MAGIC CASTLE

# Terraforming the Cloud to Teach HPC

sighpc education

acm Chapter

Thu. Feb. 8 2024

# Félix-Antoine Fortin

Principal developer of Magic Castle
Université Laval
Calcul Québec
Digital Research Alliance of Canada

he/him
@cmd-ntrf

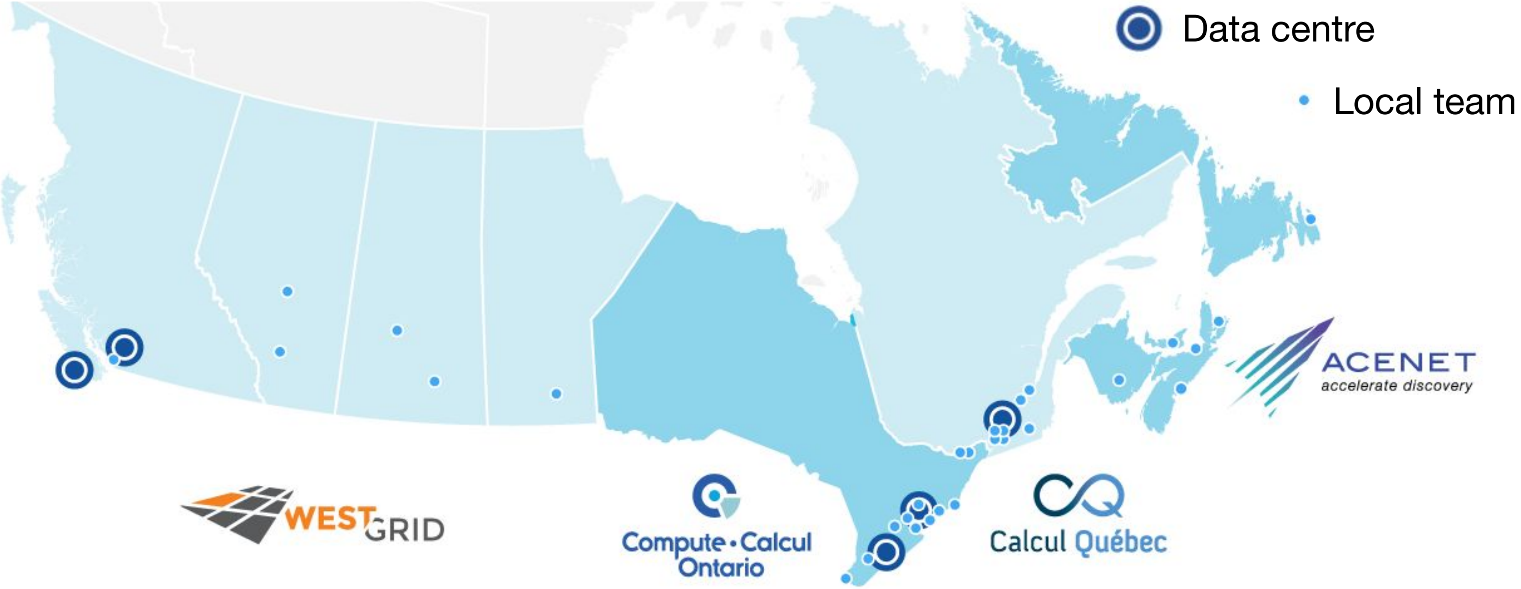
# Terraforming the Cloud to Teach HPC

- Why Cloud to teach HPC?
- Overview of existing HPC in the cloud tools
- Introduction to Magic Castle
- Magic Castle in the Wild

# Why Cloud to teach HPC?

# macro

# Advanced Research Computing (ARC)
## Research infrastructure landscape in Canada



**Digital Research Alliance** of Canada | **Alliance de recherche numérique** du Canada

# Advanced Research Computing (ARC)
## Research infrastructure landscape in Canada

150 workshops / year

95+ % usage

**How to train users at scale without impacting research?**

# micro

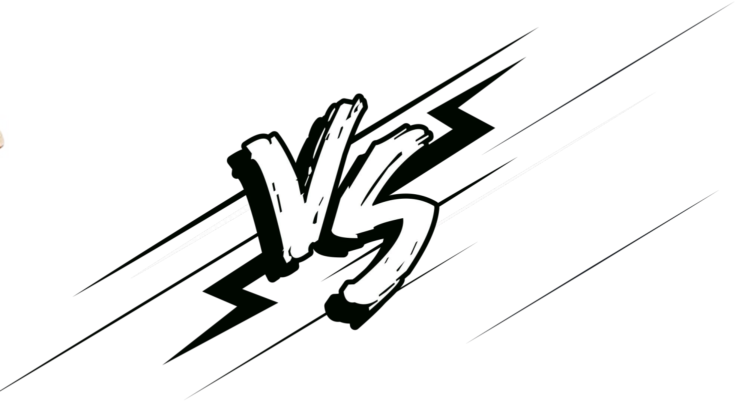
# Research support staff development

Over 200 research support staff from various scientific and engineering backgrounds, but less and less system administrators by trade.

→ ?

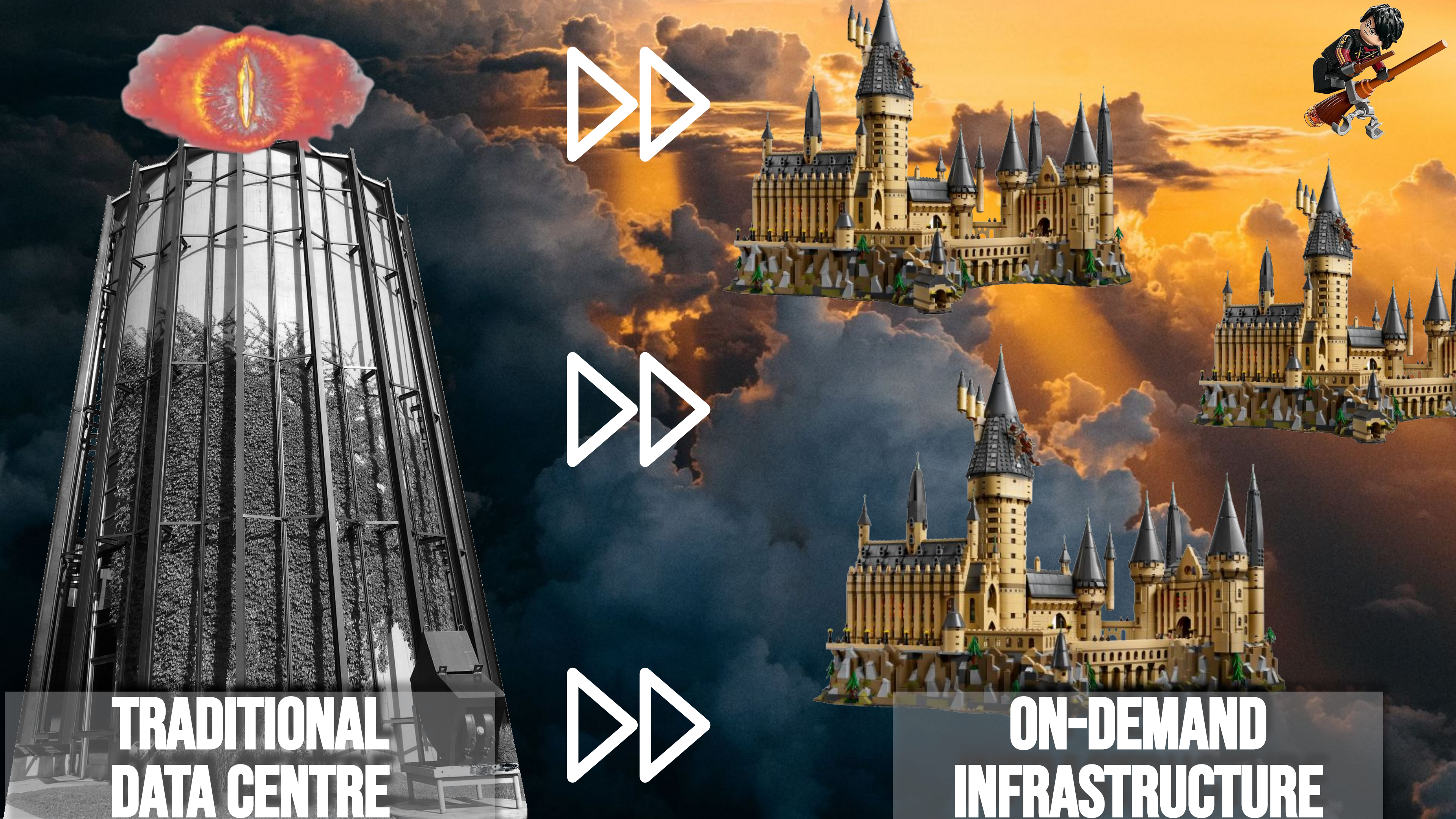
**How to provide staff inexpensive parallel computing cluster to experiment and learn?**

# Why are there more wizards in Harry Potter than in Lord of the Rings?



# Why are there more wizards in Harry Potter than in Lord of the Rings?

WIZARDRY SCHOOLS



TRADITIONAL
DATA CENTRE

ON-DEMAND
INFRASTRUCTURE

**We want accessible, inexpensive sandbox environments, designed to facilitate teaching and experimentation.**

# macro

controlled replicable
teaching environments to audiences of
variable sizes

# micro

**inexpensive sandboxes for individuals to experiment and learn at their own pace.**

# Cloud can provide the building blocks for both scale: macro and micro.

# What are the existing tools?

# Cloud specific

| Name | Creator | First public release date | Software license |
|---|---|---|---|
| AWS ParallelCluster | AWS | November 12, 2018 | Apache v2 |
| Azure CycleCloud | Microsoft | October 17, 2018 | MIT |
| Azure HPC On-Demand | Microsoft | April 23, 2021 | MIT |
| Google HPC-Toolkit | Google | May 26, 2022 | Apache v2 |
| Slurm on GCP | SchedMD | March 14, 2018 | Apache v2 |

# Multi-cloud

| Name | Creator | First public release date | Software license |
|---|---|---|---|
| Cluster in the Cloud | Matt Williams - University of Bristol | March 27, 2019 | MIT |
| ElastiCluster | Riccardo Murri - University of Zurich | July 17, 2013 | GPLv3 |

# Multi-cloud: supported providers

| Name | Alibaba Cloud | AWS | Azure | Google Cloud | Open Stack | Oracle Cloud | OVH |
|---|---|---|---|---|---|---|---|
| Cluster-in-the-Cloud | no | yes | no | yes | no | yes | no |
| ElastiCluster | no | yes | yes | yes | yes | no | - |

# Technologies

| Name | Infrastructure definition | Configuration management | Scheduler |
|---|---|---|---|
| AWS ParallelCluster | CLI generating YAML | Chef | Slurm |
| Azure CycleCloud | WebUI or CLI + templates | Chef | many |
| Azure HPC On-Demand | YAML files + shell scripts | Ansible, Packer | Open PBS, Slurm |
| Cluster in the Cloud | CLI generating TF code | Ansible, Packer | Slurm |
| ElastiCluster | CLI interpreting an INI file | Ansible | Grid Engine, Slurm |
| Google HPC-Toolkit | CLI generating TF code | Ansible, Packer | Slurm |
| Slurm GCP | Terraform modules | Ansible, Packer | Slurm |

# Why proposing another tool?

1. We wanted an open source multi-cloud solution that included OpenStack as a first class citizen.
2. We wanted Puppet to be the configuration management tool. Regional partners are Puppet-shops or at least familiar with it.
3. All cloud API interactions would have to be done by a third-party tool. No homemade CLI or wrapper.

# Designing an accessible tool for learning HPC

- Focus on re-creating the HPC environment
- Provide an accessible experience for beginners, with minimal prior HPC knowledge required
- Include key HPC features: job scheduling, data transfer, parallel and distributed computation, GPU, etc.
- Require minimal knowledge of cloud and minimal cost
- It should take a few minutes to setup a sandbox.

