

# **Présentation de Chef**

Formation F2*i*

Jaber

11 Mars 2019

# Plan Général

- Chef plate-forme de gestion de configuration
- Les composants de l'architecture Chef
- Étude d'une implémentation par `chef-solo`
- Étude d'une implémentation par le serveur Chef, un poste de travail workstation et un nœud.

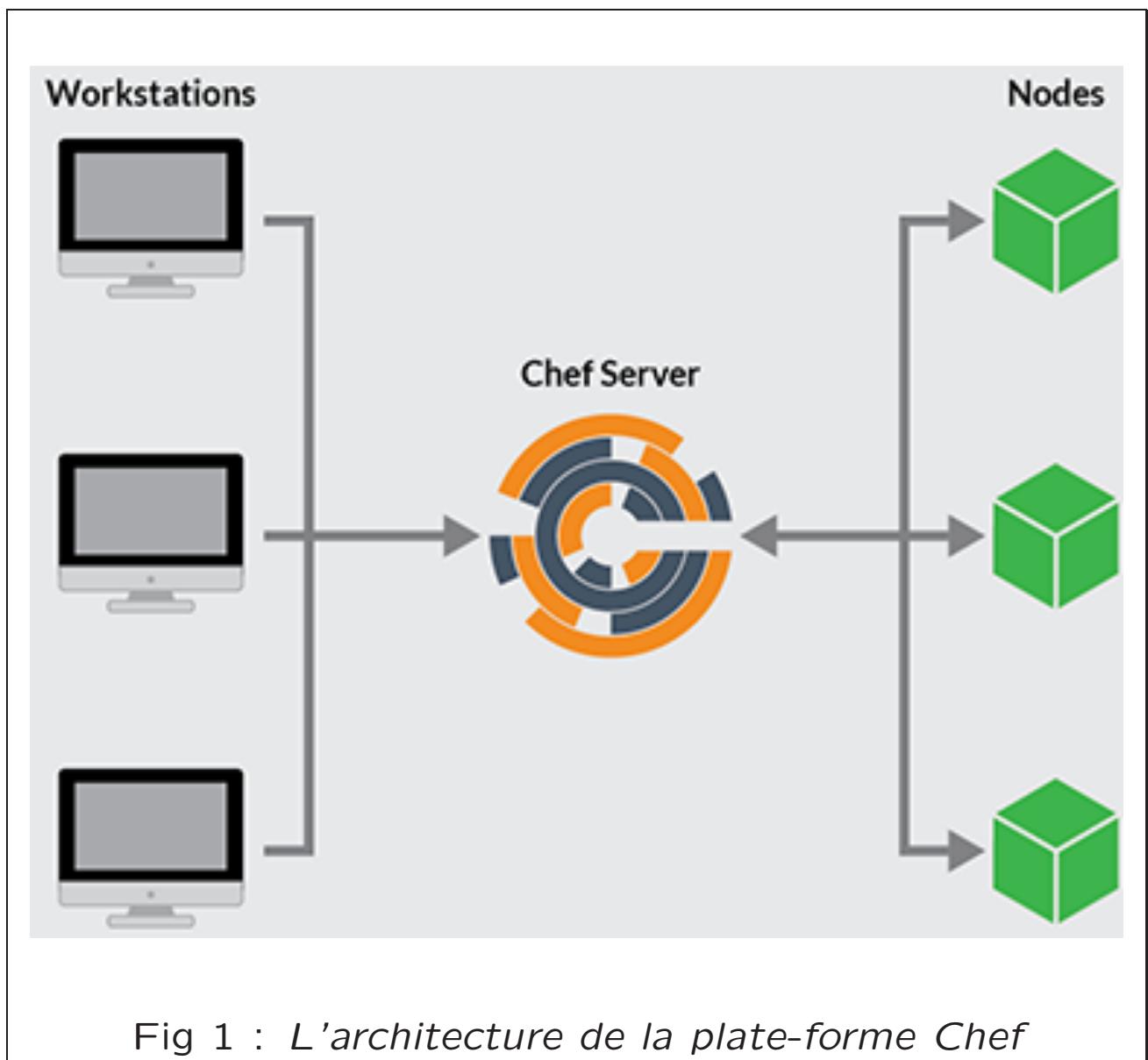
# Chef

## Plate-forme de gestion de configuration

- Chef est une plate-forme déclarative de gestion et d'automatisation de la configuration utilisée pour convertir l'*infrastructure en code*.
- Cette méthodologie permet de générer un processus avec de meilleurs tests, des déploiements efficaces et prévisibles, un contrôle de version centralisé et des environnements reproductibles sur tous les serveurs d'une infrastructure.
- Chef utilise trois composants principaux :
  - le serveur Chef ;
  - les postes de travail (Workstations) ;
  - les nœuds (Nodes).

# Chef

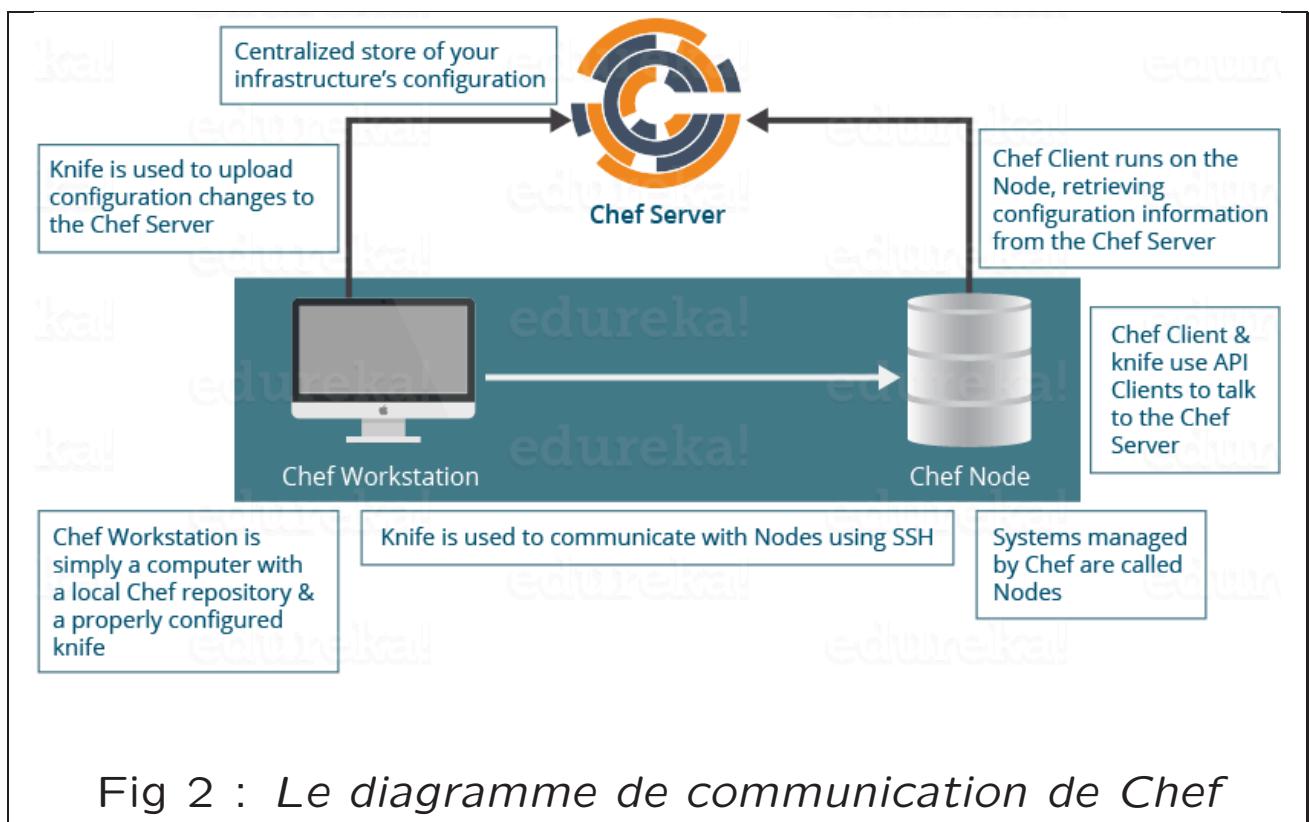
## Plate-forme de gestion de configuration



# Chef

## La gestion de configuration

- Chef est une plate-forme de gestion de configuration et d'automatisation développée par la société *Opscode*.



# Chef

## L'automatisation

- L'automatisation consiste à automatiser des opérations qui contrôlent, régulent et administrent des machines, des systèmes disparates ou des logiciels avec une intervention humaine minime, voire aucune. Autrement, l'automatisation signifie un traitement automatique avec peu ou pas d'implication humaine.
- Un système automatisé est censé exécuter une fonction de manière plus fiable, efficace et précise qu'un opérateur humain.
- La machine automatisée remplit une fonction à moindre coût et avec une efficacité supérieure à celle d'un opérateur humain. L'automatisation est donc de plus en plus répandue dans divers secteurs des services, de l'informatique et des logiciels.

# Chef

## L'automatisation

- L'automatisation aide fondamentalement une entreprise des manières suivantes :
  - réduire la complexité des processus et des étapes séquentielles ;
  - réduire les possibilités d'erreur humaine dans les tâches répétitives ;
  - améliorer de manière constante et prévisible les performances d'un système ;
  - améliore la robustesse et l'agilité du déploiement d'applications dans différents environnements et réduit le délai de commercialisation d'une application.

# Chef

## Pourquoi l'automatisation ?

- L'intensification de l'innovation dans les technologies de l'information a créé d'énormes possibilités de croissance dans les grandes organisations et les petites et moyennes entreprises.
- L'automatisation informatique est le processus d'intégration et de gestion automatisées de ressources de calcul multiformes, de middleware, d'applications d'entreprise et de services basés sur le flux de travail.
- Des méthodologies et des technologies nécessitent l'automatisation tels que : la méthodologie Agile ; la livraison continue ; la transition non efficace entre environnement de développement et environnement de production ; la communication et collaboration inefficaces entre les équipes et le *Cloud computing*.

# Chef

## Avantages de l'automatisation

- L'automatisation apporte les avantages suivants au secteur informatique :
  - Agilité et rapidité ;
  - Evolutivité ;
  - Efficacité et cohérence ;
  - Gestion efficace des ressources ;
  - Précision de déploiement.

