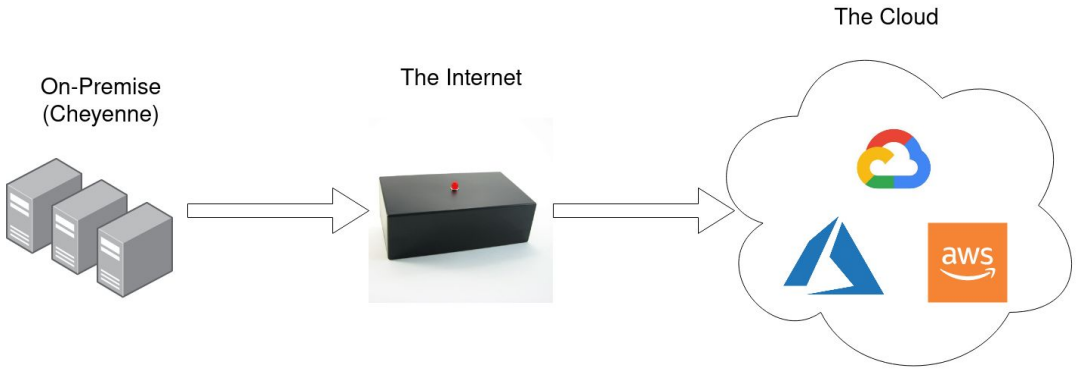
Cloud Bursting

Exploring Using the Cloud at NCAR

Will Shanks - shanks@ucar.edu
Student Visitor SSG

What is Cloud Bursting?

- When on-premise resources do not meet demand, “burst” out to a public cloud
- Users submit jobs the same way as they usually do to on-premise resources
  - Besides submitting to an alternate queue?
- Like a normal job, just runs on rent-able hardware
  - Generally short lived cloud instances (spun up for the job, then spun down afterwards)

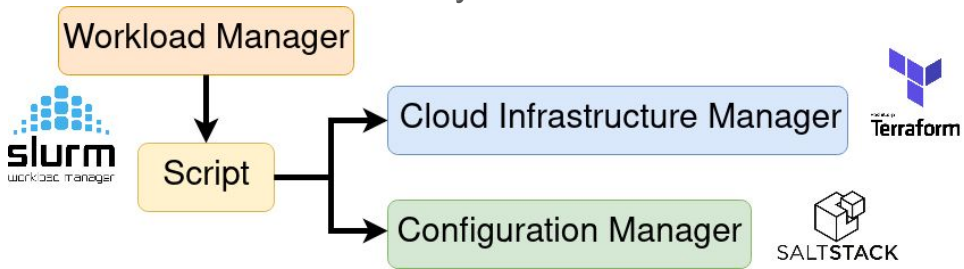


Potential Use Cases

- Don’t want to wait for local resources
  - Interactive job
  - Time sensitive jobs
  - Other?
- Reservations
  - Don’t need to reserve Cheyenne/Casper nodes all day for a workshop
  - During outages
- Trying new hardware/special cases
  - AWS has ARM
  - GCP has Tensor Processor Units
  - Azure has InfiniBand
  - etc.

High Level Design

- Workload manager decides when cloud resources should be used
  - Slurm, PBS Pro, LSF, etc.
  - Runs a program when resources should be started/stopped
  - Leveraging Slurm because it is presently deployed in the HPCFL
- Cloud resources started/stopped by a cloud infrastructure tool
  - Terraform, CloudFormation, etc.
  - Chose Terraform as it is well documented and works with multiple clouds
- Resources configured using a configuration management tool
  - SaltStack, Ansible, Puppet, Chef, etc.
  - Selected SaltStack because it is already in use in the HPCFL

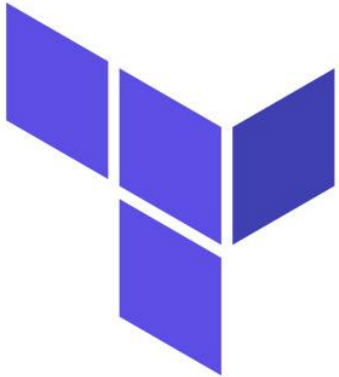


Workload Manager: Slurm Cloud Bursting

- Built off of Slurm's scheduler power saving feature
  - Calls a program to turn nodes on when they are needed (*ResumeProgram* hook)
  - Call a different program when nodes have been idle to turn them off (*SuspendProgram* hook)
- *ResumeProgram* hook
  - Start
  - Provision
  - Connect instance to Slurm
- *SuspendProgram* hook
  - Cleans up
  - Turn off node
  - Runs automatically after nodes have been idle for *SuspendTime* seconds
- Similar hooks exist in other workload managers

