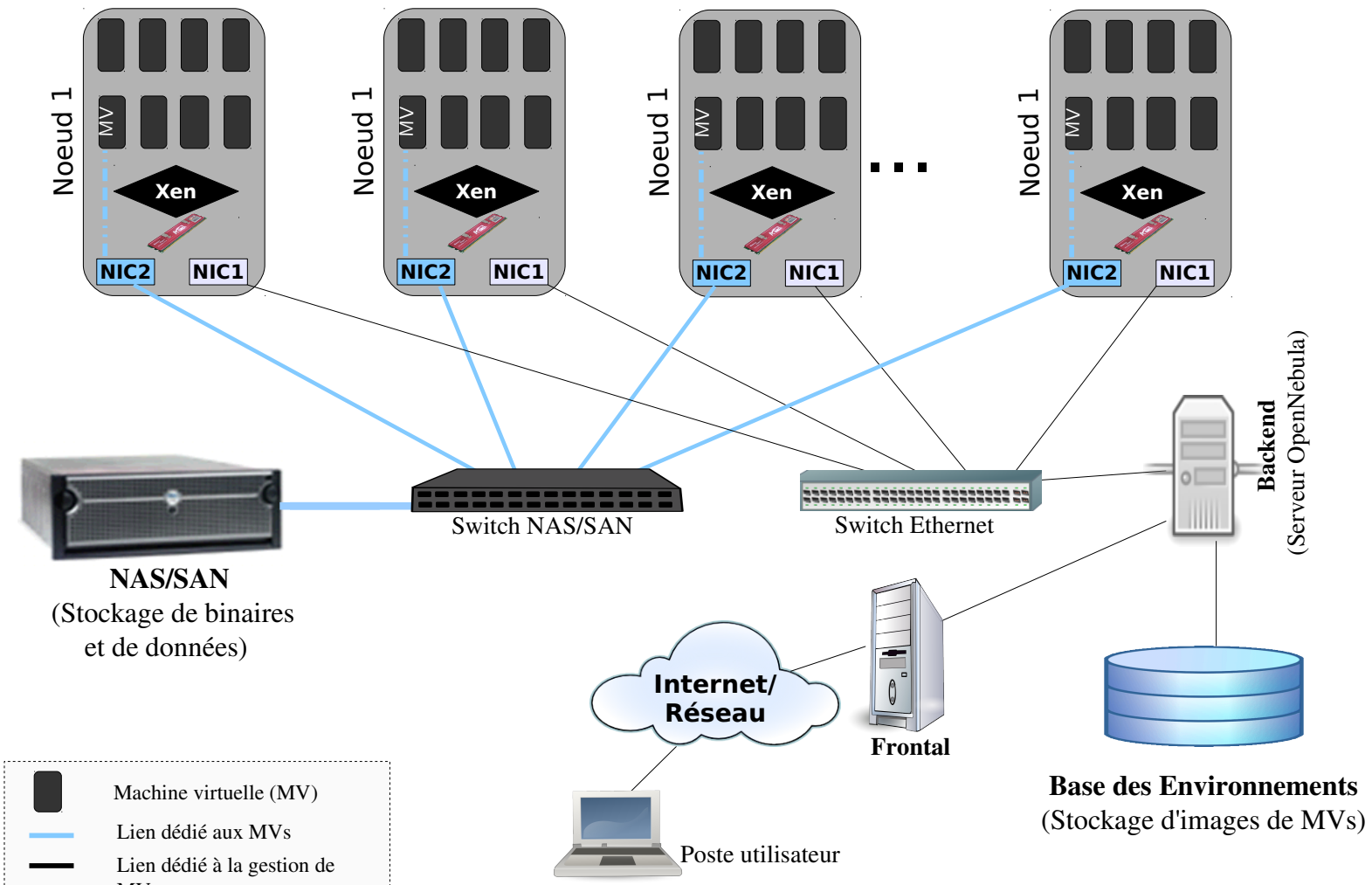
Constructing Software Environments

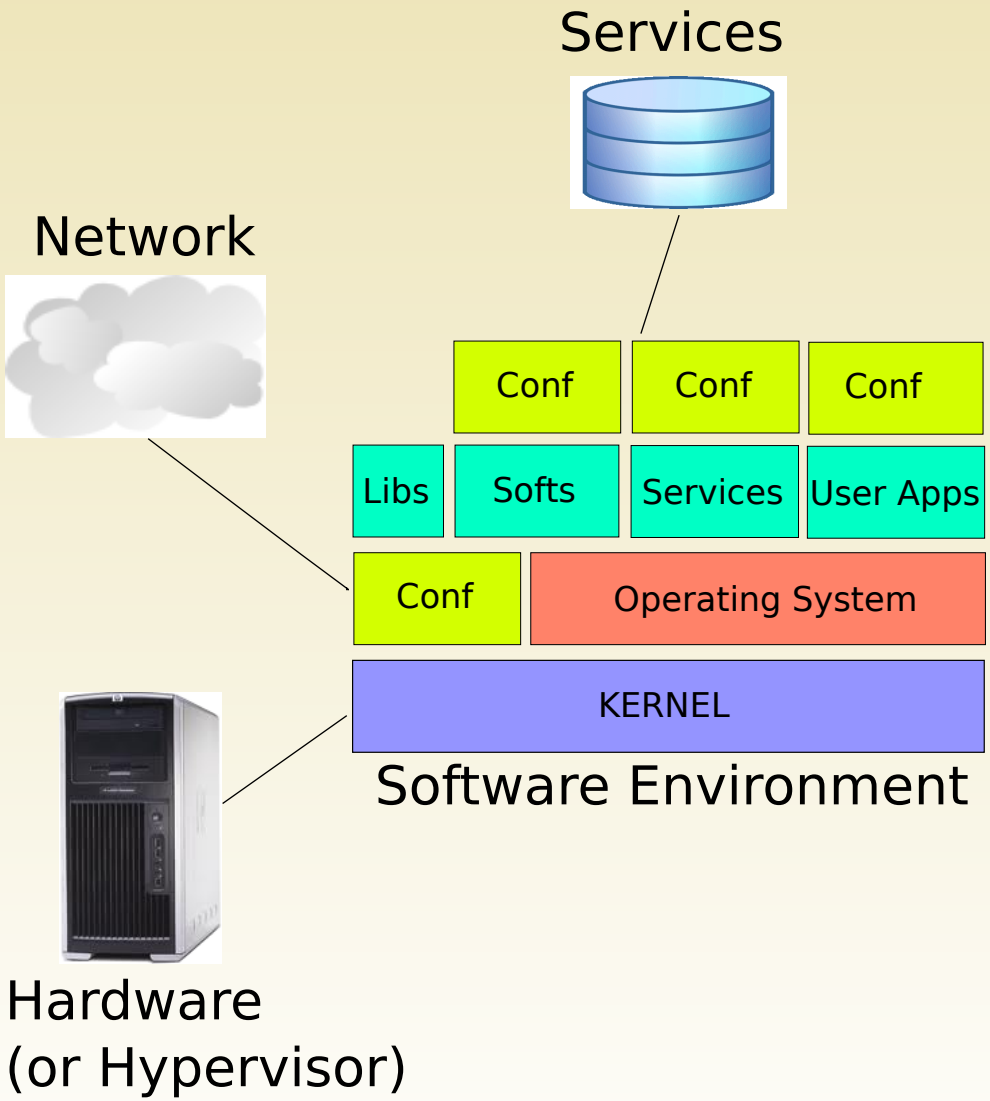
Production and User Environments in Grid'5000