# Chef

## L'automatisation & Outils

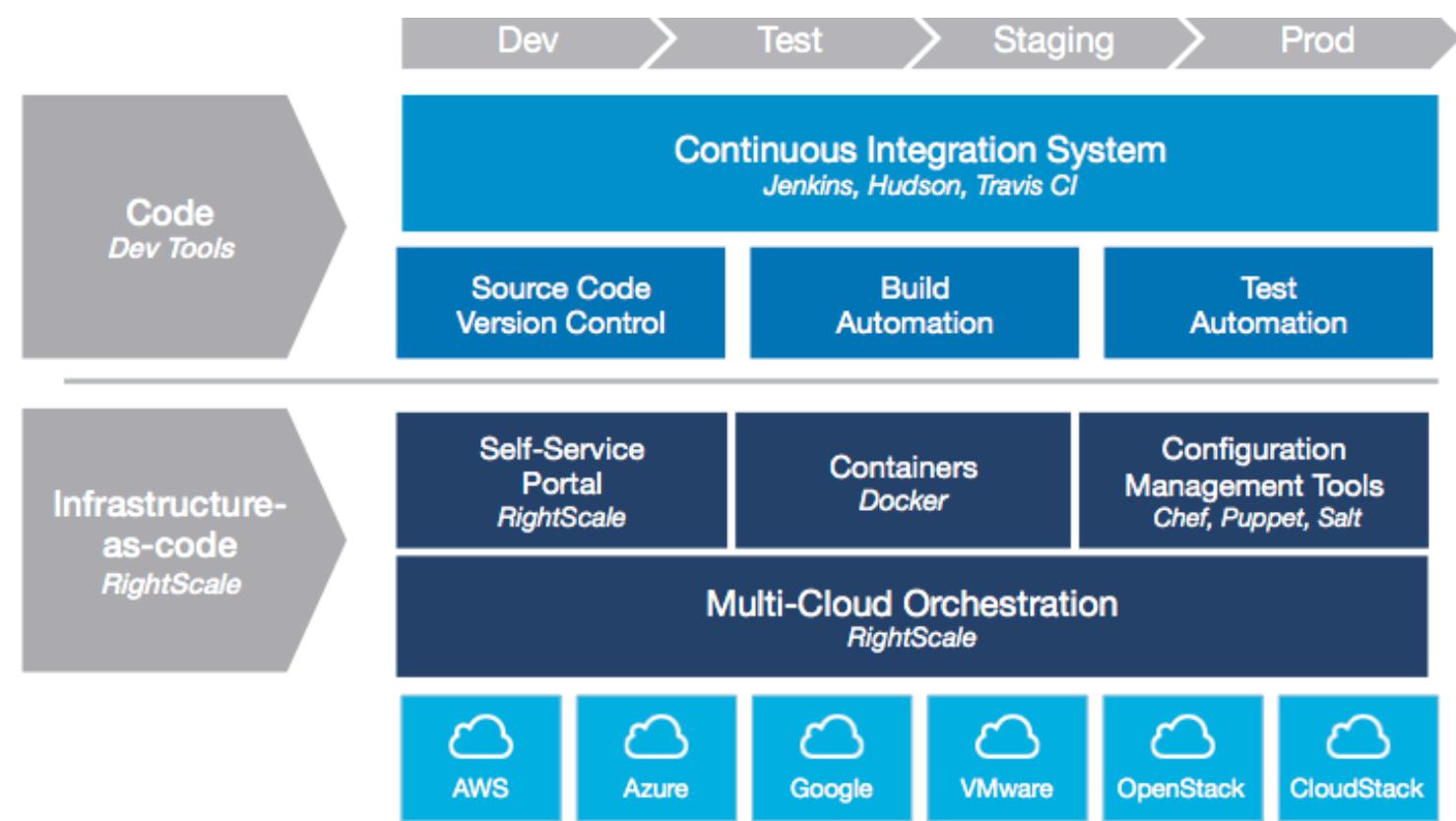


Fig 3 : *Infrastructure As A Code & Outils*

# Chef

## Infrastructures vs applications



Fig 4 : De l'infrastructure au définition d'application.

# Chef

## Comparaison

- Nous comparons certains des outils d'automatisation avancés les plus exigeants du secteur informatique actuel :

	Chef	Puppet	SaltStack	Ansible
Lic.	Apache	Apache	Apache	GPL
Lang.	Ruby/Erlang	Ruby	Python	Python
Arch.	Master/Agent	Master/Agent	Master/Agent	Standalone
Le +	Leader	Leader	Rapidité	Installation
Cloud	Oui	Oui	Oui	Oui
Util.	Facebook Linkedin Youtube Bloomberg Rackspace	Twitter Verizon Wmware Motorola Redhat	Lyft	Apple Juniper NASA SaveMart Grainger

# Chef

## La gestion de configuration

- Chef vous aide à décrire votre infrastructure avec du code. Parce que votre infrastructure est gérée avec du code, elle peut être automatisée, testée et reproduite avec facilité.

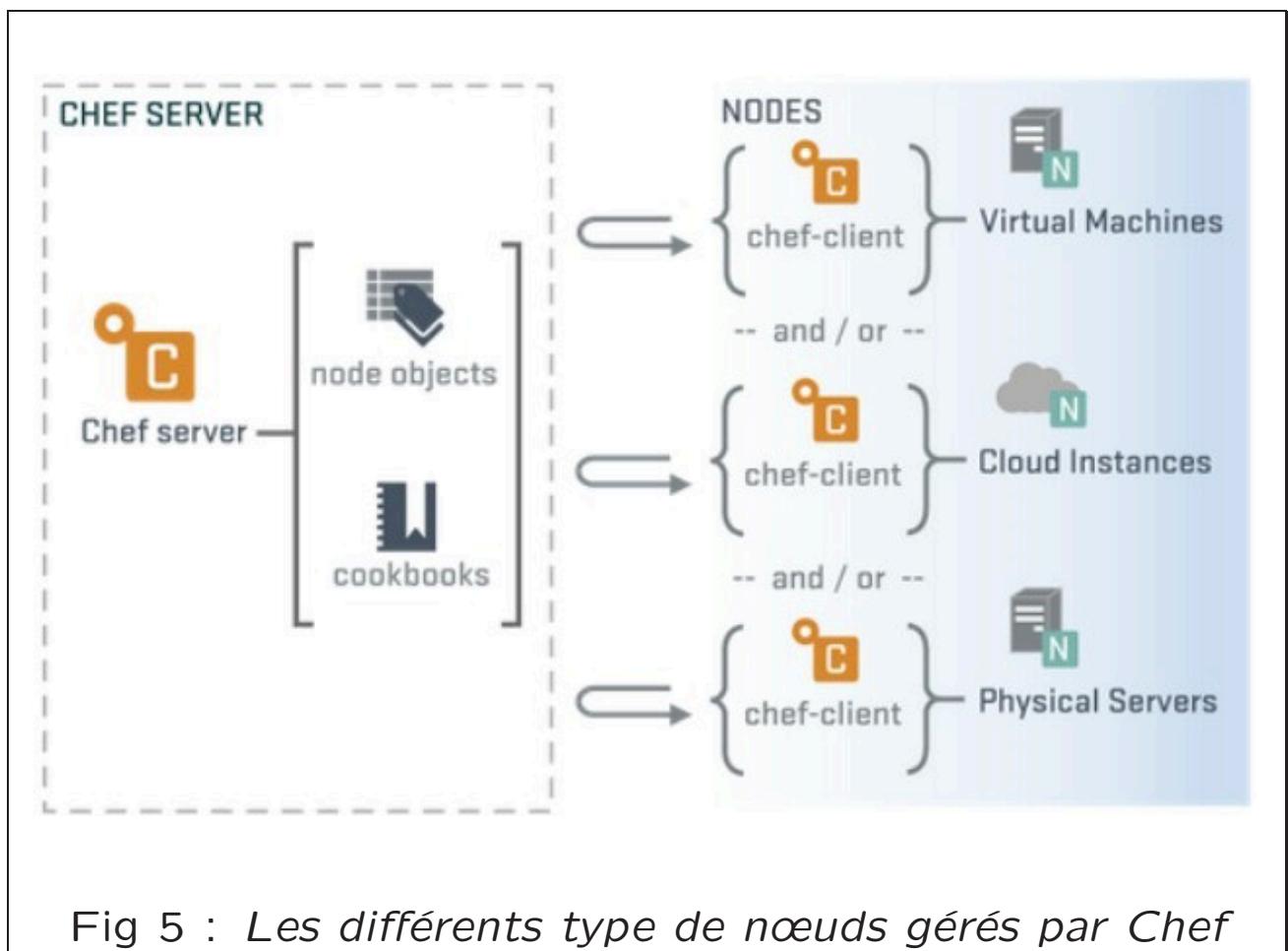


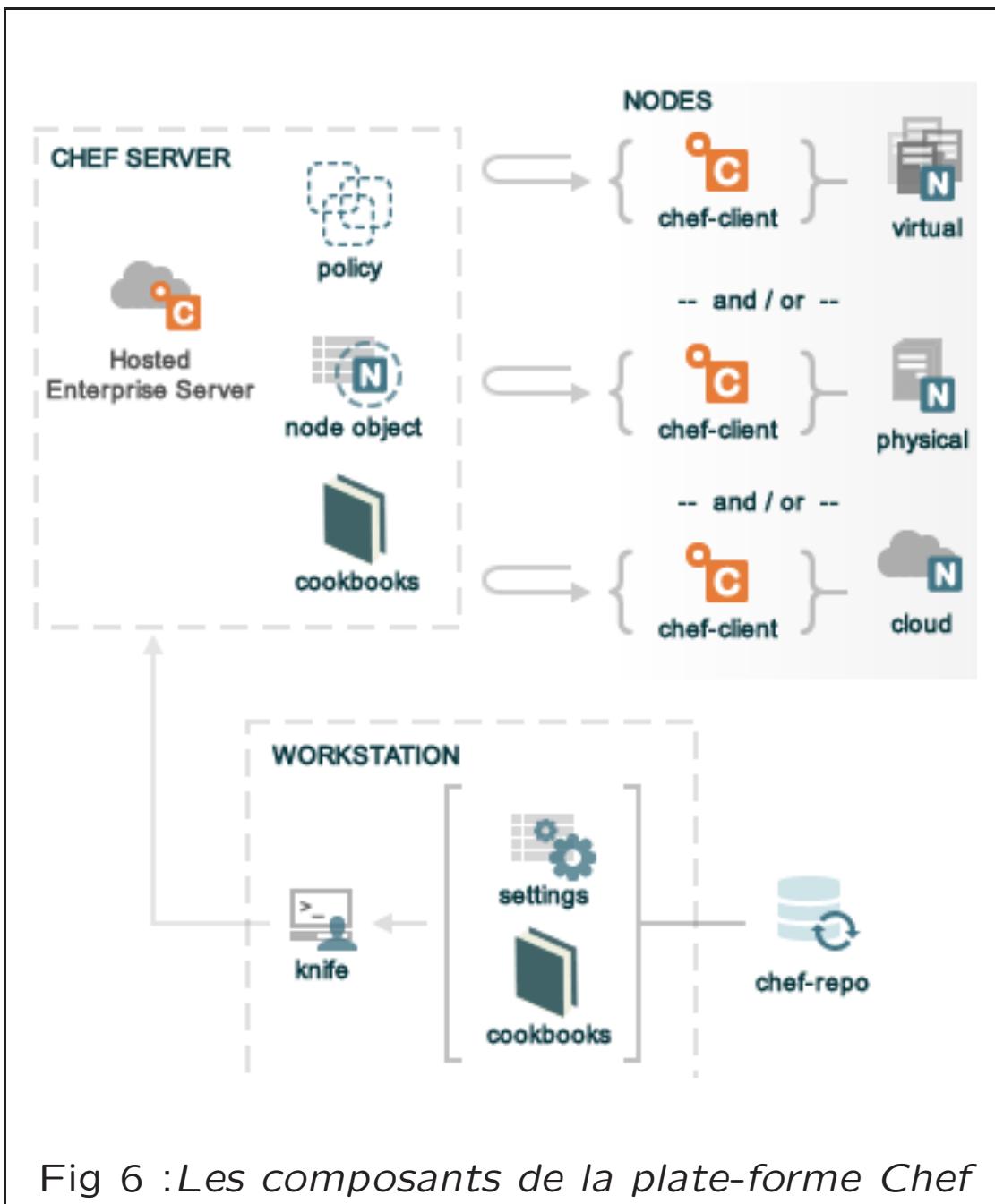
Fig 5 : Les différents type de nœuds gérés par Chef

# Chef

## Serveur Chef, Workstations & Nodes

- Ces trois composants communiquent de manière linéaire, les modifications apportées étant transférées des postes de travail (`workstations`) vers le serveur `Chef`, puis transférées du serveur vers les nœuds (`nodes`) et mises en œuvre sur chaque nœud via leur client `client-chef`.
- À leur tour, les informations sur le nœud sont transmises au serveur pour déterminer quels fichiers sont différents des paramètres actuels et doivent être mis à jour.

