chef-server-ctl (executable)¶

[edit on GitHub]

The Chef server includes a command-line utility named chef-server-ctl. This command-line tool is used to start and stop individual services, reconfigure the Chef server, run chef-pedant, and then tail Chef server log files.

Backup / Restore¶

Use the following commands to manage backups of Chef server data, and then to restore those backups.

backup¶

The backup subcommand is used to back up all Chef server data. This subcommand:

Requires rsync to be installed on the Chef server prior to running the command

Requires a chef-server-ctl reconfigure prior to running the command

Should not be run in a Chef server configuration with an external PostgreSQL database; use knife ec backup instead

Puts the initial backup in the /var/opt/chef-backup directory as a tar.gz file; move this backup to a new location for safe keeping

Options

This subcommand has the following options:

-y, --yes

Use to specify if the Chef server can go offline during tar.gz-based backups.

Syntax

This subcommand has the following syntax:

$ chef-server-ctl backup

restore¶

The restore subcommand is used to restore Chef server data from a backup that was created by the backup subcommand. This subcommand may also be used to add Chef server data to a newly-installed server. This subcommand:

Requires rsync to be installed on the Chef server prior to running the command

Requires a chef-server-ctl reconfigure prior to running the command

Should not be run in a Chef server configuration with an external PostgreSQL database; use knife ec backup instead

Note

The restore command does not support transferring backups across different versions of Chef server. Backups taken with the backup command must restore to the same version of Chef server that was in use when they were created.

Options

This subcommand has the following options:

-c, --cleanse

Use to remove all existing data on the Chef server; it will be replaced by the data in the backup archive.

-d DIRECTORY, --staging-dir DIRECTORY

Use to specify that the path to an empty directory to be used during the restore process. This directory must have enough disk space to expand all data in the backup archive.

Syntax

This subcommand has the following syntax:

$ chef-server-ctl restore PATH\_TO\_BACKUP (options)

Examples

$ chef-server-ctl restore /path/to/tar/archive.tar.gz

backup-recover¶

The backup-recover subcommand is used to force the Chef server to attempt to become the backup server. This is the opposite of the master-recover subcommand.

Warning

If this command is run on both back-end servers, it will put the back-end cluster into a state where no server holds the DRBD resource.

This subcommand has the following syntax:

$ chef-server-ctl backup-recover

cleanse¶

The cleanse subcommand is used to re-set the Chef server to the state it was in prior to the first time the reconfigure subcommand is run. This command will destroy all data, configuration files, and logs. The software that was put on-disk by the package installation will remain; re-run chef-server-ctl reconfigure to recreate the default data and configuration files.

Options

This subcommand has the following options:

--with-external

Use to specify that Chef server data on an external PostgreSQL database should be removed.

Syntax

This subcommand has the following syntax:

$ chef-server-ctl cleanse

gather-logs¶

The gather-logs subcommand is used to gather the Chef server log files into a tarball that contains all of the important log files and system information.

This subcommand has the following syntax:

$ chef-server-ctl gather-logs

ha-status¶

The ha-status subcommand is used to check the status for services running in a high availability topology. This command will verify the following:

The Keepalived daemon is enabled in the config

The DRBD process is enabled in the config

The underlying block device or logical volume for DRBD has been created and configured

The DRBD device exists

The current state of the server is master or backup; any migration processes have completed

The failover virtual IP address is correctly attached to only the master node

The DRBD state is correct based on the state of the server being master or backup

The DRBD mount point is correctly mounted to only the master node

The DRBD replication IP addresses are pingable

The runit status of the services are correct (up or down) based on the master or backup state of the server

This subcommand has the following syntax:

$ chef-server-ctl ha-status

If this command runs successfully, it will return the following:

$ [OK] all checks passed.

Otherwise it will print out a list of errors, similar to the following:

...

[OK] nginx is running correctly, and I am master.

[ERROR] redis\_lb is not running.

[OK] opscode-erchef is running correctly, and I am master.

...

[ERROR] ERRORS WERE DETECTED.

For example:

[OK] keepalived HA services enabled

[OK] DRBD disk replication enabled

[OK] DRBD partition /dev/opscode/drbd found

[OK] DRBD device /dev/drbd0 found

[OK] cluster status = master

[OK] found VIP IP address and I am master

[OK] found VRRP communications interface eth1

[OK] my DRBD status is Connected/Primary/UpToDate and I am master

[OK] my DRBD partition is mounted and I am master

[OK] DRBD primary IP address pings

[OK] DRBD secondary IP address pings

...

[OK] all checks passed.

help¶

The help subcommand is used to print a list of all available chef-server-ctl commands.

This subcommand has the following syntax:

$ chef-server-ctl help

install¶

The install subcommand is used to install premium features of the Chef server: Chef management console, Chef Analytics, chef-client run reporting, high availability configurations, Chef push jobs, and Chef server replication.

Warning

The chef-server-ctl install command no longer works in the 12.5 (and earlier) versions of the Chef server due to a change in how packages are downloaded from Chef.

