Безопасность контейнерных сред. Как мы встали на путь разработки собственного решения и что из этого получилось

Игорь Вербицкий

Руководитель отдела ИБ

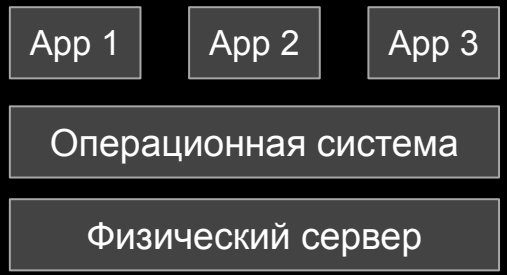
Саша Рахманный

Старший разработчик систем информационной безопасности

latech

Типовые этапы развития инфраструктуры

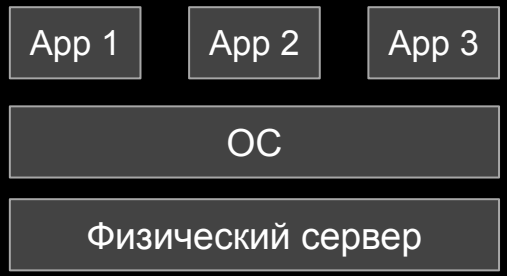
Железная
Сервисы располагаются на выделенных физических серверах

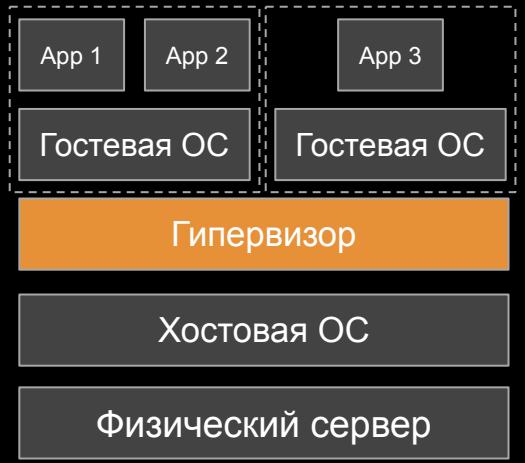


Типовые этапы развития инфраструктуры

Железная
Сервисы располагаются на выделенных физических серверах

Виртуализация
Часть сервисов располагаются на физических серверах
Часть сервисов располагаются в среде виртуализации



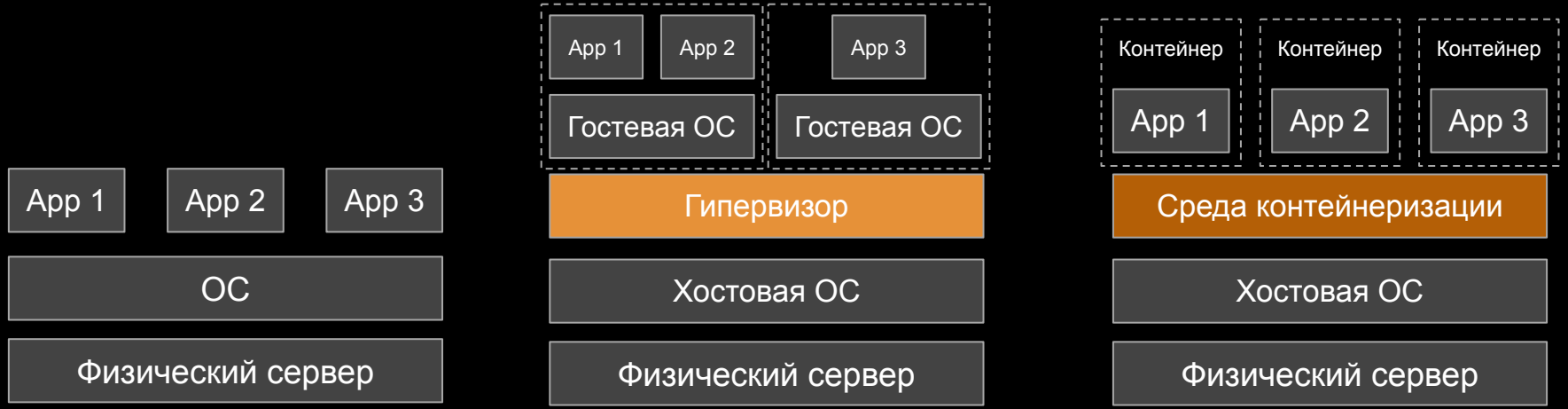


Типовые этапы развития инфраструктуры

Железная
Сервисы располагаются на выделенных физических серверах

Виртуализация
Часть сервисов располагаются на физических серверах
Часть сервисов располагаются в среде виртуализации

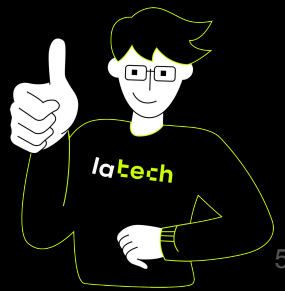
Контейнеризация
Большинство микросервисов развернуто в контейнерной инфраструктуре



Плюсы контейнеризации для команд разработки/devops

1

Ресурсы (изоляция): для развертывания контейнера требуется меньшее количество “железных” ресурсов



Плюсы контейнеризации для команд разработки/devops

1. **Ресурсы (изоляция):** для развертывания контейнера требуется меньшее количество “железных” ресурсов

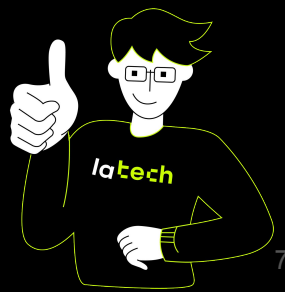
2. **Скорость и масштабируемость:** не нужно тратить время для подготовки “операционной системы”, в короткий промежуток времени можно запустить нужное количество “образов” сервиса

Плюсы контейнеризации для команд разработки/devops

1. **Ресурсы (изоляция):** для развертывания контейнера требуется меньшее количество “железных” ресурсов

2. **Скорость и масштабируемость:** не нужно тратить время для подготовки “операционной системы”, в короткий промежуток времени можно запустить нужное количество “образов” сервиса

3. **Переиспользование:** возможность использовать “чужие” образы для развертывания



“Особенности” контейнеризации для команд безопасности

1

Иная технология по отношению к классической инфраструктуре

“Особенности” контейнеризации для команд безопасности

1. Иная технология по отношению к классической инфраструктуре

2. В короткий промежуток времени в инфраструктуре может появиться большой объем уязвимых сервисов

“Особенности” контейнеризации для команд безопасности

1. Иная технология по отношению к классической инфраструктуре
2. В короткий промежуток времени в инфраструктуре может появиться большой объем уязвимых сервисов
3. Чужие образы могут принести уязвимости, которые станут нашими

Как было раньше?

Механизмы применяемые в “классической” инфраструктуре:

- Сегментация межсервисного взаимодействия (Firewall)
- Анализ сетевого трафика (IDS\IPS)
- Анализ хостов на уязвимости (Vulnerability scanners)
- Защита конечных устройств (EDR)
- Управление правами доступа

Как было раньше?

Механизмы применяемые в “классической” инфраструктуре:

- Сегментация межсервисного взаимодействия (Firewall)
- Анализ сетевого трафика (IDS\IPS)
- Анализ хостов на уязвимости (Vulnerability scanners)
- Защита конечных устройств (EDR)
- Управление правами доступа

Что нужно еще?

- Сканирование на уязвимости самих образов
- Сканирование подключаемых библиотек в ПО
- Сканирование наличия секретов в контейнерах
- Анализ настроек средства контейнеризации
- Анализ файловой системы на наличие вредоносного ПО

Какие “Enterprise” решения мы смотрели

Какие “Enterprise” решения мы смотрели

Aqua Security

Плюсы

- Зрелый продукт с большим набором функций
- Удобный инструментарий для управления

Минусы

- Нет поддержки всех необходимых для нас версий Kubernetes
- Есть риски продление\использования решения

Какие “Enterprise” решения мы смотрели

Aqua Security

Плюсы

- Зрелый продукт с большим набором функций
- Удобный инструментарий для управления

Минусы

- Нет поддержки всех необходимых для нас версий Kubernetes
- Есть риски продление\использования решения

Kaspersky Container Security

Плюсы

- Отечественный продукт с гарантированной поддержкой
- Понятный существующий роадмап и заинтересованность вендора в развитии продукта

Минусы

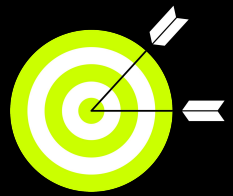
- Нет гарантированной поддержки всех необходимых для нас версий Kubernetes
- Зрелость продукта нам показалась не оптимальной

А что нам предлагает Open-Source?

