1 Le travail réalisé

1.1 Le script de réservation.

Nous avons réalisé un script qui nous permet de reserver des machines sans avoir à renseigner les commandes de l'outil OAR. Nous demandons à la personne souhaitant réserver des machines de rentrer le nom de la réservation, le nombre de machines et le temps de réservation de celles-ci. Nous passons ensuite ces arguments à oarsub avec les options requises.

```
oarsub -I -t deploy -l {"type='kavlan-local'"}/vlan=1+/nodes=$nbr_nodes,
walltime=$tmps -n "$nom"
```

Figure 1: La commande que nous passons en paramètre.

1.2 Le script d'installation sur les machines.

Ce script nous permet d'automatiser l'installation de notre cluster une fois les machines réservées.
Nous commençons l'installation de notre cluster par l'activation du DHCP de Grid5000.
Nous installons ensuite une gem Ruby pour configurer et installer le service DHCP du VLAN qui nous concerne, configuré avec notre configuration de machines. Nous allons ensuite déployer l'image ISO de

```
##Activation du DHCP g5k
kavlan -e
##Récupération du numéro VLAN
jobid=‘oarstat | grep $USER | cut -d' ' -f1‘
vlan=‘kavlan -V -j $jobid ‘
site=‘uname -n | cut -d"." -f2‘
##Génération du fichier de configuration du DHCP
export GEM_HOME=/home/$USER/.gem/ruby/1.8/
gem install ruby-ip --no-ri --no-rdoc --user-install #&>/dev/null
chmod +x ./projet/install/gen_dhcpd_conf.rb #&>/dev/null
./projet/install/gen_dhcpd_conf.rb --site $site --vlan-id $vlan #&>/dev/null
mv dhcpd-kavlan-$vlan-$site.conf $puppet_modules/dhcp/files/dhcpd.conf
```

Figure 2: On active le DHCP du Grid5000 pour créer notre fichier de configuration, qui servira pour notre propre service DHCP.

Debian que nous fournit le Grid5000 pour installer les différents services sur les noeuds. Nous affichons auparavant la liste des noeuds que nous avons réservés, avec le suffixe du vlan.

```
echo "-=-=-"
echo "Déploiement de l'environnement Linux, Debian Squeeze 64bit,
sur toutes les machines réservées (Peut prendre plusieurs minutes)."
kadeploy3 -e squeeze-x64-base -f $OAR_FILE_NODES
--vlan $vlan -d -V4 -k $HOME/.ssh/id_dsa.pub &>/dev/null
echo "-=-=-"
```

Figure 3: On utilise la commande kadeploy3 pour déployer toutes nos machines sous Debian Squeeze.

Vient ensuite l'ensemble des tests sur la configuration SSH du client qui a lancé le déploiement. Cela consiste en une vérification sur le fichier de configuration et le fichier authorized_keys, puis en une copie des clés de l'utilisateur sur toutes les machines concernées par le déploiement du cluster.

Nous créons ensuite les tunnels SSH et nous mettons à jour tous les dépôts. Cela nous permet d'installer ensuite le master puppet et de lui spécifier tous les clients qui vont recevoir des services réseau. Une fois le maître configuré, nous installons et nous configurons tous nos clients de telle sorte que chaque machine puisse recevoir la configuration qui lui est attribuée depuis le master.

```
taktuk -l root -s -m $node broadcast exec [ 'cat ~/.ssh/id_dsa.pub
>> ~/.ssh/authorized_keys' ]
```

Figure 4: Taktuk nous permet de réaliser l'insertion de la clé publique SSH dans le fichier authorized_keys à distance.

```
echo "Attribution des rôles effectués:"
echo "- "`sed -n '2 p' $puppet_clients `" : dhcp,"
echo "- "`sed -n '3 p' $puppet_clients `" : bind,"
echo "- "`sed -n '4 p' $puppet_clients `" : mysql,"
echo "- "`sed -n '5 p' $puppet_clients `" : nfs,"
echo "- "`sed -n '6 p' $puppet_clients `" : oar-server,"
echo "- "`sed -n '7 p' $puppet_clients `" : oar-frontend,"
echo "- "`sed -n '8 p' $puppet_clients `" : kadeploy."
```

Figure 5: On définit quel noeud va recevoir quel service, de manière à ce que le master puppet n'installe pas tout sur la même machine.

