Gestion des applications multiniveau avec AWS OpsWorks

Daniele Stroppa

Janvier 2015

Table des matières

Résumé

Une application a généralement besoin d'un ensemble de ressources, telles qu'un équilibreur de charge, des serveurs Web et d'application et un serveur de base de données, que vous devez globalement créer et gérer. De plus, vous devez déployer votre application sur des serveurs d'application, gérer les permissions de sécurité et des ressources, et surveiller les performances de la solution.

Le présent livre blanc explique comment utiliser AWS OpsWorks pour gérer des applications ainsi que leurs ressources associées et propose des conseils pour faciliter la diffusion de ces solutions.

Introduction

AWS OpsWorks offre une manière flexible de créer et de gérer des ressources pour vos applications. Il prend en charge des composants standard tels que les serveurs d'application, les serveurs de base de données et les équilibreurs de charge, ainsi que des composants personnalisés comme les outils de recherche et les systèmes de messagerie. AWS OpsWorks fournit également des outils pour personnaliser les configurations de package standard, installer les packages supplémentaires, automatiser les runbooks et gérer les permissions des utilisateurs au niveau du système d'exploitation.

Concepts clés

Avant de vous lancer dans l'utilisation d'AWS OpsWorks, il est utile de comprendre quelques concepts clés.

Stack

Une *pile* est un ensemble de ressources AWS qui sont gérées ensemble, par exemple les instances Amazon EC2, les volumes Amazon EBS et les équilibreurs de charge Elastic Load Balancing. AWS OpsWorks vous aide à gérer globalement ces ressources mais aussi à définir certains paramètres de configuration par défaut. Vous pouvez créer des piles séparées pour des environnements différents, par exemple une pile pour l'assurance qualité/les tests et une pour la production, et pour des applications différentes.

Couche

Chaque pile contient une ou plusieurs *couches*. Une couche spécifie comment configurer un ensemble d'instances Amazon EC2 dans un but précis comme l'hébergement d'un serveur Web.

AWS OpsWorks fournit un ensemble de couches intégrées qui prennent en charge une variété de packages standard, notamment des serveurs d'application comme Tomcat et Node.js, des serveurs de base de données comme MySQL et Amazon RDS, des équilibreurs de charge comme Elastic Load Balancing et HAProxy, etc. AWS OpsWorks vous permet de personnaliser ou d'élargir les couches intégrées en modifiant les configurations par défaut et en ajoutant des recettes Chef personnalisées. Vous pouvez créer une couche entièrement personnalisée, ce qui vous donne un contrôle total sur la configuration et l'installation de cette dernière.

Application

Vous représentez votre application au sein d'AWS OpsWorks en définissant une *application*, qui spécifie son type et contient les informations nécessaires à son déploiement depuis son référentiel vers vos instances de serveur d'application. Lorsque vous déployez une application, AWS OpsWorks exécute les recettes de déploiement sur toutes les instances de la pile, ce qui permet à ces dernières, comme les serveurs de base de données, de modifier leur configuration le cas échéant. AWS OpsWorks prend en charge la capacité à déployer plusieurs applications par pile et par couche.

Evènements du cycle de vie

Chaque couche dans une pile AWS OpsWorks possède un ensemble d'évènements *du cycle de vie* correspondant aux différentes étapes du cycle de vie d'une instance, comme l'installation ou le déploiement d'une application. Chaque évènement du cycle de vie possède un ensemble associé de recettes Chef qui sont exécutées sur chacune des instances de la couche pour réaliser les tâches requises. AWS OpsWorks fournit des recettes intégrées pour réaliser une gestion de base, et vous pouvez ajouter des recettes personnalisées à n'importe quel évènement du cycle de vie pour orchestrer le moindre changement de configuration exigé par votre application.

Installation

Lors du démarrage d'une nouvelle instance, AWS OpsWorks déclenche l'évènement d'installation, qui exécute des recettes pour installer l'instance conformément à la configuration de la couche. Par exemple, si l'instance fait partie de la couche PHP App Server, les recettes installent les packages Apache et PHP. Une fois l'installation terminée, AWS OpsWorks déclenche un évènement de déploiement, qui exécute des recettes pour déployer votre application sur la nouvelle instance.

Configuration

Lorsqu'une instance entre ou quitte l'état en ligne, AWS OpsWorks déclenche un évènement de configuration sur toutes les instances de la pile. L'évènement exécute chacune des recettes de configuration de la couche pour mettre à jour la configuration et refléter l'ensemble actuel des instances en ligne. Par exemple, les recettes de configuration de la couche HAProxy modifient la configuration de l'équilibreur de charge pour refléter toutes les instances de serveur d'application ajoutées ou supprimées.

Déploiement

AWS OpsWorks déclenche un évènement de déploiement lorsque vous exécutez une commande de déploiement généralement pour déployer votre application sur un ensemble de serveurs d'application. L'évènement exécute des recettes sur les serveurs d'application pour déployer l'application et les fichiers associés depuis son référentiel vers les instances de la couche. Vous pouvez déclencher le déploiement sur d'autres instances afin qu'elles puissent, par exemple mettre à jour leur configuration et s'adapter à l'application nouvellement déployée.

