PostgreSQL сервис с возможностями резервного копирования и восстановления данных для Cloud Foundry

Антон Козмирчук, Андрей Кокорев

Аннотация

Cloud Foundry — открытая PaaS платформа, разработанная для разработчиков приложений и помогающая избавиться от рутинной работы с настройкой инфраструктуры. Cloud Foundry предоставляет пользователям возможность запускать приложения в контейнере с доступом к СУБД, фреймворкам для разработки и тестирования. Каждой СУБД необходим сервис брокер, который реализует основной сценарий взаимодействия с приложением: работу с базой данных.

Это статья представляет архитектуру сервис брокера предоставляющего СУБД как сервис с возможностью резервного копирования и восстановления. Сервис брокер реализован для предоставления PostgreSQL и основан на предложенной архитектуре. Предлогается два типа хранения резервных копий: локальный (временный) и хранение в облаке. Разработанная архитектура может быть легко адаптирована для различных СУБД. Данный подход содержит в себе несколько преимуществ: легкая развертка, слабая связность с внутренними структурами СУБД, легкое распределение задач между модулями.

Введение

На данный момент становится очень популярной третья платформа [1] для создания IT-инфраструктуры. Эта платформа основана на различных аппаратных, программных и сетевых технологиях, включая мобильные устройсва, мобильный интернет, социальные сети, облачные инфраструктуры. Это используется для создания всех типов решений для "умных"сервисов.

В целом, третья платформа характеризуется быстрой и технологичной разработкой в 4 сферах: социальные сети, анализ больших данных, быстрый мобильный доступ к интернет, включая корпоративные

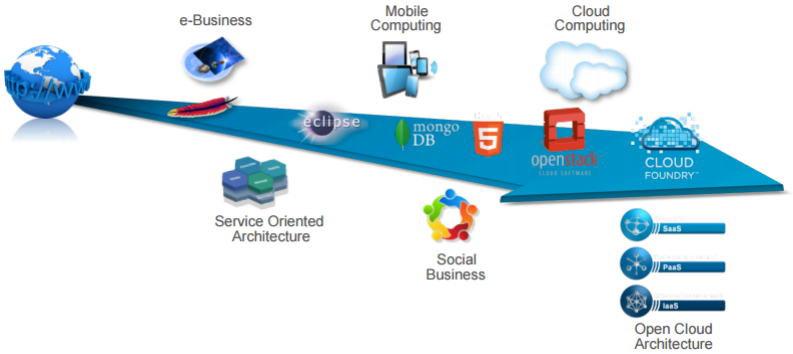

Рис. 1: Open cloud architecture

инфраструктуры, облачные вычисления и сервисы. Пользователи возрастающего количества мобильных устройств производят все больше и больше информации, которую легко хранить в облаках.

Количество мобильных устройств постоянно растет и вместе с ними растет потребность облачного хранения. Облачные технологии предлагают новый тип сервисов — обеспечение хранения и управление СУБД в одном сервисе. Также существует тип облачных сервисов, известный как PaaS (Platform as a Service)

Cloud Foundry используется для создания приложений нового поколения. Эти приложения часто основанны на современных сервисах хранения данных, таких как NoSQL, KV-хранилища, объектные хранилища и HDFS, в дополнение к более традиционным сервисам хранения данных, таких как реляционные базы данных.

Cloud Foundry поддерживает полный цикл разработки приложения, начиная от начальной разработки, через все этапы тестирования, до развертки приложения. Приложения, развернутые в Cloud Foundry, имеют доступ к внешним ресурсам через сервисы. В окружении PaaS, все внешние зависимости, такие как базы данных, сервисы передачи сообщений, файловые системы, являются сервисами.

Цель исследования заключается в разработке дополнительных сервисов Cloud Foundry и улучшение существующих сервисов, предоставляя высокую доступность, защиту данных, снэпшоты, устойчивость к сбоям. В частности, архитектура брокера, который который соеденяется с PostgreSQL и Cloud Foundry, представлена далее. Брокер также предоставляет возможность создания резервных копий для локального и внешнего хранилищ.

Related work

Рост популярности облачных вычислений связан с удобством получения сетевого доступа к общим конфигурируемых вычислительных ресурсов. Облачная вычислительная архитектура состоит из четырех слоев: Аппаратное обеспечение (управление материальными ресурсами облака, включая физические серверы, маршрутизаторы и т.д.), инфраструктура (серверы, системы хранения, сети и программного обеспечения для виртуализации, которые необходимы для поддержки требований вычислительных), платформы (операционной системы и каркасы приложений) и приложений (как правило, с помощью функции автоматического масштабирования для достижения более высокой производительности, доступности и более низкие эксплуатационные расходы). Существуют различные типы услуг облачных вычислений: Программное обеспечение как услуга (SaaS), платформа как услуга (PaaS) и инфраструктура как услуга (IaaS).