# Introduction to Magic Castle

***Open source infrastructure-as-code*** aiming to reproduce the HPC user experience in the cloud

# Design choices

- **Infrastructure**: 100% Terraform - no CLI or wrapper
  - A single interface to interact with all major cloud providers
- **Configuration**: cloud-init and Puppet
  - No knowledge of Puppet is required. The agent is autonomous.
- **Scheduler**: Slurm
  - Main scheduler used by the Alliance in Canada.
- **Cloud providers**: AWS, Azure, Google, OpenStack, OVH
  - Other providers can be added by following the documentation

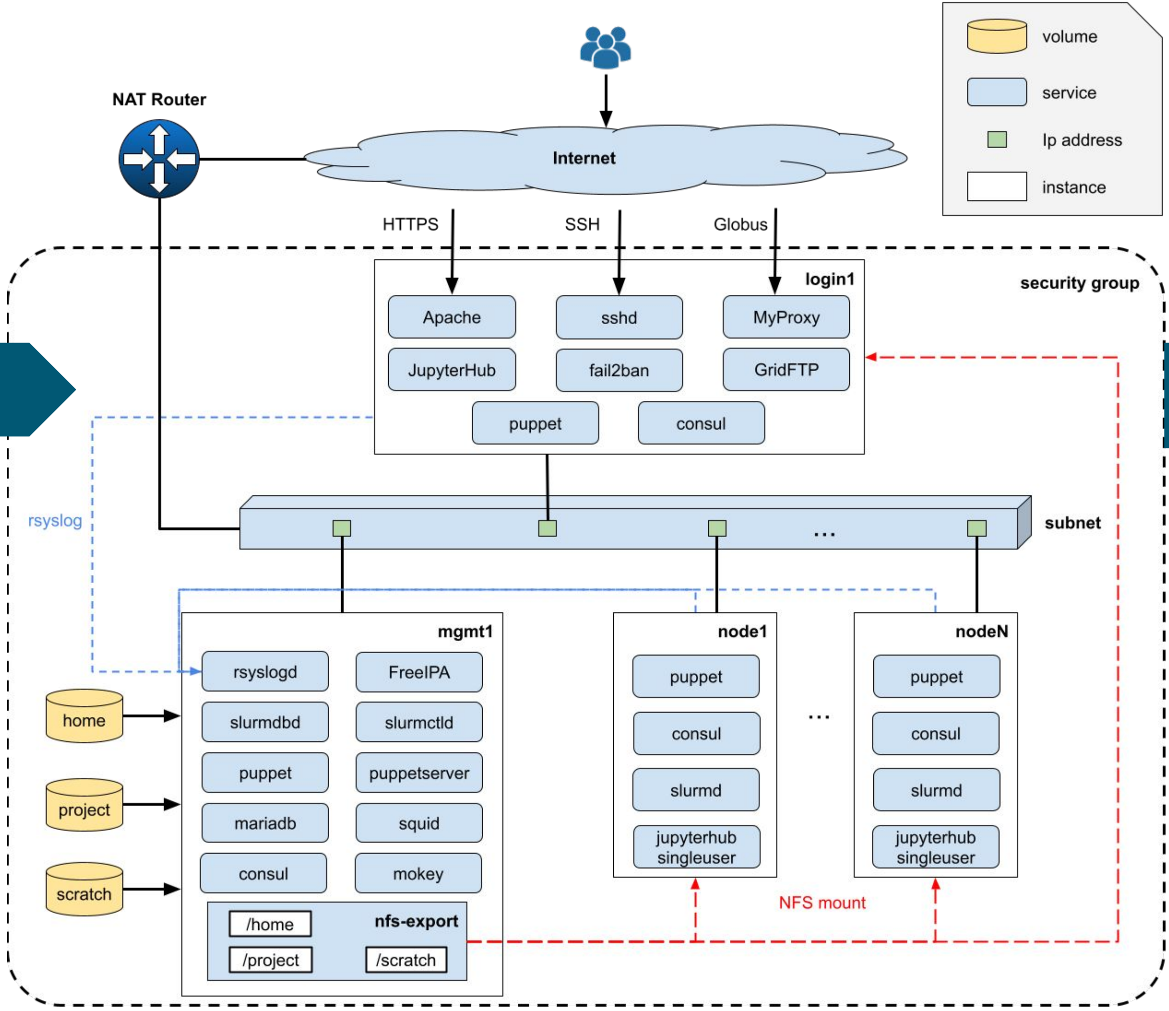
# Design choices

MAGIC CASTLE

- Spawn instances: management, login, compute, dtn, proxy, etc.
- Create volumes, network, network acls
- Create dns records
- Bootstrap passwords, certificates, secrets, keys, etc.
- Scale compute resources automatically based on job queue
- Customization via input parameters and YAML file

github.com/computecanada/magic_castle

plan



apply

aws

openstack

OVHcloud

puppet

configure

freeIPA
identity | policy | audit

slurm
workload manager

jupyterhub

Prometheus

Consul

Caddy

NVIDIA
CUDA

APPTAINER

globus

DUO

NOW

## Over 3000 scientific software are one “`module load`” away thanks to

Digital Research Alliance of Canada

EESSI
EUROPEAN ENVIRONMENT FOR
SCIENTIFIC SOFTWARE INSTALLATIONS

enjoy!

## Users can also install software using

Spack

# 1k+ workshops

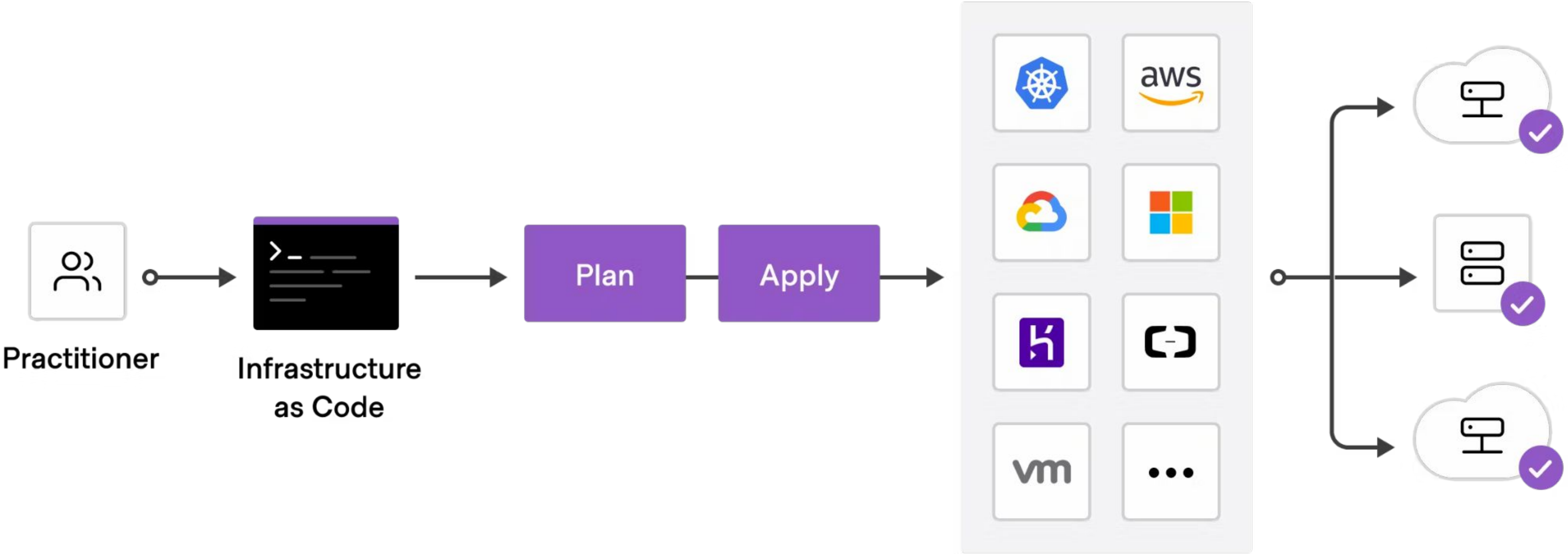
and university courses have used Magic Castle to teach advanced research computing since its initial release in 2018.

# How does it work?

# What is Terraform?

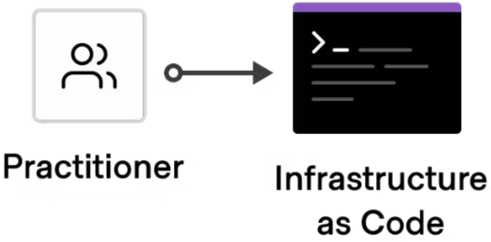
Terraform is an infrastructure-as-code software tool created by HashiCorp. Users define and provide data center infrastructure using a declarative configuration language known as HashiCorp Configuration Language (HCL).

# How does it work?



source: https://developer.hashicorp.com/terraform/tutorials/aws-get-started/infrastructure-as-code

# How does it work?

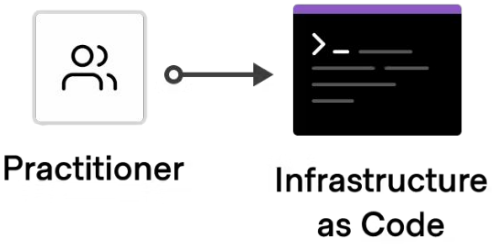


Practitioner

Infrastructure as Code

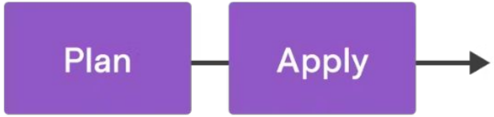
```
resource "openstack_compute_instance_v2" "mgmt01" {
  name            = "mgmt01"
  flavor_id       = "p4-6gb"
  key_pair        = "ssh-ed25519 ..."
  security_groups = ["default"]

  block_device {
    image_name            = "Rocky-8"
    source_type           = "image"
    volume_size           = "50"
    boot_index            = 0
    destination_type      = "volume"
    delete_on_termination = true
  }
}
```

# Infrastructure as code with higher level building blocks



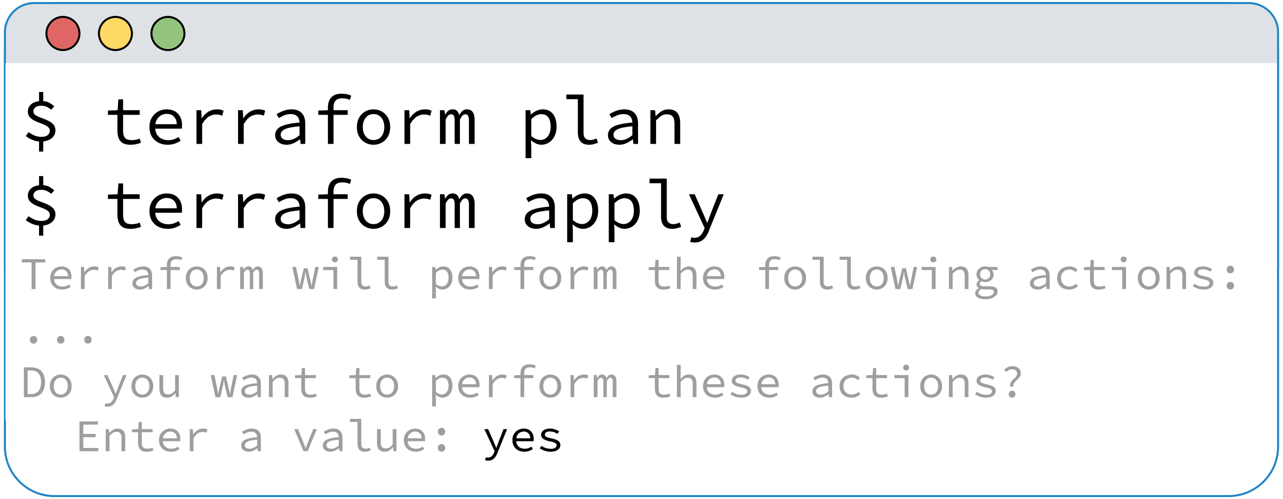
```
# IaC to create a Kubernetes cluster in GCP
module "gke" {
  source      = "..."
  project_id  = "<PROJECT ID>"
  name        = "gke-test-1"
  region      = "us-central1"
  zones       = ["us-central1-a"]
  network     = "vpc-01"
  http_load_balancing = false
  ...
}
```

# How does it work?



Plan

Apply



```
$ terraform plan
$ terraform apply
Terraform will perform the following actions:
...
Do you want to perform these actions?
  Enter a value: yes
```

# How useful is Terraform?

Terraform supports a number of cloud infrastructure providers such as Amazon Web Services, Cloudflare, Microsoft Azure, IBM Cloud, Serverspace, Google Cloud Platform, DigitalOcean, and OpenStack.

