MOEX 75.51
Рейтинг

MOEX

Инвестиции начинаются здесь

Подписан × Я работаю здесь ×

ptrvsk
11 сен в 18:35

Ускоряем разработку: 5 необычных фич DevOps-платформы

Простой 7 мин 6.3K

Блог компании MOEX, Управление разработкой*, DevOps*

Обзор

Всем привет! Меня зовут Лиза Петровская, я инженер в команде DevOps-платформы MOEX и сегодняшний рассказ будет посвящен именно ей. В статье я расскажу про пять наиболее любопытных фич и сервисов платформы, которые помогают ускорять процесс разработки и облегчают жизнь инженеров и

Давайте кратко пройдемся по фундаментальным вопросам определения платформы. Мы в MOEX рассматриваем платформу как внутренний продукт, и в ходе разработки ориентируемся главным образом на “боли” и потребности команд. Platform engineering, или развитие платформ в нашем понимании – это процесс создания и предоставления удобных сервисов самообслуживания командам и инженерам, с целью ускорения потока создания ценностей, снижения объема нецелевых активностей команд, а также повышения качества, надежности и безопасности создаваемых решений. При этом важно придерживаться принципа добровольности, при котором команды приходят на платформу без принуждения, когда сами того захотят. Конечно, это сложно и достигается путем постоянного сбора и анализа обратной связи от наших многочисленных и гетерогенных команд. Задача платформенной команды развивать и улучшать свой продукт, добавлять новый функционал, исходя из потребностей пользователей, которые мы выявляем как раз в ходе непрерывного сбора обратной связи, аудита команд, опросов, разговоров у кулера и прочее, прочее.

Немного предыстории: платформа в компании начала активно развиваться примерно в одно время с инженерной трансформацией (о которой можно прочитать здесь). В рамках этой большой трансформации сборки и пайплайны модифицировались и приводились к единым стандартам для всех команд. Вырабатывался унифицированный подход по CI/CD, наводился порядок в инфраструктуре и инструментах. Мы активно выводили инженерный слой из серой зоны и делали первые шаги к созданию платформы. Про это можно послушать в докладе нашего руководителя платформы Карапета Манасяна на DevOpsConf22:

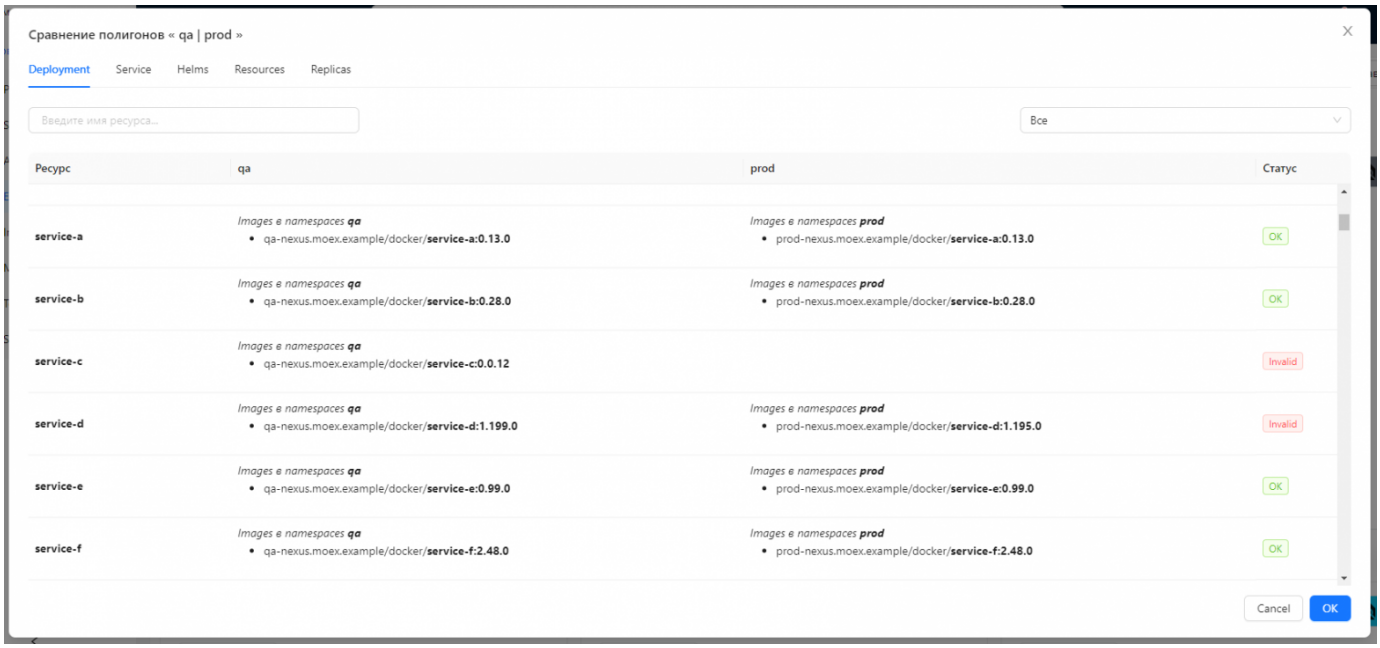
В статье об инженерной трансформации подробно описан путь, который проходит артефакт от DEV до PROD окружения, и как удалось этот путь упростить. Однако, в реальности одного конвейера недостаточно, особенно если мы говорим про огромные цифровые продукты. Команды все равно сталкивались с рядом каждодневных задач, например, узнать все различия между версиями приложений, компонентов, артефактов, сервисов на окружениях. Вот тут в игру вступает фича платформы, которая позволяет экономить часы и за пять секунд узнать в реальном времени все отличия между окружениями.