Joseph Emeras Olivier Richard

Virtual Machines Environments in Clouds (and Grids)



Software Environment Images



Outline

Outline

Environment Images

Why?

- Portability on heterogeneous platforms
- Security
- Experiments reproducibility

Environment Images

Why?

- Portability on heterogeneous platforms
- Security
- Experiments reproducibility

How?

Different approaches:

- construct from scratch
- converge from existing

Environment Images

Why?

- Portability on heterogeneous platforms
- Security
- Experiments reproducibility

How?

Different approaches:

- construct from scratch
- converge from existing

What?

- Softwares
- Configurations
- Files
- Versions, Data Provenance?

Environment Images

Virtual Machines but also physical machines.

The Grid'5000 case:

- Production environments (compute nodes)
- User environments
- Service Machines environments

Outline

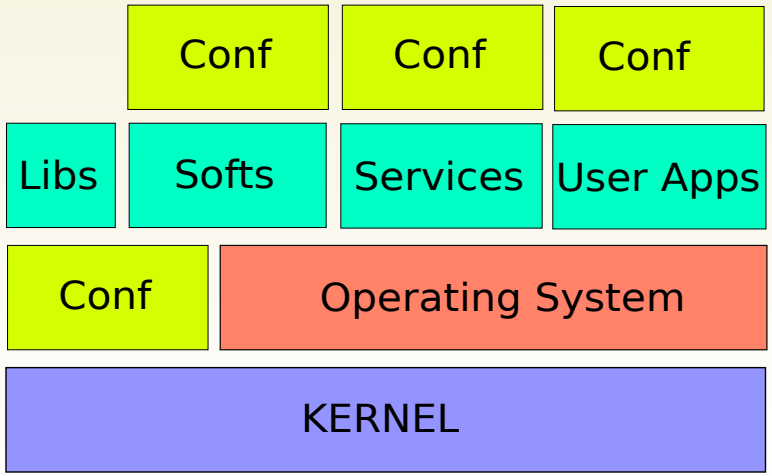
CFEngine

Concepts

- Configuration Management System
- From a policy specification: automate configuration and maintenance of computers
- Operating System-independent interface
- Describe the **FINAL** state rather than the **changes chain**
  - Promise
  - system will end up with a predictable result whatever its initial state
  - need to describe everything required in the system
- Possible actions: file operations, install package, execute command

Concepts

- Configuration Management System
- From a policy specification: automate configuration and maintenance of computers
- Operating System-independent interface
- Describe the **FINAL** state rather than the **changes chain**
  - Promise
  - system will end up with a predictable result whatever its initial state
  - need to describe everything required in the system
- Possible actions: file operations, install package, execute command

