
# preview of MC 12

Hey!

# I am Félix-Antoine

Lead of research software dev. team
Université Laval

Principal developer of Magic Castle


Open-source software that reproduces the HPC user experience in the cloud

# Straight-forward process

## Edit

Edit a single text file describing the cluster to create as code.

## Apply

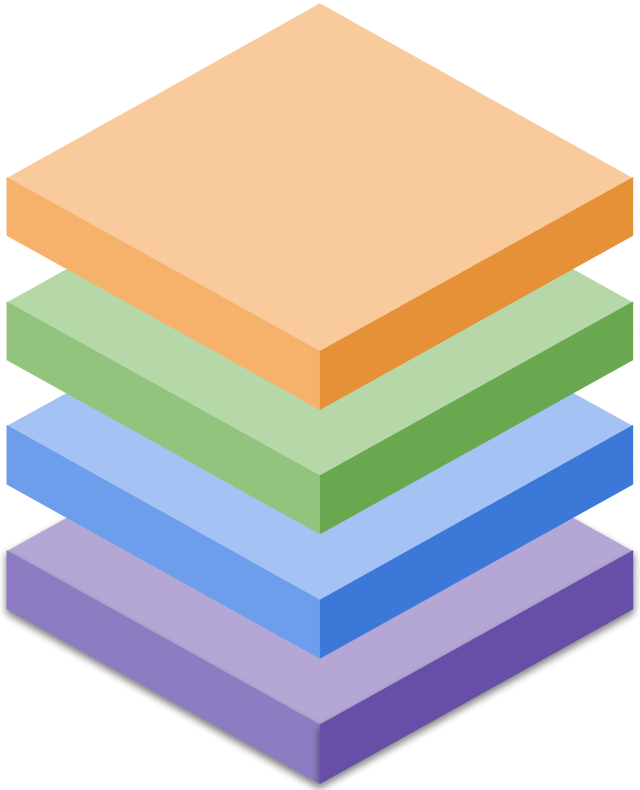
Execute the infrastructure code and create the virtual resources.

## Connect

Connect to the cluster using your protocol of choice:

