> “The first rule of any technology used in a business is that automation applied to an efficient operation will magnify the efficiency. The second is that automation applied to an inefficient operation will magnify the inefficiency.”
> *Bill Gates* (qtd. in Stancombe, 2015)

# Introduction

This report will introduce automated network concepts and considerations in relation to the current network infrastructure at ACME.com. Currently, ACME.com has four locations in New Zealand including the head office, and twelve in Australia. Each location relies heavily on the network infrastructure, which comprises of ten servers, fifty one routers, and ninety switches. This hardware is currently managed manually from ticket tracking, documentation through to performing configuration changes on the basis of the site, device or user requirement, through the command line. This approach makes it difficult to gain an accurate overview of the current network state, performance, and security. Automated network configuration tools enable efficient management of the network infrastructure, maximising its uptime, while ensuring it is resilient and secure.

# Automated Network Management

Automation is intended to enable people to focus on higher level, complex tasks by autonomously completing frequently performed, time-consuming tasks, then, monitoring/enforcing the configuration, and reporting the results (RedHat Inc, 2020). Automated network configuration software enables network administrators to describe the desired state of target devices as version controlled code (Visualpath IT, 2020).

# Advantages of Automated Network Configuration Software

1. **Simplification**: common, day to day network management tasks are consistently performed across the network
2. **Enforcement and fast failure recovery**: infrastructure code can be redeployed to return devices back to a predefined configuration/state, restoring the network with less disruption.
3. **Maintenance and scalability**: Deployment of configuration changes can scheduled to occur during low peak traffic; per time zone, network, specific subnets or VLANs.
4. **Patch management**: will allow for the management of software patches, and will allow for a incremental installation of patches on the network. This allows administrators to deploy changes to the network without disruption to services.
5. **Access and security management**: firewalls, access-lists, IP addressing is managed centrally, and is distributed where needed.
6. **Agile network management and collaboration**: network administrators are able respond to to develop, review, test and monitor network configuration changes.
7. **Monitoring and reporting**: Alert administrators of hardware issues, monitor configuration changes, report failures, unsecure or devices that are not in compliance with the standards.
8. **Version control and backup:** infrastructure code is version controlled and backed up, allowing quicker restoration of the network in the event of a natural disaster, outage, or total loss of network.
9. **Reduced network management costs**: reduction of overtime hours and time troubleshooting manual configuration issues.
10. **Performance oversight and capacity planning**: performance reporting capabilities ensure resources are utilised more efficiently, targeting areas of the network which require investment.

# Automated Network Configuration Software Concepts

## *Installation*

There are 3 System automation tools typically requires an agent to be installed on each device to be managed. Network devices are limited in that agent software cannot be installed directly, .

1. **Agent-based configuration:** With agent-based tools, an agent must be installed on every device that the configuration management tool will manage. (Gargano, 2020)
2. **Agentless configuration:** Agentless tools do not require that an agent be installed on every device; instead, they communicate via SSH or another API that a device supports. (Gargano, 2020)
3. **Proxy-agent configuration:** This type of configuration does not require an agent on every device, but it does require some type of "process" or "worker" to communicate with the master server and the remote device. (Gargano, 2020)

## *Configuration*

These technologies typically use one of two forms of language to structure configurations. (DSL) Domain-Specific Language, is specific to the platform used to configure the network. This has a steep learning curve for the network administrator to adopt the technology. (SML) Structure Mark-up Language, easily understood languages and mark-up, such as YAML. This enables network administrators and non-programmers to adopt the technology much faster, and with less errors (Gargano).

## *Communication*

1. Push Model:
2. Pull Model:
3. Convergence: only changing configurations needed to align the target with the state defined in the configuration files (Estevão & Miranda, 2020).
4. Idempotency: checking the current state of the target; if it is already in the requried state no change occurs (Estevão & Miranda, 2020).

