# Cluster in the Cloud

Easy, Scalable, Heterogeneous

CLUSTER IN
THE CLOUD

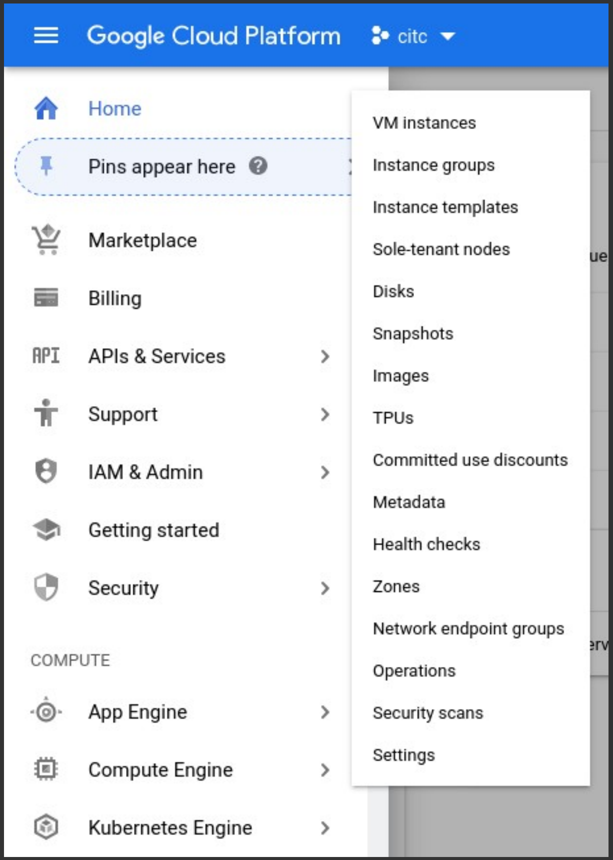
Matt Williams
Research Software Engineer
University of Bristol

