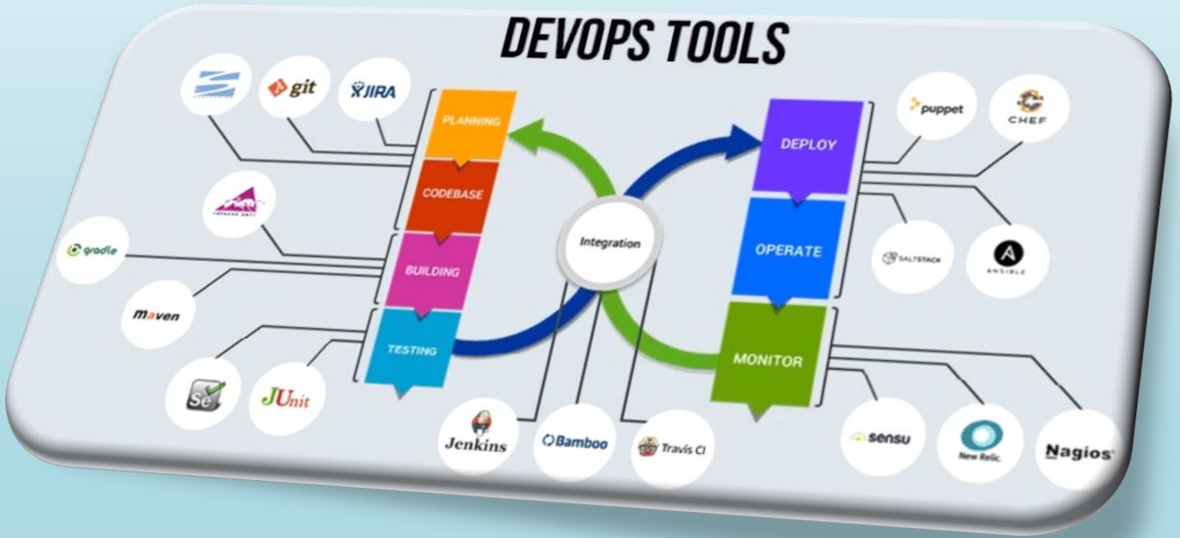
CLOUD TRAIN
ACCELERATE YOUR GROWTH

Configuration Management [Puppet]



www.thecloudtrain.com

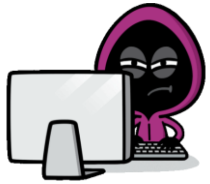
CLOUD TRAIN
ACCELERATE YOUR GROWTH

Agenda

Why Configuration Management?

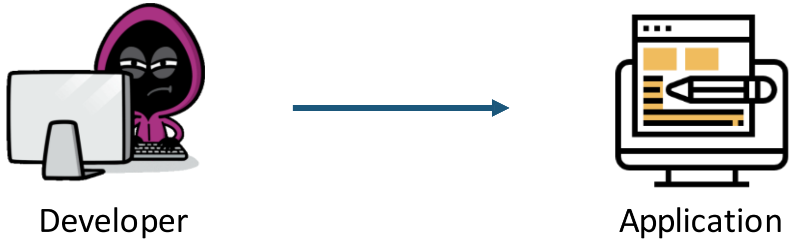
Why Configuration Management?

CLOUD TRAIN
ACCELERATE YOUR GROWTH



Developer

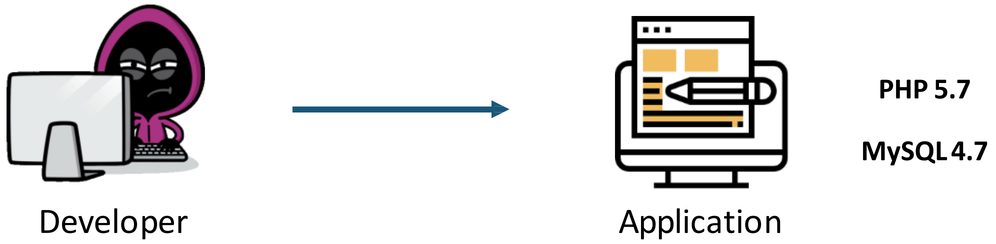
Why Configuration Management?