Combined with its ability to build infrastructure using high level building blocks, Terraform is an excellent choice for building complex environment like HPC clusters in the cloud.

# Installing Terraform

Terraform can be installed easily on all platforms as it is a single standalone Go binary.

You can download it from here :

https://developer.hashicorp.com/terraform/downloads

MAGIC CASTLE

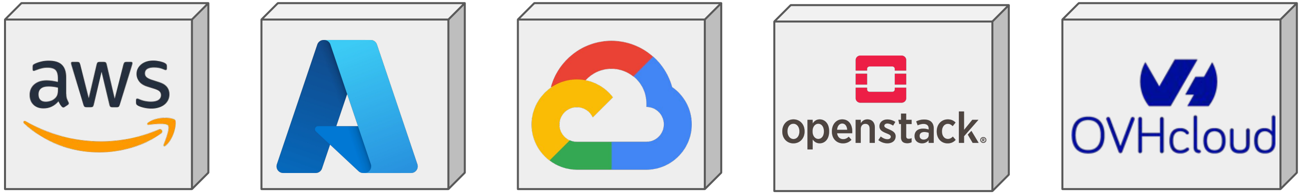
infrastructure-as-code ⇒

HashiCorp Terraform

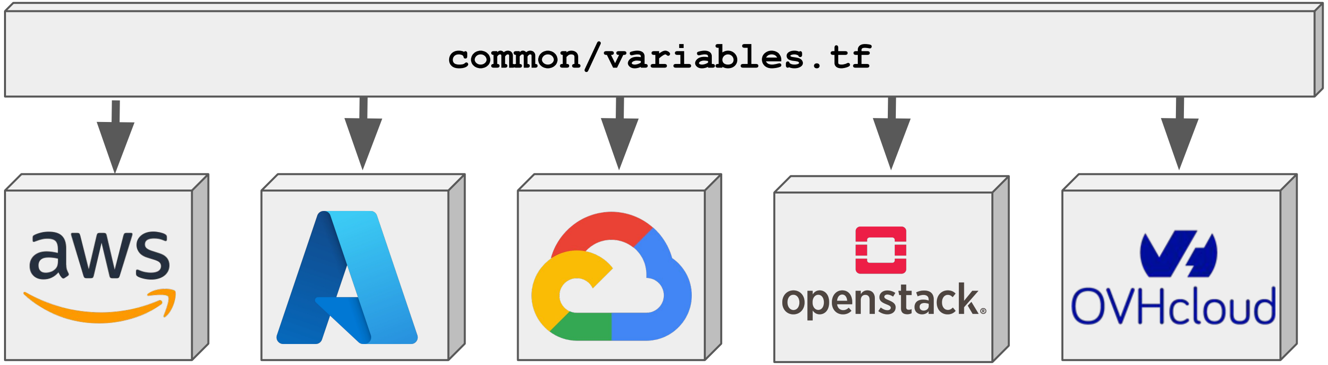
MAGIC CASTLE

HashiCorp Terraform

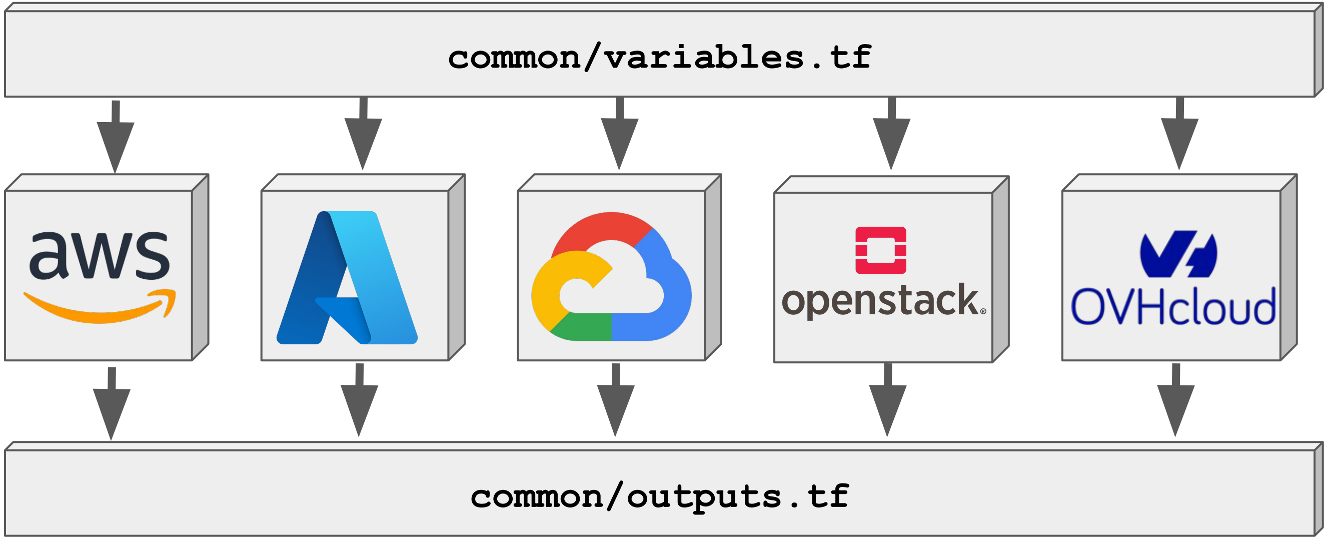
The infrastructure is defined in a main Terraform module. Each cloud provider has its dedicated main module:



## The main modules share common inputs:



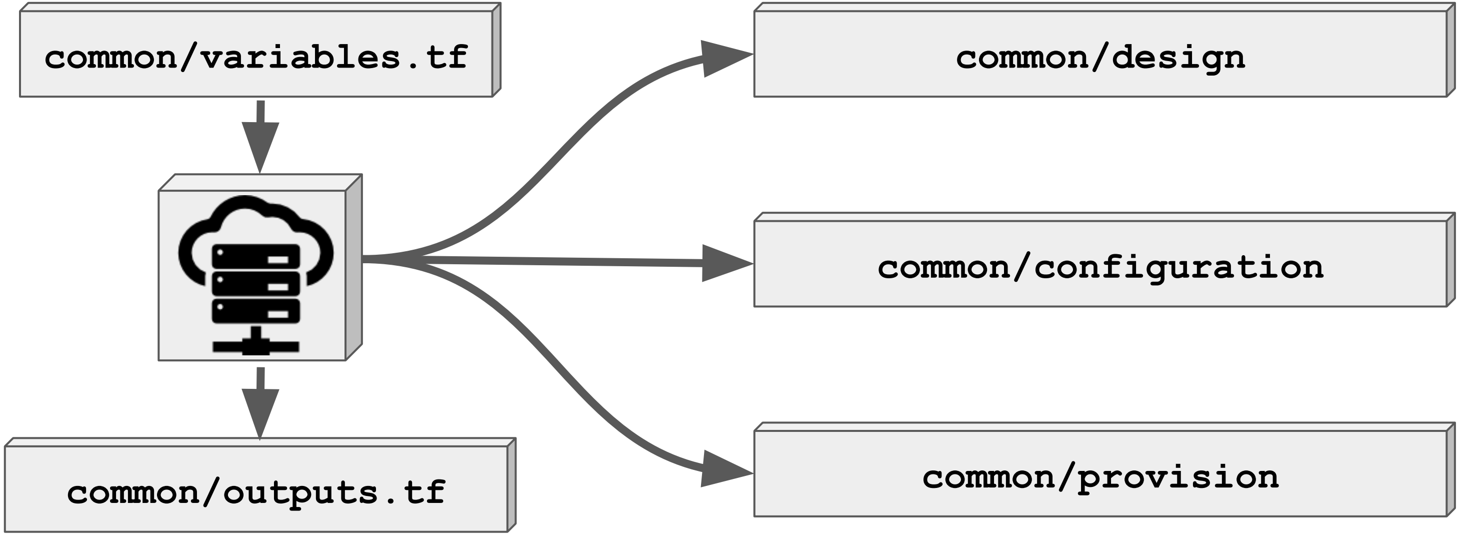
## And common outputs:



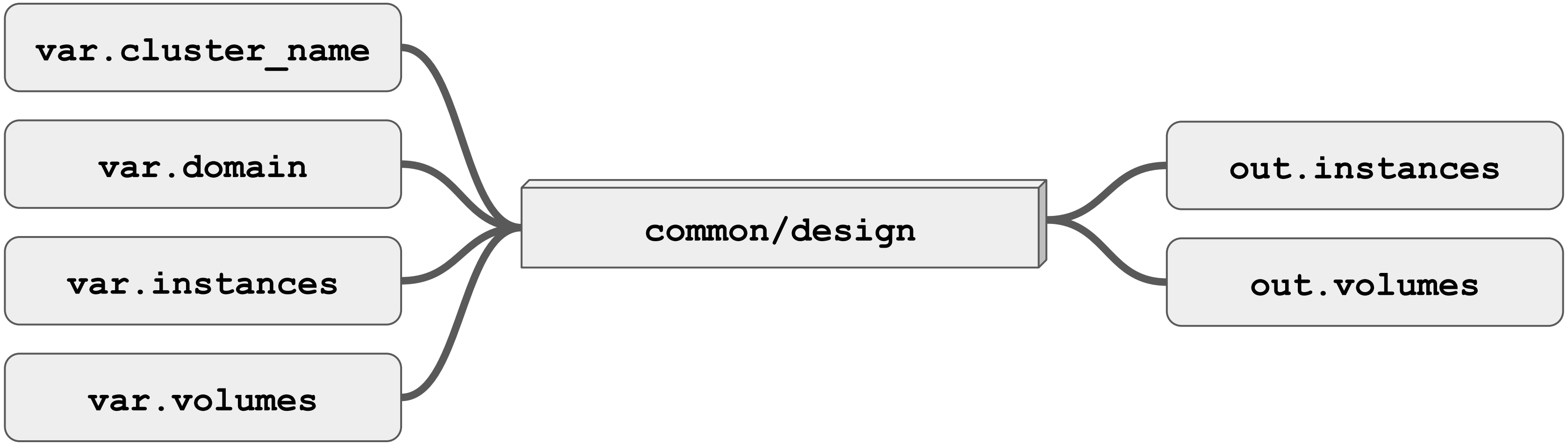
common/variables.tf

common/outputs.tf

Each main module uses 3 common sub-modules:



**`design`** sub-module transforms the inputs into **`maps`** used to generate the resources specific to each provider:



`var.cluster_name`

`var.domain`

`var.instances`

`var.volumes`

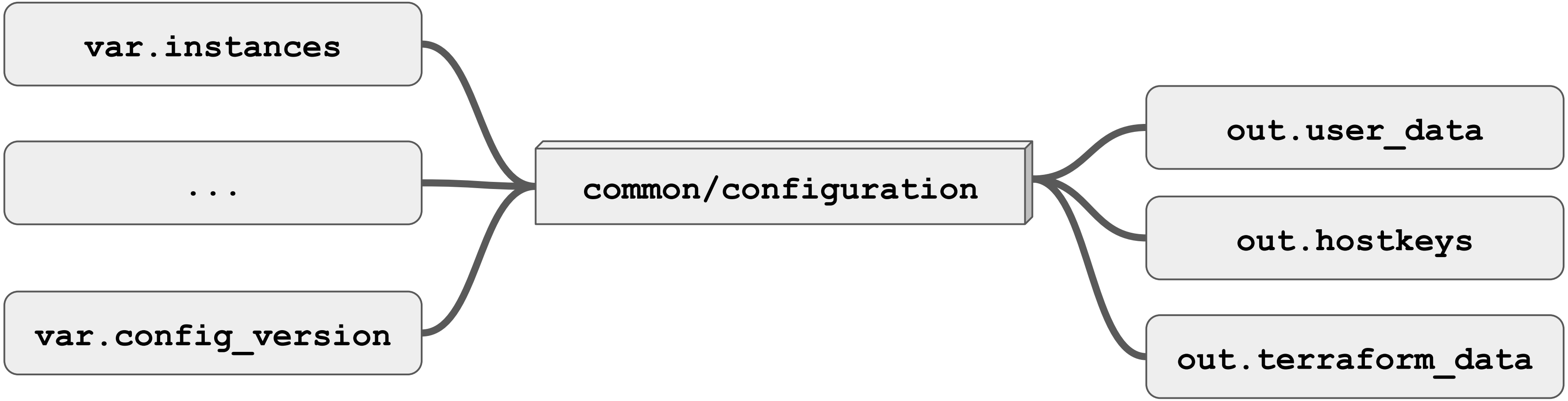
`common/design`

`out.instances`

`out.volumes`

MAGIC CASTLE

configuration sub-module creates the cloud-config file (user_data). This file configures SSH access and bootstraps Puppet on first boot.



var.instances

...