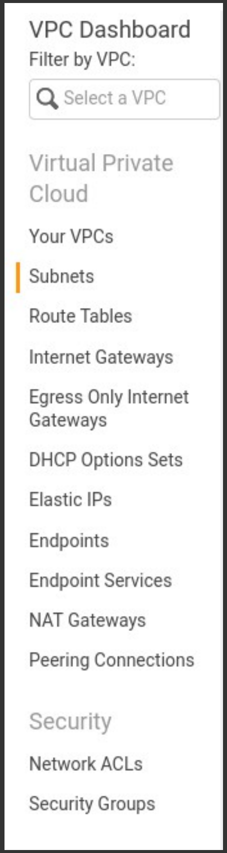
# The problem

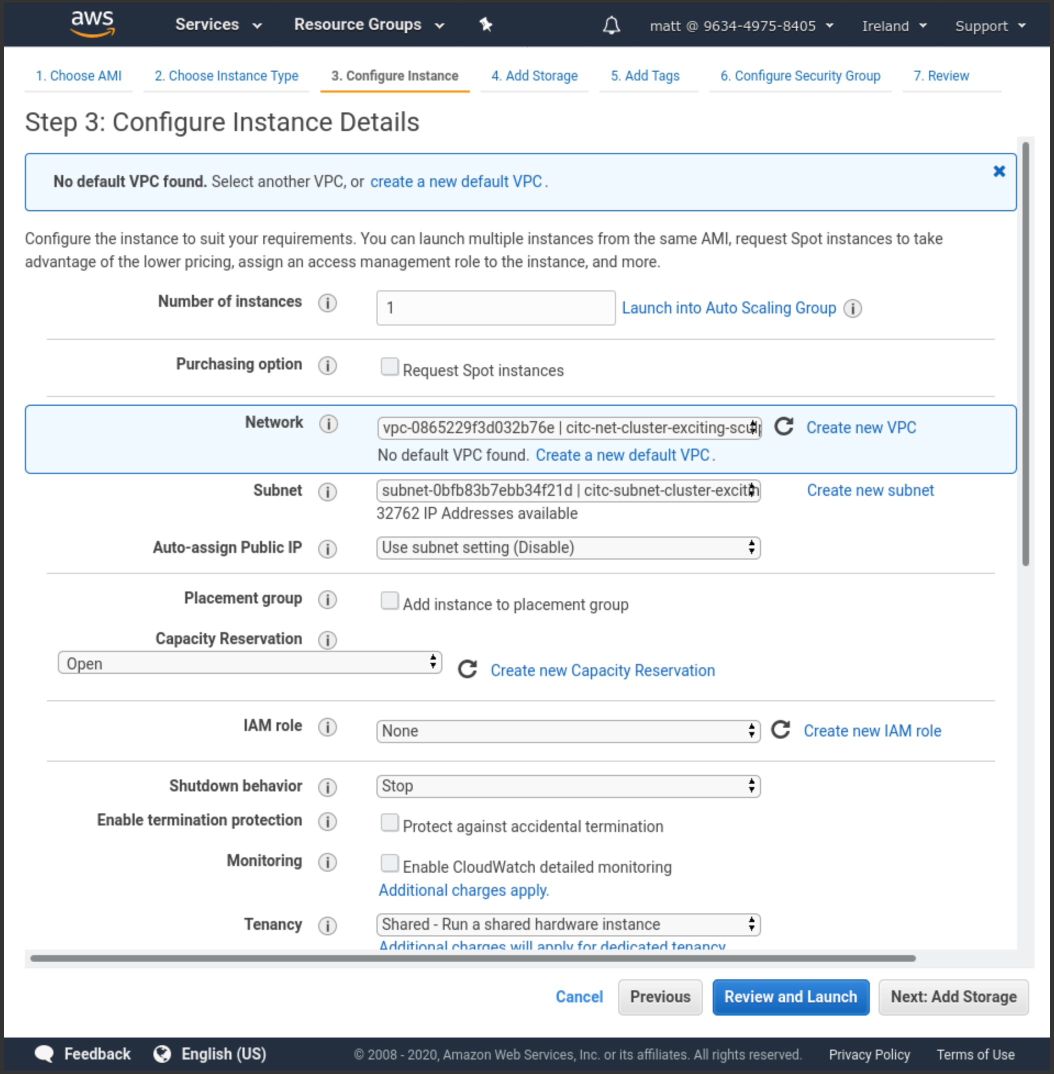
- Researchers having cloud credits

# The problem 😕

- Researchers having cloud credits
- Presented with:







Step 3: Configure Instance Details

**No default VPC found.** Select another VPC, or create a new default VPC.

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances: 1 — Launch into Auto Scaling Group

Purchasing option: □ Request Spot instances

Network: vpc-0865229f3d032b76e | citc-net-cluster-exciting-sc — Create new VPC
No default VPC found. Create a new default VPC.

Subnet: subnet-0bfb83b7ebb34f21d | citc-subnet-cluster-excit — Create new subnet
32762 IP Addresses available

Auto-assign Public IP: Use subnet setting (Disable)

Placement group: □ Add instance to placement group

Capacity Reservation: Open — Create new Capacity Reservation

IAM role: None — Create new IAM role

Shutdown behavior: Stop

Enable termination protection: □ Protect against accidental termination

Monitoring: □ Enable CloudWatch detailed monitoring
Additional charges apply.

Tenancy: Shared - Run a shared hardware instance
Additional charges will apply for dedicated tenancy.

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# The problem 😕

- What they already know:
  - Their field of research
  - Python/R/GROMACS/Relion
  - `sbatch/qsub`
- We can’t expect researchers to be professional sysadmins
  - The intersection is well handled by *Research Software Engineers*

# The solution ✅

- Give them what they are used to, but in a cloud environment
- They don’t have to know the difference
- Except:
  - No queuing
  - Only pay for what they use
- *Cluster in the Cloud*

# Cluster in the Cloud

- An automatically-provisioned Slurm cluster
- Uses Terraform to create:
  - Networking
  - Shared file system (Elastic File System)
  - Management/login VM (`t3a.medium`)
- Uses Ansible to configure the management VM and compute image

# Key Features

1. **Familiar**: known environment for researchers with Slurm, JupyterHub etc.
2. **Versatile**: Allows any number of any combination of instance types in a cluster
3. **Dynamic**: They are started only when needed
4. **Cheap**: Base cost is just one VM plus storage
5. **Cross-cloud**:  Works on AWS, Google Cloud and Oracle
6. **Open source**: Under the MIT license and is free to use.