- SSH
- HTTPS

# Magic Castle Components



Configuration

Virtuals machines

Volumes, storage, filesystems

Network, subnetwork, interfaces, DNS

# Batteries included














# Requirements

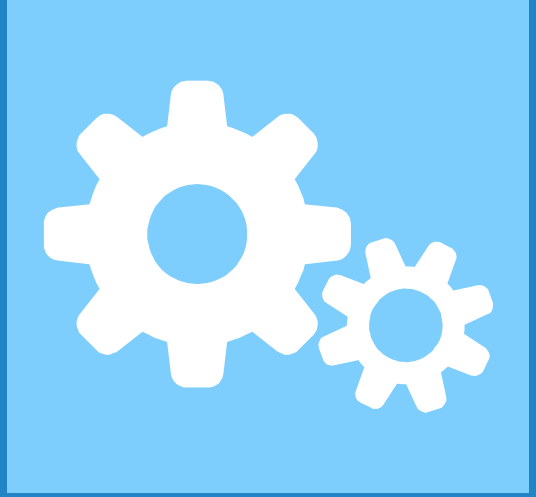
**A cloud account**

Magic Castle can be used with on-prem OpenStack and commercial clouds



**Terraform**

Read and execute Magic Castle infrastructure code



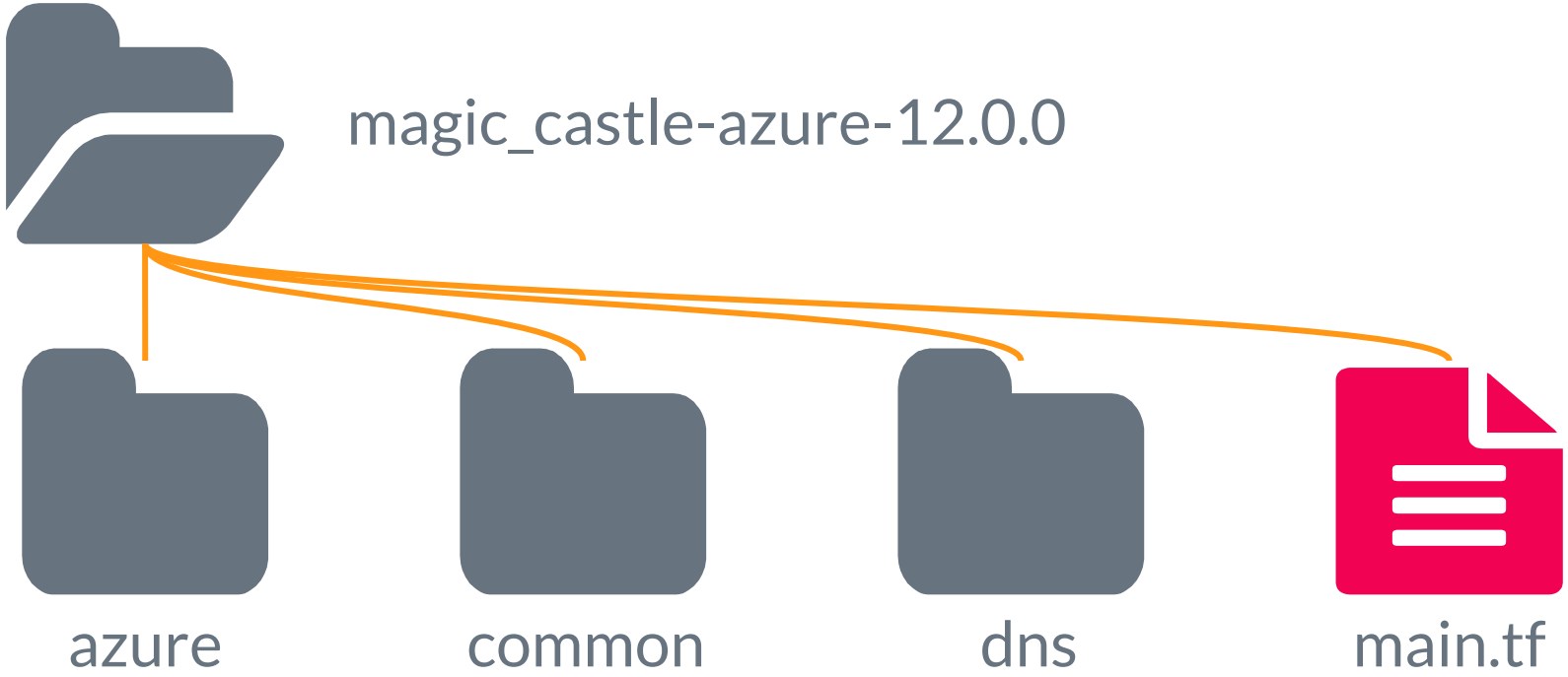
# How it works

# Magic Castle Steps

# Edit

1. Download a Magic Castle release for your cloud of choice
2. Uncompress the archive

# Edit



## Edit

main.tf

```
module "azure" {
  source          = "./azure"
  config_git_url = "..."
  config_version = "12.0.0"
```

## Edit

main.tf

```
cluster_name = "thor"
domain       = "magiccastle.dev"
image        = "CentOS-7-x64-2020-09"
```

{ AlmaLinux, Rocky, CentOS 8, RedHat }

## Edit

main.tf

```
instances = {
  mgmt  = {
    type  = "p4-7.5gb",
    tags  = ["puppet", "mgmt", "nfs"],
    count = 1
  }
```

## Edit

main.tf

```
login  = {
  type  = "p2-3.75gb",
  tags  = ["login", "public", "proxy"],
  count = 1
}
```

## Edit

main.tf

```
  node  = {
    type  = "p2-3.75gb",
    tags  = ["node"],
    count = 1
  }
}
```

# About tags and instance names

**Tags:**

- **Infrastructure**: use to attach devices and volumes
- **Configuration**: use to assign roles and responsibilities to instances

**Instance names** are arbitrary and their significance depends on the configuration environment.

They are **not limited** to *mgmt*, *login* or *node*.

## Edit

main.tf

tag

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

## Edit

main.tf

```
 public_keys   = [file("~/.ssh/id_rsa.pub")]
 nb_users      = 10
 software_stack = "eessi"
}
```