Annulation du déploiement

AWS OpsWorks déclenche un évènement d'annulation du déploiement lorsque vous supprimez une application ou exécuter une commande d'annulation du déploiement pour supprimer une application d'un ensemble de serveurs d'application. L'évènement exécute des recettes pour supprimer toutes les versions d'application et réaliser des tâches de nettoyage supplémentaires.

Arrêt

AWS OpsWorks déclenche un évènement d'arrêt lorsqu'une instance est arrêtée, mais avant que l'instance Amazon EC2 sous-jacente ne soit arrêtée. L'évènement exécute des recettes pour réaliser des tâches de nettoyage telles que l'arrêt des services. AWS OpsWorks alloue aux recettes d'arrêt un laps de temps configurable pour réaliser leurs tâches, puis il met fin à l'instance.

Conception

Une application est généralement créée à l'aide de plusieurs niveaux :

- Présentation : un serveur Web frontal qui traite du contenu statique et potentiellement du contenu dynamique en cache.
- Logique métier : un serveur d'application dans lequel le contenu dynamique est traité et généré.
- Travailleurs : un ensemble de serveurs qui exécutent des tâches en arrière-plan et de longue durée.
- Données : une base de données principal ou un magasin de données.
- Intégration : des services pour relier les autres couches, par exemple des files d'attente de messages et des rubriques.

AWS OpsWorks vous laisse modeler chaque niveau avec une couche qui définit comment configurer les ressources du niveau. Vous pouvez également associer plusieurs niveaux à une seule instance, par exemple si vous avez besoin de configurer un serveur Web administratif pour un groupe de serveurs Web.

Architecture de micro-services

L'architecture de micro-services est une approche conceptuelle pour créer une seule application comme un ensemble de petits services. Chaque service s'exécute dans son propre processus et communique avec les autres services via une interface bien définie grâce à un mécanisme léger, généralement une interface de programmation d'applications (API) basée sur HTTP. Les micro-services sont développés autour des capacités métier, et chaque service exécute une seule fonction. Les micro-services peuvent être écrits à l'aide d'infrastructures ou de langages de programmation différents, et vous pouvez les déployer indépendamment, comme un seul service ou comme un groupe de services.

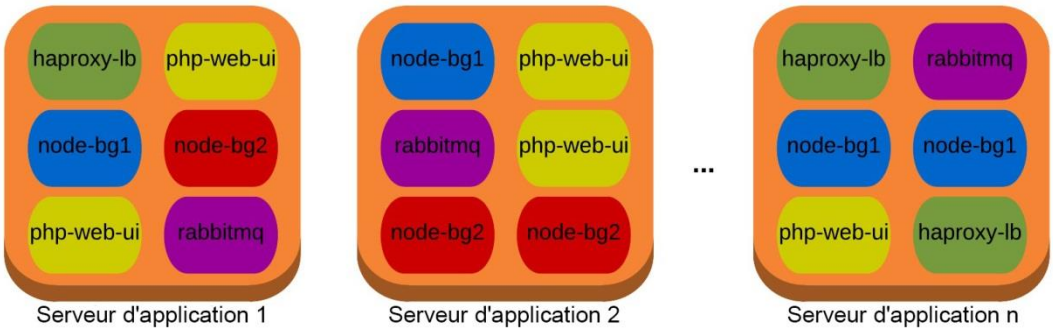

Figure 1 : un exemple d'architecture d'application utilisant les micro-services

Les micro-services peuvent être modelés dans AWS OpsWorks, avec chaque micro-service représenté par une couche séparée. Chaque couche peut être configurée indépendamment grâce à différentes recettes et chaque couche contient un ensemble d'instances. Le déploiement d'application peut être réalisé sur une seule instance, sur toutes les instances d'une couche particulière, c'est-à-dire par micro-service, ou sur les instances de plusieurs couches en même temps.

L'autre approche des architectures de micro-services consiste à utiliser des conteneurs, comme Docker. Docker est une plateforme open source qui vous permet de créer, diffuser et déployer des applications distribuées dans des environnements d'exécution légers appelés conteneurs. Ces conteneurs peuvent évoluer en fonction des besoins métier.

Le conteneur par défaut Docker, libcontainer, permet des services de gestion et de déploiement d'image grâce aux fonctions du noyau Linux telles que :

- Les cgroups pour l'isolation des ressources
- Les espaces de noms pour isoler la vue de l'application de l'environnement d'exploitation
- Les profils apparmor pour limiter les fonctions de l'application
- Les interfaces de mise en réseau pour fournir une connectivité réseau
- Les règles de pare-feu pour fournir un environnement isolé aux applications.

Les conteneurs peuvent partager le même noyau, mais chaque conteneur peut être limité à l'utilisation d'une quantité déterminée de ressources comme le processeur, la mémoire et l'E/S.

Docker permet aux développeurs de créer facilement des systèmes distribués en exécutant plusieurs applications indépendamment sur une seule machine. De nouvelles ressources sont déployées le cas échéant, ce qui permet à chaque micro-service d'évoluer de manière autonome. L'AWS Application Management Blog montre comment exécuter vos conteneurs Docker avec AWS OpsWorks.