# Technical details: Terraform

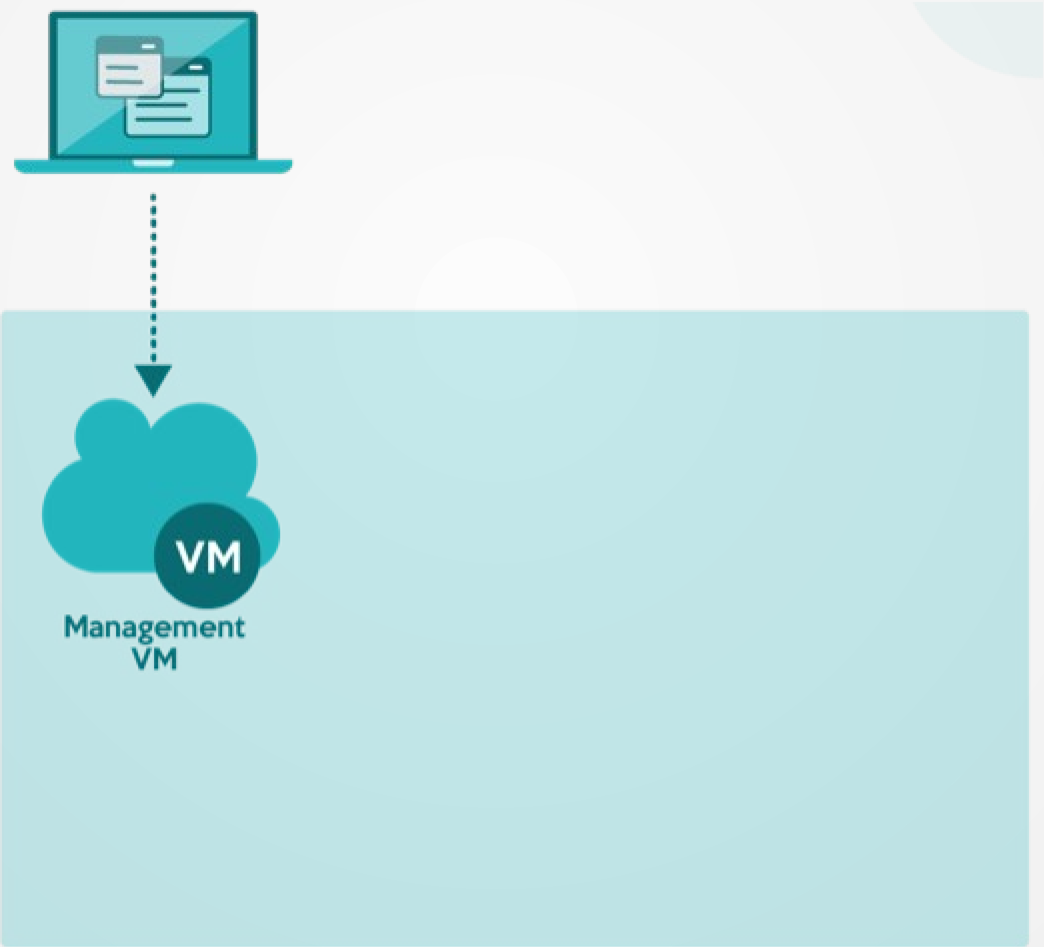
- Terraform is used to create the skeleton
- https://github.com/clusterinthecloud/terraform
  - AWS: ~500 LOC
  - Google: ~400 LOC
  - Oracle: ~450 LOC
- Written from scratch for each platform