	Сканирование установленных пакетов в образе	Сканирование подключаемых библиотек в ПО	Сканирование секретов	Встраивание в CI\CD	Сканирование FS	Анализ настроек	Экспорт в JSON
Grype	+	+	-	+	+\- (только сохраненные образы docker и Skopeo/SIF)	-	+
Trivy	+	+	+	+	+	+	+
Docker Scout	+	-	-	+	-	-	+
*Yandex	+	-	-	-	-	-	-

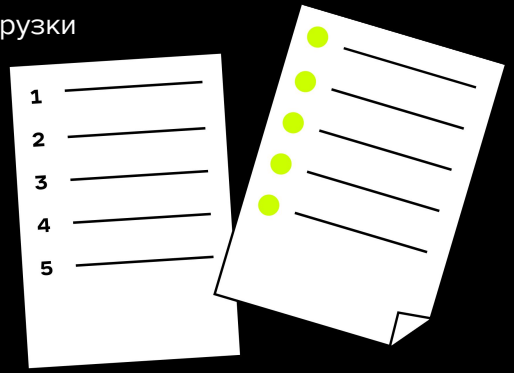
Grype - это open source проект Anchore.

Trivy - это open source проект Aqua Security.



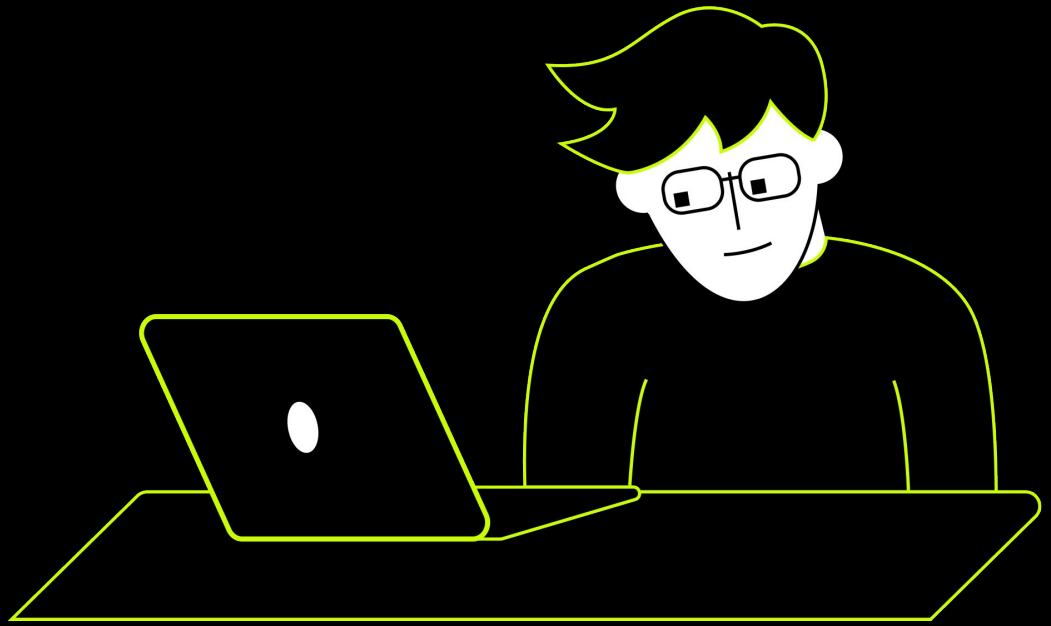
Ничего не подходит, что дальше? Пишем свой сервис!

1. Проверка образов контейнеров на разных этапах: репозиторий контейнеров, сборка CI/CD, runtime в системе оркестрации
2. Проверка образов контейнеров на наличие уязвимостей и вредоносной нагрузки
3. Проверка настроек политики безопасности системы оркестрации
4. Должна быть удобная возможность работать с результатами
5. Должна быть написано с душой и креативом: модный стек технологий, универсальность, масштабируемость...



Что дальше?

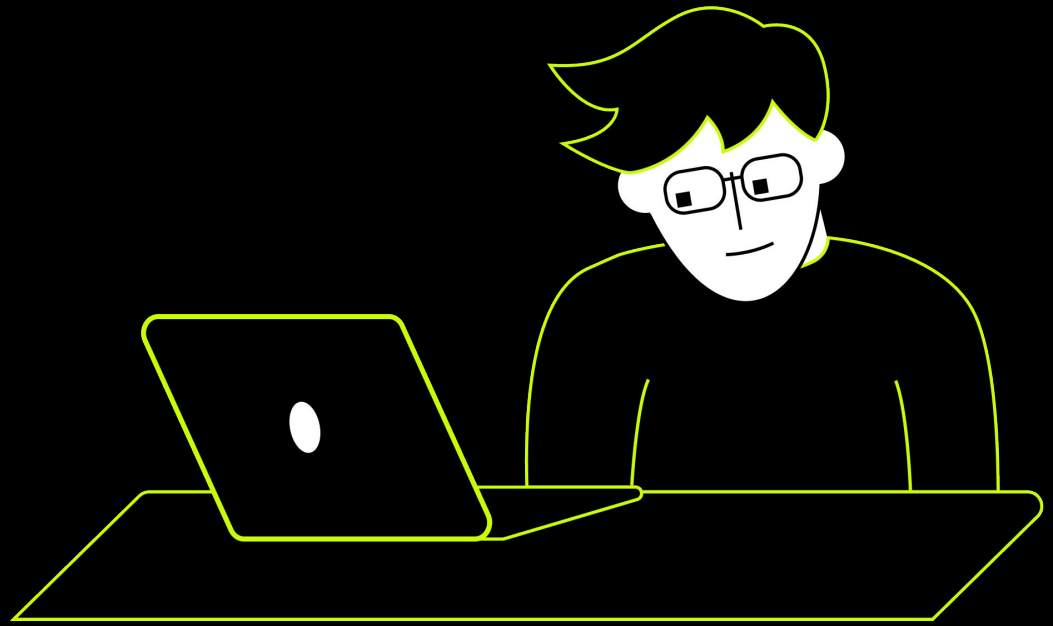
Отрицание



Что дальше?

Торг

Отрицание

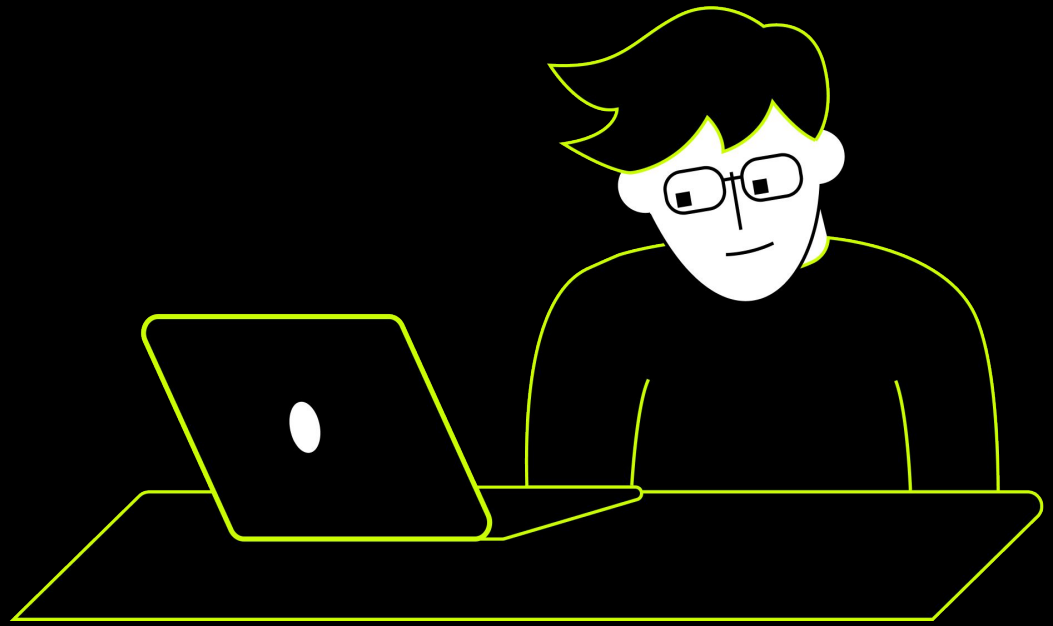


Что дальше?

