Configuration Management mit Chef

Chefkoch

Welcher Administrator träumt nicht davon, seinen kompletten Rechnerpark mit nur einem Mausklick aufsetzen und konfigurieren zu können. Einen großen Schritt in diese Richtung geht die freie Software Chef – vorausgesetzt der Koch betreibt ausschließlich Linux-Computer und beherrscht die Programmiersprache Ruby. Tim Schürmann

Im Kern besteht Chef aus einem Server, der maßgeschneiderte Konfigurationsanleitungen für Software speichert. Angeschlossene Clients rufen diese so genannten Rezepte (Recipes) ab und richten nach den darin notierten Vorschriften vollautomatisch ihre Systeme ein. Dazu passen sie nicht nur Konfigurationsdateien an, sondern werfen bei Bedarf auch den Paketmanager an. Wenn sich Rezepte später ändern oder neue hinzukommen, bringen sich alle betroffenen Clients selbstständig auf den neuesten Stand. Der Administrator muss im Idealfall nur noch die Rezepte auf dem Server pflegen.

Großmarkthalle

Vor das Vergnügen haben die Entwickler des Chefs jedoch ziemlich viel Arbeit gesetzt. So bestehen die Rezepte aus einem oder mehreren normalen Ruby-Skript(en). Wer nicht auf vorgefertigte und recht allgemein gehaltene Rezepte aus dem Internet festgenagelt bleiben will, muss also diese Skriptsprache möglichst gut beherrschen. Bis die Eigenkreationen geschrieben und gut durchgetestet sind, vergeht also einige Zeit

Eine weitere Hürde ist die Installation, die sich relativ schwierig gestaltet, weil insbesondere der Chef-Server von mehreren anderen Komponenten abhängt, die ihrerseits ebenfalls weitere Softwarepakete erfordern. Der Chef-Server selbst ist in Ruby geschrieben, nutzt aber den in Erlang programmierten Rabbit-MQ-Server sowie eine auf Java beruhende Volltextsuchmaschine und parkt seine Daten in einer Couch-DB-Datenbank.

Außerdem spielt noch das eingesetzte Betriebssystem eine Rolle. Chef bevorzugt zwar Linux als Unterbau, läuft laut Wiki [1] aber auch auf anderen Unix-stämmigen Betriebssystemen wie beispielsweise Mac OS X, Open Solaris oder Free- und Open BSD. Am schnellsten gelangt man derzeit mit den Linux-Distributionen Debian 5, Ubuntu ab Version 8.10 oder Cent OS 5.x zum Ziel. Auf allen anderen Systemen gerät insbesondere die Einrichtung des Servers zu einem kleinen Abenteuer.

Die folgenden Ausführungen beziehen sich daher überwiegend auf Debian und Ubuntu. Wer zum ersten Mal mit dem Chef kocht, sollte seine Küche zudem in eine virtuelle Maschine verlagern. So kann im Zweifelsfall nichts überkochen und den Serverraum verkleben.

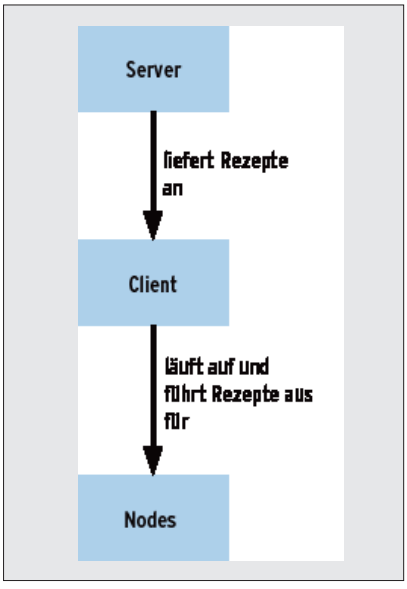


Abbildung 1: Die Zusammenarbeit von Server, Client und Nodes im Überblick.

Eine komplette Chef-Installation besteht aus den zu konfigurierenden Systemen, die Chef allgemein als Knoten bezeichnet (Nodes). Hinzu kommt der Server, der die Rezepte verwaltet und vorhält. Die Arbeit erledigen die so genannten Chef-Clients, die sich die Rezepte via REST-Schnittstelle beim Server abholen und ausführen. Jeder Client läuft auf einem Knoten, kann aber die Rezepte auf mehrere von ihnen anwenden. Das Zusammenspiel veranschaulicht Abbildung 1.

