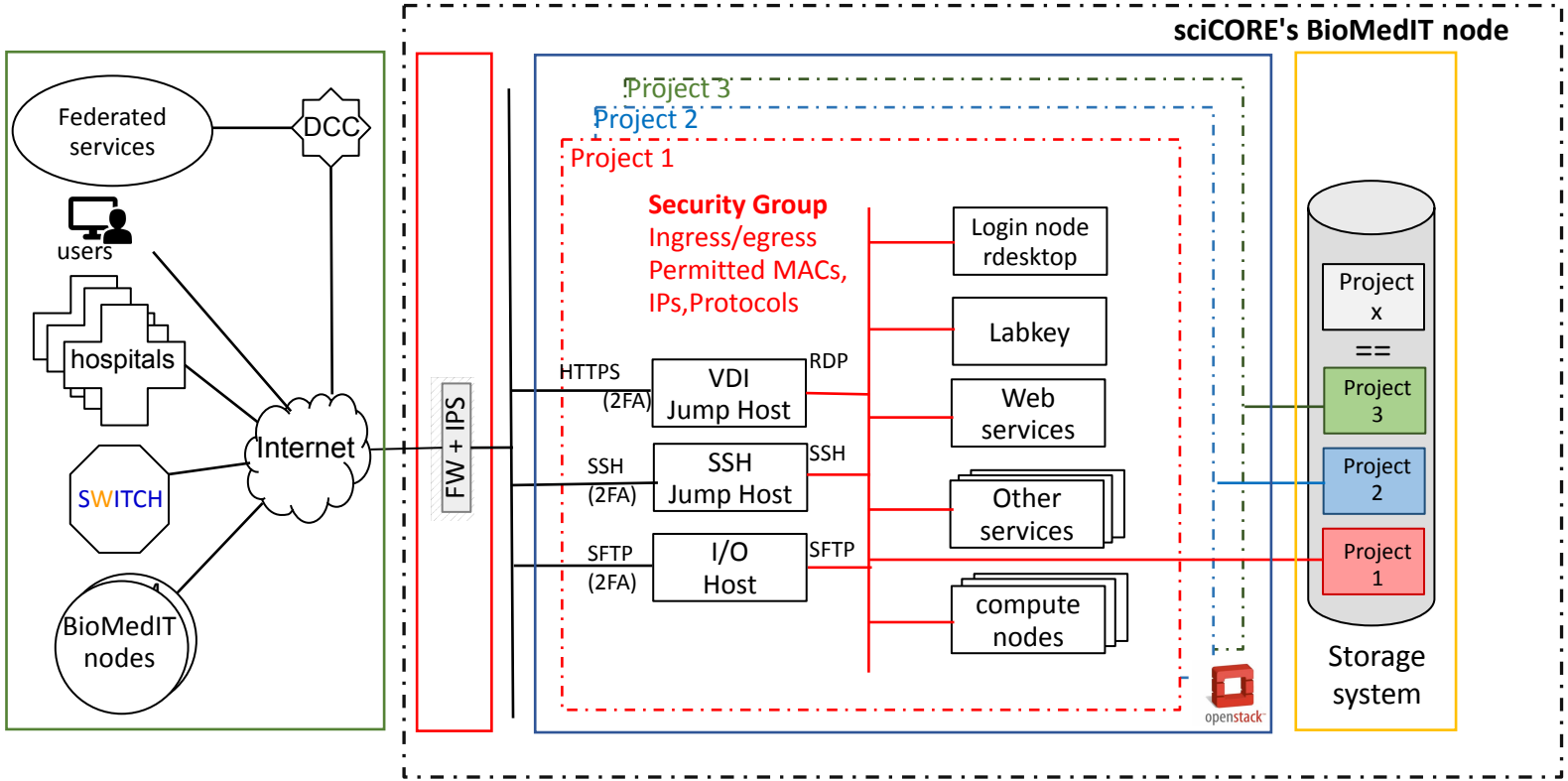
- sciCORE, operated by the University of Basel
- Romandie node operated by UNIL (previously SIB)
- SIS, operated by ETH Zurich

The BioMedIT network is coordinated by Data Coordination Centre (DCC), hosted at SIB.

