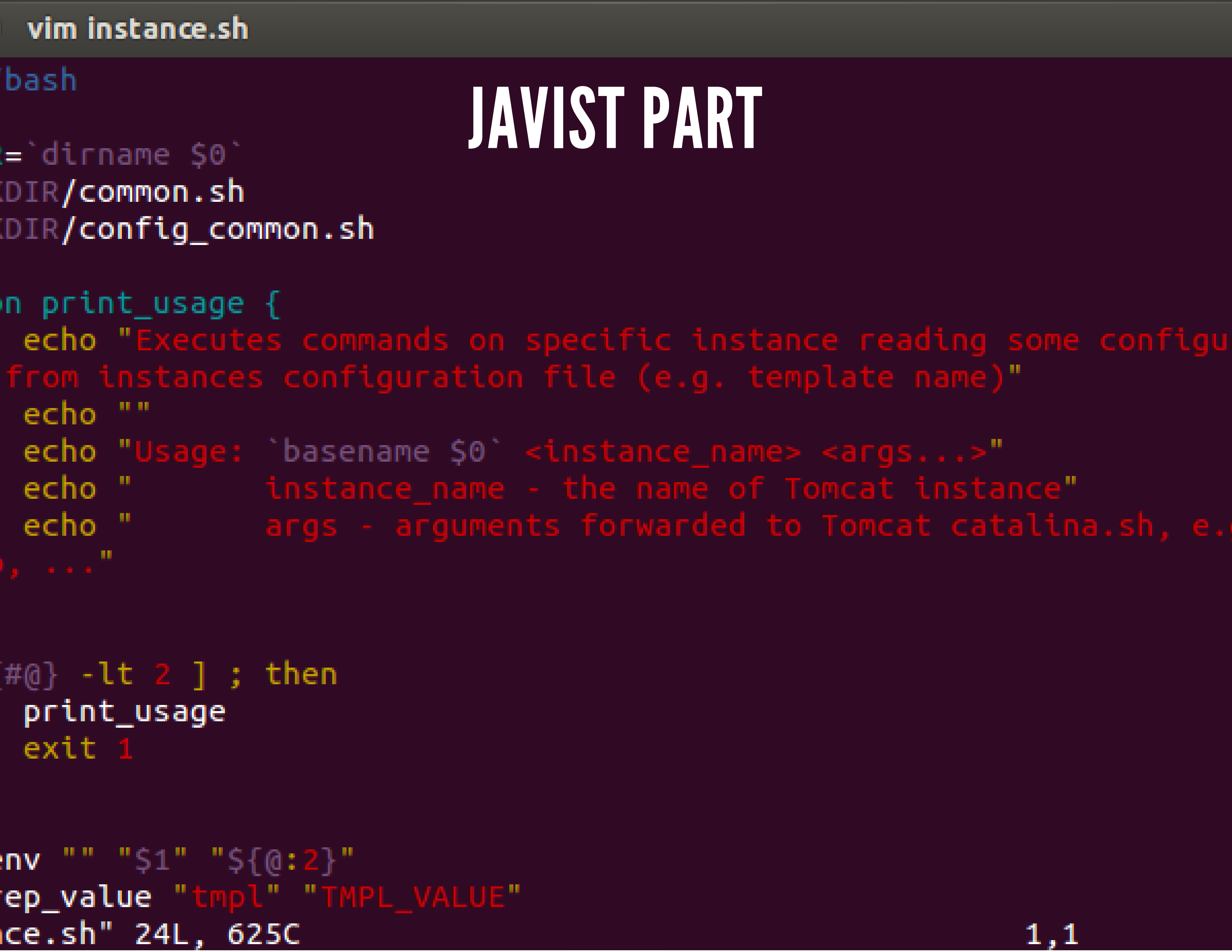
SEGES
space for solutions

JAVIST BEING CHEFTAIN

@ladislavGazo

gazo@seges.sk



vim instance.sh

JAVIST PART

```
/bash

R=`dirname $0`
CDIR/common.sh
CDIR/config_common.sh

on print_usage {
  echo "Executes commands on specific instance reading some configu
 from instances configuration file (e.g. template name)"
  echo ""
  echo "Usage: `basename $0` <instance_name> <args...>"
  echo "        instance_name - the name of Tomcat instance"
  echo "        args - arguments forwarded to Tomcat catalina.sh, e.
p, ..."


{#@} -lt 2 ] ; then
  print_usage
  exit 1


env "" "$1" "${@:2}"
rep_value "tmpl" "TMPL_VALUE"
nce.sh" 24L, 625C                                          1,1
```

