Software Engineering Conference Russia

October 2017, St. Petersburg

SECR

Автоматизация тестирования PACS-сервера с помощью DevOps

Ренат Зарипов

ЭЛЕКТРОН®
НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ ПРОИЗВОДСТВЕННАЯ КОМПАНИЯ

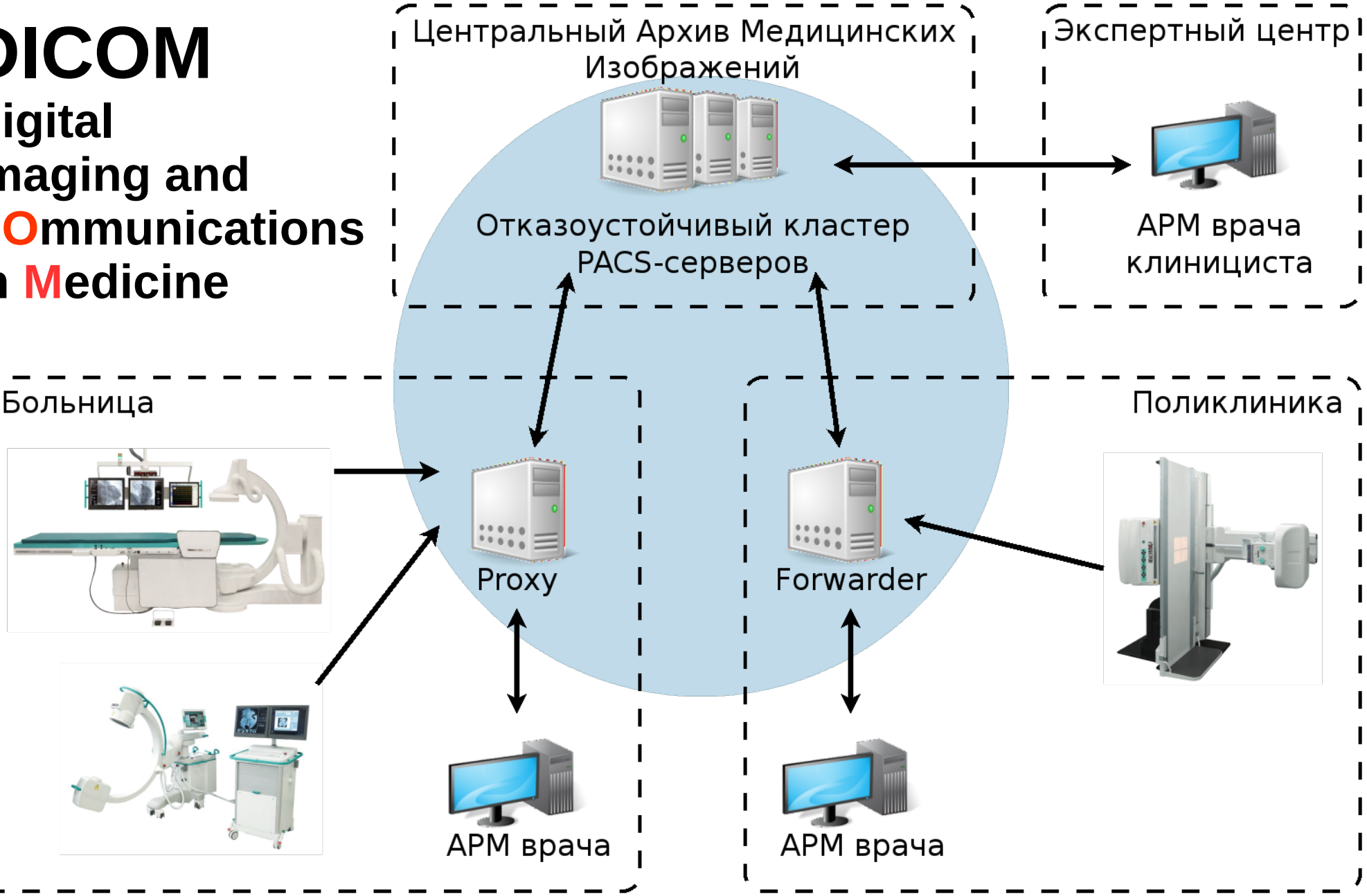
Немного о тестируемом проекте



- PACS сервер (Picture Archiving and Communication System)
  - Прием, хранение и отправка медицинских изображений
  - Выполнение поисковых запросов
- Множество дополнительных функций
- Кросс-платформенный:
  - Linux (x86, amd64, arm)
  - Windows (x86, amd64)
  - MacOS (amd64)
- Интеграция с устройствами и программным обеспечением, в т.ч. сторонними

