Guide technique

Vers la liberté en infonuagique

Table des matières

Préambule

Les nuages publics comme AWS d'Amazon, Google Cloud et Azure de Microsoft exercent de plus en plus d'influence sur les possibilités que les décideurs IT doivent envisager avant de concevoir et de déployer une application. En effet, ces nuages proposent un large éventail de services qui accélèrent certaines tâches tel le déploiement d'applications Web, réduisent les coûts de la maintenance ou atténuent les risques de défaillance grâce à une infrastructure aussi souple que dynamique.

Malheureusement cette économie de temps et cette plus grande souplesse ont un prix. En effet, bon nombre de ces fournisseurs recourent à des technologies, à des API et à des interfaces qui leur sont exclusives, donc sont incompatibles avec celles de leurs concurrents, ce qui multiplie les risques d'en devenir l'esclave.

La liberté, en infonuagique, signifie concevoir une solution afin qu'elle fonctionne sur n'importe quel nuage et ne dépende pas d'une plateforme donnée. En la concevant de la sorte, le moment venu, on pourra choisir en toute connaissance de cause soit d'exploiter les propriétés d'une plateforme en nuage, soit de créer une solution plus générique et portative. Un projet pourra donc bénéficier des avantages d'un nuage public mais garder assez de latitude pour passer facilement à celui d'un autre fournisseur, dans l'avenir.

Sujets abordés

Ce tutoriel explique comment bâtir l'infrastructure d'une application Web moderne en vue de son hébergement dans le nuage. Sans aller trop profondément dans l'application proprement dite, vous découvrirez les différents services qu'offrent les nuages publics, leurs avantages et leurs inconvénients, et apprendrez à choisir certains services ou à gérer un composant vous-même.

Le tutoriel présente une application Web (la version d'essai d'un système de scrutin en ligne) et les composants de son infrastructure en nuage. Nous comparerons l'usage d'un service d'infonuagique public pour gérer ces composants à l'autre solution, qui consiste à s'en occuper soi-même.

Ce dont vous aurez besoin

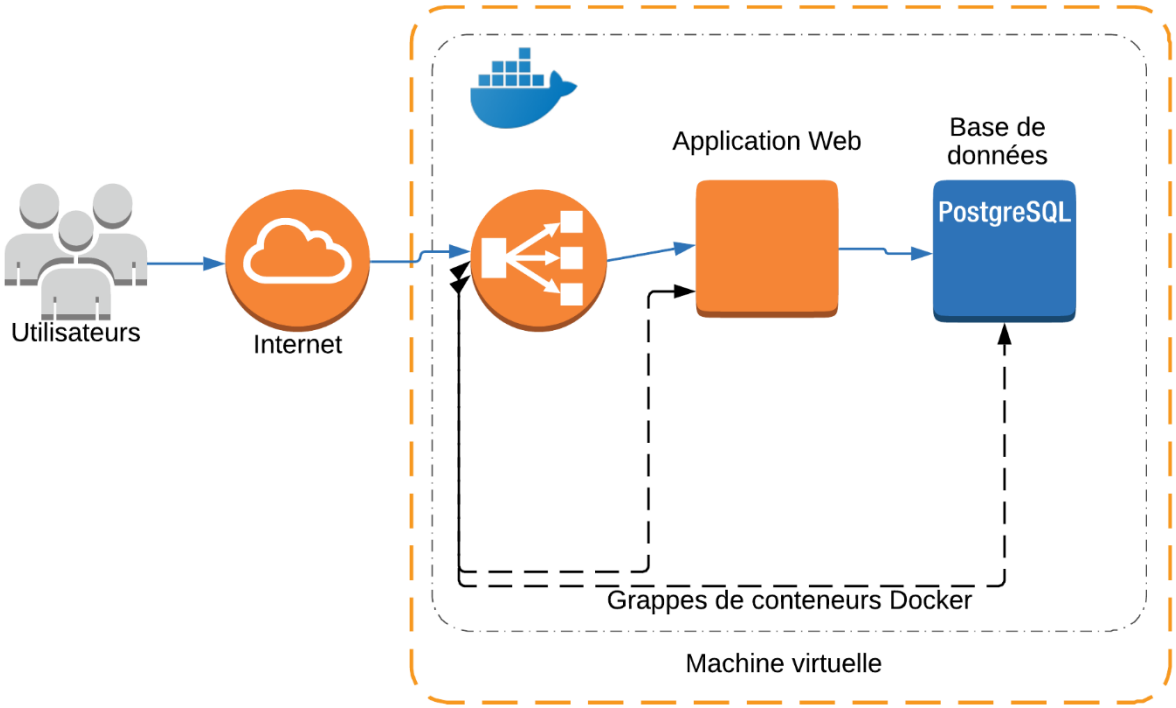
- Un peu d'expérience en infonuagique et en création d'applications
- Les justificatifs d'identité d'un fournisseur de services en nuage– par ex., nuage de l'ATIR, OpenStack, Microsoft Azure, etc. (voir les détails dans le fichier readme du dépôt de l'application servant d'illustration)
- Un compte Docker Hub
- Docker (télécharger à partir de votre compte Docker Hub)

