Using Chef Solo on Windows for AWS

Description

Lab Overview

Chef is a popular configuration management tool. It has built-in support for Windows and a thriving community providing convenient cookbooks for managing Windows configurations. This Lab will get you up and running with Chef Solo and help you configure a local Windows host on AWS using Chef. You will then explore Chef cookbook dependencies and the Chocolatey Windows package manager. Finally, you will configure the host to serve an Internet Information Services (IIS) website.

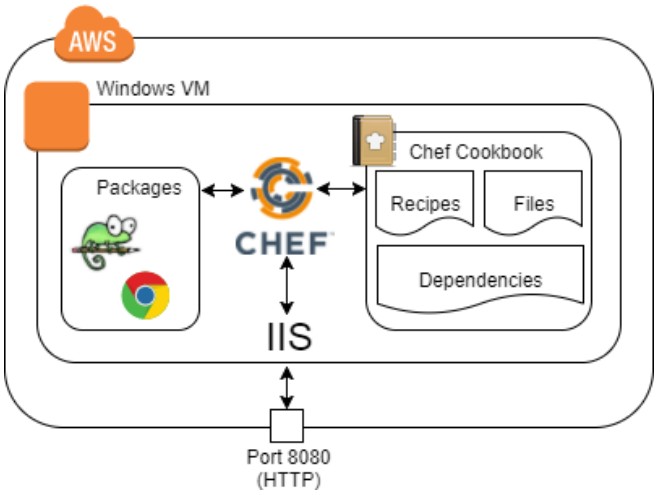
Lab Objectives

Upon completion of this lab you will be able to:

- Install and set up Chef Solo on an AWS Windows VM
- Run Chef recipes with Windows-specific resources
- Use guards to enforce Chef resource idempotence
- Manage Chef cookbook dependencies with Berkshelf
- Configure Microsoft's IIS web server with Chef Solo

Lab Environment

After completing the lab instructions, the environment should look similar to:



Lab steps

1. Logging In to the Amazon Web Services Console
2. Connecting to the Windows Instance
3. Setting Up Chef Solo on Windows
4. Running Your First Chef Solo Recipes on Windows
5. Preserving Chef Resource Idempotence with Guards

6. Managing Chef Cookbook Dependencies with Berkshelf
7. Configuring Microsoft's IIS Web Server with Chef Solo - AWS

Connecting to the Windows Instance

Introduction

In this Lab Step you will use your local operating system's Remote Desktop Protocol (RDP) client to connect to the provisioned Windows Server instance. RDP is a proprietary protocol developed by Microsoft.

The most common RDP clients for each operating system are:

- Microsoft Windows - Remote Desktop, installed by default.
- macOS - Remote Desktop, official Microsoft version available in the App Store.
- GNU/Linux - rdesktop, available at http://www.rdesktop.org, opens in a new tab.

Decrypting the Windows Administrator Password

Amazon EC2 Windows instances using standard Amazon AMIs automatically generate an Administrator password. Complete the following steps to retrieve the Administrator password.

1. Download the Privacy-enhanced Electronic Mail (PEM) key pair automatically generated by the Cloud Academy platform in the **Your lab data** section of the Lab browser window:

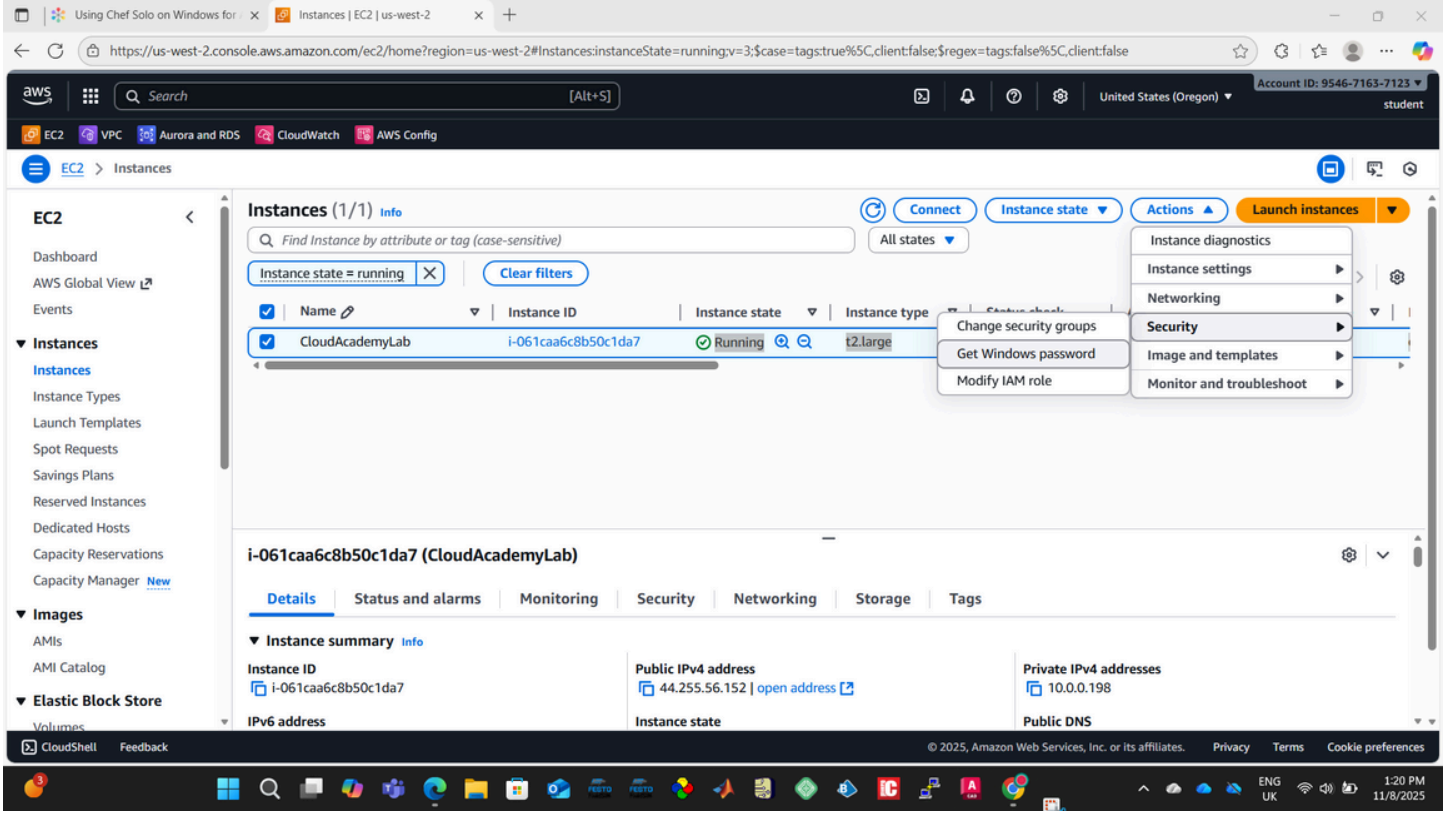
2. In the AWS Management Console search bar, enter *EC2*, and click the **EC2** result under **Services**:

3. Click the **Running Instances** link:

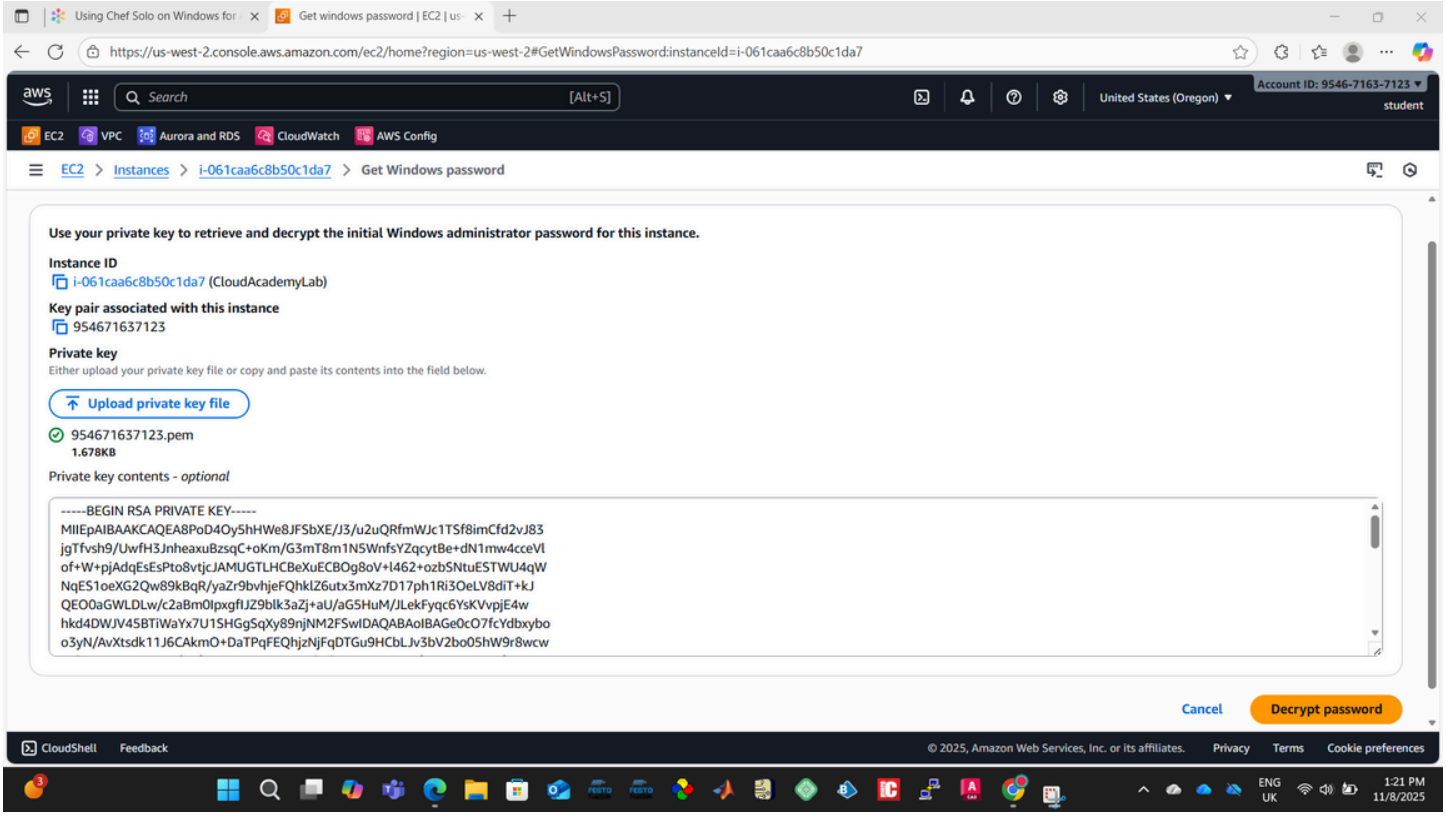
Note: It can take a few minutes for the instance to appear. Click refresh until the instance appears.

