# What is Chef..

..and how we use it at

# Who am I?

## Giedrius Rimkus

Lithuanian
PHP Developer at tripsta
Ruby enthusiast
Basketball lover

twitter @giedriusr

# I'll be talking about..

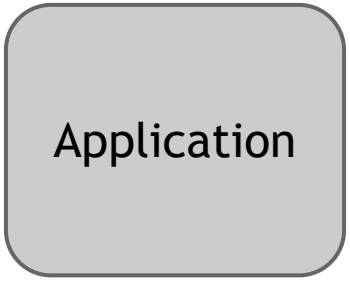
Sentiments

Pain

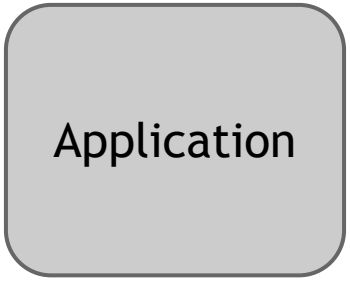
Solution

# Back in 2009..

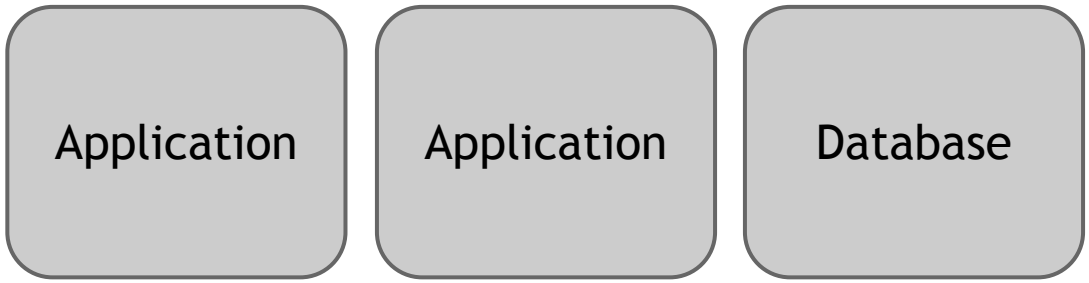
# Infrastructure



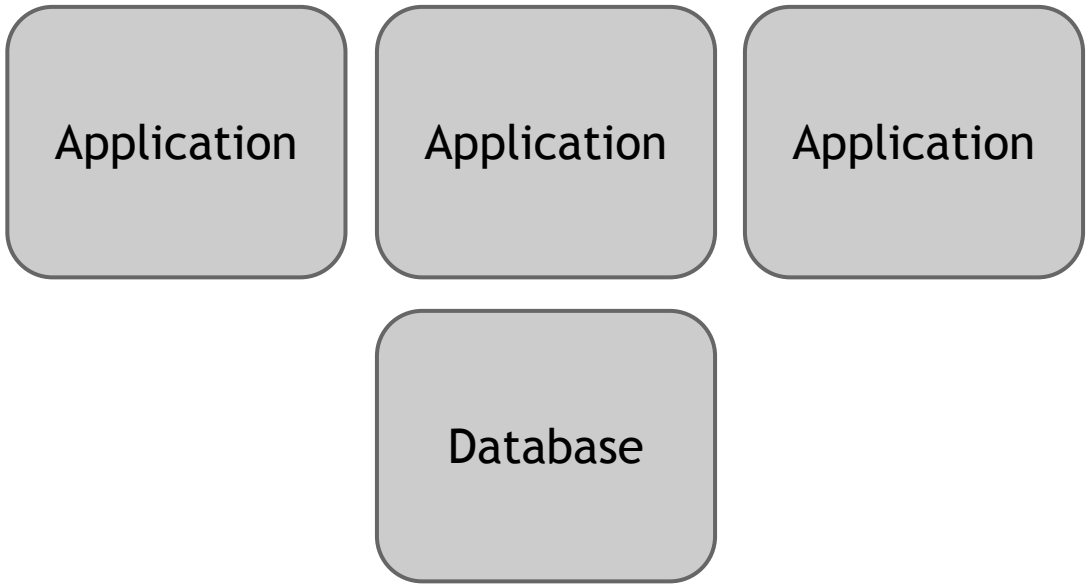
# Infrastructure



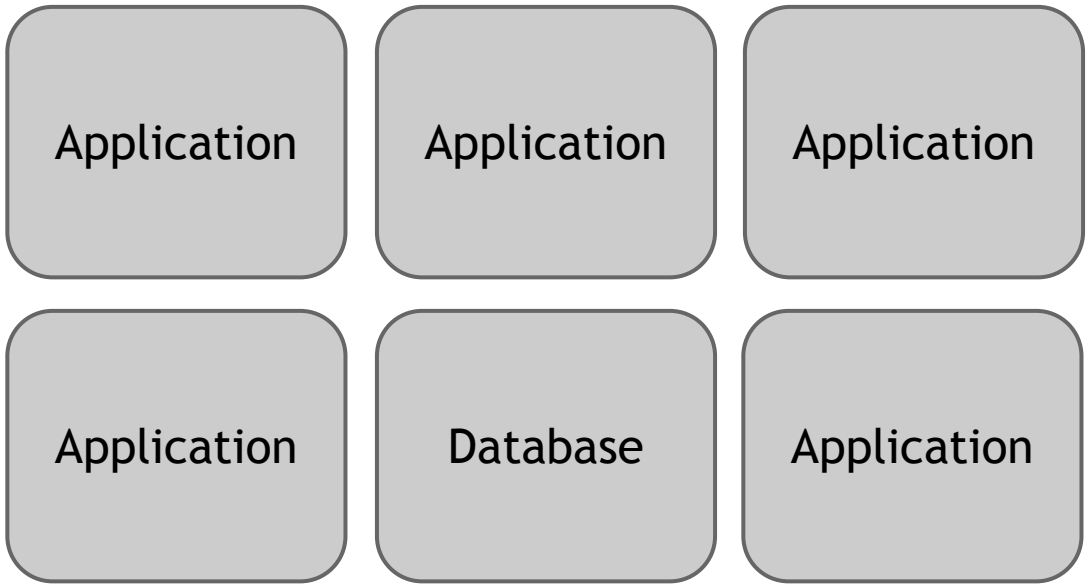
# Infrastructure



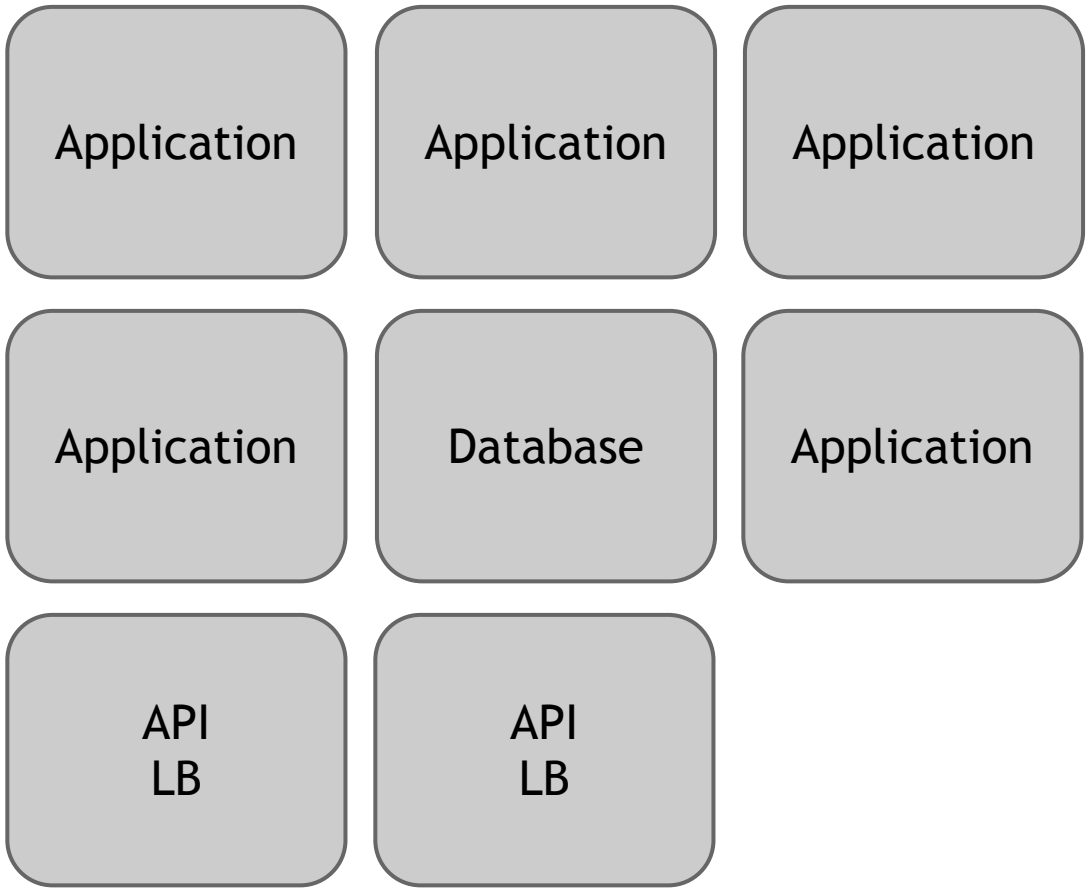
# Infrastructure



# Infrastructure



# Infrastructure



# Complexity

# Pain

- Long installation process
- Hard maintenance
- Scaling issues
- Constantly increased load and usage of resources (no load balancer)

# Solution

# CLUSTER

# WTH IS CLUSTER?

# What is a Cluster?

It's a group of linked computers, working together closely thus in many respects forming a single computer.

# Cluster categories

## High-availability (HA) clusters

High-availability clusters (also known as failover cluster) are implemented primarily for the purpose of improving the availability of services that the cluster provides.

## Load-balancing clusters

Load-balancing is when multiple computers are linked together to share computational workload or function as a single virtual computer. Logically, from the user side, they are multiple machines, but function as a single virtual machine.

## Compute clusters

Somes times called as a “Grid computing”. Basically used for calculating huge stats, etc.