Роль PACS в ИС медучреждений

DICOM
Digital
Imaging and
COmmunications
in Medicine



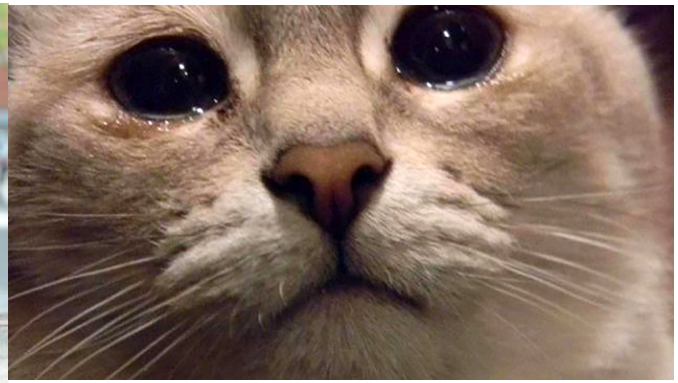
Как мы тестировали вначале

- Ручная установка и настройка ОС, ПО
- Стендов мало
- Не хватало квалификации, например Linux для части тестеров был малознакомой системой

В результате мы пропадали в серверной часами, прежде чем приступить к тестированию

Я думал этот стенд никто не использует и поэтому установил новый билд и снес базу данных...

В прошлый раз я настраивал приложение точно так же, почему же оно теперь не запускается…



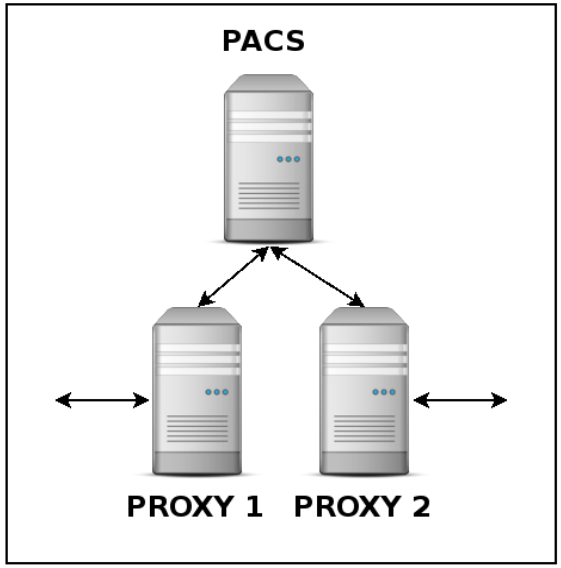
Мне нужно срочно тестировать новый билд на Debian, но не могу настроить сеть, у меня лапки...