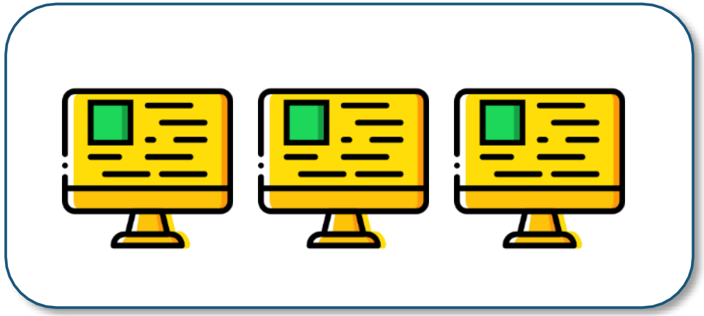
PHP 5.7

MySQL 4.7

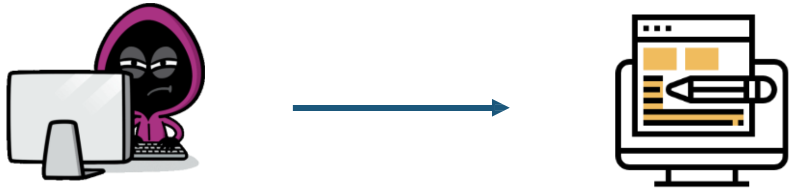
Why Configuration Management?



PHP Servers


Database Servers

Why Configuration Management?



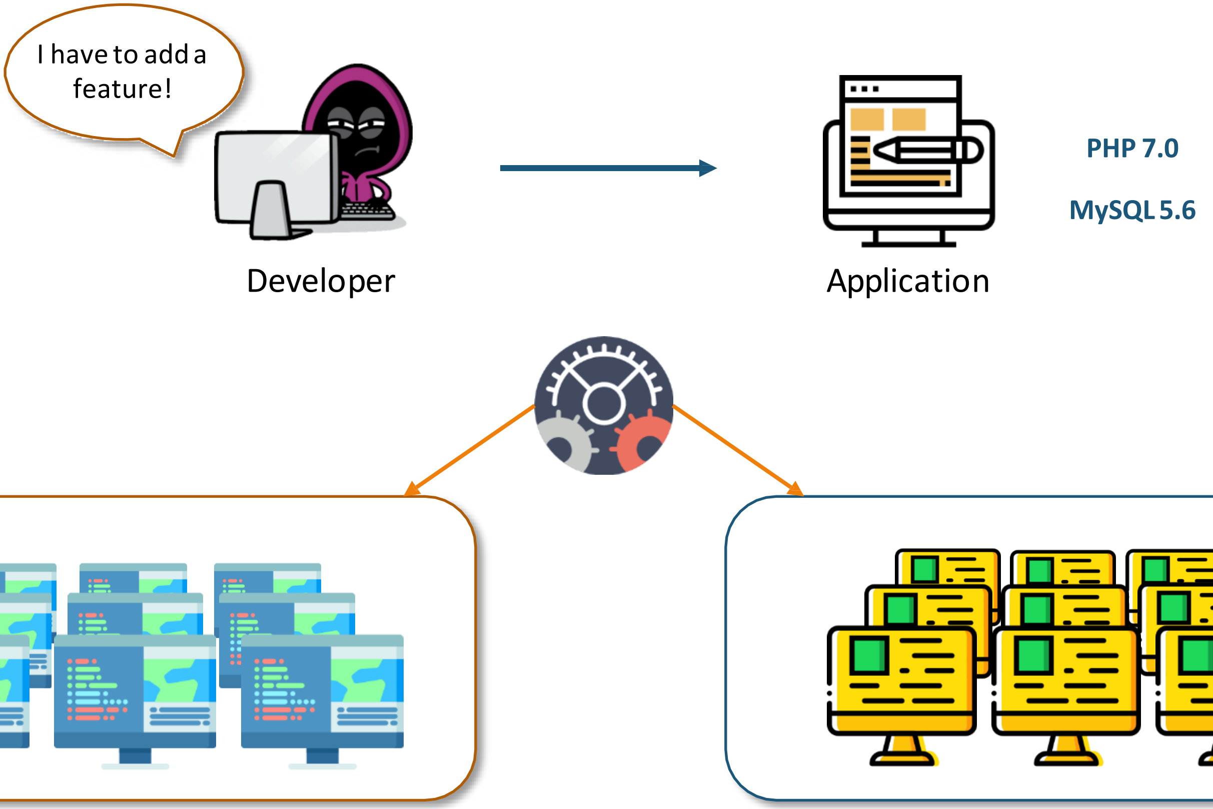
What is Configuration Management?

