Никита Жевелков
PT ISIM Tech Lead

Проектирование, разработка и поддержка тестов обновлений для продуктов с монолитной архитектурой

Никита Жевелков

За что отвечаю

Технологическая составляющая обеспечения качества продукта

Эффективая имплементация автоматизации в процесс выпуска релиза

Технологии автоматизации тестирования, инструменты и практики AQA

•01

Про продукт и проблемы проверки обновлений

PT ISIM

Система глубокого анализа технологического трафика.

Обеспечивает поиск следов нарушений информационной безопасности в сетях АСУ ТП

Распределенный монолит

Сервисы с большой связанностью между собой, вплоть до невозможности работы одного без другого

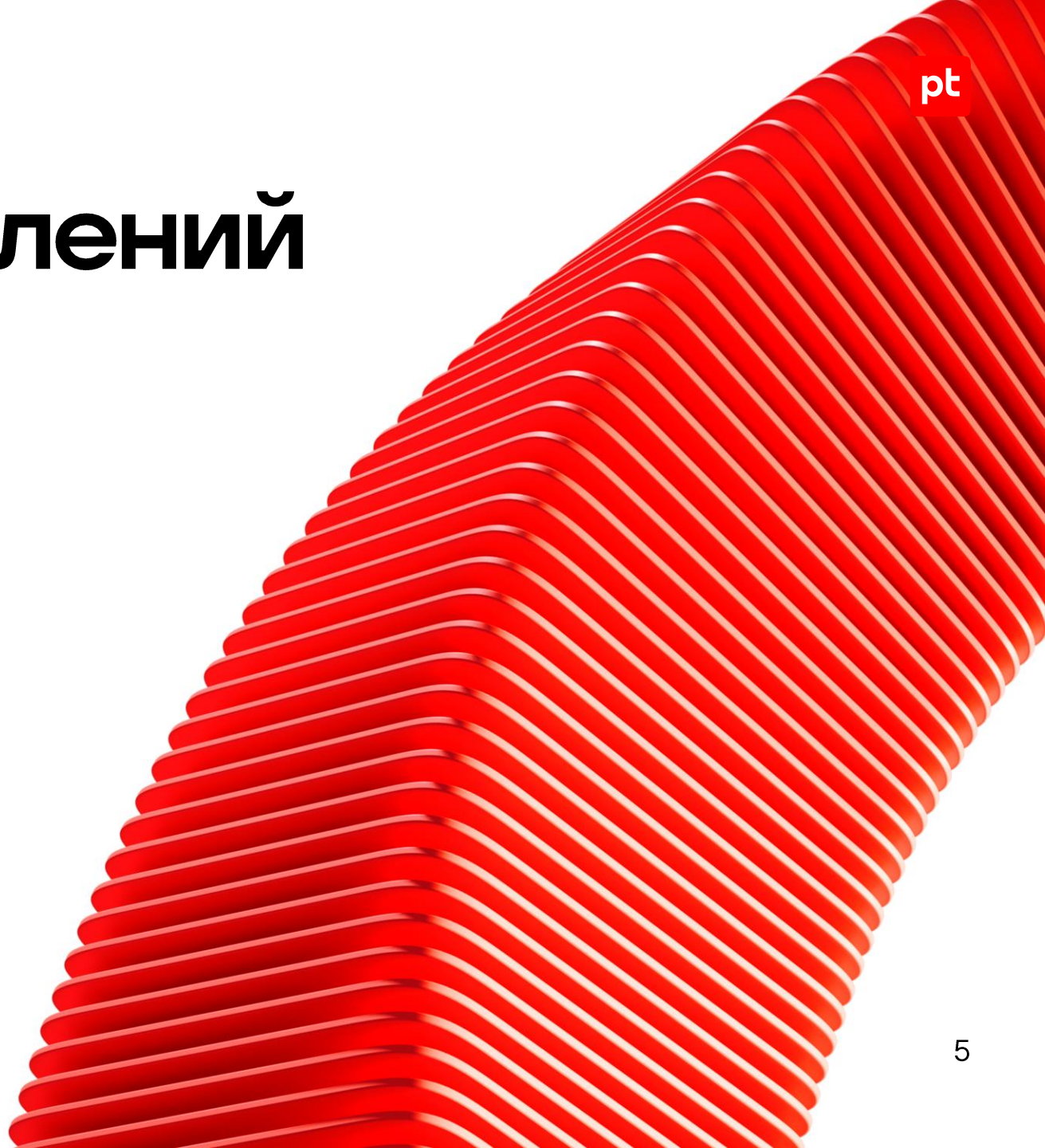
Особенности продукта с точки зрения обновлений

Есть множество различных комбинаций обновлений продукта относительно параметров, таких как версии, поставки, ОС, виды доставок пакета обновления

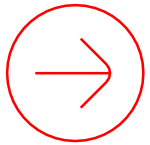
Проверка обновлений

Проверка
инсталлятора

Проверка
обновлений
в рамках
различных тестов



Проверка обновлений инсталлятора



Версии

С чего и на что мы
можем обновиться.

Обновление может быть
- мажорным (4.3.x → 4.4.x)
- минорным
(4.4.x → 4.4.y, y > x)

Поставки

- proview
- netview
- misim
- ovc
- kiosk
- экспертиза

ОС

- debian stretch
- debian buster
- astra linux

Доставка пакета обновлений

- offline
- online

Проблемы обновления

Масштабируемость

Для декартового произведения множеств продуктов, версий, поставок, ОС, доставок

- Количество продуктов = 6
- Количество версий = 4 (3 мажора + 1 минор)
- Количество доставок пакета обновлений = 2 (онлайн, офлайн)
- ОС = 4 (stretch, buster, astra voronezh, astra smolensk)
- Итого = 6 × 4 × 2 × 4 = 192

Миграция данных

Необходимо проверять, что после обновления данные мигрировали корректно

- Проверка целостности данных
- Проверка валидности данных

Проверка работоспособности

Проверка работоспособности: после обновления продукт находится в рабочем состоянии

- Проводим тестирование от простого к сложному
- Отправляем отчет о тестах на почту

Проблемы обновления

Масштабируемость

Для декартового произведения множеств продуктов, версий, поставок, ОС, доставок

- Количество продуктов = 6
- Количество версий = 4 (3 мажора + 1 минор)
- Количество доставок пакета обновлений = 2 (онлайн, офлайн)
- ОС = 4 (stretch, buster, astra voronezh, astra smolensk)
- Итого = 6 × 4 × 2 × 4 = 192

Миграция данных

Необходимо проверять, что после обновления данные мигрировали корректно

- Проверка целостности данных
- Проверка валидности данных

Проверка работоспособности

Проверка работоспособности: после обновления продукт находится в рабочем состоянии

- Проводим тестирование от простого к сложному
- Отправляем отчет о тестах на почту

Проблемы обновления

Масштабируемость

Для декартового произведения множеств продуктов, версий, поставок, ОС, доставок

- Количество продуктов = 6
- Количество версий = 4 (3 мажора + 1 минор)
- Количество доставок пакета обновлений = 2 (онлайн, офлайн)
- ОС = 4 (stretch, buster, astra voronezh, astra smolensk)
- Итого = 6 × 4 × 2 × 4 = 192

Миграция данных

Необходимо проверять, что после обновления данные мигрировали корректно

- Проверка целостности данных
- Проверка валидности данных

Проверка работоспособности

Проверка работоспособности: после обновления продукт находится в рабочем состоянии