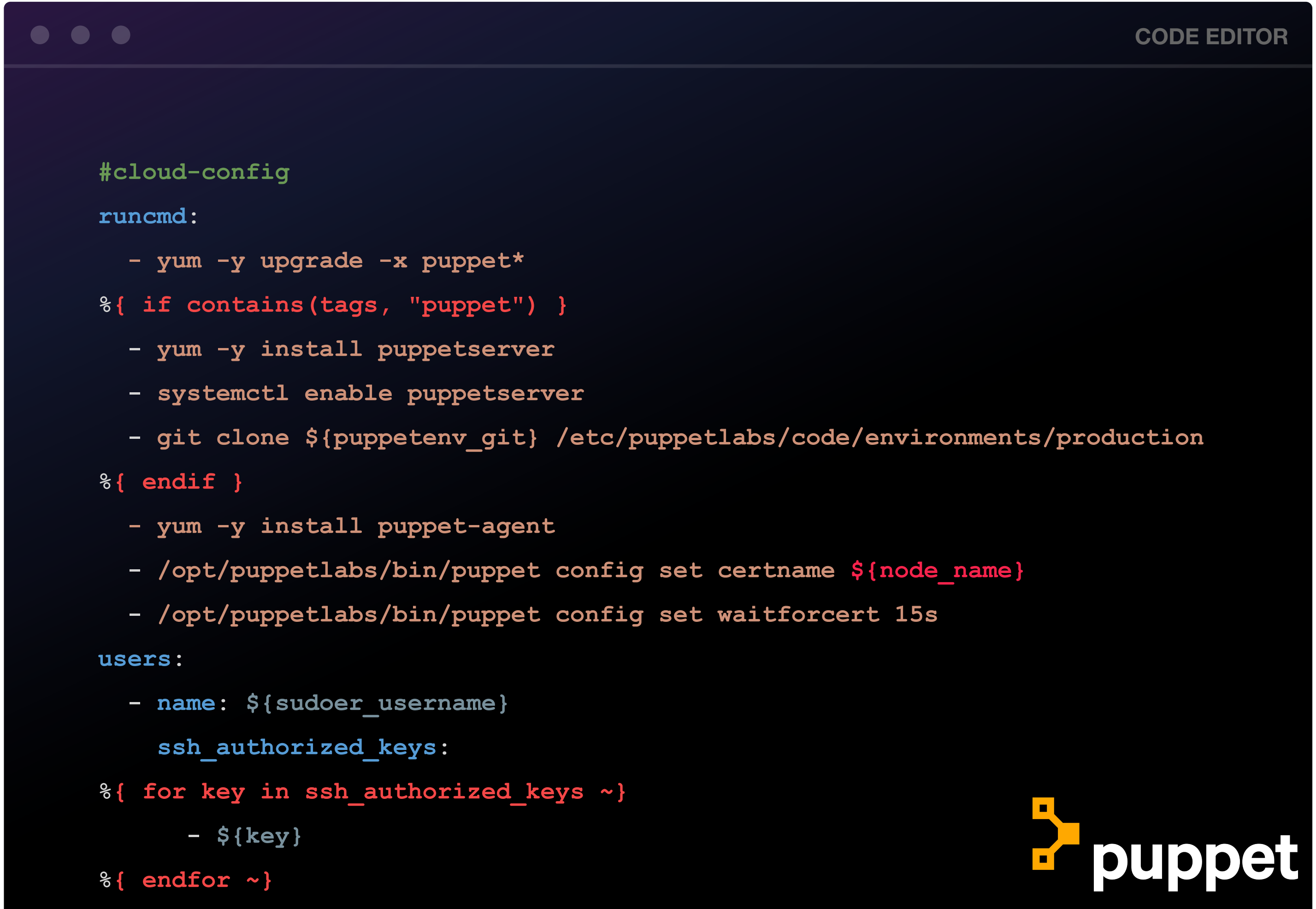
var.config_version

common/configuration

out.user_data

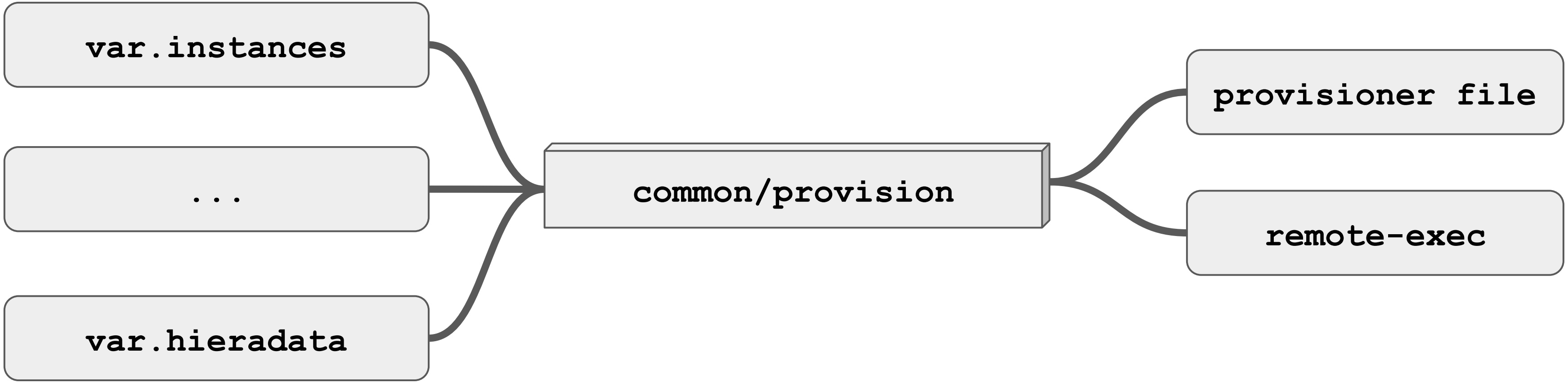
out.hostkeys

out.terraform_data



```
#cloud-config
runcmd:
  - yum -y upgrade -x puppet*
%{ if contains(tags, "puppet") }
  - yum -y install puppetserver
  - systemctl enable puppetserver
  - git clone ${puppetenv_git} /etc/puppetlabs/code/environments/production
%{ endif }
  - yum -y install puppet-agent
  - /opt/puppetlabs/bin/puppet config set certname ${node_name}
  - /opt/puppetlabs/bin/puppet config set waitforcert 15s
users:
  - name: ${sudoer_username}
    ssh_authorized_keys:
%{ for key in ssh_authorized_keys ~}
      - ${key}
%{ endfor ~}
```

`provision` copies the state (instances, #cpus, #gpus, volumes, etc.) via SSH to the Puppet server as a YAML file (`terraform_data.yaml`).