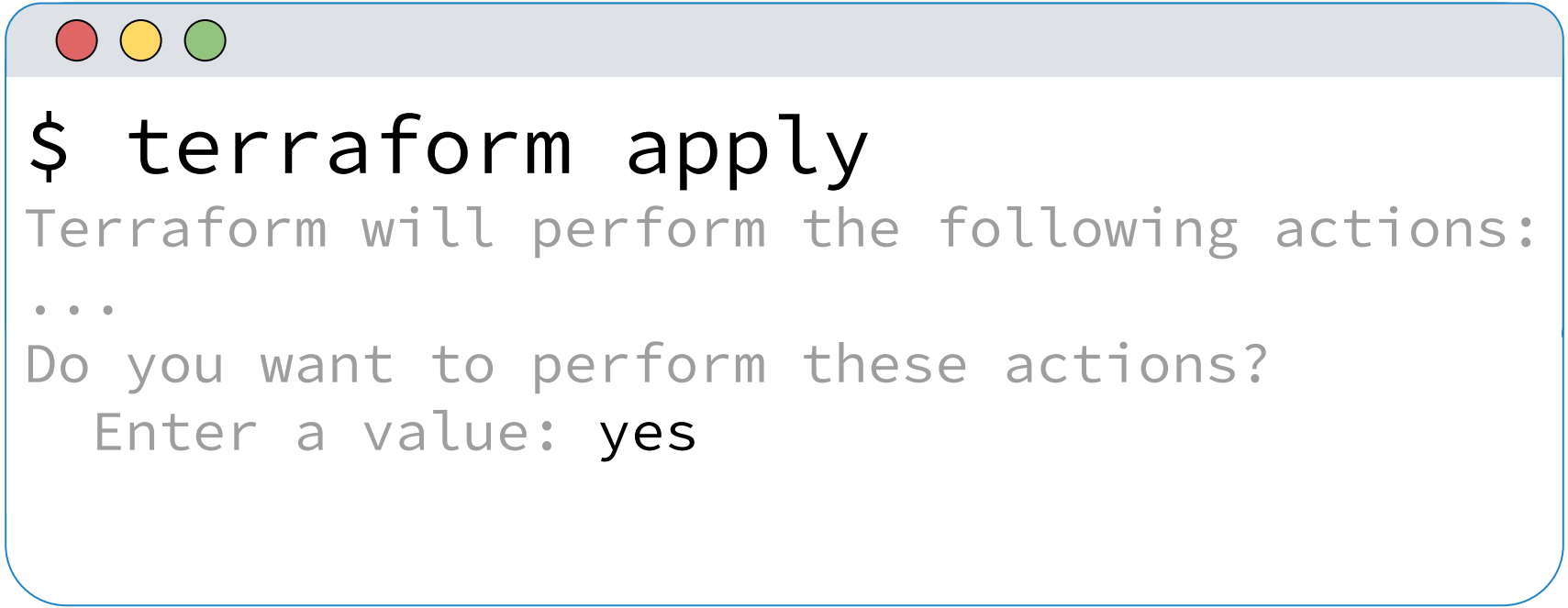
# About parameters

main.tf

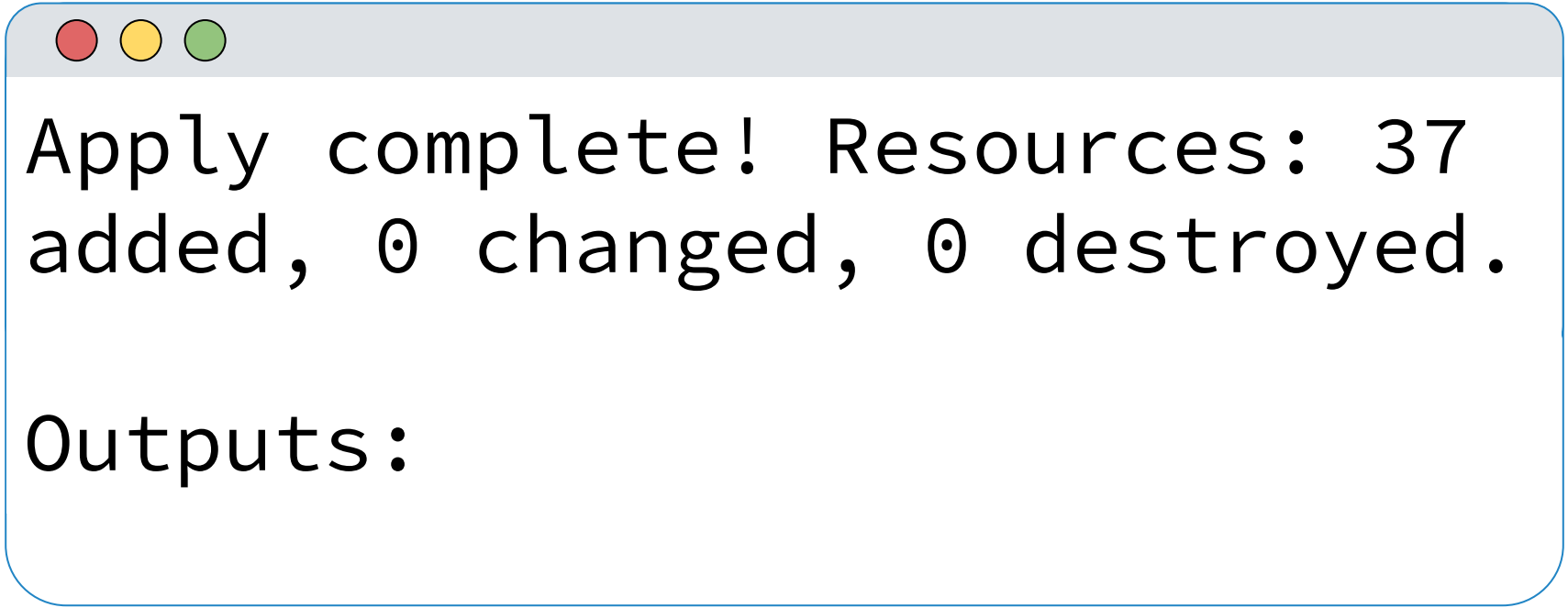
- Default values are a good starting point
- Optional parameters were not covered
- Documentation details every parameter
- Each parameter can be modified at any point in the cluster life