Syntax

This subcommand has the following syntax:

$ chef-server-ctl install name\_of\_premium\_feature (options)

where name\_of\_premium\_feature represents the command line value associated with the premium feature.

Options

This subcommand has the following options:

--path PATH

Use to specify the location of a package. This option is not required when packages are downloaded from https://packages.chef.io/.

Use Downloads¶

The install subcommand downloads packages from https://packages.chef.io/ by default. For systems that are not behind a firewall (and have connectivity to https://packages.chef.io/), these packages can be installed as described below.

Feature

Command

Chef Manage

Use Chef management console to manage data bags, attributes, run-lists, roles, environments, and cookbooks from a web user interface.

On the Chef server, run:

$ chef-server-ctl install chef-manage

then:

$ chef-server-ctl reconfigure

and then:

$ chef-manage-ctl reconfigure

Note

Starting with the Chef management console 2.3.0, the Chef MLSA must be accepted when reconfiguring the product. If the Chef MLSA has not already been accepted, the reconfigure process will prompt for a yes to accept it. Or run chef-manage-ctl reconfigure --accept-license to automatically accept the license.

Chef Push Jobs

Use Chef push jobs to run jobs—an action or a command to be executed—against nodes independently of a chef-client run.

On the Chef server, run:

$ chef-server-ctl install opscode-push-jobs-server

then:

$ chef-server-ctl reconfigure

and then:

$ opscode-push-jobs-server-ctl reconfigure

Reporting

Use Reporting to keep track of what happens during every chef-client runs across all of the infrastructure being managed by Chef. Run Reporting with Chef management console to view reports from a web user interface.

On the Chef server, run:

$ chef-server-ctl install opscode-reporting

then:

$ chef-server-ctl reconfigure

and then:

$ opscode-reporting-ctl reconfigure

Use Local Packages¶

The install subcommand downloads packages from https://packages.chef.io/ by default. For systems that are behind a firewall (and may not have connectivity to packages.chef.io), these packages can be downloaded from https://downloads.chef.io/chef-manage/, and then installed manually. First download the package that is appropriate for the platform, save it to a local path, and then run the install command using the --path option to specify the directory in which the package is located:

$ chef-server-ctl install PACKAGE\_NAME --path /path/to/package/directory

For example:

$ chef-server-ctl install chef-manage --path /root/packages

The chef-server-ctl command will install the first chef-manage package found in the /root/packages directory.

Key Rotation¶

Use the following commands to manage public and private key rotation for users and clients.

add-client-key¶

Use the add-client-key subcommand to add a client key.

Syntax

This subcommand has the following syntax:

$ chef-server-ctl add-client-key ORG\_NAME CLIENT\_NAME [--public-key-path PATH] [--expiration-date DATE] [--key-name NAME]

Warning

All options for this subcommand must follow all arguments.

Options

This subcommand has the following options:

CLIENT\_NAME

The name of the client that you wish to add a key for.

-e DATE --expiration-date DATE

An ISO 8601 formatted string: YYYY-MM-DDTHH:MM:SSZ. For example: 2013-12-24T21:00:00Z. If not passed, expiration will default to infinity.

-k NAME --key-name NAME

String defining the name of your new key for this client. If not passed, it will default to the fingerprint of the public key.

ORG\_NAME

The short name for the organization to which the client belongs.

-p PATH --public-key-path PATH

The location to a file containing valid PKCS#1 public key to be added. If not passed, then the server will generate a new one for you and return the private key to STDOUT.

add-user-key¶

Use the add-user-key subcommand to add a user key.

Syntax

This subcommand has the following syntax:

$ chef-server-ctl add-user-key USER\_NAME [--public-key-path PATH] [--expiration-date DATE] [--key-name NAME]

Warning

All options for this subcommand must follow all arguments.

Options

This subcommand has the following options:

-e DATE --expiration-date DATE

An ISO 8601 formatted string: YYYY-MM-DDTHH:MM:SSZ. For example: 2013-12-24T21:00:00Z. If not passed, expiration will default to infinity.

-k NAME --key-name NAME

String defining the name of your new key for this user. If not passed, it will default to the fingerprint of the public key.

-p PATH --public-key-path PATH

The location to a file containing valid PKCS#1 public key to be added. If not passed, then the server will generate a new one for you and return the private key to STDOUT.

USER\_NAME

The user name for the user for which a key is added.

delete-client-key¶

Use the delete-client-key subcommand to delete a client key.

Syntax

This subcommand has the following syntax:

$ chef-server-ctl delete-client-key ORG\_NAME CLIENT\_NAME KEY\_NAME

Options

This subcommand has the following arguments:

ORG\_NAME

The short name for the organization to which the client belongs.

CLIENT\_NAME

The name of the client.

KEY\_NAME

The unique name to be assigned to the key you wish to delete.

delete-user-key¶

Use the delete-user-key subcommand to delete a user key.

Syntax

This subcommand has the following syntax:

$ chef-server-ctl delete-user-key USER\_NAME KEY\_NAME

Warning

The parameters for this subcommand must be in the order specified above.

Options

This subcommand has the following arguments:

USER\_NAME

The user name.