4. Select the target virtual machine and locate the **IPv4 Public IP** address. An example IPv4 address number is 52.39.32.175. Copy your virtual machine IP address for later use (for certain RDP clients):

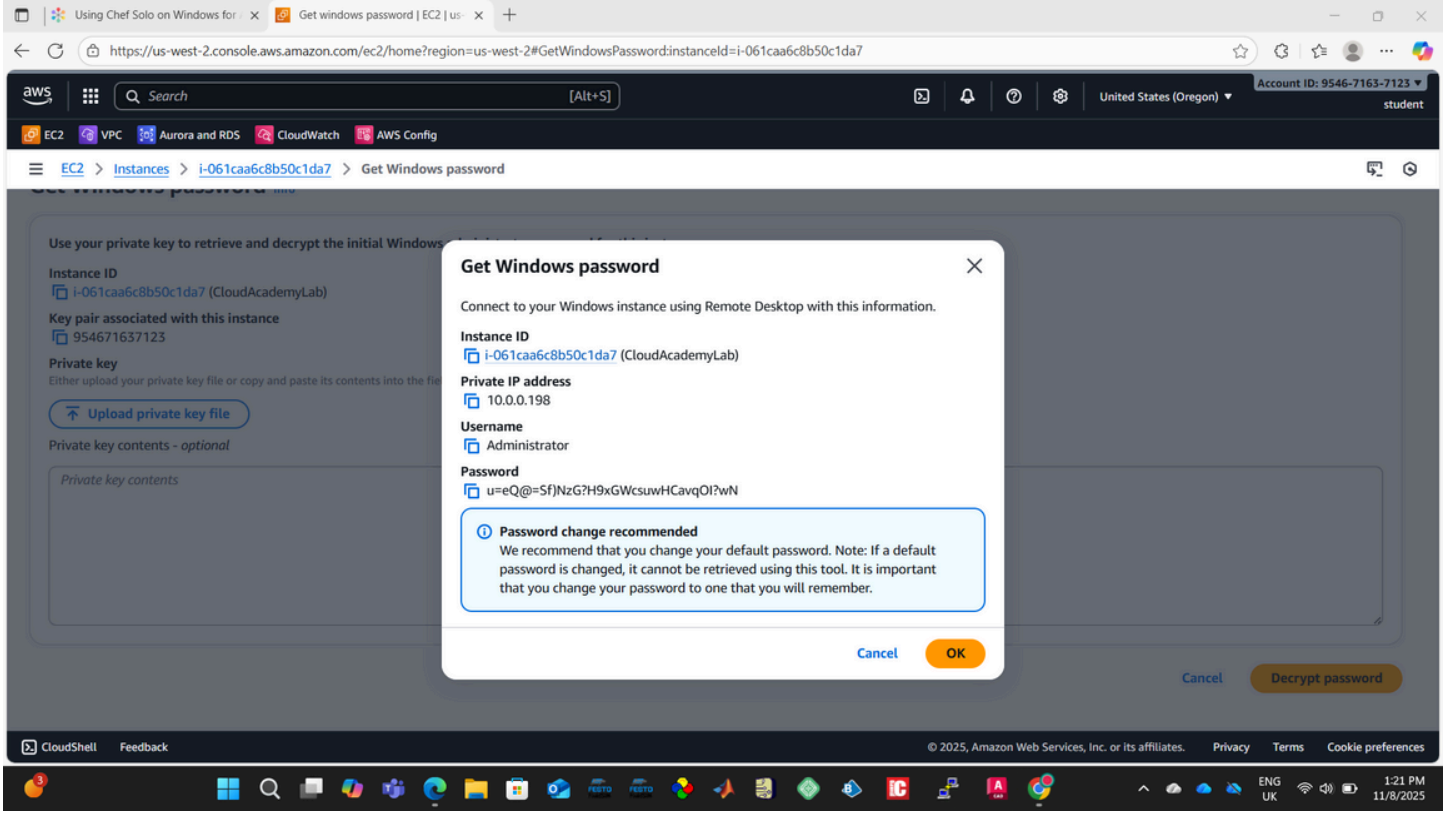
5. Click the **Actions** > **Security** > **Get Windows Password**



6. Note the **Key pair associated with this instance** in the **Get Windows password** window. Select this file after clicking the **Browse** button. Click **Decrypt Password**:



7. Copy and record the **Password**:



u=eQ@=Sf)NzG?H9xGWcsuwHCavqOI?wN

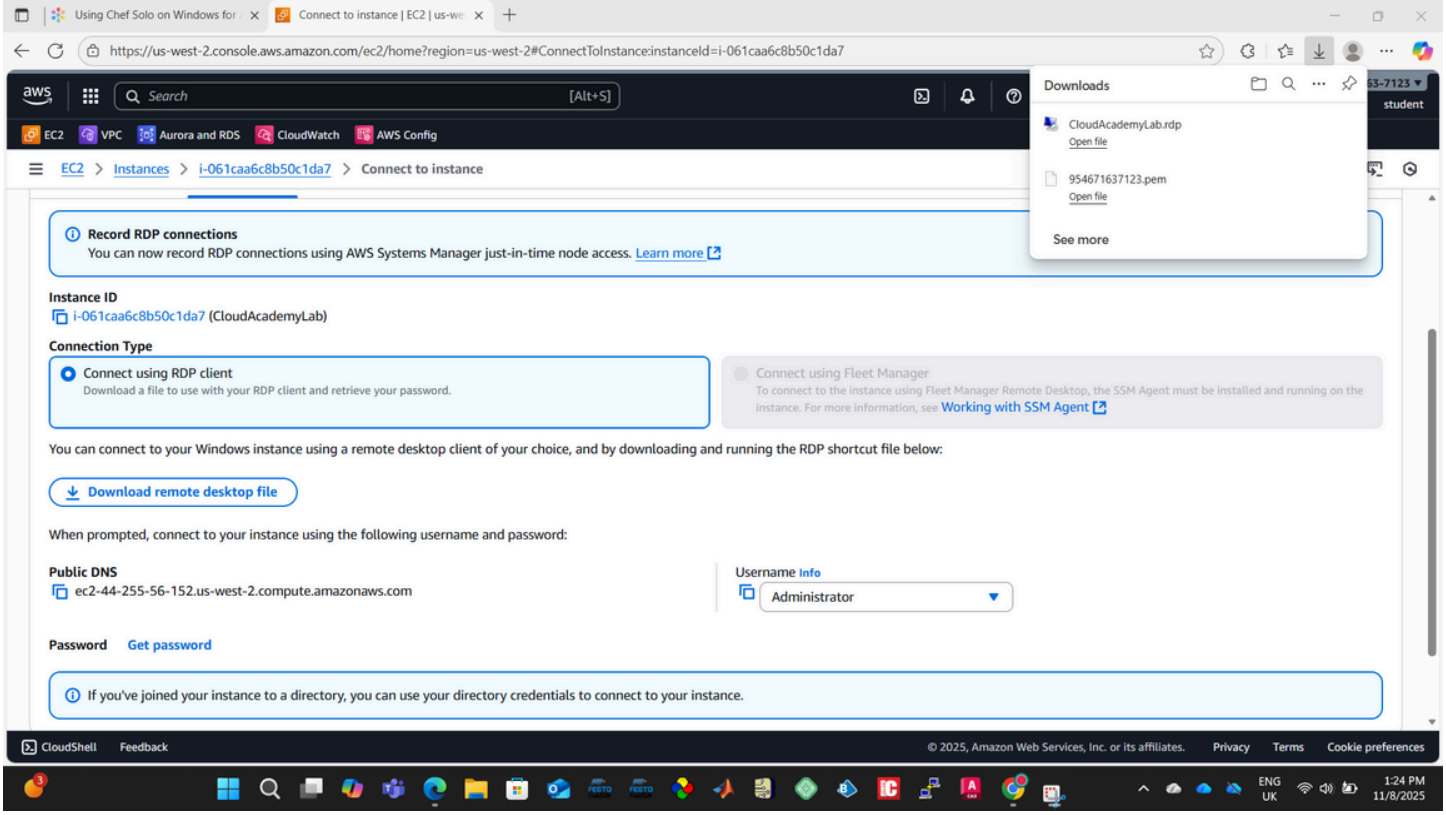
10.0.0.198

Connecting Using the Remote Desktop File

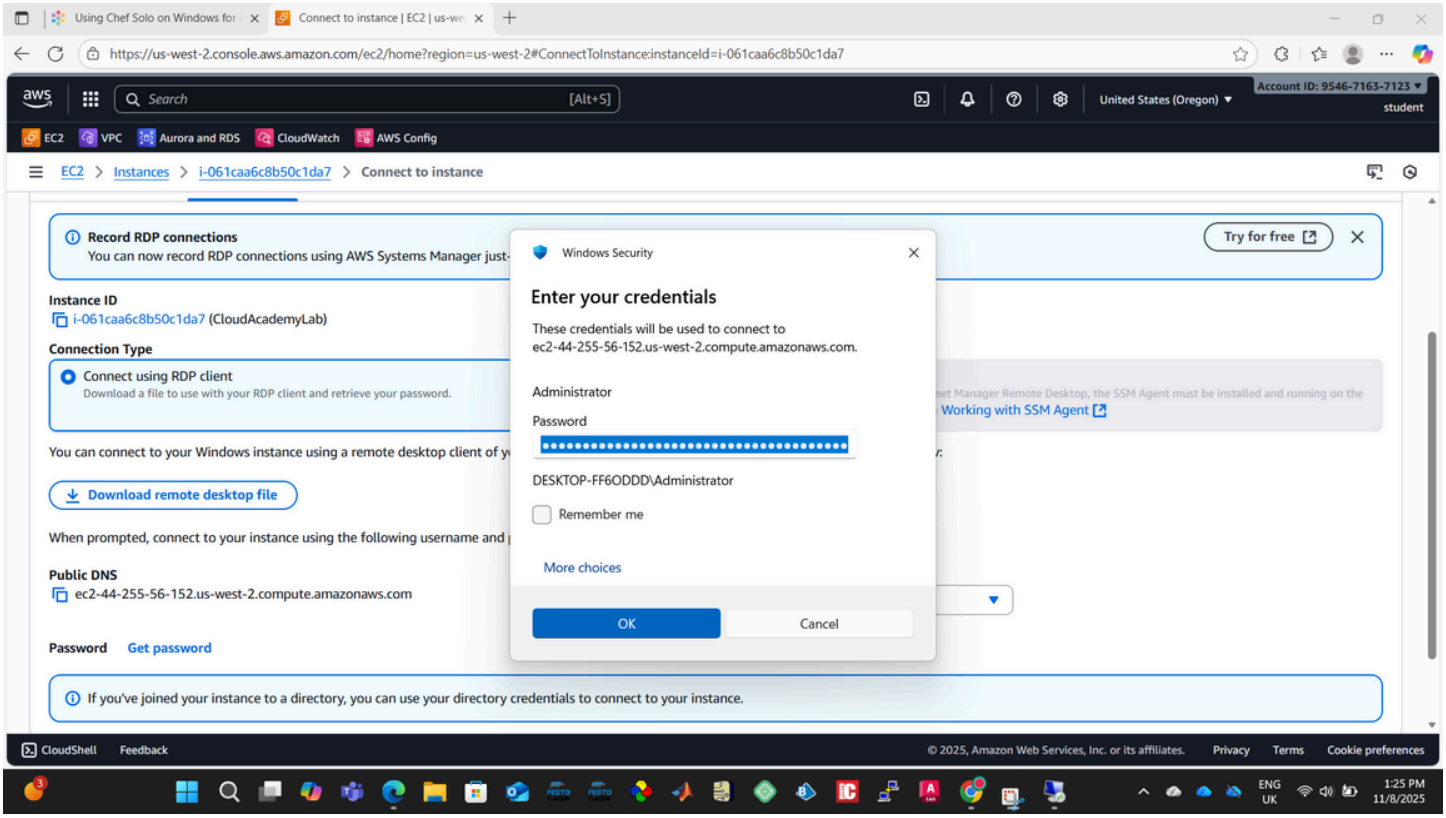
Use the Remote Desktop file to quickly connect to the virtual machine and skip several Steps in the following sections. Note that this option only works if your RDP client and operating system support .rdp files. The Windows and official Microsoft macOS clients support .rdp files.

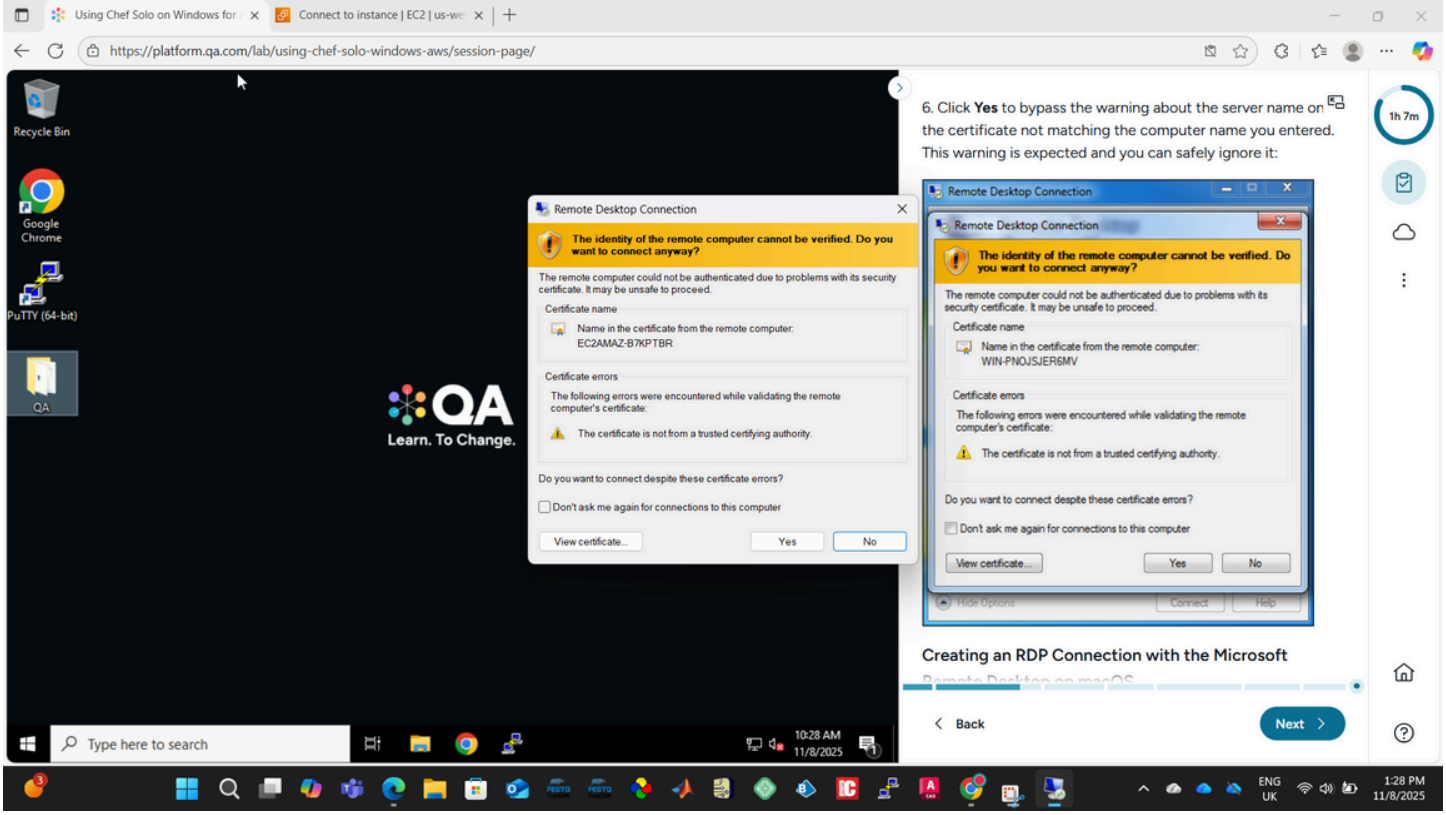
1. Return to the Instances, opens in a new tab page, select the virtual machine, and click the **Connect** button

2. Select the **RDP client tab** under **Connect to instance** and click the **Download Remote Desktop File** button.



3. Locate and launch the .rdp file to go straight to the credentials authorization screen. This screen varies slightly between clients. The only fields you must populate are entering a user of *Administrator* (if not already populated), and enter the decrypted Administrator password. Click **OK**, **Connect**, or appropriate connection authorization option.



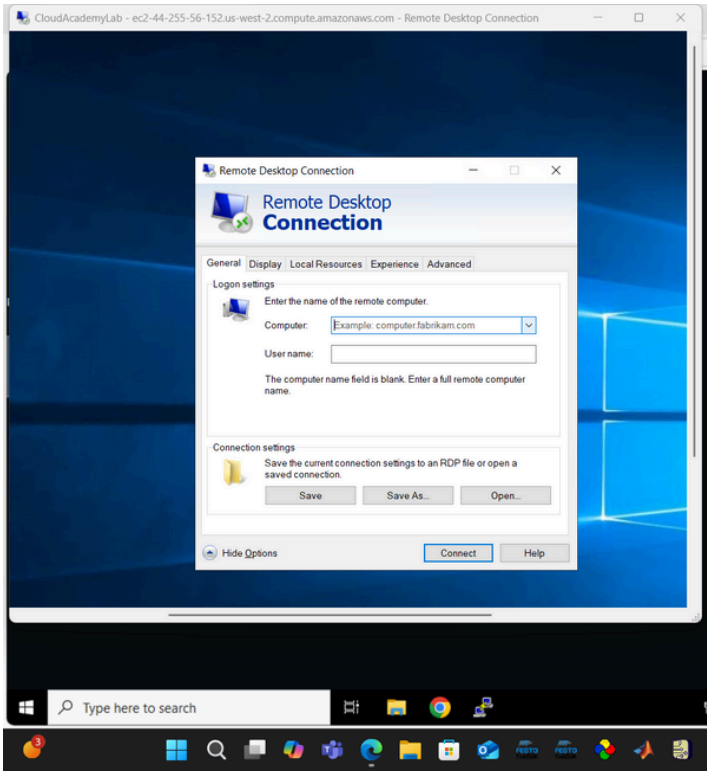


Creating an RDP Connection with the Windows Client

With the IP address and Administrator password, you can now use your RDP client to connect to the Windows virtual machine.

