Ansible, Puppet and Chef

Configuration Drift: when individual changes made over time cause a device's configuration to deviate from the standard/correct configurations as defined by the company, most of a device's configuration is usually defined in standard templates designed by network architects/engineers, it's best to have standard configuration management practices as saving the config files as text files in a shared folder, however this doesn't guarantee that the configurations actually match the standard.
Configuration provisioning refers to how configuration changes are applied to devices, this includes configuring new devices too, traditionally this is done by connecting devices one by one via SSH, configuration management tools allow changes to be made to devices on a mass scale using two essential components: templates and variables.
Configuration management tools were originally designed to enable system admins to automate the process of creating, configuring and removing VMs, they can generate configurations for new devices on a large scale, perform configuration changes on devices, check for compliance with defined standards and compare different devices' configurations and different versions of configurations on the same device.

Ansible is owned by red hat, written in python, agentless (doesn't require any special software to run on managed devices, uses SSH to connect to devices, a push model (configurations are pushed to devices through SSH), several text files must be created as playbooks: blueprints of automation tasks, they outline the logic and actions of tasks Ansible should do (YAML).
Inventory: List devices managed by Ansible as well as characteristics of each device as role (access,WAN router,.. etc), (YAML, INI or other formats).
Templates: represent a device's configuration file with no specific values written in Jinja2 format.
Ansible server is called a control node.

Puppet:written in Ruy, agent-based (specific software must be installed on managed devices), it can run agent less, where a proxy agent runs on an external host and the proxy agent uses SSH to connect to the managed devices and communicate with them., uses a pull model (clients pull configs from the puppet master), client use TCP 8140 to communicate,
a proprietary language for files is used, two text files are used: manifest → it defines the desired configuration state of a network device, templates → similar to ansible templates used to generate manifests, puppet agent provides a REST API to communicate with the device using HTTPs.

Chef: written in Ruby, agent-based, uses a pull model, server uses TCP 10002 to send configs to clients, files use a DSL (Domain-Specific language) based on Ruby.
Text files include: Resources: the configuration objects manged by Chef.
Recipes: in a cookbook outlines the logic and actions of tasks performed on the resources, cookbooks: a set of related recipes, run-list: an ordered list of recipes that are run to bring a device to the desired configuration state.
Use HTTPs via REST APIs.
All of them use a client server model.