KEY\_NAME

The unique name to be assigned to the key you wish to delete.

list-client-keys¶

Use the list-client-keys subcommand to list client keys.

Syntax

This subcommand has the following syntax:

$ chef-server-ctl list-client-keys ORG\_NAME CLIENT\_NAME [--verbose]

Warning

All options for this subcommand must follow all arguments.

Options

This subcommand has the following options:

CLIENT\_NAME

The name of the client.

ORG\_NAME

The short name for the organization to which the client belongs.

--verbose

Use to show the full public key strings in command output.

list-user-keys¶

Use the list-user-keys subcommand to list client keys.

Syntax

This subcommand has the following syntax:

$ chef-server-ctl list-user-keys USER\_NAME [--verbose]

Warning

All options for this subcommand must follow all arguments.

Options

This subcommand has the following options:

USER\_NAME

The user name you wish to list keys for.

--verbose

Use to show the full public key strings in command output.

Example

To view a list of user keys (including public key output):

$ chef-server-ctl list-user-keys applejack --verbose

Returns:

2 total key(s) found for user applejack

key\_name: test-key

expires\_at: Infinity

public\_key:

-----BEGIN PUBLIC KEY-----

MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA4q9Dh+bwJSjhU/VI4Y8s

9WsbIPfpmBpoZoZVPL7V6JDfIaPUkdcSdZpynhRLhQwv9ScTFh65JwxC7wNhVspB

4bKZeW6vugNGwCyBIemMfxMlpKZQDOc5dnBiRMMOgXSIimeiFtL+NmMXnGBBHDaE

b+XXI8oCZRx5MTnzEs90mkaCRSIUlWxOUFzZvnv4jBrhWsd/yBM/h7YmVfmwVAjL

VST0QG4MnbCjNtbzToMj55NAGwSdKHCzvvpWYkd62ZOquY9f2UZKxYCX0bFPNVQM

EvBQGdNG39XYSEeF4LneYQKPHEZDdqe7TZdVE8ooU/syxlZgADtvkqEoc4zp1Im3

2wIDAQAB

-----END PUBLIC KEY-----

key\_name: default

expires\_at: Infinity

public\_key:

-----BEGIN PUBLIC KEY-----

MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA4q9Dh+bwJSjhU/VI4Y8s

9WsbIPfpmBpoZoZVPL7V6JDfIaPUkdcSdZpynhRLhQwv9ScTFh65JwxC7wNhVspB

4bKZeW6vugNGwCyBIemMfxMlpKZQDOc5dnBiRMMOgXSIimeiFtL+NmMXnGBBHDaE

b+XXI8oCZRx5MTnzEs90mkaCRSIUlWxOUFzZvnv4jBrhWsd/yBM/h7YmVfmwVAjL

VST0QG4MnbCjNtbzToMj55NAGwSdKHCzvvpWYkd62ZOquY9f2UZKxYCX0bFPNVQM

EvBQGdNG39XYSEeF4LneYQKPHEZDdqe7TZdVE8ooU/syxlZgADtvkqEoc4zp1Im3

2wIDAQAB

-----END PUBLIC KEY-----

Secrets Management¶

Use the following commands to manage and rotate shared secrets and service credentials. The secrets file used for storing these is located at /etc/opscode/private-chef-secrets.json on your Chef server. It should be owned and readable only by root.

set-secret¶

The set-secret subcommand allows storing shared secrets and service credentials. Only secrets known to Chef server can be stored.

New in Chef server 12.14

Syntax

This subcommand has the following syntax:

$ chef-server-ctl set-secret GROUP NAME

There are various ways to pass the secret to this command:

as a third argument:

$ chef-server-ctl set-secret ldap bind\_password secretpassword

via an environment variable:

$ export LDAP.BIND\_PASSWORD="secretpassword"

$ chef-server-ctl set-secret ldap bind\_password

via an interactive prompt:

$ chef-server-ctl set-secret ldap bind\_password

Enter ldap bind\_password: (no terminal output)

Re-enter ldap bind\_password: (no terminal output)

remove-secret¶

The remove-secret subcommand allows removing a stored shared secret and service credential.

New in Chef server 12.14

Syntax

This subcommand has the following syntax:

$ chef-server-ctl remove-secret GROUP NAME

Example

$ chef-server-ctl remove-secret ldap bind\_password

show-secret¶

The show-secret subcommand allows viewing a stored shared secret and service credential.

New in Chef server 12.14

Syntax

This subcommand has the following syntax:

$ chef-server-ctl show-secret GROUP NAME

set-db-superuser-password¶

The set-db-superuser-password subcommand allows storing the database superuser password.

New in Chef server 12.14

Syntax

This subcommand has the following syntax:

$ chef-server-ctl set-db-superuser-password

Similar to set-secret, the superuser password can also be provided via the environment variable DB\_PASSWORD.

set-actions-password¶

The set-actions-password subcommand allows storing the RabbitMQ Actions password.

