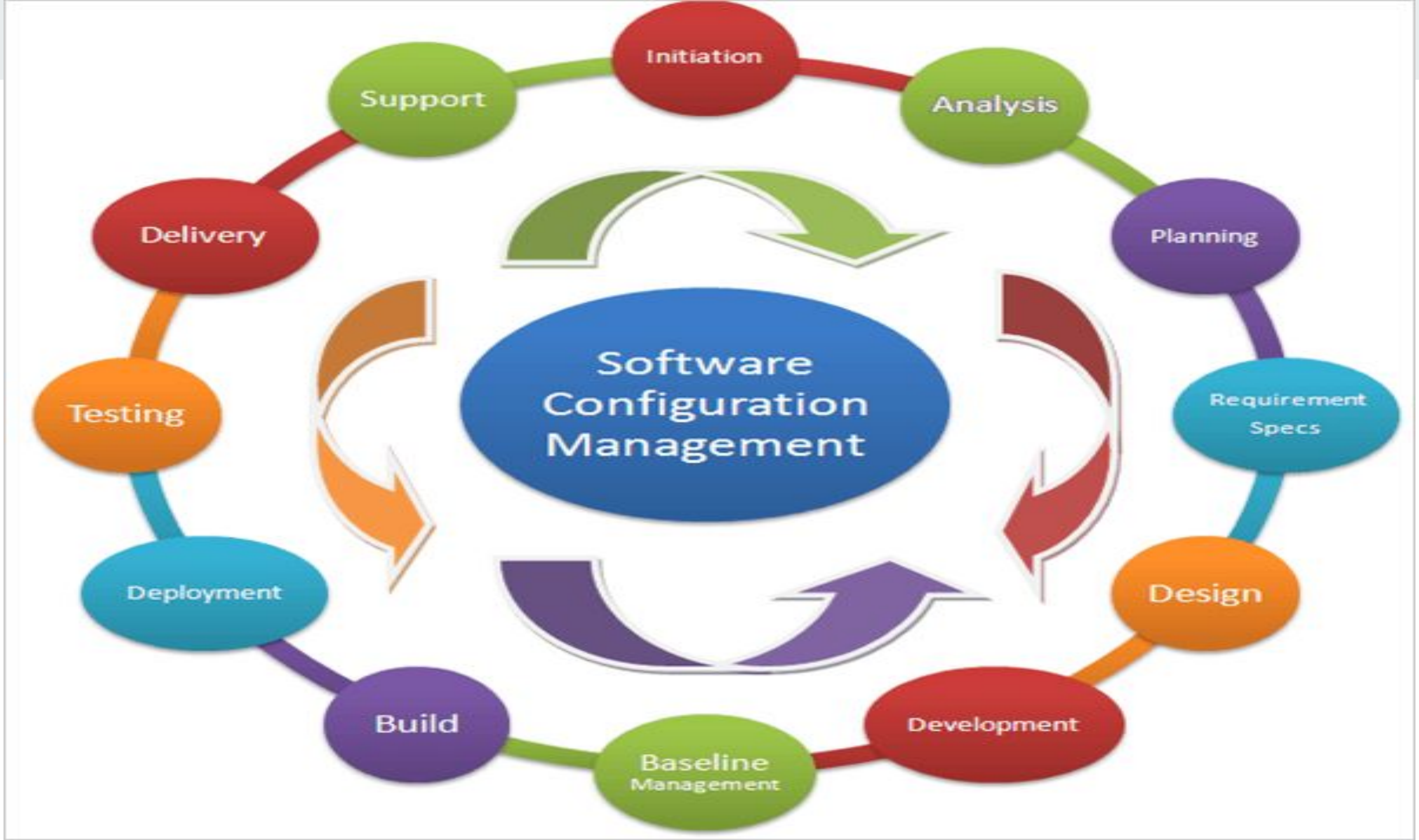
Configuration Management

Configuration Management

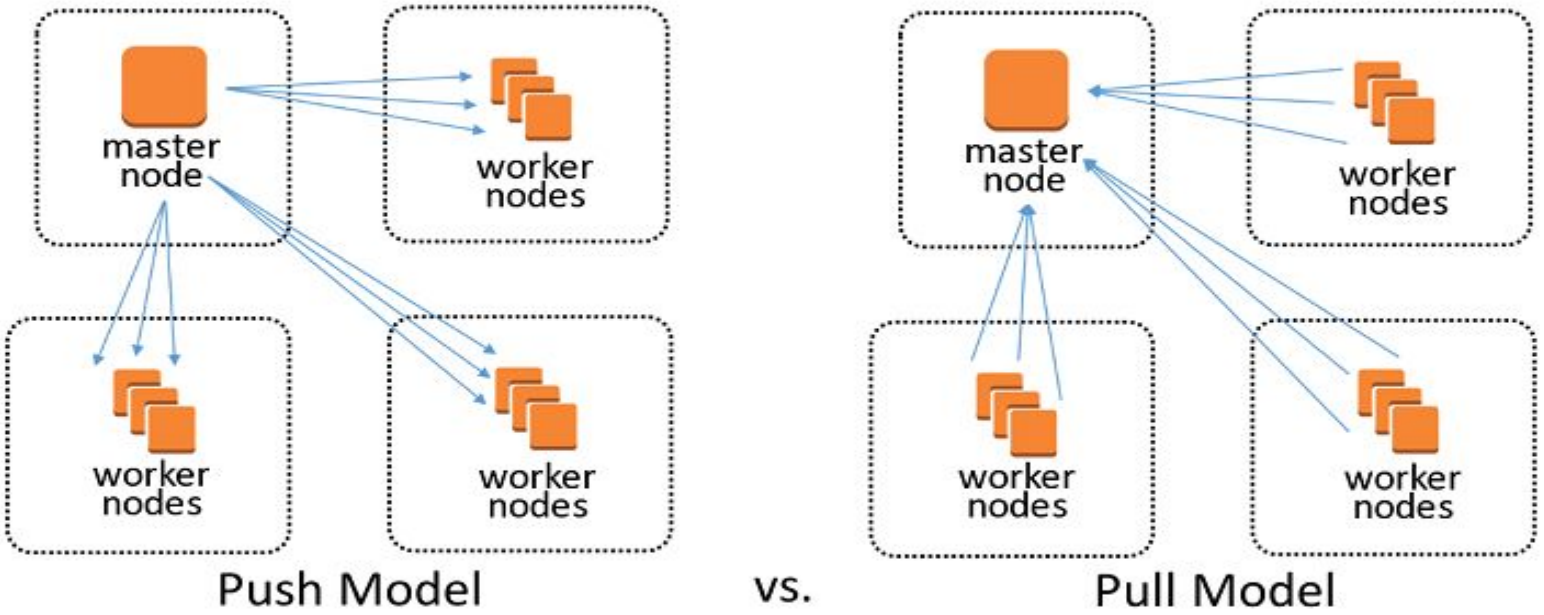
Configuration management (CM) is a systems engineering process for establishing and maintaining consistency of a product's performance, functional, and physical attributes with its requirements, design, and operational information throughout its life.

Configuration management applies to a variety of systems, but most often, you’ll be concerned with these:

- Servers
- Databases and other storage systems
- Operating systems
- Networking
- Applications
- Software


Software Configuration Management
Initiation
Analysis
Planning
Requirement Specs
Design
Development
Baseline Management
Build
Deployment
Testing
Delivery
Support

