



EXITING VACUUM

INTEGRATING CONFIGURATION MANAGEMENT

Sascha Bates

Opscode

INTERNATIONAL
SOFTWARE DEVELOPMENT
CONFERENCE

gotocon.com



sascha bates

blog.brattyredhead.com

Twin Cities Infracoders Meetup

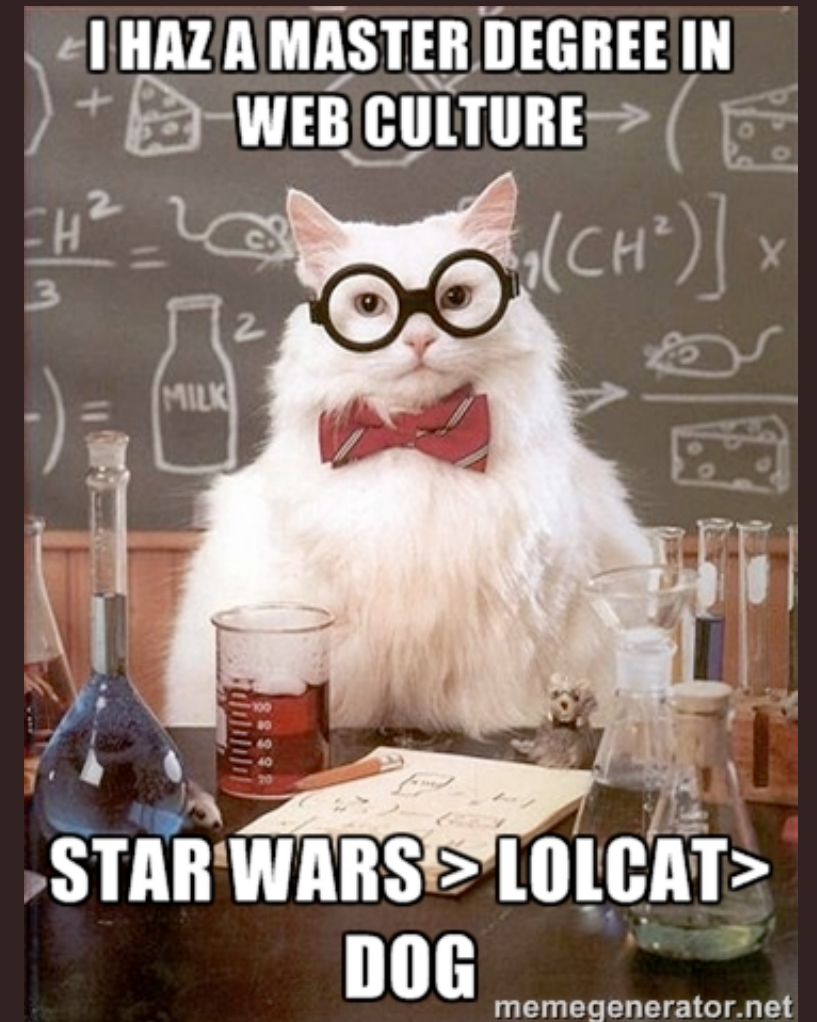
@sascha_d

The Ship Show Podcast

credentials?

In love with CM since 2010

Curating developer happiness
even longer





OPSCoDETM
CODE CAN



A tool is just
a tool

it's what
you do
with it that
matters



WHY AM I HERE?





I mess things up
so you don't have to

Wasn't it awesome when
it took 3-6 weeks to get a
dev server and you got
to share it with 60 other
people?