What is Configuration Management?

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Configuration management is a systems engineering process for establishing and maintaining consistency of a product's performance, functional and physical attributes with its requirements, design and operational information throughout its life.

What is Configuration Management?



PHP 7.0 Servers

Database 5.6 Servers

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Configuration Management Features

- Automation
- Consistency
- Software Updates
- Software Rollback

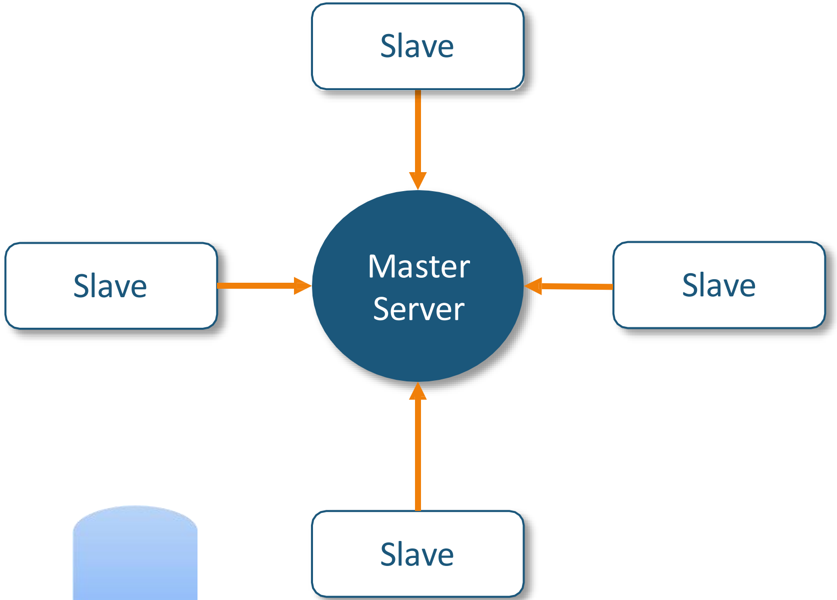
CLOUD TRAIN
ACCELERATE YOUR GROWTH

Configuration Management Tools

Types of Configuration Management Tools

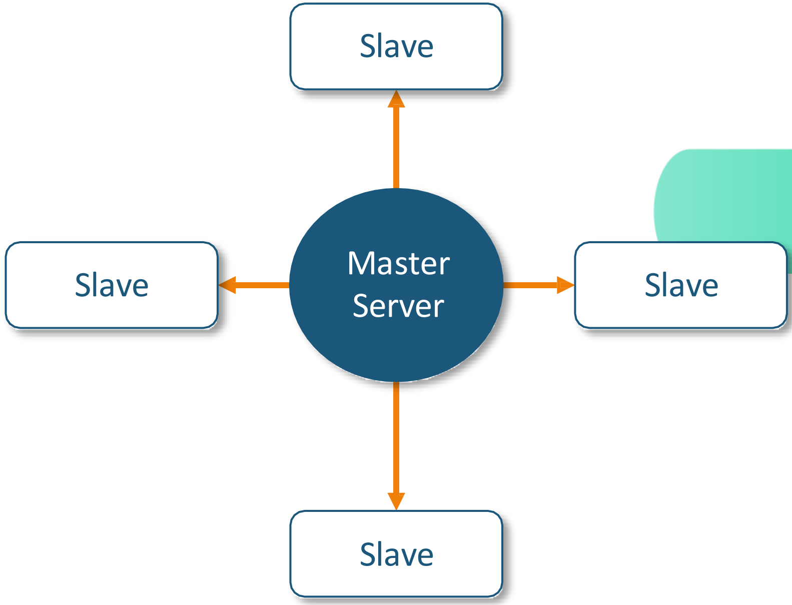
CLOUD TRAIN
ACCELERATE YOUR GROWTH

Pull Configuration



Changes arepulled

Push Configuration



Changes arepushed

Types of Configuration Management Tools

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Pull Configuration

puppet

CHEF

Push Configuration

ANSIBLE

SALTSTACK

CLOUD TRAIN
ACCELERATE YOUR GROWTH

What is Puppet?

What is Puppet?

