Configure Your Development Environment:

Google Cloud Platform

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 GCP.

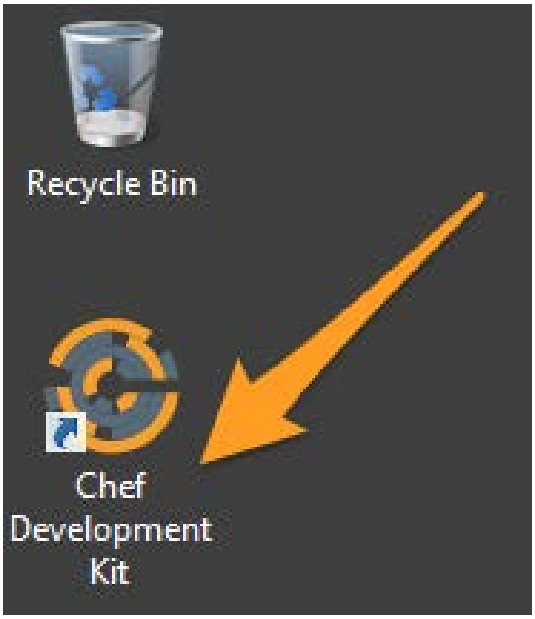
Objectives:

- Check pre-reqs
- Verify your ssh client
- Create GCP Account
- Launch Centos VM

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:

The output that you witness may be different depending on what machine image you use. If you have issues with GCP, check out their support page

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
- Optional: Git 2.8.2

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
```

Create/Access your GCP Account

You will need to log into your GCP account; if you don't have one yet you can take advantage of Google's 60-day free trial for up to $300 of use

If you are unable to use GCP, you may try using one of the other options offered for this class

Launch a CentOS 7 Instance

Spin up a CentOS instance on Google Compute Engine
If you haven't used GCP before, this tutorial will show you how to do so

Launch a CentOS-based 7.2 instance
When you choose the machine type, select micro, small or 1 vCPU

In this process when you choose the firewall settings,
Select Allow HTTP and Allow HTTPS traffic. This opens ports 22 (ssh), 80 (http), and 443 (https)

Class Workflow

For the first half of the class we will log into a GCP 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 Compute Engine 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.

Connect to your GCP Instance

You will now need to connect to your CentOS VM instance
There are a few different ways to connect

This is an example of using ssh with key-based authentication (Replace *~/.ssh/id_rsa* with the path to your key file, replace *chef* with your username, and *40.117.150.107* with your IP address

```
$ ssh -i ~/.ssh/id_rsa chef@40.117.150.107
```

Install the Chef Development Kit

From your ssh connection, run the following command to install the ChefDK:

```
$ curl https://omnitruck.chef.io/install.sh | sudo bash -s -- -P chefdk -c stable -v 0.18.30
```

After installing, check that the tools can be found by running:

```
$ chef --version
```

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
```

Clean Up

To reduce the costs of running your instances, you will need to stop your VM

At the completion of this class you will need to destroy your instances