What is Puppet?

Puppet Example

```
package { "mysql-server":
    ensure => installed,
}
```

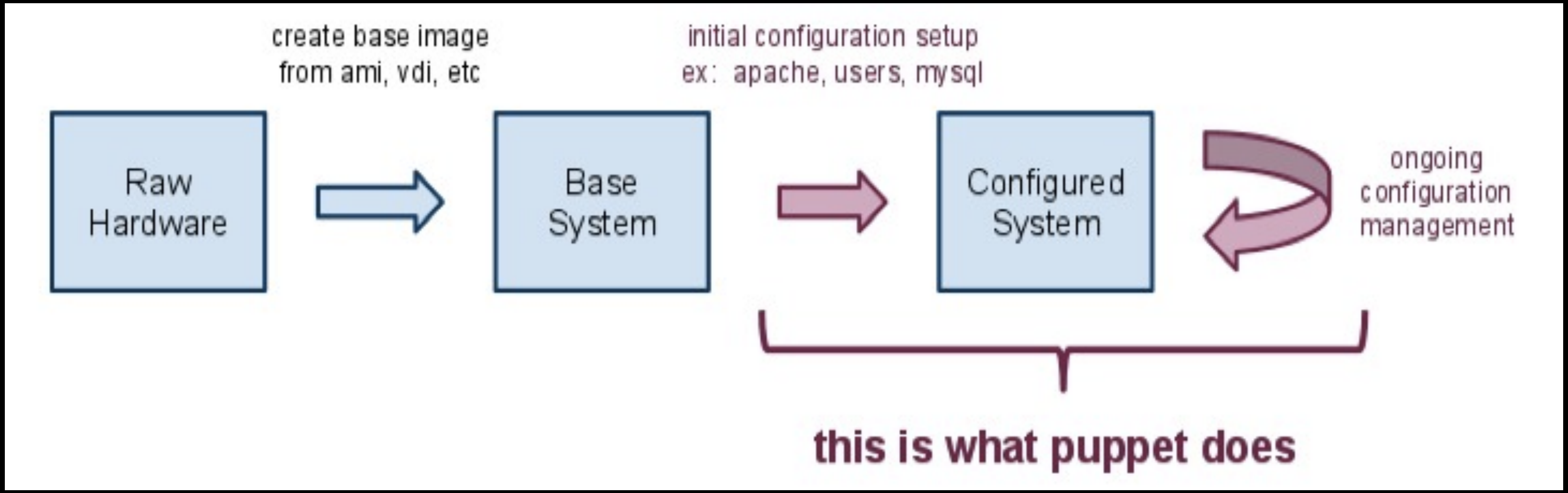
Puppet Example

```
package { "mysql-server":
    ensure => installed,
}
```

compare with

```
sudo apt-get install mysql-server
```

Configuration Management vs Provisioning



Let’s start from zero

- vagrant destroy db
- vagrant destroy web

host

Vagrant Provisioning

- Vagrant supports configuring a VM with Puppet, Chef, and others, automatically
  - They call it “provision” but it’s really “configure”
- We’ll do this so we can code in our normal editor, then invoke Vagrant to configure the node.

Configure Vagrant

- Create a directory called **manifests** that is parallel to your Vagrantfile.
- In Vagrantfile add a Puppet provisioner as shown below

```
config.vm.define :db do |my|
  my.vm.network :private_network, ip: "172.16.1.11"
  my.vm.hostname = "db"
    ...
  my.vm.provision :puppet do |puppet|
    puppet.manifest_file = "db.pp"
  end
end
```

source

pro tip

Both Vagrantfile and Puppet files are Ruby-based DSLs. Configure your editor for Ruby for syntax highlighting. :-)

Create manifest

create a file called **db.pp** in the manifests directory with the following content:

```
package { "mysql-server":
    ensure => installed,
}
```

