Servers So Easy A Caveman Can Do It

Christopher H. Laco » claco@chrislaco.com » @claco » #clerb

Follow along! http://chrislaco.com/slides/clerb-caveman.pdf

Shout out to our sponsors

leandog.com

Within3

within3.com

Your humble speaker

- Reformed Music Major Turned Nerd
- H.A.H.S.
- **H**osting **A**t **H**ome **S**yndrom
- S.I.B.D.
- **S**ervers **I**n **B**asement **D**isease
- Hardware/Software/Network
- 3 Months DevOps Free. Just a programmer now.

  Also .NET Free since June!

Is Of DevOps Kind For Hire Of Cloud Works!

Why are we here?

- Heard the word “easy”. Sorry. I lied. Servers are hard.
- Food / Drink
- Thought this was a CleAG night
- Hiding from the authorities
- You’re the “server guy/gal/victim”
- Hate working on servers
- Looking for ideas to automate your pain away

Motivations

- Configuring servers requires a “server guy”
- Testing locally is different than testing upstream
- Adding capacity takes time
- Upgrades introduce risk
- Changing deployment logic is troublesome
- Troubleshooting production is risky/difficult
- Disaster recovery is costly/long
- True “Staging” environments are difficult

Motivations continued...

- “Works on my machine” is dangerous
- Nothing is repeatable
- Managing multiple servers is tedious
- MBP Setup is different

Goals

- Any Engineer/QA can spin up machines
- Same configuration everywhere
- Add more servers when traffic increases
- Test OS/Software upgrades easily
- Tune production deployment without production
- Reproduce production problems out of band
- Recover from server failures quickly
- Duplicate Production in Staging

Goals continued...

- Test outside of the MBP bubble
- Make setup repeatable
- Manage servers in bulk
- FNG gets instance on Day #1
- FNG gets MBP Setup on Day #1

Three Steps To A New Server

Step 1: Provision

Step 2: Configure

Step 3: Deploy

Provisioning a New Server

Where Do We Put The Server?

- Amazon AWS / EC2
- Rackspace RackCloud
- SliceHost, Linode, TerraHost, OpenStack, Eucalyptus
- Local VirtualBox Install
- Existing Servers
- Managed Host / Service Provider

What Needs Provisioned?

- Operating System + root access
- Install just enough to run configure / deploy steps later
- Install Ruby / Ohai / Chef / RubyShadow / Bundler
- Remove Future Roadblocks in Configuration / Deployment
  - SSH Config: Disable Require TTY, Env Keep PATH / SSH_AUTH_LOCK
  - Disable SeLinux (KickStart Bug! / Chef Recipes / Apache)
  - Configure $PATH: environment, bashrc, profile, etc
  - LD PATH: ldconfig (bundler deployment cache issues)

How Do We Do It?

- VirtualBox 4.1.0 (4.1.2 has issues!)
  - veewee - Creates Images (“box”) - https://github.com/jedi4ever/veewee
  - vagrant - Manages Boxes / Instances - http://vagrantup.com/
- EC2
  - knife / knife-ec2 - Manages Instances - https://github.com/opscode/
- RackCloud
  - knife / knife-rackcloud - Manages Instances - https://github.com/opscode/
- Managed / Existing Servers
  - ssh / sudo / su

