MAGIC CASTLE

Canadian HPC as a Service

Félix-Antoine Fortin

Principal developer of Magic Castle
Digital Research Alliance of Canada

Director of software development
Calcul Québec

felix@calculquebec.ca

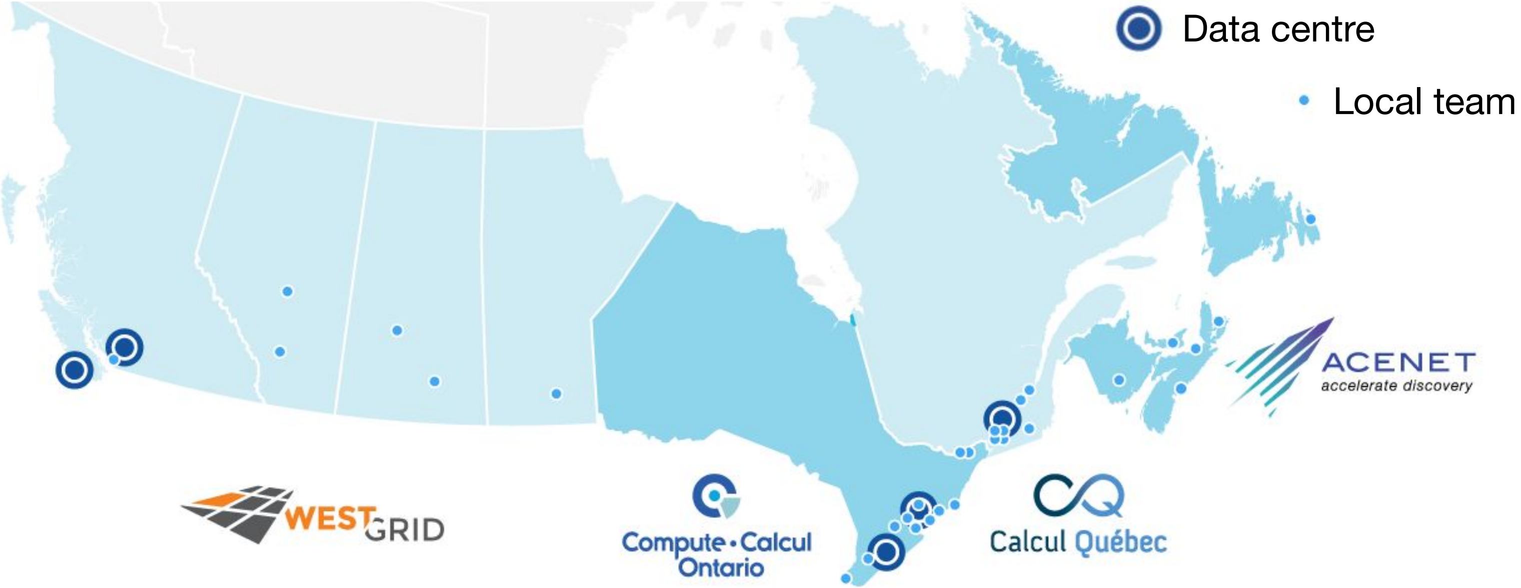
UNIVERSITÉ LAVAL

Magic Castle - Canadian HPC as a Service

1. Genesis
2. Technical overview
3. Variety of use cases

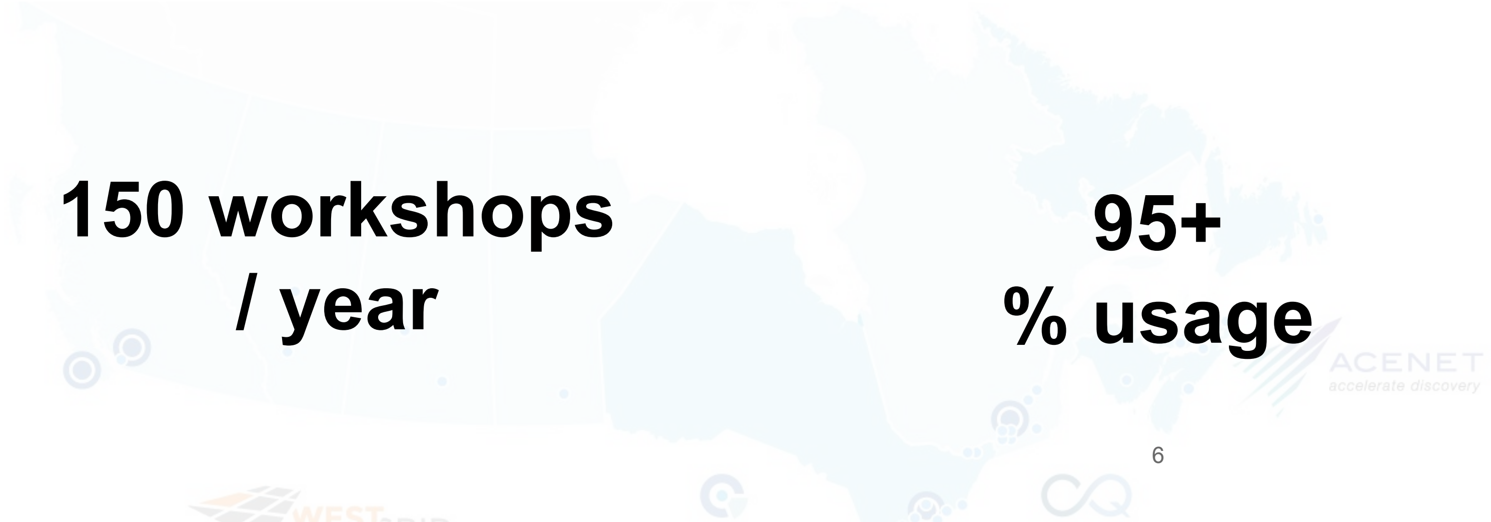
Magic Castle Genesis

High Performance Computing (HPC)
Research infrastructure landscape in Canada



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

High Performance Computing (HPC)
Research infrastructure landscape in Canada



150 workshops / year

95+ % usage

How to train users at scale without impacting research?

Design an accessible tool for learning HPC

- Focus on recreating the Alliance HPC environment
- Include key features:
  - Slurm
  - Scientific software stack
  - GPU support
- Minimal IT administration knowledge required
- Quisk setup - few minutes

We want accessible, inexpensive sandbox environments, designed to facilitate teaching to audiences of various sizes.

It should be as easy as Legos…
for adults.

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

Technical Overview

Imagine you are a wizard and you want to build a new castle.

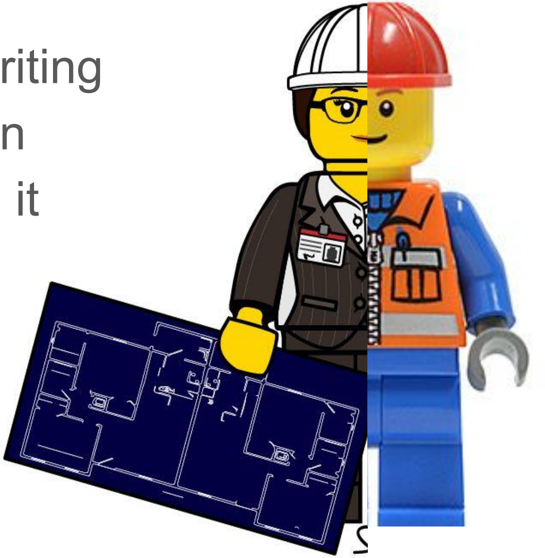
You don’t know much about building castles and/or you already have enough on your plate defeating dark forces.

If only there was someone able to take care of it all for you…

MAGIC CASTLE

Part architect :

- Puts your needs in writing
- Don’t need a dungeon right now? Can close it down temporarily



Part foreman :

- Manages the construction site
- Monitors and fixes problems regularly

With the best social skills! Will set up your castle anywhere

Design choices

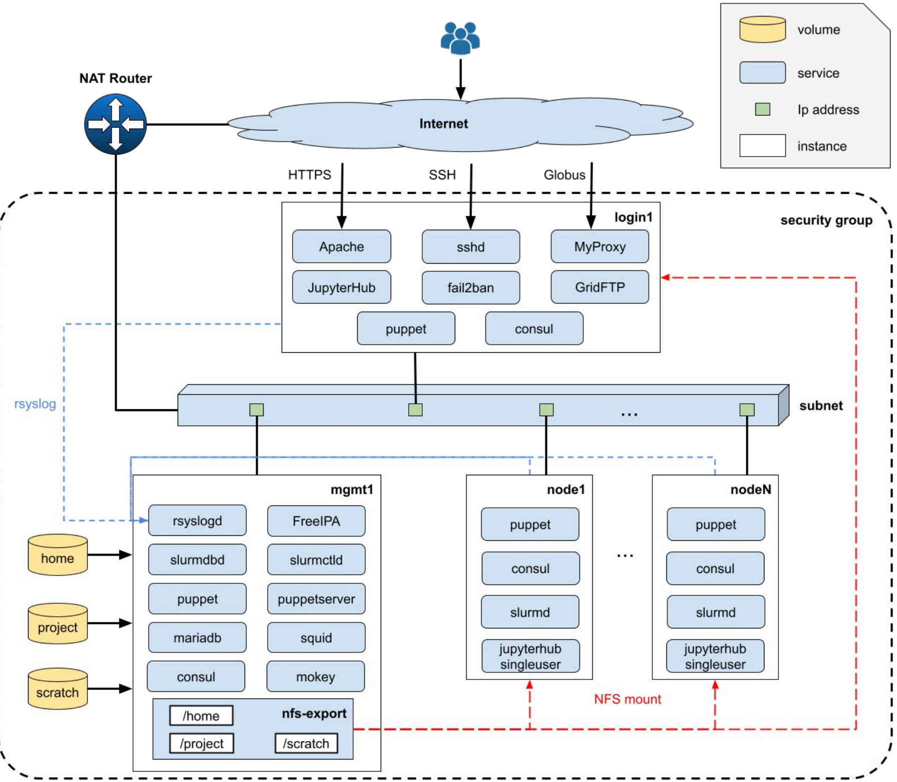
- **Infrastructure**: 100% Terraform
  - No CLI or wrapper, no API interaction
  - A single interface to interact with all major cloud providers
- **Configuration**: cloud-init and Puppet
  - No knowledge of Puppet is required. The agent is autonomous.
- **Scheduler**: Slurm
  - Support dynamic nodes
  - Main scheduler used by the Alliance in Canada.

Design choices

- **Cloud providers**: AWS, Azure, Google, OpenStack, OVH
    - Other providers can be added by following the documentation
- **Provider agnostic autoscaling**
- **Curated solution** that still allows customization
    - via input parameters and YAML file

https://github.com/computecanada/magic_castle

plan



apply

aws

openstack.

OVHcloud

configure

slurm
workload manager

freeIPA
identity | policy | audit

APPTAINER

globus

jupyterhub

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

How does it work?

plan

apply

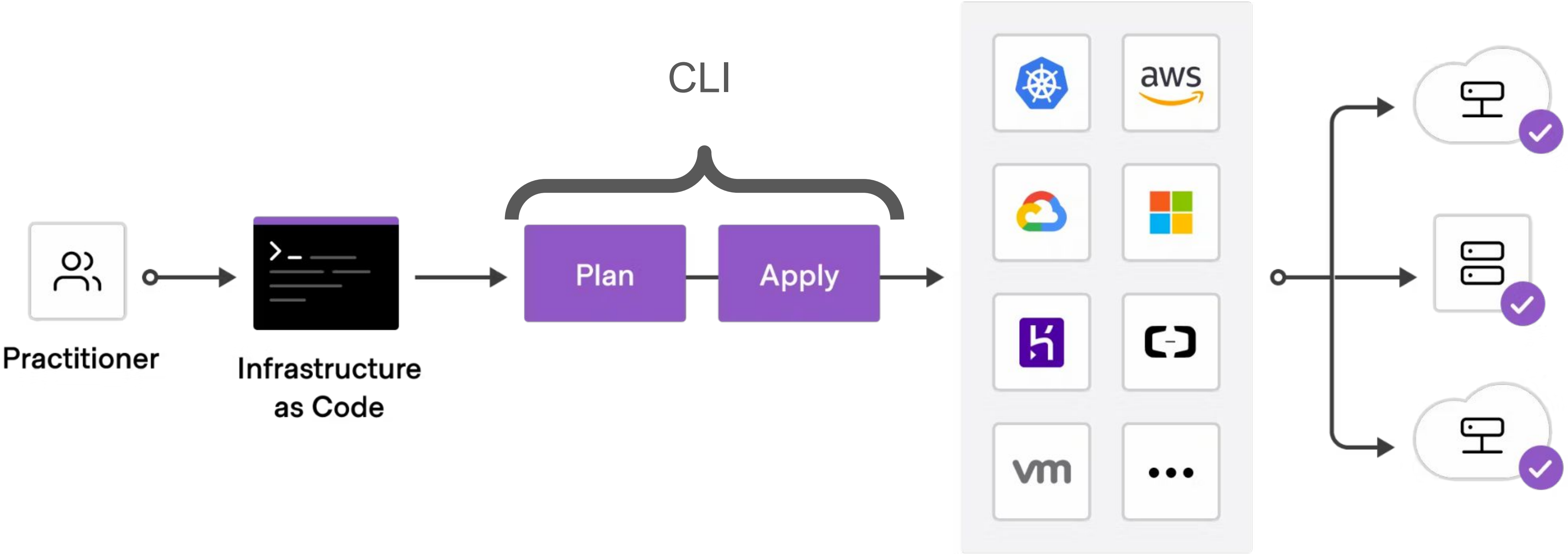
configure

What is Terraform?

