63. ANSIBLE, PUPPET, AND CHEF

CONFIGURATION DRIFT

- CONFIGURATION DRIFT is when individual changes made over time causes a device's configuration to deviate from the standard / correct configurations as defined by the company
  - Although each device will have unique parts of its configurations (IP Addresses, hostname, etc) most of a device's configuration is usually defined in standard templates designed by the network architects / engineers of the company
  - As individual engineers make changes to devices (for example, to troubleshoot and fix network issues, test configurations, etc), the configuration of a device can drift away from the standard.
  - Records of these individual changes and their reasons aren't kept
  - This can lead to future issues
- Even without automation tools, it is best to have standard configuration management practices.
  - When a change is made, save the config as a text file and place it in a shared folder
    - A standard naming system like (*hostname_yyyymmdd)* might be used.
    - There are flaws to this system, as an engineer might forget to place the new config in the folder after making changes. Which one should be considered the "CORRECT" config?
    -

      Even if configurations are properly saved like this, it doesn't guarantee that the configurations actually match the standard

CONFIGURATION PROVISIONING

- CONFIGURATION PROVISIONING refers to how configuration changes are applied to devices
  - This includes configuring new devices, too
- Traditionally, configuration provisioning is done by connecting to devices one-by-one via SSH
  - This is not practical in large networks
- Configuration management tools like Ansible, Puppet, and Chef allow us to make changes to devices on a mass scale with a fraction of time and effort.
- TWO ESSENTIAL COMPONENTS:

  - Templates
  - Variables

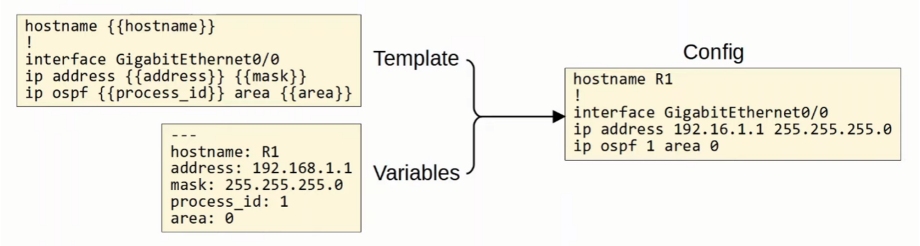


Figure 1: image

INTRO TO CONFIGURATION MANAGEMENT TOOLS

- CONFIGURATION MANAGEMENT TOOLS are network automation tools that facilitate the centralized control of large numbers of network devices
- The option you need to be aware of for the CCNA are Ansible, Puppet, and Chef
- These tools were originally developed after the rise of VMs, to enable server system admins to automate the process of creating, configuring, and removing VMs
  - However, they are also widely used to manage network devices
- These tools can be used to perform tasks such as :
  - Generate configurations for new devices on a large scale
  - Perform configuration changes on devices (all devices in your network, or certain subset of devices)
  - Check device configurations for compliance with defined standards
  - Compare configurations between devices, and between different versions of configurations on the same device

Figure 2: image

ANSIBLE

- ANSIBLE is a configuration management tool owned by Red Hat

- Ansible itself is written in Python
- Ansible is *agentless*
  - It doesn't require any special software to run on the managed devices
- Ansible uses SSH to connect to devices, make configuration changes, extract info, etc
- Ansible uses a *push* model. The Ansible server (Control node) uses SSH to connect to managed devices and *push* configuration changes to them
  - Puppet and Chef use a *pull* model
- After installing Ansible itself, you must create several text files:
  - PLAYBOOKS :
    - These files are "blueprints of automation tasks"
    - They outline the logic and actions of the tasks that Ansible should do
    - Written in YAML
  - INVENTORY :
    - These files list the devices that will be managed by Ansible, as well as characteristics of each device such as their device role (Access Switch, Core Switch, WAN Router, Firewall, etc.)
    - Written in INI, YAML, or other formats
  - TEMPLATES :
    - These files represent a device's configuration file, but specific values for variables are not provided.
    - Written in JINJA2 format
  - VARIABLES :
    - These files list variables and their values.
    - These values are substituted into the templates to create complete configuration files.
    - Written in YAML