# Chef : les composants



# Le serveur Chef

- Le serveur Chef fournit un chemin de communication entre les postes de travail workstations sur lesquels votre infrastructure est codée et les nœuds sur lesquels les configurations sont déployées par le client Chef.
- Tous les fichiers de configuration, recettes (cookbooks), métadonnées et autres informations sont créés sur les postes de travail workstations et stockés sur le serveur Chef.

# Le serveur Chef

- Le serveur Chef conserve également des informations sur l'état de tous les nœuds au moment de la dernière exécution du `chef-client`. Un poste de travail communique avec le serveur Chef à l'aide des outils de ligne de commande `Knife` et `Chef`, tandis que les nœuds communiquent avec le serveur Chef à l'aide du client `Chef` : `chef-client`.
- Toute modification apportée à votre code d'infrastructure doit passer par le serveur Chef pour pouvoir être appliquée aux nœuds. Avant d'accepter ou de transmettre des modifications, le serveur Chef authentifie toutes les communications via son API REST à l'aide du cryptage à clé publique.

# Le serveur Chef : les composants

- Le serveur Chef est composé de plusieurs composants qui l'aident à communiquer efficacement avec les postes de travail workstations et les nœuds.
- Chaque serveur Chef dispose d'un équilibrEUR de charge frontal NGINX pour acheminer toutes les demandes à l'API du serveur Chef. PostgreSQL est utilisé pour stocker des données. Une instance Apache Solr, encapsulée par chef-solr, est utilisée pour l'indexation et la recherche.

# Le serveur Chef : les composants

- Une interface Web, appelée *Chef manage*, est disponible pour les tâches de gestion de serveur Chef courantes.
- Tous ces composants contribuent à la capacité du serveur Chef de traiter les demandes de plusieurs milliers de nœuds et font du serveur Chef une application exigeant de nombreuses ressources :

| [https://docs.chef.io/chef\\_system\\_requirements.html](https://docs.chef.io/chef_system_requirements.html)

# Chef : les composants

## Chef Components

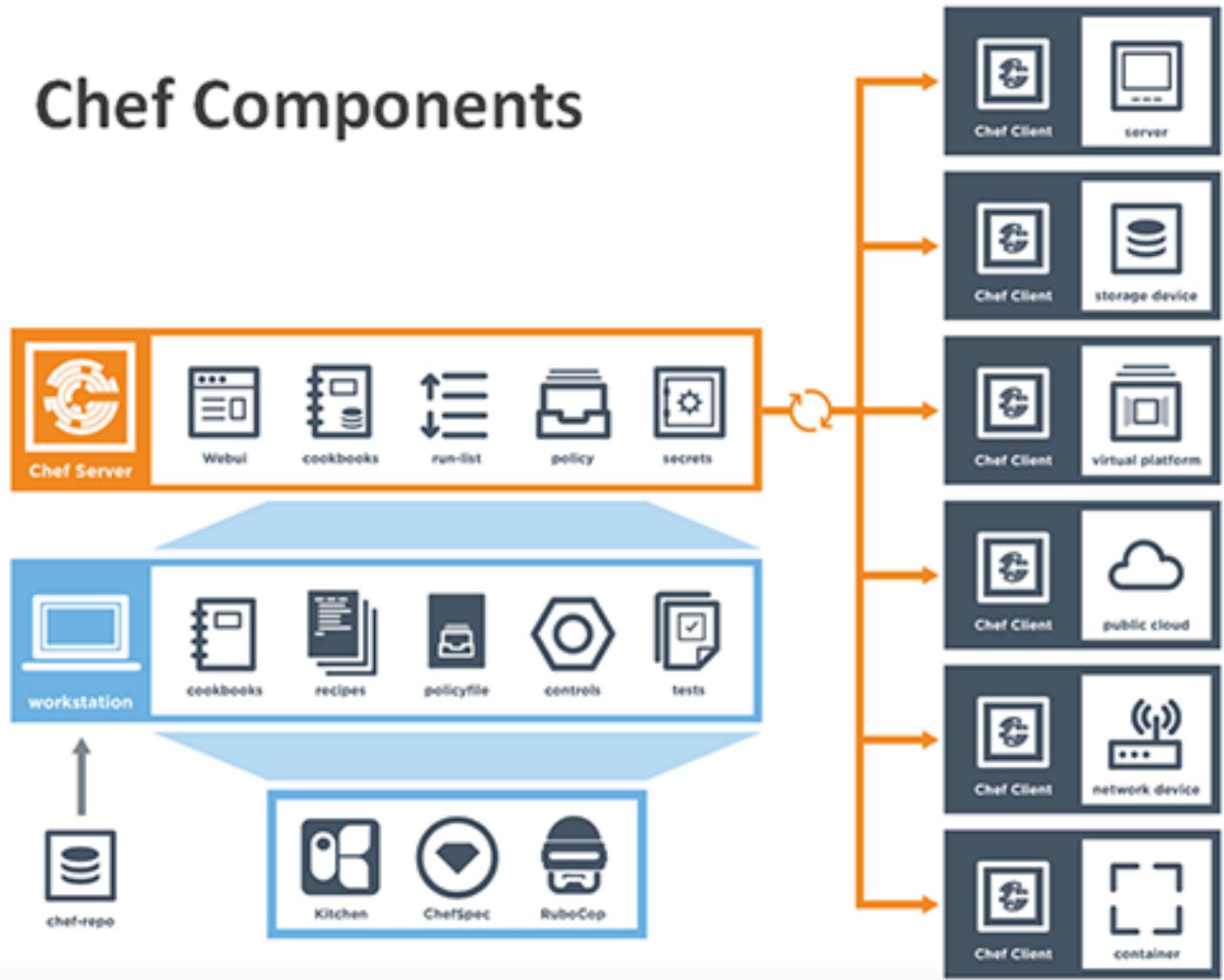


Fig 7 :Les composants de la plate-forme Chef

# Chef

## Exemple de gestion de configuration

- La façon la plus simple d'utiliser Chef est `chef-solo`. Il vous permet d'installer, de configurer et de gérer les paquets requis par votre application sans la complication de la configuration du client et du serveur, par exemple l'installation et la configuration d'un environnement Web.
- Chaque fois que vous faites cela, vous devez configurer un serveur Web, se souvenir de commandes d'installation, éditer des fichiers de configuration, aller chercher une copie du serveur et faire d'autres installations.

# Chef

## Exemple de gestion de configuration

- Comment peut-on automatiser une telle installation afin d'éviter des installations à répétitions? Avec Chef, nous pouvons définir notre infrastructure en tant que code et automatiser des tâches comme celle-ci.
- Donc avant de commencer, nous avons besoin de tester notre code ou configuration quelque part. Nous présenterons des outils qui vous aideront à gérer le développement et les tests de Chef dans la seconde partie, mais pour l'instant nous n'aurons besoin que d'un accès administrateur à une nouvelle installation.

# Chef

## Installation de Chef

### Par script

```
| curl -L https://www.opscode.com/chef/install.sh | bash
```

### Par paquet

```
| apt-get update  
| apt-get install chef
```

- Pour vérifier l'installation :

```
| chef-solo -v
```

- Les sites officiels pour d'autres types d'installations

```
| https://www.chef.io/chef/get-chef/  
| https://downloads.chef.io/chefdk
```

# Chef

## Un cookbook environnement Web

- Pour avoir un environnement *Web*, nous devons installer et configurer :
  - un serveur Apache ;
  - une serveur MySQL ;
  - un interprète PHP ;
  - Déployer le code.
- Afin de construire un cookbook, nous devrions mettre en place une structure de fichiers qui nous aidera à organiser nos différents fichiers Chef. *Opscode*, en fournit à travers le *Chef Repository*.

# Chef

## Mise en place de l'arborescence

```
cd ~  
wget http://github.com/opscode/chef-repo/tarball/master  
tar -zxf master  
mv chef-chef-repo* chef-repo  
rm master  
cd chef-repo/  
ls
```

- Les recettes cookbook de Chef résident dans le répertoire cookbooks. Nous allons l'appeler phpapp. Nous pouvons utiliser la commande chef (knife pour les versions précédentes) pour nous aider à gérer nos cookbooks.
- D'abord, nous devrions indiquer à chef où trouver notre répertoire de cookbooks.

```
cd ~/chef-repo  
mkdir .chef  
echo "cookbook_path [ '/home/dada/chef-repo/cookbooks' ]" > .chef/knife.rb
```

# Chef

## Un exemple de cookbook

- Un fichier de configuration a été créé pour chef qui lui indique d'utiliser le répertoire cookbooks. Ensuite, nous demandons à chef de créer le cookbook nommé phpapp.

```
curl -s https://omnitruck.chef.io/install.sh | bash -s -- -P chefdk
cd ~/chef-repo/cookbooks
chef generate cookbook phpapp
cd phpapp
ls
```

- Le cookbook pour installer et configurer Apache, MySQL et PHP peut être trouvé sur le site de cookbooks de *Opscode Community*.

| <https://supermarket.chef.io/>

# Chef

## Un exemple de Cookbook

- Vous trouverez des cookbooks bien conçus et testés qui feront la plupart du travail pour vous. Considérez-les comme des bibliothèques que vous utiliserez dans votre code.

## Cookbook Apache2

- Pour le cookbook Apache2, il n'est pas nécessaire de le télécharger manuellement à partir du site de la communauté, la commande `knife` a cette fonctionnalité intégrée.

# Chef

- Nous allons également installer le cookbook apt. Cela nous aidera à faire en sorte que chef-solo fasse une mise à jour apt-get avant d'installer des paquets.

```
cd ~/Chef/chef-repo/cookbooks
knife supermarket download apache2 6.0.0
tar -zxf apache2*
rm apache2*.tar.gz
knife supermarket download apt
tar -zxf apt*
rm apt*.tar.gz
cd phpapp
```

# Chef

## Un exemple de Cookbook

- Il s'agit de modifier le fichier `metadata.rb` en rajoutant à la fin :

```
| depends 'apache2'
```

- Ceci indique à Chef que notre cookbook ou recette se fonde sur le cookbook d'Apache2 pour fonctionner. Ensuite à la fin du fichier `recipes/default.rb` rajouter :

```
| include_recipe "apache2"
```

# Chef

- Cela inclut la recette – *recipe* – par défaut du cookbook Apache2 dans notre configuration (recette). La recette *recipe* Apache2 par défaut (qui peut être trouvée dans les cookbooks/apache2/recipes/default.rb) installe et configure Apache pour nous.
- Ensuite, il s'agit de créer solo.rb dans le répertoire chef-repo ; cd ~/chef-repo/ :

```
| file_cache_path "/home/dada/chef-solo"  
| cookbook_path "/home/dada/chef-repo/cookbooks"
```

# Chef

## Un exemple de Cookbook

- Ce fichier configure la commande `chef-solo`, lui indiquant où conserver son cache de fichiers et où sont nos cookbooks. Ensuite, il s'agit de créer un fichier appelé `web.json` dans le répertoire `~/chef-repo`

```
{  
  "run_list": [ "recipe[apt]", "recipe[phpapp]" ]  
}
```

- Nous indiquons à Chef de lancer le cookbook `apt` suivi par notre cookbook `phpapp`. Pourquoi n'avons-nous pas inclus le cookbook `apt` dans la recette ou `recipe` comme nous l'avons fait avec le cookbook `Apache2`?

