# **63. Ansible, Puppet, and Chef, Terraform**

## **Configuration Drift**

Configuration drift occurs when individual changes made over time cause a device’s configuration to deviate from the standard or correct configurations defined by the company.

* While each device has unique configuration elements (e.g., IP addresses, hostname), most configurations should follow standardized templates designed by network architects/engineers.
* Changes made by engineers (e.g., troubleshooting, testing) can cause configurations to drift from the standard.
* These changes are often undocumented, leading to future issues.

**Without automation tools, best practices include:**

* Saving the configuration as a text file in a shared folder whenever a change is made.
* Using a standard naming convention like ***hostname\_yyyymmdd***.
* Recognizing flaws in this approach:
  + Engineers may forget to update the shared folder.
  + It’s unclear which saved configuration is the “correct” one.
  + Even properly saved configurations might not match the company standard.

## **Configuration Provisioning**

Configuration provisioning refers to how configuration changes are applied to devices, including setting up new devices.

* Traditionally done manually via SSH, which is impractical for large networks.
* Configuration management tools like **Ansible, Puppet, and Chef** enable efficient large-scale changes.

### **Two Key Components of Configuration Provisioning:**

1. **Templates** – Standardized configuration frameworks.
2. **Variables** – Device-specific settings within templates.

## **Introduction to Configuration Management Tools**

Configuration management tools facilitate centralized control over large-scale network configurations.

* **CCNA Relevant Tools**: Ansible, Puppet, and Chef.
* Originally designed for server automation in virtualized environments but now widely used for networking.

### **Use Cases:**

✅ Automating large-scale configuration deployment for new devices.  
 ✅ Applying configuration changes to all or specific subsets of devices.  
 ✅ Ensuring configuration compliance with company standards.  
 ✅ Comparing configurations across devices and versions.

## **Ansible**

* **Owned by**: Red Hat
* **Written in**: Python
* **Model**: *Push* model (Control node pushes configurations via SSH)
* **Agentless**: No software needed on managed devices

### **How Ansible Works:**

* Uses SSH to connect, extract info, and apply configurations.
* Requires creating several text files:

| **File Type** | **Description** | **Format** |
| --- | --- | --- |
| **Playbooks** | Define automation tasks | YAML |
| **Inventory** | List managed devices & roles | INI/YAML |
| **Templates** | Base configuration files with placeholders | Jinja2 |
| **Variables** | Stores values for placeholders in templates | YAML |

## **Puppet**

* **Written in**: Ruby
* **Model**: *Pull* model (Clients pull configurations from the Puppet master)
* **Requires an agent**: Most devices need Puppet software installed (some Cisco devices may not support this).
* **Communication**: Uses **TCP 8140** for client-server interactions.
* Can be run *agentless* using a proxy agent via SSH.

### **Key Text Files in Puppet:**

| **File Type** | **Description** | **Format** |
| --- | --- | --- |
| **Manifest** | Defines desired configuration | Proprietary Puppet language |
| **Templates** | Used to generate manifests | Proprietary Puppet language |

## **Chef**

* **Written in**: Ruby
* **Model**: *Pull* model
* **Requires an agent**: Managed devices need Chef software installed (not all Cisco devices support this).
* **Communication**: Uses **TCP 10002** for client-server interactions.
* Uses a **Domain-Specific Language (DSL)** based on Ruby.

### **Key Text Files in Chef:**

| **File Type** | **Description** | **Format** |
| --- | --- | --- |
| **Resources** | Individual config elements | Ruby DSL |
| **Recipes** | Define automation tasks | Ruby DSL |
| **Cookbooks** | Collection of related recipes | Ruby DSL |
| **Run-list** | Ordered list of recipes to apply | Ruby DSL |

## **Comparison Chart for CCNA**

**Memorize this chart for the CCNA:**

| **Tool** | **Written In** | **Model** | **Requires Agent?** | **Communication Method** |
| --- | --- | --- | --- | --- |
| **Ansible** | Python | Push | ❌ No | SSH |
| **Puppet** | Ruby | Pull | ✅ Yes (or agentless with proxy) | TCP 8140 |
| **Chef** | Ruby | Pull | ✅ Yes | TCP 10002 |

