

Automating Azure virtual machine deployment with Chef

05/30/2017 7 minutes to read Contributors  all

In this article

Chef basics

Preparing your workstation

Setup Chef Server

Install Chef Workstation

Creating a Cookbook

Creating a template

Upload the Cookbook to the Chef Server

Deploy a virtual machine with Knife Azure

Note

Azure has two different deployment models for creating and working with resources: **Resource Manager and classic**. This article covers using both models, but Microsoft recommends that most new deployments use the Resource Manager model.

Chef is a great tool for delivering automation and desired state configurations.

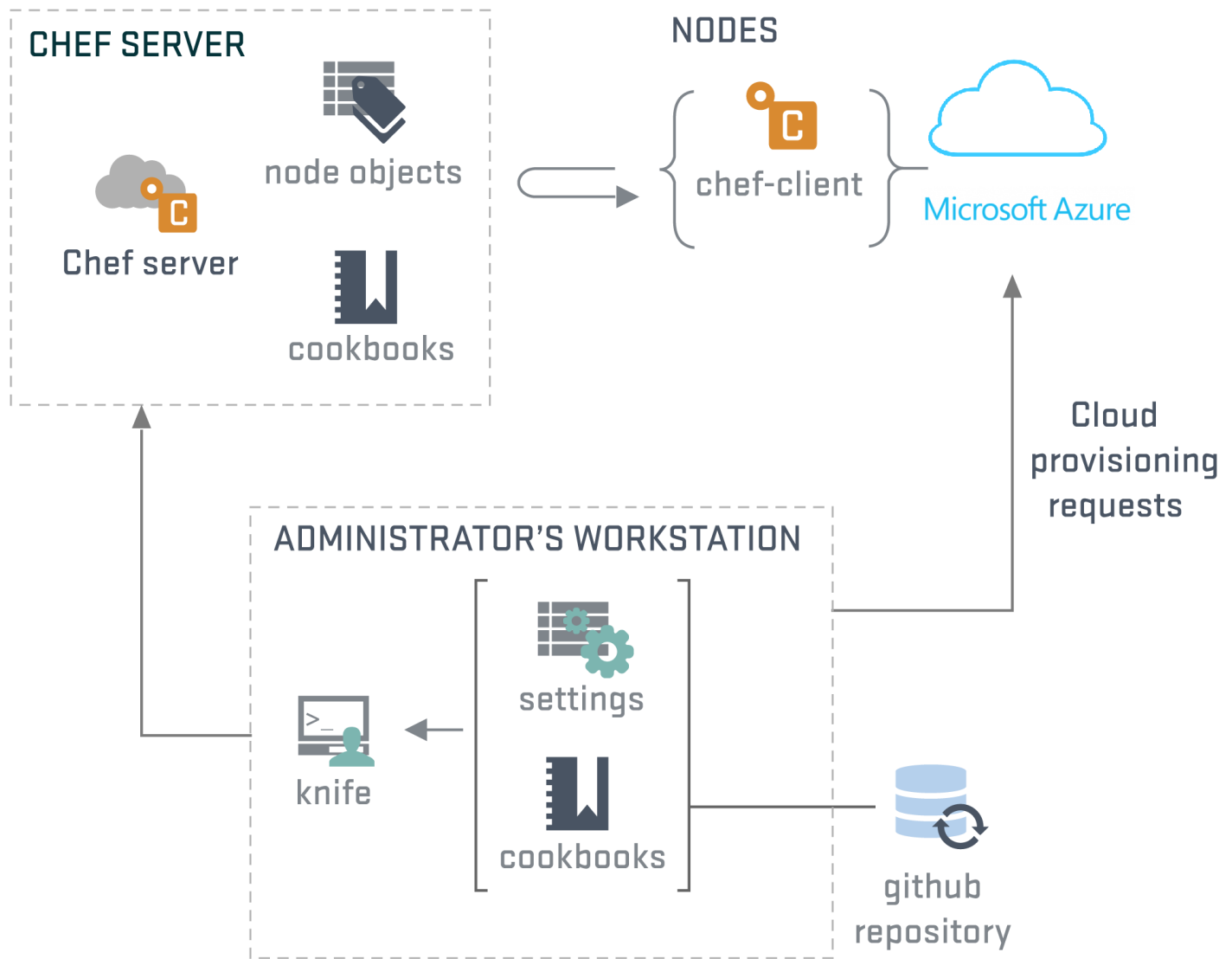
With the latest cloud API release, Chef provides seamless integration with Azure, giving you the ability to provision and deploy configuration states through a single command.

