

## Управление конфигурацией (Configuration Management)

Управление конфигурацией (CM) играет критически важную роль в обеспечении стабильности и согласованности в современных IT-инфраструктурах. Это относится не только к обеспечению стандартных конфигураций, но и к минимизации возможных рисков и ошибок.

CM становится неразделимой частью автоматизации и используется для обеспечения однородности настройки ресурсов на протяжении всего жизненного цикла разработки и эксплуатации. Инструменты управления конфигурацией, включая Ansible, Chef, и Puppet, служат мощными средствами для автоматизации задач, связанных с управлением и обслуживанием ИТ-ресурсов, и обеспечивают быстрые, надежные и масштабируемые решения.

Управление конфигурацией позволяет организациям стандартизировать настройки и конфигурации ресурсов и приложений через всю инфраструктуру. Обеспечивает согласованность и устойчивость систем, уменьшает риск ошибок и снижает затраты на обслуживание. С помощью инструментов CM, команды могут автоматизировать развертывание и обновление конфигураций, следить за изменениями и версионировать конфигурационные файлы. Это также улучшает взаимодействие между разработчиками и администраторами, так как обе стороны работают с одним и тем же определением инфраструктуры, которое затем используется для создания и обновления систем.

Так как некоторые инструменты CM имеют степень сложности, команды могут потребоваться инвестировать время в обучение и адаптацию для полного понимания и эффективного использования инструментов. Это может включать в себя разработку документации, обучающих материалов и проведение обучающих сессий для сотрудников.

Инструмент	Как это работает	Архитектура	Язык сценариев	Основные функции	Поддерживаемые платформы	Подход к установке агента
Ansible	Декларативный, Push-модель	Децентрализованная	YAML	Развертывание, оркестровка, управление конфигурациями	Linux, Windows, UNIX	Не требуется
Chef	Императивный, Pull-модель	Централизованная	Ruby	Управление конфигурациями, сбор и анализ данных об узлах	Linux, Windows, UNIX	Требуется

Инструмент	Как это работает	Архитектура	Язык сценариев	Основные функции	Поддерживаемые платформы	Подход к установке агента
Puppet	Декларативный, Pull-модель	Централизованная	Puppet DSL (на основе Ruby)	Автоматизация, управление конфигурациями, отчетность	Linux, Windows, UNIX	Требуется
SaltStack	Императивный и Декларативный, Push и Pull	Гибкая	YAML, Jinja, Python	Оркестровка, управление конфигурациями, автоматизация	Linux, Windows, UNIX	Опционально

Как это работает:

- **Декларативный:** Описывает желаемое состояние системы, не задавая шаги для достижения этого состояния.
- **Императивный:** Описывает последовательность шагов или команд, которые нужно выполнить для достижения желаемого состояния.
- **Push-модель:** Конфигурации отправляются на узлы с центрального сервера.
- **Pull-модель:** Узлы самостоятельно запрашивают конфигурации с центрального сервера.

Архитектура:

- **Централизованная:** Единый центральный сервер управляет конфигурациями всех узлов.
- **Децентрализованная:** Управление конфигурациями может быть распределено по нескольким серверам или быть полностью локальным.
- **Гибкая:** Может использовать как централизованные, так и децентрализованные модели управления.

Основные функции:

- **Развертывание:** Автоматизация процесса размещения приложений и сервисов на серверах.
- **Оркестровка:** Управление и координация множественными задачами и сервисами, работающими на различных узлах и серверах.
- **Управление конфигурациями:** Стандартизация и автоматизация конфигураций систем и приложений.
- **Сбор и анализ данных об узлах:** Сбор данных о состоянии узлов и анализ данных для оптимизации и улучшения.

# Пример

Управление конфигурацией с помощью инструментов, таких как Ansible, позволяет автоматизировать процессы, увеличивать скорость развертывания, улучшать безопасность и стабильность систем. Возьмем простой пример, который иллюстрирует применение Ansible для управления конфигурацией веб-сервера.

## Задача:

Мы хотим автоматизировать развертывание веб-сервера с использованием Apache на нескольких машинах, а также убедиться, что фаервол настроен правильно для разрешения трафика HTTP.

## Ansible Playbook:

```
---
- name: Configure Web Servers
  hosts: webservers
  become: yes
  tasks:
    - name: Install Apache
      package:
        name: apache2
        state: present

    - name: Ensure Apache is running
      service:
        name: apache2
        state: started
        enabled: yes

    - name: Open port 80 for http traffic
      ufw:
        rule: allow
        port: '80'
        proto: tcp
```

## Объяснение:

- **hosts: webserver**s: определяет группу хостов, которые следует настроить. Названия хостов и их IP-адреса определены в инвентарном файле Ansible.
- **become: yes**: обеспечивает повышение привилегий, необходимое для выполнения задач.
- **tasks**: содержат список задач, которые необходимо выполнить на целевых хостах.
  - **Install Apache**: Устанавливает пакет apache2.
  - **Ensure Apache is running**: Убеждается, что служба Apache запущена и будет запускаться при загрузке.
  - **Open port 80 for http traffic**: Открывает порт 80 на фаерволе для трафика HTTP.

## Результат:

С использованием Ansible и данного playbook, мы можем автоматически установить, настроить и запустить Apache на всех серверах, указанных в группе `webserver`s в нашем инвентаре, а также настроить фаервол для разрешения трафика HTTP.

Этот пример демонстрирует, как управление конфигурацией позволяет стандартизировать и автоматизировать настройки систем, уменьшая вероятность ошибок, связанных с человеческим фактором, и обеспечивая более быструю и надежную доставку приложений.

## Не путайте IaC и CM подходы!

IaC фокусируется на предоставлении и управлении инфраструктурой (виртуальные машины, сети, хранилище и так далее) с использованием кода и скриптов.

CM фокусируется на установке и управлении программным обеспечением и настройками на уже существующих системах.

В идеальном мире DevOps, IaC и CM работают вместе. Вы можете использовать IaC для создания и управления инфраструктурой, на которой будет работать ваше приложение, а затем применять CM для установки и настройки самого приложения и его зависимостей. Оба подхода взаимодополняют друг друга, позволяя автоматизировать весь жизненный цикл инфраструктуры и приложений.