- nobody ever

This Never Happens



wrong database connection string deployed to prod

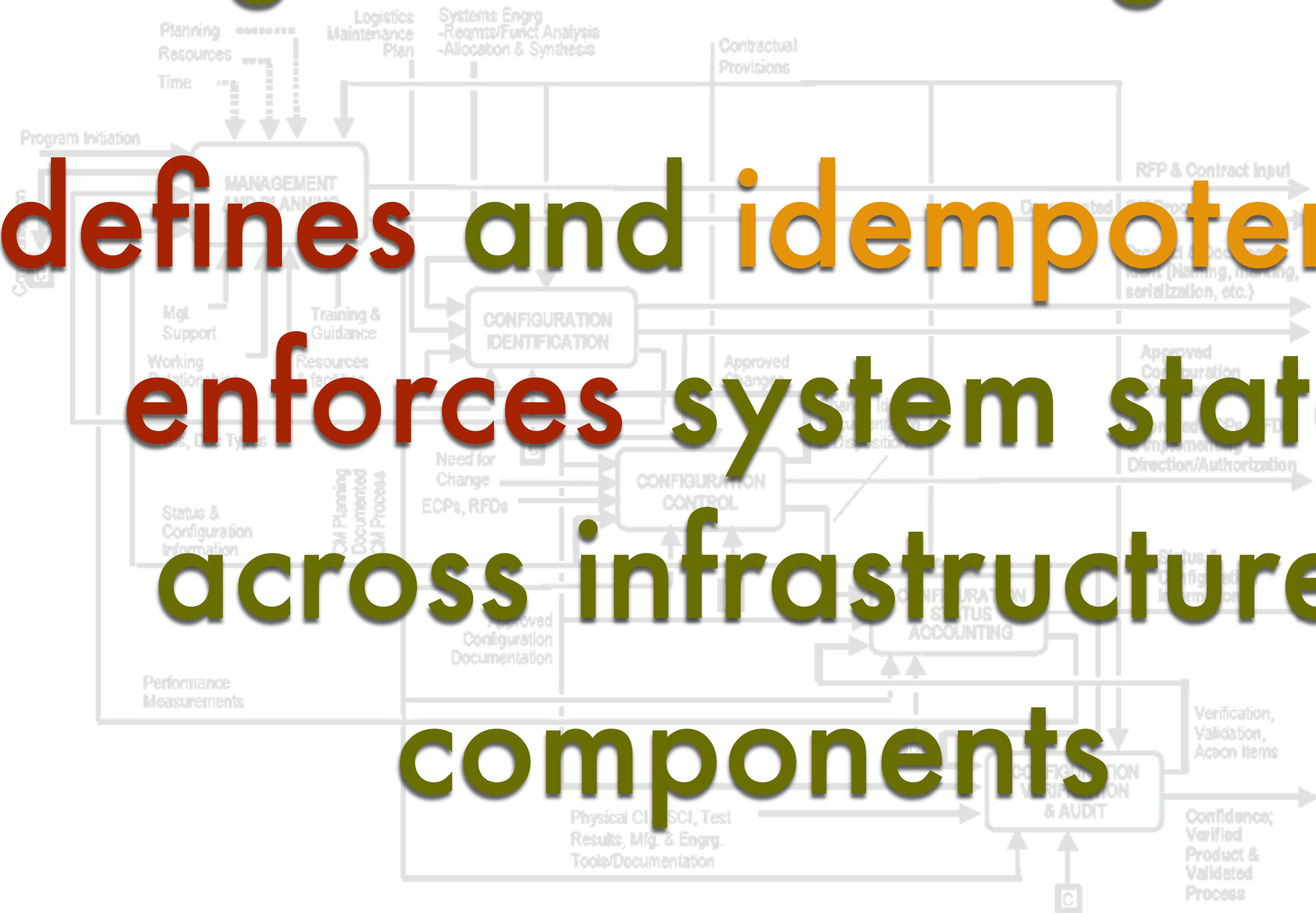
smtp server fixed by hand and forgotten

test apache server with special configs

ssh keys pushed by hand

configuration management