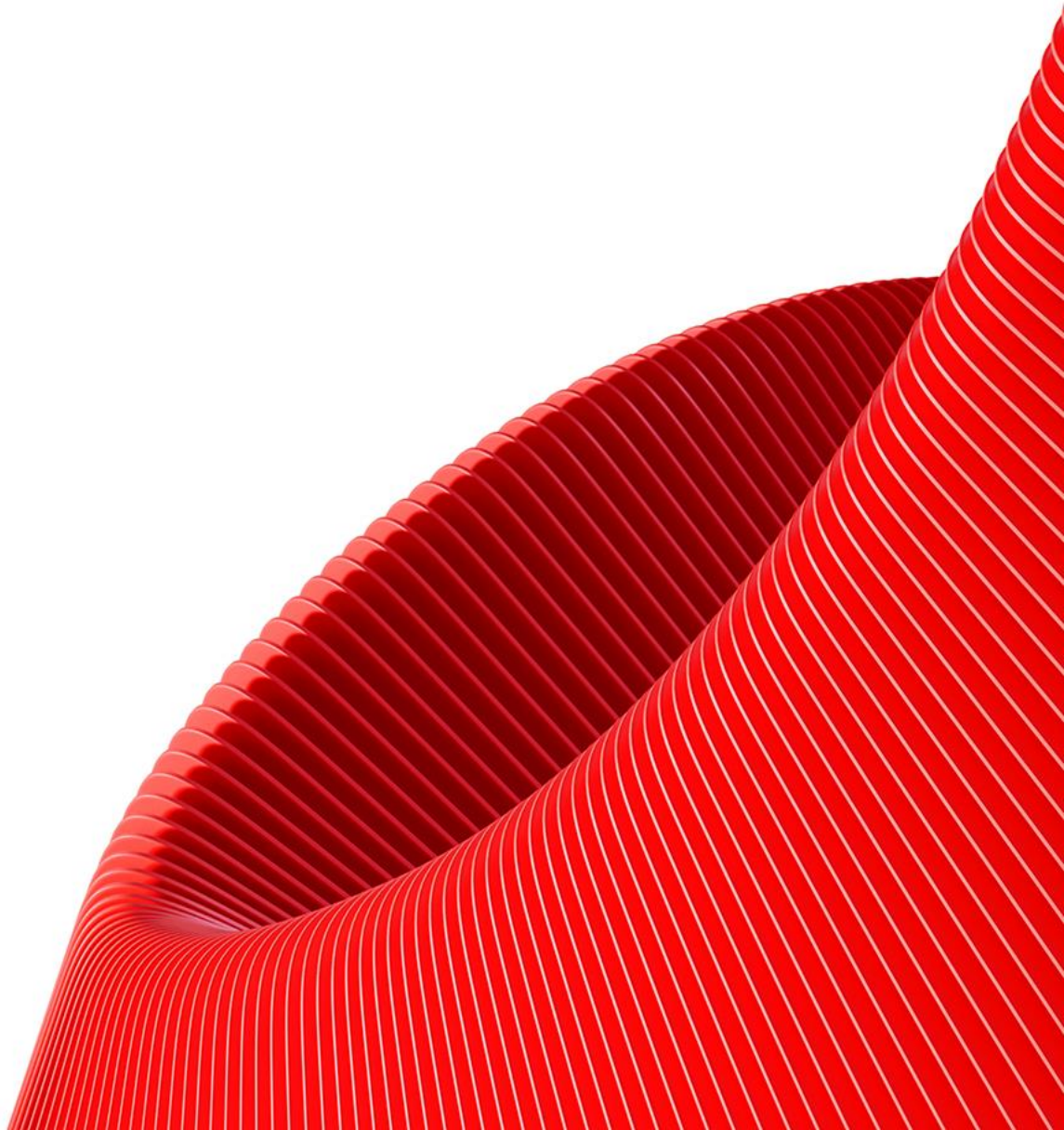
- Проводим тестирование от простого к сложному
- Отправляем отчет о тестах на почту

Проверка работоспособности

- Проверка rc-инсталятора
- Проверка логов сервисов на отсутсвие ошибок после обновления
- Проверка целостности данных
- Проверка UI
- Отрабатывают все пользовательские кейсы в интерфейсе

•02

Тестирование инсталлятора продукта



Узлы View Sensor
(proView или netView)

Узлы Overview Center,
которые подключены
к PT ISIM Overview Center

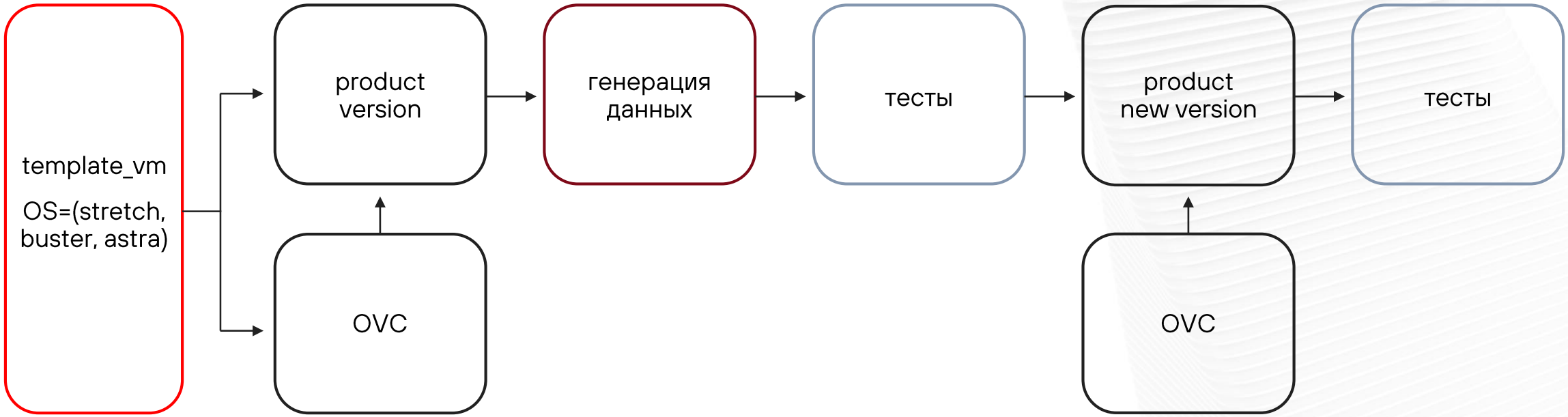
PT ISIM Overview Center

Продукт линейки PT ISIM,
предназначенный для сбора информации
с подключенных к нему узлов

OVC обновляет
подключенные
к нему сенсоры

OVC собирает
информацию
об инцидентах с сенсоров

Схема онлайн-обновления



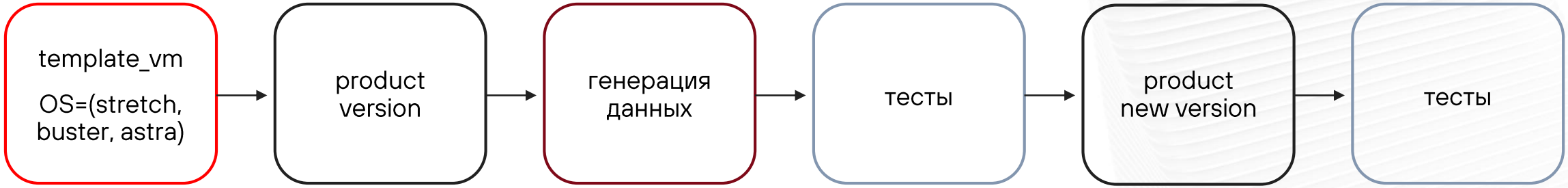
1 Одна ОС

Online-режим

> Подключаем сенсоры к OVC, скачиваем через OVC «новую версию» с GUS на сенсоры. Обновляем сенсоры через OVC