Push and Pull strategy/Model/workflow

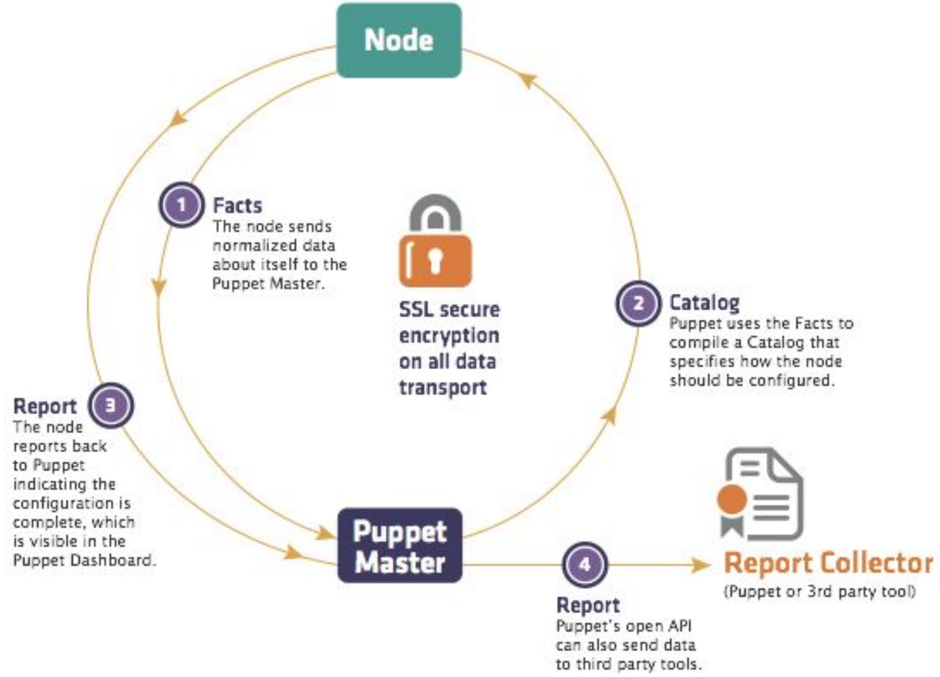


Puppet

Puppet User Stories

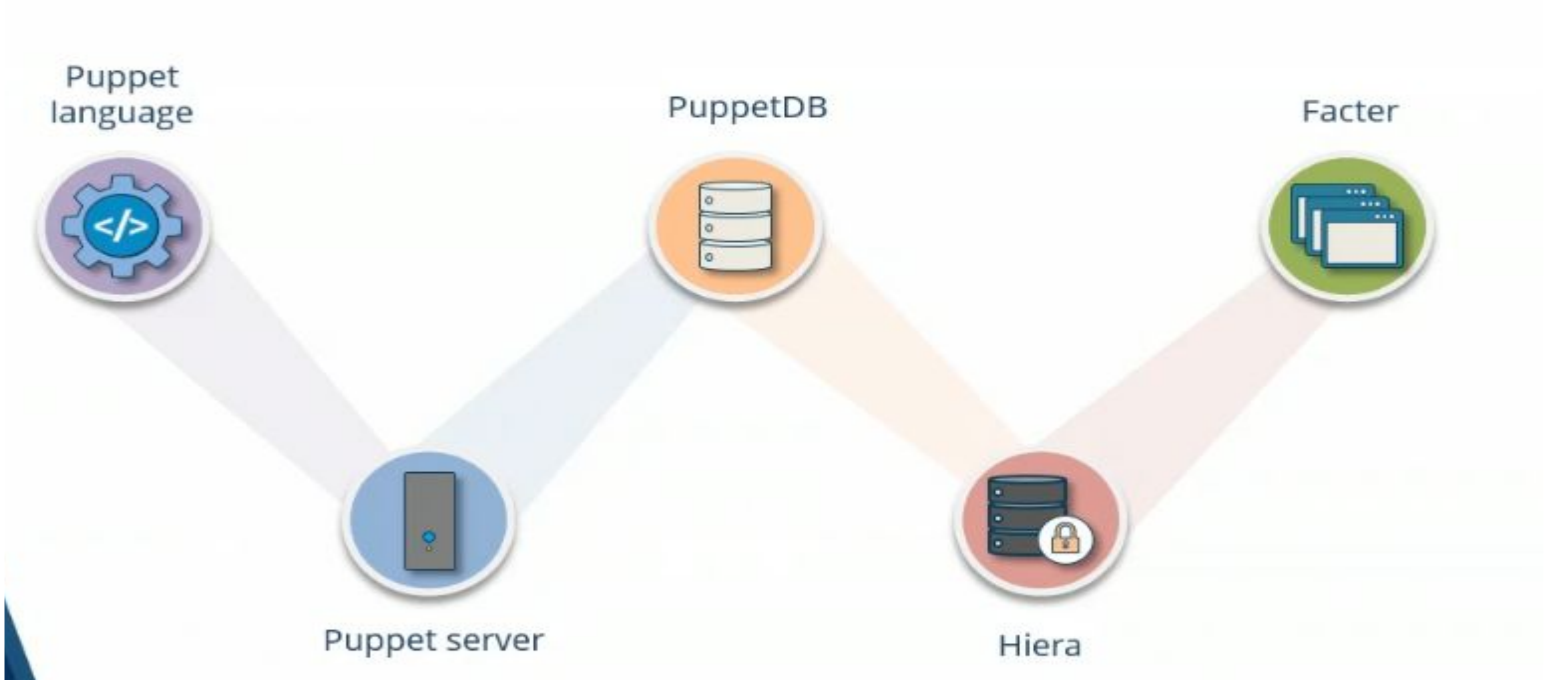
- Infrastructure management
- Configuration management
- Fail over
- Frequent releases and deployments