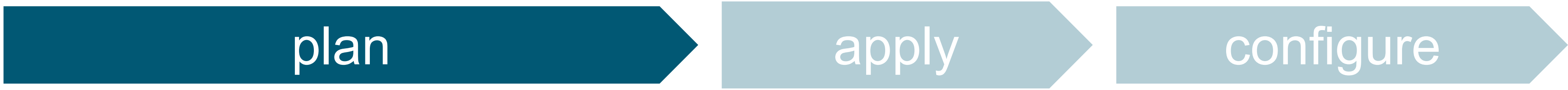
Terraform is an infrastructure-as-code software tool. Users define and provide data center infrastructure using a declarative configuration language(HCL).

It supports a number of cloud infrastructure providers such as AWS, Microsoft Azure, Google Cloud Platform, and OpenStack.

How does it work?



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



plan

apply

configure

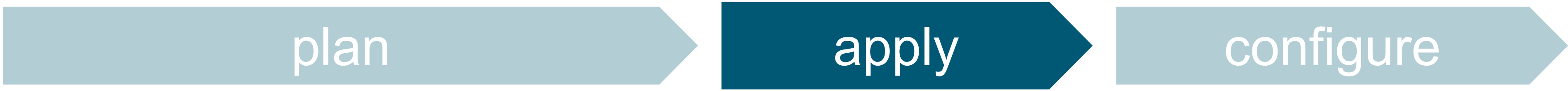
```
resource "openstack_compute_instance_v2" "mgmt01" {
  name            = "mgmt01"
  flavor_id       = "p4-6gb"
  key_pair        = "ssh-ed25519 ..."
  security_groups = ["default"]

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

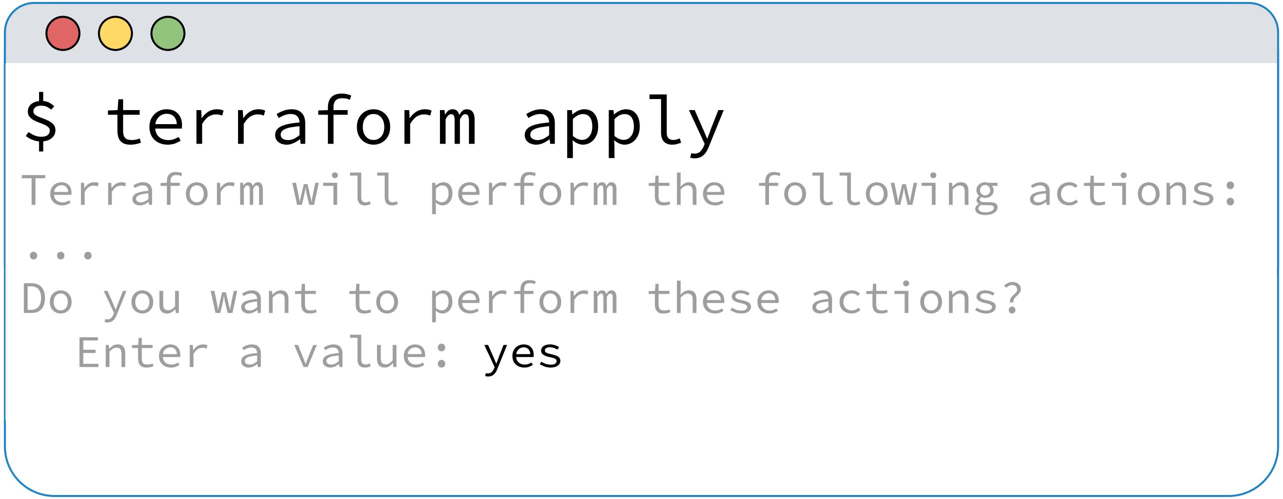
