There are encrypted data bags in json files with some values I need to change. I need to run something like.

$ knife data bag from file show --secret-file path/to/secret DATABAGNAME --config path/to/knife.rb

**Create key and databag item:**

$ openssl rand -base64 512 | tr -d '\r\n' > /tmp/encrypted\_data\_bag\_secret

$ knife data bag create mydatabag secretstuff --secret-file /tmp/encrypted\_data\_bag\_secret -z

**Enter this:**

{

"id": "secretstuff",

"firstsecret": "must remain secret",

"secondsecret": "also very secret"

}

**The json file is indeed encrypted:**

# cat data\_bags/mydatabag/secretstuff.json

{

"id": "secretstuff",

"firstsecret": {

"encrypted\_data": "VafoT8Jc0lp7o4erCxz0WBrJYXjK6j+sJ+WGKJftX4BVF391rA1zWyHpToF0\nqvhn\n",

"iv": "MhG09xFcwFAqX/IA3BusMg==\n",

"version": 1,

"cipher": "aes-256-cbc"

},

"secondsecret": {

"encrypted\_data": "Epj+2DuMOsf5MbDCOHEep7S12F6Z0kZ5yMuPv4a3Cr8dcQWCk/pd58OPGQgI\nUJ2J\n",

"iv": "66AcYpoF4xw/rnYfPegPLw==\n",

"version": 1,

"cipher": "aes-256-cbc"

}

}

**Show decrypted content with knife:**

# knife data bag show mydatabag secretstuff -z --secret-file /tmp/encrypted\_data\_bag\_secret

Encrypted data bag detected, decrypting with provided secret.

firstsecret: must remain secret

id: secretstuff

secondsecret: also very secret

Create a random encryption key:  
openssl rand -base64 512 | tr -d '\r\n' > secret\_key

Use this **to encrypt a data bag item named “passwords” located in a data bag named “production”:**  
knife data bag create −−editor /usr/bin/vi −−secret-file ./secret\_key production passwords

This will open an text editor, **example JSON data** would be:  
{  
  "id": "passwords",  
  "mysql": "yourmysqlpassword",  
  "ssh": "yoursshpassword"  
}  
Save and exit.

Show the encrypted contents of your databag:  
knife data bag show production passwords

Show the decrypted contents of your databag:  
knife data bag show −−secret-file=./secret\_key production passwords

For your chef clients to be able to decrypt the databag when needed, just copy over the secret key (replace client-node with your IP/node name):  
scp ./secret\_key client-node:/etc/chef/encrypted\_data\_bag\_secret

Given you have a recipe mysql, that will be run on your client, your .rb file (for example **recipes/default.rb**) could contain the following:  
passwords = Chef::EncryptedDataBagItem.load("production", "passwords")  
mysql = passwords[“mysql”]  
Chef::Log.info("The mysql password is: '#{mysql}'")  
This will log the password in cleartext. You can use the variable***#{mysql}***which contains the cleartext password in whatever action in your recipe.

**About Data Bags**[**¶**](https://docs.chef.io/data_bags.html#about-data-bags)

[[edit on GitHub]](https://github.com/chef/chef-web-docs/blob/master/chef_master/source/data_bags.rst)

A data bag is a global variable that is stored as JSON data and is accessible from a Chef server. A data bag is indexed for searching and can be loaded by a recipe or accessed during a search.

**Create a Data Bag**[**¶**](https://docs.chef.io/data_bags.html#create-a-data-bag)

A data bag can be created in two ways: using knife or manually. In general, using knife to create data bags is recommended, but as long as the data bag folders and data bag item JSON files are created correctly, either method is safe and effective.

**Using Knife**[**¶**](https://docs.chef.io/data_bags.html#using-knife)

knife can be used to create data bags and data bag items when the knife data bag subcommand is run with the create argument. For example:

$ knife data bag create DATA\_BAG\_NAME (DATA\_BAG\_ITEM)

knife can be used to update data bag items using the from file argument:

$ knife data bag from file BAG\_NAME ITEM\_NAME.json

As long as a file is in the correct directory structure, knife will be able to find the data bag and data bag item with only the name of the data bag and data bag item. For example:

$ knife data bag from file BAG\_NAME ITEM\_NAME.json

will load the following file:

data\_bags/BAG\_NAME/ITEM\_NAME.json

Continuing the example above, if you are in the “admins” directory and make changes to the file charlie.json, then to upload that change to the Chef server use the following command:

$ knife data bag from file admins charlie.json

In some cases, such as when knife is not being run from the root directory for the chef-repo, the full path to the data bag item may be required. For example:

$ knife data bag from file BAG\_NAME /path/to/file/ITEM\_NAME.json

**Manually**[**¶**](https://docs.chef.io/data_bags.html#manually)

One or more data bags and data bag items can be created manually under the data\_bags directory in the chef-repo. Any method can be used to create the data bag folders and data bag item JSON files. For example:

$ mkdir data\_bags/admins

would create a data bag folder named “admins”. The equivalent command for using knife is:

$ knife data bag create admins

A data bag item can be created manually in the same way as the data bag, but by also specifying the file name for the data bag item (this example is using vi, a visual editor for UNIX):

$ vi data\_bags/admins/charlie.json

would create a data bag item named “charlie.json” under the “admins” sub-directory in the data\_bags directory of the chef-repo. The equivalent command for using knife is:

$ knife data bag create admins charlie

**Store Data in a Data Bag**[**¶**](https://docs.chef.io/data_bags.html#store-data-in-a-data-bag)

When the chef-repo is cloned from GitHub, the following occurs:

* A directory named data\_bags is created.
* For each data bag, a sub-directory is created that has the same name as the data bag.
* For each data bag item, a JSON file is created and placed in the appropriate sub-directory.

The data\_bags directory can be placed under version source control.

When deploying from a private repository using a data bag, use the deploy\_key option to ensure the private key is present:

{

'id': 'my\_app',

... (truncated) ...

'deploy\_key': 'ssh\_private\_key'

}

where ssh\_private\_key is the same SSH private key as used with a private git repository and the new lines converted to \n.

**Directory Structure**[**¶**](https://docs.chef.io/data_bags.html#directory-structure)

All data bags are stored in the data\_bags directory of the chef-repo. This directory structure is understood by knife so that the full path does not need to be entered when working with data bags from the command line. An example of the data\_bags directory structure:

- data\_bags

- admins

- charlie.json

- bob.json

- tom.json

- db\_users

- charlie.json

- bob.json

- sarah.json

- db\_config

- small.json

- medium.json

- large.json

where admins, db\_users, and db\_config are the names of individual data bags and all of the files that end with .json are the individual data bag items

**knife data bag**[**¶**](https://docs.chef.io/knife_data_bag.html#knife-data-bag)

[[edit on GitHub]](https://github.com/chef/chef-web-docs/blob/master/chef_master/source/knife_data_bag.rst)

A data bag is a global variable that is stored as JSON data and is accessible from a Chef server. A data bag is indexed for searching and can be loaded by a recipe or accessed during a search.

A data bag item may be encrypted using [shared secret encryption](https://en.wikipedia.org/wiki/Symmetric-key_algorithm). This allows each data bag item to store confidential information (such as a database password) or to be managed in a source control system (without plain-text data appearing in revision history). Each data bag item may be encrypted individually; if a data bag contains multiple encrypted data bag items, these data bag items are not required to share the same encryption keys.

**Note**

Because the contents of encrypted data bag items are not visible to the chef-server, search queries against data bags with encrypted items will not return any results.

The knife data bag subcommand is used to manage arbitrary stores of globally available JSON data.

**Note**

Review the list of [common options](https://docs.chef.io/knife_common_options.html) available to this (and all) knife subcommands and plugins.

**create[¶](https://docs.chef.io/knife_data_bag.html" \l "create" \o "Permalink to this headline)**

Use the create argument to add a data bag to the Chef server.

**Syntax**[**¶**](https://docs.chef.io/knife_data_bag.html#syntax)

This argument has the following syntax:

$ knife data bag create DATA\_BAG\_NAME [DATA\_BAG\_ITEM] (options)

**Options**[**¶**](https://docs.chef.io/knife_data_bag.html#options)

This argument has the following options:

DATA\_BAG\_ITEM

The name of a specific item within a data bag.

--secret SECRET

The encryption key that is used for values contained within a data bag item. If secret is not specified, the chef-client looks for a secret at the path specified by the encrypted\_data\_bag\_secret setting in the client.rb file.

--secret-file FILE

The path to the file that contains the encryption key.

**Note**

For encrypted data bag items, use either --secret or --secret-file, not both.

**Note**

See [knife.rb](https://docs.chef.io/config_rb_knife_optional_settings.html) for more information about how to add certain knife options as settings in the knife.rb file.