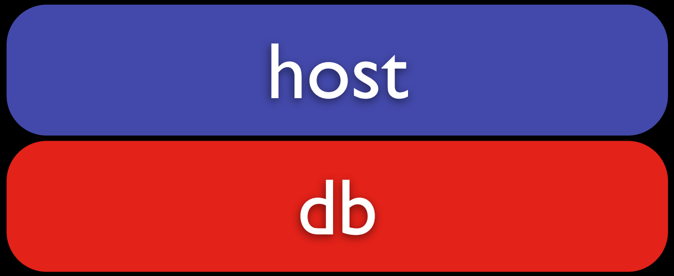
source

Apply the Manifest

- vagrant up db
- Did you see the Puppet messages?

verify that MySQL is installed with:

```
aptitude show mysql-server
```



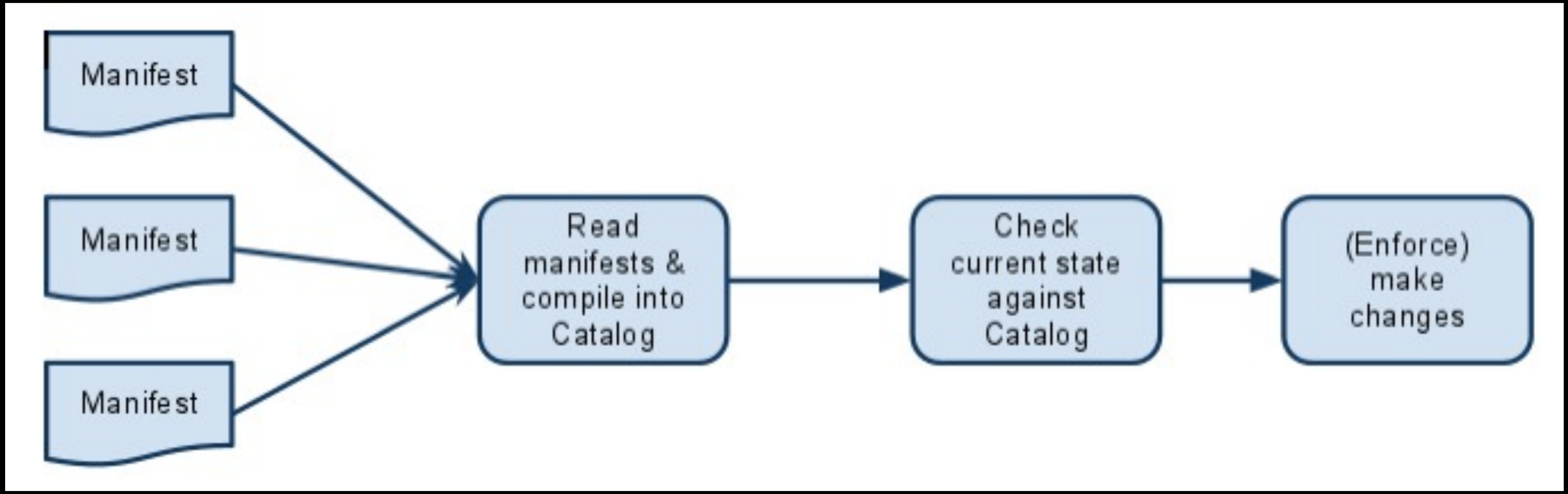
apt-get update?

```
stage { 'preinstall': before => Stage['main'] }

class apt_get_update {
  exec { '/usr/bin/apt-get -y update':
    user => 'root'
  }
}

class { 'apt_get_update': stage => preinstall }
```

How puppet works


Manifest
Manifest
Manifest
Read manifests & compile into Catalog
Check current state against Catalog
(Enforce) make changes