plan

apply

configure

```
# 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
  ...
}
```



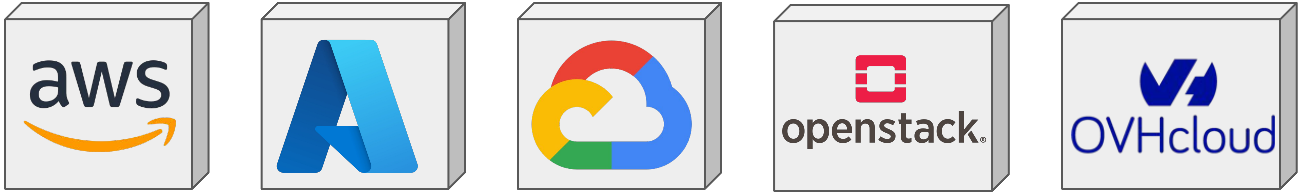


```
$ terraform apply
Terraform will perform the following actions:
...
Do you want to perform these actions?
  Enter a value: yes
```

MAGIC CASTLE

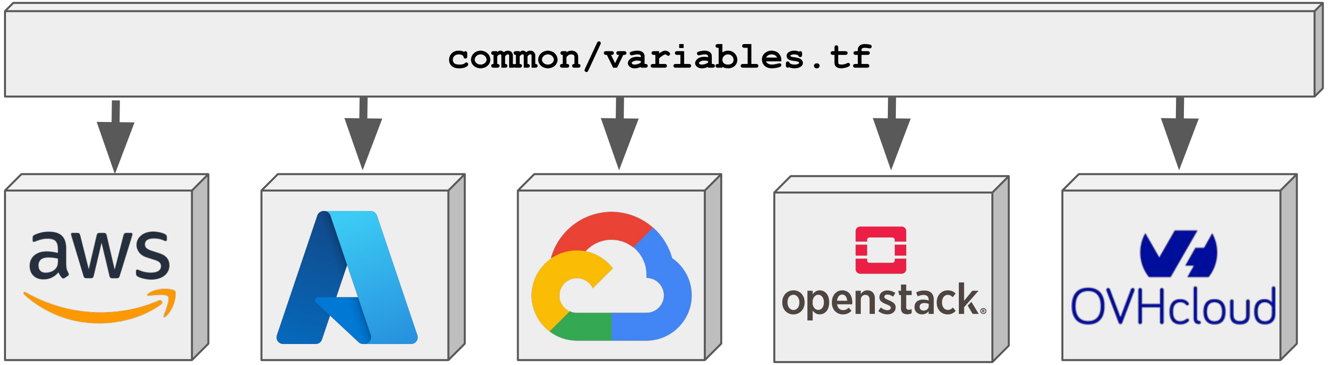
HashiCorp Terraform

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

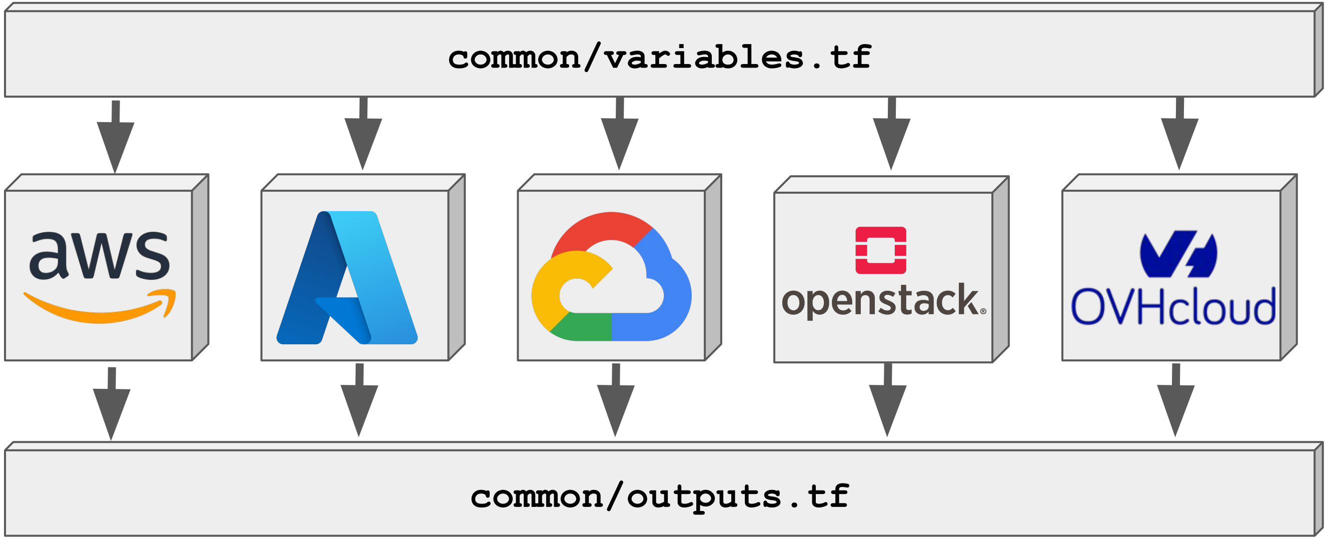


MAGIC CASTLE

The main modules share common inputs:



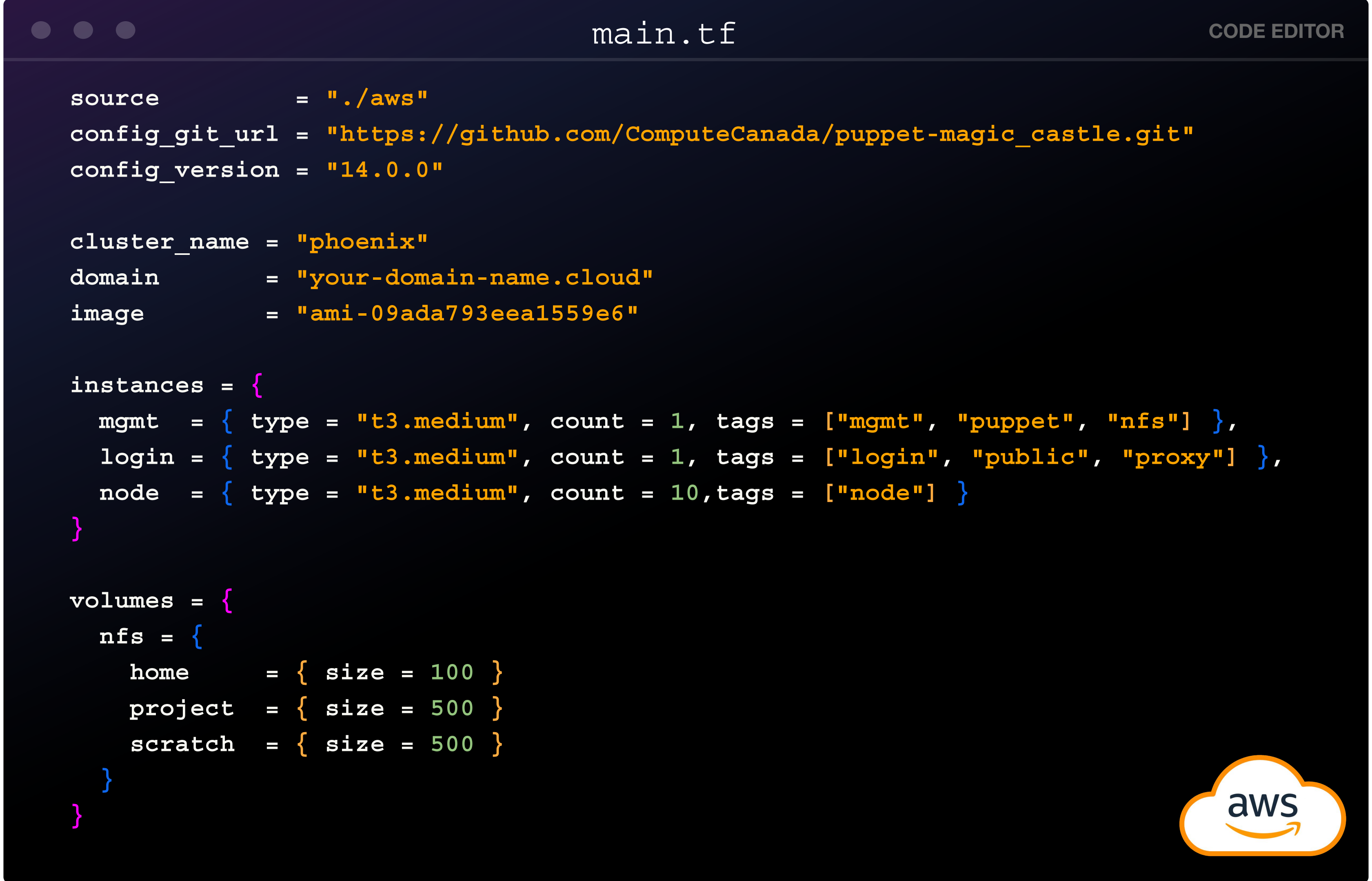
And common outputs:



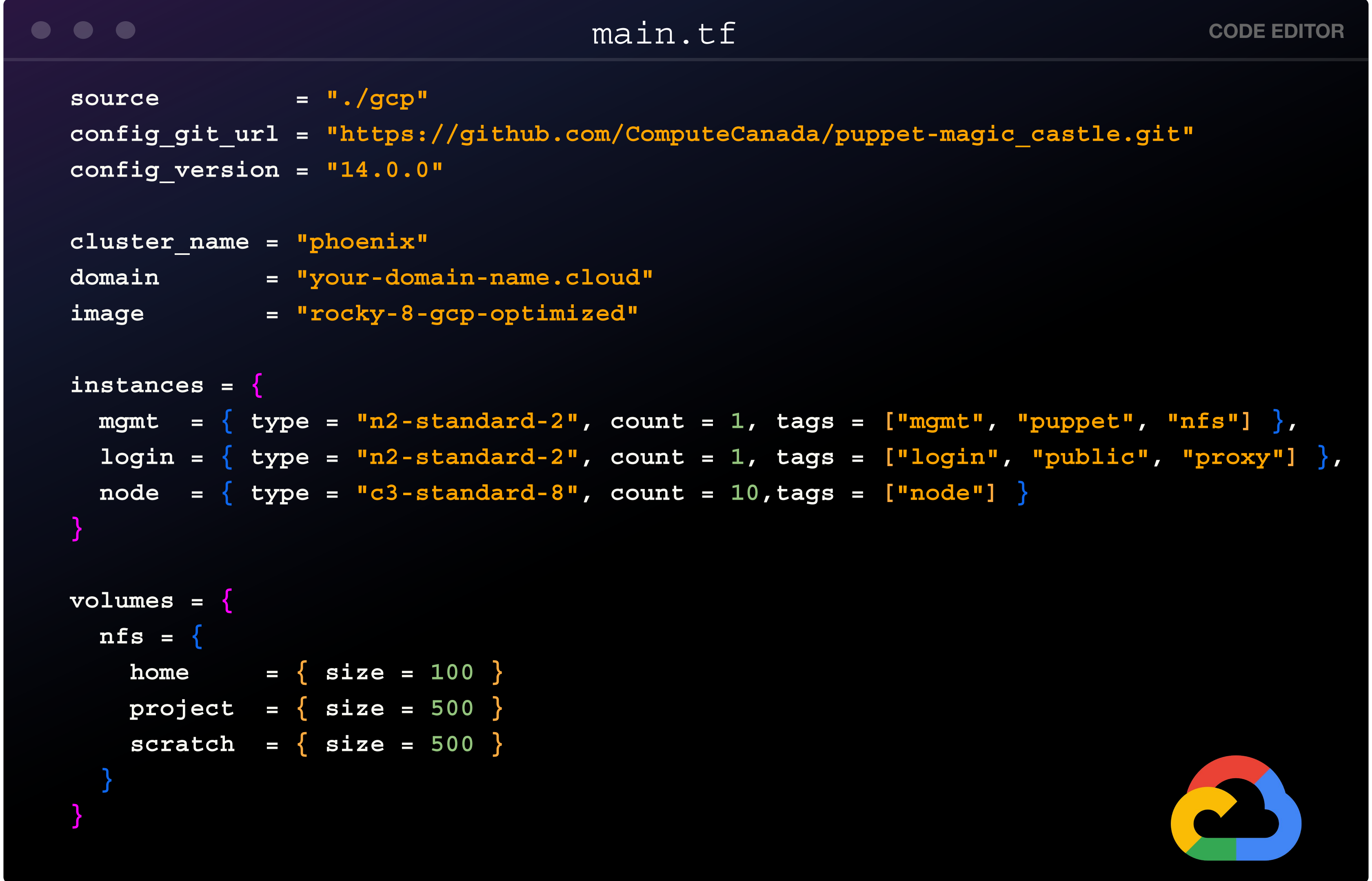
common/variables.tf

common/outputs.tf

These common inputs create an easy to use interface without vendor lock-in.



