ThoughtWorks®

Sydney Testers Bootcamp

“WORKS ON MY MACHINE”

BUILDING + PROVISIONING TEST ENVIRONMENTS WITH CHEF AND VAGRANT

BEFORE WE BEGIN

If you don't have :

VAGRANT
VIRTUALBOX
RUBY W/DEVKIT (WINDOWS ONLY)
CHEF WITH CHEFDK

Or have not yet run:

VAGRANT BOX ADD CHEF/UBUNTU-14.04

Pair with someone who has!

ThoughtWorks®

WHAT'S THE PROBLEM?

WHAT'S THE PLAN?

TESTING HEURISTICS

SPEED

REPEATABILITY

COVERAGE

RELIABILITY

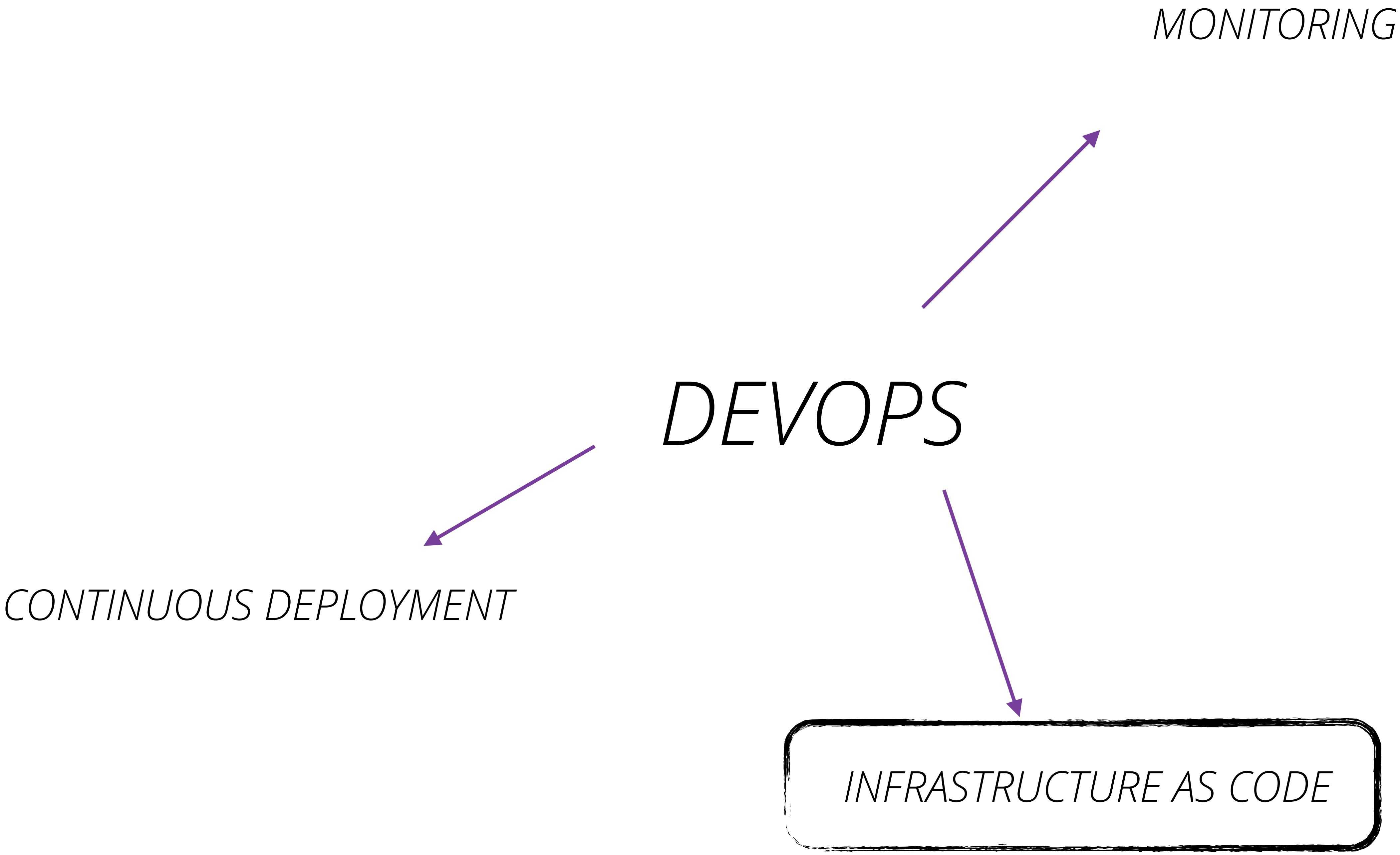
USER SATISFACTION

RELIABILITY

How likely it is that the behaviour in environment X is representative of the behaviour in environment Y:

- *Workstation*
- *CI Environment*
- *QA Environment*