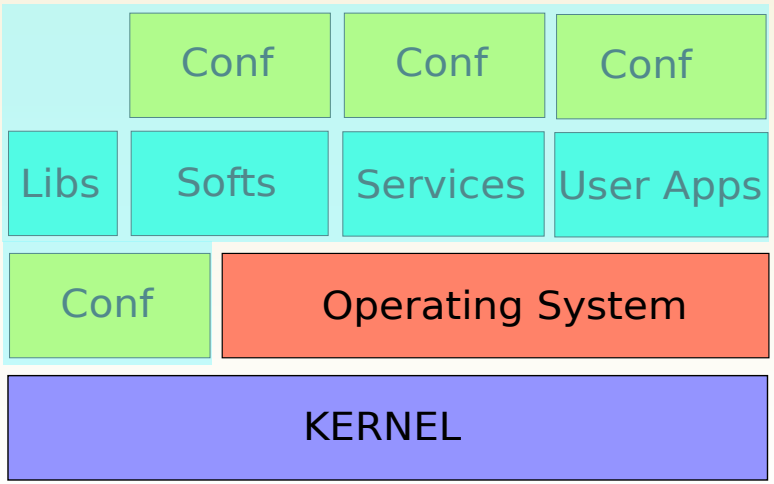


CFEngine

Concepts

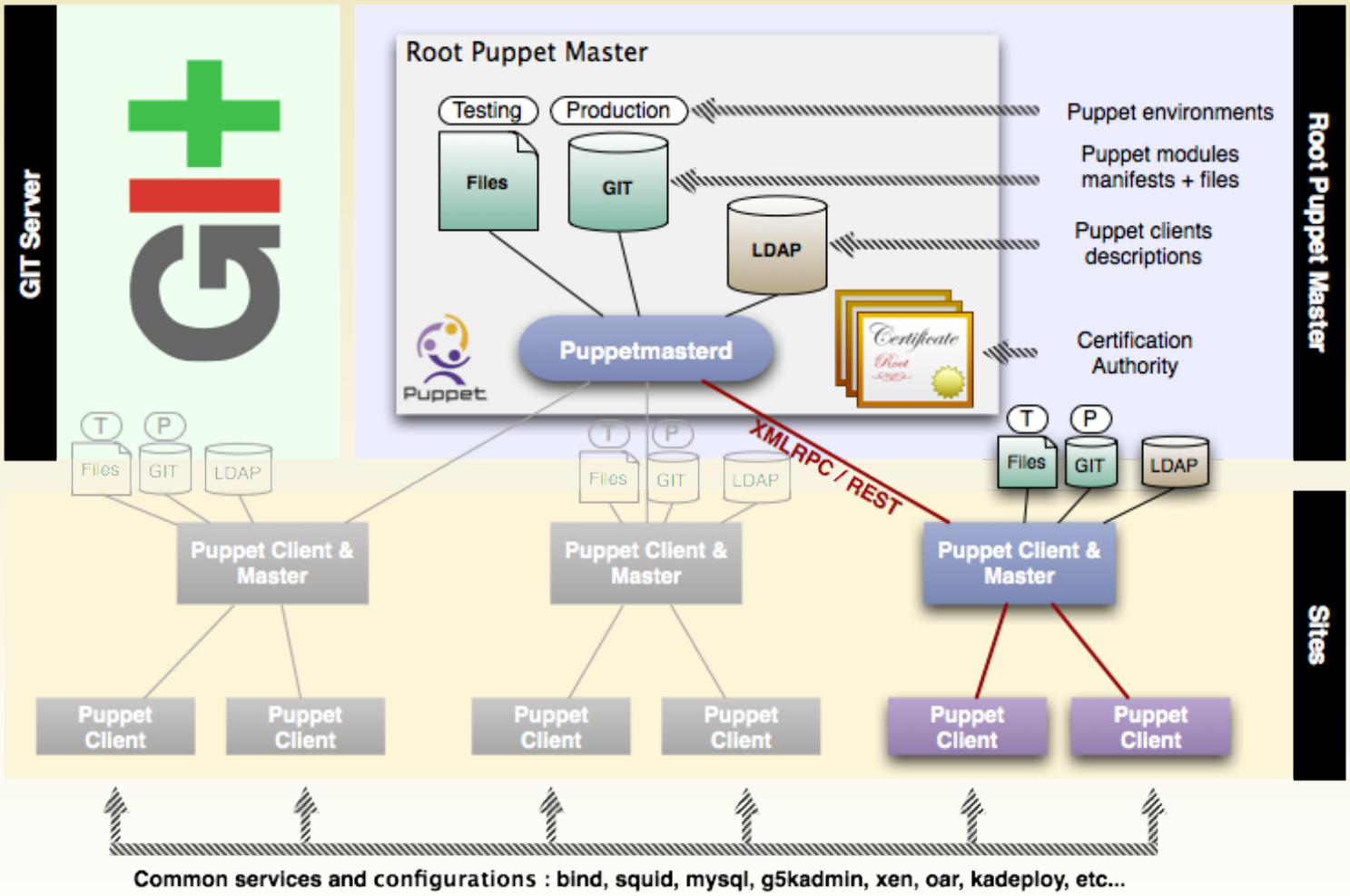
- Configuration Management System
- From a policy specification: automate configuration and maintenance of computers
- Operating System-independent interface
- Describe the **FINAL** state rather than the **changes chain**
  - Promise
  - system will end up with a predictable result whatever its initial state
  - need to describe everything required in the system
- Possible actions: file operations, install package, execute command

CFEngine



Puppet

- Automated machines configuration management
- Centralized environment specification
- Automatic clients resynchronization



Puppet

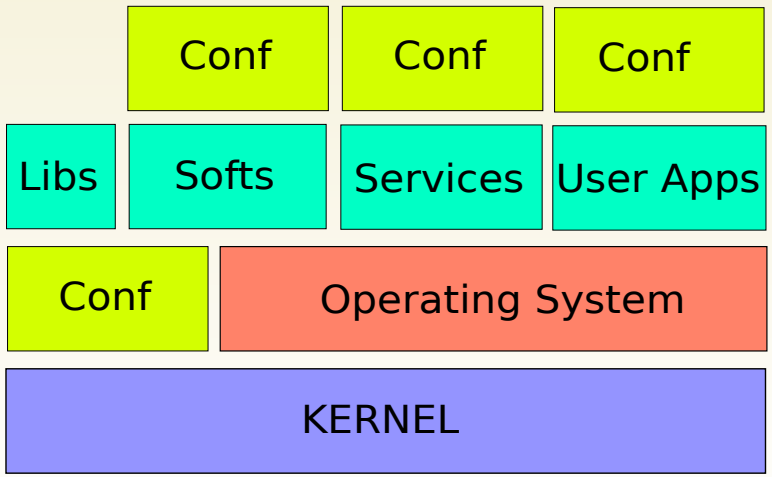
Concepts

- Same base concept as in CFEngine: focus on the **final** state
- RAL: Resource Abstraction Layer (Puppet DSL)
  - configuration in high-level terms: users, services and packages
  - platform independent language
  - support for Ruby
- Manifests: declaration of the conditions the environment requires
- Inheritance and dependencies

Puppet

Concepts

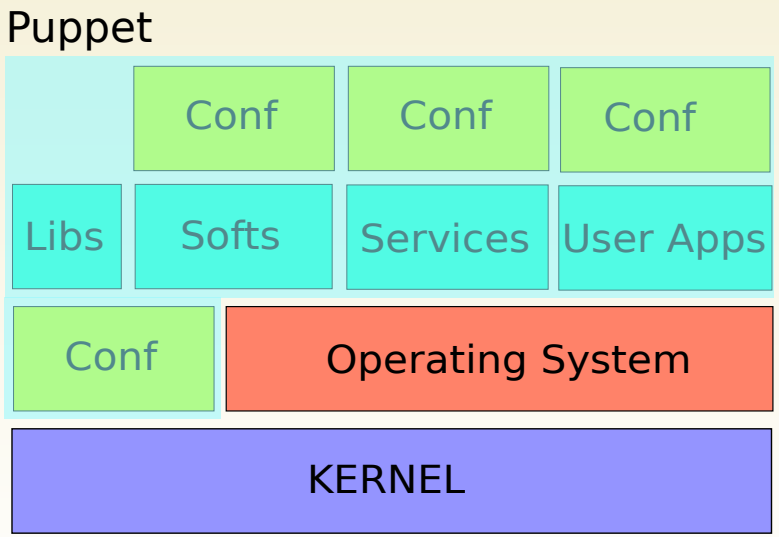
- Same base concept as in CFEngine: focus on the **final** state
- RAL: Resource Abstraction Layer (Puppet DSL)
  - configuration in high-level terms: users, services and packages
  - platform independent language
  - support for Ruby
- Manifests: declaration of the conditions the environment requires
- Inheritance and dependencies