VirtualBox Provisioning

- Install VirtualBox 4.1.0
  - http://www.virtualbox.org/wiki/Download_Old_Builds_4_1
- Install Ruby Gems
  - **`$ gem install veewee vagrant`**
- Define/Customize a new machine image
  - **`$ vagrant basebox define MyServer CentOS-5.6-x86_64-netboot`**
  - Edit KickStart Config - **`vim ks.cfg`**
  - Edit Post Install Script - **`vim postinstall.sh`**

```
# Kickstart file automatically generated by anaconda.

install
url --url=http://centos.mirror.facebook.net/5.6/os/x86_64
lang en_US.UTF-8
langsupport --default=en_US.UTF-8 en_US.UTF-8
keyboard us
xconfig --card "VMWare" --videoram 16384 --hsync 31.5-37.9 --vsync 50-70 --resolution 800x600 --depth 16
network --device eth0 --bootproto dhcp --hostname vagrant-centos64-5-6.local
rootpw --iscrypted $1$vSG8FjAu$ekQ0grf16hS4G93HTPcco/
firewall --enabled --trust eth0 --ssh
selinux --disabled
authconfig --enableshadow --enablemd5
timezone America/New_York
bootloader --location=mbr
# The following is the partition information you requested
# Note that any partitions you deleted are not expressed
# here so unless you clear all partitions first, this is
# not guaranteed to work
clearpart --all --drives=sda --initlabel
part /boot --fstype ext3 --size=100 --ondisk=sda
part pv.2 --size=0 --grow --ondisk=sda
volgroup VolGroup00 --pesize=32768 pv.2
logvol swap --fstype swap --name=LogVol01 --vgname=VolGroup00 --size=528 --grow --maxsize=1056
logvol / --fstype ext3 --name=LogVol00 --vgname=VolGroup00 --size=1024 --grow
reboot

%packages --excludedocs --nobase
grub
e2fsprogs
lvm2

%post
yum install -y sudo
/usr/sbin/groupadd vagrant
/usr/sbin/useradd vagrant -g vagrant
echo "vagrant"|passwd --stdin vagrant
echo "vagrant        ALL=(ALL)       NOPASSWD: ALL" >> /etc/sudoers
echo "PATH=/bin:/usr/bin:/usr/local/bin:/opt/ree/bin" >> /etc/environment
echo 'PATH=/opt/ree/bin:$PATH' >> /etc/bashrc
echo 'export PATH=/opt/ree/bin:$PATH' >> /root/.profile
source /root/.profile
vagrant/definitions/CentOS-5.6-x86_64/ks.cfg [unix] [cfg] (1/42,1) [Git(master)] [ree-1.8.7-2010.02@setup]
```

```
1  date > /etc/vagrant_box_build_time
2
3  fail()
4  {
5    echo "FATAL: $*"
6    exit 1
7  }
8
9  # kernel source is needed for vbox additions
10 echo "Installing base yum packages"
11 rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/x86_64/epel-release-5-4.noarch.rpm
12 yum -y install curl ftp rsync time wget which patch gcc bzip2 make kernel-devel-`uname -r` gcc-c++ zlib-devel openssl-devel readline-devel vim-minimal
13 yum -y erase wireless-tools gtk2 libX11 hicolor-icon-theme avahi freetype bitstream-vera-fonts ruby
14 yum -y clean all
15
16 # Installing ruby
17 echo "Installing ree ruby"
18 cd ~
19 wget http://rubyenterpriseedition.googlecode.com/files/ruby-enterprise-1.8.7-2011.03.tar.gz || fail "Could not download Ruby source"
20 tar xzvf ruby-enterprise-1.8.7-2011.03.tar.gz
21 cd ruby-enterprise-1.8.7-2011.03
22 ./installer --auto /opt/ree --dont-install-useful-gems --no-dev-docs
23
24 cd ~
25 rm -rf ruby-enterprise-1.8.7-2011.03
26 rm ruby-enterprise-1.8.7-2011.03.tar.gz
27
28 # Setting Gem defaults
29 echo "Settings gemrc defaults"
30 cat << 'EOF' >> /etc/gemrc
31 ---
32 :bulk_threshold: 1000
33 :verbose: true
34 :update_sources: true
35 :sources:
36 - http://rubygems.org/
37 gem: --no-ri --no-rdoc
38 :backtrace: true
39 :benchmark: false
40 EOF
41
42 # Upgrading rubmgems
vagrant/definitions/CentOS-5.6-x86_64/postinstall.sh [unix] [sh] (1/100-1) [Git(master)] [ree-1.8.7-2010.02@setup]
```

VirtualBox Provisioning cont...

