Задания по администрированию кластера Kafka

Как мы уже обсуждали на предыдущих уроках – нет ничего идеального. Любая система, даже самая надежная, рано или поздно даст сбой. И мы при всем желании не сможем защититься от всех возможных отказов на 100%.

Что же с этим делать? Внимательные ученики курса уже знают ответ – иметь disaster recovery plan и время от времени тренироваться на практике. Ребята из Netflix думали также и поэтому создали инструменты для контролируемого создания хаоса в вашей системе. Наверное, самый известный из них – **Chaos Monkey**, но в разрушительном зоопарке у Netflix есть еще и другие виды обезьян и даже Chaos Kong.

Вот что они сами инженеры Netflix пишут в своем блоге про идею, подтолкнувшую их на создание таких на первый взгляд вредных инструментов:

> "Представьте, что у вас спустило колесо. Даже если у вас в багажнике есть запаска, знаете ли вы, накачано ли она? Есть ли у вас инструменты, чтобы её поставить? И, что важнее всего, вы помните, как это правильно делать? Способ убедиться, что вы можете справиться со спущенной шиной на автостраде, под дождем, посреди ночи, - это проделать дыру в шине один раз в неделю рядом с домом в воскресенье и выполнить упражнение по замене. Это дорого и требует много времени в реальности, но может быть (почти) бесплатным и автоматизированным в облаке.
> Так мы размышляли, создавая Chaos Monkey - инструмент, который случайным образом отключает наши инстансы в продакшне, чтобы убедиться, что мы сможем пережить эту нередкую проблему без последствий для клиентов. Название происходит от идеи выпустить дикую обезьяну с оружием в вашем центре обработки данных (или регионе облака), чтобы она случайным образом уничтожала инстансы и грызла кабели – а всё это время мы продолжаем непрерывно обслуживать наших клиентов. Запустив Chaos Monkey в середине рабочего дня в тщательно контролируемой среде с инженерами, готовыми решить любые проблемы, мы все равно можем извлечь уроки о слабых местах нашей системы и создать механизмы автоматического

> восстановления для их устранения. Так что в следующий раз, когда произойдет сбой инстанса в 3 часа ночи в воскресенье, мы этого даже не заметим."
> https://netflixtechblog.com/the-netflix-simian-army-16e57fbab116

В практическом задании мы предлагаем вам вырастить несколько собственных обезьянок, выпустить их на свободу и спасти свой кластер Apache Kafka от погружения в хаос.

Задание 1

Для начала нам нужен кластер Apache Kafka. Именно в него мы будем селить наших обезьянок. Если вам потребуется помощь – пересмотрите третий урок.

- Установить и запустить кластер из трех брокеров Apache Kafka и трех ZooKeeper.

Задание 2

Предустановка: JMX Exporter + Prometheus + Grafana с настроенным дашбордом можно поставить заранее.

Обезьянки любят, когда их работа у всех на виду. На тестовых стендах уже установлены Prometheus и Grafana и настроен дашборд для кластера Apache Kafka. Но у нас нет никакого представления о том, что происходит с клиентами.

- Добавить JMX Exporter агентов для брокеров Apache Kafka и серверов ZooKeeper (не забыть про рестарт);
- Проверить метрики кластера Apache Kafka на уже готовом дашборде – `http://grafana.<ваш номер студента>.edu.slurm.io/dashboards`.

Задание 3

Уже готовы запускать обезьянок? Осталось чуть-чуть. Ронять кластер, которым никто не пользуется, как-то неинтересно. Давайте запустим несколько клиентов, которым и будем портить жизнь. Если вы уже выполнили предыдущий шаг этого

практического задания, то можете использовать клиентов из него или из урока “Клиентские библиотеки”. Если готовых клиентов у вас под рукой нет, можно воспользоваться встроенными асинхронными клиентами Kafka.

- Установить Burrow и настроить сбор метрики consumer lag через Prometheus;
- Добавить Prometheus JMX Exporter конфиг файл с запросами для сбора метрик producer/consumer.
- Создать тестовый топик с 3 партициями, фактором репликации 3, минимальным числом синхронных реплик 2 и запретом на “грязные” выборы лидера;
- Запустить ProducerPerformance клиента;
- Запустить ConsumerPerformance клиента;
- Создать новый дашборд в Grafana с метриками consumer lag и клиентскими метриками из Prometheus.

Задание 4

Окей, кластер готов, клиенты работают, мониторинг на месте – в общем все слишком хорошо. Давайте запустим пару злобных обезьян. Нашей основной задачей будет настроить клиентов и кластер так, чтобы минимизировать время неработоспособности при авариях.

Мы не будем пользоваться инструментами Netflix, так как они тянут за собой ряд зависимостей, которые усложнят наш проект. Вместо этого мы призовем на помощь дракона! Встречайте – Trogdor. Кстати, именно этим инструментом сами разработчики Apache Kafka проводят различные стресс-тесты.

- Запустить Trogdor;
- Имитировать неисправность на стороне брокера при помощи `ProcessStopFault`;
- Имитировать разрыв сети при помощи `NetworkPartitionFault`;
- Повторить имитации, изменяя продолжительность действия проблемы;
- Увеличить latency между брокерами;
- Увеличить latency между продюсером и кластером;

- Выключить один брокер руками “навсегда”. Вернуть ноду в качестве "новой". Восстановить балансировку партиций нашего топика.

Задание 5

Давайте завершим наше тестирование кластера на отказоустойчивость и немного порефлексируем.

- Как мы поняли, что с кластером все в порядке?
- Как мы поймали момент, когда начались проблемы? Какие метрики помогли нам в этом? На какие метрики мы бы хотели повесить алерты? Было ли в это время что-то интересное в логах брокеров? Был ли достаточным уровень логирования на брокерах?
- Все ли партиции были в синхронном состоянии? Как повели себя при этом асинхронные клиенты? А как бы повели синхронные с `acks=all`?
- Что еще могло пойти не так? Как бы мы это заметили?
- Была ли просадка в скорости работы наших клиентов?
- Появились ли дубли? Потеряли ли мы данные? Как можно это понять?
- Можно ли было автоматизировать восстановление после каких-то неисправностей? Есть ли уже готовые инструменты для этого?
- Подумать, что может пойти не так с вашим боевым кластером Kafka и какие требования вы к нему предъявляете. Составить disaster recovery plan. В плане описать “флаги”, сигнализирующие о наличии проблем и шаги по их устранению.