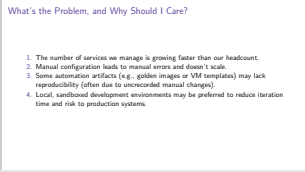
Master of Puppets

Infrastructure as Code and Configuration Management on an Open Source Budget

Mike Renfro (renfro@tntech.edu)

Tennessee Tech University

2023-04-24


What's the Problem, and Why Should I Care?
1. The number of services we manage is growing faster than our headcount.
2. Manual configuration leads to manual errors and doesn't scale.
3. Some automation artifacts (e.g., golden images or VM templates) may lack reproducibility (often due to uncrecorded manual changes).
4. Local, sandboxed development environments may be preferred to reduce iteration time and risk to production systems.

x

x

The Tradeoffs

Figure 1: XKCD 1205, "Is It Worth the Time?"

Figure 2: XKCD 1319, "Automation"

Figure 3: XKCD 2730, "Code Lifespan"

x

Minimum Standards for a Viable Infrastructure as Code (IaC) Solution

1. For any given service, define a single source of authority for:
   - installed packages
   - configuration files
   - running services
   - firewall rules
   - etc. with customization allowed for groups of servers.
2. Automatically apply *all* needed changes, but *only* when needed.
3. Maintain balance of consistency and separation of dev/test/prod environments.
4. Automatically maintain records of who made what change when (and ideally, why).
5. Prefer text over binaries (automation for base OS install instead of golden thick image or VM template).
6. Enable developers to test safely and minimize exposure to outside network.

Stretch Goals for a Viable IaC Solution

1. Allow multiple dev/test environments.
2. Give admins their choice of development platform (Windows, macOS, Linux).
3. Enable management of multiple server OSes (at least multiple Unix, or possibly Windows).
4. Manage endpoints as well as servers.
5. Secure and track secrets (e.g., local database passwords) in central location.
6. Be a good neighbor on already-installed systems (only manage what has to be) and expand scope from there.
7. Avoid vendor lock-in.

x

x

This Isn't the Only Possible Solution

- Some tools used here are derived from our production environment.
- Other tools are ones I've used or promoted in other projects and contexts.
- These tools provide a working reference implementation that's cross-platform (if not totally cross-architecture) with zero purchasing price and open-source licensing.
- Replace any of them with other tools matching your local preferences and standards (the concepts are unchanged).

Provisioning (1/2)

Oracle VM VirtualBox *without Extension Pack* (GNU General Public License v2): Type 2 hypervisor for x86 platforms.

- Runs on Windows, Linux, and macOS (M1/M2 is in developer preview, and hasn't been tested for this application).
- Extension Pack is not open source, and use requires separate license from Oracle.

x

x

VirtualBox Typical Use for Client Testing

Figure 4: Typical Use of VirtualBox on Desktop

Easy, and works fine for testing new OSes or applications.

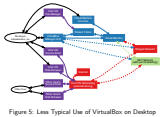


Figure 5: Less Typical Use of VirtualBox on Desktop

x

Provisioning (2/2)

HashiCorp Vagrant (MIT License) Provisioning software for virtual machines.

- Supports programmatic creation of virtual machines and networks.
- Supports in-VM provisioning via file copy, shell script, Ansible, CFEngine, Chef, Docker, Podman, Puppet, and Salt.

x

x

What's Vagrant Doing?

A Vagrant VM is just a VirtualBox VM that:

- runs in the background (headless)
- is usually derived from a base installation from https://app.vagrantup.com/
- configured with a Ruby-syntax `Vagrantfile`
- usually supports a shared folder `/vagrant` mapped from the host OS
- supports ssh from the host operating system through an automatically-forwarded port (via `vagrant ssh`)

What's the Difference with Vagrant for a Single Virtual Machine?

Figure 6: Vagrant use for a single VM

x

How Does It Scale to Multiple Virtual Machines?

Figure 7: Vagrant use for multiple VMs

x

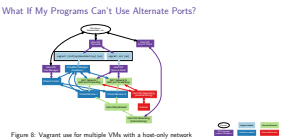


Figure 8: Vagrant use for multiple VMs with a host-only network

x

x

Toolchains

Git (GNU General Public License v2) Keeps track of and logs changes to files in folders. Allows multiple concurrent branches of development, and can push and pull code to/from remote servers (usually via ssh).

Puppet Agent (Apache License) Primarily needed for secret-management tools. Service can be stopped and disabled.

Puppet Development Kit (Apache License) Helper tools for developing and testing Puppet modules and classes.

x

Editing

Microsoft Visual Studio Code (MIT License) 800 pound gorilla of text editors, emacs for a new generation.