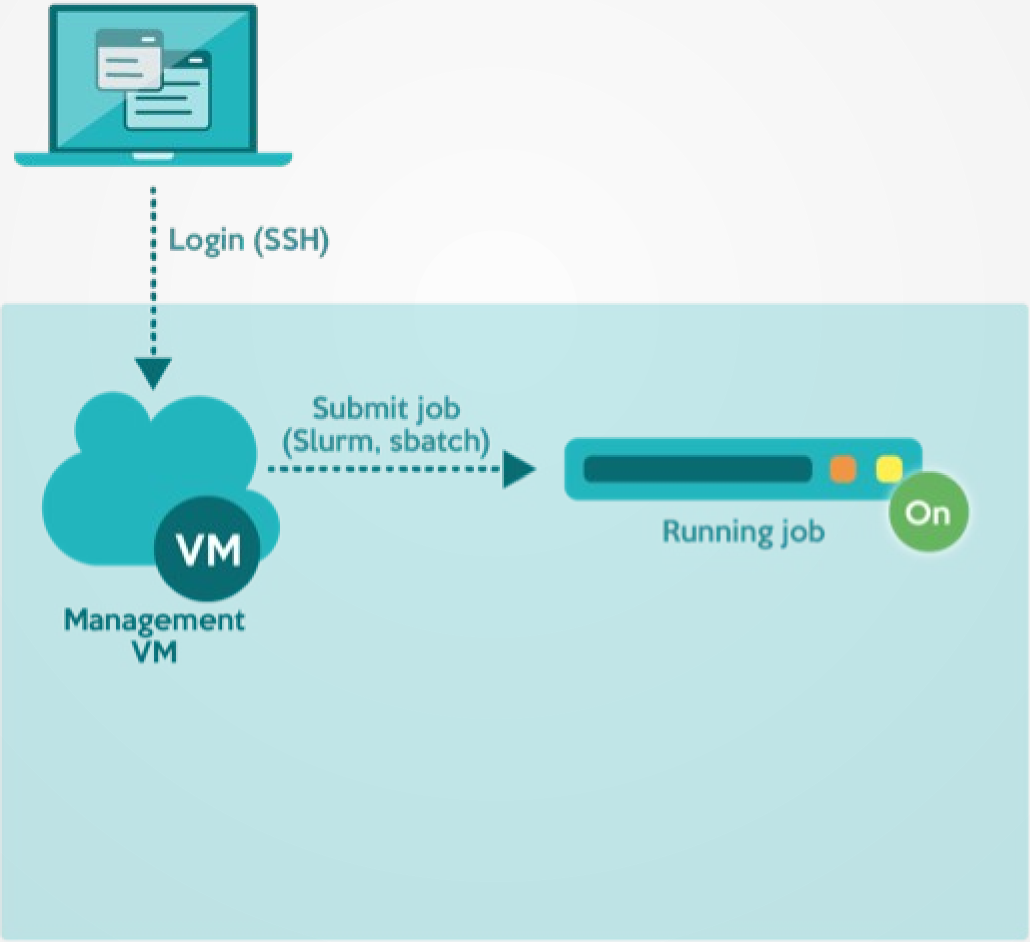
# Technical details: Ansible

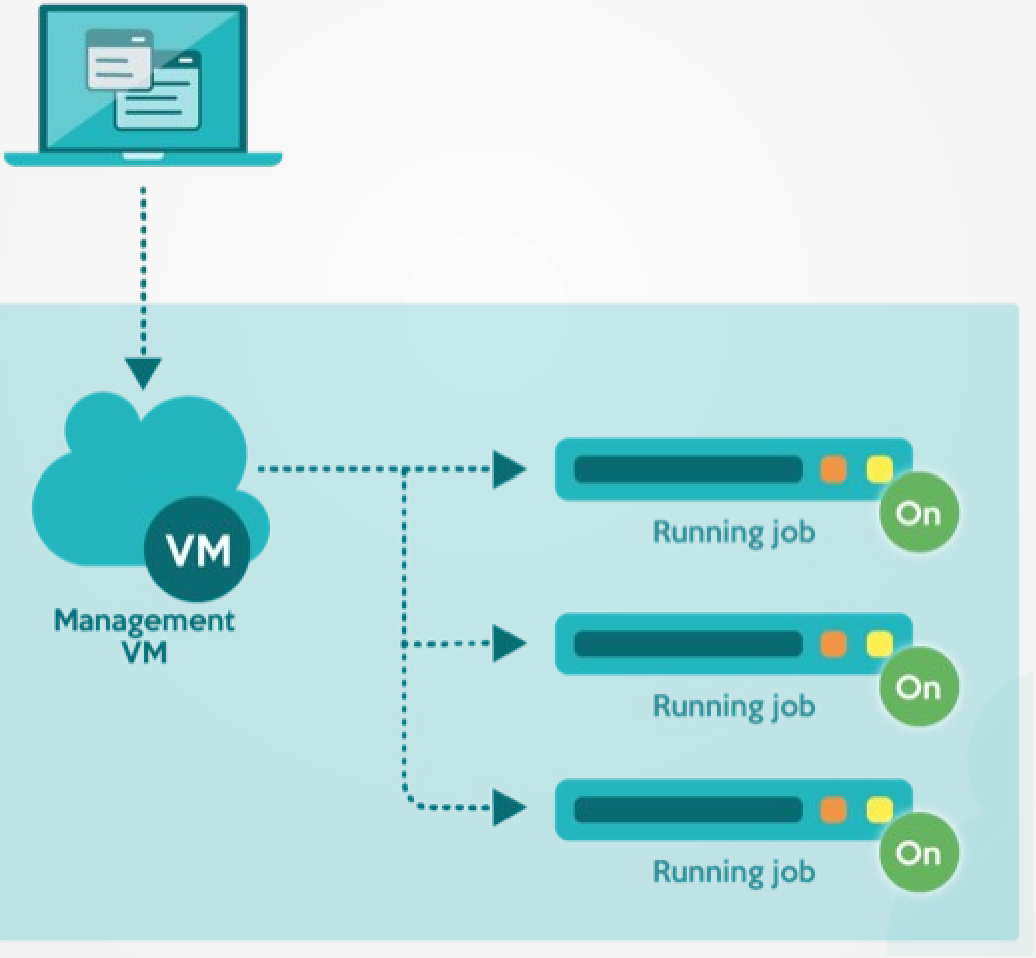
- ~1.5K lines of Ansible
- https://github.com/clusterinthecloud/ansible
- Configures:
  - Mounting shared filesystem
  - LDAP for user management
  - Slurm
    - Including node start/stop scripts
  - Monitoring (Grafana)
  - Base software set
  - And more...
- Covers both the management VM and compute image

