**Bootstrap a Node using knife**

A node is any physical, virtual, or cloud machine that is configured to be maintained by a chef-client. In order to bootstrap a node, you will first need a working installation of the [*Chef software package*](https://docs.chef.io/packages.html). A bootstrap is a process that installs the chef-client on a target system so that it can run as a chef-client and communicate with a Chef server. There are two ways to do this:

* **Use the knife bootstrap subcommand to** [***bootstrap a node using the omnibus installer***](https://docs.chef.io/install_bootstrap.html)
* **Use an unattended install to bootstrap a node from itself, without using SSH or WinRM**

**knife bootstrap**[**¶**](https://docs.chef.io/install_bootstrap.html#knife-bootstrap)

The knife bootstrap command is a common way to install the chef-client on a node. The default for this approach assumes that a node can access the Chef website so that it may download the chef-client package from that location.

The omnibus installer will detect the version of the operating system, and then install the appropriate version of the chef-client using a single command to install the chef-client and all of its dependencies, including an embedded version of Ruby, RubyGems, OpenSSL, key-value stores, parsers, libraries, and command line utilities.

The omnibus installer puts everything into a unique directory (/opt/chef/) so that the chef-client will not interfere with other applications that may be running on the target machine. Once installed, the chef-client requires a few more configuration steps before it can perform its first chef-client run on a node.

**Run the bootstrap command**

The knife bootstrap subcommand is used to run a bootstrap operation that installs the chef-client on the target node. The following steps describe how to bootstrap a node using knife.

1. Identify the FQDN or IP address of the target node. The knife bootstrap command requires the FQDN or the IP address for the node in order to complete the bootstrap operation.
2. Once the workstation machine is configured, it can be used to install the chef-client on one (or more) nodes across the organization using a knife bootstrap operation. The knife bootstrap command is used to SSH into the target machine, and then do what is needed to allow the chef-client to run on the node. It will install the chef-client executable (if necessary), generate keys, and register the node with the Chef server. The bootstrap operation requires the IP address or FQDN of the target system, the SSH credentials (username, password or identity file) for an account that has root access to the node, and (if the operating system is not Ubuntu, which is the default distribution used by knife bootstrap) the operating system running on the target system.

In a command window, enter the following:

$ knife bootstrap 123.45.6.789 -x username -P password --sudo

where 123.45.6.789 is the IP address or the FQDN for the node. Use the --distro option to specify a non-default distribution. For more information about the options available to the knife bootstrap command for Ubuntu- and Linux-based platforms, see [*knife bootstrap*](https://docs.chef.io/knife_bootstrap.html). For Microsoft Windows, the knife windows plugin is required, see [*knife windows*](https://docs.chef.io/plugin_knife_windows.html).

And then while the bootstrap operation is running, the command window will show something like the following:

Bootstrapping Chef on 123.45.6.789

123.45.6.789 knife sudo password:

Enter your password:

123.45.6.789

123.45.6.789 [Fri, 07 Sep 2012 11:05:05 -0700] INFO: \*\*\* Chef 10.12.0 \*\*\*

123.45.6.789

123.45.6.789 [Fri, 07 Sep 2012 11:05:07 -0700] INFO: Client key /etc/chef/client.pem is not present - registering

123.45.6.789

123.45.6.789 [Fri, 07 Sep 2012 11:05:15 -0700] INFO: Setting the run\_list to [] from JSON

123.45.6.789

123.45.6.789 [Fri, 07 Sep 2012 11:05:15 -0700] INFO: Run List is []

123.45.6.789

123.45.6.789 [Fri, 07 Sep 2012 11:05:15 -0700] INFO: Run List expands to []

123.45.6.789

123.45.6.789 [Fri, 07 Sep 2012 11:05:15 -0700] INFO: Starting Chef Run for name\_of\_node

123.45.6.789

123.45.6.789 [Fri, 07 Sep 2012 11:05:15 -0700] INFO: Running start handlers

123.45.6.789

123.45.6.789 [Fri, 07 Sep 2012 11:05:15 -0700] INFO: Start handlers complete.