1. From the Start menu or Start screen, click the magnifying glass or search box and initiate a Windows search for *remote*.

2. Click the **Remote Desktop Connection** search result when it appears.



3. Click the arrow next to **Show Options**

4. Enter the IP address in the **Computer:** field and *Administrator* in the **User name:** field. Click **Connect**

5. Enter the decrypted Administrator password and click **OK**.

6. Click **Yes** to bypass the warning about the server name on the certificate not matching the computer name you entered. This warning is expected and you can safely ignore it:

Creating an RDP Connection with the Microsoft Remote Desktop on macOS

1. Open the downloaded .rdp file to connect to your VM.

You will see **Connecting to...**

Warning: If you receive a connection error, wait a minute and try again until the VM is ready to accept the connection.

2. Enter the connection information before clicking **Continue**:

5. Ignore the warning about certificate verification and click the **Continue** button:

Creating an RDP Connection with rdesktop for Linux

1. Enter the following command at a terminal prompt:

```
rdesktop -u Administrator <REMOTE_SERVER_IP_ADDRESS>
```

2. Enter the decrypted Administrator password at the Windows login prompt when it appears.

Note: It is best practice not to leave passwords in your command history.

3. Click **OK**.

4. Ignore any warnings about certificate verification and connect to the server.

Note: There are several other GNU/Linux RDP clients available, some with graphical user interfaces (GUI). Most operate on the same principles as other RDP clients. The three key pieces of information you need to record and enter are:

- Public IPv4 address or public DNS hostname
- Username, in this case Administrator
- Decrypted administrator password

Summary

In this Lab Step you decrypted the default Administrator password and connected to the Windows Server virtual machine with an RDP client. You learned how to use the .rdp file to quickly make connections to servers and how to manually create connections when necessary.

Setting Up Chef Solo on Windows

Introduction

chef-solo is a command that can execute Chef cookbooks without connecting to a Chef server. It uses the same Chef client that runs in Chef's client-server architecture. Chef Solo uses the local mode of the Chef client without any centralized server dependencies. This allows Chef Solo to use cookbooks on the local file system or via a URL instead of a centralized server. This also means that everything that would be stored on the central server needs to be stored on the node running Chef Solo.

Chef Solo was actually how Chef originally worked before support for a central server was added. You can use Chef Solo for scenarios where there is only one node to manage, when under resource constraints, and for quickly practicing with Chef. You should also know that Chef Solo doesn't support any authentication or authorization.

In this Lab Step you will install Chef Solo by installing the Chef development kit (ChefDK). Although the smaller Chef client installer is all that is required to use Chef Solo, you will use some of the features that are installed with ChefDK for convenience. You will also set up the default configuration for Chef Solo.

Instructions

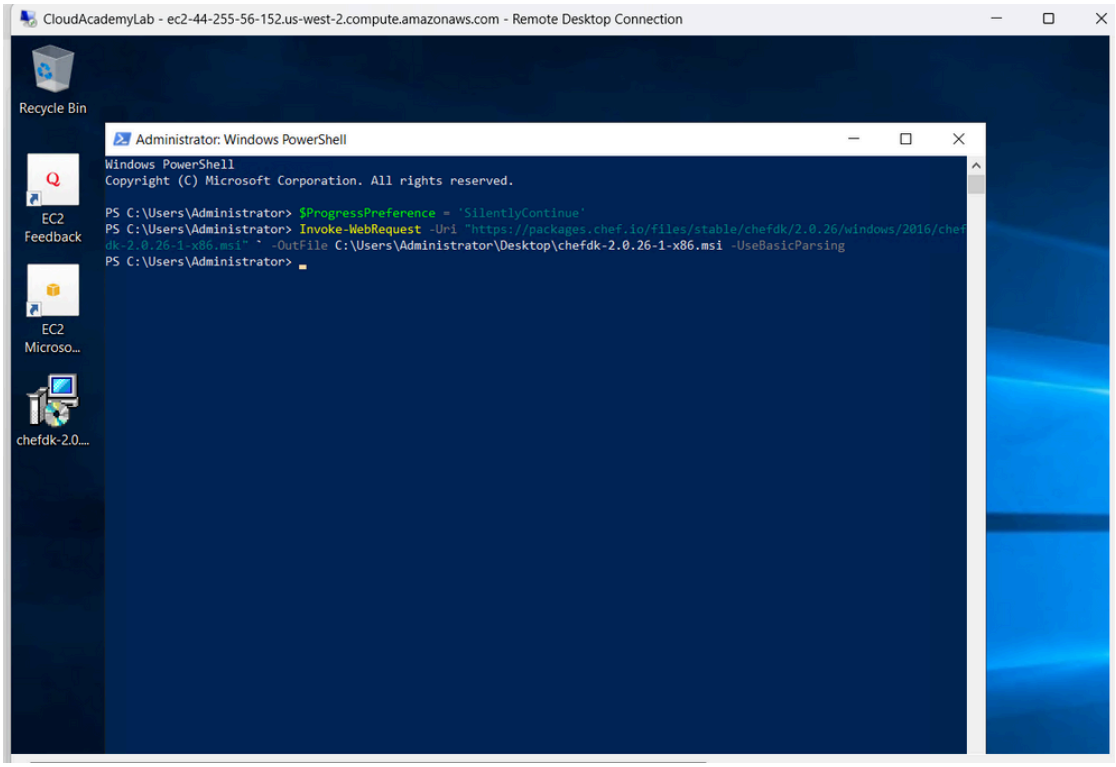
1. Open a powershell terminal by clicking the magnifying glass search icon in the lower-left corner, entering Powershell, and clicking **Windows PowerShell**:

2. Enter the following commands to download the ChefDK version 2.0.26 installer:

Copy code

```
$ProgressPreference = 'SilentlyContinue'

Invoke-WebRequest -Uri
"https://packages.chef.io/files/stable/chefdk/2.0.26/windows/2016/chefdk-2.0.26-1-x86.msi" `

            -OutFile C:\Users\Administrator\Desktop\chefdk-2.0.26-1-x86.msi -UseBasicParsing
```

The backtick ` at the end of the Invoke-WebRequest line tells PowerShell the command is spread over multiple lines.



3. Close the PowerShell window.

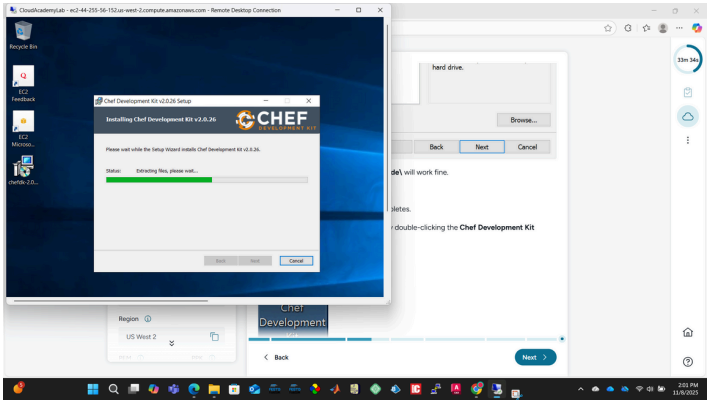
4. Double click to ChefDK installer on your desktop:

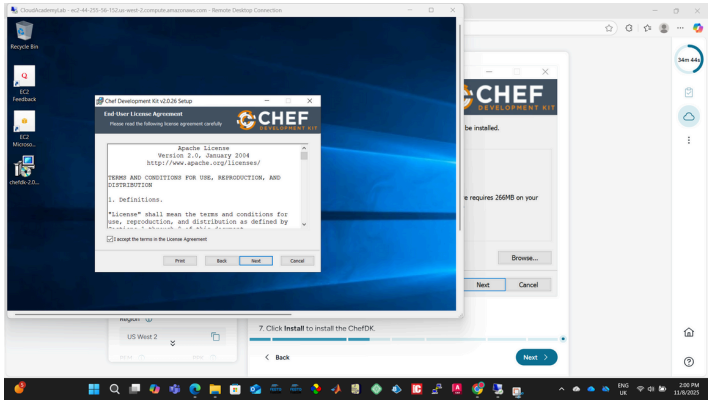
5. Check **I accept the terms in the License Agreement** and click **Next**.

6. Click **Next** on the **Custom Setup** dialog:

The default installation path of **C:\opscode** will work fine.

7. Click **Install** to install the ChefDK.





8. Click **Finish** when the installation completes.

9. Open a ChefDK PowerShell terminal by double-clicking the **Chef Development Kit** shortcut created on your Desktop:

10. Confirm that Chef Solo is installed by entering:

Copy code

```
chef-solo --version
```

The ChefDK PowerShell terminal adds all the necessary directories to your Path variable allowing you to run chef-solo from any directory.