Enfin, nous terminons l'installation par la désactivation du vlan fourni par Grid5000 pour utiliser notre propre serveur DHCP, puis par la suppression du fichier temporaire utilisé.

```
taktuk -l root -s -f $puppet_clients broadcast exec [ dhclient ] &>/dev/null
taktuk -l root -s -f $list_users broadcast exec [ dhclient ] &>/dev/null
rm lol.tmp
```

Figure 6: Fin du déploiement.

1.3 Le script de configuration du serveur.

Ce script permet d'écrire toute la configuration du maître Puppet en se basant sur tous les services à installer. Pour commencer, nous configurons le fichier /etc/hostname pour établir le nom du réseau. Nous redémarrons ensuite le service de noms d'hôte. Nous ajoutons ensuite dans le fichier de configuration de Puppet le nom du serveur maître, car notre master puppet est aussi un agent. A cela se greffe une section contenant le nom de la machine qui validera les certificats, en l'occurence la machine maître, le tout placé dans une section. Nous avons également besoin de configurer les adresses ou les noms de nos machines clientes pour que Puppet puisse les retrouver.

```
#ajout des hosts clients
y=1
for role in "puppet" "dhcp" "dns" "mysql" "nfs" "oar-frontend" "oar-server" "kadeploy"
do
node=`sed -n $y"p" puppet_clients`
ip_node=`arp $node | cut -d" " -f2 | cut -d"(" -f2 | cut -d")" -f1`
echo "$ip_node $role.ptut-grid5000.lan" >> /etc/hosts
y=$(($y+1))
done
```

Figure 7: L'ajout des machines se fait à l'aide d'une boucle.

Nous utilisons le même type de boucle pour définir les noeuds dans le fichier nodes.pp. Enfin, nous terminons la configuration du master Puppet par la configuration du service de noms de domaine. La fin du programme permet de préparer l'installation de kadeploy en modifiant le fichier de configuration.

```
for machine in $(cat list_users)
do
ip_machine=‘arp $machine | cut -d" " -f2 | cut -d"(" -f2 | cut -d")" -f1‘
echo "machine$i IN A $ip_machine" >>
/etc/puppet/modules/bind/files/db.ptut-grid5000.lan
ip_oct_4=‘echo $ip_machine |cut -d"." -f4 ‘
echo "$ip_oct_4 IN PTR machine$i.ptut-grid5000.lan." >>
/etc/puppet/modules/bind/files/db.revers
 echo "marocco-$i.ptut.grid5000.fr $ip_machines marocco" >>
 /etc/puppet/modules/kadeploy/files/confs/nodes
i=$(($i+1))
done
```

Figure 8: Une des nombreuses commandes d'association pour le DNS.

1.4 Le script de configuration des clients.

Contrairement au master, le client puppet n'a besoin que de connaître que deux adresses : celle du master puppet, et celle de son propre client.

1.5 Les modules Puppet.

Le sujet principal du projet était de déployer l'intégralité du cluster au travers de Puppet. Une fois que l'infracstructure est installée, nous allons maintenant déployer les différents services à l'aide de modules que nous avons créés. Pour éviter les répétitions, nous nous contenterons d'étudier les modules des outils spécifiques à Grid5000 et ceux qui ont une importance capitale.

Figure 9: L'arborescence des modules.

1.6 Le logiciel OAR.

Comme expliqué plus haut, OAR se décompose ici en deux parties : le serveur et la frontend.

Figure 10: Deux modules pour OAR : Le serveur, et la frontend.

Ces deux modules nécessitent un certain nombre de paramètres qui lui permettent de fonctionner :

- Une communication SSH qui permettent à la frontend de se connecter directement sur le serveur et inversement ;
- Le fichier de dépôt de logiciels d'OAR.

Nous copions donc l'intégralité des clés SSH créées pour l'occasion, ainsi que le fichier de configuration de notre utilisateur et le fichier de configuration du démon SSH. L'ensemble de cette configuration, déployée

à l'identique sur le frontend et le serveur, est définie par une classe située dans le fichier install.pp de chaque module OAR.

