# WHEN DOCKER ENDS CHEF BEGINS

Hello!

# I AM GIOVANNI TORALDO

Open Source enthusiast with SuperCow Powers
PHP/Java/whatever developer
writer of the OpenNebula book
Lead Developer @ ClouDesire

## WHAT IS CLOUDESIRE?

- Application Marketplace
  - Helps S/M software vendors
  - For simple applications it can
    - provision VM
    - on multiple cloud providers
    - monitor resources
  - For complex applications
    - expose REST API
  - For everyone
    - manage subscriptions, billing, pay-per-use, invoicing, payments

# 1 WOULD YOU BE MY CONTAINER?

## DOCKER: WHAT IS IT?

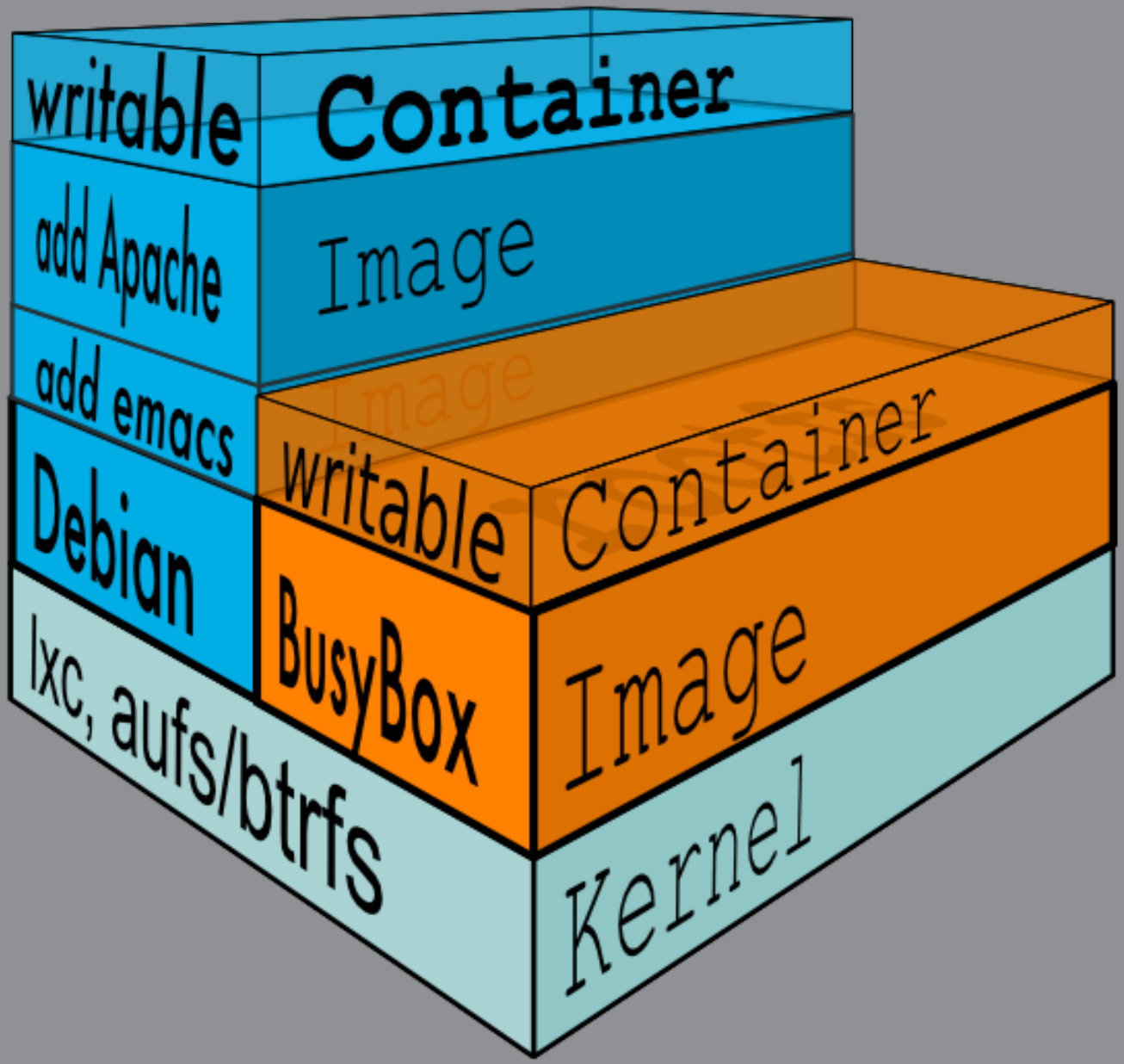
- Enables software developers to
  - package an application
  - with all dependencies
  - runs it everywhere unchanged