# HOW TO BUILD IT?


IN CASE OF
EMERGENCY
Call 911

OR


I will ask for
help before my
problems get out
of control.

RAGNARSON

PERFECT PROGRAMMERS


Yandy.com


HOT & SPICY


Chef

**What is Chef? What problem does it solve?**

Chef is an open-source systems integration framework built specifically for automating the cloud / system configuration.

# Chef types

Chef Solo

Chef client and Chef server

Hosted Chef

Private Chef

# Chef Solo..

..is an open source standalone version of Chef that runs locally on your node, detached from a Chef server.

# Chef Client and Chef Server

Chef-client connects to a Chef Server to be told what to do on the node.

# Hosted Chef

As with Chef-Server, Chef-client connects to Hosted Chef to be told what to do on the local node.

# Private Chef..

..is for Enterprises who want the power, flexibility, availability, and performance of Hosted Chef, but require that information never leave their private networks.

# Why it's an issue?

Infrastructure changes all the time.
Different operating systems
Different hardware from different vendors.

# What exactly can you do with Chef?

- Install Operating Systems on new servers.
- Install application software on servers.
- Have new software automatically configure itself to match your environment.
- Share recipes (and obtain recipes from) other people to install and configure software.

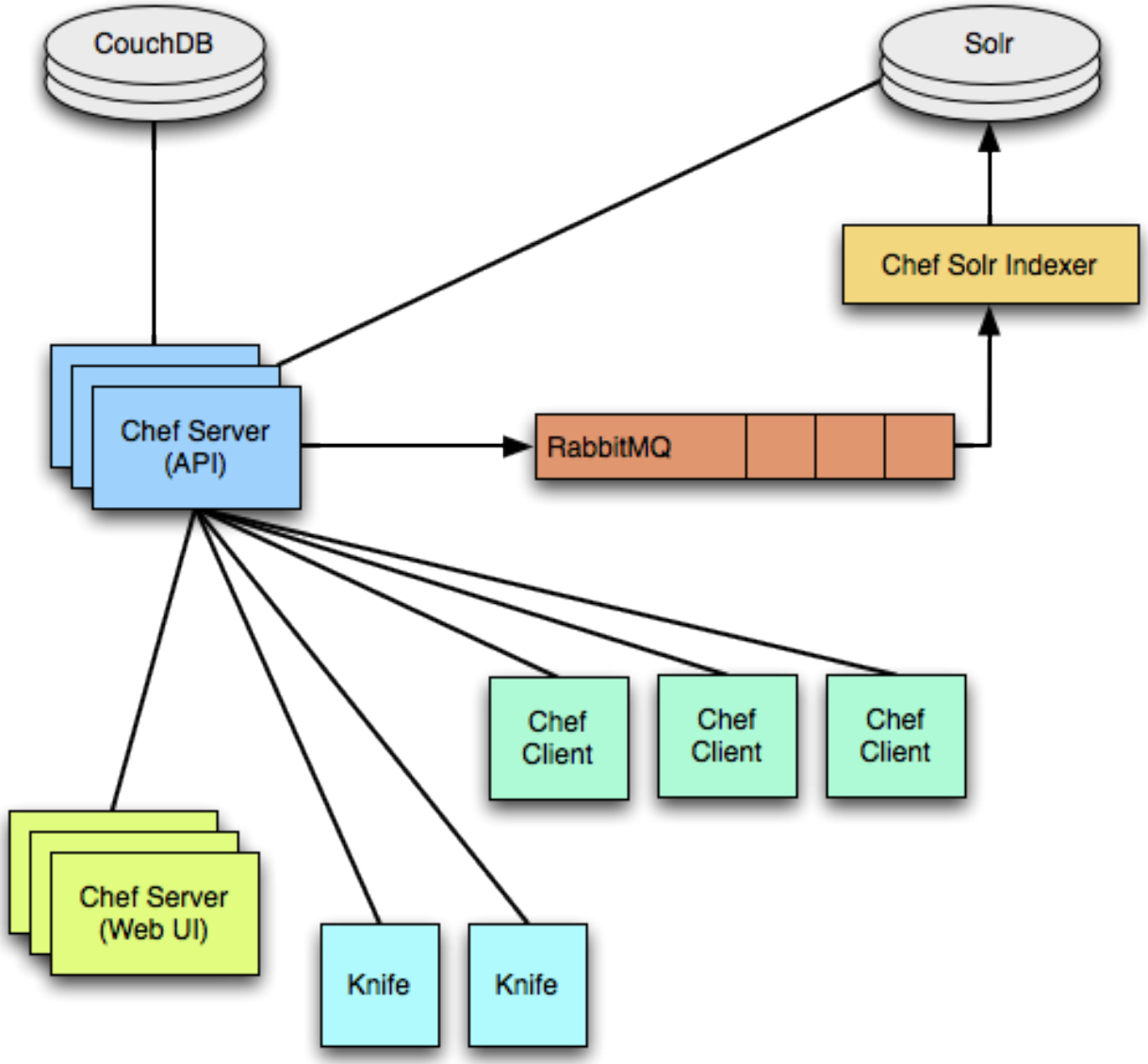
# Some goodies I like about Chef

"Manage your servers by writing code, not by running commands."