Развитие сферы облачных технологий стало причиной появления различных PaaS платформ для самых разнообразных целей. Большинство современных PaaS платформ, таких как Heroku, Google AppEngine, Amazon Elastic MapReduce, Microsoft Azure, Cloud Foundry, предоставляют различные окружения для разработки разнообразного программного обеспечения. Несмотря на похожую модель предоставления окружения, PaaS платформы не дублируют друг друга. Основными различиями выступают выбор фреймворков, выбор языков программирования, набор доступных сервисов, способ распространения.

Heroku — проприетарная платформа ориентированная на разработку Web-приложений, с большим выбором языков программирования, таких как JavaScript, Ruby, JVM languages, Go, PHP, Python. Эта PaaS обладает большим количеством сервисов, а также возможностью добавления своих собственных.

Google App Engine — предоставляет проприетарный сервис для разработки Web-приложений на языках Java, Go, Python, PHP с использованием сервисов от Google, таких как Google Search, Security Scanner, Google Cloud SQL. Предоставляется NoSQL хранилище объектов и Memcache.

Yandex Cocaine — быстрорастущая PaaS платформа для Web разработки с поддержкой большинства популярных языков программирования и контейнеров. Также платформой поддерживаются различные модули, такие как Elasticsearch, URL Fetcher, Elliptics и MongoDB.

Amazon Elastic MapReduce — проприетарная платформа, предостав-

ляющая окружения для разработки приложений, использующих модель MapReduce.

Microsoft Azure — проприетарная платформа, ориентированная на .NET Framework, при этом поддерживающая большинство популярных языков и фреймворков, предоставляющая окружение для разработки Web и Enterprise приложений. Обладает большим выбором сервисов СУБД, объектных хранилищ, сервисов архивации.

Cloud Foundry — открытая PaaS платформа, предлагающая поддержку множества языков программирования, различные сервисы для хранения и обработки данных. Cloud Foundry работает на большинстве популярных IaaS платформ и предоставляет создание собственных сервисов.

Системы обработки и хранения данных являются неотъемлимой частью практически любого приложения. В этом качестве успешно используются реляционные СУБД, такие как Oracle, MySQL, Microsoft SQL Server, PostgreSQL и другие. Также в настоящее время приобретают популярность различные NoSQL решения: MongoDB, Cassandra, Redis и другие. Облачным приложениям, так же, как и настольным, требуются такие системы. Для использования этих систем в облаках требуются специальные адаптации, называемые сервис-брокерами.

Современные реляционные СУБД, не смотря на молодые NoSQL решения, не теряют собственной актуальности и занимают твердые позиции при решении широкого круга задач. Например, на момент написания статьи для PostgreSQL (одна из наиболее популярных реляционных СУБД согласно рейтингу сайта db-engines.com[16]) существует всего несколько сервис-брокеров для облачной платформы Cloud Foundry. Однако, с нашей точки зрения, каждый из них обладает недостатками, которые мы подробнее обсуждаем далее.

Cloud Foundry

Cloud Foundry — PaaS платформа для облачных приложений, стремящаяся создать новый уровень абстрации для виртуального окружения. Cloud Foundry предоставляет возможность запускать многоцелевые приложения, которые не зависят от конкретной инфраструктуры.

Cloud Foundry предоставляет пользователям возможность запускать приложения в контейнере с различными микро-сервисами, такими как операционные системы, СУБД, фреймворки для разработки и тестирования. Контейнеры приложений являются средством логиче-

ской изоляции приложения. Операционная система виртуализируется в контейнере со всеми ресурсами и услугами, которые могут масштабироваться по мере необходимости.

Сервисы Cloud Foundry

Сервисы Cloud Foundry позволяют приложениям использовать ресурсы, предоставленные внешними источниками. Каждый сервис состоит из программного обеспечения, которое предоставляет определенный ресурс и сервис брокера — приложения, которое реалищует интеграцию сервиса с Cloud Foundry. Сервис брокер предоставляет каталог сервисов и планов, которые являются экземплярами определенных услуг. Брокер должен иметь возможность резервировать ресурсы для приложений (т.е. создавать экземпляр сервиса) и привязывать приложения к этим экземплярам (т.е. предоставить доступ для приложения и дать необходимые мета-данные для связи с сервисом).

