# Configure Your Development Environment:

## Vagrant and VirtualBox



## Chef Workflow

When building and testing Chef code a normal workflow involves managing servers directly from your workstation. In this class, you'll start by logging into a server directly to get to know the way Chef works.

In the second half of class, we'll manage remote servers (nodes) using a workstation connected to a Chef Server. For now, we'll start by managing a node using Vagrant and VirtualBox.

# Objectives:

- Check pre-reqs
- Verify your ssh client
- Install VirtualBox
- Install Vagrant
- Launch Centos VM

# Ensure your system meets the minimum requirements

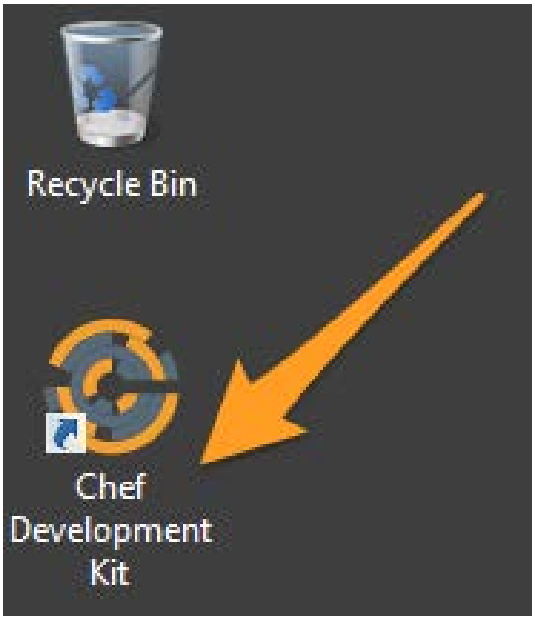
Running virtual machines can be demanding on your system's hardware. Before proceeding with the labs, please ensure that your system:

1. Supports virtualization. This is typically enabled in your BIOS
2. Meets the minimum requirements to run VirtualBox
3. Has at least 5GB of hard-disk space available
4. Has at least 512MB RAM available for each VM you would like to run (this class may have a maximum of 3 VM's running at any time, so please ensure you have at least 1.5GB RAM available)
5. If using VMware Fusion, ensure nested virtualization is enabled
6. If your system doesn't meet the pre-reqs, consider one of the other options for completing the class exercises.

# Verify your ssh client

This class uses ssh to connect to our instances. This usually involves an ssh client. If using MacOS or most Linux workstations the terminal should work.

If using Windows or having trouble connecting, the ChefDK includes Git for Windows, an ssh client.




# Disclaimers:

Chef integrates easily with VirtualBox and Vagrant, but these projects are not maintained by Chef. If something isn't working, we recommend the following:

- Check the Vagrant Issues or VirtualBox Bugtracker pages
- Refer to documentation:
  - Vagrant
  - VirtualBox

This class is known to work with the versions of the software we suggest installing. If something isn't working, check that you're running with the tested versions of the following:

- ChefDK 0.18.30
- Vagrant 1.8.6
- VirtualBox 5.1.8r111374
- Optional: Git 2.8.2

# Class Workflow

For the first half of the class we will log into a Vagrant instance and work with Chef by directly managing the virtual machine

On the virtual CentOS instance we will install the Chef Development Kit (ChefDK) and write code using a command-line text editor, like Vi, Emac or Nano

In the second half of the class we will manage several Vagrant instances remotely using a Chef Server.

For these exercises we will be using your local machine, where the ChefDK will also be installed. You can use any text editor you prefer for these exercises. I'll be using Vim or Sublime Text throughout the video demos.

# Install the Chef Development Kit

You can install ChefDK from here
or
On Windows you can run the
installation script->

```
PS > . { iwr -useb https://omnitruck.chef.io/install.ps1 } | iex; install -project chefdk -channel stable -version  1.0.3
```

After installing on Linux and MacOS, check that the tools can be found by running:

```
$ chef --version
```

On Windows, launch the ChefDK to run *chef --version* or update your PATH to include the tools in your Powershell session

# Install VirtualBox

- Windows: Consider using the Chocolatey Installer or
- Download VirtualBox directly from Oracle

Verify the installation by running:

```
$ VBoxManage --version
5.1.8r111374
```

Note: On Windows, VirtualBox is installed to C:\Program Files\Oracle\VirtualBox
If the command fails, you may need to update your system PATH:

```
PS> $path = [Environment]::GetEnvironmentVariable("PATH", "Machine")
PS> $vbox_path = "C:\Program Files\Oracle\VirtualBox"
PS> [Environment]::SetEnvironmentVariable("PATH", "$path;$vbox_path", "Machine")
```