## DOCKER: WHAT IS THE POINT?

- Enables system administrators to
  - simplify application deployment
  - ease scale-up & scale-down
  - processes separation

## DOCKER: UNDERLYING TECHNOLOGIES

- *Linux namespaces*
- *Control Groups* (cgroups)
- Layered filesystems
- *LXC (now libcontainer)*


writable
Container
add Apache
Image
add emacs
writable
Container
Debian
BusyBox
Image
lxc, aufs/btrfs
Kernel

## DOCKER: GLOSSARY

- **Image**: immutable snapshot of a container, push/pull repository
- **Container**: an instance launched from an image
- **Volume**: persistent writable area of a container
- **Registry**: repository of images (versioned via **tags**)
- **Dockerfile**: the descriptor from which an image is built

## DOCKER: HOW DO I RUN IT?

- GNU/Linux

  wget -qO- https://get.docker.com/ | sh

- Windows

  https://github.com/boot2docker/windows-installer/releases/latest

- OSX

  https://kitematic.com/download/

- Hello world

*$ docker run -ti ubuntu:14.04 /bin/bash*

## DOCKER: WHAT HAPPENS UNDER THE HOOD?

- **Pulls** the ubuntu image from registry
- Creates a **new container**
  - Allocates a rw **filesystem**
  - Allocates a **network** interface (on a bridge)
  - Sets up network (**IP address**, dns..)
- **Launch a process** in the container
- Captures and provides **application output**

Container terminates when the process exit

# DOCKER: A SIMPLE DOCKERFILE

Dockerfile

Raw

```
FROM ubuntu:14.04
MAINTAINER Giovanni Toraldo

ENV DEBIAN_FRONTEND noninteractive

RUN sed -i s/archive/it.archive/g /etc/apt/sources.list
RUN apt-get update

RUN apt-get install -y software-properties-common
RUN apt-add-repository ppa:chris-lea/node.js
RUN apt-get update
RUN apt-get install -y nodejs

RUN apt-get clean
RUN rm -rf /var/lib/apt/lists/*

RUN mkdir /var/www

ADD app.js /var/www/app.js

ENV PORT 8080
EXPOSE 8080

VOLUME ["/var/www/data"]

CMD ["/usr/bin/node", "/var/www/app.js"]
```

## DOCKER: STANDARD WORKFLOW

Build & push:

- docker build -t gionn/nodejs-app:1.0.0 .
  - a tagged image is generated
- docker push gionn/nodejs-app:1.0.0
  - publish to repository

Pull & run:

- docker pull gionn/nodejs-app:1.0.0
  - fetch from repository
- docker run gionn/nodejs-app:1.0.0
  - run container from this image

Example gist: link

## DOCKER: ROUGH EDGE #1

Service in container A needs to talk to service in container B

Docker solution:

- Use Container Links

Reality:

- Works only on the same host
- Ordered sequence to boot-up
- Can't solve cyclic dependencies

## DOCKER: ROUGH EDGE #2

My containerized application needs environment-dependant configurations

Docker solution:

- Inject environment variables

Reality:

- I need to fill YAML, XML, JSON complex structures

## DOCKER: ROUGH EDGE #3

I need to manage and upgrade a non-trivial number of containers on multiple hosts

Docker solution:

- Docker Swarm

Reality:

- currently in beta, not recommend for production

## DOCKER: RECAP

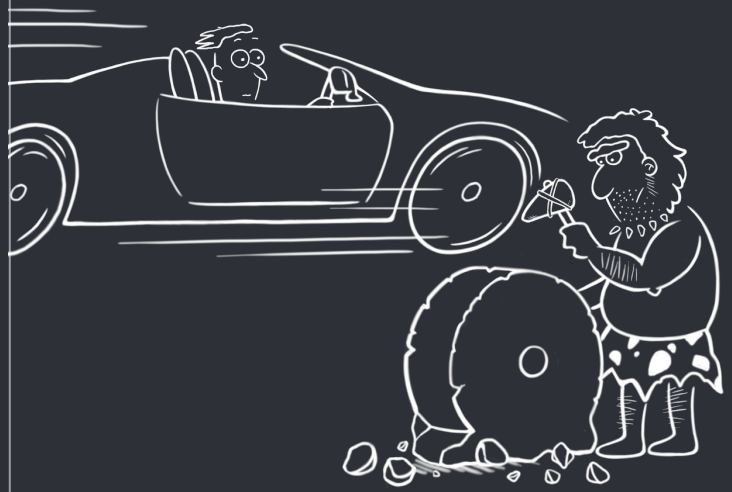
So far so good?