In this article, you set up your Chef environment to provision Azure virtual machines and walk through creating a policy or “Cookbook” and then deploying this cookbook to an Azure virtual machine.

Chef basics

Before you begin, [review the basic concepts of Chef](#).

The following diagram depicts the high-level Chef architecture.



Chef has three main architectural components: Chef Server, Chef Client (node), and Chef Workstation.

The Chef Server is the management point and there are two options for the Chef Server: a hosted solution or an on-premises solution.

The Chef Client (node) is the agent that sits on the servers you are managing.

The Chef Workstation, which is the name for both the admin workstation where you create policies and execute management commands and the software package of Chef tools.

Generally, you'll see *your workstation* as the location where you perform actions and *Chef Workstation* for the software package. For example, you'll download the knife command as part of *Chef Workstation*, but you run knife commands from *your workstation* to manage infrastructure.

Chef also uses the concepts of "Cookbooks" and "Recipes", which are effectively the policies we define and apply to the servers.

Preparing your workstation

First, prep your workstation by creating a directory to store Chef configuration files and cookbooks.

Create a directory called C:\chef.

Download your Azure PowerShell [publish settings](#).

Setup Chef Server

This guide assumes that you will sign up for Hosted Chef.

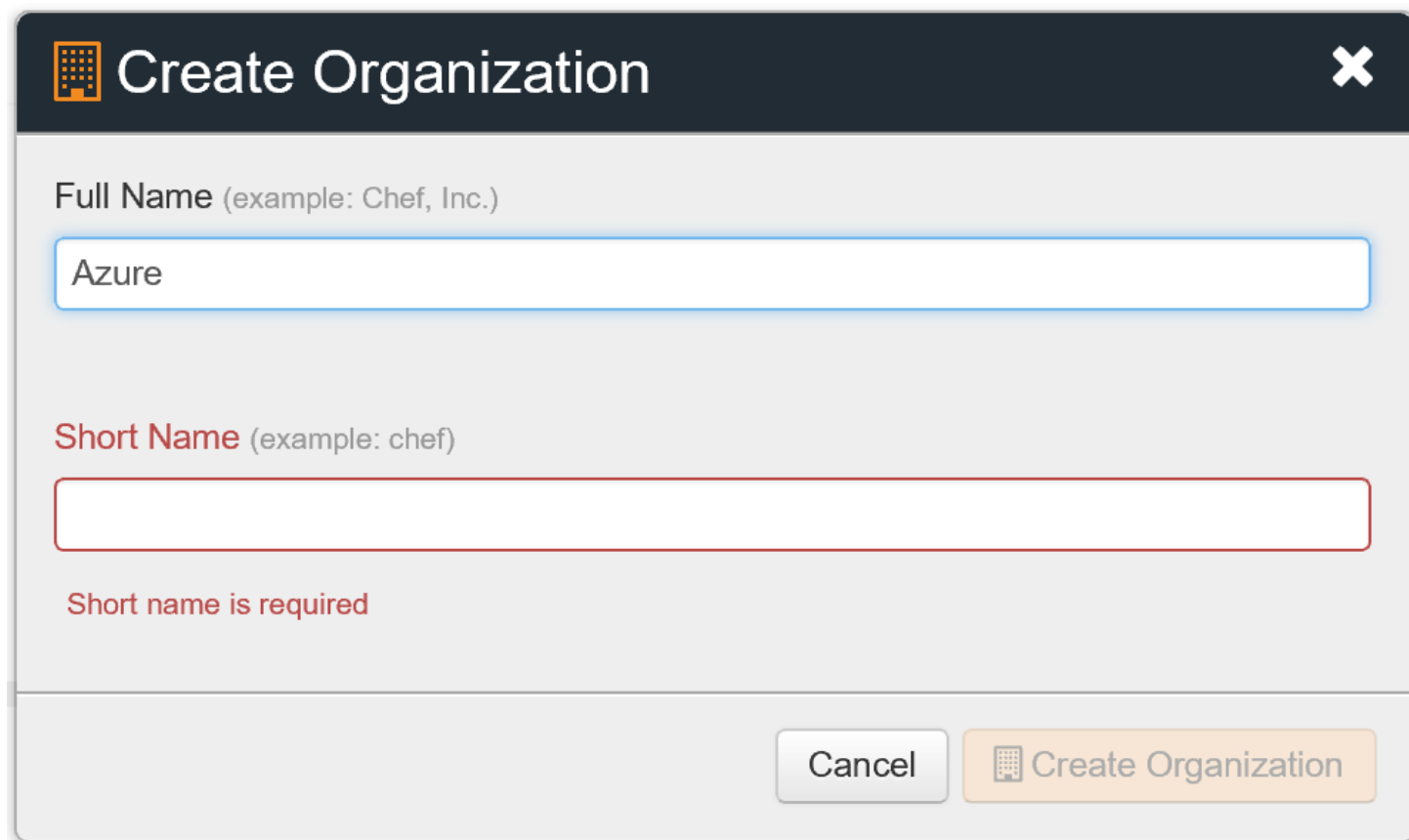
If you're not already using a Chef Server, you can:

- Sign up for Hosted Chef, which is the fastest way to get started with Chef.
- Install a standalone Chef Server on linux-based machine, following the installation instructions from Chef Docs.

Creating a Hosted Chef account

Sign up for a Hosted Chef account [here](#).

During the sign-up process, you will be asked to create a new organization.



Create Organization

Full Name (example: Chef, Inc.)

Azure

Short Name (example: chef)

Short name is required

Cancel Create Organization

Once your organization is created, download the starter kit.

Note

If you receive a prompt warning you that your keys will be reset, it's okay to proceed as we have no existing infrastructure configured as yet.

This starter kit zip file contains your organization configuration files and user key in the `.chef` directory.

The `organization-validator.pem` must be downloaded separately, because it is a private key and private keys should not be stored on the Chef Server. From [Chef Manage](#) and select "Reset Validation Key", which provides a file for you to download separately. Save the file to `c:\chef`.

Configuring your Chef workstation

Extract the content of the `chef-starter.zip` to `c:\chef`.

Copy all files under `chef-starter\chef-repo.chef` to your `c:\chef` directory.

Copy the `organization-validator.pem` file to `c:\chef`, if it is saved in `c:\Downloads`

Your directory should now look something like the following example.

PowerShell

Copy

```
Directory: C:\Users\username\chef
```

```
Mode                LastWriteTime         Length Name
----                -

```

```

d----- 12/6/2018 6:41 PM .chef
d----- 12/6/2018 5:40 PM chef-repo
d----- 12/6/2018 5:38 PM cookbooks
d----- 12/6/2018 5:38 PM roles
-a----- 12/6/2018 5:38 PM 495 .gitignore
-a----- 12/6/2018 5:37 PM 1678 azuredocs-validator.pem
-a----- 12/6/2018 5:38 PM 1674 user.pem
-a----- 12/6/2018 5:53 PM 414 knife.rb
-a----- 12/6/2018 5:38 PM 2341 README.md

```

You should now have five files and four directories (including the empty chef-repo directory) in the root of c:\chef.

Edit knife.rb

The PEM files contain your organization and administrative private keys for communication and the knife.rb file contains your knife configuration. We will need to edit the knife.rb file.

Open the knife.rb file in the editor of your choice. The unaltered file should look something like:

```
rb
```

Copy

```

current_dir = File.dirname(__FILE__)
log_level    :info
log_location STDOUT
node_name    "mynode"
client_key   "#{current_dir}/user.pem"
chef_server_url "https://api.chef.io/organizations/myorg"
cookbook_path ["#{current_dir}/cookbooks"]

```

Add the following information to your knife.rb:

```
validation_client_name "myorg-validator" validation_key ""#{current_dir}/myorg.pem"
```

Also add the following line reflecting the name of your Azure publish settings file.

```
knife[:azure_publish_settings_file] = "yourfilename.publishsettings"
```

Copy

Modify the "cookbook_path" by removing the ../ from the path so it appears as:

```
cookbook_path ["#{current_dir}/cookbooks"]
```

Copy

These lines will ensure that Knife references the cookbooks directory under `c:\chef\cookbooks`, and also uses our Azure Publish Settings file during Azure operations.