Puppet

Concepts

- Same base concept as in CFEngine: focus on the **final** state
- RAL: Resource Abstraction Layer (Puppet DSL)
  - configuration in high-level terms: users, services and packages
  - platform independent language
  - support for Ruby
- Manifests: declaration of the conditions the environment requires
- Inheritance and dependencies



Chef

Concepts

- Both High-Level and more Technical vision
- Describe the **“changes chain”**
- Files, users, packages management operations simplified
- Powerful Ruby based recipes:
  - platform independent
    - some operations are, some not
    - recipes can “make themselves” platform independent (see next slide examples)
  - use of code
  - (re-)configurable/definable DSL
- Client/Server + Solo mode
- “Recipes” are grouped into “Cookbooks”
- Chef loads a cookbook as a git repository archive (can be remote)

Chef Examples

Example: High-level

```
case node[:platform]
when "ubuntu", "debian"
  include_recipe "setup::apt"
end
```

Chef Examples

Example: High-level

```
case node[:platform]
when "ubuntu", "debian"
  include_recipe "setup::apt"
end
```

Example: execute a command

```
execute "apt-get-update" do
  command "apt-get update"
end
```

Chef Examples

Example: High-level

```
case node[:platform]
when "ubuntu", "debian"
  include_recipe "setup::apt"
end
```

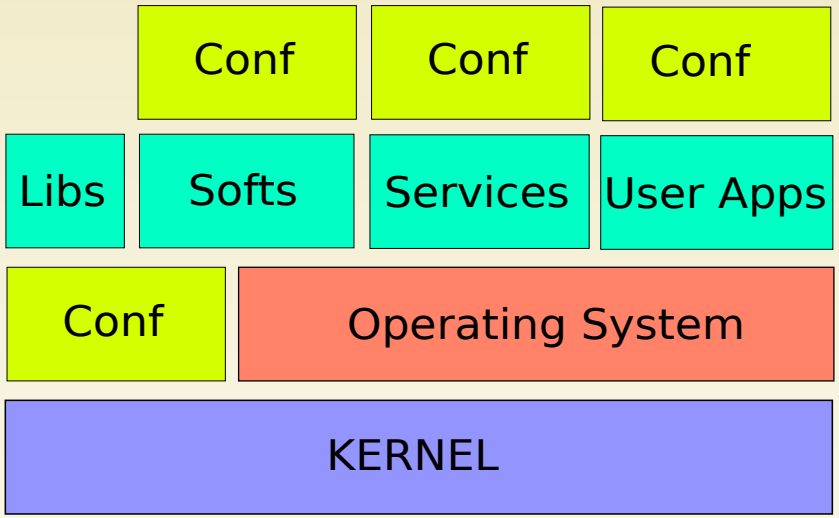
Example: execute a command

```
execute "apt-get-update" do
  command "apt-get update"
end
```

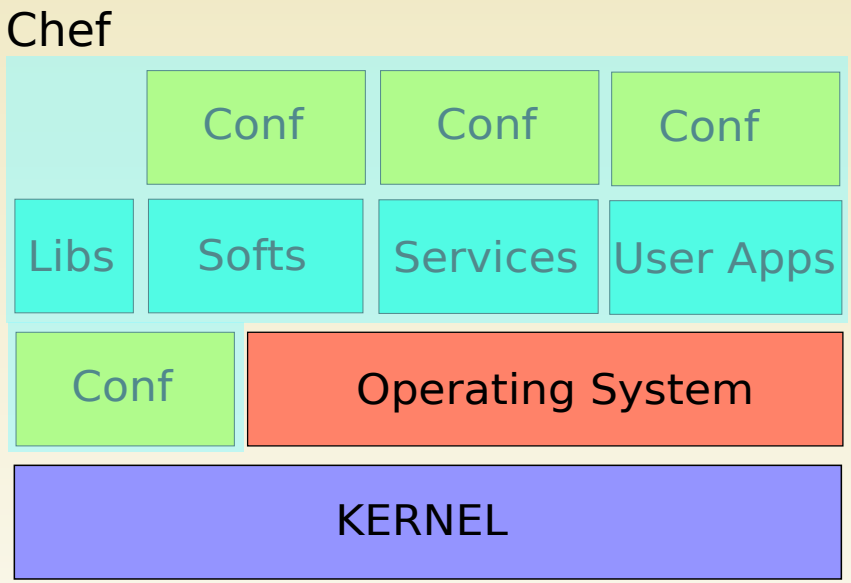
Example: install packages

```
%w{oar-common oar-doc oar-libs oar-node}.each do |pkg|
  package pkg do
    action :install
    version "2.2.16-2"
    options "--force-yes"
  end
end
```

Chef


Conf
Conf
Conf
Libs
Softs
Services
User Apps
Conf
Operating System
KERNEL

Chef


Chef
Conf
Conf
Conf
Libs
Softs
Services
User Apps
Conf
Operating System
KERNEL

Juju (formerly Ensemble)

In traditional West African religion, use of objects (charms) to perform witchcraft.

Concepts

- Package management on a higher level, around services
- Automate services deployment and configuration (cloud targeted)
- Link the services between themselves
- Ubuntu specific: Personal Package Archive (PPA)
- Charms (formulas) manage the services install/configuration part

Juju (formerly Ensemble)

In traditional West African religion, use of objects (charms) to perform witchcraft.