11. Create the default Chef configuration directory and change into it:

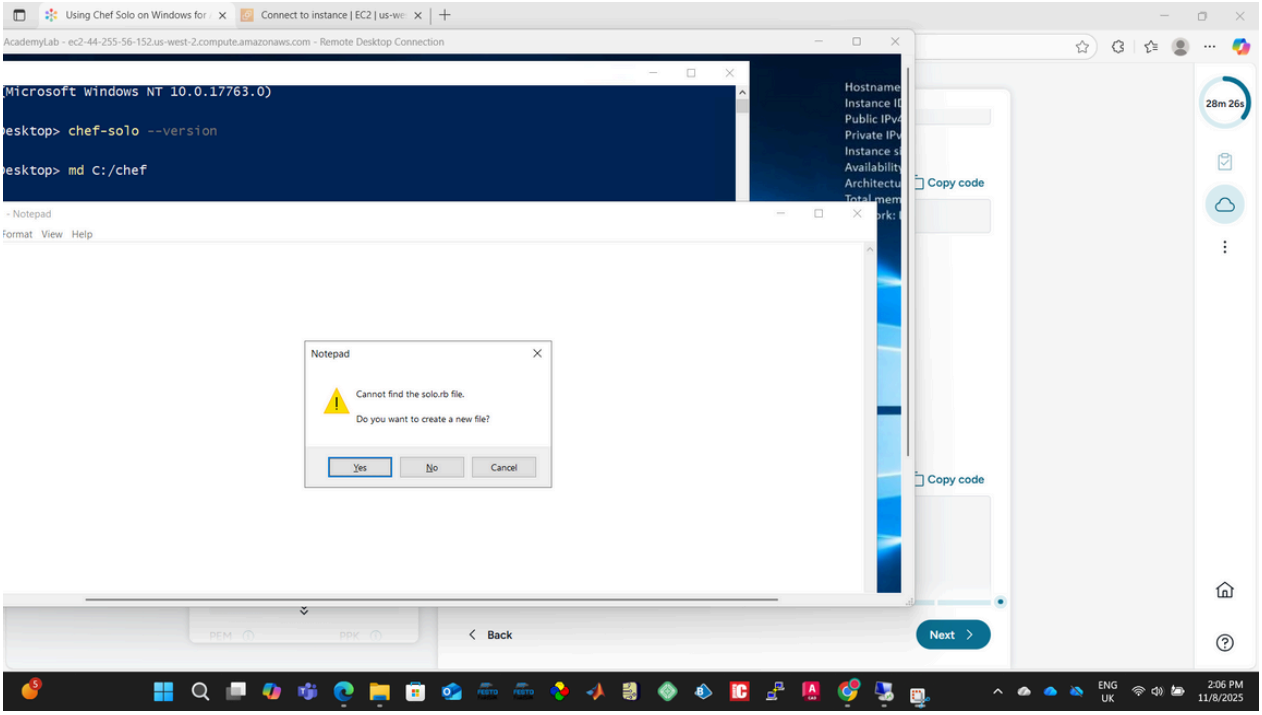
Copy code

```
md C:\chef

cd C:\chef
```

12. Create the default Chef Solo configuration file:

Copy code

```
notepad solo.rb
```



13. Click **Yes** when prompted **if you want to create a new file**:

14. Enter the following into the file:

Copy code

```
repo = 'C:\chef\repo'

# Paths

cookbook_path  repo + '\cookbooks'
```

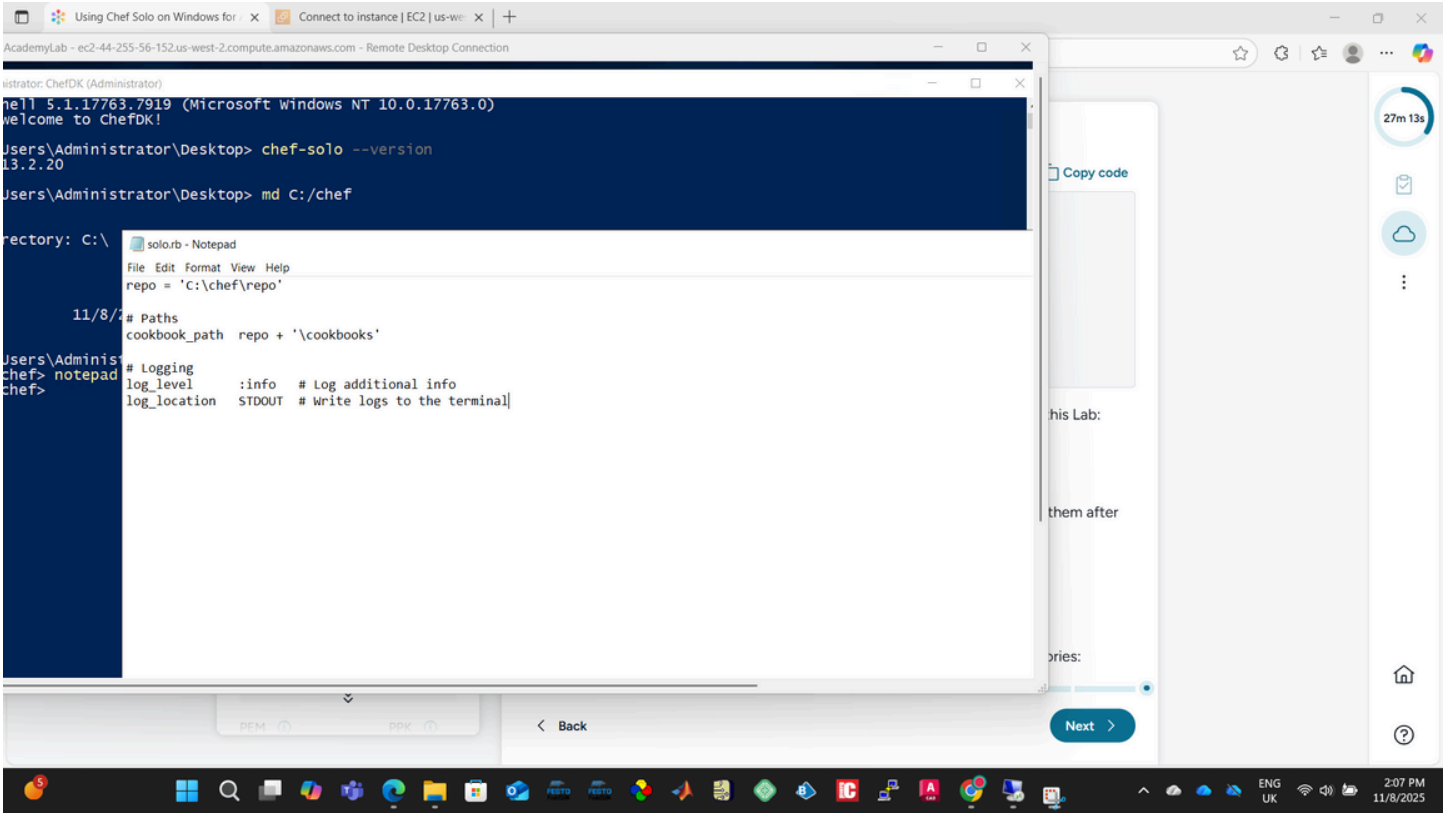
```
# Logging

log_level     :info   # Log additional info

log_location   STDOUT  # Write logs to the terminal
```

The configuration file specifies all of the Chef Solo settings that you need for this Lab:

- Paths to cookbooks and roles
- Instructions to log additional log info to the terminal

Notice that you can use Ruby variables, like repo, and operations in the file.