Puppet Master - Node Architecture



Node
1 Facts
The node sends normalized data about itself to the Puppet Master.
SSL secure encryption on all data transport
2 Catalog
Puppet uses the Facts to compile a Catalog that specifies how the node should be configured.
Report 3
The node reports back to Puppet indicating the configuration is complete, which is visible in the Puppet Dashboard.
Puppet Master
4
Report
Puppet's open API can also send data to third party tools.
Report Collector
(Puppet or 3rd party tool)

Core Components

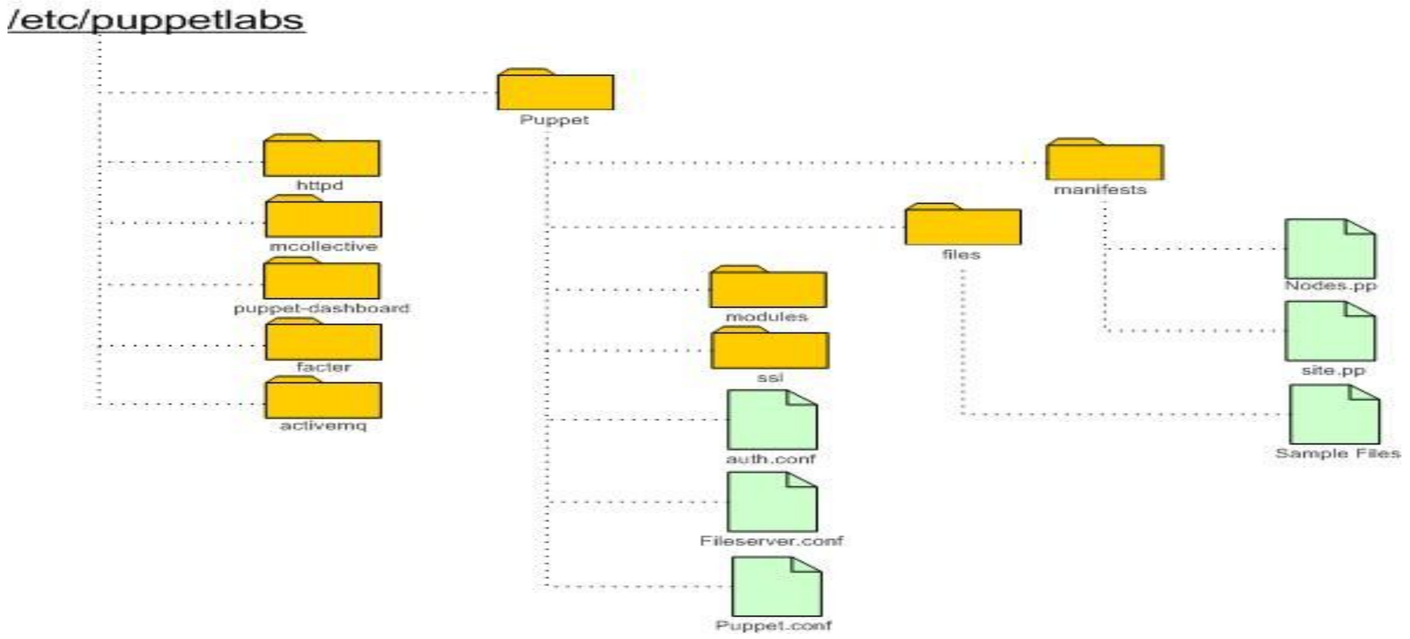


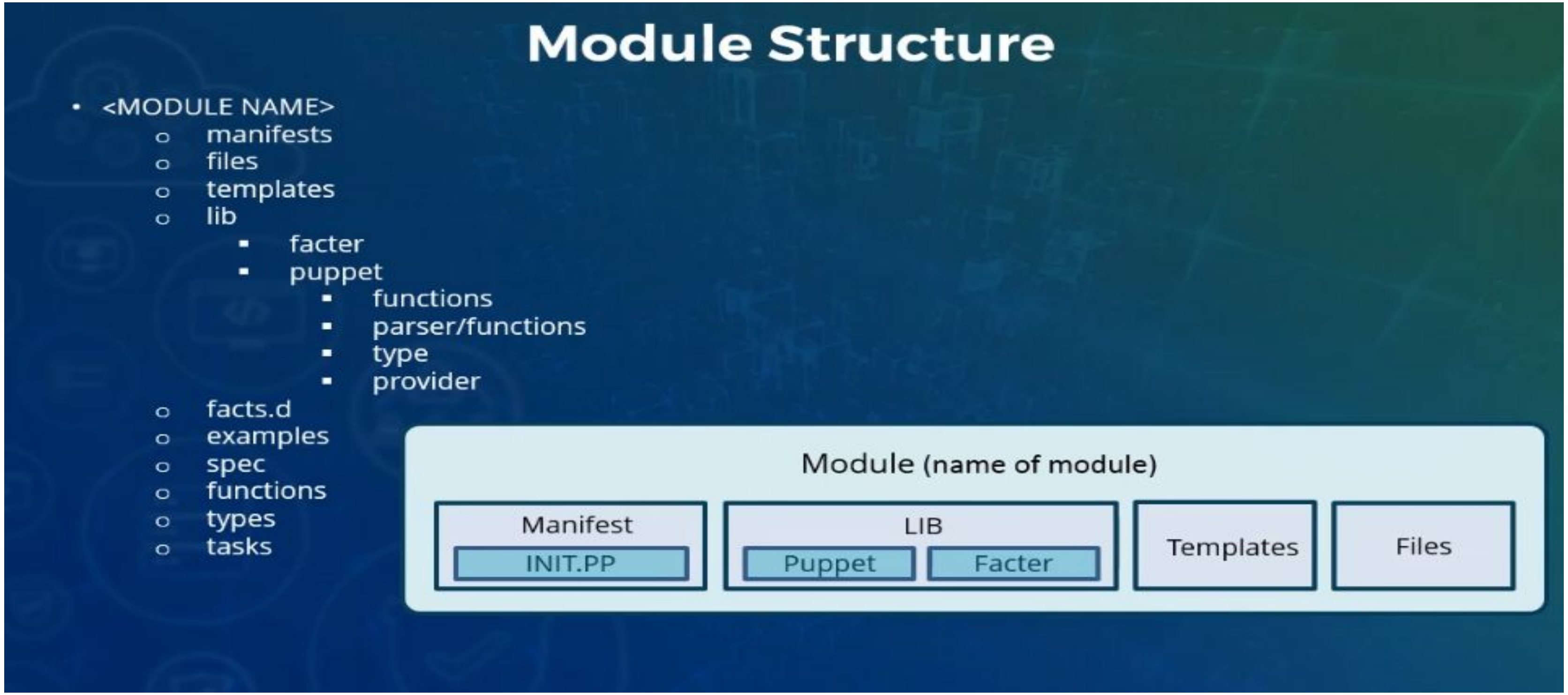
Installing Puppet

Installing puppet server - Only on Linux

Installing puppet client - Both on Windows and Linux

Puppet Folder structure





Module Structure

- <MODULE NAME>
    - manifests
    - files
    - templates
    - lib
        - facter
        - puppet
            - functions
            - parser/functions
            - type
            - provider
    - facts.d
    - examples
    - spec
    - functions
    - types
    - tasks

