Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Ansible с нуля: как мы ставим автоматизацию в офисе

Control node, инвентарь, первые плейбуки, идемпотентность и Vault — наш стартовый набор

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

К нам приходят с парком 8-30 серверов и рабочих станций, который годами настраивали руками через SSH и RDP: на каждой машине свои версии пакетов, свой sudoers, свой NTP, свой набор служб. Мы закрываем это Ansible: один control node, инвентарь и роли в Git, после чего состояние хоста становится воспроизводимым текстом, а не памятью инженера.

Почему это важно бизнесу

- Настройка нового сервера перестаёт зависеть от того, кто из инженеров её делает
- Конфигурация уезжает в Git: видно, кто, когда и что поменял на каждом хосте
- Восстановление после сбоя — прогон плейбука, а не сборка сервера по памяти
- Массовое закрытие уязвимостей и смена параметров идут за один прогон, а не за неделю
- Передача обслуживания другому инженеру не требует месяца погружения в чужую инфраструктуру

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

2.21 / 2.20

ansible-core: ставим только текущую и предыдущую поддерживаемые ветки

docs.ansible.com, Releases and maintenance

3.12-3.14

Python на control node для core 2.20 и 2.21; на целевых узлах 3.9-3.14

docs.ansible.com, матрица поддержки

5 -> 20

forks: дефолт 5, наш стандарт 20 на парк 10-50 хостов

ansible.cfg, наш стандарт

60s -> 300s

ControlPersist в ssh_args: дефолт 60s, у нас 300s плюс pipelining=true (дефолт false)

docs.ansible.com, ssh connection

jsonfile

fact_caching: дефолт memory, ставим jsonfile и обязательный fact_caching_connection

docs.ansible.com, Configuration

changed=0

критерий приёмки любой роли: второй прогон подряд без изменений

наш стандарт приёмки

РЕАЛИЗАЦИЯ 1 ИЗ 3

День первый: control node, инвентарь, ansible.cfg

Что настраиваем

Типовой старт у клиента с парком 10-30 Linux-хостов и одним-двумя Windows-серверами

Как мы это делаем

1. Отдельная VM Debian 12/Ubuntu 24.04 под control node. ansible-core ставим через pipx и фиксируем версию: пакет дистрибутива обычно отстаёт на две-три ветки
2. Сервисная учётка ansible на целевых хостах, ed25519-ключ через ssh-copy-id, become: sudo с ограниченным набором команд в отдельном файле /etc/sudoers.d/ansible
3. Инвентарь только YAML и только в Git: inventory/prod/hosts.yml, group_vars/all.yml, host_vars/. Проверка структуры — ansible-inventory -i inventory/prod --graph
4. ansible.cfg под проект: forks=20, gathering=smart (дефолт implicit), fact_caching=jsonfile с fact_caching_connection, ssh_args с ControlPersist=300s, pipelining=true
5. Приёмка дня: ansible all -m ansible.builtin.ping проходит на 100% хостов, ключи хостов заранее собраны ssh-keyscan, host_key_checking оставлен в true

РЕЗУЛЬТАТ

Клиент получает единую точку управления и машиночитаемый список инфраструктуры вместо таблицы в Excel. Дальше любая задача формулируется как «добавить задачу в роль», а не «зайти на каждый сервер».

КЛЮЧЕВОЙ НЮАНС

Инвентарь пишем сразу в целевой структуре с group_vars/host_vars. Переезд с плоского /etc/ansible/hosts на YAML-дерево потом стоит дороже, чем сделать правильно в первый день.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Роль baseline и доказанная идемпотентность

Что настраиваем

Единый стандарт хоста: пакеты, NTP, ротация логов, sudoers, агент мониторинга

Как мы это делаем

1. Скелет через ansible-galaxy init roles/baseline: defaults, tasks, handlers, templates, meta. Переменные с префиксом роли, значения по умолчанию в defaults/main.yml
2. Задачи только на модулях (package, systemd_service, user, lineinfile, template), command и shell — в последнюю очередь и всегда с creates/removes или changed_when
3. Конфиги отдаём через template с {{ ansible_managed }} в шапке, перезапуск служб — только через handlers и notify, а не отдельной задачей systemd в конце
4. Прогон 1: ansible-playbook site.yml --check --diff --limit staging. Прогон 2 и 3: боевой, затем контрольный — ждём changed=0 в recap
5. ansible-lint --profile production в pre-commit, requirements.yml с точными version: и ansible-galaxy collection install -r requirements.yml в CI

РЕЗУЛЬТАТ

Роль на 25-30 задач приводит новый сервер к корпоративному стандарту за один прогон и даёт объективный отчёт о дрейфе: любое расхождение видно как changed в отчёте регулярного прогона.

КЛЮЧЕВОЙ НЮАНС

Идемпотентность не декларируется, а измеряется. Если второй прогон даёт changed, у задачи нет creates, when или changed_when — правим до нуля, иначе роль нельзя ставить на расписание.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Секреты в Vault и подключение Windows-хостов

Что настраиваем

Смешанный парк: Linux по SSH плюс контроллер домена и RDS по WinRM

Как мы это делаем

1. Секреты в отдельном файле group_vars/all/vault.yml, шифруем ansible-vault encrypt (AES256), пароль отдаём через --vault-id prod@/usr/local/bin/vault-pass.sh
2. Отдельные значения — ansible-vault encrypt_string, чтобы остальные переменные оставались читаемыми в diff и code review
3. Windows: pywinrm[kerberos]>=0.4.0 на control node, слушатель HTTPS на 5986, ansible_winrm_transport=kerberos для доменных учёток; basic и ntlm не используем
4. Windows-задачи только модулями коллекции ansible.windows (win_feature, win_service, win_updates), для доменных операций — коллекция microsoft.ad
5. Проверка: ansible.windows.win_ping проходит по 5986, пароль Vault в CI берётся из ANSIBLE_VAULT_PASSWORD_FILE, сам файл в репозиторий не попадает

РЕЗУЛЬТАТ

Одно дерево плейбуков закрывает и Linux, и Windows-часть парка. Пароли лежат в репозитории в виде AES256-блоков, ротация сводится к ansible-vault rekey и одному коммиту.

