

Как мы изменения в сети валидировали



Дмитрий Ларин
Руководитель R&D, Nadal Project

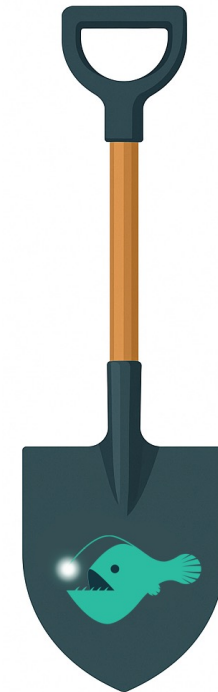
0 чем доклад

Как проверить все изменения в сети:

- Нет устройств с одинаковыми hostname
- На всех соседних портах одинаковый MTU
- На всех активных портах выставлен description
- И т.д.

О чем еще:

- Какой подход мы придумали и как реализовали
- Как построить процесс валидации изменений
- Куда мы идем — AI-аналитика



Лопата для «выкапывания»
данных о сети

Обо мне

- Руководитель R&D в Hadal Project
- Разрабатывал и внедрял автоматизацию управления оборудованием в корпоративных и операторских сетях
- PhD @ МФТИ



Знакомьтесь, Хадалыч

Валидация изменений

Изменения в сети требуют постоянного контроля:

- **Golden Config** (AAA, NTP, стандартизация названий и т.д.)
- **Проверки состояний устройств** (EoS, EoL, Uptime, версии ОС и др.)
и протоколов (LLDP, Duplex, MTU, STP, OSPF, BGP и др.)
- **Проверки на соответствие рекомендациями безопасности и лучшим практикам** (PCI DSS, CIS, и др.)



Неприметное нарушение

Почему необходим постоянный контроль

- **Гигиена** → выявление нарушений на ранних этапах, никаких «внезапностей» при траблшутинге
- **Автоматизация** → если автоматизировать хаос, то получится автоматизированный хаос
- **Безопасность** → ничто не забыто, вы знаете про каждое устройство в вашей сети



Ошибки, нарушения, отклонения, «забыли обновить доку» и т.д. затрудняют траблшутинг

Простая и важная задача. Которую сложно реализовать

Чем больше инфраструктура и «зоопарк» вендоров — тем сложнее ее валидировать

Netconf, gNMI и другие интерфейсы
либо не поддерживаются на оборудовании,
либо вторичны

Единственный выход:

- Собрать выводы команд по SSH
- Распарсить неструктурированный текст
- Нормализовать датасет, обработать, ...



«Зоопарк» вендоров добавляет
значительную сложность

Пример: статусы интерфейсов

Cisco NX OS: две команды

```
Ethernet1/1 is up  
admin state is up, Dedicated Interface  
Hardware: 100/1000/10000 Ethernet, address: 0c48.fa00.1b08 (bia 0c48.fa00.0101)  
Internet Address is 169.254.0.6/31  
MTU 1500 bytes, BW 1000000 Kbit , DLY 10 usec  
reliability 255/255, txload 1/255, rxload 1/255  
Encapsulation ARPA, medium is broadcast  
full-duplex, 1000 Mb/s  
Beacon is turned off
```

Huawei VRP: одна команда

```
Ethernet1/0/0 current state : UP (ifindex: 6)  
Line protocol current state : UP  
Last line protocol up time : 2025-05-09 14:28:12  
Link quality grade : GOOD  
Description: towards spine1  
Route Port, The Maximum Transmit Unit is 1500  
Internet Address is 169.254.0.3/31  
IP Sending Frames' Format is PKTFMT_ETHNT_2, Hardware address is 3872-655b-4a00  
Loopback: none, LAN full-duplex mode, Pause Flowcontrol: Receive Enable and Send Enable  
Last physical up time : 2025-05-09 14:28:12
```

Port	Name	Status	Vlan	Duplex	Speed	Type
Eth1/1	--	connected	routed	full	1000	10g
Eth1/2	--	connected	routed	full	1000	10g
Eth1/3	--	connected	routed	full	1000	10g
Eth1/4	--	notconnec	1	auto	auto	10g
Eth1/5	--	notconnec	1	auto	auto	10g
Eth1/6	--	notconnec	1	auto	auto	10g

**Необходимо выполнять различные команды
и писать кастомные парсеры.
Даже для самых «базовых» данных**

Простая и важная задача. Которую сложно реализовать

- «Статусы интерфейсов, активные сессии BGP... Мы могли бы выделить отдельного человека под эту задачу на 6 месяцев... 80% задачи — это сбор и парсинг»
- Очень много повторяющейся работы
- ... **но в итоге эта задача не решается:** инженеры преимущественно не разработчики и заняты задачами эксплуатации



Когда пишешь парсеры для всех
моделей и версий ОС

А что если упростить работу
с данными до одного запроса?



Мощный инструмент
в правильных руках

Что нужно валидировать

Конфигурации (aka Golden Config)

- Настройки AAA, пользователей, ACL и т.п.
- Настройки hostname в нужном формате, выставленные description на портах и т.п.

Проверки по группам устройств, наличие необходимых и отсутствие запрещенных строк конфигурации

Что нужно валидировать

Конфигурации (aka Golden Config)

- Настройки AAA, пользователей, ACL и т.п.
- Настройки hostname в нужном формате, выставленные description на портах и т.п.

Проверки по группам устройств, наличие необходимых и отсутствие запрещенных строк конфигурации

Операционное состояние

- Состояния устройств — версия ОС, недавние перезагрузки, статусы портов и т.п.
- Состояния протоколов — перезагрузки и состояния сессий (BGP, OSPF, STP, LLDP и т.п.), таблицы RIB/FIB/MAC/ARP и т.п.

Проверки по устройствам и ближайшим соседям, множество параметров

Что нужно валидировать

Конфигурации (aka Golden Config)

- Настройки AAA, пользователей, ACL и т.п.
- Настройки hostname в нужном формате, выставленные description на портах и т.п.

