OTUS

Онлайн образование

otus.ru

Проверить, идет ли запись

Меня хорошо видно && слышно?

Тема вебинара

Admin API

Непомнящий Евгений

Разработчик Java/Kotlin IT-Sense

@evgeniyN

Правила вебинара

Активно
участвуем

Off-topic обсуждаем
в TG

Задаем вопрос
в чат или **голосом**

Вопросы вижу в чате,
могу ответить не сразу

Условные обозначения

Индивидуально

Время, необходимое
на активность

Пишем в чат

Говорим голосом

Документ

Ответьте себе или
задайте вопрос

Маршрут вебинара

- Зачем это надо
- Устройство Admin API
- Управление топиками
- Сообщения и консамеры
- Авторизация
- Прочее

Цели вебинара

После занятия вы сможете

1. Рассмотреть Admin API
2. Изучить процесс управления кластером

Смысл

Зачем вам это уметь

1. Admin API нужно не слишком часто, но иногда может быть полезно

Зачем?

Можно делать как раньше

Управлять кафкой через шел-скрипты, как мы делали до сих пор

```
bin/kafka-topics.sh
  --create --topic kinaction_selfserviceTopic \
  --bootstrap-server localhost:9094 \
  --partitions 2 \
  --replication-factor 2
```

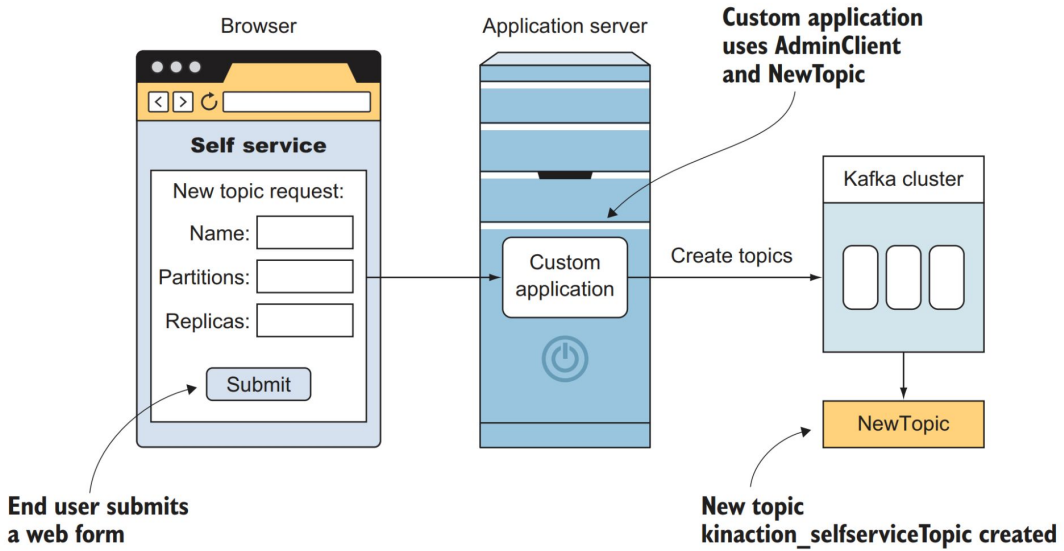
Uses the kafka-topic.sh script to create a new topic

Includes our custom integers for the number of partitions and replicas for our topic

Своя админка

Иметь свою собственную админку

- выполнять проверки
  при создании топика
  - требования к названию,
  автоматически выбирать
  фактор репликации и т.п.
- устанавливать настройки
  по шаблону - списки ACL, политики удаления и т.п.
- добавляем новый сервис - автоматически добавляется пользователь во все
  нужные ACL



Универсальный инструмент для Kafka

- Kafdrop
- Conduct
- Kafka explorer

https://github.com/obsidiandynamics/kafdrop/blob/master/src/main/java/kafdrop/service/KafkaHighLevelAdminClient.java

Kafka API

- Producer API - kafka-clients
- Consumer API - kafka-clients
- Streams API - kafka-streams
- **Admin API - kafka-clients**
- Connect API

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Устройство Admin API

Давайте приступим - создание топика

p1….Ex1CreateTopic

- параметров не так и много, самое главное - куда подключаться
- все методы асинхронные
- большинство методов работают в batch-стиле
- у нас будет много примеров - давайте уберем всю копипасту
- **p1….Ex2CreateTopic** - финальный пример. **Если по нему есть вопросы, давайте обсудим их сразу**

CreateTopicResult

- Многие результаты Admin устроены похожим образом
- Внутри набор KafkaFuture
- Доступ к ним через хелперные методы, которые возвращают kafkaFuture или их коллекции