Квест. Найди 10 Отличий

Итак, первая из нашего сегодняшнего топа, необычная, и уже незаменимая для команд фича - сравнение окружений от dev до prod.

Только представьте, у вас есть полигон, которым пользуется команда разработки, полигон для автотестов, полигон для интеграционных тестов, для нагрузочных, пре-прод, и, наконец, сам prod. В процессе поставки зачастую членам команды нужно быстро сравнить окружения. Какие есть варианты? Зайти в кубера и сравнить глазами. Да, но уйдет очень много времени. Второй вариант – сравнить yaml-манифесты репозиториев, являющихся частью
GitOps-процесса. Да, но там не вся необходимая информация, а только версии helm-чартов. В любом случае, уйдет много времени на нецелевую работу команды. Эту проблему мы решили с помощью инструмента для мониторинга и сравнения окружений. Почему не использовали готовое решение? Инструментов для мониторинга k8s множество, но вот настолько кастомных нет. Наиболее часто используется именно сравнение по версиям сервисов, но также есть возможность сравнить версии любых объектов k8s (deployment, ingress, replicaSet и т.д.), helm-чартов. При этом мы не показываем секреты или увствительную информацию, а лишь отмечаем наличие расхождений.

Кстати, само сравнение можно проводить как в режиме «помониторить один полигон», так и сравнивать неограниченное количество полигонов между собой. А отличия подсвечиваются, чтобы точно не пропустить.