Puppet VSCode Extension (Apache License) Provides syntax highlighting, code completion, and linting of Puppet code. Integrates with Puppet Development Kit.

x

Configuration Management

Puppet (primary server and agent) (Apache License) a tool that helps you manage and automate the configuration of servers.

- Code is declarative, describing the desired state of your systems, not the sequence of steps needed to get there.
- Puppet primary server stores the code defining your desired state, and compiles it with facts provided by the agent into a catalog.
- Puppet agent translates the compiled catalog into host-specific commands and executes them.
- Run the agent on the Puppet primary server to have it define its own configuration.

Version Control Server

Gitea (MIT License) self-hosted Git server, features:

- single Go binary with SQLite support.
- issue tracking and wikis.
- organizational hierarchy.
- OpenID Connect single sign-on (yes, it works with Azure Active Directory).
- branch protection and review before merge.
- webhooks to trigger automation on various events.

x

x

Continuous Deployment

Adnan Hajdarevic (adnanh)'s Webhook (MIT License) lightweight configurable tool written in Go, that allows you to easily create HTTP endpoints (hooks) on your server, which you can use to execute configured commands.

r10k (Apache License) provides a general purpose toolset for deploying Puppet environments and modules. Maps a branch in a Git repository to a Puppet environment.

Combining Git branches, Gitea webhooks, adnanh's Webhook, and r10k allows easy management of multiple Puppet environments for developing and testing services.

VS Code, Puppet VS Code Extension, VirtualBox, Vagrant, Puppet Agent, Puppet Development Kit

Use default settings for all.

x

Git

- Windows, macOS, Linux installation: you should be able to follow Software Carpentry's template workshop instructions for installation.
- Windows, macOS, Linux setup: following with Software Carpentry's Setting Up Git page, opening a command prompt and running:
  - `git config --global user.name "Your Name"`
  - `git config --global user.email "you@yourplace.edu"`
  - and either:
    - `git config --global core.autocrlf input` for macOS and Linux, or
    - `git config --global core.autocrlf false` for Windows

  should be enough to get started.

x

Git in Visual Studio Code

- File / Open Folder
- Find existing folder, or create empty one.
- View / Source Control
- Initialize Repository button

x

x

Terminal in Visual Studio Code

- View / Terminal

Places you at the top-level Git repository folder.

x

How To Make the First Vagrant VM?

We want a VM:

- running a Red Hat-family OS, version 8
- supporting a shared folder `/vagrant` mapped from the host OS

Start the definition of a new Vagrant VM in the repository folder with `vagrant init bento/rockylinux-8`, and look at the `Vagrantfile` that was just created.

(The bento Vagrant boxes are built by the Chef Bento project.)

x

The First Vagrantfile (1/2)

Generated from `vagrant init bento/rockylinux-8`, filtered down to interesting commands and comments.

```
Vagrant.configure("2") do |config|
  config.vm.box = "bento/rockylinux-8"
  # config.vm.network "forwarded_port", guest: 80, host: 8080
  # config.vm.network "forwarded_port", guest: 80, host: 8080,
  #   host_ip: "127.0.0.1"
  # config.vm.network "private_network", ip: "192.168.33.10"
  # config.vm.provider "virtualbox" do |vb|
  #   # Customize the amount of memory on the VM:
  #   vb.memory = "1024"
  # end
```

x

The First Vagrantfile (2/2)

```
  # config.vm.provision "shell", inline: <<-SHELL
  #   apt-get update
  #   apt-get install -y apache2
  # SHELL
end
```

You'll also see a 1 badge on the Source Control button in Visual Studio Code, indicating there's one pending change in the repository folder (the creation of the `Vagrantfile`).

Installing a New Vagrant VM with `vagrant up`

- Running `vagrant up` from the Visual Studio Code terminal builds the VM.
- Running `vagrant ssh` from the Visual Studio Code terminal results in you being logged into a `vagrant` account in the Linux VM, which has passwordless `sudo` rights.
- If you exit back out to your host command prompt, you can do a `vagrant destroy` to shut down and delete the VM.

Once that's working, we'd like to record the new Vagrantfile into version control in Visual Studio Code, but there are complications.

x

x

Setting Up Version Control (How to Ignore Machine-Generated Files)

- View / Source Control

Notice there are several other files in the repository folder now (the Source Control button probably has a badge of 3 now). These are from the `.vagrant` folder that Vagrant uses to track virtual machine information.