- *Production*

MEANWHILE, IN A DUNGEON NEAR YOU...



MONITORING

DEVOPS

CONTINUOUS DEPLOYMENT

INFRASTRUCTURE AS CODE

INFRASTRUCTURE-AS-CODE

A big part of the DevOps movement. Bringing the best practises from dev to infrastructure:

- Source control
- Testing
- Dependency Management
- Build once, deploy-everywhere

INFRASTRUCTURE-AS-CODE

And a big emphasis on Configuration Management tools:

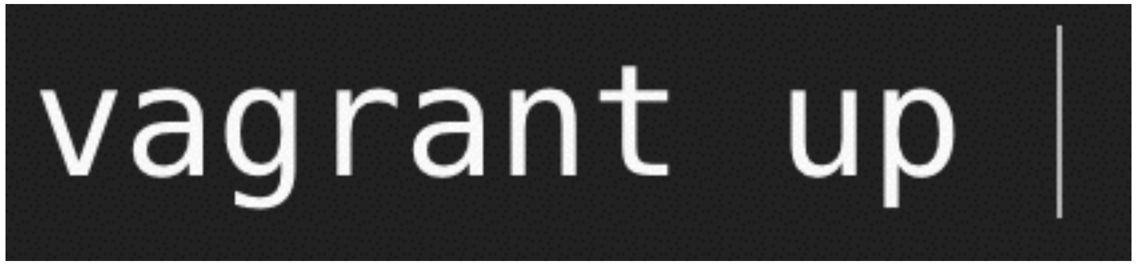
INFRASTRUCTURE-AS-CODE

And improved tooling around virtualisation and containers:

SO WHAT?

BUILDING A TEST ENVIRONMENT

Procurement
Package installation
Package configuration
Backup/Archive
Maintenance



```
vagrant up
```

