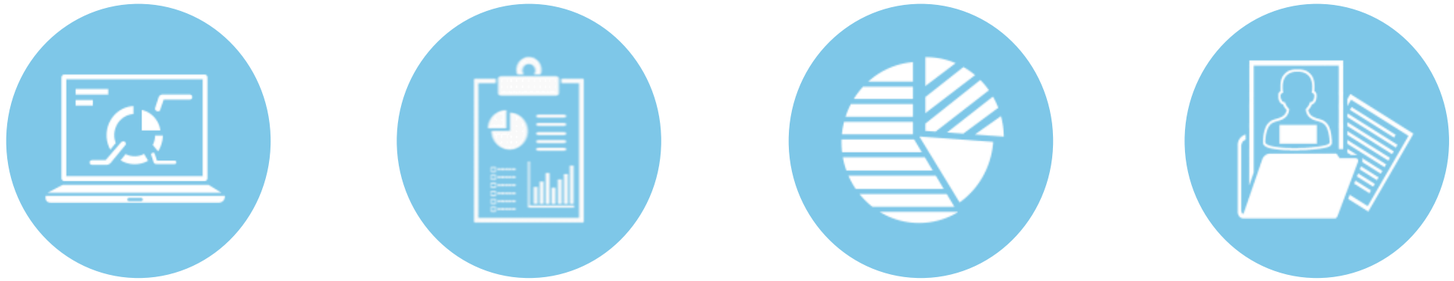
DevOps Certification Training

Lesson 05: Configuration Management Tools



Learning Objectives

By the end of this lesson, you will be able to:

- Explain the concepts of configuration management tools
- Demonstrate Puppet installation
- Demonstrate Chef installation
- Demonstrate Ansible installation and its usage
- Describe SaltStack
- Select the suitable CM tool for your organization

Configuration Management Tools

Overview of Configuration Management Tools

Configuration Management Tools

- Configuration management tools manage all configuration items in a software for all environments.
- These configuration items can be software application files, software packages, and software installations which need to be configured for specific environments.
- Configuration management tools cover both, software and server configurations.
- They help enforce standardization in software configurations without any errors.
- They also help to reduce the time taken to manage configurations manually on each and every server.

Popular Configuration Management Tools

Purpose of Configuration Management Tool

1. Helps to manage software configurations effectively
2. Helps to remove manual errors while performing configuration changes for application software
3. Helps to manage configurations in multiple environments
4. Generic configurations can be stored in version control so that they can be used in other projects as well
5. Saves time by automating generic configuration procedures
6. CM tools can be used to deploy application source codes across vast infrastructure
7. Eliminates the need of documentation process for performing configuration changes

Configuration Management Process

Configuration Regulation
Regulates the way configuration changes are made to application software

Configuration Identification
Identifies the correct configuration that needs to be managed by CM tool

Configuration Compliance
Audits and implements compliance on configuration changes made to application software

Configuration Management Tools

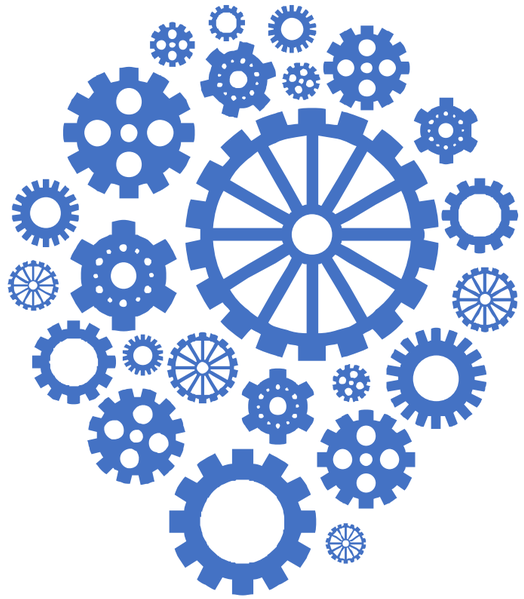
Managing Infrastructure

Role of Infrastructure as Code in DevOps Environment

Modern way to manage configuration items

Documentation for software modifications and infrastructure configurations

Admins can manage multiple environments with infrastructure scripts/codes



Considered an essential component of DevOps

Easy to integrate with version control and share with others

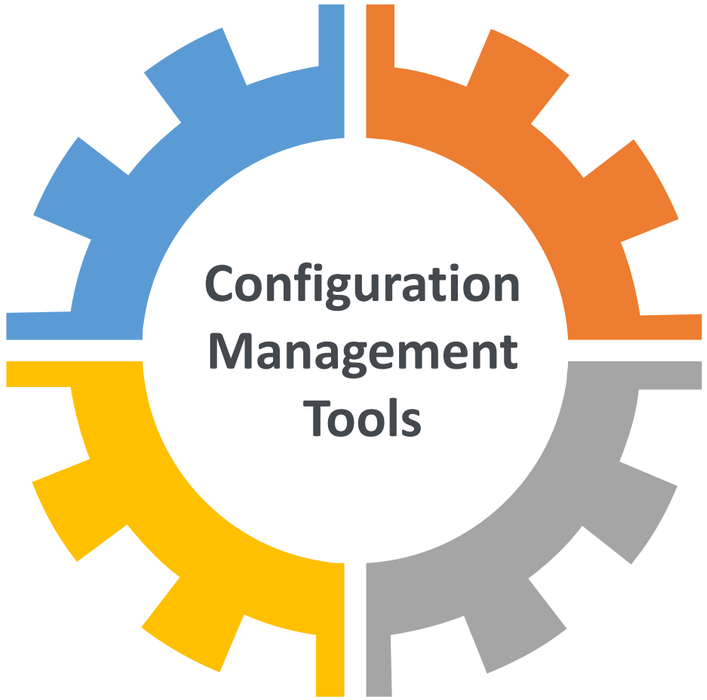
Configuration Management Tools

Types of Configuration Management tools

Types of Configuration Management Tools

Puppet
Ruby DSL-based CM tool used for managing software, systems, and network configuration items

