Open Computing Facility

Config Management (with Puppet)

Lecture 11

jaysa

(slide credits fydai)

Congratulations, you’re hired!

You’ve earned your Open Computing Facility Full-Fledged Linux Expert Certification™*, and now you have been hired to manage a computer lab…

*sorry this is not real :'(

Problem 1: Managing a fleet

Suppose you have a bunch of computers… and you decide that everybody computer in the lab needs Minecraft installed.

- Manually install on each desktop?
- Write a script to SSH into each desktop and install?

How do I deploy updates to a fleet of existing computers?

Problem 2: Provisioning new machines

Suppose your computer lab buys a new computer… and you want to set it up the same as the others

- Remembering to install each software and configure it correctly is pretty difficult!
  - Eg: Have Minecraft use more than 1Gb RAM by default

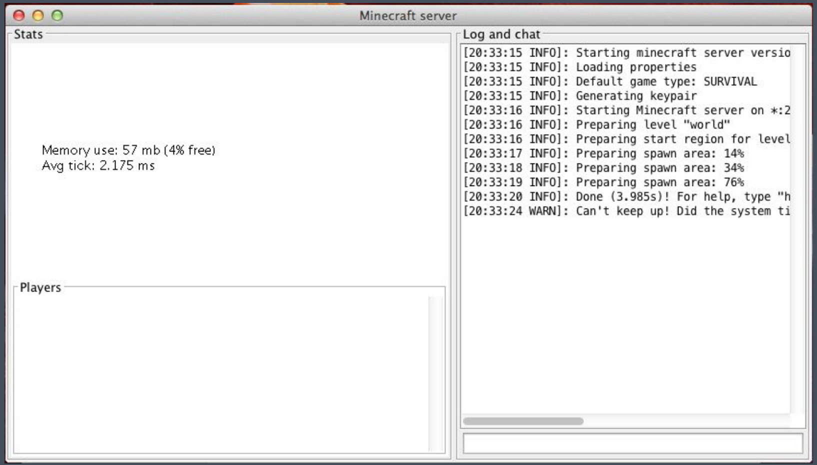
How do I provision new machines?

Problem 3: Communicating changes

Suppose you are running a Minecraft server... and you realize some change you made long ago broke a thing.

- How do you figure out what settings you had changed around that time?

