<https://www.redhat.com/en/topics/automation/what-is-configuration-management>

<https://searchitoperations.techtarget.com/definition/configuration-management-CM>

<https://www.liquidweb.com/kb/puppet-salt-chef-ansible-a-comparison/>

* Configuration management is important because undocumented changes across many systems and applications can cause instability and downtime and can negatively impact business operations and security
* Configuration management is a process for maintaining computer systems, servers, and software in a desired, consistent state
* Configuration management is a way to ensure that systems perform as they should as changes are made over time
* Without configuration management, system administrators and software developers could end up not knowing what is on a server or which software has been updated
* Configuration management helps users and administrators know where certain services exist and what the current state of applications are
* Configuration management tools help enforce a desired configuration state and provides alerts for configuration problems
* Helps system administrators and software developers understand how a change to one configuration item will affect other items

**Puppet**

* Offers an open-source edition and an enterprise edition
* Puppet master server can only be installed on Unix/Linux systems
* Puppet deployment consists of a master server and client machines known as Agents
* Uses its own language based on Ruby
* Puppet takes manifest files and compiles them into catalogs and pushes each catalog to its designated node
* The node is reconfigured to the desired state
* Puppet offers a web-based UI
* If the master server goes down, Puppet will replace it with a new master server
* Clients check the master for updated manifests and then pull new configurations down from the master server
* Steeper learning curve due to required knowledge of the Puppet DSL and Ruby programming languages

**SaltStack**

* Designed to allow low-latency, high-speed communication and data transmission between nodes for remote execution
* Designed to work with Unix/Linux and Windows, but the master server can only work on Unix/Linux machines
* SaltStack is composed of a Salt Master and clients called Salt Minions
* Salt Minions run as agents on each node machine
* Salt human-readable configurations use Python and YAML files
* The Salt Master pushes configurations to client machines
* Salt serves as an asynchronous file server, which increases the speed of file transfers that serve the Salt Minions
* Salt can run in a multi-master configuration
* If a master server goes down, agents will connect to another master listed in the configs
* Salt allows parallel execution of multiple commands at once
* Commands are encrypted via AES and pushed to client nodes via SSH protocol
* Salt can manage multiple masters
* Offers a web UI but has limited capabilities and features

**Chef**

* Has a large support community with extensive documentation
* Master and node software work on Unix/Linux systems but not Windows
* Client and workstation versions can be deployed on Windows servers
* Configuration options consist of Cookbooks and Recipes
* Recipes are definition files that can be combined with attributes, files, libraries, and other recipes to build Cookbooks
* Cookbooks can then be used for client deployment
* Chef consists of 3 main components: Chef Workstation, Chef Master Server, and Chef Client
* Chef Workstation is used to create, test, and deploy cookbooks to the master servers
* Chef Master Servers are where configuration data is stored, such as cookbooks, server data, and other relevant information
* Chef Client is the end-node machines managed by master servers that pull and execute cookbook configurations from the master server
* Chef uses a DSL (Domain Specific Language) like Puppet, but supports scripts written in Ruby
* Chef is set up with a backup master server which will take over if the master server goes down
* Chefs infrastructure is very stable and built for reliability
* Chef offers sequential execution order

**Ansible**

* Ansible supports both Windows and Unix/Linux client machines, but requires a Unix/Linux based server
* Only an Ansible master is necessary to run because it uses SSH (or RDP) protocols to open a connection to client servers
* Configuration files are Python based and use YAML files for structured data
* Configuration files are called Playbooks
* Supports a Python API
* Ansible can also be used via a CLI to perform simple tasks such as restarting a server
* For more complex tasks, a Playbook is necessary
* Usually configured as a single active master server, but a secondary master can be configured to take over if the master server goes down
* Because Ansible uses an agent-less approach, it can deploy changes or push updates quickly to all nodes
* Clients do not check for periodic config changes on the master server