Infrastructure Management: Terraform

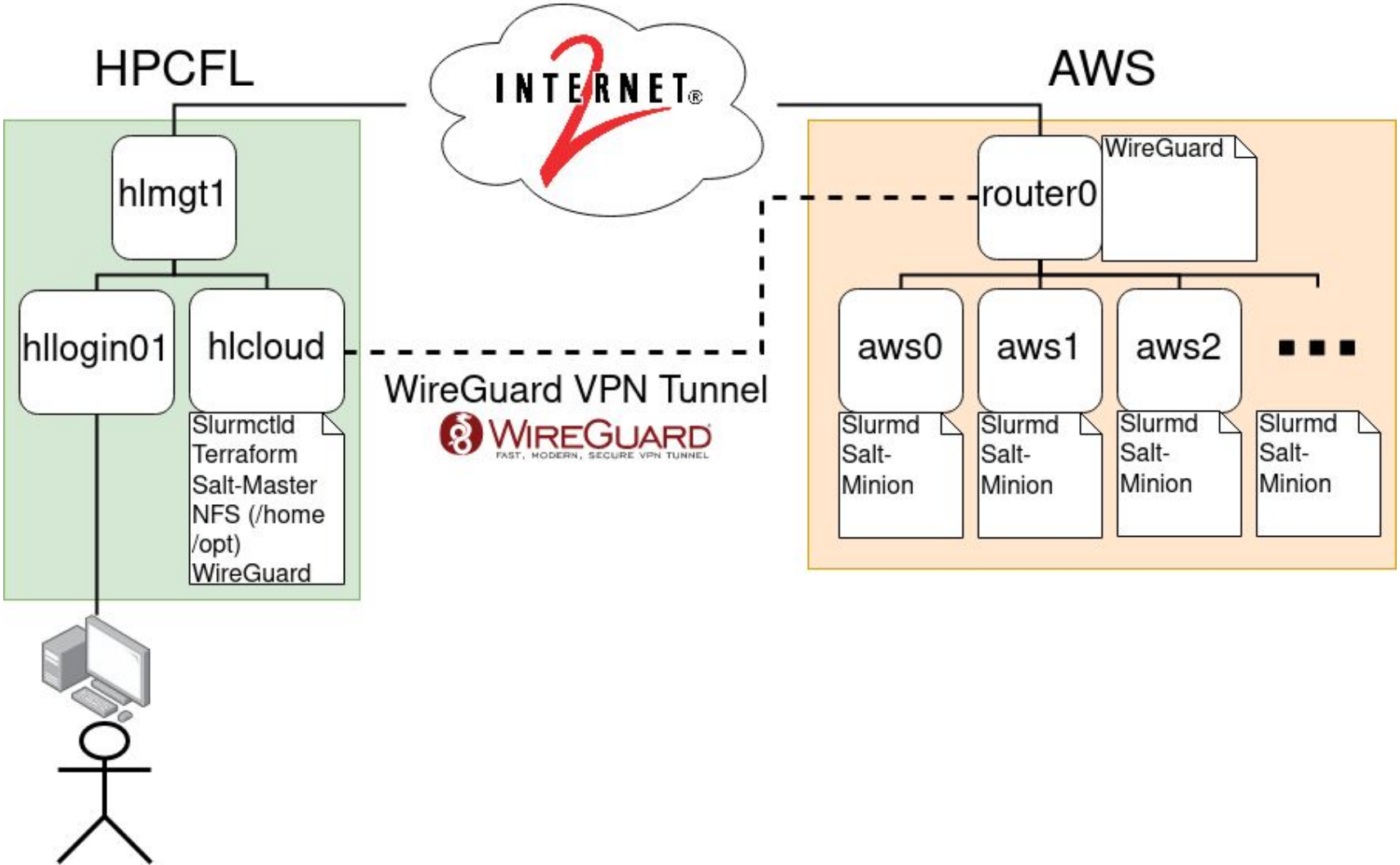
- Makes managing cloud infrastructure easy
  - Every cloud has a different API, Terraform only has **one**
  - Just update the config file when you want to change something
- Works with over 200 different resource orchestration services
  - AWS, Azure, GCP
  - VMware
  - Kubernetes
- Not easy to automate
  - No public API
  - CLI expects to be called manually