# Chef

## Un exemple de Cookbook

- C'est parce que notre application PHP nécessite Apache pour fonctionner, mais nous ne voulons pas nécessairement lier le cookbook à des plates-formes qui ne supportent que apt.

## Exécution d'une configuration

```
| cd ~/Chef/chef-repo/  
| chef-solo -c solo.rb -j web.json
```

- Chef fournit des informations complètes sur ce qu'il fait. Par défaut, les actions prises sont affichées en vert et quand il met à jour un modèle, il montre ce qui a changé. Suite à cette installation, un serveur Apache2 sera mis en place.

# Chef

## Un serveur MySQL

- Pour la suite, il s'agit de configurer un serveur MySQL. Comme le site communautaire a un cookbook pour MySQL, le processus est similaire à Apache. Encore une fois la commande `knife` se charge de chercher le cookbook sur le site de la communauté :

```
| https://supermarket.chef.io/cookbooks/mysql#knife
```

## Procédures d'installation

```
| cd ~/Chef/chef-repo/cookbooks  
| knife supermarket download mysql 4.1.2
```

# Chef

## Un serveur MySQL

- Il serait possible de spécifier une version particulière de MySQL en l'indiquant après le nom du cookbook. Il arrive que nous souhaitons utiliser une version particulière plutôt que la dernière version, comme le montre les paramètres passés à la commande `knife`.
- Nous procédons à l'installation de MySQL. Nous souhaitons installer le client MySQL et le serveur afin de tester notre application depuis la même machine. Les `recipes` ou recettes de MySQL sont dans le répertoire :

```
| cd ~/Chef/chef-repo/cookbooks/mysql/recipes  
| ls
```

- Il y a des `recipes` ou recettes de client et de serveur. Nous aurons besoin d'inclure les deux.

# Chef

## Un serveur MySQL

- Il s'agit d'indiquer dans notre configuration l'installation de MySQL à travers le fichier `metadata.rb` comme suit :

```
| cd ~/Chef/chef-repo/cookbooks/phpapp
```

- Il s'agit de rajouter la ligne qui indique l'installation

```
| depends 'mysql', '8.4.0'
```

- Ainsi, dans le fichier `recipes/default.rb` rajouter après le `recipe apache2` :

```
| include_recipe "apache2"
| include_recipe "mysql::client"
| include_recipe "mysql::server"
```

# Chef

## Exécution de la configuration

```
cd ~/Chef/chef-repo/  
chef-solo -c solo.rb -j web.json  
chef-solo -c solo.rb -j web.json -l debug
```

- Nous avons une erreur. Le cookbook build-essential est introuvable. Nous ne l'avons pas inclus dans notre cookbook, c'est probablement nécessaire pour le cookbook MySQL que nous venons d'ajouter.

# Chef

## Exécution de la configuration

- Nous devons spécifier les dépendances du cookbook dans `metadata.rb`, nous allons donc regarder le fichier `metadata.rb` dans le cookbook MySQL.

```
...
depends 'openssl'
depends 'build-essential'

depends 'homebrew'
depends 'windows'
...
```

# Chef

## Téléchargement des cookbooks

```
cd ~/Chef/chef-repo/cookbooks/
knife supermarket download openssl 1.1.0
tar zxf openssl*.tar.gz
rm openssl*.tar.gz
knife cookbook site download build-essential
tar zxf build-essential-*.tar.gz
rm build-essential-*.tar.gz
```

- Les cookbooks suivants pourraient aussi être téléchargés pour certaines versions :

```
knife cookbook site download homebrew
knife cookbook site download windows
knife cookbook site download chef_handler
knife cookbook site download chef-sugar
```

# Chef

## Exécution de la configuration

```
cd ~/Chef/chef-repo/  
chef-solo -c solo.rb -j web.json
```

- En cas d'erreurs, il pourrait s'agir, comme indiqué par le message de Chef que nous devons définir un mot de passe `root` pour MySQL.
- C'est un attribut. Dans Chef, les attributs sont des valeurs que nous utilisons pour configurer nos applications ou notre plate-forme. Un attribut pourrait être un numéro de port pour Apache. Souvent, une valeur par défaut est spécifiée dans un cookbook. Une telle valeur par défaut pour un port de serveur Web serait 80.

# Chef

## Configuration MySQL

- Il n'y a pas de mot de passe MySQL par défaut, donc nous devons en spécifier un dans le fichier `web.json`.

```
{  
  "mysql": { "server_root_password": \  
             "808052769e2c6d909027a2905b224bad", \  
             "server_debian_password": "569d1ed2d46870cc020fa87be83af98d", \  
             "server_repl_password": "476911180ee92a2ee5a471f33340f6f4"},  
  "run_list": [ "recipe[apt]", "recipe[phpapp]" ]  
}
```

# Chef

## Exécution de la configuration

```
| cd ~/Chef/chef-repo/  
| chef-solo -c solo.rb -j web.json
```

## Installation de PHP 5

```
| cd ~/Chef/chef-repo/cookbooks/  
| knife supermarket download php 5.0.0  
| tar zxf php*.tar.gz  
| rm php*.tar.gz
```

- Le cookbook PHP dépend des cookbooks XML, yum-epel, windows et iis, donc nous en aurons besoin même si nous ne les utiliserons pas tous. Nous devrons également installer des sous-dépendances yum (une dépendance de yum-epel), chef\_handler, et powershell (dépendances de windows).

# Chef

## Dépendances PHP

```
cd ~/Chef/chef-repo/cookbooks/
knife cookbook site download xml
tar zxf xml-*.tar.gz
knife cookbook site download yum
tar zxf yum-*.tar.gz
knife cookbook site download yum-epel
tar zxf yum-epel-*.tar.gz
knife cookbook site download powershell
tar zxf powershell-*.tar.gz
knife cookbook site download iis
tar zxf iis-*.tar.gz
rm *.tar.gz
```

# Chef

## Utilisation du cookbook PHP dans notre cookbook

- Pour utiliser le cookbook PHP dans notre cookbook, nous le rajoutons dans le fichier  
~/Chef/chef-repo/cookbooks/phpapp/metadata.rb :

```
depends "apache2"
depends "mysql"
depends "php"
```

- Il reste à inclure le cookbook PHP dans notre cookbook dans le fichier phpapp/recipes/default.rb

```
...
include_recipe "apache2"
include_recipe "mysql::client"
include_recipe "mysql::server"
include_recipe "php"
include_recipe "php::module_mysql"
include_recipe "apache2::mod_php5"
...
...
```

# Chef

## Utilisation du cookbook PHP dans notre cookbook

- Nous ajoutons le cookbook PHP par défaut, un pour installer l'extension PHP MySQL et une pour activer le module PHP Apache mod\_php.
- Nous activons également le site par défaut afin que nous puissions vérifier que notre installation a fonctionné.

## Exécution de la configuration

```
| cd ~/Chef/chef-repo/  
| chef-solo -c solo.rb -j web.json
```

- On peut faire un essai en créant une page de test sous /var/www/test.php par exemple.

```
| <?php phpinfo(); ?>
```

# Chef

## Processus idempotent

- Chef a mis à jour tous les fichiers de configuration pour chaque paquet avec des valeurs par défaut cohérentes. Si nous installons un autre serveur et exécutons les mêmes cookbooks, les mêmes programmes seront installés.
- Nous pouvons répéter la commande `chef-solo` à plusieurs reprises et nous obtiendrons le même résultat que la première fois que nous l'avons exécuté. Le processus est *idempotent*; il produira toujours le même résultat, peu importe le nombre de fois qu'il est exécuté.

# Chef

## Processus idempotent

- Si vous configurez un serveur manuellement, vous devez configurer une base de données, copier le code de l'application *Web*, créer un utilisateur MySQL pour le site *Web* et configurer des hôtes virtuels.
- Chef peut automatiser la configuration de notre application. Cela nous permet de configurer plusieurs serveurs et nous savons que nous obtiendrons toujours les mêmes résultats.

# **Le serveur Chef : les composants**

## **Le Bookshelf**

- Pour stocker des recettes ou des cookbooks, des fichiers et des modèles associés, le serveur Chef utilise une bibliothèque qui fonctionne comme un référentiel (avec gestion de versions). Lorsqu'un cookbook est chargé sur le serveur Chef, la nouvelle version du cookbook est comparée à celle déjà stockée. S'il y a des changements, une nouvelle version est stockée.
- Le serveur Chef ne stocke qu'une seule copie d'un fichier ou d'un modèle, ce qui signifie que si les ressources sont partagées entre les cookbooks et leurs versions, elles ne seront pas stockées plusieurs fois.

# Chef : les composants

## Les Workstations

- Les postes de travail ou `workstations` sont les environnements où les utilisateurs créent, testent et gèrent des `cookbooks` et des règles qui seront transmis au serveur Chef et extraits par des nœuds.
- La fonctionnalité de poste de travail `workstation` est disponible en téléchargeant le package Chef Workstation, autrefois connu sous le nom de ChefDK.
- Chef Workstation fournit des outils de ligne de commande de Chef et de Knife, les outils de test Test Kitchen, ChefSpec, Cookstyle, Foodcritic et InSpec, qui vous permet de rédiger des tests automatisés pour les exigences de conformité, de sécurité et de règles. En outre, le gestionnaire de dépendance pour les `cookbooks` et `berkshelf` est installé.

# Chef : les composants

## Les Workstations

- Le package Chef Workstation peut être installé sur des serveurs virtuels ou sur des machines personnelles. Les postes de travail sont configurés pour interagir avec un seul serveur Chef et la plupart des travaux sont effectués dans le répertoire `chef-repo` situé sur le poste de travail.
- Les cookbooks créés sur les postes de travail peuvent être utilisés à titre privé par une organisation ou téléchargés vers le Supermarket Chef pour que d'autres puissent les utiliser. De même, les postes de travail peuvent être utilisés pour télécharger des cookbooks créés par d'autres utilisateurs de Chef et trouvés dans le supermarché.

| [https://docs.chef.io/about\\_chefdk.html](https://docs.chef.io/about_chefdk.html)

# Chef : les composants

## Le répertoire de dépôt

- Le répertoire `chef-repo` est la zone du poste de travail où les recettes `cookbooks` sont créés et conservés. Toutes les ressources associées, telles que les rôles, les données et les environnements, sont stockées.
- Le `chef-repo` doit être contrôlé par la version avec un système de contrôle de version distant, comme Git. Chef est en mesure de communiquer avec le serveur à partir du `chef-repo` et de transmettre tout changement via la commande `knife`.
- Pour générer un référentiel Chef à l'aide de la commande suivante :

```
| chef generate repo nom-dépot
```

# Chef : les composants

## Knife

- L'outil de ligne de commande `Knife` est le moyen principal par lequel un poste de travail communique le contenu de son répertoire `chef-repo` avec un serveur Chef. Il fournit également une interface permettant de gérer les noeuds, les recettes `cookbooks`, les rôles, les environnements et les balises de données.
- Une commande `Knife` exécutée à partir d'un poste de travail utilise le format suivant :

```
| knife sous-commande [ARGUMENT] (options)
| knife supermarket download nginx
| knife user show dada
```

- `Knife` permet d'exécuter diverses autres opérations utiles sur le serveur et les nœuds Chef :

```
| https://docs.chef.io/knife.html
```

# Chef : les composants

## Tester des configurations

- Test Kitchen vous fournit un environnement de développement sur votre poste de travail pour créer, tester vos recettes avant de distribuer son contenu sur vos nœuds de production.
- Vous pouvez utiliser l'outil de ligne de commande Kitchen pour exécuter des tests d'intégration sur différentes plates-formes, ce qui vous permet de tester la diversité des noeuds exécutés sur votre infrastructure de production.

| <https://kitchen.ci/docs/getting-started/introduction/>

# Chef : les composants

## Les Nœuds/Nodes

- Un nœud est une machine gérée par un serveur Chef. Chef peut gérer des nœuds qui sont des serveurs virtuels, des conteneurs, des périphériques réseau et des périphériques de stockage.
- Un client Chef correspondant doit être installé sur chaque nœud afin d'exécuter les étapes nécessaires pour amener le nœud dans les conditions requises définies par un cookbooks.
- Les nœuds sont validés via les certificats `validator.pem` et `client.pem` créés sur le nœud lors de son amorçage.