```
source          = "./aws"
config_git_url = "https://github.com/ComputeCanada/puppet-magic_castle.git"
config_version = "14.0.0"

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 = 10,tags = ["node"] }
}

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

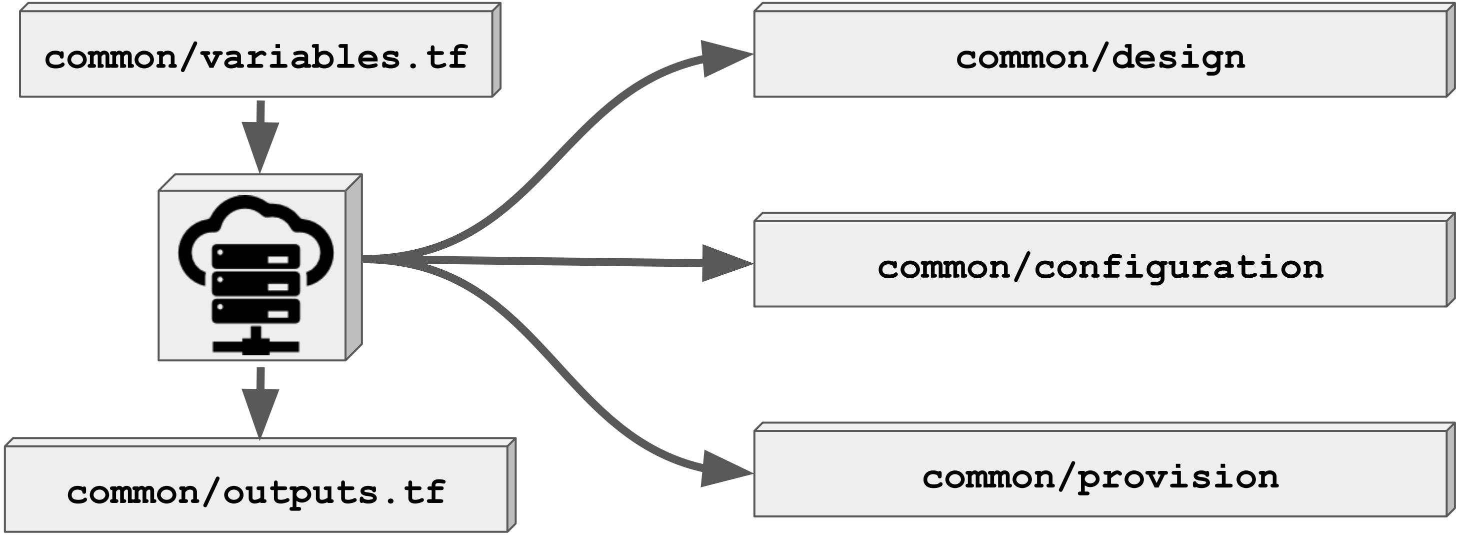


main.tf

```
source         = "./gcp"
config_git_url = "https://github.com/ComputeCanada/puppet-magic_castle.git"
config_version = "14.0.0"

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 = "c3-standard-8", count = 10,tags = ["node"] }
}

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

To facilitate the support of multiple providers, the inputs are transformed by common submodules.

Each main module uses 3 common sub-modules:



common/variables.tf

common/design

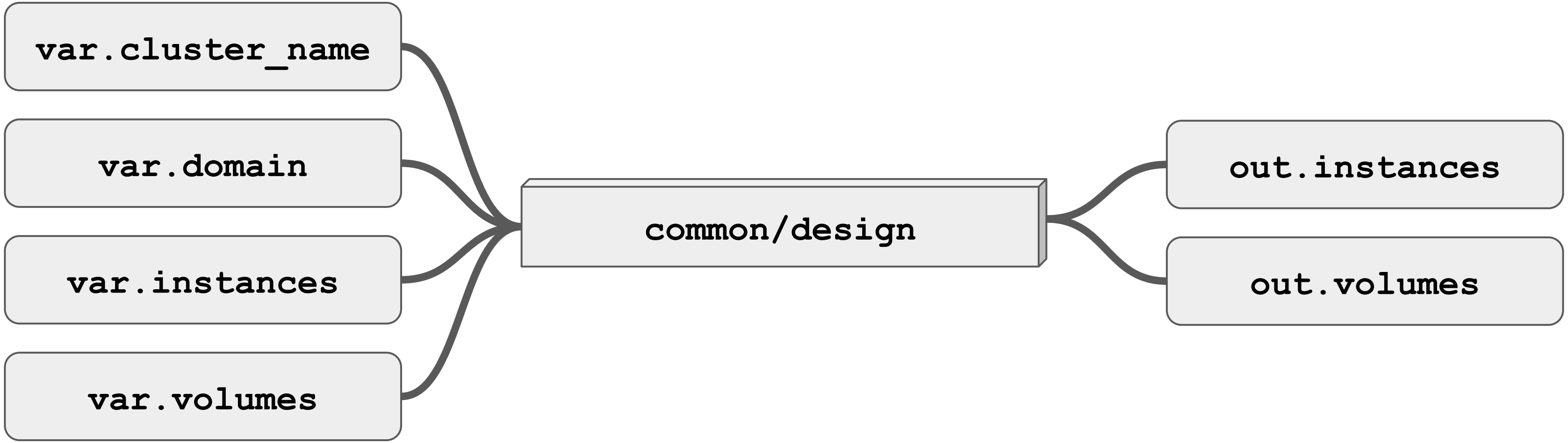
common/configuration

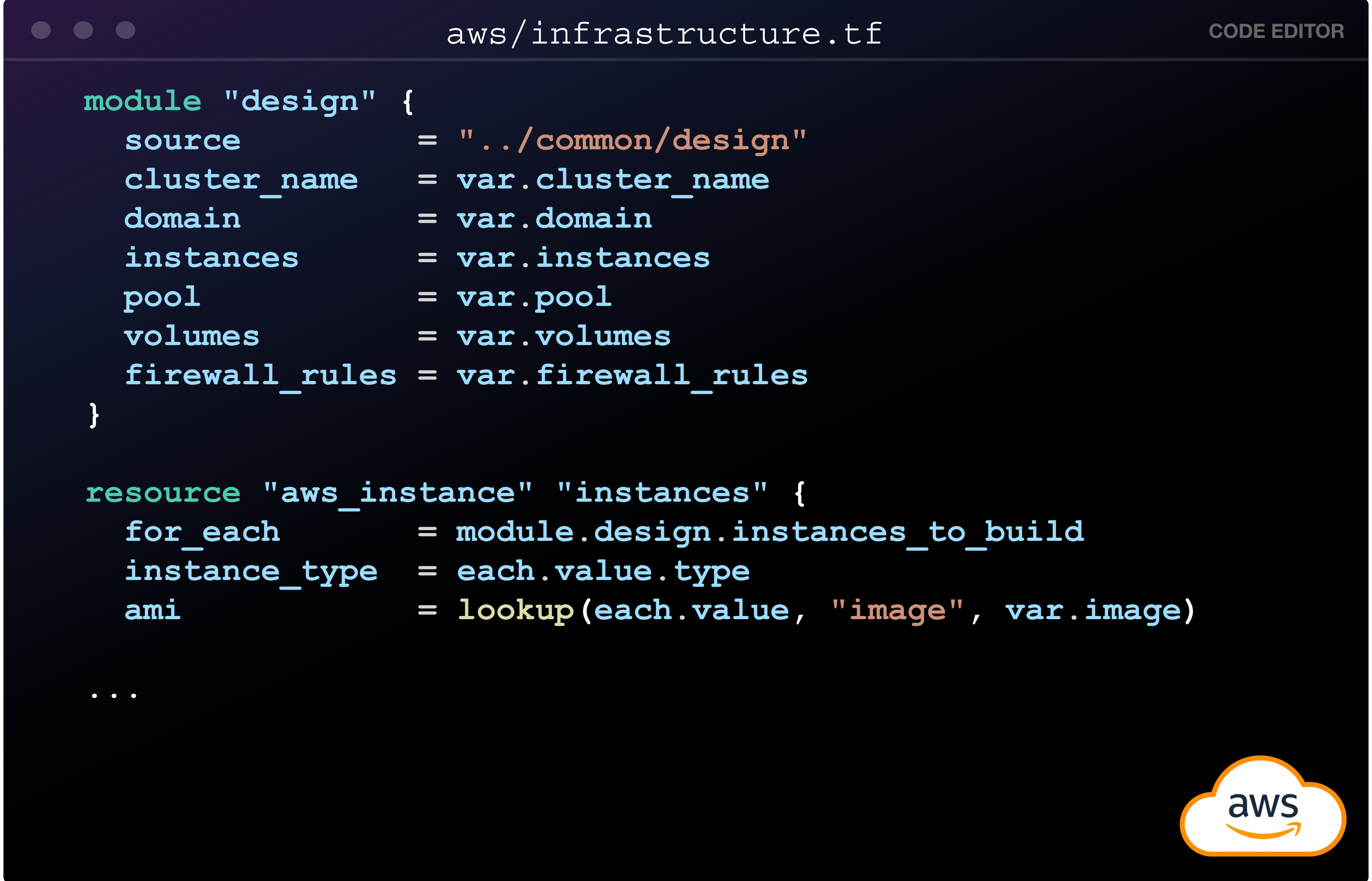
common/provision

common/outputs.tf

MAGIC CASTLE

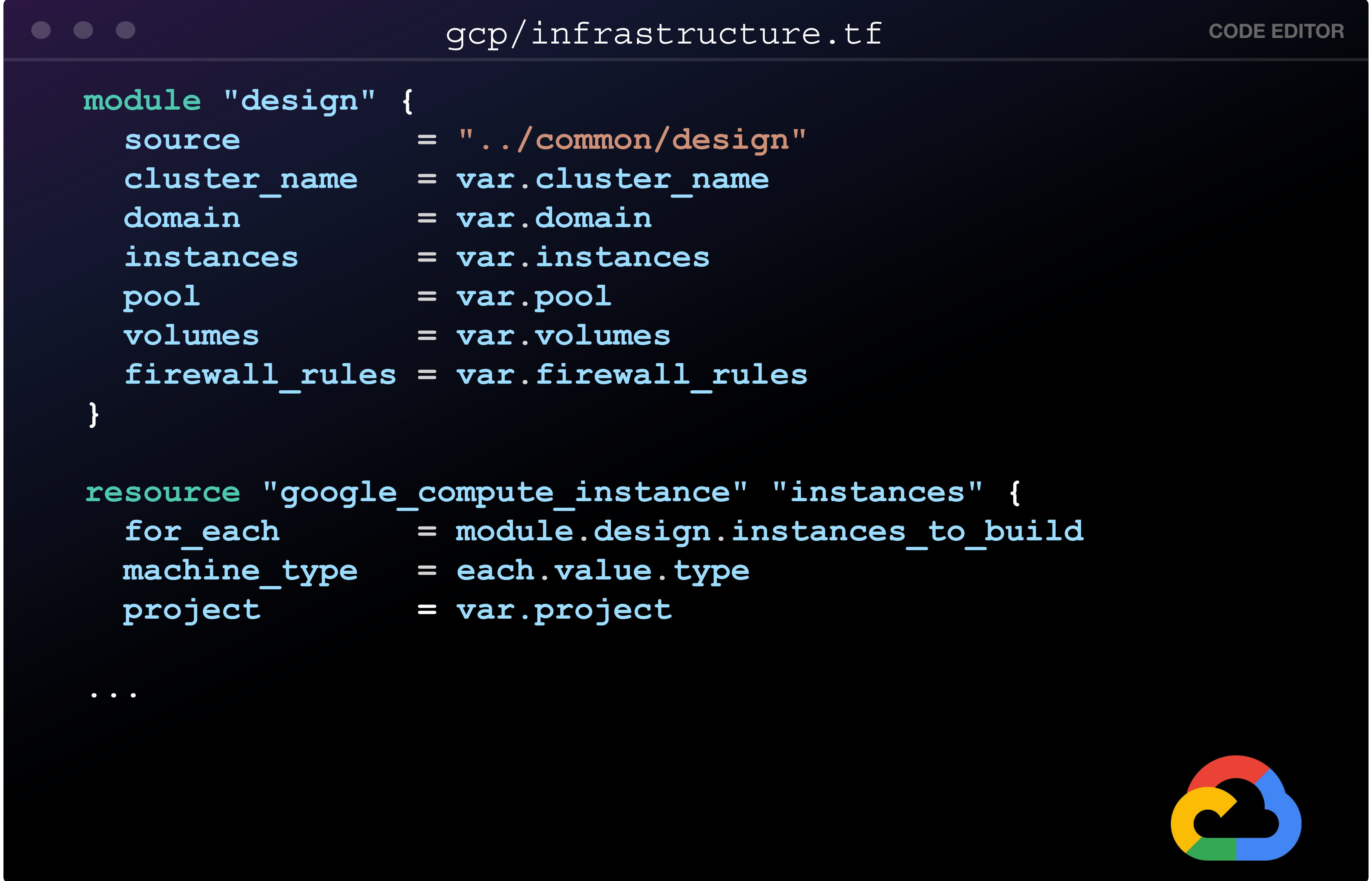
`design` sub-module transforms the inputs into **`maps`** used to generate the resources specific to each provider:





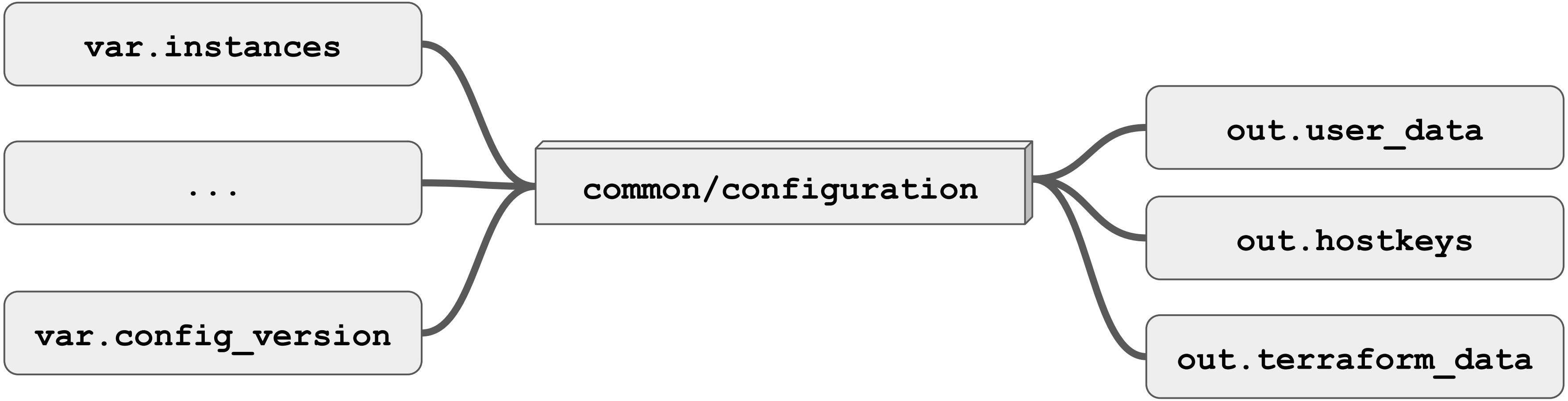
aws/infrastructure.tf

```
module "design" {
  source         = "../common/design"
  cluster_name   = var.cluster_name
  domain         = var.domain
  instances      = var.instances
  pool           = var.pool
  volumes        = var.volumes
  firewall_rules = var.firewall_rules
}

resource "aws_instance" "instances" {
  for_each       = module.design.instances_to_build
  instance_type  = each.value.type
  ami            = lookup(each.value, "image", var.image)

...
```



```
module "design" {
  source         = "../common/design"
  cluster_name   = var.cluster_name
  domain         = var.domain
  instances      = var.instances
  pool           = var.pool
  volumes        = var.volumes
  firewall_rules = var.firewall_rules
}

resource "google_compute_instance" "instances" {
  for_each       = module.design.instances_to_build
  machine_type   = each.value.type
  project        = var.project