- Install VirtualBox 4.1.0
  - http://www.virtualbox.org/wiki/Download_Old_Builds_4_1
- Install Ruby Gems
  - `$ gem install veewee vagrant`
- Define/Customize a new machine image
  - `$ vagrant basebox define MyServer CentOS-5.6-x86_64-netboot`
- Build the machine image
  - `$ vagrant basebox build MyServer`
  - `$ vagrant basebox export MyServer`

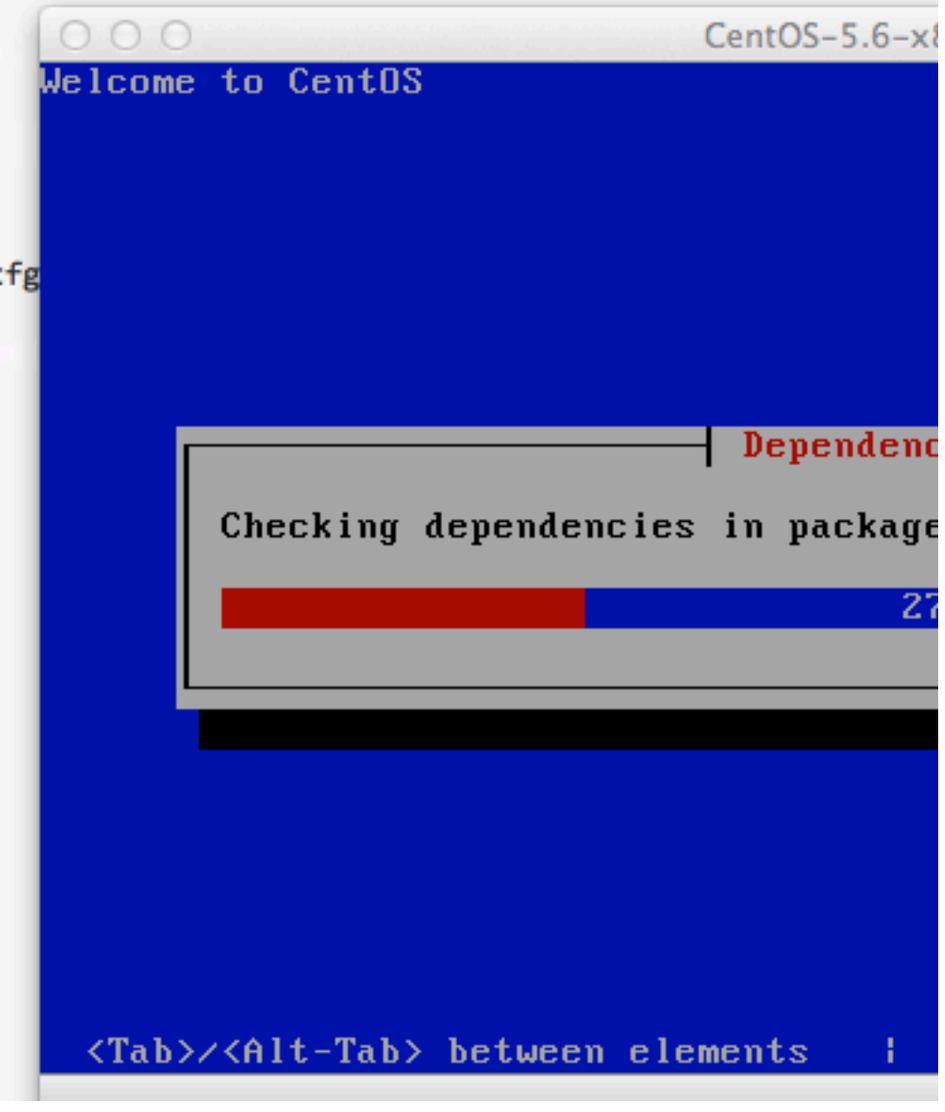
```
claco@hank vagrant (master) (ree-1.8.7-2010.02@setup) $ vagrant basebox build CentOS-5.6-x86_64

Verifying the isofile CentOS-5.6-x86_64-netinstall.iso is ok.
We found no good state so we are destroying the previous machine+disks
VBoxManage unregistervm  'CentOS-5.6-x86_64' --delete
Deleting vm CentOS-5.6-x86_64
Setting VM Flag ioapic to on
Setting VM Flag pae to on
Creating vm CentOS-5.6-x86_64 : 1024M - 1 CPU - RedHat_64
Creating new harddrive of size 40960
VBoxManage createhd --filename '/Users/claco/VirtualBox VMs/CentOS-5.6-x86_64/CentOS-5.6-x86_64.vdi' --size '40960' --format vdi > /dev/nul
Attaching disk: /Users/claco/VirtualBox VMs/CentOS-5.6-x86_64/CentOS-5.6-x86_64.vdi
Mounting cdrom: /Volumes/Within3/within3_setup/vagrant/iso/CentOS-5.6-x86_64-netinstall.iso
Waiting for the machine to boot

Typing:[1]: linux text ks=http://192.168.10.18:7122/ks.cfg<Enter>
Done typing.

Starting a webserver on port 7122
Serving file /Volumes/Within3/within3_setup/vagrant/definitions/CentOS-5.6-x86_64/ks.cfg

Waiting for ssh login with user vagrant to sshd on port => 7222 to work
```



```