Проверки по группам устройств, наличие необходимых и отсутствие запрещенных строк конфигурации

Операционное состояние

- Состояния устройств — версия ОС, недавние перезагрузки, статусы портов и т.п.
- Состояния протоколов — перезагрузки и состояния сессий (BGP, OSPF, STP, LLDP и т.п.), таблицы RIB/FIB/MAC/ARP и т.п.

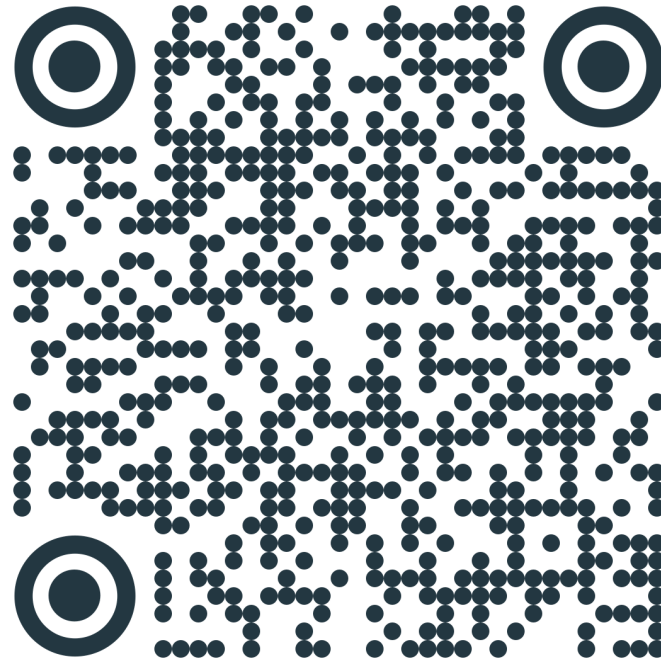
Проверки по устройствам и ближайшим соседям, множество параметров

End-to-End поведение сети

- Есть маршрут, есть резервные пути, нет петель, прямой и обратный пути совпадают и т.п.
- Выполнено разграничение по зонам, трафик ходит только до разрешенных портов, нет доступа из Интернета и т.п.

Проверки маршрутизации и коммутации по всей сети

Что нужно валидировать



nexthop 24

«Как проверить, что ваша сеть работает правильно: формальная верификация»

Что учитывать – вложенность данных

Классический подход: используем регулярки, проверяем наличие или отсутствие строчек конфигурации

Но, что делать с:

- Все IPv4 подсети на интерфейсах с YYY должны входить в ACL XXX
- Если настроен X, то должен быть настроен Y

Необходимо использовать функциональные проверки по всей конфигурации и всем состояниям



Вложенность данных

Что учитывать – топологическая связность

Многие проверки завязаны на топологию сети:

- Одинаковые MTU/Duplex на соседних портах
- Все ли анонсированные по BGP устройством префиксы были установлены в таблицу его соседа?
- Верификация End-To-End путей
- Отключенный LLDP на границе сети
- И многие другие

При валидации необходимо учитывать физическую и логическую связность между устройствами



При валидации необходимо
учитывать все связи

Что учитывать — изменения во времени

Все сломалось, но час назад все было хорошо — что произошло?

- Когда и кто изменил конфигурацию
- Когда удалился маршрут
- Когда в сети появилось новое устройство
- И другие проверки

**При валидации необходимо поддерживать
версионирование данных**



Куда пропали маршруты?

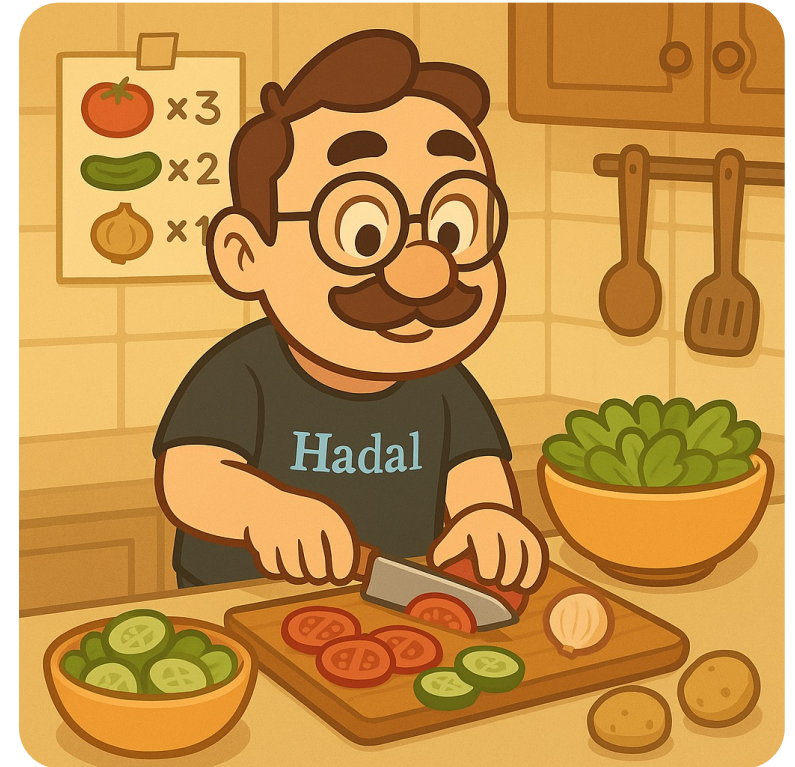
Произвольные выборки

Уже храним и версионим конфигурации, операционные состояния и топологию

Используем эти данные для произвольных выборок:

- На каких портах настроен VLAN XXX
- Любая статистика (оборудование, платы, интерфейсы, утилизация, количество и т.п.)
- Выборки по критериям (вендор, версия ОС, регион и т.п.)

