Libérez votre DBaaS PostgreSQL

Étienne BERSAC, DALIBO

#PGSession13 - 19 novembre 2020

true

Libérez votre DBaaS PostgreSQL

TITRE : Libérez votre DBaaS PostgreSQL
SOUS-TITRE :

DATE: Étienne BERSAC, DALIBO

SOMMAIRE

- Tour des DBaaS PostgreSQL
- Présentation de cornac.
- Démo !

Questions préliminaires (à distance ???):

- Qui utilise RDS ?
- Qui utilise une solution cloud privée pour l'autre SGBD ?
- Qui automatise le déploiement de base de données ?
- Qui veut déployer PostgreSQL sur kubernetes ?

Dans un premier temps, nous aborderons la définition d'un DBaaS, les critères qui nous intéressent et nous fairons notre marché parmi les solutions actuelles. Ceci expliquera notre nouvelle contribution, le projet cornac.

Avant de passer aux questions, nous aurons une rapide démonstration de la console web du projet.

QU'EST-CE QU'UN DBAAS ?

- DataBase-as-a-Service.
- Service fournissant API & console web.
- Automate du cycle de vie global.
- Libre-service.
- Le retour des infrastructures privées.

DBaaS est un produit de type cloud-computing. C'est-à-dire la dernière génération de service d'hébergement et d'infogérance. Parmi les innovations techniques permettant ce type de service, on compte l'automatisation et les API. Il faut compter aussi la facturation à la consommation.

L'API permet le libre-service, c'est-à-dire que l'utilisateur peut commander lui-même des opérations précises et les paramétrer. Le service valide et exécute automatiquement les opérations. Le service laisse l'utilisateur déclencher des opérations comme la création d'instance, la création de sauvegardes, la restauration, les mise-à-jours majeures et bien sûr la destruction de tout ça. La console web permet de guider l'utilisateur dans le libre-service.

L'API permet l'Infrastructure-as-code, un nouveau mode de consomation de services. Le cloud est un nouveau contrat entre utilisateur et opérateurs.

Depuis quelques temps, on observe un retour vers les infrastructures privées, à l'opposé du mouvement tout-public précédent. Cela s'explique par la volonté des grands acteurs de toucher des marchés dont les contraintes légales empêchent l'hébergement de services et de données sur une infrastructure publique. Et tout simplement ceux qui ont peur du cloud public. L'enjeu financer est énorme.

Néanmoins, le principe reste qu'on donne les clefs de son infrastructure aux grands du cloud. Il s'agit plus de fournir du matériel à AWS que de reprendre la main sur la technologie.

Important, le retours vers les infrastructures privées est aussi l'introduction du contract cloud dans les infras privées. Le cloud a changé les attentes : automatisation, autonomie.

NOTRE DBAAS IDÉAL

- PostgreSQL ! Pas un fork.
- Logiciel libre.
- Exploitation internalisée.
- Compatible, intégrable, accessible.

Quel serait notre DBaaS idéal ? En premier lieu, nous voulons PostgreSQL, bien sûr ! Dans sa version communautaire. Pas un fork, ni un clone. Nous voudrions pouvoir installer des extensions, et d'autres outils propres à chaque entreprise.

Ensuite, nous voulons le déployer sur notre infrastructure, être souverain de l'exploitation et du devenir des données.

Par ailleurs, le service est-il facile à opérer et à utiliser ? Que se passe-t-il en cas de problème ? Les utilisateurs sont-ils autonomes ?

Enfin, il reste la question du coût.

true

TOUR DU MARCHÉ

	Libre	API	Console
AWS Outpost, Azure Stack	NON	OUI	CENTRALE
Ark, Scalefield, …	NON	?	OUI
Operator Crunchy & Zalando	OUI	YAML	NON
solution maison	OUI*	?	?

Dans un premier temps, on peut regarder les grands acteurs du cloud. AWS et Azure sont les seuls à fournir une offre on-premise. Bien sûr, c'est le service le plus opaque et qui sacrifie le plus la souveraineté.

Ensuite, on a les spécialistes qui fournissent souvent une solution cloud privé, souvent basée sur Kubernetes.

Sinon, en libre, nous avons deux projets connus, deux jumeaux : les opérateurs Kubernetes de Crunchy et de Zalando. Ils sont fournis sans interface, avec ou sans client terminal ad-hoc. L'essentiel passe par la fameuse API de Kubernetes et ces kilos de YAML. L'intégration avec une console, un catalogue de service ou les outils d'Infrastructure-as-Code est compliquée.

Reste la solution maison. Il y en a plus qu'on ne croit. Plus ou moins abouties. Au prix d'un développement interne non négligeable, elle offre une grande liberté.

À Dalibo, nous nous sommes penché sur le sujet et voici notre contribution à la communauté, autour ce ces critères : nom de code cornac !