Ansible
Python-based CM tool, also considered as agentless CM tool



Chef
Ruby-based CM tool having integration with most of cloud-based platforms

SaltStack
Python-based open-source CM tool used to manage configuration items remotely

Configuration Management Tools

Overview of Puppet

Puppet

Puppet is an open-source CM Tool

Developed and launched by Puppet Labs in year 2005

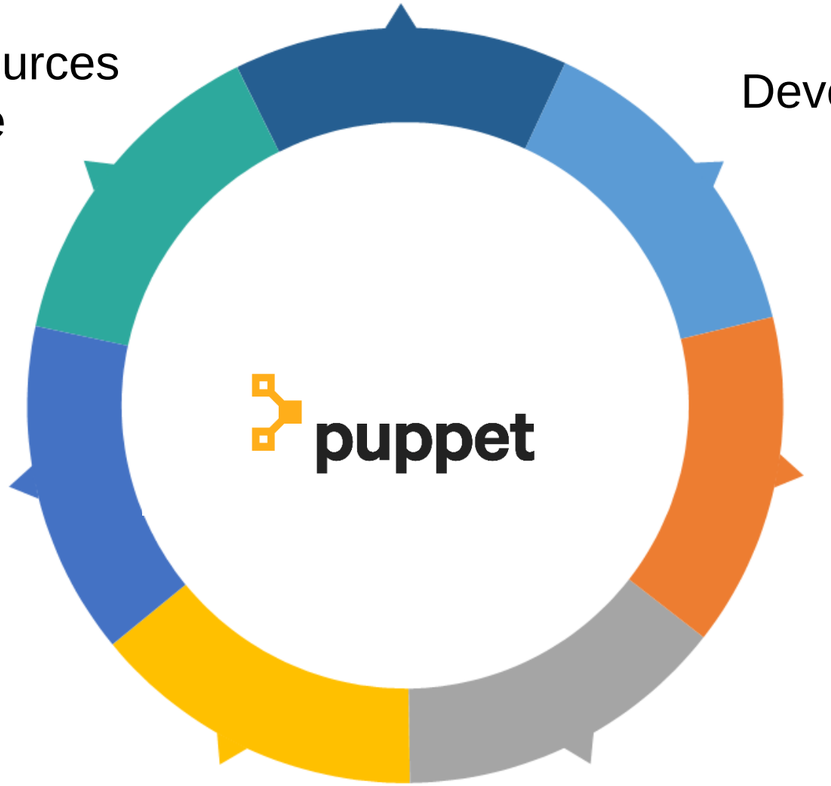
Written in Ruby DSL language. It is launched under GNU and Apache license 2.0

Used to manage both, Unix and Windows-based servers

Follows client-server architecture for managing configuration items

Designed to manage changes on software systems

Used to manage various resources and application software installations



Manifest File in Puppet

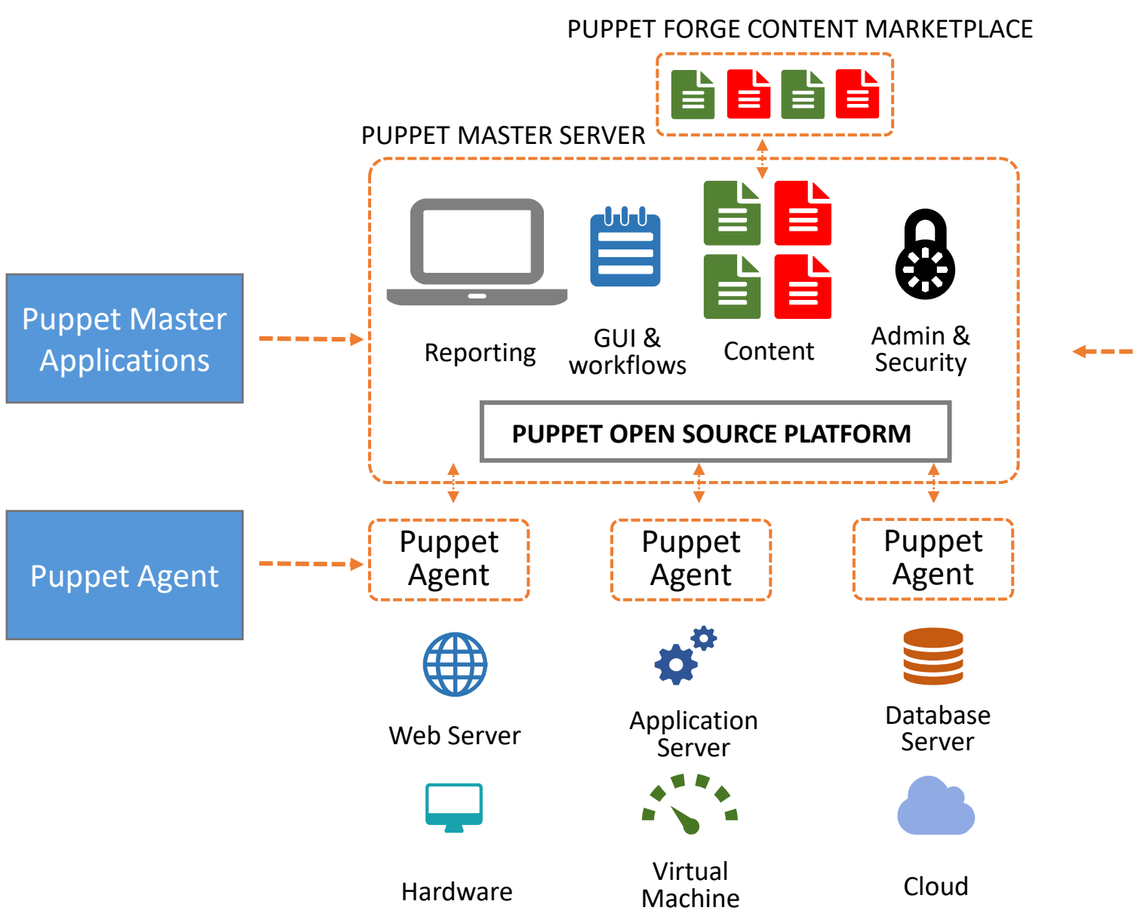
Manifest file in puppet is used to manage various resources on software systems. These resources are packages, files, and system configuration changes. Resources can be managed using the manifest script file written in Ruby DSL language.

```
# execute 'apt-get update'
exec { 'apt-update':                    # exec resource named 'apt-update'
  command => '/usr/bin/apt-get update'  # command this resource will run
}

# install apache2 package
package { 'apache2':
  require => Exec['apt-update'],        # require 'apt-update' before installing
  ensure => installed,
}

# ensure apache2 service is running
service { 'apache2':
  ensure => running,
}
```