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

The following examples show how to use this knife subcommand:

**Create a data bag**

To create a data bag named “admins”, enter:

$ knife data bag create admins

to return:

Created data\_bag[admins]

**delete[¶](https://docs.chef.io/knife_data_bag.html" \l "delete" \o "Permalink to this headline)**

Use the delete argument to delete a data bag or a data bag item from a Chef server.

**Syntax**[**¶**](https://docs.chef.io/knife_data_bag.html#id1)

This argument has the following syntax:

$ knife data bag delete DATA\_BAG\_NAME [DATA\_BAG\_ITEM] (options)

**Options**[**¶**](https://docs.chef.io/knife_data_bag.html#id2)

This argument has the following options:

DATA\_BAG\_ITEM

The name of a specific item within a data bag.

**Examples**[**¶**](https://docs.chef.io/knife_data_bag.html#id3)

The following examples show how to use this knife subcommand:

**Delete a data bag**

$ knife data bag delete data\_bag\_name

**Delete a data bag item**

To delete an item named “charlie”, enter:

$ knife data bag delete admins charlie

Type Y to confirm a deletion.

**edit[¶](https://docs.chef.io/knife_data_bag.html" \l "edit" \o "Permalink to this headline)**

Use the edit argument to edit the data contained in a data bag. If encryption is being used, the data bag will be decrypted, the data will be made available in the $EDITOR, and then encrypted again before saving it to the Chef server.

**Syntax**[**¶**](https://docs.chef.io/knife_data_bag.html#id4)

This argument has the following syntax:

$ knife data bag edit DATA\_BAG\_NAME [DATA\_BAG\_ITEM] (options)

**Options**[**¶**](https://docs.chef.io/knife_data_bag.html#id5)

This argument has the following options:

DATA\_BAG\_ITEM

The name of a specific item within a data bag.

--secret SECRET

The encryption key that is used for values contained within a data bag item. If secret is not specified, the chef-client looks for a secret at the path specified by the encrypted\_data\_bag\_secret setting in the client.rb file.

--secret-file FILE

The path to the file that contains the encryption key.

**Note**

For encrypted data bag items, use either --secret or --secret-file, not both.

**Note**

See [knife.rb](https://docs.chef.io/config_rb_knife_optional_settings.html) for more information about how to add certain knife options as settings in the knife.rb file.

**Examples**[**¶**](https://docs.chef.io/knife_data_bag.html#id6)

The following examples show how to use this knife subcommand:

**Edit a data bag**

To edit the contents of a data bag, enter:

$ knife data bag edit dogs tibetanspaniel

where dogs is the name of the data bag and tibetanspaniel is the name of the data bag item. This will return something similar to the following in the knife editor:

{

"name":"data\_bag\_item\_dogs\_tibetanspaniel",

"json\_class":"Chef::DataBagItem",

"chef\_type":"data\_bag\_item",

"data\_bag":"dogs",

"raw\_data":

{

"description":"small dog that likes to sit in windows",

"id":"tibetanspaniel"

}

}

Make the necessary changes to the key-value pairs under raw\_data and save them.

**Edit a data bag item**

To edit an item named “charlie” that is contained in a data bag named “admins”, enter:

$ knife data bag edit admins charlie

to open the $EDITOR. Once opened, you can update the data before saving it to the Chef server. For example, by changing:

{

"id": "charlie"

}

to:

{

"id": "charlie",

"uid": 1005,

"gid": "ops",

"shell": "/bin/zsh",

"comment": "Crazy Charlie"

}

**from file**[**¶**](https://docs.chef.io/knife_data_bag.html#from-file)

Use the from file argument to:

Add a data bag item to a data bag

Update the contents of an existing data bag item

The data bag itself must already exist on the Chef server and must be specified as part of the command. The contents of the data bag item are specified using a JSON file. This JSON file may be located at a relative or absolute path; its location must be specified as part of the command. The JSON file that defines the contents of the data bag item must at least contain the name of the data bag item—"id": "name".

**Warning**

A chef-client must be version 11.6 (or higher) when using the knife data bag from file argument with the Enterprise Chef or Open Source Chef version 11 servers.

**Syntax**[**¶**](https://docs.chef.io/knife_data_bag.html#id7)

This argument has the following syntax:

$ knife data bag from file DATA\_BAG\_NAME\_or\_PATH JSON\_FILE

**Options**[**¶**](https://docs.chef.io/knife_data_bag.html#id8)

This argument has the following options:

-a, --all

Upload all data bags found at the specified path.

