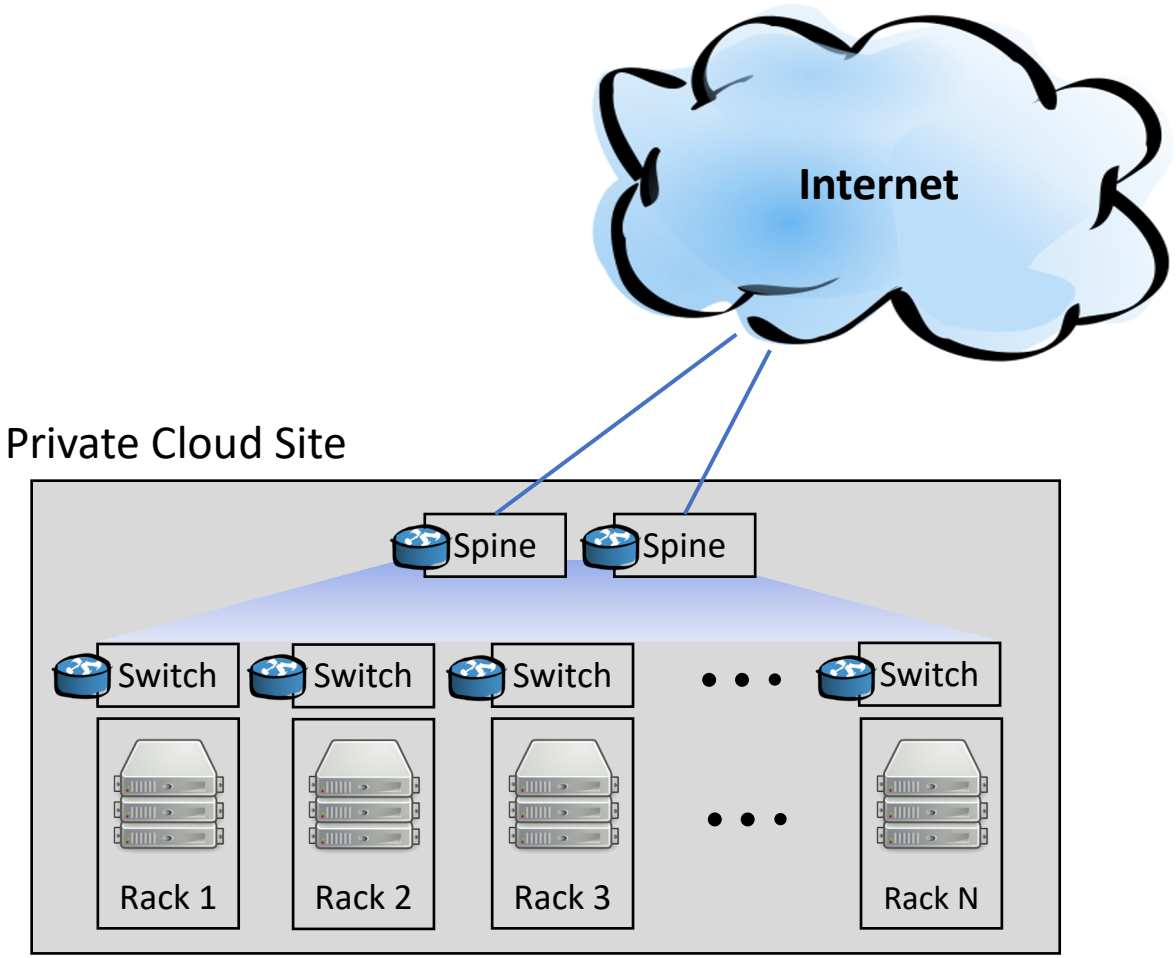
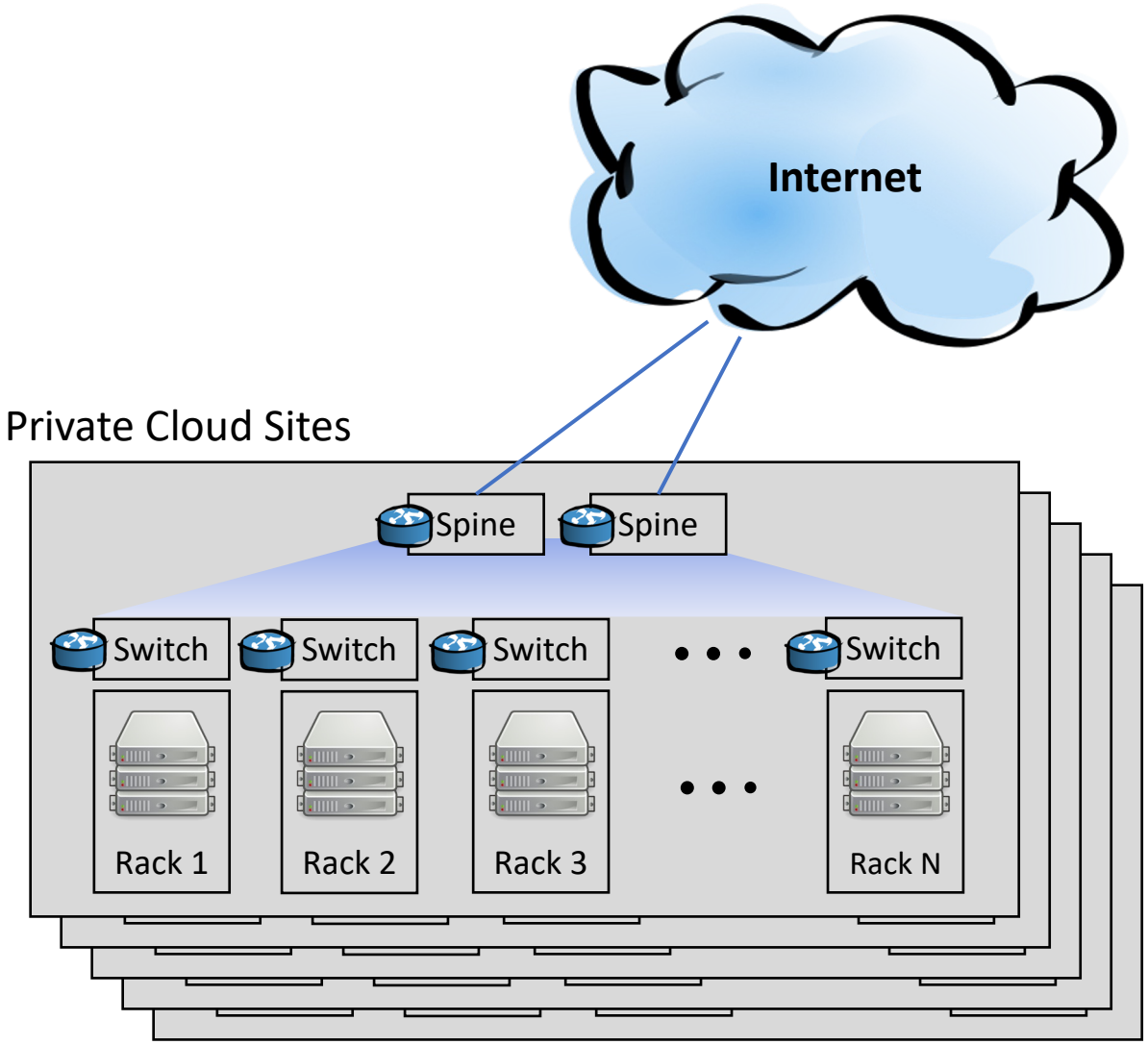
# Provisioning Private Clouds

