Привет!

1 декабря, в 23:59 по московскому времени мы запускаем наш новый сервис - API хранилища истории сессий нашего онлайн кинотеатра «Фильмопоиск». Дату запуска сдвинуть нельзя, наш PR уже активно продвигает этот запуск. От тебя потребуется развернуть продуктовую инсталляцию этого сервиса.

Наш подрядчик "Horns&Hooves Soft inc" пишет для нас этот новый сервис. Неделю назад подрядчик провёл демонстрационную презентацию. На ней он показал *почти* корректно работающее приложение, и презентовал HTTP эндпоинт, который отвечает на **GET /ping** кодом **200**, если приложение работает корректно и кодом **500**, если нет.

Мы попросили внести небольшие изменения: нужно, чтобы запрос **GET /long\_dummy** в 75% случаев работал быстрее секунды, при этом нас устроит закешированный ответ не старше минуты. На презентации он работал дольше. Кроме того, подрядчик сообщил, что потребуется внести некоторые технологические изменения для повышения удобства эксплуатации, а так же починить несколько некритичных багов для повышения стабильности в работе.

Вчера должна была состояться приёмка, но подрядчик на связь не вышел и перестал отвечать на письма, сообщения и звонки. Нам удалось выяснить, что у подрядчика возникли серьёзные форс-мажорные обстоятельства. Скорее всего получится возобновить взаимодействие не раньше 2 декабря, то есть уже после согласованной даты запуска. Подрядчик не успел предоставить документацию к приложению, и не смог развернуть у нас своё приложение в срок, как ранее обещал. Тот стенд, на котором проводилась демонстрация, уже успели разобрать. К счастью, у нашего менеджера остался email с бинарником приложения, который использовали на демо.

<https://storage.yandexcloud.net/final-homework/bingo> – вот ссылка на этот бинарник.

Твоя задача развернуть отказоустойчивую инсталляцию приложения из имеющегося бинарника до даты запуска продукта. Планируется *стабильная* нагрузка в **60 RPS**, *пиковая* в **120 RPS**.

В эту пятницу, 24 ноября, выходит из отпуска наш тестировщик Петя, который работал с подрядчиком и умеет тестировать это приложение. Он сможет проверить твою и инсталляцию и подсказать, что с ней не так, чтобы тебе было удобнее готовиться к финальному запуску.

Петя интроверт, не любит живое общение, поэтому он обещал сделать автоматику и помогать тебе с помощью специального сервиса - <https://devops.yactf.ru>

Посредством этого сервиса он и будет принимать решение о том, насколько тебе удалось справиться с требованиями технического задания.

**Требования** (в порядке убывания важности):

- Отказоустойчивость: сервис должен быть развернут на **двух нодах**, отказ любой из них должен быть незаметен пользователю. Допускается просадка по RPS до *стабильного* значения в момент отказа любой из нод. При живости обеих нод, инсталяция обязана выдерживать *пиковую* нагрузку. Так же нужно обеспечить восстановление работоспособности любой отказавшей ноды быстрее, чем за минуту.

- Сервис должен переживать *пиковую* нагрузку в **120 RPS** в течение 1 минуты, *стабильную* в **60 RPS**.

- Запросы **POST /operation {"operation": <operation\_id: integer>}** должны возвращать незакешированный ответ. Сервер должен обрабатывать такие запросы и отдавать результат быстрее, чем за **400 миллисекунд** в **90%** случаев при **120 RPS**, гарантируя **не более 1%** ошибок.

- Запросы **GET /db\_dummy** должны возвращать незакешированный ответ. Сервер должен обрабатывать такие запросы и отдавать результат быстрее, чем за **400 миллисекунд** в **90%** случаев при **120 RPS**, гарантируя **не более 1%** ошибок.

- Запросы **GET /api/movie/{id}** должны возвращать незакешированный ответ. Сервер должен обрабатывать такие запросы и отдавать результат быстрее, чем за **400 миллисекунд** в **90%** случаев при **120 RPS**, гарантируя **не более 1%** ошибок.

- Запросы **GET /api/customer/{id}** должны возвращать незакешированный ответ. Сервер должен обрабатывать такие запросы и отдавать результат быстрее, чем за **400 миллисекунд** в **90%** случаев при **120 RPS**, гарантируя **не более 1%** ошибок.

- Запросы **GET /api/session/{id}** должны возвращать незакешированный ответ. Сервер должен обрабатывать такие запросы и отдавать результат быстрее, чем за **400 миллисекунд** в **90%** случаев при **120 RPS**, гарантируя **не более 1%** ошибок.

