L'offre EC2 d'Amazon, des machines dans le nuage

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 9 novembre 2011

https://www.bortzmeyer.org/amazon-cloud.html

Vous avez remarqué ? Tout le monde est dans les nuages, désormais. Ou plutôt dans le nuage. C'est-à-dire qu'on met ses machines à l'extérieur de l'entreprise. Avec un peu de marketing, ça peut passer pour une révolution. Alors, plutôt que de me lancer dans un long article sur le "*Cloud*" en général, je vais me contenter de présenter le service EC2 d'Amazon, pas seulement parce que j'en suis assez content, mais aussi parce que cette offre ne m'a pas semblé claire du tout lorsque j'ai commencé à l'évaluer, puis à l'utiliser, et que cela vaut donc la peine d'écrire ce que j'ai compris.

Donc, le principe d'EC2 ("*Elastic Compute Cloud*", un des composants d'AWS, "*Amazon Web Services*"), c'est qu'Amazon possède un grand nombre de machines physiques dans des hôtels d'ordinateurs à quelques endroits sur la planète, fait tourner sur ces machines physiques des machines virtuelles (apparemment en utilisant Xen). Vous pouvez, si vous êtes inscrit, créer et supprimer à la demande de telles machines virtuelles. Vous vous connectez ensuite par SSH sur ces machines et les administrez comme vous voulez (Amazon fournit apparemment du Windows mais j'ai juste testé leur offre Linux). En jargon "*cloud*" on appelle cela l'IaaS (alors que Gmail est du SaaS et Google App Engine du PaaS). En gros, avec l'IaaS, on est root et on fait ce qu'on veut de sa machine.

Une des innovations d'Amazon est que la gestion des machines virtuelles (création, destruction, etc) peut se faire via une API. De nombreux logiciels ont donc été développés pour l'automatisation du processus. En gros, vous avez une application sur le Web, vous commencez avec une seule instance (dans la terminologie EC2, une instance est une machine virtuelle), et vous pouvez rajouter, via une interface de votre choix, ou bien via un programme à vous, des instances si la charge augmente, et les supprimer lorsqu'elle diminue (si votre application fonctionne bien en mode distribué...).

Mais combien ça coûte ? Bonne question, surtout qu'on ne peut pas dire que cela soit parfaitement clair. Le principe est qu'on ne paie rien si on ne fait rien. Pas d'abonnement minimum, les périodes d'inactivité permettent d'avoir une facture de zéro dollar. Ensuite, on paie ce qu'on utilise et on paie vraiment tout. L'heure de fonctionnement d'une machine est facturé (à un prix qui dépend de la taille de la machine), mais aussi le stockage permanent (les disques de la machine virtuelle disparaissent à son arrêt), le débit sur le réseau, etc. Vous pouvez voir tous les prix en ligne <http://aws.amazon.com/ec2/pricing/>. Un outil sympathique pour tenter d'estimer ses coûts est le calculateur <http:

//calculator.s3.amazonaws.com/calc5.html>. Pour comprendre ces tarifs, rappelez-vous que les données stockées sur le « disque » de la machine virtuelle disparaissent avec elles. Vous serez donc sans doute obligés d'acheter du stockage, soit sous forme de disques traditionnels, les EBS ("*Elastic Block Store*"), soit sous forme d'un autre service nuagique d'Amazon, S3 ("*Simple Storage Service*").

Le résultat peut être assez cher (surtout le stockage sur les disques). EC2 n'est **pas** pour les budgets serrés. Toutefois, si vous comparez avec une autre solution, n'oubliez pas de **tout** prendre en compte. Par exemple, si vous comparez un petit serveur Web hébergé chez EC2 (en octobre 2011, vous arrivez facilement à 40 $ par mois) à une machine stockée chez vous et connectée en ADSL, n'oubliez pas d'intégrer des choses comme le coût de l'électricité...