Запросы к сети как к Базе Данных



Рецепт салата из VLAN

Реализация



Для работы необходимо подобрать
правильный инструмент

Подход №1 – решаем «в лоб»

Сбор данных: выполняем show-команды по SSH, собираем выводы

Валидация: набор регулярных выражений (например, строчек конфигураций), которые должны присутствовать в выводах команд или не должны

Результат: набор устройств на которых прошли/не прошли проверки

NTP Huawei

Name:

NTP Huawei

Description:

Check NTP server configuration at Huawei devices.

Site:

DEMO

Vendor:

Huawei

Model:

*

OS:

VRP

OS Version:

VRP 8.X

Obligatory checks:

system ntp server 10.10.15.8.*
system ntp server 10.10.15.8.*
system ntp server 10.10.15.8.*
system ntp server 10.10.15.8.*

Prohibited checks:

system ntp server .*

Edit

List of checked devices

List of all devices which were under test

Device	Site	Vendor	Model	OS	OS ver...	Severity	
huawei 1	DEMO	huawei	model 1	VRP	5.170	PASSED	show details
huawei 1	DEMO	huawei	model 1	VRP	5.170	FAILED	show details

Rows per page: 100 1-2 of 2

Подход №1 – решаем «в лоб»

Преимущества

- Простота реализации:
достаточно собрать
выводы команд
и написать регулярки
- Подходит для
простых задач вида «есть
или нет» (например,
в конфигурациях)

Подход №1 – решаем «в лоб»

Преимущества

- Простота реализации: достаточно собрать выводы команд и написать регулярки
- Подходит для простых задач вида «есть или нет» (например, в конфигурациях)

Недостатки

- Отсутствуют функциональные проверки («если, то, иначе»)
 - Что делать если данные нужны из нескольких show-команд?
 - Нельзя делать произвольные выборки данных (помимо проверок)
 - Все равно нужно писать парсеры под каждого вендора, модель, ОС, версию ОС
 - Нет возможности проводить проверки с учетом топологической связности
- Итого: таким подходом можно покрыть только малую часть всех необходимых проверок + он очень трудоемкий**

Переосмыслиаем

НЕСТРУКТУРИРОВАННЫЕ ДАННЫЕ

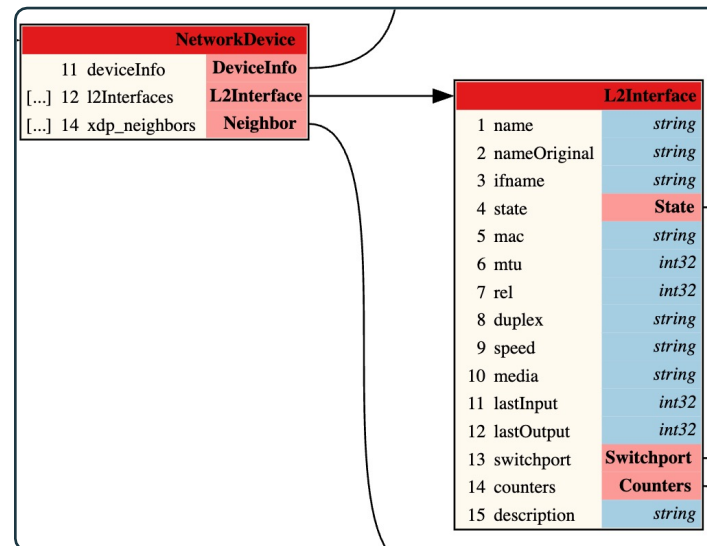
```
Ethernet1/0/0 current state : UP (ifindex: 6)
Line protocol current state : UP
Last line protocol up time : 2025-05-09 14:28:12
Link quality grade : GOOD
Description: towards spine1
Route Port,The Maximum Transmit Unit is 1500
Internet Address is 169.254.0.3/31
IP Sending Frames' Format is PKTFMT_ETHNT_2, Hardware address
Loopback:none, LAN full-duplex mode, Pause Flowcontrol: Rece
Last physical up time : 2025-05-09 14:28:12
Last physical down time : 2025-05-09 14:27:52
Current system time: 2025-05-09 14:37:22
Statistics last cleared:never
  Last 300 seconds input rate: 0 bits/sec, 0 packets/sec
  Last 300 seconds output rate: 0 bits/sec, 0 packets/sec
  Input peak rate 0 bits/sec, Record time: -
  Output peak rate 0 bits/sec, Record time: -
  Input: 0 bytes, 0 packets
  Output: 0 bytes, 199 packets
  Input:
    Unicast: 0 packets, Multicast: 0 packets
    Broadcast: 0 packets, JumboOctets: 0 packets
    CRC: 0 packets, Symbol: 0 packets
```

Переосмыслиаем

НЕСТРУКТУРИРОВАННЫЕ ДАННЫЕ

```
Ethernet1/0/0 current state : UP (ifindex: 6)
Line protocol current state : UP
Last line protocol up time : 2025-05-09 14:28:12
Link quality grade : GOOD
Description: towards spine1
Route Port,The Maximum Transmit Unit is 1500
Internet Address is 169.254.0.3/31
IP Sending Frames' Format is PKTFMT_ETHNT_2, Hardware address
Loopback:none, LAN full-duplex mode, Pause Flowcontrol: Rece
Last physical up time : 2025-05-09 14:28:12
Last physical down time : 2025-05-09 14:27:52
Current system time: 2025-05-09 14:37:22
Statistics last cleared:never
  Last 300 seconds input rate: 0 bits/sec, 0 packets/sec
  Last 300 seconds output rate: 0 bits/sec, 0 packets/sec
  Input peak rate 0 bits/sec, Record time: -
  Output peak rate 0 bits/sec, Record time: -
  Input: 0 bytes, 0 packets
  Output: 0 bytes, 199 packets
  Input:
    Unicast: 0 packets, Multicast: 0 packets
    Broadcast: 0 packets, JumboOctets: 0 packets
  CRC: 0 packets, Symbols: 0 packets
```