Puppet Resources

```
resource type : package
resource name : mysql-server
```

to see current system state try:

```
puppet resource package mysql-server
```

```
package { 'mysql-server':
    ensure => '5.1.54-1ubuntu4'
}
```

db

Resource Examples

```
puppet resource package svn
```

```
package { 'svn':
    ensure => 'purged'
}
```

```
puppet resource user
```

db

Core Resource Types

file

package

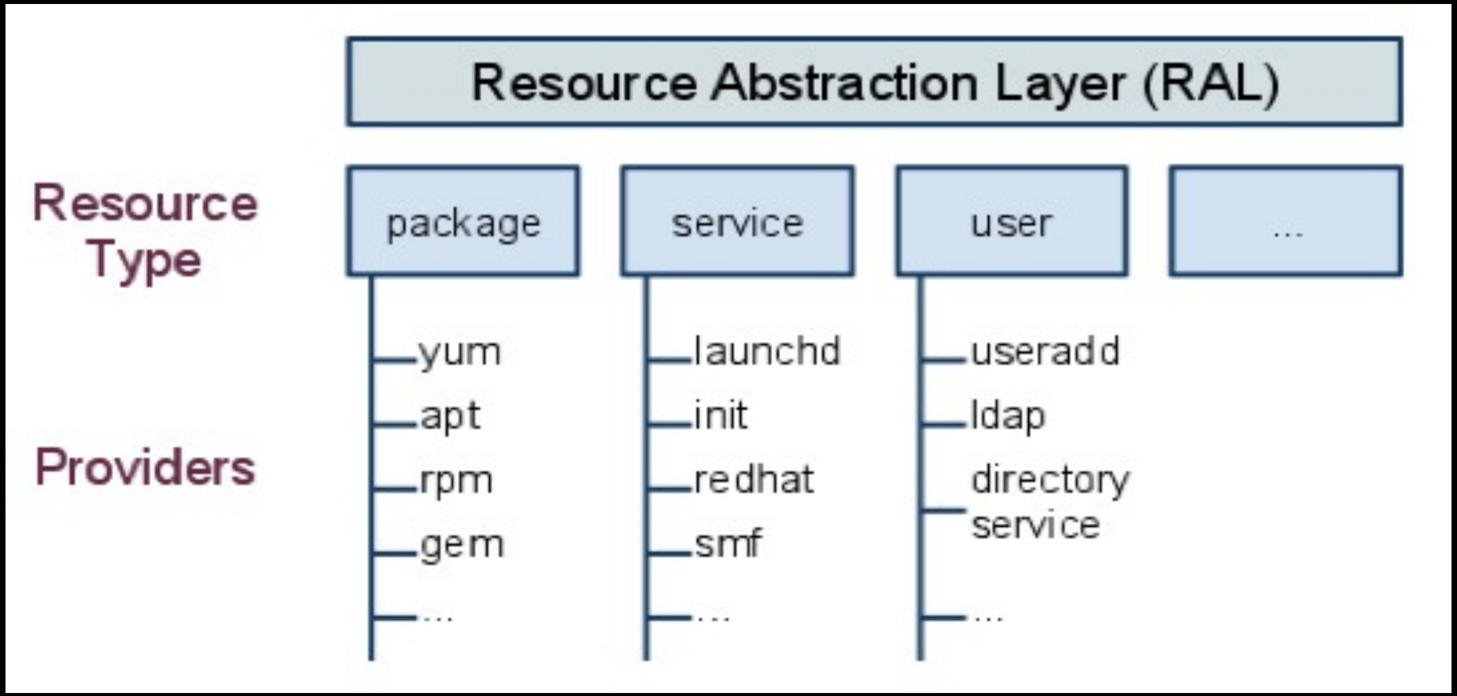
service

users / groups

cron

exec (for everything else)

Puppet providers



Configure MySQL

add following to **db.pp**:

```
file {"/etc/mysql/conf.d/allow_external.cnf":
      owner   => mysql,
      group   => mysql,
      mode    => 0644,
      content => "[mysqld]\n   bind-address = 0.0.0.0",
}
```

then apply the manifest

- *vagrant* ***provision*** *db*

source

host

Backups

puppet automatically stores back-ups of modified files

the md5 checksum is a unique identifier of the backup

Templates

use ERB (built-in to Ruby)