Mais le point le plus dangereux avec EC2, c'est le fait que la facture n'est pas bornée (je n'ai en tout cas pas trouvé de moyen de placer une limite). Si vous lancez une grosse machine pour un essai, que vous oubliez de l'éteindre, et que vous y pensez trois jours après, vous vous retrouvez avec une facture de 70 $... Ce problème est d'autant plus sérieux que la plate-forme EC2 a été conçue pour l'automatisation : si votre programme qui crée des machines virtuelles lorsque c'est nécessaire a une bogue, et se met à créer des dizaines d'instances, votre comptable n'appréciera pas... J'ai vu une fois une image EC2 qui créait automatiquement des volumes EBS (des disques) pour ses besoins et ne les détruisait pas à l'arrêt de la machine virtuelle. La surprise est désagréable.

Bref, il est nécessaire de surveiller de près sa consommation (Amazon fournit une excellente page Web pour cela, dans la console d'AWS - "*Account Activity*").

Bien, maintenant, si vous avez envie d'essayer, créez-vous un compte sur <http://aws.amazon.com/account/>. Il va y avoir des vérifications, notamment une vérification téléphonique où il faudra taper le PIN dicté au téléphone (avec un écho terrible). Une fois que c'est fait, et que votre compte marche, rappelez-vous mon conseil : gardez un œil sur la facturation !

D'abord, il va falloir quelques éléments de sécurité (lettres de créance, "*access credentials*") à présenter pour vous authentifier. Le mot de passe de votre compte ne permet que l'accès à la console Web. Pour les autres activités, il faut d'autres éléments de sécurité. Vous pourrez en créer plusieurs, de manière à déléguer certaines tâches à diverses personnes ou programmes. Mais gardez-les bien secrets ! Quelqu'un qui met la main dessus peut, non seulement accéder à vos machines mais surtout créer des ressources AWS que vous devrez payer.

Donc vous créez ces éléments de sécurité (mots de passe - Amazon dit "*access keys*", certificats X.509 et clés SSH - "*key pairs*", ces dernières pouvant désormais être générées chez vous et chargées chez Amazon <http://alestic.com/2010/10/ec2-ssh-keys>) via la console <https://aws-portal.amazon.com/gp/aws/developer/account?ie=UTF8&action=access-key> (à partir d'ici, un certain nombre de liens ne marcheront que si vous avez un compte et êtes logué). Pour les clés cryptographiques, stockez bien la partie privée en sécurité sur votre disque dur, Amazon ne la garde pas.

Vous pouvez désormais créer une première machine virtuelle. Il existe plusieurs moyens pour cela. Le plus simple pour commencer est la console Web. On choisit d'abord une image (Amazon dit AMI pour "*Amazon Machine Images*"), c'est-à-dire un cliché d'un système d'exploitation complet, avec ses applications et ses réglages, qui va s'exécuter sur la machine virtuelle. Un des avantages d'EC2 est qu'il existe d'innombrables AMI. Chaque communauté de développeurs a créé la sienne et fournit donc une véritable "*appliance*" logicielle prête à l'emploi. Attention, toutefois, ces AMI ne sont pas forcément validées et vous ne devez donc exécuter que celles en qui vous avez, pour une raison ou une autre, confiance. Un exemple d'AMI développées par la communauté est donné par les AMI Ubuntu d'Alestic

<http://alestic.com/>. Quant à Debian, il faut aller voir du côté du groupe Debian EC2 <http://groups.google.com/group/ec2debian>.

