PostgreSQL для администраторов баз данных и разработчиков

otus.ru

Меня хорошо видно & слышно?

Защита проекта
Тема:
Создание и тестирование высоконагруженного, отказоустойчивого кластера PostgreSQL на базе Patroni в Яндекс облаке

Ананьев Дмитрий

Архитектор
МТС Digital

План защиты

- Цели проекта
- Что планировалось
- Используемые технологии
- Что получилось
- Схемы/архитектура
- Выводы

Цели проекта

Какие цели вы поставили и какие задачи решили своим проектом

1. Приобрести основные навыки работы с инструментарием кластера Patroni
2. Настроить собственную недорогую инфраструктуру для экспериментов с кластером Patroni доступную из любой точки (в ЯО)
3. Изучить общую теоретическую часть кластера Patroni, HA Proxy, ECTD
4. Изучить технологии, связанные с кластером Patroni

Что планировалось

Что было в начале, что знали до курса, сколько времени заняло выполнение проекта

1. Создать кластер Patroni в Яндекс облаке на одной ВМ на базе Docker Swarm, чтобы иметь возможность масштабировать на несколько ВМ.
   Заряжай Patroni. Тестируем Patroni + Zookeeper кластер (Часть первая)
   Заряжай Patroni. Тестируем Patroni + Zookeeper кластер (Часть вторая)
   Не удалось настроить сервис Zookeeper в DockerSwarm.
2. Создать кластер Patroni в Docker наиболее простым способом:
   Создание масштабируемой и высокодоступной системы Postgres с помощью Patroni 3.0 и Citus
3. Протестировать кластер Patroni переключением, отключением нод
4. Протестировать кластер под небольшой нагрузкой
5. Загрузить на кластер тестовую БД и поработать с ней

Используемые технологии

1. Docker Compose, Portainer, Docker Swarm
2. Patroni, Citus, PostgreSQL, HA Proxy, etcd
3. GitHub, DockerHub

Какие технологии использовались и
какое у вас мнение о новых технологиях

HA Proxy

Серверное программное обеспечение для обеспечения высокой доступности и балансировки нагрузки для TCP- и HTTP-приложений посредством распределения входящих запросов на несколько обслуживающих серверов.[1] Программа написана на C[2].

HAProxy - это бесплатное, очень быстрое и надежное решение для обратного прокси-сервера, балансировки нагрузки и проксирования для приложений на основе TCP и HTTP. Он особенно подходит для веб-сайтов с очень высоким трафиком и поддерживает значительную часть самых посещаемых сайтов в мире. С годами он стал де-факто стандартным балансировщиком нагрузки с открытым исходным кодом, теперь поставляется с большинством основных дистрибутивов Linux и часто развертывается по умолчанию на облачных платформах.

Citus

Citus горизонтально масштабирует PostgreSQL на нескольких компьютерах, используя сегментирование и репликацию. Его механизм запросов распараллеливает входящие SQL-запросы на этих серверах, чтобы обеспечить человеку ответы в режиме реального времени (менее секунды) на больших наборах данных.

Citus - это, по сути, беспроблемный Postgres, созданный для масштабирования. Это расширение для Postgres, которое распределяет данные и запросы в кластере из нескольких компьютеров. В качестве расширения (а не форка) Citus поддерживает новые версии PostgreSQL, позволяя пользователям извлекать выгоду из новых функций, сохраняя совместимость с существующими инструментами PostgreSQL.