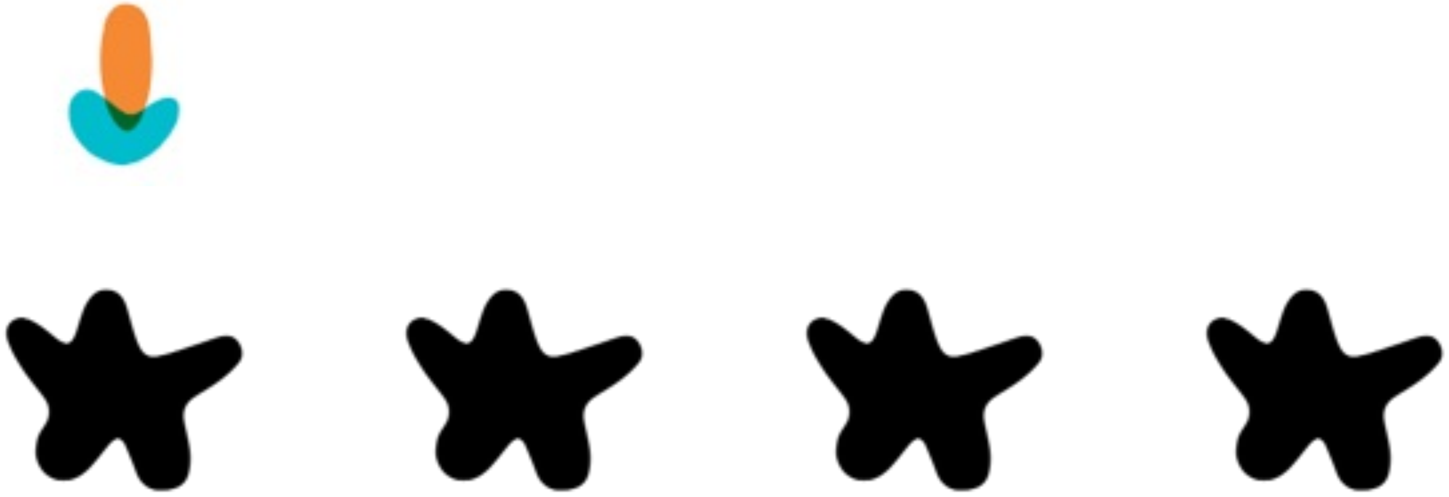
ThoughtWorks®

WHAT'S THE PROBLEM?

WHAT'S THE PLAN?

PLAN

4 Iterations - 20 minutes each



Prepare ingredients

Adding some sugar

Little bit of Chilli

A taste test

Our destiny

ITERATION 1 - THE VAGRANTFILE

VAGRANT

GOOGLE: "VAGRANT GETTING STARTED"

- For creating and configuring development environments
- Is a wrapper around virtualbox/vmware etc

ITERATION 1 - THE VAGRANTFILE

VAGRANT *GOOGLE: "VAGRANT GETTING STARTED"*

- cd to your favourite directory
- run 'vagrant init' - generates Vagrantfile
- configure vagrant to use base box: chef/ubuntu-14/04
- put the box on a private network, on an ip of your choosing
- run 'vagrant up'
- ssh into the box using "vagrant ssh" (linux/mac) or by using Putty on windows:

  host: **{{ip-from-vagrantfile}}** port: **22**

  user: **vagrant** password: **vagrant**

ITERATION 1 - THE VAGRANTFILE

VAGRANT *GOOGLE: "VAGRANT GETTING STARTED"*

- destroy your box with "vagrant destroy"
- Modify your Vagrantfile to increase the memory on the box, and if you have multiple cores on your host, adjust the number of CPUs to match
- vagrant up
- enable shared folders (see git repo)
- check your work with:

  'cat /proc/meminfo'

  'cat /proc/cpuinfo | grep processor | wc -l'

  and by dropping files from host into guest

ITERATION 1 - THE VAGRANTFILE

```
VAGRANTFILE_API_VERSION = "2"

Vagrant.require_version ">= 1.6.3"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|

   config.vm.hostname = "testbox”

   config.omnibus.chef_version = "11.6”

   config.vm.box = "chef/ubuntu-14.04"

   config.vm.network :private_network, ip: '33.33.33.10'

   config.vm.boot_timeout = 120

   config.berkshelf.enabled = true
end
```

ITERATION 2 - CHEF

CHEF™

GOOGLE: "GET CHEF"

- Tooling and language for automating provisioning and configuring of machines
- Runs in pure ruby