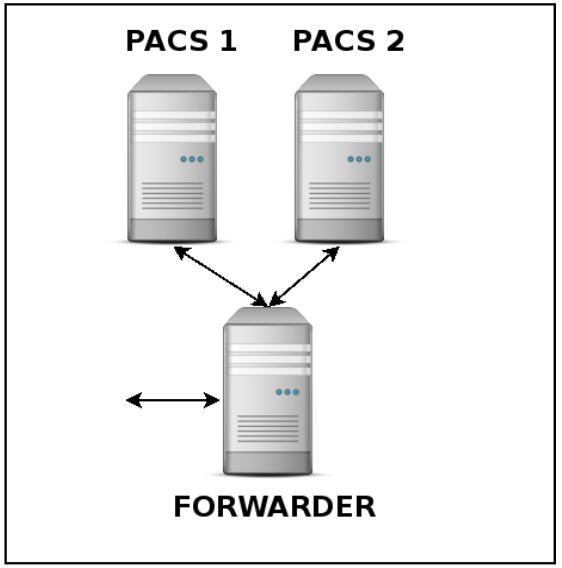
Что у нас уже было

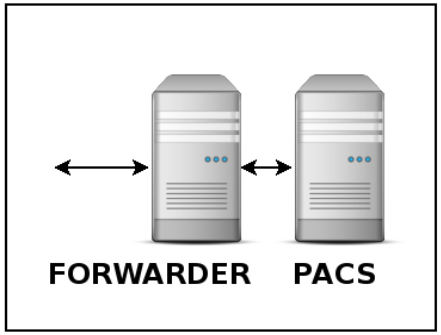
- Достаточно производительные рабочие ПК тестеров с Windows 7 на борту
- ПК для выполнения тестов ~ 15 шт. самых разных конфигураций
- TFS 2013:
  - Хранилище тест-кейсов
  - Хранилище результатов прогона тестов
  - Баг-трекер
  - Team Foundation Version Control - Система управления версиями

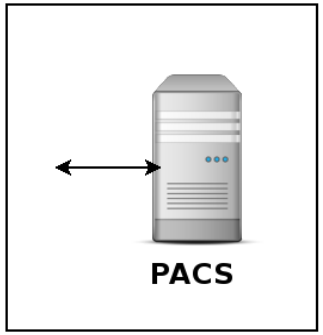
Целевые ОС и конфигурации стендов

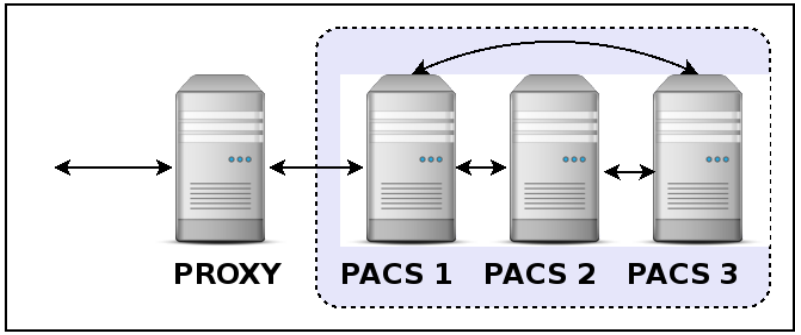
- Debian 8, 9 (amd64)
- Windows 7 (x86)
- Windows 7 (amd64)
- Windows 10 (amd64)
- MacOS (планируется)
- Linux (arm,планируется)