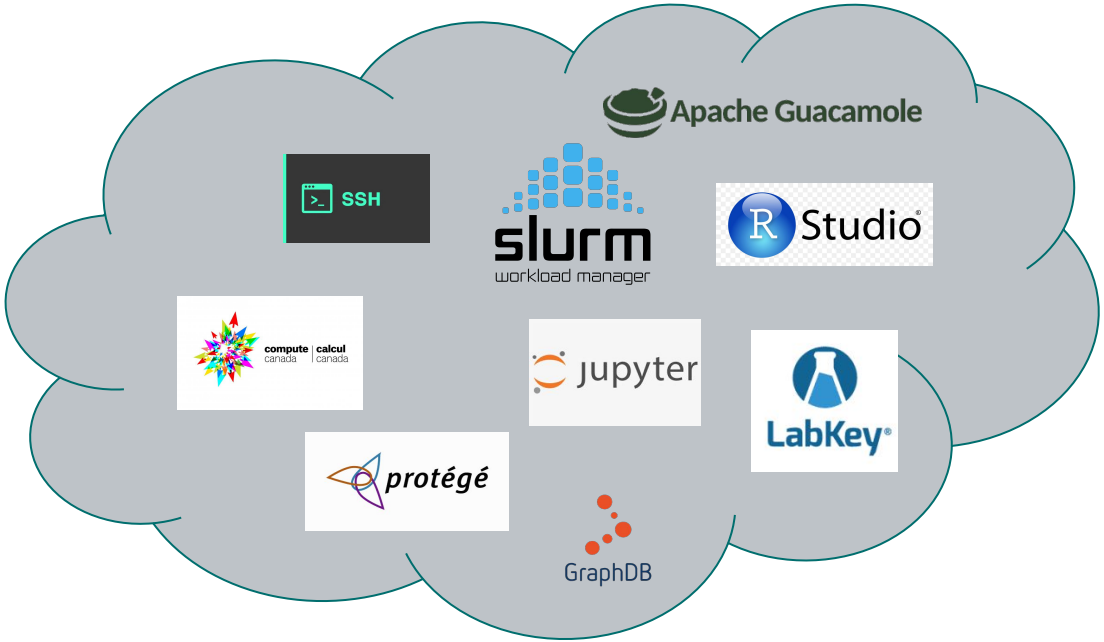
SPHN Swiss Personalized Health Network

SIB Swiss Institute of Bioinformatics

OpenStack project architecture at sciCORE BioMedIT node



sciCORE BioMedIT project space features



Challenges and requirements

Infrastructure challenges

- How to automate deployment, use resources efficiently, when to use external consultant?

Projects challenges

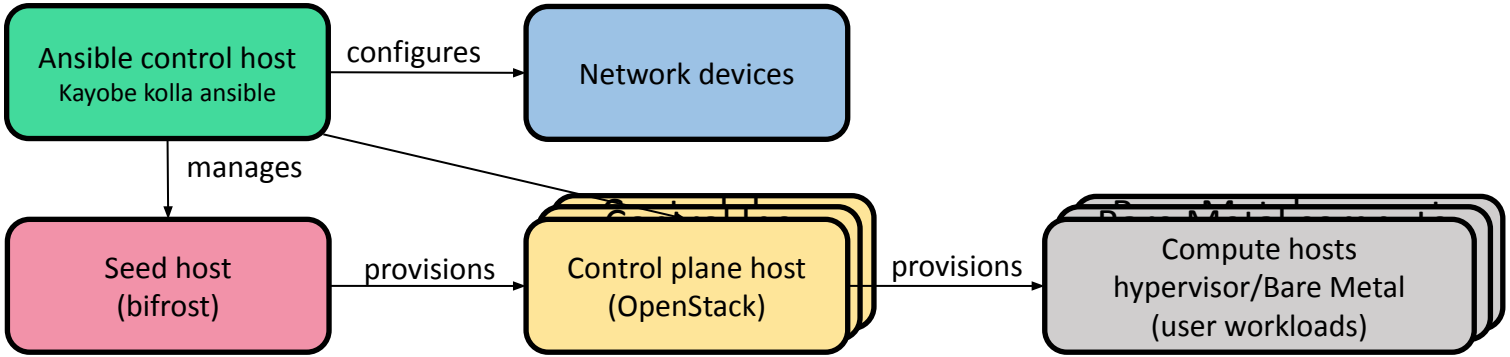
- How to automate project onboarding and deployment process?
- How to keep track on changes made on OpenStack?
- Where to put the focus on system administration?
- How to maximise usable resources (sysadmin time, services on top, using external contracts, etc.)

