Configuration Management Tools

- Configuration management systems are designed to make controlling large numbers of devices easy for administrators and operations teams.
- They allow you to control many different systems in an automated way from one central location.
- Popular options (open source and free, with paid for Enterprise editions available):
  - Ansible
  - Puppet
  - Chef
  - Terraform (free Business Source License)

Configuration Management Tool Benefits

Configuration Management Tools:

- Can automate the provisioning and deployment of servers and network devices
- Require little knowledge of programming
- Have established development practices including version control and testing

Ansible

- Can be run from any machine with Python 2 or Python 3 installed
- Agentless
- Push model
- Communicates over SSH by default
- Simpler than most other tools
- Originally released in 2012

Ansible (Cont.)

- Ansible **modules** are pre-built Python scripts.
- Many pre-built network modules exist.
- Ansible **inventory** files define all hosts that will be managed by the control workstation.
- Ansible **playbooks** are YAML files that outline the instructions it needs to run.
- **ansible.cfg** is Ansible's default configuration file

Puppet

- Typically uses an agent on the target devices
- ‘Puppet Master’ runs on Linux server
- Pull model, agent checks in every 30 mins by default
- Written in Ruby
- Uses proprietary DSL rather than YAML
- A ‘Manifest’ defines the device’s properties
- It can check configuration consistency
- Created in 2005

Chef

- An agent must be installed on the target devices
- Pull model
- Written in Ruby
- Terminology is Cook Book > Recipe
- Released in 2009

Terraform

- Terraform is an infrastructure-as-code (IaC) software tool from HashiCorp.
- Administrators provision infrastructure using HashiCorp Configuration Language (HCL) or JSON
- Primarily designed for public and private cloud, can also work with on-premises platforms
- Written in Go, it can be installed on Linux, Windows or Mac
- Communicates over REST APIs, NETCONF, SSH, SNMP
- Agentless
- Push model
- Released in 2014

Terraform Components

- **Configuration files** define the final desired state using ‘resource blocks’. Resources are virtual machines, network devices etc.
- The **State file** provides lifecycle management by tracking the deployed resources.
- **Providers** are plug-ins that interface with platforms such as AWS or IOS-XE, define their available resources and the API calls to manage them.

How Terraform Works

(Code): Administrator creates the configuration files.

1) **Plan:** Terraform checks the differences between the current state in the state file and the desired state in the configuration files, then determines the necessary actions to apply. Administrator checks the actions.
2) **Apply:** The actions are executed.
3) **Consolidate:** The state file is updated.

Configuration Management Tool Support

- Ansible, Puppet, Chef and Terraform were designed primarily for server system administration
- Ansible and Terraform are more suitable for network environments than Puppet and Chef because they do not require an agent. They are also simpler to learn and use
- There is limited support on Cisco devices to run agents

Ansible vs Terraform

- Terraform's primary focus is as an infrastructure provisioning tool.
- It uses a 'declarative' approach where a desired state is defined and the tool takes actions to achieve it.
- It is lifecycle aware.
- Ansible's primary focus is as a configuration tool.
- It uses a 'procedural' approach with a granular sequence of steps that are followed to reach the desired end result.
- It is not lifecycle aware.

Ansible vs Terraform: Mutable vs Immutable

- A mutable object can be changed after it's created, an immutable object cannot. To make changes to an immutable object it must be destroyed then replaced with a new version.
- Ansible is considered mutable, Terraform immutable by default.
- (Both tools can conduct mutable operations and have immutable elements.)

Ansible vs Terraform

- Ansible and Terraform can be used together, for example:
- Use Terraform to provision an infrastructure then call Ansible to push configurations to the resources. Use Ansible for future configuration changes
- Or call Terraform in the first step of an Ansible playbook to provision the infrastructure then continue with configuration from there