Automating Web App Deployments with Opscode Chef and iControl

George Watkins, 2011-08-07

Chef is a systems integration framework developed here in Seattle by Opscode. It provides a number of configuration management facilities for deploying systems rapidly and consistently. For instance, if you want 150 web servers configured identically (or even with variances), Chef can make that happen. It also curtails the urge to make "one-off" changes to individual hosts or to skip over checking those configuration changes into revision control. Chef will revert any changes made out-of-band upon its next convergence. As a former systems administrator with "OCD-like" tendencies, these features make me happy.

We were introduced to the folks at Opscode through a mutual friend and we got to chatting about their products and ours. Eventually the topic of Ruby emerged (Chef is built on Ruby). We started tossing around ideas about how to use Ruby to make Chef and BIG-IP a big happy family. What if we could use Chef to automatically add our web servers to an LTM pool as they are built? Well, that's exactly what we did.

We wrote a Chef recipe to automatically add our nodes to our pool. We were able to combine this functionality with the Apache cookbook provided by the Opscode Community and create a role that handles all these actions simultaneously. Combine this with your PXE installation and you've got a highly efficient system for building loads of web servers in a hurry.

Chef Basics

Chef consists of a number of different components, but we will deduce them collectively to the Chef server, Chef client, and knife, the command-line tool. Chef also provides access to configurations via a management console (web interface) that provides all the functionality of knife in a GUI, but I prefer the command-line, so that's what we'll be covering.

Chef uses cookbooks of recipes to perform automated actions against its nodes (clients). The recipe houses the logic for how resources (pre-defined and user-defined) should perform actions against the nodes. A few of the more common resources are file, package, cron, execute, and Ruby block. We could define a resource for anything though and that is what makes Chef so powerful: its extensibility. Using recipes and resources we can perform sweeping changes to system, but they aren't very "personal" at this point. That is where attributes come into play. Attributes define the node specific settings that are "personalize" the recipe for that node or class of nodes.

Once we have cookbooks to support our node configurations, we can group those recipes into roles. For instance we might want to build a "base_server" role that should be applied to all of my servers regardless of their specialized purpose. This "base_server" role might include recipes for installing and configuring OpenSSH, NTP, and VIM. We would then create a "web_server" role that installs and configure Apache and Tomcat. Our "database_server" role would install MySQL and load my default database. If we wanted to take this a step further, we could organize these roles into environments, so that we could rapidly deploy development, staging, and production servers. This makes building up and tearing down environments very efficient.

That was a very short introduction to Chef and its features. For more information on the basics of Chef, check out this Opscode wiki entry.

Chef meets F5's Ruby iControl Library

Now that we've got a fair number of web servers built with our "web_server" role, we need them to start serving traffic. We could go to our LTM and add them all manually, but that wouldn't be any fun would it? Wouldn't it be cool if we could somehow auto-populate our LTM pool with our new web servers? This is where things get cool.

We created a Chef cookbook called "f5-node-initiator" that we can add to our server roles. Whenever the node receives this recipe, it will automatically install our f5-icontrol gem, copy an "f5-node-initiator" script to /usr/local/bin/, and add the node to the LTM defined in attributes section of the server's role.

Chef Installation

The installation of Chef server and it constituents is a topic beyond the scope of this article. The Opscode folks have assembled a great quick start guide, which they update regularly. We followed this guide and had no trouble getting things up and running. Using Ubuntu 10.04 LTS, the install was exceptionally easy using Aptitude (apt-get) to install the chef-server and chef packages on the Chef server and client(s), respectively. After installing the packages, we cloned the Chef sample repository, copied our user keys to the ~/.chef/ directory (covered in the quick start guide), created a knife.rb configuration file in .chef (also in quick start), finally we filled in the values in ~/chef-repo/config/rake.rb. I would encourage everyone to read the quick start guide as well as a few others here and here.

Note: Our environment was Ubuntu 10.04 LTS server installs running on a VMWare ESXi box (Intel Core i7 with 8GB of RAM). This was more than enough to run a Chef server, 10 nodes, an F5 LTM VE instance, as well a few other virtual machines.

The f5-node-initiator Cookbook

The "f5-node-initiator" cookbook (recipe can be used interchangeably here as the cookbook only contains one recipe) is relatively simple compared to some of the examples I encountered while demoing Chef. Let's look at the directory structure:

```
f5-node-initiator (dir)
  |--> attributes (dir)
       |--> default.rb – contains default attribute values
  |--> files (dir)
       |--> default (dir)
            |--> f5-icontrol-10.2.0.2.gem – F5 Ruby iControl Library
            |--> f5-node-initiator – script to add nodes to BIG-IP pool; source in Codeshare
  |--> recipes (dir)
       |--> default.rb – core logic of recipe
  |--> metadata.rb – information about author, recipe version, recipe license, etc.
  |--> README.rdoc – README document with description, requirements, usage, etc.
```

That's it. Our recipe contains 4 directories and 6 files. If we did our job in creating this cookbook, you shouldn't need to modify anything within it. We should be able to change the default attributes in our role or our node definition to enact any changes to the defaults.