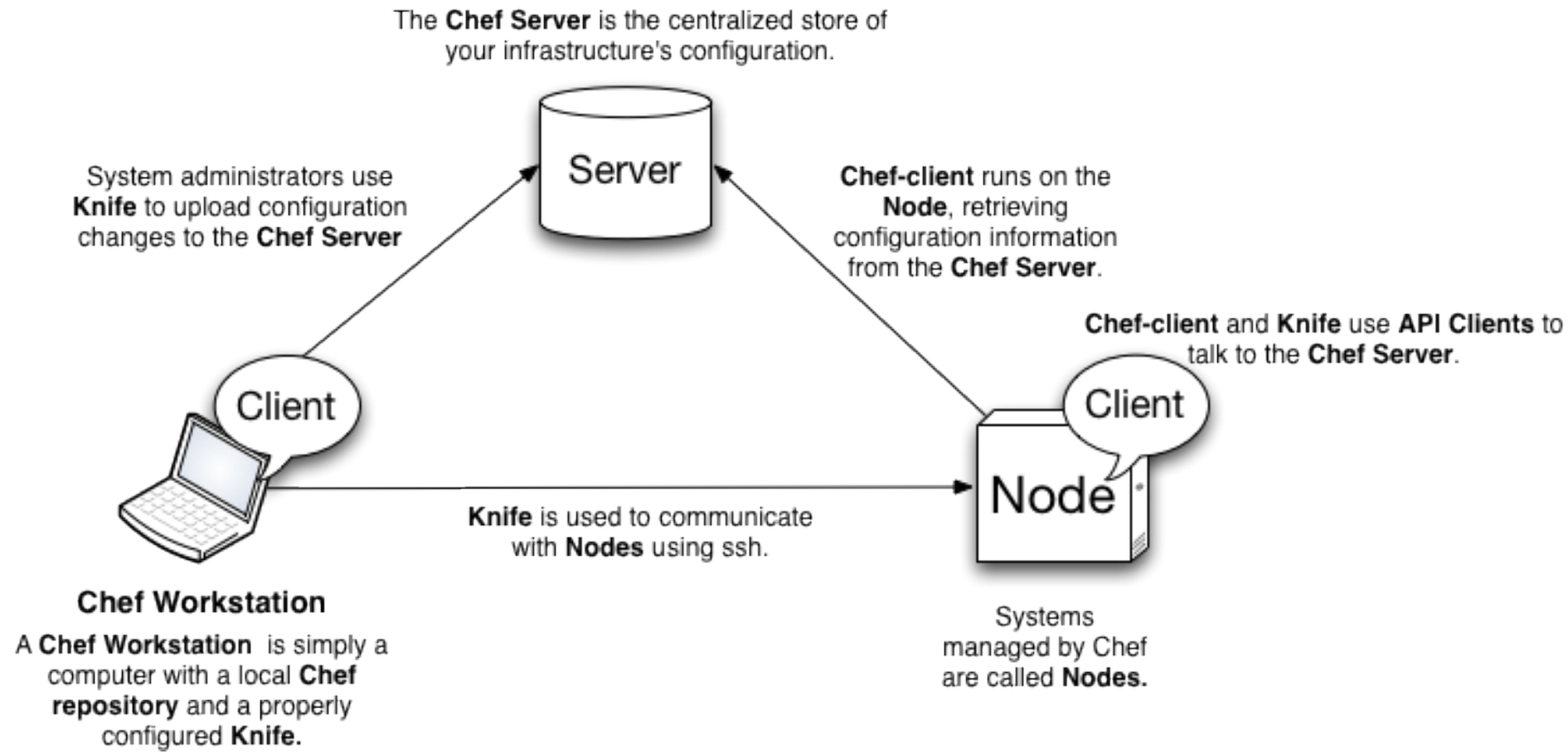
Chef is idempotent

Built on top of Ruby

# Chef Server Architecture



# Architecture visualization



# Basic structure of Chef

Environments
Nodes
Cookbooks
Recipes
Files / Templates
Attributes
Data Bags
Search

# What is what?

What is recipe?
What is cookbook?
What is resource?
What is node?
What is chef-client?
What is knife?
What is data bag?
What is template?

# Modeling your infrastructure

# Nodes

A **node** is a host that runs the Chef client. The primary features of a node, from Chef's point of view, are its **attributes** and its **run list**. Nodes are the thing that Recipes and Roles are applied to.

# Roles

A **role** means grouping similar features of similar nodes.

# Run list

A list of recipes that a node will run.

# Configuring Nodes

# Cookbooks

A **cookbook** is a collection of recipe, resource definition, attribute, library, cookbook file and template files that chef uses to configure a system. Cookbooks are typically grouped around configuring a single package or service.

The MySQL cookbook, for example, contains recipes for both client and server.

```
giedrius@lithuania:~/Projects/chef (master) $ tree -a cookbooks/mysql
cookbooks/mysql
├── README.rdoc
├── attributes
│   └── server.rb
├── metadata.rb
├── recipes
│   ├── client.rb
│   ├── default.rb
│   └── server.rb
└── templates
    └── default
        ├── debian.cnf.erb
        ├── grants.sql.erb
        ├── my.cnf.erb
        ├── mysql-server.seed.erb
        └── slow_log_rotate.erb

4 directories, 11 files
```

# Recipes

Recipes are the files where you write your resources (code).

```
package "nginx"

directory node[:nginx][:log_dir] do
  mode 0755
  owner node[:nginx][:user]
  action :create
end

%w{nxensite nxdissite}.each do |nxscript|
  template "/usr/sbin/#{nxscript}" do
    source "#{nxscript}.erb"
    mode 0755
    owner "root"
    group "root"
  end
end
```

# Another example

chef/cookbooks/git/recipes/default.rb

```
package "git-core"
# apt-get install git-core
# yum install git-core
# etc..
```

# Another example

```
directory "/home/new_folder" do
  mode 0755
  owner "someuser"
  group "www"
  action :create
end
```

# Metadata

Cookbooks often rely on other cookbooks for pre-requisite functionality. In order for the server to know which cookbooks to ship to a client, a cookbook that depends on another one needs to express that dependency somewhere. That "somewhere" is in cookbook metadata.

# Resources

A resource is usually a cross platform abstraction of the thing you're configuring on the host.

Chef's resources are mostly just containers for data, with some basic validation functionality.

