DevOps

PUPPET

By
ADITYA PRABHAKARA

Introducing Myself

Aditya S P (sp.aditya@gmail.com)
Freelance trainer and technologist

Boring Stuff about me:

- 14+ years of experience in development and training
- Started with Java, moved to Android and now working on Big Data Technologies

Interesting Things about me:

- Actually Nothing !

Getting to know you

Show of hands please!

- Any freshers in this group?
- What is the general development experience of this group
  - 0-2 years, 0-5 years, 5 and above
- What programming area are you currently working on?
  - Java, Web Stack, Analytics, Big data, any other
- Why are you learning python programming?
  - Sys admin, Web development, Data Analytics, IoT, any other

Puppet

A GPL Open Source Project written in Ruby

●A declarative language for expressing system configuration

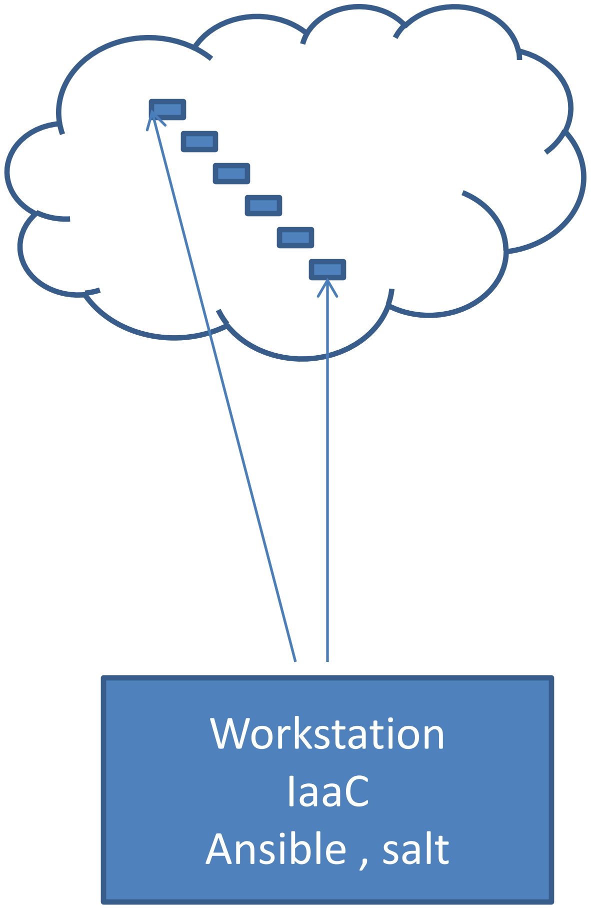
● A Client and server

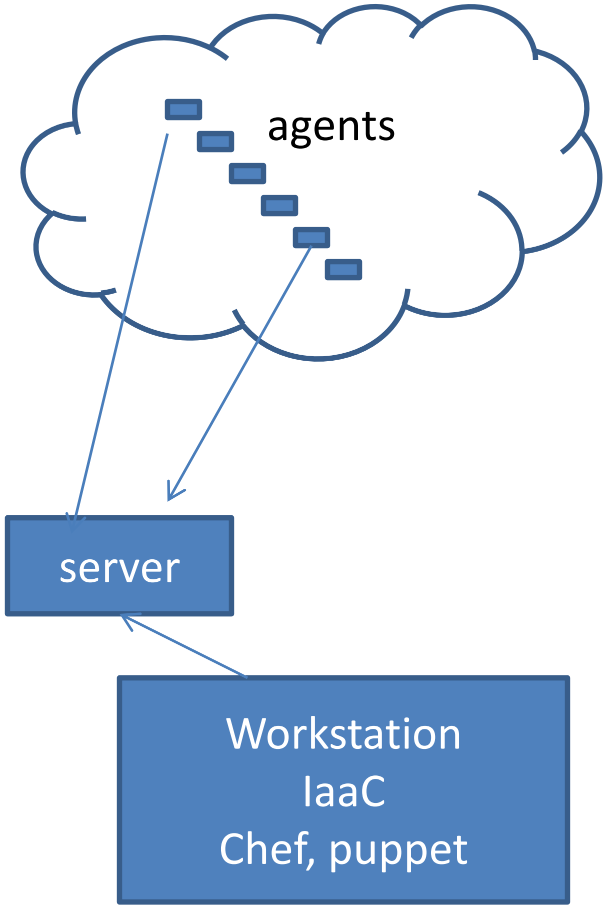
● A library to realize the configuration