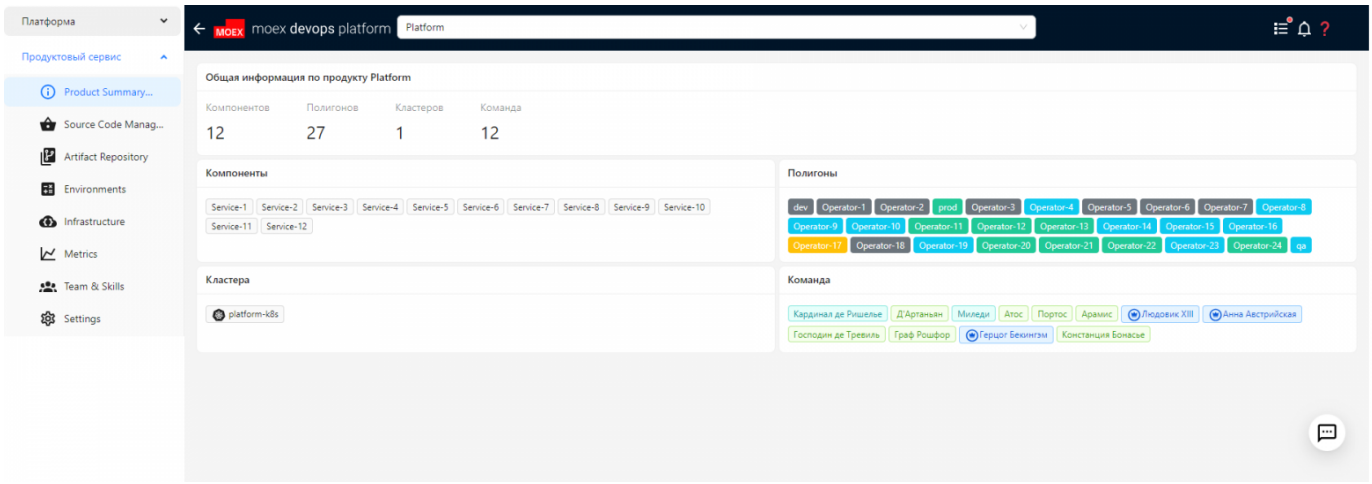
Сравнение тестовых полигонов

Мультитул для тимлида

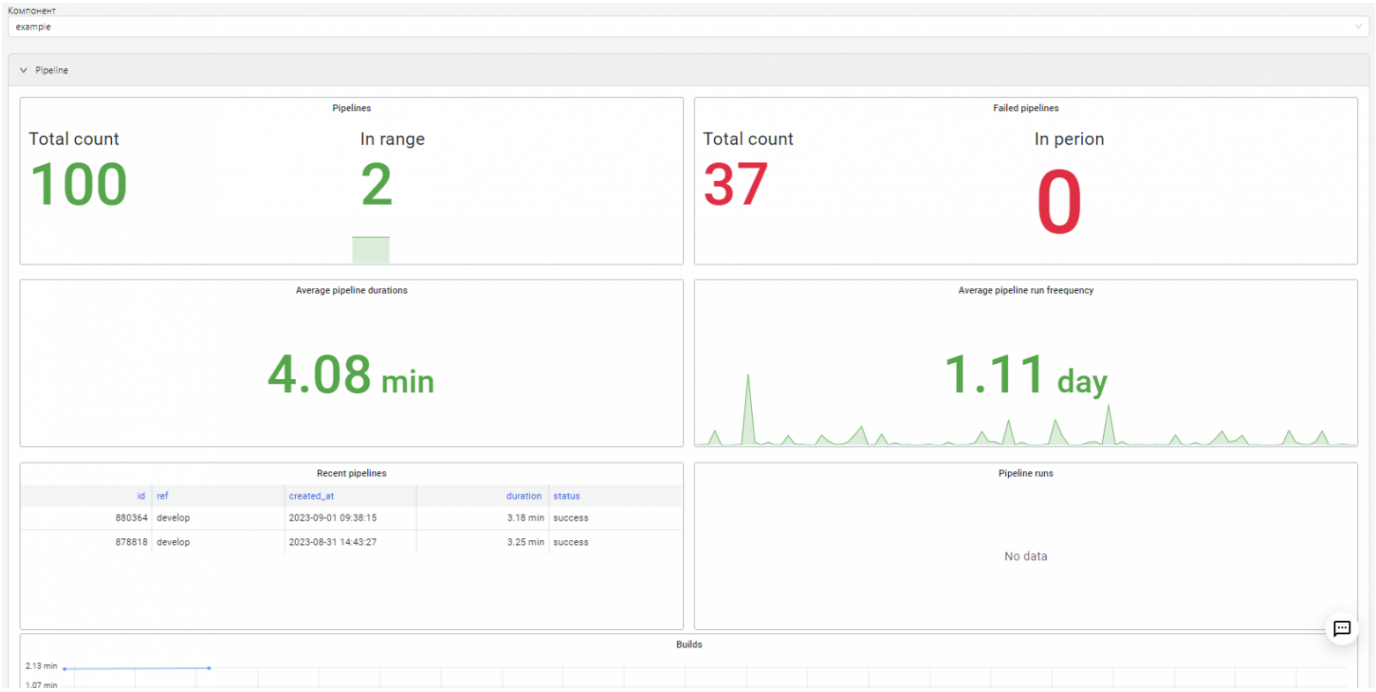
В MOEX разрабатываются сильно отличающиеся друг от друга продукты. Например, высоконагруженные торговые системы и их ядра с особенно высокими требованиями к производительности разворачиваются на железных серверах, а микросервисные Финуслуги разрабатываются в куберах. Тем не менее разнородность ИТ-ландшафта, кардинальные различия в подходах и технологическом стеке от команды к команде не мешают нам делать одну платформу, удобную для всех. Платформа строится вокруг абстракции продуктов и проектов, а не процессов и инструментов.