- Git
- Terraform

Composants d'une application Web de base

L'application Web qui nous servira d'exemple et que vous déploierez est un clone du tutoriel Django. **Django** est un cadriciel Web populaire rédigé en langage Python (plus sur le projet Django ici).

Commençons par examiner l'architecture générale de l'application que vous créerez avec ce tutoriel.

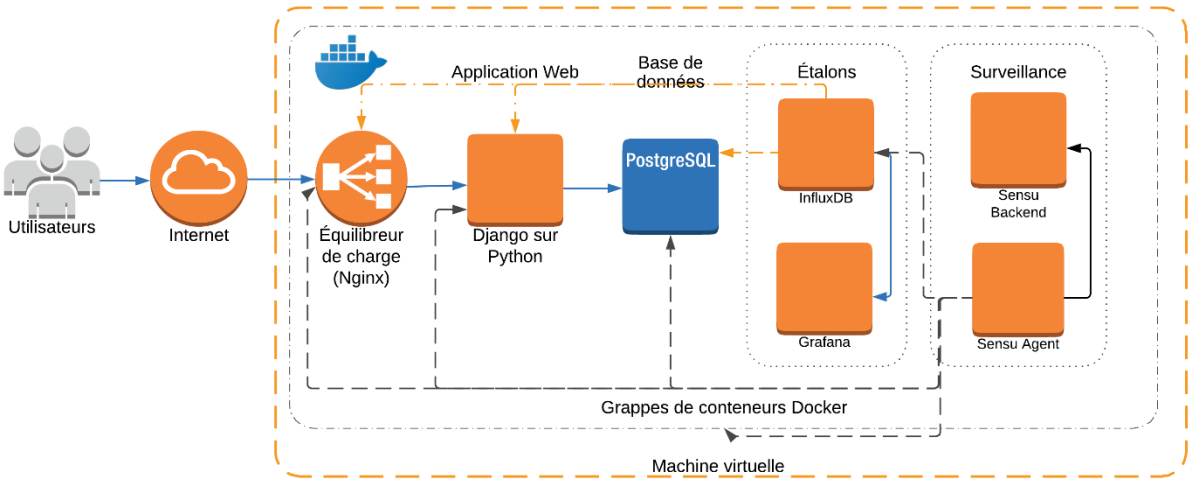


Comme vous pouvez le constater, l'application comporte trois composants principaux :

- **un équilibreur de charge :** il accepte le trafic venant du réseau et le répartit entre les serveurs de l'application;
- **l'application proprement dite :** le code permettant à l'utilisateur d'interagir avec l'application par le truchement de son navigateur;
- **une base de données :** le service qui emmagasine les données d'une manière fiable et durable.