Configuration management overview

- The main tool to manage configurations changes in sciCORE BioMedIT OpenStack is git.
- Everything that is automated is in a Git repository.
- OpenStack infrastructure installation and configuration tools Kayobe/Kolla-Ansible are initially cloned from GitHub
    - Later our specific populated configs are stored in local branch in our gitlab.
- Almost all our deployment tools are written in Ansible.
- For project infrastructure provisioning we use Terraform.

OpenStack deployment tool Kayobe

- Kayobe enables deployment of containerized OpenStack to bare metal.
- Based on bifrost, Kolla and Kolla-ansible *(bifrost is a self contained Ironic service)*
- Heavily automated using Ansible
- Deployment of a *seed* VM used to deploy the OpenStack control plane
- Configuration of physical network infrastructure
- Discovery, introspection and provisioning of control plane hardware using OpenStack bifrost
- Deployment of an OpenStack control plane using OpenStack Kolla-Ansible
- Discovery, introspection and provisioning of bare metal compute hosts using OpenStack ironic



Deployment of projects in OpenStack
Choosing the right tool

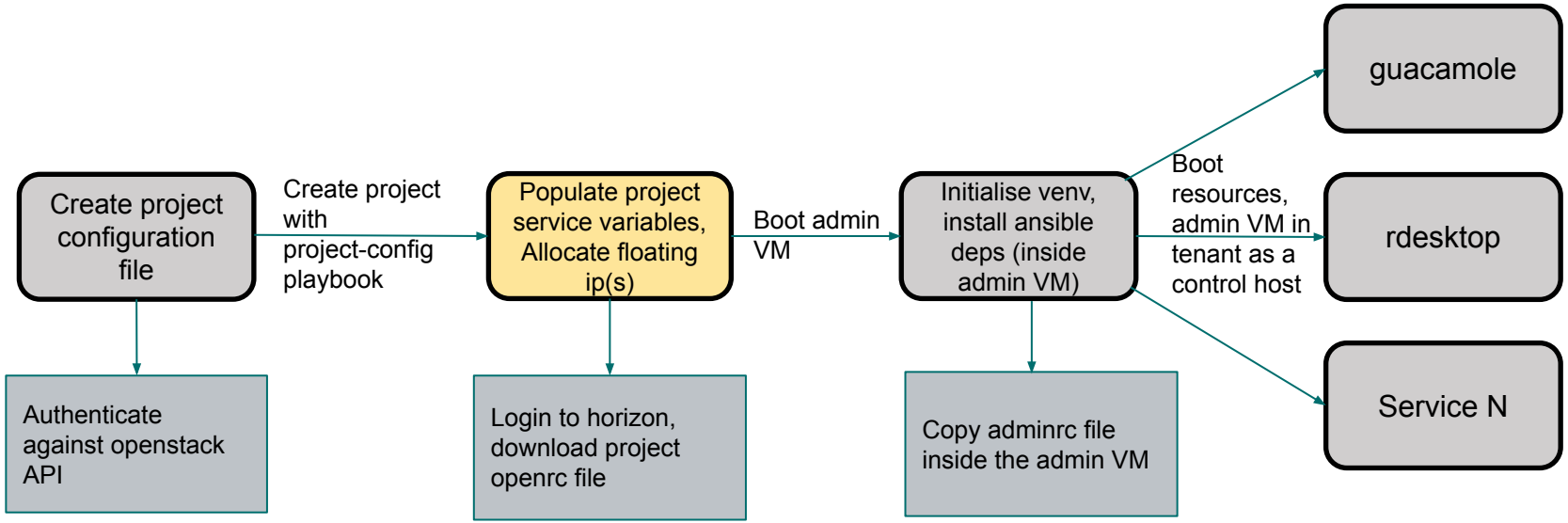
- We had previous experience using Ansible.
- When BioMedIT was started, we chose Kolla-Ansible for OpenStack deployment.
- Ansible provides many modules to interact with OpenStack API:
  https://docs.ansible.com/ansible/latest/collections/openstack/cloud/index.html
- For us the natural choice was to use Ansible for project bootstrap and configuration .
  DISCLAIMER: In our case, it ended up being non-optimal choice ;)