1.6.1 Le module OAR-frontend.

Il est configuré par deux programmes : les fichiers prologue et epilogue. Ils nécessitent absolument une configuration manuelle car vierges par défaut. Tout comme le serveur OAR, le frontend est configuré par deux scripts puppet, l'un permettant d'initialiser le module, l'autre se contentant de vérifier et d'adapter la configration sur le client Puppet.

Figure 11: Deux scripts d'installation pour chaque module OAR : L'initialisation, qui appelle l'installation.

1.6.2 Le module OAR-server.

Le serveur OAR dépend activement d'une base de données, en l'occurence ici une base mysql, afin de gérer les noeuds. Cette base est localisée et configurée dans le fichier oar.conf, que nous allons copier depuis le maître Puppet vers le client. Deux autres fichiers de configuration sont nécesaires pour le serveur OAR. Il s'agit de :

- Monika.conf, qui permet de montrer les jobs soumis par la frontend au serveur ainsi que l'état de chaque noeud ;
- drawgantt.conf, qui permet de présenter les différentes tâches dans le temps suivant un Diagramme de Gantt.

Le serveur OAR nécessite également plusieurs autres services :

- un serveur Web du type apache2 sur la machine déployée ;
- une base de données de type mysql ou postgre qui peut être sur une autre machine.

1.7 Le module Kadeploy3.

Ce module est installé par Puppet à l'aide de 5 fichiers de scripts Puppet. Cela permet de définir simplement ce qu'un client et un serveur doivent récupérer.

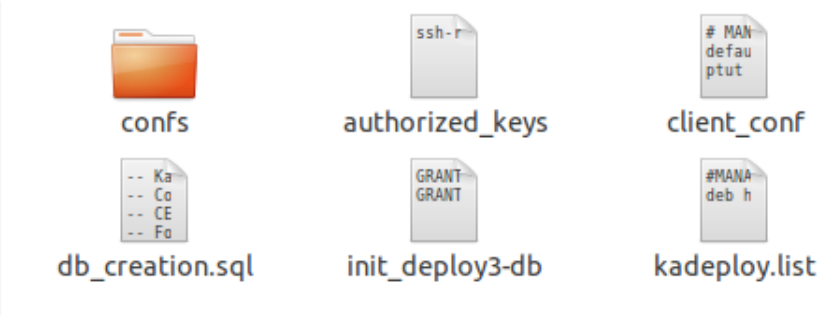

Figure 12: Un dossier contenant la configuration pour kadeploy3, et les fichiers nécessaires à sa bonne installation.

Kadeploy3 possède son propre dépôt qu'il faut ajouter pour pouvoir l'installer. il utilise également une base de données créé et configurée par Puppet à l'aide des fichiers db_ creation.sql et init_ deploy3-db. Nous rajoutons également le fichier authorized_ keys de manière à permettre une communication SSH. Dans le dossier conf des fichers utiles à la configuration de kadeploy par Puppet, nous retrouvons les fichiers qui permettent d'indiquer le nom du cluster, les partitions à appliquer, les commandes spécifiques pour certains nœuds, etc.

1.8 Le Puppet Dashboard.

C'est une fonctionnalité qui n'était pas prévue intialement dans le projet, mais plutôt considéré comme un extra. Il nécessite d'installer la clé publique pour le dépôt du Dashboard, qui n'est pas inclus par défaut dans les dépôts logiciels du Grid5000. Nous nous assurons le bon fonctionnement du module sur le nœud au travers de 4 scripts Puppet, qui permettent d'installer, de configurer et de lancer le puppet dashboard.

1.9 Les autres modules.

D'autres modules ont dû être définis pour assurer le bon fonctionnement des outils utilisés. Nous avons donc réalisé ces modules en se basant sur des exemples fournis par le site de Puppet, que nous avons réadapté avec une configuration adaptée à nos besoins. Cela nous a permis d'installer ces services très rapidement, avec la souplesse d'utilisation de ces modules.

2 Les problèmes rencontrés.

Quelques problèmes se sont présentés durant les 6 semaines de projet. beaucoup de problèmes posés avec l'utilisation de taktuk. Cela était principalement dû à la ocnfiguration spécifique des nœuds dans un VLAN, nous demandant alors de créer un tunnel SSH avec la machine locale pour permettre d'installer nos services.