Поясню, платформа - единая точка входа к множеству услуг, сервисов и инфраструктуре, но работа на платформе начинается с выбора/создания продукта.

Заводим продукт, добавляем компоненты, добавляем зависимости, интеграции, тестовые полигоны, команду. И метрики у нас сюда подключаются, и дашборды из графаны рисуются.



Дашборд продукта

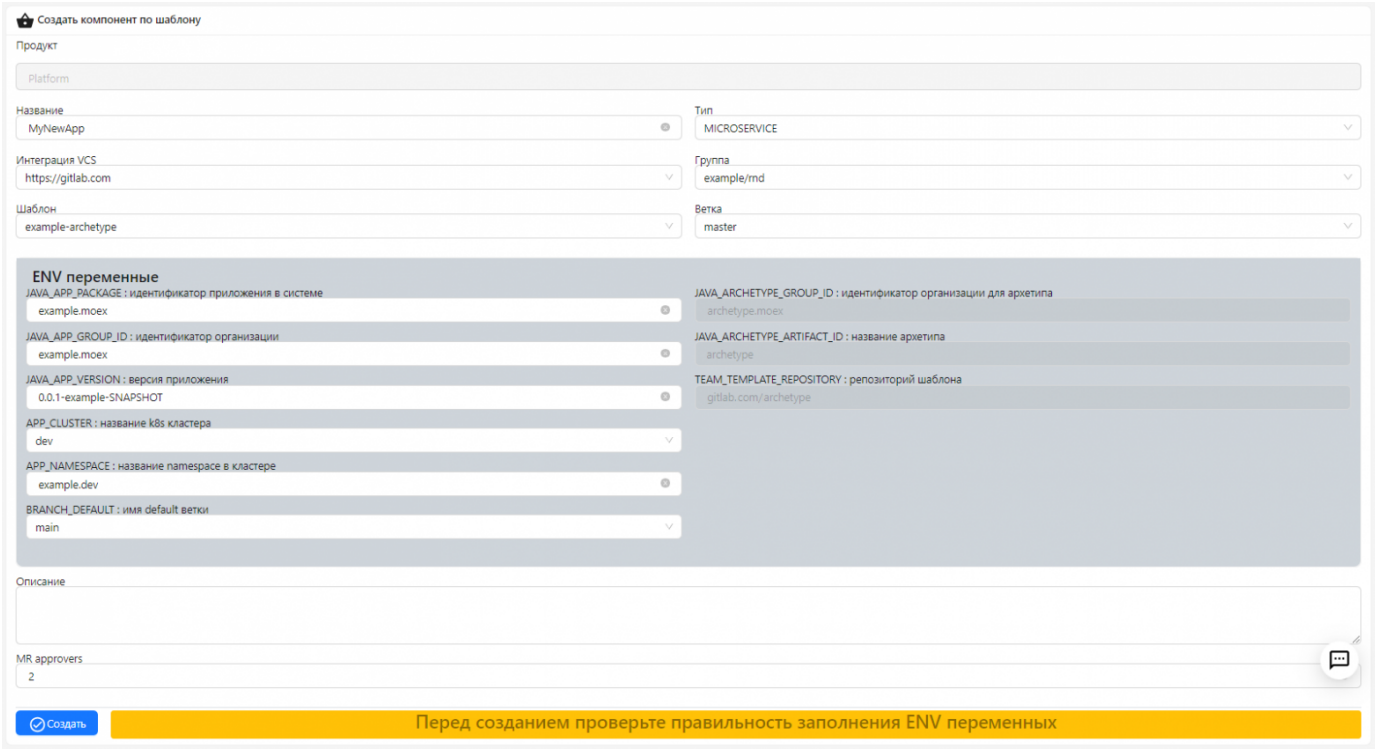


CI/CD метрики по продукту

Вот так получаем продуктовое observability, как для инженера, так и прозрачность разработки и ИТ-процессов для бизнеса.