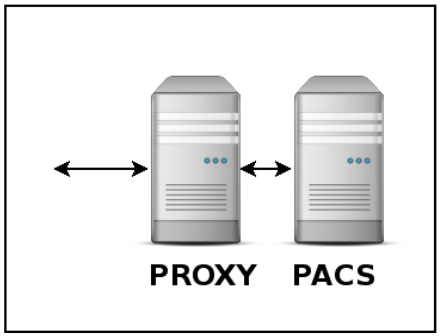


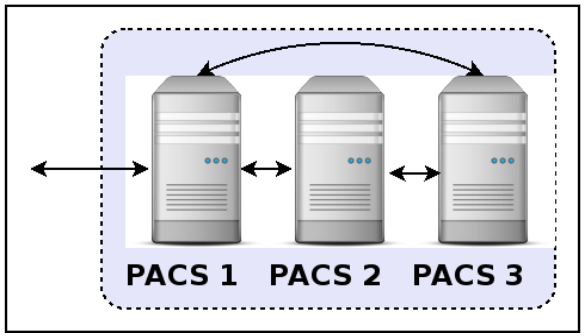


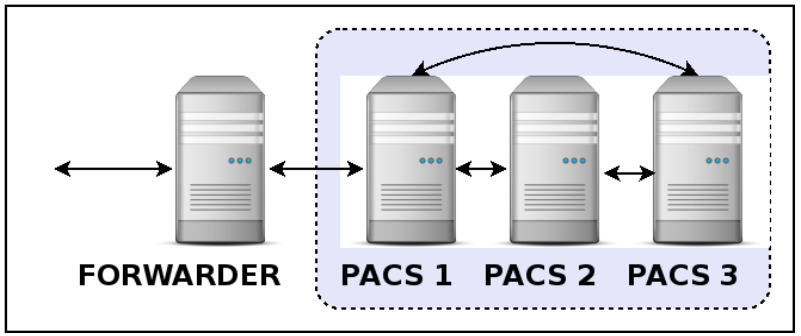












Попытка сделать ситуацию лучше

- Сохранение и повторное использование успешных конфигураций
- Инструкции по установке и настройке
- Частичная автоматизация установки ОС и ПО

Хорошая попытка, но нет:

- Функциональность менялась быстрее, чем мы писали инструкции
- Конфигурационные файлы не покрывали все типы стендов
- Стендов требовалось все больше

Мы проанализировали и поняли, что нам нужно

- Автоматизировать рутинные тесты
- Автоматизировать создание тестовых стендов:
  - Гибкое управление настройками ПО и ОС на стендах
  - Запуск виртуальных тестовых стендов
- Интеграция с TFS

Автоматизация рутинных тестов