Гнев

Торг

Отрицание



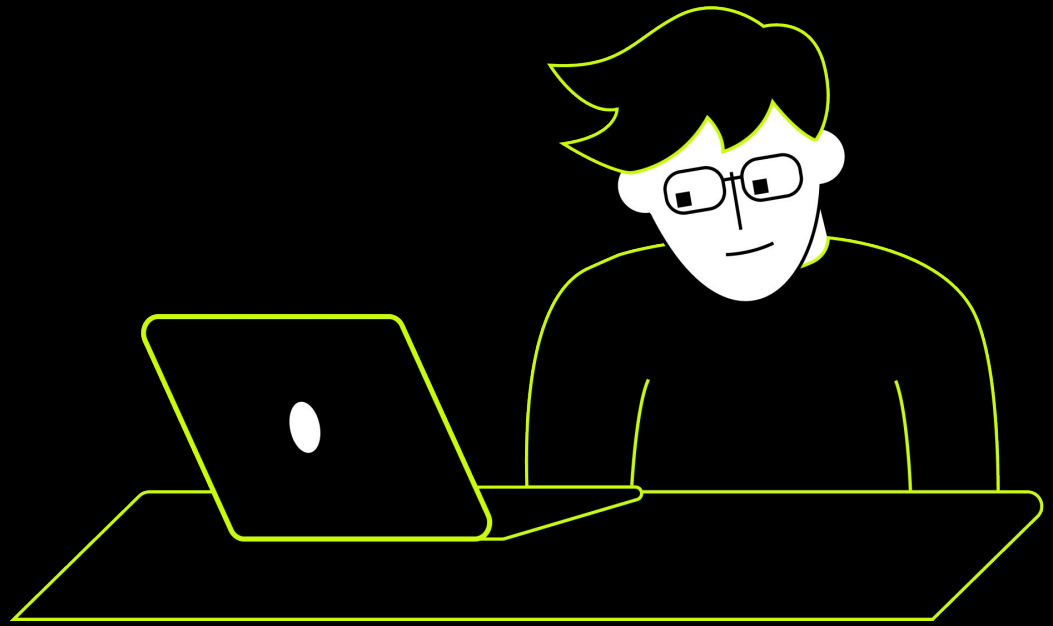
Что дальше?

Гнев

Депрессия

Торг

Отрицание



Что дальше?

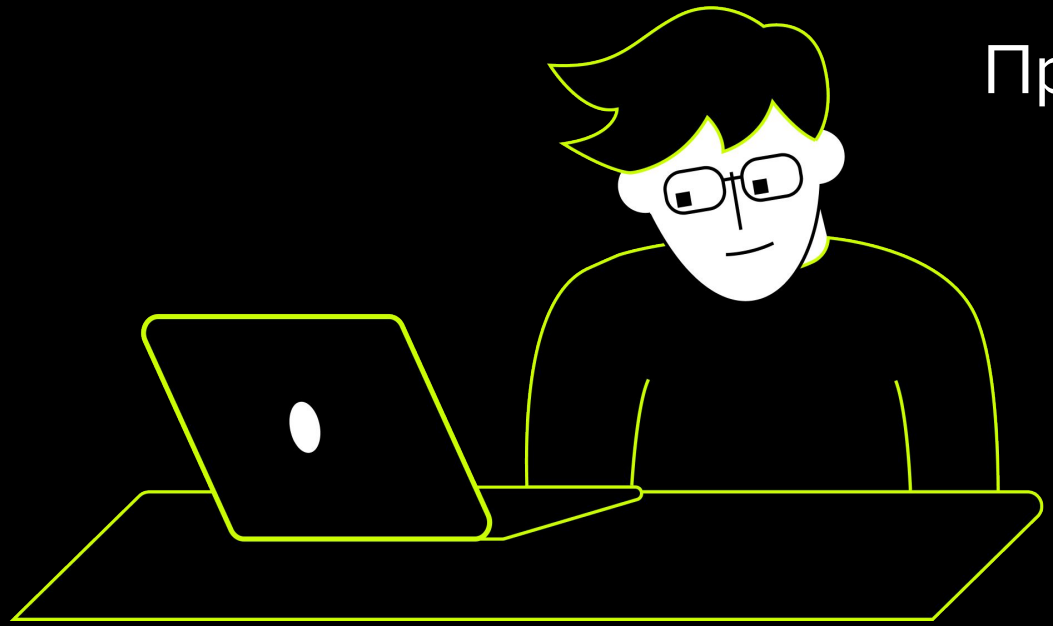
Гнев

Депрессия

Принятие

Торг

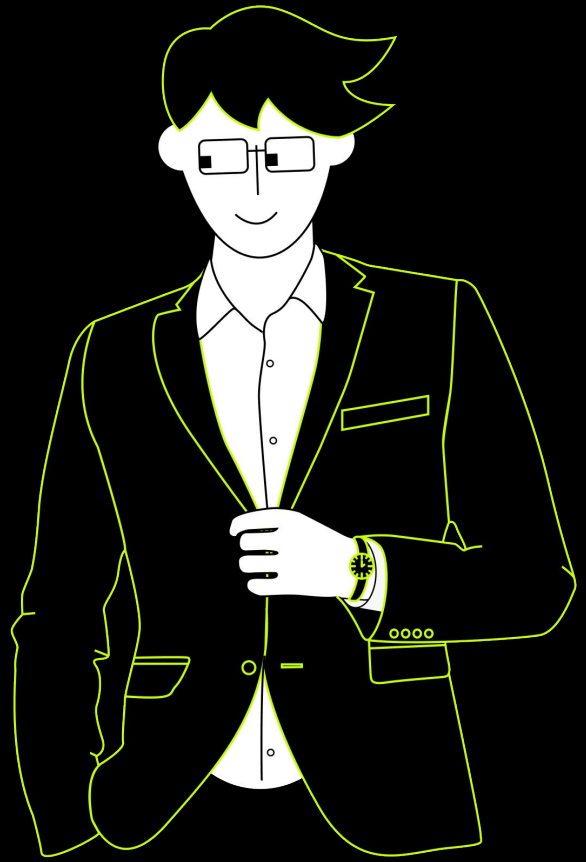
Отрицание



На самом деле, все не так!

Оценка нашей экспертизы и затрат на разработку, привела к пониманию, что мы «могём». Приступили к планированию ресурсов и реализации.

А что из этого получилось и как оно в итоге работает, расскажем далее...



