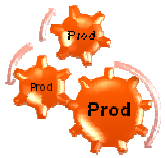
Sophia-Conf 2012 – Cfengine

Portail / Hebex - 2/7/2012

Bernard Brandl
Fabrice Clement

Sommaire

- Présentation Environnement
- Les process avant les outils
- Vue d'ensemble
- Gestion de configuration : chiffres clés
- Cfengine

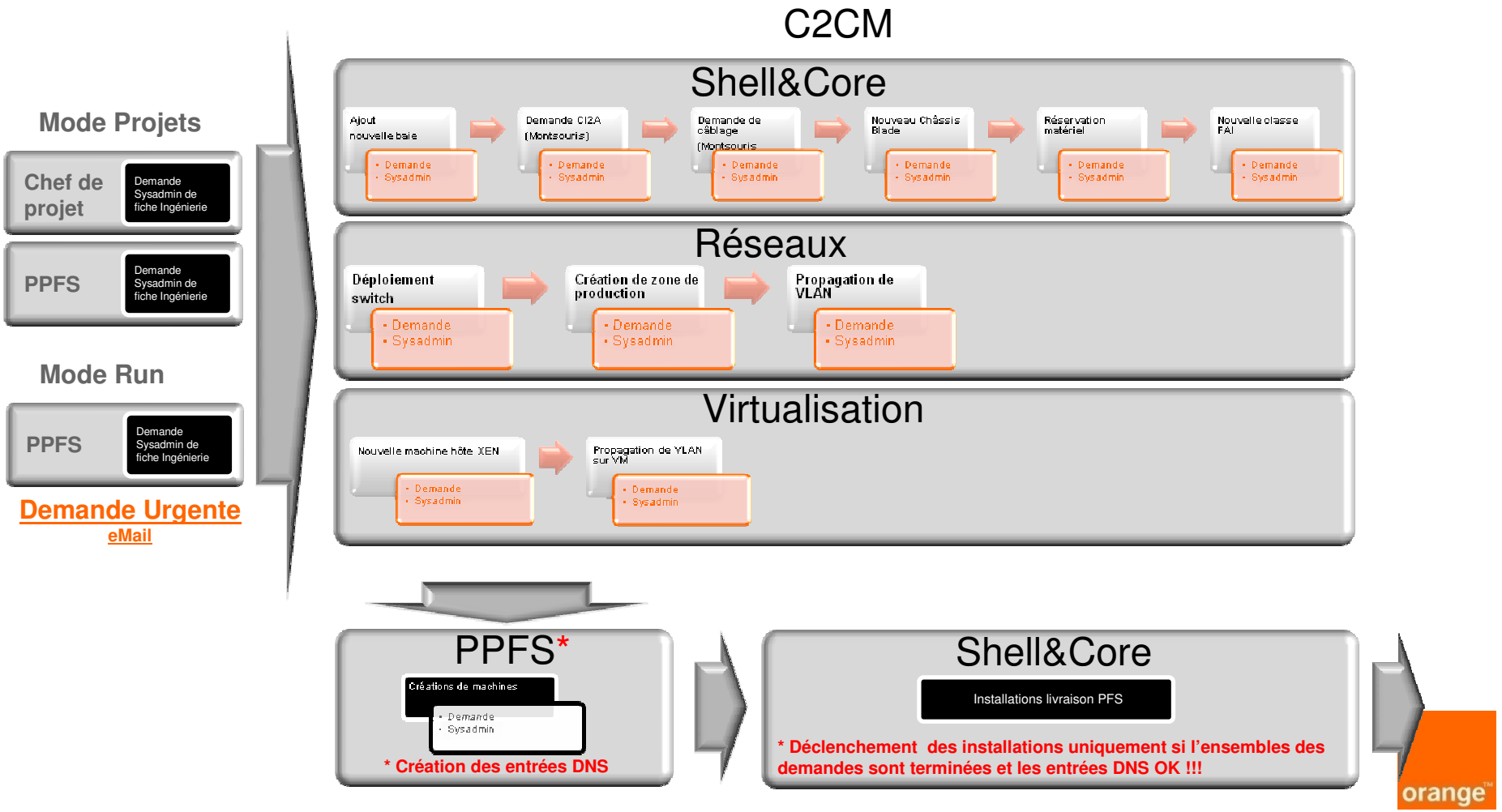


Présentation Portail / Hebex

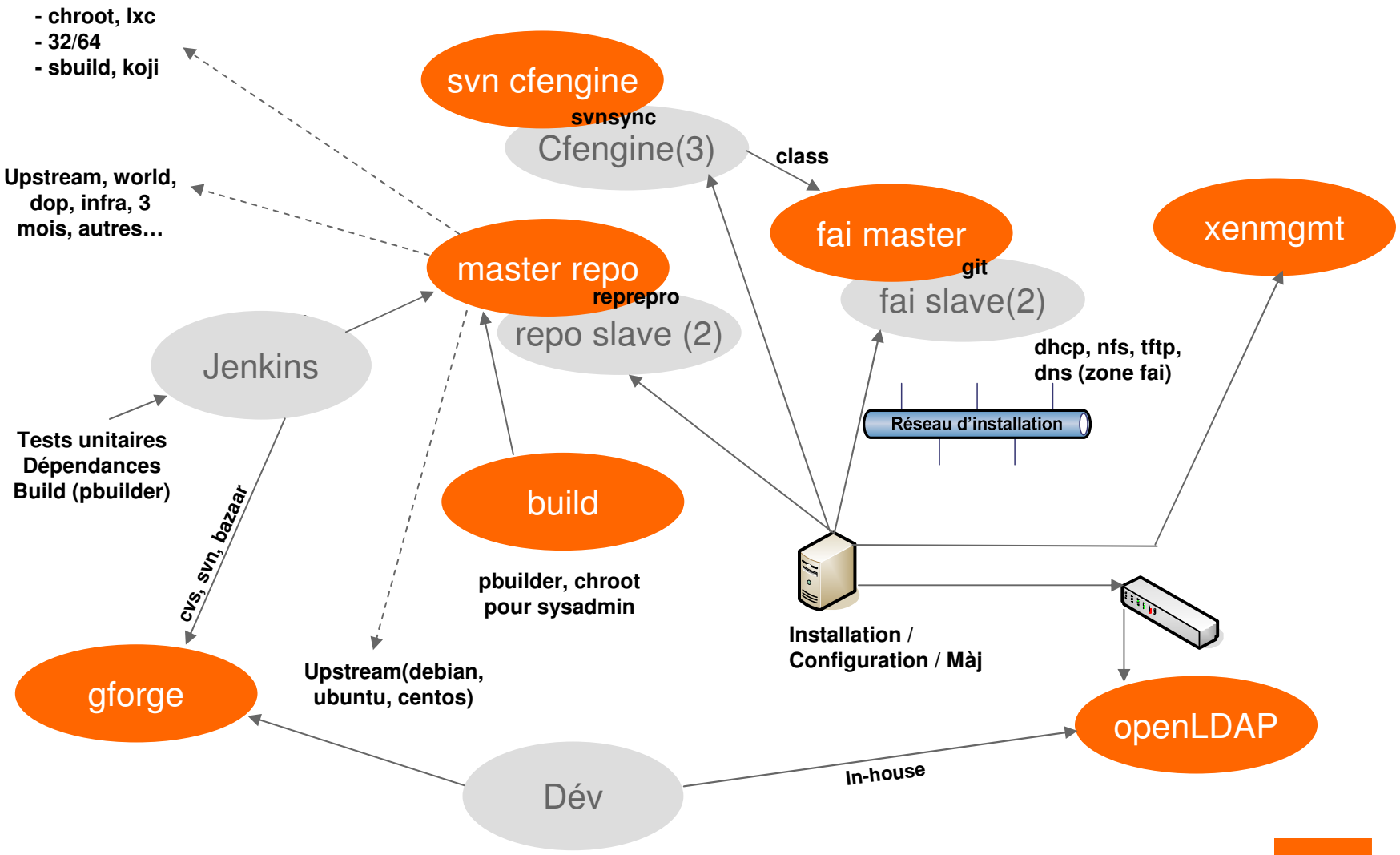
- Portail orange.fr
  - sauf messagerie
  - moteur de recherche
  - ~ 2700 URLs, 300 services
  - Contraintes : haute-disponiblité et charge (60 M pv/jour)
- #3000 serveurs, 3 DCs, 88 PFS
- People
  - 10 % sur cmdb (dév interne : sysinfo, sup, netstat…)
  - 3 % infra (sup, sauvegarde, dns, netstat, ldap…)
  - 1 % « install os » (xen, cfengine, fai…)
- 46 applications d'admin sys
- Ubuntu LTS / CentOS / Debian