La plupart des fournisseurs de services d'infonuagique publics proposent des variantes de ces composants. Vous pouvez aussi recourir à une solution de source ouverte qui n'adhère à aucun nuage public ou à un mélange des deux.

Ainsi, pour déployer une application Web écrite en Python qui recourt à une base de données Postgres assortie d'un égalisateur de charge comme NGINX, le résultat final, incluant la surveillance et la collecte de données, pourrait ressembler à ce qui suit.



Vous pourriez choisir de profiter à la fois des services d'un nuage public et de composants autonomes en confiant un ou plusieurs des services indiqués ci-dessus à un fournisseur spécifique. Le diagramme resterait le même, un bloc remplacerait simplement un ou plusieurs composants. Nous y reviendrons plus loin.

Application Web

Le premier composant à examiner est l'application Web qui nous sert d'illustration. Elle a été rédigée en Python, un langage robuste, simple à utiliser, qui laisse beaucoup de latitude au niveau de l'hébergement dans un nuage. Notre application, déployée dans un conteneur Docker, suivra les connexions de la clientèle sur le port 8000.

Docker permet d'emballer, d'expédier et d'utiliser aisément n'importe quelle application sous la forme d'un petit conteneur portatif, exécutable presque partout. Des millions d'applications ont adopté Docker depuis sa diffusion en 2014. L'aide, la documentation et le soutien ne manquent donc pas, si vous en avez besoin. Les principaux fournisseurs de services d'infonuagique proposent tous une vaste gamme de services compatibles avec Docker, ce qui vous permettra d'aller aisément de l'un à l'autre avec votre application « conteneurisée ».

Un avantage des conteneurs Docker est qu'ils peuvent être hébergés de diverses manières dans le nuage. La façon la plus élémentaire et la plus souple consiste à créer l'**instance d'une machine virtuelle (MV)** et à installer manuellement Docker dans celle-ci afin qu'elle accueille des applications en conteneur. Cette méthode fonctionnera d'un fournisseur de services d'infonuagique à l'autre, mais elle requière une solide connaissance de l'administration des systèmes et la maintenance de l'application, durant sa vie utile, exigera beaucoup de temps.

Pour rendre la gestion de Docker plus facile, les fournisseurs proposent divers services qui y sont adaptés comme **AWS Fargate** et **Azure Container Instances**.

Ces solutions occupent des créneaux différents du continuum, comme on peut le voir ci-dessous. En général, plus la gestion d'une solution est prise en main, plus elle sera dispendieuse. Sa mise en œuvre sera aussi plus rigide, cependant la plus faible maintenance qui en résulte compense parfois ces inconvénients.

Gestion intégrale	Gestion partielle	Autogestion
Google App Engine	AWS Fargate	Docker sur une instance
Heroku	Azure Containers	
Serverless (AWS Lamba, AWS Functions)		

Exécuter une grappe de conteneurs dans un **service d'orchestration** comme AWS Fargate ou Azure Containers coûtera plus cher, mais la maintenance sera moins grande pour le personnel IT, car le fournisseur s'occupera de la pile de conteneurs et gèrera la grappe, de même que les MV qui y sont associées. Il se pourrait néanmoins que les types de conteneur d'un fournisseur ne puissent facilement être exportés vers un autre fournisseur, donc songez-y sérieusement avant de choisir une solution ou une autre.

Équilibreur de charge

On pourrait donner à l'utilisateur directement accès à l'application de scrutin, mais une meilleure solution à long terme consiste à l'installer après un égalisateur de charge. En passant par un égalisateur pour interagir avec l'application, on pourra changer plus facilement d'échelle par la suite et on créera un tampon immédiat entre l'Internet et l'application. Pareille séparation ajoute une couche de sécurité, absente avec une connexion directe comme avec NGINX (ou un autre équilibreur). Les équilibreurs de charge ont une couche réseau plus résistante qui fait habituellement défaut à la majorité des applications.

Les équilibreurs de charge fonctionnent en deux modes : couche 4 ou couche 7.