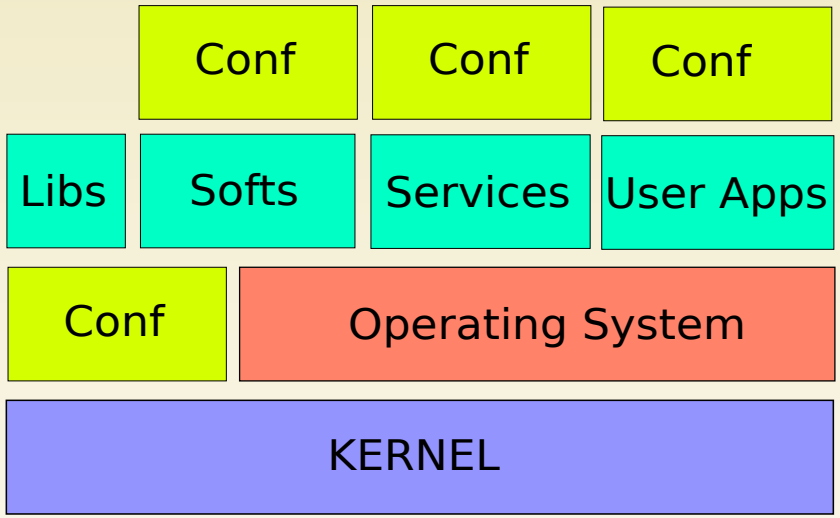
Concepts

- Package management on a higher level, around services
- Automate services deployment and configuration (cloud targeted)
- Link the services between themselves
- Ubuntu specific: Personal Package Archive (PPA)
- Charms (formulas) manage the services install/configuration part

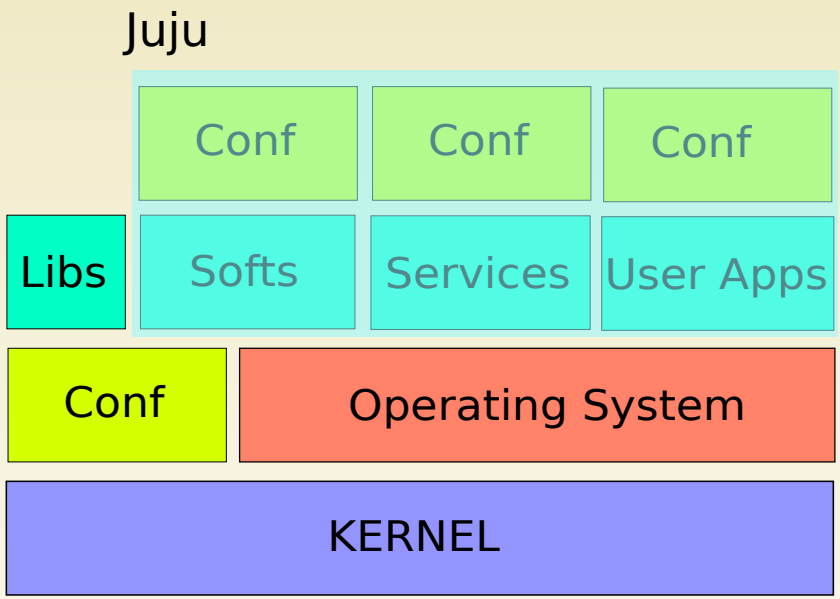
Charms

- Description of a service integration and reaction to juju events (services linking)
- Metadata
  - info, version
  - **dependencies**
- Hooks (code)
  - install, start, stop
  - relation management (services linking)

Juju


Conf
Conf
Conf
Libs
Softs
Services
User Apps
Conf
Operating System
KERNEL

Juju

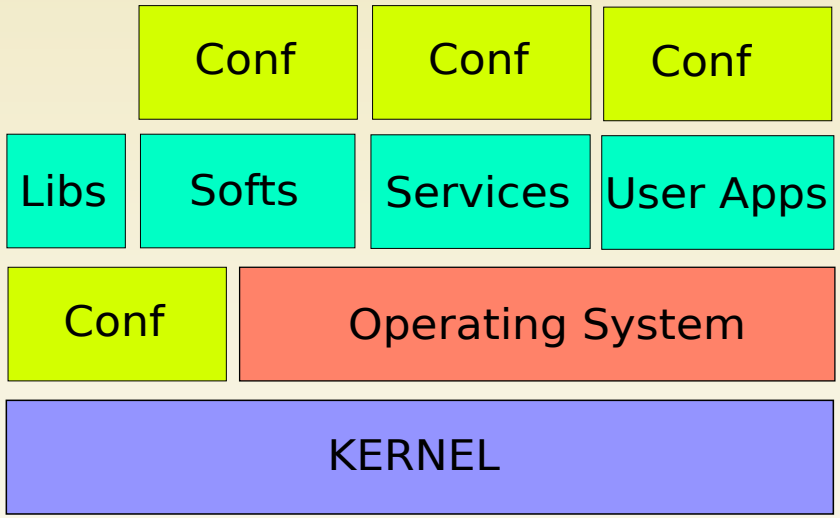

Juju
Conf
Conf
Conf
Libs
Softs
Services
User Apps
Conf
Operating System
KERNEL

UShareSoft UForge

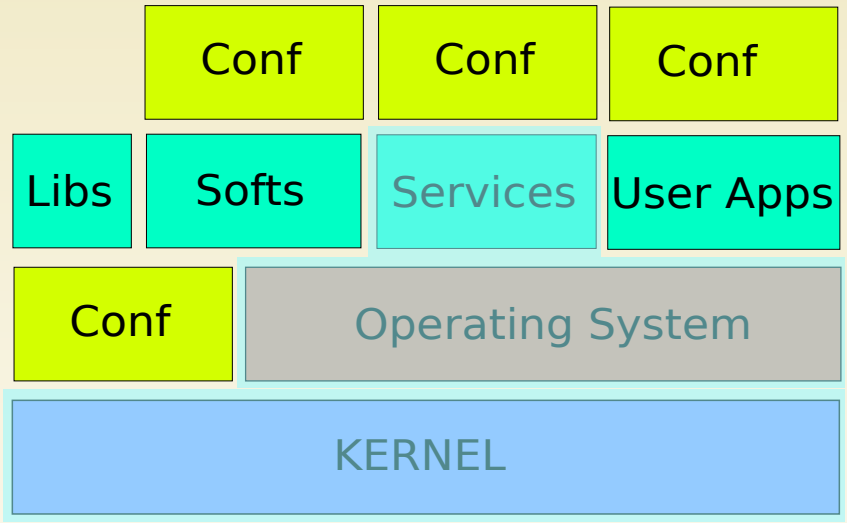
Concepts

- Commercial solution (but a restricted free account is available)
- Generate software appliances for several OS in several output formats
  - CentOS, Debian, Ubuntu, Fedora ...
  - Amazon, cloud.com, KVM, raw, VirtualBox, Xen, VMWare, iso ...
- 2 ways:
  - online: simple default image generator
  - UForge: custom image generator (choose applications to install)
- REST API for automating and streamlining software build
- Custom bootscripts + auto or custom environmental configuration
- Software image templates: update, clone and share
- Easy but blackbox

UForge


Conf
Conf
Conf
Libs
Softs
Services
User Apps
Conf
Operating System
KERNEL

UForge


Conf
Conf
Conf
Libs
Softs
Services
User Apps
Conf
Operating System
KERNEL

Kameleon

Concepts

- Model the software environment: recreate it **in the same way**
- From scratch or from an existing "basis" environment
- Kameleon describes the **"changes chain"**

Kameleon

Concepts

- Model the software environment: recreate it **in the same way**
- From scratch or from an existing "basis" environment
- Kameleon describes the **"changes chain"**

Entities

- Recipe (environment description: high level, semantic, YAML)
  - combination of steps that lead to environment construction
- Steps (low level, technical, shell code)
  - one technical action (software installation, configuration ...)