Микросервисы на кончике клика

Не будем уходить далеко от разработки. Тут речь пойдет о шаблонах для быстрого создания микросервисов. Это удобно, эффективно, избавляет от ручных рутинных действий. Как я писала выше, команды и продукты на Бирже сильно отличаются друг от друга, поэтому на платформе есть как общие шаблоны по языкам программирования (наподобие тех, что предоставляет Gitlab), так и специальные шаблоны для команд, которые создаются и добавляются по обращению и в ходе совместной проработки шаблона. Большинство шаблонов параметризованные. Вот пример шаблона для определенной команды, с требуемыми им настройками.



Создание микросервиса из шаблона

Польза очевидна - не тратятся часы на копирование кода из старого проекта, все готово, как говорится, из коробки. Процесс создания микросервиса, уже преднастроенного под конкретную команду, занимает от силы три минуты. Дальше только бизнес-логику писать.

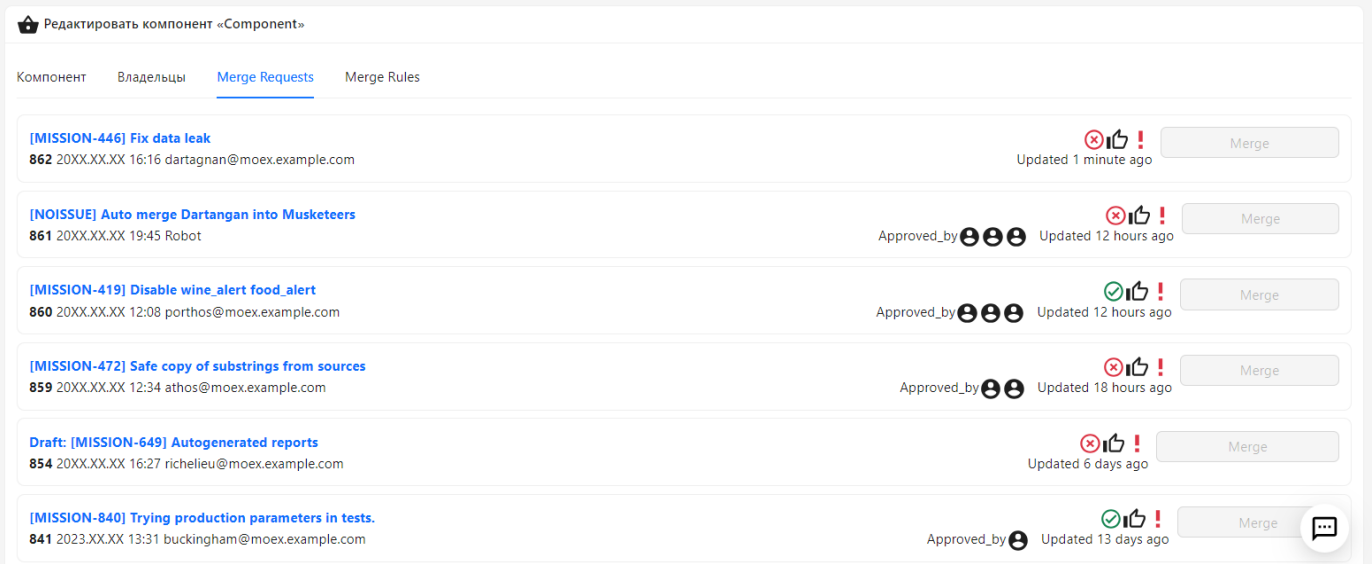
Мерж реквесты. Важен каждый голос

Платформа дает нам площадку и подспорье для экспериментов. Так что по просьбам пользователей мы сделали на платформе возможность настроить кастомные правила для Merge Request (MR) на ветку/группы веток в проекте. Вдохновлялись тем, как это устроено в Gitlab Premium, но добавили кое-что от себя, например, регулярные выражения для настройки правил и более гибкие политики для указания конкретных «аппруверов».