> Обновление считается успешным, если пройдены все этапы тестирования от rc=0 до e2e-тестов

Короткая схема офлайн-обновления

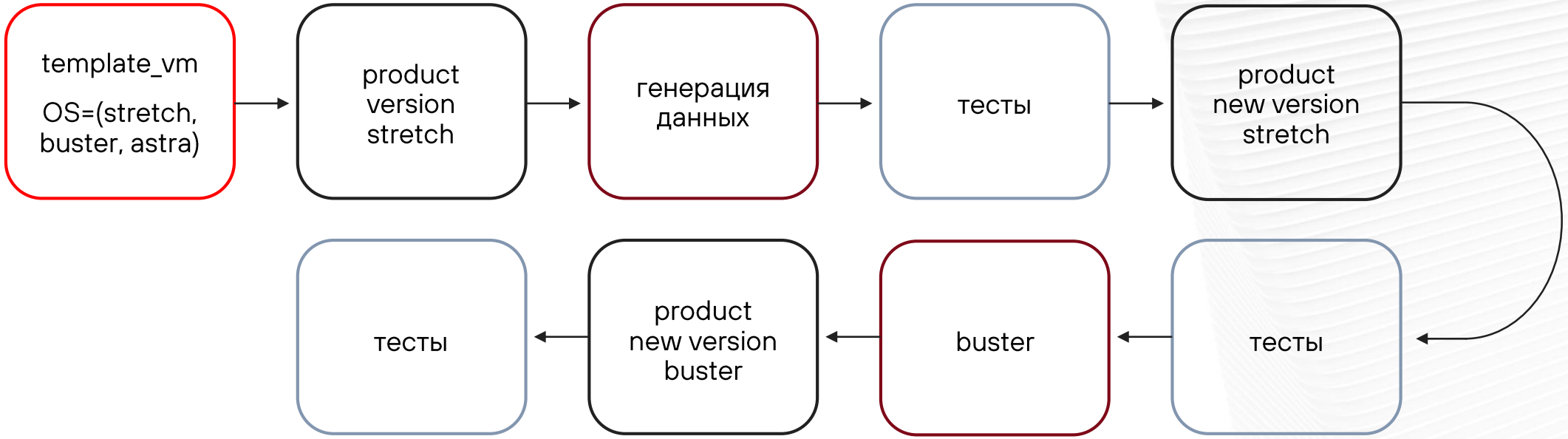


1 Одна ОС

Offline-режим

> Обновление считается успешным, если пройдены все этапы тестирования от rc=0 до e2e-тестов

Схема офлайн-обновления



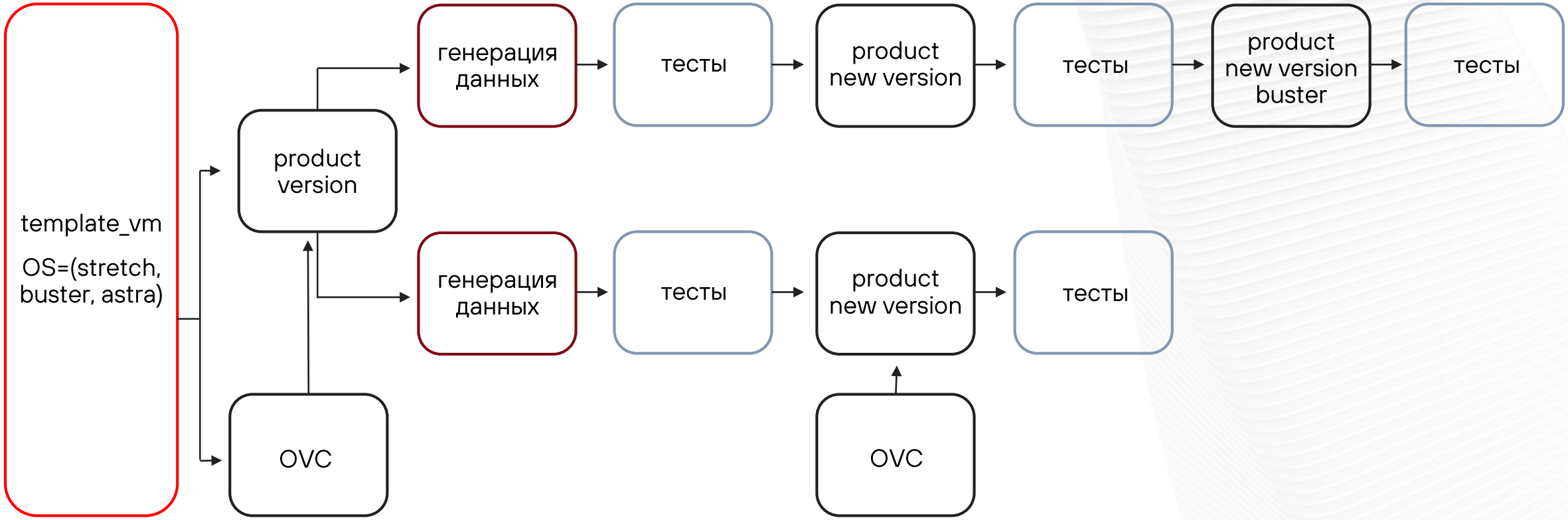
2 Две ОС: stretch → buster

Offline-режим

> Обновляем x.y на z.t.deb9, stretch → buster, z.t.deb9 → z.t.deb10; x >= z, y > t

> Обновление считается успешным, если пройдены все этапы тестирования от rc=0 до e2e-тестов

Полная схема обновлений



> В этом варианте заключены все возможные варианты обновления продукта до «новой версии»

> Таким образом мы приходим к гибкому формату обновлений, которые регулируются конфигами и переменными в рамках CI/CD

Инструменты для тестирования обновлений

Для CI/CD-части проверки обновлений

Нужна система управления репозиториями кода

Наш выбор
GitLab

Аналоги
GitHub, Bitbucket

Нужна система оркестрации для управления виртуальными машинами

Наш выбор
Ansible

Аналоги
Terraform

Инструменты для тестирования обновлений

Для CI/CD-части проверки обновлений

Нужен гипервизор и пакет управления виртуальными машинами

Наш выбор
VMware vSphere

Аналоги
Yandex Cloud, VK Cloud

Нужен язык программирования для расширения функциональности Ansible

Наш выбор
Python

Требования к инструменту для запуска тестов обновлений

Для проверки инсталлятора

- Может по заданной ему конфигурации создавать нужный пайплайн для тестирования обновлений
- Должен быть легко поддерживаемым
- Легко расширяемым
- Нативным для любого пользователя

```
products:
  Package:
    upgrades_from:
      stretch:
        proview:
          versions:
            4.2: link
            4.3: link
      buster:
        netview:
          versions:
            4.3: link
    upgrades_to:
      stretch:
        proview:
          versions:
            4.4: link
      buster:
        netview:
          versions:
            4.4: link
```