...
```

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



var.instances

...

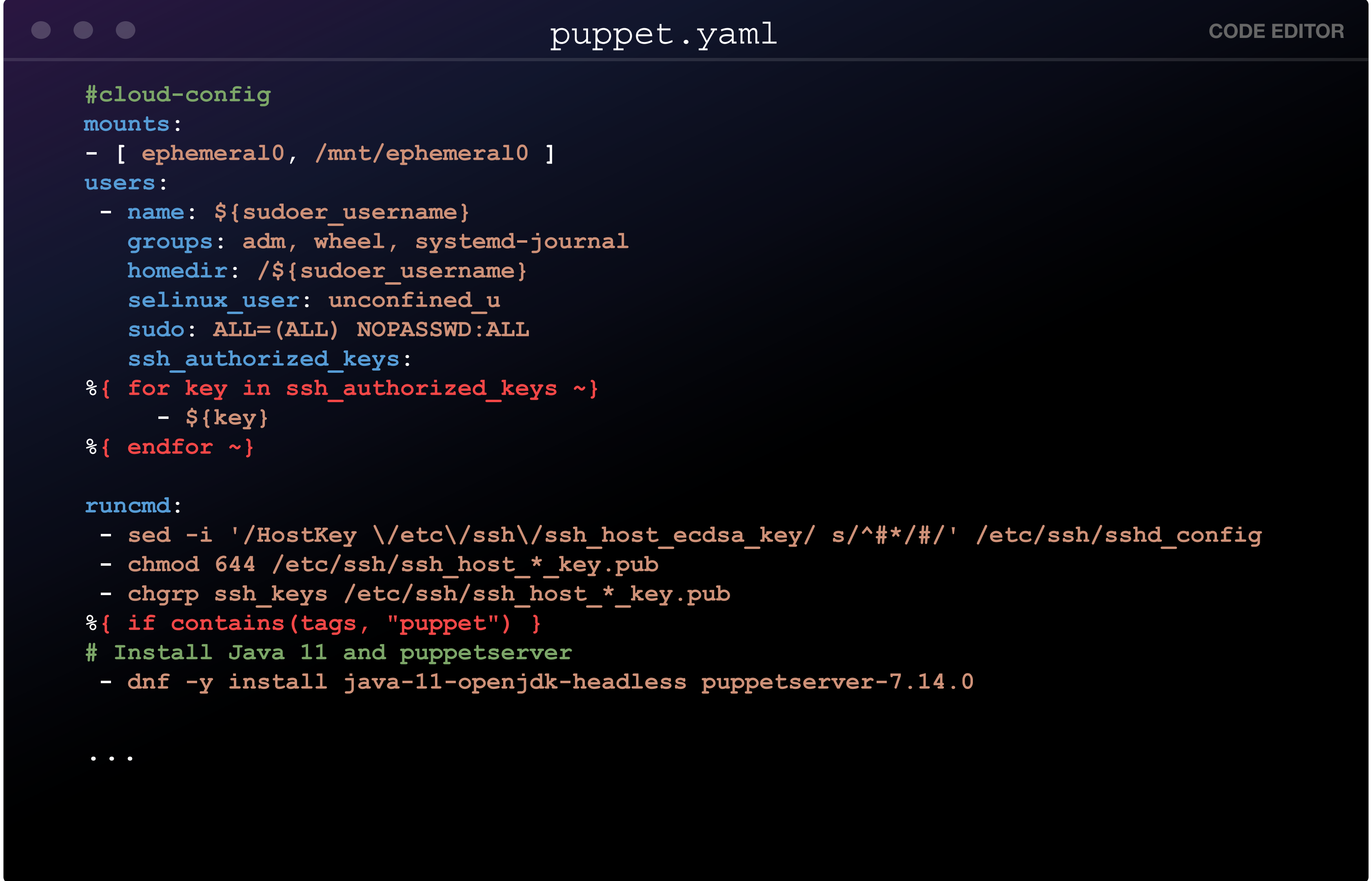
var.config_version

common/configuration

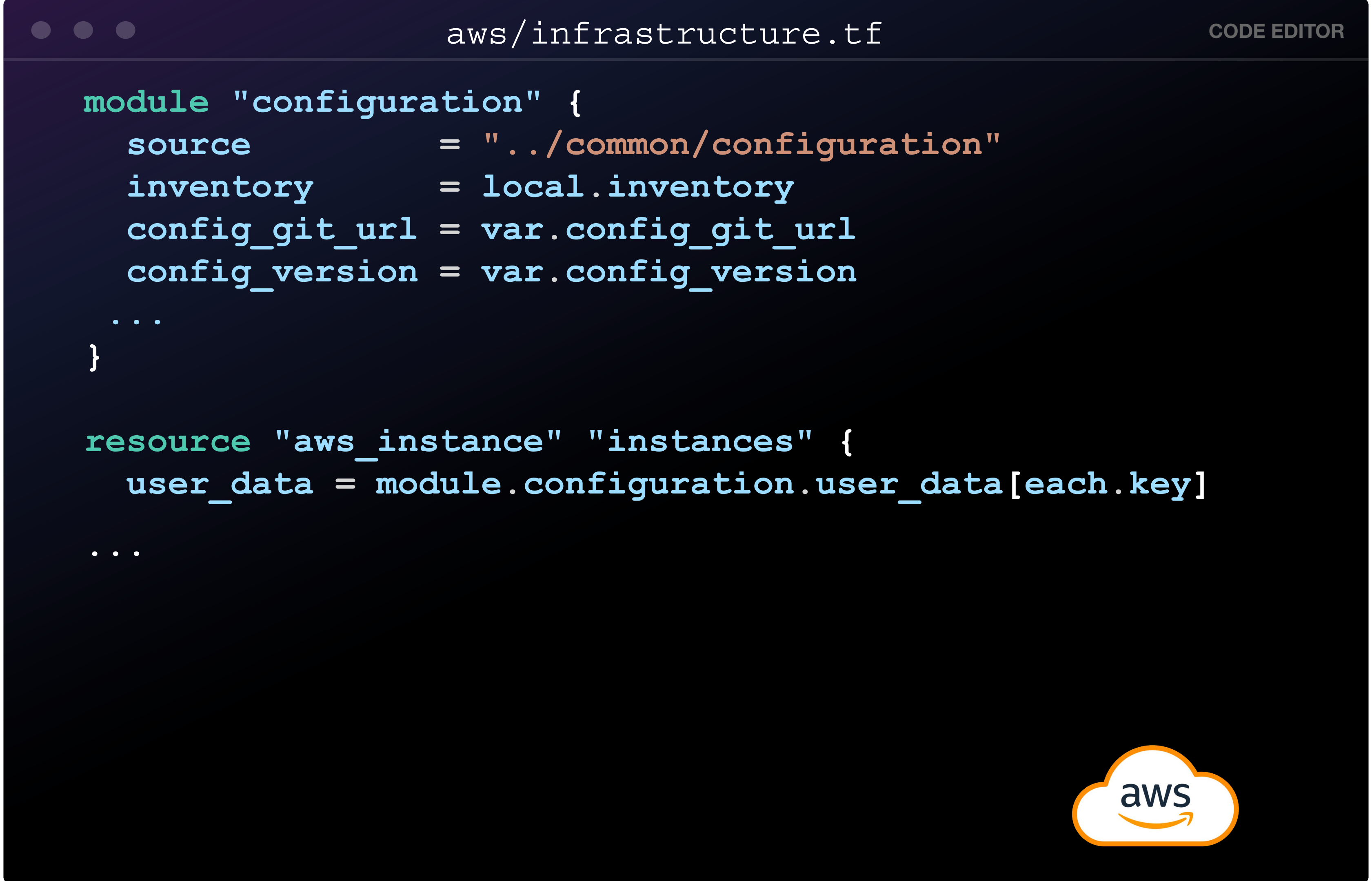
out.user_data

out.hostkeys

out.terraform_data

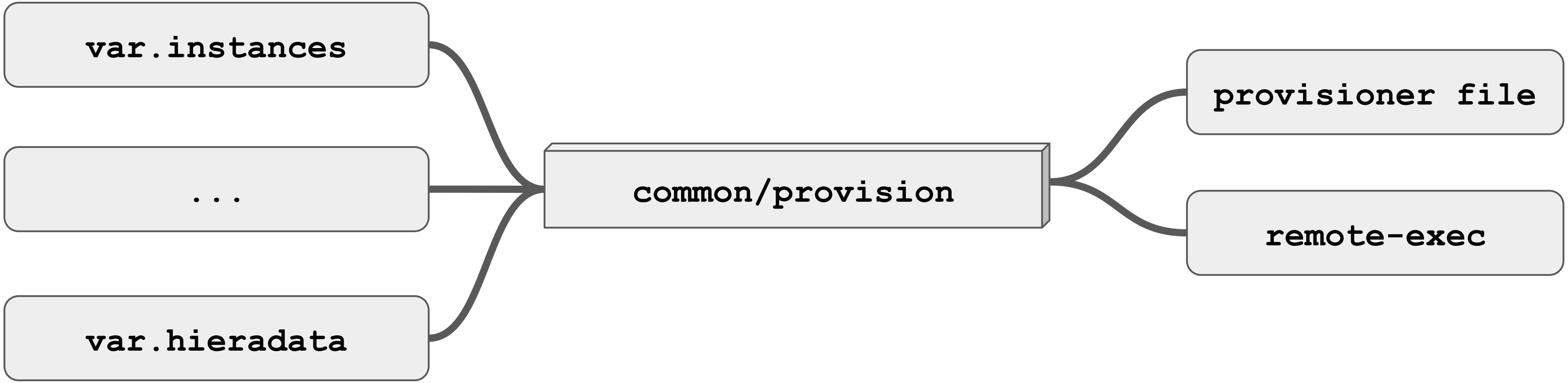


```
#cloud-config
mounts:
- [ ephemeral0, /mnt/ephemeral0 ]
users:
 - name: ${sudoer_username}
   groups: adm, wheel, systemd-journal
   homedir: /${sudoer_username}
   selinux_user: unconfined_u
   sudo: ALL=(ALL) NOPASSWD:ALL
   ssh_authorized_keys:
%{ for key in ssh_authorized_keys ~}
     - ${key}
%{ endfor ~}

runcmd:
 - sed -i '/HostKey \/etc\/ssh\/ssh_host_ecdsa_key/ s/^#*/#/' /etc/ssh/sshd_config
 - chmod 644 /etc/ssh/ssh_host_*_key.pub
 - chgrp ssh_keys /etc/ssh/ssh_host_*_key.pub
%{ if contains(tags, "puppet") }
# Install Java 11 and puppetserver
 - dnf -y install java-11-openjdk-headless puppetserver-7.14.0


...
```



```
module "configuration" {
  source         = "../common/configuration"
  inventory      = local.inventory
  config_git_url = var.config_git_url
  config_version = var.config_version
 ...
}

resource "aws_instance" "instances" {
  user_data = module.configuration.user_data[each.key]

...
```

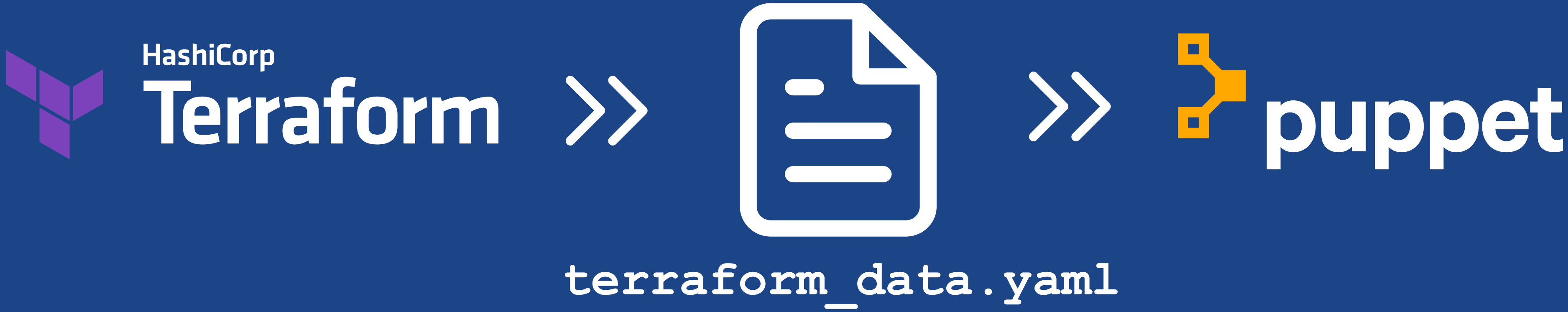
`provision` copies the state (instances, #cpus, #gpus, volumes, etc.) via SSH to the Puppet server as a YAML file (`terraform_data.yaml`).



MAGIC CASTLE

terraform_data.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"]
```