`var.instances`

`...`

`var.hieradata`

`common/provision`

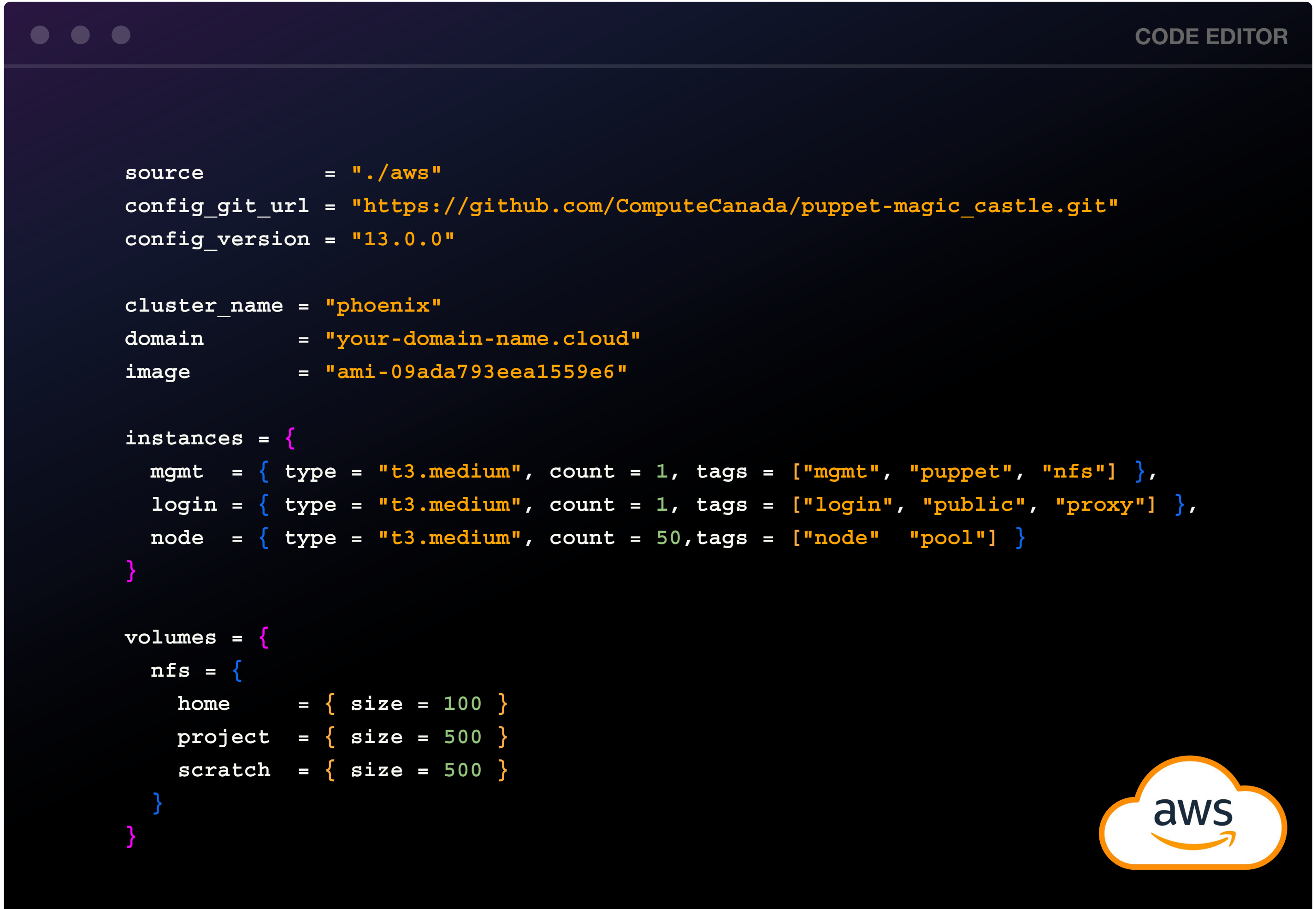
`provisioner file`

`remote-exec`

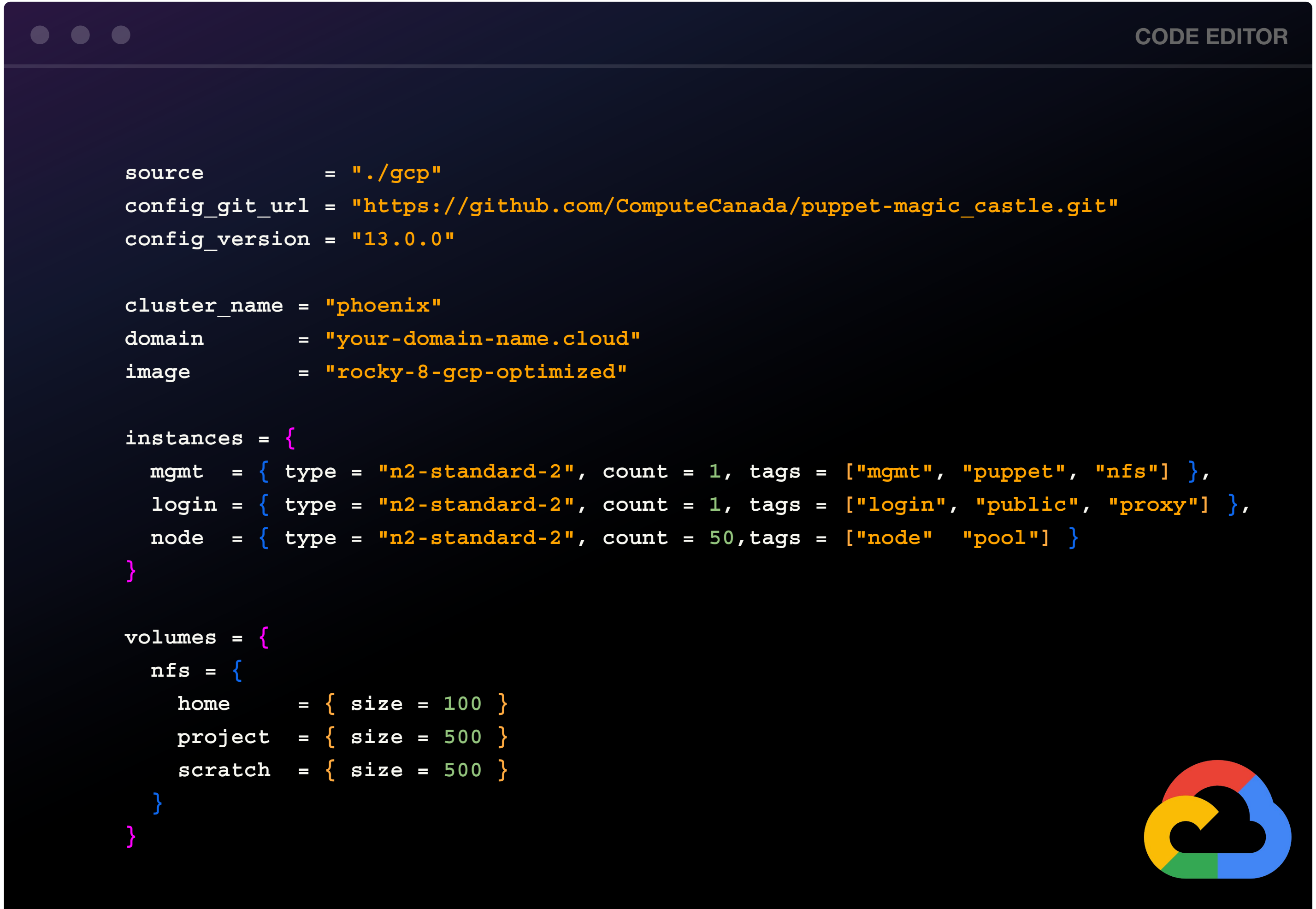
## terraform_data.yaml

```yaml
  "node4":
    "hostkeys":
      "ed25519": ssh-ed25519 …
      "rsa": ssh-rsa …
    "id": "droid-node4"
    "local_ip": "10.0.0.11"
    "public_ip": ""
    "specs": { "cpus": "2", "gpus": 0, "ram": "8000" }
    "tags": ["node", "pool"]
```

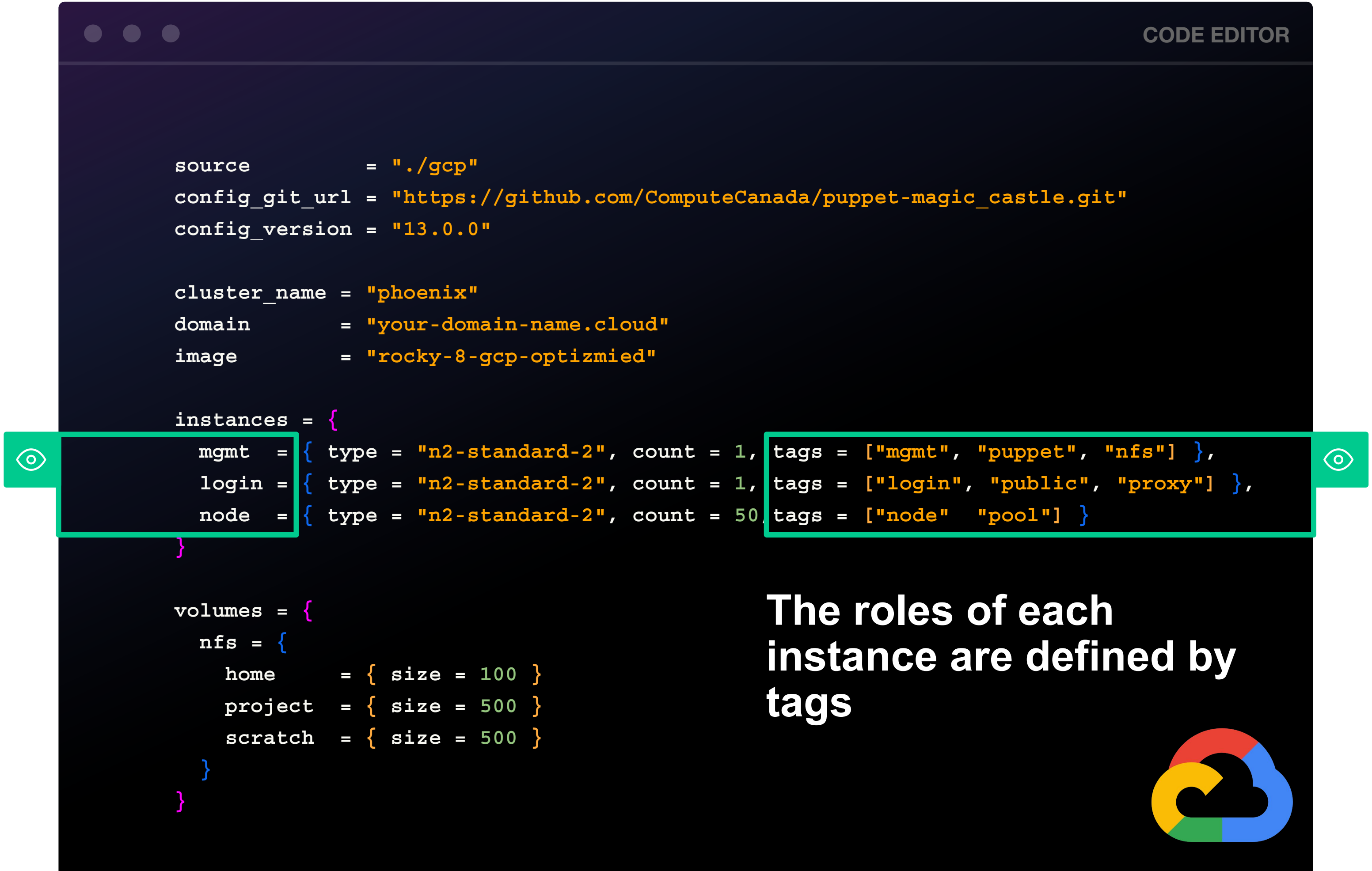
This set of common submodules creates an easy to use interface without vendor lock-in.



```
source         = "./aws"
config_git_url = "https://github.com/ComputeCanada/puppet-magic_castle.git"
config_version = "13.0.0"

cluster_name = "phoenix"
domain       = "your-domain-name.cloud"
image        = "ami-09ada793eea1559e6"

instances = {
  mgmt  = { type = "t3.medium", count = 1, tags = ["mgmt", "puppet", "nfs"] },
  login = { type = "t3.medium", count = 1, tags = ["login", "public", "proxy"] },
  node  = { type = "t3.medium", count = 50,tags = ["node"  "pool"] }
}

volumes = {
  nfs = {
    home     = { size = 100 }
    project  = { size = 500 }
    scratch  = { size = 500 }
  }
}
```

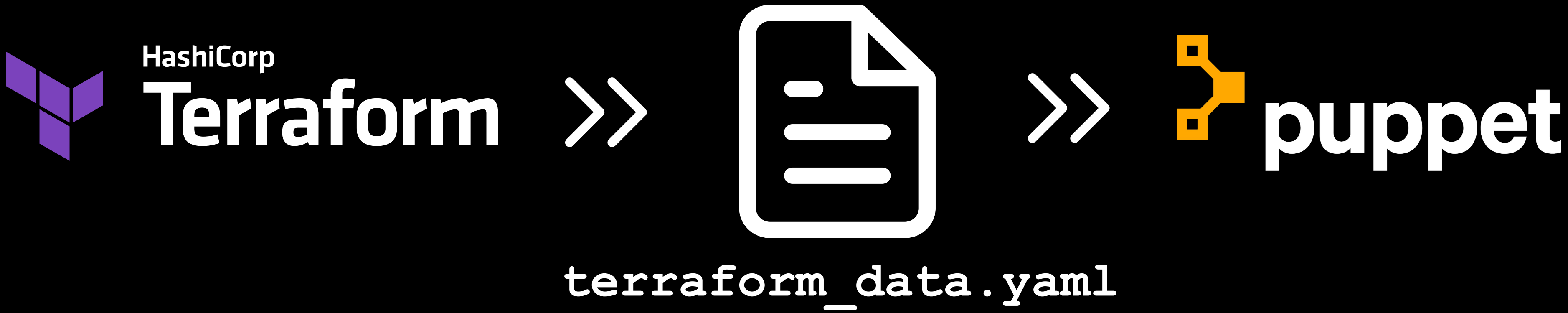


```
source         = "./gcp"
config_git_url = "https://github.com/ComputeCanada/puppet-magic_castle.git"
config_version = "13.0.0"

cluster_name = "phoenix"
domain       = "your-domain-name.cloud"
image        = "rocky-8-gcp-optimized"

instances = {
  mgmt  = { type = "n2-standard-2", count = 1, tags = ["mgmt", "puppet", "nfs"] },
  login = { type = "n2-standard-2", count = 1, tags = ["login", "public", "proxy"] },
  node  = { type = "n2-standard-2", count = 50,tags = ["node"  "pool"] }
}

volumes = {
  nfs = {
    home     = { size = 100 }
    project  = { size = 500 }
    scratch  = { size = 500 }
  }
}
```



```
source         = "./gcp"
config_git_url = "https://github.com/ComputeCanada/puppet-magic_castle.git"
config_version = "13.0.0"

cluster_name = "phoenix"
domain       = "your-domain-name.cloud"
image        = "rocky-8-gcp-optizmied"

instances = {
  mgmt   = { type = "n2-standard-2", count = 1, tags = ["mgmt", "puppet", "nfs"] },
  login  = { type = "n2-standard-2", count = 1, tags = ["login", "public", "proxy"] },
  node   = { type = "n2-standard-2", count = 50, tags = ["node"  "pool"] }
}

volumes = {
  nfs = {
    home     = { size = 100 }
    project  = { size = 500 }
    scratch  = { size = 500 }
  }
}
```