Approvisionnement et déploiement

Vous devez gérer l'approvisionnement automatisé d'infrastructure comme n'importe quel autre code source, en utilisant un système de contrôle de versions. L'approvisionnement des ressources, la configuration logicielle et le déploiement d'applications doivent être déterministes, répétables, flexibles et prévisibles.
AWS OpsWorks et les autres services AWS tels qu'AWS CloudFormation, vous permettent d'approvisionner l'infrastructure nécessaire pour exécuter votre application et gérer son déploiement continu.

Gestion de plusieurs environnements avec des modèles AWS CloudFormation

Vous pouvez utiliser des modèles AWS CloudFormation pour modéliser vos composants AWS OpsWorks (piles, couches, instances et applications) et les approvisionner comme piles AWS CloudFormation. Cela vous donne la possibilité de suivre les changements dans votre infrastructure grâce à un outil de contrôle de versions et de partager votre configuration AWS OpsWorks. De plus, vous avez la flexibilité de créer des ressources et services associés, comme Elastic Load Balancing et les bases de données Amazon RDS, avec vos composants AWS OpsWorks grâce à un seul modèle AWS CloudFormation ou des modèles AWS CloudFormation imbriqués. Ces exemples de modèles montrent comment modéliser votre pile AWS OpsWorks à l'aide d'AWS CloudFormation.

Grâce à un modèle AWS CloudFormation, vous pouvez créer plusieurs piles identiques, une pour chaque environnement de déploiement, par exemple les tests et la production. De cette manière, vous veillez à ce que votre application soit testée dans un environnement identique à l'environnement de production. Vous pouvez également définir des paramètres dans votre modèle AWS CloudFormation, afin de pouvoir personnaliser et différencier les piles pour les différents environnements.

Intégration, diffusion, pipelines et déploiement continus

Avec l'intégration continue, les membres d'une équipe de développement intègrent fréquemment leur travail, en général au quotidien. Chaque intégration est créée et ensuite testée pour repérer les erreurs aussi vite que possible.

Avec la diffusion continue, vous créez un logiciel afin qu'il puisse être mis en production à tout moment. Le code peut être déployé tout au long de son cycle de vie, ce qui vous fournit des commentaires rapides et automatiques sur la disponibilité de production de vos systèmes. La diffusion continue est possible en intégrant continuellement votre code, en créant des packages et en exécutant des tests automatiques pour détecter les problèmes. Ensuite, vous déployez ces packages dans un environnement aussi proche que possible de votre environnement de production grâce à un pipeline de déploiement.

Un pipeline de déploiement vous permet de diviser votre processus de création en étapes, avec une confiance accrue à chaque étape. En général, la première étape d'un pipeline de déploiement compile le code et fournit des packages pour les étapes ultérieures. L'étape finale déploie les packages pour la production. La transition entre les étapes peut être automatique ou exiger une autorisation humaine. Vous pouvez généralement repérer des problèmes de création aux premiers stades de votre pipeline, ce qui vous fournit des commentaires plus rapides et vous permet à ce moment de mettre fin au processus. Lors des étapes ultérieures, le dépannage exige une analyse précise et davantage de temps.

Vous pouvez également considérer votre pipeline comme une seule transaction : après le démarrage d'une création, cette dernière est déployée dans un environnement de production en cas de succès, ou alors tous les changements à chaque étape sont restaurés. Cela vous permet de garder une certaine cohérence entre vos environnements à tout moment.

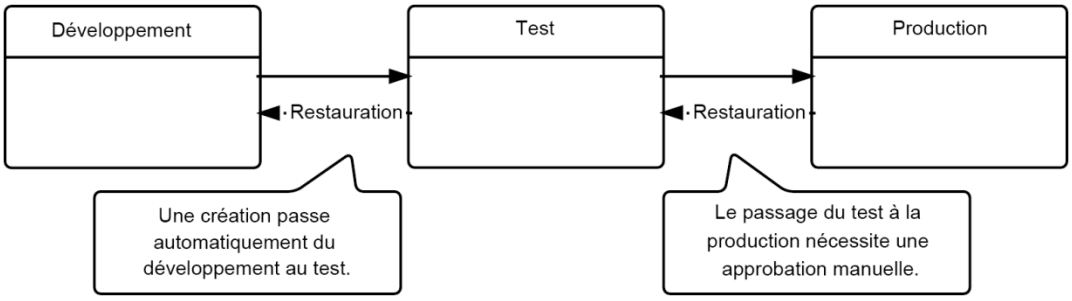

Figure 2 : un exemple de pipeline de déploiement

Le déploiement continue s'appuie sur des concepts de diffusion continue. Avec la diffusion continue, le pipeline ne déploie pas nécessairement de code en production ; vous choisissez d'en déployer ou non. Avec le déploiement continu, chaque changement passant avec succès par le pipeline est automatiquement déployé en production, ce qui entraine généralement plusieurs déploiements par jour.

