**Chef Server Migration**

This document outlines the steps involved to migrate the data from an existing Chef Server into a new Chef Server:

**Tools to be used:**

[**knife-ec-backup**](https://github.com/chef/knife-ec-backup#knife-ec-backup)

Knife-ec-backup is a Ruby gem which creates a full object-based export of your Chef Server (e.g., organizations, users, cookbooks, nodes, clients & keys, roles, acl’ s, etc.) and can restore them to another Chef Server.

Unlike file-based backups ([chef-server-ctl backup](https://docs.chef.io/server_backup_restore.html#chef-server-ctl)) which are optimized for speed, knife-ec-backup was designed to maximize compatibility and enable admins to clean up errors and bad data after it has been exported. The downside is that knife-ec-backup can be quite a bit slower than other backup strategies.

[**knife-tidy**](https://github.com/chef-customers/knife-tidy#knife-tidy)

Knife-tidy is an essential sidekick tool for Chef Server migrations.

Knife-tidy has several modes of operation:

* **knife tidy backup clean** can validate and eliminate errors within a knife-ec-backup data set. Over time Chef Server has become better about validating data, but doesn’t know how to clean up existing objects that were stored within it. This function fixes those objects for you so that they will cleanly import into the latest Chef server.
* **knife tidy server report** will check your server for unused nodes and cookbook versions, which are normally the largest objects in your Chef Server data. It identifies unused cookbook versions by evaluating the run\_lists of all the nodes and environment constraints, therefore providing a high degree of safety in its recommendations.
* **knife tidy server clean** is like server report but will take action for you, removing the unused nodes and cookbooks.

**Installing Prerequisites**

1. Install the required development packages for your platform:
   * RHEL/Centos/Oracle 6/7  
     $ sudo yum -y install gcc postgresql-devel
   * RHEL/Centos/Oracle 6 w/ Enterprise Chef 11  
     $ sudo yum -y install https://download.postgresql.org/pub/repos/yum/9.5/redhat/rhel-6-x86\_64/pgdg-redhat95-9.5-2.noarch.rpm  
     $ sudo yum -y install gcc postgresql-devel postgresql95-devel
2. Install a separate ruby environment from the embedded ruby packaged with the Chef Server. For simplicity, we are going to use the ChefDK.  
   $ curl -L https://chef.io/chef/install.sh | sudo bash -s -- -P chefdk
3. Install the latest knife-ec-backup and knife-tidy gems into your ruby environment (we are using the ruby embedded in the ChefDK we installed) on your existing Chef server:  
   $ sudo /opt/chefdk/embedded/bin/gem install knife-ec-backup -- --with-pg-config=/opt/opscode/embedded/postgresql/<PG\_VERSION>/bin/pg\_config  
   $ sudo /opt/chefdk/embedded/bin/gem install knife-tidy

### Migration Process

Your migration will happen in two phases:  the Initial Transfer phase, and the Synchronization phase (which consists of many small “catch up” syncs). Those familiar with the unix rsync tool will find this process to be identical in concept.

### Initial Transfer

The Initial Transfer phase can be pretty slow during both the backup and restore phases. It’s strongly recommended that you use a shell session manager like tmux orscreen to maintain your session in the event your computer is disconnected. Taking this one step further,you might configure those tools to capture all of the session history, or use a tool like script or tee to do that for you.

**Initial backup and restore process:**

* 1. Ensure you have installed the required prerequisites.
  2. Export all of your Chef Server data:
     + $ /opt/chefdk/embedded/bin/knife ec backup my\_backup\_destination --with-user-sql --with-key-sql --concurrency 20 -c /etc/opscode/pivotal.rb
     + Note: The default concurrency is 10. Pay special attention to your erchef logs and back off the concurrency number if you notice 502 or 412 errors sent to clients as.you don’t want to overload the Chef Server and effect existing traffic.
  3. Run knife-tidy on the export to resolve all compatibility issues
     + $ /opt/chefdk/embedded/bin/knife tidy backup clean --backup-path my\_backup\_destination
  4. Import the data on to the new cluster
     + Perform step 1 to install the latest knife-ec-backup and knife-tidy gems
     + $ /opt/chefdk/embedded/bin/knife ec restore my\_backup\_destination --with-user-sql --with-key-sql -c /etc/opscode/pivotal.rb

### Synchronization Phase

This phase will use the exact same steps as the Initial Transfer phase, optionally you can add the --purge flag to knife-ec-backup (**but not on restore\*)** to delete objects in the backup folder that have been deleted on the source Chef Server.

The Synchronization phase is much shorter than a full transfer, because only the changed objects need to be transferred. Schedule periodic syncs using cron during this phase while you plan your final cut-over. The time it takes to complete one full synchronization cycle will determine the length of the maintenance window (Chef server downtime) needed for the cut-over.

**\*Using –purge while restoring can have unintended effects because of the way Chef Server de-duplicates cookbook files between versions of a cookbook.**

**Plan and execute the cut-over**

The cut-over phase is essentially a Synchronization phase but with two additional steps added:

1. **Before the synchronization, stop all traffic going to your existing Chef server:** This is most simply accomplished with a load balancer or firewall rule that can be quickly enabled and disabled.
2. **After synchronization, change the DNS entry for your Chef Server to point at the new server or cluster:** As long as SSL certificates match, Chef clients will have no problems communicating with the new server. If the SSL certificates are different and issued by an internal PKI, you can pre-seed the certs from the cert chain in /etc/chef/trusted\_certs to all the clients so that they will be trusted.