Installing the Cookbook

1. Download the f5-node-initiator cookbook: f5-node-initiator.tgz
2. Untar it into your chef-repo/cookbooks/ directory

   ```
   tar -C ~/chef-repo/cookbooks/. -zxvf f5-node-initiator.tgz
   ```

3. Add the new cookbook to your Git repository

   ```
   git commit -a -m "Adding f5-node-initiator cookbook"
   ```

4. Install the cookbook on the Chef server

```
rake install
```

5. Ensure that the cookbook is installed and available on the Chef server

```
knife cookbook list
```

The “web_server” Role

Once we have our cookbook uploaded to our server, we need to assign it to our “web_server” role in order to get it to do anything. In this example, we are going to install and configure Apache, mod_php for Apache, and the f5-node-initiator. Here are the steps to create this role:

1. Create a file called “web_server.rb” in ~/chef-repo/roles/

```
vi ~/chef-repo/roles/web_server.rb
```

2. Add the following contents to the “web_server.rb” role file

```
name "web_server"
description "Common web server configuration"


run_list(
  "recipe[apache2]",
  "recipe[f5-node-initiator]"
)


default_attributes(
  "bigip" => {
    "address" =>   "10.0.0.245",
    "user" =>      "admin",
    "pass" =>      "admin",
    "pool_name" => "chef_test_http_pool"
  }
)
```

Note: Don't forget to create the targret HTTP pool on the LTM. If there isn't a pool to add the nodes to, the f5-node-initiator recipe will fail.

3. Add the role to your Git and commit it to the repository

```
git add web_server.rb
git commit -m "Adding web_server role"
```

4. Install the "web_server" role

```
rake install
```

Applying the “web_server” role to a node

The f5-node-initiator cookbook is now in place and the recipe has been added to our “web_server” role. We will now take the role and apply it to our new node, which we’ll call “web-001”. At the conclusion of this section, if everything goes as planned, we should have a web server running Apache and serving traffic as a pool member of our LTM. Let’s walk through the steps of adding the role to our node:

1. Add the “web_server” role to the node’s run_list

```
knife node run_list add web-001 "role[webserver]"
```

2. Manually kick-off convergence on the node

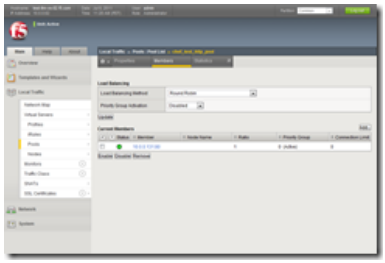
```
ssh root@web-001
root@web-001:~# chef-client
```

Note: Convergence happens by default automatically every 30 minutes, but it is best to test at least one node to ensure things are working as expected.

3. Watch the output to ensure that everything runs successfully

```
[Fri, 08 Jul 2011 11:17:21 -0700] INFO: Starting Chef Run (Version 0.9.16)
[Fri, 08 Jul 2011 11:17:24 -0700] INFO: Installing package[apache2] version 2.2.14-5ubuntu8.4
[Fri, 08 Jul 2011 11:17:29 -0700] INFO: Installing gem_package[f5-icontrol] version 10.2.0.2
[Fri, 08 Jul 2011 11:17:38 -0700] INFO: gem_package[f5-icontrol] sending run action to execute[f
[Fri, 08 Jul 2011 11:17:40 -0700] INFO: Ran execute[f5-node-initiator] successfully
[Fri, 08 Jul 2011 11:17:40 -0700] INFO: Chef Run complete in 18.90055 seconds
[Fri, 08 Jul 2011 11:17:40 -0700] INFO: cleaning the checksum cache
[Fri, 08 Jul 2011 11:17:40 -0700] INFO: Running report handlers
[Fri, 08 Jul 2011 11:17:40 -0700] INFO: Report handlers complete
```

4. Verify that the node was added to the “chef_test_http_pool” on our LTM



Conclusion

This example used a web server as the example, but the role and attributes could be easily modified to support any number of systems and protocols. If you can pass traffic through a BIG-IP and create a pool for it, then you should be able to use the f5-node-initiator cookbook to automate additions of those nodes to the LTM pool. Give it a shot with SMTP, SIP, etc. and let us know how it goes.

Chef and iControl are both incredibly powerful and versatile tools. When combined, they can perform a number of labor-intensive tasks almost effortlessly. The initial configuration of Chef may seem like a lot of work, but it will save you work in the long run. Starting with a good foundation can make large projects later on seem much more approachable. Trust us, it is worth it. Until next time, keep automating!

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc. Corporate Headquarters info@f5.com	F5 Networks Asia-Pacific apacinfo@f5.com	F5 Networks Ltd. Europe/Middle-East/Africa emeainfo@f5.com	F5 Networks Japan K.K. f5j-info@f5.com