**The roles of each instance are defined by tags**

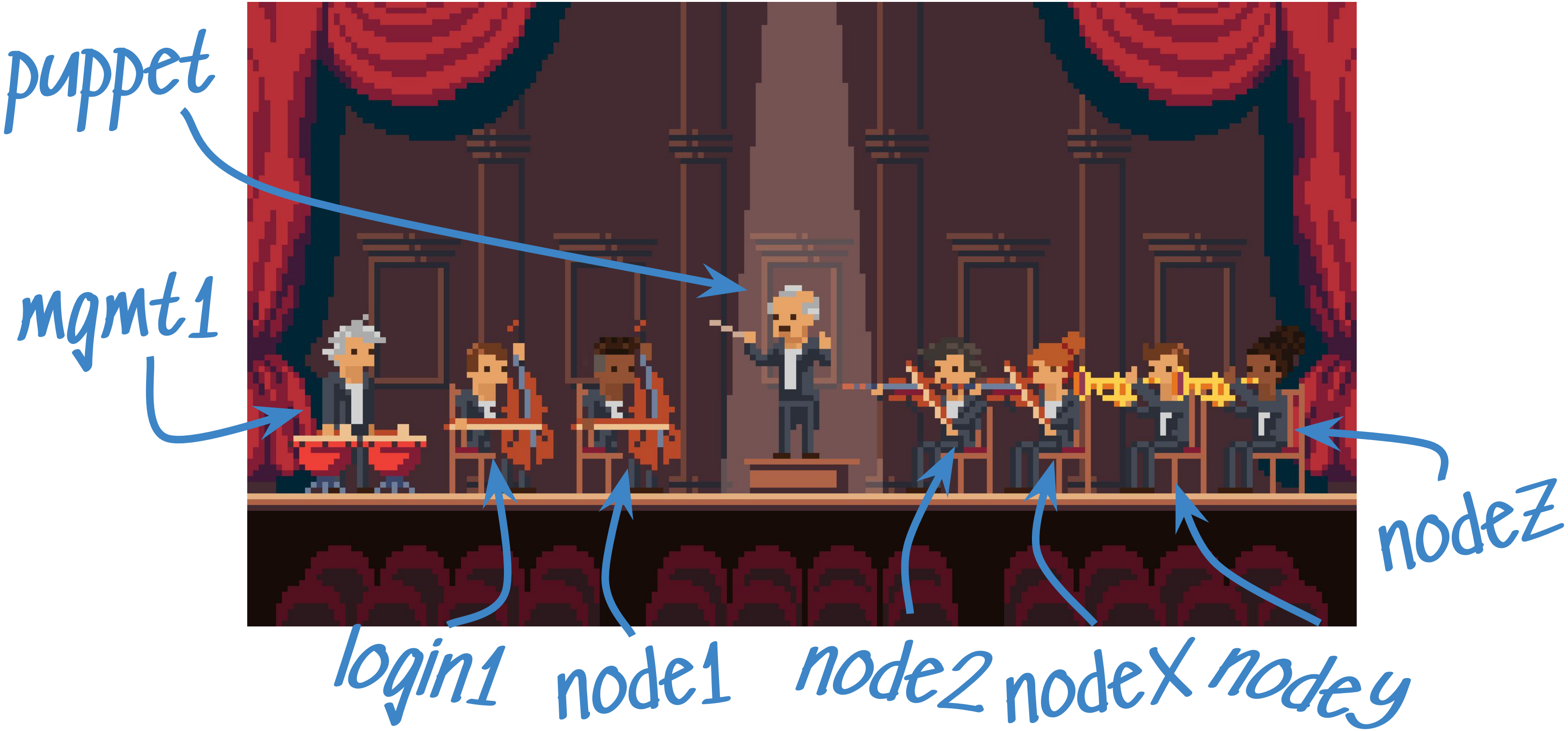


HashiCorp Terraform

terraform_data.yaml

puppet

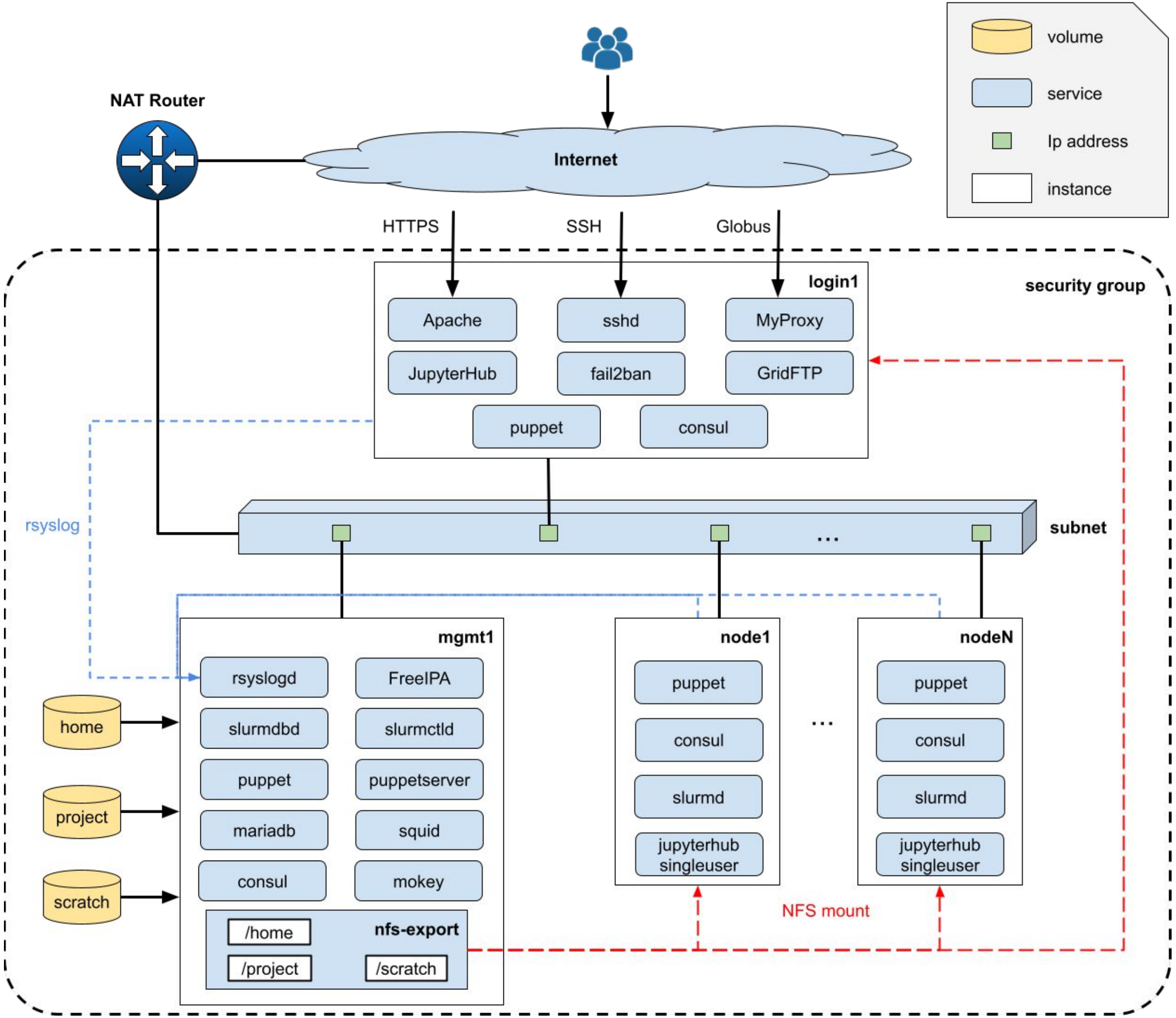
# Puppet manages the configuration



# Puppet configuration customization: YAML

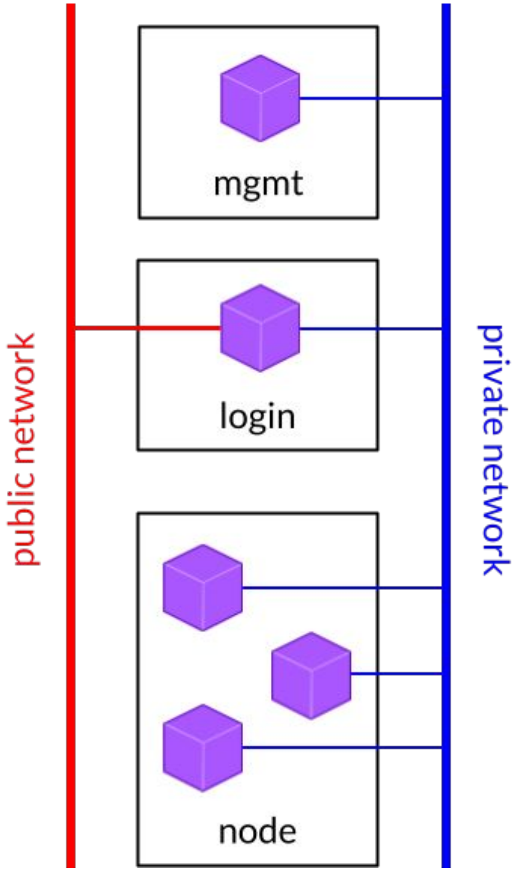
- Magic Castle configuration is done entirely through Puppet classes.
- There are over 40 classes that can be customized.
- Customization can happen before a cluster is launched or after.

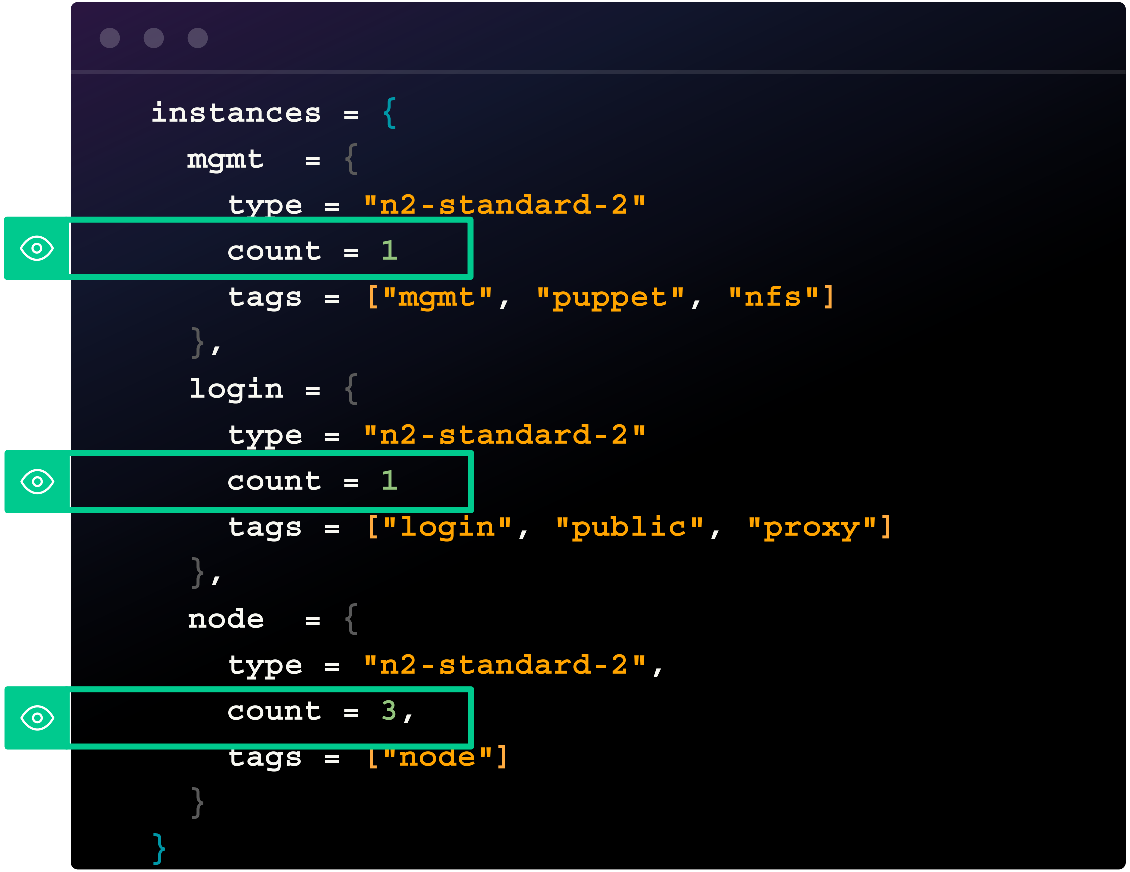
```
---
profile::users::ldap::users
  alice
    groups: ['engineering']
    public_keys: ['ssh-rsa ... user@local'  'ssh-ed25519 ...']
profile::fail2ban::ignoreip
    132.203.0.0/16
```