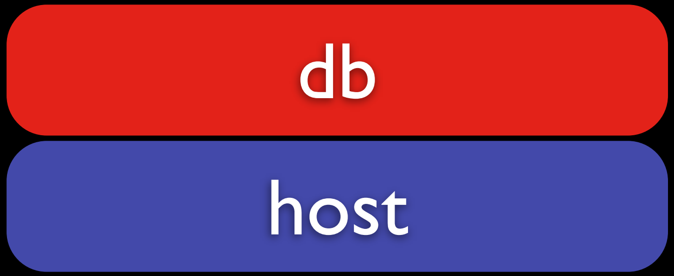
instead of inlining content in manifest:

```
content => template(“/vagrant/allow_external.cnf”)
```

```
cp /etc/mysql/conf.d/allow_external.cnf /vagrant/allow_external.cnf
```

```
vagrant provision db
```

and confirm it worked



Dependencies

order in manifest does not matter

puppet might apply resources randomly

specify dependencies explicitly with:

```
require => Package[“mysql-server”]
```

Now change ‘file’ in db.pp to depend on MySQL package as above and *vagrant provision db*

source

Services

check if mysql is running:

```
sudo service mysql status
```

db

Services

puppet can manage running services

makes sure they are running

makes sure they start on boot

can restart them if things change

Define MySQL Service

implement a puppet service for mysql

- always running
- start on boot

http://j.mp/puppet-service

source

Refreshing Services

restart services when configs change

```
notify => Service[“mysql”]
```

or

```
subscribe => File[“cfgfile”]
```

source

apply puppet - what happens?

change bind-address back to 0.0.0.0

apply puppet

db

What color is your bar?

Create app DB & user

Back to MDD:

Let’s do a richer check of MySQL and confirm we can log in

check_mysql_database

/etc/nagios-plugins/config/mysql.cfg

```
define command {
    command_name  check_mysql_database
    command_line  /usr/lib/nagios/plugins/check_mysql
        -d '$ARG3$' -H '$HOSTADDRESS$' -u '$ARG1$' -p '$ARG2$'
}
```

monitor

Execute by hand

```
/usr/lib/nagios/plugins/check_mysql
      -d opencart -H db
      -u opencart -p openpass
```

monitor

Add MySQL service

```
/etc/nagios3/conf.d/opencart.cfg
```

monitor

Ok, let's create the DB and user

Procedurally ....

```
create database opencart;
```

database

```
grant all on opencart.*
```

username

```
to 'opencart'@'%'
identified by 'openpass';
```

password

Executing commands

can use the `exec` resource

making sure it is **idempotent**:

- unless specifies guard for execution
- creates specifies resulting files

Sample exec statement

```
exec { "some-name":
  unless => "/usr/bin/foo that returns 0 or 1",
  onlyif => “/usr/bin/foo that returns 0 or 1”,
  command => "/usr/bin/bar that does something",
  require => Service["baz"],
}
```

unless runs on 1, ***onlyif*** runs on 0

MySQL database

create: `mysqladmin -uroot create opencart`

confirm: `mysqlshow -uroot opencart`

source

Create DB user...

```
grant all on opencart.*
to 'opencart'@'%'
identified by 'openpass';
```

http://j.mp/puppet-exec

What color is your bar?

Extra credit...

use puppet to set MySQL root password

extract passwords out of manifests

Puppet Organization

Classes

singleton collections of resources

can be applied (or not) as a unit

have nothing to do with OO classes

think of them like “roles” or “aspects”

Using Classes

define a class

```
class foo {
  package {...}
  file {...}
}
```

use a class

1. `include foo`

2. `class { "foo": }` (like defining a resource)

Re-factor db.pp

Wrap generic MySQL resources in a class

- package
- file
- service

source

Defined Types

re-usable collection of resources

can be used multiple times in a manifest

analogous to macros in other languages

used like native Puppet resources

Using Defined Types

define the type

```
define mytype($arg1, $arg2) {
 exec …
 exec …
}
```

declare resources of this type

```
mytype { “name”:
   arg1 => “val1”,
   arg2 => “val2”,
}
```

Refactor db.pp

extract creating database and user

use a defined type with parameters

- database name
- username
- password

source

Modules

self-contained collections of

- manifests
- templates
- other (files, plugins, tests)