●Puppet is the abstraction layer between the system admini strator and the system

● Puppet requires only Ruby and Facter

● Client runs every 30 minutes by default


Workstation
IaaC
Ansible , salt


agents
server
Workstation
IaaC
Chef, puppet

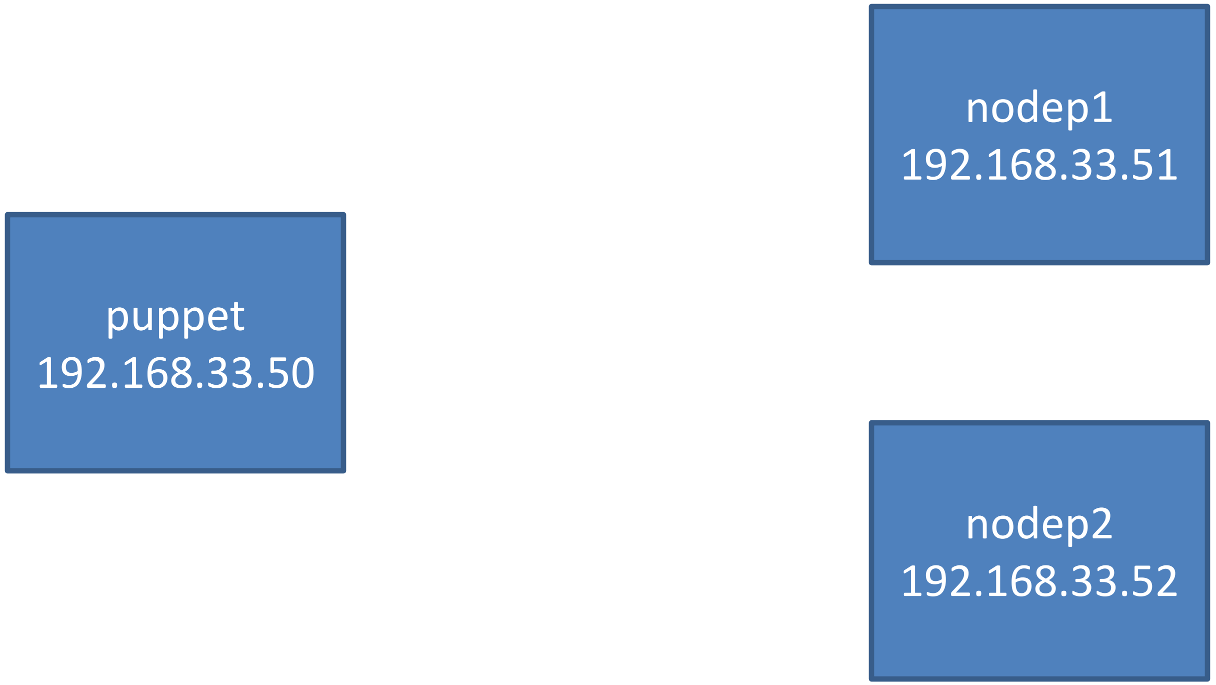
Our Setup

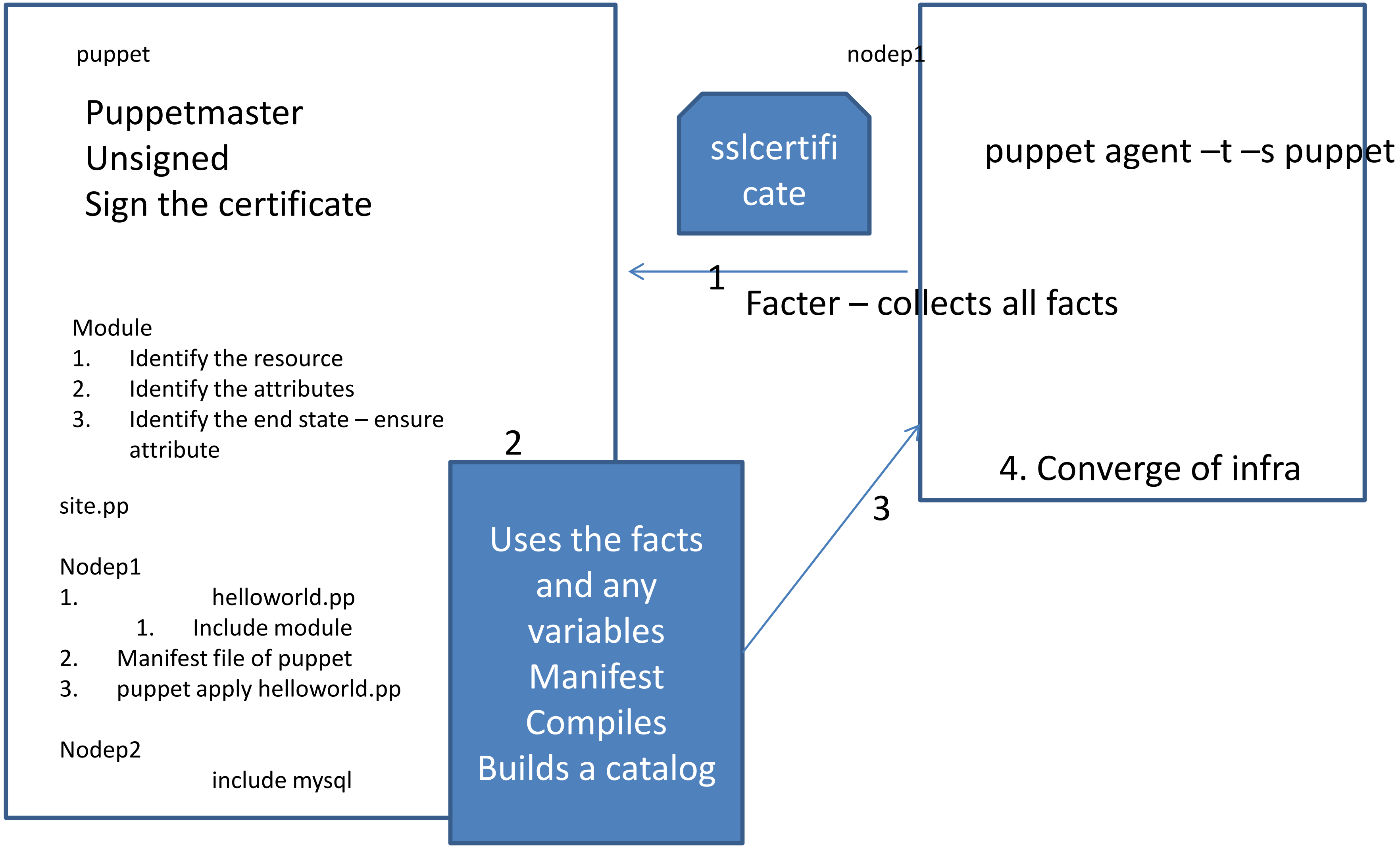
Vagrant and Virtual Box

Vagrant will help

- Bring up VMs with configuration
- Sets Network interfaces
- Run some scripts
- Memory
- shared folder setup