Kameleon

Concepts

- Model the software environment: recreate it **in the same way**
- From scratch or from an existing "basis" environment
- Kameleon describes the **"changes chain"**

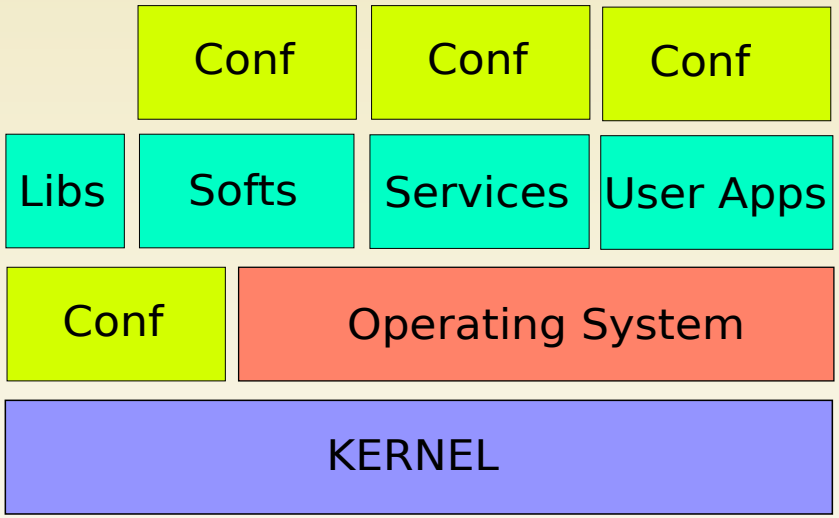
Entities

- Recipe (environment description: high level, semantic, YAML)
  - combination of steps that lead to environment construction
- Steps (low level, technical, shell code)
  - one technical action (software installation, configuration ...)

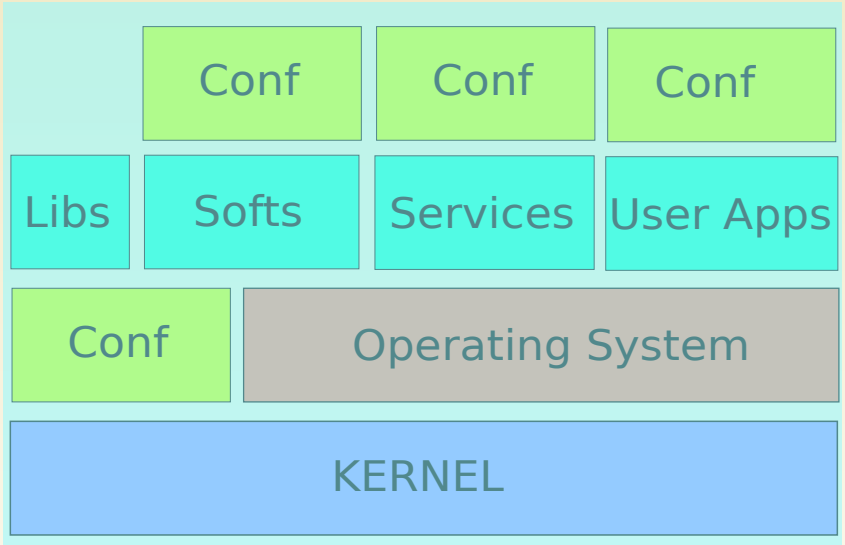
More about Kameleon

- Recipes and steps for Debian, OAR, SLURM, G5K
- Output formats: KVM, Grid'5000, Xen, VirtualBox, raw, iso
- Able to load, modify and save a UForge image

Kameleon


Conf
Conf
Conf
Libs
Softs
Services
User Apps
Conf
Operating System
KERNEL

Kameleon


Conf
Conf
Conf
Libs
Softs
Services
User Apps
Conf
Operating System
KERNEL

Other tools

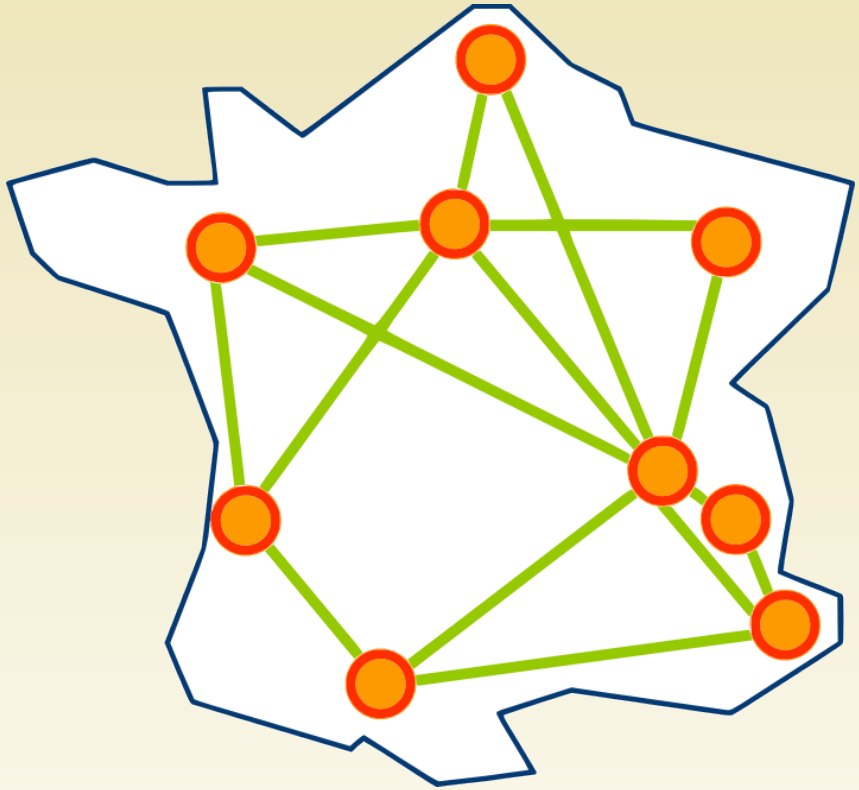
Difficult to know them all!