defines and idempotently
enforces system state
across infrastructure
components



freedom



not bondage

confidence



Configuration Management is NOT

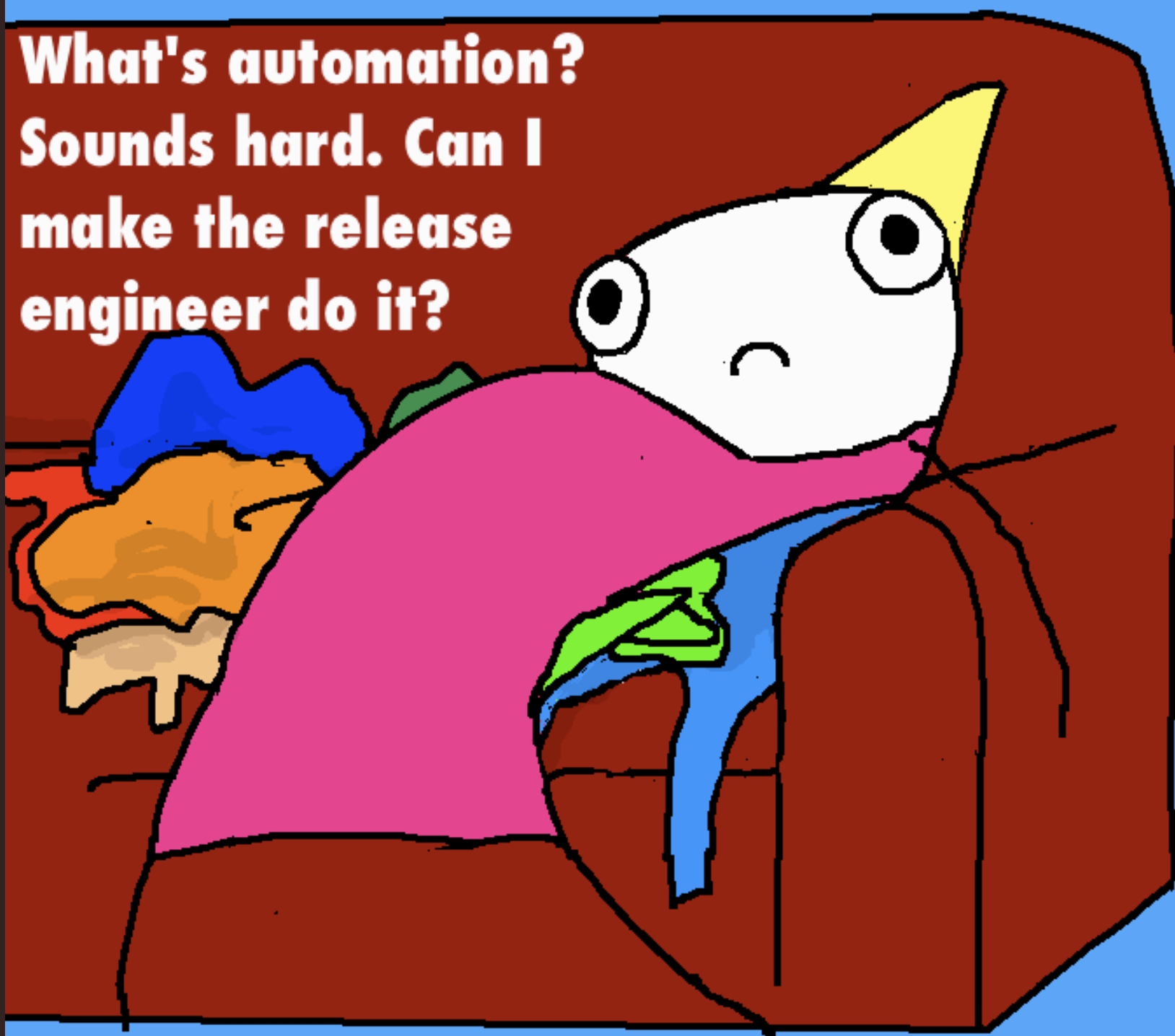


a magic rainbow pixie dusted unicorn
coming to save you



where do you stand?