Tout au long des étapes de votre pipeline de déploiement, vous devez déployer votre code plusieurs fois sur différents environnements. Il existe différentes méthodes de déploiement :

- Préparez une image personnalisée, comme des Amazon Machine Images (AMI) ou un conteneur Docker, qui inclut tous les packages et configurations requis. Grâce aux images préparées, vous pouvez fournir des images cohérentes pour vos environnements et réduire le temps de démarrage d'une instance. Vous pouvez automatiser le processus de préparation grâce à des outils tels que Netflix Aminator, pour créer des AMI personnalisées, ou Packer, qui vous permet également de préparer des images Docker. Ici, vous pouvez trouver un exemple de modèle Packer que vous pouvez utiliser pour préparer des AMI.
- Déployez sur place, ce qui signifie que vous lancez vos instances à l'aide d'AMI de base, que vous installez tous les packages et que vous effectuez la configuration requise grâce aux recettes Chef. Cette méthode peut ralentir le temps de démarrage de l'instance, mais cela vous donne plus de flexibilité lors de la configuration de vos environnements.

Vous pouvez choisir d'implémenter l'une de ces méthodes ou d'utiliser une combinaison des deux : préparer des AMI personnalisées avec les packages et les configurations qui changent moins fréquemment et utiliser des recettes Chef pour configurer les composants les plus dynamiques.

Déploiements sans temps d'arrêt

Lorsque vous déployez sur votre environnement de production plusieurs fois par jour, vous ne voulez pas que votre application subisse de temps d'arrêt. Lors de la session 2014 d'AWS re:Invent « Scalable Site Management Using AWS OpsWorks », nous avons présenté un script qui simplifie et automatise les déploiements sans temps d'arrêt avec AWS OpsWorks.

Le script utilise le kit de développement logiciel AWS pour Python pour faire des appels à l'API AWS OpsWorks. Dès l'exécution, le script circule à travers les instances dans une couche AWS OpsWorks et envoie la commande de déploiement à chacun d'entre eux. Si la couche est associée à un équilibreur de charge Elastic Load Balancing, le script désenregistre tout d'abord l'instance de l'équilibreur de charge, attend la durée configurée par le paramètre de délai d'attente du drainage de la connexion et lance ensuite le déploiement. Une fois le déploiement terminé, il réenregistre l'instance avec l'équilibreur de charge Elastic Load Balancing. Le script lit également la configuration de la vérification d'intégrité de l'équilibreur de charge et attend que l'instance soit considérée comme intègre, c'est-à-dire l'intervalle de la vérification d'intégrité, avant de la remettre en ligne.

Déploiements bleu-vert

Les déploiements bleu-vert sont un moyen efficace de réduire les risques de déploiement. L'environnement existant de production en directe est désigné comme *bleu*, tandis que l'environnement utilisé pour tester une nouvelle version de votre application est désigné comme *vert*. Lorsque la nouvelle version a été testée avec succès et est prête à être mise en ligne, vous pouvez basculer tout le trafic utilisateur de l'environnement bleu vers l'environnement vert. L'environnement bleu peut être inactif et supprimé après un certain temps si aucune restauration n'est requise.

Vous pouvez implémenter des déploiements bleu-vert grâce à AWS OpsWorks en association avec un pool d'équilibreurs de charge Elastic Load Balancing et Amazon Route 53.

- Les environnements bleus et verts sont représentés par des piles AWS OpsWorks. L'environnement bleu exécute la version actuelle, tandis que l'environnement vert exécute la nouvelle version. Avec AWS CloudFormation, chaque pile peut effectuer des changements dans la configuration des logiciels et des ressources. Au départ, les instances reçoivent du trafic uniquement dans la pile bleue.
- Créez un pool d'équilibreurs de charge Elastic Load Balancing pouvant être dynamiquement attaché à une couche dans chaque pile et préchauffé pour s'adapter au volume attendu de trafic.
- Utilisez la fonction routage pondéré fournie par Amazon Route 53 pour créer un ensemble d'enregistrements dans une zone hébergée qui inclut tous vos équilibreurs regroupés. Attribuez un poids nul à tous les équilibreurs de charge inutilisés, et un poids non nul à ceux attachés à votre environnement en direct.

Lorsque vous êtes prêt pour un basculement bleu-vert :

- Créez une pile AWS OpsWorks verte identique à la pile bleue existante ; vous pouvez clôner la pile bleue ou utiliser un modèle AWS CloudFormation. Une fois la pile bleue créée, lancez des instances dans les couches requises et déployez la nouvelle version de votre application.
- Attachez un équilibreur de charge Elastic Load Balancing du pool vers la couche d'application de la pile verte.
- Lorsque les instances apparaissent comme intègres dans l'équilibreur de charge, changez le poids dans l'ensemble d'enregistrements Amazon Route 53 afin que l'équilibreur de charge attaché à l'environnement vert affiche un poids non nul, et que l'équilibreur de charge attaché à l'environnement bleu affiche un poids nul.
- Détachez l'équilibreur de charge Elastic Load Balancing de la couche d'application de la pile bleue afin qu'il retourne dans le pool.
- Lorsqu'il n'est plus nécessaire de conserver la pile bleue, c'est-à-dire qu'une restauration n'est pas requise, supprimez-la.

Supervision

La supervision de vos ressources est considérée comme une bonne pratique et permet aux organisations d'identifier et de résoudre les problèmes d'infrastructure et d'application avant qu'ils n'affectent les processus métier critiques.