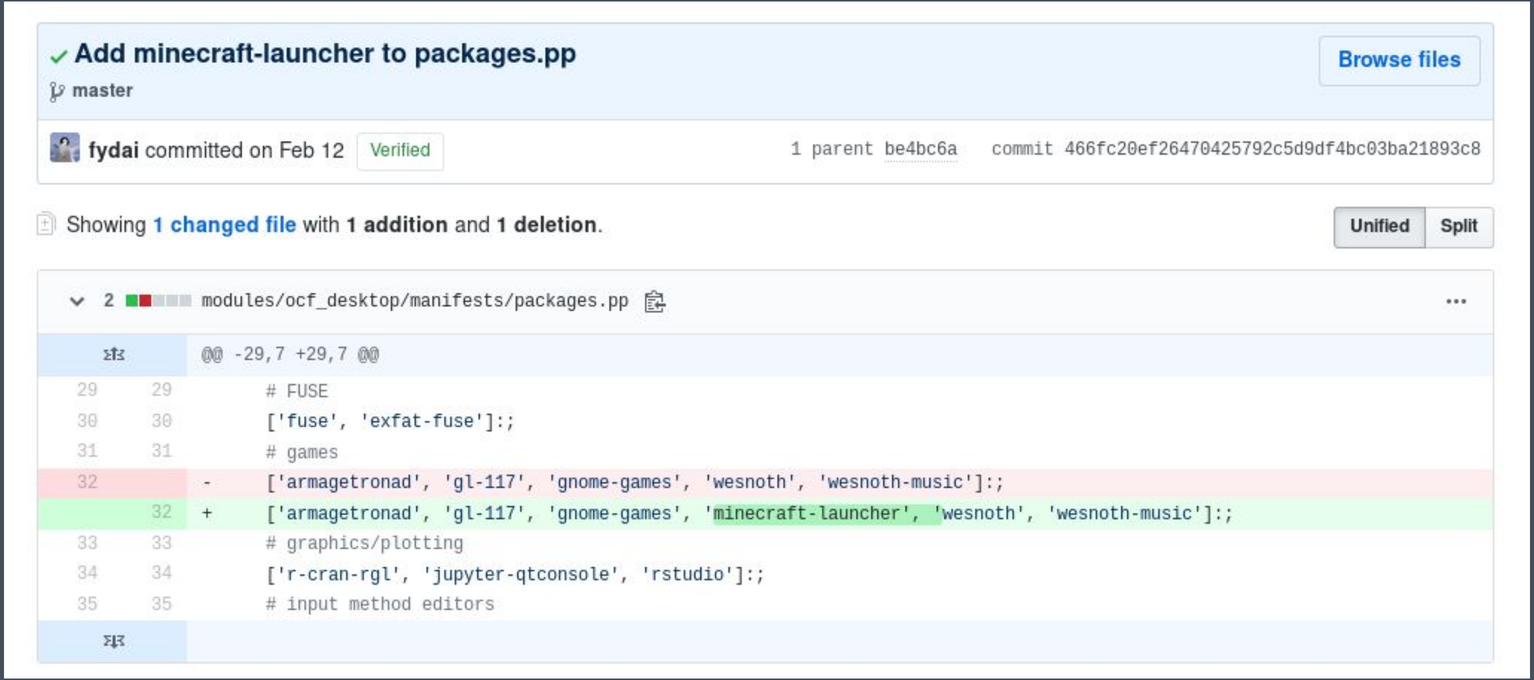
How do you communicate changes in software configuration to future you (and others)?



```
Memory use: 57 mb (4% free)
Avg tick: 2.175 ms
```

```
[20:33:15 INFO]: Starting minecraft server versio
[20:33:15 INFO]: Loading properties
[20:33:15 INFO]: Default game type: SURVIVAL
[20:33:15 INFO]: Generating keypair
[20:33:16 INFO]: Starting Minecraft server on *:2
[20:33:16 INFO]: Preparing level "world"
[20:33:16 INFO]: Preparing start region for level
[20:33:17 INFO]: Preparing spawn area: 14%
[20:33:18 INFO]: Preparing spawn area: 34%
[20:33:19 INFO]: Preparing spawn area: 76%
[20:33:20 INFO]: Done (3.985s)! For help, type "h
[20:33:24 WARN]: Can't keep up! Did the system ti
```

Config Management!



Add minecraft-launcher to packages.pp
master

Browse files

fydai committed on Feb 12 Verified

1 parent be4bc6a commit 466fc20ef26470425792c5d9df4bc03ba21893c8

Showing **1 changed file** with **1 addition** and **1 deletion**.

Unified Split

2 modules/ocf_desktop/manifests/packages.pp

```
@@ -29,7 +29,7 @@
      # FUSE
      ['fuse', 'exfat-fuse']:;
      # games
-     ['armagetronad', 'gl-117', 'gnome-games', 'wesnoth', 'wesnoth-music']:;
+     ['armagetronad', 'gl-117', 'gnome-games', 'minecraft-launcher', 'wesnoth', 'wesnoth-music']:;
      # graphics/plotting
      ['r-cran-rgl', 'jupyter-qtconsole', 'rstudio']:;
      # input method editors
```

Config Management

Config management software makes it as easy as possible to bootstrap new machines, configure running software, and allows configuration to be stored as code “configuration as code” philosophy

- Solves problem 1 (updating computers) by having an unified update mechanism
- Solves problem 2 (bootstrapping new computers) by having all the changes necessary in some centralized repository
- Solves problem 3 (communication) by allowing you to use standard development practices (mainly git) to record your changes, and communicate with others