Docker is a piece of cake for wrapping together the technologies of Linux containers, multi-layered filesystems and an image build system, in an unique tool easy and fast to use.

## DOCKER: RECAP

But what about the environment?

Being a (relatively) young project, the ecosystem of tools is pretty scattered and inconsistent.



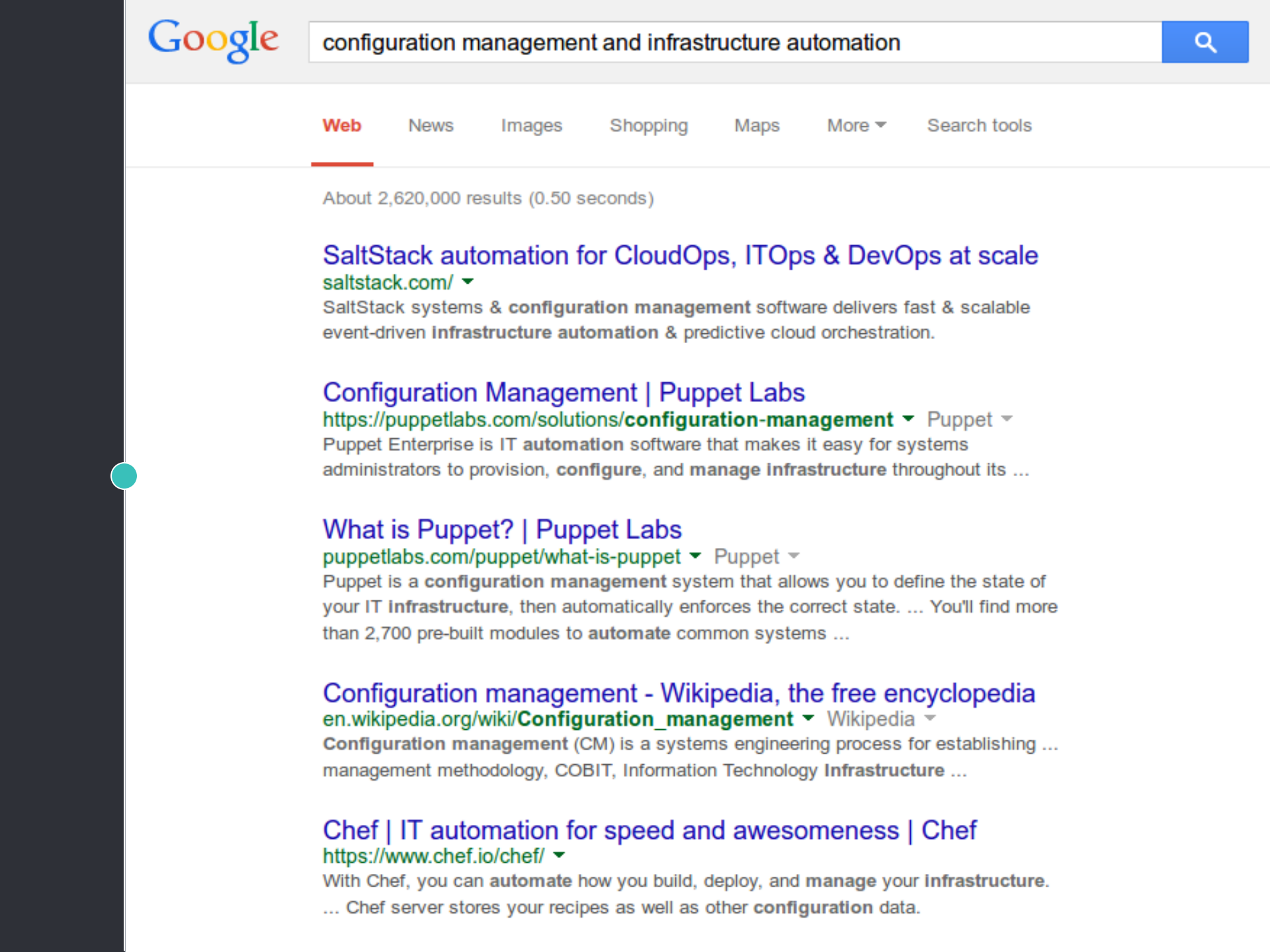
# 2 WHO YOU GONNA CALL?