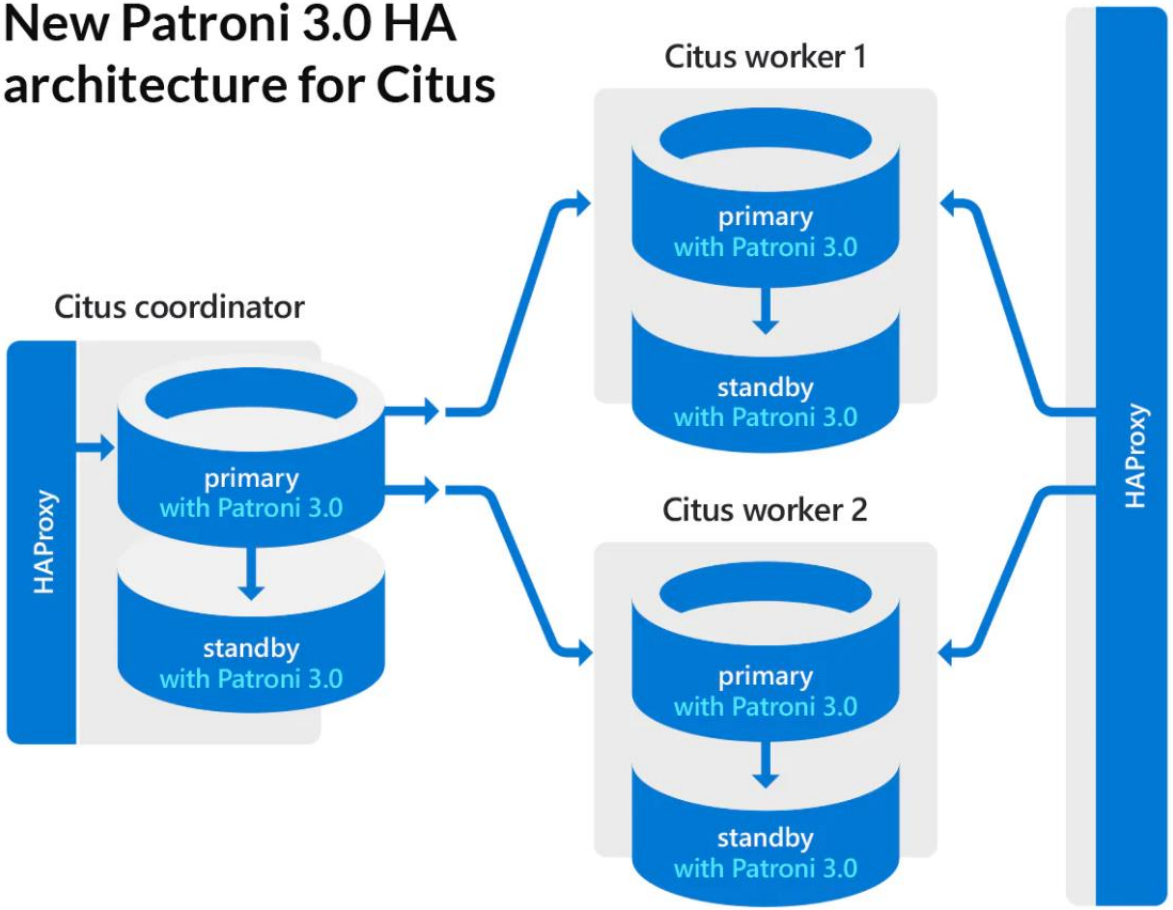
Patroni

Patroni — это автоматическая система аварийного переключения для PostgreSQL. Patroni обеспечивает автоматическое и ручное переключение при отказе и хранит все важные данные в распределенном хранилище конфигурации (DCS) на базе систем ETCD, Consul и т.д.. Соединения приложения и базы данных не осуществляются напрямую, а маршрутизируются через прокси-сервер соединения, такой как HAProxy. Прокси определяет активный/главный узел, который в данный момент времени готов обрабатывать соединения. Использования прокси-сервера сводит к 0 шансы встретить split-brain в кластере баз данных.