ITERATION 2 - CHEF - TERMS OF REFERENCE

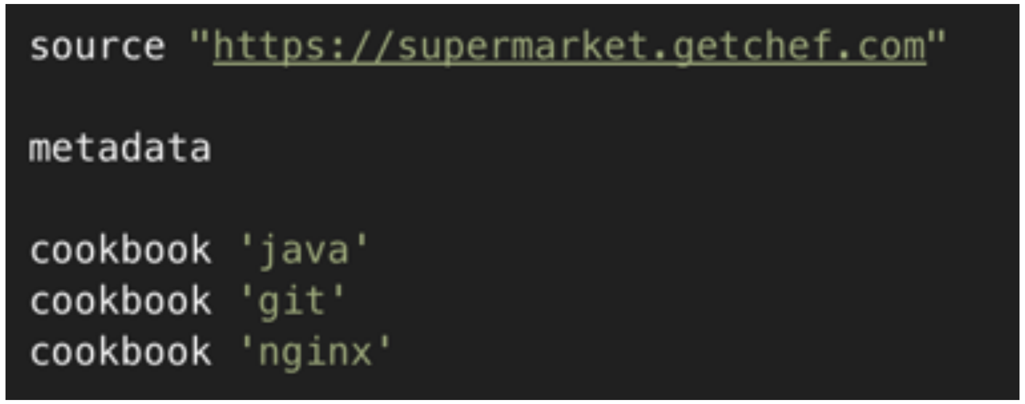
CHEF™

GOOGLE: "GET CHEF"

- A '**recipe**' is a list of installation and configuration activities that configure a server to a known state
- Collections of recipes are called '**cookbooks**'
- **"Berkshelf"** is a tool to manage cookbooks

ITERATION 2 - CHEF

- Cookbooks available on the web can be used with Berkshelf
- We don’t have to write our own recipes for many standard packages (e.g. Java)
- We can add them to our **Berksfile**:



```
source "https://supermarket.getchef.com"

metadata

cookbook 'java'
cookbook 'git'
cookbook 'nginx'
```

- And Berkshelf will make these available to us!

ITERATION 2 - CHEF

CHEF™

GOOGLE: "GET CHEF"

- cd to git repo /02…./machines
- remove 'Vagrantfile' and rename 'Vagrantfile_I2'
- ensure chefdk is at the front of your path:

    *nix: "PATH=/opt/chefdk/bin:$PATH"

    win: "SET PATH=C:\opscode\chefdk\bin;%PATH%"

ITERATION 2 - BY OUR POWERS COMBINED....

- Configure vagrant to use the chef provisioner, adding recipes we want to run:

```
config.vm.provision :chef_solo do |
chef|
    chef.run_list = [
        'recipe[git]'
    ]
    chef.log_level = :debug
end
```

- ‘vagrant destroy’ then ‘vagrant up’

ITERATION 2 - BY OUR POWERS COMBINED….

- SSH into the box - then see if git is installed:

```
git —version
```

ITERATION 3 - YOUR OWN RECIPES

- Create a new file in the ‘recipes’ directory
- Add the recipe name to the Vagrantfile:

    ‘recipe[machines::myrecipe]’

ITERATION 3 - YOUR OWN RECIPES

Using the Chef 'Resource' Documentation as a guide, create a recipe that does the following:

Create the following user:

Username: sydneytester

Home folder: /home/sydneytester

Password: Tester@123

Create the following directory:

Owner: sydneytester

Path: /vagrant_data/quicktest

Permission: 664

ITERATION 3 - YOUR OWN RECIPES

Using the Chef ‘Resource’ Documentation as a guide, create a recipe that does the following:

Starts the service “apache2”

Creates a directory: “/var/www/sydneytester”