15. Save the file by clicking on **File > Save** and close it by clicking **File > Exit**.



16. Return to the command prompt and create the cookbooks and roles directories:

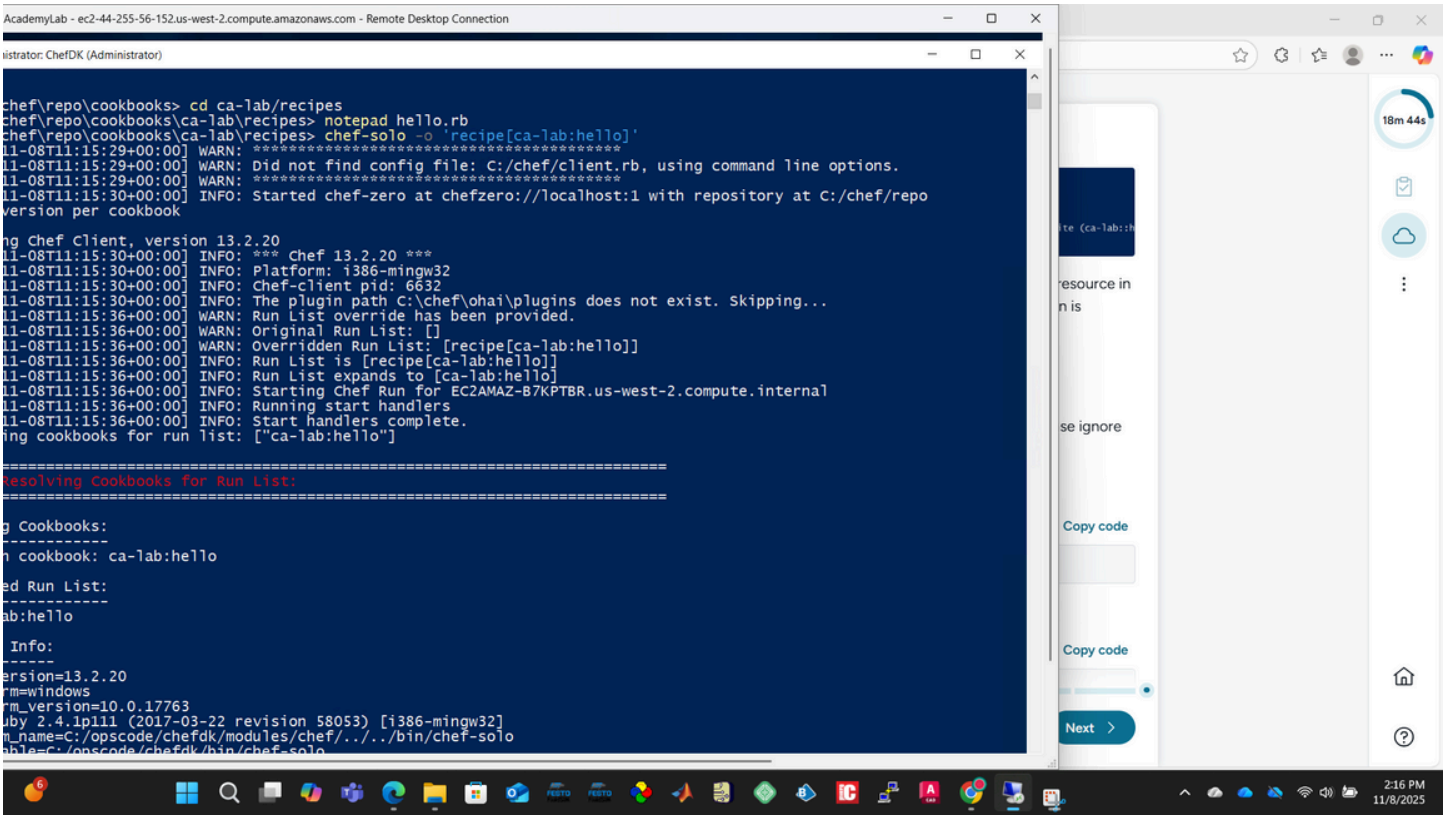
Copy code

```
md repo\cookbooks
```

Now you are ready to use Chef Solo.

Summary

In this Lab Step, you installed the Chef client, which includes Chef Solo. You also created the default Chef Solo configuration file C:\chef\solo.rb. You declared the location for your cookbook files and made the directory in preparation of future Lab Steps.



Running Your First Chef Solo Recipes on Windows

Introduction

In this Lab Step, you will create your first Chef cookbook. You will add a few recipes to the cookbook and run them with Chef Solo.

Instructions

1. Change into the cookbooks directory of your repository:

Copy code

```
cd repo\cookbooks
```

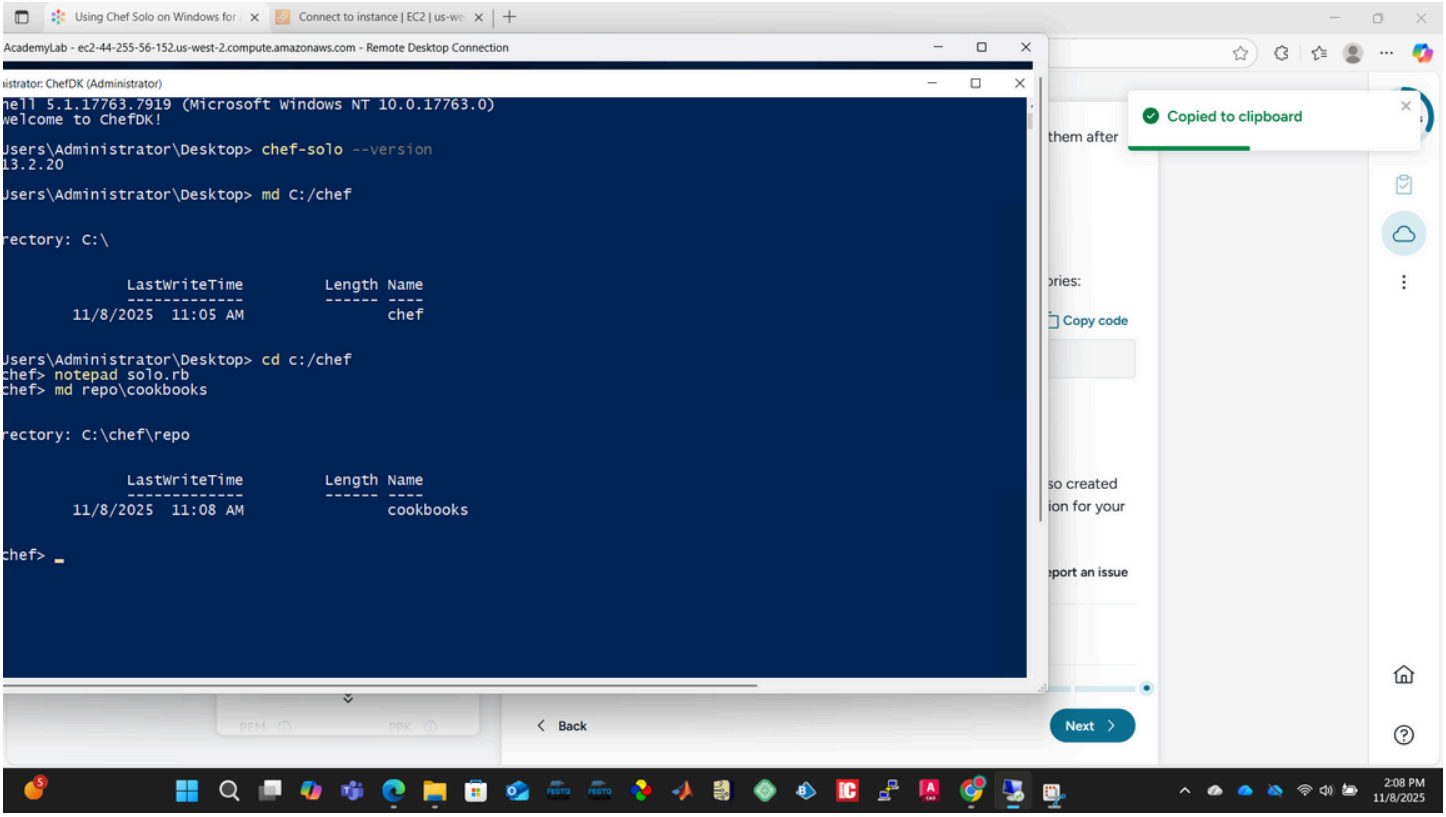
2. Create the ca-lab cookbook, a recipes subdirectory, and change into the recipes subdirectory:

Copy code

```
md ca-lab\recipes
```

```
cd ca-lab\recipes
```

Note that a cookbook is just a directory with a specific structure. You can use Chef's knife utility to quickly create a template cookbook with all the files and directories added, for example knife cookbook create ca-lab. In this Lab you will build out the cookbook step by step and avoid any unnecessary clutter.



3. Create a recipe file called hello.rb:

Copy code

```
notepad hello.rb
```

4. Enter the following Chef recipe in the file:

Copy code

```
log 'message' do
  message  'Hello Chef'
  level   :info
end
```

5. Save the file and exit the editor.

6. Run the recipe with Chef Solo:

Copy code

```
chef-solo -o 'recipe[ca-lab::hello]'
```

The -o specifies the run list, which lists the recipes to run. You will see the following in the output:

```
Synchronizing Cookbooks:
  - ca-lab (0.0.0)
Installing Cookbook Gems:
Compiling Cookbooks...
Converging 1 resources
Recipe: ca-lab::hello
  * log[message] action write[2017-07-11T05:58:11+00:00] INFO: Processing log[message] action write (ca-lab::h
ello line 1)
[2017-07-11T05:58:11+00:00] INFO: Hello Chef
```

This shows when Chef Solo is reading the **ca-lab** cookbook and converging the resource in the **hello** recipe. The line beginning with *** log[message]** shows the default action is **write** and the last line is the message you specified written to the log.

The rest of the output is useful for understanding how Chef Solo works. You are encouraged to quickly read the other **INFO** messages.

Note: you will see some warnings before and after the Hello Chef message: please ignore them as they won't interfere with the lab.

7. Create another recipe:

Copy code

```
notepad editor.rb
```

8. Enter the following into the file:

Copy code

```
windows_package 'notepad++' do

  source 'https://github.com/notepad-plus-plus/notepad-plus-plus/releases/download/v7.8.8/npp.7.8.8.Installer.x64.exe'

end
```

This uses the windows_package resource to install a Microsoft Installation (MSI) package for the editor Notepad++. The resource uses the default action of :install and the source attribute is a URL where the package is located. windows_package is one of the built-in Chef resources dedicated to the Windows platform. Others include: batch, chocolatey_package, dsc_script, powershell_script, and registry_key.

9. Save the file and exit Notepad.

10. Run the editor recipe:

Copy code

```
chef-solo -o 'recipe[ca-lab::editor]'
```

You will see the following within the output telling you that the package was downloaded and installed:

```
[2017-07-11T20:14:36+00:00] INFO: remote_file[c:\chef\local-mode-cache\cache\package\npp.7.4.2.Installer.x64.exe] create
d file c:\chef\local-mode-cache\cache\package\npp.7.4.2.Installer.x64.exe

      - create new file c:\chef\local-mode-cache\cache\package\npp.7.4.2.Installer.x64.exe[2017-07-11T20:14:36+00:00] IN
FO: remote_file[c:\chef\local-mode-cache\cache\package\npp.7.4.2.Installer.x64.exe] updated file contents c:\chef\local-
mode-cache\cache\package\npp.7.4.2.Installer.x64.exe

      - update content in file c:\chef\local-mode-cache\cache\package\npp.7.4.2.Installer.x64.exe from none to bb89c0
      (new content is binary, diff output suppressed)
[2017-07-11T20:14:42+00:00] INFO: windows_package[notepad++] installed notepad++ at latest
    - install version latest of package notepad++
```

The green text allows you to easily see the actions Chef Solo performed.