Manifest File in Puppet (Contd.)

```
# install mysql-server package
package { 'mysql-server':
  require => Exec['apt-update'],        # require 'apt-update' before installing
  ensure => installed,
}

# ensure mysql service is running
service { 'mysql':
  ensure => running,
}
# install php5 package
package { 'php':
  require => Exec['apt-update'],        # require 'apt-update' before installing
  ensure => installed,
}
# ensure info.php file exists
file { '/var/www/html/info.php':
  ensure => file,
  content => '<?php  phpinfo(); ?>',    # phpinfo code
  require => Package['apache2'],        # require 'apache2' package before creating
}
```

Puppet Architecture



- Describes the desired system state of the computer. Examples: Software, libraries, and configuration setup.
- Lists all resources as well as dependencies between those resources in a catalog. Example: A Windows OS upgrade will affect the software Visual Studio Code.
- Puppet uses several resources of information to compile a catalog.

Configuration Management Tools

Overview of Chef

Chef Architecture

- Chef was developed in Ruby and Erlang languages. It was initially launched in year 2009.
- It uses pure-Ruby DSL based language for managing system resources.
- It’s extensive integration with latest cloud platforms like AWS, GoogChef runs both in Client-Server and solo architecture for managing both infra and software resources.
- We can manage both Unix based and Windows based systems using Chef.

Chef Architecture (Contd.)

- Chef is comprised of three main components:

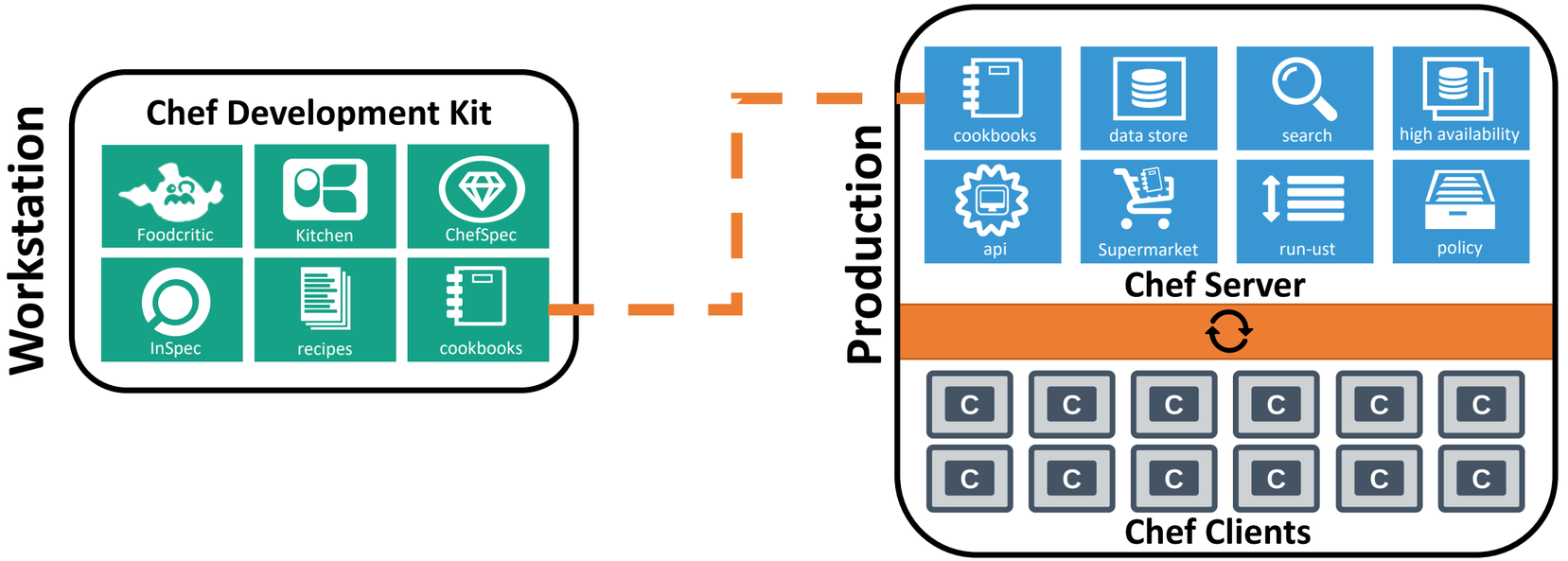
  Chef-server, Chef-client, and Workstation.