Comité C2CM (Core and Capacity mgmt)

- Différents DC avec des services différents
- Coordination des équipes (s&c, net, virt, installOS, achat) dans le processus de mise à disposition d'infrastructure de production => cout dans les délais



Vue d'ensemble



FI et FAI

- FI
  - PFS, nom serveur, physique/virtuel, serveur hôte, 32/64, OS, FS, Classe FAI, Partitionnement, Nb CPU, Core, HT, RAM, RAID, disques, volume, salle baie
  - Bulle, switch, port, vip, vlan, gw
- FAI
  - Installation non interactive
  - Partionne (LVM), création filesystem, lilo/grub
  - Installation des packages
  - Notion de class, hooks
  - Shell, perl, expect, cfengine configuration
  - Réseau d'installation (PXE, temporaire)

gestion de conf & cfengine =>

Gestion de conf : A quoi çà sert?

- La gestion de configuration permet de définir et de **maintenir dans le temps** la cohérence d'un système ou d'une application en fonction de son cahier des charges initial, des politiques de sécurité générales de l'entreprise ou spécifiques à une application, des bonnes pratiques du métier et **des exigences de conformité** au sein d'une entreprise.

- **En automatisant les procédures de configuration existantes**, il est possible de faire un pas vers une gestion industrialisée du SI.

- Cfengine offre "une vigilance continue, qui permet de mettre en oeuvre des **réparations automatiques** ou de lever des alertes si la conformité des règles établies venait à défaillir."

Gestion de conf : A quoi çà sert?

- **Le serveur est HS, il faut le réinstaller**

« Cela ne pourra pas être fait avant 1 semaine, la personne qui a installé et configuré ce serveur est en vacances actuellement. »

- **On a une faille critique 0 day sur l'ensemble de nos serveurs, il faut passer le patch immédiatement !**

« On va mobiliser l'ensemble du personnel technique pendant 3 jours pour mettre à jour les 3000 serveurs »

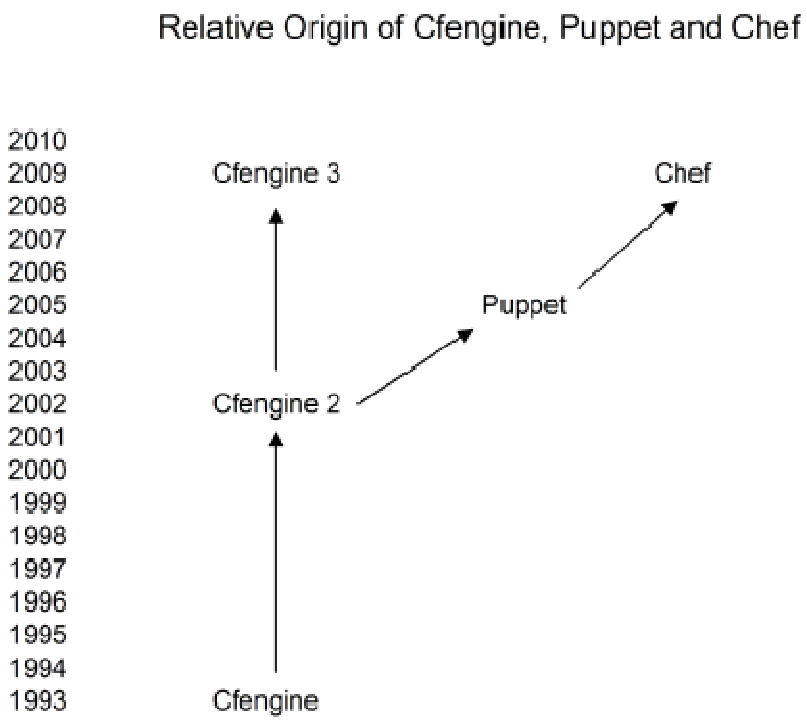
- **Quelles sont les configurations préconisées pour cette série de logiciels ?**

« Il faut demander aux différents experts techniques et à la sécurité »

- **Ce serveur n'a pas la bonne configuration**

« pourquoi et depuis quand ? »

Histoire


Relative Origin of Cfengine, Puppet and Chef
2010
2009
2008
2007
2006
2005
2004
2003
2002
2001
2000
1999
1998
1997
1996
1995
1994
1993
Cfengine 3
Chef
Puppet
Cfengine 2
Cfengine