NOUVEAU PROJET : CORNAC

- Service et console pour implémenter un DBaaS.
- Libre et public : gitlab.com/dalibo/cornac.
- Compatible Amazon RDS.
- Adaptable à votre déploiement de PostgreSQL.

Voici notre contribution. À l'instar d'une implémentation S3, nous avons fait le choix d'implémenter l'API RDS. Cette API est mature et l'écosystème autour est très riche.

Nous publions une implémentation de ce service et un console, basée sur le SDK d'AWS RDS, qui consomme directement ce service.

Le service orchestre le cycle de vie des instances et délègue les opérations à un greffon implémentant une API Python ad-hoc, appelée *opérateur*. C'est similaire à ce qu'on a dans Kubernetes, mais c'est basé sur les VM.

DÉMO !

Préparatifs:

- dans un terminal, exécuter `bin/démo` pour tout réinitialiser.
- dans un autre terminal, lancer `make devserver` dans cornac/service
- dans un autre terminal, lancer `make devworker`
- Ouvrir deux terminaux pour la démo, côtes-à-côtes, prêt à lancer aws cli
- Ouverture sur la console web. http://local1.gilwell.virt:8001/cornac
- Plein écran, Zoom à 200%.
- Se connecter à temBoard en admin. https://localhost:8888/, Zoom à 175%.

ATTENTION à ne pas créer plus de 3 VM ! La 5e ne démarre pas faute de RAM.

Fil rouge de démo:

- **Identification avec remi.root@acme.tld** dans la console web.
- Présentation de l'interface :
  - Identifiant de région.
    - Le service gère une région.
  - Databases & Administration
  - Menu utilisateur.
  - Sélecteur de compte.
    - Le service implémente le modèle bastion.
    - Le compte Master centralise les accès.
    - Les bases de données sont regroupées par comptes.
    - Le compte Master contient une seule base, la base de cornac.
    - On donne les accès *assumeRole* par utilisateur.
- **Bascule sur le compte Équipe A**
  - La liste des instances a changé.
  - Liste des sauvegardes.
- **Création d'une instance en HA.**
  - Expliquer chaque champs.
  - Expliquer la création de VM, patroni.
  - Expliquer que ça prend quelques minutes.
- Visiter une autre base.
  - Présenter les détails d'une instance.
  - Présenter les détails de connexion.
  - Présenter les actions.
  - Expliquer la coopération console - temboard
  - **Suivre vers temBoard.**
    - Naviguer un peu dans temBoard : statements.

        * Revenir dans la console.
- **Lancer une sauvegarde.**
- **Lister les sauvegardes.**
    - Présenter les sauvegardes automatiques.
- **Restaurer la sauvegarde initiale.**
- Visite de la page utilisateurs.
- Visite de la page compte.
- Visite de la page Virtual IPs. Une IP est allouée.
- **Revenir au détails de l'instance en HA.**

Bonus:

- aws rds describe-db-instances
- switchaccount 2
- aws rds describe-db-instances
- pqenv ha bash
- psql
- haping
- dans un autre terminal: virsh reboot cornac-ha-65ab377ea
- Observer la bascule et la réintégration.
- dans un autre terminal: virsh reboot cornac-ha-65ab377eb

true

ARCHITECTURE DE CORNAC

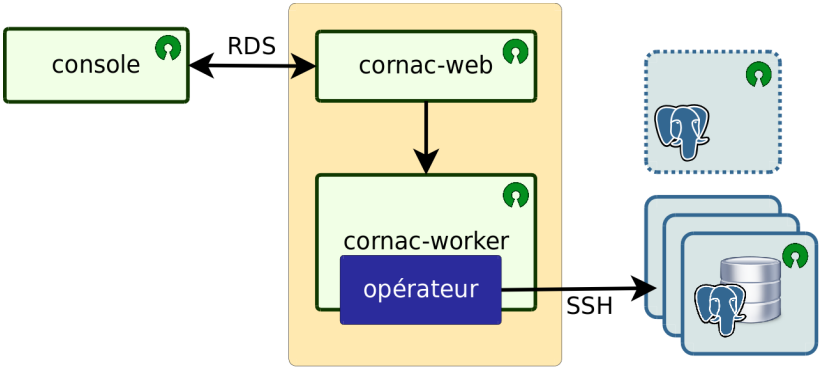

Figure 1: Architecture de cornac

Un opérateur *basic* est fournis, avec des fonctionnalités limitées. La démo suivante utilisera un moteur utilisant l'offre socle Dalibo.

QUESTIONS ? REMARQUES !

- gitlab.com/dalibo/cornac
- DaliboLabs #PGSession13
- etienne.bersac@dalibo.com

N'hésitez-pas à venir vers nous si le projet vous intéresse.