Сервис брокер должен реализовывать API, который использует для взаимдойствея с Cloud Foundry (Рис. 2). API состоит из 5 функций: получение предлагаемых услуг, создание экземпляра сервиса, удаление экземпляра сервиса, привязки экземпляра сервиса к приложению, отвязывание экземпляра сервиса от приложения. Любая допустимая реализация этого API является сервис брокером, который может быть развернут как пользовательское приложение или установлен с помощью BOSH на виртуальную машину, или быть развернут дистанционно.

PostgreSQL и Cloud Foundry

Существует несколько простых способов интегрировать PostgreSQL [3] в Cloud Foundry. Первый из них представляет собой ручное управление СУБД, при котором нужно добавлять отдельный экземпляр сервиса для каждого приложения, который обеспечивает единственный экземпляр базы данных. Другой способ заключается в использовании платформы SaaS ElephantSQL [4], которая позволяет использовать базы данных на внешних серверах для приложений. Есть также открытые реализации брокеров, которые позволяют получить ресурсы базы данных для приложения.

Первый способ имеет недостаток: администратор должен вручную создать базу данных для каждого приложения, которому необходимы

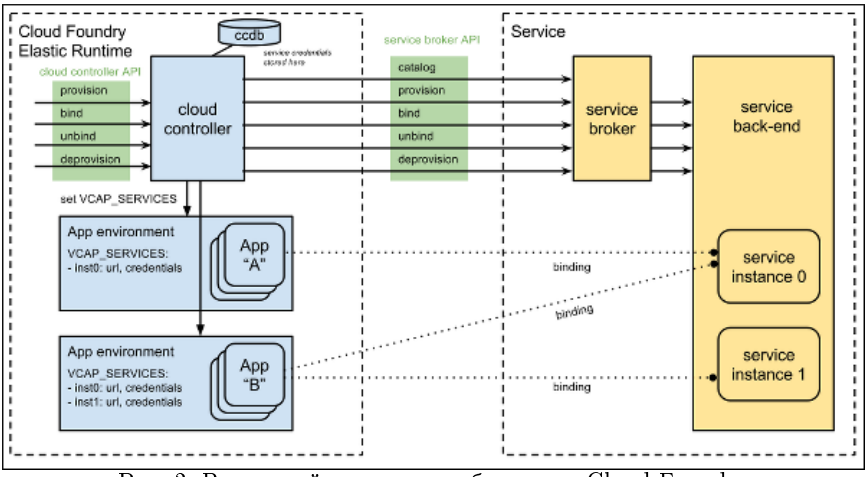

Рис. 2: Взаимодействие сервис брокера с Cloud Foundry

ресурсы СУБД, а также настроить БД для конкретных потребностей приложения. Это утомительная и сложная задача, которая требует специальных знаний SQL диалекта, используемого в PostgreSQL. Преимущества этого метода включают возможность настроить базу данных произвольным образом в соответствии с требованиями приложения.

Использование платформы SaaS ElephantSQL позволяет забыть о ручной конфигурации базы данных и поддержке ее эффективности. ElephantSQL предлагает широкий спектр тарифов, которые определяют параметры прежоставляемого сервиса. Кроме того, пользователь имеет возможность создать резервную копию для восстановления данных приложения. К недостаткам этого метода можно отнести проприетарность ElephantSQL: каждый пользователь должен совершать ежемесячный платеж за использование ресурсов, а так же невозможность прямого доступа к СУБД. Таким образом, разработчик приложения не имеет возможности для настройки базы данных под свои специфические нужды.

Третий подход, предусматривающий использование брокеров с открытым исходным кодом хорош тем, что разработчик может легко получить необходимые ресурсы, и администратор Cloud Foundry может добавить новые функциональные возможности, которые могут понадобиться в приложениях. К сожалению, на момент написания статьи не существует брокера с открытым исходным кодом, который поддерживал бы создание резервных копий.

Таблица 1: Сравнение решений

	Легкая развертка	Точная настройка	Открытость	Backup & Restore
ElephantSQL	+	-	-	+
Ручной подход	-	+	?	-
Другие сервис брокеры	+	-	+	-