Ceux de la **couche 4** opèrent dans la **couche de transport** qui achemine le trafic du réseau, peu importe le contenu.

Les égalisateurs de la **couche 7** opèrent dans la **couche des applications,** qui régule le contenu réel du trafic et autorise des décisions plus intelligentes concernant la destination ultime de ce dernier.

Ainsi, les demandes concernant une page précise émises par un équilibreur de charge de la couche 7 pourraient être acheminées au serveur d'une application Web donnée, alors que l'égalisateur de la couche 4 redirigera les demandes vers les serveurs d'applications sans tenir compte de la nature de la demande. Les deux types d'équilibreurs peuvent établir vers quels serveurs il est préférable d'acheminer la demande.

La plupart des options d'équilibrage que proposent les services d'infonuagique publics se situent dans la couche 4 (AWS Elastic Load Balancer, Azure Load Balancer), mais l'équilibrage dans la couche 7 gagne en popularité depuis peu et il est désormais plus facile de l'obtenir (par ex., AWS App Mesh avec Envoy, Azure Application Gateway) en sus des possibilités d'égalisation reposant sur DNS (AWS Route 53, Azure Traffic Manager).

Parmi les équilibreurs de charge de source ouverte, mentionnons HAProxy, NGINX et Envoy.

Comme on ne cessera de le répéter dans ce tutoriel, choisir entre le service payant d'un fournisseur et l'autogestion de l'équilibrage dépend de l'expertise de l'équipe et du temps dont elle dispose pour cela, ainsi que de l'importance que cette activité présente pour la réalisation du but recherché. Les quelques remarques que voici comparent des services entièrement gérés à la gestion personnelle des mêmes services.

Gestion intégrale du service	Autogestion
Conception plus simple et meilleur maintien de l'application	Complexité supérieure mais meilleur contrôle
Économie de coûts comparativement à l'embauche de professionnels en IT chevronnés	Économie de coûts si l'entreprise a déjà des employés expérimentés
Mise en œuvre rapide, mais sujétion au nuage du fournisseur	Mise en œuvre plus lente, mais portabilité d'un nuage à l'autre
Forte disponibilité intégrée ou services réseau, ce qui engendre une solution plus robuste et plus fiable	Investissement supérieur au niveau du développement et des essais; forte disponibilité pour le passage à la production

Dans ce tutoriel, nous utiliserons le conteneur Docker NGINX autogéré et de source ouverte, qui permet d'équilibrer la charge à la fois dans la couche 4 *et* dans la couche 7. Pour plus de simplicité, nous nous en servirons strictement comme égalisateur de charge de la couche 4. Le conteneur Docker NGINX est facile à utiliser et permet aisément d'économiser tant sur le coût des opérations que sur celui de la largeur de bande. En outre, sa portabilité autorise le transfert de la configuration d'un fournisseur de services à l'autre.

En retenant le conteneur Docker NGINX de source ouverte, on renonce néanmoins aux fonctionnalités propres au réseau du fournisseur, notamment la forte disponibilité intrinsèque et le contrôle des accès et identités.

Base de données

Pour que l'application de scrutin suive la situation à mesure qu'elle évolue, les votes des utilisateurs doivent être stockés dans une base de données. Comme c'est le cas pour l'application Web et l'équilibreur de charge, il est possible de gérer la base de données soi-même ou d'en confier la gestion à un fournisseur de services d'infonuagique.

La première option vous donnera plus de marge de manœuvre et de contrôle sur le déploiement de la base de données, mais l'administration et la maintenance de celle-ci pourraient s'avérer une lourde tâche. Les problèmes de performance, de résilience, d'espace-disque, de sauvegarde ou d'accès peuvent en effet absorber beaucoup de temps qui pourrait être utilisé à meilleur escient pour faire avancer le projet. Une base de données mal entretenue peut soulever des difficultés appréciables en ralentissant l'application, laquelle pourrait alors connaître une défaillance ou tomber en panne, voire devenir une faille par où pourraient fuir les données consécutivement à un piratage.