Pour commencer, on peut se limiter à celles fournies par Amazon. Le choix est limité mais c'est un début. Par exemple, pour les systèmes à base de Linux, Amazon ne fournit officiellement que des images Red Hat. On choisit l'une d'elles et on la lance. Via l'interface Web, il faudra répondre à quelques questions, notamment le nom de la "*key pair*" (créée à l'étape précédent) à utiliser. Une minute plus tard, la console montre que l'instance est lancée ("*running*") et nous donne le nom public de la machine. On peut alors s'y connecter en SSH :

```
% slogin -l root -i test-a.pem ec2-46-137-46-53.eu-west-1.compute.amazonaws.com
```

(la "*key pair*" utilisée se nommait `test-a` et la partie privée était stockée en `test-a.pem`.) Le nom d'utilisateur (ici `root`) dépend de l'AMI utilisée (ce n'est pas forcément `root` par exemple, sur celles de CloudBioLinux <http://cloudbiolinux.org/>, une AMI faite pour la bio-informatique, c'est `ubuntu`).

Si vous n'arrivez pas à vous connecter, d'abord soyez patient (les machines virtuelles ne démarrent pas instantanément et le serveur SSH n'est pas prêt tout de suite) mais vérifiez aussi que la politique de sécurité est correcte. Vous créez votre machine avec un certain "*security group*" et ce groupe est la politique du pare-feu qui protège votre machine. S'il n'autorise pas SSH, la machine tourne mais vous ne pourrez pas la joindre.

Si vous utilisez, non pas le client SSH OpenSSH mais le client PuTTY sur Windows, rappelez-vous qu'il utilise un format de clé différent de celui fabriqué par Amazon. Si vous avez le message "*Unable to use key file*", il faut donc d'abord convertir les clés avec Puttygen <https://forums.aws.amazon.com/thread.jspa?threadID=32083>.

N'oubliez pas de terminer votre machine ensuite ! Rappelez-vous qu'EC2 est cher pour les distraits. Il est prudent de vérifier de temps en temps qu'on n'a pas oublié des machines ou, surtout, des disques EBS. La facture à la fin du mois peut être une mauvaise surprise (pour la petite histoire, la mise au point de cet article, avec vérifications et tout ça, a coûté dans les cinq dollars ; de nombreux démarrages et arrêts de machines virtuelles ont été nécessaires).

Et si on ne veut pas utiliser l'interface Web ? C'est justement un des principaux avantages d'EC2 : il existe une API officielle et documentée et des tas d'outils ont été créés pour en profiter et permettre d'automatiser la gestion des machines virtuelles. C'est par exemple le cas de l'extension Firefox Elastic Fox <http://aws.amazon.com/developertools/9302537431253167>, très agréable.

Il existe de la même façon des interfaces graphiques à EC2 pour tous les systèmes (je n'ai pas testé sur mon Android, la seule application étant payante).

Continuons avec les outils fournis par Amazon, les "*EC2 API tools*" <http://aws.amazon.com/developertools/351>. Ils sont disponibles sous forme de paquetage pour beaucoup de systèmes, par exemple Ubuntu, où il suffit d'un `aptitude install ec2-api-tools`. Ils sont écrits en Java et offrent un grand nombre de commandes CLI commençant par `ec2-`. Par exemple, `ec2-run-instances` permet de créer une instance (une machine virtuelle) et `ec2-terminate-instances` de la terminer. Avant tout, il faut être autorisé. Les "*EC2 API tools*" imposent apparemment les certificats X.509 (les autres lettres de créance possibles ne semblent pas utilisables avec ces programmes). On les crée depuis l'interface Web (nul besoin d'une signature par une AC, je vous rassure), on met deux fichiers en local (ne perdez pas la clé privée, Amazon n'en garde pas de copie) et on utilise des variables d'environnement (ou bien des options sur la ligne de commande) pour indiquer où ces fichiers se trouvent. On peut ensuite utiliser les commandes :

```
% export EC2_CERT=cert-OAT5EB2CMK5JF2SFJYNOONSLFJDSVEW2.pem
% export EC2_PRIVATE_KEY=pk-OAT5EB2CMK5JF2SFJYNOONSLFJDSVEW2.pem

% ec2-describe-regions
REGION  eu-west-1       ec2.eu-west-1.amazonaws.com
REGION  us-east-1       ec2.us-east-1.amazonaws.com
REGION  ap-northeast-1  ec2.ap-northeast-1.amazonaws.com
REGION  us-west-1       ec2.us-west-1.amazonaws.com
REGION  ap-southeast-1  ec2.ap-southeast-1.amazonaws.com
```