# Resources

Have a type

Have a name

Have parameters

Take action to put the resource in the declared state

# Type

```
package "apache2" do
  version "2.2.11-2ubuntu2.6"
  action :install
end

template "/etc/apache2/apache2.conf" do
  source "apache2.conf.erb"
  owner "root"
  group "root"
  mode 0644
  action :create
end
```

# Name

```
package "apache2" do
  version "2.2.11-2ubuntu2.6"
  action :install
end

template "/etc/apache2/apache2.conf" do
  source "apache2.conf.erb"
  owner "root"
  group "root"
  mode 0644
  action :create
end
```

# Parameters

```
package "apache2" do
  version "2.2.11-2ubuntu2.6"
  action :install
end

template "/etc/apache2/apache2.conf" do
  source "apache2.conf.erb"
  owner "root"
  group "root"
  mode 0644
  action :create
end
```

# Action

```
package "apache2" do
  version "2.2.11-2ubuntu2.6"
  action :install
end

template "/etc/apache2/apache2.conf" do
  source "apache2.conf.erb"
  owner "root"
  group "root"
  mode 0644
  action :create
end
```

# Providers

The provider is the platform-specific implementation of the thing a resource abstracts.

On Red Hat or CentOS - yum

Debian and Ubuntu - apt package manager will be used

# Search

Search is built by the Chef Server, and allow you to query arbitrary data about your infrastructure

# Data Bags

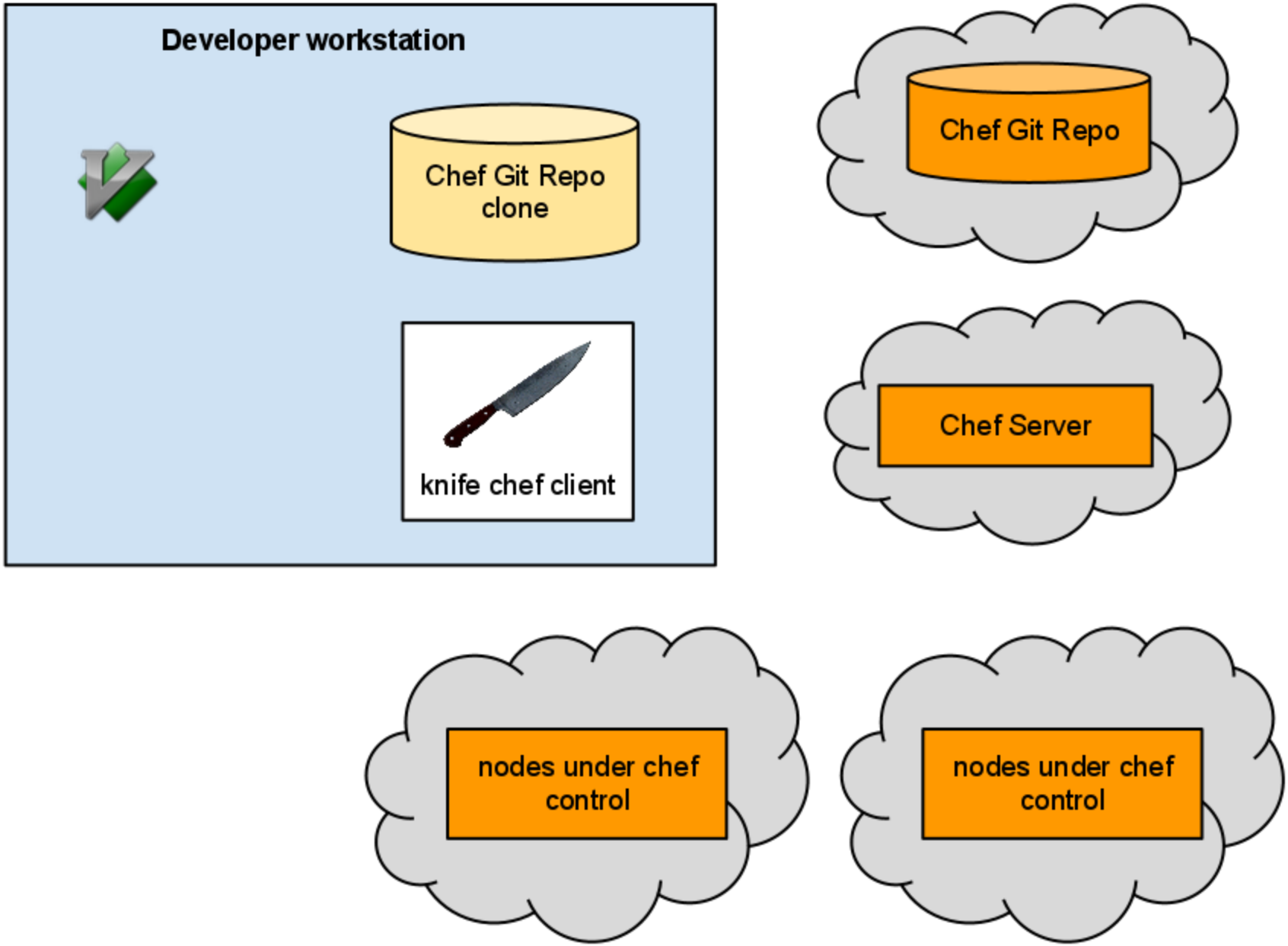
Data Bags store nested key–value data on the chef server. Data Bag data are searchable, and can also be loaded directly by name in a recipe. Data Bags are global for your chef-server installation–you can think of them as attributes for your whole infrastructure.