Для написания тестов был выбран Python:

- Простой язык с высокой скоростью разработки
- Используем unittest
  - Поставляется вместе с Python
  - Управление тест-кейсами
  - Тест-раннер
  - Фикстуры
- Генерация отчетов через unittest-xml-reporting

Гибкое управление настройками ПО и ОС на стендах

В качестве системы управления конфигурациями и координации работы серверов был выбран **SaltStack**

Плюсы для нас:

- Написан на Python
- Наличие API
- Легкая разработка новых модулей

Управление сервисами

Инсталляция ПО

Параметризация конфигурационных файлов

Настройка сети

Загрузка\сохранение артефактов

Анализ логов

Запуск скриптов

Запуск виртуальных тестовых стендов

Выбрали Vagrant + VirtualBox + Packer

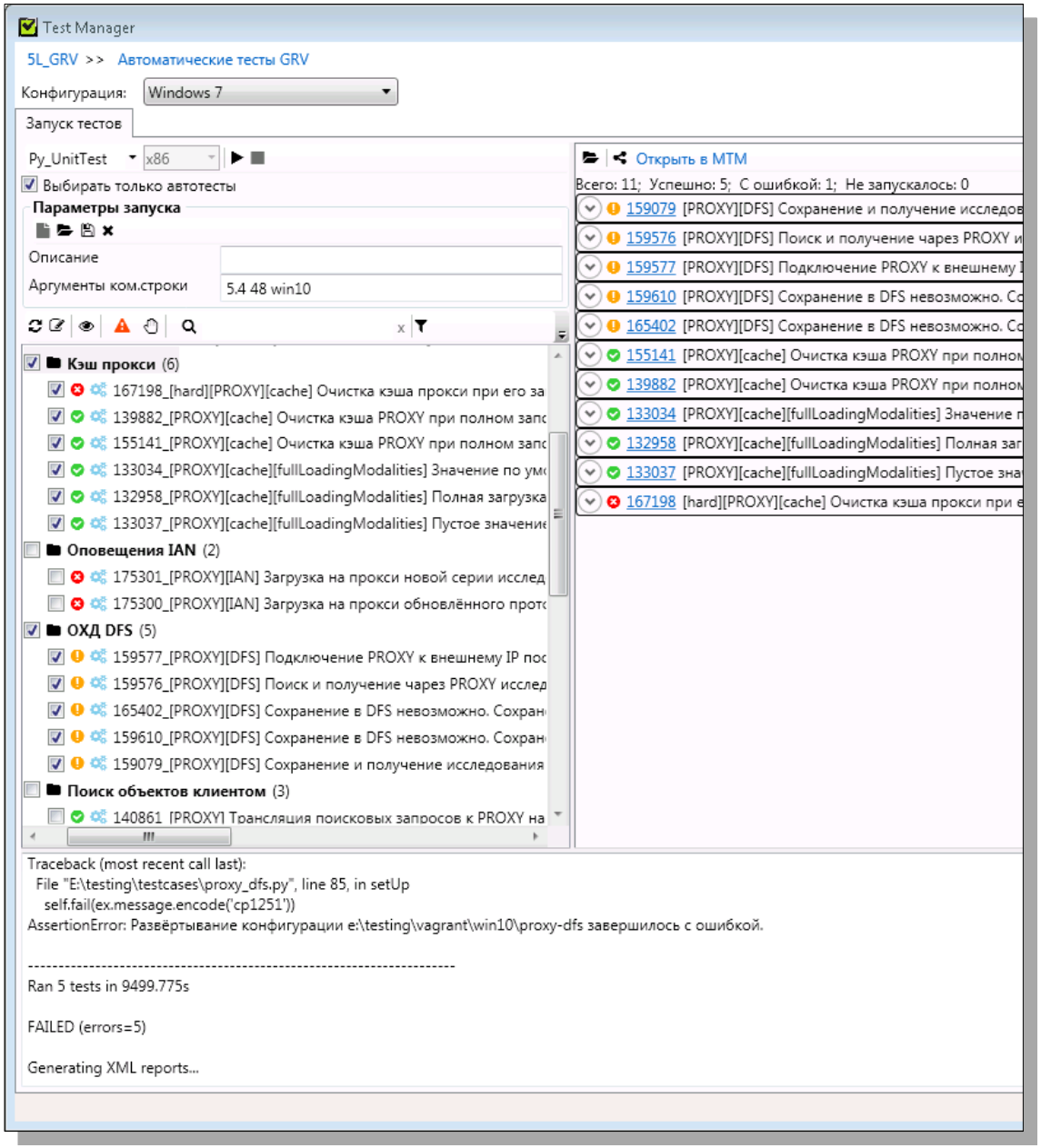
Почему не выбрали Docker:

- Docker на Windows 7... 
- Windows контейнер в Docker... 
- MacOS контейнер в Docker... 

Интеграция с TFS

Advanced Test Manager (ATM)

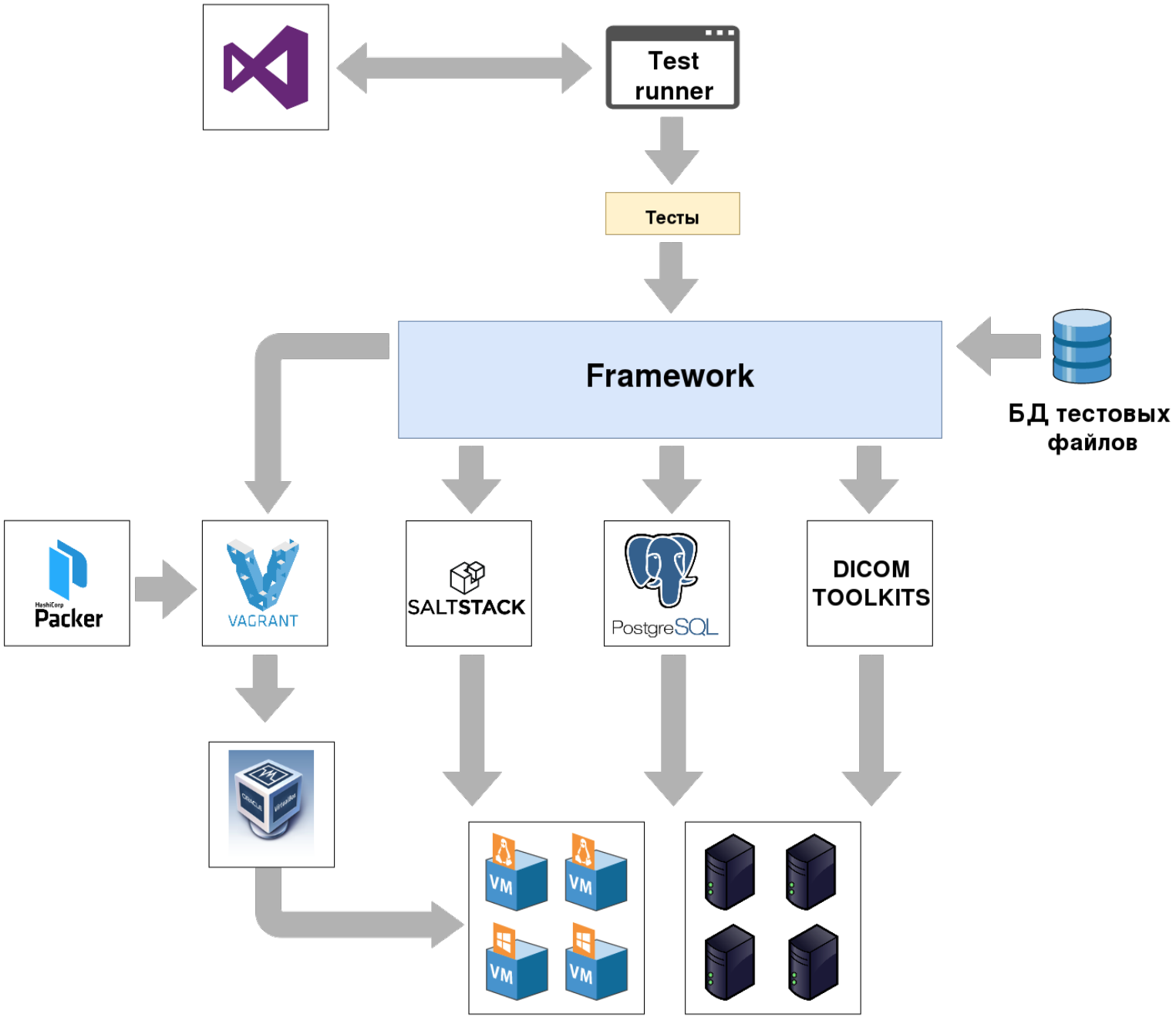
- Привязка автоматического теста к существующему тест-кейсу в TFS
- Создание тест-кейсов в TFS на основе автоматических тестов
- Прогон выбранных тест-кейсов
- Публикация результатов прогона тест-кейсов
- Поддерживаемые форматы автоматических тестов:
  - MS UnitTest
  - MS CodedUITest
  - Python unittest
  - QT unittest
  - NUnit