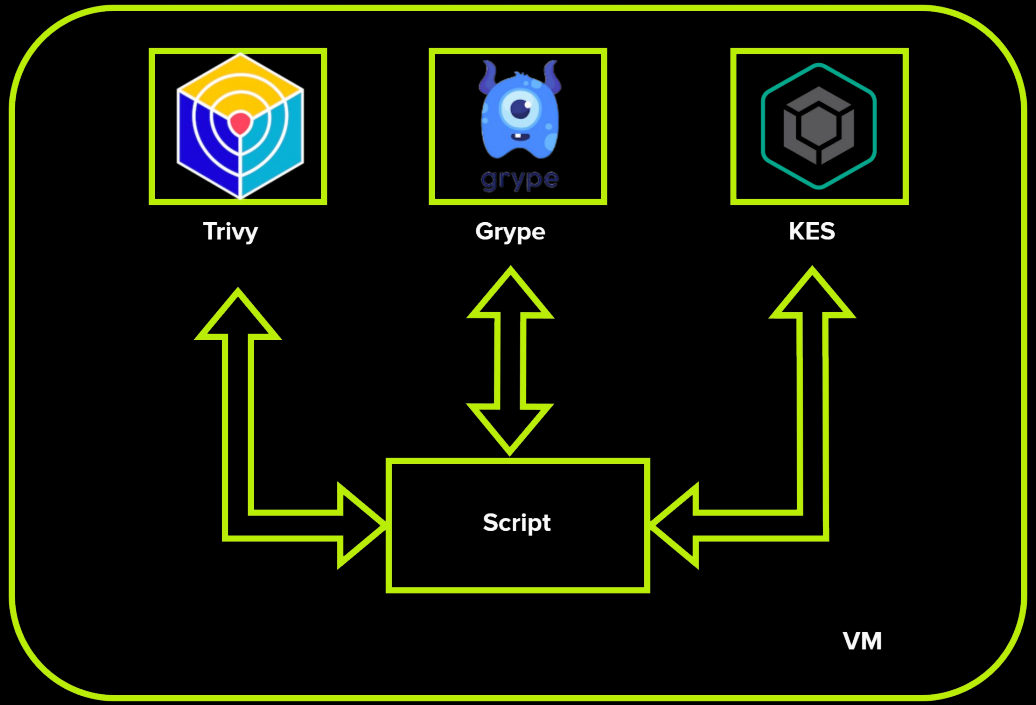
Наша реализация

Напишем скрипт и просканируем весь репозиторий

Подготовили ВМ с установленными сканерами:

- Trivy
- Grype
- Kaspersky Endpoint Protection

Написали скрипт, взаимодействующий со сканерами и репозиторием

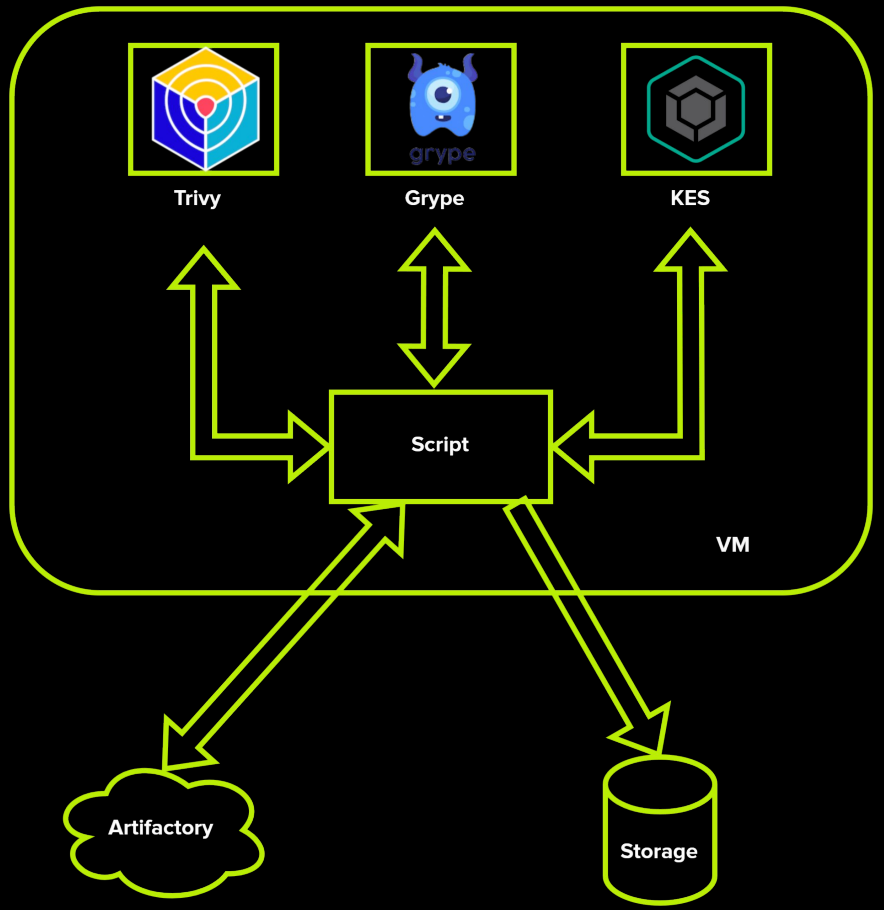


Напишем скрипт и просканируем весь репозиторий

Запустили анализ всех последний версий образов в репозитории
Выгрузили вариант в хранилище для последующей загрузки в SIEM

Проблемы:

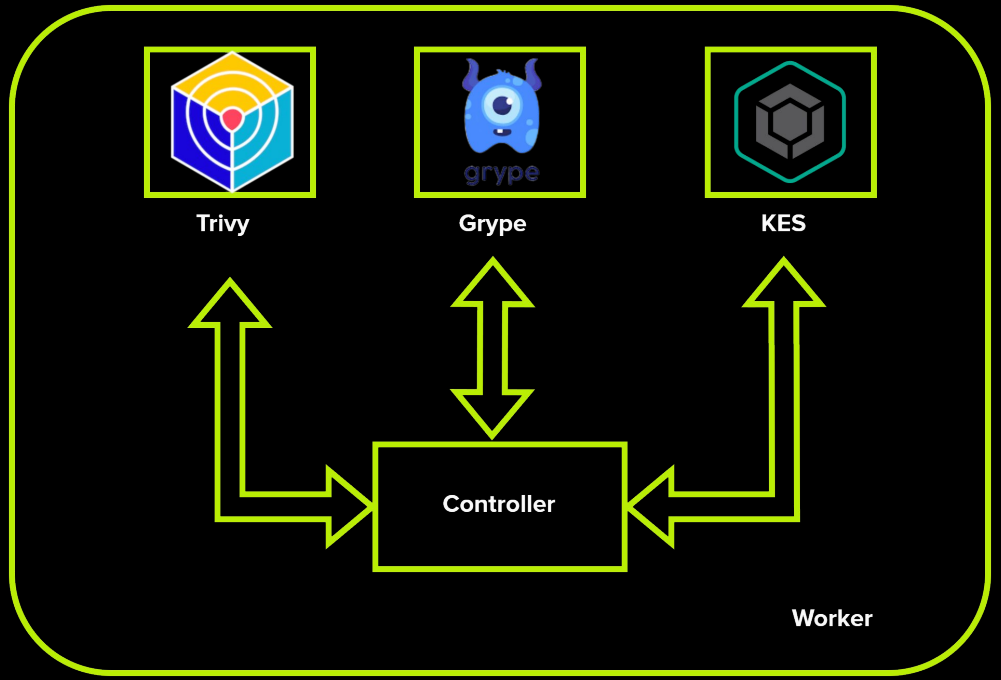
- Долгое сканирование(3 недели)
- Нет информации об актуальности образов
- Данные для импорта в SIEM нужно нормализовать



Распределенная архитектура

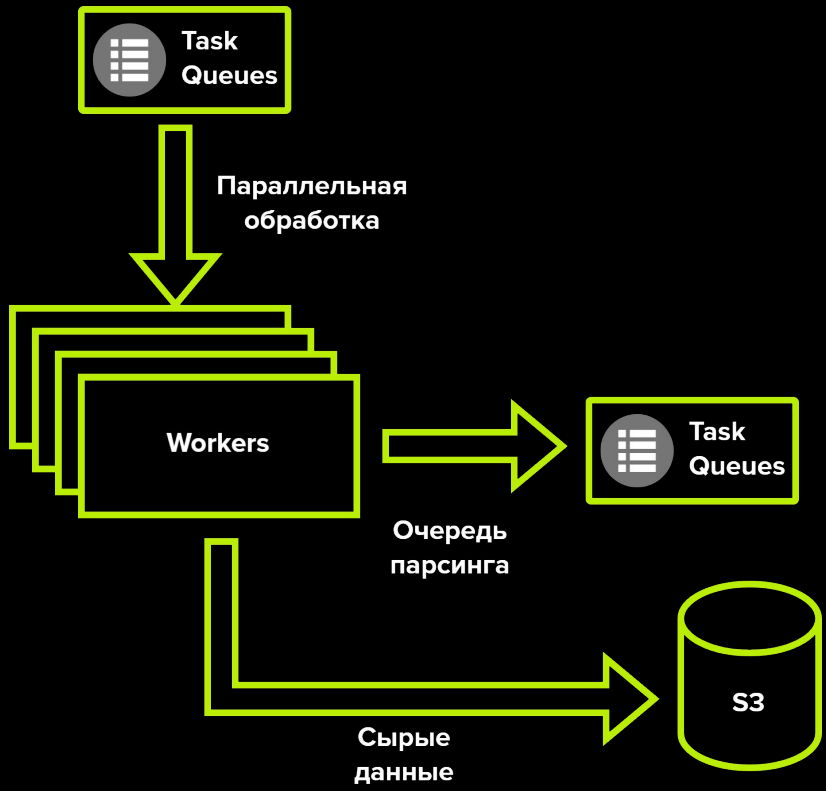
Разработали новую архитектуру:

- Собрали контейнер (Worker), включающий в себя сканеры.
- Написали контроллер, взаимодействующий с очередями Amazon SQS и хранилищами Amazon S3



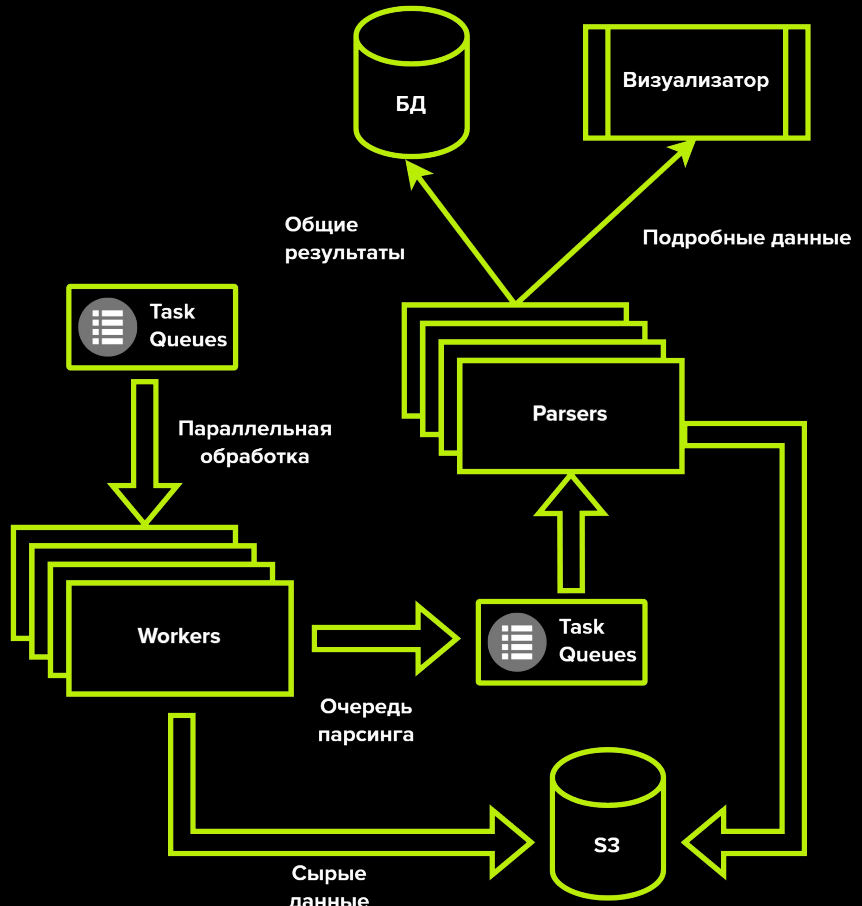
Распределенная архитектура

- Воркеры получают задание из очереди сканирования.
- Сырые результаты складываются в S3 хранилище
- Информация о результатах сканирования образа помещается в очередь парсинга



Распределенная архитектура

Парсеры получают информацию из очереди, скачивают сырые данные из S3, нормализуют для SIEM и записывают статистическую информацию в БД



Преимущества использования распределенной инфраструктуры

Гибкое масштабирование

Использование очередей позволило гибко масштабировать количество нод сканеров и парсеров. Мы можем динамически изменять количество нод в зависимости от размера очереди

Преимущества использования распределенной инфраструктуры

Гибкое масштабирование

Использование очередей позволило гибко масштабировать количество нод сканеров и парсеров. Мы можем динамически изменять количество нод в зависимости от размера очереди

Консистентность данных

Очереди отвечают за консистентность данных - при критических ошибках на ноде задание вернется в очередь.

Преимущества использования распределенной инфраструктуры

Гибкое масштабирование

Использование очередей позволило гибко масштабировать количество нод сканеров и парсеров. Мы можем динамически изменять количество нод в зависимости от размера очереди

Консистентность данных

Очереди отвечают за консистентность данных - при критических ошибках на ноде задание вернется в очередь.

Гибкая настройка парсеров

Парсинг результатов сканера вынесен отдельно, что позволило настраивать правила нормализации без изменения сканеров и более гибко масштабировать (4 ноды сканера на 1 ноду парсера)

На каком жизненном этапе будем сканировать?

На этапе сборки

Самый простой и логичный вариант.

+Можно остановить дальнейшие шаги развертывания

-Не все образы собираются в CI

На каком жизненном этапе будем сканировать?

На этапе сборки

Самый простой и логичный вариант.

+Можно остановить дальнейшие шаги развертывания

-Не все образы собираются в CI

При помещении нового образа в репозиторий

Используя веб-хук при появлении нового образа

+Сканируем до развертывания в прод
+Можем применить политику запрета деплоя данного артефакта через API

-Нет информации об актуальности образа

На каком жизненном этапе будем сканировать?

На этапе сборки

Самый простой и логичный вариант.

+Можно остановить дальнейшие шаги развертывания

-Не все образы собираются в CI

При помещении нового образа в репозиторий

Используя веб-хук при появлении нового образа

+Сканируем до развертывания в прод
+Можем применить политику запрета деплоя данного артефакта через API

-Нет информации об актуальности образа

При запуске нового контейнера

Подключаемся к k8s кластерам

+Получаем информацию об актуальных образах после развертывания нового контейнера.
+Можем запретить развертывание уязвимых контейнеров.

-Контейнер уже будет в проде на этапе сканирования

На каком жизненном этапе будем сканировать?

На этапе сборки

Самый простой и логичный вариант.

+Можно остановить дальнейшие шаги развертывания

-Не все образы собираются в CI

При помещении нового образа в репозиторий

Используя веб-хук при появлении нового образа

+Сканируем до развертывания в прод
+Можем применить политику запрета деплоя данного артефакта через API

-Нет информации об актуальности образа

При запуске нового контейнера

Подключаемся к k8s кластерам

+Получаем информацию об актуальных образах после развертывания нового контейнера.
+Можем запретить развертывание уязвимых контейнеров.

-Контейнер уже будет в проде на этапе сканирования

Периодически

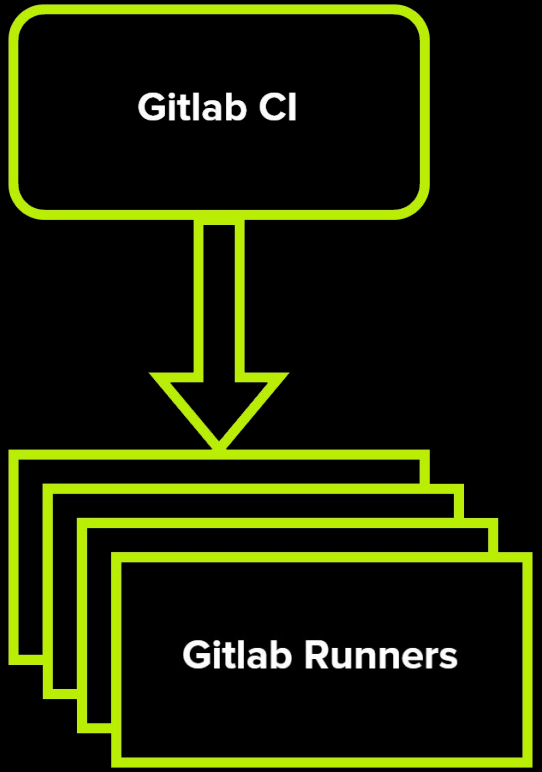
Проверяем образы с актуальными базами

+Периодическое сканирование позволяет найти новые уязвимости в старых образах

Интеграция с CI/CD

Настроили пайплайны в Gitlab CI:

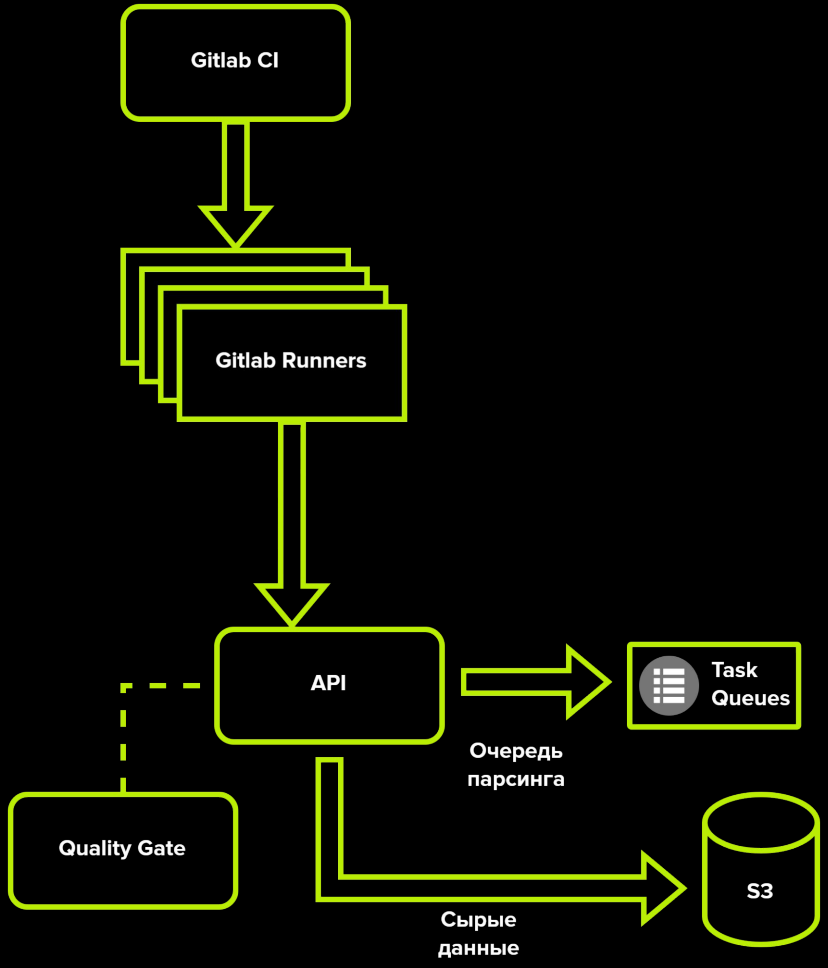
- В образ воркера добавили возможность запуска с передачей параметров через stdin
- Добавили шаг сканирования следующим за сборкой
- После сканирования передаем обратно в CI общие результаты сканирования



Интеграция с CI/CD

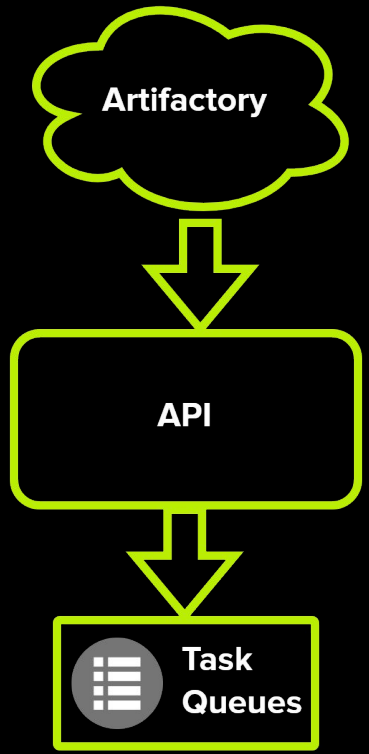
Добавили API:

- Получает информацию о начале сканирования образа
- Получает сырые данные
- Формирует задачи для парсинга
- Заложили возможность оценки уязвимостей с использованием Quality Gate с остановкой дальнейших шагов пайплайна



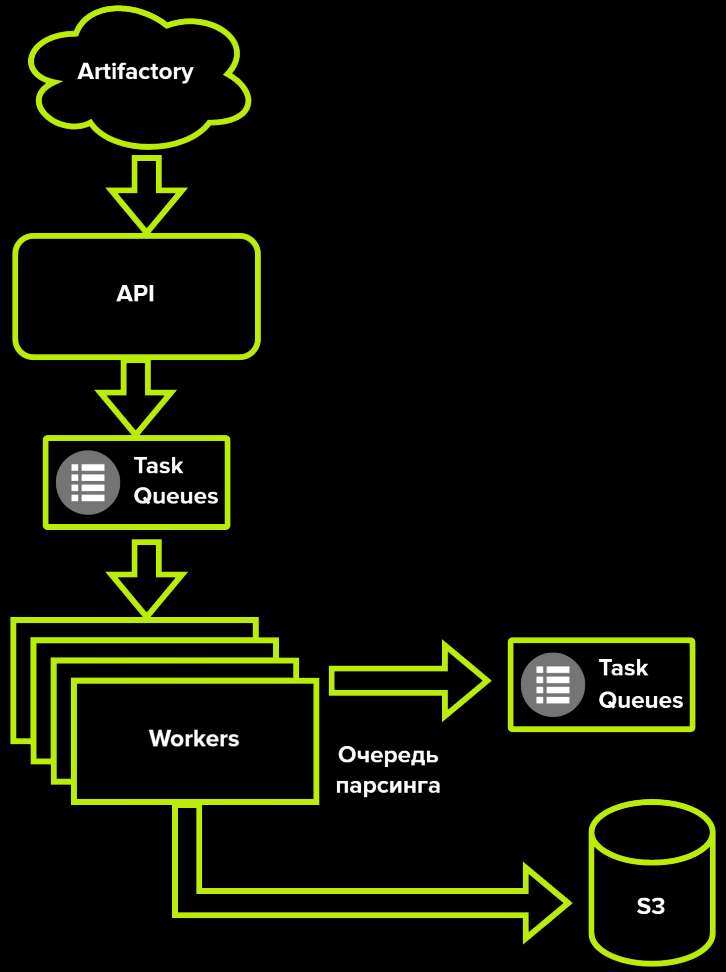
Интеграция с репозиторием

- Настроили отправку веб-хуков при появлении нового образа в репозитории
- Добавили в API функцию приема вебхуков
- API формирует задачу на сканирование образа из веб-хука



Интеграция с репозиторием

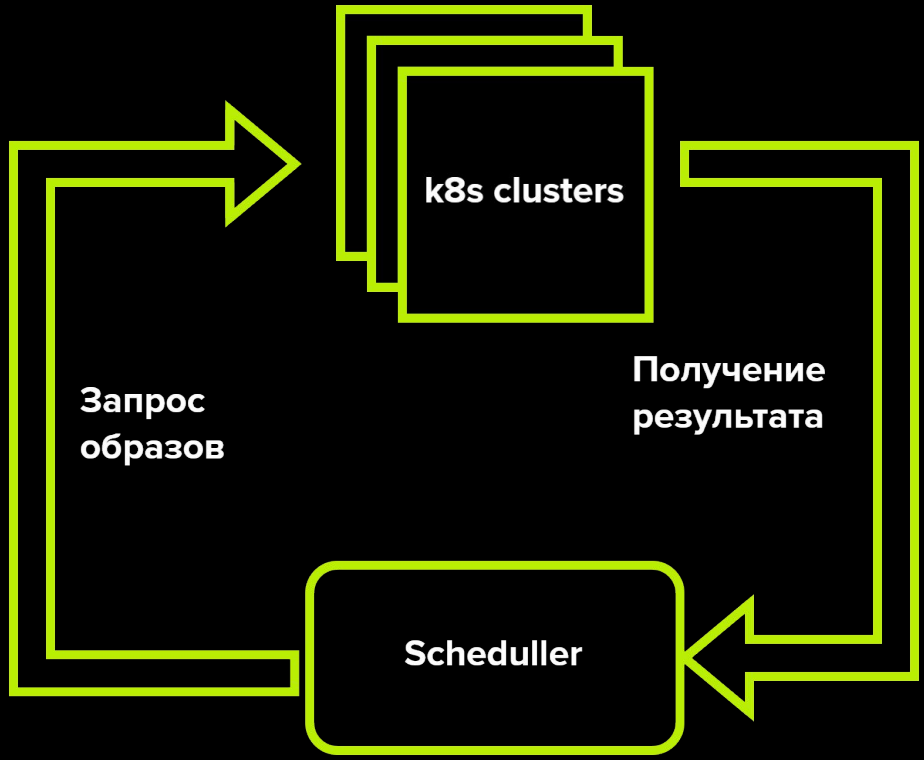
- Воркеры получают задания из очереди и производят сканирование образов
- Сырые результаты сканирования отправляются в S3 Хранилище
- Воркер также формирует задачу на парсинг данных