PUPPET

- PUPPET is a configuration management tool written in RUBY
- Puppet is typically agent-based
  - Specific software must be installed on the managed devices
  - Not all Cisco devices support a Puppet agent
- It CAN be run *agentless,* in which a proxy agent runs on an external host, and a proxy agent uses SSH to connect to the managed devices and communicate with them
- The Puppet server is called the "Puppet master"
- Puppet uses a PULL model (clients "pull" configurations from the Puppet master)
  - Clients use TCP 8140 to communicate with the Puppet master
- Instead of YAML, it uses a proprietary language for files
- Text files required on the Puppet master include:
  - MANIFEST :

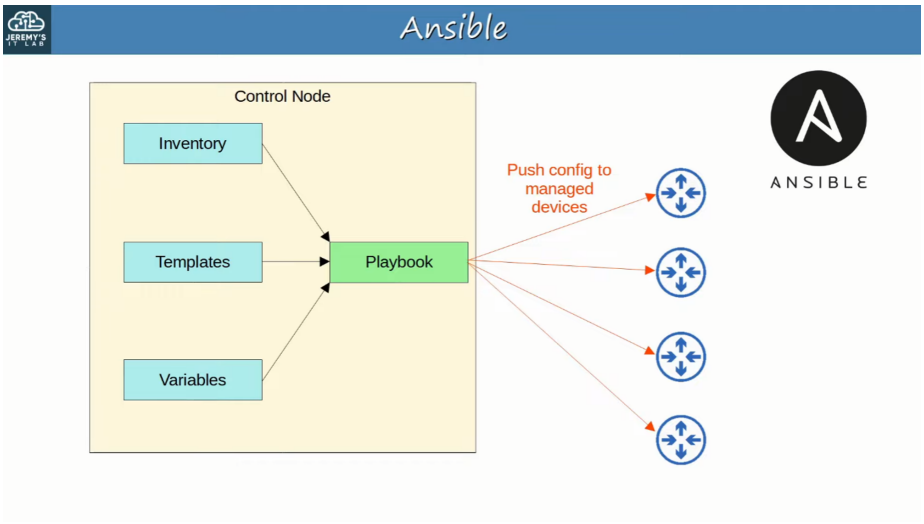


Figure 3: image

    - The file defines the desired configuration state of a network device
  - TEMPLATES :
    - Similar to Ansible templates.
    - Used to generate MANIFESTS

CHEF

- CHEF is a configuration management tool written in RUBY
- CHEF is Agent-Based
  - Specific software must be installed on the managed devices
  - Not all Cisco devices support a CHEF agent
- CHEF uses a PULL model
- The server uses TCP 10002 to send configurations to clients
- Files use a DSL (Domain-Specific Language) based on Ruby
- Text files used by CHEF include:
  - RESOURCES :
    - The “ingredients” in a RECIPE.
    - Configuration objects managed by CHEF
  - RECIPES :
    - The “recipes” in a COOKBOOK.
    - Outlines the logic and actions of the tasks performed on the resources
  - COOKBOOKS :
    - A set of related RECIPES grouped together
  - RUN-LIST :