НОРМАЛИЗИРОВАННАЯ ВЕНДОР-НЕЗАВИСИМАЯ СХЕМА ДАННЫХ

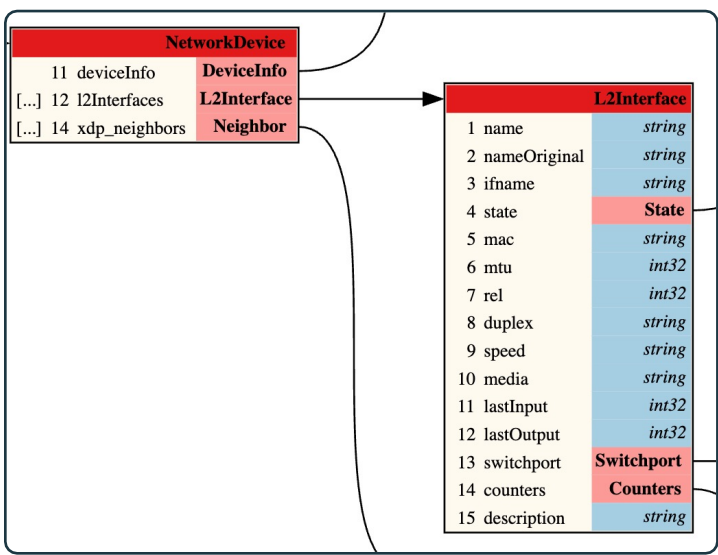


Переосмыслиаем

НЕСТРУКТУРИРОВАННЫЕ ДАННЫЕ

```
Ethernet1/0/0 current state : UP (ifindex: 6)
Line protocol current state : UP
Last line protocol up time : 2025-05-09 14:28:12
Link quality grade : GOOD
Description: towards spine1
Route Port,The Maximum Transmit Unit is 1500
Internet Address is 169.254.0.3/31
IP Sending Frames' Format is PKTFMT_ETHNT_2, Hardware address
Loopback:none, LAN full-duplex mode, Pause Flowcontrol: Rece
Last physical up time : 2025-05-09 14:28:12
Last physical down time : 2025-05-09 14:27:52
Current system time: 2025-05-09 14:37:22
Statistics last cleared:never
  Last 300 seconds input rate: 0 bits/sec, 0 packets/sec
  Last 300 seconds output rate: 0 bits/sec, 0 packets/sec
  Input peak rate 0 bits/sec, Record time: -
  Output peak rate 0 bits/sec, Record time: -
  Input: 0 bytes, 0 packets
  Output: 0 bytes, 199 packets
  Input:
    Unicast: 0 packets, Multicast: 0 packets
    Broadcast: 0 packets, JumboOctets: 0 packets
  CRC: 0 packets, Symbol: 0 packets
```

НОРМАЛИЗИРОВАННАЯ ВЕНДОР-НЕЗАВИСИМАЯ СХЕМА ДАННЫХ



ЗАПРОС ДАННЫХ О СЕТИ И ВИД ОТВЕТА

Простой запрос (концепт)

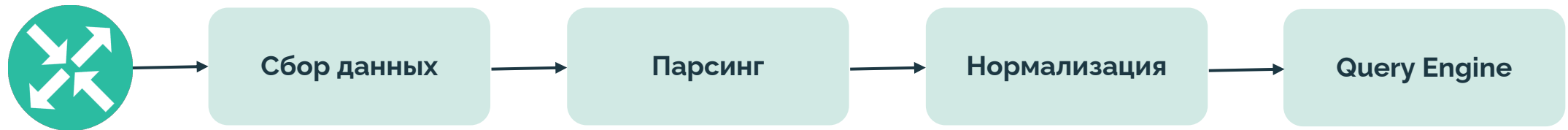
На каких интерфейсах не выставлен description?

Структурированный вывод

HOSTNAME	INTERFACE	DESCRIPTION	VIOLATION
viosl2-15.2-4	GigabitEthernet0/3	to server 1	false
viosl2-15.2-4	GigabitEthernet1/0	to server 2	false
viosl2-15.2-4	Vlan1		true
viosl2-15.2-2	GigabitEthernet0/1		true

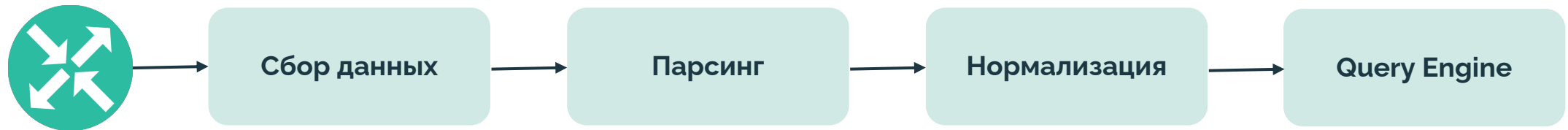
Как это реализовать?

В теории это просто



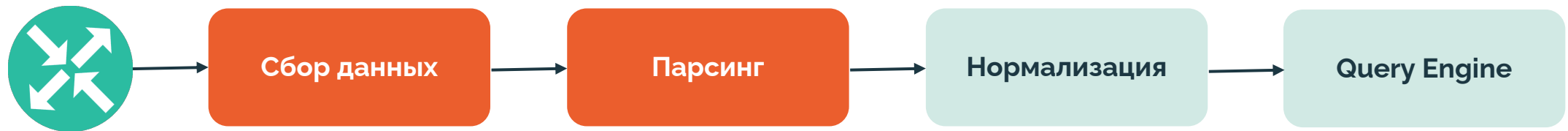
Как это реализовать?

В теории это просто



На практике: челленджи на каждом этапе

Как это реализовать?