Your `knife.rb` file should now look similar to the following example:

```
current_dir = File.dirname(__FILE__)
log_level      :info
log_location   STDOUT
node_name      [REDACTED]
client_key     "#{current_dir}/[REDACTED].pem"
validation_client_name "a-[REDACTED]-validator"
validation_key  "#{current_dir}/a-[REDACTED]-validator.pem"
chef_server_url "https://api.opscode.com/organizations/a-[REDACTED]"
cache_type     'BasicFile'
cache_options( :path => "#{ENV['HOME']}/.chef/checksums" )
cookbook_path  ["#{current_dir}/cookbooks"]
knife[:azure_publish_settings_file] = "[REDACTED].publishsettings"
```

rb

Copy

```
knife.rb
current_dir = File.dirname(__FILE__)
log_level      :info
log_location   STDOUT
node_name      "myorg"
client_key     "#{current_dir}/user.pem"
chef_server_url "https://api.chef.io/organizations/myorg"
validation_client_name "myorg-validator"
validation_key  "#{current_dir}/myorg.pem"
cookbook_path  ["#{current_dir}/cookbooks"]
knife[:azure_publish_settings_file] = "yourfilename.publishsettings"
```

Install Chef Workstation

Next, [download and install](#) Chef Workstation. Install Chef Workstation the default location. This installation may take a few minutes.

On the desktop, you'll see a "CW PowerShell", which is an environment loaded with the tool you'll need for interacting with the Chef products. The CW PowerShell makes new ad-hoc commands available, such as `chef-run` as well as traditional Chef CLI commands, such as `chef`. See your installed version of Chef Workstation and the Chef tools with `chef -v`. You can also check your Workstation version by selecting "About Chef Workstation" from the Chef Workstation App.

`chef --version` should return something like:

```
Chef Workstation: 0.2.29
  chef-run: 0.2.2
  Chef Client: 14.6.47x
  delivery-cli: master (6862f27aba89109a9630f0b6c6798efec56b4efe)
  berks: 7.0.6
  test-kitchen: 1.23.2
  inspec: 3.0.12
```

Note

The order of the path is important! If your opscore paths are not in the correct order you will have issues.

Reboot your workstation before you continue.

Install Knife Azure

This tutorial assumes that you're using the Azure Resource Manager to interact with your virtual machine.

Install the Knife Azure extension. This provides Knife with the "Azure Plugin".

Run the following command.

```
chef gem install knife-azure --pre
```

Note

The `--pre` argument ensures you are receiving the latest RC version of the Knife Azure Plugin which provides access to the latest set of APIs.

It's likely that a number of dependencies will also be installed at the same time.

```
Command Prompt

c:\Chef>chef gem install knife-azure --pre

Temporarily enhancing PATH to include DevKit...
Building native extensions. This could take a while...
Successfully installed eventmachine-1.0.4
Successfully installed em-winrm-0.6.0
Successfully installed winrm-s-0.2.2
Successfully installed knife-windows-0.8.2
Fetching: knife-azure-1.4.0.rc.0.gem (100%)
Successfully installed knife-azure-1.4.0.rc.0
Parsing documentation for eventmachine-1.0.4
Installing ri documentation for eventmachine-1.0.4
Parsing documentation for em-winrm-0.6.0
Installing ri documentation for em-winrm-0.6.0
Parsing documentation for winrm-s-0.2.2
Installing ri documentation for winrm-s-0.2.2
Parsing documentation for knife-windows-0.8.2
Installing ri documentation for knife-windows-0.8.2
Parsing documentation for knife-azure-1.4.0.rc.0
Installing ri documentation for knife-azure-1.4.0.rc.0
Done installing documentation for eventmachine, em-winrm, winrm-s, knife-windows, knife-azure after 16 seconds
5 gems installed
```

To ensure everything is configured correctly, run the following command.

[Copy](#)

```
knife azure image list
```

If everything is configured correctly, you will see a list of available Azure images scroll through.

Congratulations. Your workstation is set up!

Creating a Cookbook

A Cookbook is used by Chef to define a set of commands that you wish to execute on your managed client. Creating a Cookbook is straightforward, just use the **chef generate cookbook** command to generate the Cookbook template. This cookbook is for a web server that automatically deploys IIS.

Under your C:\Chef directory run the following command.

[Copy](#)

```
chef generate cookbook webserver
```

This command generates a set of files under the directory C:\Chef\cookbooks\webserver. Next, define the set of commands for the Chef client to execute on the managed virtual machine.

The commands are stored in the file default.rb. In this file, define a set of commands that installs IIS, starts IIS and copies a template file to the wwwroot folder.

Modify the C:\chef\cookbooks\webserver\recipes\default.rb file and add the following lines.