Grundzutat: Ruby 1.8

Der Einfachheit halber setzen die folgenden Beispiele neben einem Chef-Server einen einzigen Client auf. Letzterer konfiguriert ausschließlich den Computer, auf dem er läuft. Auf beiden beteiligten Systemen muss als erstes Ruby in Version 1.8.5 oder 1.8.7 (mit SSL-Bindings) her, unter Ruby 1.9 läuft der Chef derzeit noch nicht. Zusätzlich übersetzen einige später noch nachgeholte Rubygems verschiedene Erweiterungen und Bibliotheken, was die Anwesenheit von »make«, »gcc«, »g++« sowie der Ruby-Entwicklungspakete erfordert. Hinzu kommt noch das Hilfsprogramm »wget« für diverse Downloads. Unter Debian und Ubuntu Linux beginnt den Reigen der folgende Befehl:

```
sudo apt-get install ruby ruby1.8-dev ➥
libopenssl-ruby1.8 rdoc ri irb build-➥
essential wget ssl-cert
```

Unter Open Suse heißend die Pakete »ruby«, »ruby-devel«, »wget«, »openssl-certs«, »make«, »gcc« und »g++«. Die Zertifikate aus »ssl-cert« kommen später noch zum Einsatz.

Laut Anleitung [1] hätte der Chef gerne Rubygems in Version 1.3.6 oder neuer, aber nicht die 1.3.7. Die enthält einen Fehler, der die gleich folgende Installation mittendrin abwürgt. Da die meisten Distributionen eine ältere Rubygems-Version an Bord haben, greift der Koch am besten direkt zum Sourcecode-Archiv:

```
cd /tmp
wget http://rubyforge.org/frs/download.➥
php/69365/rubygems-1.3.6.tgz
tar zxf rubygems-1.3.6.tgz
cd rubygems-1.3.6
sudo ruby setup.rb
```

Wenn der letzte Befehl den Gems-Befehl als »/usr/bin/gem1.8« installiert hat (wie es unter Debian und Ubuntu der Fall ist), sorgt ein symbolischer Link für eine kleine Verbesserung:

```
sudo ln -sfv /usr/bin/gem1.8 /usr/bin/gem
```

Nun lässt sich mit dem folgenden Gems-Aufruf das Chef-Paket installieren:

```
sudo gem install chef
```

In Zukunft sollte bei jedem Gem-Update dem Json-Gem besondere Beachtung gelten. Die mittlerweile via Rubygems erhältliche Version 1.4.3 führt zu einem Fehler in Chef. Installiert ein »gem update« auf der Festplatte das betroffene Json-Paket, kehren die folgenden zwei Befehle wieder zur alten Fassung zurück:

```
sudo gem uninstall -aIx json
sudo gem install -v1.4.2 json
```

Die bis hierhin gezeigten Schritte haben jedoch erst die Basis für den Chef-Betrieb geschaffen. Die Installation insbesondere des Servers steht noch an.

Mit Chef lassen sich komplette Programme einspielen und automatisiert

Server-Installation zu Fuß

Wer den Chef-Server per Hand aufzusetzen muss, installiert zunächst den Messaging-Server Rabbit MQ. Liegt der Distribution kein entsprechendes Paket bei, findet sich ein passendes unter [2]. Open-Suse-Nutzer sollten »rabbit-mq« über den hauseigenen Buildservice installieren [3]. Dabei bindet Yast auch gleich noch andere für den weiteren Verlauf nötige Repositories ein. Sobald Rabbit MQ auf der Platte weilt, folgt die Konfiguration für Chef:

```
sudo rabbitmqctl add_vhost /chef
sudo rabbitmqctl add_user chef testing
sudo rabbitmqctl set_permissions -p /chef chef ".*" ".*" ".*"
```

Als Nächstes folgt die Couch-DB-Datenbank über das gleichnamige Paket. Falls notwendig starten Sie den Dienst noch per Hand, unter Open Suse etwa mit »rccouchdb start«. Der Chef-Server verlangt zudem noch das Sun Java SDK in Version 1.6.0. Bei einigen Distributionen liegt das Paket in einem externen oder speziellen Repository. Unter Debian aktivieren Sie dazu die Paketquelle »non-free«, in Ubuntu 10.04 binden Sie wie folgt das Partner-Repository ein:

```
sudo add-apt-repository "deb http://archive.canonical.com/ ➥
lucid partner"
sudo apt-get update
```

Jetzt installieren Sie das JDK. In Debian und Ubuntu versteckt es sich im Paket »sun-java6-jdk«, bei Open Suse heißt es hingegen »java-1_6_0-sun-devel«. Anwender letztgenannter Distribution sollten sicherheitshalber noch die Open-JDK-Pakete »java-1_6_0-openjdk« und »java-1_6_0-openjdk-devel« löschen.