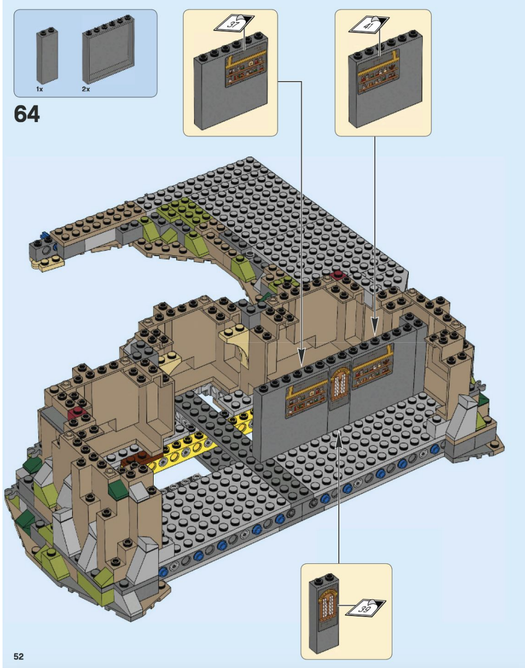
HashiCorp Terraform

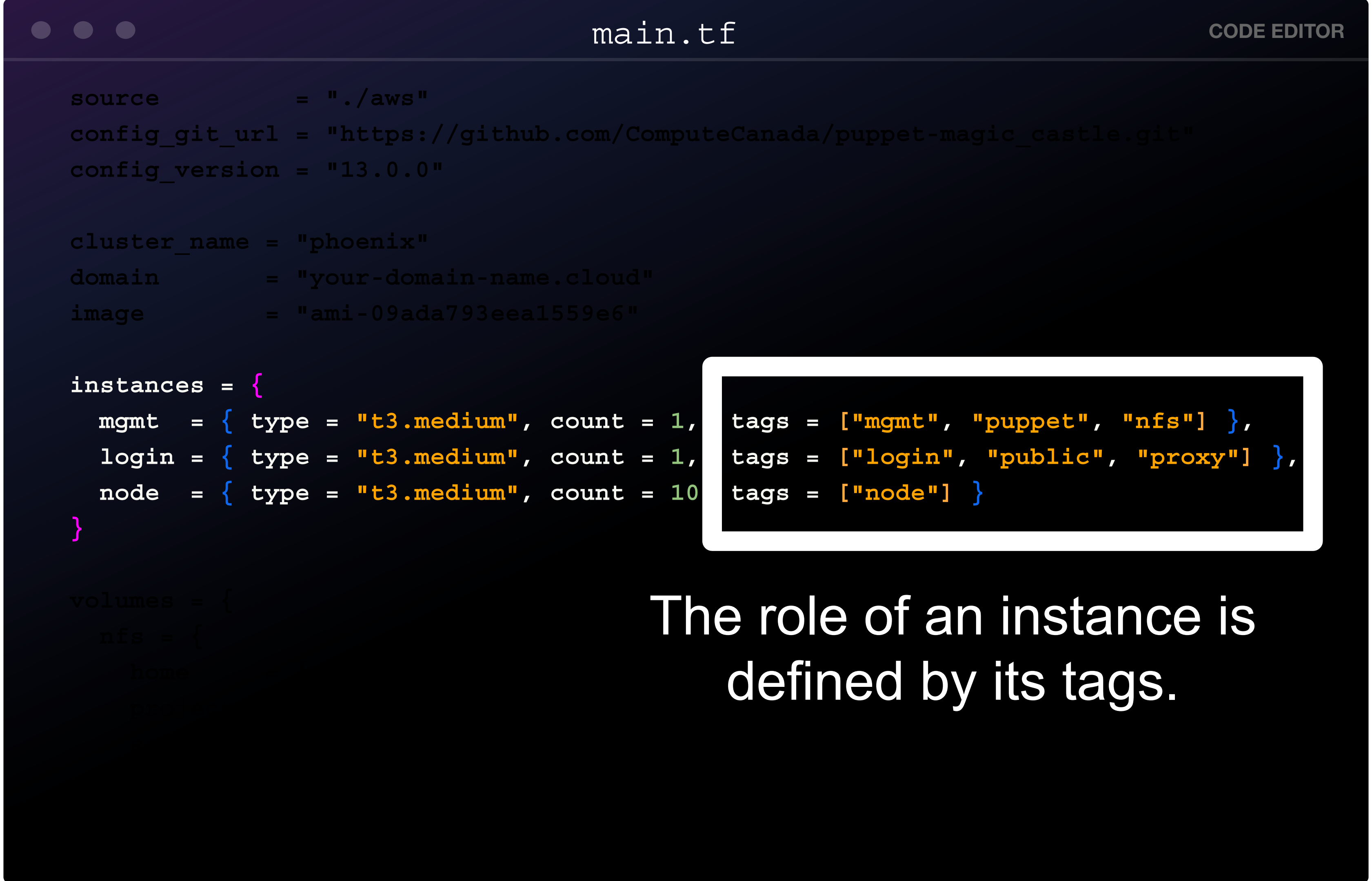
terraform_data.yaml

puppet

plan → apply → configure

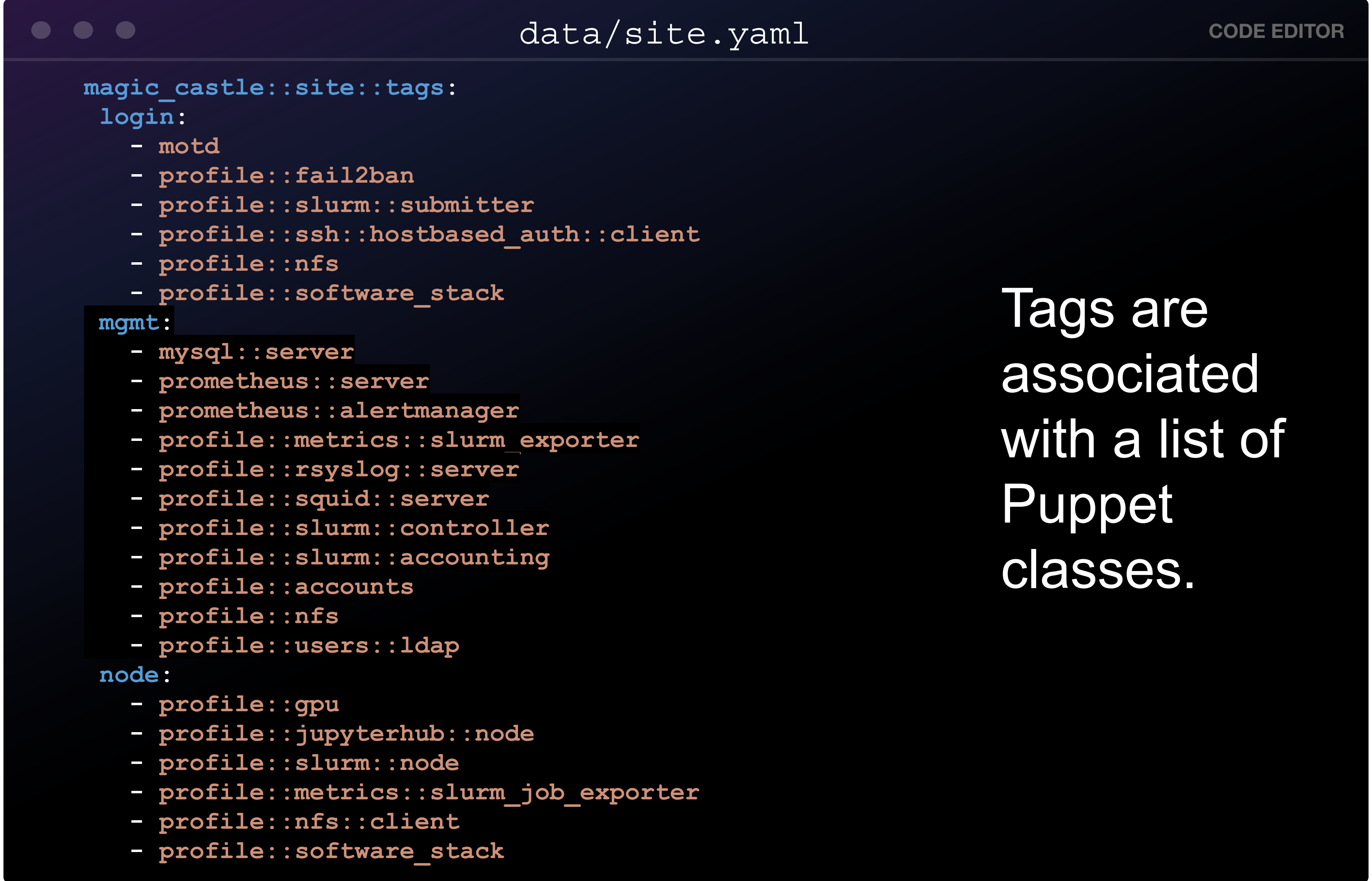
Puppet manages the configuration





```
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 = 10 tags = ["node"] }
}

volumes = {
  nfs = {
    home
    proj
```

The role of an instance is defined by its tags.



```yaml
magic_castle::site::tags:
 login:
   - motd
   - profile::fail2ban
   - profile::slurm::submitter
   - profile::ssh::hostbased_auth::client
   - profile::nfs
   - profile::software_stack
 mgmt:
   - mysql::server
   - prometheus::server
   - prometheus::alertmanager
   - profile::metrics::slurm_exporter
   - profile::rsyslog::server
   - profile::squid::server
   - profile::slurm::controller
   - profile::slurm::accounting
   - profile::accounts
   - profile::nfs
   - profile::users::ldap
 node:
   - profile::gpu
   - profile::jupyterhub::node
   - profile::slurm::node
   - profile::metrics::slurm_job_exporter
   - profile::nfs::client
   - profile::software_stack
```

Tags are associated with a list of Puppet classes.

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.
- New tags can also be added or old tags can be redefined.