- Right-click one of the files, select "Add to .gitignore"
- Notice the file is absent from the Source Control button, and there's a new file `.gitignore`.
- Go back to the folder view (View / Explorer)
- Select the file `.gitignore` to open it in the editor
- Edit the only line in `.gitignore` to be just `.vagrant/` (no filename) and save the file.

x

Setting Up Version Control (Adding and Committing Useful Files)

- Notice the Source Control badge now reads 2 (one for `Vagrantfile`, one for `.gitignore`).
- Right-click `Vagrantfile`, select "Stage Changes" (or hit + sign to right of `Vagrantfile`).
- Repeat for `.gitignore`.
- In the "Message" text entry, enter `Define initial Vagrantfile and .gitignore` and select the "Commit" button.

In theory, Git commits should be "atomic", *i.e.*, a single, complete unit of work that can be described in a single sentence. In practice, we're often not that disciplined about it.

Git commit messages should be short and imperative, completing the sentence, "when applied, this commit will . . . ."

x

What Would We Like to Change?

Things to Fix

1. VM hostname is currently `localhost`, would like that to change.
2. Need some actual configuration (packages, config files, services, etc.)
3. Need *multiple* VMs (Git server, Puppet primary server, Puppet client, etc.)
4. Need Puppet agent to poll Puppet primary server for changes (requires name resolution from DNS, host file, etc.)

Tools for Fixing Things

1. `Vagrantfile` settings
2. VM provisioners (`shell` shown by default, but `puppet` provisioner also available)
3. Provisioners can also read from files (shell scripts, Puppet manifests)
4. Files in the repository folder (show up in `/vagrant` in the VMs)
5. DRY (don't repeat yourself) principle (avoid copy/paste)

x

Reference Architecture

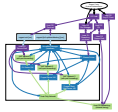


Figure 9: Reference architecture

- Administrator only directly interacts with VS Code, host file manager, and any host IP clients (*e.g.*, web browser)
- VMs communicate over shared internal network

Minimum Viable IaC Part 1: Bootstrapping a Git Server

Minimum Viable IaC Part 1: Bootstrapping a Git Server

- Vagrant allows for multiple provisioning blocks in the Vagrantfile.
- We'll use the `aba11` provisioner to install the Puppet agent in each VM (later, we'll let the `puppet` provisioner to do the rest of the setup in each VM).

x

Initial Vagrantfile for Git server

```
Vagrant.configure("2") do |config|
  # general settings for all VMs
  config.vm.box = "bento/rockylinux-8"
  config.vm.provision "shell", path: "provision.sh"
  # settings specific to the git VM
  config.vm.define "git" do |git|
    git.vm.hostname = "git"
    git.vm.network "private_network", ip: "10.234.24.2",
      netmask: "255.255.255.0"
  end
end
```

x

Contents of provision.sh

```
#!/bin/bash
# Ensure working local DNS
nmcli con modify 'eth0' ipv4.dns-search 'theita23.remf.ro' \
  ipv4.ignore-auto-dns yes ipv4.dns '10.234.24.254'
systemctl restart NetworkManager
YUM="yum -q -y"
# Install puppet
${YUM} install http://yum.puppet.com/puppet7-release-el-8.noarch.rpm
${YUM} install puppet-agent
```

x

x

Build Git Server, Verify Puppet Exists, Then Commit Changes

- `vagrant up git` to build
- `vagrant ssh git` to log in
- `sudo -i puppet --version` to see Puppet is installed
- `exit` to log out

In VS Code:

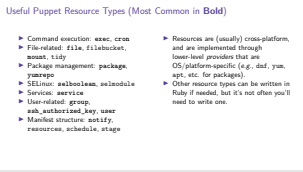
- View / Source Control
- add `Vagrantfile` and `provision.sh` to the staged changes
- commit changes with message `Define initial Git server and install puppet agent`

Adding a Second Provisioner

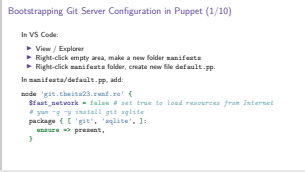
- Vagrant allows for multiple provisioning blocks in the Vagrantfile.
- Now that the Puppet agent is installed in the VM, we can use it to install and configure other items.
- Add a line below the shell provisioner in the Vagrantfile containing:

```
config.vm.provision "puppet" # defaults to manifests/default.pp
```

x


Useful Puppet Resource Types (Most Common in Bold)
Command execution: exec, cron
File-related: file, filebucket, mount, tidy
Package management: package, yumrepo
SELinux: selboolean, selmodule
Services: service
User-related: group, ssh_authorized_key, user
Manifest structure: notify, resources, schedule, stage
Resources are (usually) cross-platform, and are implemented through lower-level providers that are OS/platform-specific (e.g., dnf, yum, apt, etc. for packages).
Other resource types can be written in Ruby if needed, but it's not often you'll need to write one.