Deployment of projects in OpenStack
Initial approach

- We started by writing all the automation for projects deployment in one big Ansible playbook
- Our initial playbook included (too) many different tasks:
  - Bootstrap the tenants. Create networks, subnets, routers, disk volumes, security group rules and boot VMs and create any other OpenStack resource
  - Query the resources using an ansible dynamic inventory plugin (custom developed)
  - Install project specific software/applications/services on VMs
  - Deploy user accounts

Deployment of projects in OpenStack

First iteration



Deployment of projects in OpenStack
Issues encountered when using single playbook

- The code quickly became big, complex and hard to maintain:
  - Booting resources and configuring them in the same playbook increased complexity.
  - Some parts of the playbook used OpenStack API to create resources while other parts were configuring the VMs.
  - We had to switch to different OpenStack credentials to work with different tenants and to be able to use the dynamic inventory.
  - Ansible is *stateless*. This is great for some use cases but it was a pain for our use case. Our playbook had to keep track of every deleted resource e.g. security group rules
- The Ansible dynamic inventory plugin was yet another piece of code to maintain

Deployment of projects in OpenStack
Rethinking our initial approach

- Once the code became too complex we realized (and accepted, this was harder ;) that it was difficult to maintain and it did not scale well. Therefore we decided to refactor.

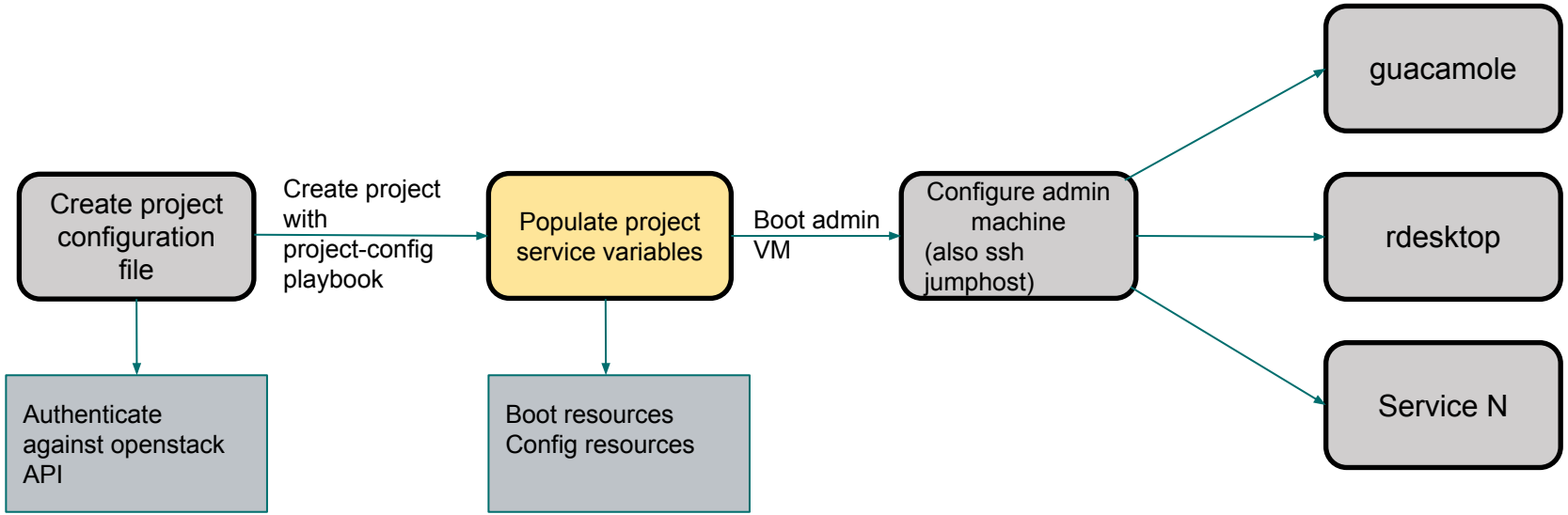
- First we tried to split our ansible code in smaller independent playbooks:
  - Playbook 1 only interacting with OpenStack API (creating of networks, routers, VMs..etc)
  - Playbook 2 configuring all the VMs