--secret SECRET

The encryption key that is used for values contained within a data bag item. If secret is not specified, the chef-client looks for a secret at the path specified by the encrypted\_data\_bag\_secret setting in the client.rb file.

--secret-file FILE

The path to the file that contains the encryption key.

**Note**

For encrypted data bag items, use either --secret or --secret-file, not both.

**Note**

See [knife.rb](https://docs.chef.io/config_rb_knife_optional_settings.html) for more information about how to add certain knife options as settings in the knife.rb file.

**Examples**[**¶**](https://docs.chef.io/knife_data_bag.html#id9)

The following examples show how to use this knife subcommand:

**Create a data bag from a file**

To create a data bag on the Chef server from a file:

$ knife data bag from file "path to JSON file"

**Create an encrypted data bag from a file**

To create a data bag named “devops\_data” that contains encrypted data, enter:

$ knife data bag from file devops\_data --secret-file "path to decryption file"

**Create an encrypted data bag for use with chef-client local mode**

To generate an encrypted data bag item in a JSON file for use when the chef-client is run in local mode (via the --local-mode option), enter:

$ knife data bag from file my\_data\_bag /path/to/data\_bag\_item.json -z --secret-file /path/to/encrypted\_data\_bag\_secret

this will create an encrypted JSON file in:

data\_bags/my\_data\_bag/data\_bag\_item.json

**list[¶](https://docs.chef.io/knife_data_bag.html" \l "list" \o "Permalink to this headline)**

Use the list argument to view a list of data bags that are currently available on the Chef server.

**Syntax**[**¶**](https://docs.chef.io/knife_data_bag.html#id10)

This argument has the following syntax:

$ knife data bag list

**Options**[**¶**](https://docs.chef.io/knife_data_bag.html#id11)

This argument has the following options:

-w, --with-uri

Show the corresponding URIs.

**Examples**[**¶**](https://docs.chef.io/knife_data_bag.html#id12)

The following examples show how to use this knife subcommand:

**View a list of data bags**

$ knife data bag list

**show[¶](https://docs.chef.io/knife_data_bag.html" \l "show" \o "Permalink to this headline)**

Use the show argument to view the contents of a data bag.

**Syntax**[**¶**](https://docs.chef.io/knife_data_bag.html#id13)

This argument has the following syntax:

$ knife data bag show DATA\_BAG\_NAME (options)

**Options**[**¶**](https://docs.chef.io/knife_data_bag.html#id14)

This argument has the following options:

DATA\_BAG\_ITEM

The name of a specific item within a data bag.

--secret SECRET

The encryption key that is used for values contained within a data bag item. If secret is not specified, the chef-client looks for a secret at the path specified by the encrypted\_data\_bag\_secret setting in the client.rb file.

--secret-file FILE

The path to the file that contains the encryption key.

**Note**

For encrypted data bag items, use either --secret or --secret-file, not both.

**Note**

See [knife.rb](https://docs.chef.io/config_rb_knife_optional_settings.html) for more information about how to add certain knife options as settings in the knife.rb file.

**Examples**[**¶**](https://docs.chef.io/knife_data_bag.html#id15)

The following examples show how to use this knife subcommand:

**Show a data bag**

$ knife data bag show admins

to return something like:

charlie

**Show a data bag item**

To show the contents of a specific item within data bag, enter:

$ knife data bag show admins charlie

to return:

comment: Crazy Charlie

gid: ops

id: charlie

shell: /bin/zsh

uid: 1005

**Show a data bag, encrypted**

To show the contents of a data bag named passwords with an item that contains encrypted data named mysql,enter:

$ knife data bag show passwords mysql

to return:

{

"id": "mysql",

"pass": "trywgFA6R70NO28PNhMpGhEvKBZuxouemnbnAUQsUyo=\n",

"user": "e/p+8WJYVHY9fHcEgAAReg==\n"

}

**Show a data bag, decrypted**

To show the decrypted contents of the same data bag, enter:

$ knife data bag show --secret-file /path/to/decryption/file passwords mysql

to return:

{

"id": "mysql",

"pass": "thesecret123",

"user": "fred"

}

**Show a data bag as JSON**

To view information in JSON format, use the -F common option as part of the command like this:

$ knife data bag show admins -F json

Other formats available include text, yaml, and pp.