Connecting to dlc.sun.com.edgesuite.net|205.234.225.152|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 44957696 (43M) [application/octet-stream]
Saving to: `VBoxGuestAdditions_4.1.0.iso'

100%[======================================>] 44,957,696   507K/s   in 86s

2011-09-11 18:10:05 (511 KB/s) - `VBoxGuestAdditions_4.1.0.iso' saved [44957696/44957696]

Verifying archive integrity... All good.
Uncompressing VirtualBox 4.1.0 Guest Additions for Linux.........
VirtualBox Guest Additions installer
Removing existing VirtualBox DKMS kernel modules[  OK  ]
Removing existing VirtualBox non-DKMS kernel modules[  OK  ]
Building the VirtualBox Guest Additions kernel modules
Not building the VirtualBox advanced graphics driver as this Linux version is
too old to use it.
Building the main Guest Additions module[  OK  ]
Building the shared folder support module[  OK  ]
Doing non-kernel setup of the Guest Additions[  OK  ]
Starting the VirtualBox Guest Additions [  OK  ]
Installing the Window System drivers[FAILED]
(Could not find the X.Org or XFree86 Window System.)
Disabling requirgetty in sudoers
Disabling SELinux
Disabling DNS in sshd
Aliasing vi to vim
Locking root account
Locking password for user root.
passwd: Success

Step [1] was successfully - saving state
CentOS-5.6-x86_64 was built successfully.

Now you can:
- verify your box by running               : vagrant basebox validate CentOS-5.6-x86_64
- export your vm to a .box file by running : vagrant basebox export   CentOS-5.6-x86_64
claco@hank vagrant (master) (ree-1.8.7-2010.02@setup) $
```