КЛЮЧЕВОЙ НЮАНС

WinRM с самоподписанным сертификатом на 5986 — временная мера на стенде. В домене мы сразу выпускаем сертификат из внутреннего CA и работаем транспортом kerberos.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **command/shell вместо модуля**
  Без creates, removes или changed_when задача рапортует changed на каждом прогоне и убивает идемпотентность роли.

- ✕ **Первый прогон сразу в проде**
  Не сделали --check --diff и --limit на одном хосте: ошибка в шаблоне уезжает на весь парк за секунды.

- ✕ **Плавающие версии коллекций**
  requirements.yml без version: сегодня плейбук работает, завтра приезжает мажор коллекции с другим поведением модуля.

- ✕ **host_key_checking=false**
  Отключение проверки ключа хоста ради удобства снимает защиту от подмены цели прогона. Правильно — предзаполнить known_hosts через ssh-keyscan.

- ✕ **Секреты в открытых group_vars**
  Пароли и токены уезжают в Git в чистом виде. Всё чувствительное — только через ansible-vault или lookup из внешнего хранилища.

- ✕ **Дефолтные forks=5**
  На 30 хостах прогон растягивается впятеро. Поднимаем forks и включаем pipelining, иначе автоматизация ощущается медленнее рук.

- ✕ **pipelining при requiretty**
  Включили pipelining, но в sudoers остался requiretty — задачи с become падают. Сначала правим sudoers, потом включаем.

- ✕ **Апгрейд ветки core вслепую**
  2.19 сменил движок шаблонов и ввёл Data Tagging, 2.20 убрал Python 3.11 с control node. Сначала стенд и porting guide, потом прод.

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- ansible-core поддерживаемой ветки на выделенном control node, не на ноутбуке инженера
- Инвентарь и плейбуки в Git, доступ по SSH-ключам, пароли не в репозитории
- Любой прогон в проде начинается с --check --diff и --limit на одном хосте
- Роль baseline: пакеты, NTP, sudoers, локали, ротация логов, агент мониторинга

НОРМАЛЬНО

- Разделение окружений: inventory/prod и inventory/staging с раздельными group_vars
- ansible-vault с --vault-id по окружениям, файл пароля вне репозитория
- ansible-lint --profile production в pre-commit, requirements.yml с точными версиями
- Приёмка каждой роли по правилу двух прогонов: второй даёт changed=0

ХОРОШО

- Еженедельный прогон из CI с отчётом о дрейфе конфигураций по всему парку
- serial и max_fail_percentage на критичных группах, блоки block/rescue для отката
- Кеш фактов jsonfile с fact_caching_connection на диске вместо дефолтного memory
- Тестирование ролей на одноразовых VM или контейнерах перед мержем в main

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Control node вынесен в отдельную VM, а не живёт на рабочем ноутбуке инженера?
- [ ] Инвентарь и все роли лежат в Git, а история изменений привязана к конкретным людям?
- [ ] Каждая роль проходит правило двух прогонов и на втором даёт changed=0?
- [ ] В плейбуках нет ни одного пароля в открытом виде, всё чувствительное в ansible-vault?
- [ ] Версия ansible-core относится к поддерживаемой ветке, а коллекции закреплены в requirements.yml?
- [ ] Есть отдельный staging-инвентарь, где новая роль обкатывается до продакшена?
- [ ] Для критичных групп задан serial, чтобы ошибка не легла разом на все серверы?
- [ ] Сервисная учётка ansible имеет только необходимые sudo-права и отдельный ключ?
- [ ] Windows-хосты подключены по WinRM HTTPS 5986 с kerberos, а не по basic-аутентификации?
- [ ] Кто в компании кроме одного инженера может запустить плейбук и понять его вывод?

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).

- Разворачиваем control node, инвентарь и ansible.cfg под ваш парк за один рабочий день
- Пишем роль baseline под ваш стандарт хоста и доказываем идемпотентность двумя прогонами
- Заводим Vault, разделение prod/staging и pre-commit с ansible-lint в вашем Git
- Подключаем Windows-часть парка по WinRM HTTPS 5986 с сертификатом внутреннего CA
- Обучаем вашего инженера читать вывод, вести роли и безопасно запускать прогоны

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 — 2026)

02 Ansible Configuration Settings: forks, gathering, fact_caching (docs.ansible.com — 2026)

03 ansible.builtin.ssh: ssh_args, ControlPersist, pipelining (docs.ansible.com — 2026)

04 Windows Remote Management: транспорты, порты 5985/5986, pywinrm (docs.ansible.com — 2026)

05 Protecting sensitive data with Ansible Vault: AES256 и vault-id (docs.ansible.com — 2026)

06 Ansible Lint: профили min/basic/moderate/safety/shared/production (docs.ansible.com — 2026)

07 Porting Guides ansible-core 2.19 и 2.20: шаблоны, Data Tagging (docs.ansible.com — 2025)

08 Шаблон ITfresh: inventory-дерево, ansible.cfg и роль baseline (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.