GS LABS
GS GROUP

Публичный - L0

DRE Advanced Media Platform CACHE SERVER

Руководство администратора

Индекс	2060-CACHESERVER-AG
Секретность	Публичный - L0
Ревизия	1.0
Статус	Согласован
Подразделение	Департамент по разработке сервисов
Компания	GS Labs

Содержание

1. Аннотация

Документ предназначен для технических специалистов, занимающихся администрированием сервиса и обладающих навыками работы с компьютером на профессиональном уровне.

Данный документ опубликован для ознакомления с компонентами системы. Все настройки и параметры окружения предоставляются по запросу заказчика.

2. Термины и сокращения

Термин, сокращение	Определение, расшифровка
DREAMPlatform	(DRE Advanced Media Platform) Комплексное решение для телесмотрения, состоящее из взаимосвязанных компонентов, обеспечивающих генерацию, хранение и доставку контента (OTT и VOD) до телезрителя.
DRM	(DRM DREPLUS) Система управления цифровыми правами.
MDS	(DRE Advanced Media Platform META DATA SERVER) - сервер метаданных.
CACHE SERVER /CS	(DRE Advanced Media Platform CACHE SERVER) - единая точка входа для устройств абонента при обращении к API сторонних сервисов, таких как MDS, сервер авторизации, DRM и т.д.
ProDG	(Profile Data Guide) Сервис, осуществляющий ведение профилей абонентов, хранение данных, привязанных к профилям (избранное, история просмотров и т.д.). Также предназначается для выделения профилей внутри домохозяйства.
БД	Базы данных.
Устройство	Приемник, телевизор со SmartTV, смартфон или планшет с установленным приложением, через которые осуществляется просмотр контента.

3. Общее описание

DRE Advanced Media Platform CACHE SERVER (далее - CACHE SERVER) предназначен для того, чтобы служить единой точкой входа для приёмников при обращении к API сторонних сервисов, таких как сервер метаданных DRE Advanced Media Platform META DATA SERVER (далее - MDS), сервер авторизации, DRM и т.д.

Приёмники интегрируются с API только сервера CACHE SERVER, который в свою очередь интегрируется со сторонними сервисами. Это позволяет сократить время для получения и отображения контента на стороне клиента, а также уменьшить нагрузку на остальные сервисы DREAMPlatform (у клиентов необходимость держать только одну сессию, а не сессии со всеми компонентами платформы).

4. Компоненты CACHE SERVER

Компонент	Описание
cache-server	Сервис для обработки запросов метаданных.
redis-sentinel-cache	Хранилище redis для кэширования запросов метаданных.
redis-sentinel-queue	Хранилище redis для данных, используемых для контроля интенсивности потока запросов, отправляемых на бэкенд-сервера.
static-proxy	Сервис для кэширования статических файлов изображений (Nginx).
keepalived	Сервис для реализации "плавающего" между узлами кластера виртуального IP адреса.
ingress-nginx	Балансировщик нагрузки NGINX Ingress Controller.
default-http-backend	Сервис обработки запросов, для которых в правилах балансировщика ingress-nginx явно не определен сервис обработки.
fas	Внешний сервис ФАС.

5. Администрирование CACHE SERVER

5.1. Общие сведения

Установка и обновление сервера производится посредством Helm - менеджера пакетов (чартов) Kubernetes. При установке на основе чарта на кластере создаются его экземпляры - релизы (releases).

Список запущенных на кластере релизов, их состояние и версии соответствующих чартов можно получить, выполнив на мастер-ноде команду:

```
helm ls
```

Удаление компонентов сервера CACHE SERVER производится путем удаления соответствующих релизов.

Администрирование установленного сервера CACHE SERVER производится через командный интерфейс Kubectl на любой ноде кластера. Подробную информацию по работе с Kubectl можно найти в документации продукта Kubernetes.

