Continuously integrating Perl projects with Vagrant

Mike Schilli, Venkat Venkataraju
Yahoo!

YAPC 2013, Austin, TX

06/04/2013

Summary

- Unit Tests vs. Integration
- Infrastructure as Code
- What’s Vagrant?
- Provisioning Tools (Puppet, Chef, Salt)
- Example and Demo

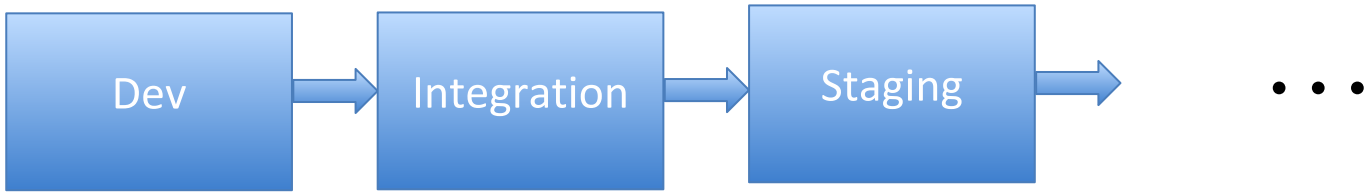
Unit Tests

```
$ perl Makefile.PL
$ make
$ make test
t/Unit.t .. ok
t/Live.t .. skipped: only with LIVE
All tests successful.
Files=2, Tests=2,  0 wallclock secs (
0.03 usr  0.01 sys +  0.03 cusr  0.00
csys =  0.07 CPU)
Result: PASS
```

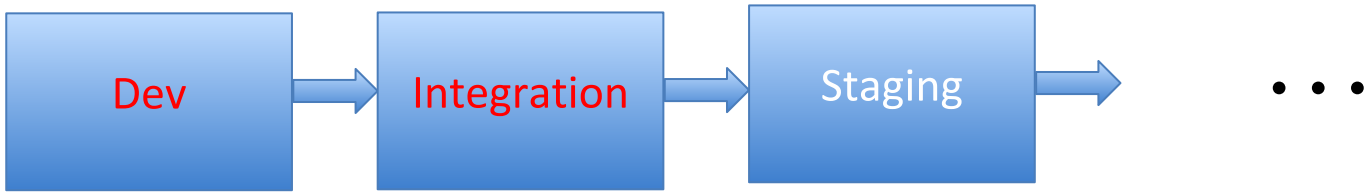
Unit Test Limitations

- “Works for me” syndrome
- Tied to development box setup
- But what about …
  - A different OS?
  - Interaction with a database?
  - Other people’s code?

CI Pipeline



CI Pipeline



Integration Tests

- Full application stack installed
- Integrate early, integrate often, integrate continuously
- Problem: Can’t configure your devbox for many different project environments

What you need to integrate with

- New supporting package releases
- Other people’s code
- Config changes
- Services
- External Systems

Virtual Machines

- Easy to spin up on many platforms to set up integration environments
- But setting up the application stack
  - is manual work
  - needs to be documented
  - slows down potential contributors

Infrastructure as Code

- Store machine setup in source control
- One-command spin-up
  - Creates VM
  - Provisions VM based on checked-in file
    - Packages
    - Configuration files
    - Start Services
    - Create user account