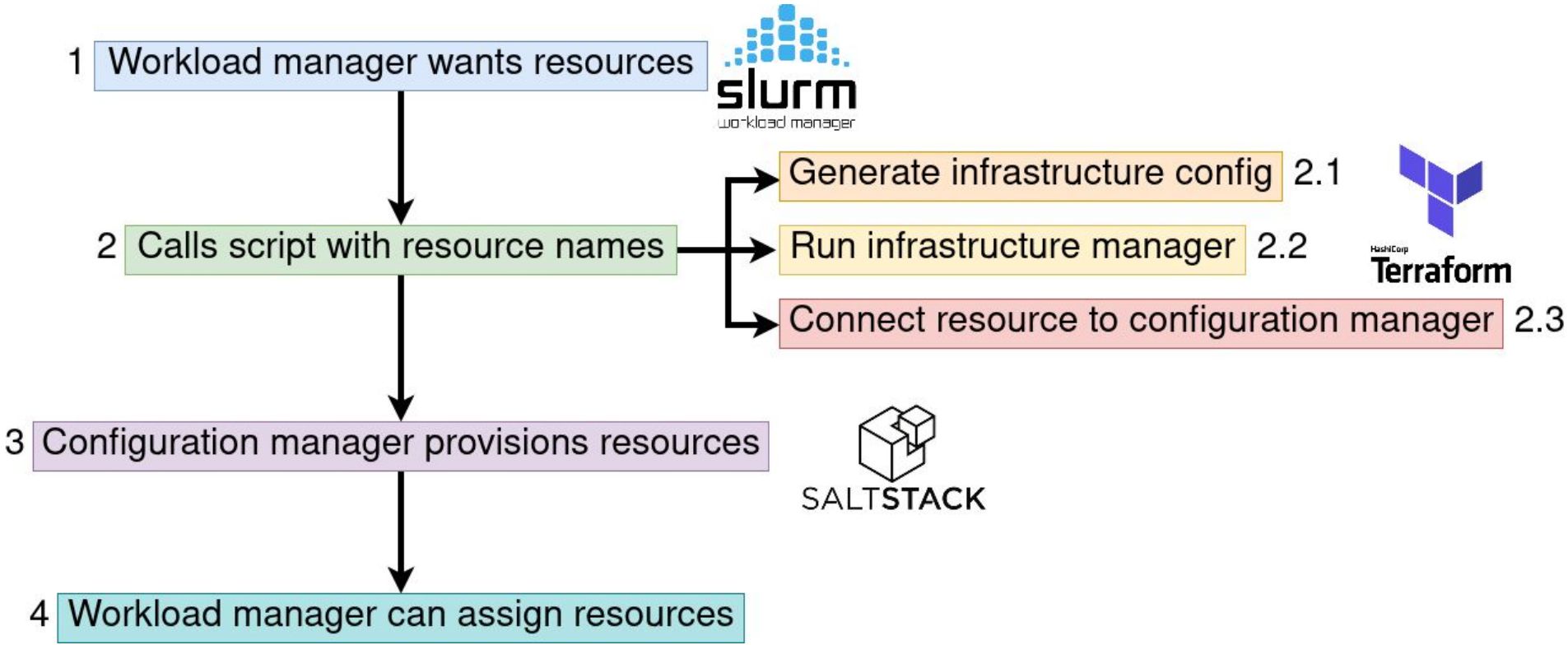
HashiCorp Terraform

Current Setup

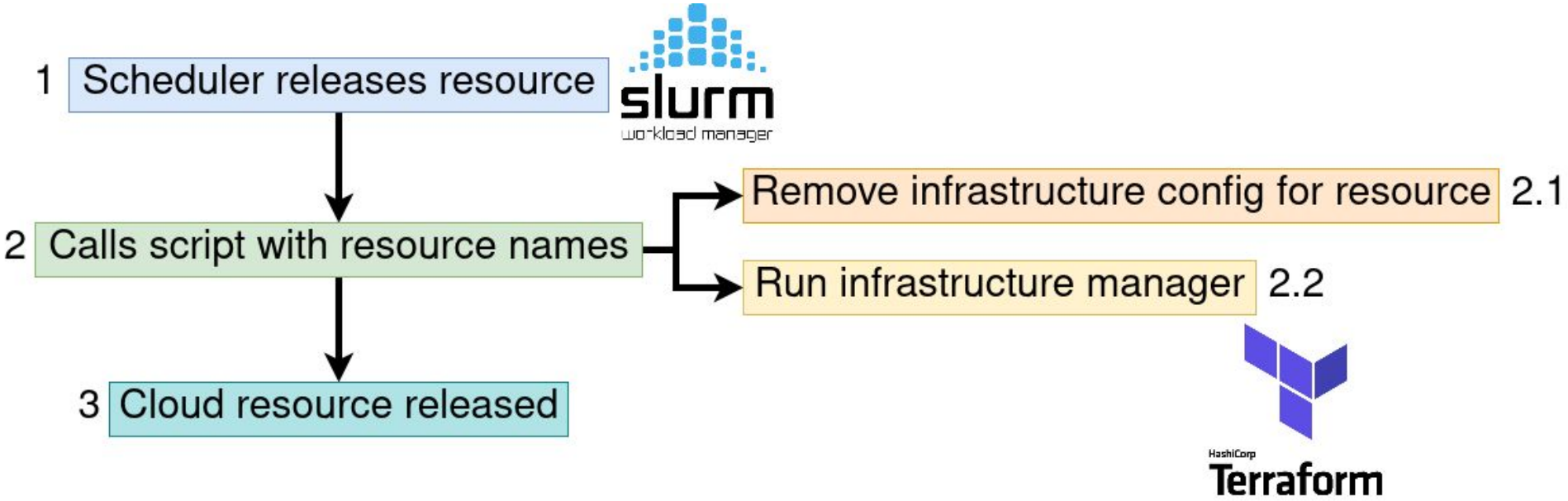
- Using AWS EC2 cloud instances - Already had AWS account for other projects
- VPN connection from HPCFL to an AWS EC2 instance *router0*
  - Runs over Internet2
  - Using WireGuard because it designed to be lightweight, secure, and will be in Linux kernel (5.6+)
- Multiple compute instances communicate with on-premise services via this connection *aws[0-9]*
- *Router0* instance is started manually
- Compute instances automatically spun up/down by workload manager(Slurm)


