Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Ansible на парке из 40 серверов: наш рабочий стандарт

Инвентарь, коллекции, vault-id, forks и serial, аудит дрейфа и CI-прогон плейбуков

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Парк в 35-45 серверов ломается не на первом плейбуке, а на второй год эксплуатации: инвентарь разрастается, роли расползаются по копиям, все секреты лежат под одним паролем, прогон на 40 хостов идёт часами на дефолтных forks = 5, а внесённые руками правки никто не видит. Мы строим Ansible-контур так, чтобы он оставался управляемым при росте парка и смене инженера.

Почему это важно бизнесу

- Обновление всего парка укладывается в окно обслуживания, а не растягивается на рабочий день
- Конфигурация серверов лежит в Git: видно кто, когда и что менял — это готовый ответ аудитору
- Ручные правки на серверах вскрываются ночным аудитом, а не всплывают в момент аварии
- Знания не заперты в голове одного админа: роли документированы, прогон повторяем любым инженером
- Ошибка в плейбуке останавливается на первой партии хостов и не выносит весь парк разом

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

2.20

ansible-core управляющего узла = пакет Ansible 13; ветка 2.21 вышла, Ansible 14 в разработке

docs.ansible.com, Releases and maintenance

3.12-3.14

Python управляющего узла для ansible-core 2.20 и 2.21; на целевых узлах 3.9-3.14

docs.ansible.com, матрица версий

5 -> 25

forks: значение по умолчанию 5, поднимаем до 25 в ansible.cfg под парк 40 хостов

docs.ansible.com + наш стандарт

1, 5, 25%

serial-лестница раскатки: канарейка, малая партия, затем четверть парка за шаг

наш стандарт раскатки

86400 с

fact_caching_timeout: держим значение по умолчанию для jsonfile-кэша при gathering = smart

docs.ansible.com, Configuration Settings

1 раз/сутки

ночной аудит дрейфа: прогон всего парка в --check --diff, отчёт по changed

наш стандарт эксплуатации

РЕАЛИЗАЦИЯ 1 ИЗ 3

Инвентарь и группы: два измерения на 40 хостов

Что настраиваем

Проектируем структуру так, чтобы один хост попадал и в срез по роли сервиса, и в срез по клиенту, без дублей переменных

Как мы это делаем

1. Разносим окружения: inventories/production и inventories/staging, в каждом свой hosts.yml, group_vars/ и host_vars/
2. Даём хосту членство сразу в двух измерениях: функциональное (web, db, cache, mon) и клиентское (cl_alfa, cl_beta), плюс зонтичные группы через children
3. Общие значения кладём в group_vars/all.yml, специфику клиента — в group_vars/cl_alfa/vars.yml, уникальное железо — только в host_vars
4. Проверяем результат до прогона: ansible-inventory -i inventories/production --graph --vars и ansible all --list-hosts с нужным --limit
5. В плейбуках адресуемся группами: --limit 'web:&cl_alfa' даёт пересечение, --limit '!cl_beta' исключает клиента целиком

РЕЗУЛЬТАТ

Любая операция адресуется одной строкой: обновить все веб-серверы, только одного клиента или пересечение того и другого. Переменные не копируются между файлами, а наследуются по приоритету group_vars -> host_vars: добавление 41-го сервера — это строка в hosts.yml и ноль правок в ролях.

КЛЮЧЕВОЙ НЮАНС

Инвентарь переделывать дороже всего: групповые имена мы фиксируем на старте и запрещаем адресовать прогоны по IP. Хост без группы — дефект, он не получит ни одной переменной окружения.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Секреты и раскатка: vault-id по клиентам, serial-лестница

Что настраиваем

Один мастер-пароль на весь парк мы не используем: разграничиваем доступ по клиентам и окружениям, а прогон ведём партиями