# Install Vagrant

Windows: Consider using the Chocolatey Installer
Download Vagrant directly from HashiCorp

Verify the installation by running:

```
$ vagrant --version
Vagrant 1.8.6
```

Note: On Windows, Vagrant is installed to
*C:\HashiCorp\Vagrant\bin*
To add Vagrant to your system PATH, run:

```
PS> $path = [Environment]::GetEnvironmentVariable("PATH", "Machine")
PS> $vagrant_path = "C:\HashiCorp\Vagrant\bin"
PS> [Environment]::SetEnvironmentVariable("PATH", "$path;$vagrant_path", "Machine")
```

# Download a CentOS 7.2 Vagrant Box

```
$ vagrant box add bento/centos-7.2 --provider=virtualbox

==> box: Loading metadata for box 'bento/centos-7.2'
    box: URL: https://atlas.hashicorp.com/bento/centos-7.2
==> box: Adding box 'bento/centos-7.2' (v2.3.0) for provider: virtualbox
    box: Downloading: https://atlas.hashicorp.com/bento/boxes/centos-7.2/versions/2.3.0/providers/virtualbox.box
```

This downloads a VirtualBox-compatible CentOS 7.2 Vagrant box. The bento/centos-7.2 image is retrieved from the HashiCorp Atlas, and refers to a Chef project that provides Vagrant boxes to make testing on common platforms easy.

Note: On Windows, if the command fails with a blank error message you may need to install Microsoft Visual C++ 2010 SP1. See the Vagrant issues here.

# Launch a CentOS 7.2 Instance

```
$ vagrant init bento/centos-7.2

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

Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'bento/centos-7.2'...
==> default: Checking if box 'bento/centos-7.2' is up to date...
==> default: Setting the name of the VM: root_default_1476898305221_53382
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
    default:
    default: Vagrant insecure key detected. Vagrant will automatically replace
    default: this with a newly generated keypair for better security.
    default:
    default: Inserting generated public key within guest...
    default: Removing insecure key from the guest if it's present...
    default: Key inserted! Disconnecting and reconnecting using new SSH key...
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
==> default: Mounting shared folders...
    default: /vagrant => /root
```

# Log Into the CentOS 7.2 Instance

```
$ vagrant ssh
```

# And Install the ChefDK

```
[vagrant@localhost ~]$ curl https://omnitruck.chef.io/install.sh | sudo bash -s -- -P chefdk -c stable -v 0.18.30

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed

  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 20051  100 20051    0     0  48583      0 --:--:-- --:--:-- --:--:-- 48549
warning: /tmp/install.sh.12800/chefdk-0.18.30-1.el7.x86_64.rpm: Header V4 DSA/SHA1 Signature, key ID 83ef826a: NOKEYel 7 x86_64
Getting information for chefdk stable 0.18.30 for el...
downloading https://omnitruck.chef.io/stable/chefdk/metadata?v=0.18.30&p=el&pv=7&m=x86_64
  to file /tmp/install.sh.12800/metadata.txt
[...]
Installing chefdk 0.18.30
installing with rpm...
Preparing...                          ########################################
Updating / installing...
chefdk-0.18.30-1.el7                  ########################################
Thank you for installing Chef Development Kit!
```

# Setup Your Text Editor

We'll be writing code in this class to configure remote machines. Install your text editor of choice.

If you're new to command-line text editors, we recommend trying Nano.

### Learn Vim

```
[vagrant@localhost ~]$ sudo yum install vim -y
```

### Learn Emacs

```
[vagrant@localhost ~]$ sudo yum install emacs -y
```

### Learn Nano

```
[vagrant@localhost ~]$ sudo yum install vim -y
```

# Manage Your Vagrant Instance

Run these common commands in the same directory as your Vagrantfile:

`$ vagrant init`
creates a Vagrantfile, used to specify virtual machine settings

`$ vagrant up`
spins up the virtual machine using the Vagrantfile

`$ vagrant ssh-config`
list connection details for running instances

`$ vagrant status`
lists virtual machines and current status. 'running' means machine is available for ssh

`$ vagrant suspend`
save machine state and shut down

`$ vagrant destroy --force`
destroy all running virtual machines