Правило представляет из себя:

- ветку или несколько веток
- минимальное количество «аппрувов», требуемых для разрешения MR
- группу людей, чьи «аппрувы» будут учитываться (инициатора MR можно учитывать или нет)

Как это выглядит:

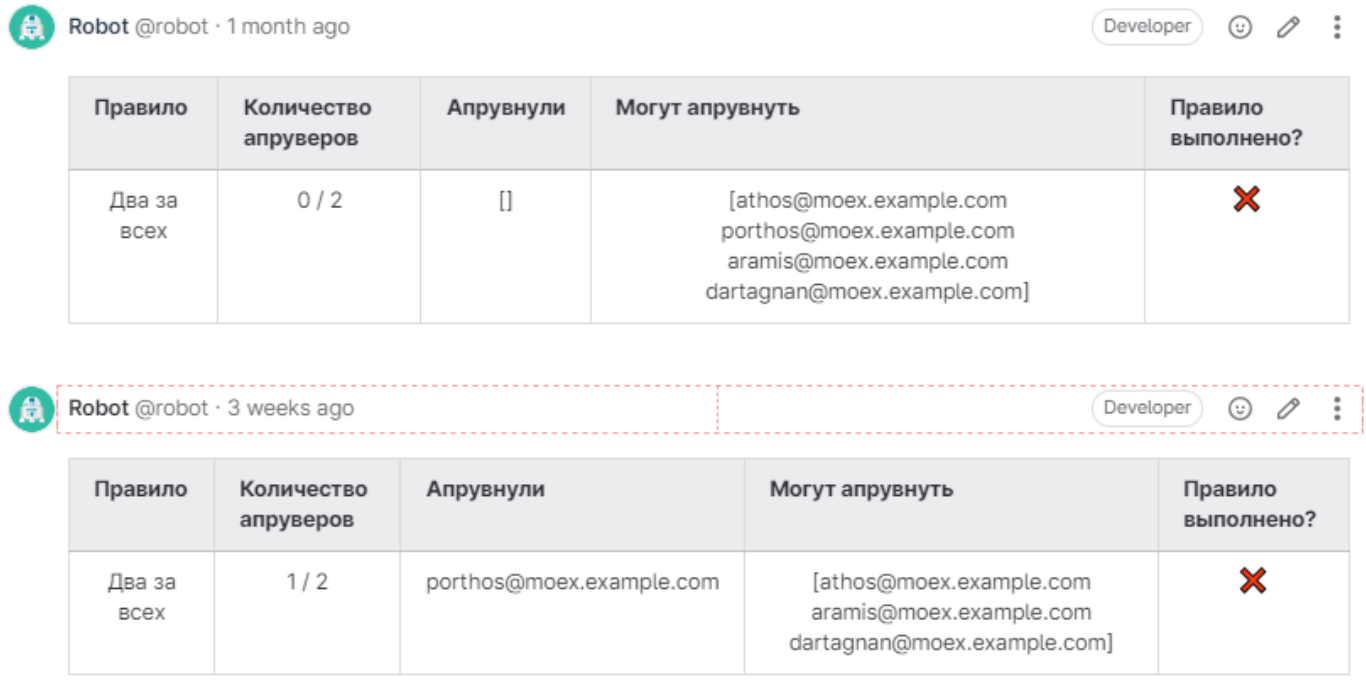


Мерж-реквесты на платформе

На платформе транслируется информация из Gitlab:

- отображается статус MR
- количество «аппрувов»
- наличие конфликтов
- возможность автоматического вливания MR

Со стороны Gitlab это выглядит как дискуссия, блокирующая MR:



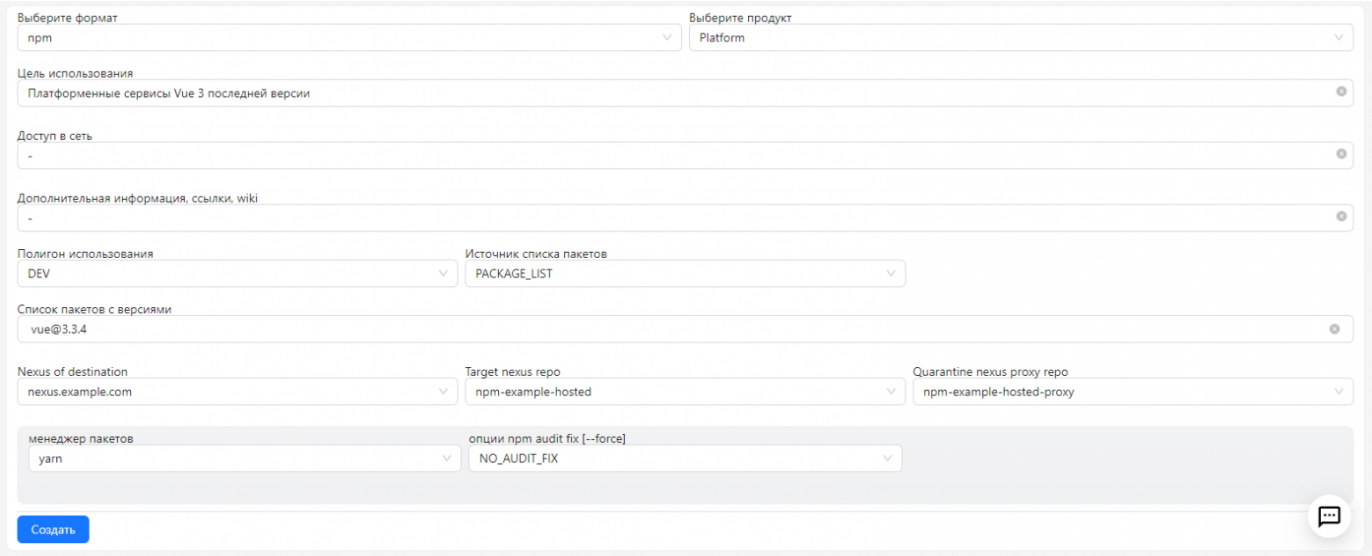
Безопасное проникновение в контур

Вот уже более полутора лет мы живем в условиях новой реальности и предпочитаем находиться в позиции недоверия: доступ в публичный интернет ограничен из внутренней сети. Процесс получения зависимостей, артефактов из интернета, обновления библиотек у нас был довольно сложным с точки зрения времени ожидания и количества задействованных людей.

Там, где раньше было «docker pull», теперь «для проекта N просьба скачать докер-образ NN для использования в сегменте NNN».

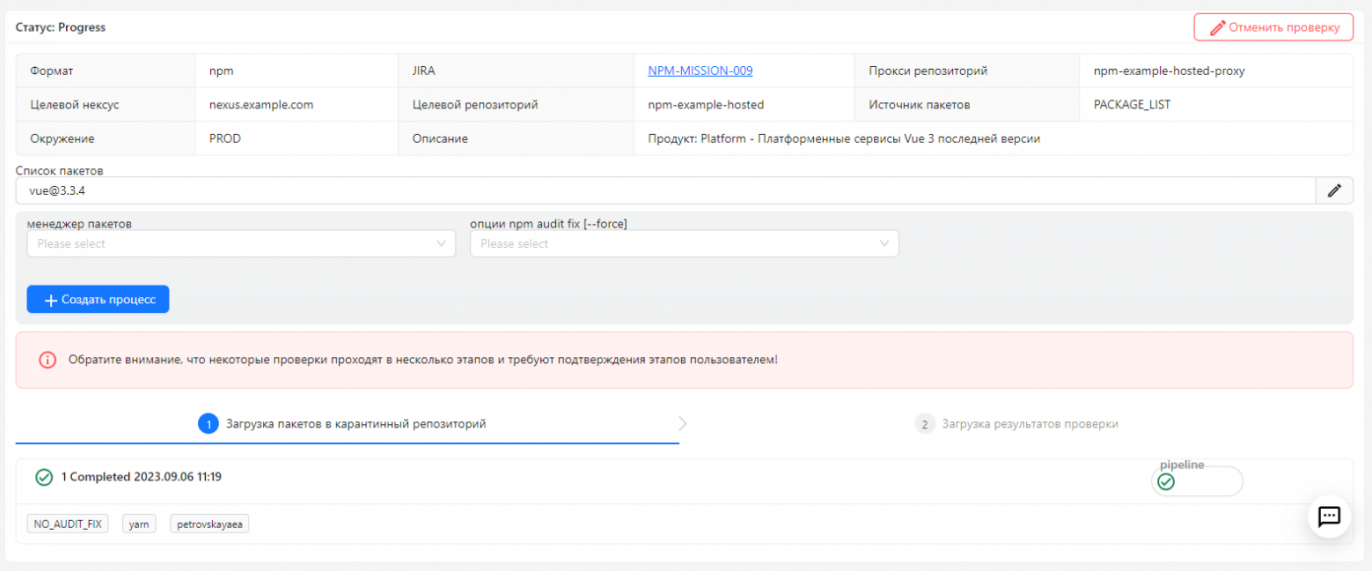
Радости такой процесс, конечно же, не приносит, силы отнимает, поэтому на платформе мы сделали сервис безопасной загрузки артефактов из интернета.