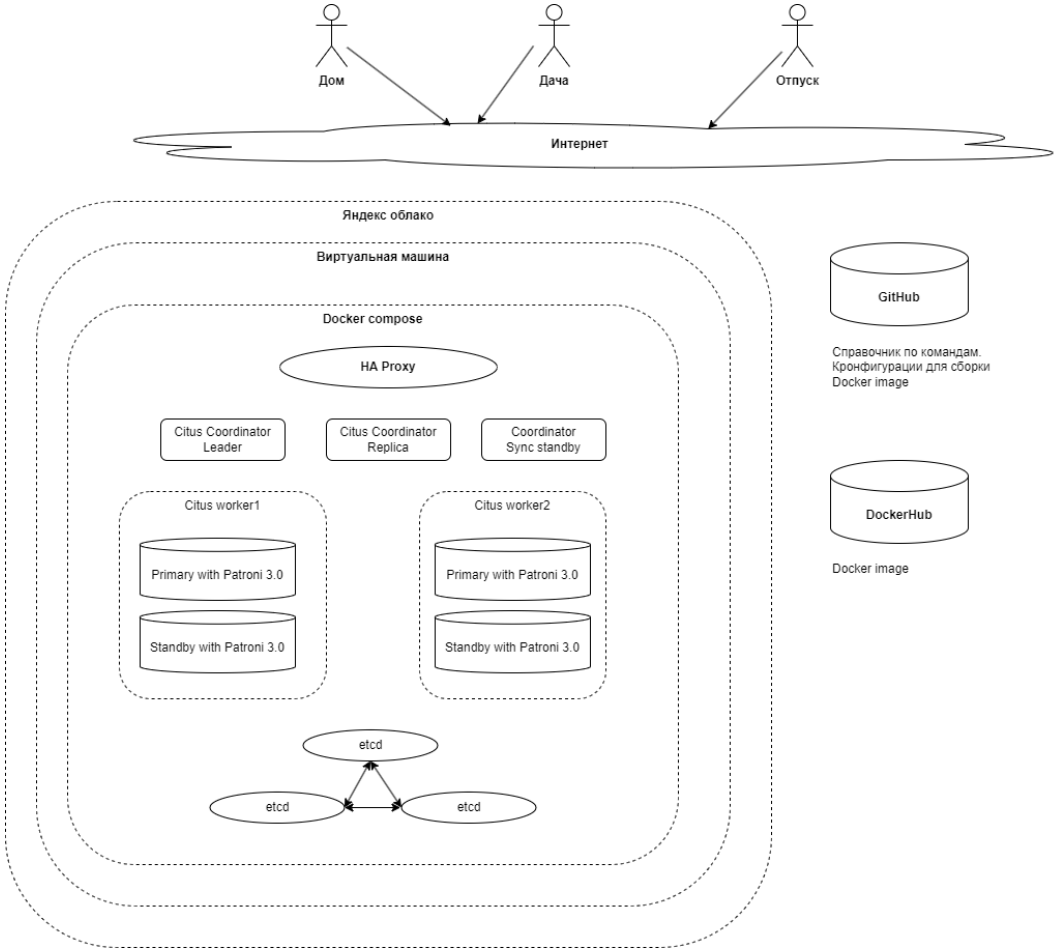
etcd

Строго согласованное распределенное хранилище ключей-значений, обеспечивающее надежный способ хранения данных, доступ к которым требуется распределенной системе или кластеру компьютеров. Он корректно обрабатывает выборы лидера во время сетевых разделов и может допускать сбой компьютера даже в узле лидера.