# Chef : les composants

## Les Nœuds/Nodes

Tous les nœuds doivent être démarrés sur SSH en tant qu'utilisateur `root` ou utilisateur disposant de privilèges élevés.

- Les nœuds sont tenus à jour grâce à l'utilisation de `chef-client`, qui établit une convergence entre le nœud et le serveur Chef.
- Les cookbooks et les rôles que le nœud va prendre dépendent de la liste d'exécution et de l'environnement définis pour le nœud en question.

# Chef : les composants

## chef-client

- La commande application `chef-client` vérifie la configuration actuelle du nœud avec les recettes (`recipes`) et les stratégies (`policies`) stockées sur le serveur Chef et le met à jour.
- Le processus commence par la vérification par le `chef-client` de la liste d'exécution (`run list`) du nœud, le chargement des recettes `cookbooks` requis, puis la vérification et la synchronisation des recettes `cookbooks` avec la configuration actuelle du nœud.
- Le `chef-client` doit être exécuté avec des privilèges élevés pour pouvoir configurer correctement le nœud. Il doit être exécuté périodiquement pour garantir que le serveur est toujours à jour. Cette opération est souvent réalisée via une planification `cron job` ou en configurant le `chef-client` en tant que service.

# Chef : les composants

## Run Lists

- Les listes d'exécution (Run Lists) définissent les recettes (recipes) qu'un nœud utilisera.
- La liste d'exécution est une liste ordonnée de tous les rôles (roles) et recettes (recipes) que le chef-client doit extraire du serveur Chef pour s'exécuter sur un nœud.
- Les rôles (roles) sont utilisés pour définir des modèles et des attributs sur plusieurs nœuds.

# Chef : les composants

## Ohai

- Ohai collecte des informations sur les nœuds pour le client Chef. Il est nécessaire qu'il soit présent sur chaque nœud et qu'il soit installé dans le cadre du processus d'amorçage (bootstrap process).
- Les informations collectées incluent l'utilisation du réseau et de la mémoire, les données de la CPU, les données du noyau, les noms d'hôte, les noms de domaine complets et d'autres attributs automatiques permettant au client Chef de déterminer l'état du nœud avant d'appliquer la liste d'exécution de ces nœuds.

# Chef : les composants

## Environnements

- Les environnements Chef imitent le flux de travail réel, permettant aux nœuds d'être organisés en différents *groupes* qui définissent le rôle que le nœud joue dans la flotte.
- Cela permet aux utilisateurs de combiner des environnements et des recettes cookbooks avec une gestion des versions afin d'avoir différents attributs pour différents nœuds.
- Les environnements sont définis dans le fichier `chef-repo/environnements` et enregistrés en tant que fichiers Ruby ou JSON.
- Tous les nœuds sont automatiquement définis sur l'environnement *par défaut* au démarrage. Pour changer cela, l'environnement doit être défini dans le fichier `client.rb` qui se trouve dans le répertoire `/etc/chef` sur les nœuds.

# Chef : les composants

## Cookbooks

- Les cookbooks constituent la base de la gestion des configurations sur n'importe quel nœud. Les cookbooks contiennent des valeurs et des informations sur l'état souhaité d'un nœud, puis le serveur Chef et le client Chef garantissent que l'état défini est atteint.
- Les cookbooks sont composés de recettes (recipes), métadonnées (metadata), attributs (attributes), ressources, modèles (templates), bibliothèques (libraries) et de tout autre élément permettant de créer un système fonctionnel.

# Chef : les composants

## Cookbooks

- Les attributs et les recettes constituant les deux parties essentielles d'un cookbook.
- Les composants d'un cookbook doivent être modulaires, les recettes doivent rester petites et associées.
- Les cookbooks doivent être contrôlés par la version. Les versions peuvent vous aider lorsque vous utilisez différents environnements Chef et vous permettent de distribuer et de collaborer sur des cookbooks avec d'autres membres de l'équipe.

# Chef : les composants

## Recipes

- Un cookbook Chef contient des recettes indiquant l'état souhaité par les nœuds.
- Les recettes sont écrites en Ruby et contiennent des informations sur tout ce qui doit être exécuté, modifié ou créé sur un nœud.
- Les recettes fonctionnent comme un ensemble de ressources qui déterminent la configuration ou la politique d'un nœud, les ressources étant un élément de configuration de la recette.
- Pour qu'un nœud puisse exécuter une recette, il doit figurer dans la liste d'exécution de ce nœud (`run list`).

# Chef : les composants

## Attributes

- Les attributs (`attributes`) définissent des valeurs spécifiques concernant un nœud et sa configuration et sont utilisés par le client Chef pour appliquer ces attributs aux nœuds via sa liste d'attributs.
- Le client Chef peut recevoir des attributs de nœuds, fichiers d'attributs, recettes, environnements et rôles.
- Les attributs sont souvent utilisés avec des modèles et des recettes pour définir les paramètres :

| <https://docs.chef.io/attributes.html>

# Chef : les composants

## Files

- Ce sont des fichiers statiques qui peuvent être téléchargés sur des nœuds.
- Les fichiers peuvent être des fichiers de configuration, des scripts, des fichiers de sites Web. Par exemple, vous pouvez avoir une recette qui utilise un fichier `index.php`.
- Vous pouvez utiliser un bloc de ressources `cookbook_file` dans une recette pour créer le fichier sur un nœud. Tous les fichiers statiques doivent être stockés dans un répertoire `files` de la recette `cookbook`.

# Chef : les composants

## Libraries

- Bien que Chef soit livré avec un certain nombre de bibliothèques intégrées, des bibliothèques supplémentaires peuvent être définies.
- Les bibliothèques vous permettent d'écrire du code Ruby à inclure dans un cookbook.
- Les bibliothèques constituent un moyen pratique d'inclure du code auxiliaire pour vos recettes existantes.
- Les bibliothèques offrent un moyen puissant d'étendre les ressources créées par vos recettes.

# Chef : les composants

## Resources

- Les ressources sont écrites en Ruby et définies dans des fichiers de recette.
- Les ressources doivent contenir un type, un nom, une ou plusieurs propriétés et une ou plusieurs actions.
- Les ressources sont les composants clés de chaque recette :

| <https://docs.chef.io/resource.html>

# Chef : les composants

## Templates

- Les modèles ou `templates` sont des fichiers Ruby incorporés ou embarqués (`.erb`) utilisés pour créer dynamiquement des fichiers texte statiques.
- Pour utiliser un modèle dans un `cookbook`, vous devez déclarer une ressource de modèle dans une recette et inclure un fichier de modèle `.erb` correspondant dans un sous-répertoire de modèle `template`.
- Votre ressource de modèle peut contenir des variables qui seront ensuite utilisées par le modèle pour fournir dynamiquement ces valeurs en fonction d'un contexte particulier de nœuds.

# Le serveur Chef

- Chef est un outil de gestion de la configuration basé sur Ruby utilisé pour définir l'infrastructure en tant que code.
- Cela permet aux utilisateurs d'automatiser la gestion de nombreux nœuds et de maintenir la cohérence entre ces nœuds.
- Les recettes déclarent l'état souhaité pour les nœuds gérés et sont créées sur le poste de travail d'un utilisateur à l'aide du package Chef Workstation.
- Vos recettes sont réparties sur les nœuds via un serveur Chef.
- Un client Chef, installé sur chaque nœud, est chargé d'appliquer la recette au nœud correspondant.

# Le serveur Chef

## Installation d'un serveur Chef

- Une machine serveur Chef avec suffisemment de mémoire (par exemple 8Go). Attribuer un domaine au serveur Chef. S'assurer que le domaine a une zone de domaine, un enregistrement NS et un enregistrement A/AAA correspondants. S'assurer que le nom d'hôte du serveur Chef est identique à son nom de domaine. Le serveur Chef créera automatiquement des certificats SSL basés sur le nom d'hôte.

## Workstation & Node

- Une machines poste de travail Workstation et l'autre un nœud qui sera géré par Chef.

## Mise à jour

- La mise à jour sur les différentes machines avant l'installation des composant de Chef :

```
| apt update && apt upgrade
```

# **Le serveur Chef**

## **Installation d'un serveur Chef**

- Le serveur Chef est le centre d'interaction entre tous les postes de travail et les nœuds sous la gestion Chef.
- Les modifications apportées au code de configuration sur les postes de travail sont transférées vers le serveur Chef, puis extraites par le client Chef du nœud pour appliquer les configurations.