**What's automation?
Sounds hard. Can I
make the release
engineer do it?**

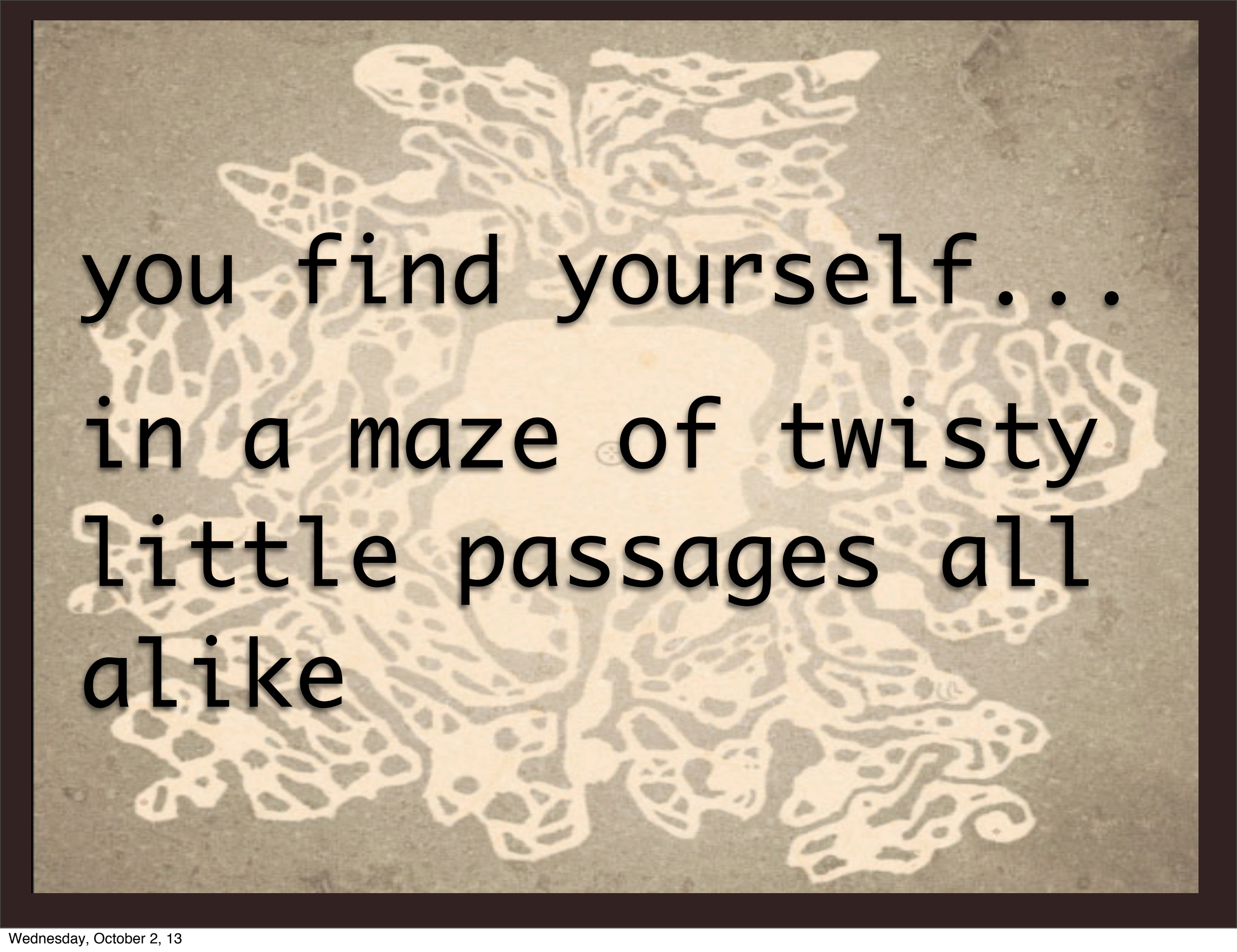


**if I'm really
quiet**



**The
automation
will NEVER
FIND ME!**





you find yourself...
in a maze of twisty
little passages all
alike

Map the Journey

Infrastructure

Greenfielding and Brownfielding

A Balanced Ecosystem

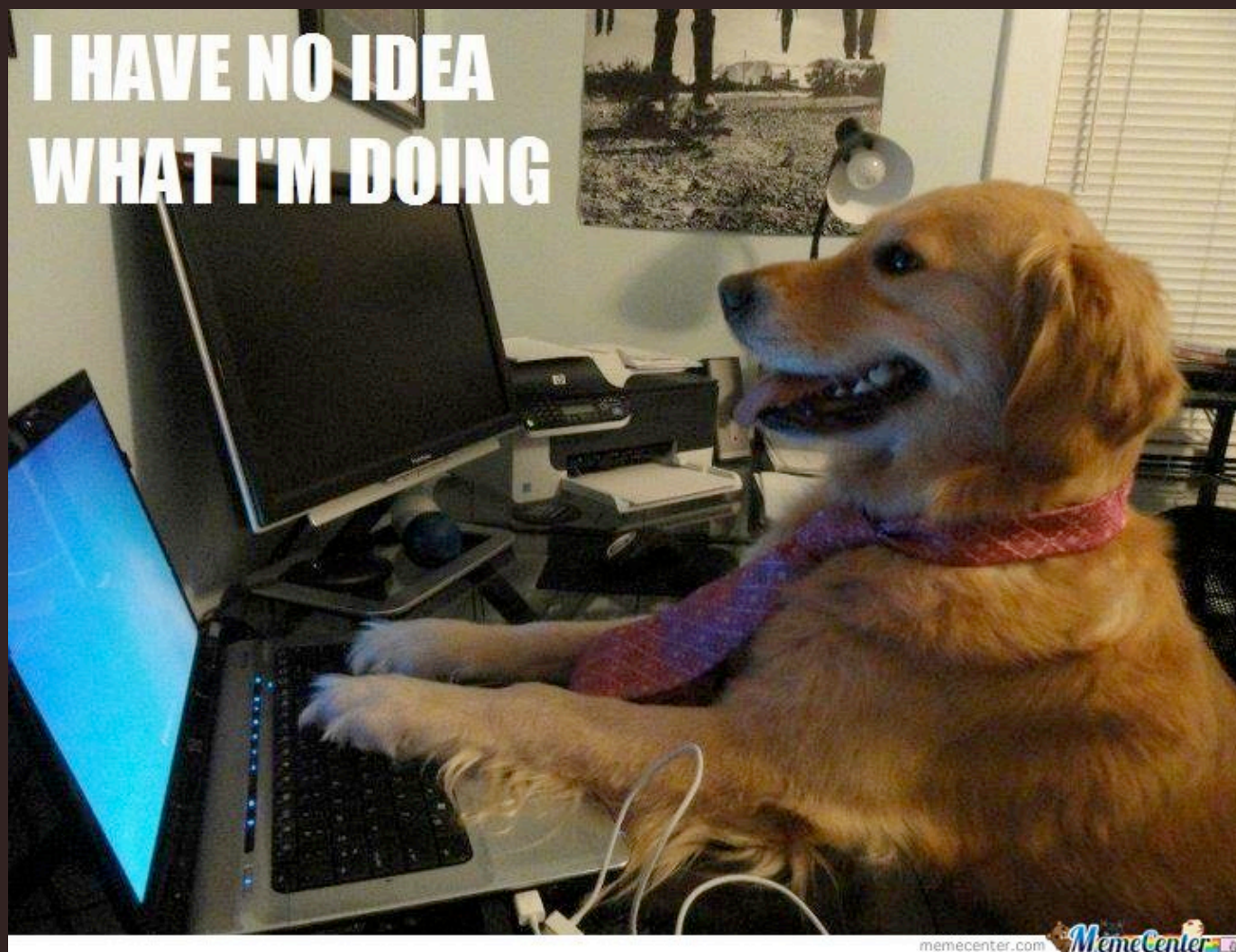
Practical CM

Infrastructure who cares?