Les régions, affichées ci-dessus, permettent de répartir les risques : les différentes régions ne partagent **rien** et ne doivent normalement donc pas tomber en panne en même temps. Si on a des serveurs dans différentes régions, on est normalement à l'abri de pas mal de problèmes. En contrepartie, comme les régions ne partagent rien, une AMI n'est que dans une seule région, comme les clés ou les autres ressources. C'est parfois un peu agaçant, mais c'est cohérent.

Continuons. Cette commande crée une instance :

```
% ec2-run-instances --region eu-west-1 ami-02103876
RESERVATION     r-277f2651      047928833755    default
INSTANCE        i-32dd217b      ami-02103876                    pending         0       m1.small2011-10-09T2
```

La commande renvoie une table qui contient notamment l'identificateur de l'instance (ici `i-32dd217b`). Mais la machine n'a pas encore démarré réellement (état "*pending*"). Je ne connais pas de moyen de bloquer la commande en attendant que la machine soit disponible. Il faut l'interroger régulièrement avec `ec2-describe-instances` :

```
% ec2-describe-instances i-32dd217b
RESERVATION     r-277f2651      047928833755    default
INSTANCE        i-32dd217b      ami-02103876    ec2-46-137-4-184.eu-west-1.compute.amazonaws.com \
      ip-10-227-3-65.eu-west-1.compute.internal       running \
      0               m1.small2011-10-09T20:22:24+0000        eu-west-1a ...
```

On voit qu'elle a fini par passer dans l'état "*running*", on peut alors s'en servir (le nom public dans le DNS est indiqué, c'est ici `ec2-46-137-4-184.eu-west-1.compute.amazonaws.com`). Une fois que c'est fait, et pour éviter les grosses factures, bien penser à arrêter la machine :

```
% ec2-terminate-instances i-32dd217b
INSTANCE        i-32dd217b      running shutting-down
```

Il y a des tas d'autres options à ces commandes comme :

```
% ec2-run-instances  --key  Stephane --region "us-east-1"  --instance-type m1.large ami-6011e409
```

Et, si on ne veut pas utiliser la région par défaut, et pourtant ne pas taper son nom à chaque fois :

```
% export EC2_URL=https://ec2.eu-west-1.amazonaws.com
```

Une fois qu'on a ces commandes, on peut tout automatiser, par exemple depuis un "*shell script*". En voici un qui crée une machine, y exécute les commandes spécifiées dans la variable `COMMANDS` et détruit la machine. Pour savoir quand la machine est prête, il la "*polle*" régulièrement : (en ligne sur `https://www.bortzmeyer.org/files/amazon-ec2-run-commands.sh`).

Autre exemple, un script qui lance une machine, y installe le moteur de rendu POV-Ray, lui fait calculer une scène et rapatrie le résultat. À la main, on aurait plus ou moins fait :

```
% scp -i id_test table.pov ec2-72-44-37-142.compute-1.amazonaws.com:tmp
% ssh -i id_test ec2-72-44-37-142.compute-1.amazonaws.com "(cd tmp; povray table.pov)"
% scp -i id_test ec2-72-44-37-142.compute-1.amazonaws.com:tmp/table.png /tmp
```

Ici, le script (en ligne sur `https://www.bortzmeyer.org/files/amazon-ec2-povray.sh`) fait tout cela automatiquement. Notez qu'il commence par installer POV-Ray sur la machine (je n'ai pas trouvé d'AMI publique ayant déjà POV-Ray), ce qui aurait aussi bien pu se faire en mettant l'installation de POV-Ray dans le `user_data` (script exécuté à la création) Voir par exemple CloudInit en <`http://stackoverflow.com/questions/6475374/how-do-i-make-cloud-init-startup-scripts-run-every-ti` ou bien <`https://help.ubuntu.com/community/CloudInit`>.