AWS OpsWorks vous permet de superviser vos piles et applications via Amazon CloudWatch, qui fournit des métriques comme la charge, le processeur et la mémoire, ou grâce à des outils tiers comme Ganglia ou Nagios.

Amazon CloudWatch

AWS OpsWorks utilise Amazon CloudWatch pour fournir des métriques personnalisées avec une supervision détaillée de chaque instance dans la pile, et présente les données agrégées dans la section de supervision de chaque pile. Vous pouvez consulter les métriques de la pile entière ou d'une couche en particulier, ou vous pouvez zoomer sur une instance spécifique.

Vous pouvez également définir des métriques personnalisées et les publier sur Amazon CloudWatch avec la commande put-metric-data. Amazon CloudWatch stocke des données métriques comme une série de points de données, chacun avec un horodatage associé. Vous pouvez publier un ou plusieurs points de données avec chaque appel à `put-metric-data`, et vous pouvez également publier un ensemble agrégé de points de données, appelé ensemble de statistiques. Une fois les métriques publiées sur Amazon CloudWatch, vous pouvez consulter les graphiques de ces dernières dans l'AWS Management Console.

Pour publier vos métriques personnalisées sur Amazon CloudWatch depuis une pile AWS OpsWorks, vous écrivez habituellement une recette personnalisée pour créer une tâche cron afin de publier vos métriques personnalisées grâce aux commandes de l'interface de ligne de commande (CLI) AWS, aux scripts de supervision Amazon CloudWatch pour Linux ou à n'importe quel kit de développement logiciel AWS disponible. L'exemple suivant montre une recette qui publie les métriques personnalisées de l'espace disque à l'aide des scripts de supervision Amazon CloudWatch. La ressource cron crée une tâche cron pour exécuter le script toutes les cinq minutes :

```
# Install Perl dependecies
#
if platform?("ubuntu")
  package "unzip"
  package "libwww-perl"
  package "libcrypt-ssleay-perl"
  package "libswitch-perl"
elsif platform?("amazon")
  package "perl-Switch"
  package "perl-Sys-Syslog"
  package "perl-LWP-Protocol-https"
end

# Download the Amazon CloudWatch Monitoring Scripts for
Linux
#
remote_file "/opt/CloudWatchMonitoringScripts-v1.1.0.zip" do
  source "http://ec2-downloads.s3.amazonaws.com/cloudwatch-
samples/CloudWatchMonitoringScripts-v1.1.0.zip"
  mode '0644'
end

# Unzip the Amazon CloudWatch Monitoring Scripts for Linux
#
execute "unzip" do
  command "unzip CloudWatchMonitoringScripts-v1.1.0.zip"
  creates "/opt/aws-scripts-mon"
  cwd "/opt"
end

# Add the script to cron so that it runs every 5 minutes
#
cron "cloudwatch-disk-space" do
  hour "*"
  minute "*/5"
  weekday "*"
  command "/opt/aws-scripts-mon/mon-put-instance-data.pl --
disk-space-used --disk-space-avail --disk-space-util -- disk-
path=/ --from-cron"
end
```

Notez que les instances Amazon EC2 dans la pile AWS OpsWorks doivent avoir un rôle AWS Identity and Access Management (IAM) avec une stratégie qui autorise les actions `cloudwatch:PutMetric*` .

Utilisation d'Amazon CloudWatch pour superviser les journaux de pile

Chaque instance de votre pile produit différents fichiers journaux, tels que les journaux système, les journaux d'application et peut-être des journaux personnalisés. Vous souhaitez probablement superviser certains de ces journaux pour repérer des comportements ou des erreurs inattendus, par exemple lorsqu'un serveur d'application génère davantage de code de statut 404 HTTP que prévu. AWS OpsWorks prend en charge Amazon CloudWatch Logs pour superviser les journaux sélectionnés sur plusieurs instances.

Avec Amazon CloudWatch Logs, vous pouvez superviser un journal pour l'occurrence d'un modèle spécifique. Par exemple, vous pouvez superviser vos journaux d'application pour l'occurrence d'un terme littéral comme ERROR. L'agent Amazon CloudWatch Logs envoie les journaux à Amazon CloudWatch Logs, et vous pouvez ensuite utiliser la console Amazon CloudWatch Logs ou la CLI pour configurer des métriques afin qu'elles vous envoient une notification lorsqu'une certaine condition est respectée, par exemple lorsque le nombre d'erreurs dans le journal dépasse un seuil défini.

Pour activer la supervision Amazon CloudWatch Logs sur votre pile AWS OpsWorks, vous devez suivre les étapes suivantes :

- Mettez à jour votre profil d'instance afin que l'agent Amazon CloudWatch Logs dispose des permissions adaptées. Vous pouvez utiliser le même profil mis à jour pour toutes vos instances.
- Créez un fichier de configuration qui précise des détails tels que les journaux à superviser, et installez-le dans le répertoire /tmp de chaque instance.
- Installez et lancez l'agent Amazon CloudWatch Logs sur chaque instance.

Vous pouvez automatiser les deux dernières étapes grâce à des recettes personnalisées pour gérer les tâches requises et les attribuez aux évènements d'installation de la couche appropriée. Chaque fois que vous lancez une nouvelle instance sur ces couches, AWS OpsWorks exécute automatiquement vos recettes après le démarrage de l'instance, ce qui active Amazon CloudWatch Logs. Consultez la section Quick Start: Install the CloudWatch Logs Agent Using AWS OpsWorks and Chef dans la documentation AWS OpsWorks pour plus d'informations.