EC2/RackCloud Provisioning

- Install Ruby Gems *(Bundler Issue! JSON Lock!)*
  - `$ gem install chef knife-ec2 knife-rackspace`
- Configure API Keys in `~/.chef/knife.rb`
  - `knife[:aws_access_key_id]  = "Your AWS Access Key ID"`
  - `knife[:aws_secret_access_key] = "Your AWS Access Key"`
- Customize the server image
  - `$ vim ~/.chef/bootstrap/centos56.rb`

```
# /* vim: set filetype=sh : */
echo "Installing sudo"
yum install -y sudo

echo "Updating paths for ree bin"
echo "PATH=/bin:/usr/bin:/usr/local/bin:/opt/ree/bin" >> /etc/environment
echo 'PATH=/opt/ree/bin:$PATH' >> /etc/bashrc
echo 'export PATH=/opt/ree/bin:$PATH' >> /root/.profile
source /root/.profile

echo "Installing base yum packages"
rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/x86_64/epel-release-5-4.noarch.rpm
yum -y install curl ftp rsync time wget which patch gcc bzip2 make kernel-devel-`uname -r` gcc-c++ zlib-devel openssl-devel readline-devel vim-minimal
yum -y erase wireless-tools gtk2 libX11 hicolor-icon-theme avahi freetype bitstream-vera-fonts ruby
yum -y clean all

# Installing ruby
echo "Installing ree ruby"
cd ~
wget http://rubyenterpriseedition.googlecode.com/files/ruby-enterprise-1.8.7-2011.03.tar.gz || fail "Could not download Ruby source"
tar xzvf ruby-enterprise-1.8.7-2011.03.tar.gz
cd ruby-enterprise-1.8.7-2011.03
./installer --auto /opt/ree --dont-install-useful-gems --no-dev-docs

cd ~
rm -rf ruby-enterprise-1.8.7-2011.03
rm ruby-enterprise-1.8.7-2011.03.tar.gz

# Setting Gem defaults
echo "Settings gemrc defaults"
cat << 'EOF' >> /etc/gemrc
---
:bulk_threshold: 1000
:verbose: true
:update_sources: true
:sources:
- http://rubygems.org/
gem: --no-ri --no-rdoc
:backtrace: true
:benchmark: false
EOF

.chef/bootstrap/centos-5.4.erb [unix] [sh] (1/195-1) [Git(master)] [ree-1.8.7-2010.02@setup]
-- INSERT --
```

EC2/RC Provisioning cont...

- Install Ruby Gems *(Bundler Issue! JSON Lock!)*
  - `$ gem install chef knife-ec2 knife-rackspace`
- Configure API Keys in `~/.chef/knife.rb`
  - `knife[:aws_access_key_id]  = "Your AWS Access Key ID"`
  - `knife[:aws_secret_access_key] = "Your AWS Access Key"`
- Customize the server image
  - `$ vim ~/.chef/bootstrap/centos56.rb`
- Create the machine image
  - `$ knife ec2 create -I ami-0a59bb63 -d centos-5.4 ...`

Login To Your New Server

- VirtualBox / Vagrant
  - `$ vagrant ssh  [ssh vagrant@localhost -p 2222]`
- EC2
  - `$ ssh root@ec2-xxx-xxx-xxx-xxx.compute-1.amazonaws.com -i ec2-group-key.pem`
- Rackspace
  - `$ ssh root@xxx-xxx-xxx-xxx.staticip.rackspace.com`
- Questions?

Configuring Your New Server

What Is Chef?

- Configuration management for “Nodes” or servers
- It is a “Cookbook” full of configuration “Recipes” plus “Data Bags”
  - Install “build” user. Set password. Configure ssh key. Configure github access.
- Cookbooks, Recipes, Data Bags stored upstream on OpsCode server
- Client downloads recipes and runs them on each server
- Configure things differently by “Environment”: production, staging, development
- Configure “Roles” or groups of recipes: app, db, caching, services, etc
- Manage Cookbooks, Recipes, Roles and Nodes from command line

What Does A Data Bag Do?

```
{
  "id": "build",
  "uid": 1000,
  "gid": 1000,
  "comment": "Build User",
  "shell": "/bin/bash",
  "password": "$1$31Pf4SgRy$edFhgUyhUBDE3%eUSD4rmk1",
  "ssh_keys": "ssh-rsa AAAABC2TbS43DAAABD4ER3DH4WT....default",
  "sudoers": "ALL=(ALL) ALL"
}
```