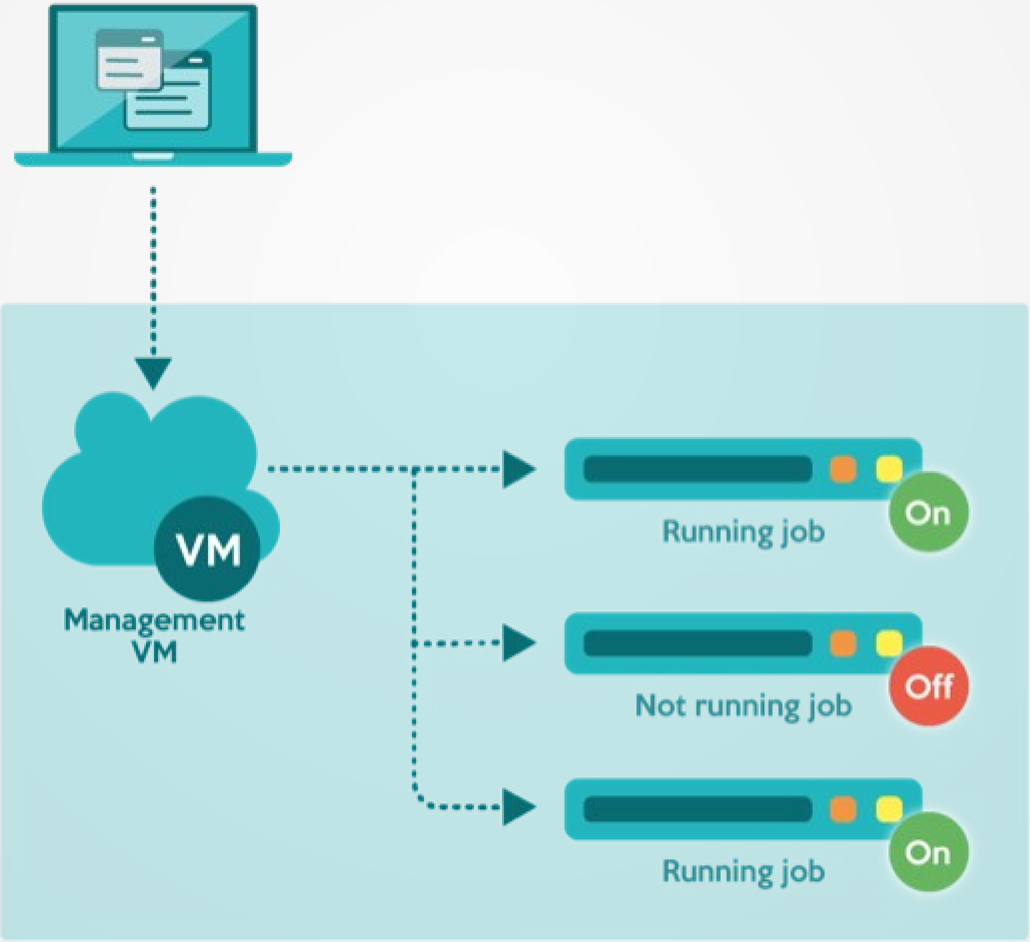
# ⚡ Slurm power management

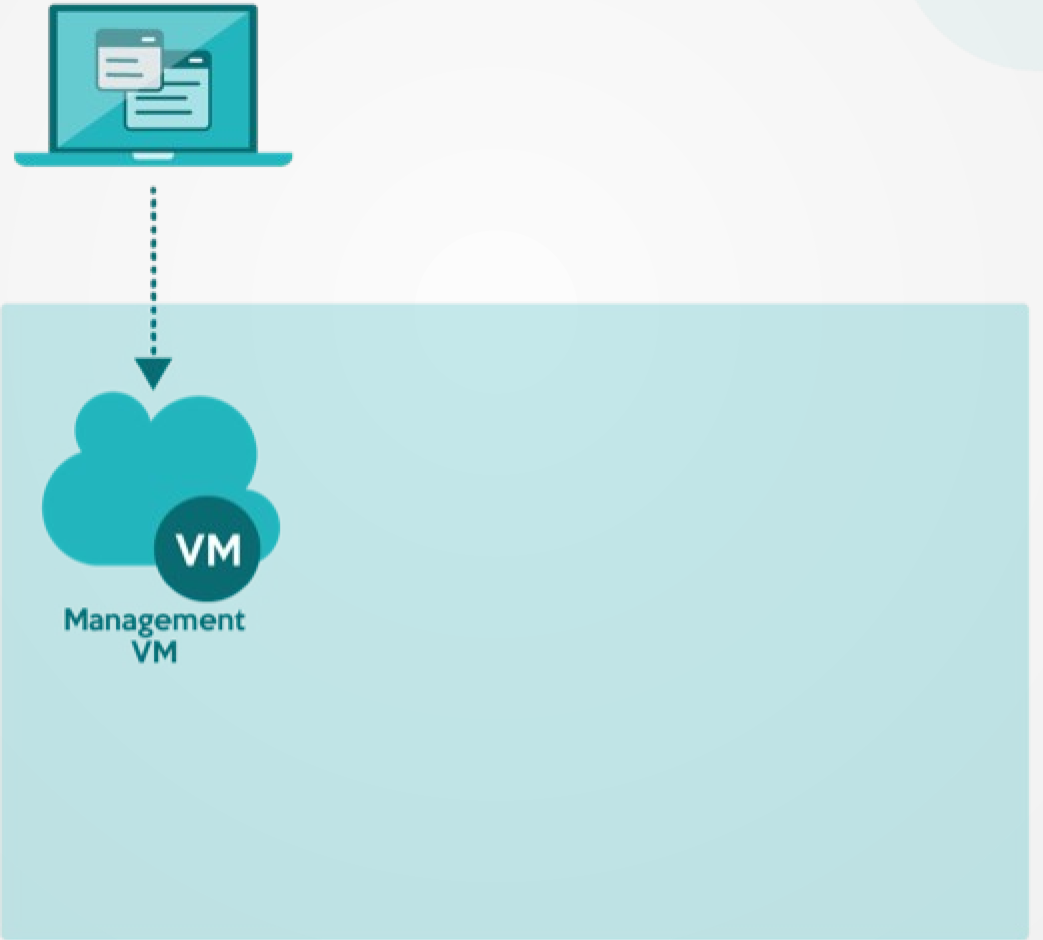
- Python plugin calls the AWS API
- Initial configuration creates any number of *potential* nodes of each desired type:
  - e.g. 1000 32-core, 1000 16-core, 1000 GPU etc.
- On job submission Slurm
  1. Chooses a node type
  2. Creates an instance from an image
  3. Runs the job
  4. Destroys it (after a timeout)