Les fournisseurs de services d'infonuagique proposent diverses **bases de données en tant que service (DBaaS)** « de production », gérées à différents niveaux et assorties de particularités supplémentaires comme une forte disponibilité ou la résilience. Le **Relational Database Service (RDS)** d'Amazon Web Services, par exemple, offre la plupart des bases de données de fond habituelles, dont **Aurora**, la base de données exclusive du fournisseur. Les fonctionnalités sont similaires, mais Aurora n'appartient qu'à Amazon et, soutient-on, est nettement plus performante que ses contreparties.

Le prix des bases de données est complexe en infonuagique. En planifier le coût suppose facturer le prix du stockage, de la largeur de bande, du nombre de demandes, de l'exportation des données et ainsi de suite. Par ailleurs, si recourir au service de base de données du fournisseur vous économisera le temps et l'argent que suppose la gestion d'une telle application, vous courrez le risque que vos données se trouvent confinées dans la solution d'un fournisseur unique.

Conserver la propriété des données devrait être une priorité. C'est pourquoi ce tutoriel se concentrera sur la gestion de notre propre base de données. La souveraineté des données est un autre point important que beaucoup prennent en considération. Si vous êtes dans ce cas, vous devrez déterminer comment faire en sorte que les données restent strictement dans un centre canadien. Le tutoriel

utilise un conteneur Docker PostgreSQL. Tout comme l'application Web et l'égalisateur de charge, l'usage d'un conteneur Docker facilitera le transfert des composants d'un nuage public à l'autre. Voici un bref tableau comparant les avantages d'une base de données entièrement gérée à une dont vous prendrez en charge la gestion vous-même.

Gestion intégrale	Autogestion
– La BD pourrait être exclusive, si bien que vous pourriez vous retrouver « prisonnier » du fournisseur.	– Contrôle direct sur la BD et accès à sa coquille
– Amazon Aurora est une solution naturelle à forte disponibilité, optimisée, venant avec des bases auxiliaires, doublées dans de nombreuses régions	– Peut être utilisée pour un formatage particulier ou un moteur de stockage spécifique que ne pourrait supporter un service entièrement géré. Amazon Aurora ne supporte qu'innoDB, par exemple.
– Dépendance absolue sur le fournisseur pour le débogage et les mises à niveau	– Temps et ressources requis pour maintenir la BD et mettre à niveau ou sécuriser le système d'exploitation
– Pas besoin d'un administrateur	– Administrateur indispensable
– Sauvegardes et mises à l'échelle automatiques – Enregistrement constant dans le stockage d'objets, sans répercusssion sur la performance – Met fin à la nécessité de fixer des moments pour la sauvegarde – Inutile de planifier les capacités à l'avance	– Travail et planification supplémentaires pour les sauvegardes, la journalisation, la mise à l'échelle lors d'une expansion et la surveillance du rendement
– Aucun contrôle ou très peu sur la souveraineté des données	– Contrôle absolu sur la souveraineté des données

Outils indépendants

Dans la partie qui précède, nous avons examiné en détail les composants de notre application Web. Nous avons aussi parlé des solutions que proposent les fournisseurs de services publics d'infonuagique, des solutions que vous pouvez prendre en charge personnellement ainsi que des compromis que cela entraîne.

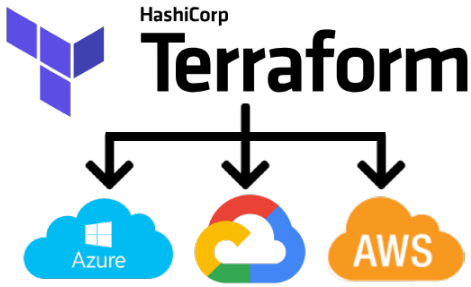
Tout en comprenant l'importance de ces composants pour parvenir à une solution efficace, il est impérieux également de savoir *comment* créer et déployer ladite solution. C'est pourquoi nous examinerons maintenant les outils dont vous disposez pour vous aider dans cette tâche et dans la gestion de la solution.