[Copy](#)


```
powershell_script 'Install IIS' do
  action :run
  code 'add-windowsfeature Web-Server'
end

service 'w3svc' do
  action [ :enable, :start ]
end

template 'c:\inetpub\wwwroot\Default.htm' do
  source 'Default.htm.erb'
  rights :read, 'Everyone'
end
```

Save the file once you are done.

Creating a template

In this step, you'll generate a template file to used as the default.html page.

Run the following command to generate the template:

[Copy](#)

```
chef generate template webserver Default.htm
```

Navigate to the `C:\chef\cookbooks\webserver\templates\default\Default.htm.erb` file. Edit the file by adding some simple "Hello World" HTML code, and then save the file.

Upload the Cookbook to the Chef Server

In this step, you make a copy of the Cookbook that you have created on the local machine and upload it to the Chef Hosted Server. Once uploaded, the Cookbook appears under the **Policy** tab.

[Copy](#)

```
knife cookbook upload webserver
```

Nodes		Reports	Policy	Administration
Showing All Cookbooks				
Cookbook		Current Version		
webserver		0.1.2		

Deploy a virtual machine with Knife Azure

Deploy an Azure virtual machine and apply the “Webserver” Cookbook which installs the IIS web service and default web page.

In order to do this, use the **knife azure server create** command.

An example of the command appears next.

[Copy](#)

```
knife azure server create --azure-dns-name 'diegotest01' --azure-vm-name 'testserver01' --az
```

The parameters are self-explanatory. Substitute your particular variables and run.

Note

Through the command line, I’m also automating my endpoint network filter rules by using the `--tcp-endpoints` parameter. I’ve opened up ports 80 and 3389 to provide access to my web page and RDP session.

Once you run the command, go to the Azure portal to see your machine begin to provision.

testserver01	** Starting	Visual Studio Ultimate with MSDN	Southeast Asia	diegotest01.cloudapp.net
--------------	-------------	----------------------------------	----------------	---

The command prompt appears next.