New in Chef server 12.14

Syntax

This subcommand has the following syntax:

$ chef-server-ctl set-actions-password

Similar to set-secret, the action password can also be provided via the environment variable ACTIONS\_PASSWORD.

oc-id-show-app¶

The oc-id-show-app subcommand allows for retrieving the client ID and client secret for applications known to oc-id. Note that with insecure\_addon\_compat \_disabled\_, this data will no longer be written to /etc/opscode/oc-id-applications/APP.json.

New in Chef server 12.14

Syntax

This subcommand has the following syntax:

$ chef-server-ctl oc-id-show-app APP

Example

$ chef-server-ctl oc-id-show-app supermarket

{

"name": "supermarket",

"uid": "0bad0f2eb04e935718e081fb71asdfec3681c81acb9968a8e1e32451d08b",

"secret": "17cf1141cc971a10ce307611beda7ffadstr4f1bc98d9f9ca76b9b127879",

"redirect\_uri": "https://supermarket.mycompany.com/auth/chef\_oauth2/callback"

}

require-credential-rotation¶

The require-credential-rotation subcommand takes the Chef server offline and requires a complete service credential rotation before the Chef server(s) in your cluster can restart again. Run rotate-shared-secrets to create a new shared secret, salt, and generate the new service credentials. Then copy the secrets file to each Chef server and run sudo chef-server-ctl reconfigure on each server to complete the rotation process.

Note

Credential rotation does not rotate the pivotal, user, or client keys, or remove any Chef server policy or cookbooks that have been uploaded.

New in Chef server 12.7

Syntax

This subcommand has the following syntax:

$ chef-server-ctl require-credential-rotation (options)

Options

This subcommand has the following options:

-y, --yes

Bypass a prompt in the terminal and agree that you want to disable the Chef server, and require credential rotation.

rotate-all-credentials¶

The rotate-all-credentials subcommand generates new credential values for all service credentials by incrementing the credential version number and creating a new hash value. You can choose whether to copy the updated secrets file to each node in the cluster and reconfiguring or by running this subcommand on all the nodes.

New in Chef server 12.7

Syntax

This subcommand has the following syntax:

$ chef-server-ctl rotate-all-credentials

rotate-credentials¶

The rotate-credentials subcommand generates new credential values for all credentials for a given service by incrementing the value and creating a new hash value. You can choose whether to copy the updated secrets file to each node in the cluster and reconfiguring or by running this subcommand for that specific service on all the nodes.

New in Chef server 12.7

Syntax

This subcommand has the following syntax:

$ chef-server-ctl rotate-credentials SERVICE\_NAME

rotate-shared-secrets¶

The rotate-shared-secrets subcommand creates a new shared secret and salt, in addition to generating new service credentials. It also resets the credential\_version number for the services to 0. After you have run this subcommand, a new shared secret has been created, so you must copy the secrets file to each Chef server and run sudo chef-server-ctl reconfigure on them to complete the rotation process.

New in Chef server 12.7

Syntax

This subcommand has the following syntax:

$ chef-server-ctl rotate-shared-secrets

show-service-credentials¶

The show-service-credentials subcommand shows all of the service credentials for services running on the local Chef server.

New in Chef server 12.7

Syntax

This subcommand has the following syntax:

$ chef-server-ctl show-service-credentials

master-recover¶

The master-recover subcommand is used to force the Chef server to attempt to become the master server. This command is typically run in tandem with the backup-recover subcommand on the back-end peer, unless the back-end peer is no longer available.

This subcommand has the following syntax:

$ chef-server-ctl master-recover

Organization Management¶

Use the org-create, org-delete, org-list, org-show, org-user-add and org-user-remove commands to manage organizations.

org-create¶

The org-create subcommand is used to create an organization. (The validation key for the organization is returned to STDOUT when creating an organization with this command.)

Syntax

This subcommand has the following syntax:

$ chef-server-ctl org-create ORG\_NAME "ORG\_FULL\_NAME" (options)

where:

The name must begin with a lower-case letter or digit, may only contain lower-case letters, digits, hyphens, and underscores, and must be between 1 and 255 characters. For example: chef.

The full name must begin with a non-white space character and must be between 1 and 1023 characters. For example: "Chef Software, Inc.".

Options

This subcommand has the following options:

-a USER\_NAME, --association\_user USER\_NAME

Associate a user with an organization and add them to the admins and billing\_admins security groups.

-f FILE\_NAME, --filename FILE\_NAME

Write the ORGANIZATION-validator.pem to FILE\_NAME instead of printing it to STDOUT.

Examples

$ chef-server-ctl org-create prod Production

$ chef-server-ctl org-create staging Staging -a chef-admin

$ chef-server-ctl org-create dev Development -f /tmp/id-dev.key

$ chef-server-ctl org-create dev Development --association\_user grantmc

org-delete¶

The org-delete subcommand is used to delete an organization.

Syntax

This subcommand has the following syntax:

$ chef-server-ctl org-delete ORG\_NAME

Examples

$ chef-server-ctl org-delete infra-testing-20140909

$ chef-server-ctl org-delete pedant-testing-org

org-list¶

The org-list subcommand is used to list all of the organizations currently present on the Chef server.