In a perfect universe

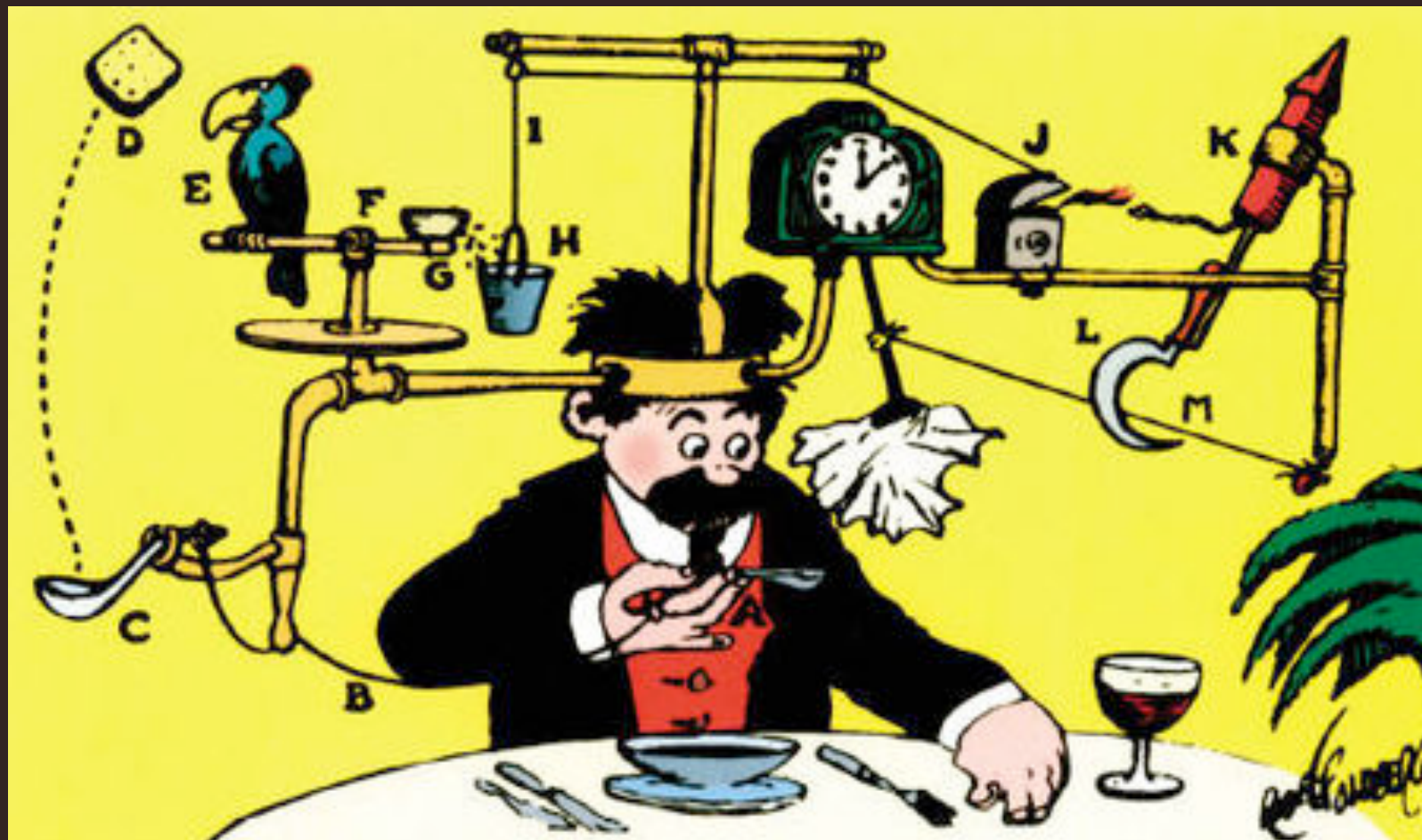
Provision Identically



the brains behind your servers

In a perfect universe

One deployment process to rule them all



because deployments are complicated enough

In a perfect universe

Repositories for all OS packages

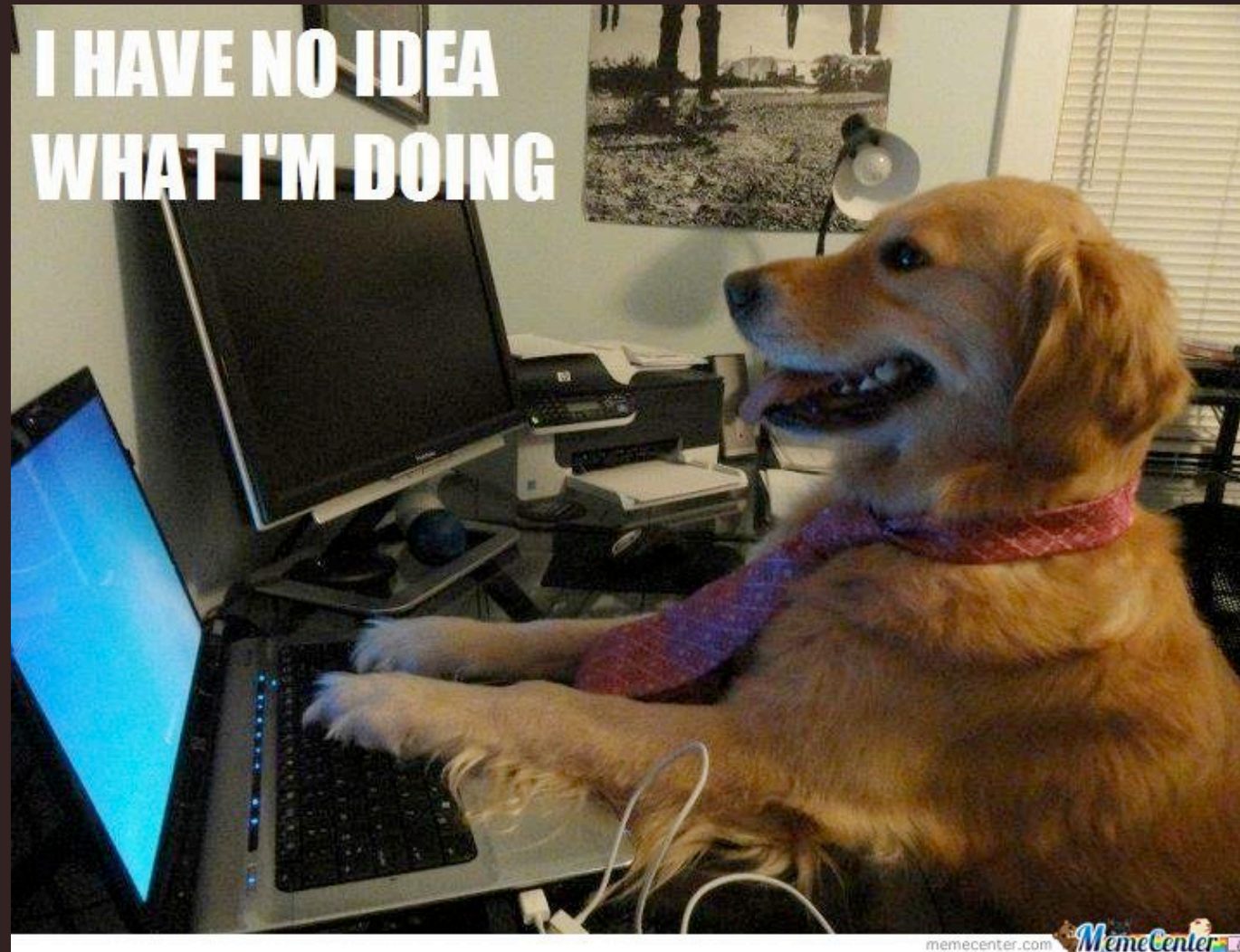
```
yum install tomcat
```

trumps

```
curl -o http://some-tomcat-url  
tar -xvf tomcat.gz
```

In a perfect universe

Hands-off the servers!