x


Bootstrapping Git Server Configuration in Puppet (1/10)
In VS Code:
View / Explorer
Right-click empty area, make a new folder manifests
Right-click manifests folder, create new file default.pp.
In manifests/default.pp, add:
node 'git.theits23.rwmf.ro' {
$fast_network = false # set true to load resources from Internet
yum -y -y install git sqlite
package { ['git', 'sqlite',]:
ensure => present,
}

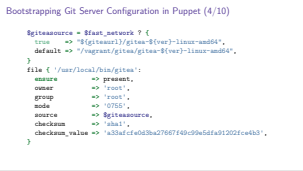
x


Bootstrapping Git Server Configuration in Puppet (2/10)
user { 'git': # adduser --system --create-home git
ensure => present,
system => true,
managehome => true,
}
['', 'custom', 'data', 'log'].each |$dir| {
file { "/var/lib/gitea/${dir}":
ensure => directory,
owner => 'git',
group => 'git',
mode => '0750',
require => User['git'],
} # install -d -m git -g git -m 0750 /var/lib/gitea/{custom,data,log}
}

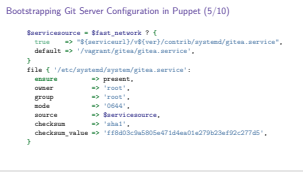
x


Bootstrapping Git Server Configuration in Puppet (3/10)
file { '/etc/gitea': # install -d -o root -g git -m 0770 /etc/gitea
ensure => directory,
owner => 'root',
group => 'git',
mode => '0770',
require => User['git'],
}
$ver = '1.18.5'
$giteaurl = "https://dl.gitea.com/gitea/${ver}"
$serviceurl = "https://raw.githubusercontent.com/go-gitea/gitea"

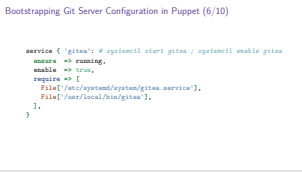
x


Bootstrapping Git Server Configuration in Puppet (4/10)
$gitesource = $fast_network ? {
true => "${giteasrl}/gitea-${ver}-linux-amd64",
default => "/vagrant/gitea/gitea-${ver}-linux-amd64",
}
file { '/usr/local/bin/gitea':
ensure => present,
owner => 'root',
group => 'root',
mode => '0755',
source => $giteasource,
checksum => 'sha1',
checksum_value => 'a33afcfe0d3ba27667f49c99e5dfa91202fce4b3',
}

x


Bootstrapping Git Server Configuration in Puppet (5/10)
$servicesource = $fast_network ? {
true => "${servicecurl}/v${ver}/contrib/systemd/gitea.service",
default => '/vagrant/gitea/gitea.service',
}
file { '/etc/systemd/system/gitea.service':
ensure => present,
owner => 'root',
group => 'root',
mode => '0644',
source => $servicesource,
checksum => 'sha1',
checksum_value => 'ff8d03c9a5805e471d4ea01e279b23ef92c277d5',
}

x


Bootstrapping Git Server Configuration in Puppet (6/10)
service { 'gitea': # systemctl start gitea ; systemctl enable gitea
ensure => running,
enable => true,
require => [
File['/etc/systemd/system/gitea.service'],
File['/usr/local/bin/gitea'],
],
}

x

Bootstrapping Git Server Configuration in Puppet (7/10)

```
$localnet = '10.234.24.0/24'
exec { 'allow webhooks to local network':
  command  => 'echo -e \ # wrapped for slide, should be one line
  '[webhook]\nALLOWED_HOST_LIST = ${localnet}' >> /etc/gitea/app.ini",
  provider => 'shell',
  onlyif   => 'test \ # wrapped for slide, should be one line
  $(grep -q ALLOWED_HOST_LIST /etc/gitea/app.ini; echo $?) -eq 1',
  notify   => Service['gitea'],
}
```

x

Bootstrapping Git Server Configuration in Puppet (8/10)

```
exec { 'fix root URL for Gitea':
  command  => 'perl -pi.bak \ # wrapped for slide, should be one line
  -e \'s#localhost:3000#10.234.24.2:3000#g\' /etc/gitea/app.ini',
  provider => 'shell',
  onlyif   => 'test \ # wrapped for slide, should be one line
  $(grep -q 10.234.24.2:3000 /etc/gitea/app.ini; echo $?) -eq 1',
  notify   => Service['gitea'],
}
```

x

Bootstrapping Git Server Configuration in Puppet (9/10)

```
exec { 'fix domain, ssh domain for Gitea':
  command  => 'perl -pi.bak \ # wrapped for slide, should be one line
  -e \'s#= localhost#= 10.234.24.24#g\' /etc/gitea/app.ini',
  provider => 'shell',
  onlyif   => 'test \ # wrapped for slide, should be one line
  $(grep -q "= 10.234.24.2" /etc/gitea/app.ini;echo $?) -eq 1',
  notify   => Service['gitea'],
}
```