Конфиг для управления обновлениями

>

upgrade_config.yml
конфиг, в котором структурированно представлена модель, содержащая в себе иерархию: продукт, поставка, версия (версии)

```
products:
  Package:
    upgrades_from:
      stretch:
        proview:
          versions:
            4.2: link
            4.3: link
      buster:
        netview:
          versions:
            4.3: link
    upgrades_to:
      stretch:
        proview:
          versions:
            4.4: link
      buster:
        netview:
          versions:
            4.4: link
```

Конфиг для управления обновлениями

upgrade_config.yml
Конфиг, в котором структурированно представлена модель, содержащая в себе иерархию: продукт, поставка, версия (версии)

Products
OVC, Package, Rules, misim, Kiosk

```
products:
  Package:
    upgrades_from:
      stretch:
        proview:
          versions:
            4.2: link
            4.3: link
        buster:
          netview:
            versions:
              4.3: link
    upgrades_to:
      stretch:
        proview:
          versions:
            4.4: link
        buster:
          netview:
            versions:
              4.4: link
```

Конфиг для управления обновлениями

upgrade_config.yml
Конфиг, в котором структурированно представлена модель, содержащая в себе иерархию: продукт, поставка, версия (версии)

Products
OVC, Package, Rules, misim, Kiosk

upgrades_from
С чего хотим обновляться

upgrades_to
На что хотим обновляться

```
products:
  Package:
    upgrades_from:
      stretch:
        proview:
          versions:
            4.2: link
            4.3: link
      buster:
        netview:
          versions:
            4.3: link
    upgrades_to:
      stretch:
        proview:
          versions:
            4.4: link
      buster:
        netview:
          versions:
            4.4: link
```

Конфиг для управления обновлениями

upgrade_config.yml
Конфиг, в котором структурированно представлена модель, содержащая в себе иерархию: продукт, поставка, версия (версии)

Products
OVC, Package, Rules, misim, Kiosk

upgrades_from
С чего хотим обновляться

upgrades_to
На что хотим обновляться

OS
stretch, buster, smolensk, voronezh

```
products:
  Package:
    upgrades_from:
      stretch:
        proview:
          versions:
            4.2: link
            4.3: link
      buster:
        netview:
          versions:
            4.3: link
    upgrades_to:
      stretch:
        proview:
          versions:
            4.4: link
      buster:
        netview:
          versions:
            4.4: link
```

Конфиг для управления обновлениями

upgrade_config.yml
Конфиг, в котором структурированно представлена модель, содержащая в себе иерархию: продукт, поставка, версия (версии)

Products
OVC, Package, Rules, misim, Kiosk

upgrades_from
С чего хотим обновляться

upgrades_to
На что хотим обновляться

OS
stretch, buster, smolensk, voronezh

Поставки
proview, netview

```
products:
  Package:
    upgrades_from:
      stretch:
        proview:
          versions:
            4.2: link
            4.3: link
      buster:
        netview:
          versions:
            4.3: link
    upgrades_to:
      stretch:
        proview:
          versions:
            4.4: link
      buster:
        netview:
          versions:
            4.4: link
```

Конфиг для управления обновлениями

upgrade_config.yml
Конфиг, в котором структурированно представлена модель, содержащая в себе иерархию: продукт, поставка, версия (версии)

Products
OVC, Package, Rules, misim, Kiosk

upgrades_from
С чего хотим обновляться

upgrades_to
На что хотим обновляться

OS
stretch, buster, smolensk, voronezh

Поставки
proview, netview

Versions
Версии, которые мы поддерживаем в этом релизе при обновлениях

```
products:
  Package:
    upgrades_from:
      stretch:
        proview:
          versions:
            4.2: link
            4.3: link
      buster:
        netview:
          versions:
            4.3: link
    upgrades_to:
      stretch:
        proview:
          versions:
            4.4: link
      buster:
        netview:
          versions:
            4.4: link
```

Конфиг для управления обновлениями

upgrade_config.yml
Конфиг, в котором структурированно представлена модель, содержащая в себе иерархию: продукт, поставка, версия (версии)

Products
OVC, Package, Rules, misim, Kiosk

upgrades_from
С чего хотим обновляться

upgrades_to
На что хотим обновляться

OS
stretch, buster, smolensk, voronezh

Поставки
proview, netview

Versions
Версии, которые мы поддерживаем в этом релизе при обновлениях

Link
Ссылка в хранилище на артефакт, подходящий под всю иерархию

Структура CI/CD

.gitlab-ci.yml содержит последовательный набор всевозможных этапов пайплайна для прохождения любого варианта тестов обновлений

Также в него импортируются файлы, содержащие декомпозированные наборы этапов прохождения пайплайна и переменные, определенные внутри этих файлов и пришедшие из запуска пайплайна

По умолчанию в тестах обновления берется продукт ISIM поставки proview. Остальные поставки определяются отдельными переменными и задачами внутри структуры файлов CI/CD

.gitlab-ci.yml 29.73 KiB

```
# ------------------------------ INCLUDES
include:
 - 'releases/OVC.yml'
 - 'releases/3_1.yml'
 - 'releases/3_3.yml'
 - 'releases/4_1.yml'
 - 'releases/4_2.yml'
 - 'releases/4_3.yml'
 - 'releases/4_4.yml'
 - 'rules_vars.yml'

# ------------------------------ STAGES --

stages:
  - clone VM
  - install product
  - add VMs to OVC
  - connect OVC to GUS
  - collect and dump data
  - product update
  - dump data
  - compare collected data
  - send report email
  - OS update
  - product update for buster
  - dump data buster
  - compare collected data buster
  - rules update
  - detects tests
  - send report email data buster

# ------------------------------ JOBS ----
```

Структура CI/CD

На примере одной из задач всего пайплайна тестов обновлений рассмотрим структуру файлов и декомпозицию задач

- В файле .gitlab-ci.yml определяем задачу по клонированию машины для проверки обновлений продукта релизной версии 4.3

- Создаем три варианта задачи:
  - → для онлайн-обновления
  - → для офлайн-обновления
  - → для офлайн-обновления поставки OVC

```
75  # ---------------------------- release/4.3 --------------------------
76  clone VM for 4.3 (offline):
77    extends:
78      - .clone_vm_release_4_3_offline
79
80  clone VM for OVC 4.3 (offline):
81    extends:
82      - .clone_ovc_release_4_3_offline
83
84  clone VM for 4.3 (online):
85    extends:
86      - .clone_vm_release_4_3_online
87
88  # ---------------------------- release/4.4 --------------------------
```

Структура CI/CD

Так выглядит структура файла releases/4_3.yml, который мы импортировали в .gitlab-ci.yml

- Определены общие переменные для всех задач, которые присущи релизу 4.3
- Виден еще один уровень вложенности у якорей при декомпозиции задач из .gitlab-ci.yml. Они включают в себя другие якоря из файла vars_and_anchors.yml

4_3.yml 4.18 KiB

```yaml
include:
  - 'vars_and_anchors.yml'

#----------------------------VARS-----------------------
.4-3-offline-variables: &4-3-offline-variables
  VERSION: "4.3"
  OS: $TARGET_OS
  POSTFIX: $COMMON_OFFLINE_POSTFIX

.4-3-online-variables: &4-3-online-variables
  VERSION: "4.3"
  OS: $TARGET_OS
  POSTFIX: $COMMON_ONLINE_POSTFIX

.dump-directory: &dump-directory
  MINOR_OS_DIRECTORY: /root/data_dump_for_4_3_$TARGET_OS

#----------------------- CLONE VM ----------------------
.clone_vm_release_4_3_offline:
  variables:
    <<: *4-3-offline-variables
  extends:
    - .clone_vm
    - .rules_for_offline_common

.clone_ovc_release_4_3_offline:
  variables:
    <<: *4-3-offline-variables
  extends:
    - .clone_vm
    - .rules_for_ovc_offline

.clone_vm_release_4_3_online:
  variables:
    <<: *4-3-online-variables
  extends:
    - .clone_vm
    - .rules_for_online

#----------------------- INSTALL PRODUCT ---------------
```