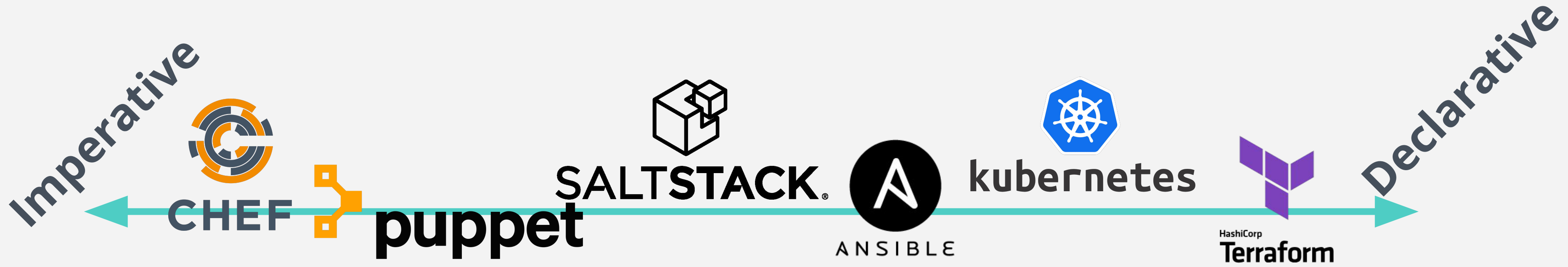
Configuration Management Philosophies

Imperative

- Write the **set of tasks**"
- “Install minecraft”, then “add a line to the config file”, then “run minecraft”

Declarative

- Write the **desired state**
- “Ensure minecraft is installed, the config file has line <X> in it, and ensure that minecraft is running”

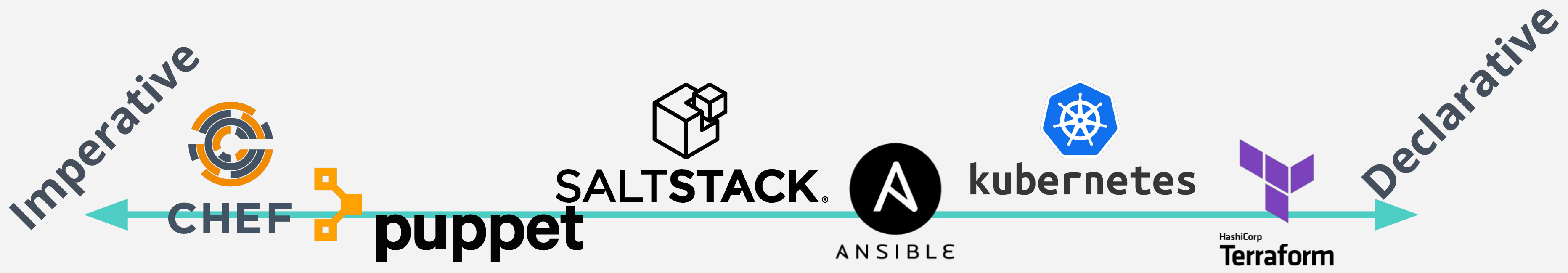


Configuration Management Software

from Stack Overflow Developer Survey 2023:

% of respondents who use:

- Kubernetes - 19.02%
- Terraform - 11.3%
- Ansible - 8.62%
- Puppet - 1.12%
- Chef - 0.94%



Let’s learn

puppet !

Puppet

- Released 2005; has open source and enterprise closed source version
- Declarative philosophy, with some Imperative components when necessary
- Originally built on Ruby, now its own configuration language
- “**Pull model**” - Configured machines ask for an update
  - At OCF, puppet is scheduled to run every 30 minutes

Used at places like

- OCF (github.com/ocf/puppet)
- CS 162 (github.com/Berkeley-CS162/vagrant/tree/master/modules/cs162)
- Wikimedia (github.com/wikimedia/puppet)
- Github
- Lyft