123.45.6.789

123.45.6.789 [Fri, 07 Sep 2012 11:05:17 -0700] INFO: Loading cookbooks []

123.45.6.789

123.45.6.789 [Fri, 07 Sep 2012 11:05:17 -0700] WARN: Node name\_of\_node has an empty run list.

123.45.6.789

123.45.6.789 [Fri, 07 Sep 2012 11:05:19 -0700] INFO: Chef Run complete in 3.986283452 seconds

123.45.6.789

123.45.6.789 [Fri, 07 Sep 2012 11:05:19 -0700] INFO: Running report handlers

123.45.6.789

123.45.6.789 [Fri, 07 Sep 2012 11:05:19 -0700] INFO: Report handlers complete

123.45.6.789

1. After the bootstrap operation has finished, verify that the node is recognized by the Chef server. To show only the node that was just bootstrapped, run the following command:
2. $ knife client show name\_of\_node

where name\_of\_node is the name of the node that was just bootstrapped. The Chef server will return something similar to:

admin: false

chef\_type: client

json\_class: Chef::ApiClient

name: name\_of\_node

public\_key:

and to show the full list of nodes (and workstations) that are registered with the Chef server, run the following command:

knife client list

The Chef server will return something similar to:

workstation

workstation

...

client

name\_of\_node

...

client

**Validatorless Bootstrap**[**¶**](https://docs.chef.io/install_bootstrap.html#validatorless-bootstrap)

The ORGANIZATION-validator.pem is typically added to the .chef directory on the workstation. When a node is bootstrapped from that workstation, the ORGANIZATION-validator.pem is used to authenticate the newly-created node to the Chef server during the initial chef-client run. Starting with Chef client 12.1, it is possible to bootstrap a node using the USER.pem file instead of the ORGANIZATION-validator.pem file. This is known as a “validatorless bootstrap”.

To create a node via the USER.pem file, simply delete the ORGANIZATION-validator.pem file on the workstation. For example:

$ rm -f /home/lamont/.chef/myorg-validator.pem

and then make the following changes in the knife.rb file:

* Remove the validation\_client\_name setting
* Edit the validation\_key setting to be something that isn’t a path to an existent ORGANIZATION-validator.pem file. For example: /nonexist.

As long as a USER.pem is also present on the workstation from which the validatorless bootstrap operation will be initiated, the bootstrap operation will run and will use the USER.pem file instead of the ORGANIZATION-validator.pem file.

When running a validatorless knife bootstrap operation, the output is similar to:

desktop% knife bootstrap 10.1.1.1 -N foo01.acme.org \

-E dev -r 'role[base]' -j '{ "foo": "bar" }' \

--ssh-user vagrant --sudo

Node foo01.acme.org exists, overwrite it? (Y/N)

Client foo01.acme.org exists, overwrite it? (Y/N)

Creating new client for foo01.acme.org

Creating new node for foo01.acme.org

Connecting to 10.1.1.1

10.1.1.1 Starting first Chef Client run...

[....etc...]

**knife bootstrap Options**[**¶**](https://docs.chef.io/install_bootstrap.html#knife-bootstrap-options)

Use the following options with a validatorless bootstrap to specify items that are stored in chef-vault:

--bootstrap-vault-file VAULT\_FILE

The path to a JSON file that contains a list of vaults and items to be updated.

--bootstrap-vault-item VAULT\_ITEM

A single vault and item to update as vault:item.

--bootstrap-vault-json VAULT\_JSON

A JSON string that contains a list of vaults and items to be updated.

For example:

--bootstrap-vault-json '{ "vault1": ["item1", "item2"], "vault2": "item2" }'

**Examples**[**¶**](https://docs.chef.io/install_bootstrap.html#examples)

The --bootstrap-vault-\* options add the client identify of the bootstrapping node to the permissions list of the specified vault item. This enables the newly-bootstrapped chef-client to be able to read items from the vault. Only a single client is authorized at a time for acces to the vault. (The -S search query option with the knife vault create subcommand does the same.)