Our Setup



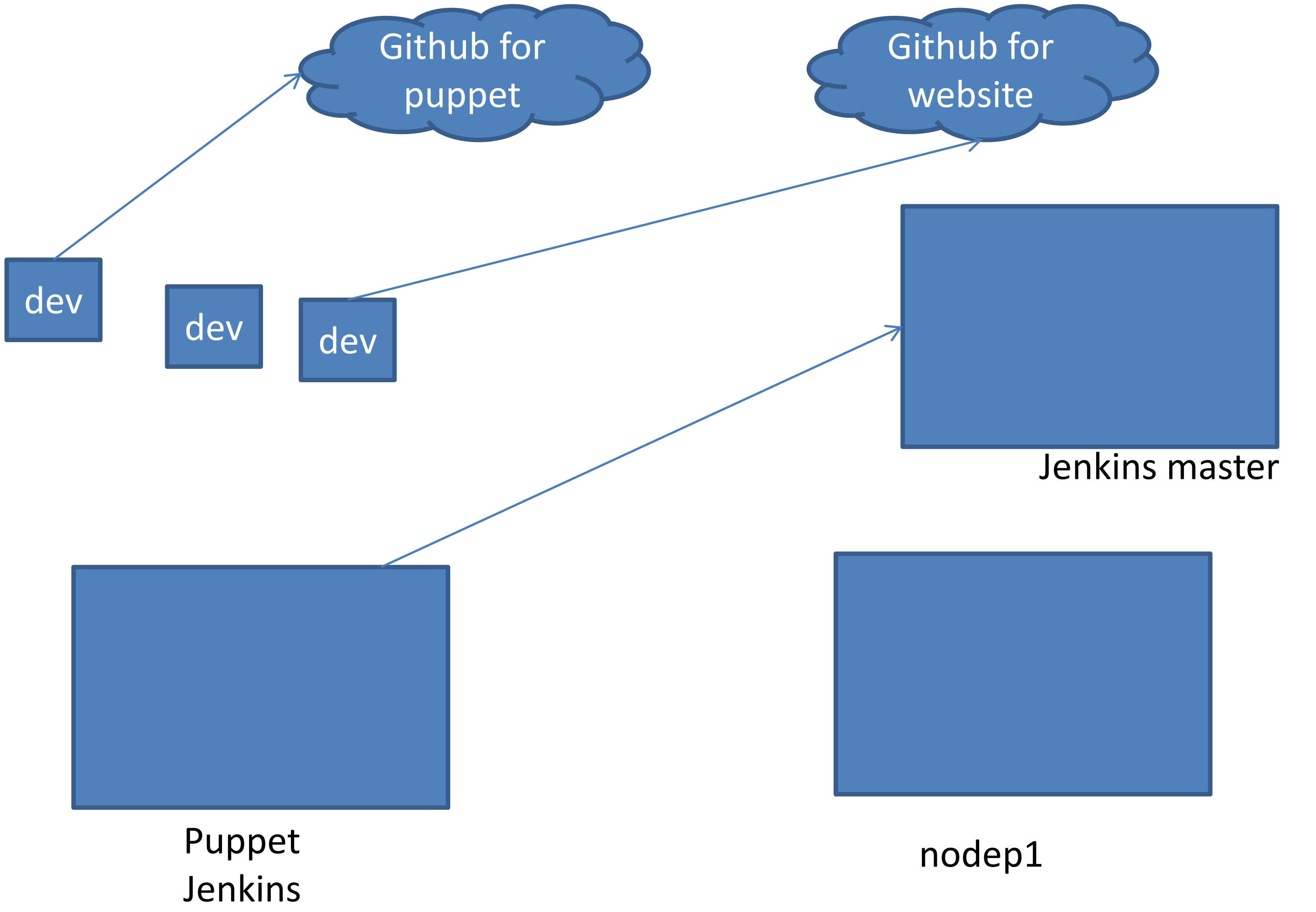


Module
1. Identify the resource
2. Identify the attributes
3. Identify the end state – ensure attribute

site.pp

Nodep1
1. helloworld.pp
    1. Include module
2. Manifest file of puppet
3. puppet apply helloworld.pp

Nodep2
include mysql


Github for puppet
Github for website
dev
dev
dev
Jenkins master
Puppet Jenkins
nodep1

Our Setup

Some vagrant commands

1. **vagrant up**
   Brings up all the nodes
2. **vagrant up nodep1**
   Brings up only nodep1
3. **vagrant halt**
   Stops all nodes
4. **vagrant halt nodep1**
   Stops only nodep1
5. **vagrant destroy**
   Destroys the VM. All changes will be lost

Understanding Puppet Components

Puppet master

Puppet Client

Puppet Types

A Type is the actual work horse that Puppet knows how to configure

- Files (content, permissions, ownership)
- Packages (ensure installed or absent)
- Services (enabled/disabled, running/stopped)
- Exec (run commands)

Types are used in manifest files

Writing a manifest file

```
root@nodep1:/home/vagrant# cat helloworld.pp
file {'/tmp/hello.txt':
        content => 'Hello World',
        ensure => present,
        mode => '0655',
        owner => 'vagrant',
        group => 'vagrant'
}
root@nodep1:/home/vagrant#
```