Notez que l'exemple avec POV-Ray n'est pas forcément convaincant si on a une machine locale rapide. Une scène assez simple m'a pris 60 secondes sur mon PC et plus de 3 minutes sur une machine EC2 de la catégorie `m1.small`. EC2 est plus intéressant pour des calculs plus longs.

Les outils d'Amazon sont documentés en ligne <`http://docs.amazonwebservices.com/AWSEC2/latest/CommandLineReference/index.html`>. Et si on n'aime pas les outils en ligne de commande d'Amazon, notons qu'il existe des concurrents comme aws <`http://timkay.com/aws/`>.

Et si on veut écrire des programmes plus complexes, faisant des choses plus raffinées avec les machines EC2 ? Il existe des bibliothèques pour cela dans tous les langages de programmation. En Python, j'utilise boto <`http://code.google.com/p/boto/`>, par exemple dans un script qui permet lui aussi de créer une machine et d'exécuter des commandes ensuite : (en ligne sur `https://www.bortzmeyer.org/files/amazon-ec2-run-commands.py`). Les programmeurs Perl pourront, eux, utiliser `Net::Amazon::EC2`.

Sinon, si on veut installer les outils Amazon ligne de commande à la main, et non pas depuis un paquetage, ceci semble marcher : télécharger depuis <`http://aws.amazon.com/developertools/351`>, `export JAVA_HOME=/usr` (ou l'endroit où on a son environnement Java, ce point ne semble pas documenté), définir la variable d'environnement `EC2_HOME`. Par exemple, si on a mis les outils en `/usr/local/amazon/api` :

```
% export EC2_HOME=/usr/local/amazon/api
% export PATH=$PATH:$EC2_HOME/bin
% export EC2_PRIVATE_KEY=...comme avant...
% export EC2_CERT=...comme avant...
```

On notera que, comme avec la plupart des programmes écrits en Java, ils ne marchent qu'avec le Java de Sun. Avec gij, on a :

```
% ec2-describe-regions
Exception in thread "main" java.lang.NoClassDefFoundError: org.codehaus.xfire.aegis.type.xml.SourceType
   at java.lang.Class.initializeClass(libgcj.so.10)
   at org.codehaus.xfire.aegis.type.DefaultTypeMappingRegistry.createDefaultMappings(DefaultTypeMappingRegis
   at org.codehaus.xfire.aegis.type.DefaultTypeMappingRegistry.createDefaultMappings(DefaultTypeMappingRegis
   at org.codehaus.xfire.aegis.type.DefaultTypeMappingRegistry.<init>(DefaultTypeMappingRegistry.java:131)
   at org.codehaus.xfire.aegis.type.DefaultTypeMappingRegistry.<init>(DefaultTypeMappingRegistry.java:137)
   at org.codehaus.xfire.aegis.type.DefaultTypeMappingRegistry.<init>(DefaultTypeMappingRegistry.java:118)
```

Et si on veut de l'aide, quelles sont les ressources disponibles ? Il y a un bon forum IRC sur Freenode, `##aws` (oui, deux croisillons). Et il y a les forums de discussion d'Amazon <https://forums.aws.amazon.com/forum.jspa?forumID=30&start=0> où les utilisateurs peuvent dialoguer entre eux (voir un exemple avec mes problèmes sur les security groups <https://forums.aws.amazon.com/thread.jspa?threadID=47125&tstart=0>.