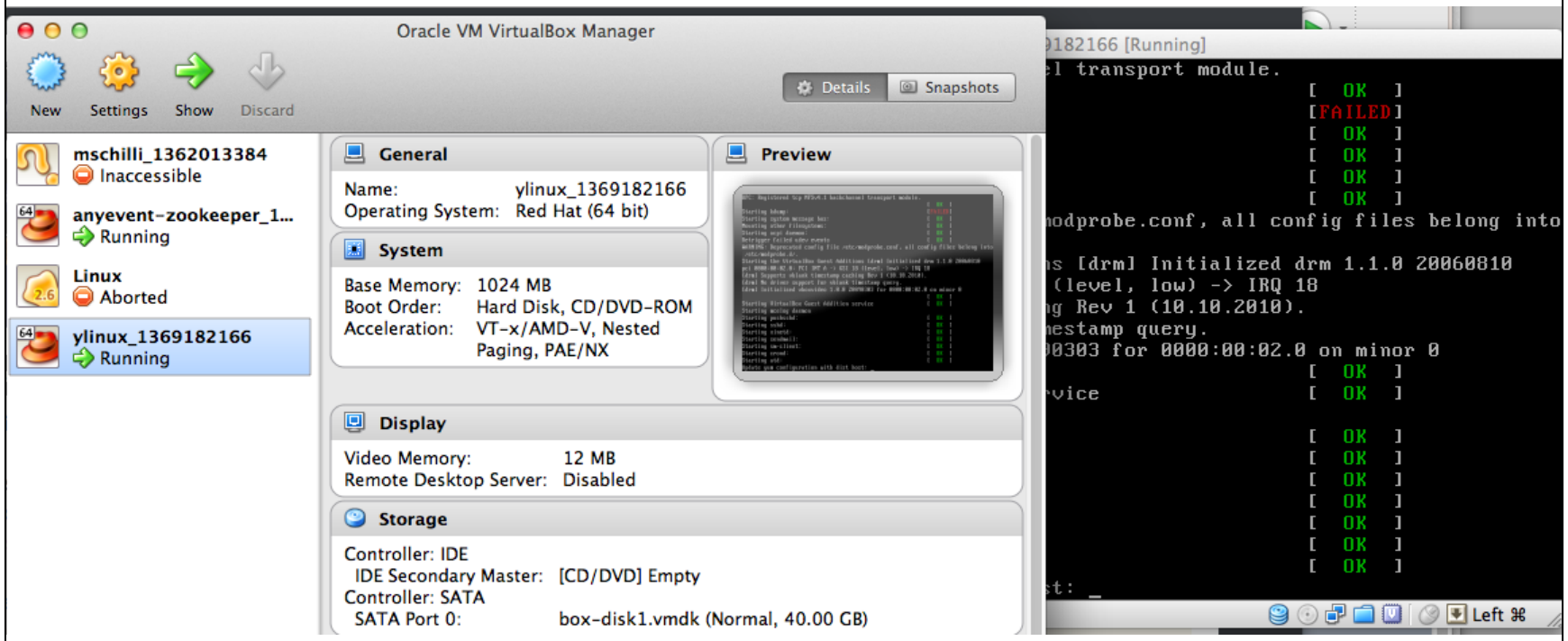
Infrastructure as code - why?

- Restore Infrastructure from scratch after catastrophic incidents
- The New Guy doesn’t need to read documentation to get started
- Get a working development environment back right away after coming back to a project

VirtualBox and Vagrant

+

On top of VirtualBox



```
$ vagrant up
```

Vagrantfile

Vagrantfile

```
Vagrant::Config.run do |config|
  config.vm.box = "ubuntu2"
  config.vm.box_url = "http://files.vagrantup.com/precise64.
box"
end
```