## WHO YOU GONNA CALL?

Probably someone has solved this kind of issues far time ago, even before Docker existed?

Those kind of problems are all about **configuration management** and **automation**.

So use the tools already available.


Google
configuration management and infrastructure automation
Web
News
Images
Shopping
Maps
More
Search tools
About 2,620,000 results (0.50 seconds)
SaltStack automation for CloudOps, ITOps & DevOps at scale
saltstack.com/
SaltStack systems & configuration management software delivers fast & scalable event-driven infrastructure automation & predictive cloud orchestration.
Configuration Management | Puppet Labs
https://puppetlabs.com/solutions/configuration-management Puppet
Puppet Enterprise is IT automation software that makes it easy for systems administrators to provision, configure, and manage infrastructure throughout its ...
What is Puppet? | Puppet Labs
puppetlabs.com/puppet/what-is-puppet Puppet
Puppet is a configuration management system that allows you to define the state of your IT infrastructure, then automatically enforces the correct state. ... You'll find more than 2,700 pre-built modules to automate common systems ...
Configuration management - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/Configuration_management Wikipedia
Configuration management (CM) is a systems engineering process for establishing ... management methodology, COBIT, Information Technology Infrastructure ...
Chef | IT automation for speed and awesomeness | Chef
https://www.chef.io/chef/
With Chef, you can automate how you build, deploy, and manage your infrastructure. ... Chef server stores your recipes as well as other configuration data.

## CHEF

Chef enables you to:

- **Version** your **infrastructure** on SCM, build an artifact
- Apply testing, CI, CD to infrastructure
- Keep it aligned with your software
- Automation via **repeatable** actions (e.g. click to deploy)

## CHEF: THE TOOLS

Everything you need in a single package:
https://downloads.chef.io/chef-dk/

For (automated) testing
https://www.vagrantup.com
https://www.virtualbox.org

## CHEF: EVERYTHING IS IN A REPOSITORY

The chef-repo is a standard repo layout and contains:

- Cookbooks
- Environments
- Data bags
- Roles

## CHEF: WHAT IS A COOKBOOK

Each cookbook is coupled with a service (e.g. mysql).

Contains:

- **Attributes**: they are like global variables (e.g. version to install)
- **Recipes**: an atomic unit of configuration
- **Templates**: patterns to generate real files, filled with data
- **Files**: static configuration

## CHEF RECIPES

Each recipe contains **behaviour** expressed by **resources** (and Ruby code)

```
user_name = 'gionn'
user user_name do
  supports :manage_home => true
  uid 1000
  gid 'users'
  home "/home/#{user_name}"
  shell '/bin/bash'
  password '$1$JJsvHslV$szsCjVEroftprNn4JHtDi'
end
```

## CHEF COMPONENTS

- The remaining components:
  - **Environments**: contains common attributes for a group of nodes
  - **Roles**: contains attributes for nodes sharing a particular behaviour
  - **Data bags**: general-purpose JSON data, optionally encrypted, usually to store credentials

## COOKBOOKS FOR EVERY NEEDS

- All cookbooks are usually hosted on GitHub
  - Maintained by Opscode
    https://github.com/opscode-cookbooks
  - by vendors
    https://github.com/elastic/cookbook-elasticsearch
  - by the community
    https://supermarket.chef.io

Community Stats (07/04/2015)

2,120 Cookbooks ~ 62,086 Chefs

# 3 ONE DOES NOT SIMPLY COOK A WHALE

## DOCKER COOKBOOK

- https://github.com/bflad/chef-docker
  - Install docker daemon on supported platforms
    - Ubuntu/Debian
    - RHEL/CentOS/Fedora
    - Amazon Linux
  - Expose attributes for fine-tuning (e.g. TLS certificates, DNS)
  - Manage images & containers lifecycle via ad-hoc resources