Puppet is an open-source software configuration management tool. It runs on many Unix-like systems, as well as on Microsoft Windows, and includes its own declarative language to describe the system configuration.

puppet

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Key Features of Puppet

- Large UserBase
- Big Open-source Community
- Documentation
- Platform Support

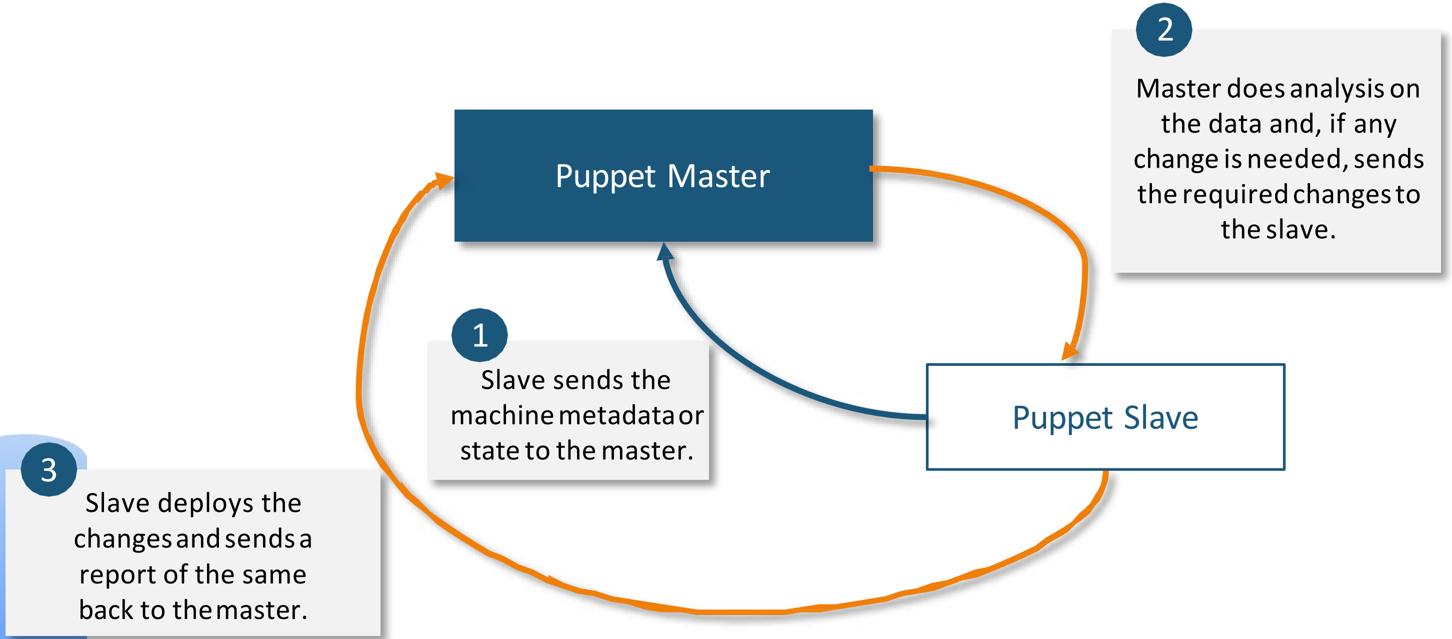
Refer Puppet Documentation here:
https://puppet.com/docs/puppet/7/puppet_index.html

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Puppet Architecture

Puppet Architecture

Puppet follows a Master–Slave architecture, the working of which has been explained in the below diagram.