- Quattor (CERN)
- RBuilder (CernVM)
- VMBuilder (Ubuntu)
- Kiwi (OpenSuse - http://kiwi.berlios.de/)
- Probably more than we may imagine...

Outline

A nation-wide platform

9 sites



Sites

Bordeaux, Grenoble, Lille, Lyon, Nancy, Orsay, Rennes, Sophia, Toulouse

(+Brazil and Luxembourg)

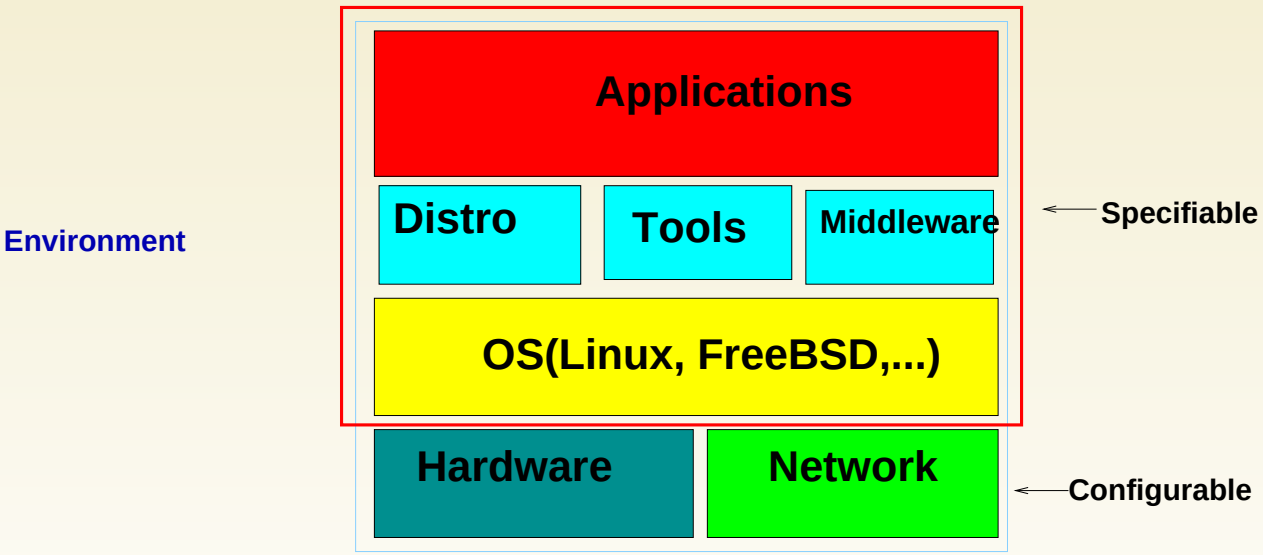
Grid'5000 Basic Design Concepts

- Research Platform dedicated to Experiments
- Large-Scale and distributed
- Heterogeneous hardware resources
- Dedicated network links between sites
  - isolate Grid'5000 from the rest of the Internet
  - let packets fly inside Grid'5000 without limitation
- **Deep reconfiguration** mechanism for experiments on all layers of the software stack
- User has **full control** of the reserved experimental resources: deployable environments

Management of the environments

Users can use:

- Default production environment
- One of the environments provided by the staff
- An environment created by another user
- An environment they created themselves



Grid'5000 environments management by the Tech Team

Service Nodes Environments

- Services: NFS, Mysql, OAR, Kadeploy...
- Configured with Puppet.
- Deployed with Capistrano (tool for deploying web applications, basically: Makefile+ssh).

Grid'5000 environments management by the Tech Team

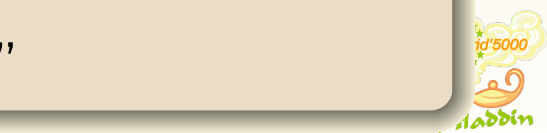
Service Nodes Environments

- Services: NFS, Mysql, OAR, Kadeploy...
- Configured with Puppet.
- Deployed with Capistrano (tool for deploying web applications, basically: Makefile+ssh).

Compute Nodes Environments

Production environments: environment installed on the compute nodes.
User environment: special environments deployable on the compute nodes.
Management: on a node of the platform itself

- manual bootstrap
- chroot, ruby and Chef install
- configuration via Chef
- different recipes for different environment "flavors"



Example

Grenoble Production environment

```
{
  "grid5000": { "site": "grenoble" },
  "kaenv": {
    "version": "1.0",
    "name": "prod",
    "site": "grenoble",
    "description": "Debian 6. Production environment. Generated with Chef.",
    "author": "support-staff@lists.grid5000.fr",
    "kernel": "/boot/vmlinuz-2.6.32-5-amd64",
    "initrd": "/boot/initrd.img-2.6.32-5-amd64",
    "postinstall": "/grid5000/postinstalls/userpostinstall-prod.tgz|tgz| \
                    traitement.ash /rambin"
  },
  "oar": { "version": "2.4" },
  "recipes": [ "setup", "oar", "g5kchecks", "ganglia", "kernel", "nvidia", \
               "fastnetwork::infiniband", "fastnetwork::openmpi" , \
               "g5kcode" , "g5ksubnets", "drivers::initramfs-up" ]
}
```

Other option: Kameleon

Full Kameleon

- Use of Kameleon to generate environment from scratch.
- Customize it.
- Output format: Grid'5000.
- Bootstrap, chroot and export as deployable image is managed by Kameleon.

Other option: Kameleon

Full Kameleon

- Use of Kameleon to generate environment from scratch.
- Customize it.
- Output format: Grid'5000.
- Bootstrap, chroot and export as deployable image is managed by Kameleon.