Структура CI/CD

vars_and_anchors.yml — файл, включающий в себя все возможные переменные, максимально декомпозированные задачи и правила

- Определены правила для включения задачи в пайплайн или исключения из пайплайна
- Определены переменные различного рода, начиная переменными, задающими параметры стенда, заканчивая переменными, которые указывают путь к скрипту запуска
- Определены якоря, с которых начинается «матрешка» вложенности, доходящая до файла .gitlab-ci.yml

```yaml
.clone_vm:
  stage: clone VM
  image: docker-isim.ptsecurity.ru/isim-ansible:release-4-4
  allow_failure: false
  extends: .deploy-servers
  variables:
    <<: *deploy-variables
    <<: *ansible-variables
    <<: *scripts-variables
    <<: *product-version-variables
    <<: *postfix-variables
  script:
    - bash $RUN_SCRIPT_CLONE_VM
  artifacts:
    paths:
      - $CI_PROJECT_DIR/stand_ip
  when: on_success

.install_product:
```

```yaml
.scripts-variables: &scripts-variables
    RUN_SCRIPT_CLONE_VM: "deploy_scripts/clone_vm.sh"
    RUN_SCRIPT_INSTALL_PRODUCT: "deploy_scripts/install_product.sh"
    RUN_SCRIPT_ADD_SENSORS_TO_OVC: "deploy_scripts/add_sensors_to_ovc.sh"
    RUN_SCRIPT_ENV_PREPARE: "deploy_scripts/env_prepare.sh"
    RUN_SCRIPT_ENV_DUMP: "deploy_scripts/env_dump.sh"
    RUN_SCRIPT_COMPARE_DATA: "deploy_scripts/compare_data.sh"
    RUN_SCRIPT_ONLINE_UPDATE: "deploy_scripts/online_update.sh"
    RUN_SCRIPT_OVC_TO_GUS: "deploy_scripts/ovc_to_gus.sh"
    RUN_SCRIPT_OS_UPGRADE: "deploy_scripts/os_upgrade.sh"
    RUN_RULES_UPGRADE: "deploy_scripts/rules_upgrade.sh"
    RUN_SCRIPT_SEND_REPORT: "deploy_scripts/send_report.sh"
```

```yaml
# ---------------------- RULES -----------------------------------
.rules_for_offline_common:
  rules:
    - if: $CI_PIPELINE_SOURCE == "web" && ($MAJOR_VERSION == $VERSION || $FULL == "true") && ($OFFLINE == "true") && $PRODUCT != "OVC"

.rules_for_offline_package:
  rules:
    - if: $CI_PIPELINE_SOURCE == "web" && ($MAJOR_VERSION == $VERSION || $FULL == "true") && $PRODUCT == "Package" && ($OFFLINE == "true")

.rules_for_offline_package_update_on_buster:
```

Запуск Ansible-плейбука

```
ansible-playbook $playbooks_dir/test-upgrade/clone_vm.playbook \
    -e "vmware_guest_vm_name=$vm_name" \
    -e "vm_name_to_clone=$vm_name_to_clone" \
    -e "ansible_user=$vm_user" \
    -e "ansible_password=$vm_password" \
    -e "domain_name=$vm_domain_name" \
    -e "vm_owner=$vm_owner" \
    -e "vm_ttl_action=$vm_ttl_action" \
    -e "vm_ttl=$vm_ttl" \
    -e "{\"resource_pool\": \"$resource_pool\"}" \
    -e "cluster=$cluster" \
    -e "{\"upload_licence\": $upload_licence}" \
    -e "{\"disable_firewall\": $disable_firewall}" \
    -e "{\"disable_training_mode\": $disable_training_mode}" \
    -e "pipeline_id=$CI_PIPELINE_ID" -vvv 2>&1 | tee stdout.log
```

Статья «Пособие по Ansible» на Хабре

>

Все шелл-скрипты, которые определены в директории deploy_scripts/ служат для вызова Ansible-плейбука и передачи в этот плейбук нужных переменных для его выполнения

>

Запуск плейбуков происходит в докер-контейнере Ansible, образ которого мы собираем в отдельном репозитории

>

Все манипуляции с виртуальными машинами происходят на VMware vSphere через API в отдельной роли Ansible

Итоги тестирования инсталлятора

Время и выводы

40 МИНУТ min

Минимальное время

90 МИНУТ max

Максимальное время

Все виды проверок
запускаются параллельно

Структура CI/CD сделана так,
чтобы можно было легко масштабировать
конфигурации проверок инсталлятора

•03

Тестирование продукта после обновления

Тестирование продукта после обновления

Виды проверок и тестов для определения качества продукта после обновления

Проверка успешного завершения инсталлятора

Проверка, что rc=0 и что в логах инсталятора нет ошибок

Проверка логов сервисов на отсутсвие ошибок после обновления

Проводим bvt-тестирование

Тестирование продукта после обновления

Виды проверок и тестов для определения качества продукта после обновления

Проверка миграции данных

Написали фреймворк для сравнения дампов данных со стенда

Проверка UI

Проводим smoke-тестирование

Тестирование продукта после обновления

Виды проверок и тестов для определения качества продукта после обновления

Отрабатывают все пользовательские кейсы в интерфейсе

Проводим e2e-тестирование

Проверка обновлений экспертизы

Тесты экспертизы с помощью написанного нами для этого фреймворка

Проверка миграции данных

- От версии к версии продукта данные могут менять свой формат, наполнение
- Важно проверять, что данные не изменились и не потерялись во время обновлений, так как продукт следит за активностью в сети, копит и хранит важные данные для детектирования угроз
- Потеря таких данных критична и несет за собой репутационные риски для компании

Проверка миграции данных

- Анализ трафика сетей АСУ ТП. Поиск следов нарушений ИБ и кибератак
- Анализаторы трафика, которые слушают сетевой интерфейс и записывают результаты разбора трафика

Проверка миграции данных

- Сенсор мы наполняем данными, пуская трафик через сетевой интерфейс
- Трафик готовится заранее с помощью библиотеки PCAP
- Эмулируем различную пользовательскую активность на сенсоре
- Делаем слепок данных на сенсоре, используя API, так как API меняется намного реже, чем БД
- REST API — формат данных JSON

Проверка миграции данных

- Перед началом обновления наполнить сенсор данными и сделать их дамп
- После каждого пройденного обновления в рамках одного пайплайна снова делаем дамп данных с сенсора
- После этого нужно сравнить их целостность с дампом, полученным на предыдущем этапе в рамках одного пайплайна

Проверка миграции данных

- Data mapping
- Целостность данных для всех возможных записей выборки
- Никакие данные не должны дублироваться во время миграции