What Does A Recipe Do?

```
home_dir = "/home/#{u['id']}"

group u['id'] do
  gid u['gid']
end
user u['id'] do
  uid u['uid']
  gid u['gid']
  shell u['shell']
  password u['password']
  home home_dir
end
directory "#{home_dir}/.ssh" do
  owner u['id']
  group u['gid'] || u['id']
  mode "0700"
end
template "#{home_dir}/.ssh/authorized_keys" do
  source "authorized_keys.erb"
  owner u['id']
  group u['gid'] || u['id']
  mode "0600"
  variables :ssh_keys => u['ssh_keys']
end
```

What Does A Role Do?

```
name "app"

description "App role for all web servers."

run_list
  "role[base]",
  "recipe[apache2]",
  "recipe[apache2::mod_ssl]",
  "recipe[mysql::client]",
  "recipe[passenger_apache2]",
  "recipe[passenger_apache2::mod_rails]",
  "recipe[sphinx]",
  "recipe[company::ssl]",
  "recipe[company::mainsite]",
  "recipe[company::mobilesite]"
```

What Does An Environment Do?

```
name "development"
description "The development environment"
default_attributes
  "company" => {
    "mainsite"  => { "virtual_host" => "localhost" }
  }


name "staging"
description "The staging environment"
default_attributes
  "company" => {
    "mainsite"  => { "virtual_host" => "mainsite-staging.company.com" }
  }


name "production"
description "The production environment"
default_attributes
  "company" => {
    "mainsite"  => { "virtual_host" => "www.company.com" }
  }
```

Manage Everything Via Terminal

```
$ knife node list

   app1, ec2-claco, services2, staging-db, vagrant-claco-mainsite, ....

$ knife cookbook list

   apache, xml, xslt, imagemagic, company::users, mysql, ....

$ knife search node "chef_environment:production AND role:services"

   Node Name:   services1
   Environment: production
   FQDN:        services1.company.com
   IP:          172.16.2.3
   Run List:    role[base], role[services]
   Roles:       cache, queue, services, search, base
   Recipes:     company::users, ntp, postfix, java, memcached
   Platform:    redhat 5.6

$ knife ssh “name:app*” “pwd” -x build

   app1.company.com Mon Sep 12 10:07:51 CDT 2011
   app2.company.com Mon Sep 12 10:07:51 CDT 2011
   app3.company.com Mon Sep 12 10:07:51 CDT 2011
```

Running Chef

- VirtualBox / Vagrant
  - chef-client automatically runs after vagrant up
  - **`vagrant provision`** to manually reconfigure the server
- EC2 / Rackspace
  - chef-client automatically runs after knife create bootstrap
  - **`sudo chef-client`** to manually reconfigure the server
- Managed Servers
  - **`sudo chef-client`** to manually configure the servers
- Automate from afar! **`knife ssh "name:mynode" "sudo chef-client"`**

What About My Mac!

- Install Using Homebrew! - https://github.com/mathie/chef-homebrew
- Install Using Dmg! - https://github.com/opscode/cookbooks/tree/master/dmg

```
dmg_package "Google Chrome" do
  dmg_name "googlechrome"
  source "https://dl-ssl.google.com/chrome/mac/stable/GGRM/googlechrome.dmg"
  checksum "7daa2dc5c46d9bfb14f1d7ff4b33884325e5e63e694810adc58f14795165c91a"
  action :install
end

dmg_package "Dropbox" do
  volumes_dir "Dropbox Installer"
  source "http://www.dropbox.com/download?plat=mac"
  checksum "b4ea620ca22b0517b75753283ceb82326aca8bc3c86212fbf725de6446a96a13"
  action :install
end

dmg_package "Virtualbox" do
  source "http://dlc.sun.com.edgesuite.net/virtualbox/4.0.8/VirtualBox-4.0.8-71778-OSX.dmg"
  type "mpkg"
end
```