# Le serveur Chef

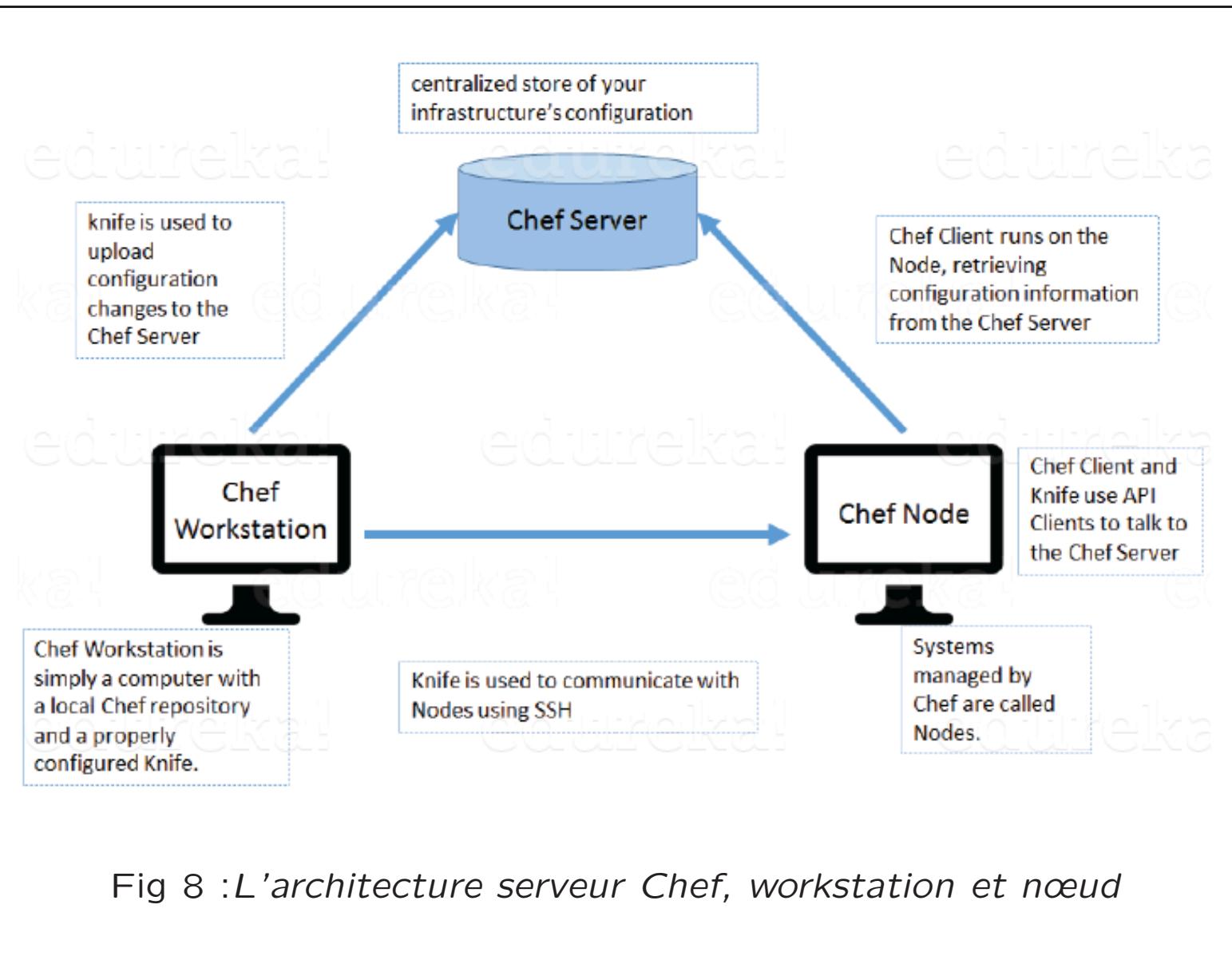


Fig 8 :L'architecture serveur Chef, workstation et noeud

# Le serveur Chef

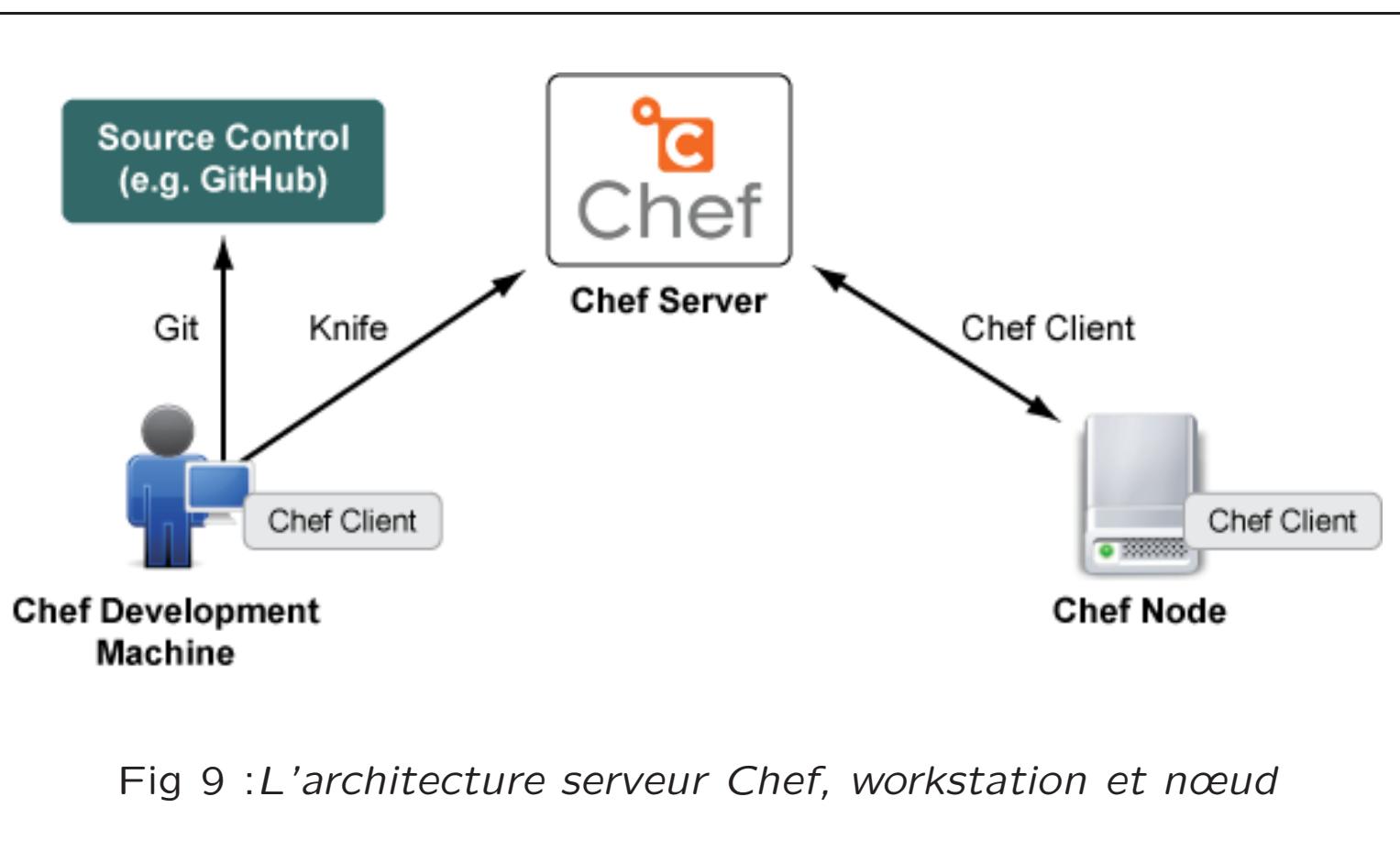


Fig 9 :L'architecture serveur Chef, workstation et nœud

# Le serveur Chef

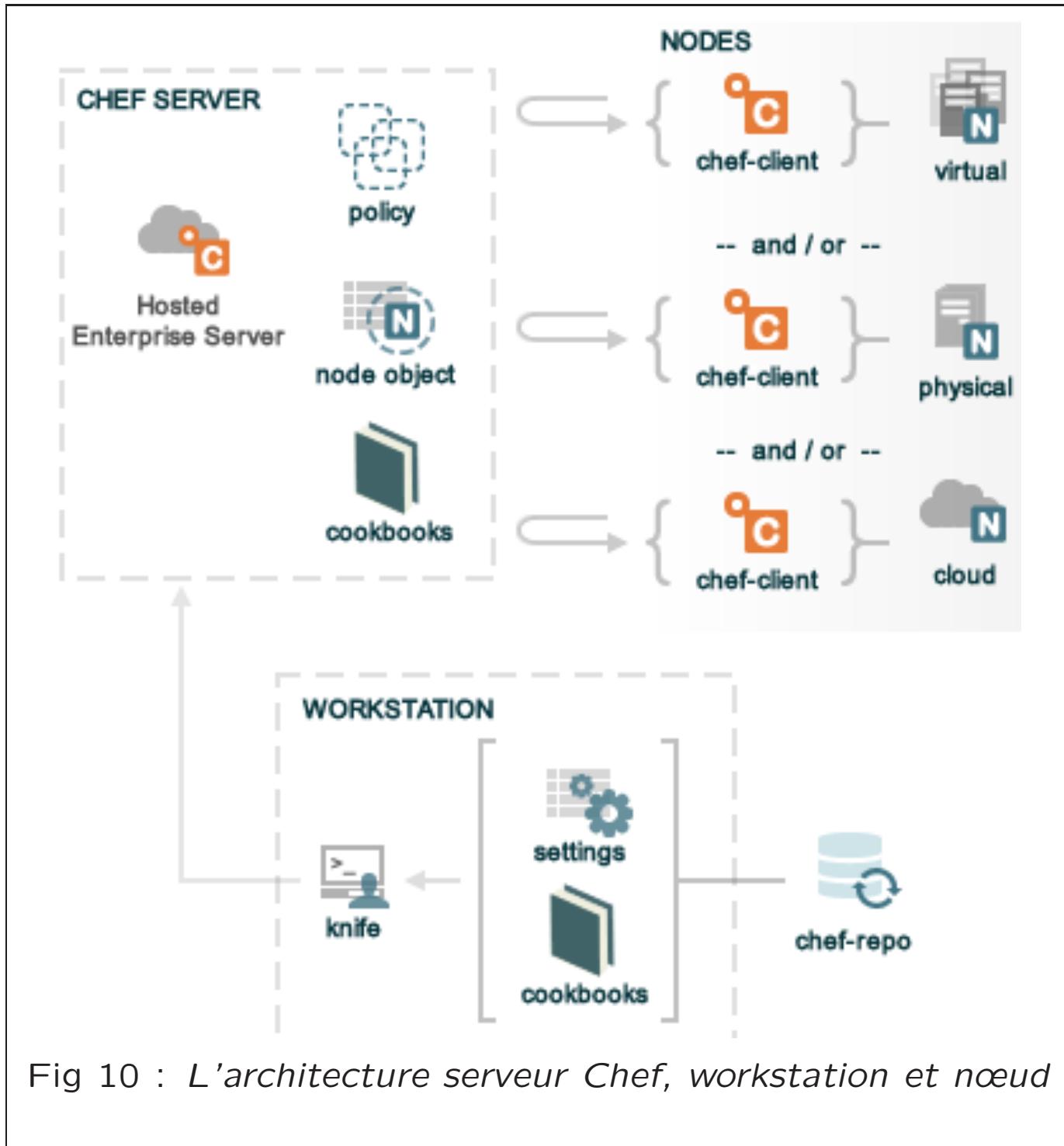


Fig 10 : L'architecture serveur Chef, workstation et noeud

# Chef

## Architecture et composants de Chef

- Le système Chef est défini par les rôles que chaque machine ou ressource joue dans le processus de déploiement :
  - **Chef Server** : Il s'agit de l'emplacement central qui stocke les **cookbooks** de configuration, et de recettes et les définitions du **Workstation** et de **node**.  
C'est la machine centrale que toutes les autres machines de l'organisation utiliseront pour la configuration du déploiement.

# Chef

## Architecture et composants de Chef

- **Chef Nodes:** les nœuds Chef sont les cibles de déploiement configurées par Chef. Chaque nœud représente un environnement de machine séparé et confiné qui peut être sur du matériel physique ou virtuel.

Ces environnements de système d'exploitation contiennent chacun une application *client Chef* pouvant communiquer avec le *serveur Chef*.

- **Chef Workstation :** Les postes de travail de Chef sont où les détails de configuration de Chef sont créés ou édités. Les fichiers de configuration sont ensuite transmis au *serveur Chef*, où ils seront disponibles pour être déployés sur tous les nœuds.

# Chef

## Architecture et composants de Chef

- La configuration de ces différents composants vous permet d'avoir plusieurs postes de travail ou Workstations et nœuds ou nodes. Les nœuds peuvent être configurés dès qu'ils sont en ligne et connectés au serveur.
- Bien que cette description donne l'impression qu'il s'agit d'entités distinctes, il est possible qu'une machine remplisse deux ou tous ces rôles. Il existe un outil appelé chef-solo qui vous permet de renoncer à l'utilisation d'un serveur et de fonctionner en configurant la machine sur laquelle il est installé.