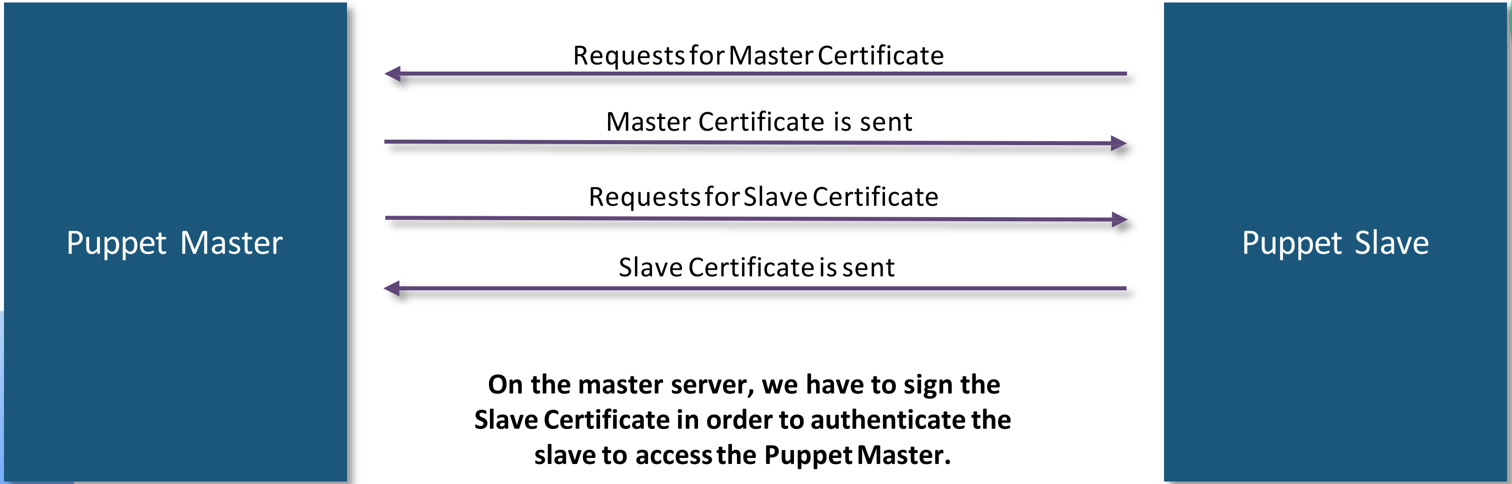


Puppet Master
Puppet Slave
1
Slave sends the machine metadata or state to the master.
2
Master does analysis on the data and, if any change is needed, sends the required changes to the slave.
3
Slave deploys the changes and sends a report of the same back to the master.

Puppet Architecture: SSL Connection

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Because Puppet nodes have to interact with the master, all the information which is communicated between the master node and slave nodes are encrypted using SSL certificates.
The certificate signing process is as follows:


Puppet Master
Requests for Master Certificate
Master Certificate is sent
Requests for Slave Certificate
Slave Certificate is sent
Puppet Slave

On the master server, we have to sign the Slave Certificate in order to authenticate the slave to access the Puppet Master.

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Setting up Puppet Master–Slave

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Code Basics for Puppet

Code Basics for Puppet: Resource

The most basic component of Puppet Code is a **resource**. A resource describes something about the state of the system, such as if a certain user or file should exist, or a package should be installed, etc.

Syntax

```
resource_type { ‘resource_name’:

      attribute => value,
                 …
  }
```

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Code Basics for Puppet: Resource Example

Example

```
package { ‘nginx’:
ensure => ‘installed’,
}
```

Sample nginx package

Code Basics for Puppet: Resource Types

CLOUD TRAIN
ACCELERATE YOUR GROWTH

There are three kinds of resource types:

Puppet core or built-in resource types:
Core or built-in resource types are the pre-built puppet resource types shipped with puppet software. All of the core or built-in Puppet resource types are written and maintained by Puppet team.

Puppet defined resource types:
Defined resource types are lightweight resource types written in Puppet declarative language using a combination of existing resource types.

Puppet custom resource types:
Custom resource types are completely customized resource types written in Ruby.