# Environments

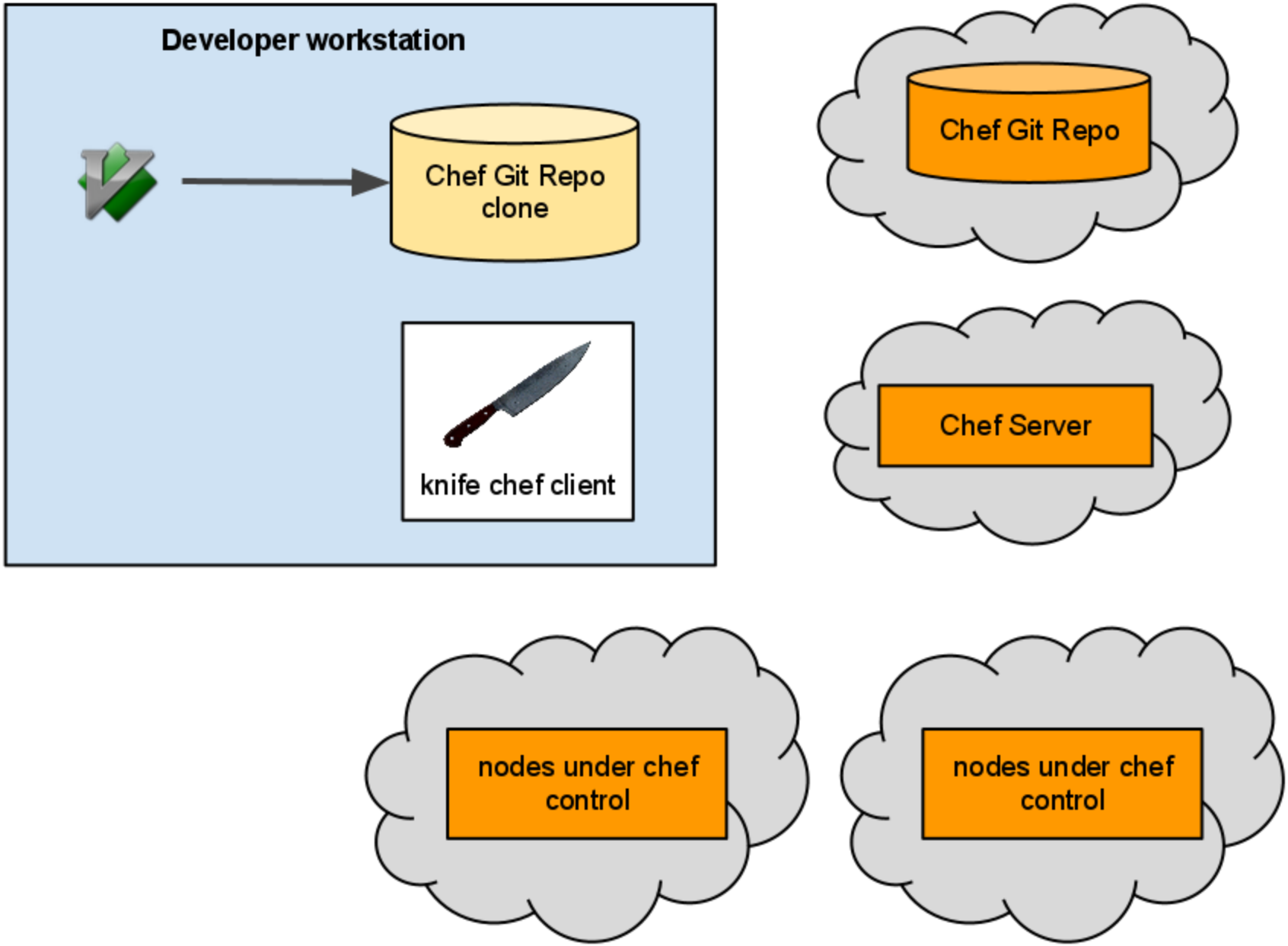
Provides a mechanism for managing different architectural segmented spaces such as production, staging, development, and testing, etc with one Chef setup.