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

Autoscaling

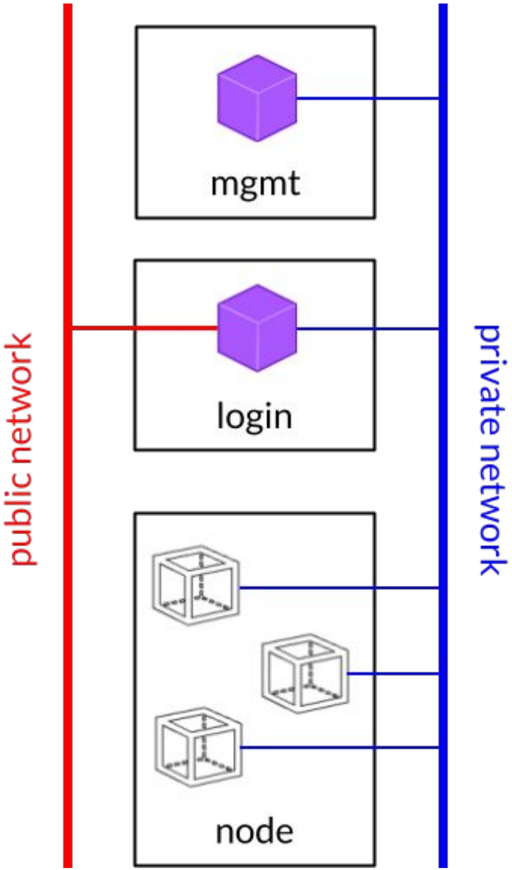
HashiCorp Terraform

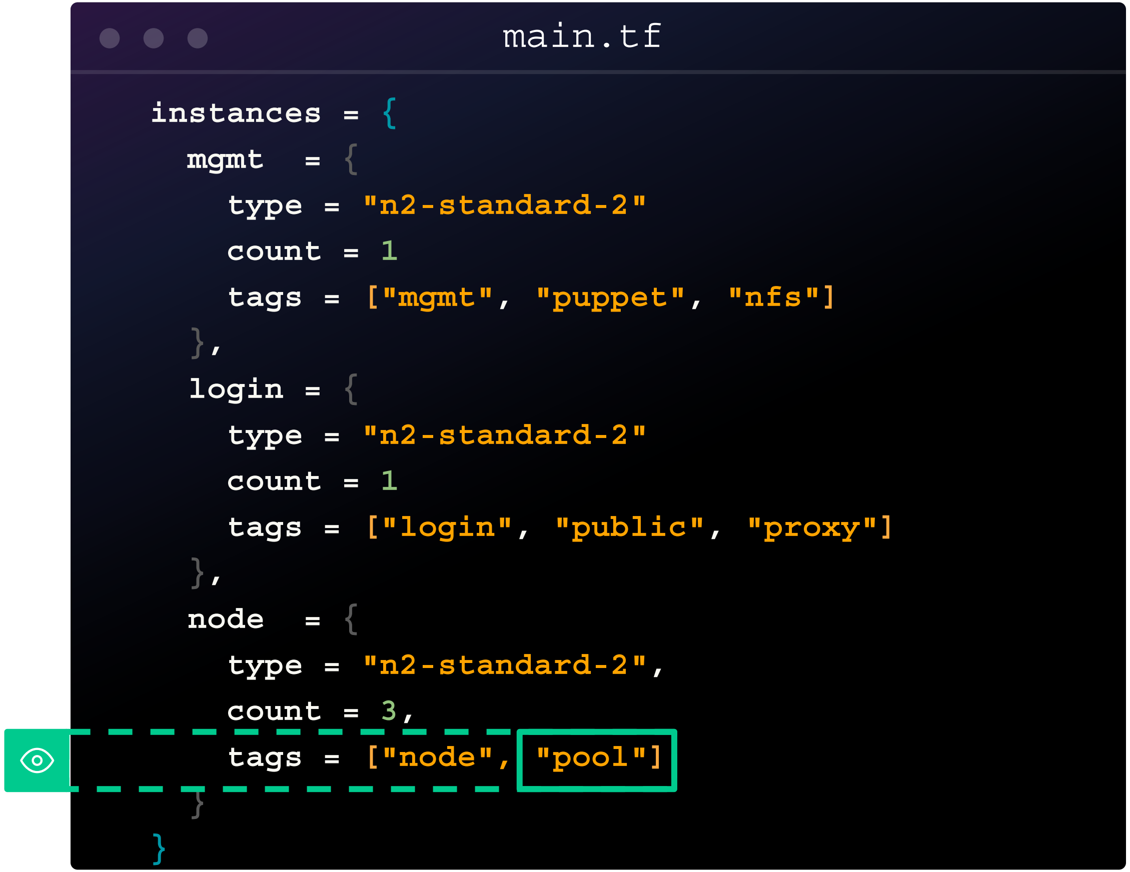
Autoscaling with Terraform Cloud

- Terraform CLI runs in a cloud
- A single API for Slurm to interact with

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

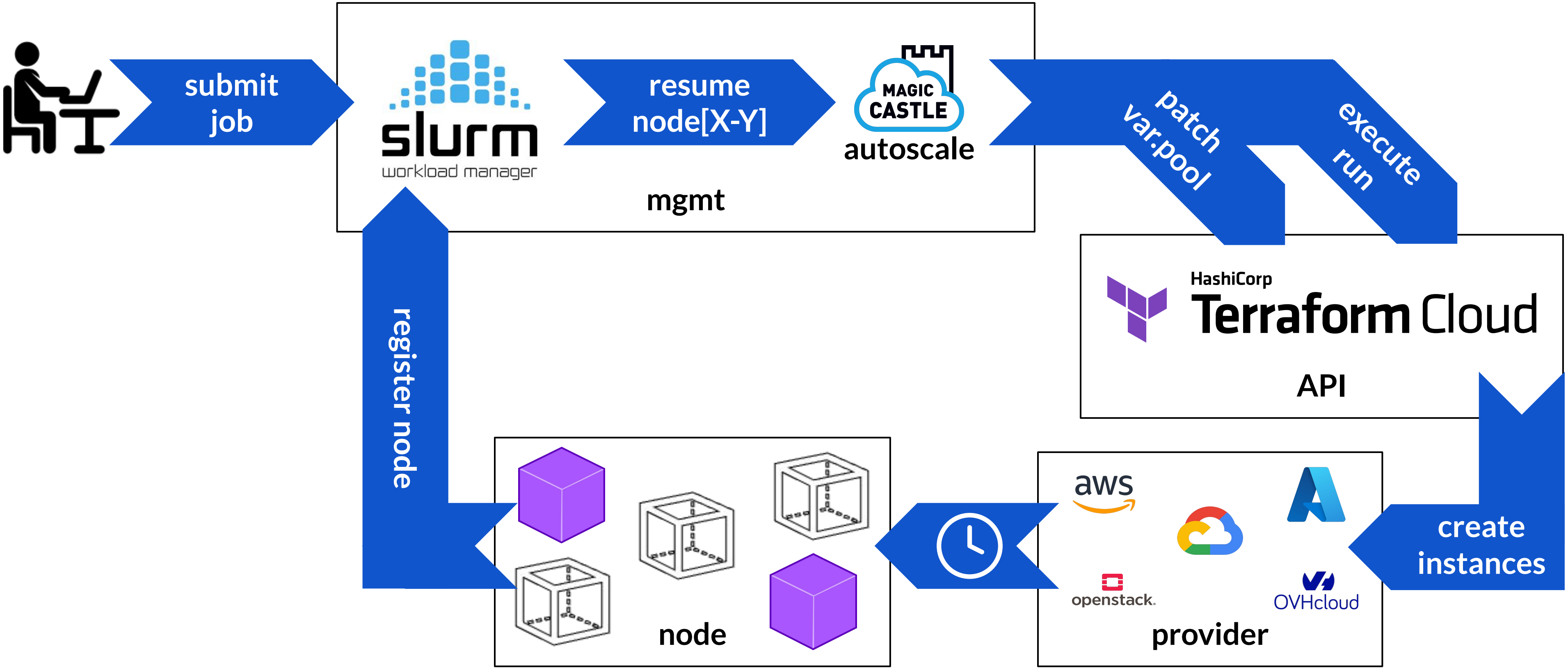
Autoscaling



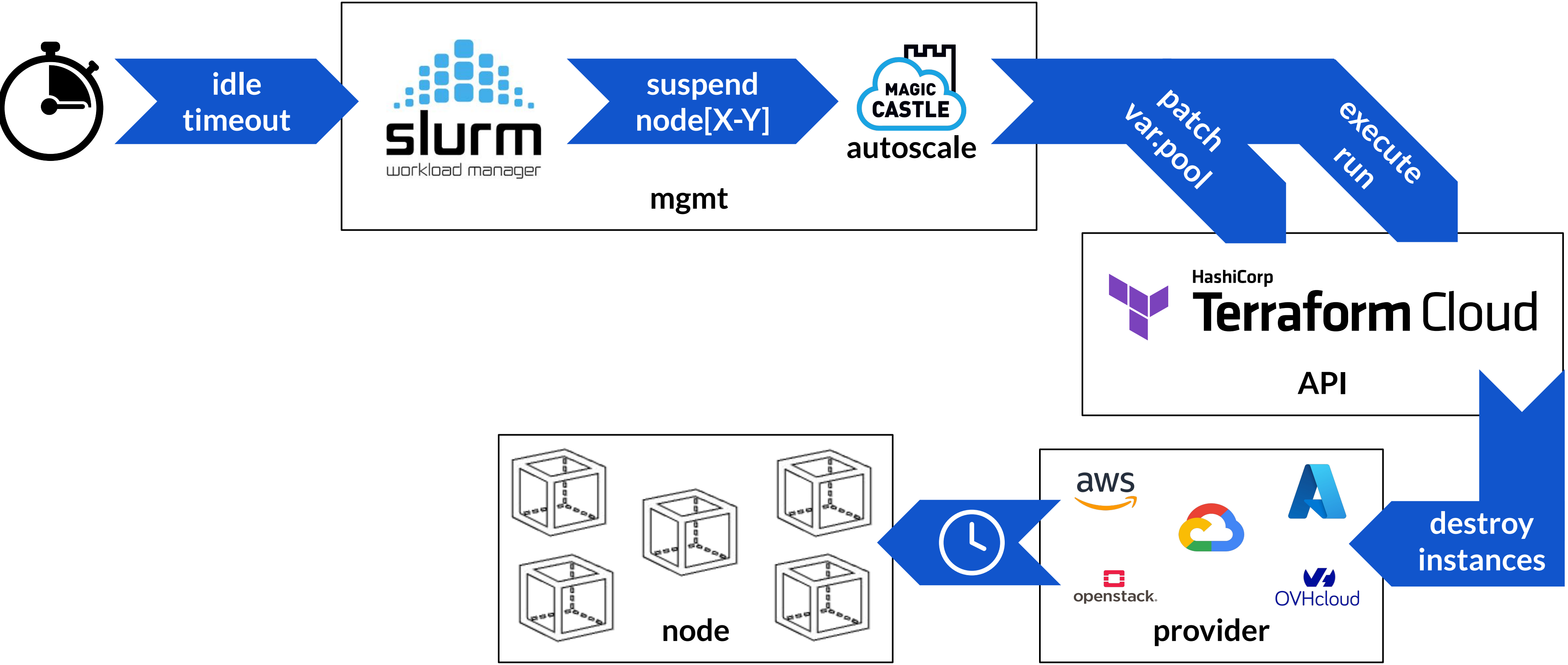


```
instances = {
  mgmt   = {
    type = "n2-standard-2"
    count = 1
    tags = ["mgmt", "puppet", "nfs"]
  },
  login = {
    type = "n2-standard-2"
    count = 1
    tags = ["login", "public", "proxy"]
  },
  node   = {
    type = "n2-standard-2",
    count = 3,
    tags = ["node", "pool"]
  }
}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

Autoscaling: resume



Autoscaling: suspend



- The autoscaling logic is *cloud-agnostic* and is expressed in 200 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.

MIG Configuration with Cloud Nodes

MIG Configuration with cloud nodes

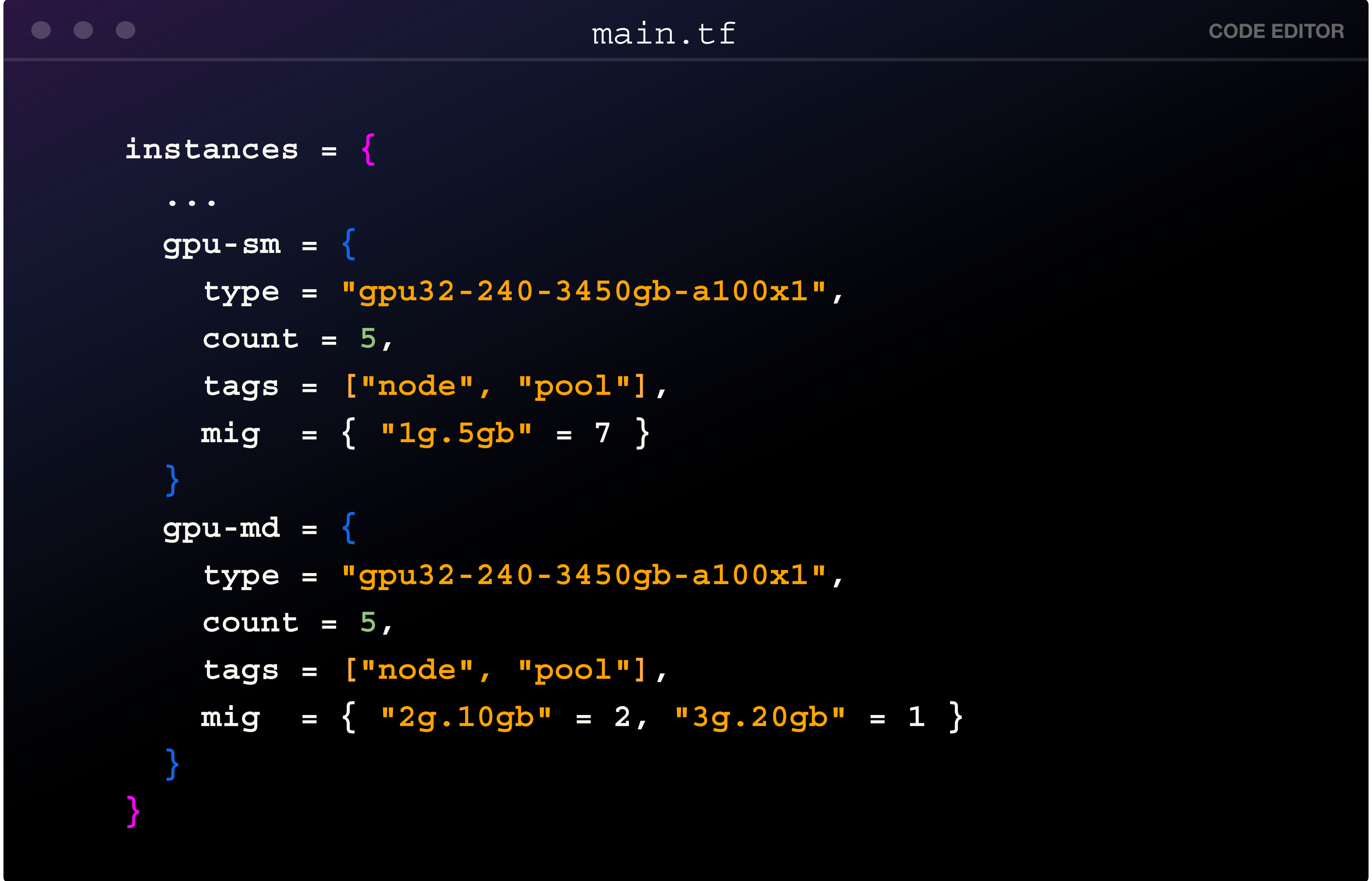
Problem:

- To configure MIGs in Slurm, specify `AutoDetect=nvml` in gres.conf
- **But** AutoDetect cannot be used with cloud nodes.

Solution:

1. Define MIG Profiles in Terraform (main.tf)
2. [compute] Puppet installs NVIDIA drivers
3. [all] Puppet generates the slurm.conf from terraform_data.yaml
4. Puppet generates the gres.conf
   - [controller] using the information from terraform_data.yaml
   - [compute] using nvidia_gres.sh which is based on nvidia-smi
5. [compute] Puppet uses nvidia-mig-parted to apply config

Combined with autoscaling, a user can request a specific MIG profile