Next step is to get Terraform to read the infrastructure code and spawn the cluster.

# Apply


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

# Apply



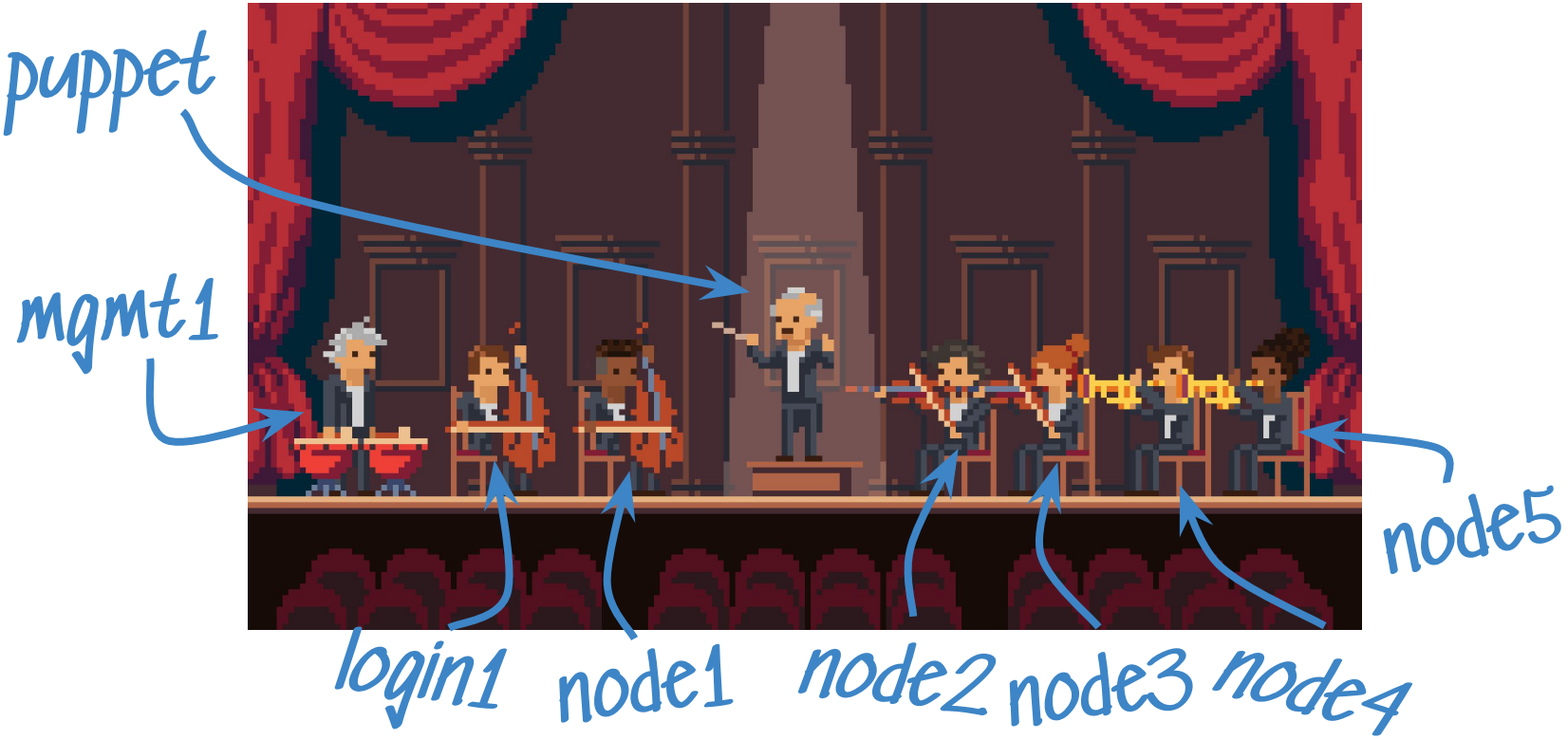
```
Apply complete! Resources: 37
added, 0 changed, 0 destroyed.

Outputs:
```

# Once the cluster is built...

Its automatic configuration begins.