# Chef

## Chef Server

- Le serveur est le point de contrôle central auquel accèdent toutes les autres machines Chef, que ce soit en tant que client ou gestionnaire. Il s'agit essentiellement d'un dépôt (*repository*) ou d'une base de données de tous les détails de configuration.
- Il gère les connexions et les autorisations des nœuds et des postes de travail Workstations et organise les données de façon à être extraites facilement par les clients. Le serveur peut également inclure une interface Web pour gérer ou configurer certains détails.

# Chef

## Chef Node

- Un nœud peut être une machine physique ou virtuelle. Ses seules exigences sont qu'il a accès au réseau et peut communiquer avec le serveur Chef. L'utilisateur qui exécute le logiciel Chef doit également être en mesure d'installer le logiciel et d'apporter des modifications au système.
- Chaque nœud communique avec le serveur à l'aide d'une application appelée `chef-client`. Cela permet de retirer les données du serveur et d'exécuter les étapes de configuration nécessaires pour mettre le nœud dans son état final. Le programme `chef-client` et le serveur Chef communiquent grâce à l'utilisation de l'authentification par clé RSA.

# Chef

## Chef Node

- `chef-client` utilise un outil appelé `ohai` pour obtenir des statistiques sur le nœud. Ils sont utilisés pour configurer certains détails de configuration et remplir les variables contenues dans les fichiers.

# Chef

## Chef Workstation

- Un poste de travail ou `Workstation` dispose des outils nécessaires pour créer et modifier les détails de configuration de l'un des nœuds disponibles et peut communiquer avec le serveur `Chef` pour les rendre disponibles.
- Un outil important pour gérer `Chef` sur un poste de travail est appelé `knife`. `knife` agit comme une passerelle dans laquelle vous pouvez configurer tout ce qui serait stocké sur le serveur.  
Il peut gérer des nœuds et des configurations et peut généralement être utilisé pour accéder au serveur d'une manière *spécifique à Chef*.

# Chef

## Chef Workstation

- Bien qu'il soit possible de se connecter au serveur avec SSH et d'apporter des modifications à toutes les données qu'il gère manuellement, cela n'adhère pas vraiment aux processus que Chef met en œuvre.
- Les configurations et les définitions créées et modifiées sur un poste de travail sont validées par le contrôle de version, puis transmises au serveur. Le dépôt ou *repository* s'appelle `chef-repo`. Il contient toutes les données nécessaires à la configuration du Chef.

# Chef

## Structure du dépôt Chef : chef-repo

- Chef gère ses informations de configuration et de dépendance sur un poste de travail ou *Workstation* dans une structure de répertoire spécifiée. Il est important de comprendre cette hiérarchie afin de créer efficacement des recettes et de valider les changements.
- Les fichiers de configuration du serveur doivent être conservés dans le dépôt ou *repository* de contrôle de version appelé *chef-repo*. Ceci est juste un répertoire qui contient les fichiers de Chef.
- Dans ce répertoire, nous pouvons trouver une structure qui ressemble à ceci :
  - *certificates/* : Contient les certificats SSL pouvant être associés aux clients pour l'authentification

# Chef

## Structure du dépôt Chef : chef-repo

- `chefignore` : Répertorie les fichiers et les répertoires de la structure qui ne doivent pas être inclus dans l'envoie ou `push` vers le serveur.
- `config/` : Contient l'un des deux fichiers de configuration du *repository* ou dépôt.
  - `rake.rb`: Définit des déclarations de variables pour la création de certificats SSL et quelques options générales.
- `cookbooks/` : Contient les `cookbooks` de recettes qui configurent l'infrastructure pour votre organisation.

# Chef

## Structure du dépôt Chef : chef-repo

- `data_bags/` : Contient diverses données pour votre configuration. Les data bags de données sont des sous-répertoires protégés qui contiennent des détails de configuration sensibles. Ils sont uniquement accessibles aux nœuds qui ont des certificats SSL correspondants et contiennent des fichiers JSON avec des détails de configuration.
- `environments/` : Contient les détails de déploiement de l'environnement.  
Chaque environnement qui diverge de l'environnement par défaut doit être défini dans ce répertoire.
- `Rakefile` : Ce fichier définit les tâches que Chef peut effectuer dans ses configurations.
- `roles/` : Contient les fichiers qui définissent les rôles pouvant être assignés aux nœuds.

# Chef

## Structure d'un fichier cookbook de Chef

- Dans le répertoire des cookbooks du chef-repo, les sous-répertoires définissent des cookbooks de recettes spécifiques pour les applications.
- Dans chaque répertoire de configuration d'application distinct est une structure qui définit comment ce service doit être installé et quelles modifications doivent être apportées pour le faire fonctionner correctement.
- Dans l'application, vous trouverez des fichiers et des définitions qui définissent comment une application doit être installée et configurée.

# Chef

## Le fichier metadata

- Les fichiers `metadata.rb` ou `metadata.json` contiennent des informations de métadonnées sur le service. Cela inclut des informations de base comme le nom du `cookbook` et la version, mais c'est aussi l'endroit où les informations de dépendance sont stockées.
- Si ce `cookbook` dépend d'autres `cookbooks` à installer, il peut les lister dans ce fichier et Chef les installera et les configurera avant le `cookbook` de recettes actuel.

## Le répertoire attributes

- Le répertoire `attributes` contient des définitions d'attribut qui peuvent être utilisées pour remplacer ou définir des paramètres pour les nœuds qui auront ce service.

# **Chef**

## **Le répertoire definitions**

- Le répertoire `definitions` contient des fichiers qui déclarent des ressources. Cela signifie que vous pouvez regrouper les fonctionnalités sous un seul titre.

## **Le répertoire files**

- Le répertoire `files` décrit comment Chef doit distribuer les fichiers sur le nœud sur lequel ce cookbook de recettes est déployé.

# Chef

## Le répertoire `recipes`

- Le répertoire `recipes` contient les *recettes* qui définissent comment le service doit être configuré. Les recettes sont généralement des fichiers qui configurent des aspects spécifiques du système.

## Le répertoire `templates`

- Le répertoire `templates` est utilisé pour fournir une gestion de configuration plus complexe. Vous pouvez fournir des fichiers de configuration entiers contenant des commandes et variables Ruby incorporées. Les variables peuvent être définies dans d'autres fichiers.

# Installation d'un serveur Chef

- À mesure que vos besoins d'infrastructure augmentent, la gestion manuelle de chaque serveur devient une tâche de plus en plus difficile. Cette difficulté est aggravée par l'exigence de reproductibilité, qui devient nécessaire en cas de défaillance d'un nœud ou si une mise à reproduction de configuration, *horizontal scaling*, est nécessaire.
- Les solutions de gestion de configuration sont conçues pour résoudre ces problèmes en transformant votre administration d'infrastructure à base de code. Au lieu d'effectuer des tâches individuelles sur un certain nombre de machines, ces outils vous permettent de valider vos exigences dans un emplacement central où chaque composant peut se connecter, dérouler sa configuration et l'appliquer.

# Installation d'un serveur Chef

- Nous procérons pour installer et mettre en place un serveur Chef centralisé qui stockera et servira les instructions de configuration et les informations de profilage des nœuds.
- Nous allons également mettre en place un poste de travail Workstation où l'administrateur peut travailler avec la base de code et modifier les caractéristiques de l'infrastructure. Nous suivrons cela en amorçant un nouveau nœud pour l'amener sous la gestion de l'écosystème Chef.

## Pré-requis

- La documentation de Chef nous indique que votre serveur Chef doit avoir au moins 4 cœurs et 4 Go de RAM. Il devrait également avoir un système d'exploitation 64 bits.

`https://docs-archive.chef.io/release/server_12-8/\install_server_pre.html#hardware-software`

# Installation d'un serveur Chef

- Le poste de travail `Workstation` et les nœuds ont peu d'exigences.

## Objectifs

- Le but est l'installation d'un serveur `Chef` centralisé pour stocker et servir nos données de configuration.
- Le poste de travail `Workstation` sera utilisé pour apporter des modifications, les télécharger sur le serveur et amorcer et gérer les nouveaux nœuds.
- Le nœud représente un seul serveur dans notre infrastructure.

# Installation d'un serveur Chef

## Accès par nom d'hôte

- Assurez-vous que le serveur est accessible par nom d'hôte. Une fois connecté au serveur sur lequel vous envisagez d'installer le serveur Chef, la première tâche à effectuer consiste à vérifier que le nom d'hôte du serveur est un nom de domaine complet (*FQDN*) ou une adresse IP résolvable. Vous pouvez utiliser une configuration par serveur DNS ou par le fichier /etc/hosts.

## Téléchargement du serveur

```
 wget https://packages.chef.io/files/stable/chef-server/12.18.14/ubuntu/\
18.04/chef-server-core_12.18.14-1_amd64.deb
```

# Installation d'un serveur Chef

## Installation du serveur

```
| sudo dpkg -i chef-server-core_*.deb
```

- Une fois l'installation terminée, vous devez appeler la commande `reconfigure`, qui configure les composants qui composent le serveur pour qu'ils fonctionnent ensemble dans votre environnement spécifique:

```
| sudo chef-server-ctl reconfigure
```

# Installation d'un serveur Chef

## Création d'un utilisateur & d'une organisation d'administration

- Nous pouvons le faire en utilisant l'action `user-create` de la commande `chef-server-ctl`. La commande nécessite un certain nombre de champs à passer au cours du processus de création. La syntaxe générale est:

```
mkdir /.chef  
chef-server-ctl user-create username first_name \  
last_name email password --filename /.chef/username.pem
```

- Nous inclurons cette information, et nous ajouterons également une option supplémentaire `-f`, à la fin, afin de spécifier un nom de fichier dans lequel afficher la clé RSA privée de notre nouvel utilisateur. Nous en aurons besoin pour l'authentification lors de l'utilisation de la commande de gestion `knife`.

# Installation d'un serveur Chef

## Création d'un utilisateur & d'une organisation d'administration

- Vous devriez avoir une clé privée dans un fichier nommé `admin.pem` par exemple dans votre répertoire actuel :

```
sudo chef-server-ctl user-create admin admin admin\  
admin@devops.lan adminpassword -f admin.pem  
sudo chef-server-ctl user-list
```

- Maintenant que vous avez un utilisateur, vous pouvez créer une organisation avec l'action `org-create`. Une organisation est simplement un regroupement d'infrastructure et de configuration au sein de Chef. La commande a la syntaxe g'nérale suivante :

```
chef-server-ctl org-create SHORTNAME LONGNAME\  
--association_user USERNAME
```

# Installation d'un serveur Chef

## Création d'un utilisateur & d'une organisation d'administration

- Le nom court est le nom que vous utiliserez pour vous référer à l'organisation depuis Chef. Le nom long est le nom réel de l'organisation. Le `--association_user` spécifie le nom d'utilisateur qui a accès à l'administration de l'organisation.
- L'option `-f` est utilisée afin que nous puissions spécifier le nom du fichier pour placer la clé privée. La clé qui sera créée est utilisée pour valider de nouveaux clients dans le cadre de l'organisation jusqu'à ce qu'ils puissent obtenir leur propre clé de client unique.

```
sudo chef-server-ctl org-create devops "DevOps Org"\  
  --association_user admin -f devops-validator.pem  
sudo chef-server-ctl org-list
```

# Installation d'un serveur Chef

- À cette étape de l'installation, vous devriez avoir deux fichiers de clé .pem dans votre répertoire d'installation. Dans notre cas, ils seront nommés `admin.pem` et `devops-validator.pem`.
- Nous devrons nous connecter à ce serveur et télécharger ces clés sur notre poste de travail `Workstation`. Pour l'instant, l'installation de notre serveur `Chef` est terminée.