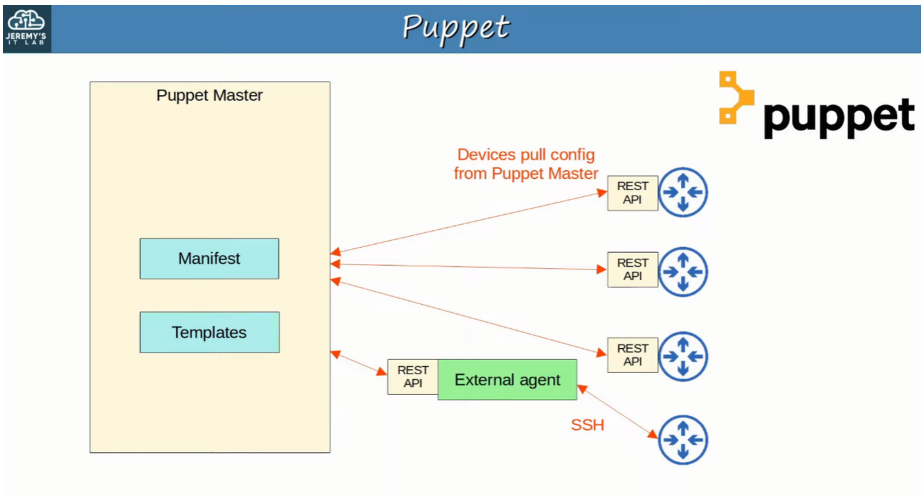


Figure 4: image

* An ordered list of RECIPES that are run to bring a device to the desired configuration state

---

MEMORIZE THIS CHART FOR THE CCNA

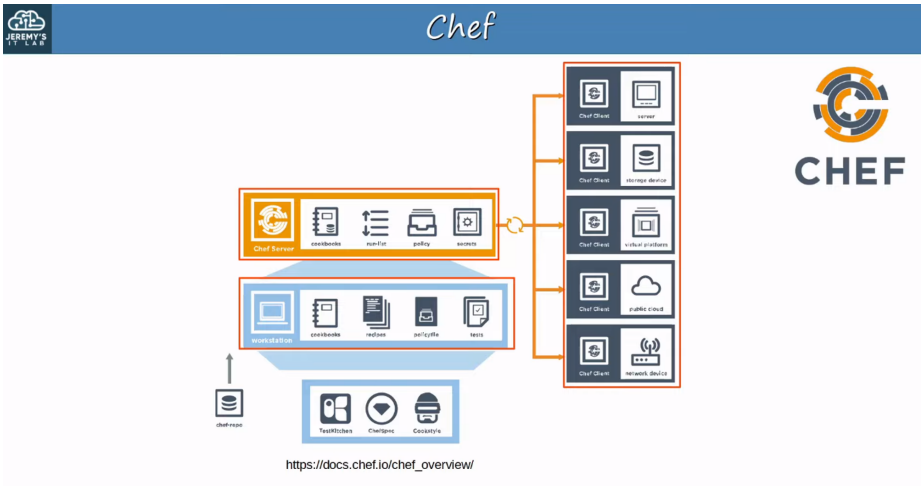
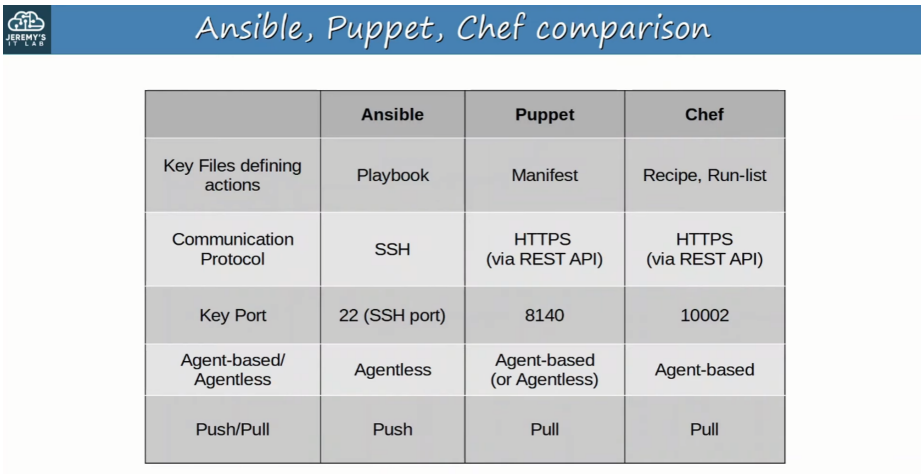


Figure 5: image



	Ansible	Puppet	Chef
Key Files defining actions	Playbook	Manifest	Recipe, Run-list
Communication Protocol	SSH	HTTPS (via REST API)	HTTPS (via REST API)
Key Port	22 (SSH port)	8140	10002
Agent-based/ Agentless	Agentless	Agent-based (or Agentless)	Agent-based
Push/Pull	Push	Pull	Pull

Figure 6: image