Infrastructure Prototyping with Bolt and Vagrant

Steven Pritchard

Vagrant

- From Hashicorp
  - Makers of Terraform, Vault, and others
- Easy infrastructure prototyping
  - Define one or more VMs of any type with any arbitrary configuration
  - Unified configuration language for VirtualBox, libvirt, and others

https://vagrantup.com/

- From Puppet
  - Makers of, well, Puppet
- Orchestrates actions across systems
  - Can use any combination of commands, scripts, or even Puppet code
  - Flexible inventory, with support for custom plugins

https://puppet.com/open-source/bolt/

Getting Started

Bolt
https://puppet.com/docs/bolt/latest/bolt_installing.html

Vagrant
https://www.vagrantup.com/downloads

Vagrant provider

- VirtualBox
  https://www.virtualbox.org/wiki/Downloads

Building VMs

- Create an empty directory to work from
- **`cd`** into it
- Run **`vagrant init centos/8`**
  - Creates a *Vagrantfile* with a CentOS 8 VM configured

```
Vagrant.configure("2") do |config|
 config.vm.box = "centos/8"
end
```

- Edit the *Vagrantfile* to add additional VMs

Building VMs

Example *Vagrantfile*:

```
Vagrant.configure("2") do |config|
 config.vm.define "centos" do |centos|
   centos.vm.box = "centos/8"
   centos.vm.provider "virtualbox" do |vb|
     vb.memory = "1024"
   end
 end

 config.vm.define "windows" do |windows|
   windows.vm.box = "gusztavvargadr/windows-server"
   windows.vm.provider "virtualbox" do |vb|
     vb.memory = "1024"
   end
 end
end
```

To start the VMs, run **`vagrant up`**.

Useful vagrant subcommands

- `vagrant init`
- `vagrant up`
- `vagrant rsync`
- `vagrant provision`
- `vagrant reload`
- `vagrant halt`
- `vagrant destroy`
- `vagrant ssh`
- `vagrant status`
- `vagrant global-status`

Setting up Bolt

```
$ bolt project init
```

or

```
$ bolt project init projectname
```

Creates two files

- *bolt-project.yaml*
- *inventory.yaml*

Adding the `bolt_vagrant` module

```
$ bolt module add dylanratcliffe-bolt_vagrant
```

- Updates *bolt-project.yaml*
- Creates a Bolt-managed *Puppetfile*
- Installs the module (plus any dependencies) in the *.modules* directory

https://forge.puppet.com/modules/dylanratcliffe/bolt_vagrant

Enabling `bolt_vagrant` inventory

- Add the following to *inventory.yaml*:

```
targets:
  - _plugin: task
    task: bolt_vagrant::targets
```

- Run **`bolt inventory show -t all`**
  - You should see **`centos`** and **`windows`** in the output.

Running ad-hoc commands

Run a command across all of our VMs:

```
$ bolt command run hostname -t all
```

Running scripts

Run arbitrary scripts against our VMs:

```
$ echo hostname > myscript.ps1

$ bolt script run myscript.ps1 -t all
```

Running tasks

Bolt tasks are scripts with some associated metadata that is used to specify script arguments, etc.

Several tasks are shipped with Bolt. You can see the current list with `bolt task show`.

To add additional tasks, either install additional Puppet modules that contain tasks or create a directory named *tasks* and drop in the necessary files.

Running tasks

tasks/hostname.json:

```
{
 "puppet_task_version": 1,
 "supports_noop": false,
 "description": "Run hostname",
 "implementations": [
   { "name": "hostname.sh", "requirements": ["shell"] },
   { "name": "hostname.ps1", "requirements": ["powershell"] }
 ],
 "parameters": {
 }
}
```

tasks/hostname.sh:

```
#!/bin/sh

hostname
```

tasks/hostname.ps1:

```
hostname
```

Running tasks

Once those files are in place, you can run the task on all nodes with the following command:

```
$ bolt task run demo::hostname -t all
```

Plans

- Plans can run a set of commands, scripts, tasks, or other plans in a set sequence on a list of targets.
- Bolt supports two different kinds of plans
  - Plans written in YAML
  - Plans written in the Puppet language

Writing Plans

Generate a new YAML plan template:

```
$ bolt plan new demo::hostname1
```

Generate a Puppet-language plan template:

```
$ bolt plan new demo::hostname2 --pp
```

Alternatively, add plan files in a directory named *plans* in the Bolt project

Plans

For **`demo::hostname1`**, we can modify *plans/hostname1.yaml* to look like this:

```
description: Run hostname via a YAML plan
parameters:
 targets:
   type: TargetSpec
   description: A list of targets to run actions on
   default: all
steps:
 - message: Hello from demo::hostname1
 - name: command_step
   command: hostname
   targets: $targets
return: $command_step
```

Plans

To get the same functionality out of **`demo::hostname2`**, modify *plans/hostname2.pp* to look like this:

```
plan demo::hostname2 (
 TargetSpec $targets = 'all',
) {
 out::message('Hello from demo::hostname2')
 $command_result = run_command('hostname', $targets)
 return $command_result
}
```

Plans

You can run these plans with

```
$ bolt plan run demo::hostname1
```

or

```
$ bolt plan run demo::hostname2
```

Note that we do not need to supply a list of targets for either of these plans because they default to a target list of `all`.

Using a Bolt Plan for Provisioning

With a couple of minor tweaks to our *Vagrantfile*, we can use a Bolt plan for provisioning.

First, we need to enable an experimental Vagrant feature - **`typed_triggers`**. We can do this by adding the following line to the beginning of our *Vagrantfile*:

```
ENV['VAGRANT_EXPERIMENTAL'] = 'typed_triggers'
```

Using a Bolt Plan for Provisioning

Next, we want to make sure that any Puppet modules that Bolt needs (like **`bolt_vagrant`**) are installed before we try to use them. Add the following to the **`Vagrant.configure(2)`** block:

```
config.trigger.before [:up, :provision, :reload], type: :command do |trigger|
  trigger.info = 'Initializing bolt'
  trigger.run = { inline: 'bolt module install' }
end
```

Using a Bolt Plan for Provisioning

Finally, we want to run our plan after all of the VMs have started:

```
config.trigger.after [:up, :provision, :reload], type: :command do |trigger|
  trigger.info = 'Running bolt plan'
  trigger.run = { inline: 'bolt plan run demo' }
end
```

Examples

You can find examples of this pattern on GitHub.

https://github.com/topics/puppet-bolt?q=vagrant-environments&type=topics

sicura.us

LinkedIn