volume
service
Ip address
instance
NAT Router
Internet
HTTPS
SSH
Globus
security group
login1
Apache
sshd
MyProxy
JupyterHub
fail2ban
GridFTP
puppet
consul
rsyslog
subnet
...
mgmt1
rsyslogd
FreeIPA
slurmdbd
slurmctld
puppet
puppetserver
mariadb
squid
consul
mokey
nfs-export
/home
/project
/scratch
home
project
scratch
node1
puppet
consul
slurmd
jupyterhub singleuser
...
nodeN
puppet
consul
slurmd
jupyterhub singleuser
NFS mount

# Scaling & Autoscaling

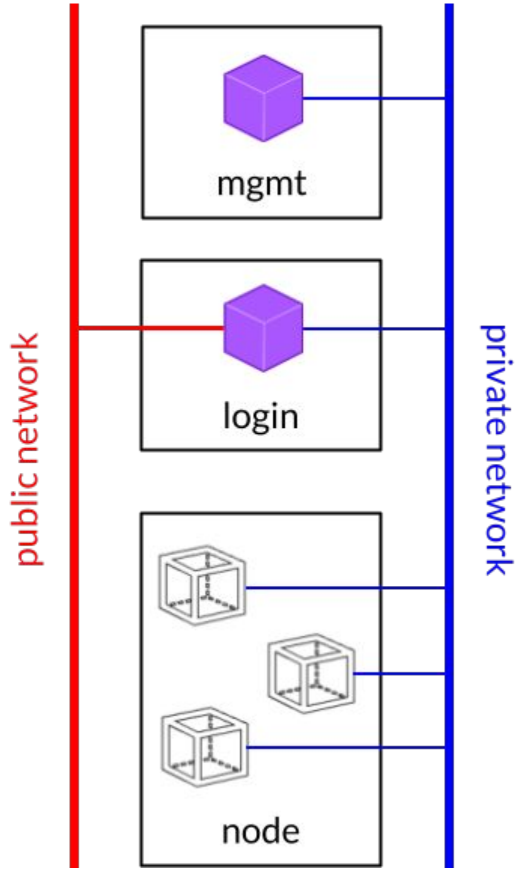
# Scaling

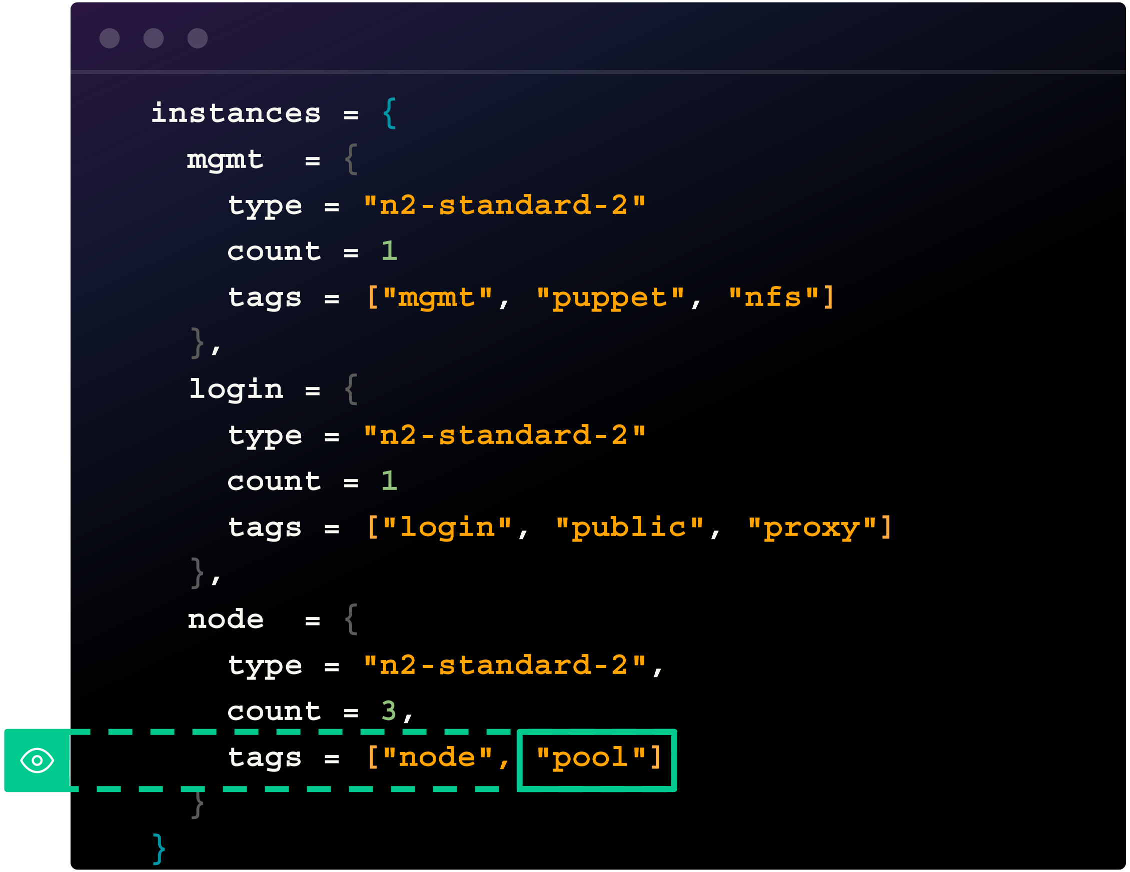




```
instances = {
  mgmt   = {
    type  = "n2-standard-2"
    count  = 1
    tags  = ["mgmt", "puppet", "nfs"]
  },
  login  = {
    type  = "n2-standard-2"
    count  = 1
    tags  = ["login", "public", "proxy"]
  },
  node   = {
    type  = "n2-standard-2",
    count  = 3,
    tags  = ["node"]
  }
}
```