Сбор данных

Тут все виды челленджей:

- Некоторые девайсы могут отказывать
- Некоторые девайсы могут отдавать данные только раз в 30 секунд
- Пагинация, способы подключения, не у всех есть CLI, ...
- И это только вершина айсберга



Иногда приходится
работать лопатой

Парсинг данных

Масштаб: 19 вендоров, 23 операционных системы, 68 версий ОС

Пример: только на одном типе устройств (Cisco NX-OS) больше **120 тысяч вариантов** задать валидные высокоуровневые команды конфигурации

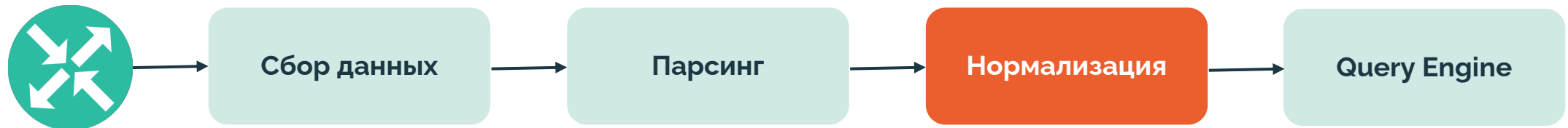
Критично: максимально упростить процесс парсинга выводов команд

Одно из основных направлений в Hadal



Качественные данные –
современное золото

Как это реализовать?



Нормализация данных

Не изобретаем колесо

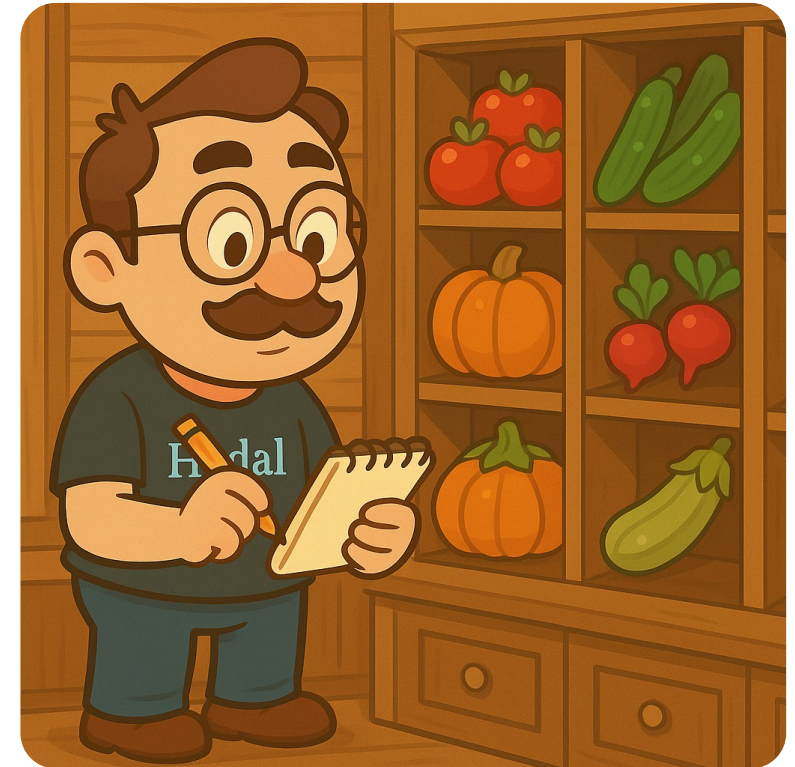
В качестве формата схемы данных
используем YANG-модели
OpenConfig + IETF

Упрощаем схемы для Read-Only
режима и добавляем топологию.
Берем лучшее из обоих миров



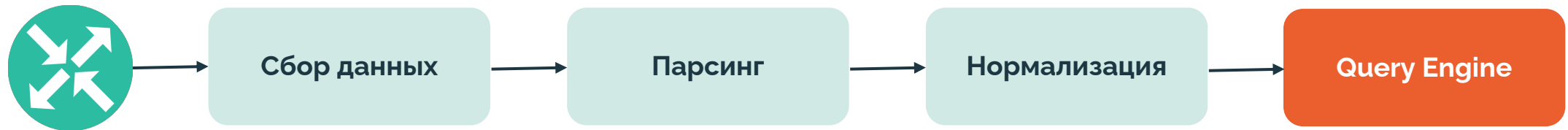
YangModels/**yang**

YANG modules from standards organizations such as the IETF, The IEEE, The Metro Ethernet Forum, open source such as Open...



Раскладываем данные
«по полочкам»

Как это реализовать?



Query Engine: модель данных

- Собираем JSON-схему данных на основе YANG-модели
- Можно обращаться к каждому полю по-отдельности
- Можно делать запросы по топологии
- Оптимизируем запросы — если пользователю нужно только поле X, то выгружаем из хранилища только значение этого поля (GraphQL-like)



Query Engine: язык запросов

- Язык запросов — Python
- Работа с данными — как с JSON (прозрачно для пользователя)
- Код проверок хранится в системе (+импорт/экспорт), не нужно хранить скрипты во внешних системах
- За счет нормализованных данных — даже сложные проверки по всем вендорам делаются в несколько строчек кода



```
Editor ▶ Execute

1 def validation_query():
2     results = []
3
4     for device in network['devices']:
5         for interface in device['l2Interfaces']:
6             if interface['state']['admin'] == 'up':
7                 violation = interface['description'] == ""
8                 results.append({
9                     "hostname": device['deviceInfo']['hostname'],
10                    "interface": interface["name"],
11                    "description": interface["description"],
12                    "violation": violation,
13                })
14
15     return results
```


Query Engine: результат запроса

- Результат запроса — таблица
- Пользователь сам задает необходимые для отображения поля в запросе
- Можно использовать для:
 - 1) Проверки (столбец violation)
 - 2) Произвольные выборки
- Результаты можно выгружать при помощи API (REST/gRPC)