CFEngine vs Chef vs Puppet

- Chef / Puppet : Ruby, lourdes dependances / moins de perf
- CFengine : C, faible conso, scalable par design (10kclients/hub)
- CFengine : Apprehension du langage -> go O'Reilly Learning CFengine3
- Communauté Puppet / Chef : des tonnes de recettes et de scripts déjà disponibles
- Communauté Cfengine (3) : Design Center 2012
- Cfengine : Normal ordering, PT, PULL,
- Puppet est une réaction à cfengine2
- Chef est une réaction à Puppet

Cfengine : introduction

- Solution multi-plateforme (Unix en Opensource)

- **Économe** (consommation inférieure à 1% des serveurs actuels) avec une architecture adaptée à la montée en charge.

- Stable. La **version 3** marque un tournant dans son histoire (15 ans tout de même)

- Cfengine est **open source**, core/github + paquets dispo sur *cfengine.com/downloads*

- Le **support** entreprise existe (Windows)

- C, openssl, tokyocabinet et c'est tout.

CF2/CF3:Convergence et Promesses

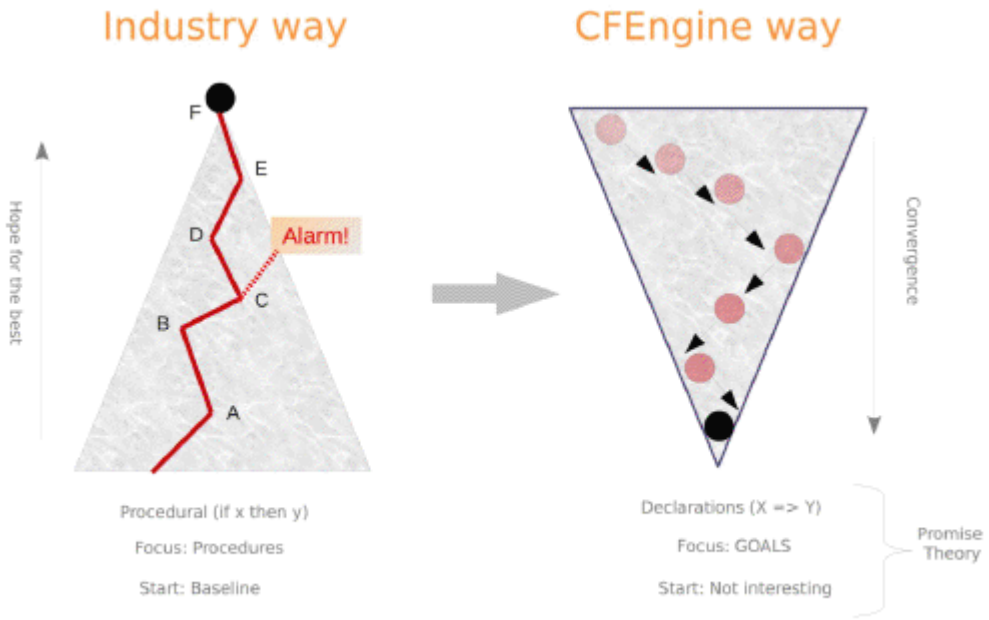
- CFengine 2 :
  - Modele sur le systeme immunitaire
  - Principe de convergence abouti
  - Tolérance aux pannes

- Cfengine 3 : œuvres sur un modèle théorique appelé *Théorie de la promesse* (http://research.iu.hio.no/promises.php).

- Ce modèle théorique relate du comportement **d'agents autonomes** dans un environnement **sans autorité centrale**, basée uniquement sur des promesses de comportement effectuées par chaque agent, et montre que, même sans contrôle central, le système peut converger vers un état stable.