11. Open Notepad++, and click **No** if asked to update:

Copy code

```
start 'C:\Program Files\Notepad++\notepad++.exe'
```

You will use Notepad++ for editing Chef files due to several advantages. It includes Ruby syntax highlighting and tabs to edit multiple files.

Summary

In this Lab Step, you ran your first Chef Solo recipes on Windows. The first demonstrated the log resource and the second demonstrated using windows_package to install an MSI package for the Notepad++ editor.

Preserving Chef Resource Idempotence with Guards

Introduction

An important concept to understand when working with Chef is *idempotence*. A resource is idempotent if it won't try to execute when it is already in the desired state. Idempotence is important for several reasons including:

- Re-executing resources that already converged to their desired state has a negative performance impact
- Re-executing resources can potentially break the configuration of the system

Many Chef resources are idempotent. Some are not idempotent by their nature. For example, running a command inside of a script resource is not idempotent as they are typically unique to the environment in which they are run. Other resources may not be idempotent only because of limitations in verifiying the state of a resource.

You can use *guards* when you want to enforce idempotence on resources that are not idempotent. In general, guards are used to evaluate the state of a resource. The resource will

execute or not based on how the guard evaluates. Guards are declared as attributes of a resource. There are two guard attributes:

- **not_if**: Use to prevent a resource from executing when the condition returns true
- **only_if**: Use to allow a resource to execute only if the condition returns true

In this Lab Step you will see how idempotence works and learn how to preserve idempotence for resources that do not support it automatically.

Instructions

1. In Notepad++, create a new recipe by clicking **File > New**.

2. Let Notepad++ know it is a Ruby file by going to **File > Save As...** and entering the following for the **File name**:

Copy code

```
C:\chef\repo\cookbooks\ca-lab\recipes\file.rb
```

3. Enter the following recipe in the file and press ctrl+s to save the file:

Copy code

```
file 'C:\Users\Administrator\Desktop\idempotent.txt' do

  content 'Do not edit this file!'

  rights :write, 'Administrator'

end
```

The file resource's name is used as the path for the file, by default. The default action is to create the file. The content attribute defines the contents of the file and the rights define any access rights to the file. By default, the Administratoruser has write permission as an Administrator, but you will see how this rights definition can ensure the Administrator isn't denied write permission later.

4. Execute the recipe from the PowerShell terminal:

Copy code

```
chef-solo -o 'recipe[ca-lab::file]'
```

You will see output similar to the following:

```
    - create new file C:\Users\student\Desktop\idempotent.txt[2017-07-11T21:59:01+00:00] INFO: file[C:\Users\student\Des
ktop\idempotent.txt] updated file contents C:\Users\student\Desktop\idempotent.txt

    - update content in file C:\Users\student\Desktop\idempotent.txt from none to 0898f8
    --- C:\Users\student\Desktop\idempotent.txt 2017-07-11 21:59:01.000000000 +0000
    +++ C:\Users\student\Desktop/chef-idempotent20170711-2220-y0xyj5.txt        2017-07-11 21:59:01.000000000 +0000
    @@ -1 +1,2 @@
    +Do not edit this file![2017-07-11T21:59:01+00:00] INFO: file[C:\Users\student\Desktop\idempotent.txt] permissions c
hanged to [ca-lab-vm\student/flags:0/mask:40000000]

    - change dacl
```

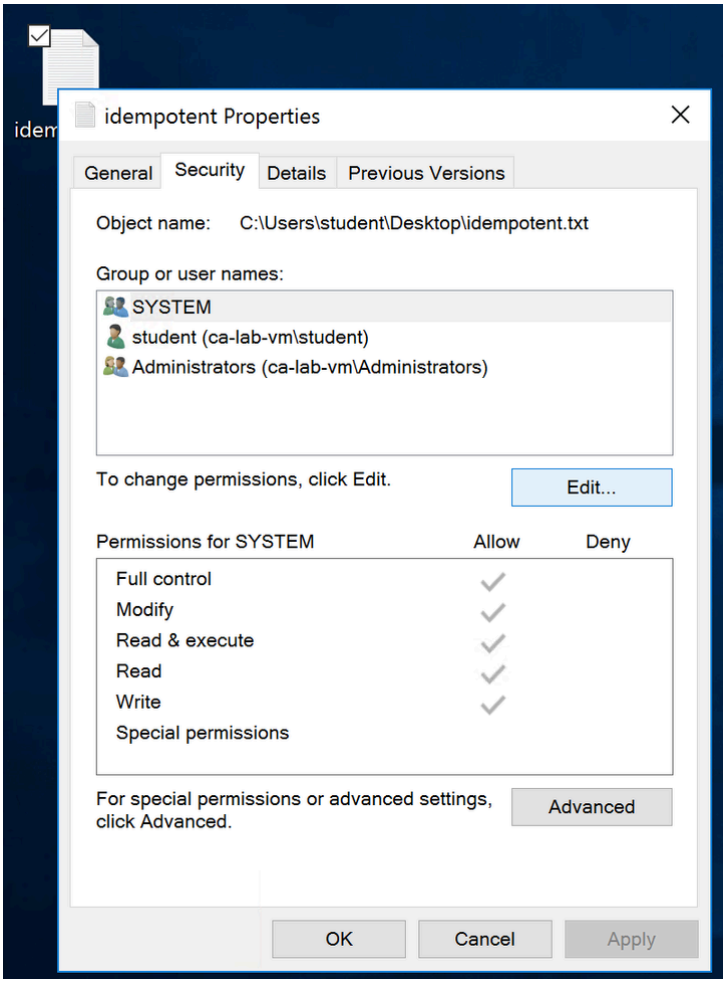
The **change dacl** action refers to changing permissions in Windows' Discretionary Access Control List (DACL).

5. Demonstrate the idempotency of the file resource by running the recipe again and observing the output.

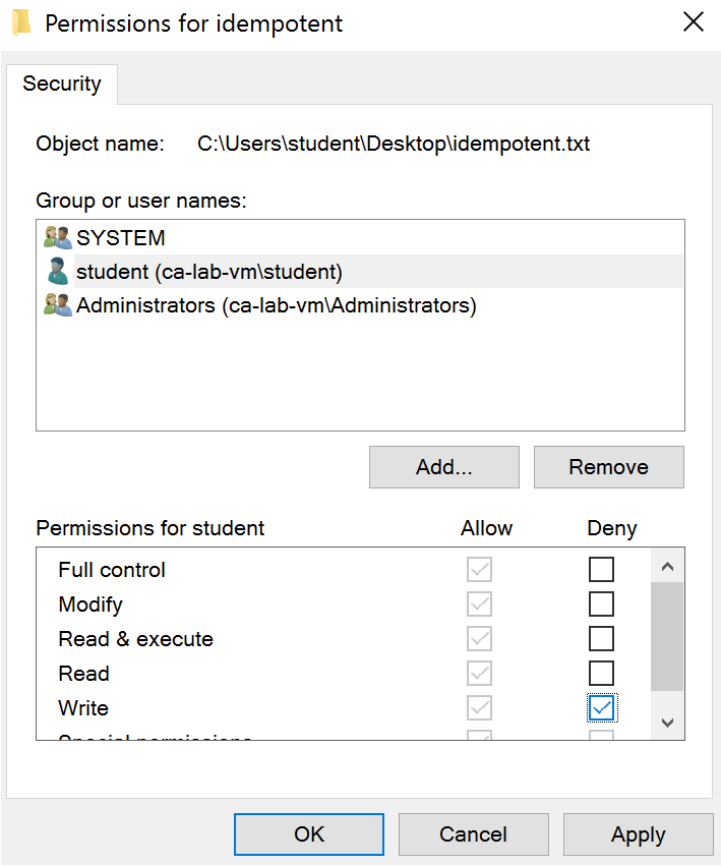
You will notice no green actions are displayed. Only an **(up to date)** message is written because the resource is idempotent.

6. To demonstrate how the file resource can converge back to its desired state when it detects changes, edit the content of the file by double-clicking the file on your Desktop and saving the file when you are finished.

7. Modify the Administrator file permissions by right-clicking on the file icon, selecting **Properties**, clicking the **Security** tab, and clicking **Edit**:



8. In the Edit dialog, select the **Administrator(**Administrator**)** user in the upper list and check **Deny** for the **Write** row in the **Permissions** table.



9. Click **OK** to save the permission change, click **Yes** to accept that you are denying permissions in the **Windows Security** pop-up, and click **OK** again to close the **Properties** dialog.

If you try to modify the file again, you will be denied access when attempting to save the file.

10. Re-execute the file recipe.

This time Chef detects the changes in the file contents and permissions and performs actions to converge the file resource to its desired state:

The single - and + symbols indicate the lines removed and added to converge the file contents to its desired state. You can also verify that you have been granted write permission to the idempotent.txt file.

11. Now re-execute the editor recipe:

Copy code

```
chef-solo -o 'recipe[ca-lab::editor]'
```

Do you notice anything unexpected in the output? Even though Notepad++ is already installed, Chef will notify you that it performed the action **install version latest of package notepad++**. You will also see **Chef Client finished, 1/2 resources updated**. The installation

package **remote_file** resource is the one that didn't get updated. Chef uses the if-modified-since HTTP header to determine if the installation package has changed since it was originally downloaded. Chef fails to verify the package is installed and then installs the package again as a result.

Note: You can set the log_level to :debug in the C:\chef\solo.rb file to see more log messages that show how Chef determines if a resource has changed.