Note: all of the preceding code is geared for bootstrapping and demonstrating fundamental resource types. Normally we use much higher-level abstractions for configuration where possible.

x

Bootstrapping Git Server Configuration in Puppet (10/10)

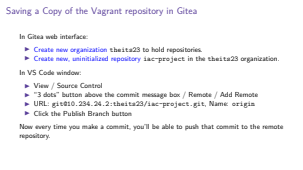
- At host terminal, run `vagrant provision git --provision-with puppet`
- If this fails due to the Vagrantfile having changed while the VM was running, run `vagrant reload`.
- Watch Puppet download, install, and configure Gitea.
- At host terminal, re-run `vagrant provision git --provision-with puppet`
- Watch Puppet determine no further changes need to be made.
- Add and commit the changes to `Vagrantfile` and `default.pp`.

x

Final Configuration of Gitea Through the Web

- Head to http://10.234.24.2:3000/ in the host browser for initial setup of Gitea.
  - Database Type: SQLite3
  - Gitea Base URL: http://10.234.24.2:3000/
  - Administrator Username: gitadmin
  - Password: (anything)
  - Email Address: (anything)
- On the host, generate an ssh key with `ssh-keygen -t ed25519`, then `cat ~/.ssh/id_ed25519.pub` (if you already have an ssh public key, you can cat it instead).
- Copy/paste the public key content into http://10.234.24.2:3000/user/settings/keys.
- At the host terminal, re-run `vagrant provision git --provision-with puppet` to finalize the last Gitea settings from Puppet.

x


Saving a Copy of the Vagrant repository in Gitea
In Gitea web interface:
Create new organization theita23 to hold repositories.
Create new, uninitialized repository iac-project in the theita23 organization.
In VS Code window:
View / Source Control
"3 dots" button above the commit message box / Remote / Add Remote
URL: git@10.234.24.2:theita23/iac-project.git, Name: origin
Click the Publish Branch button
Now every time you make a commit, you'll be able to push that commit to the remote repository.

x

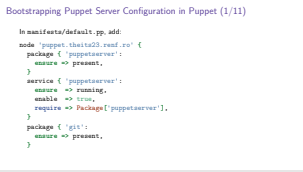
Minimum Viable IaC Part 2: Bootstrapping a Puppet Primary Server

Minimum Viable IaC Part 2: Bootstrapping a Puppet Primary Server

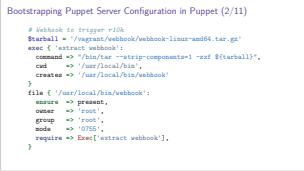
Repeat the same procedure to make a new Vagrant VM for Puppet. Realistically, a Puppet server needs at least 4 GB RAM, and extra cores don't hurt. Add the following to the `Vagrantfile`, right below the Git server definition:

```
config.vm.define "puppet" do |puppet|
  puppet.vm.provider "virtualbox" do |vb|
    vb.cpus = "2"
    vb.memory = "4096"
  end
  puppet.vm.hostname = "puppet"
  puppet.vm.network "private_network", ip: "10.234.24.3",
    netmask: "255.255.255.0"
end
```

x


Bootstrapping Puppet Server Configuration in Puppet (1/11)
In manifests/default.pp, add:
node 'puppet.theita23.rsmf.ro' {
package { 'puppetserver':
ensure => present,
}
service { 'puppetserver':
ensure => running,
enable => true,
require => Package['puppetserver'],
}
package { 'git':
ensure => present,
}

x


Bootstrapping Puppet Server Configuration in Puppet (2/11)
Webhook to trigger r10k
$tarball = '/vagrant/webhook/webhook-linux-amd64.tar.gz'
exec { 'extract webhook':
command => "/bin/tar --strip-components=1 -xzf ${tarball}",
cwd => '/usr/local/bin',
creates => '/usr/local/bin/webhook'
}
file { '/usr/local/bin/webhook':
ensure => present,
owner => 'root',
group => 'root',
mode => '0755',
require => Exec['extract webhook'],
}

x

Bootstrapping Puppet Server Configuration in Puppet (3/11)

```
file { '/etc/webhook.yaml':
  ensure => present,
  owner  => 'root',
  group  => 'root',
  mode   => '0644',
  source => '/vagrant/webhook/webhook.yaml',
}
file { '/etc/systemd/system/webhook.service':
  ensure => present,
  owner  => 'root',
  group  => 'root',
  mode   => '0644',
  source => '/vagrant/webhook/webhook.service',
}
```