# Configuration with Puppet



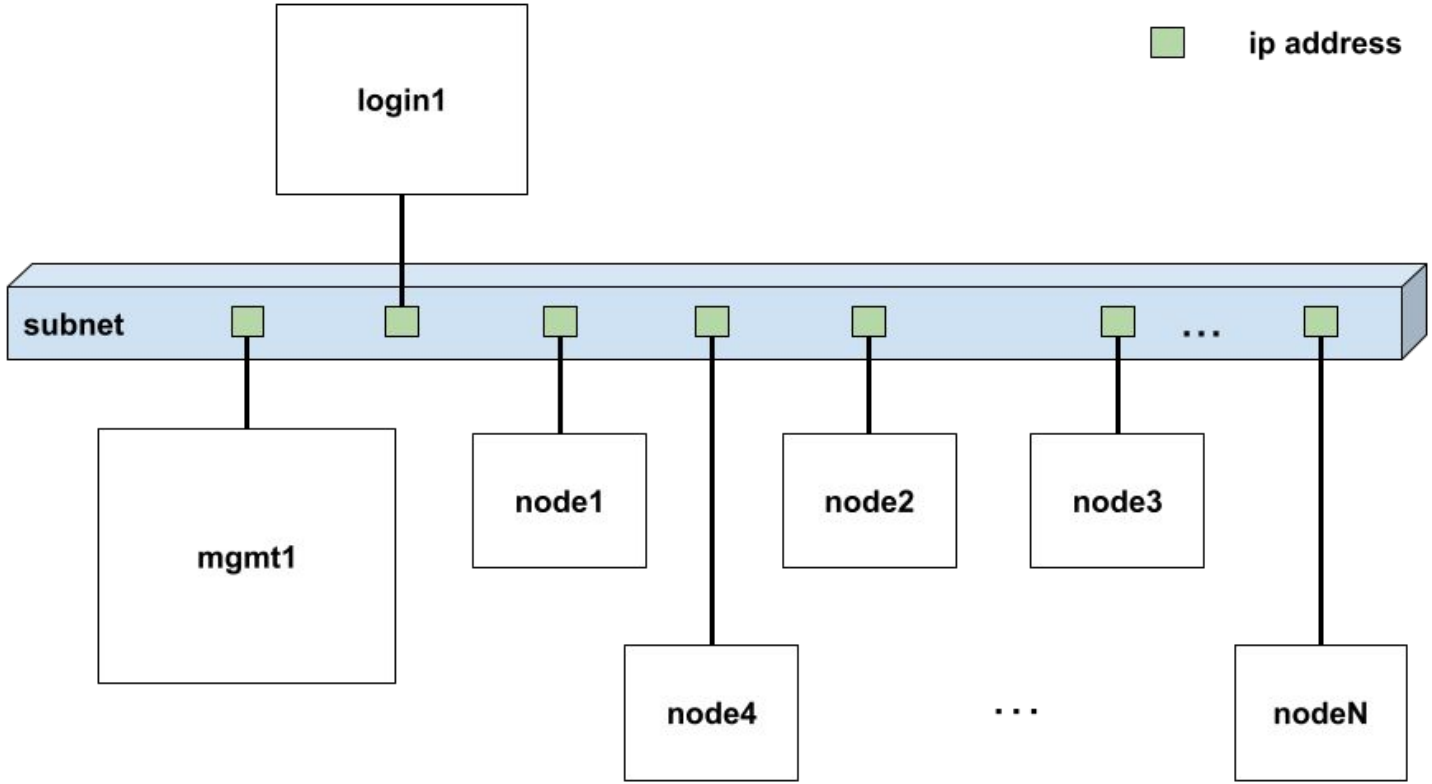
# Abouts tags in Puppet

- During the apply phase, Terraform copies on the puppet server a file containing the information about tags, instances and user input parameters.
  **`terraform_data.yaml`**
- This information is injected in Puppet information hierarchy and can be used to determine an instance role in place of its hostname.