5.2. Очистка кэша

Чтобы очистить кэш статических файлов, удалите поды сервиса static-proxy (новые поды создаются автоматически).

```
kubectl delete pods -l app=static-proxy -n NAMESPACE
```

Очистка кэша запросов в Redis производится через запуск скрипта очистки clean_cache.py из любого пода сервиса cache-server:

```
kubectl exec -ti PODNAME -n NAMESPACE python clean_cache.py
```

В случае необходимости сброса счетчика исходящих запросов, находящихся в обработке нижележащими серверами, запустите скрипт очистки clean_queue.py из любого пода сервиса cache-server:

```
kubectl exec -ti PODNAME -n NAMESPACE python clean_queue.py
```

6. Настройки CACHE SERVER

6.1. Описание работы

Все создаваемые контейнеры сервиса cache-server используют общее хранилище данных - Redis Sentinel. Каждый создаваемый контейнер сервиса static-proxy работает с собственным внутренним кэшем статических файлов.

Настройки кэширования задаются в helm-файле для сервисов cache-server и static-proxy.

6.2. Cache-server

Основные заголовки ответов на запросы метаданных от сервиса cache-server:

- Date: Fri, 06 Apr 2018 11:46:49 GMT - дата и время генерации ответа на запрос.
- Cache-Control: max-age=600 - период актуальности данных в секундах от момента текущего запроса. При анализе необходимо учитывать содержимое заголовка X-Caching.
- Expires: Fri, 06 Apr 2018 21:46:10 GMT - время истечения срока актуальности данных.
- X-Caching: true, если ответ на запрос получен из кэша, false, если сделан запрос к нижележащему серверу данных.
- X-Correlation-Id: 7c37012e-c342-47ec-9c37-e42dad70f83c - идентификатор запроса.
- ETag: "0bbb6e208526e295e677564c82e062f0" - тег (идентификатор) версии запрошенного ресурса.
- Last-Modified: Fri, 06 Apr 2018 11:46:10 GMT - дата и время последней модификации данных на кэш-сервере.
- Server: CacheServer v3.3.0 - имя и версия сервера.

Заголовок X-Correlation-Id присутствует во всех ответах на запросы от сервиса cache-server, кроме некоторых случаев ошибок обработки запроса. При этом если заголовок передан клиентом в исходном запросе, то он передается на нижележащий сервер и также будет указан в ответе. Если в исходном запросе заголовок отсутствует, то он генерируется по правилу UUID4 и отправляется на нижележащий сервер.

При прерывании запроса клиентом или истечении таймаута балансировщика запрос от сервиса cache-server к нижележащему серверу не будет прерван и в случае его корректного выполнения ответ будет сохранен в кэш.

Сервис cache-server работает в неблокирующем асинхронном режиме. Длительное ожидание получения ответа от нижележащих серверов не препятствует получению данных из кэша.

Задать правила обработки и кэширования запросов можно также через настройки конфигурации proxy-conf.yml схемы URL. Параметры настроек приведены в таблице ниже.

Изменение конфигурации proxy-conf.yml, заданной по умолчанию в чарте CacheServer, производится добавлением в helmfile CacheServer параметра ProxyConf, в котором приводится полная конфигурация, либо редактированием Config Map (после редактирования Config Map необходимо пересоздать поды сервиса cache-server).

6.3. Static-proxy

Основные заголовки ответов на запросы файлов от сервиса static-proxy:

- Date: Fri, 06 Apr 2018 12:57:49 GMT - время генерации ответа на запрос.
- ETag: "5ab90ed6-9f" - идентификатор файла.
- Last-Modified: Mon, 26 Mar 2018 15:16:38 GMT - время последней модификации данных на бэкенд сервере.
- Server: nginx/1.11.13 - имя и версия сервера.
- X-Cache-Status: HIT - состояние данных запроса в кэше (HIT, MISS, EXPIRED).