Сравнение дампов данных


```
{
    "x": null,
    "name": "#10 Fujitsu",
    "interfaces": [
     {
      "mac": "90:1B:0E:A2:8F:E5",
      "vlan": [],
      "order": 1,
      "id": "16",
      "last_activity": "2024-03-06T09:05:07.938Z",
      "ip": null,
      "interface_vendor": "Fujitsu",
      "description": null
     }
    ],
    "group_id": null,
    "os": null,
    "group_name": null,
    "description": null,
    "importance": 0,
    "has_connected_devices": false,
    "id": "10",
    "y": null,
    "vendor": "Fujitsu",
    "incidents_relation": "source",
    "last_activity": "2024-03-06T09:05:07.938Z",
    "exclusions_counter": 0,
    "authorized": false,
    "host_type": "unknown",
    "violations_counter": 0,
    "violations": []
}
```

```
{
    "x": null,
    "host name": "#10 Fujitsu ",
    "interfaces": {
        "new":
     {
      "mac": "90:1B:0E:A2:8F:E5",
      "vlan": [],
      "order": 1,
      "id": 16,
      "last_activity": "2024-03-06T09:05:07.938Z",
      "ip": null,
      "interface_vendor": "Fujitsu",
      "description": null
     }
    },
    "group_id": null,
    "OS": null,
    "group_name": null,
    "importance": 0,
    "has_connected_devices": false,
    "id": "10",
    "y": null,
    "vendor": "Fujitsu",
    "incidents_relation": "source",
    "last_activity": "2024-03-06T09:05:07.938Z",
    "exclusions_counter": 0,
    "authorized": false,
    "host_type": "unknown",
    "description": null,
}
```

Сравнение дампов данных


```
{
    "x": null,
    "name": "#10 Fujitsu",
    "interfaces": [
      {
        "mac": "90:1B:0E:A2:8F:E5",
        "vlan": [],
        "order": 1,
        "id": "16",
        "last_activity": "2024-03-06T09:05:07.938Z",
        "ip": null,
        "interface_vendor": "Fujitsu",
        "description": null
      }
    ],
    "group_id": null,
    "os": null,
    "group_name": null,
    "description": null,
    "importance": 0,
    "has_connected_devices": false,
    "id": "10",
    "y": null,
    "vendor": "Fujitsu",
    "incidents_relation": "source",
    "last_activity": "2024-03-06T09:05:07.938Z",
    "exclusions_counter": 0,
    "authorized": false,
    "host_type": "unknown",
    "violations_counter": 0,
    "violations": []
}
```

```
{
    "x": null,
    "host name": "#10 Fujitsu ",
    "interfaces": [
      {
        "mac": "90:1B:0E:A2:8F:E5",
        "vlan": [],
        "order": 1,
        "id": 16,
        "last_activity": "2024-03-06T09:05:07.938Z",
        "ip": null,
        "interface_vendor": "Fujitsu",
        "description": null
      }
    ],
    "group_id": null,
    "OS": null,
    "group_name": null,
    "importance": 0,
    "has_connected_devices": false,
    "id": "10",
    "y": null,
    "vendor": "Fujitsu",
    "incidents_relation": "source",
    "last_activity": "2024-03-06T09:05:07.938Z",
    "exclusions_counter": 0,
    "authorized": false,
    "host_type": "unknown",
    "description": null,
}
```

Сравнение дампов данных


```
{
    "x": null,
    "name": "#10 Fujitsu",
    "interfaces": [
      {
        "mac": "90:1B:0E:A2:8F:E5",
        "vlan": [],
        "order": 1,
        "id": "16",
        "last_activity": "2024-03-06T09:05:07.938Z",
        "ip": null,
        "interface_vendor": "Fujitsu",
        "description": null
      }
    ],
    "group_id": null,
    "os": null,
    "group_name": null,
    "description": null,
    "importance": 0,
    "has_connected_devices": false,
    "id": "10",
    "y": null,
    "vendor": "Fujitsu",
    "incidents_relation": "source",
    "last_activity": "2024-03-06T09:05:07.938Z",
    "exclusions_counter": 0,
    "authorized": false,
    "host_type": "unknown",
    "violations_counter": 0,
    "violations": []
}
```

```
{
    "x": null,
    "hostname": "#10 Fujitsu",
    "interfaces": {
        "new":
      {
        "mac": "90:1B:0E:A2:8F:E5",
        "vlan": [],
        "order": 1,
        "id": 16,
        "last_activity": "2024-03-06T09:05:07.938Z",
        "ip": null,
        "interface_vendor": "Fujitsu",
        "description": null
      }
    },
    "group_id": null,
    "OS": null,
    "group_name": null,
    "importance": 0,
    "has_connected_devices": false,
    "id": "10",
    "y": null,
    "vendor": "Fujitsu",
    "incidents_relation": "source",
    "last_activity": "2024-03-06T09:05:07.938Z",
    "exclusions_counter": 0,
    "authorized": false,
    "host_type": "unknown",
    "description": null,
}
```

Сравнение дампов данных


```
{
    "x": null,
    "name": "#10 Fujitsu",
    "interfaces": [
      {
        "mac": "90:1B:0E:A2:8F:E5",
        "vlan": [],
        "order": 1,
        "id": "16",
        "last_activity": "2024-03-06T09:05:07.938Z",
        "ip": null,
        "interface_vendor": "Fujitsu",
        "description": null
      }
    ],
    "group_id": null,
    "os": null,
    "group_name": null,
    "description": null,
    "importance": 0,
    "has_connected_devices": false,
    "id": "10",
    "y": null,
    "vendor": "Fujitsu",
    "incidents_relation": "source",
    "last_activity": "2024-03-06T09:05:07.938Z",
    "exclusions_counter": 0,
    "authorized": false,
    "host_type": "unknown",
    "violations_counter": 0,
    "violations": []
}
```

Сравнение дампов данных


```
{
    "x": null,
    "name": "#10 Fujitsu",
    "interfaces": [
      {
        "mac": "90:1B:0E:A2:8F:E5",
        "vlan": [],
        "order": 1,
        "id": "16",
        "last_activity": "2024-03-06T09:05:07.938Z",
        "ip": null,
        "interface_vendor": "Fujitsu",
        "description": null
      }
    ],
    "group_id": null,
    "os": null,
    "group_name": null,
    "description": null,
    "importance": 0,
    "has_connected_devices": false,
    "id": "10",
    "y": null,
    "vendor": "Fujitsu",
    "incidents_relation": "source",
    "last_activity": "2024-03-06T09:05:07.938Z",
    "exclusions_counter": 0,
    "authorized": false,
    "host_type": "unknown",
    "violations_counter": 0,
    "violations": []
}
```

```
{
    "x": null,
    "hostname": "#10 Fujitsu",
    "interfaces": [
      {
        "mac": "90:1B:0E:A2:8F:E5",
        "vlan": [],
        "order": 1,
        "id": 16,
        "last_activity": "2024-03-06T09:05:07.938Z",
        "ip": null,
        "interface_vendor": "Fujitsu",
        "description": null
      }
    ],
    "group_id": null,
    "OS": null,
    "group_name": null,
    "importance": 0,
    "has_connected_devices": false,
    "id": "10",
    "y": null,
    "vendor": "Fujitsu",
    "incidents_relation": "source",
    "last_activity": "2024-03-06T09:05:07.938Z",
    "exclusions_counter": 0,
    "authorized": false,
    "host_type": "unknown",
    "description": null,
}
```