Running a manifest file

Use --noop for a dry run . Noop stands for no operation
The below output says that it **"would have"** created a file if we ran without --noop

```
root@nodep1:/home/vagrant# puppet apply helloworld.pp --noop
Notice: Compiled catalog for nodep1.belkin in environment production
in 0.06 seconds
Notice: /Stage[main]/Main/File[/tmp/hello.txt]/ensure: current_value
absent, should be present (noop)
Notice: Class[Main]: Would have triggered 'refresh' from 1 events
Notice: Stage[main]: Would have triggered 'refresh' from 1 events
Notice: Finished catalog run in 0.10 seconds
```

DevOps

Running a manifest file

An actual run

```
root@nodep1:/home/vagrant# puppet apply helloworld.pp
Notice: Compiled catalog for nodep1.belkin in environment production
in 0.06 seconds
Notice: /Stage[main]/Main/File[/tmp/hello.txt]/ensure: created
Notice: Finished catalog run in 0.10 seconds
root@nodep1:/home/vagrant# ls -ltra /tmp/hello.txt
-rw-r-xr-x 1 vagrant vagrant 11 Mar 15 01:11 /tmp/hello.txt
```

Modules

Modules are resuable components

Modules are self-contained bundles of code and data.
These reusable, shareable units of Puppet code are a basic building block for Puppet.

Creating a module

Using puppet given scaffolding to create a module
The module name should be <author>-<name of the module> as this is how its stored in puppet forge
We need to remote “sp-” from the module name while running the modules

```
root@nodep1:/home/vagrant/demo# puppet module generate sp-apache2
Notice: Generating module at /home/vagrant/demo/sp-apache2
sp-apache2
sp-apache2/Modulefile
sp-apache2/README
sp-apache2/manifests
sp-apache2/manifests/init.pp
sp-apache2/spec
sp-apache2/spec/spec_helper.rb
sp-apache2/tests
sp-apache2/tests/init.pp
root@nodep1:/home/vagrant/demo# mv sp-apache2 apache2
```

Adding resources to module

```
root@nodep1:/home/vagrant/demo# cat apache2/manifests/init.pp
class apache2 {
 package{'apache2':}
}
```

DevOps

Including the module in a manifest file

```
root@nodep1:/home/vagrant/demo# ls
apache2  installapache.pp
root@nodep1:/home/vagrant/demo# cat installapache.pp
include apache2

root@nodep1:/home/vagrant/demo#
```

DevOps

Run the manifest installapache.pp

In the below output it empty as apache2 was already installed on nodep1

```
root@nodep1:/home/vagrant/demo# puppet apply --modulepath
/home/vagrant/demo installapache.pp
Notice: Compiled catalog for nodep1.belkin in environment production
in 0.01 seconds
Notice: Finished catalog run in 0.09 seconds
root@nodep1:/home/vagrant/demo#
```

DevOps

Creating a module

Using puppet given scaffolding to create a module
The module name should be <author>-<name of the module> as this is how its stored in puppet forge
We need to remote "sp-" from the module name while running the modules

```
root@nodep1:/home/vagrant/demo# puppet module generate sp-apache2
Notice: Generating module at /home/vagrant/demo/sp-apache2
sp-apache2
sp-apache2/Modulefile
sp-apache2/README
sp-apache2/manifests
sp-apache2/manifests/init.pp
sp-apache2/spec
sp-apache2/spec/spec_helper.rb
sp-apache2/tests
sp-apache2/tests/init.pp
```

Facter

Facter gathers information about the client, which can be used as variables within puppet.

- You can add custom facts as needed.

Class

- A named collection of type objects
- Can include or inherit from other classes

```
class sudo_class {
        include foo_class file { “/etc/sudoers”: ... }
        package{ “sudo”: ... }
}
```

Class Inheritance

```
class afile {
        file { “/tmp/foo”: ensure => file source =>
“/src/versionA” }
        }
class another_file inherits afile {
        File[“/tmp/foo”] { source => “/src/versionB” }
}
```

Node

A configuration block matching a client

- Can contain types, classes
- “default” node matches any client without a node block

```
node “ohad.myself” {
            include sudo_class
            include other_class
}
```