Un thème qui revient couramment dans nos tutoriels est celui de l'infrastructure en tant que code. Dans cette optique, la création, la gestion et l'exploitation de l'application Web qui nous sert d'illustration seront définies comme un *code*, avec l'application proprement dite. Quand la solution au grand complet est établie sous forme de code, on peut recourir à des outils pour lire et exécuter ce code, donc obtenir constamment, de façon prévisible et fiable, les mêmes résultats durant le déploiement. Si, par exemple, l'application et son infrastructure étaient déployées par une suite de touches du clavier et de clics de souris, chaque étape serait à la merci de coquilles typographiques ou de touches ratées.

Définir une solution au moyen d'un code permet aussi d'adopter d'autres pratiques de développement exemplaires, notamment la **gestion, le balisage et la diffusion des versions**. Ces pratiques s'avéreront d'une grande aide quand viendra le temps de gérer les changements, la maintenance, les mises à niveau et les essais.

Enfin, définir l'infrastructure sous forme de code facilite le passage d'un fournisseur de services d'infonuagique à l'autre, au besoin. Quand on code le cœur de l'application, il est possible d'utiliser des outils comme Terraform pour la déployer à différents endroits ou sur diverses plateformes (développement, essai, production), avec de légères modifications. Changer de fournisseur ne vous obligera qu'à changer le nom et les fonctions des ressources.

Examinons les principaux outils que nous utiliserons pour déployer l'application Web qui nous sert d'illustration.



Terraform

Terraform est un outil avec lequel on peut bâtir une infrastructure, la modifier et en gérer les versions de façon efficace et sécuritaire au moyen d'un code. Les services d'infonuagique publics les plus populaires tolèrent cet outil aussi bien que les solutions internes, réalisées sur mesure.

Terraform recourt à un langage déclaratif courant pour créer des ressources et les gérer d'une plateforme en nuage à l'autre et c'est tant mieux car, autrement, ces ressources devraient être créées et gérées avec un clavier et une souris ou avec le langage de programmation naturel du fournisseur (CloudFormation d'AWS, par exemple), ce qui accroîtrait les risques d'asservissement à ce dernier.

Terraform peut aussi donner un aperçu ou un plan de vos actions, ce qui permet de déterminer les effets éventuellement destructeurs d'une modification de l'infrastructure avant qu'ils se produisent.

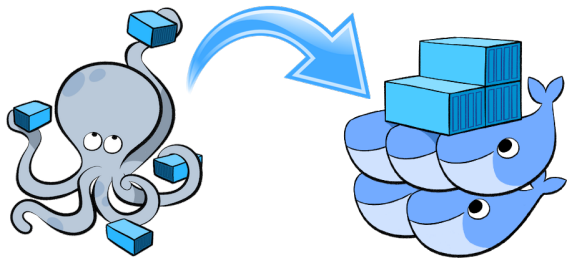
Enfin, Terraform autorise un déploiement plus évolué de la solution (du type blue-green par exemple), de même que le déploiement parallèle de plateformes de production et de développement.

Docker

Docker est un jeu d'outils avec lesquels il est possible de créer et de gérer le code des applications sous forme de *conteneurs*. Le conteneur renferme le code du logiciel, les services qui le soutiennent, les ressources dont il a besoin (l'unité de traitement centrale et la mémoire, par exemple) et les contraintes d'accès (privilèges de l'administrateur et accès au réseau), le tout dans un paquet que l'on peut transporter.

Docker est une solution populaire permettant de gérer et de déployer des logiciels, surtout des applications Web. Tel qu'indiqué précédemment, beaucoup de services d'infonuagique et d'outils indépendants sont compatibles avec Docker, et les experts ne manquent pas en plus d'être très dynamiques. Recourir à Docker, c'est garantir que l'application, dans son emballage, pourra être gérée et déployée presque n'importe où et qu'une collectivité est là, prête à vous épauler si vous en avez besoin.