Prof. Ítalo Cunha

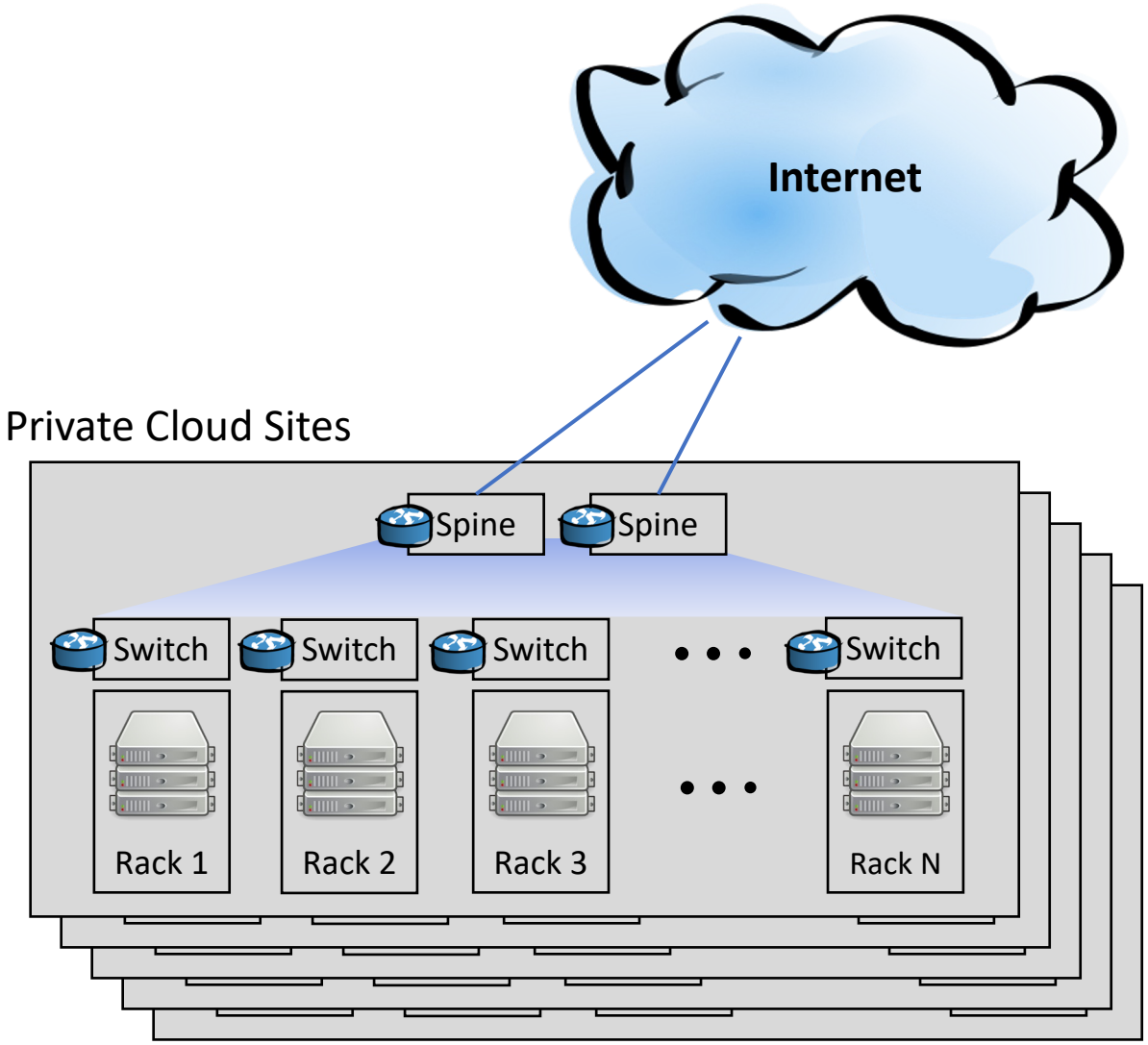
# Control and Management

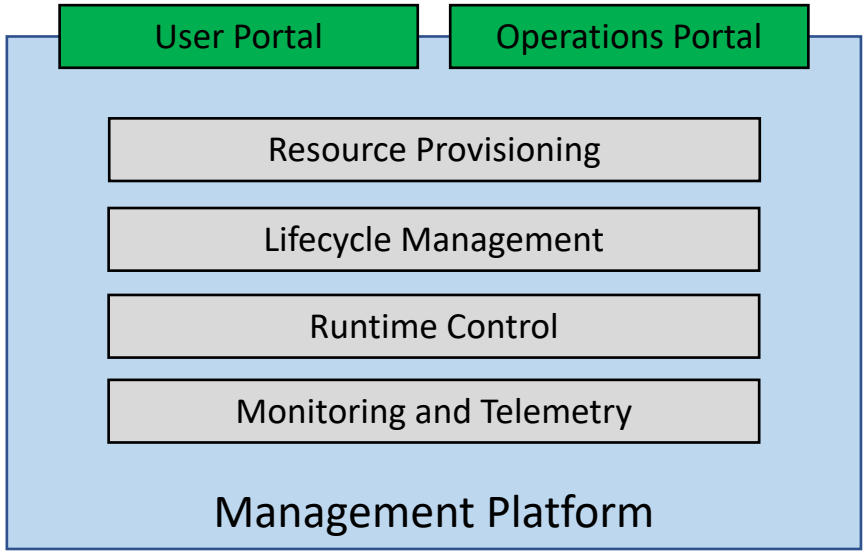


# Control and Management



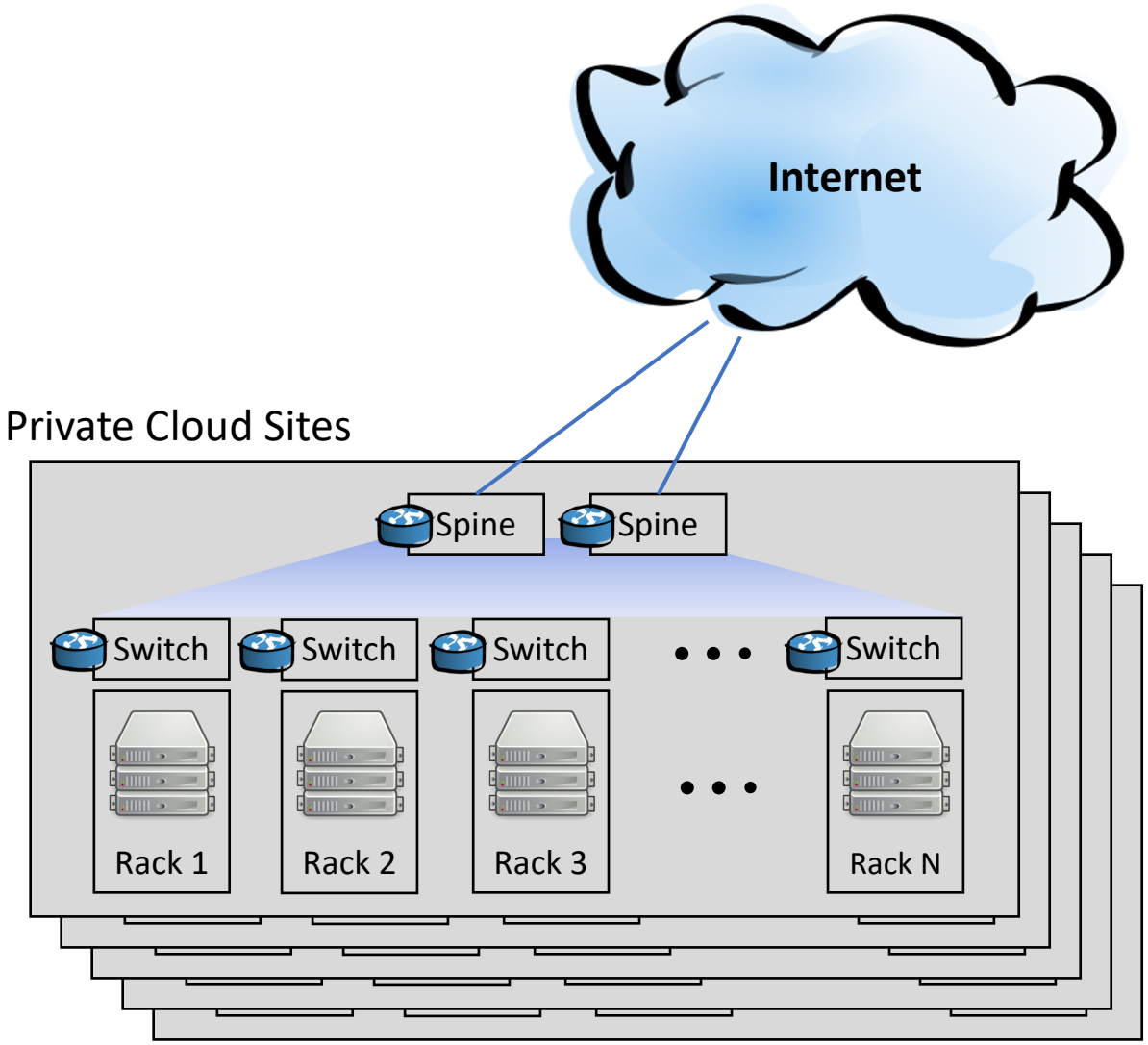
# Control and Management

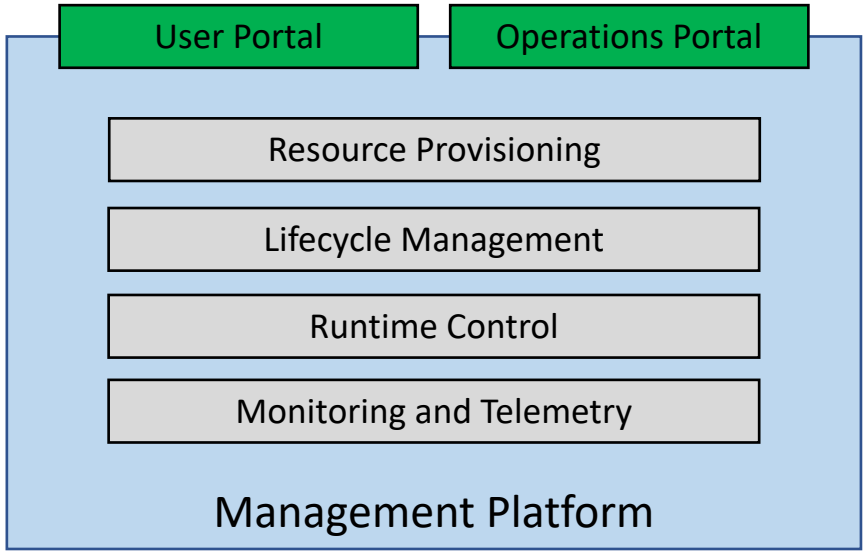


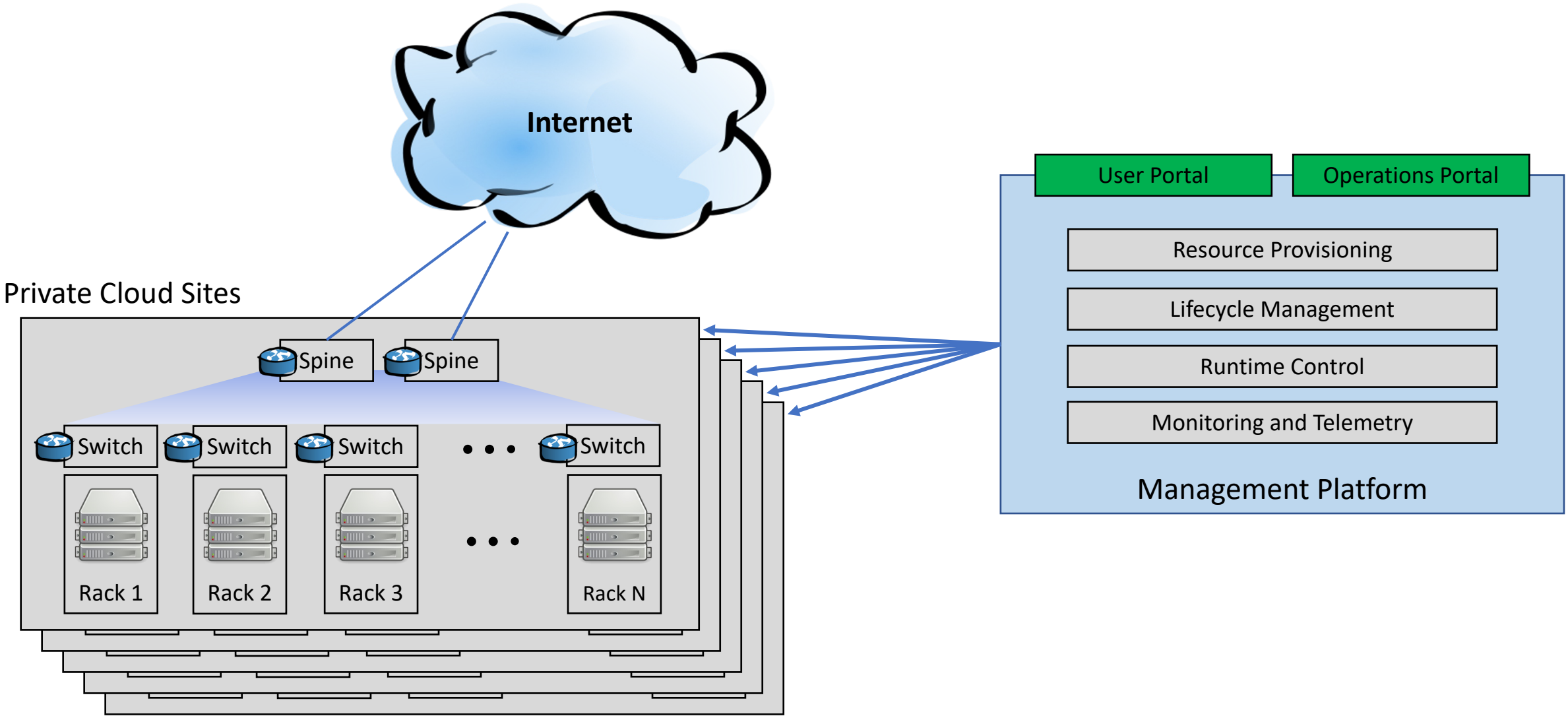
# Control and Management

Stakeholders:
- Users
- Operators
- Service providers

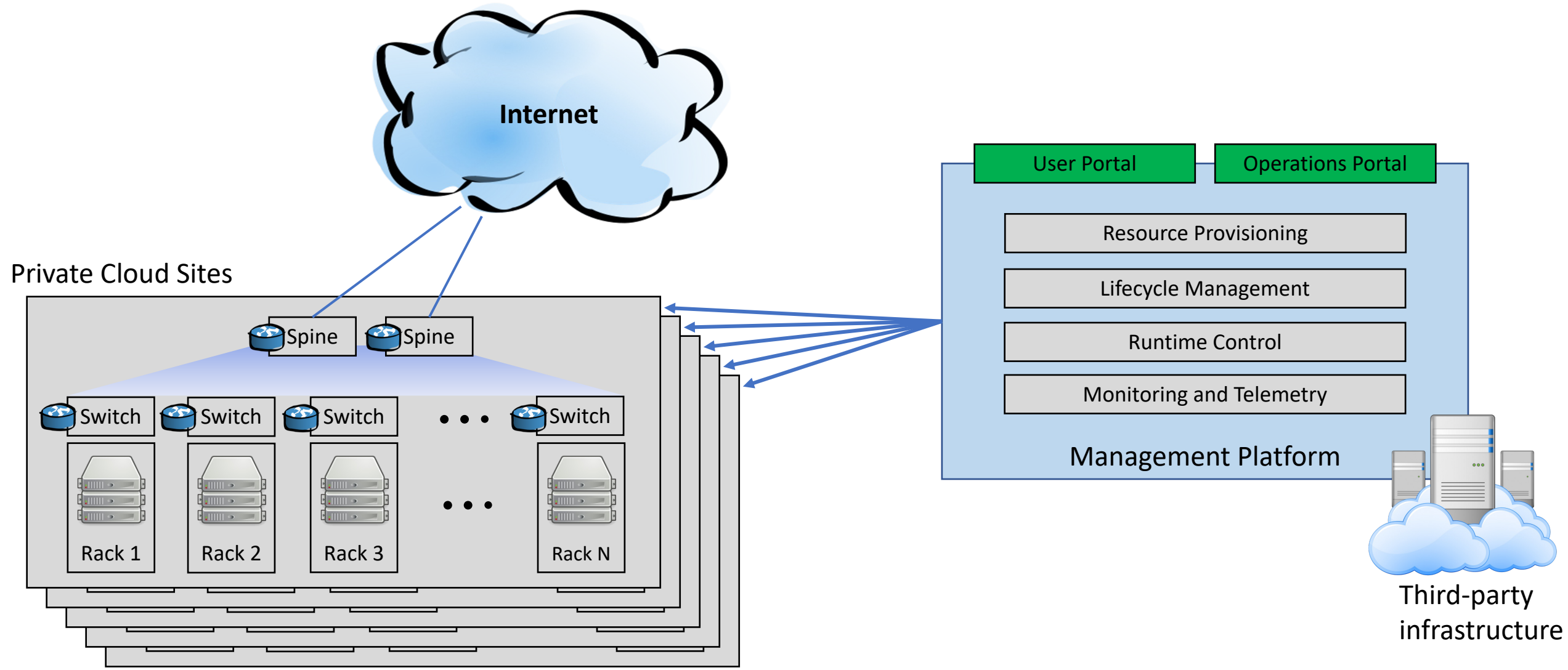


# Control and Management



# Control and Management



# Resource Provisioning

Prof. Ítalo Cunha

# Resource provisioning

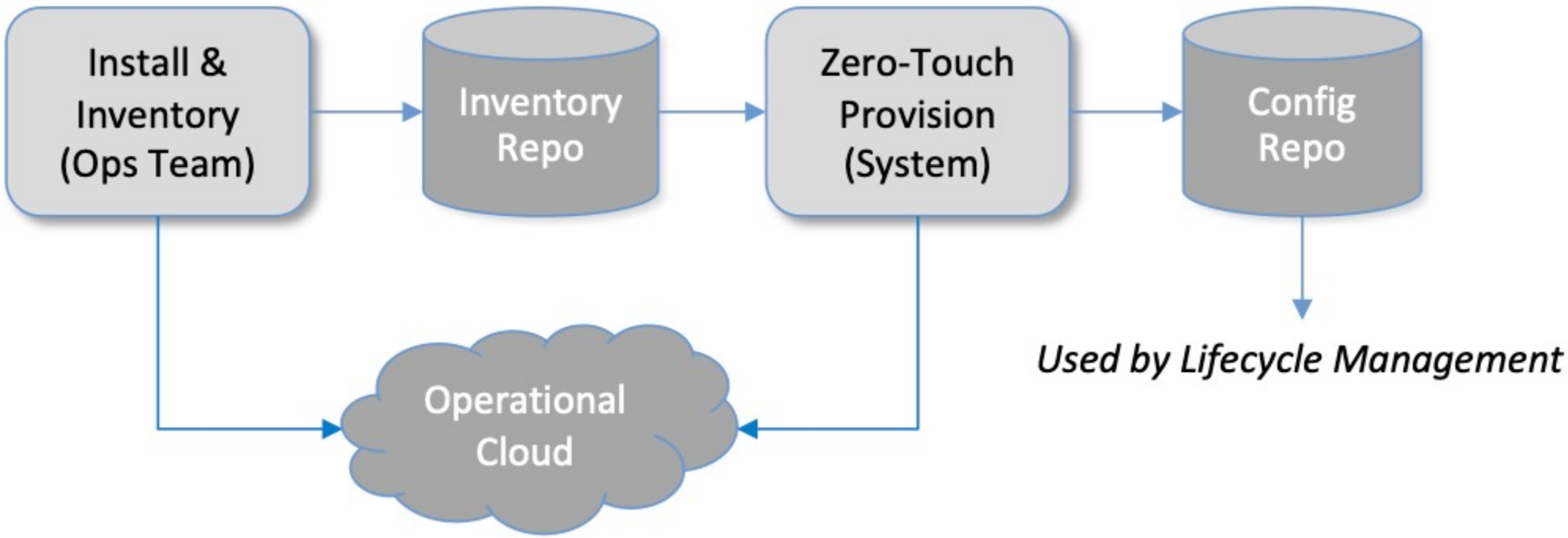
- Bringing virtual and physical resources online
  - Hands-on deployment for physical resources
    - “Racking” and connecting power and network cables
  - Bootstrapping
    - Getting resources into a “ready” state (e.g., reachable over the network)

# Resource provisioning

- Bringing virtual and physical resources online
  - Hands-on deployment for physical resources
    - “Racking” and connecting power and network cables
  - Bootstrapping
    - Getting resources into a “ready” state (e.g., reachable over the network)

- Resource provisioning also happens incrementally over time
  - Upgrades, removal of obsolete resources, deployment of new resources

# Overview of resource provisioning




“Day 0 operations”

# Installation and Inventory

Prof. Ítalo Cunha

# Installation and inventory

- Cannot be entirely zero-touch for physical infrastructures
  - Assume we are dealing with commodity general-purpose resources
  - For virtual infrastructures, cloud provider APIs are used to provision resources
    - “Infrastructure as Code”
  - For plug-and-play appliances, configuration may be preinstalled

# Installation and inventory

- Cannot be entirely zero-touch for physical infrastructures
  - Assume we are dealing with commodity general-purpose resources
  - For virtual infrastructures, cloud provider APIs are used to provision resources
    - “Infrastructure as Code”
  - For plug-and-play appliances, configuration may be preinstalled
- Goal is to minimize the amount of manual handling
  - Focus on getting the device connected and reachable
  - Zero-touch provisioning tools take it from there