x

Bootstrapping Puppet Server Configuration in Puppet (4/11)

```
service { 'webhook':
  ensure  => running,
  enable  => true,
  require => [
    File['/etc/systemd/system/webhook.service'],
    File['/etc/webhook.yaml'],
    File['/usr/local/bin/webhook'],
  ],
}
```

x

Bootstrapping Puppet Server Configuration in Puppet (5/11)

```
# r10k and dependencies
package { 'r10k':
  ensure   => 'present',
  provider => 'puppetserver_gem',
}
['r10k', 'r10k-deploy-ref'].each |$f| {
  file { "/usr/local/bin/${f}":
    ensure => present,
    source => "/vagrant/webhook/${f}",
    owner  => 'root',
    group  => 'puppet',
    mode   => '0750',
  }
}
```

x

Bootstrapping Puppet Server Configuration in Puppet (6/11)

```
file { '/etc/puppetlabs/r10k':
  ensure => directory,
  owner  => 'root',
  group  => 'puppet',
  mode   => '0750',
}
file { '/etc/puppetlabs/r10k/r10k.yaml':
  source => '/vagrant/webhook/r10k.yaml',
  owner  => 'root',
  group  => 'puppet',
  mode   => '0640',
}
```

x

Bootstrapping Puppet Server Configuration in Puppet (7/11)

```
# Prepare ssh keys for r10k git operations
$puppet_home = '/opt/puppetlabs/server/data/puppetserver'
file { "${puppet_home}/.ssh":
  ensure  => directory,
  owner   => 'puppet',
  group   => 'puppet',
  mode    => '0700',
  require => Package['puppetserver'],
}
```

x

Bootstrapping Puppet Server Configuration in Puppet (8/11)

```
$algo = 'ed25519'
$keyfile = "${puppet_home}/.ssh/id_${algo}"
$makekey = "ssh-keygen -t ${algo} -f ${keyfile}"
exec { 'create puppet ssh deploy key':
  command  => "${makekey} -q -N ''",
  provider => 'shell',
  user     => 'puppet',
  creates  => ${keyfile},
  require  => File["${puppet_home}/.ssh"],
}
```

x

Bootstrapping Puppet Server Configuration in Puppet (9/11)

```
-> exec { 'import git ssh host key':
  command  => "ssh-keyscan -t ecdsa git.theita23.renf.ro,10.234.24.2 \
  # wrapped for slide, should be one line
  > ${puppet_home}/.ssh/known_hosts",
  provider => 'shell',
  user     => 'puppet',
  creates  => "${puppet_home}/.ssh/known_hosts",
}
```

x

Bootstrapping Puppet Server Configuration in Puppet (10/11)

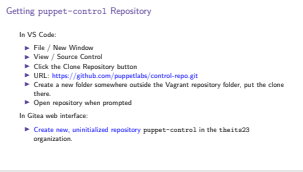
```
# Prepare initial clone directories for r10k
file { [
  '/etc/puppetlabs/code/environments',
  '/etc/puppetlabs/code/environments/production'
  ]:
  ensure => directory,
  owner  => 'puppet',
  group  => 'puppet',
  mode   => '0750',
}
```

x

Bootstrapping Puppet Server Configuration in Puppet (11/11)

- At host terminal, run `vagrant up puppet`
- Puppet server will get installed in one run (OS, shell provisioner, puppet provisioner).
- At host terminal, re-run `vagrant provision puppet --provision-with puppet`
- Watch Puppet determine no further changes need to be made.
- Add and commit the changes to `Vagrantfile` and `default.pp`.

x


Getting puppet-control Repository
In VS Code:
File / New Window
View / Source Control
Click the Clone Repository button
URL: https://github.com/puppetlabs/control-repo.git
Create a new folder somewhere outside the Vagrant repository folder, put the clone there.
Open repository when prompted
In Gitea web interface:
Create new, uninitialized repository puppet-control in the theita23 organization.

x

x

Saving a Local (Gitea) Copy of puppet-control

In VS Code window with the `puppet-control` repository open:

- View / Source Control
- "3 dots" button above the commit message box / Remote / Remove Remote
- Select `origin` as the remote to remove
- "3 dots" button above the commit message box / Remote / Add Remote
- URL: `git@10.234.24.2:theita23/puppet-control.git`
- Name: `origin`
- Click the Publish Branch button

Webhook Between Gitea and Puppet

Webhook Between Gitea and Puppet

We want to automatically have changes to a Puppet code repository stored in Gitea automatically show up on the Puppet server.