Pour revenir au concept de l'infrastructure en tant que code, les conteneurs Docker sont définis par ce qu'on appelle des Dockerfile, fichiers texte qui énumèrent les étapes aboutissant à la création du conteneur. Le Dockerfile peut être géré comme n'importe quel autre élément de code.



Source: https://blog.emsa-cloud.com/docker-compose

Docker Compose

Docker Compose est un outil servant à définir et à orchestrer de nombreux conteneurs Docker. Cet outil rassemble tous les attributs du conteneur dans un fichier YAML, ce qui permet d'établir un lien entre les différents conteneurs au moyen d'un code.

Déploiement et configuration

À présent que nous avons examiné l'architecture de l'application Web et les outils dont on se servira pour la déployer dans le nuage, voyons comment procéder.

Comme le mentionnait le préambule, l'application a été conçue pour s'exécuter dans de multiples conteneurs Docker. Bien que les services publics d'infonuagique proposent diverses solutions pour héberger de tels conteneurs, nous installerons plutôt ceux-ci nous-mêmes, sur une instance de base (machine virtuelle).

En utilisant Docker, on garde la possibilité de recourir à une solution Docker spécifique au fournisseur, si jamais c'est ce qu'on souhaite. De cette manière, on n'asservira à aucun fournisseur en particulier et à ses solutions naturelles, ce qui va dans le sens de notre philosophie, qui est la liberté en infonuagique.

Remarque sur les moteurs d'orchestration de conteneurs

Les **moteurs d'orchestration de conteneurs (COE)** comme Docker Swarm ou Kubernetes ajoutent une couche d'automatisation et de gestion à Docker. Nous ne recourrons pas à un COE dans ce tutoriel.

Dépôt du code

Le code de l'application qui nous sert d'illustration se trouve dans le dépôt de solutions de l'ATIR. Vous pouvez télécharger un dossier ZIP ou le cloner sur votre poste de travail avec un client Git en exécutant la commande que voici sur votre terminal :

```
git clone https://solutions.cloud.canarie.ca:3000/DAIR/dair-cloud-independence-tutorial
```

Si vous utilisez un poste de travail Mac ou Linux, le dépôt renferme tout ce dont vous avez besoin pour déployer l'application. Si votre poste fonctionne sur la plateforme Windows, vous pourriez envisager le lancement d'une MV Linux, mais ceci déborde du sujet du tutoriel.

Dans le dépôt Git se trouve un fichier README détaillé dont vous devriez prendre connaissance. Suivez ses instructions. Ce fichier décrit comment obtenir les justificatifs d'identité du nuage AWS, Azure ou OpenStack. Nous vous recommandons d'obtenir les identifiants d'au moins deux nuages.

Déployer l'application

Les étapes suivantes supposent que vous avez obtenu les identifiants des nuages OpenStack et Microsoft Azure.

Déploiement dans OpenStack

Suivez les instructions du fichier README pour vous authentifier sur OpenStack et modifiez les variables de votre plateforme en conséquence. Ensuite, du terminal, changez les répertoires dans le dépôt Git local que vous avez consulté et exécutez la commande suivante :

```
make apply ENV=openstack
```

Lorsque l'exécution est terminée, vous verrez quelque chose ressemblant à ceci :

```
Apply complete! Resources: 11 added, 0 changed, 0 destroyed.


Outputs:

polls_url = http://111.203.33.44/polls

public_ip = 111.203.33.44
```

Une autre adresse IP que 111.203.33.44 devrait apparaître à l'écran. Copiez et collez « polls_url » dans votre navigateur pour voir l'application en action.

Avant de passer à la partie suivante, vous pouvez laisser l'application se déployer sur OpenStack ou la supprimer avec cette commande :

```
make destroy ENV=openstack
```

Déploiement dans Azure

Suivez les instructions du fichier README pour vous authentifier dans Azure et modifiez les variables de votre plateforme en conséquence.