# Chef Workflow

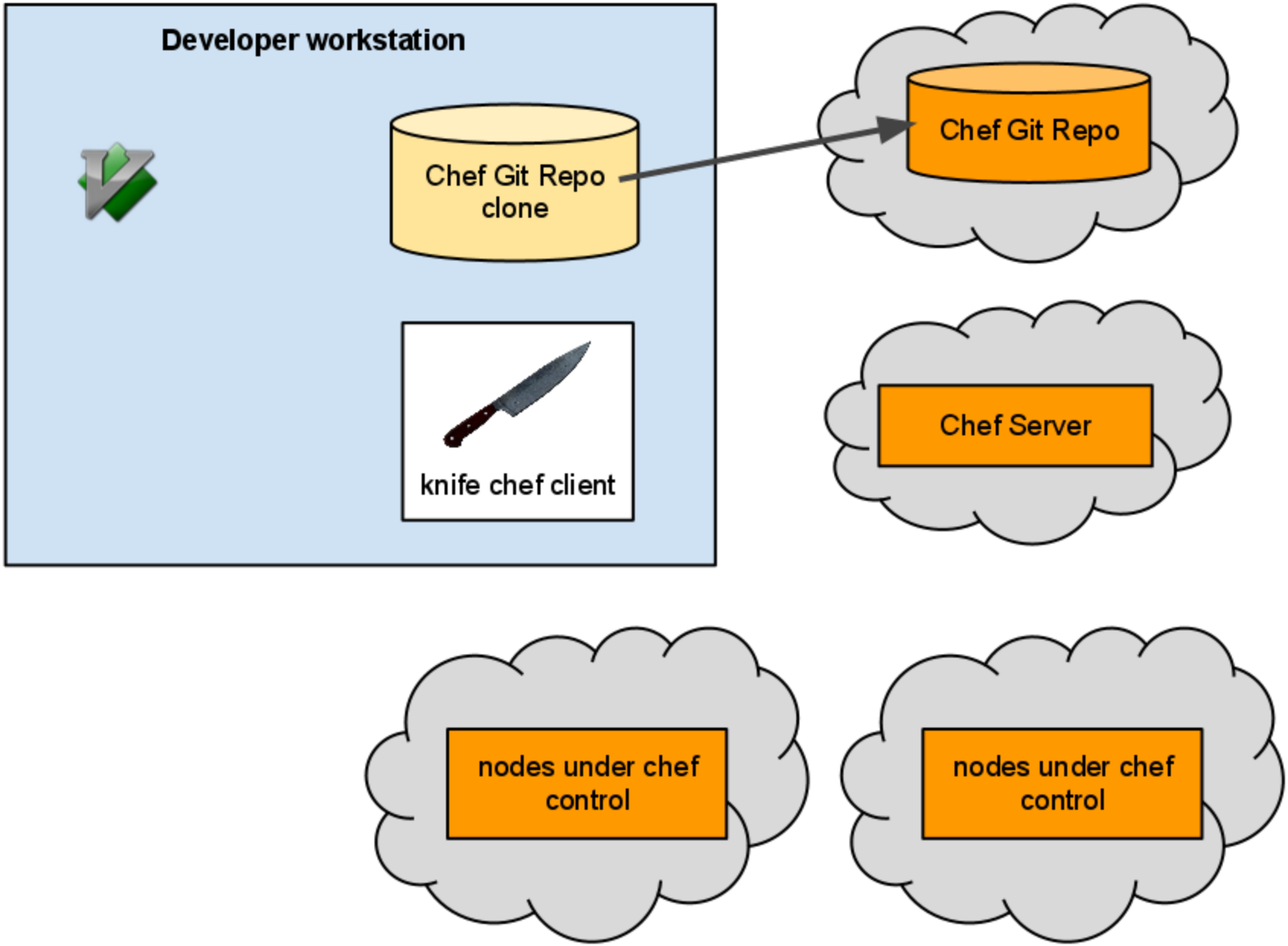
# Everyday Chef Workflow for developers



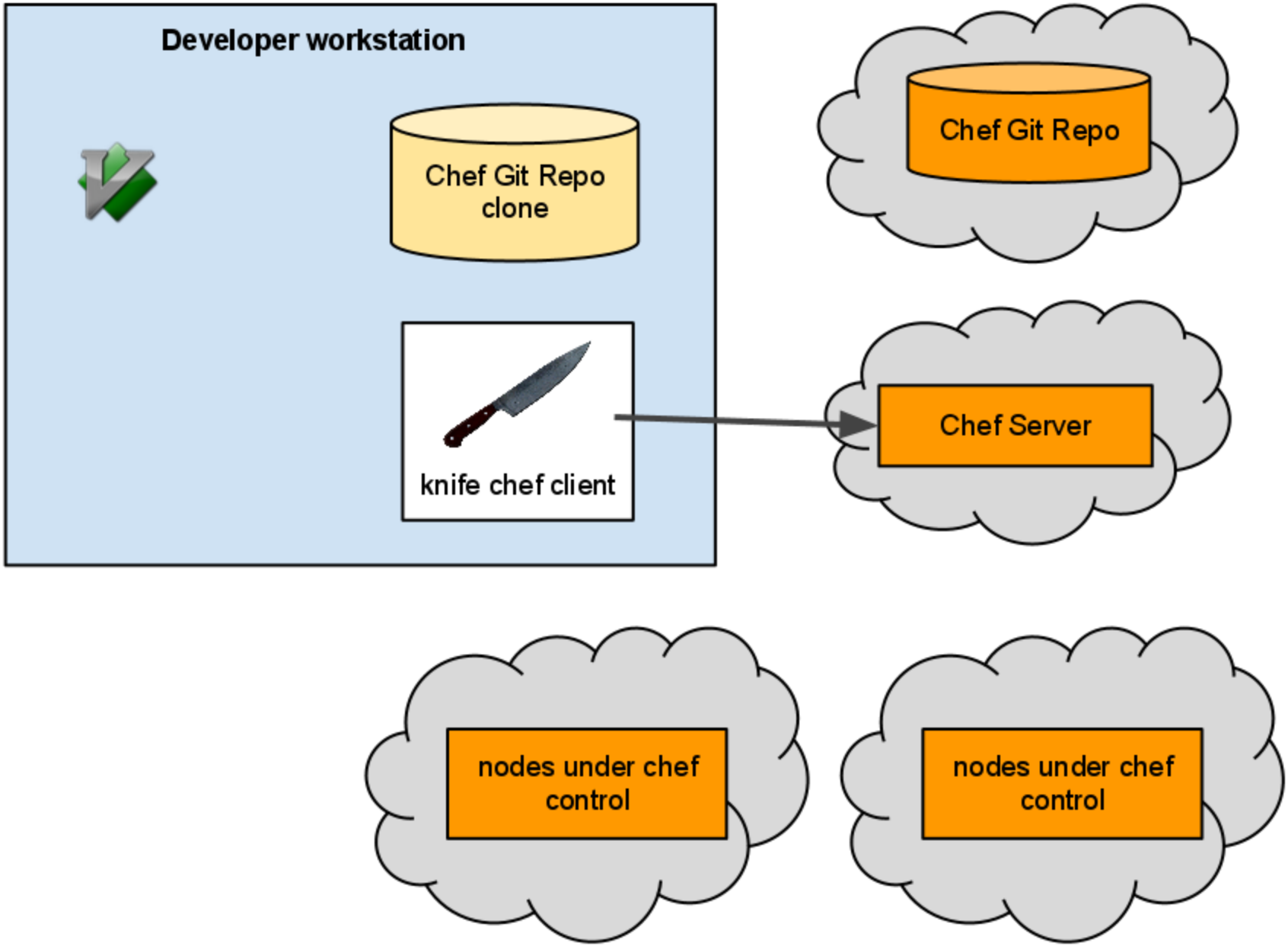
# Developer makes changes to Chef recipes, roles, etc.



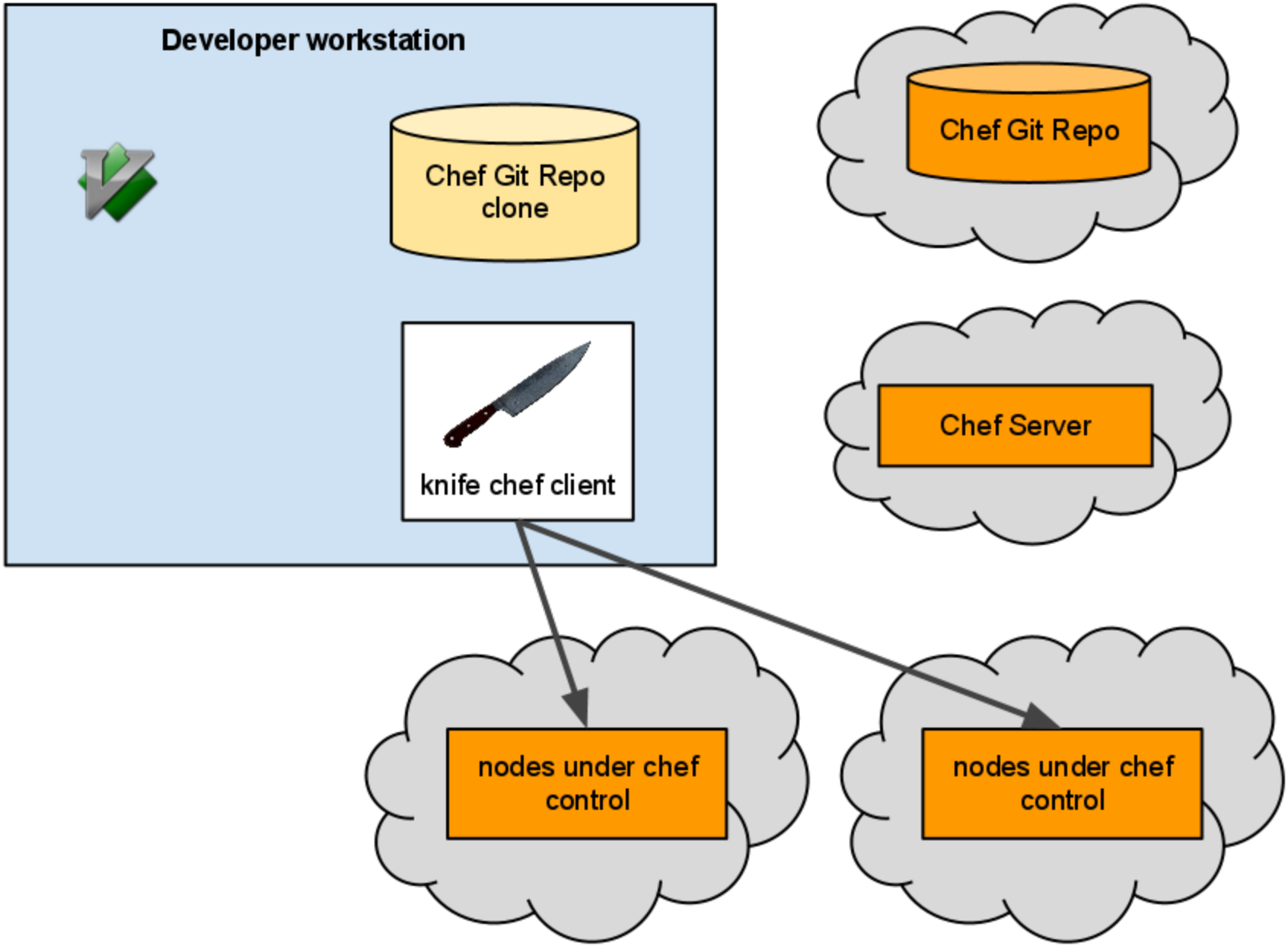
# Developer pushes changes to origin Chef repository