Syntax

This subcommand has the following syntax:

$ chef-server-ctl org-list (options)

Options

This subcommand has the following options:

-a, --all-orgs

Show all organizations.

-w, --with-uri

Show the corresponding URIs.

org-show¶

The org-show subcommand is used to show the details for an organization.

Syntax

This subcommand has the following syntax:

$ chef-server-ctl org-show ORG\_NAME

org-user-add¶

The org-user-add subcommand is used to add a user to an organization.

Syntax

This subcommand has the following syntax:

$ chef-server-ctl org-user-add ORG\_NAME USER\_NAME (options)

Options

This subcommand has the following options:

--admin

Add the user to the admins group.

Examples

$ chef-server-ctl org-user-add prod john\_smith

$ chef-server-ctl org-user-add preprod testmaster

$ chef-server-ctl org-user-add dev grantmc --admin

org-user-remove¶

The org-user-remove subcommand is used to remove a user from an organization.

Warning

A user who belongs to the admins group must be removed from the group before they may be removed from an organization. To remove a user from the admins group, run the following:

$ EDITOR=vi knife edit /groups/admins.json

make the required changes, and then save the file.

Syntax

This subcommand has the following syntax:

$ chef-server-ctl org-user-remove ORG\_NAME USER\_NAME (options)

Options

This subcommand has the following options:

--force

Force the removal of a user from the organization’s admins and billing-admins groups.

Examples

$ chef-server-ctl org-user-remove prod john\_smith

$ chef-server-ctl org-user-remove prod testmaster

$ chef-server-ctl org-user-remove grantmc --force

password¶

The password subcommand is used to change a user’s password. When Active Directory or LDAP is enabled, this command enables (or disables) the system recovery password for that user. For example:

This subcommand has the following syntax:

$ chef-server-ctl password USERNAME

This subcommand has the following options:

--disable

Use this option to disable a user’s system recovery password.

Examples

For example, to change a user’s password, enter:

$ chef-server-ctl password adamjacobs

and then enter the password and confirm it:

Enter the new password: \*\*\*\*\*\*

Enter the new password again: \*\*\*\*\*\*

to return:

Password for adamjacobs successfully set.

To disable a system recovery password:

$ chef-server-ctl password adamjacobs --disable

to return:

Password for adamjacobs successfully disabled for System Recovery.

psql¶

The psql subcommand is used to log into the PostgreSQL database associated with the named service. This subcommand:

Uses psql (the interactive terminal for PostgreSQL)

Has read-only access by default

Is the recommended way to interact with any PostgreSQL database that is part of the Chef server

Automatically handles authentication

Syntax

This subcommand has the following syntax:

$ chef-server-ctl psql SERVICE\_NAME (options)

Options

This subcommand has the following options:

--write

Use to enable write access to the PostgreSQL database.

reconfigure¶

The reconfigure subcommand is used when changes are made to the chef-server.rb file to reconfigure the server. When changes are made to the chef-server.rb file, they will not be applied to the Chef server configuration until after this command is run. This subcommand will also restart any services for which the service\_name['enabled'] setting is set to true.

This subcommand has the following syntax:

$ chef-server-ctl reconfigure

reindex¶

The reindex subcommand is used to reload Chef server data from PostgreSQL to Apache Solr.

This subcommand has the following syntax:

$ chef-server-ctl reindex

Options

This subcommand has the following options:

-a, --all-orgs

Use to reindex all organizations on the Chef server. This option will override any organization specified as part of the command, i.e. chef-server-ctl reindex ORG\_NAME -a will reindex all organizations and not just the specified organization.

-d, --disable-api

Use to disable the Chef server API to prevent writes during reindexing.

-t, --with-timing

Use to print timing information for the reindex processes.

-w, --wait

Use to wait for the reindexing queue to clear before exiting.

Server Admins¶

The server-admins group is a global group that grants its members permission to create, read, update, and delete user accounts, with the exception of superuser accounts. The server-admins group is useful for users who are responsible for day-to-day administration of the Chef server, especially user management via the knife user subcommand. Before members can be added to the server-admins group, they must already have a user account on the Chef server.

Scenario¶

The following user accounts exist on the Chef server: pivotal (a superuser account), alice, bob, carol, and dan. Run the following command to view a list of users on the Chef server:

$ chef-server-ctl user-list

and it returns the same list of users:

pivotal

alice

bob

carol

dan

Alice is a member of the IT team whose responsibilities include day-to-day administration of the Chef server, in particular managing the user accounts on the Chef server that are used by the rest of the organization. From a workstation, Alice runs the following command:

$ knife user list -c ~/.chef/alice.rb

and it returns the following error:

ERROR: You authenticated successfully to <chef\_server\_url> as alice

but you are not authorized for this action

Response: Missing read permission

Alice is not a superuser and does not have permissions on other users because user accounts are global to organizations in the Chef server. Let’s add Alice to the server-admins group:

$ chef-server-ctl grant-server-admin-permissions alice

and it returns the following response:

User alice was added to server-admins.