- Chef-server is used to manage various chef resources, information on nodes, and cookbooks.

- Chef-client should be deployed on various systems in a Chef environment.

- Workstation is one of the components used to manage all Chef resources.

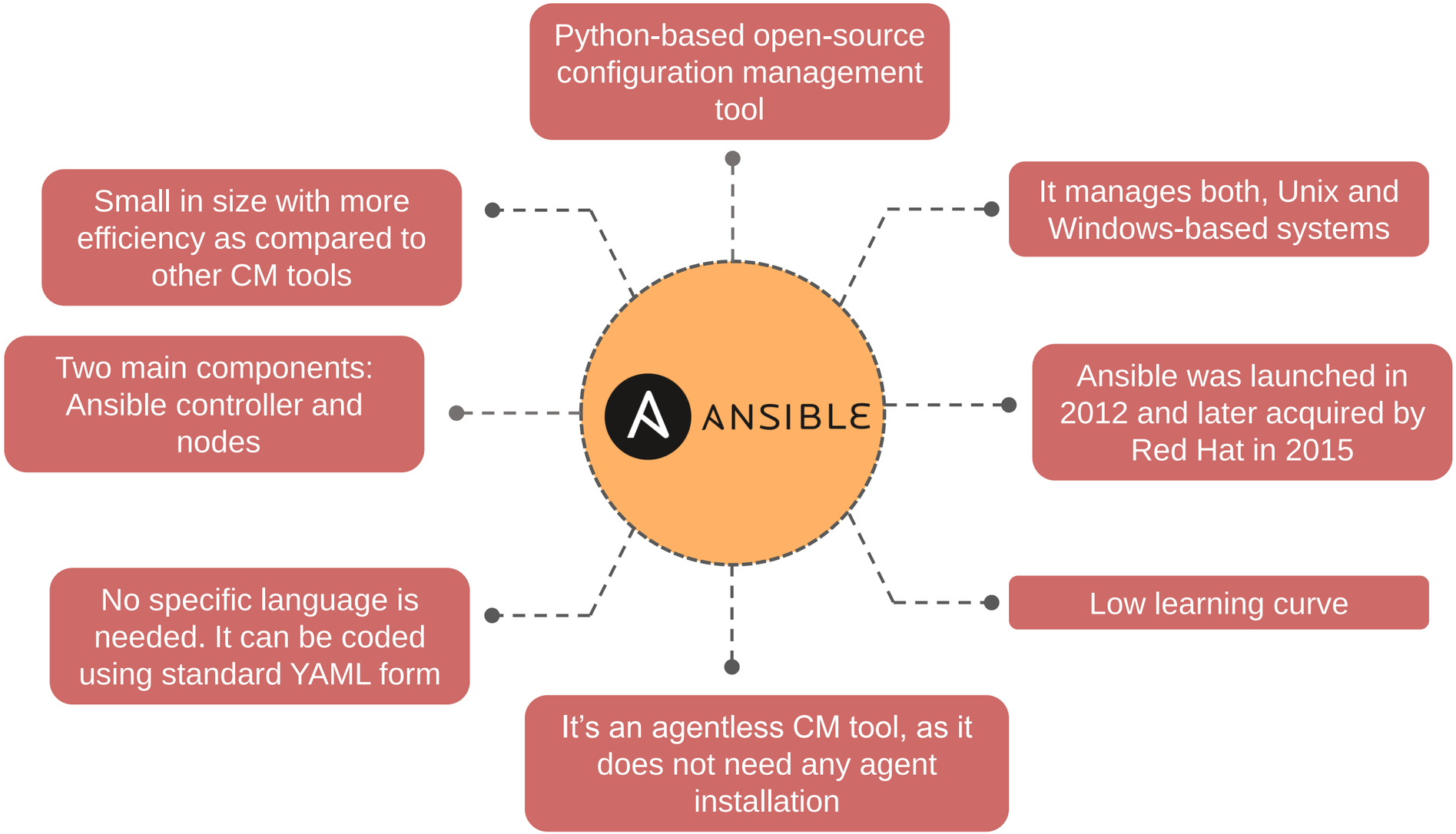
Chef Architecture (Contd.)