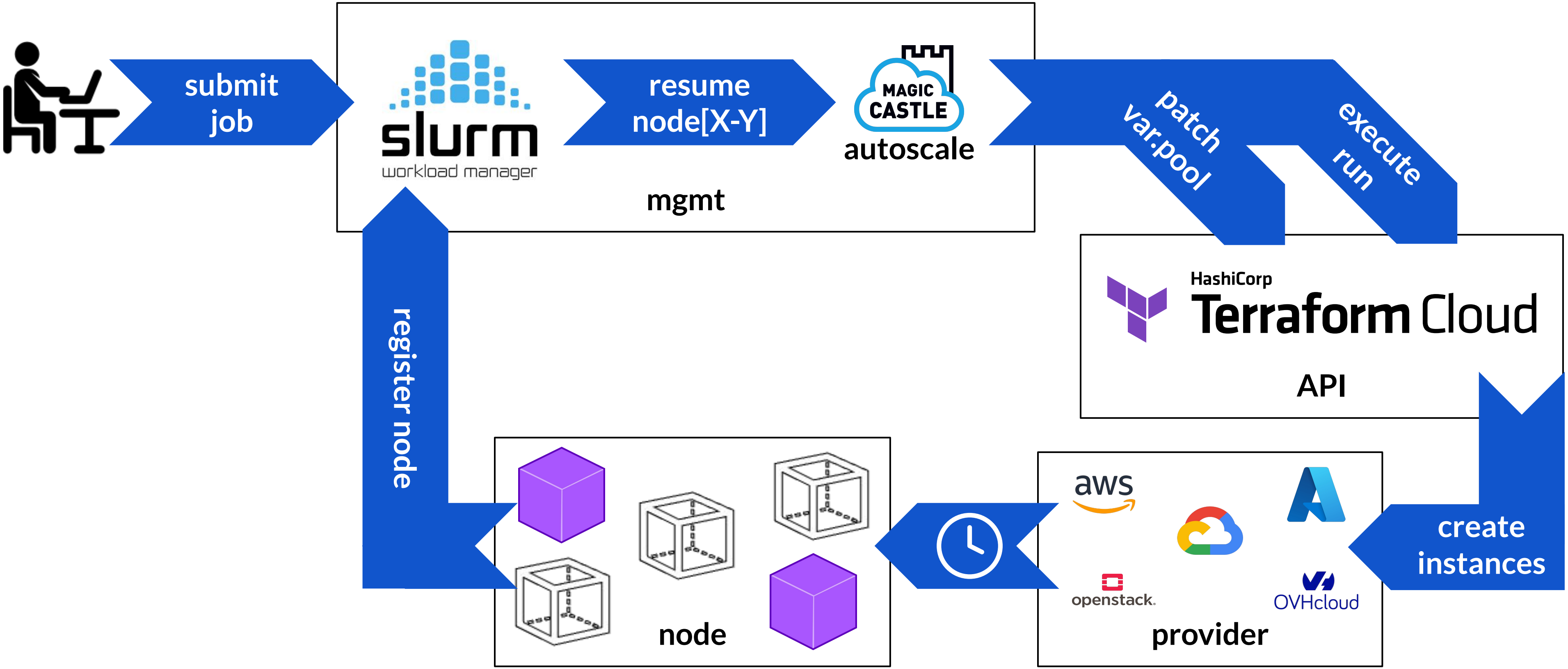
# Autoscaling



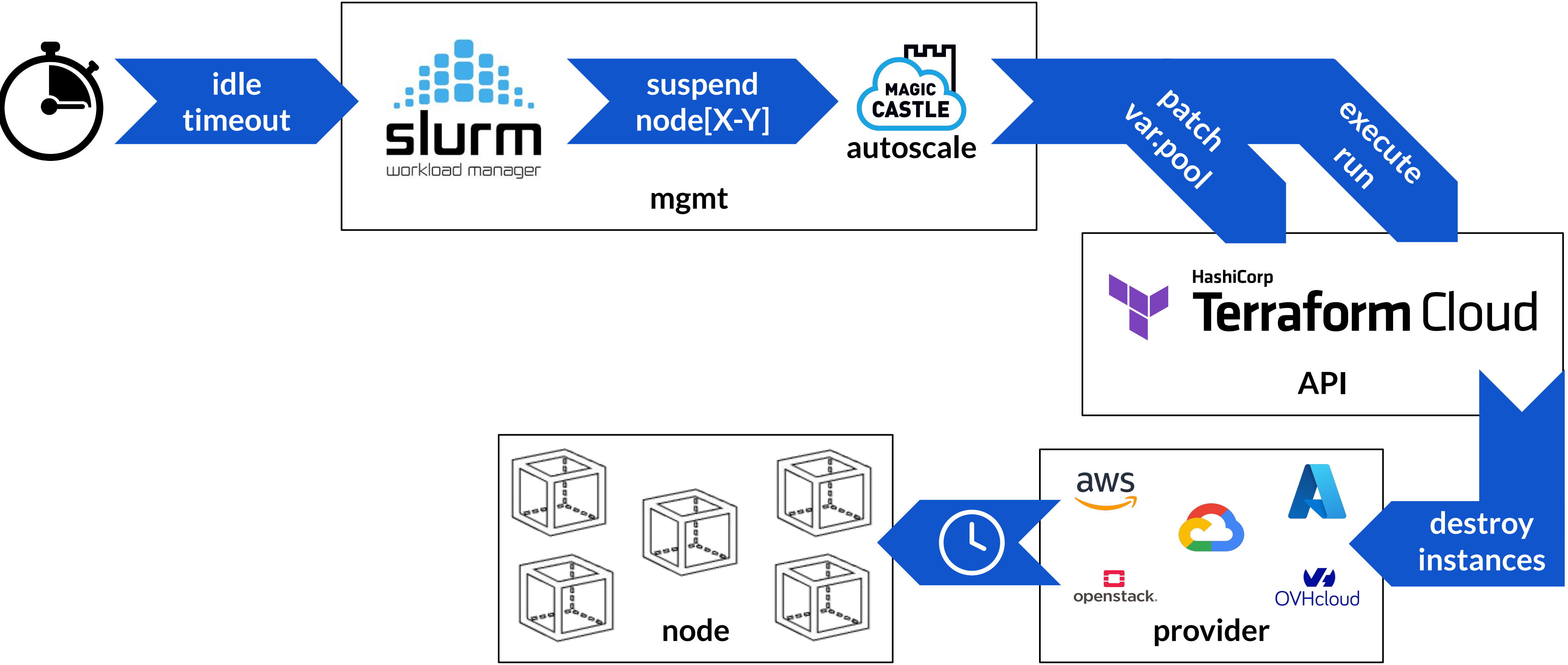


```
instances = {
  mgmt   = {
    type = "n2-standard-2"
    count = 1
    tags = ["mgmt", "puppet", "nfs"]
  },
  login = {
    type = "n2-standard-2"
    count = 1
    tags = ["login", "public", "proxy"]
  },
  node   = {
    type = "n2-standard-2",
    count = 3,
    tags = ["node", "pool"]
  }
}
```

# Autoscaling: resume



# Autoscaling: suspend



# What is Terraform Cloud ?

Terraform Cloud manages Terraform runs in a consistent and reliable environment, and includes easy access to shared state and secret data, access controls for approving changes to infrastructure.

Teams can connect Terraform to version control, share variables, run Terraform in a remote environment, and securely store remote state.

Terraform Cloud is available as a hosted service at https://app.terraform.io.

# Autoscaling

HashiCorp Terraform Cloud

1. Initialize a new git repository on GitLab or GitHub with the Magic Castle release for your cloud provider
2. Add `data.yaml` to the repo, you will use this with `main.tf` to define your cluster
3. Link the repo with a Terraform cloud workspace
4. Configure credentials for your providers, and a workspace variable: **pool = []**
5. Define the workspace ID and an API token in `data.yaml`
6. Launch the run execution in Terraform Cloud

**See autoscaling documentation**

▷ The autoscaling logic is *cloud-agnostic* and is expressed in 100 lines of Python.

▷ The API token requires only 2 permissions: modify a variable and create a plan.

▷ The compute nodes can be heterogeneous (GPU, x86, ARM64). Slurm determines which nodes to power-up based on the job queue.

# IN THE WILD

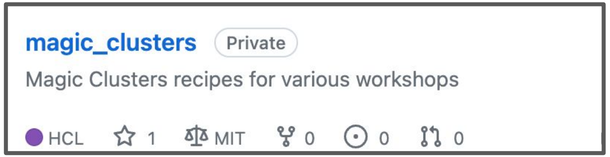
Calcul Québec

A **regional partner** of the

**Digital Research Alliance** of Canada

- Uses Magic Castle as the hands-on exercise platform for their entire 2023-2024 training program

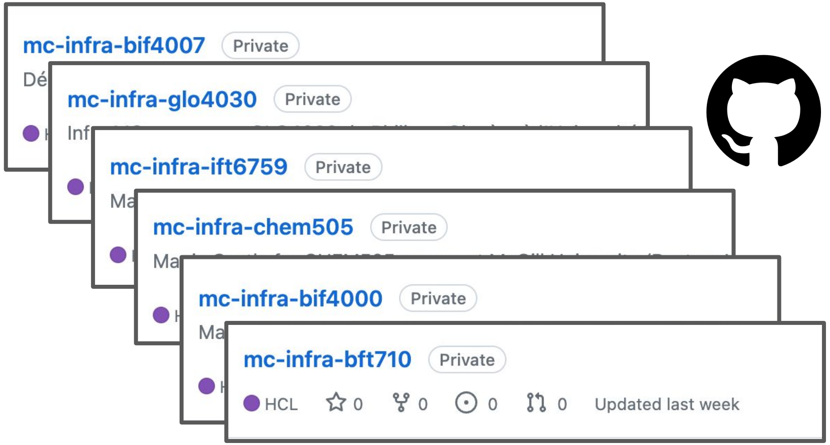
- Provides and administers Magic Castle clusters to graduate courses from various disciplines: AI, bioinformatics, neuroscience



magic_clusters Private

Magic Clusters recipes for various workshops

● HCL ☆ 1 MIT 0 0 0



mc-infra-bif4007 Private

Dé

mc-infra-glo4030 Private

Inf

mc-infra-ift6759 Private

Ma

mc-infra-chem505 Private

Ma

mc-infra-bif4000 Private

Ma

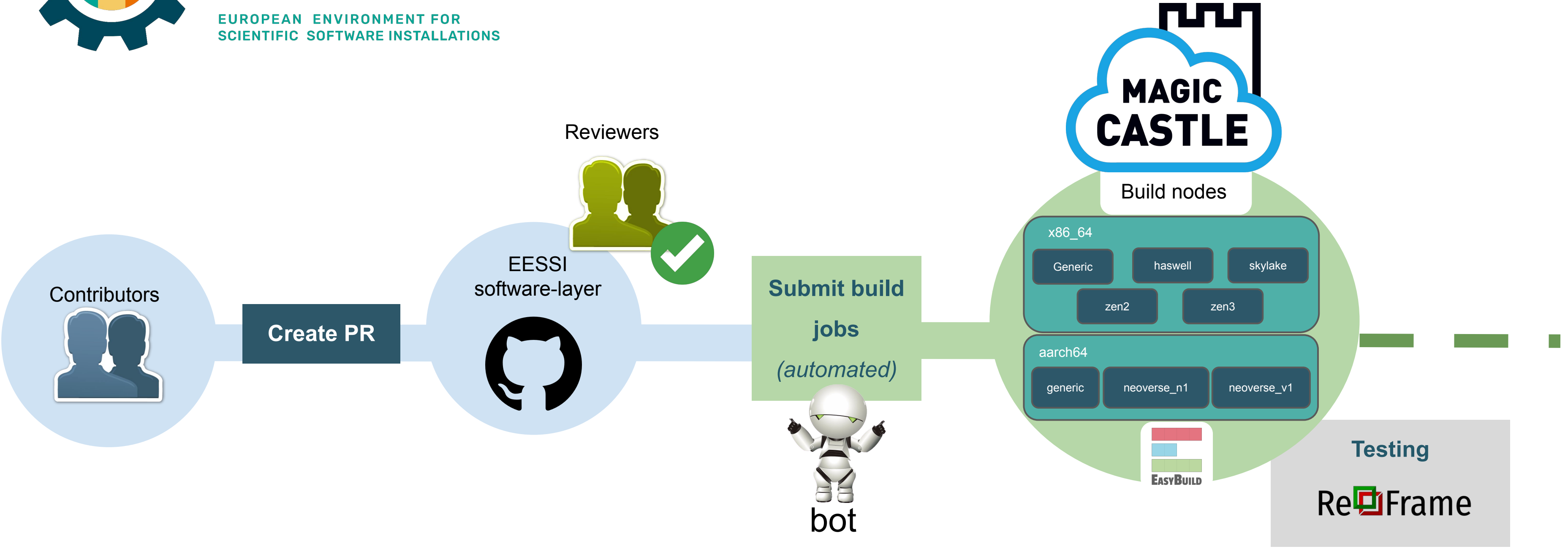
mc-infra-bft710 Private

● HCL ☆ 0 0 0 0 Updated last week

EESSI

EUROPEAN ENVIRONMENT FOR SCIENTIFIC SOFTWARE INSTALLATIONS

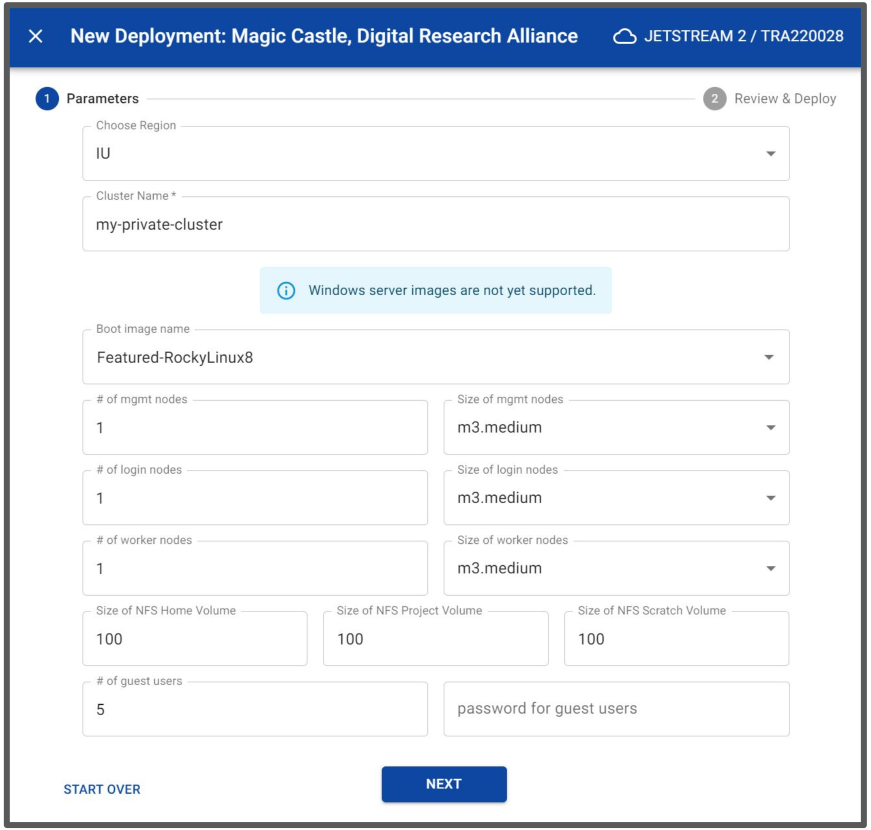
uses Magic Castle as its platform to compile and test software built with EasyBuild before deploying them on CVMFS



Submit build jobs *(automated)*

https://www.eessi.io/

Magic Castle is integrated in CACAO and can be launched easily in Jetstream cloud.



New Deployment: Magic Castle, Digital Research Alliance

JETSTREAM 2 / TRA220028

1 Parameters — 2 Review & Deploy

Choose Region: IU

Cluster Name *: my-private-cluster

ⓘ Windows server images are not yet supported.

Boot image name: Featured-RockyLinux8

# of mgmt nodes: 1 — Size of mgmt nodes: m3.medium

# of login nodes: 1 — Size of login nodes: m3.medium

# of worker nodes: 1 — Size of worker nodes: m3.medium

Size of NFS Home Volume: 100 — Size of NFS Project Volume: 100 — Size of NFS Scratch Volume: 100

# of guest users: 5 — password for guest users

https://docs.jetstream-cloud.org/ui/cacao/deployment_magic_castle/

MAGIC CASTLE

- ★ Simple to use
- ★ Autoscaling infrastructure
- ★ Ideal software environment to teach and learn HPC

**cloud-agnostic** and **open source**

https://www.github.com/computecanada/magic_castle