Интеграция с k8s

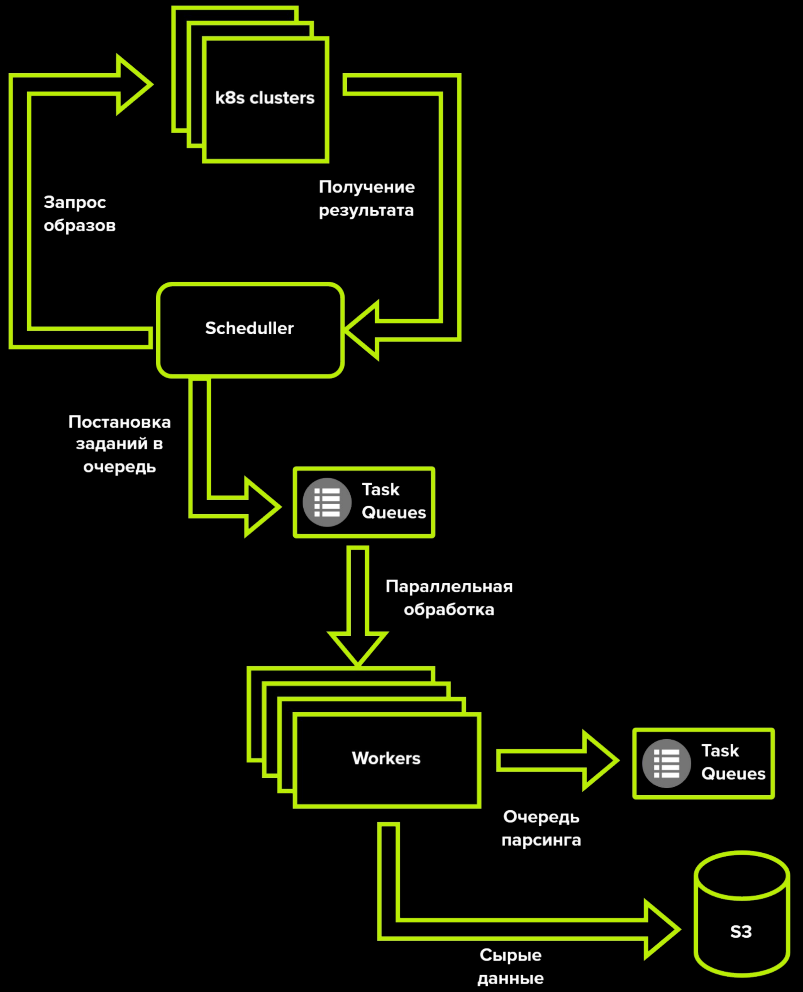
Добавили компонент планировщик:

- Подключается к кластерам
- Запрашивает новые поды и контейнеры, созданные с момента последней синхронизации
- Проверяет, были ли просканированы образы контейнеров ранее
- Также формирует список образов, которые были просканированы более 30 дней назад



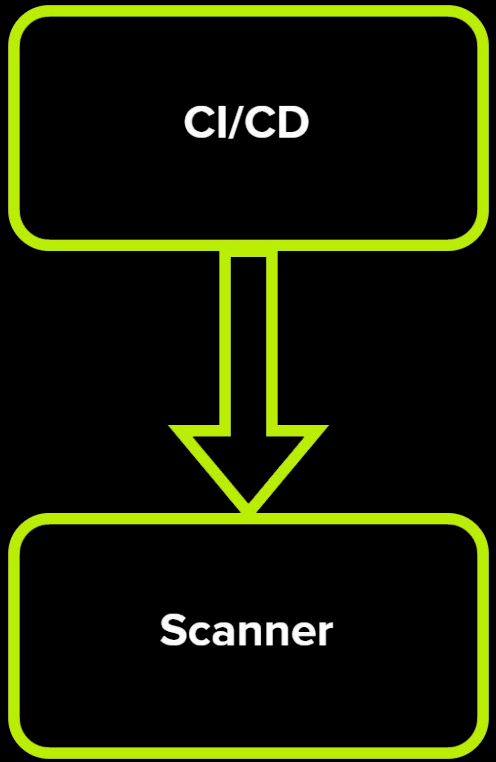
Интеграция с k8s

- Планировщик формирует список образов для сканирования и отправляет в очередь
- Воркеры скачивают образы из репозитория и производят сканирование
- Сырые результаты отправляются в S3
- Формируется задача в очередь парсинга



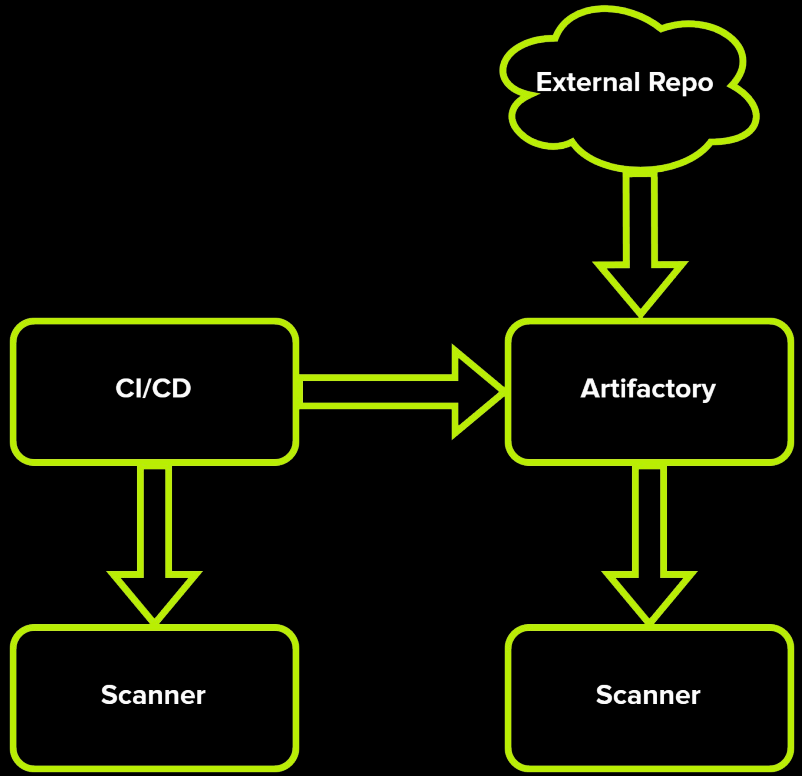
Собираем все вместе

- Проверяем образ на уязвимости на этапе сборки.
- Результат и sha256 хэш записываем в БД. Отправляем образ в корпоративный репозиторий