Couches intégrées et personnalisées Ganglia

Outre l'utilisation d'Amazon CloudWatch pour superviser les ressources de votre pile, vous pouvez également utiliser la couche AWS OpsWorks Ganglia pour une supervision supplémentaire des applications. La couche Ganglia est un plan pour une instance maître Ganglia qui supervise votre pile grâce à la supervision distribuée Ganglia.

En général, une pile inclut seulement une instance maître Ganglia. Les recettes standard AWS OpsWorks installent un client Ganglia à faible charge sur chaque instance. Si votre pile inclut une couche Ganglia, le client Ganglia le signale automatiquement à l'instance maître Ganglia lorsqu'elle est en ligne. L'instance maître Ganglia utilise ces données pour calculer différentes statistiques et affiche les résultats via une interface graphique Web.

La communauté Chef fournit également des livres de recettes pour les autres outils populaires de supervision comme Nagios, Monit et Munin. Vous pouvez facilement ajouter l'un de ces outils à votre pile grâce à une couche personnalisée.

Sécurité

La sécurité est l'exigence fonctionnelle principale qui protège les informations critiques des manipulations accidentelles ou délibérées telles que les vols, les fuites et les suppressions.

AWS fournit une infrastructure globale et des services fondamentaux (calcul, stockage, mise en réseau et base de données) ainsi que des services de niveau supérieur. AWS fournit une gamme de services et de fonctions de sécurité que les clients peuvent utiliser pour sécuriser leurs actifs. Sous le modèle de responsabilité partagée, les clients AWS sont responsables de la protection de leurs données dans le cloud et du respect des exigences de protection des informations.

AWS OpsWorks peut exploiter les fonctions standard de sécurité AWS pour répondre aux exigences de sécurité des clients et fournir un environnement de déploiement sûr.

Amazon Virtual Private Cloud

Amazon Virtual Private Cloud (Amazon VPC) vous permet d'encapsuler des ressources dans un réseau virtuel que vous définissez et qui est isolé de manière logique des autres réseaux virtuels du cloud AWS.

Amazon VPC inclut un ou plusieurs sous-réseaux, chacun avec une table de routage associée qui dirige le trafic en fonction de son adresse IP de destination.

- Les instances au sein d'Amazon VPC peuvent communiquer les unes avec les autres quel que soit leur sous-réseau.
- Les sous-réseaux dont les instances peuvent communiquer avec Internet sont appelés *sous-réseaux publics.* Ils communiquent avec Internet via une passerelle
- Les sous-réseaux dont les instances ne peuvent pas communiquer directement avec Internet sont appelés *sous-réseaux privés*. Ils peuvent communiquer avec d'autres instances dans Amazon VPC, mais doivent communiquer avec Internet via une instance de traduction d'adresses réseau (NAT).

AWS OpsWorks exige qu'Amazon VPC soit configuré de façon à ce que toutes les instances de la pile, notamment celles des sous-réseaux privés, accèdent aux points de terminaison AWS OpsWorks et Amazon S3. Vous devez également accéder au répertoire du package pour votre système d'exploitation et vos autres dépendances telles que les gems Ruby. Pour les instances des sous-réseaux privés, vous pouvez utiliser une NAT pour fournir une connectivité sortante à Internet. Consultez la section Running a Stack in a VPC dans la documentation pour plus d'informations.

Amazon VPC fournit deux fonctions pouvant être utilisées pour augmenter la sécurité de vos ressources :

- Les groupes de sécurité agissent en tant que pare-feu virtuel pour votre instance afin de contrôler le trafic entrant et sortant.
- Les listes de contrôle d'accès réseau (ACL) agissent en tant que pare-feu pour les sous-réseaux associés, en contrôlant le trafic entrant et sortant au niveau du sous-réseau.

Vous pouvez associer un ou plusieurs groupes de sécurité à vos instances, et chaque instance d'Amazon VPC peut appartenir à un ensemble différent de groupes de sécurité. De plus, vous pouvez sécuriser davantage vos instances en ajoutant des ACL réseau aux sous-réseaux d'Amazon VPC.

Gestion de l'accès aux instances

L'accès direct à vos instances via SSH ou RDP ne fait généralement pas partie des bonnes pratiques et doit être évité. Toutefois, à des fins de dépannage et lorsque les informations des journaux ne suffisent pas, il se peut que vous ayez besoin de vous connecter à vos instances. L'utilisation de paires de clés constitue l'option par défaut (et préférée) pour accéder à vos instances car elle réduit les chances pour un individu d'accéder à l'instance en devinant le mot de passe.

La section « Managing OS-level Access to Amazon EC2 Instances » du livre blanc sur les bonnes pratiques AWS en matière de sécurité explique comment les paires de clés peuvent être générées et comment elles s'adaptent au processus de démarrage des instances.