**Recreate a data bag item**[**¶**](https://docs.chef.io/install_bootstrap.html#recreate-a-data-bag-item)

The following example shows how to recreate a data bag item:

$ knife vault delete sea power

Do you really want to delete sea/power? (Y/N) Y

Deleted chef\_vault\_item[sea/power]

$ echo "{\"some\":\"content for them\"}" > sea-power-content.json

$ cat sea-power-content.json

{"some":"content for them"}

$ knife vault create sea power -M client -A sean\_horn,angle -J sea-power-content.json

No clients, because the -S option was not specified while creating the vault.

At this time, only the users sean\_horn and angle are authorized to read and manage the vault.

$ knife vault show sea power --mode client -p all

admins:

sean\_horn

angle

clients:

id: power

search\_query:

some: content for them

It is definitely an encrypted databag, see?

$ knife data\_bag show sea power

WARNING: Encrypted data bag detected, but no secret provided for decoding. Displaying encrypted data.

id: power

some:

cipher: aes-256-cbc

encrypted\_data: c7Axnyg+1KDxBPOZdYN9QuIYx6dmSmK70unAQbn12Lygvsv2g9DPJJbueXVh

+yxL

iv: ONoVR7OjPZiAzaqOZ30bjg==

version: 1

**Use –bootstrap-vault-file**[**¶**](https://docs.chef.io/install_bootstrap.html#use-bootstrap-vault-file)

Use the sea:power recreation step above first, to follow the difference in the vault permissions.

echo "{\"sea\":\"power\"}" > sea-power-bootstrap-vault-file.json

$ knife bootstrap localhost -p 2200 -N ubuntu-12.04 -r 'role[group1]' --ssh-user vagrant --sudo --bootstrap-vault-file sea-power-bootstrap-vault-file.json

Node ubuntu-12.04 exists, overwrite it? (Y/N) Y

Client ubuntu-12.04 exists, overwrite it? (Y/N) Y

Creating new client for ubuntu-12.04

Creating new node for ubuntu-12.04

Connecting to localhost

localhost -----> Existing Chef installation detected

localhost Starting first Chef Client run...

localhost Starting Chef Client, version 12.2.1

localhost resolving cookbooks for run list: ["delay-test-reporting"]

localhost Synchronizing Cookbooks:

localhost - delay-test-reporting

localhost Compiling Cookbooks...

localhost Converging 1 resources

localhost Recipe: delay-test-reporting::default

localhost \* execute[sleep 30] action run

localhost - execute sleep 30

localhost

localhost Running handlers:

localhost Running handlers complete

localhost Chef Client finished, 1/1 resources updated in 34.307257232 seconds

The client ubuntu-12.04 was added to the chef-vault during the bootstrap.

$ knife vault show sea power --mode client -p all

admins:

sean\_horn

angle

clients: ubuntu-12.04

id: power

search\_query:

some: content for them

**Use –bootstrap-vault-item**[**¶**](https://docs.chef.io/install_bootstrap.html#use-bootstrap-vault-item)

Use the sea:power re-creation step above first, to follow the difference in the vault permissions.

$ knife bootstrap localhost -p 2200 -N ubuntu-12.04 -r 'role[group1]' --ssh-user vagrant --sudo --bootstrap-vault-item sea:power

Node ubuntu-12.04 exists, overwrite it? (Y/N) Y

Client ubuntu-12.04 exists, overwrite it? (Y/N) Y

Creating new client for ubuntu-12.04

Creating new node for ubuntu-12.04

Connecting to localhost

localhost -----> Existing Chef installation detected

localhost Starting first Chef Client run...

localhost Starting Chef Client, version 12.2.1

localhost resolving cookbooks for run list: ["delay-test-reporting"]

localhost Synchronizing Cookbooks:

localhost - delay-test-reporting

localhost Compiling Cookbooks...

localhost Converging 1 resources

localhost Recipe: delay-test-reporting::default

localhost \* execute[sleep 30] action run

localhost - execute sleep 30

localhost

localhost Running handlers:

localhost Running handlers complete

localhost Chef Client finished, 1/1 resources updated in 34.322229474

seconds

During the above run, the sea:power vault item was updated with the ubuntu-12.04 client during the validatorless bootstrap. Previously, it only had the two admins authorized to view the content

$ knife vault show sea power -p all

admins:

sean\_horn

angle

clients: ubuntu-12.04

id: power

search\_query: role:stuff

some: secret stuff for them

Then, let’s check the ubuntu-12.04 client. Install the chef-vault gem in the embedded chef-client:

$ sudo /opt/chef/embedded/bin/gem install chef-vault --no-ri --no-rdoc

Fetching: chef-vault-2.6.1.gem (100%)

Successfully installed chef-vault-2.6.1

1 gem installed

The client itself can decrypt and read the encrypted databag contents as well.