Mix Chef-Kameleon

- Use of Kameleon to manage minimal distrib install.
- Step for installing Chef.
- Retrieve Grid'5000 cookbooks.
- Apply Chef cookbooks inside Kameleon run (chroot).

Outline

Questions, thoughts and problems risen

Cloud computing pushes towards using virtual machines.
Grid has default configured environment.
Use provided software environments.

Questions, thoughts and problems risen

Cloud computing pushes towards using virtual machines.
Grid has default configured environment.
Use provided software environments.

Environment is Black Box

- Loose control/knowledge
- Loose efficiency
- Reproducibility impacted?
- Debug is harder

Questions, thoughts and problems risen

Cloud computing pushes towards using virtual machines.
Grid has default configured environment.
Use provided software environments.

Environment is Black Box

- Loose control/knowledge
- Loose efficiency
- Reproducibility impacted?
- Debug is harder

Make it crystal clear to the user

- Give the access to the environment construction
- Even if the user can't modify it, he will understand its behavior
- Paranoia from the users too... (Tony Cass' "trusted images" talk)

Questions, thoughts and problems risen

Cloud computing pushes towards using virtual machines.
Grid has default configured environment.
Use provided software environments.

Environment is Black Box

- Loose control/knowledge
- Loose efficiency
- Reproducibility impacted?
- Debug is harder

Make it crystal clear to the user

- Give the access to the environment construction
- Even if the user can't modify it, he will understand its behavior
- Paranoia from the users too... (Tony Cass' "trusted images" talk)

Reproducibility

Thank you for your attention

Examples

Entities:

Promises

```
promiser:
      CFEngine_word => user_defined_value;
```

Bodies

```
body CFEngine_word user_defined_value
{
      CFEngine_word => user_defined_value;
      CFEngine_word => user_defined_value;
}
```

Bundles

Set of Promises

CFEngine file management and code execution

```
files:
  "/tmp/promiser"
    perms => myexample;


body perms myexample
{
mode => "644";
owners => { "mark", "sarah", "angel" };
groups => { "users", "sysadmins", "mythical_beasts" };

#And Command Execution:
cfruncommand => "$(sys.workdir)/root/myscript.sh";
}
```

CFEngine packages install

```
control:
    any::
          actionsequence = ( packages )
          DefaultPkgMgr = ( rpm )
          RPMcommand = ( /bin/rpm )
          RPMInstallCommand = ( "/usr/bin/yum -y install %s" )


packages:
    any::
          ganglia-gmond action=install

    grid::
          lcg-CA action=install
```

Puppet

```
class kadeploy::server inherits kadeploy {

  package {
    ["kadeploy-server", "tftpd-hpa", "syslinux"]:
      ensure => installed,
      require => [User["deploy"], File["source kadeploy"],
                  Exec["sources update"]];
  }

  file {
    "/var/lib/tftpboot/kernels":
      ensure => directory,
      mode => 775, owner => root, group => deploy,
      require => Package["tftpd-hpa"];
    "/var/lib/tftpboot/pxelinux.cfg":
      ensure => directory,
      mode => 775, owner => root, group => deploy,
      require => Package["tftpd-hpa"];
  }
```

Chef

```
execute "apt-get update" do
  command "apt-get update"
end

%w{ oar-common oar-node }.each do |pkg|
  package pkg do
    action :install
    version "2.4.0"
    options "--force-yes"
  end
end

execute "usermod -U oar" do
  command "usermod -U oar"
end
```

Juju Metadata

```
name: drupal
revision: 1
summary: "Drupal CMS"
description: |
    Installs the drupal CMS system, relates to the mysql charm
    provided in examples directory. Can be scaled to multiple
    web servers
requires:
  db:
    interface: mysql
```

Juju Hooks: install

```
#!/bin/bash

set -eux # -x for verbose logging to juju debug-log
juju-log "Installing drush,apache2,php via apt-get"
apt-get -y install drush apache2 php5-gd libapache2-mod-php5 \
php5-cgi mysql-client-core-5.1 a2enmod php5
/etc/init.d/apache2 restart
juju-log "Using drush to download latest Drupal"
cd /var/www && drush dl drupal --drupal-project-rename=juju
```

Juju Hooks: relation-changed

```
#!/bin/bash
set -eux # -x for verbose logging to juju debug-log
hooksdir=$PWD
user=‘relation-get user‘
password=‘relation-get password‘
host=‘relation-get host‘
database=‘relation-get database‘
# All values are set together, so checking on a single
# value is enough
# If $user is not set, DB is still setting itself up,
# we exit awaiting next run
[ -z "$user" ] && exit 0
juju-log "Setting up Drupal for the first time"
cd /var/www/juju && drush site-install -y standard \
--db-url=mysql://$user:$password@$host/$database \
--site-name=juju --clean-url=0
cd /var/www/juju && chown www-data sites/default/settings.php
open-port 80/tcp
```

Kameleon Recipe

Recipe sample: Creation of a KVM Debian image

```
# ### debian.yaml Kameleon recipe sample ###
global:
 distrib: debian
 # Debian specific
 debian_version_name: squeeze
 distrib_repository: http://ftp.fr.debian.org/debian/
 #
 # Architecture
 arch: amd64
 kernel_arch: "amd64"
 #
steps:
 - bootstrap
 - system_config
 - root_passwd
 - mount_proc
 - kernel_install
 - strip
 - umount_proc
 - build_appliance:
    - clean_udev
    - create_raw_image
    - create_nbd_device
    - mkfs
    - mount_image
    - copy_system_tree
    - install_grub
    - umount_image
    - save_as_raw
 - clean
```

Kameleon Steps

Debian Kernel installation step

```
kernel_install:
 - kernel-img_conf:
   - write_file:
     - /etc/kernel-img.conf
     - |
       do_symlinks = yes
       relative_links = yes
       do_bootloader = yes
       do_bootfloppy = no
       do_initrd = yes
       link_in_boot = no
 - kernel_install:
   - exec_chroot: bash -c "DEBIAN_FRONTEND
     =noninteractive apt-get -y --force-yes
     install linux-image-$$kernel_arch"
```

Debian basis installation step

```
bootstrap:
 - debootstrap:
   - exec_appliance:  debootstrap --arch=
     $$arch $$debian_version_name
     $$chroot/ $$distrib_repository
```