This repository Search Explore Gist Blog Help gionn

# bflad / chef-docker

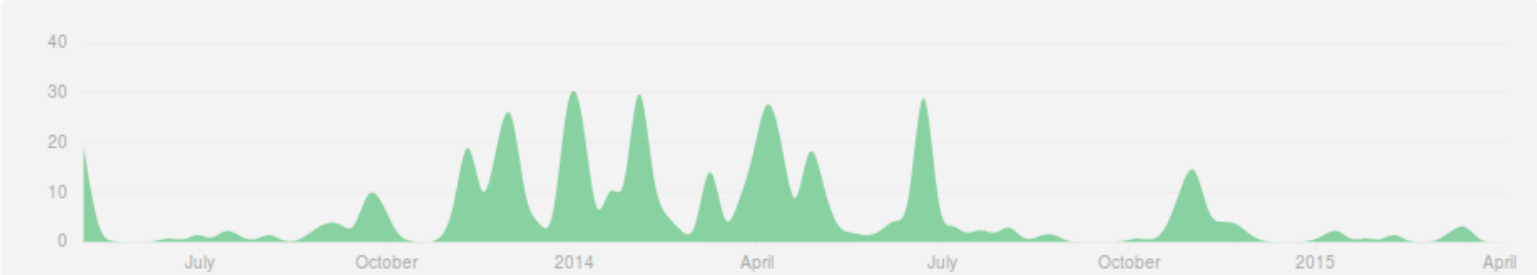
Watch 37 Unstar 363 Fork 167

Contributors | Commits | Code frequency | Punch card | Network | Members

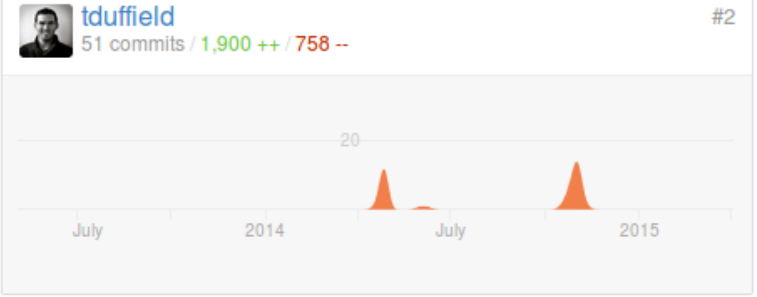
## May 5, 2013 – Apr 4, 2015

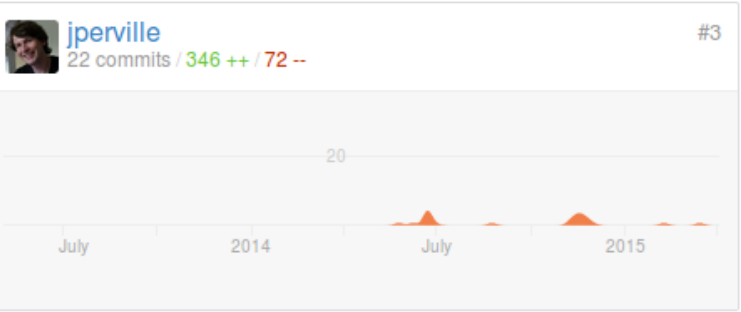
Contributions: Commits

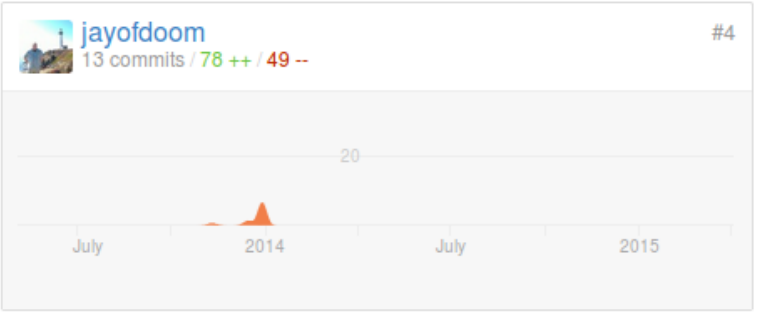
Contributions to master, excluding merge commits







# PROBLEM #0: IMAGE DISTRIBUTION / RUN CONTAINER

```
docker_image 'registry:0.9.0' do
  action :pull
  notifies :redeploy, 'docker_container[registry]', :immediately
end

docker_container 'registry' do
  container_name 'registry'
  image 'registry:0.9.0'
  detach true
  port '5000:5000'
  volume '/srv/registry:/tmp/registry'
  env 'SETTINGS_FLAVOR=local'
  env 'SEARCH_BACKEND=sqlalchemy'
  action :run
end
```

## PROBLEM #0: IMAGE DISTRIBUTION / RUN CONTAINER

- Check that docker has such tagged image and if not:
  - Download that version
  - Stop existing container (if any)
  - Run new container
  - Raise error if anything goes wrong