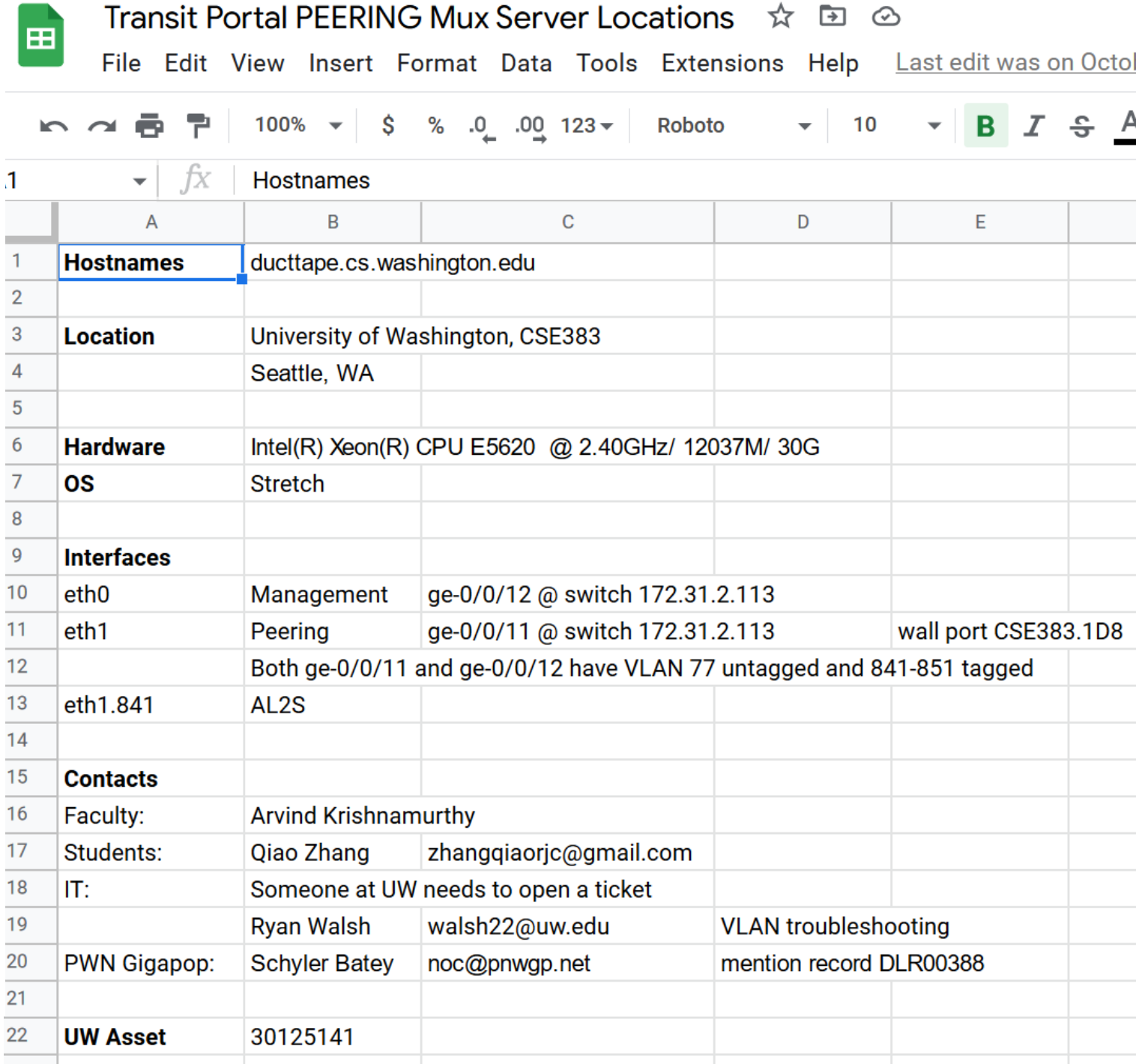
# Documenting the infrastructure

- Managing infrastructure requires ground truth about it
  - Organizations
  - Sites
  - Racks
    - Switches
    - Servers
    - Storage
    - Other equipment
  - Deployment
  - Power and networking

# Documenting the infrastructure

- Managing infrastructure requires ground truth about it
  - Organizations
  - Sites
  - Racks
    - Switches
    - Servers
    - Storage
    - Other equipment
  - Deployment
  - Power and networking

**Device:**
- Rack and rack position
- Manufacturer
- Model
- Serial number
- Device type
- MAC addresses
- Power outlet
- Switch ports and VLANs



Transit Portal PEERING Mux Server Locations

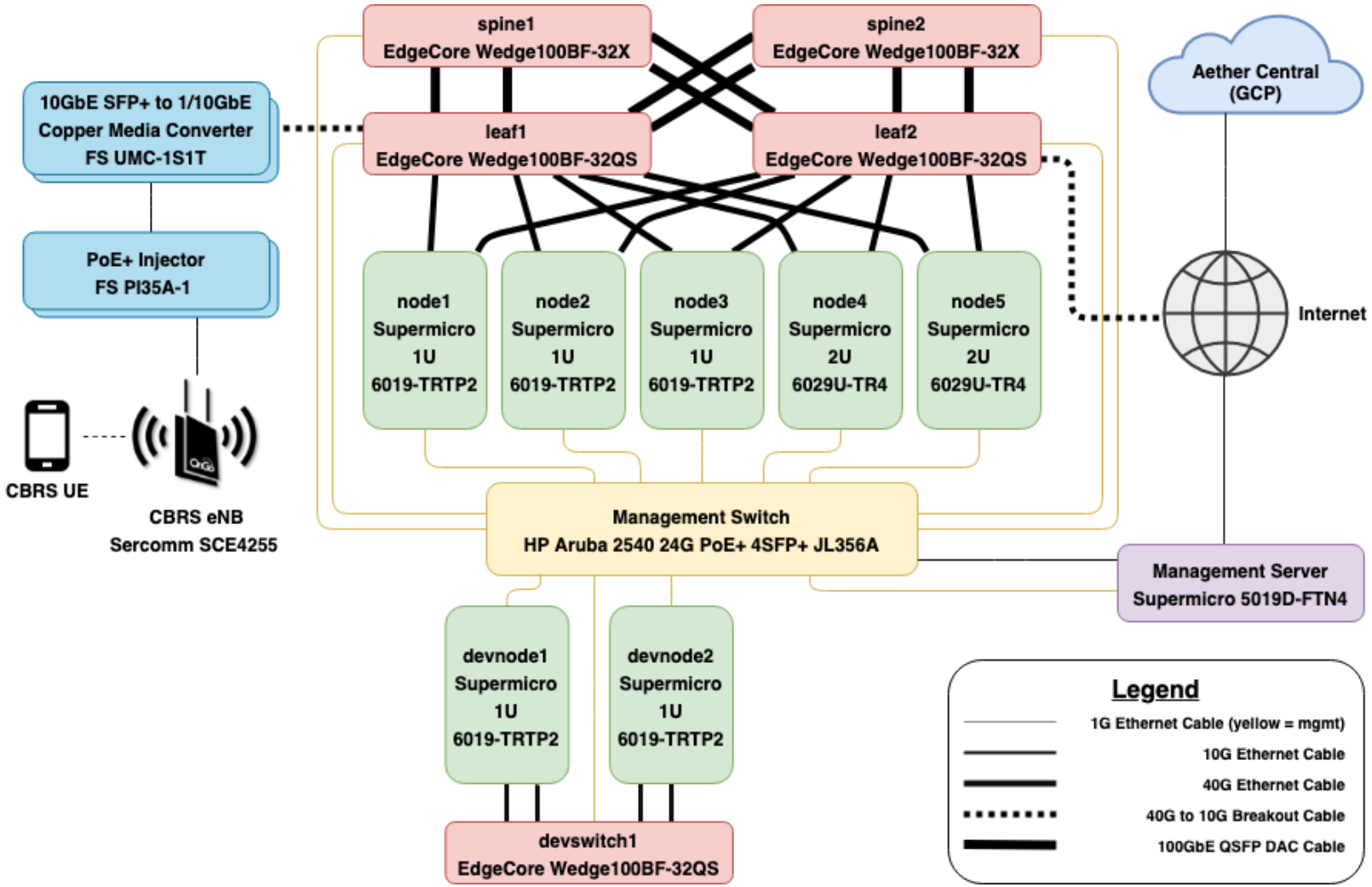
File Edit View Insert Format Data Tools Extensions Help Last edit was on Octol