Puppet searches for them in modulepath

Using modules

```
.
├── Vagrantfile
├── manifests
│   └── db.pp
├── modules
│   └── database  <--- Module Name
│       ├── manifests
│       │   ├── appdb.pp
│       │   ├── init.pp
│       │   └── mysql.pp
│       └── templates
│           └── etc
│               └── mysql
│                   └── conf.d
│                       └── allow_external.cnf
└── setup_node.sh
```

host

Configure Vagrant

- In Vagrantfile add a module path definition

```
config.vm.define :db do |my|
  my.vm.network :private_network, ip: "172.16.1.11"
  my.vm.hostname = "db"

  my.vm.provision :puppet do |puppet|
    puppet.manifest_file = "db.pp"
    puppet.module_path = "modules"
  end
end
```

- We'll create the modules directory next...

source

Creating base structure

```
mkdir -p modules/database/{manifests,templates}
mkdir -p modules/database/templates/etc/mysql/conf.d
touch modules/database/manifests/{init,mysql,appdb}.pp
```

```
create allow_external.cnf in modules/database/
templates/etc/mysql/conf.d
```

```
[mysqld]
    bind-address = 0.0.0.0
```

host

move mysql class definition to mysql.pp

move appdb defined type to appdb.pp

leave use of appdb in db.pp

init.pp

```
import "mysql"
import "appdb"
```

source

In mysql.pp switch allow_external.cnf to to module qualified template

```
content => template("database/etc/mysql/conf.d/
allow_external.cnf")
```

modulename/fully/qualified/path/in/templates/dir

source

Add following to tell db.pp to use the module:

```
import "database"
```

Apply the manifest:

```
vagrant reload db
```

* we are forced to use reload on the first apply because Vagrant needs to mount the module path in the VM

source

Test from scratch

```
vagrant destroy db

vagrant up db
```

host

Configure Web Node

What can we test for the web node?

How about accepting http requests with check_http?

Conveniently we have already set this up

Confirm it is still failing in Nagios

To install OpenCart

- Install Apache, PHP & dependencies
- Download & unzip OpenCart distribution
- Move upload to /var/www/
- Configure config.php
- Create database schema
- Remove /install
- Set permissions

Start with packages

needed packages:

apache2 php5 mysql-client

```
php5-mysql php5-gd php5-curl
```

php5 depends on apache2

```
php5-mysql php5-gd php5-curl depend on php5
```

create a file called **web.pp** in the manifests directory with the following content:

```
package{ "apache2":
    ensure => installed
}

package { "php5":
    ensure => installed,
    require => Package["apache2"],
}

package { [‘php5-mysql’, ‘php5-gd’, ‘php5-curl’]:
    ensure => installed,
    require => Package["php5"],
}
```

source

Variables, arrays, hashes

Variables

```
$v = "/root/thefile.txt"
```

Arrays

```
$a = ['a', 'b', 'c']
```

Hashes (maps / dictionaries)

```
$h = {k1 => 'v1', k2 => 'v2'}
```

Configure Apache

Create Apache service

new php package requires restarting apache

Configure Vagrant

- In Vagrantfile add a Puppet provisioner as shown below

```
config.vm.define :web do |my|
  my.vm.network :private_network, ip: "172.16.1.10"
  my.vm.hostname = "web"
  ...
  my.vm.provision :puppet do |puppet|
    puppet.manifest_file = "web.pp"
    puppet.module_path = "modules"
    puppet.options = "--verbose --debug"
  end
end
```

source

vagrant up web

host

What color is your bar?