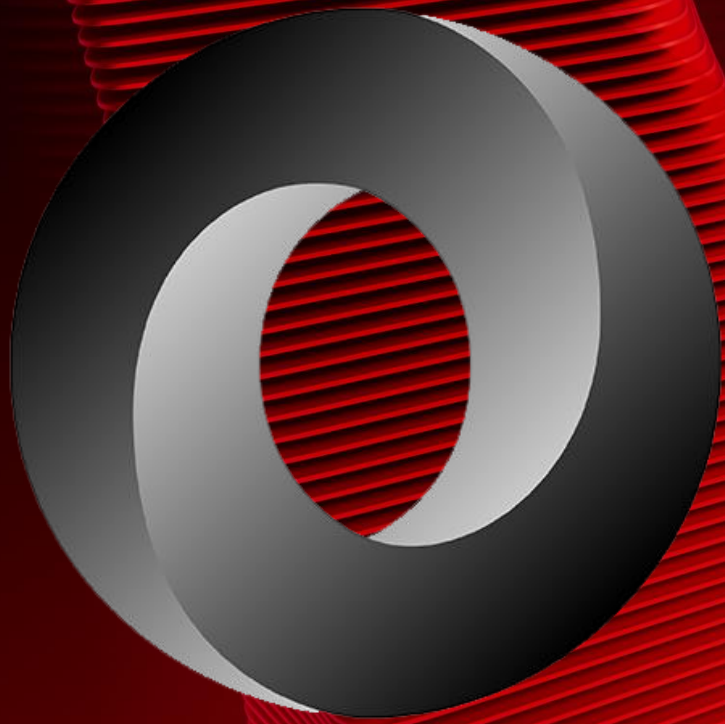
Сравнение дампов данных


```
{
    "x": null,
    "name": "#10 Fujitsu",
    "interfaces": [
      {
        "mac": "90:1B:0E:A2:8F:E5",
        "vlan": [],
        "order": 1,
        "id": "16",
        "last_activity": "2024-03-06T09:05:07.938Z",
        "ip": null,
        "interface_vendor": "Fujitsu",
        "description": null
      }
    ],
    "group_id": null,
    "os": null,
    "group_name": null,
    "description": null,
    "importance": 0,
    "has_connected_devices": false,
    "id": "10",
    "y": null,
    "vendor": "Fujitsu",
    "incidents_relation": "source",
    "last_activity": "2024-03-06T09:05:07.938Z",
    "exclusions_counter": 0,
    "authorized": false,
    "host_type": "unknown",
    "violations_counter": 0,
    "violations": []
}
```

```
{
    "x": null,
    "hostname": "#10 Fujitsu",
    "interfaces": [
      {
        "mac": "90:1B:0E:A2:8F:E5",
        "vlan": [],
        "order": 1,
        "id": 16,
        "last_activity": "2024-03-06T09:05:07.938Z",
        "ip": null,
        "interface_vendor": "Fujitsu",
        "description": null
      }
    ],
    "group_id": null,
    "OS": null,
    "group_name": null,
    "importance": 0,
    "has_connected_devices": false,
    "id": "10",
    "y": null,
    "vendor": "Fujitsu",
    "incidents_relation": "source",
    "last_activity": "2024-03-06T09:05:07.938Z",
    "exclusions_counter": 0,
    "authorized": false,
    "host_type": "unknown",
    "description": null,
}
```

Проблемы при сравнении дампов

- Разный уровень вложенности данных
- Разный тип данных
- Разный нейминг данных
- Разный порядок данных
- Разное наполнение данными



JSON

Маппинг данных

- По каждой компоненте продукта и для каждого релиза подготовливаем список данных для миграции
  - Известно, какие данные не нужно сравнивать
  - Известны уровни вложенности
  - Известно, какие типы данных в какие мигрировали
- Благодаря этим знаниям мы можем манипулировать данными

Таким образом мы имитируем миграцию данных от «старых» релизов к «новым» и в итоге сравниваем дампы эмулированных и мигрированных данных между собой

```
Base_Topology.__init__(self)
self.dump_path = os.path.join(dump_dir, 'topology.json')
self.extra_hosts_fields = {
    'hosts': [
        'violations',
        'description',
        'exclusions_counter',
        'importance',
        'violations_counter'
    ],
    'interfaces': [
        'description',
        'vlan'
    ],
}
self.fields_for_swap = [
    'source_mac',
    'source_authorized',
    'source_type',
    'source_ip',
    'source_text',
    'target_mac',
    'target_authorized',
    'target_type',
    'target_ip',
    'target_text',
]
self.data_dump = self.swap_old_fields_to_new(self.data_dump)
self.alerts = self.load_data_to_template(
    'alerts',
    self.data_dump,
    self.alerts_template,
    self.extra_hosts_fields
)
```

```
alerts:
   impact:
id:
critical:
progress:
status:
type:
status_changed:
start:
end:
group:
capacity:
change:
severity:
label:
source_mac:
source_authorized:
source_type:
source_ip:
source_text:
target_mac:
target_authorized:
target_type:
target_ip:
target_text:
```

Шаблон для сортировки данных

Файл **alerts_template.yml** содержит все возможные ключи из json-файла по данной компоненте.
Это нужно для того, чтобы список ключей был одинаковым для всех данных, чтобы приводить их к одному виду

Это работает, если фрагмент данных имеет в себе **id**.
Только в этом случае мы сможем правильно сортировать и сравнивать данные между собой, иначе будет невозможно понять, что с чем сравнивать и как сортировать

Сравнение данных

Check_Difference.py 9.62 KiB

```python
import json

from jsondiff import diff
from Messages import (error_by_section, error_by_section_without_id,
                      error_key_not_found, error_len)
from releases.release_3_1 import Release_3_1
from releases.release_3_3 import Release_3_3
from releases.release_4_1 import Release_4_1
from releases.release_4_2 import Release_4_2
from releases.release_4_3 import Release_4_3
from releases.release_4_4 import Release_4_4


class Check_Diffrenece():
    """
    Class contains comparing methods for compare data between two releases.

    init - method lets the class initialize the object's attributes and serves
    """

    def __init__(self):
        """
        Class setups.

        skip_msg - message for skiped fields in compare
        ignore_fields - fields dont needed in comparing
        """
        self.skip_msg = 'This field dont exist in previous sensor version'
        self.ignore_fields = ['last_activity']

    @staticmethod
    def compare_two_values(first, second):
        """
        Compare two values.

        return True if values equals
```

> **Check_Difference —** класс, который содержит все возможные методы, чтобы сравнивать максимально глубоко и выборочно данные JSON-формата

> Сравнения **списков**, внутри которых находятся словари

> Сравнение словарей, если есть элементы **с id**

> Сравнение **значений**

> Мало знать, что данные изменились — нужно знать **точное место этих изменений**. Ведь данные могут быть размером в мегабайты

> Сравнение **словарей**, внутри которых находятся списки

> Сравнение словарей **без id**

Все эти методы как помогают друг другу сравнивать объекты между собой, так и работают атомарно