Utilisation des permissions AWS OpsWorks pour configurer les clés SSH publiques des utilisateurs

AWS OpsWorks vous permet de sélectionner des utilisateurs IAM pour chaque pile et de définir les permissions de chaque utilisateurs grâce à la page des permissions AWS OpsWorks ou en attachant une stratégie IAM adaptée. Grâce à la page des permissions AWS OpsWorks, vous pouvez contrôler quels utilisateurs disposent d'un accès SSH et de privilèges sudo sur chaque instance dans une pile AWS OpsWorks. Chaque utilisateur IAM peut enregistrer une clé SSH publique avec AWS OpsWorks. Une fois la clé publique enregistrée pour un utilisateur IAM, vous pouvez accorder des privilèges sur une base par pile afin d'utiliser cette clé pour se connecter aux instances de la pile. Pour chaque instance, AWS OpsWorks crée un utilisateur de système d'exploitation, place la clé publique dans le fichier authorized_keys et met à jour la clé publique si l'utilisateur la modifie.

Hôte bastion

L'utilisation de paires de clés avec un hôte bastion peut-être difficile. Il est notamment préférable d'éviter d'utiliser l'hôte bastion pour stocker des clés privées nécessaires pour se connecter à l'instance. L'une des solutions pour surmonter ce problème est d'utiliser un transfert d'agent SSH sur le client. Cela vous permet de vous connecter depuis l'hôte bastion à vos instances sans avoir besoin de stocker la clé privée sur l'hôte bastion. Les détails AWS Security Blog sur la façon de configurer et d'utiliser le transfert d'agent SSH pour vous connecter à vos instances.

Lorsque vous ajoutez de nouvelles instances à vos piles, vous devez conserver le fichier hôte sur l'hôte bastion à jour. Pour faciliter ce processus, vous pouvez utiliser ce livre de recettes, qui crée une tâche cron pour mettre à jour périodiquement le fichier hôte.

De plus, souvenez-vous des bonnes pratiques suivantes lors de la configuration de votre hôte bastion :

- Lorsque vous configurez le groupe de sécurité sur l'hôte bastion, appliquez le principe de moindre privilège, en autorisant des connexions SSH, c'est-à-dire port TCP/22, uniquement à partir d'adresses IP connues et fiables, telles que votre réseau d'entreprise.
- Vous devez disposer d'un bastion dans chacune des zones de disponibilité contenant vos instances. Si votre déploiement profite d'une connexion Amazon VPC virtual private network (VPN), ayez également un bastion sur site.
- Configurez les instances dans Amazon VPC pour accepter des connexions SSH provenant uniquement d'une instance d'hôte bastion.
- Utilisez l'instance d'hôte bastion uniquement en tant qu'hôte bastion et pas pour autre chose. De plus, vous pouvez renforcer davantage la sécurité des instances, par exemple activez SELinux, utilisez un serveur syslog distant pour les journaux et configurez une détection d'intrusion basée sur l'hôte.

Gestion des secrets

Les secrets, tels que les mots de passe de base de données, les informations d'identification AWS ou d'API tierces, doivent être stockés et accessibles par les serveurs qui en ont besoin. Si un serveur n'a pas besoin d'un secret spécifique, il n'a pas à y accéder.

Des sacs de données AWS OpsWorks Custom JSON ou Chef peuvent être utilisés pour que les nœuds puissent accéder aux données dans une pile AWS OpsWorks. Toutefois, ces méthodes ne sont pas adaptées pour partager des secrets, car ils sont accessibles en clair par tous les utilisateurs de la pile.

Variables d'environnement AWS OpsWorks

Avec AWS OpsWorks, vous pouvez définir jusqu'à 20 variables d'environnement pour chaque application. Ces variables sont transmises aux instances du serveur d'application lors de l'installation de l'instance et peuvent être mises à jour sur chaque déploiement d'application. Des couches personnalisées peuvent utiliser une recette pour récupérer une valeur de variable grâce à une syntaxe de nœud Chef standard et la stocker ensuite sous une forme accessible par les instances de la couche.

Vous pouvez également définir des variables d'environnement comme valeurs protégées, afin qu'elles ne puissent pas être lues par les utilisateurs AWS OpsWorks. Par exemple, vous pouvez configurer des variables d'environnement séparées pour l'identifiant et le mot de passe de votre base de données. La variable d'environnement du mot de passe peut être définie comme une valeur protégée, et peut donc être consultée dans la console, le kit de développement logiciel AWS ou la CLI et être accessible uniquement par une application spécifique via JSON qui est transféré aux instances spécifiques pour être utilisé dans des recettes Chef.

Chiffrement des sacs de données avec un fichier secret sur Amazon S3 avec Server- side Encryption (SSE)

Les sacs de données chiffrés utilisent un secret partagé et un chiffrement symétrique des valeurs du sac de données. Les données sont chiffrées à l'aide du protocole AES-256-CBC grâce à un vecteur d'initialisation aléatoire chaque fois qu'une valeur est chiffrée pour les aider à se protéger de toutes les formes de cryptanalyse. Seules les valeurs de l'élément d'un sac de données sont déchiffrées, tandis que les clés peuvent toujours faire l'objet d'une recherche. Les sacs de données peuvent être chiffrés uniquement par un nœud ou un utilisateur avec le même secret partagé.