Monitoring using Functional Tests

“Semantic Monitoring”

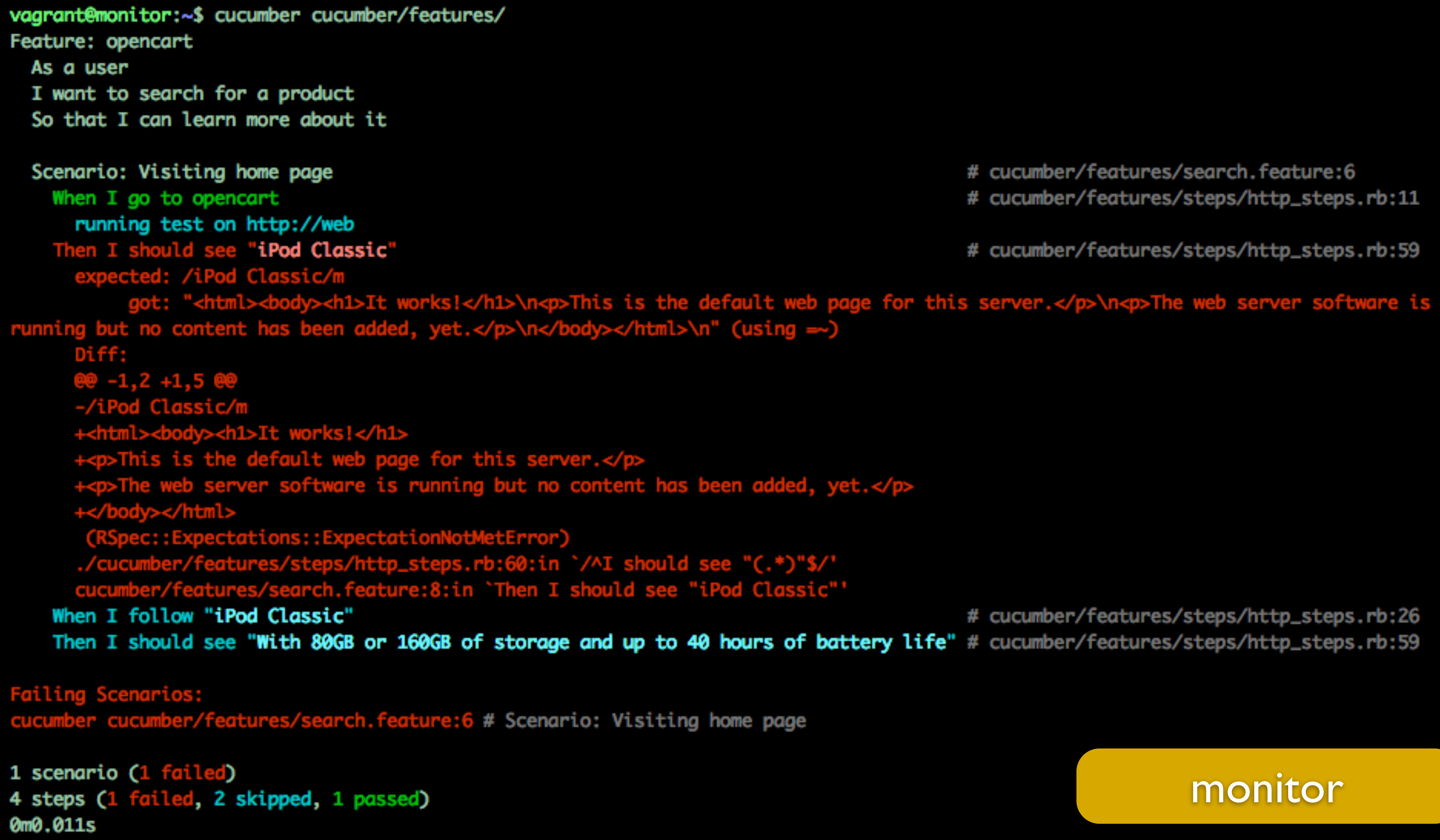
cucumber

```
Feature: Opencart Search
  As a user
  I want to search for a product
  So that I can learn more about it

  Scenario: Visiting home page
    When I go to "http://web"
    Then I should see "Apple Cinema 30"
    When I follow "Apple Cinema 30"
    Then I should see "The 30-inch Apple Cinema"
```