$ sudo /opt/chef/bin/knife vault show sea power -c /etc/chef/client.rb -M client -p all

admins:

sean\_horn

angle

clients: ubuntu-12.04

id: power

search\_query: role:group1

some: secret stuff for them

Success! The client is authorized to view the content of the sea:power databag item

**Use –bootstrap-vault-json[¶](https://docs.chef.io/install_bootstrap.html" \l "use-bootstrap-vault-json" \o "Permalink to this headline)**

Use the sea:power re-creation step above first, to follow the difference in the vault permissions.

$ knife bootstrap localhost -p 2200 -N ubuntu-12.04 -r 'role[group1]' --ssh-user vagrant --sudo --bootstrap-vault-json '{"sea": "power"}'

Node ubuntu-12.04 exists, overwrite it? (Y/N) Y

Client ubuntu-12.04 exists, overwrite it? (Y/N) Y

Creating new client for ubuntu-12.04

Creating new node for ubuntu-12.04

Connecting to localhost

localhost -----> Existing Chef installation detected

localhost Starting first Chef Client run...

localhost Starting Chef Client, version 12.2.1

localhost resolving cookbooks for run list: ["delay-test-reporting"]

localhost Synchronizing Cookbooks:

localhost - delay-test-reporting

localhost Compiling Cookbooks...

localhost Converging 1 resources

localhost Recipe: delay-test-reporting::default

localhost \* execute[sleep 30] action run

localhost - execute sleep 30

localhost

localhost Running handlers:

localhost Running handlers complete

localhost Chef Client finished, 1/1 resources updated in 33.732784033 seconds

$ knife vault show sea power -M client -p all

admins:

sean\_horn

angle

clients: ubuntu-12.04

id: power

search\_query:

some: content for them

**Unattended Installs**[**¶**](https://docs.chef.io/install_bootstrap.html#unattended-installs)

The chef-client can be installed using an unattended bootstrap. This allows the chef-client to be installed from itself, without using SSH. For example, machines are often created using environments like AWS Auto Scaling, AWS CloudFormation, Rackspace Auto Scale, and PXE. In this scenario, using tooling for attended, single-machine installs like knife bootstrap or knife CLOUD\_PLUGIN create is not practical because the machines are created automatically and someone cannot always be on-hand to initiate the bootstrap process.

When the chef-client is installed using an unattended bootstrap, remember that the chef-client:

* Must be able to authenticate to the Chef server
* Must be able to configure a run-list
* May require custom attributes, depending on the cookbooks that are being used
* Must be able to access the chef-validator.pem so that it may create a new identity on the Chef server
* Must have a unique node name; the chef-client will use the FQDN for the host system by default

When the chef-client is installed using an unattended bootstrap, it is typically built into an image that starts the chef-client on boot. The type of image used depends on the platform on which the unattended bootstrap will take place.

Use settings in the [*client.rb file*](https://docs.chef.io/config_rb_client.html)—chef\_server\_url, http\_proxy, and so on—to ensure that configuration details are built into the unattended bootstrap process.

**Setting the initial run-list**

A node’s initial run-list is specified using a JSON file on the host system. When running the chef-client as an executable, use the -j option to tell the chef-client which JSON file to use. For example:

$ chef-client -j /etc/chef/file.json --environment \_default

where file.json is similar to:

{

"resolver": {

"nameservers": [ "10.0.0.1" ],

"search":"int.example.com"

},

"run\_list": [ "recipe[resolver]" ]

}

and where \_default is the name of the environment that is assigned to the node.