CHANGELOG

```
 # change host records
    vim /etc/hosts
    vim /etc/hostname

    # optionally reboot
    reboot

    # to refresh repo list
    sudo apt-get update
    # to install updates
    sudo apt-get upgrade

    sudo apt-get install tmux mc
http://www.andrewault.net/2010/05/17/securing-an-ubuntu-server/

    sudo sysctl -w net.ipv4.conf.all.accept_source_route=0
    sudo sysctl -w net.ipv4.conf.default.accept_source_route=0

    sudo aptitude -y install denyhosts
    sudo aptitude -y install tiger
    sudo aptitude -y install psad
    sudo chkrootkit
```

... BUT WHAT IF

- there are more servers
  - change log for every one
- I want to setup development environment
  - manually go through the changelog
  - and probably do mistakes
- the changelog has some blank places
  - usually it is not that detailed

Holy crap... now what?

Chef

COMPARISON?

Only subjective:

Chef feels more community friendly
&
I like community projects

ALTERNATIVES

Of course there are many:

- Puppet
- CFEngine
- Capistrano
- Fabric
- glu
- ...

There is a difference between infrastructure management and deployment management.

Chef Solo vs. Chef Server

Knife

CHEF

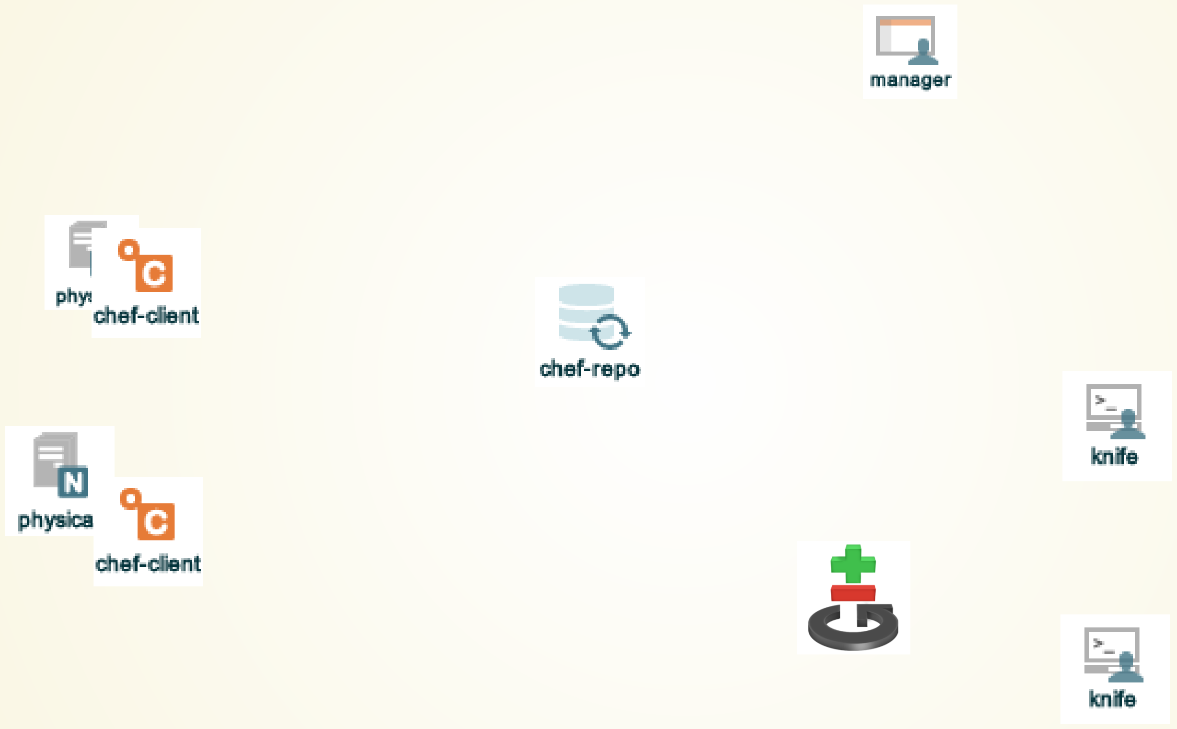
Solo

- for local/one node
- no central repository of configuration

Server

- installed internally or bought from Opscode
- central repository of cookbooks
- easy to install on supported OS
  - Ubuntu

COMPONENTS


manager
phys
chef-client
chef-repo
knife
physica
chef-client
knife

COMPONENTS II.

- attributes
- run-lists
- roles
- data bags
- environments

THE ESSENCE

COOKBOOKS:

Attributes
Recipes
Templates
Files

Metadata + Version + Dependencies

COOKBOOKS

- managed by Knife
- source code stored in Git repository
- uploaded to the Chef Server
- downloaded by Chef Client on a particular Node

HOW TO START COOKING?

You **do not** need to know Ruby up-front

=

wheew for Javist ;)

but

http://docs.opscode.com/just_enough_ruby_for_chef.html

NEXT

Install Chef Workstation if it does not exist

Set up Git repo (BB | GitHub | ...)
git clone git://github.com/opscode/chef-repo.git

Follow the guide precisely

Note: I keep separate user for development and separate for deployment

WRITE FIRST COOKBOOK

It is simple

```