this guy again

Getting Started

making mud pies



Infrastructure Crafting

Server Provisioning



Infrastructure Crafting

App Layer Configuration

keep configuration data separate from code

different configs in different environments

deployments controlled by different teams

Infrastructure Crafting

Dynamic Discovery Across Tiers

application instances
noticed by
web instances
noticed by
load balancer configs



Infrastructure Crafting

Workstation Automation



make onboarding a fast happy process

eliminate stale epic-length wiki pages

Infrastructure Crafting

Superior Local Testing



vagrant

virtualbox/ec2/vmware

chef/puppet/ansible

Infrastructure Crafting

Beef Up Your Pipeline

Jenkins + Configuration Management = power

bootstrap/deploy

automated integration/functional testing ftw

Getting Started

don't do this



Pick a Sane Use Case

don't try to automate the world

small
achievable
measurable
impactful

Pick a Sane Use Case

stay agile and visible



demo your impactful automation
show time/frustration saved

Keep an Open Mind

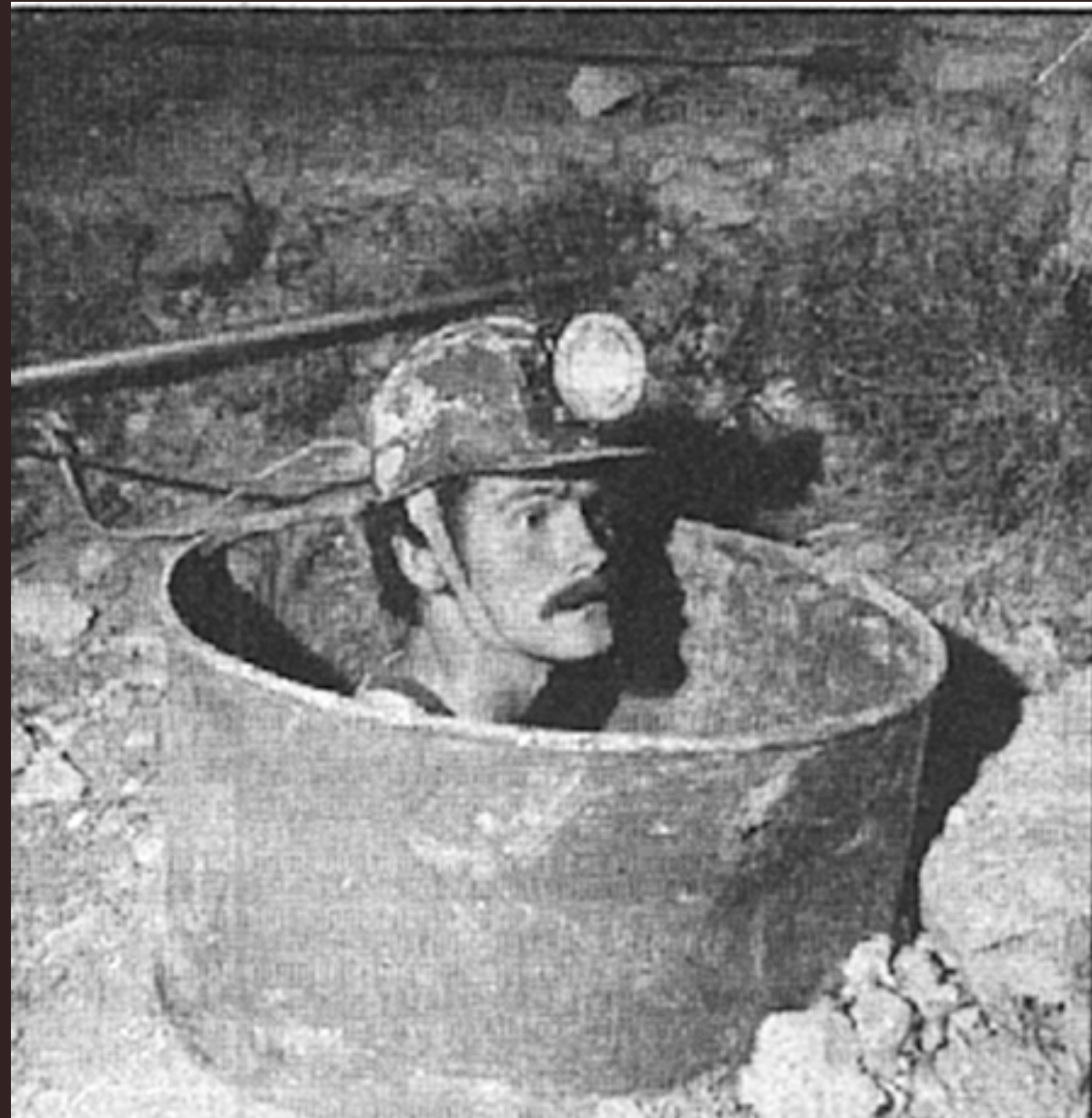


“because we’ve always done it that way” is no longer acceptable

Refactoring Happens



Brownfielding



your biggest challenge is people

Brownfielding

coloring inside the lines



Brownfielding

collaborating



legacy apps have possessive owners
be inclusive, ask questions
listen when they tell you what will work
mute criticism

A Balanced Ecosystem

automation can't live in a vacuum



Configuration Management is not

a package manager

a package repository

a substitute for version control

A Balanced Ecosystem

Package Repos



insert
package
repository
rant