Cela fait, lancez la commande :

```
make apply ENV=azure
```

Une fois l'exécution terminée, vous verrez quelque chose de presque identique à ce qui était apparu à l'écran lors du déploiement de l'application dans OpenStack. Si vous visitez la page « polls_url », vous obtiendrez le même résultat!

Félicitations, vous venez de déployer la même application Web sur deux nuages différents! Vous avez pu constater la rapidité de l'infrastructure en tant que code et des applications en conteneurs ainsi que la portabilité de leur déploiement.

Déploiement dans AWS

Voyez si vous parvenez à suivre les instructions du fichier README en vous authentifiant et en déployant l'application dans AWS. La procédure ressemble à celle que nous venons de voir pour OpenStack et Azure. Elle nécessitera la modification des variables de votre environnement.

Apporter des changements à l'application

Déployer l'application, c'est bien et pouvoir le faire dans plusieurs nuages, c'est mieux, cependant, même déployée, une application devra inévitablement subir des changements. Peut-être quelqu'un voudra-t-il y ajouter une fonctionnalité, à moins que vous vouliez réorganiser le site ou que quelqu'un rapporte un bogue.

Pouvoir gérer et déployer les changements d'une manière contrôlée est un aspect important du développement de l'infrastructure. Si important, en fait, que nous y consacrerons d'autres tutoriels. Dans l'immédiat, voici une façon très élémentaire de déployer des modifications.

Le dépôt Git que vous avez téléchargé du dépôt de solutions de l'ATIR comporte plusieurs dossiers. Celui appelé « apps » renferme tous les fichiers nécessaires pour gérer les applications qui constituent l'infrastructure complète du nuage. Dans le dossier « app » s'en trouve un autre baptisé « docker_files » et dans celui-ci, un autre dossier « app » où se trouve l'application Django. Si vous devez apporter des changements à cette dernière, c'est l'endroit où le faire.

Pour modifier la page « polls » en y ajoutant une rubrique au sommet annonçant « Sondages », par exemple, ouvrez le fichier « apps/docker_files/app/polls/templates/polls/index.html » dans l'éditeur de texte de votre choix.

Sous la troisième ligne, insérez-en une nouvelle qui se lit comme suit :

```
<h1>Sondages</h1>
```

Enregistrez le fichier et fermez l'éditeur.

Après ce changement, vous devrez redéployer l'application afin de la mettre à jour. À la racine du dépôt, au même endroit où se trouve Makefile, exécutez les deux commandes que voici :

```
make deploy_app ENV=openstack
make restart_app ENV=openstack
```

Cela fait, rendez-vous à l'adresse http://111.203.33.44/polls et voyez la nouvelle en-tête.

Aussi simples que puissent paraître les étapes qui ont mené à ce changement, beaucoup de choses se sont passées en coulisses. Terraform a créé la machine virtuelle, copié les fichiers Docker, bâti les conteneurs et connecté les services. Et nous n'avons qu'effleuré la surface. Gérer et déployer des modifications est un vaste sujet que nous aborderons dans d'autres tutoriels. Dans l'intervalle, en guise d'exercice, examinez les tâches « deploy_app » et « restart_app » dans Makefile et voyez si vous pouvez suivre exactement ce que font ces commandes.

Récapitulation

Avec ce tutoriel, nous vous avons expliqué ce que signifie la liberté en infonuagique et pourquoi elle est importante. Nous avons illustré cette indépendance avec l'architecture d'une application Web moderne et vous avons présenté les outils qui vous aideront à conserver cette indépendance. Nous avons abordé la façon de maintenir une application Web au moyen d'outils adaptés à l'infrastructure en tant que code et nous vous avons montré comment déployer l'application dans le nuage de trois fournisseurs, pour vous prouver que oui, la liberté est possible en infonuagique.

Ne ratez pas nos prochains tutoriels sur le même sujet et d'autres de même nature.