Alice can now create, read, update, and delete user accounts on the Chef server, even for organizations to which Alice is not a member. From a workstation, Alice re-runs the following command:

$ knife user list -c ~/.chef/alice.rb

which now returns:

pivotal

alice

bob

carol

dan

Alice is now a server administrator and can use the following knife subcommands to manage users on the Chef server:

knife user-create

knife user-delete

knife user-edit

knife user-list

knife user-show

For example, Alice runs the following command:

$ knife user edit carol -c ~/.chef/alice.rb

and the $EDITOR opens in which Alice makes changes, and then saves them.

Superuser Accounts¶

Superuser accounts may not be managed by users who belong to the server-admins group. For example, Alice attempts to delete the pivotal superuser account:

$ knife user delete pivotal -c ~/.chef/alice.rb

and the following error is returned:

ERROR: You authenticated successfully to <chef\_server\_url> as user1

but you are not authorized for this action

Response: Missing read permission

Alice’s action is unauthorized even with membership in the server-admins group.

Manage server-admins Group¶

Membership of the server-admins group is managed with a set of chef-server-ctl subcommands:

chef-server-ctl grant-server-admin-permissions

chef-server-ctl list-server-admins

chef-server-ctl remove-server-admin-permissions

Add Members¶

The grant-server-admin-permissions subcommand is used to add a user to the server-admins group. Run the command once per user added.

This subcommand has the following syntax:

$ chef-server-ctl grant-server-admin-permissions USER\_NAME

where USER\_NAME is the user to add to the list of server administrators.

For example:

$ chef-server-ctl grant-server-admin-permissions bob

returns:

User bob was added to server-admins. This user can now list,

read, and create users (even for orgs they are not members of)

for this Chef Server.

Remove Members¶

The remove-server-admin-permissions subcommand is used to remove a user from the server-admins group. Run the command once per user removed.

This subcommand has the following syntax:

$ chef-server-ctl remove-server-admin-permissions USER\_NAME

where USER\_NAME is the user to remove from the list of server administrators.

For example:

$ chef-server-ctl remove-server-admin-permissions bob

returns:

User bob was removed from server-admins. This user can no longer

list, read, and create users for this Chef Server except for where

they have default permissions (such as within an org).

List Membership¶

The list-server-admins subcommand is used to return a list of users who are members of the server-admins group.

This subcommand has the following syntax:

$ chef-server-ctl list-server-admins

and will return a list of users similar to:

pivotal

alice

bob

carol

dan

show-config¶

The show-config subcommand is used to view the configuration that will be generated by the reconfigure subcommand. This command is most useful in the early stages of a deployment to ensure that everything is built properly prior to installation.

This subcommand has the following syntax:

$ chef-server-ctl show-config

uninstall¶

The uninstall subcommand is used to remove the Chef server application, but without removing any of the data. This subcommand will shut down all services (including the runit process supervisor).

This subcommand has the following syntax:

$ chef-server-ctl uninstall

Note

To revert the uninstall subcommand, run the reconfigure subcommand (because the start subcommand is disabled by the uninstall command).

upgrade¶

The upgrade subcommand is used to upgrade the Chef server.

Syntax

This subcommand has the following syntax:

$ chef-server-ctl upgrade (options)

Options

Note

Options for the upgrade subcommand may only be used when upgrading from Open Source Chef 11 to Chef server 12.

This subcommand has the following options:

-d DIRECTORY, --chef11-data-dir DIRECTORY

The directory in which Open Source Chef 11 data is located. Default value: a temporary directory.

-e DIRECTORY, --chef12-data-dir DIRECTORY

The directory in which Chef server 12 data is located. Default value: a temporary directory.

-f FULL\_NAME, --full-org-name FULL\_NAME

The full name of the Chef server organization. The full name must begin with a non-white space character and must be between 1 and 1023 characters. For example: Chef Software, Inc.. If this option is not specified, the upgrade command will prompt for it.

-h, --help

Use to show help for the chef-server-ctl upgrade subcommand.

-k KEY\_PATH, --key KEY\_PATH

The Open Source Chef 11 admin.pem key for the API client. This is the key used to download Open Source Chef 11 data. Default value: /etc/chef-server/admin.pem.

-o ORG\_NAME, --org-name ORG\_NAME

The name of the Chef server organization. The name must begin with a lower-case letter or digit, may only contain lower-case letters, digits, hyphens, and underscores, and must be between 1 and 255 characters. For example: chef. If this option is not specified, the upgrade command will prompt for it.

-s URL, --chef11-server-url URL

The URL for the Open Source Chef or Enterprise Chef server, version 11. Default value: https://localhost.

-t NUMBER, --upload-threads NUMBER

The number of threads to use when migrating cookbooks. Default value: 10.

-u USER, --user

Create a client as an admin client. This is required for any user to access Open Source Chef as an administrator.

-x URL, --chef12-server-url URL

The URL for the Chef server, version 12. Default value: https://localhost.

-y, --yes

Use to skip confirmation prompts during the upgrade process.

User Management¶

Use the user-create, user-delete, user-edit, user-list and user-show subcommands to manage users.

user-create¶

The user-create subcommand is used to create a user. (The validation key for the organization may be returned to STDOUT when creating a user with this command.)