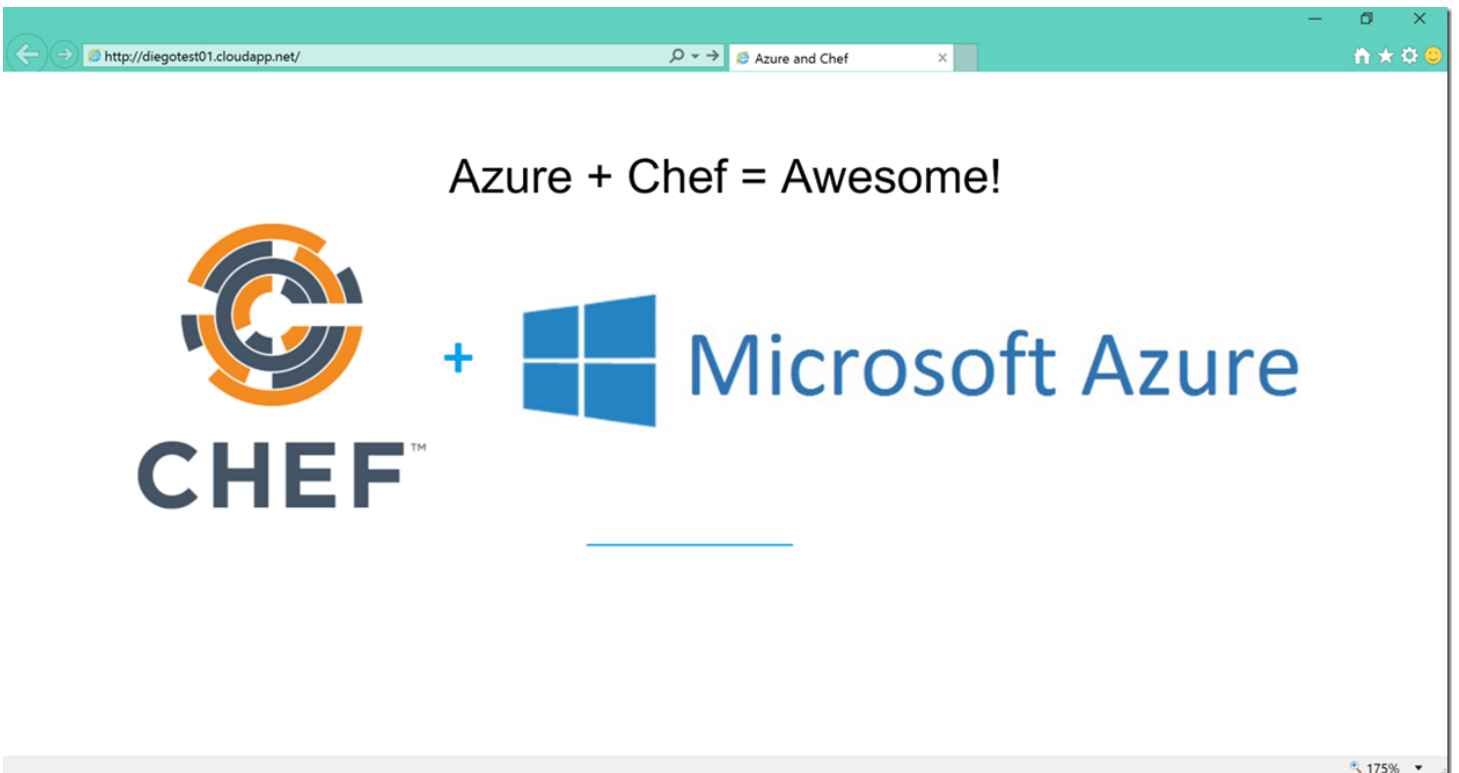
```

c:\Chef>knife azure server create --azure-dns-name 'diegotest01' --azure-vm-name 'testserver01' --azure-vm-size 'Small' --azure-storage-account 'portal' --bootstrap-protocol 'cloud-api' --azure-source-image 'a699494373c04fc0bc8f2bb1389d6106__Windows-Server-2012-Datacenter-201411.01-en.us-127GB.vhd' --azure-service-location 'Southeast Asia' --winrm-user azureuser --winrm-password 'myPassword123' --tcp-endpoints 80,3389 --r 'recipe[webserver]'
.....
Waiting for virtual machine to reach status 'provisioning'.....vm state 'provisioning' reached after 2.79 minutes.
Waiting for virtual machine to reach status 'ready'.....vm state 'ready' reached after 2.23 minutes.
DNS Name: diegotest01.cloudapp.net
VM Name: testserver01
Size: Small
Azure Source Image: a699494373c04fc0bc8f2bb1389d6106__Windows-Server-2012-Datacenter-201411.01-en.us-127GB.vhd
Azure Service Location: Southeast Asia
Public Ip Address: 104.43.9.88
Private Ip Address: 100.72.52.27
WinRM Port: 5985
TCP Ports: [{"Name"=>"tcpport_3389_testserver01", "Uip"=>"104.43.9.88", "PublicPort"=>"3389", "LocalPort"=>"3389"}, {"Name"=>"tcpport_80_testserver01", "Uip"=>"104.43.9.88", "PublicPort"=>"80", "LocalPort"=>"80"}]
Environment: _default
Runlist: ['recipe[webserver]']

Waiting for Resource Extension to reach status 'wagent provisioning'....Resource extension state 'wagent provisioning' reached after 0.09 minutes.
Waiting for Resource Extension to reach status 'provisioning'.....Resource extension state 'provisioning' reached after 2.02 minutes.
Waiting for Resource Extension to reach status 'ready'.....Resource extension state 'ready' reached after 4.86 minutes.
DNS Name: diegotest01.cloudapp.net
VM Name: testserver01
Size: Small
Azure Source Image: a699494373c04fc0bc8f2bb1389d6106__Windows-Server-2012-Datacenter-201411.01-en.us-127GB.vhd
Azure Service Location: Southeast Asia
Public Ip Address: 104.43.9.88
Private Ip Address: 100.72.52.27
WinRM Port: 5985
TCP Ports: [{"Name"=>"tcpport_3389_testserver01", "Uip"=>"104.43.9.88", "PublicPort"=>"3389", "LocalPort"=>"3389"}, {"Name"=>"tcpport_80_testserver01", "Uip"=>"104.43.9.88", "PublicPort"=>"80", "LocalPort"=>"80"}]
Environment: _default
Runlist: ['recipe[webserver]']

```

Once the deployment is complete, you should be able to connect to the web service over port 80, because you opened the port when you provisioned the virtual machine with the Knife Azure command. As this virtual machine is the only virtual machine in this cloud service, connect to it with the cloud service url.



This example uses creative HTML code.

Don't forget you can also connect through an RDP session from the Azure portal via port 3389.