Abschließend bleiben noch die Entwicklerpakete für »zlib« und »libxml« einzuspielen. Bei Debian und Ubuntu heißen sie »zlib1g-dev« und »libxml2-dev«, bei Open Suse »zlib-devel« und »libxml-devel«. Jetzt können Sie endlich den Chef-Server installieren:

```
sudo gem install chef-server chef-server-api chef-server chef-solr
```

Außerdem noch die überaus praktische Weboberfläche hinzuholen:

```
sudo gem install chef-server-webui
```

Ist auch dies erledigt, erstellen Sie die Konfigurationsdatei »/etc/chef/server.rb«. Eine Vorlage zeigt Listing 1. In ihr müssen Sie mindestens den Domainnamen hinter »chef_server_url« gegen die Ausgabe von »hostname -f« austauschen und hinter »web_ui_admin_default_password« ein selbst gewähltes Passwort eintragen. Alle übrigen Einstellungen können Sie bedenkenlos übernehmen. Das gilt insbesondere für die Pfade, die der Server später gegenebenfalls selbst anlegt.

Im nächsten Schritt erzeugt das Skript aus Listing 2 ein paar notwendige SSL-Zertifikate. Den Suchindex erzeugt dann der folgende Aufruf:

```
sudo chef-solr-indexer
```

Ein weiterer Befehl startet den SOLR-Server:

```
sudo chef-solr
```

Dann den Chef-Server selbst:

```
sudo chef-server -N -e production
```

Dann die grafische Weboberfläche:

```
sudo chef-server-webui -p 4040 -e production
```

Nun ist im Webbrowser die Chef-Administrations-Oberfläche über den Port 4040 erreichbar.

konfigurieren. Was liegt da also näher, als Chef sich selbst installieren zu lassen. Die Entwickler nennen dieses Verfahren Bootstrapping. Passende Rezepte, die auf diese Weise den Server einrichten, existieren allerdings nur für Debian 5, Ubuntu ab 8.10 und Cent OS 5.x. Auf anderen Distributionen muss der Anwender alle Schritte per Hand durchführen, wie es der Kasten „Server-Installation zu Fuß“ vorführt.

Mit einem der offiziell unterstützten Koch-Kollegen geht es etwas flotter: Zunächst ist sicherzustellen, dass die beteiligten Computer über Fully Qualified Domain Names (FQDN) verfügen, also einen Namen der Art »chefserver.example.com« tragen. Sonst hagelt es später Fehlermeldungen der Art: »Attribute domain is not defined! (ArgumentError)«. Auch müssen die Repositories das Programm »runit« in einem Paket namens »runit« bereithalten (dieses nicht selbst installieren).

Kaffee-Ersatz

Abschließend setzt der Chef-Server noch das Sun Java SDK in Version 1.6.0 voraus, das die Distributionen jedoch gerne in einem speziellen Repository verstecken. Debian-User aktivieren dafür die Paketquelle »non-free«, Ubuntu-Benutzer binden das Partner-Repository mit folgenden zwei Zeilen ein:

```
sudo add-apt-repository "deb http://➥
archive.canonical.com/ lucid partner"
sudo apt-get update
```

Prinzipiell läuft der Chef-Server auch mit dem Open JDK, die Entwickler übernehmen dafür aber keine Garantie.

Tabelle 1: Verzeichnisse eines Repository

Verzeichnis	Inhalt
certificates/	SSL Zertifikate (in der Regel von »rake ssl_cert« erstellt)
config/	Allgemeine Konfigurationsdateien des Repository
cookbooks/	Fertige Kochbücher
roles/	Rollendefinitionen
site-cookbooks/	Modifizierte Kochbücher; die hier gespeicherten Kochbücher überschreiben oder verändern die unter »cookbooks« gelagerten

Sind alle Voraussetzungen getroffen, soll jetzt auf dem Server und auf dem Client je eine Konfigurationsdatei für Chef Solo entstehen. Diese Chef-Variante führt die Rezepte direkt auf dem Client ohne Beteiligung des Servers aus. Ohne den Server eignet sich Chef Solo höchstens als Hilfe bei der Erstellung einfacher Skripte – oder zur Installation des vollständigen Server und Clients. Dazu ist auf beiden beteiligten Systemen jeweils die Datei »~/solo.rb« mit drei Zeilen zu erstellen:

```
file_cache_path "/tmp/chef-solo"
cookbook_path "/tmp/chef-solo/cookbooks"
recipe_url "http://s3.amazonaws.com/➥
chef-solo/bootstrap-latest.tar.gz"
```

Sie teilen Chef Solo mit, wo die Rezepte für die Installation lagern.