Vagrant Boxes For Download

www.vagrantbox.es

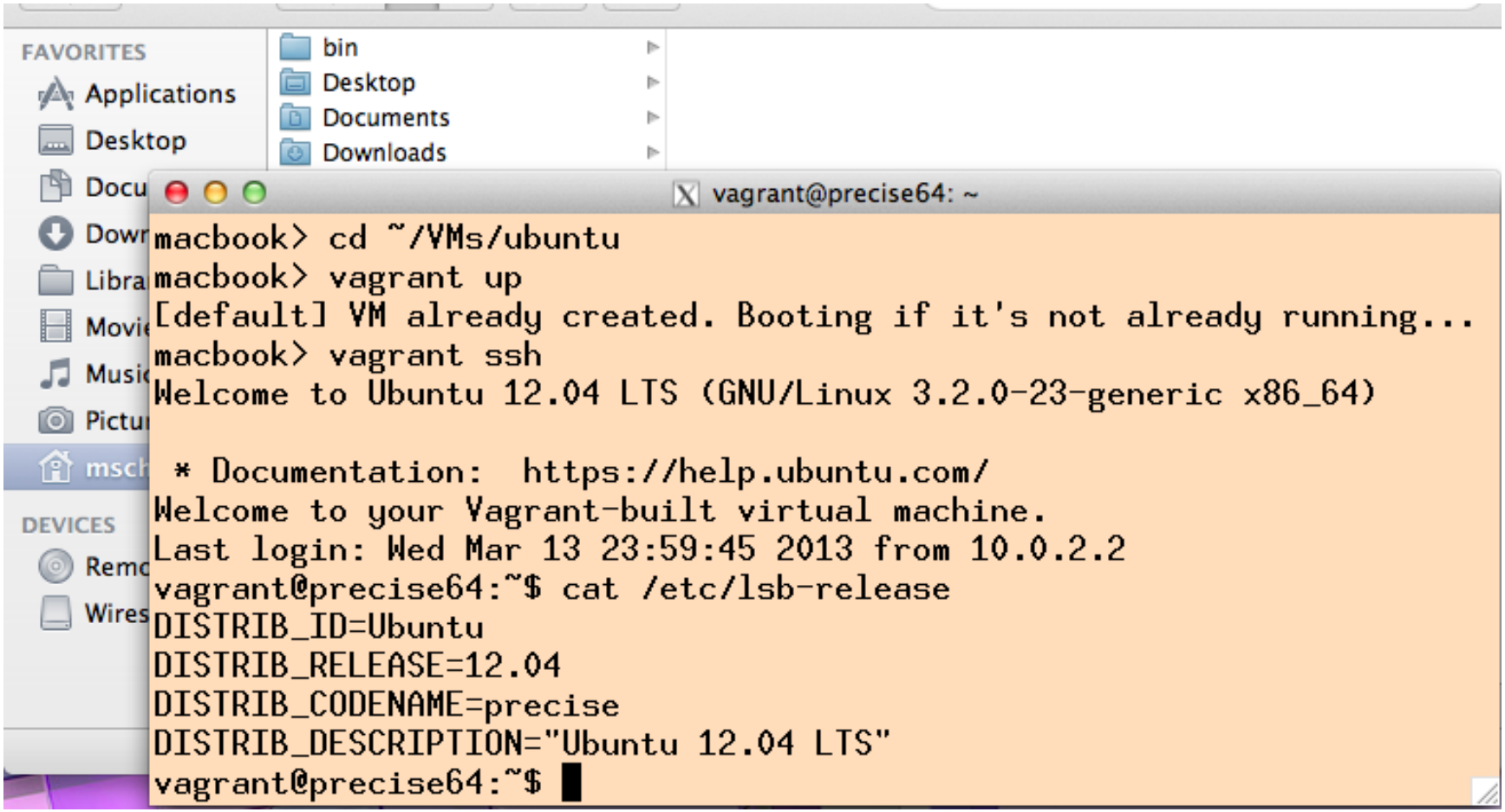
Lustig ist das Zigeu

Debian squeeze 64	http://dl.dr
Debian Squeeze 64 with Puppet 2.7	http://andr
Debian Squeeze amd64 (with Puppet, Chef and VirtualBox 4.2.1)	http://goo.
Debian Squeeze amd64 (with VirtualBox 4.2.4)	http://dl.dr
Debian Wheezy amd64 minimal (base install, no puppet, no chef, no ruby, no gems, just an install')	https://gith
Fedora 17 i386 (Puppet, Chef, VirtualBox 4.2.6)	http://dl.dr
Fedora 17 x86_64 (Puppet, Chef, VirtualBox 4.2.6)	http://dl.dr
Fedora 18 x86_64 Minimal (with Puppet, VirtualBox 4.2.6 and rpmfusion yum repo)	https://dl.c
FreeBSD 9.1 amd64 - UFS (Puppet, Chef, VirtualBox 4.2.6)	https://s3.
FreeBSD 9.1 amd64 - ZFS (Puppet, Chef, VirtualBox 4.2.6)	https://s3.
Gentoo 2013.01.30 amd64 (Puppet, Chef, VirtualBox 4.2.6)	http://dl.gu
Gentoo 3.3.8 i686	http://dl.dr
Heroku Celadon Cedar	http://dl.dr
Minimal CentOS 5.6	http://dl.dr
Minimal CentOS 6.0	http://dl.dr
Official Ubuntu 12.04 daily Cloud Image amd64 (Guest Additions)	http://clou
Official Ubuntu 12.04 daily Cloud Image i386 (Guest Additions)	http://clou

Box Install

```
$ mkdir myubuntu
$ cd myubuntu
$ /opt/vagrant/bin/vagrant box add precise32 http://files.vagrantu
p.com/precise32.box
$ vagrant init precise32
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
$ vagrant up
[default] Importing base box 'precise32'...
[default] The guest additions on this VM do not match the install
version of
VirtualBox! This may cause things such as forwarded ports, shared
folders, and more to not work properly. If any of those things fai
l on
this machine, please update the guest additions and repackage the
box.
Guest Additions Version: 4.2.0
VirtualBox Version: 4.2.8
[default] Matching MAC address for NAT networking...
[default] Clearing any previously set forwarded ports...
[default] Forwarding ports...
[default] -- 22 => 2222 (adapter 1)
[default] Creating shared folders metadata...
[default] Clearing any previously set network interfaces...
[default] Booting VM...
[default] Waiting for VM to boot. This can take a few minutes.
$ vagrant ssh
vagrant@precise32:~$
```

Vagrant: Remote-controls VirtualBox



Magic Mount

- /vagrant in the VM maps to the start directory on the host system.