Gitea can be configured to make a web request when certain events occur on any or all branches in a repository. In our case:

- Every change to every branch will trigger a web request to the webhook service on the Puppet server.
- The webhook service will be told what branch was changed.
- The webhook service will run a defined script with a command parameter including the branch name.
- The script will run the `r10k` program to check out that branch, pull down prewritten modules from Puppet Forge or Git repositories, and deploy an entire Puppet environment defining multiple servers.

x

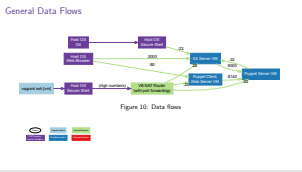


Figure 10: Data flows

x

/etc/webhook.yaml

```yaml
- id: r10k
  execute-command: /usr/local/bin/r10k-deploy-ref
  command-working-directory: /tmp
  pass-arguments-to-command:
  - source: payload
    name: ref
```

x

/usr/local/bin/r10k-deploy-ref

```
#!/bin/bash
if [ $# -ne 1 ]; then
  echo "Usage: $0 REF"
  echo "Example: $0 refs/heads/production"
  exit 1
else
  # "How to split a string in shell and get the last field"
  # -- https://stackoverflow.com/q/918251#918
  REF="$1"; BRANCH=$(echo "${REF}" | rev | cut -d/ -f1 | rev)
  /usr/local/bin/r10k deploy environment "${BRANCH}" -p
  /opt/puppetlabs/bin/puppet generate types --environment "${BRANCH}" \
    --codedir /etc/puppetlabs/code
fi
```

x

/usr/local/bin/r10k

```
#!/bin/bash
/opt/puppetlabs/bin/puppetserver ruby \
  /opt/puppetlabs/server/data/puppetserver/jruby-gems/bin/r10k $@
```

x

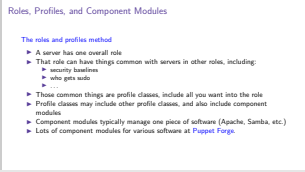
Gitea configuration

- Add the puppet user's public key as a deploy key for the `puppet-control` repository.
- Configure an outgoing webhook in the `puppet-control` repository, pointed at `http://puppet.theits23.renf.ro:9000/hooks/r10k`.

x

x

Figure 11: GitHub Flow applied to Puppet development


Roles, Profiles, and Component Modules
The roles and profiles method
A server has one overall role
That role can have things common with servers in other roles, including:
security baselines
who gets sudo
...
Those common things are profile classes, include all you want into the role
Profile classes may include other profile classes, and also include component modules
Component modules typically manage one piece of software (Apache, Samba, etc.)
Lots of component modules for various software at Puppet Forge.

x

x

Provisioning a New Web Server in Puppet (1/10)

Modify the `Vagrantfile` to include a web server VM:

```
config.vm.define "web" do |web|
  web.vm.hostname = "web"
  web.vm.network "private_network", ip: "10.234.24.4",
    netmask: "255.255.255.0"
end
```

and the Puppet `manifests/default.pp` to include:

```
node 'web.theits23.rmf.ro' {
  notify { 'web to be configured from puppet server':
    message => 'Web server to be configured from the Puppet server.',
  }
}
```

x

Provisioning a New Web Server in Puppet (2/10)

In VS Code window for the `iac-project` repository:

- Run `vagrant up web` and verify the web server VM is created and prints the warning message about getting configured from the Puppet server.
- Add, commit, and push the changes to `Vagrantfile` and `default.pp`.

In VS Code window for `puppet-control` repository:

- View / Source Control
- "3 dots" button above the commit message box / Branch / Create Branch From
- Use `production` branch as source, name the new branch `new_webserver`
- Click the Publish Branch button

x

Provisioning a New Web Server in Puppet (3/10)

In `Puppetfile`, ensure the lines

```
mod 'puppetlabs-apache', '9.1.2'
mod 'puppetlabs-concat', '7.4.0'
mod 'puppetlabs-inifile', '6.0.0'
mod 'puppetlabs-stdlib', '8.5.0'
```

exist (all these modules are from Puppet Forge). Add and commit this change with a message like `set up current module versions`.

x

Provisioning a New Web Server in Puppet (4/10)

In the `puppet-control` `manifests/site.pp`, replace the `node default` entry with:

```
node default {
  $classes = lookup('classes', Variant[String])
  case $classes {
    String[1]: { include $classes }
    default: { fail('This node did not receive any classification') }
  }
}
```

Save, add, and commit this change with a message like `derive node classes from Hiera` — this is a simpler version of Updating Puppet classification with hiera to use the modern lookup command.