6.4. Балансировщик нагрузки

Настройка балансировщика нагрузки производится путем редактирования helm-файла. Правила направления запросов клиентов на бэкенд-сервисы задаются в ingress.yaml. Подробнее см. Руководство по установке. Просмотр текущих правил и их редактирование (без сохранения в репозиторий) доступны по командам:

```
kubectl describe ing INGRESSNAME -n NAMESPACE kubectl edit ing INGRESSNAME -n NAMESPACE
```

Описание доступных настроек балансировщика нагрузки доступно в документации продукта NGINX Ingress Controller.

Далее приведены некоторые доступные параметры, задаваемые в helmfile:

6.5. Keepalived

С помощью сервиса keepalived реализуются "плавающие" виртуальные IP-адреса между нодами кластера, по которым происходит обращение к CACHE SERVER через dns.

Общая схема работы следующая:

- На каждой ноде кластера разворачивается по одному контейнеру балансировщика нагрузки и сервиса keepalived. Как только контейнеры разворачиваются, keepalived присваивает одной из нод указанный виртуальный IP-адрес.
- Эта нода становится master нодой, остальные - backup нодами.
- Каждый keepalived-контейнер периодически отправляет запросы к балансировщику, который расположен с ним на хосте, и проверяет, что ответ имеет определенный код. Если в результате двух последовательных проверок от балансировщика на master ноде не получены ответы по истечении назначенного таймаута или получены ответы с кодом, отличным от установленного, keepalived переназначает виртуальный IP-адрес на одну из backup нод с работающим балансировщиком. Соответствующая нода переходит в статус master.

Текущая конфигурация keepalived доступна через соответствующий объект configmap.

Виртуальные адреса задаются на этапе установки и могут быть изменены согласно инструкции в Руководстве по установке.

6.6. Хранилище данных Redis

Для хранения данных CACHE SERVER использует два хранилища Redis Sentinel.

Одно из хранилищ (Redis Sentinel Cache) используется для хранения запросов метаданных. Хранение организовано в ключах вида cache_server:url_<hash_url>.

Конфигурационные настройки Redis Sentinel доступны через соответствующий Config Map. При необходимости изменения конфигурации отредактируйте настройки в configmap и выполните команду удаления имеющихся подов Redis, запустив таким образом процесс создания новых подов.

Второе хранилище Redis Sentinel Queue используется для хранения данных об активных исходящих запросах кэш-сервера, находящихся в обработке бэкенд-сервисами. Ключи вида cache_server: in_progress_<server>_<hash_url> указывают на то, что запрос с данным url уже находится в обработке бэкенд-сервисом <server>, ключ cache_server:total_in_progress_<server> содержит счетчик общего количества активных запросов от CACHE SERVER на отдельном бэкенд-сервисе.

7. Логирование и структура данных

Сбор логов и отправка на серверы-агрегаторы производится сервисом filebit. Настройка и конфигурирование сервиса происходит при помощи helmfile и в данном руководстве не рассматривается. Параметры и конфигурация сбора логов указываются отделом автоматизации GS Labs.

7.1. Cache-server

В таблице ниже перечисляются основные поля в логах cache-server:

Название	Описание
application	Название программы, оставившей запись в логе (CACHE SERVER).
application_version	Версия кэш-сервера.
bytes	Длина тела ответа в байтах.
caching	Данные получены из кеша (true, false).
cid	Correlation-Id, присвоенный запросу.
compressed	Клиенту отправлен сжатый ответ (true, false).
error_text	Текст сообщения об ошибке.
hash_url	Hash для url запроса.
headers.Accept-Encoding headers.X-hwid headers.X-Correlation-Id headers.X-Serial-Number headers.X-domain-code headers.X-aptp-code	Соответствующие заголовки в клиентском запросе.
http_status_code	HTTP статус код ответа.
http_version	Версия протокола HTTP.
method	Метод HTTP запроса.
path	URI запроса без схемы и доменного имени (источника host).
query_string	Строка параметров запроса.
scheme	Схема запроса (http/https).

time_backend_response	Длительность обработки запроса нижележащим сервером или хранилищем redis в миллисекундах.
time_duration	Длительность обработки запроса после его передачи от балансировщика сервису cache-server в миллисекундах. Включает в себя время time_backend_response.
user_agent	Содержимое заголовка User-Agent от клиента.