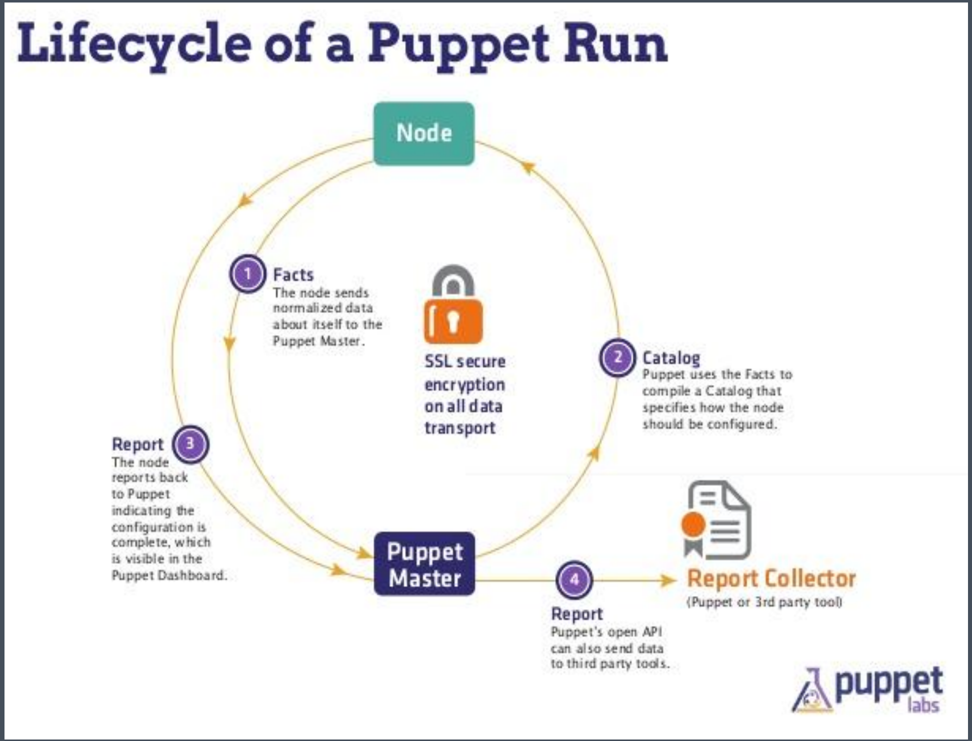
What happens when Puppet is run?

1. Client asks server for an update
2. Server asks client for a list of Facts
3. Client responds with the facts
4. Server responds with configuration
5. Client makes the necessary changes to ensure its current configuration matches the configuration given by the server

1. “I want to be configured as a Minecraft Server”
2. “Ok, send me your hostname, and RAM”
3. “My hostname is zombies.ocf.berkeley.edu and I have 4GB RAM”
4. “Ensure the Minecraft server is running, with hostname zombies.ocf.berkeley.edu, 4GB RAM, with this configuration file
5. “The minecraft server is currently running, but the configuration file has been updated, I will fetch the updated version”

What happens when Puppet is run?

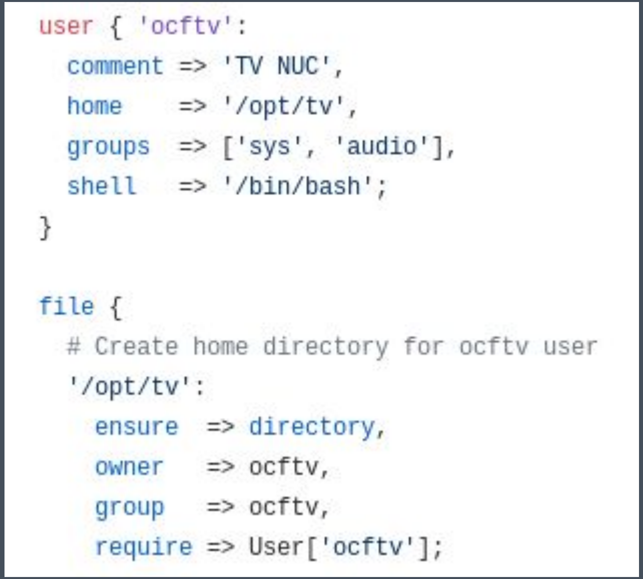
1. Client = Node
2. Server = Puppet Master


Lifecycle of a Puppet Run
Node
1 Facts
The node sends normalized data about itself to the Puppet Master.
SSL secure encryption on all data transport
2 Catalog
Puppet uses the Facts to compile a Catalog that specifies how the node should be configured.
3 Report
The node reports back to Puppet indicating the configuration is complete, which is visible in the Puppet Dashboard.
Puppet Master
4 Report
Puppet's open API can also send data to third party tools.
Report Collector
(Puppet or 3rd party tool)
puppet labs

Puppet Code

- Mostly, we have:
  - **Files** - contains static files
  - **Templates** - contain templates (Ruby style)
  - **Manifests** - the heart of the configuration, specifies the desired states
- Other sections that are occasionally used
  - **Facts** - Ways to extract data needed for configuration
  - **Functions** - if you need extra something fancy data structure manipulating
- Dependencies need to be explicitly described
  - Puppet is allowed to run code in any order that it sees fit
  - If you have code installing Minecraft, and running Minecraft, you need to tell puppet to install Minecraft before running it