- Second idea was to use Terraform to bootstrap resources (networks, routers, VMs…etc) and use ansible to configure them. This was the final choice.

- At this stage we finished firewall automated configuration which improved the process

Deployment of projects in OpenStack

Second iteration



Deployment of projects in OpenStack
Moving to Terraform + cookiecutter + Ansible

- We split the code in two smaller, simpler and easier to maintain pieces
  - Terraform interacts with the OpenStack API to bootstrap cloud resources (networks, VMs..)
  - Ansible only configures machines.

- Terraform's stateful design helped to simplify the code interacting with the OpenStack API and managing cloud resources.

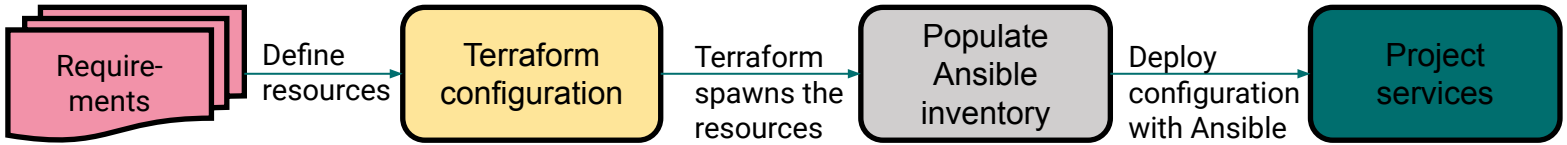
- We also took the opportunity to move from ansible dynamic inventory to ansible static inventory

- Structure of Ansible configuration was refactored

Deployment of projects in OpenStack

Workflow to create a new project from scratch

- Collect project requirements.
- Create the terraform project using our cookiecutter template and adapt it to the project's needs (add or remove extra VMs and resources).
- Bootstrap the project with terraform (networks, routers, VMs, security group rules…) . Each tenant has a dedicated git repo for the Terraform config and another private git repo to keep track infrastructure changes and the state for each tenant.
- Populate the ansible static inventory with the new booted VMs.
- Execute ansible to deploy required software and users accounts for the project.



Lessons learned

- Dedicate people only for OpenStack, it is a full time job!
- Using external support is okay! (StackHPC)
- OpenStack is an excellent way to fully utilise available hardware but has an administrative overhead
- Deployment automation is a must.
- Automation keeps the environment homogenous, reduces mistakes and facilitates auditing
- Separate *provisioning* of infrastructure from *configuration of services* (keep it simple…)
- Services are what brings in the users and keeps the users! (focus on services, not on infrastructure)
    - Applies specifically to our project/case

Acknowledgements

People:

Pablo Escobar Lopez
Sudershan Thirunavukkarasu
Martin Jacquot
Thierry Sengstag
sciCORE team

Collaborators:

CoreIT at UNIL and SIB
SIS at ETHZ
StackHPC UK
SPHN DCC at SIB

Links:

BioMedIT https://www.biomedit.ch
SPHN https://sphn.ch
sciCORE https://scicore.unibas.ch
stackHPC https://stackhpc.com
Kayobe https://docs.openstack.org/kayobe/latest/
Kolla-Ansible https://docs.openstack.org/kolla-ansible/latest/
Terraform: https://www.terraform.io/

University of Basel

Thank you

for your attention.