100% $ % .0 .00 123 Roboto 10 B I S A

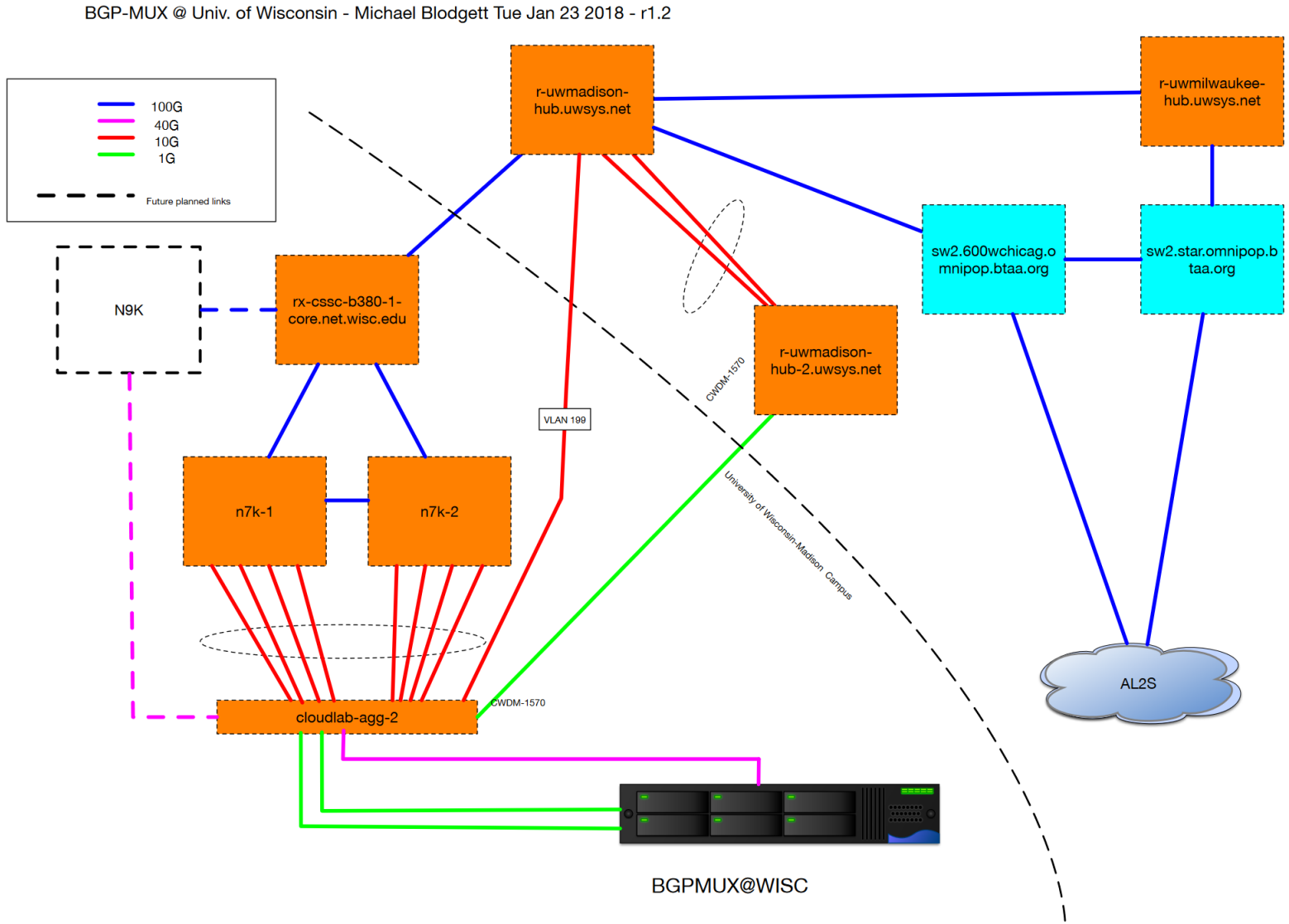
1 fx Hostnames

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | **Hostnames** | ducttape.cs.washington.edu | | | |
| 2 | | | | | |
| 3 | **Location** | University of Washington, CSE383 | | | |
| 4 | | Seattle, WA | | | |
| 5 | | | | | |
| 6 | **Hardware** | Intel(R) Xeon(R) CPU E5620 @ 2.40GHz/ 12037M/ 30G | | | |
| 7 | **OS** | Stretch | | | |
| 8 | | | | | |
| 9 | **Interfaces** | | | | |
| 10 | eth0 | Management | ge-0/0/12 @ switch 172.31.2.113 | | |
| 11 | eth1 | Peering | ge-0/0/11 @ switch 172.31.2.113 | | wall port CSE383.1D8 |
| 12 | | Both ge-0/0/11 and ge-0/0/12 have VLAN 77 untagged and 841-851 tagged | | | |
| 13 | eth1.841 | AL2S | | | |
| 14 | | | | | |
| 15 | **Contacts** | | | | |
| 16 | Faculty: | Arvind Krishnamurthy | | | |
| 17 | Students: | Qiao Zhang | zhangqiaorjc@gmail.com | | |
| 18 | IT: | Someone at UW needs to open a ticket | | | |
| 19 | | Ryan Walsh | walsh22@uw.edu | VLAN troubleshooting | |
| 20 | PWN Gigapop: | Schyler Batey | noc@pnwgp.net | mention record DLR00388 | |
| 21 | | | | | |
| 22 | **UW Asset** | 30125141 | | | |

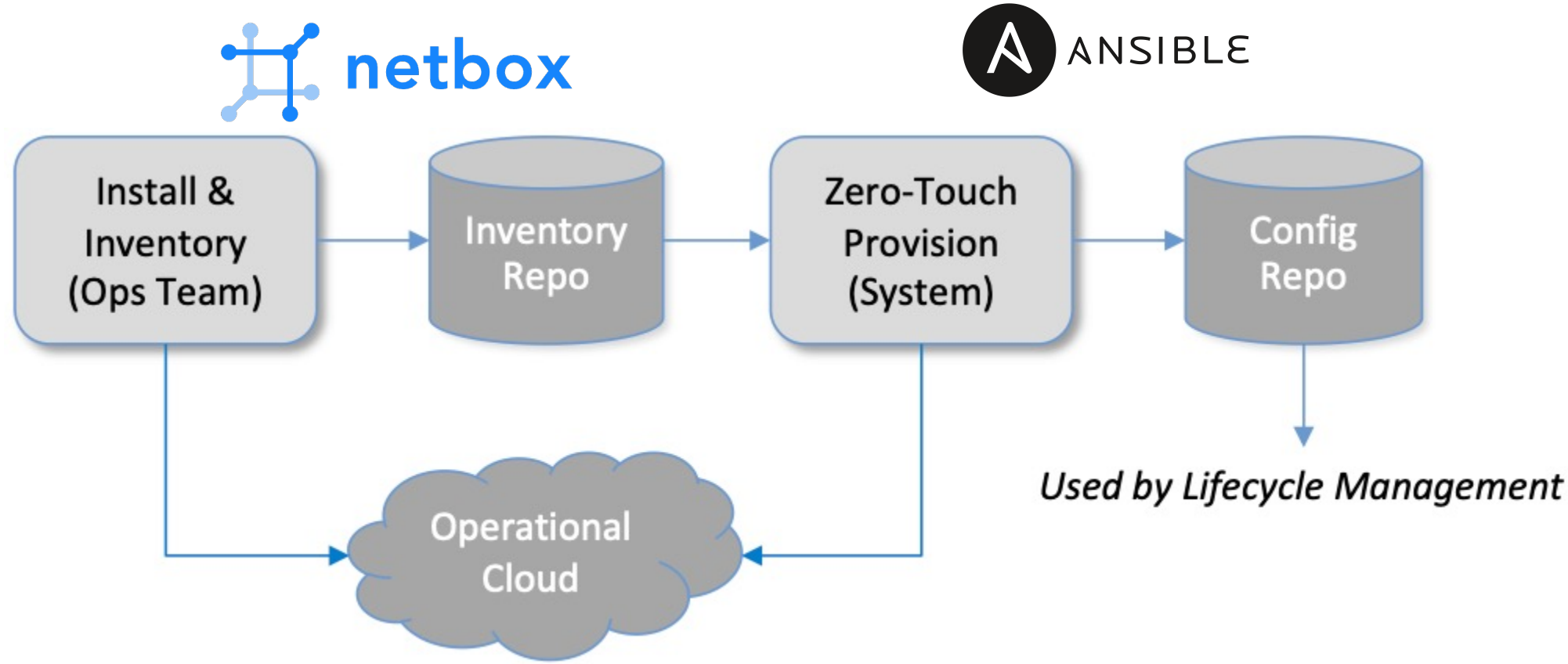
# Infrastructure planning



# Infrastructure documentation (after the fact)



# Overview of resource provisioning



# Example: Cabling in NetBox

## Cables

| ID | Label | Side A | Termination A | Side B | Termination B | Status | Type | Length | Color |
|---|---|---|---|---|---|---|---|---|---|
| 165 | — | mgmtswitch1.prod1.stanford1 | gbe3 | node1.prod1.stanford1 | bmc | Connected | CAT6 | 2 Feet | |
| 166 | — | mgmtswitch1.prod1.stanford1 | gbe4 | node2.prod1.stanford1 | bmc | Connected | CAT6 | 2 Feet | |
| 167 | — | mgmtswitch1.prod1.stanford1 | gbe5 | node3.prod1.stanford1 | bmc | Connected | CAT6 | 2 Feet | |
| 168 | — | mgmtswitch1.prod1.stanford1 | gbe6 | node4.prod1.stanford1 | bmc | Connected | CAT6 | 3 Feet | |
| 169 | — | mgmtswitch1.prod1.stanford1 | gbe7 | node5.prod1.stanford1 | bmc | Connected | CAT6 | 3 Feet | |
| 170 | — | mgmtswitch1.prod1.stanford1 | gbe11 | spine1.prod1.stanford1 | eth0 | Connected | CAT6 | 5 Feet | |
| 171 | — | mgmtswitch1.prod1.stanford1 | gbe12 | spine2.prod1.stanford1 | eth0 | Connected | CAT6 | 5 Feet | |
| 172 | — | mgmtswitch1.prod1.stanford1 | gbe13 | leaf1.prod1.stanford1 | eth0 | Connected | CAT6 | 5 Feet | |
| 173 | — | mgmtswitch1.prod1.stanford1 | gbe14 | leaf2.prod1.stanford1 | eth0 | Connected | CAT6 | 5 Feet | |
| 174 | — | mgmtswitch1.prod1.stanford1 | gbe15 | node1.prod1.stanford1 | gbe0 | Connected | CAT6 | 2 Feet | |
| 175 | — | mgmtswitch1.prod1.stanford1 | gbe16 | node2.prod1.stanford1 | gbe0 | Connected | CAT6 | 2 Feet | |
| 176 | — | mgmtswitch1.prod1.stanford1 | gbe17 | node3.prod1.stanford1 | gbe0 | Connected | CAT6 | 2 Feet | |
| 177 | — | mgmtswitch1.prod1.stanford1 | gbe18 | node4.prod1.stanford1 | gbe0 | Connected | CAT6 | 3 Feet | |
| 178 | — | mgmtswitch1.prod1.stanford1 | gbe19 | node5.prod1.stanford1 | gbe0 | Connected | CAT6 | 3 Feet | |