Example Puppet Code 1

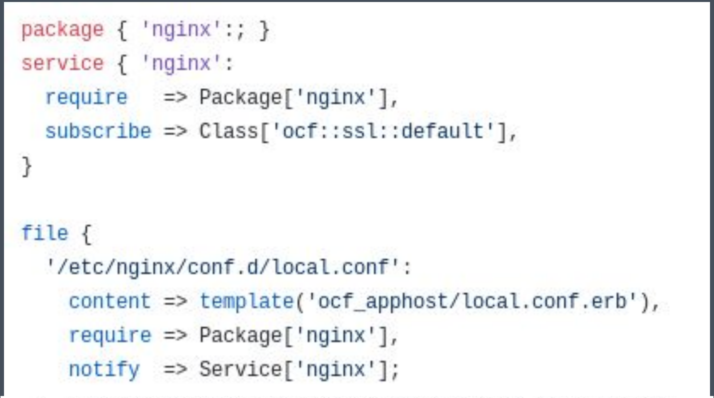


```
user { 'ocftv':
  comment => 'TV NUC',
  home    => '/opt/tv',
  groups  => ['sys', 'audio'],
  shell   => '/bin/bash';
}

file {
  # Create home directory for ocftv user
  '/opt/tv':
    ensure  => directory,
    owner   => ocftv,
    group   => ocftv,
    require => User['ocftv'];
```

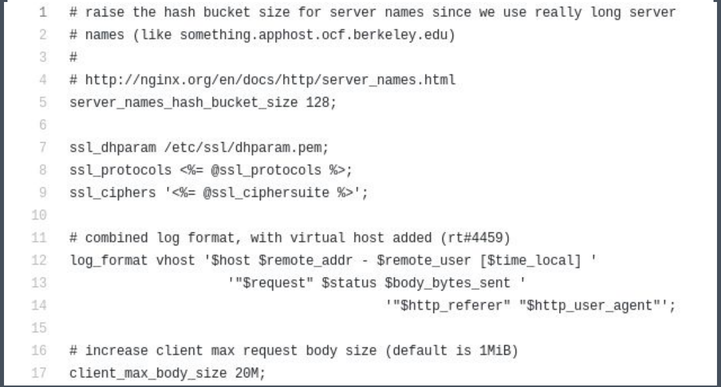
What does this do?

A: Adds a user (ocftv) and a home directory (/opt/tv)

Example Puppet Code 2



```
package { 'nginx':; }
service { 'nginx':
  require   => Package['nginx'],
  subscribe => Class['ocf::ssl::default'],
}

file {
  '/etc/nginx/conf.d/local.conf':
    content => template('ocf_apphost/local.conf.erb'),
    require => Package['nginx'],
    notify  => Service['nginx'];
```



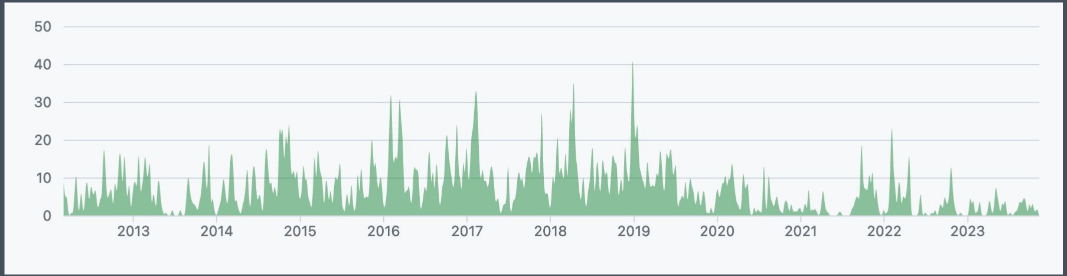
```
# raise the hash bucket size for server names since we use really long server
# names (like something.apphost.ocf.berkeley.edu)
#
# http://nginx.org/en/docs/http/server_names.html
server_names_hash_bucket_size 128;

ssl_dhparam /etc/ssl/dhparam.pem;
ssl_protocols <%= @ssl_protocols %>;
ssl_ciphers '<%= @ssl_ciphersuite %>';

# combined log format, with virtual host added (rt#4459)
log_format vhost '$host $remote_addr - $remote_user [$time_local] '
                    '"$request" $status $body_bytes_sent '
                                        '"$http_referer" "$http_user_agent"';

# increase client max request body size (default is 1MiB)
client_max_body_size 20M;
```

What does this do?