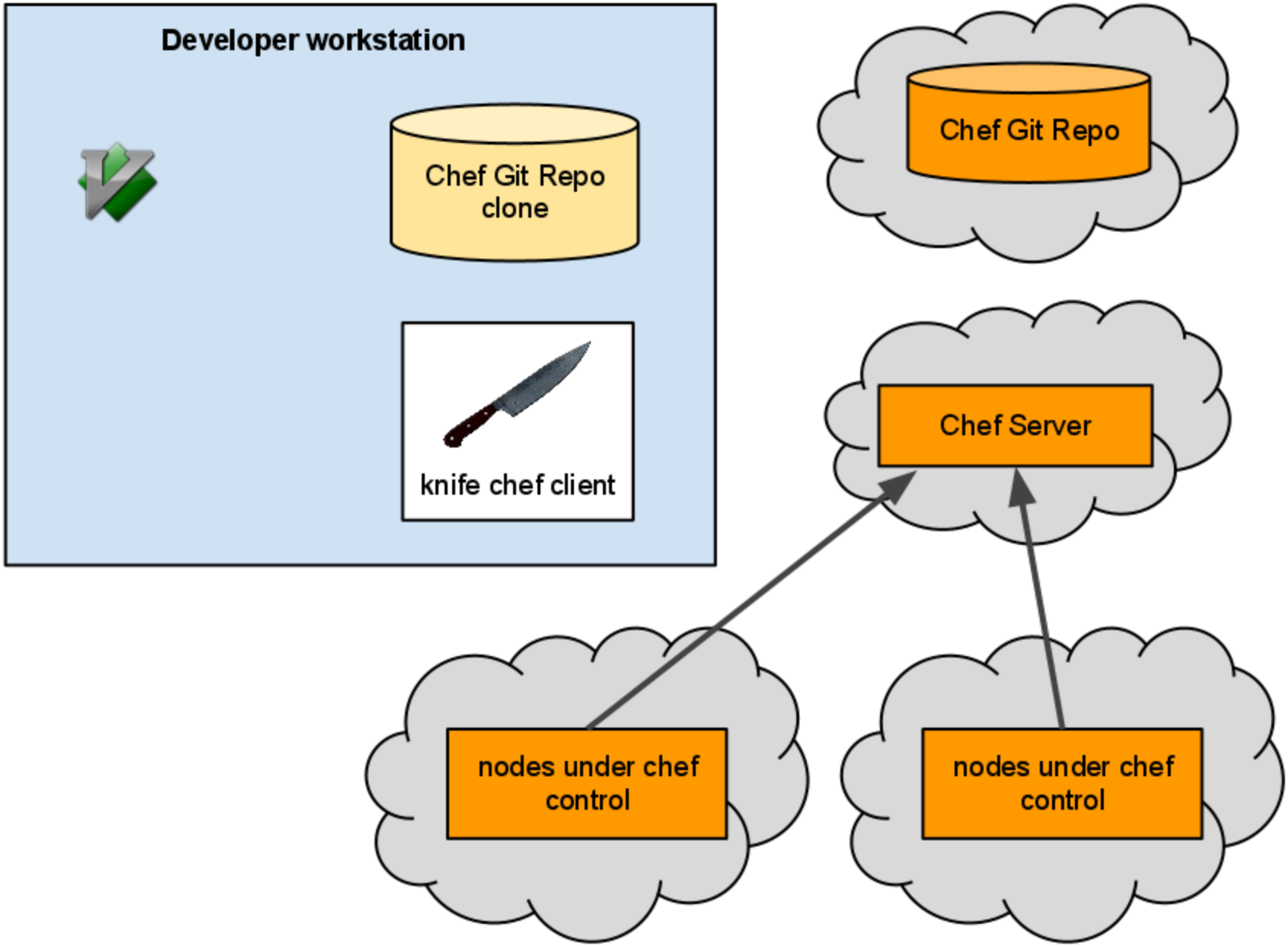
# Developer uses Knife to push new code to the Chef Server



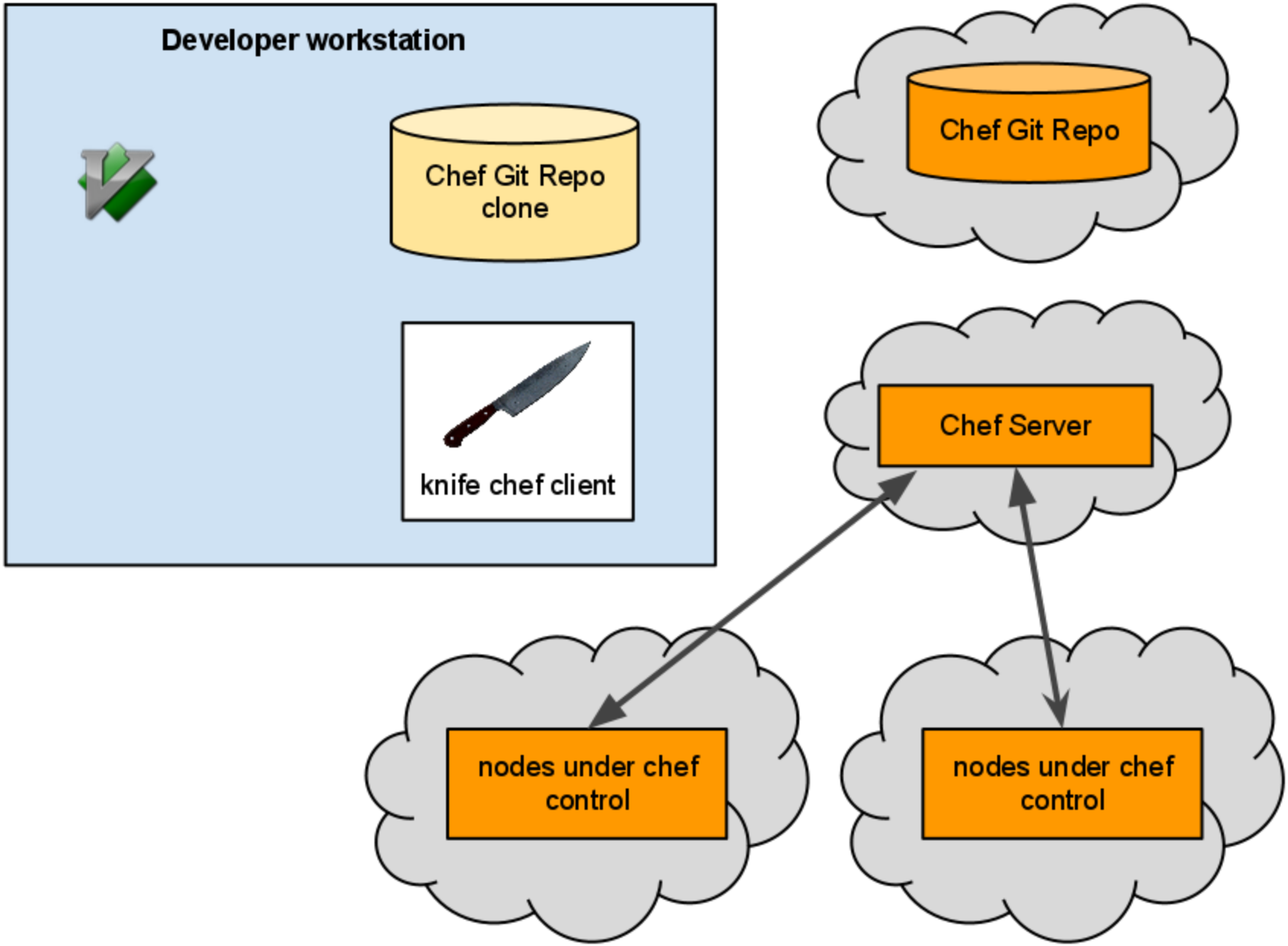
# Developer uses Knife to tell Chef-clients to update themselves