here

A Balanced Ecosystem

Version Control

configuration management
code is

CODE

put it where it belongs

A Balanced Ecosystem

Build Tools



A Balanced Ecosystem

Virtualization



Practical CM testing

you can write tests for CM

unit testing w/**rspec**

functional/integration testing with **minitest/bats**

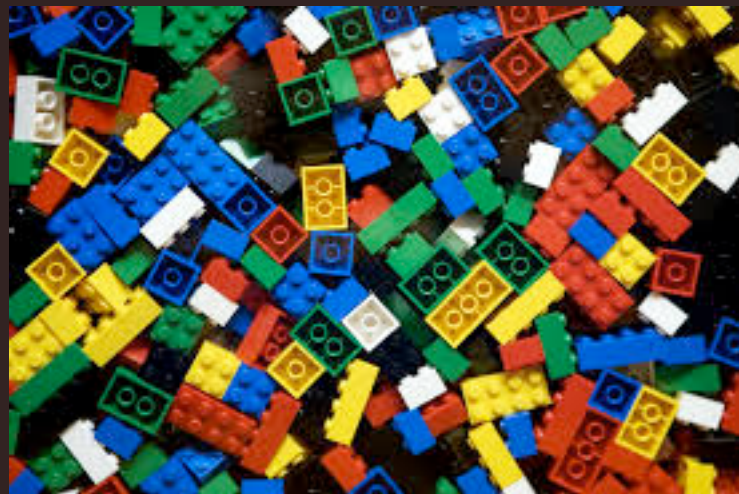
Practical CM dependency resolution

Librarian for both Puppet and Chef

Berkshelf for Chef

There could be others

Practical CM primitives



Practical CM primitives

file, user, package, template, directory

built-in idempotence

readability

operating system cross-functionality

Practical CM exec blocks



Practical CM

exec vs primitives

```
bash 'install_my_package' do  
  command "yum -y install my_package"  
end
```

NEVER DO THIS

Practical CM

exec vs primitives

```
package 'apache' do  
  action :install  
end
```

ALWAYS DO THIS


```
bash "install_tomcat6" do
  tomcat_version_name = "apache-tomcat-#{node.tomcat.version}"
  tomcat_version_name_tgz = "#{tomcat_version_name}.tar.gz"
  user "root"
  code <<-EOH
    curl --proxy https://aproxy.com:8080/ --user user:pass
https://myartifactoryurl.com/artifactory/ext-release-local/
apache-tomcat/apache-tomcat/#{node.tomcat.version}/
#{tomcat_version_name_tgz} -o /tmp/#{tomcat_version_name_tgz}
    tar -zxvf /tmp/#{tomcat_version_name_tgz} -C /tmp
    rm /tmp/#{tomcat_version_name_tgz}
    mv /tmp/#{tomcat_version_name} #{node.tomcat.install_path}
    chown -R #{node.tomcat.run_user}:#{node.tomcat.run_group}
#{node.tomcat.install_path}
    chmod -R 755 #{node.tomcat.install_path}
    rm -rf #{node.tomcat.install_path}/webapps/ROOT
    EOH
end
```