L'informatique en nuage en général et Amazon EC2 en particulier ont parfois été critiqués au nom de la fiabilité : si le nuage est en panne, on est fichu. Mais rien n'indique qu'EC2 soit plus souvent en panne que des machines gérées dans une salle de machines locale. Prenons l'exemple de la grande panne EC2 d'avril 2011. Amazon avait très bien documenté cette panne (qui frappait EBS et, par ricochet, EC2) dans un excellent rapport technique <http://aws.amazon.com/message/65648/>. Quelques points que j'ai retenus :

- Amazon reconnait la perte de données (« pour 0,07 % des clients », ce qui n'est pas forcément incompatible avec les chiffres annonçant que des centaines de clients avaient perdu des données, car EC2 a beaucoup de clients),
- la panne (et l'article d'Amazon) expliquent très bien l'intérêt du système des régions (une des fonctions les moins comprises d'EC2), puisque, ne partageant rien, les autres régions n'ont pas été touchées ; un client d'Amazon malin pouvait donc assurer la redondance de son service,
- très bonnes explications techniques (ah, le problème de la synchronisation des clients, se précipitant tous au même moment dès que le système remarche, le faisant replanter) : faire un gros système distribué est **dur**.

Sur le même sujet, deux bons articles (parmi les milliers déjà publiés) étaient « "*5 Lessons We've Learned Using AWS*" <http://techblog.netflix.com/2010/12/5-lessons-weve-learned-using-aws.html> » (qui inclut la description du désormais célèbre Singe du Chaos) et « "*Amazon's EBS outage*" <http://storagemojo.com/2011/04/29/amazons-ebs-outage/> ».

Outre cette question de la fiabilité, qu'en est-il de problèmes plus politiques avec l'informatique en nuage comme la protection de la vie privée ou bien le risque de contrôle et de censure ? Richard Stallman, dans un interview <http://www.guardian.co.uk/technology/2008/sep/29/cloud.computing.richard.stallman>, avait fortement critiqué l'idée même d'informatique en nuage, notamment par la perte de contrôle qui en résultait (ses remarques concrètes s'appliquaient plutôt au SaaS comme Gmail plutôt qu'à toute l'informatique en nuage). Il est clair que c'est un problème réel à prendre en compte. Par exemple, Amazon peut lire toutes vos données stockées sur une machine EC2 (sauf si vous les avez déposé déjà chiffrées et jamais déchiffrées depuis EC2). L'informatique en nuage, c'est de la sous-traitance et elle présente donc les risques et les avantages de la sous-traitance.

Plus grave, le risque de censure. Comme WikiLeaks l'a découvert, Amazon peut à tout moment décider de couper le service. Le fameux premier amendement de la constitution des États-Unis dit que l'État n'a pas le droit de censurer. Mais les entreprises privées comme Amazon sont parfaitement libres de le faire. (Voir l'opinion de l'EFF <https://www.eff.org/deeplinks/2010/12/amazon-and-wikileaks-f> et celle d'Amazon <http://aws.amazon.com/message/65348/>, défendant la censure. Clients potentiels, vous êtes prévenus.)

Et si, compte-tenu de cela et d'autres problèmes, on ne veut pas utiliser Amazon du tout ? Un concurrent possible est Eucalyptus mais attention, c'est un concurrent d'Amazon ayant la même API (donc on peut utiliser les mêmes outils) mais pas un service : il faut se bâtir son nuage à soi. Autre approche du problème de la dépendance vis-à-vis de l'API d'Amazon : utiliser une bibliothèque qui masque les différences entre fournisseurs, comme libcloud <http://libcloud.apache.org/>.

Quelques lectures :

— Le site officiel d'EC2 <http://aws.amazon.com/ec2> et celui d'AWS <http://aws.amazon.com/>, et la documentation officielle <http://aws.amazon.com/documentation/ec2/>.
— Un bon guide pour le débutant <http://paulstamatiou.com/how-to-getting-started-with-amazon-e très détaillé.
— Mes liens divers sur EC2 <http://delicious.com/bortzmeyer/ec2>.