HPCFL
INTERNET2
AWS
hlmgt1
hllogin01
hlcloud
Slurmctld
Terraform
Salt-Master
NFS (/home
/opt)
WireGuard
WireGuard VPN Tunnel
WIREGUARD
FAST, MODERN, SECURE VPN TUNNEL
router0
WireGuard
aws0
aws1
aws2
Slurmd
Salt-
Minion
Slurmd
Salt-
Minion
Slurmd
Salt-
Minion
Slurmd
Salt-
Minion

How Instances are Acquired



How Instances are Released



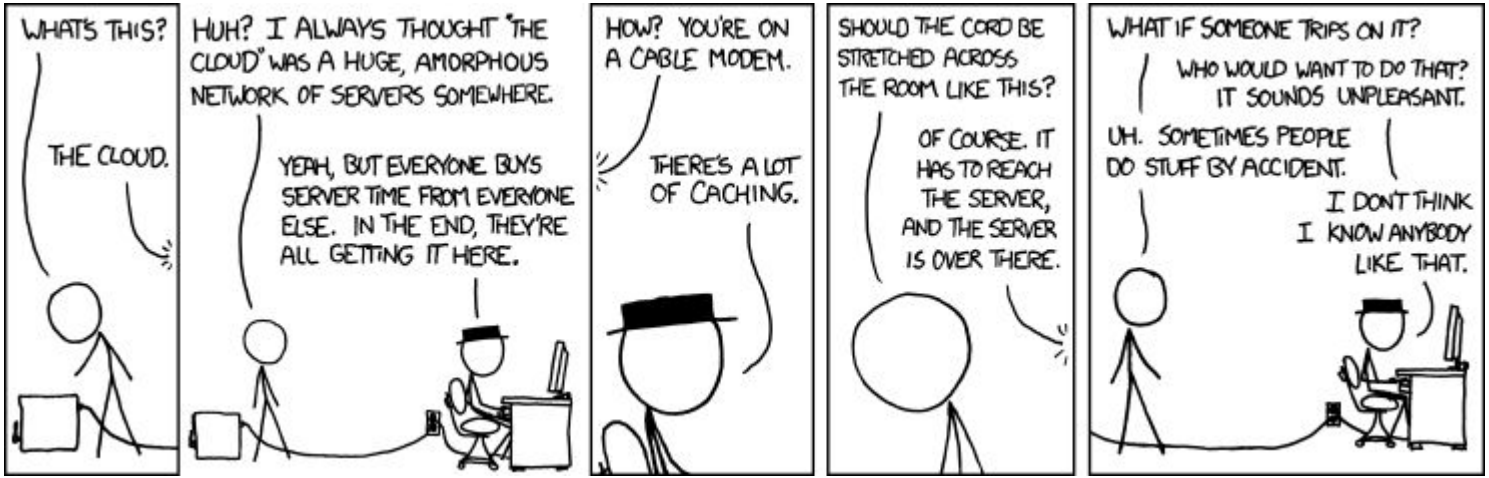
Work In Progress

- Switch to CentOS from Amazon Linux
- Improve security
- Make a custom AWS machine image (AMI)
  - Reduce resource provisioning time
  - Less network usage - don’t need to install as much during setup
- Automate infrastructure
  - Router AWS instance has to be manually started
  - Need safety checks
- Make Terraform successfully concurrently callable
  - Required for actual use
  - Needs better locking
- Make everything public

Future Work

- Minimize data egress/ingress by moving data out to the cloud
  - Costs money and time to move data in/out of cloud networks
  - Burst Buffer?
- Benchmark applications on cloud
- Test at scale
  - Number of instances
  - Network bandwidth
  - etc.
- Test different cloud instances - small/big
- Try different clouds - Azure, GCP
- Try different workload manager - PBS Pro

Questions?



Will shanks
shanks@ucar.edu