# Automated Network Configuration Software Options

## *Ansible*

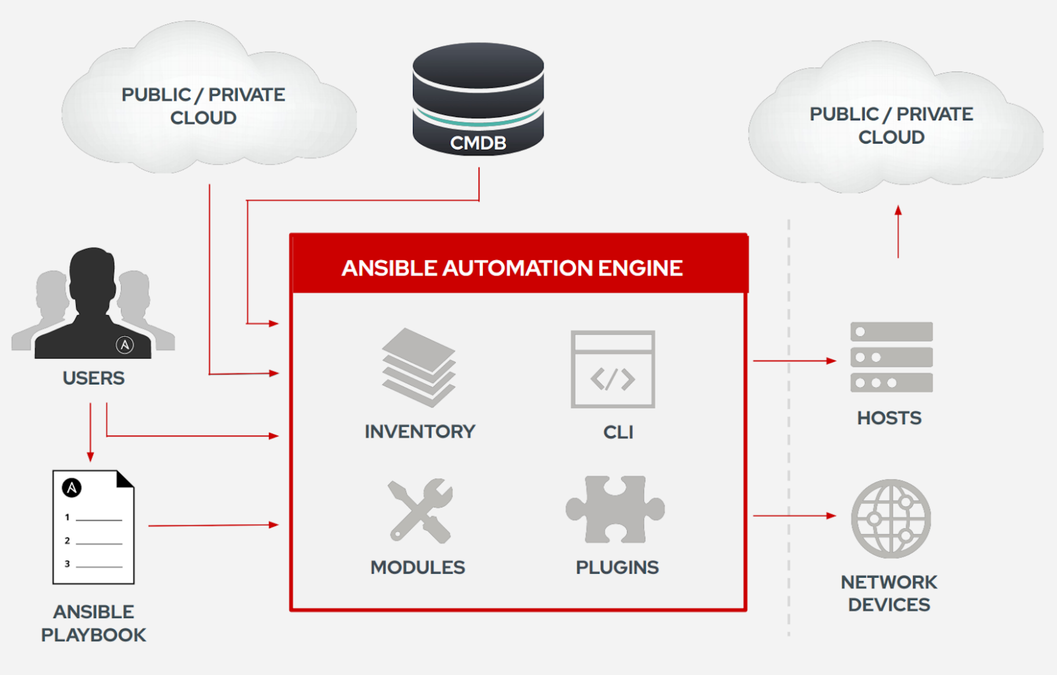
To set up Ansible, you will want to designate a single node as your control node. In reality, any of your nodes can be the control node. There is no need to set up client software on your other nodes. Create an SSH key pair on your control node, then copy it onto the rest of your nodes. Create an inventory file for your Ansible nodes. Typically, this goes into /etc/ansible/hosts on Linux OSes like Red Hat Linux, Ubuntu, and Debian. Ansible will use SSH connections to your controlled nodes to run your configuration management playbooks.

It is an automation platform that is capable of deployment of configuration, monitoring and security management. It works with python programming language and YAML data format. Ansible is an agentless tool, which means there is no software or agent that needs to be installed on the client machines. Being agentless allows the user to push configuration to one of the network devices on the network.

Ansible uses SSH as a remote communication protocol. It uses push models to get the configurations from network devices. Push model means no agent software on the nodes. Since it is agentless, any device can be an Ansible Controller on the network.

Ansible does not require agent software to be installed on target nodes.