A: Runs a web server!

Puppet at the OCF (1/2)

github.com/ocf/puppet



- Originally started in 2012, from the “Configuration is edited directly on the server, and desktops manually” model
  - We only had 10 desktops so this was kinda okay
- 11 years later, all of the OCF’s machines* runs off the puppet repository
  - Desktops
  - Thing behind the TV
  - Hypervisors (things running the VMs)
  - VMs (Running all the Networked Services you learned about)
    - Including the puppet server itself

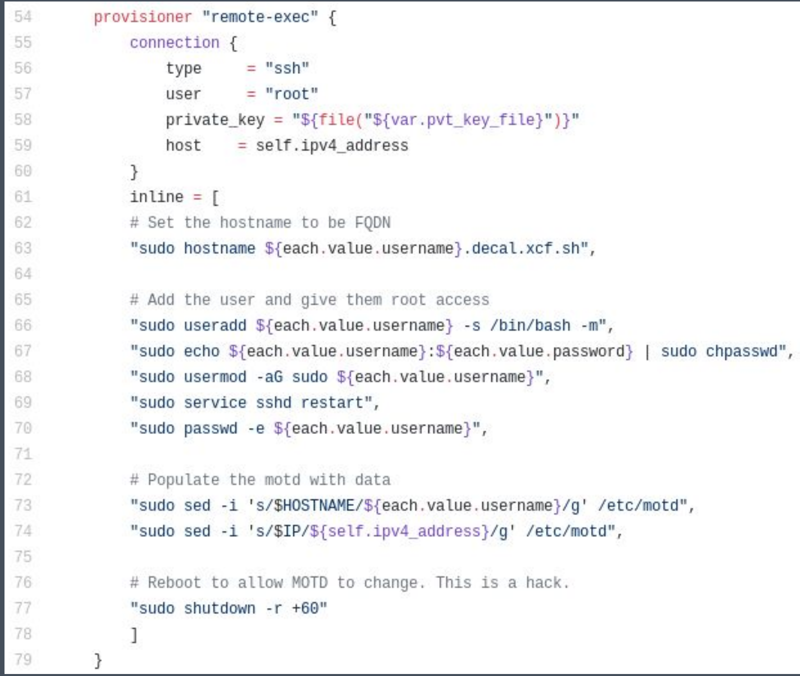
* actually in 2024 we moved a bunch of things to nix and plan to move the rest by 2025

Puppet at the OCF (2/2)

- All the code is split into modules
  - ocf_tv
  - ocf_desktop
  - ocf_www
  - ocf_printhost
- Common OCF modules for shared configuration
  - ocf::ssl for (I need a web certificate)
  - ocf::auth for LDAP and Kerberos and sudoers configuration

Bonus Slide: **Terraform**

- Has integrated APIs to provision machines declaratively on cloud platforms
- This is the code used to generate your decal VMs!
- The alternative (which we seriously considered) was clicking “New droplet” 80 times.
- If you’re interested in seeing more of the decal VM Terraform code, here’s the repository: github.com/0xcf/decal-automation



```
    provisioner "remote-exec" {
        connection {
            type     = "ssh"
            user     = "root"
            private_key = "${file("${var.pvt_key_file}")}"
            host    = self.ipv4_address
        }
        inline = [
        # Set the hostname to be FQDN
        "sudo hostname ${each.value.username}.decal.xcf.sh",

        # Add the user and give them root access
        "sudo useradd ${each.value.username} -s /bin/bash -m",
        "sudo echo ${each.value.username}:${each.value.password} | sudo chpasswd",
        "sudo usermod -aG sudo ${each.value.username}",
        "sudo service sshd restart",
        "sudo passwd -e ${each.value.username}",

        # Populate the motd with data
        "sudo sed -i 's/$HOSTNAME/${each.value.username}/g' /etc/motd",
        "sudo sed -i 's/$IP/${self.ipv4_address}/g' /etc/motd",

        # Reboot to allow MOTD to change. This is a hack.
        "sudo shutdown -r +60"
        ]
    }
```