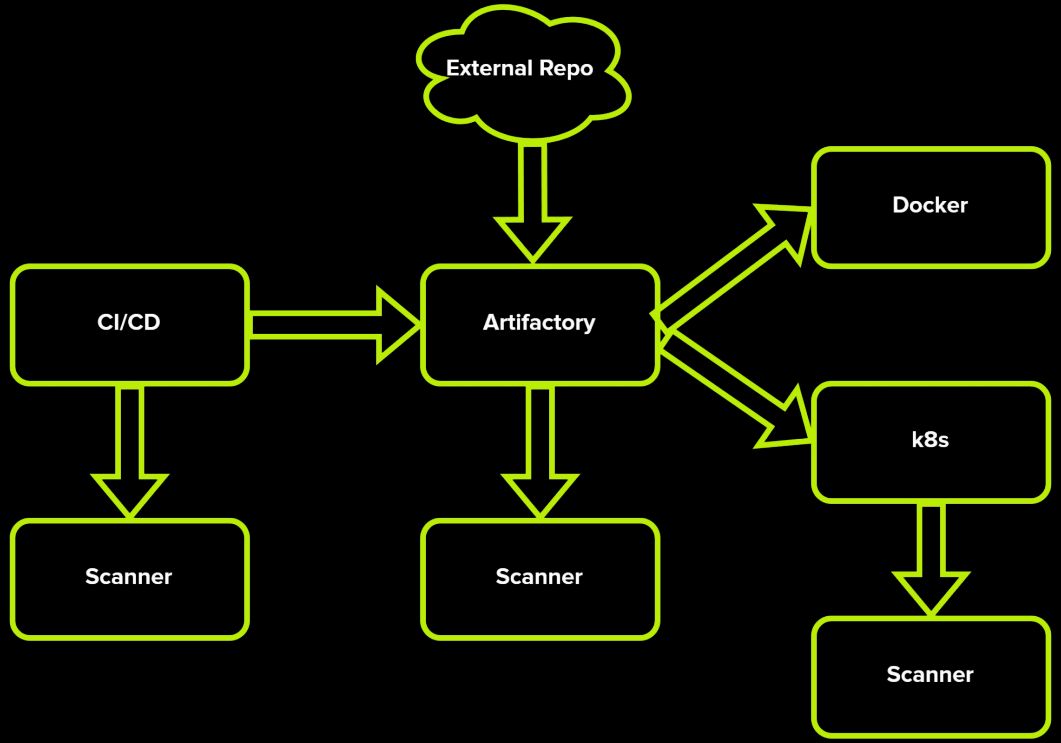
Собираем все вместе

- Образы из внешних репозиториев загружаем в корпоративный.
- При появлении нового образа в корпоративном репозитории отправляем информацию в сканер.
- Сканируем, если нет информации от CI/CD



Собираем все вместе

- Деплоим контейнеры из образов в Docker и k8s.
- Проверяем в кластерах k8s все новые образы по sha256.
- Если дата сканирования более 30 дней назад проводим повторное сканирование



Cloud Ready

1

Ускорение разработки: Развернуть cloud management ресурсы быстро и малозатратно. Бонус - нетарифицируемый лимит (например, 100K запросов в Yandex Queue)

Cloud Ready

1

Ускорение разработки: Развернуть cloud management ресурсы быстро и малозатратно. Бонус - нетарифицируемый лимит (например, 100K запросов в Yandex Queue)

2

Снижение затрат на поддержку: заботы о высокой доступности, резервном копировании перекладываем на инженеров облака с прописанным SLA

Cloud Ready

1. **Ускорение разработки:** Развернуть cloud management ресурсы быстро и малозатратно. Бонус - нетарифицируемый лимит (например, 100K запросов в Yandex Queue)

2. **Снижение затрат на поддержку:** заботы о высокой доступности, резервном копировании перекладываем на инженеров облака с прописанным SLA

3. **Pay-as-you-go:** снижаем затраты на инфраструктуру, платим только за использование ресурсов в момент сканирования

Cloud Ready

Yandex Queue

Построение очередей сканирования и парсинга

Позволяет гибко масштабировать количество подключенных сканеров, путем подписки на очередь и механизму exactly once

Yandex Backet

Хранение сырых и нормализованных результатов

Дешевое распределенное хранилище с доступом по Rest API

Yandex MDB

Храним историю сканирования, состав k8s кластеров

Yandex VM

Запускаем сканеры, возможность автоскейла через API облака

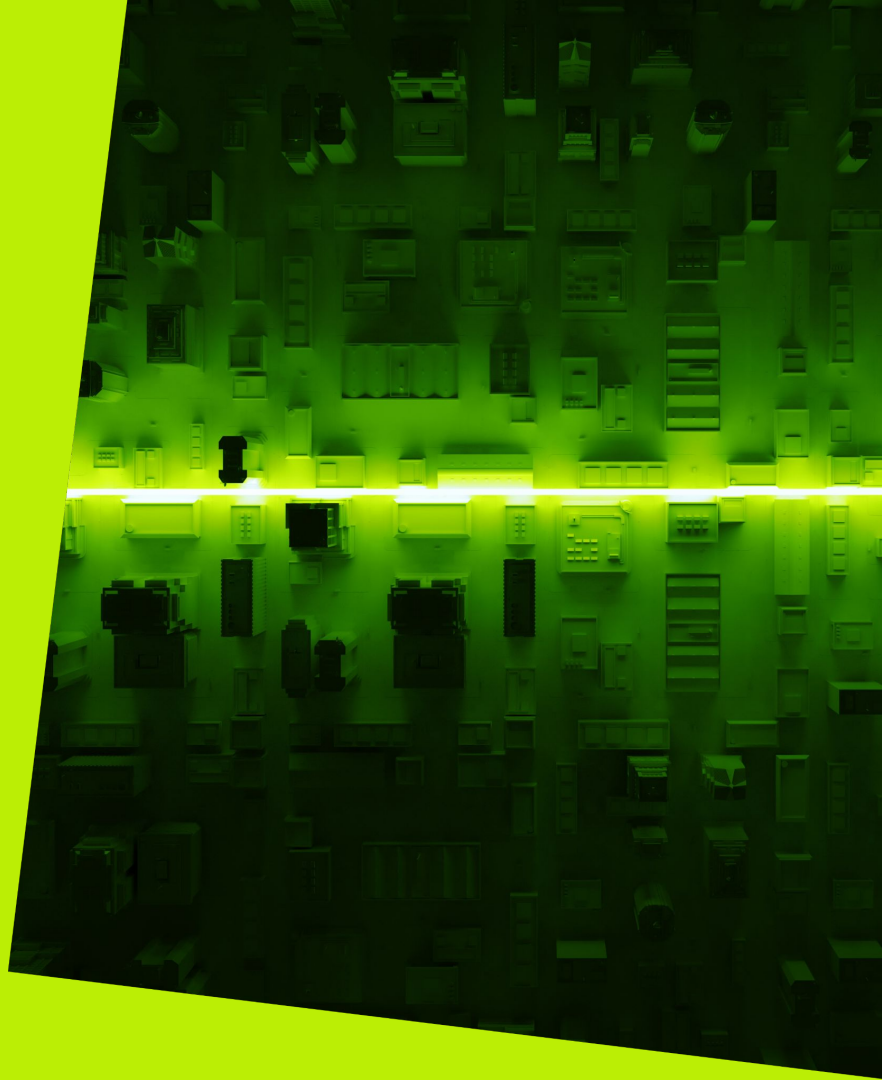
API позволяет включать ноды сканера при появлении заданий в очереди и выключать при отсутствии, тем самым экономя бюджет

Yandex Serverless Containers

Может получать задание из очереди, запускает контейнеры (автоскейл+лимит), после работы при пустой очереди завершается.

Ограничение на время работы контейнера 10 минут, не используем

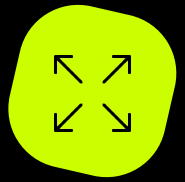
Демо



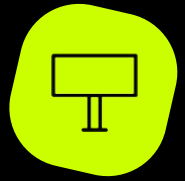
latech

Что получили на выходе?

Сервис, который решает поставленные нами задачи



Возможность масштабирования и подключения новых движков сканирования



Возможность отдачи результатов в разные визуализаторы

Инструмент, который позволит количественно оценивать уязвимости для процесса их управления

Спасибо!

://code
the
lifestyle

Контакты

Игорь Вербицкий

Руководитель отдела ИБ

@i_verbitsky

igor.verbitsky@lamoda.ru

Саша Рахманный

Старший разработчик систем информационной безопасности

@arahmanny

aleksandr.rahmanny@lamoda.ru

latech

Q&A