

# 自動化とプログラマビリティ

# 学習内容

---

## ネットワーク自動化の核心技術を理解する

- 1 REST APIとは何か
- 2 RESTの4つの設計原則
- 3 REST APIのデータ形式とステータスコード
- 4 構成管理ツールとは
- 5 Ansibleの概要と特徴
- 6 主要ツールの比較とまとめ

# 01

## Chapter 1: REST APIの基礎

# REST APIとは？

---

Webの技術を用いた柔軟なシステム連携の仕組み

Webシステムを外部から操作するためのAPIの一種

**REST** (Representational State Transfer) の設計原則に基づいている

クライアントとサーバが**HTTP**を利用してやり取りを行う

APIは「ソフトウェア同士が共通のルールに従ってやり取りをするための仕組み」と理解すると分かりやすい

# RESTの4つの主要な設計原則

試験で頻出するRESTのコアコンセプト

ハイパーメ  
ディアの活  
用

JSON/XMLなどで表現し、関連データをリンクとして含める

HTTPメソッドによる統一  
的インターフェース

リソース操作をGET/POST/PUT/DELETEに統一

リソースは一意のURIで識別

データ（リソース）をURIで特定（例：  
/users/123）

ステートレスなクライアント/サーバモデル

サーバは過去の通信状態を保持しない（各リクエストが完結）

# CRUD操作とHTTPメソッドの対応

リソースに対する操作を一貫して行うためのマッピング

## CRUD操作

Create

作成

Read

読み取り

Update

更新

Delete

削除

## HTTPメソッド

POST / PUT

GET

PUT / PATCH

DELETE

# REST APIにおける主要なデータ形式

クライアントとサーバ間の情報伝達フォーマット

## JSON (JavaScript Object Notation)

軽量で人間にも読みやすい。  
記述が簡潔で、Cisco SDNコントローラでも多用される。

## XML (Extensible Markup Language)

タグを用いてデータの構造を明確に表現。自由度が高く、情報交換に広く使われる。

## YAML (YAML Ain't Markup Language)

インデントで階層を表現。シンプルで人間に読みやすく、**Ansible**などの設定ファイルに利用される。

02

## Chapter 2: 構成管理ツールの活用



# 構成管理ツール導入のメリット

---

## 効率化と安定運用を実現するアプローチ

### 設定の自動化と効率化

複数機器へ一括で設定を反映し、工数を大幅に削減する

### 構成ドリフトの防止

人為的ミスによる設定のズレ（ドリフト）を防ぎ、正しい構成を維持する

### 統一的な運用

大規模環境でも設定の標準化を容易に行える

# Ansibleの主な特長

---

ネットワーク自動化で最も注目されるツール

STEP 1

エージェントレス  
(管理対象機器に専用ソフトが不要)

STEP 2

豊富なモジュール  
(ネットワーク、クラウドなど幅広い自動化に対応)

STEP 3

簡単な記述方式  
(設定ファイルは**YAML**形式で分かりやすい)

# 主要な構成管理ツールの比較

Ansible、Puppet、Chefのアーキテクチャの違い

比較項目	Ansible	Puppet	Chef
開発組織	RedHat	Puppet Labs	Chef Software
アーキテクチャ	エージェントレス	エージェント	エージェント
管理方式	Push型（サーバから配布）	Pull型（対象機器が取得）	Pull型
記述言語	YAML	独自DSL	Ruby