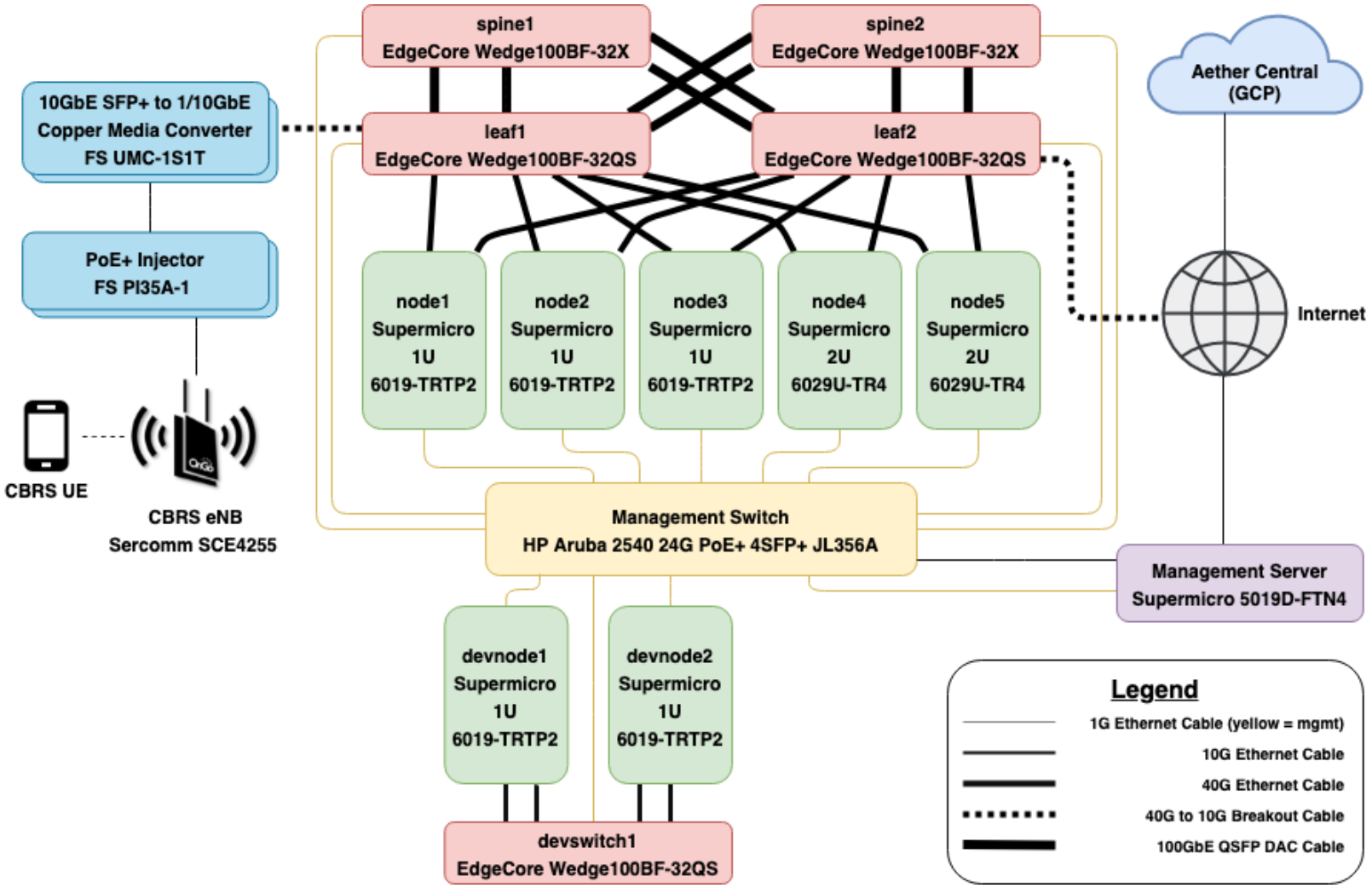
# Bootstrapping

Prof. Ítalo Cunha

# Bootstrapping → From bricks to “ready”

- Goal: Minimize amount of manual configuration
  - Want to get devices manageable over the network
- Cannot avoid manual configuration

# Bootstrapping



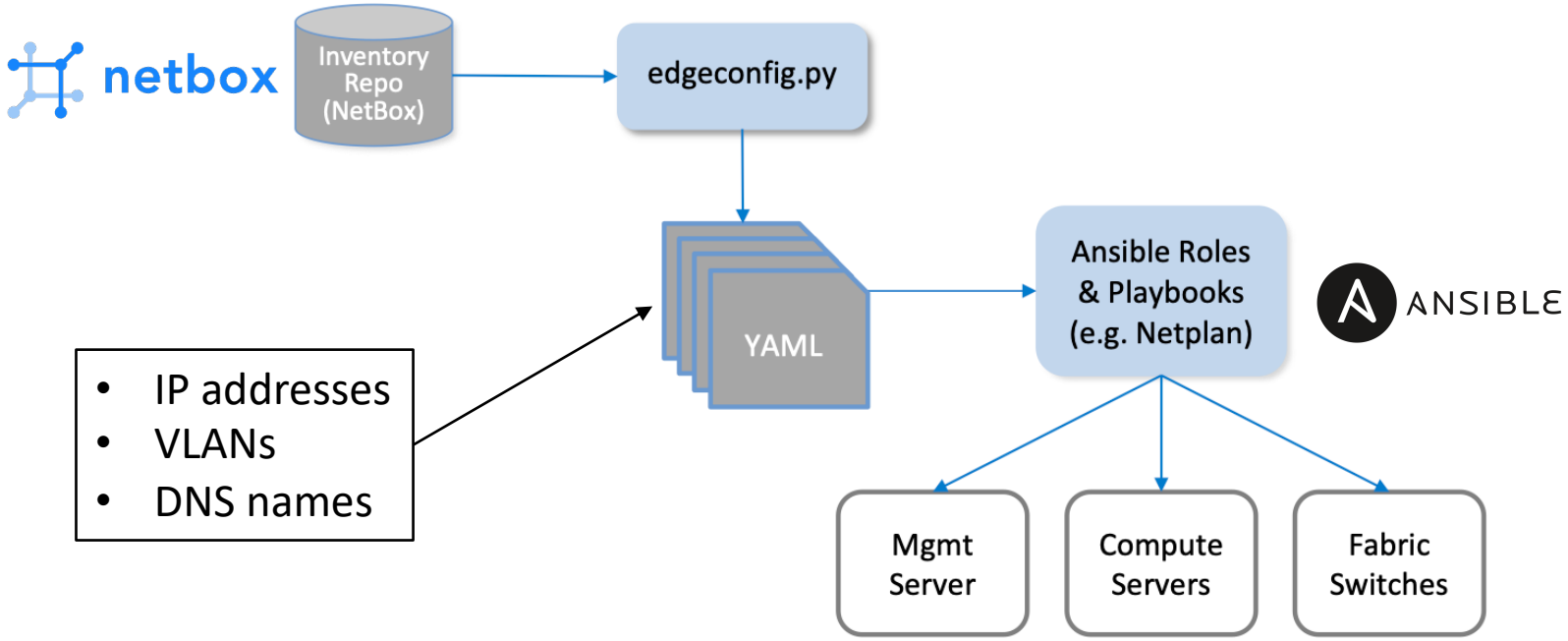

- Configure VLANs on management switch
- Configure management server needs to boot from USB stick
- Configure management services on management server using Ansible
- Configure Compute Servers to boot from the mgmt server via iPXE
- Configure fabric switches to boot from the mgmt server via Nginx

# Bootstrapping → From bricks to “ready”

- Goal: Minimize amount of manual configuration
  - Want to get devices manageable over the network
- Cannot avoid manual configuration
  - But can still try to minimize work
  - Preconfigure as much as possible locally before shipping devices to the field
  - Use DHCP with MAC addresses to avoid configuring interfaces on hosts

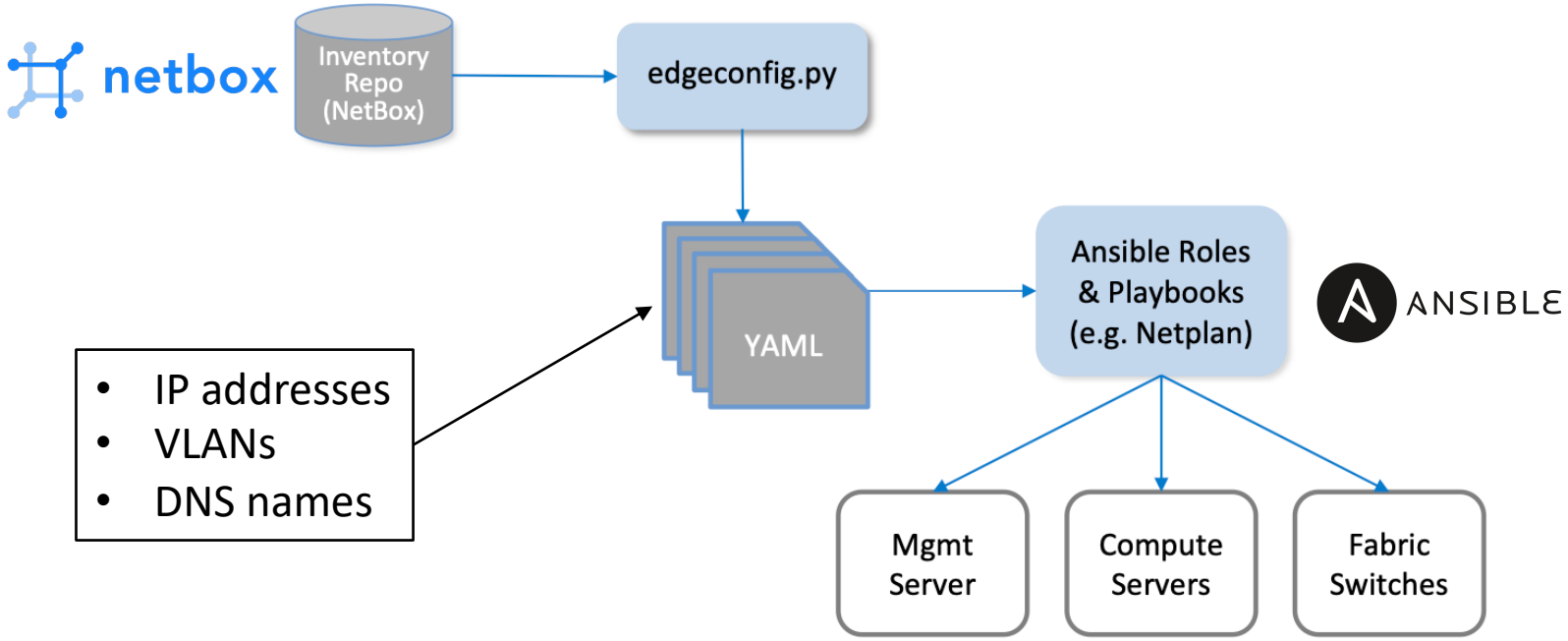
# Automated configuration

- Automate configuration as soon as machines are reachable
  - Ansible, Puppet, Chef
- Generate inputs to configuration system from inventory