# Preview of MC 12

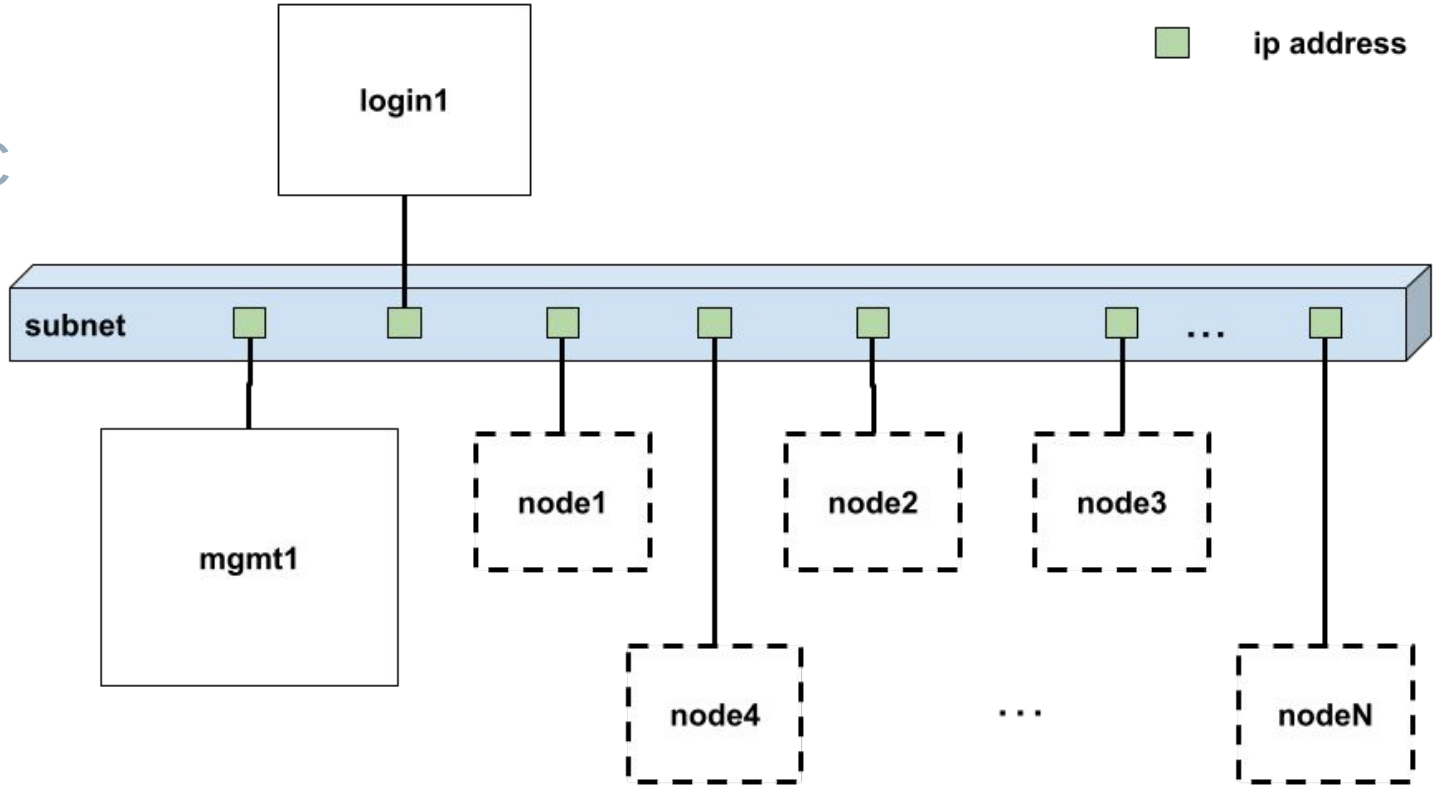
## Autoscaling compute with Terraform Cloud