Jetzt geht’s auf dem künftigen Server weiter. Dort soll eine Json-Konfigurationsdatei namens »~/chef.json« entstehen, die Informationen über diesen Knoten liefert. Ihren Inhalt zeigt Listing 3.

Der Servername neben »server_fqdn« ist entsprechend anzupassen. Jetzt genügt der Aufruf von

```
sudo chef-solo -c ~/solo.rb -j ~/chef.json
```

um den kompletten Chef-Server aufzusetzen. Bricht der Befehl mit einer kryptischen Fehlermeldung ab, hilft ein erneuter Aufruf – in den Tests lief die Installation dann ohne Probleme durch. Das Kommando setzt zunächst einen Chef-Client auf, installiert Rabbit MQ, Couch DB, die Entwicklerpakete zu »zlib« und »xml«, installiert den Chef-Server samt Indexer, eine Weboberfläche, über die sich der Chef-Server später einfach bedienen lässt (Web-UI), erstellt passende Konfigurationsdateien sowie einige benötigte Verzeichnisse und bindet noch den Server in die Init-Skripte ein.

Am Ende der gesamten Prozedur lauscht der Chef-Server auf Port 4000, die Weboberfläche ist über Port 4040 zu erreichen. Java und Rabbit MQ nutzen übrigens die im Chef-Server eingebaute, auf Apache SOLR basierende Volltextsuchmaschine. Sie liefert unter anderem Informationen zur vorhandenen Infrastruktur, die sich wiederum gezielt in den Rezepten weiterverarbeiten lässt. Details zur Suchfunktion liefert die Wiki-Seite unter [4].

Küchenhilfe

Sobald der Server steht, geht es auf dem Client weiter. Dort entsteht zunächst wieder eine Json-Konfigurationsdatei »~/chef.json«. Ihren Inhalt zeigt Listing 4.

Listing 1: Vorlage für »server.rb«

```
log_level          :info
log_location       STDOUT
ssl_verify_mode    :verify_none
chef_server_url    "http://chef.example.com:4000"

signing_ca_path    "/var/chef/ca"
couchdb_database   'chef'

cookbook_path      [ "/var/chef/cookbooks", "/var/chef/site-cookbooks" ]

file_cache_path    "/var/chef/cache"
node_path          "/var/chef/nodes"
openid_store_path  "/var/chef/openid/store"
openid_cstore_path "/var/chef/openid/cstore"
search_index_path  "/var/chef/search_index"
role_path          "/var/chef/roles"

validation_client_name "validator"
validation_key         "/etc/chef/validation.pem"
client_key             "/etc/chef/client.pem"
web_ui_client_name     "chef-webui"
web_ui_key             "/etc/chef/webui.pem"

web_ui_admin_user_name "admin"
web_ui_admin_default_password "somerandompasswordhere"

supportdir = "/srv/chef/support"
solr_jetty_path File.join(supportdir, "solr", "jetty")
solr_data_path  File.join(supportdir, "solr", "data")
solr_home_path  File.join(supportdir, "solr", "home")
solr_heap_size  "256M"

umask 0022

Mixlib::Log::Formatter.show_time = false
```

Der »server_fqdn« muss hier den Rechnernamen des Servers enthalten und nicht den des Clients. Damit lässt sich nun Chef Solo anwerfen:

```
sudo chef-solo -c ~/solo.rb -j ~/chef.json -r➥
 http://s3.amazonaws.com/chef-solo/bootstrap-➥
latest.tar.gz
```

Das Werkzeug erstellt ein paar Verzeichnisse, rückt die Konfigurationsdateien gerade und bindet »chef-client« in die Init-Skripte ein. Letzteres stellt sicher, dass sich der Client nach jedem Systemstart einmal beim Server meldet und die dort zwischenzeitlich vorgenommenen Rezeptänderungen ausführt.
Abschließend muss sich der Client noch beim Server registrieren. Dazu wandert eine Kopie der Datei »/etc/chef/validation.pem« vom Server in das Verzeichnis »/etc/chef/« auf dem Client, der dann einmal kurz per Hand startet:

```
sudo chef-client
```

Der Client erzeugt dabei automatisch einen Schlüssel, der in die Datei »/etc/chef/client.pem« wandert und ab sofort jede Transaktion mit dem Server signiert. Anschließend ist es aus Sicherheitsgründen am besten, die Datei »validation.pem« wieder zu löschen.