Как мы это делаем

1. Каждому клиенту и окружению — свой vault-id вида label@source: ansible-playbook --vault-id cl_alfa@~/.vault/cl_alfa.sh --vault-id prod@prompt site.yml
2. Постоянные метки прописываем в vault_identity_list и включаем vault_id_match, чтобы пароль применялся только к своей метке
3. Открытые переменные держим в vars.yml со ссылками вида db_password: "{{ vault_db_password }}", шифрованные значения — в vault.yml рядом
4. Раскатка ступенями: serial: [1, 5, "25%"] плюс max_fail_percentage, чтобы партия с отказами прерывала play, а не докатывалась до конца
5. Точечные шаги ограничиваем throttle и run_once, а сам плейбук сначала гоняем с --check --diff и --tags на одной канарейке

РЕЗУЛЬТАТ

Уход инженера или компрометация закрывается ротацией одного vault-id (ansible-vault rekey --new-vault-id), а не сменой всех секретов парка. Ошибочная роль останавливается на первом-втором хосте: до 40 машин изменение не доезжает, откат — git revert и повторный прогон по тому же --limit.

КЛЮЧЕВОЙ НЮАНС

Пароль от vault не лежит в обёртке открытым текстом: vault-скрипт берёт его из системного keyring и печатает в stdout. Шифруем файл целиком, а не отдельные строки — так diff в Git не выдаёт структуру секретов.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Дрейф конфигураций и CI: ночной аудит плюс проверка изменений

Что настраиваем

Ручные правки на серверах неизбежны, поэтому мы их не запрещаем, а обнаруживаем — и не пускаем в парк непроверенный код

Как мы это делаем

1. Ночью по расписанию гоним весь парк в режиме --check --diff; каждый changed — кандидат в дрейф, а не изменение
2. Для аудита ставим stdout_callback = ansible.posix.jsonl и разбираем вывод машинно; в обычных прогонах — log_path и callbacks_enabled = ansible.posix.profile_tasks
3. Задачам на command и shell проставляем check_mode: false и changed_when, иначе в check-режиме они пропускаются и дрейф не виден
4. Каждый push проходит пайплайн: yamllint, ansible-lint 26.x, ansible-playbook --syntax-check, затем molecule test для изменённых ролей
5. Версии коллекций пинуем в requirements.yml: ansible-galaxy collection install -r requirements.yml -p ./collections

РЕЗУЛЬТАТ

Каждое утро видно, какие серверы разошлись с эталоном за ночь и по какой именно задаче. Изменение роли не попадает в production, пока не пройдёт линтер и прогон molecule в контейнере, а состав коллекций одинаков на управляющем узле и в CI-раннере.

КЛЮЧЕВОЙ НЮАНС

Дрейф ловится только идемпотентными ролями: пока повторный прогон даёт changed на ровном месте, отчёт бесполезен. Мы доводим роль до нулевого changed на втором запуске и только потом ставим её в ночной аудит.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **forks остались дефолтные, равные 5**
  40 хостов обрабатываются восемью волнами. Поднимаем forks в ansible.cfg и следим за RAM и числом SSH-сессий управляющего узла.

- ✕ **serial есть, а max_fail_percentage нет**
  Партия падает наполовину, но play уходит на следующую. Порог допустимых отказов задаём явно, иначе serial не защищает парк.

- ✕ **--check доверяют вслепую**
  Модули command и shell в check-режиме пропускаются. Без check_mode: false и changed_when прогон рисует ложную картину.

- ✕ **Один vault-пароль на весь парк**
  Доступ к репозиторию = доступ ко всем клиентам. Разделяем метками vault-id, ставим vault_identity_list и vault_id_match = True.

- ✕ **Коллекции ставятся без пинов версий**
  requirements.yml без version даёт разное поведение на ноутбуке инженера и в CI. Версию пиним, каталог collections держим локальным.

- ✕ **Идемпотентность не проектировалась**
  Роль перезапускает сервис на каждом прогоне. Изменения ведём через handlers и notify, а не безусловными задачами restart.

- ✕ **gathering = smart без кэша фактов**
  Сбор фактов на 40 хостах съедает основное время прогона. Включаем fact_caching = jsonfile с таймаутом, при нужде --flush-cache.

- ✕ **Прогоны нигде не логируются**
  Через месяц не восстановить, что и когда каталось. Включаем log_path и profile_tasks, а отчёт прикладываем к задаче.