KafkaFuture

- Это Future
- Плюс часть механик из CompletionStage, которые нужны для реализации CreateTopicResult и его аналогов
- Можно получить полноценный CompletionStage (`toCompletionStage()`), если вам нужна удобная асинхронная работа
- Если вы пишите обычный синхронный код, просто вызывайте get() и не мучайтесь
- Если вы пишите асинхронный код, можно использовать project reactor (у которого значительно больше возможностей):
  `Mono.fromCompletionStage(kafkaFuture.toCompletionStage())`

см **RemoveAll**

Две перегрузки у многих методов

```
default CreateTopicsResult createTopics(Collection<NewTopic> newTopics) {  /*1*/
   return createTopics(newTopics,  new CreateTopicsOptions());
}

CreateTopicsResult createTopics(Collection<NewTopic> newTopics, CreateTopicsOptions options); /*2*/

public class CreateTopicsOptions extends AbstractOptions<CreateTopicsOptions> { /*3*/
   private boolean validateOnly = false;
   private boolean retryOnQuotaViolation = true;


public abstract class AbstractOptions<T extends AbstractOptions> {   /*4*/
   protected Integer timeoutMs = null;
```

InterfaceStability

```
public class InterfaceStability {
  public @interface Stable { } // может меняться при изменении majo r версии

  public @interface Evolving { } // может меняться при изменении minor версии

  public @interface Unstable { } // вообще никаких гарантий
}
```

```
@InterfaceStability.Evolving
public interface Admin extends AutoCloseable {
```

```
@InterfaceStability.Evolving
public class CreateTopicsOptions extends AbstractOptions<CreateTopicsOptions> {
```

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Управление топиками

Создание топика

```
new NewTopic("ex3-topic-1", 1, (short) 1) /* настройка параметров топика */
        .configs(Map.of(
                TopicConfig.SEGMENT_MS_CONFIG, Integer.valueOf(1000 * 60 * 60).toString()
        )),

new NewTopic("ex3-topic-2", Optional.empty(), Optional.empty()), /* настройки по умолчанию */

new NewTopic("ex3-topic-3", Map.of(0, List.of(2, 3), 1, List.of(4, 5))) /* ручное распределение */
```

Описание топика и список топиков

```
DescribeTopicsResult describeTopics(Collection<String> topicNames)
DescribeTopicsResult describeTopics(Collection<String> topicNames, DescribeTopicsOptions options)
public class TopicDescription {
   private final String name;
   private final boolean internal;
   private final List<TopicPartitionInfo> partitions;
   private final Set<AclOperation> authorizedOperations;
   private final Uuid topicId;
public class TopicPartitionInfo {
   private final int partition;
   private final Node leader;
   private final List<Node> replicas;
   private final List<Node> isr;

ListTopicsResult listTopics(ListTopicsOptions options)
public class TopicListing {
   private final String name;
   private final Uuid topicId;
   private final boolean internal;
```

см **p2.topic.Ex4DescribeTopic**

Удаление топика

```
DeleteTopicsResult deleteTopics(Collection<String> topics)

DeleteTopicsResult deleteTopics(TopicCollection topics)
```

Удаление топика выполняется не сразу, а только через некоторое время

см **p2.topic.Ex5DeleteTopic**

Добавление партиций

```
CreatePartitionsResult createPartitions(Map<String, NewPartitions> newPartitions)

static NewPartitions increaseTo(int totalCount)

// для каждой новой партиции указываем раскладку по брокерам
// размер внешнего списка -  totalCount - prevCount
// размеры внутренних списков - replicationFactor
static NewPartitions increaseTo(int totalCount, List<List<Integer>> newAssignments)
```

см **p2.topic.Ex6CreatePartitions**

Перемещение партиций

```
AlterPartitionReassignmentsResult  alterPartitionReassignments (
   Map<TopicPartition, Optional<NewPartitionReassignment >> reassignments)
public class  NewPartitionReassignment  {
   private final  List<Integer> targetReplicas;

ListPartitionReassignmentsResult  listPartitionReassignments ()
ListPartitionReassignmentsResult  listPartitionReassignments (Set<TopicPartition> partitions)
public class  PartitionReassignment  {
   private final  List<Integer> replicas;
   private final  List<Integer> addingReplicas;
   private final  List<Integer> removingReplicas;
```

см **p2.topic.Ex7MovePartitions**

Хранение в файловой системе

```
DescribeLogDirsResult describeLogDirs(Collection<Integer> brokers)
public class LogDirDescription {
   private final Map<TopicPartition, ReplicaInfo> replicaInfos;
   private final ApiException error;
   private final OptionalLong totalBytes;
   private final OptionalLong usableBytes;

DescribeReplicaLogDirsResult describeReplicaLogDirs(Collection<TopicPartitionReplica> replicas)
static public class ReplicaLogDirInfo {
   private final String currentReplicaLogDir;
   private final long currentReplicaOffsetLag;
   private final String futureReplicaLogDir;
   private final long futureReplicaOffsetLag;

AlterReplicaLogDirsResult alterReplicaLogDirs(Map<TopicPartitionReplica, String>
replicaAssignment)
```