Speisekammer

Nachdem Server und Client laufen, kommt als Nächstes eine Ablage für alle Rezepte auf dem Server an die Reihe. Dieses so genannte Repository besteht einfach aus mehreren genormten (Unter-)Verzeichnissen. Die ließen sich alle per Hand anlegen, schneller geht es jedoch mit einer von Opscode bereitgestellten Vorlage, die nur herunterzuladen und zu entpacken ist:

```
wget http://github.com/opscode/chef-repo/➥
tarball/master
tar -zxf opscode-chef-repo-123454567878.➥
tar.gz
```

Die kryptische Zahl ist für die tägliche Arbeit nur schwer zu merken, es ist also besser, das Verzeichnis umzubenennen (die Zahl stammt übrigens von der Versionsverwaltung und repräsentiert die Commit-ID):

```
mv opscode-chef-repo-123454567878 chef-repo
cd chef-repo
```

Listing 2: SSL-Zertifikate für den Chef-Server

```
01 server_ssl_req="/C=US/ST=Several/L=Locality/O=Example/OU=Operations/CN=chef.example.com/
   emailAddress=ops@example.com"
02 openssl genrsa 2048 > /etc/chef/validation.key
03 openssl req -subj "${server_ssl_req}" -new -x509 -nodes -sha1 -days 3650 -key /etc/chef/validation.key
   > /etc/chef/validation.crt
04 cat /etc/chef/validation.key /etc/chef/validation.crt > /etc/chef/validation.pem
05 openssl genrsa 2048 > /etc/chef/webui.key
06 openssl req -subj "${server_ssl_req}" -new -x509 -nodes -sha1 -days 3650 -key /etc/chef/webui.key > /
   etc/chef/webui.crt
07 cat /etc/chef/webui.key /etc/chef/webui.crt > /etc/chef/webui.pem
```

Die Verzeichnisstruktur innerhalb von »chef-repo« erklärt Tabelle 1. Die hierin gelagerten Rezepte impft ein kleines Werkzeug namens »knife« dem Server ein. Um es auf seinen Einsatz vorzubereiten, ruft der Anwender den Befehl

```
knife configure -i
```

auf und bestätigt die vorgegebenen Antworten mit der Eingabetaste – bis auf zwei Ausnahmen: Bei »Your client user name?« gibt er seinen Benutzernamen ein, während er die Frage »Path to a chef repository (or leave blank)?« mit ».« beantwortet.
Der Befehl »knife« meldet jetzt auf dem Chef-Server einen neuen Client an, legt das dabei erzeugt Zertifikat in »/.chef/my-knife.pem« ab und erstellt die Konfigurationsdatei »/.chef/knife.rb«.

Fertiggericht

Mehrere Rezepte, die ein gemeinsames Ziel verfolgen, fasst Chef in einem Kochbuch (Cookbook) zusammen. So finden sich beispielsweise im Kochbuch »mysql« alle Rezepte, die notwendig sind, um die freie Datenbank zu installieren und einzurichten.
Für einen ersten Test eignet sich am besten ein einfaches, fertiges Cookbook [5]. Im Folgenden soll das Cookbook für »emacs« aus der Applications-Gruppe als Beispiel dienen. Es installiert über den Paketmanager den beliebten Texteditor Emacs. Das heruntergeladene Cookbook-Archiv lässt sich einfach ins Unterverzeichnis »cookbooks« entpacken. Anschließend erhält der Server Kenntnis von den neuen Rezepten:

```
rake upload_cookbooks
```

Das Kommando »rake« ruft »knife« automatisch mit den passenden Parametern auf, das wiederum alle Cookbooks aus dem gleichnamigen Verzeichnis hochlädt. Ein einzelnes Cookbook schiebt

```
rake upload_cookbook[emacs]
```

auf den Server. Das Target »upload_cookbook« ist übrigens in dem über das Repository mitgelieferten »Rakefile« bereits definiert.

Glasur

Der Server kennt damit das Cookbook »emacs«, die Clients stehen aber noch wie der Ochs vorm Berg. Um das zu ändern, öffnet der Anwender einen Browser und betritt die Weboberfläche über die Adresse [http://chefserver.example.