Cet exemple montre comment modifier le guide de démarrage de la pile du serveur d'application simple basée sur RDS à partir d'AWS OpsWorks pour utiliser un sac de données chiffré afin de stocker l'identifiant et le mot de passe DB et conserver le fichier secret sur un compartiment Amazon S3 chiffré. Vous devez installer le kit de développement Chef (ChefDK) pour créer le sac de données chiffré à l'aide de la commande `knife` ; suivez les instructions suivantes pour installer ChefDK.

Tout d'abord, créez le fichier secret qui sera utilisé pour chiffrer le sac de données.

```
$ openssl rand -base64 512 | tr -d '\r\n' > secret_file
```

Le fichier de résultat peut être chargé dans un compartiment Amazon S3, en veillant à activer le chiffrement côté serveur pour l'objet.

Veillez à ce que le rôle IAM attaché aux instances de votre pile (le rôle par défaut est `aws-opsworks-ec2-role`) inclut une stratégie pour autoriser l'accès au compartiment Amazon S3 dans lequel vous chargez votre fichier secret :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt112233445566",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::bucket_name/secret_file"
      ]
    }
  ]
}
```

Créez un fichier texte brut JSON contenant les secrets à partager :

```
$ echo "{\"id\": \"rdscredentials\", \"user\":
\"username\", \"password\": \"Passw0rd\"}" >>
plain_text.json
```

Créez et chiffrez le sac de données grâce au fichier secret créé auparavant :

```
$ knife data bag create rdscredentials -z
Created data_bag[rdscredentials]
$ knife data bag from file rdscredentials
/path/to/plain_text.json --secret-file /path/to/secret_file
-z
Updated data_bag_item[rdscredentials::rdscredentials]
```

Modifiez le JSON personnalisé pour la pile pour inclure votre sac de données et les détails de la clé secrète sur Amazon S3.

```
{
  "deploy": {
    "simplephpapp": {
      "database": {
        "database": "my_rds_db",
        "host": "my-rds.xxxxxxxxxx.eu-west-
1.rds.amazonaws.com",
        "adapter": "mysql"
      }
    }
  },
  "opsworks": {
    "data_bags": {
      "rdscredentials": {
        "rdscredentials": {
          "id": "rdscredentials",
          "user": {
            "encrypted_data":
"zVrVESc4NDR9nHSQxY1YLL5aodcx+9r68J1wnJh2tEY=\n",
            "iv": "iDNa3Cr4X0GttXPtoFEOaQ==\n",
            "version": 1,
            "cipher": "aes-256-cbc"
          },
          "password": {
            "encrypted_data":
"cBYeshea0ps9JS4YBNsxt9VQEQTMkNF/q+6B4ZKWmS4=\n",
            "iv": "FtZGMPPOkoxKG9xQ3EF3xg==\n",
            "version": 1,
            "cipher": "aes-256-cbc"
          }
        }
      }
    }
  },
  "secret": {
    "bucket": "bucket_name",
    "object": "secret_file"
  }
}
```

Modifiez la recette appsetup.rb afin que la ressource de modèle lise le fichier secret à partir du compartiment Amazon S3 et que les valeurs provenant du sac de données chiffrées soient utilisées pour compiler le modèle.

```
require 'rubygems'
require 'aws-sdk'

node[:deploy].each do |app_name, deploy|

  script "install_composer" do
    interpreter "bash"
    user "root"
    cwd "#{deploy[:deploy_to]}/current"
    code <<-EOH
    curl -sS https://getcomposer.org/installer | php
    php composer.phar install --no-dev
    EOH
  end

  template "#{deploy[:deploy_to]}/current/db-connect.php"
do
    source "db-connect.php.erb"
    mode 0660
    group deploy[:group]

    if platform?("ubuntu")
      owner "www-data"
    elsif platform?("amazon")
      owner "apache"
    end

    s3 = AWS::S3.new()
    secret =
s3.buckets[node[:secret][:bucket]].objects[node[:secret][:o
bject]].read.strip

    rdscredentials =
Chef::EncryptedDataBagItem.load("rdscredentials",
"rdscredentials", secret)

    variables(
      :host =>      (deploy[:database][:host] rescue nil),
      :user =>      (rdscredentials['user']),
      :password => (rdscredentials['password']),
```

```
      :db =>        (deploy[:database][:database] rescue
nil),
      :table =>     (node[:phpapp][:dbtable] rescue nil)
    )

   only_if do
     File.directory?("#{deploy[:deploy_to]}/current")
   end
  end
end
```

Conclusion

Dans le présent livre blanc, nous avons montré comment vous pouviez utiliser AWS OpsWorks pour gérer des applications multiniveaux complexes, de la conception d'architectures évolutives et flexibles à l'approvisionnement et au déploiement continus d'infrastructures et d'applications. Nous avons également souligné combien la supervision et la sécurité jouaient un rôle important dans de tels déploiements et comment AWS OpsWorks vous permet de gérer facilement ces aspects.

Suggestions de lecture

Pour plus d'informations sur la gestion des applications multiniveaux avec AWS OpsWorks, consultez les sources suivantes.

- Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation
- Continuous Integration and Deployment Best Practices on AWS
- Scalable Site Management Using AWS OpsWorks
- AWS Security Best Practices