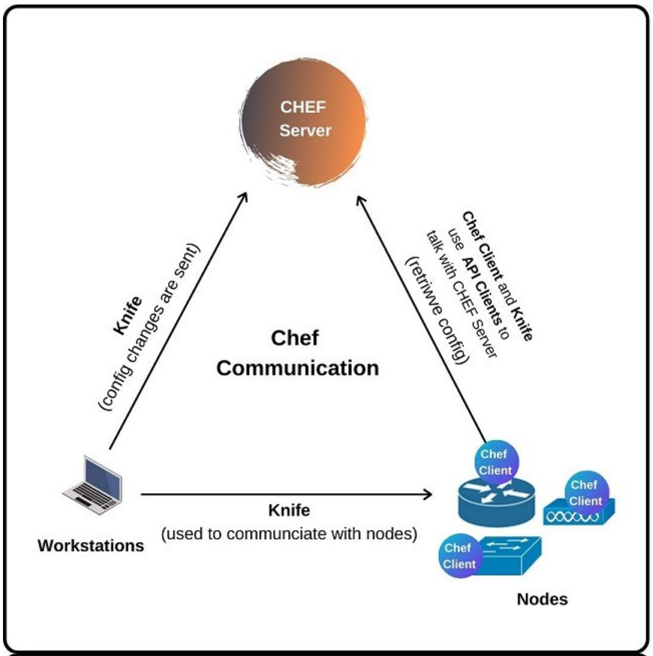
- **Network:** A10, Arista, Aruba, Cumulus, Bigswitch, CISCO, Dell, Extreme, F5, Lenovo, MikroTik, Juniper, OpenSwitch and others
- **Security:** Checkpoint, Cisco, Cyberark, F5, Fortinet, Juniper, IBM, Palo Alto, Snort and others
- **Monitoring:** LogicMonitor, New Relic, Sensu, Coralogix, and others
- **DevOps:** Jira, GitHub, Vagrant, Jenkins, Slack, and others
- **Cloud:** AWS, Azure, Digital Ocean, Google, OpenStack, Rackspace, and others
- **Operating Systems:** RHEL, Linux Flavors, MS Windows, and others