# Automated configuration

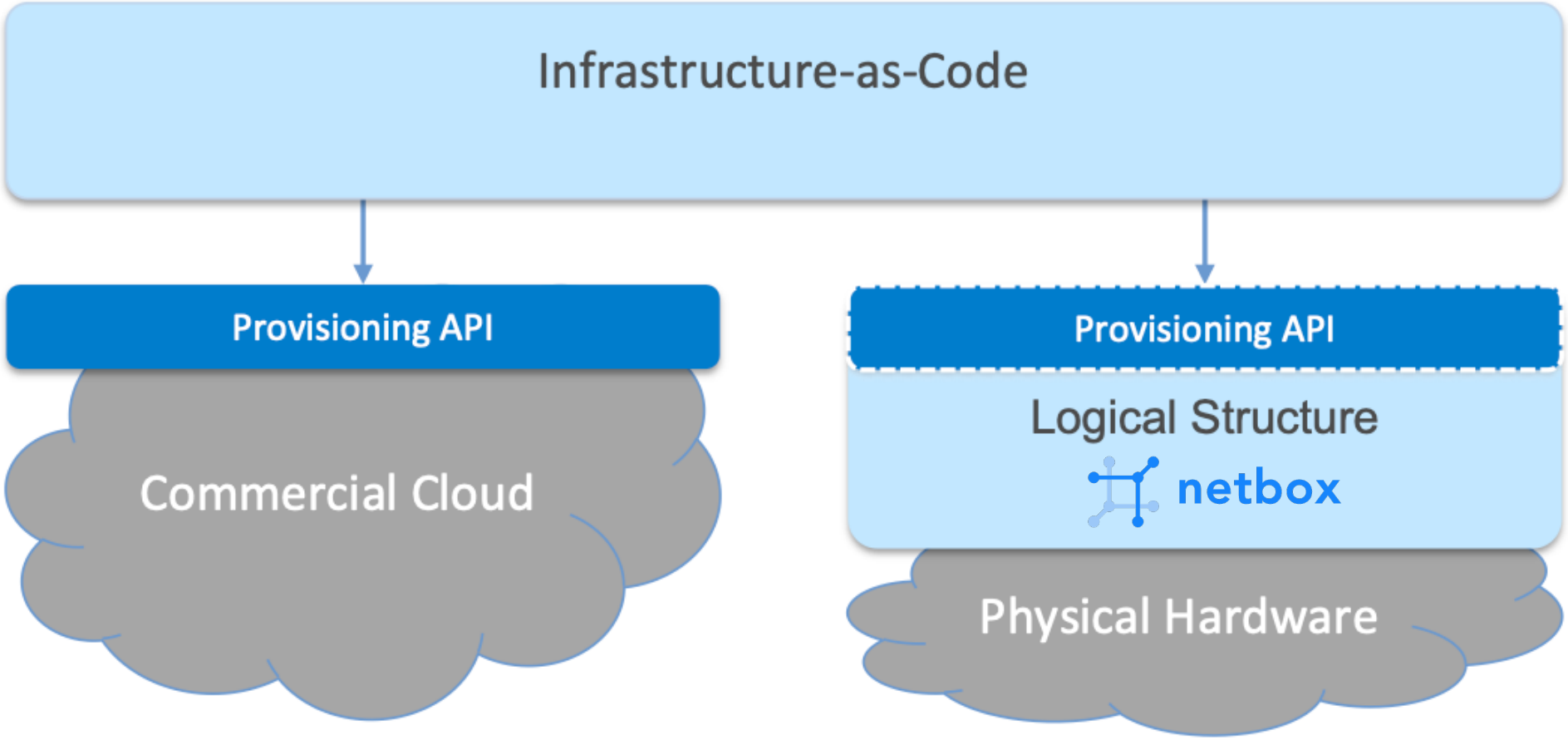
- Automate configuration as soon as machines are reachable
  - Ansible, Puppet, Chef
- Generate inputs to configuration system from inventory



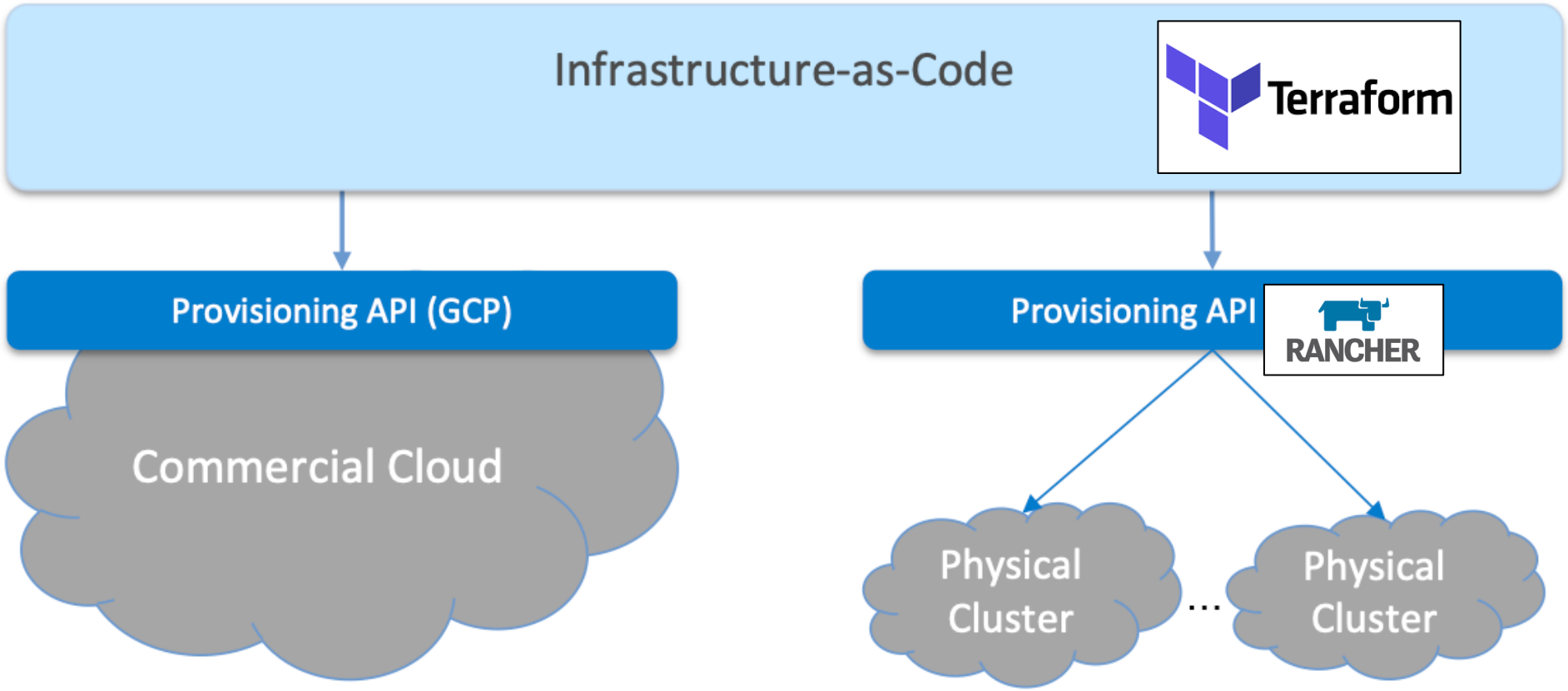
# Provisioning

- After servers are configured, we need to provision the cluster
  - Kubernetes deployment
  - OpenStack deployment
- Automation during the provisioning phase
  - Require a provisioning API
    - Like the ones provided by cloud providers

# Provisioning APIs



# Provisioning APIs



# Provisioning in the public vs private clouds

**Public clouds**

- On-demand capacity
- Untrusted tenants and apps

**Private clouds**

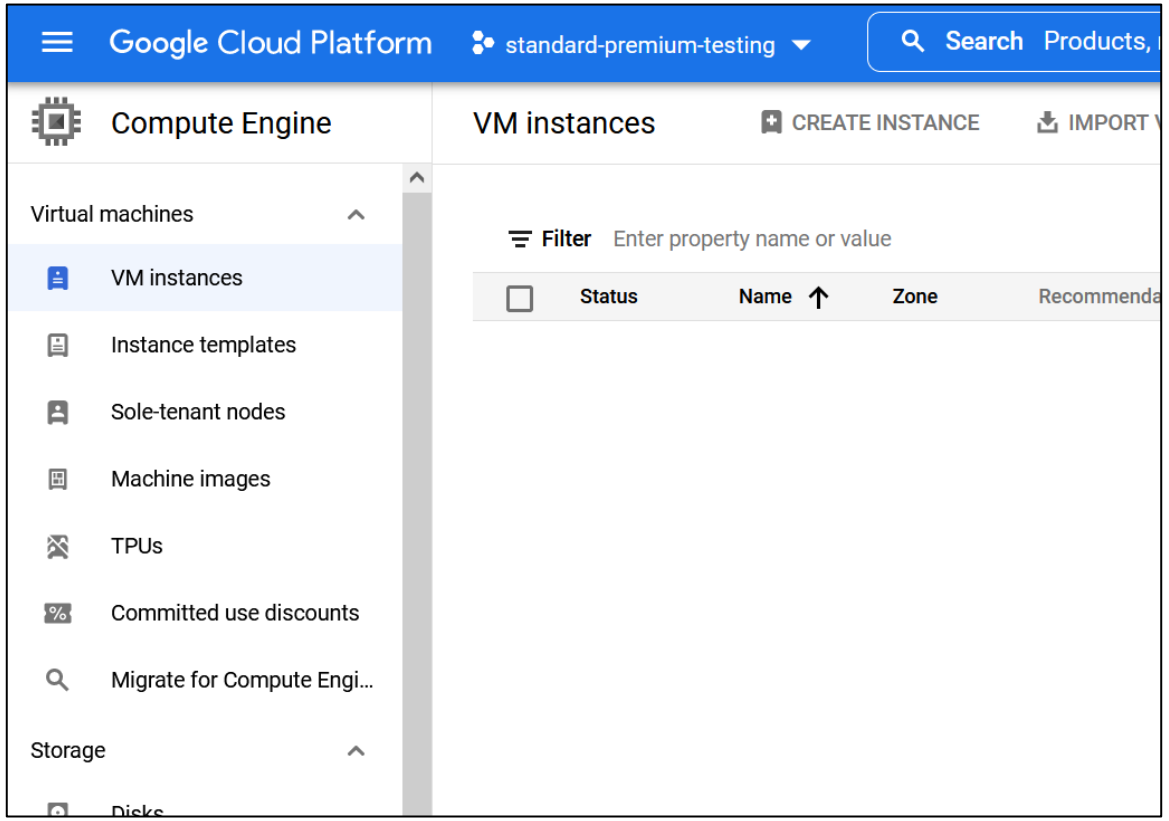
- Planned capacity
- Trusted tenants and applications