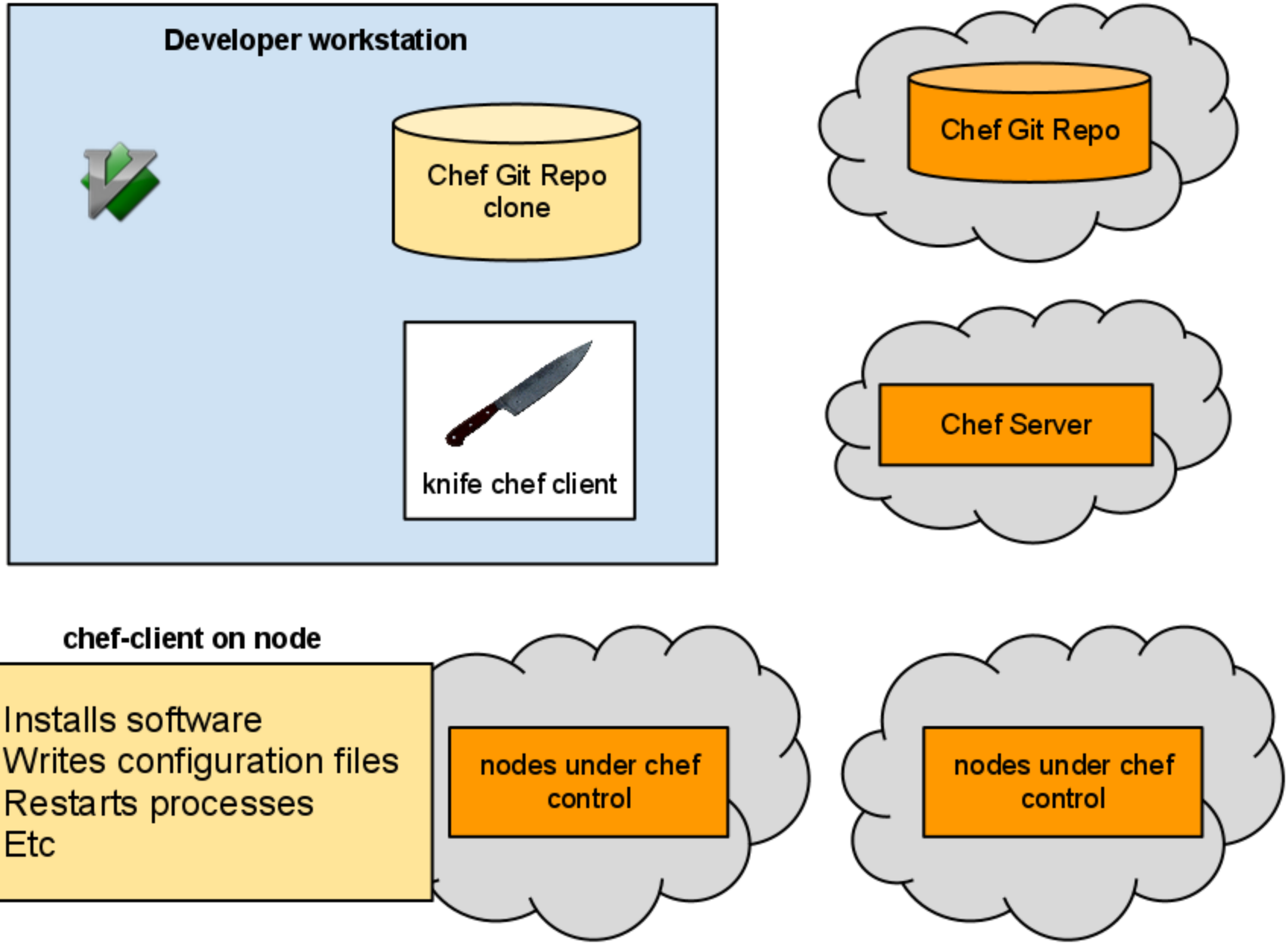
# Chef-client on Chef nodes contact the Chef Server



# Chef-client pulls latest code from the Chef Server



# Chef-client on the node updates the system



# Practical example

# Update authorized_keys

```
$ cat cookbooks/bootstrap/files/default/authorized_keys
```

```
giedrius@lithuania:~/Projects/chef (master) $ cat cookbooks/bootstrap/files/default/authorized_keys
ssh-rsa dev1-public-key
ssh-rsa dev2-public-key
ssh-rsa dev3-public-key
```

```
$ echo 'ssh-rsa dev4-public-key' >> !$
```

# Committing and pushing

```
$ git diff
```

```
giedrius@lithuania:~/Projects/chef (master) $ git diff
diff --git a/cookbooks/bootstrap/files/default/authorized_keys b/cookbooks/bootstrap/files/default/authorized_keys
index 91cc004..8481dc6 100644
--- a/cookbooks/bootstrap/files/default/authorized_keys
+++ b/cookbooks/bootstrap/files/default/authorized_keys
@@ -1,3 +1,4 @@
 ssh-rsa dev1-public-key
 ssh-rsa dev2-public-key
 ssh-rsa dev3-public-key
+ssh-rsa dev4-public-key
```

```
git add .
git commit -m 'adding new public key'
git push
```

# Updating Chef Server

```
bundle exec knife cookbook upload bootstrap

cap configure:all
```

or

```
ssh root@127.0.0.1
$ chef-client
```

# How Chef helped us?

# We don't care that much about

Infrastructure changes all the time

Scalability, because..

- adding new nodes is painless
- it's fast (takes minutes or hours, not days or weeks)
- there is no need in buying more machines with every new website

# Clear separation between servers

with roles:

Frontend
Backend
Application
Memcached
Database
other..

# Chef requirements

**Chef-client is supported on the following platforms**

- Ubuntu (10.04, 10.10, 11.04, 11.10)
- Debian (5.0, 6.0)
- RHEL & CentOS (5.x, 6.x)
- Fedora 10+
- Mac OS X (10.4, 10.5, 10.6, 10.7)
- Windows 7
- Windows Server 2003 R2, 2008 R2

**Ruby**

Ruby 1.8.7, 1.9.1 or 1.9.2 with SSL bindings is required.

**RubyGems**

Version 1.3.7 or greater. On Ubuntu and Debian Rubygems should be installed from source

**Git?**

# Parts I didn't cover

- Setting up and running chef-server / chef-client
- Shef
- Knife plugins
- Chef + capistrano
- All the rest Chef goodies

# Chef Alternatives

- Puppet
- Sprinkle
- Rubber
- Sunzi

# Great resources

https://github.com/opscode/cookbooks
http://wiki.opscode.com/display/chef/Home
http://railscasts.com/episodes/339-chef-solo-basics
https://github.com/ctshryock/capistrano-chef

# Thank You

tripsta®

travelplanet24.com®

τόσο απλά, τόσο φθηνά