Пользователь выбирает тип артефакта, заполняет заявку в диалоговом окне, после чего запускается процесс загрузки пакетов в карантин. Строится дерево зависимостей, проверяется наличие файлов во внутренних репозиториях, осуществляется проверка контрольных сумм. После чего заказчик подтверждает, что ему нужны именно эти артефакты. Далее скачанные файлы и отчет отправляются на проверку команде информационной безопасности.



Заявка на получение артефактов

Созданные заявки отображаются на платформе, можно посмотреть, проверить статус, отменить.



Заявка на получение артефактов

Этим сервисом мы постарались снять нагрузку с дежурных, разгрузить ИБ. Сам по себе процесс ускорить, исключив, где возможно, человеческий фактор, ожидание, автоматизировав ручные действия. Избавились от ситуации с повторными запросами пользователей, когда не догрузили транзитивные зависимости, проект не собирается, человек возвращается и по новой ждет проверки новых пакетов.

Может возникнуть вопрос, почему этот сервис располагается на платформе с UI, а не вызывается только через API с ПК разработчика. Мы считаем, что, заполняя сперва диалоговую форму, человек сильнее задумывается о вопросах безопасности и необходимости использования того или иного пакета. То есть он понимает свою причастность к тому или иному выбранному артефакту. А когда этот процесс полностью скрыт от него, то ответственность смещается на всех, кроме самого разработчика.

В планах по развитию этого сервиса прикрутить к процессу скачивания и первичной проверки различные DevSecOps инструменты.

Платформа - это также про постоянное улучшение DevExp, про прозрачность и избавление от узких мест в виде процессов и людей.

Конечно, у нас были ошибки. Во-первых, в самом начале пути мы выпускали фичи, которые, как нам казалось, будут наиболее полезны для команд, вместо проведения аудита и сбора реальных запросов и потребностей. А также мы начинали разрабатывать новый функционал сразу в «чистовом виде» без PoC. Тратили много ресурсов, не имея уверенности, взлетит фича или нет. Так что, хозяйке на заметку, самое главное - решать реальные проблемы, устранять реальные боли разработчиков, то есть общаться, спрашивать, проводить аудит, собирать обратную связь и статистику использования.

Нет единого рецепта для всех, нет двух одинаковых платформ, но определенные подходы и идеи можно и нужно пробовать внедрять. Это были пять наиболее

любопытных и необычных возможностей и сервисов нашей внутренней инженерной платформы, которыми хотелось поделиться с миром. Главное, что эти фичи экономят нам время, силы, ресурсы и сокращают TTM. И самое-самое главное, ключевой фактор успеха - это наша большая сплоченная команда настоящих профессионалов, без которых ничего этого невозможно было бы построить и развивать!

> Мы искренне уверены, что инвестиции в команду и в людей – это основной фактор успеха.

В следующих статьях расскажем о других возможностях нашей платформы, а пока пишите в комментариях, если стоит подробнее осветить какую-то из представленных фич.

Теги: platform engineering, idp, internal developer platform, platform as a service, devsecops, cicd, devops, разработка

Хабы: Блог компании MOEX, Управление разработкой, DevOps