VM
Management
VM


Login (SSH)
Submit job
(Slurm, sbatch)
VM
Management
VM
Running job
On


Management VM
VM
Running job
On
Running job
On
Running job
On

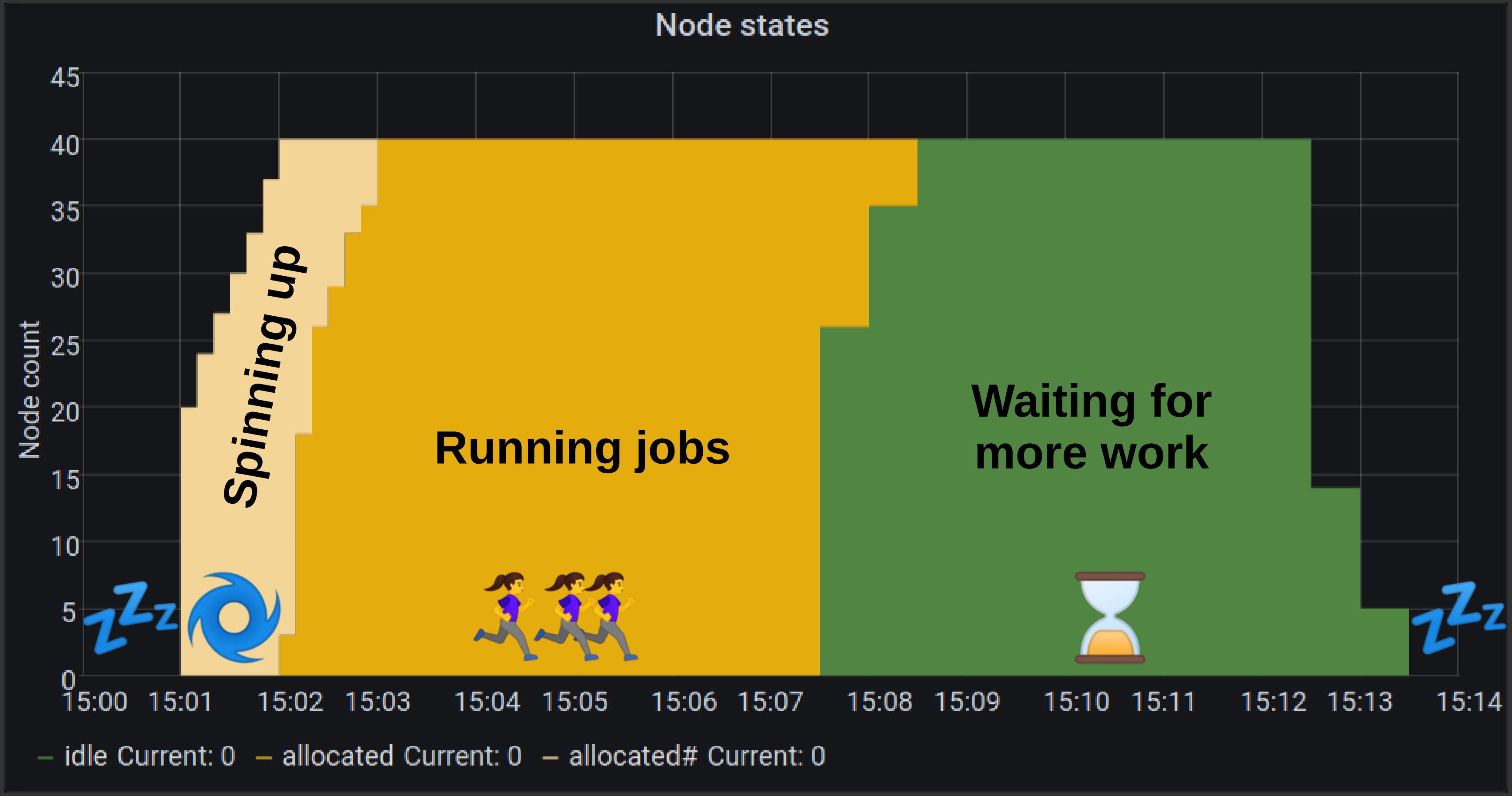

VM
Management VM
Running job
On
Not running job
Off
Running job
On



VM
Management VM

# Node states

- 40-node array job, 5 minute runtime



# Timings ⏱️

- Full system test ~17 minutes on AWS
  1. Create cluster from scratch, including node images
  2. Run test job
  3. Check other system statuses
  4. Tear down whole cluster
- Job submit → job start: 1 minute

# Performance characteristics 🏎️

- ✓ Best-suited to heterogeneous high-throughput tasks
  - Pipelines needing different node type for different parts
  - Can be much more specific than the average on-premise cluster
  - Always access to latest hardware, e.g Graviton 2
- ✖ At present is not optimised for multi-node workloads
  - No fast interconnect support
  - Only cheap shared storage
- ✓ Great for teaching clusters and benchmarking
- ✓ Suitable for Dask, Spark, Singularity

# Users

- **Smoking cessation**: A General Mechanism for Signal Propagation in the Nicotinic Acetylcholine Receptor Family
  10.1021/jacs.9b09055
- **Vaccine delivery**: Synthetic self-assembling ADDomer platform for highly efficient vaccination by genetically encoded multiepitope display
  10.1126/sciadv.aaw2853
- **Other projects**:
  - COVID research
  - Molecular dynamics
  - Carbon sequestration
  - Radiotherapy research

# Hackathon streams

**1) Spot instances**

*Allow creating compute nodes on the spot market*

**2) Benchmarking**

*Automatically profile and benchmark workloads*

**3) Shared storage**

*Implement a persistent shared storage layer*

**4) Elastic Fabric Adapter**

*Integrate AWS EFA for multi-node workloads*

# Getting help

- During this workshop, ask in the chat and we can direct to GitHub if needed
- First stop should be github.com/clusterinthecloud/support/issues
- We can triage them there and move as necessary
- If it’s a feature request or a bug report, I’m happy to guide you to fixing it yourself!

# Demo time

Let’s see how this works in practice

# Thank you

Find out more at
cluster-in-the-cloud.readthedocs.io

Thanks to AWS, Google and Oracle for supporting development and to the Bristol RSE team