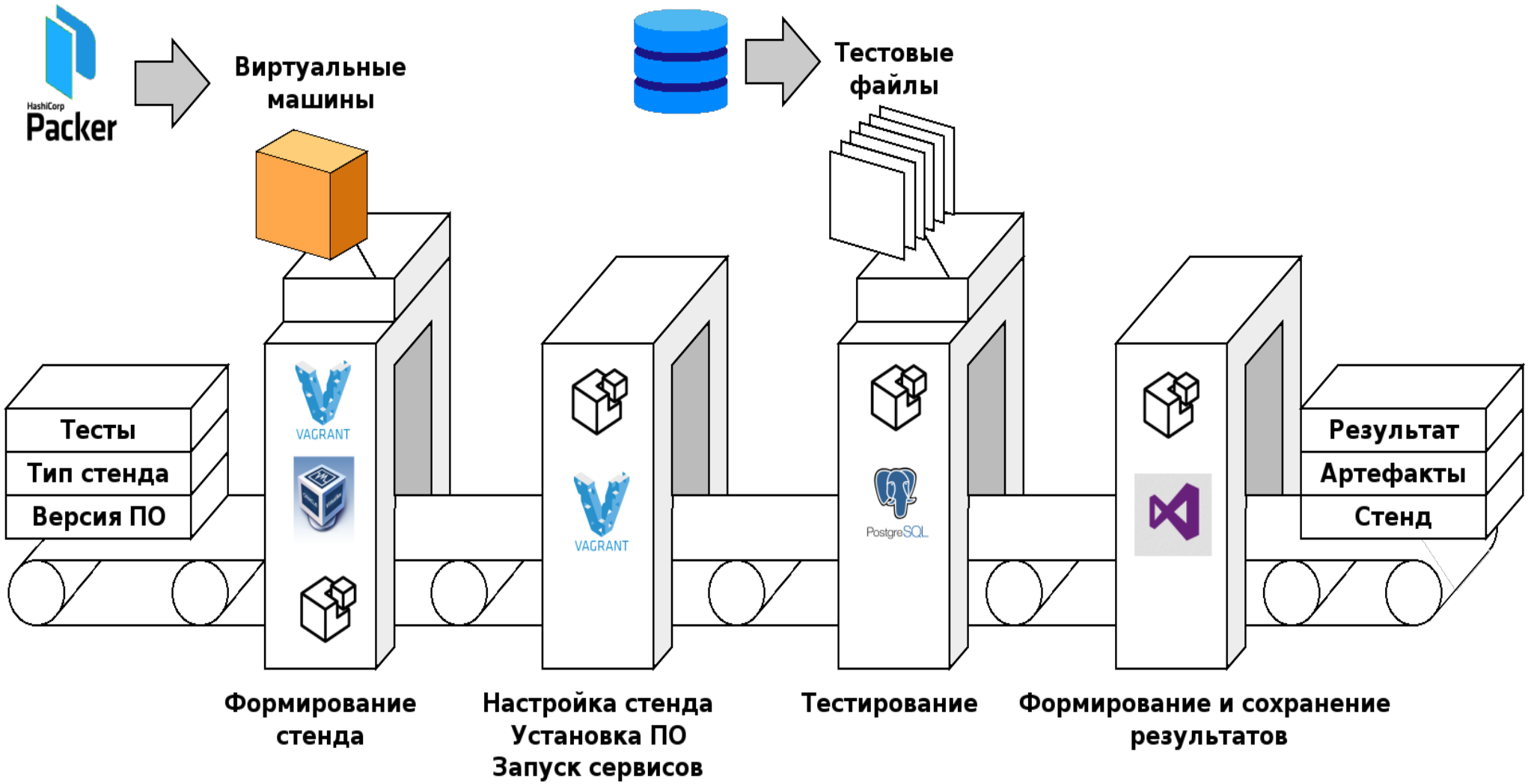
Что еще используем для тестирования?

- Поддержка нескольких DICOM тулкитов
  - dcmtk
  - dcm4che
  - pynetdicom
  - gdcm
- БД тестовых файлов
  - Более 480.000 (~3 Тб) изображений разных типов, протоколов, презентаций
  - Централизованное хранение и управление тестовыми данными

Что в итоге получилось



Как выполняются тесты



Сравнение временных затрат на тестирование

Конфигурация с одним сервером, smoke test

	Вручную	Автотесты ВМ	Автотесты ПК
Подготовка стенда	15-35 мин.	1-10 мин.	2-8 мин.
Выполнение тестов	45 мин.	25 мин.	20 мин.
Сохранение артефактов теста	5 мин.	1-3 мин.	1-3 мин.
Суммарное время	От 65 мин. До 100 мин.	От 27 мин. (**-59%**) До 38 мин. (**-62%**)	От 23 мин. (**-65%**) До 31 мин. (**-69%**)

Конфигурация с тремя серверами, smoke test

	Вручную	Автотесты ВМ	Автотесты ПК
Подготовка стенда	45 мин. - 1 ч.	1-10 мин.	2-8 мин.
Выполнение тестов	3 ч.	1 ч. 20 мин.	1 ч. 10 мин.
Сохранение артефактов теста	10 мин.	1-3 мин.	1-3 мин.
Суммарное время	От 225 мин. До 250 мин.	От 82 мин. (**-64%**) До 93 мин. (**-63%**)	От 63 мин. (**-72%**) До 81 мин. (**-78%**)

Какие еще выгоды от использования DevOps мы получили?

Перевели в код и храним в СКВ:

- Тестовое окружение
- Тесты (в т.ч. описание и шаги, интегрированные в TFS)
- Документация

Виртуальные стенды используются для инкрементного, регрессионного, исследовательского тестирования.

Стенды из реальных ПК для нагрузочных тестов и тестов с большими объемами данных

Спасибо за внимание!

Присылайте свои вопросы, делитесь мнением:


r.r.zaripov@gmail.com


rrzaripov


@rrzaripov