Validation Results			
HOSTNAME	INTERFACE	DESCRIPTION	VIOLATION
viosl2-15.2-4	GigabitEthernet0/2		true
viosl2-15.2-4	GigabitEthernet0/3	to server 1	false
viosl2-15.2-4	GigabitEthernet1/0	to server 2	false
viosl2-15.2-4	Vlan1		true
viosl2-15.2-2	GigabitEthernet0/1		true
viosl2-15.2-2	GigabitEthernet0/3		true
viosl2-15.2-2	GigabitEthernet1/0		true
viosl2-15.2-2	GigabitEthernet1/1		true
viosl2-15.2-2	Vlan1		true
viosl2-15.2-3	GigabitEthernet0/0		true
viosl2-15.2-3	GigabitEthernet0/1		true
viosl2-15.2-3	GigabitEthernet0/2		true
viosl2-15.2-3	GigabitEthernet0/3		true
viosl2-15.2-3	GigabitEthernet1/0		true
viosl2-15.2-3	GigabitEthernet1/1		true
viosl2-15.2-3	Vlan1		true

Rows per page: 100 1-50 of 50

Результат

← → ↺ ⚠ Not Secure 10.199.199.113/validation/edit/a2713498-6cb1-47e7-9bb2-5da2ca4cca8d?from=07%2F05%2F2024+11%3A03%3A45&to=07%2F05%2F2025+11%3A03%3A45&ru... 🔍 ☆ 📌 👤 New Chrome available ⋮

≡ Hadal 🔑

20/03/2025, 23:05:57 ▾ 07/05/2024 11:03:45 → 07/05/2025 11:03:45 📅

👤

Validation check

Name
Interface descriptions

Intent
Descriptions at all active interfaces must be present

Priority
Medium ▾

Save and close

Cancel

Editor ▶ Execute

```
1 def validation_query():
2     results = []
3
4     for device in network['devices']:
5         for interface in device['l2Interfaces']:
6             if interface['state']['admin'] == 'up':
7                 violation = interface['description'] == ""
8                 results.append({
9                     "hostname": device['deviceInfo']['hostname'],
10                    "interface": interface["name"],
11                    "description": interface["description"],
12                    "violation": violation,
13                })
14
15     return results
```

Results Data Model

Validation Results

HOSTNAME 4	INTERFACE rnet0/2	DESCRIPTION	VIOLATION
viosl2-15.2-4	GigabitEthernet0/3	to server 1	false
viosl2-15.2-4	GigabitEthernet1/0	to server 2	false
viosl2-15.2-4	Vlan1		true
viosl2-15.2-2	GigabitEthernet0/1		true
viosl2-15.2-2	GigabitEthernet0/3		true
viosl2-15.2-2	GigabitEthernet1/0		true

Rows per page: 100 ▾ 1-50 of 50 < >

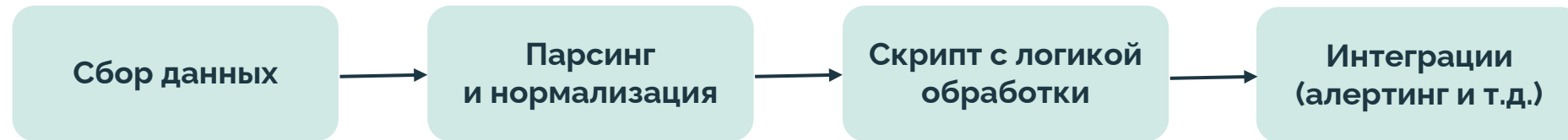
Пример: где настроен VLAN 10

Editor ▶ Execute

```
1 def validation_query():
2     results = []
3
4     for device in network['devices']:
5         for vlan in device['vlanTable']:
6             if vlan["id"] == 10:
7                 results.append({
8                     "hostname": device['deviceInfo']['hostname'],
9                     "id": vlan["id"],
10                    "name": vlan['name']
11                })
12
13     return results
```