# Static




```
node={type="...", tags=["node"], count=N}
```

# Elastic




```
node={type="...", tags=["node", "draft"], count=N}
```

# Node specifications as puppet data

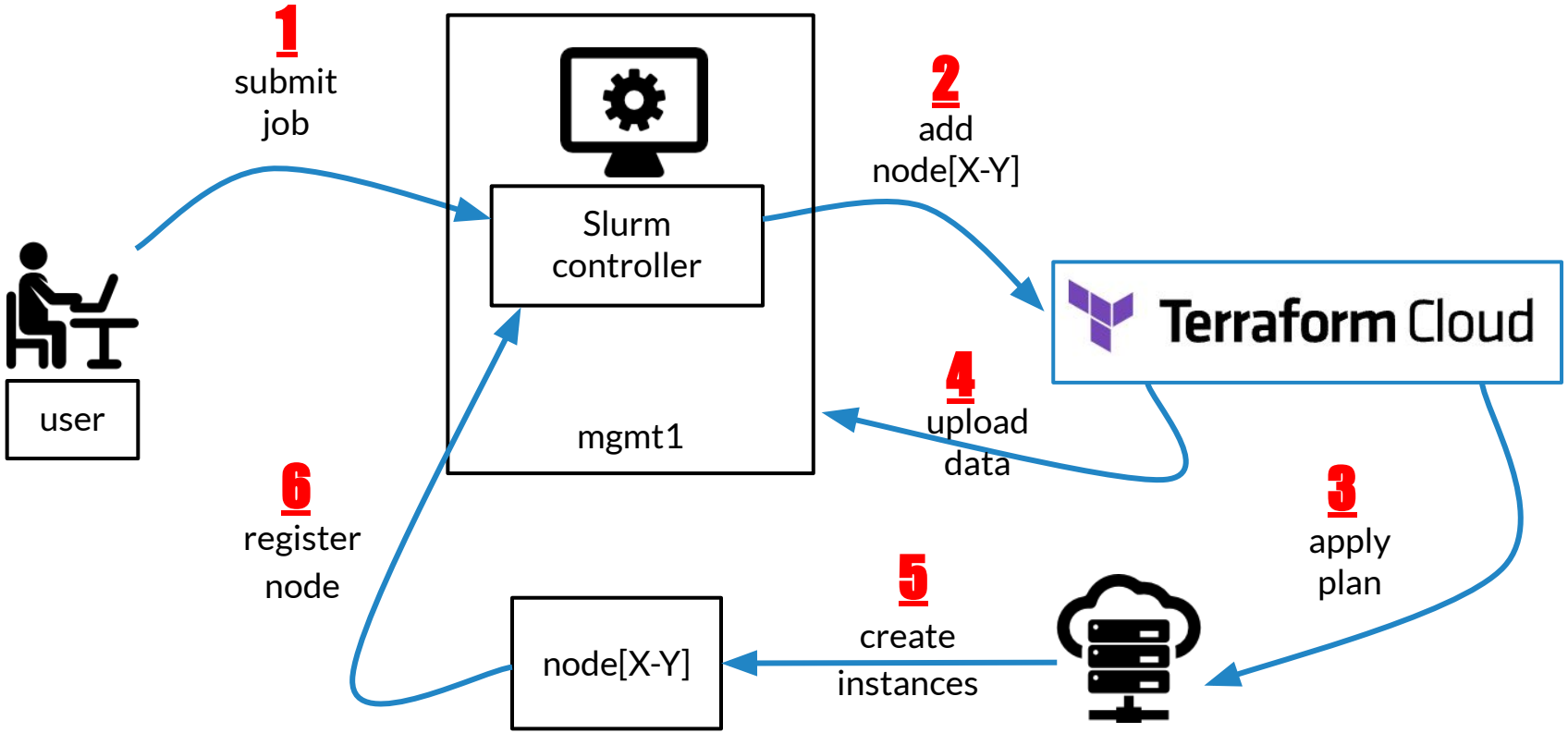
```
"node4":
  "hostkeys":
    "ed25519": ssh-ed25519 …
    "rsa": ssh-rsa …
  "id": "droid-node4"
  "local_ip": "10.0.0.11"
  "public_ip": ""
  "specs": { "cpus": "2", "gpus": 0, "ram": "8000" }
  "tags": ["node", "draft"]
```