Deploy Your Application

Configure Capistrano

- Create a multistage environment configuration (multistage plugin or inline tasks)

- `gem install capistrano`
- `cd MyApp`
- `Capify .`
- `vim config/deploy.rb`

Override Defaults Per Destination

```
role :web, "localhost"
role :app, "localhost"
role :db,  "localhost", :primary => true

task :vagrant do
  set :port, 2222
end

task :ec2 do                    # same as :rackspace
  role :web, ENV['address']
  role :app, ENV['address']
  role :db,  ENV['address'], :primary => true

  ssh_options[:keys] = "~/.ssh/your-default-key-pair.pem"
end

task :production do
  role :web, "app1.company.com", "app2.company.com.com", "app3.company.com.com"
  role :app, "app1.company.com", "app2.company.com.com", "app3.company.com.com"
  role :db,  "app1.company.com.com", :primary => true

  ssh_options[:keys] = "~/.ssh/your-default-key-pair.pem"
end
```

Deploy Your Application

- `cap <environment> <action> branch=value address=value rails_env=environment`
- `cap ec2 deploy:initial address=xxx.xxx.xxx.xxx branch=mybranch`
- `cap rackspace deploy:initial address=xxx.xxx.xxx.xxx branch=mybranch`
- `cap vagrant deploy:initial [address=localhost] branch=mybranch`
- `cap production deploy:update [branch=master]`
- `cap deploy:update branch=mybranch  (uses defaults)`
- open http://address/ and enjoy!

Capistrano Deploy Tasks

- deploy:initial
  - deploy:setup, deploy:update, db:setup, sphinx:reindex, starling:restart, workling:restart, deploy:restart
  - Run on fresh instances: vagrant, ec2, rackspace.
  - db:setup is disabled for production environment.
- deploy:web:enable / deploy:web:disable
  - Also does cluster:put / cluster:pull
- deploy:tests:environment, deploy:test:connections. Anything!

Setup Project Contents

- **`.chef`** - Preconfigured to talk to OpsCode, EC2, Rackspace!
- **`chef`** - Company Cookbooks, Recipes, Roles, Environments, DataBags
- **`vagrant/definitions`** - Preconfigured CentOS 5.6 x64 Machine!
- **`vagrant/instances/mainsite`** - Preconfigured Vagrant Site Instance!
  - **`cd vagrant/instances/mainsite; vagrant up; cap vagrant deploy:initial`**

Gitify Your Work. Clone And Go

- `git clone git@github.com:Company/company_setup.git`
- `cd company_setup`
- `bundle install; gem install chef  (JSON Issue!)`
  - `rake ec2/rackspace instance:create`
  - `cd vagrant/instances/mainsite && vagrant up`
- `cd company_mainsite`
- `cap ec2/rackspace deploy:initial address=xxx.xxx.xxx.xxx branch=gerbilsauce`

Meeting Our Goals

Goals Revisited

- Any Engineer/QA can spin up machines
- Same configuration everywhere
- Add more servers when traffic increases
- Test OS/Software upgrades easily
- Tune production deployment without production
- Reproduce production problems out of band
- Recover from server failures quickly
- Duplicate Production in Staging