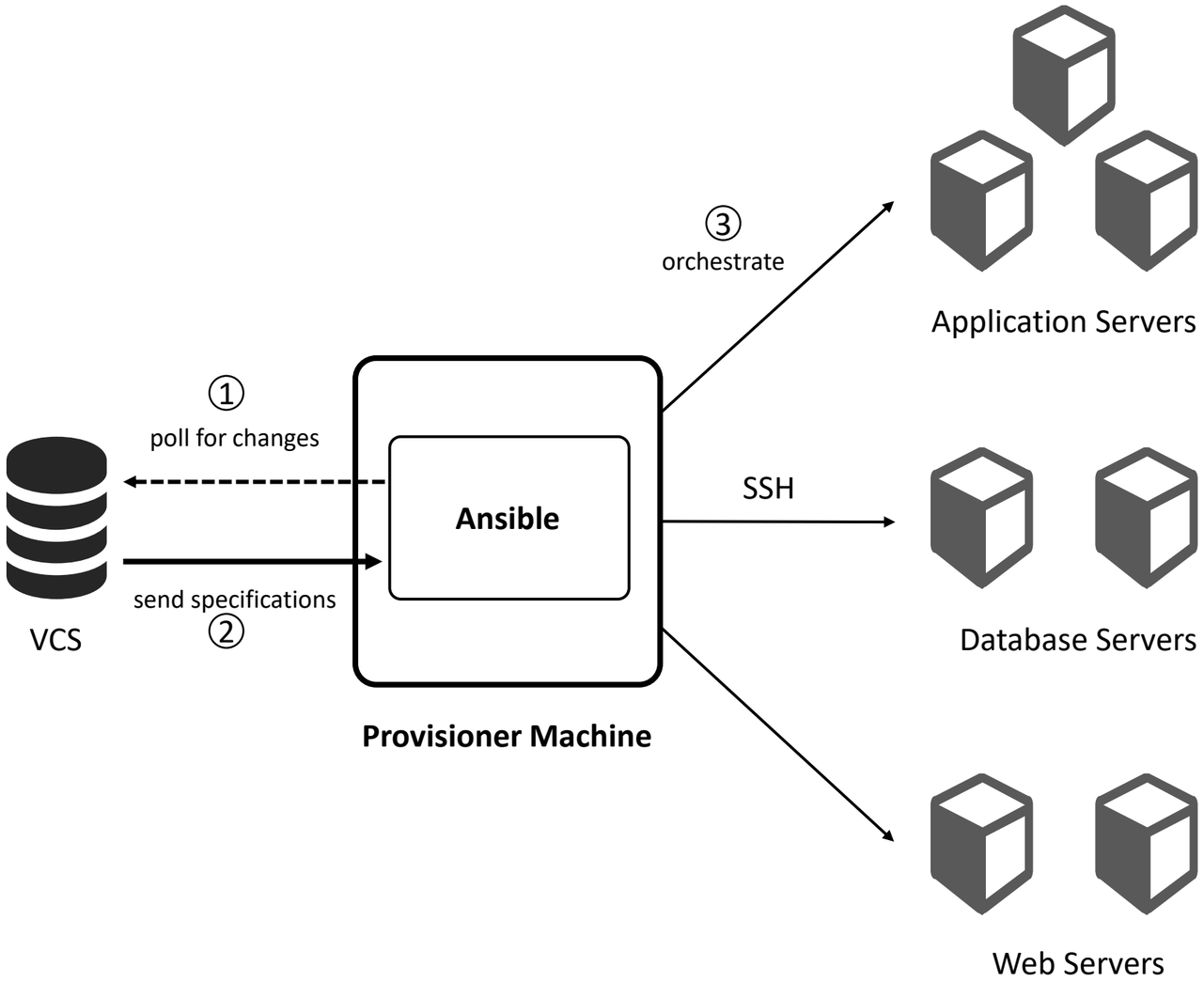
Configuration Management Tools

Overview of Ansible

Ansible



Ansible Architecture



Ansible Installation Steps

```
root@ansible:~# apt-get install software-properties-common
Reading package lists... Done
Building dependency tree
Reading state information... Done
software-properties-common is already the newest version (0.96.24.32.5).
The following package was automatically installed and is no longer required:
  grub-pc-bin
Use 'apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 9 not upgraded.
root@ansible:~# apt-add-repository ppa:ansible/ansible
 Ansible is a radically simple IT automation platform that makes your applications and sy
om code to deploy and update your applications— automate in a language that approaches pl
remote systems.

http://ansible.com/
 More info: https://launchpad.net/~ansible/+archive/ubuntu/ansible
Press [ENTER] to continue or Ctrl-c to cancel adding it.

Hit:1 http://us-east1.gce.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://us-east1.gce.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://us-east1.gce.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:5 http://archive.canonical.com/ubuntu bionic InRelease
Get:6 http://ppa.launchpad.net/ansible/ansible/ubuntu bionic InRelease [15.9 kB]
Get:7 http://ppa.launchpad.net/ansible/ansible/ubuntu bionic/main amd64 Packages [540 B]
Get:8 http://ppa.launchpad.net/ansible/ansible/ubuntu bionic/main Translation-en [344 B]
Fetched 16.8 kB in 1s (20.5 kB/s)
Reading package lists... Done
root@ansible:~# apt update
```

Ansible Installation Steps (Contd.)

```
root@ansible:~# apt install -y ansible
Reading package lists... Done
Building dependency tree
Reading state information... Done
ansible is already the newest version (2.7.1-1ppa~bionic).
The following package was automatically installed and is no longer required:
  grub-pc-bin
Use 'apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 9 not upgraded.
root@ansible:~# ansible --version
ansible 2.7.1
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.15rc1 (default, Apr 15 2018, 21:51:34) [GCC 7.3.0]
root@ansible:~#
```

Ansible Installation Steps (Contd.)

```
root@ansible:~# apt install -y ansible
Reading package lists... Done
Building dependency tree
Reading state information... Done
ansible is already the newest version (2.7.1-1ppa~bionic).
The following package was automatically installed and is no longer required:
  grub-pc-bin
Use 'apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 9 not upgraded.
root@ansible:~# ansible --version
ansible 2.7.1
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.15rc1 (default, Apr 15 2018, 21:51:34) [GCC 7.3.0]
root@ansible:~#
```

Ansible Installation Steps (Contd.)