# Made available in Slurm

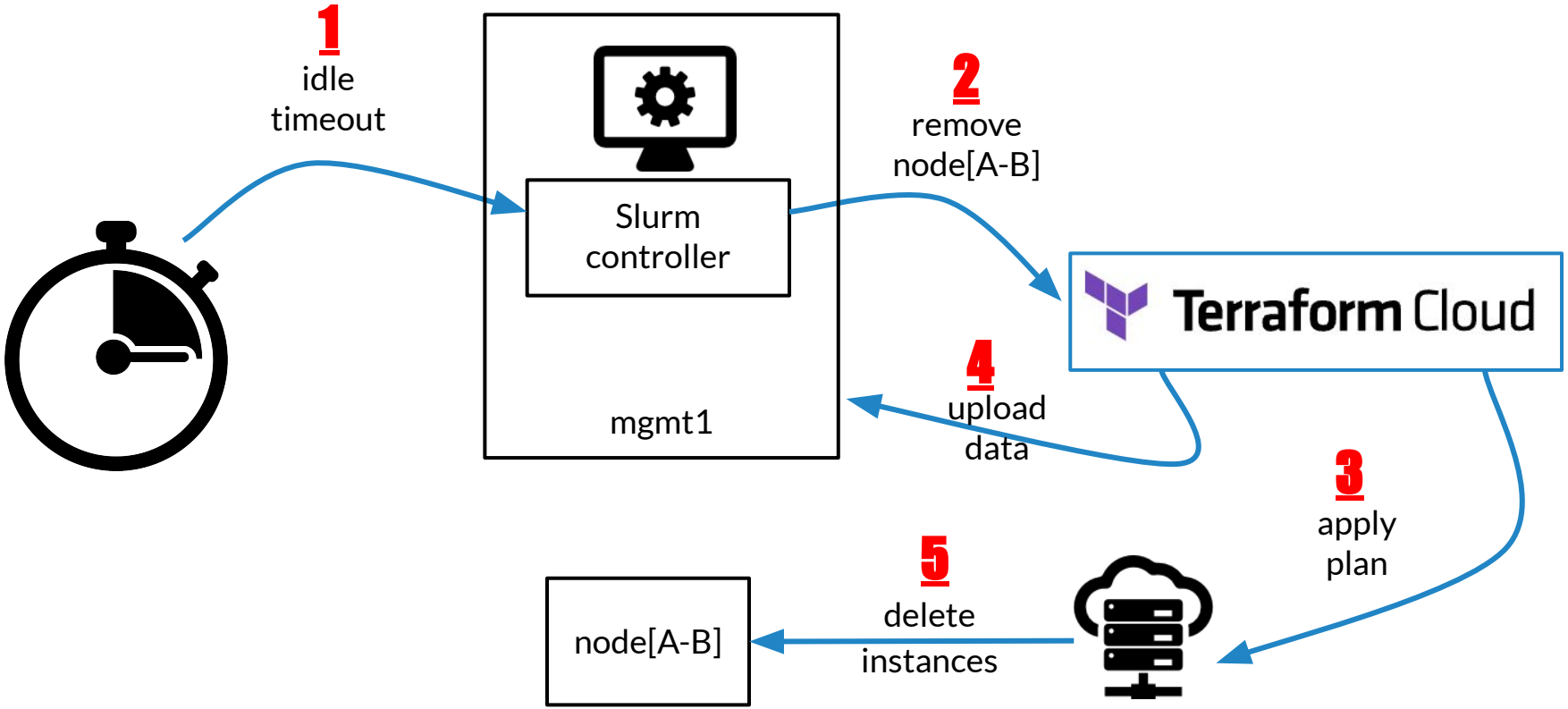
```
NodeName=DEFAULT MemSpecLimit=512 State=CLOUD


NodeName=node1 CPUs=2 RealMemory=8000 Gres=gpu:0 Weight=1
NodeName=node2 CPUs=2 RealMemory=8000 Gres=gpu:0 Weight=1
NodeName=node3 CPUs=2 RealMemory=8000 Gres=gpu:0 Weight=1
NodeName=node4 CPUs=2 RealMemory=8000 Gres=gpu:0 Weight=1
NodeName=node5 CPUs=2 RealMemory=8000 Gres=gpu:0 Weight=1
```

# Scaling up on job submission



# Scaling down



# More details

- Cluster side scaling logic is cloud provider agnostic.

- Only uses a Terraform Cloud API token to modify a single Terraform variable containing the list of compute nodes

slurm
workload manager

- Compatible with Slurm >= 20.11

# More details


- Compute infrastructure can be heterogeneous. Slurm will determine which instances to power up based job's specification and nodes' weight.


- Resume-suspend script interacting with TF Cloud is extremely simple and easy to maintain.

Ref: https://github.com/MagicCastle/elastic-slurm-tf-cloud/blob/main/src/elastic_slurm_tf_cloud/__init__.py

# How?

1. Create a git repo with MC code for your cluster
2. Link a Terraform Cloud workspace with the repo
3. Configure the workspace variables
4. Add workspace info and token to hieradata
5. Start apply run in Terraform cloud

Reference: https://github.com/ComputeCanada/magic_castle/blob/elastic/docs/terraform_cloud.md

# How fast before the job starts?

- 10 minutes when nodes are configured from a base OS image

- 3 minutes when using a snapshot of a previously configured node

```
node={type="...", tags=["node", "draft"], count=N, image="node"}
```

# Summary


- Next release of Magic Castle will allow the definition of minimal infrastructure that can be scaled up and down automatically based on the Slurm queue using Terraform Cloud.
- The scaling logic will be cloud provider agnostic.
- Terraform Cloud free tier is sufficient for MC needs.
- Release is expected by end of September 2022.

# Contact & References

- Félix-Antoine Fortin (@cmd-ntrf)
  felix@calculquebec.ca / felix-antoine.fortin.1@ulaval.ca
- Magic Castle git repository:
  https://github.com/ComputeCanada/magic_castle
- Puppet Magic Castle git repository
  https://github.com/ComputeCanada/puppet-magic_castle
- Magic Castle puppet environments:
  https:/github.com/MagicCastle
- MC-Hub
  https://www.github.com/computecanada/mc-hub