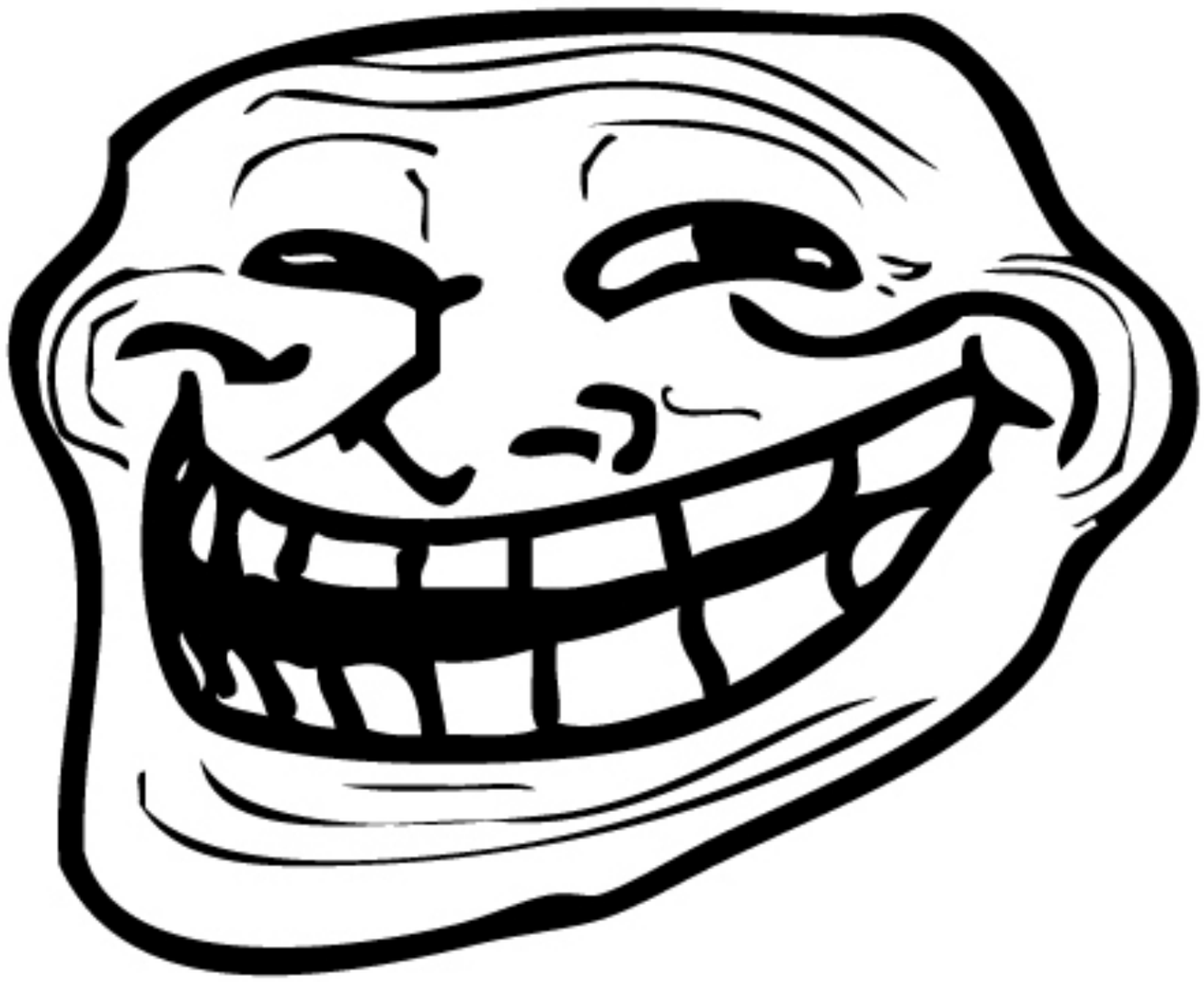
Goals Revisited continued...

- Test outside of the MBP bubble
- Make setup repeatable
- Manage servers in bulk
- FNG gets instance on Day #1
- FNG gets MBP Setup on Day #1

This Is “Easy”?

- Sorry. I lied. Servers are hard. We make it that way.
- This is an investment in your infrastructure.
- Get the knowledge from one person into something accessible.
- Make this stuff “easy-er” so you can focus on something else.

I Forgot The Last Step...



Step 4: Drink Beers!

Thanks for participating!

Slides: http://chrislaco.com/slides/clerb-caveman.pdf

Email: claco@chrislaco.com

Twitter: @claco