SSH connectivity have to be set up between Ansible controller machine and node machines to manage configuration items.

- Generate SSH Key Pair on Ansible controller machine using the command “ssh-keygen –t rsa”.
- Now copy public key to “authorized_keys” file on other node machines.
- Once the key is copied, it can be verified using the SSH command. It should work without a password.

Ansible Installation Steps (Contd.)

- The same procedure needs to be repeated for all other nodes one by one.
- Once the setup is completed, the node's IP address should be mentioned in the Ansible inventory file.

```
root@ansible:~# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
```

```
root@ansible:~# ssh 10.142.0.3 "date"
Sat Nov 10 03:34:26 UTC 2018
```

Ansible Configuration and Example Commands

```
root@ansible:~# cat /etc/ansible/hosts
[servers]
10.142.0.3
10.142.0.2
root@ansible:~# ansible -m ping all
10.142.0.3 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
10.142.0.2 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
root@ansible:~# ansible -m shell -a "hostname" all
10.142.0.3 | CHANGED | rc=0 >>
node

10.142.0.2 | CHANGED | rc=0 >>
ansible

root@ansible:~# ansible -m shell -a "uname -a" servers
10.142.0.2 | CHANGED | rc=0 >>
Linux ansible 4.15.0-1023-gcp #24-Ubuntu SMP Wed Oct 10 13:28:59 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux

10.142.0.3 | CHANGED | rc=0 >>
Linux node 4.15.0-1023-gcp #24-Ubuntu SMP Wed Oct 10 13:28:59 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux

root@ansible:~#
```

Ansible Modules

Ansible has different types of module libraries. It can be used directly, or it can be included in playbooks. These modules are predefined source codes provided by Ansible to execute a set of steps. It’s easy to use Ansible modules as they do not need any coding experience.

- The modules use key-value arguments while managing resources.
- The modules can be used to manage services, packages, and files.

Ansible Modules (Contd.)

ansible-doc –l (Command to fetch all modules in Ansible)

ansible-doc <module_name> (Command to get documentation of a specific module in Ansible)

```
root@ansible:~# ansible-doc -l| head -n 4
a10_server                                      Manage A10 Networks AX/SoftAX/Thunder/vThunder devices' server object.
a10_server_axapi3                               Manage A10 Networks AX/SoftAX/Thunder/vThunder devices
a10_service_group                               Manage A10 Networks AX/SoftAX/Thunder/vThunder devices' service groups.
a10_virtual_server                              Manage A10 Networks AX/SoftAX/Thunder/vThunder devices' virtual servers.
ERROR! Unexpected Exception, this is probably a bug: [Errno 32] Broken pipe
root@ansible:~# ansible-doc -l| grep shell
shell                                           Execute commands in nodes.
vmware_vm_shell                                 Run commands in a VMware guest operating system
win_psmodule                                    Adds or removes a Powershell Module
win_shell                                       Execute shell commands on target hosts
root@ansible:~#
```

Ansible Modules (Contd.)

```
root@ansible:~# ansible-doc ping
> PING    (/usr/lib/python2.7/dist-packages/ansible/modules/system/ping.py)

        A trivial test module, this module always returns `pong' on successful contact. It does not make sense in playbooks, but it is useful from
        `/usr/bin/ansible' to verify the ability to login and that a usable Python is configured. This is NOT ICMP ping, this is just a trivial test module that
        requires Python on the remote-node. For Windows targets, use the [win_ping] module instead. For Network targets, use the [net_ping] module instead.

OPTIONS (= is mandatory):

- data
        Data to return for the `ping' return value.
        If this parameter is set to `crash', the module will cause an exception.
        [Default: pong]


NOTES:
      * For Windows targets, use the [win_ping] module instead.
      * For Network targets, use the [net_ping] module instead.

AUTHOR: Ansible Core Team, Michael DeHaan
        METADATA:
          status:
          - stableinterface
          supported_by: core


EXAMPLES:
# Test we can logon to 'webservers' and execute python with json lib.
# ansible webservers -m ping

# Example from an Ansible Playbook
- ping:

# Induce an exception to see what happens
- ping:
    data: crash

RETURN VALUES:


ping:
    description: value provided with the data parameter
    returned: success
    type: string
    sample: pong
```

Ansible Playbook

Ansible playbook is a YAML template used to manage system resources automatically. There is no need to login to each and every server. These templates can be scheduled for both, system configuration and application deployment.

Below is the example of one YAML template used to install and manage the NTP package on Ansible systems.

```
root@ansible:~# cat ntp.yaml
---
- hosts: all
  tasks:
    - name: Run the equivalent of "apt-get update" as a separate step
      apt: update_cache=yes
    - name: Install NTP package
      apt: name=ntp state=present
    - name: Restart NTP service
      service: name=ntp state=restarted
```