CFengine: la convergence

- On converge vers l'état final désiré.
- On fait une promesse à propos de l'état que l'on souhaite avoir.
- On ne décrit pas les étapes pour y arriver



CFengine: la théorie des promesses

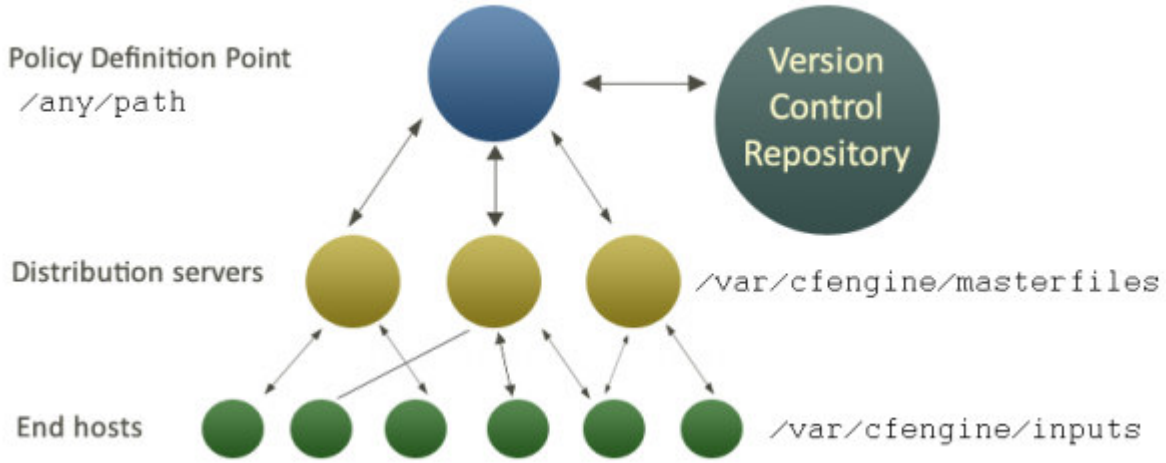
- Faire une promesse à propos de quelque chose
- CFEngine va essayer de la tenir
- **Tout est promesse**
- Promettre qu'un service tourne
- Promettre qu'un paquet est installé
- Promettre qu'un fichier existe avec certains droits
- Promettre qu'une ligne existe dans un fichier de configuration
- Promettre une certaine valeur d'une variable dans un fichier de configuration

Composants

- Un agent
- Un serveur
- Un exécuteur
- L’agent et l’exécuteur doivent être sur chaque système
- L’exécuteur réveille l’agent régulièrement
- L’agent communique avec le serveur pour obtenir la politique de configuration
- Le serveur n’est pas obligatoire
- L’agent peut utiliser une politique locale
- On peut mettre en place notre propre architecture de déploiement

Fonctionnement

Mise a jour des politiques -> PULL



De l'intention à la promesse

- Pilotage par la connaissance.

  qui, quoi, où, comment et pourquoi ?

- L'agent lit notre intention, et va nous faire une promesse

```
bundle component name(parameters)
{
what_type:
 where_when::

  # Traditional comment

  "promiser" -> { "promisee1", "promisee2" },
        comment => "The intention ....",
         handle => "unique_id_label",
    attribute_1 => body_or_value1,
    attribute_2 => body_or_value2;
}
```

```
                bundle component name(para
                {
WHAT →          what_type:

WHEN,WHERE →     where_when::

WHAT AFFECTED →  "promiser"

WHY →                  comment => "The in

                    attribute_1 => body_or
HOW →               attribute_2 => body_or
                }
```

Langage : Exemple

```
bundle agent garbage_collection

{

files:

linux.freespace_var_low::

  "$(sys.workdir)/outputs" -> {inner lopping @(sys.workdir array) is optionnal}

    delete => tidy,

    file_select => days_old("3");

}

body delete tidy

{

dirlinks => "delete";

rmdirs   => "true";

}

body file_select days_old(days)

{

mtime       => irange(0,ago(0,0,"$(days)",0,0,0));

file_result => "mtime";

}
```

Normal ordering : agent

vars
classes
outputs
interfaces
files
packages
guest_environments
methods
processes
services
commands
storage
databases
reports

vars
classes
delete_lines
field_edits
insert_lines
replace_patterns
reports

orange™

CFengine : Design Center::Sketches

- **Point central de la communauté (github)**

- **les Sketches** sont des composants directement importable et paramétrables dans vos politiques

- Outil dedie : cf-sketch.

- Revus et guarantis par l'equipe Cfengine AS.

1. Install and configure MySQL on a Linux server, using the db_mysql sketch.
2. Install and configure Wordpress on a Linux server, using the wordpress sketch.
3. Configure sshd on a Unix server, using the ssh sketch.
4. Configure DNS Settings, the local hosts table, or the system clock timezone using the config_resolver, etc_hosts or tzconfig sketches, respectively.