New Patroni 3.0 HA architecture for Citus



Архитектура проекта



Что получилось

1. Скрины основных экранов приложения и действий

или

1. Демонстрация приложения и исходных кодов

или

1. Ссылка на репозиторий с исходными кодами или просто удачные кусочки

Switchover / ручное переключение /

workers

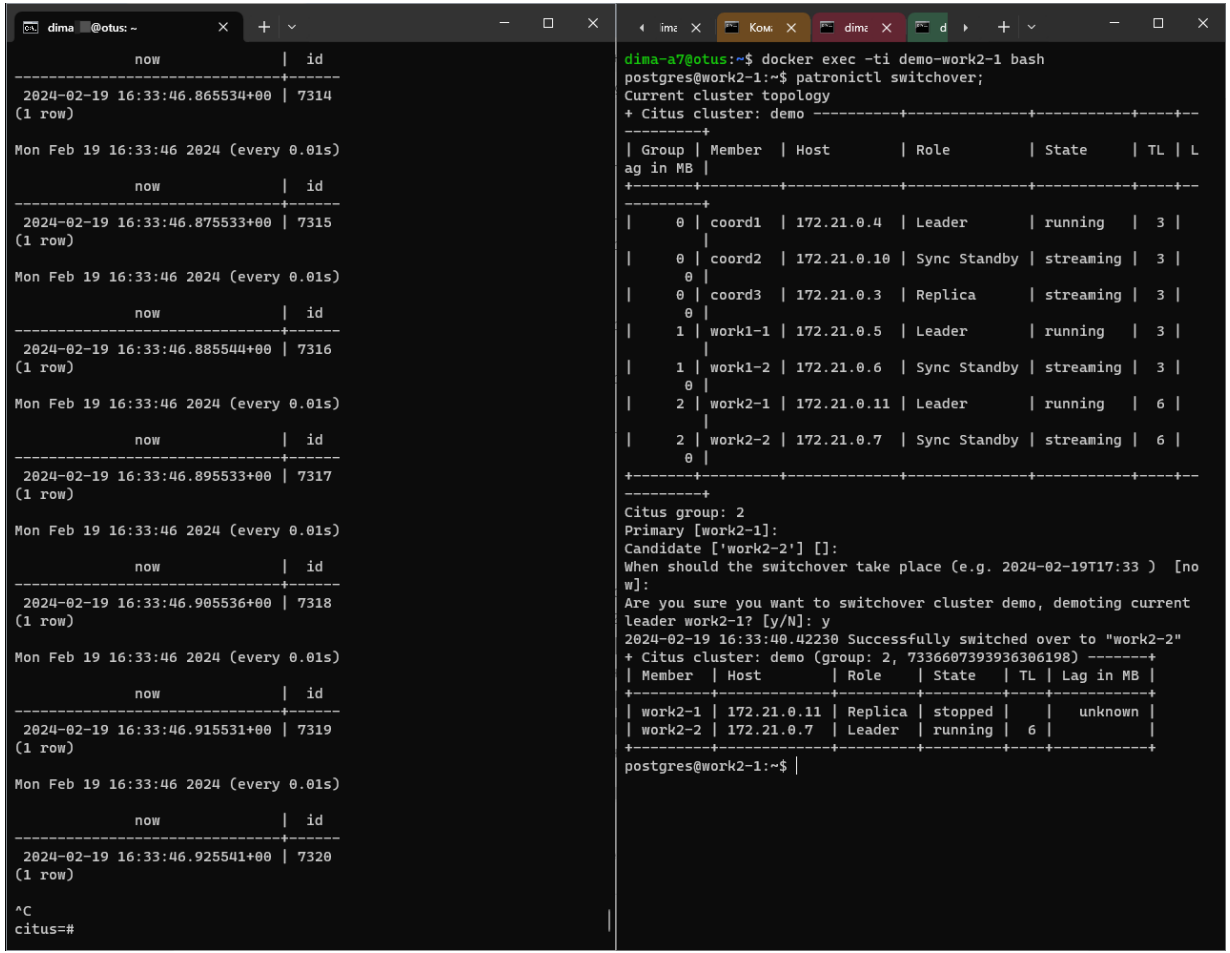
	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	0	-	0	0	100	0			0	0	0	0	0					OPEN								
work1-1	0	0	-	0	0		0	0	100	0	0	?	0	0		0		0	0	0	0	3h14m UP	L7OK/200 in 46ms	1/1	Y	-	1	1	15s	-
work1-2	0	0	-	0	0		0	0	100	0	0	?	0	0		0		0	0	0	0	3h14m DOWN	L7STS/503 in 47ms	1/1	Y	-	1	1	3h14m	-
work2-1	0	0	-	0	0		0	0	100	0	0	?	0	0		0		0	0	0	0	29s UP	L7OK/200 in 46ms	1/1	Y	-	1	1	3h14m	-
work2-2	0	0	-	0	0		0	0	100	0	0	?	0	0		0		0	0	0	0	26s DOWN	L7STS/503 in 46ms	1/1	Y	-	4	2	42s	-
Backend	0	0		0	0		0	0	10	0	0	?	0	0	0	0		0	0	0	0	3h14m UP		2/2	2	0		1	14s	