Run cucumber test by hand

cucumber cucumber/features/search.feature



```
vagrant@monitor:~$ cucumber cucumber/features/
Feature: opencart
  As a user
  I want to search for a product
  So that I can learn more about it

  Scenario: Visiting home page                                         # cucumber/features/search.feature:6
    When I go to opencart                                              # cucumber/features/steps/http_steps.rb:11
      running test on http://web
    Then I should see "iPod Classic"                                   # cucumber/features/steps/http_steps.rb:59
      expected: /iPod Classic/m
           got: "<html><body><h1>It works!</h1>\n<p>This is the default web page for this server.</p>\n<p>The web server software is
running but no content has been added, yet.</p>\n</body></html>\n" (using =~)
      Diff:
      @@ -1,2 +1,5 @@
      -/iPod Classic/m
      +<html><body><h1>It works!</h1>
      +<p>This is the default web page for this server.</p>
      +<p>The web server software is running but no content has been added, yet.</p>
      +</body></html>
       (RSpec::Expectations::ExpectationNotMetError)
      ./cucumber/features/steps/http_steps.rb:60:in `/^I should see "(.*)"$/'
      cucumber/features/search.feature:8:in `Then I should see "iPod Classic"'
    When I follow "iPod Classic"                                       # cucumber/features/steps/http_steps.rb:26
    Then I should see "With 80GB or 160GB of storage and up to 40 hours of battery life" # cucumber/features/steps/http_steps.rb:59

Failing Scenarios:
cucumber cucumber/features/search.feature:6 # Scenario: Visiting home page

1 scenario (1 failed)
4 steps (1 failed, 2 skipped, 1 passed)
0m0.011s
```

monitor

Run test with cucumber-nagios

```
cucumber-nagios
    cucumber/features/search.feature
```

```
CUCUMBER CRITICAL - Critical: 1, Warning: 0, 1 okay | passed=1; failed=1;
nosteps=0; total=2; time=0

Failed: Then I should see "iPod Classic" in cucumber/features/
search.feature:6 on cucumber/features/steps/http_steps.rb:59
```

monitor

Add cucumber nagios service

First, define the check_cucumber command in /*etc/nagios3/conf.d/cucumber-nagios.cfg*:

```
define command{
   command_name   check_cucumber
   command_line   cucumber-nagios /home/vagrant/cucumber/features/search.feature
}
```

monitor

How $HOSTADDRESS$ got passed to cucumber

```
path = ENV['NAGIOS_HOSTADDRESS'] || "web"
```

Nagios sets all of its standard arguments ($HOSTADDRESS$, etc) as env variables with the prefix NAGIOS_

|| "*web*" is a hack to let it work when you tested it from the command line

Let’s fix the test

Configure Web Nodes continued

To install OpenCart

- wget http://bit.ly/opencart-zip
- unzip opencart-zip
- sudo rm -rf /var/www/*
- sudo mv upload/* /var/www
- sudo chown -R www-data /var/www

exec as antipattern

puppet isn't very natural for the opencart install process. using lots of execs is considered a smell in puppet.

how else could we do this?

Install OpenCart package

```
wget http://j.mp/opencart-deb -O opencart.deb
```

install package with:

```
provider => dpkg,
source => "/path/to/opencart.deb"
```

note this installs the package in /var/opencart

web

source

Enable opencart site

disable default apache site by deleting

/etc/apache2/sites-enabled/000-default

enable opencart by symlinking

/etc/apache2/sites-enabled/opencart to

/etc/apache2/sites-available/opencart

source

Template Syntax

Variables

```
<%= variable %>
```

Loops

```
<% values.each do |val| -%>

<% end -%>
```

Conditionals

```
<% if foo != “BAR” %>
```

config.php

- Manually copy /var/www/config.php from web to /vagrant/config.php
- Edit config.php and change all
  - "/var/www" to "/var/opencart"
  - "http://web/" to "/"
- Using puppet place config.php in /var/opencart/config.php, substituting values for db host, user, password & name
  - template syntax: `<%= variable %>`

web

source

vagrant provision web

Then browse to http://web and check if opencart is running

host

DDL & DML

Save http://j.mp/opencart-sql into */vagrant/opencart.sql*

Load using mysql in db.pp

source

*vagrant provision **db***

Then browse to http://web and check if opencart is running

host

What color is your bar?

refactor to classes, defined types and module

Participant Demo