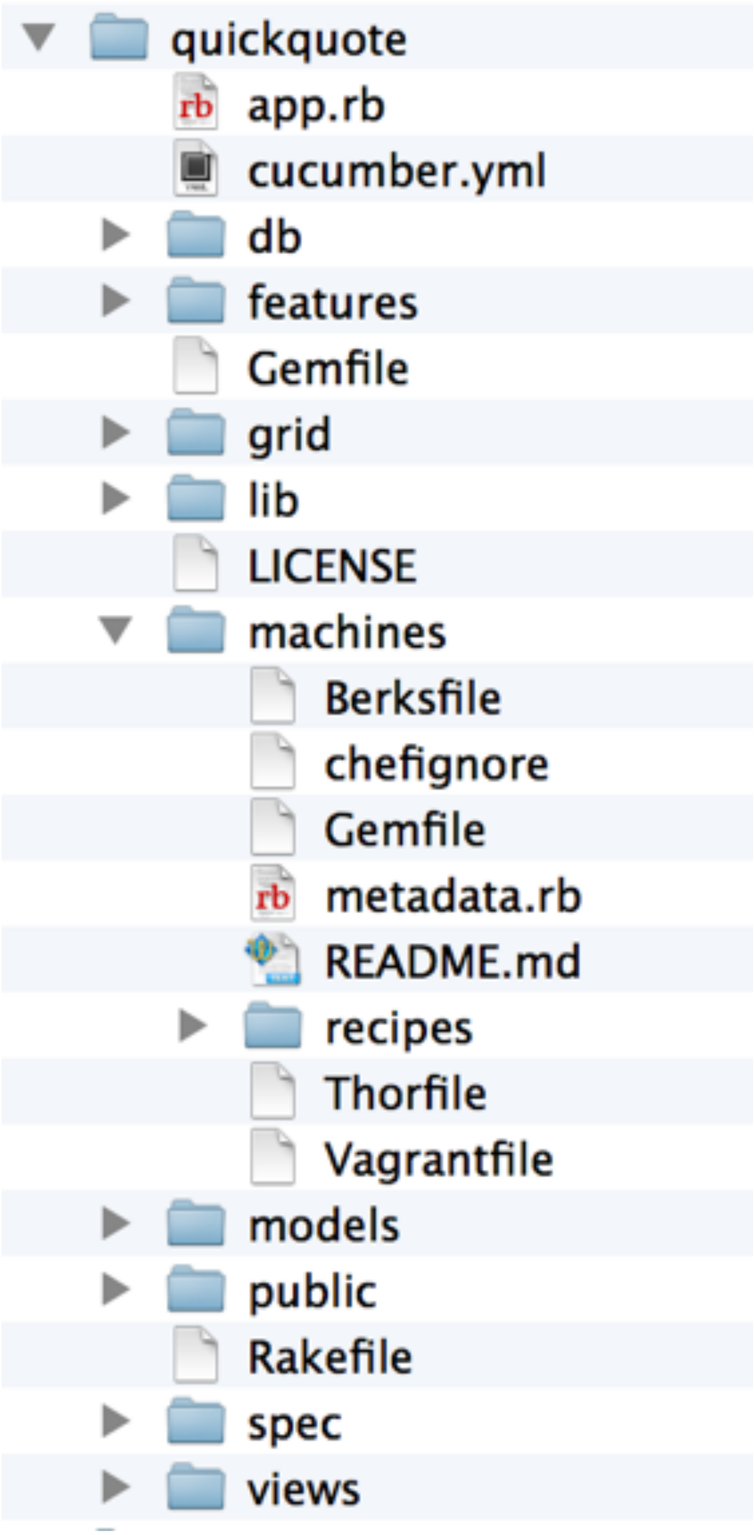
Create file “index.html” -> “<h1>Yo!</h1>

Add the following recipes in your Vagrantfile:

“apache2”

ITERATION 4 - INTEGRATE INTO THE PROJECT

- Vagrant destroy!
- Add files in the image to your fork of the ***'quickquote'*** app, into a folder called ***'machines'***
- ***cd to machines***
- ***vagrant up***



```
▼ quickquote
      app.rb
      cucumber.yml
    ▶ db
    ▶ features
      Gemfile
    ▶ grid
    ▶ lib
      LICENSE
    ▼ machines
          Berksfile
          chefignore
          Gemfile
          metadata.rb
          README.md
        ▶ recipes
          Thorfile
          Vagrantfile
    ▶ models
    ▶ public
      Rakefile
    ▶ spec
    ▶ views
```

THANK YOU

Next week:

"CONTINUOUS INTEGRATION"

with Hans and Leo

ThoughtWorks®