# Configuration d'une Workstation Chef

- Maintenant que notre serveur Chef est opérationnel l'étape suivante consiste à configurer un poste de travail **Workstation**. La coordination et la configuration réelles de l'infrastructure n'ont pas lieu sur le serveur Chef. Ce travail est effectué sur un poste de travail qui télécharge ensuite les données sur le serveur pour modifier l'environnement Chef.

## Cloner le repository Chef

- La configuration de Chef de votre infrastructure est gérée dans une structure de fichiers hiérarchique connue collectivement sous la forme d'un *repository Chef* ou *repo Chef*.

# Configuration d'une Workstation Chef

- La structure générale peut être trouvée dans un dépôt GitHub fourni par l'équipe de Chef. Nous utiliserons git pour *cloner* ce repo sur notre poste de travail Workstation afin de servir de base au *repository* ou dépôt de Chef de notre infrastructure.

```
wget https://packages.chef.io/files/stable/chef-workstation/
0.2.43/ubuntu/18.04/chef-workstation_0.2.43-1_amd64.deb
chef generate repo chef-repo
mkdir /chef-repo/.chef
cd chef-repo
ssh-keygen -b 4096
ssh-copy-id utilisateur@IPServeurChef
sudo apt-get install git
cd ~
git clone https://github.com/chef/chef-repo.git
```

- La structure de *repository* Chef de base sera copiée dans un répertoire appelé **chef-repo** dans votre répertoire personnel.

# Configuration d'une Workstation Chef

## Mettre le repo Chef sous contrôle de version

- Les configurations créées dans le repo Chef lui-même sont mieux gérées dans un système de contrôle de version de la même manière que vous géreriez le code. Depuis que nous avons cloné le repo ci-dessus, un repo git a déjà été initialisé.
- Pour configurer votre station de travail pour de nouveaux *commits*, il faut définir le nom et l'adresse e-mail que git utilisera pour marquer les validations que vous effectuez. C'est une exigence pour git d'accepter les commits. Nous définissons ceci globalement de sorte que tout repo git que nous créons utilise ces valeurs :

```
| git config --global user.name "DevOps"  
| git config --global user.email "ajaber@openepo.net"
```

# Configuration d'une Workstation Chef

## Mettre le repo Chef sous contrôle de version

- Ensuite, nous indiquons à git d'ignorer toute information contenue dans le répertoire `~/chef-repo/.chef`. dans le fichier `.gitignore` afin que git ne stocke pas les données qui ne devraient pas être exposées à d'autres personnes :

```
| echo ".chef" >> ~/chef-repo/.gitignore
```

- Ensuite, ajoutez tous les fichiers modifiés dans la zone de transfert en cours :

```
| cd ~/chef-repo  
| git add .
```

# Configuration d'une Workstation Chef

## Mettre le repo Chef sous contrôle de version

- Il s'agit de faire un commit de changements. Nous utiliserons l'option `-m` pour spécifier un message de validation en ligne décrivant les changements que nous effectuons :

```
git commit -m "Excluding the ./chef directory from\\
version control"
```

- Notre repo Chef est maintenant sous contrôle de version. Comme nous créons des configurations pour notre infrastructure, nous pouvons utiliser les deux commandes ci-dessus pour garder notre repo git à jour.

# Installation de ChefDK

- Nous devons installer le kit de développement Chef, une suite de logiciels conçus pour les stations de travail Workstation Chef.
- ChefDK inclut de nombreux utilitaires qui seront utiles lors de la conception de configurations pour votre infrastructure. L'outil qui nous intéresse à ce stade est la commande `knife`, qui peut communiquer et contrôler à la fois le serveur Chef et tous les clients Chef.

```
| https://downloads.chef.io/chefdk#/
```

- L'installation et vérification comme suit :

```
sudo dpkg -i chefdk_*.deb  
chef verify
```

```
echo 'eval "$(chef shell-init bash)"' >> ~/.bash_profile  
source ~/.bash_profile
```

# Chef Workstation

## Télécharger les clés d'authentification

- Il s'agit de télécharger les clés d'authentification sur le poste de travail `Workstation`
- À cette étape, votre poste de travail dispose de tous les logiciels nécessaires pour interagir avec un serveur `Chef` et composer des configurations d'infrastructure. Cependant, il n'est pas encore configuré pour interagir avec votre serveur `Chef` et votre environnement.
- Vous allez télécharger les informations d'identification que vous avez créées sur le serveur `Chef`.

# Chef Workstation

## Télécharger les clés d'authentification

- Vous pouvez utiliser une méthode de votre choix pour télécharger la clé utilisateur et la clé de validation d'organisation créées sur le serveur Chef. Avant cela, nous allons créer le répertoire caché où nous allons stocker ces fichiers :

```
| mkdir ~/chef-repo/.chef
```

# Chef Workstation

## Configuration de gestion d'environnement Chef

- Configuration de `knife` pour gérer votre environnement Chef :

À ce niveau où vous avez vos informations d'identification Chef disponibles sur votre poste de travail Workstation, nous pouvons configurer la commande `knife` avec les informations dont elle a besoin pour se connecter et contrôler votre infrastructure Chef.

# Chef Workstation

- Ceci est réalisé grâce au fichier `knife.rb` que nous placerons dans le répertoire `~/chef-repo/.chef` avec nos clés, comme suit :

```
current_dir = File.dirname(__FILE__)
log_level          :info
log_location        STDOUT
node_name           "admin"
client_key          "#{current_dir}/admin.pem"
validation_client_name "devops-validator"
validation_key       "#{current_dir}/devops-validator.pem"
chef_server_url     "https://chef.devops.lan/organizations/devops"
syntax_check_cache_path "#{$ENV['HOME']}/.chef/syntaxcache"
cookbook_path        ["#{current_dir}/../cookbooks"]
```

- Les éléments suivants correspondent :

- `node_name` : spécifie le nom que `knife` utilisera pour se connecter à votre serveur Chef et correspondant à votre nom d'utilisateur.

# Chef Workstation

- `client_key` : Le nom et le chemin d'accès à la clé utilisateur que vous avez copiée depuis le serveur Chef. Nous pouvons utiliser `#{{current_dir}}` pour renseigner le chemin si la clé se trouve dans le même répertoire que le fichier `knife.rb`.
- `validation_client_name` : C'est le nom du client de validation que `knife` utilisera pour amorcer de nouveaux nœuds. Cela prendra la forme du nom abrégé de votre organisation, suivi de `-validator`

# Chef Workstation

## Test de configuration

```
cd ~/chef-repo  
knife client list  
  
knife ssl fetch  
knife client list
```

- À ce niveau, votre poste de travail Workstation est maintenant configuré pour contrôler votre environnement Chef.

# Chef Workstation

## Un nouveau nœud

- Avec le serveur et le poste de travail Chef configurés, nous pouvons commencer à utiliser Chef pour configurer de nouveaux serveurs dans notre infrastructure.
- Il s'agit d'un processus d'amorçage ou de *bootstrapping* dans lequel l'exécutable du client Chef est installé sur le nouvel ordinateur et la clé du validateur organisationnel est également transmise.
- Le nouveau nœud contacte alors le serveur Chef avec la clé de validation et, en retour, reçoit sa propre clé client unique et toute configuration qui lui a été affectée. Ce processus met le nouveau serveur dans son état initial et le configure pour toute gestion à venir.

# Chef Workstation

## Un nouveau nœud

```
knife bootstrap nodeNameOrIP -N testing -x demo\  
-P password --sudo --use-sudo-password  
knife bootstrap nodeNameOrIP -x root -A
```

- Une fois que votre nouveau nœud est amorcé, vous devriez avoir un nouveau client :

```
| knife client list
```

- Vous pouvez utiliser la procédure ci-dessus pour configurer de nouveaux clients Chef quelque soit le nombre de nouveaux serveurs.

# Exemple de cookbook

- Les cookbooks de recettes ou configuration sont organisés dans une structure de répertoires complètement autonome. Il existe de nombreux répertoires et fichiers différents utilisés à différentes fins, par exemple :
  - **Recipes** : Une recette principale du cookbook. Un cookbook de recettes peut contenir plus d'une recette ou dépendre de recettes extérieures. Les recettes sont utilisées pour déclarer l'état de différentes ressources.
  - **Attributes** : Les attributs dans Chef sont essentiellement des paramètres. Considérez-les comme de simples paires clé-valeur pour tout ce que vous pourriez vouloir utiliser dans votre cookbook de recettes.

# Exemple de cookbook

- **Files** : Le sous-répertoire `files` du cookbook contient tous les fichiers statiques que nous allons placer sur les nœuds qui utilisent le cookbook de recettes.
- **Templates** : Les modèles sont similaires aux fichiers, mais ils ne sont pas statiques. Les fichiers modèles se terminent par l'extension `.erb`, ce qui signifie qu'ils contiennent du Ruby.
- **Metadata.rb** : Le fichier `metadata.rb` est utilisé pour gérer les métadonnées d'un package. Ceci inclut le nom du paquet, une description entre autres.

# Création d'un Cookbook : nginx

```
cd ~/chef-repo
knife cookbook create nginx
cd cookbooks/nginx
ls
```

## Création d'un recipe

```
cd recipes
ls
nano default.rb

package 'nginx' do
  action :install
end
service 'nginx' do
  action [ :enable, :start ]
end

cookbook_file "/usr/share/nginx/www/index.html" do
  source "index.html"
  mode "0644"
end
```

## Création du fichier index.html

```
cd ~/chef-repo/cookbooks/nginx/files/default
nano index.html
<html>
<body>
<h1>Bonjour le monde</h1>
</body>
</html>
```

## Création d'un Cookbook : apt

- L'utilisation du gestion de *packages* est nécessaire à mettre en place pour la gestion de configuration : procédez comme le cookbook `nginx` pour le mettre en place.

# Ajout du cookbook au nœud

- Maintenant que nos cookbooks de base sont complets, nous pouvons les télécharger sur notre serveur Chef.

```
knife cookbook upload apt
knife cookbook upload nginx
# ou bien knife cookbook upload -a
```

- Nous pouvons modifier la liste de lancement de nos nœuds :

```
knife node list
knife node edit NodeName

export EDITOR=nano
```

# Ajout du cookbook au nœud

- Nous pouvons ajouter notre cookbook Nginx à ce tableau en utilisant le format suivant :

```
"recipe[name_of_recipe]"  
# donc dans le fichier rajouter :  
...  
"run_list": [  
    "recipe[nginx]"  
]
```

- Maintenant, nous pouvons nous connecter en SSH à notre nœud et exécuter le client Chef. Cela entraînera le client à vérifier dans le serveur Chef. Une fois cela fait, il verra la nouvelle liste de lancement qui lui a été assignée.

- Il s'agit de se connecter au nœud et exécuter :

```
| sudo chef-client
```

- Le résultat sera affiché sur le navigateur à l'adresse de votre nœud.