$patronictl switchover
Citus group: 2

workers

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	0	-	0	0	100	0			0	0	0	0	0					OPEN								
work1-1	0	0	-	0	0		0	0	100	0	0	?	0	0		0		0	0	0	0	4h31m UP	L7OK/200 in 46ms	1/1	Y	-	1	1	15s	-
work1-2	0	0	-	0	0		0	0	100	0	0	?	0	0		0		0	0	0	0	4h31m DOWN	* L7STS/0 in 46ms	1/1	Y	-	1	1	4h31m	-
work2-1	0	0	-	0	0		0	0	100	0	0	?	0	0		0		0	0	0	0	1h15m DOWN	L7STS/503 in 46ms	1/1	Y	-	4	2	4h29m	-
work2-2	0	0	-	0	0		0	0	100	0	0	?	0	0		0		0	0	0	0	1h15m UP	L7OK/200 in 46ms	1/1	Y	-	4	2	1m36s	-
Backend	0	0		0	0		0	0	10	0	0	?	0	0	0	0		0	0	0	0	4h31m UP		2/2	2	0		1	14s	

Переключение на ходу



Failover отключением контейнера

workers

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	0	-	0	0	100	0			0	0	0	0	0					OPEN								
work1-1	0	0	-	0	0		0	0	100	0	0	?	0	0		0		0	0	0	0	4h31m UP	L7OK/200 in 46ms	1/1	Y	-	1	1	15s	-
work1-2	0	0	-	0	0		0	0	100	0	0	?	0	0		0		0	0	0	0	4h31m DOWN	* L7STS/0 in 46ms	1/1	Y	-	1	1	4h31m	-
work2-1	0	0	-	0	0		0	0	100	0	0	?	0	0		0		0	0	0	0	1h15m DOWN	L7STS/503 in 46ms	1/1	Y	-	4	2	4h29m	-
work2-2	0	0	-	0	0		0	0	100	0	0	?	0	0		0		0	0	0	0	1h15m UP	L7OK/200 in 46ms	1/1	Y	-	4	2	1m36s	-
Backend	0	0		0	0		0	0	10	0	0	?	0	0	0	0		0	0	0	0	4h31m UP		2/2	2	0		1	14s	

$ docker stop demo-work2-2

workers

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	0	-	0	0	100	0			0	0	0	0	0					OPEN								
work1-1	0	0	-	0	0		0	0	100	0	0	?	0	0		0		0	0	0	0	4h41m UP	L7OK/200 in 46ms	1/1	Y	-	1	1	15s	-
work1-2	0	0	-	0	0		0	0	100	0	0	?	0	0		0		0	0	0	0	4h41m DOWN	L7STS/503 in 46ms	1/1	Y	-	1	1	4h41m	-
work2-1	0	0	-	0	0		0	0	100	0	0	?	0	0		0		0	0	0	0	6s UP	L7OK/200 in 46ms	1/1	Y	-	4	2	4h40m	-
work2-2	0	0	-	0	0		0	0	100	0	0	?	0	0		0		0	0	0	0	1h26m UP 1/3 ↓	* L4TOUT in 3001ms	1/1	Y	-	6	2	1m36s	-
Backend	0	0		0	0		0	0	10	0	0	?	0	0	0	0		0	0	0	0	4h41m UP		3/3	3	0		1	14s	

SplitBrain

Остановка Patroni на узле Standby

Создание на Standby таблицы

Включение Patroni на узле Standby

Должно быть автоматическое восстановление. Таблица удалена.

Выводы и планы по развитию

1. Удалось создать относительно недорогой стенд в ЯО
2. Планирую использовать Citus при наработке опыта работы с шардированием
3. Планирую освоить Vagrant + Docker Swarm

Запланируйте пару минут на рефлексию в конце защиты проекта и расскажите о планах по развитию

Спасибо за внимание!