Ansible Playbook Execution

```
root@ansible:~# ansible-playbook ntp.yaml

PLAY [all] ********************************************************************************************************************************

TASK [Gathering Facts] ********************************************************************************************************************
ok: [10.142.0.2]
ok: [10.142.0.3]

TASK [Run the equivalent of "apt-get update" as a separate step] **************************************************************************
changed: [10.142.0.3]
changed: [10.142.0.2]

TASK [Install NTP package] ****************************************************************************************************************
ok: [10.142.0.2]
ok: [10.142.0.3]

TASK [Restart NTP service] ****************************************************************************************************************
changed: [10.142.0.2]
changed: [10.142.0.3]

PLAY RECAP ********************************************************************************************************************************
10.142.0.2                 : ok=4    changed=2    unreachable=0    failed=0
10.142.0.3                 : ok=4    changed=2    unreachable=0    failed=0

root@ansible:~# ansible -m shell -a "service ntp status | head -n 3" all
 [WARNING]: Consider using the service module rather than running service.  If you need to use command because service is insufficient you
can add warn=False to this command task or set command_warnings=False in ansible.cfg to get rid of this message.

10.142.0.2 | CHANGED | rc=0 >>
● ntp.service - Network Time Service
   Loaded: loaded (/lib/systemd/system/ntp.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2018-11-10 03:57:43 UTC; 12s ago

10.142.0.3 | CHANGED | rc=0 >>
● ntp.service - Network Time Service
   Loaded: loaded (/lib/systemd/system/ntp.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2018-11-10 03:57:43 UTC; 12s ago
```

Assisted Practice

Duration: 70 mins

Set Up Apache Web Server Using Ansible

Problem Statement: You are given a project to demonstrate the use of Ansible by setting up the Apache Web server.

Access: Click on the **Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Assisted Practice: Guidelines to Set Up Apache Web Server Using Ansible

1. Login to your Ubuntu Lab and open the terminal.
2. Install Ansible in Ubuntu.
3. Establish connection between Ansible controller and the node machine.
4. Write Ansible YAML script to install Ansible software.
5. Run Ansible YAML script.