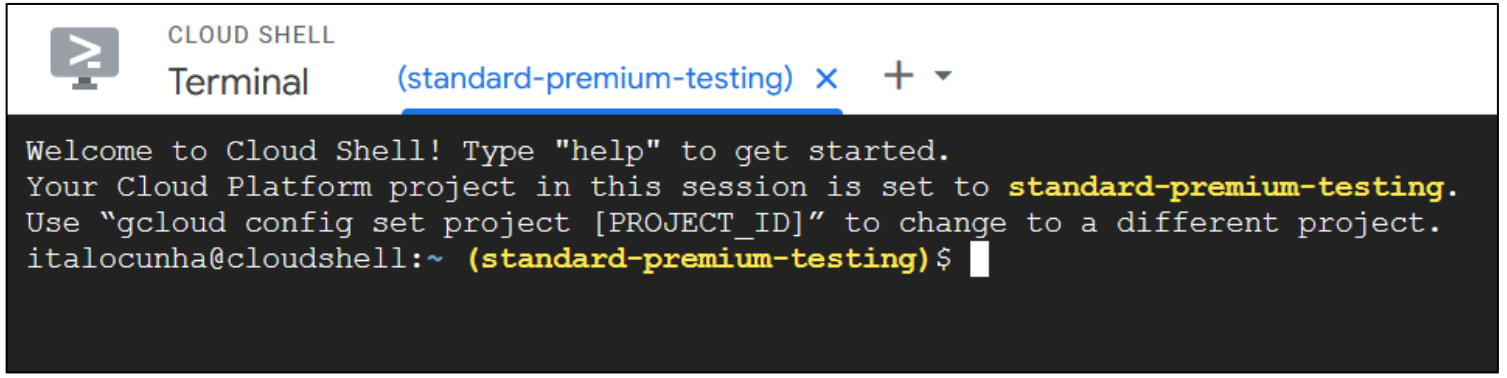
# Infrastructure as Code

Prof. Ítalo Cunha

# Provisioning

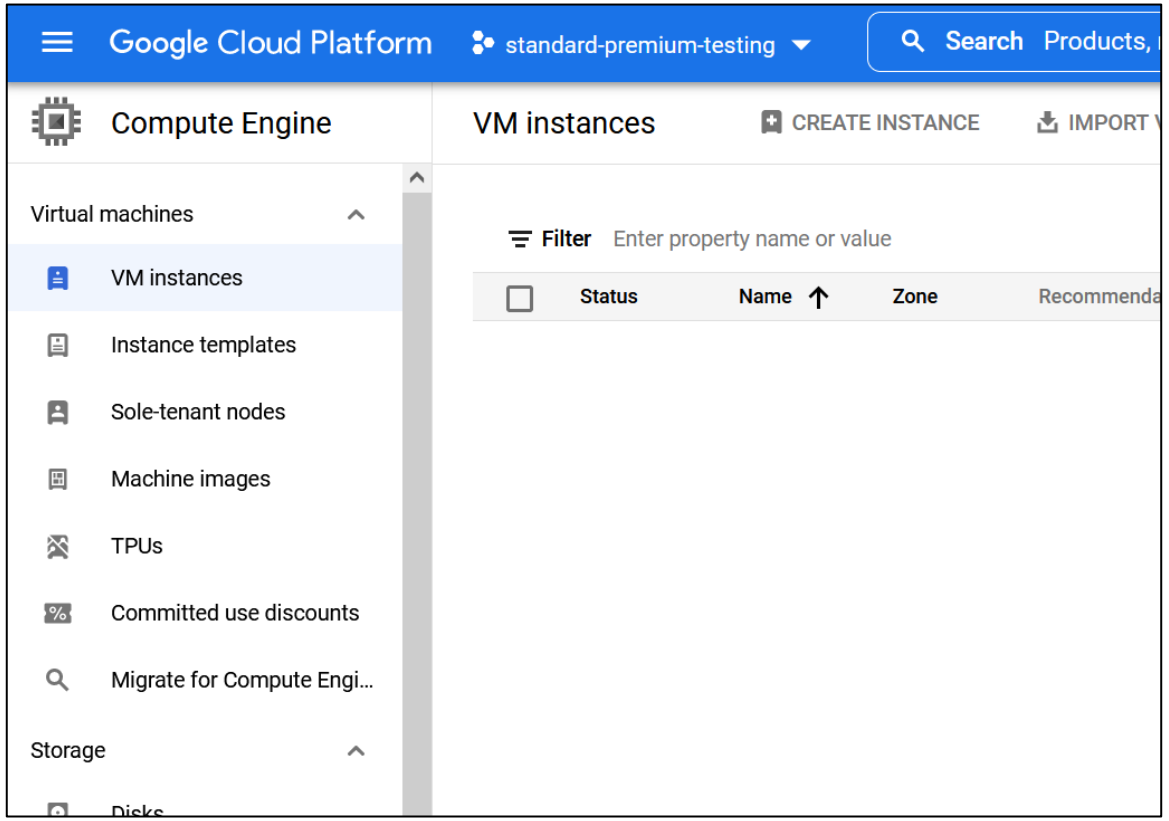
- Command-line interface
- Graphical user interface
- Programmatic API



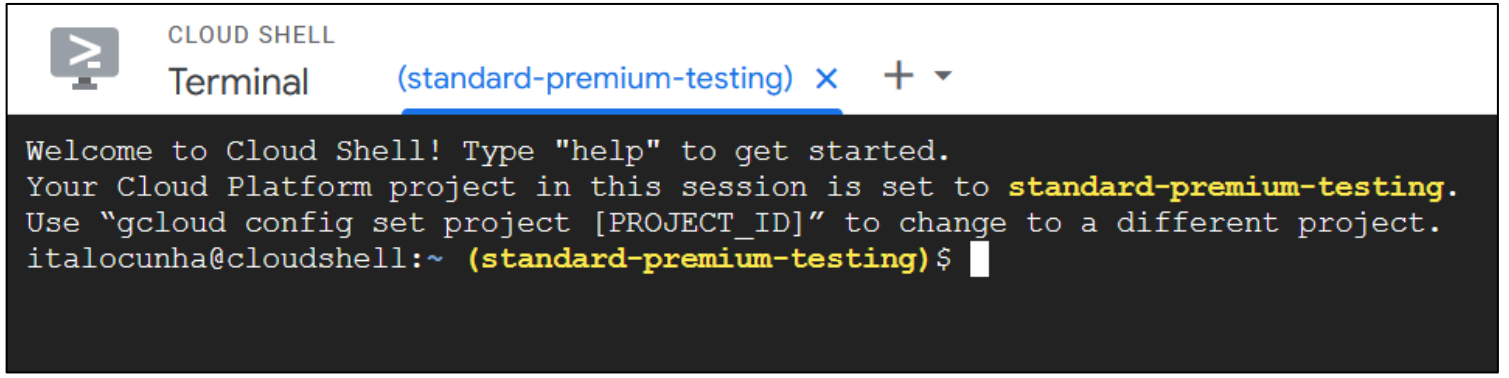



# Provisioning

- Command-line interface
- Graphical user interface
- Programmatic API




Human interaction with a CLI or GUI is time-consuming and error prone.



# Declarative interface to provisioning

- Specify provisioning in a declarative language
  - What Kubernetes clusters should be instantiated
  - On which resources
  - And specific configuration
- Automate calls to programmatic provisioning API

# Declarative interface to provisioning


- Specify provisioning in a declarative language
  - What Kubernetes clusters should be instantiated
  - On which resources
  - And specific configuration
- Automate calls to programmatic provisioning API

# Terraform

- Declare infrastructure providers  (GCP, Rancher)
- For each provider
  - Declare resources and their baseline configuration
    - IP addresses, VLANs, access keys, roles
  - Declare Kubernetes configuration
    - Controller nodes, worker nodes, CNI plugin (container network interface), proxies

# Terraform

- Declare infrastructure providers  (GCP, Rancher)
- For each provider
  - Declare resources and their baseline configuration
    - IP addresses, VLANs, access keys, roles
  - Declare Kubernetes configuration
    - Controller nodes, worker nodes, CNI plugin (container network interface), proxies

- Files containing these declarative commands are checked into code repositories → Infrastructure as Code

# Terraform

- Declare infrastructure providers  (GCP, Rancher)
- For each provider
  - Declare resources and their baseline configuration
    - IP addresses, VLANs, access keys, roles
  - Declare Kubernetes configuration
    - Controller nodes, worker nodes, CNI plugin (container network interface), proxies

- Files containing these declarative commands are checked into code repositories → Infrastructure as Code
  - Changes to configuration trigger execution of the provisioner

# Platform Definition

Prof. Ítalo Cunha

# What is a platform?

Applications


Physical Resources 




# What is a platform?

- What about providing additional functionality for applications?

Applications


Physical Resources 




# What is a platform?

- What about providing additional functionality for applications?

Applications

OpenVINO™ APACHE Spark™ kafka Vitess

kubernetes

Physical Resources 

Terraform RANCHER

netbox ANSIBLE