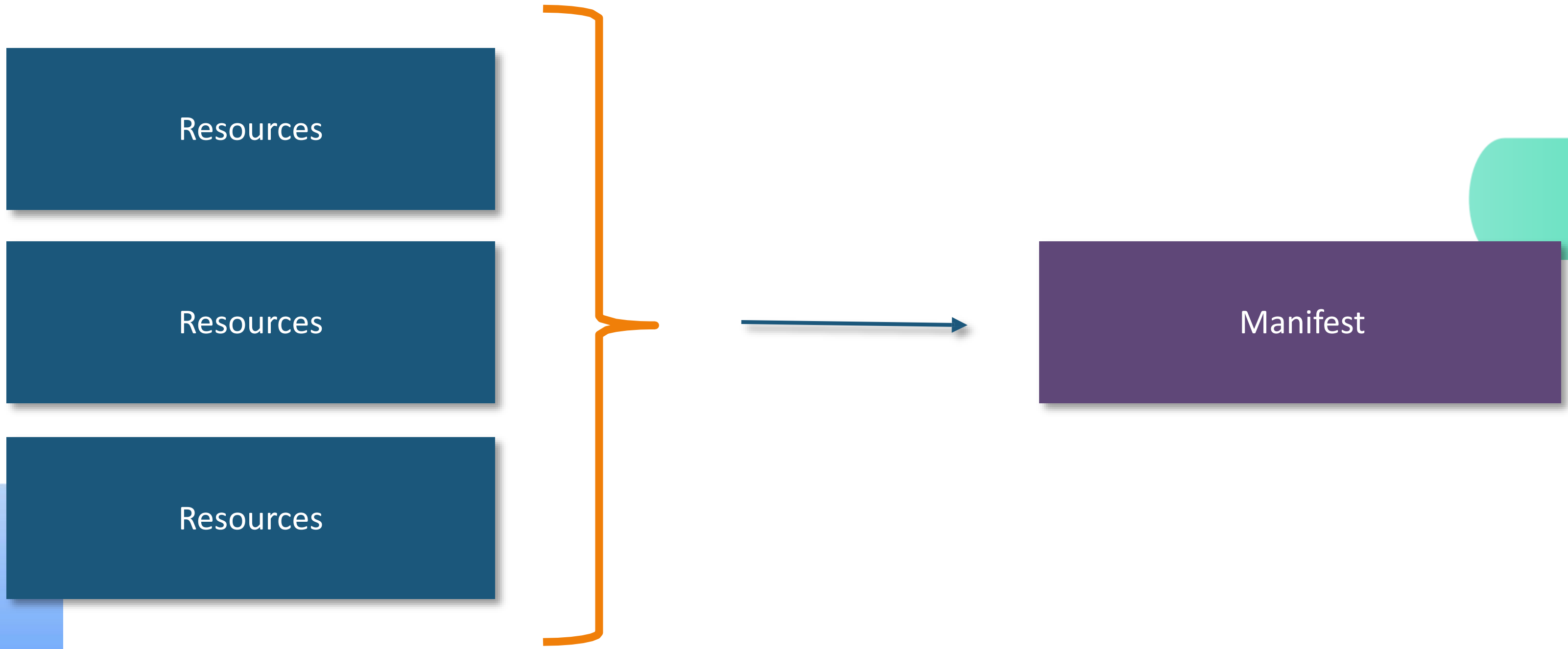
Code Basics for Puppet: Resource Commands

CLOUD TRAIN
ACCELERATE YOUR GROWTH

```
puppet --help
puppet resource –help
puppet resource –types
puppet describe <resource type name>
puppet apply <file>

puppet config set server puppetserver.example.com --section main
```

Code Basics for Puppet: Manifest



Code Basics for Puppet: Manifest

Manifests are basically a collection of resource declarations, using the extension **.pp**.

Example

```
package {  'nginx':
ensure => 'installed',
}

file {'/tmp/hello.txt':
ensure => present,
content => 'hello world',
mode => '0644',

}
```

Sample ManifestFile

Code Basics for Puppet: Manifest

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Variables

Loops

Conditions

Variables can be defined at any point in a manifest. The most common types of variables are strings and arrays of strings, but other types are also supported, such as Booleans and hashes.

Example

```
$text = “hello world”

file {‘/tmp/hello.txt’:
  ensure => present,
  content => $text,
  mode => ‘0644’,
}
```

Code Basics for Puppet: Manifest

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Variables

Loops

Conditions

Loops are typically used to repeat a task using different input values. For instance, instead of creating 10 tasks for installing 10 different packages, you can create a single task and use a loop to repeat the task with all different packages you want to install.

Example

```
$packages = [‘nginx’,’mysql-server’]

package { $packages:
 ensure => installed,
}
```

Code Basics for Puppet: Manifest

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Variables

Loops

Conditions

Conditions can be used to dynamically decide whether or not a block of code should be executed, based on a variable or an output from a command, for instance.

Example

```
exec { "Test":
command =>'/bin/echo apache2 is installed > /tmp/status.txt',
onlyif => '/bin/which apache2',
}
```

Code Basics for Puppet: Manifest

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Variables

Loops

Conditions

Conditions can be used to dynamically decide whether or not a block of code should be executed, based on a variable or an output from a command, for instance.

Example

```
exec { “Test”:
command =>‘/bin/echo apache2 is not installed > /tmp/status.txt’,
unless => ‘/bin/which apache2’,
}
```

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Applying Configuration Using Modules

What are Modules?

A collection of manifests and other related files organized in a predefined way to facilitate sharing and reusing parts of a provisioning

1. sudo puppet module generate <name>
2. Edit the init.pp with a class, and build the module
3. Finally, install the module

What are Classes?