12. Open the editor.rb recipe in Notepad++ by clicking **File > Open** and entering *C:\chef\repo\cookbooks\ca-lab\recipes\editor.rb* as the **File name**.

13. Add an only_if guard that checks whether the notepad++.exe file exists in it's default location and declare **powershell_script** as the **guard_interpreter** (The file changes are shown in bold):

Copy code

```
windows_package 'notepad++' do
  source 'https://notepad-plus-plus.org/repository/7.x/7.4.2/npp.7.4.2.Installer.x64.exe'
guard_interpreter :powershell_script
  not_if "Test-Path 'C:\\Program Files\\Notepad++\\notepad++.exe'"
end
```

The guard_interpreter instructs Chef to evaluate the guard using PowerShell. Recall that powershell_script is a Chef resource dedicated to the Windows platform. powershell_script is the most common guard interpreter when using Windows.

Note: You need to carefully choose the guard conditions you use. The guard chosen here can easily be fooled but will work for the environment you have set up so far.

14. Save the editor recipe, run it again, and confirm that no resources are updated.

Summary

In this Lab Step, you demonstrated idempotence using a Chef file resource. You also used a not_if guard to make the editor recipe idempotent. You will use another resource to install packages from a package manager that is easier to work with and more resilient.

Managing Chef Cookbook Dependencies with Berkshelf

Introduction

The Chef community has created a wide variety of cookbooks to cover many common configuration management scenarios. The Chef Supermarket, opens in a new tab is where the

community shares their cookbooks. You can use the Berkshelf dependency manager to easily take advantage of the community cookbooks. To use Berkshelf, you will need to add two files to your cookbook:

- **metadata.rb**: This is where you put metadata abour your cookbook, including its dependencies
- **Berksfile**: Tell Berkshelf where to look for cookbooks and where to find the dependencies

You will use Berkshelf to install the Chocolatey Windows package manager and use the chocolatey_package resource to install the Google Chrome web browser.

Instructions

1. In Notepad++ create a new file and save it as *C:\chef\repo\cookbooks\ca-lab\metadata.rb*.

2. Enter the following contents and save the file:

Copy code

```
name 'ca-lab' # Name of your cookbook

chef_version '>= 12.7' # Minimum Chef version

depends 'chocolatey', '~> 1.2.0'  # ca-lab depends on chocolatey cookbook
```

The text after # are Ruby comments to explain the file contents.

3. Create a new file with the following contents and save it as *C:\chef\repo\cookbooks\ca-lab\Berksfile* and be sure to select **All types (*.*)** in the **Save as type** drop-down menu to avoid Notepad++ adding .txt on the end:

Copy code

```
source 'https://supermarket.chef.io'

metadata

cookbook 'chocolatey', '~> 1.2.0'
```

The source tells Berkshelf to use the public Chef Supermarket for finding cookbooks. The metadata line loads the local cookbooks metadata.rb file and processes any dependencies specified in it. The cookbook line declares a dependency on the chocolatey cookbook between version 1.2.0 and 2.0.0. Multiple dependencies can be declared in a Berksfile.

4. Enter the following commands into your ChefDK PowerShell terminal to change into your cookbook root directory and install the dependency using Berkshelf:

Copy code

```
cd C:\chef\repo\cookbooks\ca-lab\
```

```
berks install
```

The output will show that three cookbooks are installed: **chocolatey**, **windows**, and **ohai**. The chocolatey cookbook depends on the windows cookbook, which in turn depends on the ohai cookbook. Berkshelf makes it easy to manage all the dependencies. The cookbooks are downloaded to your user's .berkshelf directory (C:\Users\Administrator\.berkshelf\cookbooks).

5. Create a new recipe in Notepad++ with the following contents and saved as *C:\chef\repo\cookbooks\ca-lab\recipes\browser.rb*:

Copy code

```
include_recipe 'chocolatey'

chocolatey_package 'googlechrome'
```

The include_recipe instruction includes the chocolatey default recipe in the recipe, which installs Chocolatey on the system. The second line installs Google Chrome using Chocolatey.

6. Vendor/group your cookbook and the cookbooks you depend on into a single cookbook:

Copy code

```
berks vendor C:\chef\repo\cookbooks\vendored
```

In order for Chef Solo to locate the included chocolatey recipe, it needs to know where to find it. Vendoring the cookbooks is one way to overcome this. The vendored cookbook is created in C:\chef\repo\ca-lab\berks-cookbooks.

7. Change the default cookbook path in your C:\chef\solo.rb to the vendored cookbook:

Copy code

```
cookbook_path  repo + '\cookbooks\vendored'
```

8. Run the browser recipe:

Copy code

```
chef-solo -o 'recipe[ca-lab::browser]'
```

After a minute the recipe will complete and a Google Chrome shortcut will appear on your Desktop:

Double-click the Google Chrome icon to launch the program and verify Chocolatey and Google Chrome installed successfully. Compared to the windows_package resource, it is more

convenient to work with package names and versions with chocolatey_package instead of using URLs to reach MSI files. Also, if you re-run the recipe you can verify that the chocolatey_package resource is idempotent without any guard.

Summary

In this Lab Step, you used Berkshelf to manage your cookbook dependencies. You added a dependency on Chocolatey and used it to install Google Chrome. You also learned how to vendor cookbooks to allow Chef Solo to locate included recipes.

Configuring Microsoft's IIS Web Server with Chef Solo - AWS

Introduction

In this Lab Step, you will create the ca-lab default recipe that installs and configures IIS web server to run a custom website. You will add a dependency on the Chef Supermarket IIS cookbook to simplify the process.

Instructions

1. Add a dependency on IIS in your Berksfile:

Copy code

```
cookbook 'iis', '~> 6.7.2'
```

The default recipe in the IIS cookbook, opens in a new tab installs IIS on the host. It also includes a variety of resources that make managing IIS with Chef possible. You will use the iis_site resource which can manage IIS sites.

2. Declare the dependency in your metadata.rb file:

Copy code

```
depends 'iis', '~> 6.7.2'
```

3. Create a cookbook files directory for storing your website files:

Copy code

```
md C:\chef\repo\cookbooks\ca-lab\files
```

4. Create and save an *index.html* file for your website in the files directory with the following contents:

Copy code

```
<!DOCTYPE html>
```

```html
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Cloud Academy Chef Lab Page</title>
  </head>
  <body>
    Come smell <a href="https://cloudacademy.com/library/?q=chef" target="_blank">what's cooking with Chef at Cloud Academy</a>
  </body>
</html>
```

5. Create the ca-lab default recipe *C:\chef\repo\cookbooks\ca-lab\recipes\default.rb* with the following contents:

Copy code

```ruby
include_recipe 'iis' # default iis recipe installs IIS


site_name = "ca-lab"


# first, the physical path for the site files must exist
directory "#{node['iis']['docroot']}\\#{site_name}" do
  action :create
end


# second, add the index page cookbook file
cookbook_file "#{node['iis']['docroot']}\\#{site_name}\\index.html" do
  source "index.html"
end
```

```
# third, add and start the site

# binding to http port 8080

iis_site "#{site_name} site" do

  bindings "http/*:8080:"

  path "#{node['iis']['docroot']}\\#{site_name}"

  action [:add,:start]

end
```

6. Vendor the cookbooks:

Copy code

```
berks vendor C:\chef\repo\cookbooks\vendored
```

7. Run the default ca-lab recipe:

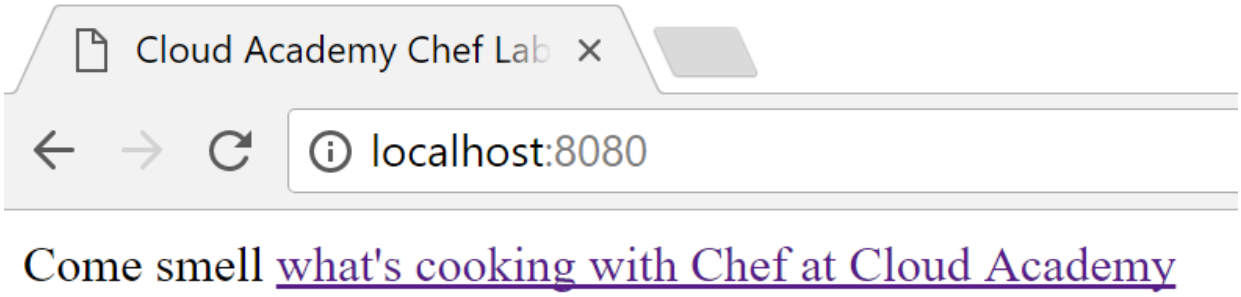
Copy code

```
chef-solo -o 'recipe[ca-lab]'
```

Sift through the output paying particular attention to the actions Chef performs in executing the recipe.

Note: If you don't see any output after two minutes, try pressing enter to coax it out.

8. Open Google Chrome and navigate to *localhost:8080*:



Summary

In this Lab Step, you configured Microsoft's IIS web server using Chef Solo. You used a Chef Supermarket cookbook for IIS and added a dependency to your Berksfile for Berkshelf to manage. You created a cookbook file for your website index page. You tied everything together in a default recipe for the ca-lab cookbook that uses:

- The default IIS recipe to install IIS
- A Chef directory resource to create the physical path of the site
- A Chef cookbook_file for the site index page
- An iis_site resource from the IIS cookbook to create the site

Now you have configured a Windows host with multiple packages and configured a web server using Chef Solo. After you complete the Lab, see what other Chocolatey packages, opens in a new tab and Chef Supermarket Windows cookbooks, opens in a new tab would be useful for your Windows configuration management needs.