7.2. Static-proxy

В таблице ниже перечисляются основные поля в логах static-proxy:

Название	Описание
application	Название программы, оставившей запись в логе (StaticProxy).
application_version	Версия static-proxy.
bytes_sent	Размер тела ответа в байтах.
cache_status	Состояние данных запроса в кэше (HIT, MISS, EXPIRED).
headers.X-hwid headers.X-Correlation-Id headers.X-Serial-Number headers.X-domain-code headers.X-aptp-code	Соответствующие заголовки в клиентском запросе.
http_status_code	HTTP статус код ответа.
log_level	Уровень логирования.
method	Метод HTTP запроса.
path	URI запроса без схемы и доменного имени (источника host).
request_proto	Протокол запроса.
time_request	Длительность в секундах обработки запроса после его передачи от балансировщика сервису static-proxy.
user_agent	Содержимое заголовка User-Agent от клиента.

7.3. Балансировщик нагрузки

Описание структуры данных логирования балансировщика можно найти в документации продукта Nginx. В таблице ниже перечисляются поля в логах балансировщика nginx-ingress-controller, включенные по умолчанию:

Название	Описание
bytes_sent	Количество байтов, отправленных клиенту (включая заголовки).
connection_requests	Количество запросов, полученных через текущее соединение.
http_status_code	HTTP статус код ответа.
host	Источник запроса (доменный адрес сервера).
method	Метод HTTP запроса.
origin	Название компонента, оставившего запись в логе (nginx).
path	URI запроса без схемы и доменного адреса (источника host).
query_string	Строка параметров запроса.
request_proto	Протокол запроса.
scheme	Схема запроса (http/https).
time_backend_connect	Длительность установки соединения в секундах с нижележащим сервером, обработавшим запрос.
time_backend_response	Длительность обработки запроса нижележащим сервером в секундах (без учета времени на установку соединения).
time_request	Длительность обработки запроса в секундах с момента получения первого байта запроса от клиента до отправки последнего байта клиенту.
upstream_addr	Адрес и порт нижележащего сервера, обработавшего запрос.
upstream_http_server	Содержимое заголовка Server, полученного от нижележащего сервера, обработавшего запрос.
user_agent	Содержимое заголовка User-Agent от клиента.

8. Метрики

Сервис реализован в виде API для целей мониторинга. Формат построения ответа - JSONAPI.

Для получения метрик необходимо наличие Prometheus в системе.

8.1. Количество запросов в обработке на бэкенд-серверах

8.1.1. Получение количества запросов, находящихся в обработке бэкенд-серверами

Метрика для получения количества исходящих запросов кэш-сервера, находящихся в обработке бэкенд-серверами.

Формат:

GET /metrics/queue_size/

Пример ответа:

```
HTTP 200 OK
Content-Type: application/json
[
  {"blackout":0},
  {"mds_manager":0},
  {"mds_storage":1},
  {"mds_search":0},
  {"fas":0}
]
```

8.2. Доступность сервисов

8.2.1. Получение статуса подключения к Redis Sentinel Queue

Метрика-индикатор доступности Redis Sentinel Queue - хранилища данных о количестве исходящих запросов, находящихся в обработке бэкенд-серверами (true, false).

Формат:

GET /metrics/service_backend/

Пример ответа:

```
HTTP 200 OK
Content-Type: application/json
{
  "service_backend": true
 }
```

8.2.2. Получение статуса подключения к Redis Sentinel Cache

Метрика-индикатор доступности Redis Sentinel Cache (true, false).

Формат:

GET /metrics/cache_backend/

Пример ответа:

```
HTTP 200 OK
Content-Type: application/json

{
        "cache_backend": true
}
```