- Запросы **GET /api/movie** должны возвращать незакешированный ответ. Сервер должен обрабатывать такие запросы и отдавать результат гарантируя **не более 1%** ошибок. Требований по времени ответа нет, планируем делать не более одного такого запроса одновременно.

- Запросы **GET /api/customer** должны возвращать незакешированный ответ. Сервер должен обрабатывать такие запросы и отдавать результат гарантируя **не более** **1%** ошибок. Требований по времени ответа нет, планируем делать не более одного такого запроса одновременно.

- Запросы **GET /api/session** должны возвращать незакешированный ответ. Сервер должен обрабатывать такие запросы и отдавать результат гарантируя **не более** **5%** ошибок. Требований по времени ответа нет, планируем делать не более одного такого запроса одновременно.

- Запросы **POST /api/session** должны возвращать незакешированный ответ. Сервер должен обрабатывать такие запросы и отдавать результат гарантируя **не более 1%** ошибок. Требований по времени ответа и RPS нет.

- Запросы **DELETE /api/session/{id}** должны возвращать незакешированный ответ. Сервер должен обрабатывать такие запросы и отдавать результат гарантируя **не более 1% ошибок**. Требований по времени ответа и RPS нет.

- **Задача со звёздочкой**: сделать так, чтобы сервис работал на отдельном домене по https протоколу, и по http без редиректа на https (допускается самоподписанный сертификат).

- **Задача со звёздочкой**: сделать http3.

- **Задача со звёздочкой**: сделать так, чтобы запросы **GET /long\_dummy** возвращали ответ не старше 1 минуты и отвечали быстрее, чем за 1 секунду в 75% случаев.

- **Задача со звёздочкой**: желательно обеспечить наблюдаемость приложения: графики RPS и ошибок по каждому эндпоинту.

- **Задача со звёздочкой**: автоматизировать развёртывание при помощи devops инструментов, с которыми вы успели познакомиться ранее.

Немножко комментариев и рекомендаций:

- Для нас важно получить работающую инсталляцию, и менее важно как эта работоспособность была обеспечена. Мы засчитываем только рабочие решения. В качестве решения помимо эндпоинта мы хотим увидеть, как эта инсталляция была развернута. В минимальном варианте это должно быть сочинение на тему "как я это развернул руками". В идеале же, это должен быть репозиторий, который при помощи различных devops инструментов способен развернуть всю инсталляцию по одной команде. Мы не ожидаем от всех участников идеального варианта, но автоматизированные решения при прочих равных, мы оценим выше неавтоматизированных. При этом работающее неавтоматизированное решение будет засчитано, а неработающее автоматизированное - нет;

- Бинарник умеет в развернутой базе данных создавать необходимые для работы таблицы и наполнять их тестовыми данными. Нужно его только об этом правильно попросить, уверен вы сможете разобраться как. Наличие в базе этих тестовых данных - одно из необходимых условий успешного выполнения задания. Вы можете в рамках тестов создавать некоторые сущности в базе и их же удалять. Однако не стоит удалять ничего из изначально там заведённого - это может не понравиться автоматике, которая будет считать Ваш рейтинг;

- Важно убедиться в том, что на сервере настроено корректно точное время. Таймзона не имеет значения. В противном случае некоторые наши проверки решат, что вы считерили, добавив кеширование там, где его быть не должно;

- В конфигурации есть параметр **student\_email** - тут должен быть ваш email, который вы использовали при регистрации на тренировки - это важно для проверки задания;

- Хэлсчек в приложении есть, живёт на **GET /ping** и свою функцию выполняет;

- Можешь на своё усмотрение упаковать бинарник в docker image, либо этого не делать;

- Для тех, кто разворачивается в Яндекс.Облаке - разворачивайтесь в одной АЗ;

- Для самостоятельной проверки RPS и времени ответа своей инсталяции можете пользовать wrk / ab;

- Для SSL, http3 и кеширования мы рекомендуем использовать reverse proxy, которые это поддерживают, например nginx;

- Некоторый софт не умеет отдавать метрики в формате prometheus, для такого софта сбоку разворачивают что-то, что умеет собрать и отдать за него эти метрики. Гуглить по запросу "<название софта> export metrics".  
  
- Нам не важно в какой именно инфраструктуре вы поднимете сервис, можете делать в том числе на любой личной. Если для выполнения проекта вам необходимы ресурсы в Яндекс.Облаке - можете запросить их по инструкции из второго домашнего задания, мы добавим вас в организацию DevOps-тренировок - https://github.com/krya-kryak/y-y-devops-trainings-cloud-1#технические-моменты .