Общая архитектура проекта

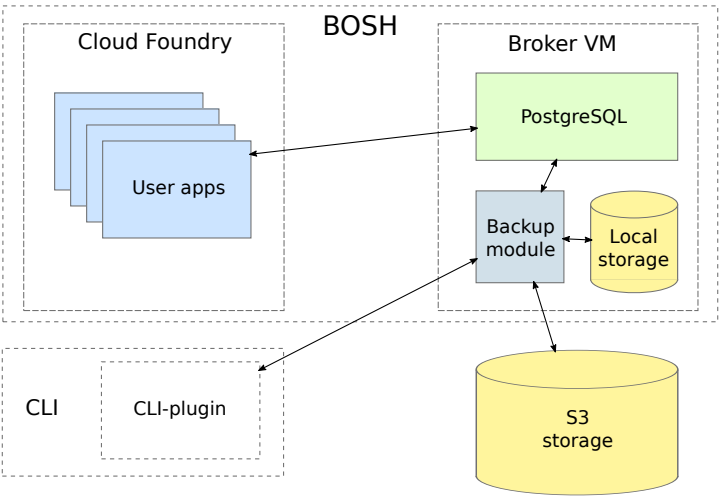

Рис. 3: Общая архитектура

Проект состоит из двух отдельных частей: плагин для интерфейса командной строки (CF CLI) и сервис-брокер (Рис. 3). Плагин может быть установлен любым пользователем CF CLI. Брокер - это так называемая BOSH job (которая разворачивается в виртуальную машину инфраструктуры), подразделяющаяся на RESTful-сервис, PostgreSQL в стандартной конфигурации, модуль бэкапа и локальное хранилище. Также брокер может использовать внешнее хранилище через S3-API.

Брокер

Брокер состоит из следующих частей:

- CF-брокер. Обеспечивает связь CF $\leftrightarrow$ Сервис брокер.
- Модуль бэкапа. Занимается резервным копированием, восстановлением и передачей данных.
- Интерфейс хранилищ. Описывает способы взаимодействия с хранилищами.

Хранилища бэкапов

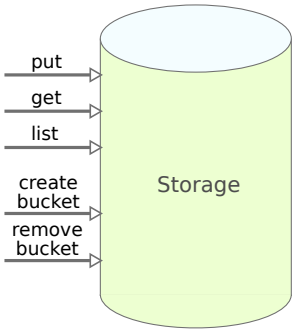

Рис. 4: API хранилищ

Мы предлагаем две возможности для сохранения резервных копий: локальное и внешнее хранилища. Брокер взаимодействует с хранилищами при помощи несложного API (Рис. 4). Хранилища предназнчены для разных потребностей (сравнение в Таблице 2):

Локальное хранилище позволяет делать резервное копирование быстрее (т.к. не происходит передачи данных через сеть) и может быть использовано в качестве временного, если внешнее хранилище недоступно. Мы должны заметить, что это хранилище не предназначено для длительного хранения, в некторых случаях данные могут быть утеряны, например если будет перезапущена виртуальная машина брокера.

Внешнее (поддерживающее S3-API) хранилище разработано для надежного хранения данных. Резервная копия сохраняется непосредственно в хранилище, не создавая локальных копий. Для использования этого хранилища требуется надежное сетевое соединение.

Таблица 2: Сравнение локального и внешнего хранилищ

Хранилище	**Локальное**	**Внешнее**
Продолжительность создания копии	фиксированное	зависит от качества соединения
Надежность	ненадежное	надежное
Доступность	всегда	зависит от внешних факторов
Дисковое пространство	малое, ограничено платформой	потенциально неограниченное

Модуль резервного копирования

Резервное копирование и восстановление данных производятся при помощи стандартных утилит, распространяемых вместе с PostgreSQL: *pg_dump*, *pg_restore*. Этот подход позволяет сохранить все гарантии, предоставленные утилитами (такие как консистентность полученной копии и не-блокирующее выполнение). К тому же, процессы резервного копирования и восстановления остаются прозрачными для опытных администраторов баз данных.

Когда брокер получает запрос резервного копирования, он осуществляется при утилитой *pg_dump* и сохраняется сразу в указанное хранилище. В случае успеха хранилище возвращает имя только что созданного файла и время создания. Иначе пользователю возвращается ошибка, произошедшая в утилите или хранилище (например если не осталось свободного места на диске). Аналогичный процесс происходит при восстановлении данных.

Модуль CF-брокера

Модуль CF-брокера - это RESTful сервис, реализующий API брокера платформы CF: он обрабатывает запросы catalog, provision, deprovision, bind и unbind. В разработаннм брокере каждый экземпляр сервиса (в терминах платформы) соответствует базе данных в PostgreSQL.