Configuration Management Tools

Overview of SaltStack

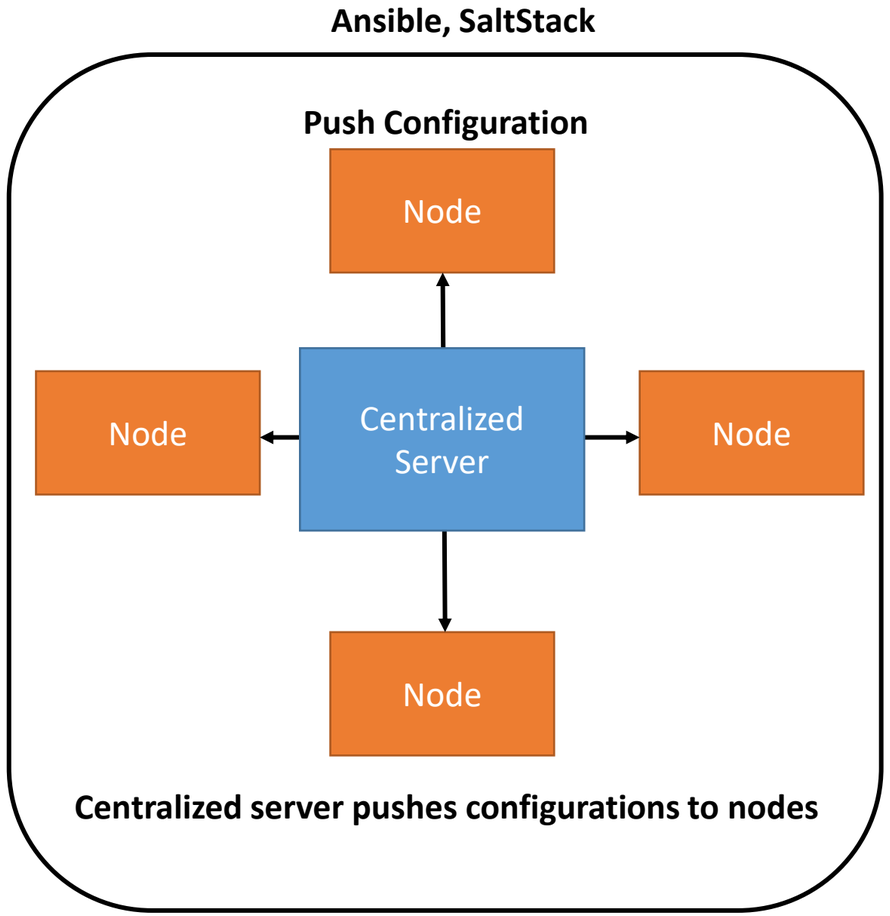
SaltStack Configuration Management Tool

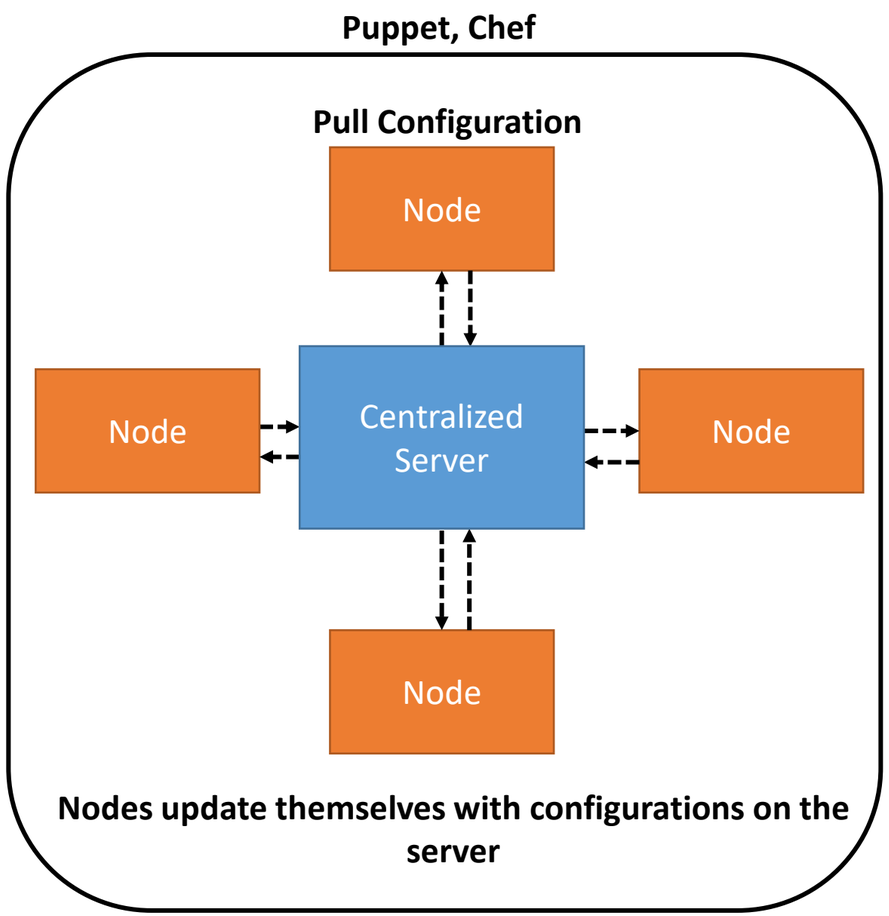
- SaltStack, like Ansible, is also a Python-based system provisioning tool.
- Tom Hatch, an IT Architect, built SaltStack in 2011. He was involved in both, Puppet and Chef configuration management tools.
- SaltStack is an open-source CM tool that comes under Apache 2.0 license.
- Like Ansible, YAML or Python-based scripts are used to manage system resources.

SaltStack Configuration Management Tool (Contd.)

- SaltStack is one of the CM tools competing with Chef, Puppet, and Ansible.
- Like Ansible, learning curve is less in case of SaltStack, as the same modules are here to manage system resources.
- SaltStack modules can be used to manage services, packages, and file resources.

Architecture Similarity





Configuration Management Tools

Comparison of Ansible, Puppet, Chef, and SaltStack

Comparison of Configuration Management Tools

CM Tools	Pros	Cons	Enterprise annual cost for an average of 100 nodes
Ansible	• Simple architecture • Low learning curve	• No Windows support for controller machine • GUI is not good as compared to Chef and Puppet	Ansible open-source for 100 nodes → $0 Ansible Tower for 100 nodes → $3000 Ansible Tower Enterprise for 100 nodes → $10,000
Salt	• Scalable and fast • Easy to manage	• GUI is not good	Salt Enterprise for 100 nodes → $15,000
Chef	• More features • Perfect GUI with lot of features	• Needs knowledge of Ruby • Takes more time to understand • Learning curve is more in Chef • Only 10 nodes allowed in open-source license	Chef Enterprise for 100 nodes → $7200
Puppet	• Oldest tool in market • GUI with better features	• Difficult to understand and configure • Has less number of integrations with other tools	Puppet Enterprise for 100 nodes → $10,500

Key Takeaways

You are now able to:

- Explain the concepts of configuration management tools
- Demonstrate Puppet installation
- Demonstrate Chef installation
- Demonstrate Ansible installation and its usage
- Describe SaltStack
- Select the suitable CM tool for your organization

Knowledge Check

Knowledge Check 1	**Which configuration item cannot be managed using configuration management tool?**

a. Git Package

b. Server Bounce

c. Changing RAM hardware for server

d. Creating file on file system

Knowledge Check 1	**Which configuration item cannot be managed using configuration management tool?**

a. Git Package

b. Server Bounce

c. Changing RAM hardware for server

d. Creating file on file system

The correct answer is **c. Changing RAM hardware for server**

Only system-related configuration items can be managed using a CM tool. Hardware-related items can't be managed using these tools.

Knowledge Check

2

What is the main objective of configuration management tools?

a. Managing configuration items

b. Controlling modifications on server

c. Auditing configurations done to the system

d. All of the above

Knowledge Check

2

What is the main objective of configuration management tools?

a. Managing configuration items

b. Controlling modifications on server

c. Auditing configurations done to the system

d. All of the above

The correct answer is **d. All of the above**

All of the above options can be used to manage various configurations on the system.

Knowledge Check 3	**Which one of the following is a pull-based configuration management tool ?**

a. Chef

b. Ansible

c. Salt

d. Jaro

Knowledge Check 3	**Which one of the following is a pull-based configuration management tool ?**

a. Chef

b. Ansible

c. Salt

d. Jaro

The correct answer is **a. Chef**

Chef-client continuously pings the master node for configuration changes and pulls the configuration scripts.

Knowledge Check 4	**Which one of the following is an open-source configuration management tool ?**

a. Chef

b. Ansible

c. SourceForge

d. Puppet

Knowledge Check 4	**Which one of the following is an open-source configuration management tool ?**

a. Chef

b. Ansible

c. SourceForge

d. Puppet

The correct answer is **b. Ansible**

Ansible is an open-source software used to manage production environment.

Lesson-End Project

Set Up MySQL Database Using Ansible

Duration: 40 mins

Problem Statement:

Perform the following actions:

- Install Ansible in Ubuntu if it is not installed.
- Establish connection between Ansible controller and the node machine.
- Write Ansible YAML script to install Ansible software.
- Run Ansible YAML script.

Access: Click on the **Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Thank You