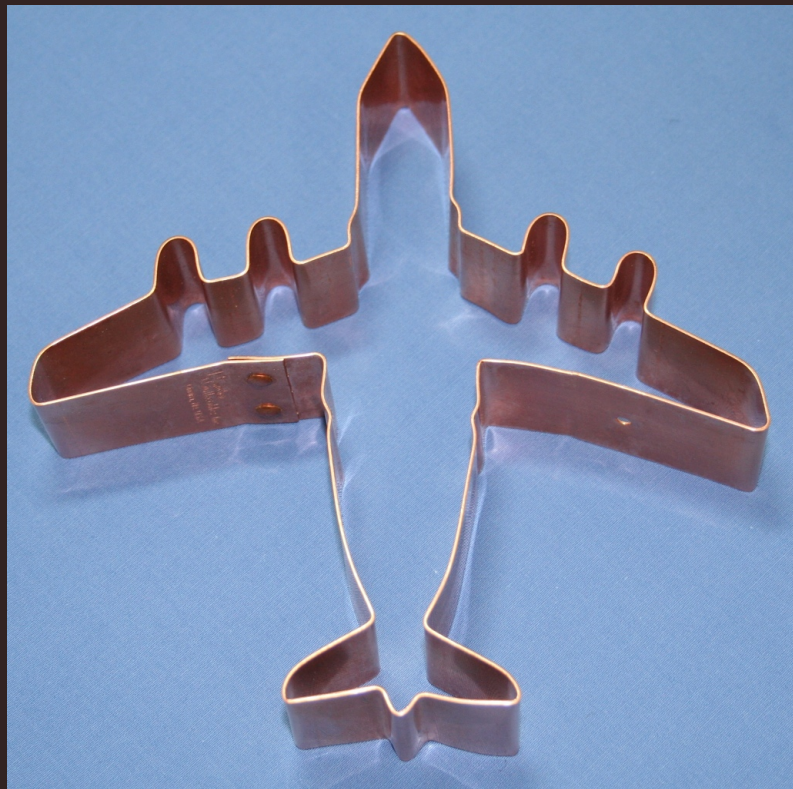
wtf was that?!



```
package 'tomcat7' do  
  action :install  
end
```

Practical CM

template primitive



templates allow you to **write**
flat files with **varied** configs
across **different** environments


```
<% @sudoers_users.each do |user| -%>
<%= user %>    ALL=(ALL) <%= "NOPASSWD:" if
@passwordless %>ALL
<% end -%>

# Members of the sysadmin group may gain root
privileges
%sysadmin      ALL=(ALL) <%= "NOPASSWD:" if
@passwordless %>ALL
```

```
bash "update_ssh" do
  code <<-E0H
  sed -i -e 's/
AuthorizedKeysFile.*authorized_keys/
AuthorizedKeysFile \\\/.keys\\/%u\\//
authorized_keys/g' /etc/ssh/sshd_config
E0H
end
```

```
bash "ssh_dns" do
  code <<-E0H
  sed -i -e 's/#UseDNS.yes/UseDNS
no/g' /etc/ssh/sshd_config
E0H
end
```

primitives trump execs

```
package "ssh" do
  action :install
end

service "sshd" do
  action [:enable, :start]
end

template "/etc/ssh/sshd_config" do
  action :create
  mode 0644
  notifies :restart, "service[sshd]"
end
```

Practical CM

extending and abstracting

CM tools are easy to extend
understand repeatable patterns
abstract them into libraries, resources, custom types
keep front line code readable


```

# Cookbook Name:: keys
# Recipe:: common
# Author:: Sascha Bates

keys = []
search('public_keys','tags:common').each { |k| keys << k }
search('public_keys','tags:chef AND tags:#{node.env}').each { |
k| keys << k }

keys.each do |k|
  key_type, key_part, key_comment = k['pub_key'].split(' ')
  ruby_block "root_keys_#{k['id']}" do
    Chef::Log.debug("test condition: grep #{key_part}
#{keyfile}")
    not_if "grep #{key_part} #{keyfile}"
    block do
      File::open(keyfile, 'a') do |f|
        Chef::Log.debug("Adding #{key_comment} to
#{f.path}")
        f << k["pub_key"] << "\n"
      end
    end
  end
end

```

dsl trumps code

```
# Cookbook Name:: keys  
# Recipe:: common  
# Author:: Sascha Bates
```

```
authkey "common_key" do  
  action :add  
  user "root"  
end
```

If you don't remember
anything else

start small, stay visible, communicate

craft a holistic ecosystem

use the tool wisely and well

bonus slide

```
# -*- mode: ruby -*-  
# vi: set ft=ruby  
Vagrant.configure("2") do |config|  
  config.vm.hostname = "goto-example"  
  config.vm.box = "opscode_centos-6.4_provisionerless"  
  config.vm.network :private_network, ip: "33.33.33.10"  
  config.vm.network "forwarded_port", guest: 8080, host: 8080,  
    auto_correct: true  
  config.omnibus.chef_version = :latest  
  config.ssh.max_tries = 40  
  config.ssh.timeout = 120  
  config.berkshelf.enabled = true  
  
  config.vm.provision :chef_solo do |chef|  
    chef.log_level = :debug  
    chef.run_list = [  
      "recipe[goto::default]"  
    ]  
  end  
end
```