При получении запросов от компоненты cloud controller платформы CF брокер производит следующий операции:

1. catalog: возвращает определенную мета-информацию

2. provision: брокер создает новую базу данных (CF cloud controller создает новый экземпляр сервиса)

3. bind: брокер создает нового пользователя в соответствующей базе данных и возвращает jdbc url, содержащий необходимые данные для подключения приложения

4. unbind: удаление пользователя, с этого момента база данных становится недоступной приложению

5. deprovision: база данных и все пользователи удаяются (CF cloud controller удаляет экземпляр сервиса)

Для нужд резервного копирования и восстановления данных мы разработали специальный API:

1. `/backup_instance/{instance_id}?storage={storage_type}`
   Создать резервную копию в указанном хранилище и получить ее имя

2. `/restore_instance/{instance_id}/{backup_name}?storage={storage_type}`
   Восстановить данные из указанной резервной копии, находящейся в указанном хранилище

3. `/upload_backup/{instance_id}/{backup_name}?storage={storage_type}`
   Загрузить резервную копию в указанное хранилище

4. `/download_backup/{instance_id}/{backup_name}?storage={storage_type}`
   Скачать указанную резервную копию

5. `/backups/{instance_id}`
   Получить список доступных резервных копий

В запросах выше используются следующие параметры:

- `instance_id`
  Уникальный идентификатор экземпляра сервиса, предоставляется CF cloud controller

- `backup_name`
  Имя резевной копии

- `storage_type` Тип хранилища, опциональное. Возможные значения: 'local', 's3'. Значение по умолчанию – 'local'.

Плагин командной строки

Плагин для CF CLI выполняется на компьютере пользователя и имеет аналогичный доступ к API платформы. Это позволяет пользователям автоматизировать различные последовательности действий. В этой работе плагин используется как инструмент взаимдействия пользователя с сервис-брокером. Плагин предоставляет набор команд для вызова API брокера.

Результаты

В этой статье мы решаем проблему сохранности данных в облачных приложениях при помощи резервного копирования в СУБД. Множество современных облачных платформ используют модель сервис-брокеров для предоставления приложениям различных СУБД. В нашем исследовании мы используем Cloud Foundry – платформу с открытым исходным кодом. Мы разработали сервис-брокер PostgreSQL с возможностями резервного копирования и восстановления данных.

Разработанная нами архитектура брокера может быть легко адаптирована к различным СУБД. У нашего подхода различные достоинства: простота дальнейшего развития, ограниченная зависимость от структуры СУБД, четкое разделение модулей. Незначительные недостатки архитектуры вызваны ограничениями платформы, которые могут быть исправлены в будущем.

Список литературы

[1] Frank Gens. The 3rd Platform: Enabling Digital Transformation, `http://www.tcs.com/SiteCollectionDocuments/White-Papers/3rd-Platform-Enabling-Digital-Transformation.pdf`

[2] Cloud Foundry Documentation, `http://docs.cloudfoundry.org/`

[3] PostgreSQL Documentation, `http://www.postgresql.org/docs/`

[4] ElephantSQL Documentation, `https://www.elephantsql.com/docs/index.html`

[5] AWS Documentation, `https://aws.amazon.com/documentation/`

[6] Microsoft Azure Documentation, https://azure.microsoft.com/en-us/documentation/

[7] Product Documentation for OpenShift Enterprise, https://access.redhat.com/documentation/en/openshift-enterprise/

[8] IBM Bluemix Documentation, https://www.ng.bluemix.net/docs/

[9] Salesforce Developers Documentation, https://developer.salesforce.com/docs/

[10] 20 Thiago Cordeiro, Douglas Damalio, Nadilma Pereira, Patricia Endo, Andre Palhares, Glauco Gonçalves, Djamel Sadok, Judith Kelner, Bob Melander, Victor Souza, Jan-Erik Mangs. Open Source Cloud Computing Platforms. 2010 Ninth International Conference on Grid and Cloud Computing. IEEE, 2010. ———–XCP, Eucalyptus and Open Nebula

[11] Z. Shu-Qing, Xu Jie-Bin. The Improvement of PaaS Platform, First International Conference on Networking and Distributed Computing. IEEE, 2010.

[12] M. Boniface, B. Nasser, J. Papay, S. C. Phillips, A. Servin, X. Yang, Z. Zlatev, S. V. Gogouvitis, G. Katsaros, K. Konstanteli, G. Kousiouris, A. Menychtas, and D. Kyriazis, "Platform-as-a-Service Architecture for Real-Time Quality of Service Management in Clouds," in Proceedings of the 2010 Fifth International Conference on Internet and Web Applications and Services, Washington, DC, USA, 2010, pp. 155–160.

[13] Nadir K Salih, Tianyi Zang. Variable service process for SaaS Application. Research Journal of Applied Sciences, Engineering and Technology.2012.

[14] Daniel Krook. OpenStack and Cloud Foundry - Pair the leading open source IaaS and PaaS

[15] B. Furht and A. Escalante, Handbook of Cloud Computing. Springer, 2010.

[16] http://db-engines.com/en/ranking/ (March 24, 2016)