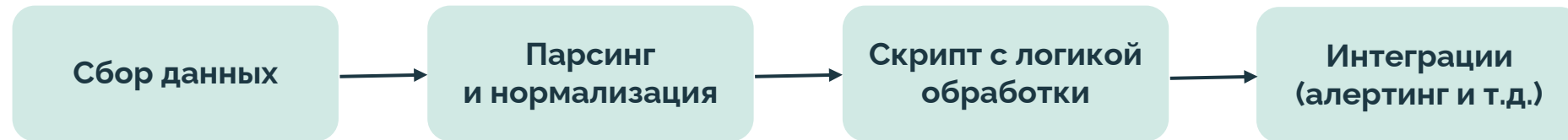
Процесс валидации

Весь процесс — инженер задействован на каждом этапе

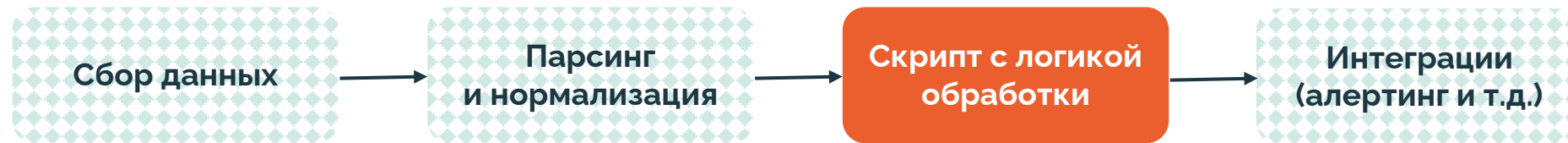


Процесс валидации

Весь процесс — инженер задействован на каждом этапе



Как мы его упростили



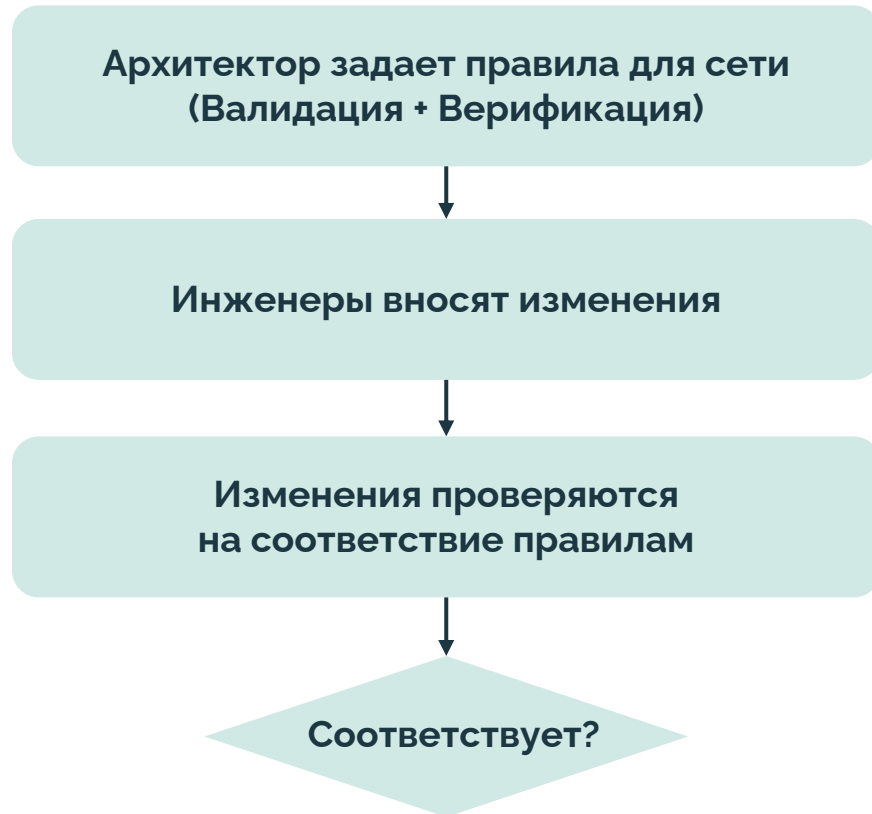
Инженер занимается только обработкой данных и интерпретацией результатов

Сценарии использования

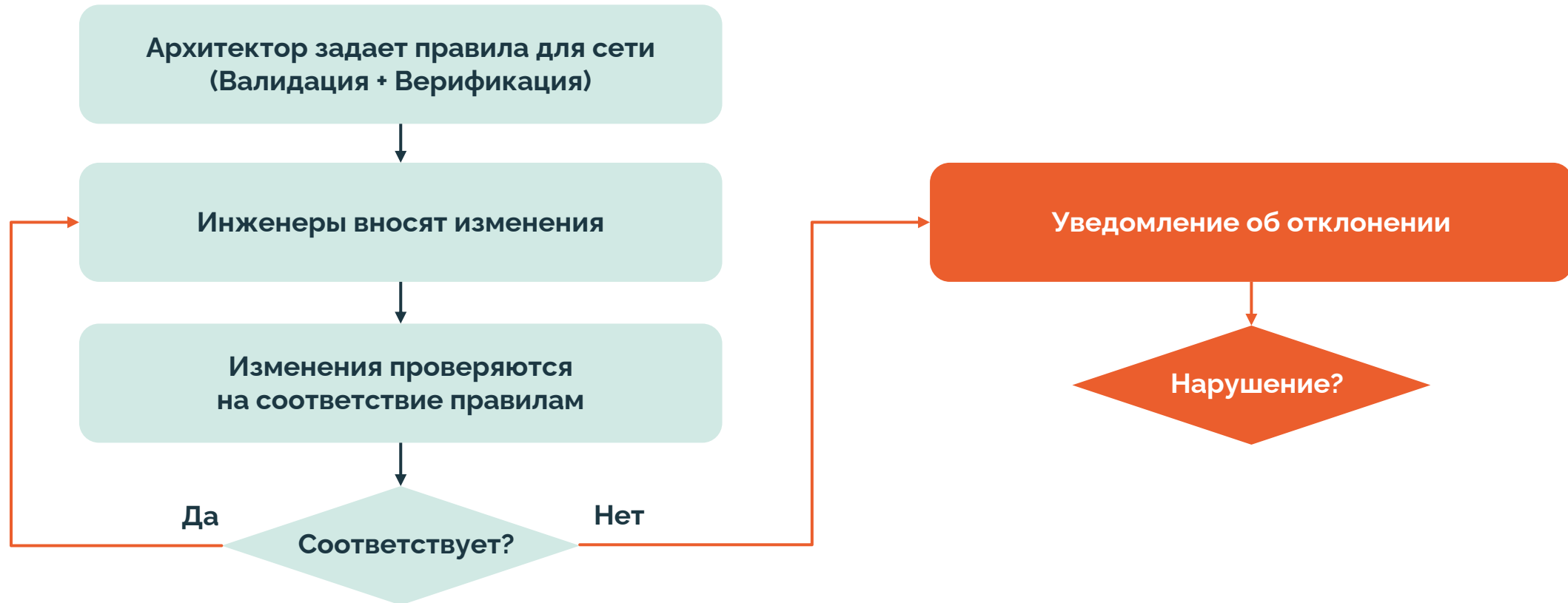


Автоматизацию в каждый огород!