knife cookbook create hyperic
```

```
cd hyperic
vim recipes/default.rb
```

you get along with 90% of what is already there for most recipes
you are fine with basic programming techniques

```
kitchen init
```

testing is important

TESTING

```
gem install test-kitchen --pre
gem install berkshelf<br>
gem install kitchen-vagrant<br>
```

SSH to running Vagrant machine when test-kitchen is executed

```
/chef-repo/cookbooks/hyperic/.kitchen/kitchen-vagrant/default-ubuntu-1204$ va
```

When everything is ready, "fire in the hole"

```
vim .kitchen.yml
kitchen test
```

CACHE

http://fgrehm.viewdocs.io/vagrant-cachier

It saves time!

```
vagrant plugin install vagrant-cachier
```

But does not work with current Kitchen version without hacking configuration file

NOTABLE RESOURCES

http://docs.opscode.com/resource.html

```
directory "/tmp/folder" do
    owner "root"
    group "root"
    mode 0755
    action :create
end
```

```
user "hyperic" do
    supports :manage_home => true
    home "/home/#{hyperic_user}"
    shell "/bin/bash"
    action :create
end
```

```
remote_file "hyperic_bundle" do
    path hyperic_src
    owner hyperic_user
    source node['hyperic']['agent']['bundle_url']
    mode 00644
end
```

OTHER

```
template "/etc/init.d/#{service_name}" do
    action :create_if_missing
    owner "root"
    mode 00700
    source "hyperic-agent.erb"
    variables(
    :service_name => service_name,
    :agentdir => hyperic_agentdir,
    :user => hyperic_user,
    :java_home => java_home
    )
end
```

```
service service_name do  pattern "agent-#{hyperic_version}"
    action [ :enable, :start ]
end
```

```
bash "extract_tcc" do  cwd ::File.dirname(tcc_down_path)
    code <<-EOH
    chown -R #{node.tcc.user}:#{node.tcc.group} #{node.tcc.location}
    EOH
end
```

NOTABLE HINTS

- **(re)create, not update**
  - rather don’t update file, find a way how to create it at once
- **more detailed steps then you expected**
  - if you thought there are 10 steps what to execute in such an automated way in your head, there are 60 at least
- **cookbook wrapper pattern**
  - describe the possibility to override resources in the “cookbook wrapper”

QUESTIONS?

@ladislavGazo
gazo@seges.sk

THANK YOU.... FOR...

ATTENTION