Networking – Simple

```
# Forward a port from the guest to the host,
# computers to access the VM, whereas host c
config.vm.forward_port 8080, 8080
```

Integration Tests

```
WriteMakefile(
    NAME                => 'Foo',
    VERSION_FROM        => 'lib/Foo.pm', # finds $VERSION
    PREREQ_PM           => {}, # e.g., Module::Name => 1.1
    ($] >= 5.005 ?       ## Add these new keywords supported since 5.005
      (ABSTRACT_FROM    => 'lib/Foo.pm', # retrieve abstract from module
       AUTHOR           => 'Mike Schilli <m@perlmeister.com>') : ()),
);

sub MY::postamble {
    "livetest:\n" .
    "\tvagrant ssh -c " .
    "'cd /vagrant; perl Makefile.PL; LIVE=1 make test'" ;
}
~
```

Integration Test Suite

```
use strict;
use warnings;

BEGIN {
    use Test::More;

    if( $ENV{ LIVE } ) {
        plan tests => 1;
    } else {
        plan "skip_all", "only with LIVE";
    }
}

ok( 1, "live passes" );
```

Unit Test

```
$ make test
t/Unit.t .. ok
t/Live.t .. skipped: only with LIVE
All tests successful.
```

Live Test

```
$ make livetest
vagrant ssh –c ‘cd /vagrant; perl Makefile.PL;
make test’
```

Live Test

```
$ make livetest
vagrant ssh –c ‘cd /vagrant; …’
t/Live.t .. ok
t/Unit.t .. ok
```

```
$ cd /vagrant
$ perl Makefile.PL
$ LIVE=1 make test
…
```

Infrastructure As Code

Cleanup

```
$ vagrant destroy
Are you sure you want to destroy the 'default' VM? [Y/N]
```

Provisioning Tools

Tool	Client	Server	Config Format
Puppet (Puppet Labs)	puppet-agent	Puppet master	DSL
Chef (Opscode)	chef-client & knife	Chef solo, server or hosted	Ruby
Salt Stack (saltstack.com)	Salt minion	Salt server	Python + YAML

…

Provisioning with Puppet

```
# Basic Puppet Mojo manifest

class mojo {
  exec { 'apt-get update':
    command => '/usr/bin/apt-get update'
  }

  package { "libmojolicious-perl":
    ensure => present
  }

  file { '/usr/bin/mymojo':
    ensure => link,
    target => "/vagrant/mymojo",
    force  => true
  }
}

include mojo
```

Provisioning with Puppet

```
#
config.vm.provision :puppet do |puppet|
  puppet.manifests_path = "manifests"
  puppet.manifest_file  = "ubuntu2.pp"
end
```

Provisioning with Puppet

```
[default] Running provisioner: Vagrant::Provisioners::Puppet...
[default] Running Puppet with /tmp/vagrant-puppet/manifests/ubuntu2.pp
```

Chef, cookbooks and recipes

- Chef - Infrastructure automation framework
- Cookbooks/Recipe - Abstract definition for installing package(s) and configuring them
- Recipes contain resource primitives that enable us to define any workflow

Little more about resources

```
package "name" do
   action :install
end

execute "Starting service" do
   command "svc -u /service/name"
end

cookbook_file "/path/to/file" do
   source "/path/to/source/file"
end
```

Test kitchen

- Integration test harness for Chef
- Abstracts creation and provisioning of VMs
- Manage multiple VMs
- Easy to write YAML file
- Beta software

Kitchen config (.kitchen.yml)

```yaml
---
driver_plugin: vagrant
driver_config:
   require_chef_omnibus: true
platforms:
- name: centos-6.4
   driver_config:
      box: opscode-centos-6.4
suites:
- name: default
   run_list:
   - "recipe[yapc2013-cookbook::default]"
    attributes: {}
```

Test Kitchen (cont.)

Supports

- **`kitchen create`** – creates VMs
- **`kitchen converge`** – provisions VMs
- **`kitchen verify`** – runs test suite
- **`kitchen destroy`** – deletes VMs

Bats

- Bash automated testing system
- TAP compliant test framework written in Bash
- Can test any scripts or programs that run in Bash shell

Simple Bats test suite

```
#!/usr/bin/env bats

@test "Running make test" {
  make -C /path/to/Makefile/ test
}
```

Demo

- Creates "centos-6.4" virtual machine
- Installs "perl"
- Installs cpan modules "Plack" and "Dancer"
- Runs verification tests

References

- Slides: https://github.com/mschilli/yapc2013
- Demo code: https://github.com/mschilli/yapc2013/tree/master/yapc2013-cookbook
- Michael Hütterman, “DevOps for Developers”, 2012
- Stephen Nelson-Smith, “Test-driven Infrastructure with Chef”, 2011
- Puppetlabs.com
- Saltstack.com
- http://www.opscode.com/chef/
- Facebook & Chef Talk: http://www.youtube.com/watch?v=SYZ2GzYAw_Q
- Chef Docs: http://docs.opscode.com/
- Test-Kitchen: https://github.com/opscode/test-kitchen
- Bash automated testing system: https://github.com/sstephenson/bats

The End

- Q&A