ОС UNIX. Лекция 5
«Хитрости» микросервисной архитектуры

Михаил Пожидаев

2 октября 2023 г.

Микросервисная архитектура

Компоненты рассматриваемой архитектуры:

1. Контроллер: центральный узел, получающий команды от внешних клиентских соединений.
2. Брокер сообщений: диспетчер обмена сообщениями между контроллером и остальными контейнерами.
3. Контейнеры-работники: динамически изменяемое множество контейнеров разных типов для выполнения работ, необходимых для обслуживания запросов.

Задача «горячего» обновления

Требования к решению задачи «горячих» обновлений (rolling updates):

1. Контейнеры, не являющиеся контроллером и брокером, периодически получают обновления. Чем чаще, тем лучше.
2. Работу сервиса останавливать нельзя, пользователи должны наблюдать непрерывное функционирование с нормальной обработкой запросов.
3. Близкой является задача динамического изменения набора контейнеров-работников для увеличения или уменьшения задействованных вычислительных мощностей.

Процедура

Порядок выполнения обновления

1. Механизм реализуется в связке с используемым брокером, выполняющим обслуживание межконтейнерного взаимодействия.
2. Инструмент оркестрации получает команду изменить конфигурацию системы, определяет набор контейнеров для остановки.
3. Контейнеры получают сигнал завершить свою работу (`SIGTERM`), после чего уведомляют брокер о своём новом статусе.
4. Контейнеры завершают выполнение тех задач, которые были в работе на момент получения сигнала.
5. Контейнеры либо добровольно завершают свою работу, либо их убивают по timeout при помощи `SIGKILL`.

Команды и ключи Docker

Основная команда обновления конфигурации:
`docker service update --image`

Пара значимых параметров:

- `update-delay`: задержка между успешными обновлениями контейнеров;
- `stop-grace-period`: задержка после отправление `SIGTERM` до отправления `SIGKILL`.

Запуск контейнера

1. Брокер должен быть уведомлён о готовности контейнера не ранее того момента, когда контейнер фактически становится готовым принять задачи для обработки. Этот момент наступает с задержкой после старта контейнера.
2. Контейнер должен корректно идентифицировать себя в системе, чтобы точно передать эту информацию брокеру. Проще всего это сделать на основе содержимого `/etc/hosts`.

Остановка

Обработка сигнала завершения работы

Java:

```
Runtime.getRuntime()
.addShutdownHook(new Thread(()->{...});
```

Python с AIOHttp:

```
app.on_shutdown.append(on_shutdown)
```

Server Message Block

Server Message Block (SMB, альтернативное название CIFS) — протокол разделяемого доступа к файлам и принтерам в Microsoft Windows.

1. Работает поверх протокола NetBIOS.
2. Существует в двух версиях: SMB1 и SMB2.

SAMBA

Samba — реализация SMB для UNIX-подобных систем.

1. Поддерживает функции контроллера доменов и Active Directory.
2. Поддерживает права доступа в стиле POSIX.
3. Иногда хвастается более высокой скоростью работы чем у Windows.

NFS

Network File System

NFS — протокол сетевого доступа от Sun Microsystems.

1. Работает поверх протокола ONC RPC.
2. Встречается в версиях 3 и 4.
3. Достаточно устойчиво обрабатывает временные потери соединений.
4. Подразумевается единое пространство UID для всех клиентов хранилища.

Атомарные операции

1. Поддержка файловых блокировок существенно различается между разными реализациями NFS. В случае Samba информация достаточно противоречива.
2. При этом доступны некоторые атомарные операции (например, `renameat2()` умеет атомарно менять файлы местами).

Исследование Ленардом Поттерингом поддержки файловых блокировок в Linux, включая использование NFS:
http://0pointer.net/blog/projects/locking.html

Inotify

Позволяет с большим числом ограничений ловить события, отражающие изменения в файловой системе.

1. Не работает на сетевых хранилищах.
2. Использование сильно усложнено отсутствием возможности понять причину произошедшего события.
3. Не работает на таких файловых системах как `/proc` и `/sys`.

Брокер

Его функции и назначение

В микросервисной архитектуре брокер используется для:

1. Гарантированной доставки сообщений между микросервисами.
2. Балансирования нагрузки с толерантностью отказов собственных реплик.
3. Унификации представления структур для работы на разных языках и платформах.

Примеры

Наиболее известные реализации

- *Apache Kafka:* написанный на Java и Scala наиболее распространённый брокер сообщений проекта Apache.
- *Rabbit MQ:* написанный на Erlang брокер с расширенными функциями маршрутизации сообщений.
- *centrifugo:* написанный на Go брокер сообщений с российскими корнями.

Apache Kafka

Модель передачи информации

Топик-подписчик

Множество подписчиков подписывается на топики, из которых получают сообщения в формате ключ-значение, публикуемые источниками.

Группы

Каждый подписчик должен отнести себя к некоторой группе. Каждое сообщение получает только один подписчик из группы. Допускается выделение каждого подписчика в свою группу.

Apache Zookeeper

Пример для Docker Compose

```
zookeeper:
  image: confluentinc/cp-zookeeper:latest
  environment:
    ZOOKEEPER_CLIENT_PORT: 2181
    ZOOKEEPER_TICK_TIME: 2000
  ports:
    - 22181:2181
```

Apache Kafka

Пример для Docker Compose

```
kafka:
  image: confluentinc/cp-kafka:latest
  depends_on:
    - zookeeper
  ports:
    - 29092:29092
  environment:
    KAFKA_BROKER_ID: 1
    KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
    KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:9092,PLAINTEXT_HOST://localhost:29092
    KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
    KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
    KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
    KAFKA_AUTO_CREATE_TOPICS_ENABLE: 'true'
```

Node.js

Пример подключения к Apache Kafka

```
const delay = ms => new Promise(resolve => setTimeout(resolve, ms));
await delay(5000);
this.kafka = new Kafka({
  logLevel: logLevel.ERROR,
  brokers: ["kafka:9092"],
  clientId: "disp",
  })
this.producer = this.kafka.producer();
await this.producer.connect();
```

Спасибо за внимание!

Всё о курсе: https://marigostra.ru/materials/unix.html
E-mail: msp@luwrain.org
Канал в Телеграм: https://t.me/MarigostraRu