Listing 3: »~/chef.json« für den Server

```
01 {
02   "bootstrap": {
03     "chef": {
04       "url_type": "http",
05       "init_style": "runit",
06       "path": "/srv/chef",
07       "serve_path": "/srv/chef",
08       "server_fqdn": "chefserver.example.com",
09       "webui_enabled": true
10     }
11   },
12   "run_list": [ "recipe[bootstrap::server]" ]
13 }
```

Listing 4: »~/chef.json« für den Client

```
01 {
02   "bootstrap": {
03     "chef": {
04       "url_type": "http",
05       "init_style": "runit",
06       "path": "/srv/chef",
07       "serve_path": "/srv/chef",
08       "server_fqdn": "chefserver.example.com"
09     }
10   },
11   "run_list": [ "recipe[bootstrap::client]" ]
12 }
```

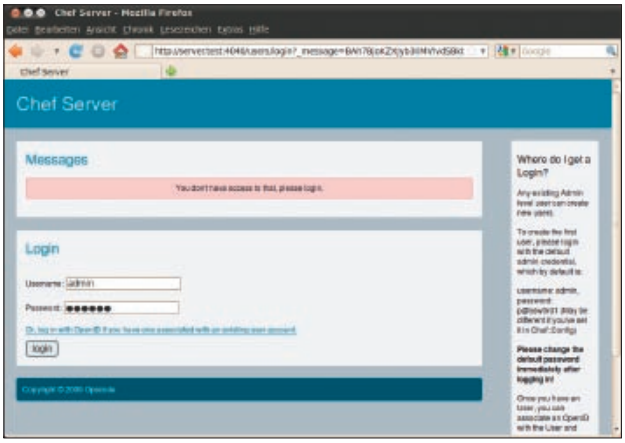


Abbildung 2: Die Anmeldeseite der Weboberfläche: Das in der rechten Leiste genannte Default-Passwort stimmt nicht.

com:4040]. Chef selbst bietet an dieser Stelle allerdings keine SSL-Verschlüsselung. Wer hier auf Nummer sicher gehen möchte, muss zur Absicherung Apache als Proxy vorschalten.

Am erscheinenden Formular meldet sich der Benutzer mit dem Namen »admin« an (Abbildung 2). Das zugehörige Passwort nennt die Datei »/etc/chef/server.rb« in der Zeile »web_ui_admin_default_password«. Die etwas kryptische Vorgabe lässt sich am besten gleich nach dem ersten Login ändern.

Anschließend geht es weiter zum Menüpunkt »Nodes«. Dort klickt der Benutzer den Namen des Clients an, wechselt auf das Register »Edit« und zieht schließlich das anzuwendende Rezept aus »Available Recipes« per Drag & Drop in die »Run List« (das Rezept rastet am oberen Rand der Liste ein). In dem Beispiel sollte dort jetzt »emacs« stehen (Abbildung 3). Um die Zuweisung zu speichern, ist noch ganz unten links auf der Seite die Schaltfläche »Save Node« zu aktivieren.

Per Hand erfolgt danach der Aufruf des Werkzeugs »chef-client« auf dem Client :

```
sudo chef-client
```

Es verbindet sich umgehend mit dem Server, holt die ihm zugewiesenen Rezepte (im Moment nur »emacs«) und führt sie aus (Abbildung 5). Damit das regelmäßig passiert, lässt sich der Client als Daemon in Intervallen starten:

```
chef-client -i 3600 -s 600
```

In diesem Beispiel meldet sich der Client alle 3600 Sekunden beim Server. Der Parameter »-s« sorgt dafür, dass der Zeitraum leicht variiert. Andernfalls könnte es dazu kommen, dass mehrere Clients gleichzeitig beim Server anklopfen und sich dabei gegenseitig blockieren.

Ausrollen

Mehrere Cookbooks lassen sich zu einer so genannten Rolle (Role) zusammenfassen. Dazu dient im Repository unter »roles« eine neue Datei – etwa »beispiel.rb« – mit folgendem Inhalt:

```
name "beispiel"
description "Eine kleine Beispielrolle"
run_list("recipe[emacs]", "recipe[zsh]",➥
 "recipe[git]")
```

Hier würden die Rezepte »emacs«, »zsh« und »git« unter dem Namen »beispiel« zusammengefasst. Von der neuen Rolle erfährt der Server dann via:

```
rake roles
```

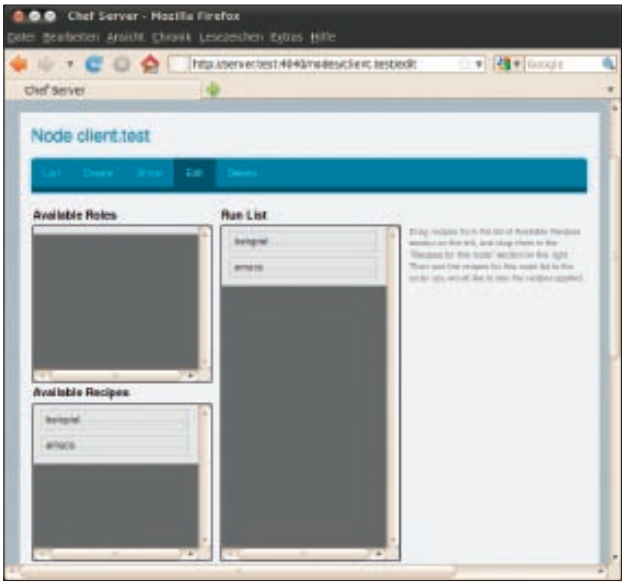
In der Weboberfläche lassen sich jetzt die Rollen genau wie die Cookbooks per Drag & Drop einem »Node« zuweisen.