Example static inventory

```
##################################

 [tenant_demo]
 demo-admin ansible_host=192.168.250.123 ansible_ssh_common_args='-F {{ playbook_dir }}/ssh_config_files/tenant_demo.ssh'
 demo-rdesktop ansible_host=192.168.250.124 ansible_ssh_common_args='-F {{ playbook_dir }}/ssh_config_files/tenant_demo.ssh'
 demo-guacamole ansible_host=192.168.250.125 ansible_ssh_common_args='-F {{ playbook_dir }}/ssh_config_files/tenant_demo.ssh'

[tenant_demo_admin]
demo-admin

[tenant_demo_rdesktop]
demo-rdesktop

[tenant_demo_guacamole]
demo-guacamole

[tenant_demo_nfs_clients]
demo-admin
demo-rdesktop

####  GLOBAL GROUPS  ####
#########################
#### Below this line you only define groups with children groups

[admin:children]
tenant_demo_admin

[rdesktop:children]
tenant_demo_rdesktop

[guacamole:children]
tenant_demo_guacamole
```

Example dynamic inventory

./openstack_inventory.py –list

```
"meta_project_demo": [
     "60fb9a69-86c3-4b94-bc25-a9b604c1a5d0",
     "7139c663-5f8b-4df3-bced-d791d7ceb502"
  ],
  "meta_role_admin": [
     "60fb9a69-86c3-4b94-bc25-a9b604c1a5d0"
  ],
  "meta_role_guacamole": [
     "7139c663-5f8b-4df3-bced-d791d7ceb502"
  ],etc.
```

Ansible playbook snippet launching guacamole:

```
- name: Launch the guacamole instance and attach a floating ip to
it
        os_server:
        state: present
        name: "{{ openstack_project }}_guacamole"
        region_name: "{{ openstack_region }}"
        image: "{{ openstack_image_guacamole }}"
        key_name: "{{ openstack_key_name }}"
        flavor: "{{ openstack_flavor_guacamole }}"
        security_groups:
        - "{{ openstack_project }}_default"
        - "{{ openstack_project }}_guacamole"
        - "{{ openstack_project }}_allow_outgoing_traffic"
        network: "{{ openstack_project_network }}"
        wait: yes
        floating_ips: "{{ guacamole_public_ip }}"
        meta:
        hostname: "{{ openstack_project }}_guacamole"
        project: "{{ openstack_project }}"
        role: "guacamole"
        group: "nfs_clients"
        register: guacamole_machine_info
```

University of Basel

Configuration management on a secure OpenStack environment

Jani Heikkinen

hpc-ch forum 19/05/2022

Deployment of projects in OpenStack

Workflow to create a new project from scratch

- Collect project requirements.
- Create the terraform project using our cookiecutter template and adapt it to the project's needs (add or remove extra VMs and resources).
- Bootstrap the project with terraform (networks, routers, VMs, security group rules…) . Each tenant has a dedicated git repo for the Terraform code to keep track infrastructure changes for each tenant.
- Populate the ansible static inventory with the new booted VMs.
- Execute ansible to deploy required software and users accounts for the project.

Configuration management in BioMedIT sciCORE

- We try to apply the “infrastructure as code” (IaC) principle.
  - Useful for collaboration in the sysadmin team
  - Useful for change management and auditing
  - Useful for reproducibility

What is sciCOREMed

- Secure IT environment for sensitive data analysis
- Based on OpenStack private cloud
- Projects are isolated from each other (OpenStack multitenancy)
- Projects have no direct internet access