Just like with regular programming languages, classes are used in Puppet to better organize the provisioning and make it easier to reuse portions of the code.

Example

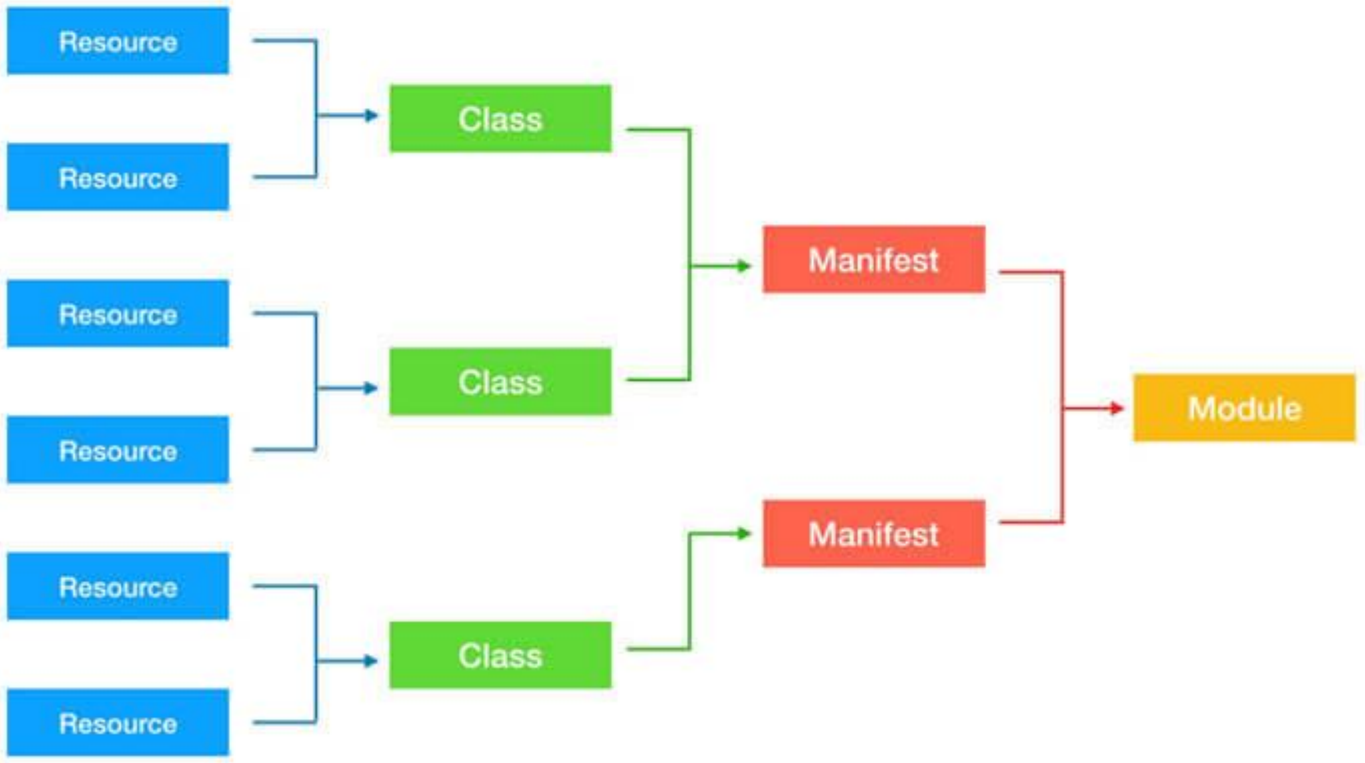
```
Class hello{

                        exec { “Test”:
    command =>‘/bin/echo apache2 is installed > /tmp/status.txt’,
                unless => ‘/bin/which apache2’,
            }

}
```

Overall Picture

CLOUD TRAIN
ACCELERATE YOUR GROWTH



CLOUD TRAIN
ACCELERATE YOUR GROWTH

Hands-on: Applying Configuration Using Modules

Hands-on: Invoking Module’s Classes

CLOUD TRAIN
ACCELERATE YOUR GROWTH

Got queries or need more info?

Contact us

TO ACCELERATE YOUR CAREER GROWTH

For questions and more details:

please call @ +91 98712 72900 or

visit https://www.thecloudtrain.com/ or

email at join@thecloudtrain.com or

WhatsApp us >>