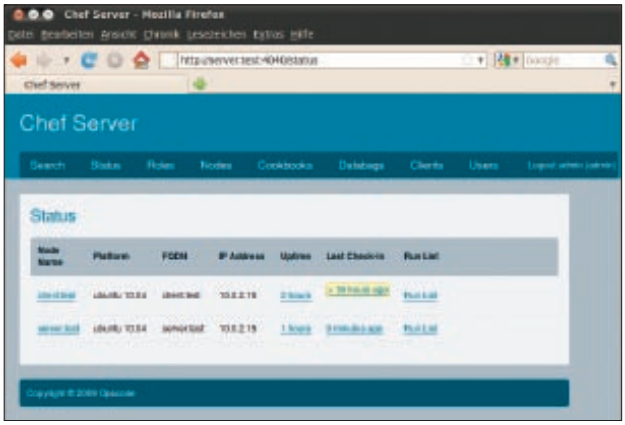
Frisch angerührt

Die vorgefertigten Rezepte und Kochbücher aus dem Internet decken meist nur Standardfälle ab. Für besondere beziehungsweise individuelle Konfigurationen muss ein eigenes Kochbuch her. Das folgende, extrem einfache Beispiel erstellt in Anlehnung an [6] auf dem Client die Textdatei »/tmp/gedanken.txt« und bestückt sie mit einem teilweise dynamisch generierten Satz. Dazu ist zunächst im »chef-repo« ein neues Kochbuch namens »beispiel« anzulegen:

```
rake new_cookbook COOKBOOK=beispiel
```

Der Befehl erzeugt unter »cookbooks« den Ordner »beispiel«, füllt ihn mit allen benötigten Unterverzeichnissen und legt schließlich noch ein leeres Rezept namens »default.rb« an.





▲ **Abbildung 4: Das Register »Status« führt alle Knoten mit ihren letzten Kontaktversuchen und zugewiesenen Rezepten (hinter »Run List«) auf.**

◀ **Abbildung 3: Rezepte weist man einem Knoten einfach per Drag & Drop zu. Hier führt der Client erst das Rezept »beispiel«, danach »emacs« aus.**

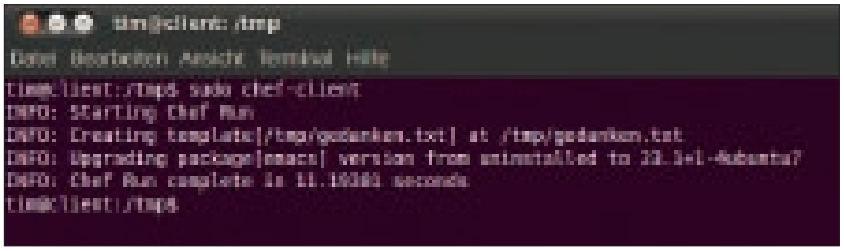


Abbildung 5: Der Client hat hier das für ihn gültige Rezept vom Server geholt und ausgeführt. Damit landet Emacs fertig konfiguriert auf der Festplatte.

Um es später mit Inhalt zu füllen, ist zunächst noch eine Vorlage (Template) für die zu erstellende Datei »/tmp/gedanken.txt« erforderlich. Sie soll später den Satz

```
Heutiger Gedanke:
```

enthalten, dem das Rezept dann einen entsprechenden Geistesblitz anhängt. Das vollständige Template dazu lautet kurz und knapp:

```
Heutiger Gedanke: <%= @gedanke %>
```

Den Platzhalter ersetzt das Rezept später durch einen passenden Text. Das neue Template gehört in das Verzeichnis »templates/default/« und soll den Namen »gedanken.txt.erb« tragen. Die meisten Rezepte nutzen derartige Templates – oder um die Entwickler zu zitieren: „We love templates."