см **p2.topic.Ex8LogDirs**

Управление транзакциями

```
DescribeTransactionsResult  describeTransactions (Collection<String> transactionalIds)

AbortTransactionResult  abortTransaction (AbortTransactionSpec  spec)

ListTransactionsResult  listTransactions ()
```

На следующем занятии

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Сообщения и консамеры

Удаление записей

```
DeleteRecordsResult deleteRecords(Map<TopicPartition, RecordsToDelete> recordsToDelete)

public class RecordsToDelete {
   private long offset;
```

см **p3.records.Ex9DeleteRecords**

Consumer groups

```
ListConsumerGroupsResult listConsumerGroups()
DescribeConsumerGroupsResult describeConsumerGroups(Collection<String> groupIds)
DeleteConsumerGroupsResult deleteConsumerGroups(Collection<String> groupIds)

ListConsumerGroupOffsetsResult listConsumerGroupOffsets(String groupId)
ListConsumerGroupOffsetsResult listConsumerGroupOffsets(Map<String,
ListConsumerGroupOffsetsSpec> groupSpecs)

DeleteConsumerGroupOffsetsResult deleteConsumerGroupOffsets(String groupId, Set<TopicPartition>
partitions)
AlterConsumerGroupOffsetsResult alterConsumerGroupOffsets(String groupId, Map<TopicPartition,
OffsetAndMetadata> offsets)
```

см **p3.records.Ex10ConsumerGroups**

Авторизация

ACL

```
Principal {P} is [Allowed|Denied] Operation {O} From Host {H} on any
Resource {R} matching ResourcePattern {RP}
```

- Принципал: `<principalType>:<principalName>, User:*` - все пользователи
- Операция: `Describe|Create|Delete|Alter|Read|Write|DescribeConfigs|AlterConfigs`
- Хост: `ip-адрес, * - все`
- Ресурс: `Cluster|Topic|Group|TransactionalId|DelegationToken`
- Шаблон: `Literal|Prefixed`

Управление ACL

```
CreateAclsResult createAcls(Collection<AclBinding> acls)

public class AclBinding {
   private final ResourcePattern pattern;
   private final AccessControlEntry entry;

ResourcePattern(ResourceType resourceType, String name, PatternType patternType)
AccessControlEntry(String principal, String host, AclOperation operation, AclPermissionType
permissionType)
```

см **p4.auth.Ex12ACL**

Управление ACL

```
DeleteAclsResult deleteAcls(Collection<AclBindingFilter> filters)
DescribeAclsResult describeAcls(AclBindingFilter filter)


class AclBindingFilter {
   private final ResourcePatternFilter patternFilter;
   private final AccessControlEntryFilter entryFilter;

ResourcePatternFilter(ResourceType resourceType, String name, PatternType patternType)
AccessControlEntryFilter(String principal, String host, AclOperation operation,
AclPermissionType permissionType)
```

см **p4.auth.Ex12ACL**

Управление квотами

```
AlterClientQuotasResult alterClientQuotas(Collection<ClientQuotaAlteration> entries)

ClientQuotaAlteration(ClientQuotaEntity entity, Collection<Op> ops)
ClientQuotaEntity(Map<String, String> entries) // тип по имени
Op(String key, Double value) // если value null, то удаляет квоту
```

```
DescribeClientQuotasResult describeClientQuotas(ClientQuotaFilter filter)
ClientQuotaFilter(Collection<ClientQuotaFilterComponent> components, boolean strict)
ClientQuotaFilterComponent(String entityType, Optional<String> match)
```

Прочее

Настройка топиков и брокеров

```
AlterConfigsResult incrementalAlterConfigs(Map<ConfigResource, Collection<AlterConfigOp>>
configs)

ConfigResource(Type type, String name)
ConfigEntry(String name, String value)
```

```
DescribeConfigsResult describeConfigs(Collection<ConfigResource> resources)
```

см **p5.misc.Ex13Configs**

Описание кластера

```
DescribeClusterResult describeCluster()
```

```
DescribeClusterResult {
    private final KafkaFuture<Collection<Node>> nodes;
    private final KafkaFuture<Node> controller;
    private final KafkaFuture<String> clusterId;
    private final KafkaFuture<Set<AclOperation>> authorizedOperations;
```

см **p5.misc.Ex14DescribeCluster**

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Рефлексия

Ключевые тезисы

1. KafkaFuture, две версии метода, batch-операции
2. АПИ не устоялось, много экспериментального
3. Позволяет выполнять операции по управлению кластером, топиками и т.п.

Заполните, пожалуйста, опрос о занятии по ссылке в чате

Спасибо за внимание!

Приходите на следующие вебинары

Непомнящий Евгений

Разработчик Java/ Kotlin IT-Sense

@evgeniyN