Syntax

This subcommand has the following syntax:

$ chef-server-ctl user-create USER\_NAME FIRST\_NAME [MIDDLE\_NAME] LAST\_NAME EMAIL 'PASSWORD' (options)

Options

This subcommand has the following options:

-f FILE\_NAME, --filename FILE\_NAME

Write the USER.pem to a file instead of STDOUT.

Examples

$ chef-server-ctl user-create john\_smith John Smith john\_smith@example.com p@s5w0rD!

$ chef-server-ctl user-create jane\_doe Jane Doe jane\_doe@example.com p@s5w0rD! -f /tmp/jane\_doe.key

$ chef-server-ctl user-create waldendude Henry David Thoreau waldendude@example.com excursions

user-delete¶

The user-delete subcommand is used to delete a user.

Syntax

This subcommand has the following syntax:

$ chef-server-ctl user-delete USER\_NAME

Examples

$ chef-server-ctl user-delete john\_smith

$ chef-server-ctl user-delete jane\_doe

Options

This subcommand has the following options:

-R, --remove-from-admin-groups

Removes a user who is in one or more ‘admin’ groups unless that user is the only member of the ‘admin’ group(s).

New in Chef server 12.9.

user-edit¶

The user-edit subcommand is used to edit the details for a user. The data will be made available in the $EDITOR for editing.

Syntax

This subcommand has the following syntax:

$ chef-server-ctl user-edit USER\_NAME

Examples

$ chef-server-ctl user-edit john\_smith

$ chef-server-ctl user-edit jane\_doe

user-list¶

The user-list subcommand is used to view a list of users.

Syntax

This subcommand has the following syntax:

$ chef-server-ctl user-list (options)

Options

This subcommand has the following options:

-w, --with-uri

Show the corresponding URIs.

user-show¶

The user-show subcommand is used to show the details for a user.

Syntax

This subcommand has the following syntax:

$ chef-server-ctl user-show USER\_NAME (options)

Options

This subcommand has the following options:

-l, --with-orgs

Show all organizations.

Service Subcommands¶

This command has a built in process supervisor that ensures all of the required services are in the appropriate state at any given time. The supervisor starts two processes per service and provides the following subcommands for managing services: hup, int, kill, once, restart, service-list, start, status, stop, tail, and term.

Warning

The following commands are disabled when an external PostgreSQL database is configured for the Chef server: hup, int, kill, once, restart, start, stop, tail, and term.

hup¶

The hup subcommand is used to send a SIGHUP to all services. This command can also be run for an individual service by specifying the name of the service in the command.

This subcommand has the following syntax:

$ chef-server-ctl hup SERVICE\_NAME

where SERVICE\_NAME represents the name of any service that is listed after running the service-list subcommand.

int¶

The int subcommand is used to send a SIGINT to all services. This command can also be run for an individual service by specifying the name of the service in the command.

This subcommand has the following syntax:

$ chef-server-ctl int SERVICE\_NAME

where SERVICE\_NAME represents the name of any service that is listed after running the service-list subcommand.

kill¶

The kill subcommand is used to send a SIGKILL to all services. This command can also be run for an individual service by specifying the name of the service in the command.

This subcommand has the following syntax:

$ chef-server-ctl kill SERVICE\_NAME

where SERVICE\_NAME represents the name of any service that is listed after running the service-list subcommand.

once¶

The supervisor for the Chef server is configured to restart any service that fails, unless that service has been asked to change its state. The once subcommand is used to tell the supervisor to not attempt to restart any service that fails.

This command is useful when troubleshooting configuration errors that prevent a service from starting. Run the once subcommand followed by the status subcommand to look for services in a down state and/or to identify which services are in trouble. This command can also be run for an individual service by specifying the name of the service in the command.

This subcommand has the following syntax:

$ chef-server-ctl once SERVICE\_NAME

where SERVICE\_NAME represents the name of any service that is listed after running the service-list subcommand.

restart¶

The restart subcommand is used to restart all services enabled on the Chef server or to restart an individual service by specifying the name of that service in the command.

Warning

When running the Chef server in a high availability configuration, restarting all services may trigger failover.

This subcommand has the following syntax:

$ chef-server-ctl restart SERVICE\_NAME

where SERVICE\_NAME represents the name of any service that is listed after running the service-list subcommand. When a service is successfully restarted the output should be similar to:

$ ok: run: service\_name: (pid 12345) 1s

service-list¶

The service-list subcommand is used to display a list of all available services. A service that is enabled is labeled with an asterisk (\*).

This subcommand has the following syntax:

$ chef-server-ctl service-list

start¶

The start subcommand is used to start all services that are enabled in the Chef server. This command can also be run for an individual service by specifying the name of the service in the command.