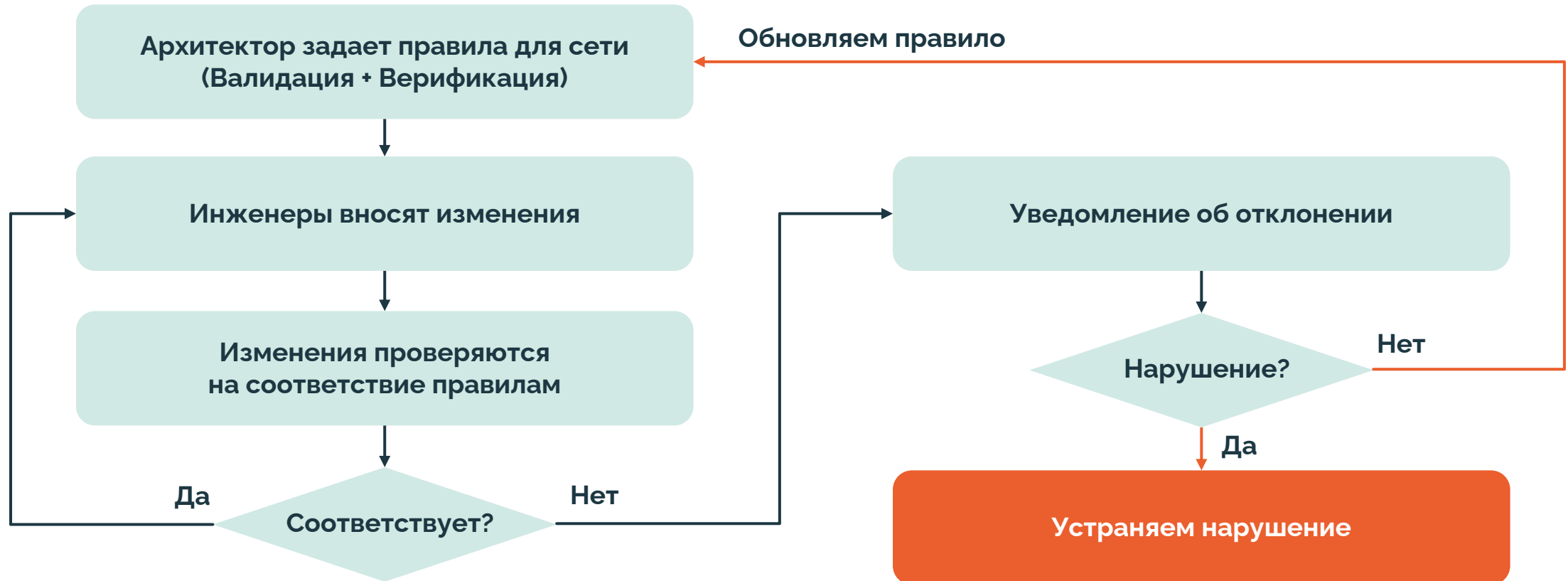
Процесс контроля изменений



Процесс контроля изменений



Процесс контроля изменений



Автоматизация устранения нарушений

HOSTNAME	INTERFACE	NEIGHBOR
spine1	GigabitEthernet1	leaf1
spine1	GigabitEthernet2	leaf2
spine1	GigabitEthernet3	edge1
Router	GigabitEthernet2.1	viosl2-15.2-2

Автоматизация устранения нарушений

HOSTNAME	INTERFACE	NEIGHBOR	COMMANDS
spine1	GigabitEthernet1	leaf1	conf t interface GigabitEthernet1 description "to device leaf1" exit
spine1	GigabitEthernet2	leaf2	conf t interface GigabitEthernet2 description "to device leaf2" exit
spine1	GigabitEthernet3	edge1	conf t interface GigabitEthernet3 description "to device edge1" exit
Router	GigabitEthernet2.1	viosl2-15.2-2	conf t interface GigabitEthernet2.1 description "to device viosl2-15.2-2"

Команды для устранения нарушения

Выгружаем результаты проверки по API

Остается разлить на оборудование (scrapli, ansible и т.п.)

Выборки и импортозамещение

Снизить расходы на закупку нового оборудования

- Сколько занятых портов в сегменте, сколько единиц оборудования необходимо купить и можно ли сократить количество?
- Какие ресурсы устройств задействованы сейчас? Можно ли купить более дешевые модели?

Подтвердить требования к оборудованию

- Какой фича-сет требуется от импортозамещенного оборудования? Все ли учли? Не требуем лишнего?



Огурцы или помидоры?

Что дальше: валидация сети

- Расширяем датасет доступных в модели данных
- Исследуем способы еще большего упрощения написания запросов
- Исследуем методы автоматической спецификации сети (например, определение Golden Config)



Что дальше: ХХ

Сейчас:

- Документируем сеть
- Контролируем изменения
- Решаем задачи планирования и моделирования

Очень много детализированных и скоррелированных данных о сети



Что дальше: XX



Что дальше: AI



Вместо заключения

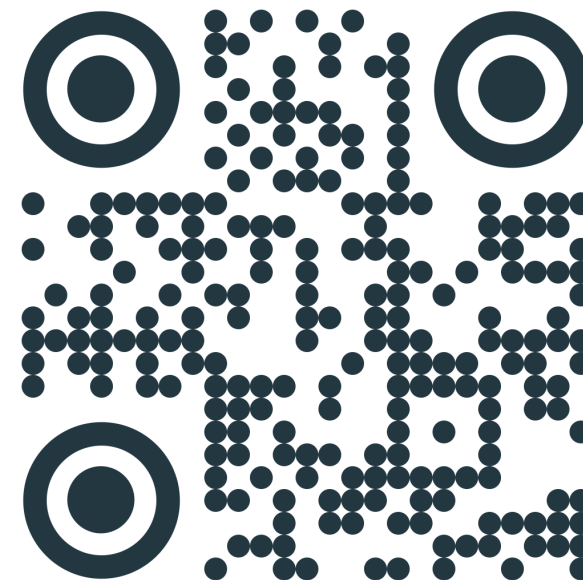
Валидация изменений

- Контроль всех изменений
- Произвольные выборки
- Исторический анализ

Корреляция событий и Root-Cause нарушений

Много логов и метрик среди которых сложно найти нарушения и их причины

Вступайте в нашу
группу в Telegram!



@hadal_project

Спасибо за внимание!

Дмитрий Ларин, Hadal Project

📍 @menetelko

✉ larin.dv@hadal.tech

🌐 hadal.tech



Оставить отзыв
на доклад можно
по QR-коду