Время тестирования

Проверка сборки на работоспособность

30 минут min

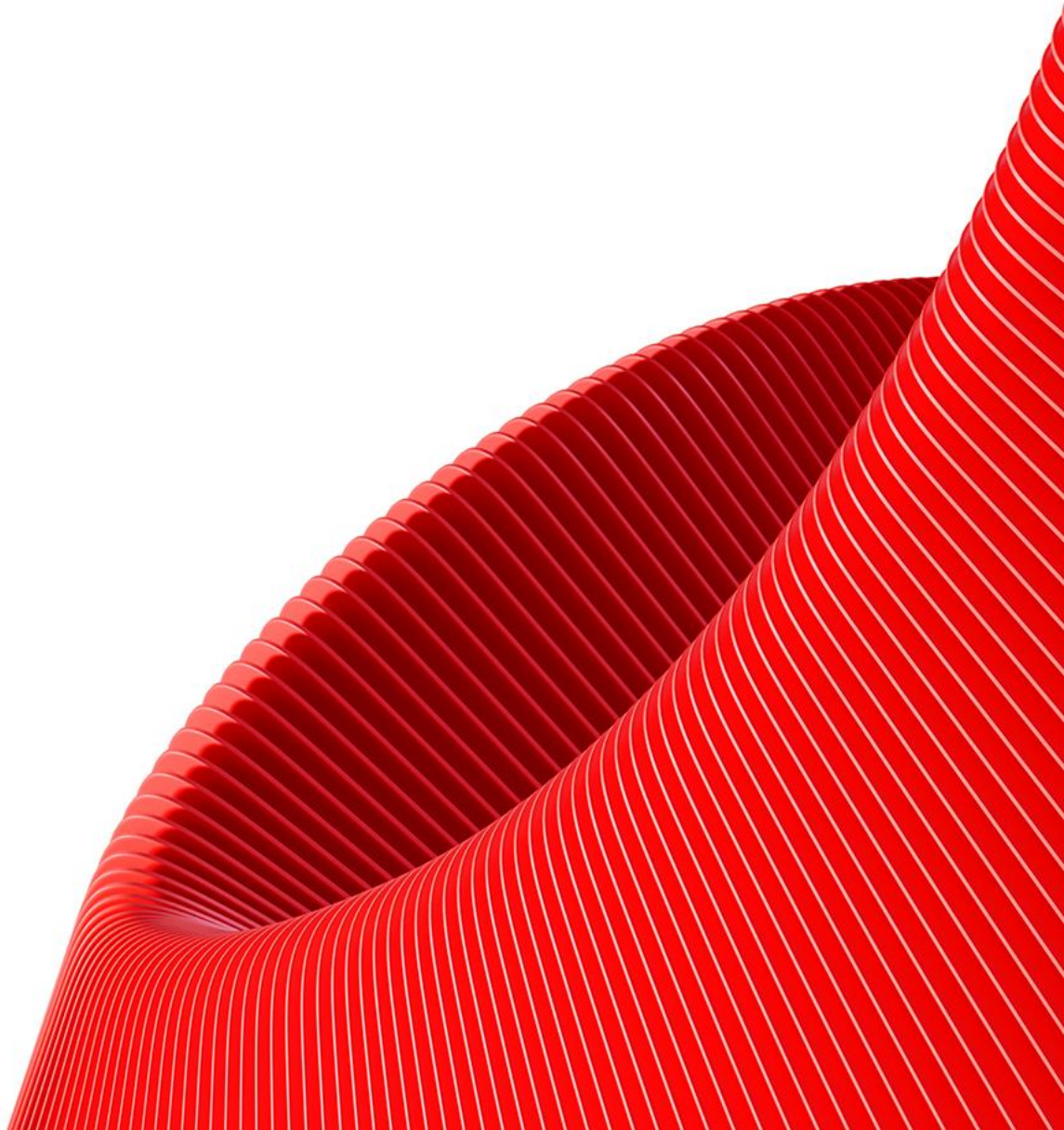
Минимальное время

24 часа max

Максимальное время

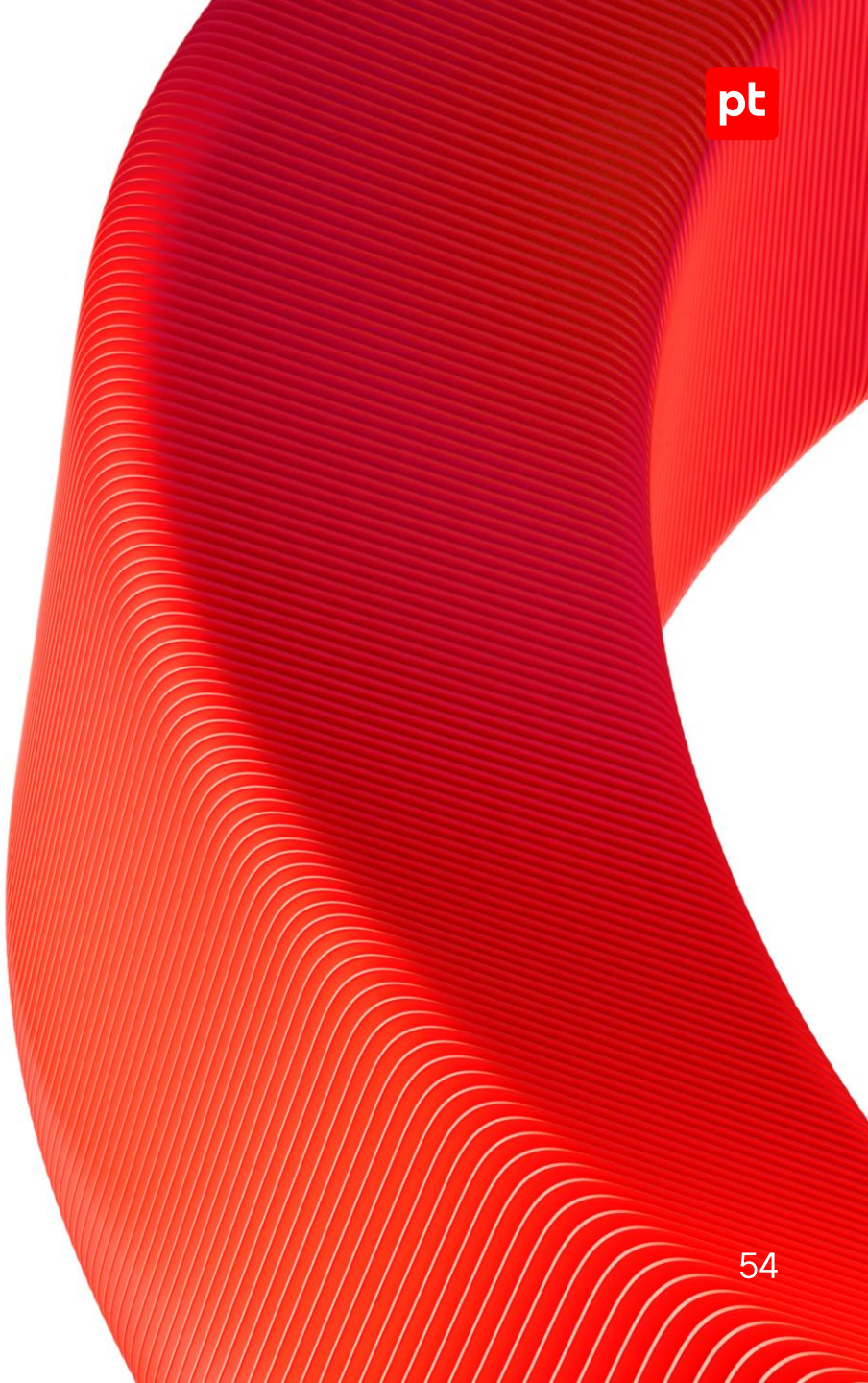
•04

Выводы



Выводы

- Не забывайте проверять обновления в двух частях: инсталлятор и работоспособность
- Проблему масштабируемости нужно решать
- Проверка миграции данных — это важно



Никита Жевелков
PT ISIM Tech Lead