Jetzt schreibt der Anwender ein passendes Rezept, das aus der Vorlage die fertige Datei »/tmp/gedanken.txt« bastelt. Es spart Arbeit, einfach das schon bestehende, aber noch leere Rezept »default.rb« im Unterverzeichnis »recipes« zu erweitern. Für das Beispiel sieht das Rezept wie folgt aus:

```
template "/tmp/gedanken.txt" do
        source "gedanken.txt.erb"
        variables :gedanke => node[:gedanke]
        action :create
end
```

Für Ruby-Kenner sollte es weitgehend selbsterklärend sein: Es erstellt die Datei »/tmp/gedanken.txt« aus der Vorlage »gedanken.txt.erb«, in der es den Platzhalter durch den Inhalt der Variablen »gedanke« ersetzt. Bleibt nur noch zu klären, welcher Gedanke hier zum Einsatz kommt.

Gewürzregal

In diesem kleinen Beispiel soll »gedanke« ein Attribut sein. Solche Attribute speichern in einem Cookbook bestimmte Einstellungen für die Nodes, die das Rezept wiederum auswerten und benutzen kann. Ein typisches Attribut wäre beispielsweise ein Kommandozeilen-Parameter für ein automatisch vom Rezept gestartetes Programm.

Attribute sind bei jeder Rezeptausführung gleich – und somit unterm Strich nichts anderes als Konstanten, die der Rezeptschreiber vorgibt. Im Gegensatz zu echten Ruby-Konstanten lassen sich Attribute über das Webinterface ändern (auf der gleichen Seite, auf er man einem Knoten ein Cookbook zuweist).

Ein Kochbuch sammelt alle Attribute in seinem Unterverzeichnis »attributes«. Für das Beispiel entsteht dort die Datei »beispiel.rb« mit folgendem Inhalt:

```
gedanke "Reden ist Silber ..."
```

Das war bereits alles. Jetzt bleibt nur noch, das neue Cookbook beim Server anzumelden

```
rake upload_cookbooks
```

und dann über die Weboberfläche einem oder mehreren Nodes zuweisen. Dort sollte nach einem neuen Durchlauf von »chef-client« die Datei »/tmp/gedanken.txt« auftauchen.

Variationen

Das vorgestellte Rezept bietet noch viel Raum für Verbesserungen. Beispielsweise könnte man den Gedanken per Zufall wählen lassen. Für geübte Ruby-Programmierer sollte dies eine einfache Fingerübung sein. Da Chef-Rezepte reine Ruby-Skripte sind, kann der Administrator hier aus dem vollen Sprachumfang sowie allen Rubygems schöpfen. Bei Letzteren sollte das Rezept aber zunächst prüfen, ob das Gem auf dem Client überhaupt existiert, und es falls nötig nachinstallieren.

Meta-Informationen über das neue Cookbook speichert die Datei »beispiel/metadata.json«. Wer das Cookbook weitergibt beziehungsweise in den Produktivbetrieb übernimmt, sollte hier die entsprechenden Informationen ergänzen. Wie die Datei-Endung schon andeutet, nutzt die Datei das Json-Format.

Fazit

Chef ist eine recht komplexe Software zum Konfigurationsmanagement. Läuft sie jedoch erst einmal und sind angepasste Rezepte geschrieben, erleichtert sie die Arbeit eines Administrators enorm – zumindest auf Linux-Systemen.

Leider legen die Entwickler die Hürden für Einsteiger recht hoch. So gibt sich die Online-Dokumentation von Chef ziemlich chaotisch und lückenhaft [7]. Wer mehr über den Aufbau eines Cookbook wissen möchte, lädt sich deshalb am besten fertige Bücher herunter und studiert diese. Das Cookbook für »emacs« zeigt beispielsweise, wie man mit »action :upgrade« ein Paket installiert:

Darüber hinaus bleibt der Anwender weitgehend auf sich gestellt, Anleitungen oder Hilfen im Netz sind rar. Bei Problemen sollten sich die Betroffenen daher an eine der beiden englischsprachigen Mailinglisten wenden [8]. *(ofr)* ■

Infos

[1] Installationsanleitung im Chef-Wiki: [http://wiki.opscode.com/display/chef/Installation]

[2] Rabbit MQ: [http://www.rabbitmq.com/server.html]

[3] Open Suse Buildservice für Rabbit MQ: [http://software.opensuse.org/search?q=rabbitmq-server&baseproject=openSUSE%3A11.2]

[4] Informationen zur Chef-Volltextsuche: [http://wiki.opscode.com/display/chef/Search+Indexes]

[5] Verzeichnis mit fertigen Cookbooks: [http://cookbooks.opscode.com/]

[6] Cookbook Quick Start: [http://wiki.opscode.com/display/chef/Cookbook+Quick+Start]

[7] Chef-Wiki: [http://wiki.opscode.com/display/chef/]

[8] Chef-Mailinglisten: [http://lists.opscode.com/sympa/lists/opensource/chef]