```
instances = {
  ...
  gpu-sm = {
    type = "gpu32-240-3450gb-a100x1",
    count = 5,
    tags = ["node", "pool"],
    mig  = { "1g.5gb" = 7 }
  }
  gpu-md = {
    type = "gpu32-240-3450gb-a100x1",
    count = 5,
    tags = ["node", "pool"],
    mig  = { "2g.10gb" = 2, "3g.20gb" = 1 }
  }
}
```

Use case 1:
Education

Since Magic Castle initial release in **2018**

1k+ workshops

and university courses have used Magic Castle to teach advanced research computing.

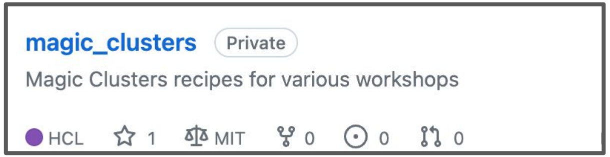
Calcul Québec

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]

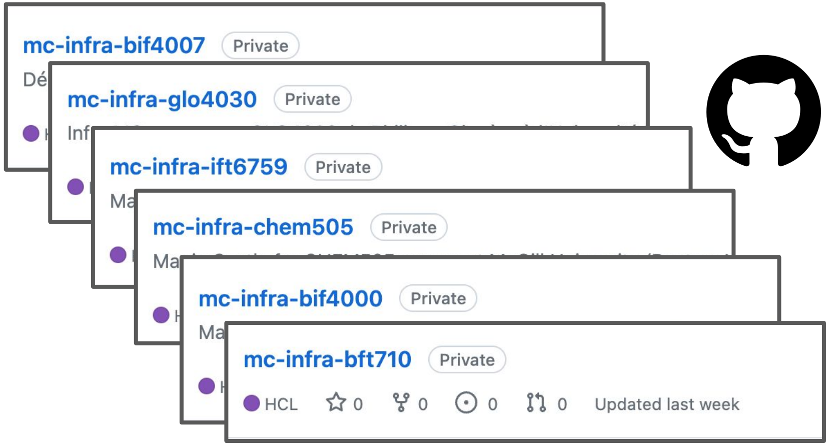
- Provides and administers Magic Castle clusters to graduate courses from various disciplines: AI, bioinformatics, neuroscience, chemistry



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

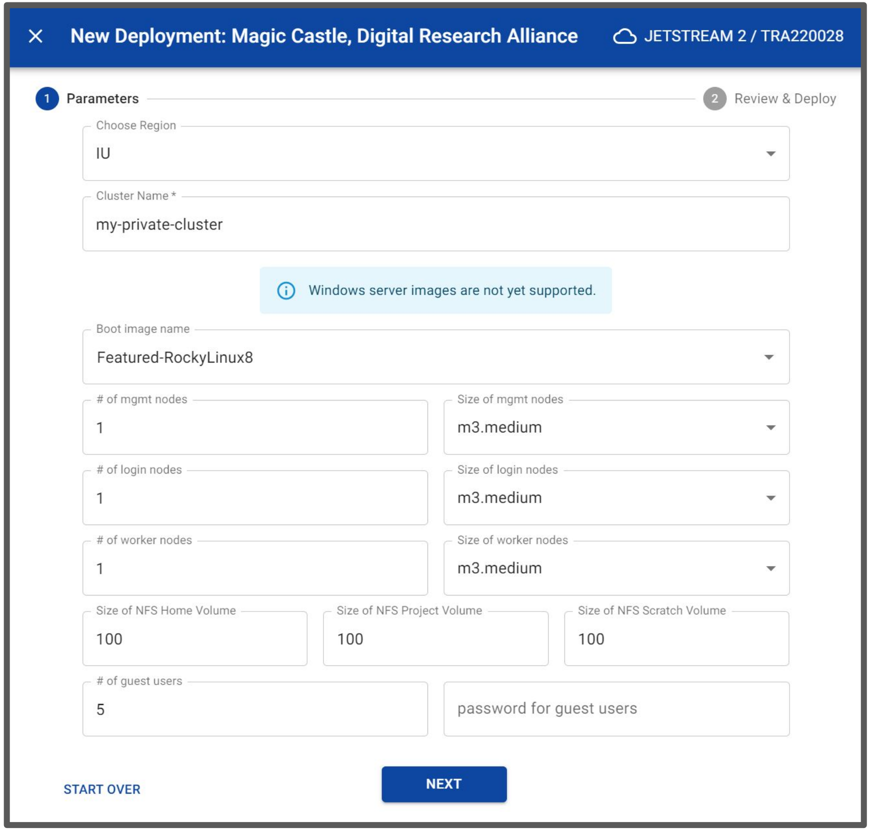
● HCL ☆ 0 0 0 0 Updated last week

Use case 2:

Self-service HPC cluster creation platforms

Jetstream2

Magic Castle is integrated in CACAO and can be launched easily in Jetstream2 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

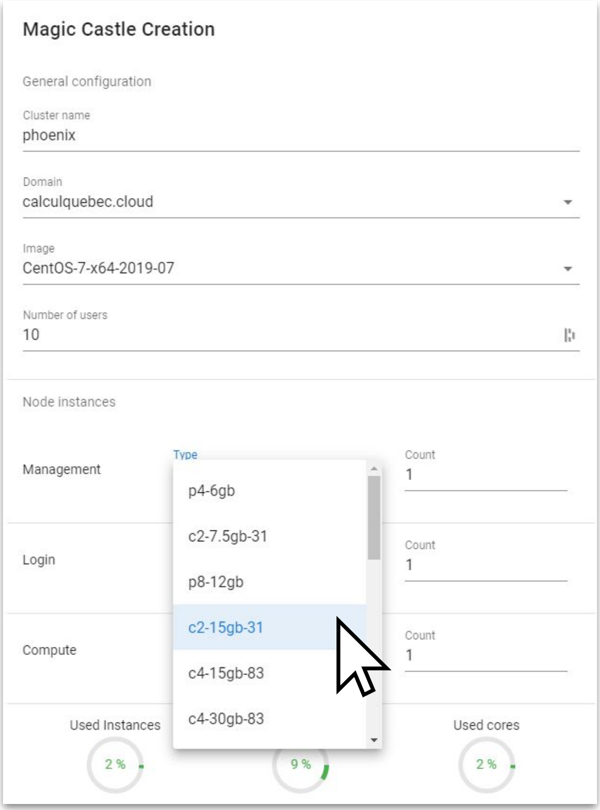
password for guest users

https://docs.jetstream-cloud.org/general/virtualclusters
https://github.com/edwins/magic_castle
https://docs.jetstream-cloud.org/ui/cacao/deployment_magic_castle/

Digital Research Alliance of Canada

Digital Research Alliance of Canada sponsors the development of Magic Castle own platform for spawning virtual HPC clusters: MC-Hub

https://github.com/computeCanada/mc-hub



Magic Castle Creation

General configuration

Cluster name: phoenix

Domain: calculquebec.cloud

Image: CentOS-7-x64-2019-07

Number of users: 10

Node instances

Management — Type — Count: 1

Login — Count: 1

Compute — Count: 1

Used Instances: 2 %

9 %

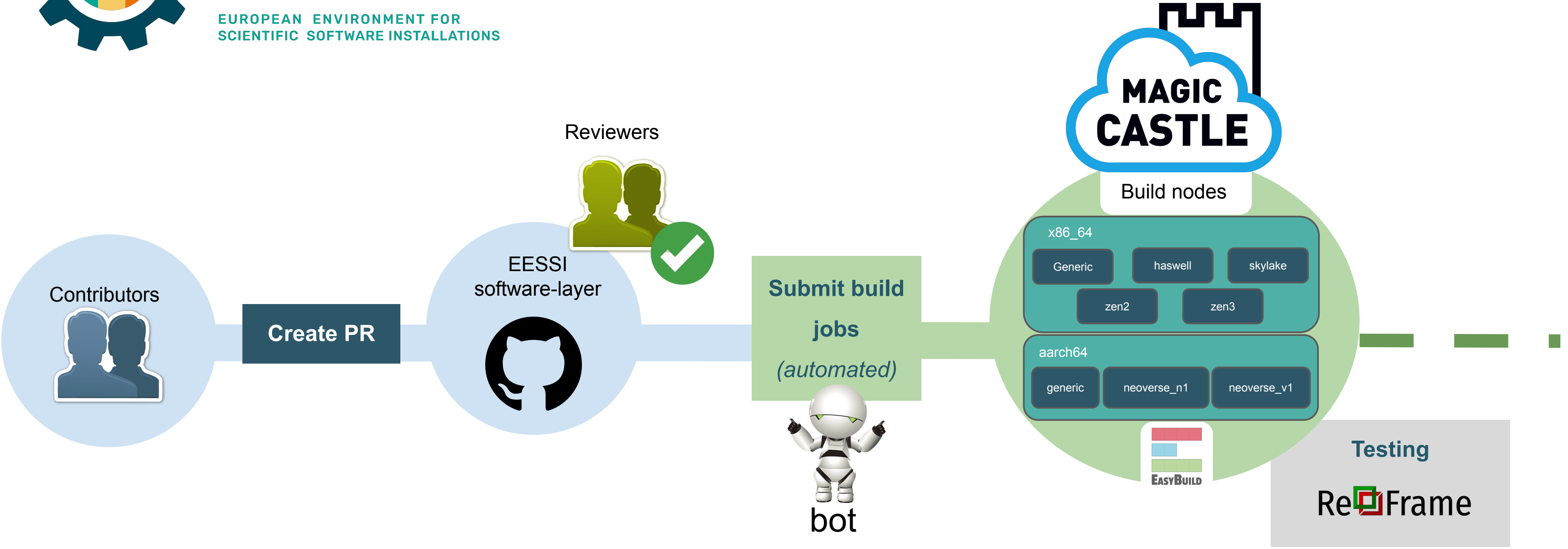
Used cores: 2 %

Use case 3:
Scientific platforms

EESSI

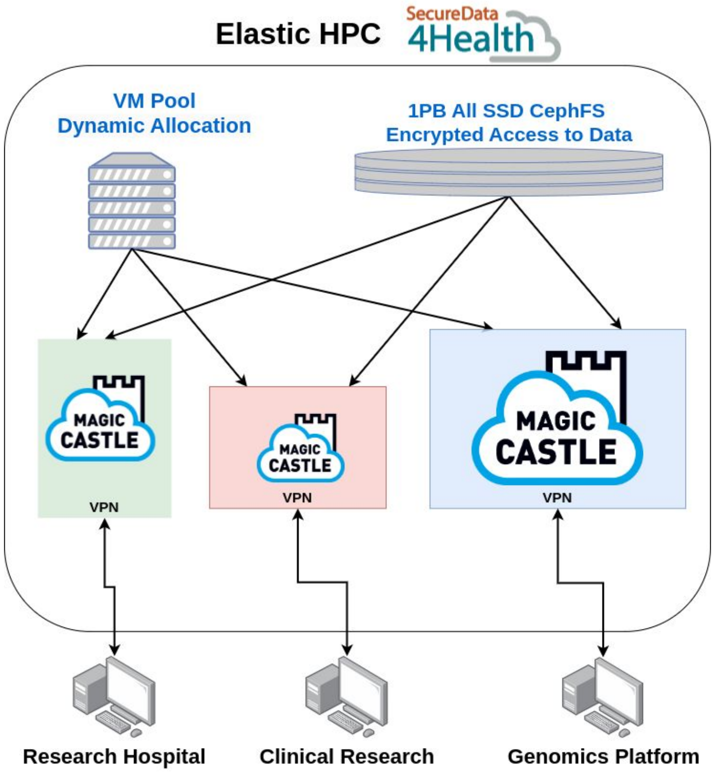
EUROPEAN ENVIRONMENT FOR SCIENTIFIC SOFTWARE INSTALLATIONS

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



https://www.eessi.io/

SecureData 4 Health: cancer patient genome sequencing



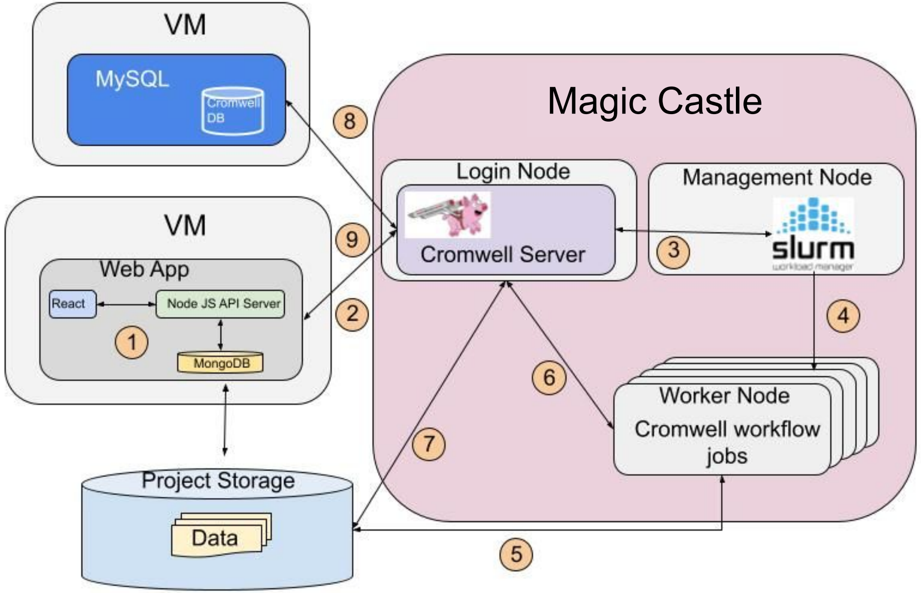
https://www.sd4health.ca/

- Single infrastructure - OpenStack
- Fully isolated project per research client
- Fulfilled hospitals cybersecurity requirements
- One Magic Castle cluster per client
- Client example: Marathon of hope Cancer Network
  - Comparison of healthy vs cancerous cells
  - 2000 cores
  - 120k jobs so far in 2024

Canadian Centre for Computational Genomics

McGill UNIVERSITY

National Microbiome Data Collaborative EDGE platform



- Allows researchers to process data with standard NMDC bioinformatics workflows
- Workflows are configured through the platform
- The jobs are scheduled in a Magic Castle cluster via Cromwell Server
- Magic Castle cluster is spawned via CACAO in Jetstream2

https://nmdc-edge.org/home

Los Alamos NATIONAL LABORATORY

MAGIC CASTLE

★ Simple to use

★ Batteries included: software, scaling, MIG, etc.

★ Ideal software environment to integrate HPC into platforms and for teaching

cloud-agnostic and **open source**

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