This subcommand has the following syntax:

$ chef-server-ctl start SERVICE\_NAME

where SERVICE\_NAME represents the name of any service that is listed after running the service-list subcommand. When a service is successfully started the output should be similar to:

$ ok: run: service\_name: (pid 12345) 1s

The supervisor for the Chef server is configured to wait seven seconds for a service to respond to a command from the supervisor. If you see output that references a timeout, it means that a signal has been sent to the process, but that the process has yet to actually comply. In general, processes that have timed out are not a big concern, unless they are failing to respond to the signals at all. If a process is not responding, use a command like the kill subcommand to stop the process, investigate the cause (if required), and then use the start subcommand to re-enable it.

status¶

The status subcommand is used to show the status of all services available to the Chef server. The results will vary based on the configuration of a given server. This subcommand has the following syntax:

$ chef-server-ctl status

and will return the status for all services. Status can be returned for individual services by specifying the name of the service as part of the command:

$ chef-server-ctl status SERVICE\_NAME

where SERVICE\_NAME represents the name of any service that is listed after running the service-list subcommand.

When service status is requested, the output should be similar to:

$ run: service\_name: (pid 12345) 12345s; run: log: (pid 1234) 67890s

where

run: is the state of the service (run: or down:)

service\_name: is the name of the service for which status is returned

(pid 12345) is the process identifier

12345s is the uptime of the service, in seconds

For example:

$ down: opscode-erchef: (pid 35546) 10s

By default, runit will restart services automatically when the services fail. Therefore, runit may report the status of a service as run: even when there is an issue with that service. When investigating why a particular service is not running as it should be, look for the services with the shortest uptimes. For example, the list below indicates that the opscode-erchef should be investigated further:

run: oc-id

run: opscode-chef: (pid 4327) 13671s; run: log: (pid 4326) 13671s

run: opscode-erchef: (pid 5383) 5s; run: log: (pid 4382) 13669s

run: opscode-expander: (pid 4078) 13694s; run: log: (pid 4077) 13694s

run: opscode-expander-reindexer: (pid 4130) 13692s; run: log: (pid 4114) 13692s

High Availability¶

On back-end servers in a high availability topology, Keepalived is used by the clustering service to determine whether a service should be running. If the status subcommand is run against any of these nodes, a few things change:

On the back-end node that is currently the backup server, it is normal to see only one running process: Keepalived

On the back-end node that is currently the master server, it is normal to see all services running. It is also normal to see some services in a down state if the server, on reboot, did not attempt to start the services because Keepalived determines which services are restarted based on the state of the cluster

A sample status line for a service that is running on the master server in a high availability topology:

run: opscode-solr4: (pid 25341) 239s, normally down; run: log: (pid 5700) 145308s

Log Files¶

A typical status line for a service that is running any of the Chef server front-end services is similar to the following:

run: name\_of\_service: (pid 1486) 7819s; run: log: (pid 1485) 7819s

where:

run describes the state in which the supervisor attempts to keep processes. This state is either run or down. If a service is in a down state, it should be stopped

name\_of\_service is the service name, for example: opscode-solr4

(pid 1486) 7819s; is the process identifier followed by the amount of time (in seconds) the service has been running

run: log: (pid 1485) 7819s is the log process. It is typical for a log process to have a longer run time than a service; this is because the supervisor does not need to restart the log process in order to connect the supervised process

If the service is down, the status line will appear similar to the following:

down: opscode-solr4: 3s, normally up; run: log: (pid 1485) 8526s

where

down indicates that the service is in a down state

3s, normally up; indicates that the service is normally in a run state and that the supervisor would attempt to restart this service after a reboot

stop¶

The stop subcommand is used to stop all services enabled on the Chef server. This command can also be run for an individual service by specifying the name of the service in the command.

This subcommand has the following syntax:

$ chef-server-ctl stop SERVICE\_NAME

where SERVICE\_NAME represents the name of any service that is listed after running the service-list subcommand. When a service is successfully stopped the output should be similar to:

$ ok: diwb: service\_name: 0s, normally up

For example:

$ chef-server-ctl stop

will return something similar to:

ok: down: nginx: 393s, normally up

ok: down: opscode-chef: 391s, normally up

ok: down: opscode-erchef: 391s, normally up

ok: down: opscode-expander: 390s, normally up

ok: down: opscode-expander-reindexer: 389s, normally up

ok: down: opscode-solr4: 389s, normally up

ok: down: postgresql: 388s, normally up

ok: down: rabbitmq: 388s, normally up

ok: down: redis\_lb: 387s, normally up

tail¶

The tail subcommand is used to follow all of the Chef server logs for all services. This command can also be run for an individual service by specifying the name of the service in the command.

This subcommand has the following syntax:

$ chef-server-ctl tail SERVICE\_NAME

where SERVICE\_NAME represents the name of any service that is listed after running the service-list subcommand.

term¶

The term subcommand is used to send a SIGTERM to all services. This command can also be run for an individual service by specifying the name of the service in the command.

This subcommand has the following syntax:

$ chef-server-ctl term SERVICE\_NAME

where SERVICE\_NAME represents the name of any service that is listed after running the service-list subcommand.

* © Copyright: This work is licensed under a Creative Commons Attribution 3.0 Unported License. This page is about: current version of Chef. Provide feedback on Chef documentation.