Provisioning a New Web Server in Puppet (5/10)
Make a new file site-modules/profile/manifests/apache.pp in the puppet-control repository. Add the following lines to it:
# Summary Configures Apache to a site-specific standard
class profile::apache {
class {'apache': }
# other things can go here, like mounting
# partitions for logs or content
}
Save, add, and commit this change with a message like define apache profile.

x

x

Provisioning a New Web Server in Puppet (6/10)

Edit the file `site-modules/role/manifests/webserver.pp` in the puppet-control repository. Add the following lines to it below `include profile::base`:

```
include profile::apache
```

Save, add, and commit this change with a message like `update webserver role to use Apache`.

x

Provisioning a New Web Server in Puppet (7/10)

Use Hiera to Separate Data from Code

- Stores site-specific data in YAML, JSON, or HOCON formats
- Supports a lookup hierarchy by hostname, domain, OS, OS family, etc.
- Supports public-key encrypted data (admins encrypt values with shared public key, Puppet decrypts on the fly with private key)


Provisioning a New Web Server in Puppet (8/10)
Make a new file data/nodes/web.theits23.remf.ro.yaml in the puppet-control repository. Add the following lines to it:
---
classes: role::webserver
Save, add, and commit this change with a message like make 'web' a web server. Then push all the commits to the remote Git repository with the Sync Changes button.

x

x

Provisioning a New Web Server in Puppet (9/10)

- Log into the web server and edit the Puppet agent settings with `vagrant ssh web -c "sudo vi /etc/puppetlabs/puppet/puppet.conf"` to add lines

  ```
  [agent]
  environment=new_webserver
  ```

- Generate a certificate signing request (CSR) for the Puppet agent on the new web server with `vagrant ssh web -c "sudo -i puppet agent -t"` and log out.
- Sign the CSR on the Puppet primary server with `vagrant ssh puppet -c "sudo -i puppetserver ca sign --certname web.theits23.renf.ro"`
- Apply changes to the web server through the Puppet agent with `vagrant ssh web -c "sudo -i puppet agent -t"`

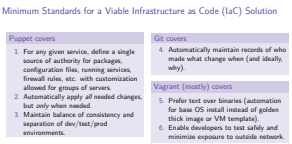
x

Provisioning a New Web Server in Puppet (10/10)

- Verify you've got a working web server by pointing the host web browser to http://10.234.24.4/

Once we're happy with the changes to the web server, we can merge them into production in Gitea:

- Make a new pull request from the new_webserver into production
- Merge the pull request and delete the new_webserver branch
- Edit the web VM's *puppet.conf* with `vagrant ssh web -c "sudo vi /etc/puppetlabs/puppet/puppet.conf"` and remove the environment=new_webserver line.


Minimum Standards for a Viable Infrastructure as Code (IaC) Solution
Puppet covers
1. For any given service, define a single source of authority for packages, configuration files, running services, firewall rules, etc. with customization allowed for groups of servers.
2. Automatically apply all needed changes, but *only* when needed.
3. Maintain balance of consistency and separation of dev/test/prod environments.
Git covers
4. Automatically maintain records of who made what change when (and ideally, why).
Vagrant (mostly) covers
5. Prefer test over binaries (automation for base OS install instead of golden thick image or VM template).
6. Enable developers to test safely and minimize exposure to outside network.

x

x

Stretch Goals for a Viable IaC Solution

Puppet covers

1. Allow multiple dev/test environments.
2. Give admins their choice of development platform (Windows, macOS, Linux).
3. Enable management of multiple server OSes (at least multiple Unix, or possibly Windows).
4. Manage endpoints as well as servers.
5. Secure and track secrets (e.g., local database passwords) in central location.
6. Be a good neighbor on already-installed systems (only manage what has to be) and expand scope from there.
7. Avoid vendor lock-in.

Things We Didn't Get To

- Cross-platform support in Puppet
- Running Puppet agent as a service
- Encrypted data in Hiera
- Options for separating Hiera data by OS, OS family, domain, etc.
- Enterprise Node Classifiers to centrally control which nodes get which environment
- Deeper dive into "facts" gathered by each node that inform the compiled catalog
- Distributing files and templates from Puppet
- Adding parameters to profile classes (usually populated from Hiera)
- Puppet Development Kit for building your own component modules
- More Gitea settings (branch protection, centralized authentication, ...)

x

Software Credits

All presentation content edited with Visual Studio Code, tracked with Git, and hosted in GitHub.

Terminal output captured with asciinema on Unix or PowerSession on Windows.

Slides created in Pandoc's Markdown format, converted to PDF with Pandoc, TeX Live and LaTeX Beamer. Slide theme: Cookeville, presented with Skim.

Graphs created in DOT format with Brewer paired12 color scheme, converted with Graphviz to PDF.

Questions?

x