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Git-репозиторий с inventories/, group_vars/, host_vars/, roles/, playbooks/
- Сервисная учётка с SSH-ключом и sudo вместо root-логина на управляемых узлах
- Секреты только в Ansible Vault, файл с паролем вне репозитория и вне Git
- Любой прогон сначала --check --diff, затем --limit на одну канарейку

НОРМАЛЬНО

- Группы в двух измерениях (сервис и клиент), переменные вынесены из hosts.yml
- forks, pipelining и ssh_args с ControlPersist заданы в ansible.cfg явно
- serial-лестница с max_fail_percentage в каждом массовом плейбуке
- Отдельные vault-id на клиента и окружение, vault_id_match = True

ХОРОШО

- Свои роли упакованы в коллекцию, версии зависимостей пинованы в requirements.yml
- CI на каждый push: yamllint, ansible-lint, --syntax-check, molecule test
- Ночной аудит дрейфа в --check --diff со сводным отчётом по changed
- fact_caching = jsonfile и profile_tasks: видно, какая роль съедает окно

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Есть ли Git-репозиторий, из которого воспроизводится конфигурация любого сервера парка?
- [ ] Каждый ли хост входит и в функциональную группу, и в группу клиента, без адресации по IP?
- [ ] Заданы ли forks, pipelining и ControlPersist явно или прогон идёт на дефолтных пяти потоках?
- [ ] Есть ли в массовых плейбуках serial и max_fail_percentage, чтобы отказ не ушёл на весь парк?
- [ ] Разделены ли секреты по vault-id между клиентами и окружениями, где лежит файл пароля?
- [ ] Даёт ли повторный прогон роли нулевой changed, то есть идемпотентна ли она?
- [ ] Пинованы ли версии коллекций в requirements.yml и совпадают ли они с версиями в CI?
- [ ] Проходит ли изменение роли ansible-lint и molecule до попадания в production?
- [ ] Проверяется ли парк на дрейф конфигураций по расписанию и кто читает этот отчёт?
- [ ] Сохраняются ли логи прогонов и можно ли поднять, что каталось три месяца назад?

Если хотя бы на два вопроса ответ «нет» или «не знаю» — тема требует внимания.

ЧТО МЫ ДЕЛАЕМ

Как поможет ITFresh

ITFresh — ИТ-аутсорсинг для юридических лиц до 50 рабочих мест в Москве и области. 15+ лет практики, собственная инфраструктура в дата-центре МТС (8 серверов Dell Xeon Platinum).

- Аудит парка и проектирование инвентаря: группы, переменные, окружения production и staging
- Управляющий узел: ansible-core 2.20, пин коллекций, сервисный доступ по ключу на все хосты
- Настройка Vault: метки vault-id по клиентам, хранение пароля в keyring, регламент ротации
- Пайплайн проверки: yamllint, ansible-lint, molecule, прогон по канарейке перед всем парком
- Ночной аудит дрейфа с отчётом и обучение вашего инженера работе с репозиторием

15+
лет в ИТ-поддержке

50
рабочих мест — наш профиль

МТС
дата-центр, Москва

КОНТАКТЫ

Обсудить вашу задачу

Сайт	**itfresh.ru**
Телефон	**+7 903 729-62-41**
Telegram	**@ITfresh_Boss**

Бесплатно посмотрим вашу инфраструктуру по этому чек-листу и скажем, где тонко — без обязательств.

itfresh.ru

ДОКУМЕНТАЦИЯ И НАШ СТЕК

Техническая база

01 Ansible Core: Releases and maintenance — матрица версий и Python (docs.ansible.com — 2.20/2026)

02 Configuration Settings: forks, log_path, callbacks_enabled, fact_caching (docs.ansible.com — 2.20/2026)

03 Playbook Guide: strategies, serial, max_fail_percentage, throttle (docs.ansible.com — 2.20/2026)

04 Vault Guide: Managing vault passwords, vault-id и vault_id_match (docs.ansible.com — 2.20/2026)

05 Cache plugins и Discovering variables: gathering, fact_caching (docs.ansible.com — 2.20/2026)

06 Ansible Lint 26.x и Molecule: правила и тестирование ролей (ansible.readthedocs.io — 26.x/2026)

07 Шаблон репозитория ITfresh: инвентарь, роли, регламент прогонов (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.