## PROBLEM #1: MY APPLICATION NEEDS CONFIGURATION

# Populate configuration files with proper values (and automatically restart on changes)

```
template node['tomcat']['host'] + '/cmw.xml' do
  source 'tomcat/tomcat-context.xml.erb'
  variables(
    :resource_name    => 'jdbc/datasource',
    :username         => node['cloudesire']['name'],
    :password         => node['cloudesire']['pass']
    :url              => node['cloudesire']['url'],
  )
  notifies :redeploy, 'docker_container[cmw]'
end
```

# PROBLEM #1: MY APPLICATION NEEDS CONFIGURATION

**tomcat-context.xml.erb**

```
<Context>
  <Resource auth="Container"
    driverClassName="org.postgresql.Driver"
    initialSize="<%= @node['cloudesire']['backend']['db_pool']['min'] %>"
    maxActive="<%= @node['cloudesire']['backend']['db_pool']['max'] %>"
    maxIdle="<%= @node['cloudesire']['backend']['db_pool']['idle'] %>"
    maxWait="<%= @node['cloudesire']['backend']['db_pool']['wait'] %>"
    name="<%= @resource_name %>"
    url="<%= @url %>"
    username="<%= @username %>"
    password="<%= @password %>"
    type="javax.sql.DataSource"
    validationQuery="select 1"
    testOnBorrow="true"
  />
</Context>
```

## PROBLEM #1: MY APPLICATION NEEDS CONFIGURATION

Templates consist of:

- an .ERB template
- a template resource declared in a recipe

The template is evaluated using the variables passed directly or via the global node object.

PROBLEM #1: MY APPLICATION NEEDS CONFIGURATION

# Inject a single file or entire folders

```
dst = node['tomcat']['base'] + '/conf/Catalina/' +
'localhost/cmw.xml'

docker_container 'cmw' do
  image image_name
  container_name 'cmw'
  detach true
  env LOG debug
  volume [
    "#{node['tomcat']['host']}/cmw.xml:#{dst}",
    "/etc/cloudesire:/etc/cloudesire"
  ]
end
```

## PROBLEM #1: MY APPLICATION NEEDS CONFIGURATION

Docker permits defining **volumes** to be used for persistent data (e.g. database files), but may be used to inject configurations into the container at runtime.

Definitevely **avoid** the needs of image rebuilding to adjust a setting.

## PROBLEM #2: CONTAINERS RUNNING ON MULTIPLE HOSTS

Each node has its own **run_list**, defining which recipes should be executed (in JSON):

```
{
  “run_list”: [
    "cd-infrastructure::docker-cmw",
    "cd-infrastructure::docker-deployer",
    "cd-infrastructure::docker-monitor",
    "cd-infrastructure::docker-logger"
  ],
  “cloudesire”: {
    “key”: “value”
  }
}
```

## PROBLEM #2: CONTAINERS RUNNING ON MULTIPLE HOSTS

# Same recipe on different nodes (attributes may change)

**node1.json**

```
{
  “run_list”: [
    "cd-infrastructure::docker-cmw",
    "cd-infrastructure::docker-logger"
  ]
}
```

**node2.json**

```
{
  “run_list”: [
    "cd-infrastructure::docker-deployer",
    "cd-infrastructure::docker-monitor",
    "cd-infrastructure::docker-logger"
  ]
}
```

## MAY NOT BE GOLD BUT IT'S A START FOR SURE!

- It's easy to getting started with Chef by using kitchen-ci or plain Vagrant:
  - Initialize a chef repo
  - Create a new cookbook
  - Start hacking
  - Play on kitchen-ci or vagrant
  - Repeat last 2

KITCHEN.YML EXAMPLE

# $ kitchen converge

```
driver:
  name: vagrant

provisioner:
  name: chef_solo

platforms:
  - name: ubuntu-1404

suites:
  - name: default
    run_list:
      - recipe[mycookbook::docker-whatever]
    attributes: { foo: "bar" }
```

## READY TO USE CHEF REPOSITORY

A starting repository for aspiring whale cookers:

https://github.com/gionn/cooking-docker

## DOCKER APPENDIX: GOLDEN RULES

- Only one process per Image
- No embedded configuration
- No, you don't need SSH
- No, you don't need syslog
- No, you won't touch a running container to adjust a thing
- No, you will not use a community-contributed image without looking at what it do

Thanks!

# ANY QUESTIONS?

You can find me at:

@gionn on twitter / github

me@gionn.net

giovanni.toraldo@cloudesire.com