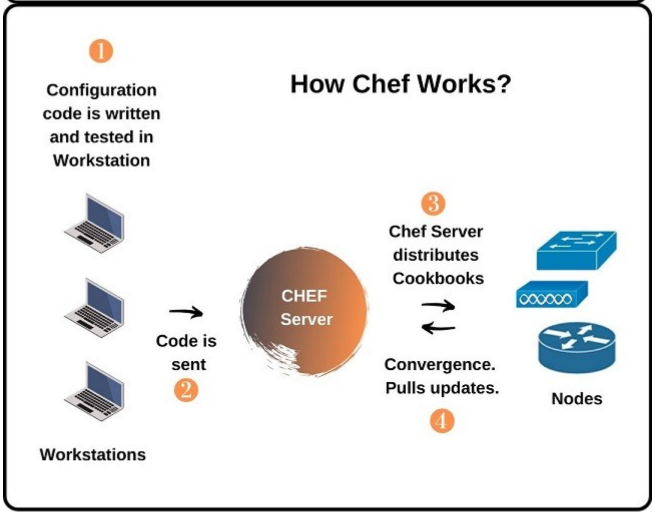


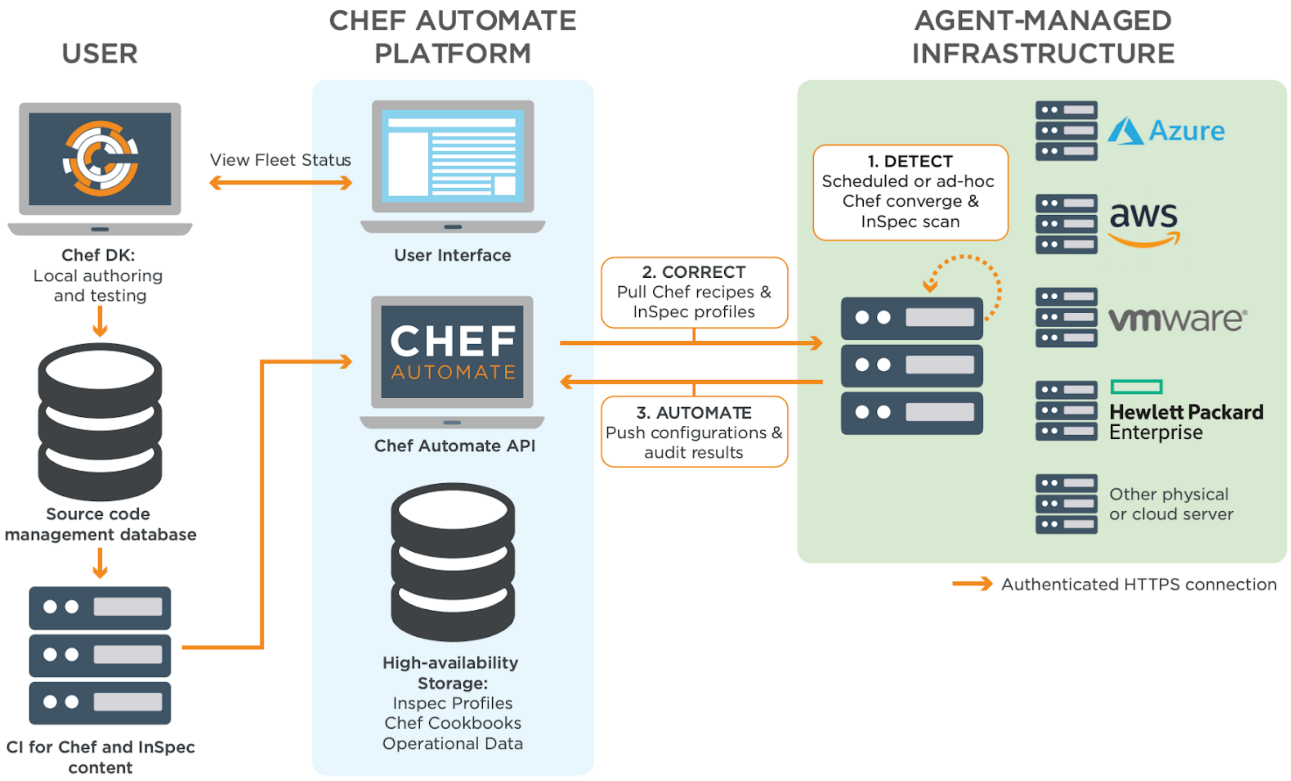
- **Playbook:** Written in YAML, *playbooks* are a set of one or more tasks that are executed in sequence. Tasks execute an Ansible module. In cases where the resource being managed does not have Python installed or no API is available, shell commands can be executed through the raw module. The “playbook” name is an analogy to a Rugby game, where the list of plays that you execute during a match is called a playbook.
- **Module:** Can be written in Python, PowerShell (for Windows resources), and any language that is able to generate a JSON formatted output. They are the core of the Ansible stack. They can be part of the main Ansible code base or external.
- **Inventory:** List of the resources in your infrastructure that will be managed with Ansible. Ansible allows static inventory files or dynamic inventories, through the use of scripts and inventory plugins. Dynamic inventories are generated at execution time from sources like a Configuration Management Database (CMDB), Satellite (or Foreman) managed hosts, and VMware vCenter virtual machines and hosts.
- **AD-HOC:** The execution of ad-hoc commands (through the **ansible cli** command) is a feature for remotely executing simple tasks (using modules) without having a playbook.
- **Plugins:** “Plugins are pieces of code that augment Ansible’s core functionality. Ansible uses a plugin architecture to enable a rich, flexible and expandable feature set” like mail, Slack notifications, enhancing inventories, and sending events to Foreman, Grafana, Logstash or Jabber.

# *Chef*

It is written by Ruby DSL (Domain Specific Language).


CHEF Server
Knife (config changes are sent)
Chef Communication
Chef Client and Knife use API Clients to talk with CHEF Server (retriwve config)
Workstations
Knife
(used to communciate with nodes)
Chef Client
Chef Client
Chef Client
Nodes


1
Configuration code is written and tested in Workstation
How Chef Works?
Workstations
Code is sent
2
CHEF Server
3
Chef Server distributes Cookbooks
Convergence. Pulls updates.
4
Nodes


USER
CHEF AUTOMATE PLATFORM
AGENT-MANAGED INFRASTRUCTURE
View Fleet Status
Chef DK:
Local authoring and testing
Source code management database
CI for Chef and InSpec content
User Interface
CHEF AUTOMATE
Chef Automate API
High-availability Storage:
Inspec Profiles
Chef Cookbooks
Operational Data
1. DETECT
Scheduled or ad-hoc Chef converge & InSpec scan
2. CORRECT
Pull Chef recipes & InSpec profiles
3. AUTOMATE
Push configurations & audit results
Azure
aws
vmware
Hewlett Packard Enterprise
Other physical or cloud server
Authenticated HTTPS connection

### *Puppet*

## References