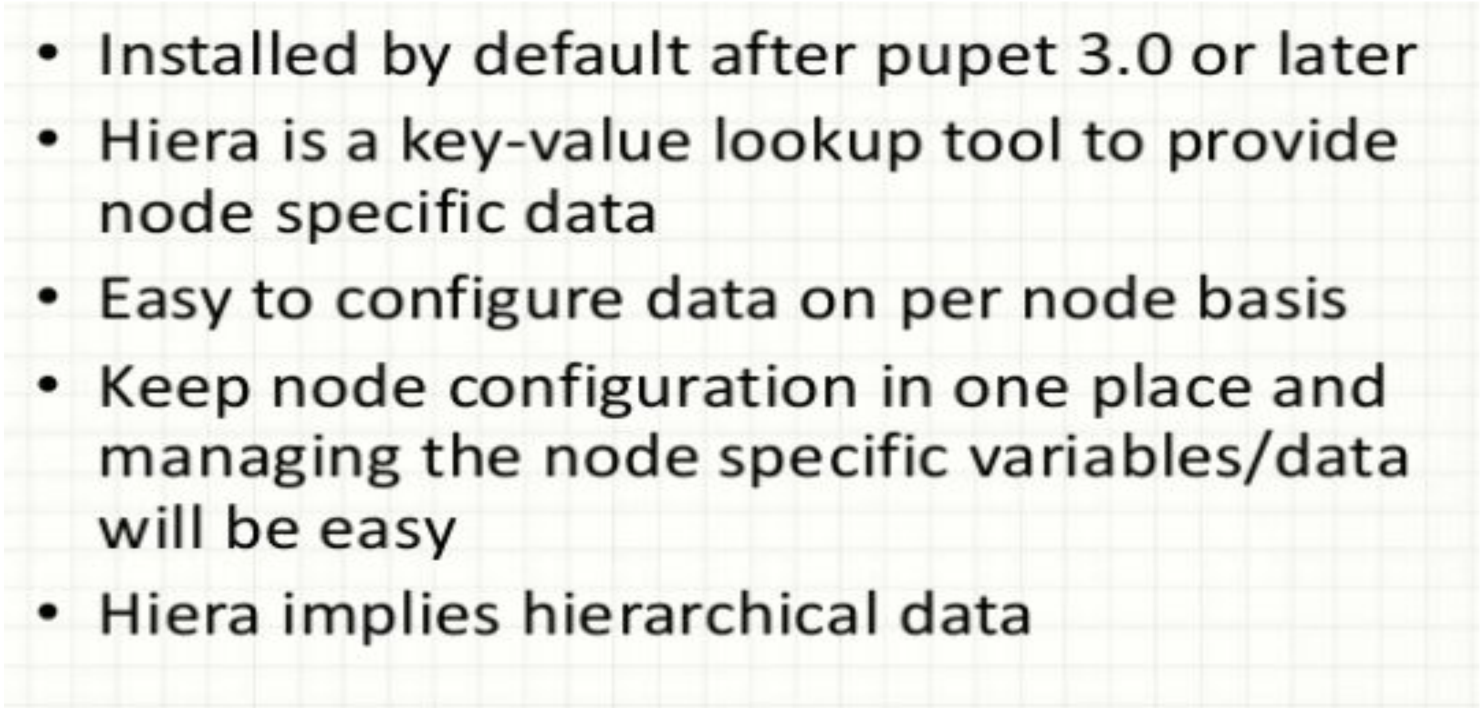
Defining puppet modules and classes

Module name	Filepath to class or defined type	Class or defined type name
username-my_module	my_module/manifests/init.pp	my_module
username-my_module	my_module/manifests/other_class.pp	my_module::other_class
puppetlabs-apache	apache/manifests/security/rule_link.pp	apache::security::rule_link
puppetlabs-apache	apache/manifests/fastcgi/server.pp	apache::fastcgi::server

Facter

- Puppet uses facter to gather information about node
- Run $facter to know what are the factors on your system
- The returned key value pairs are called Facter
- You can use facts in manifests

Hiera



- Installed by default after pupet 3.0 or later
- Hiera is a key-value lookup tool to provide node specific data
- Easy to configure data on per node basis
- Keep node configuration in one place and managing the node specific variables/data will be easy
- Hiera implies hierarchical data

Ansible

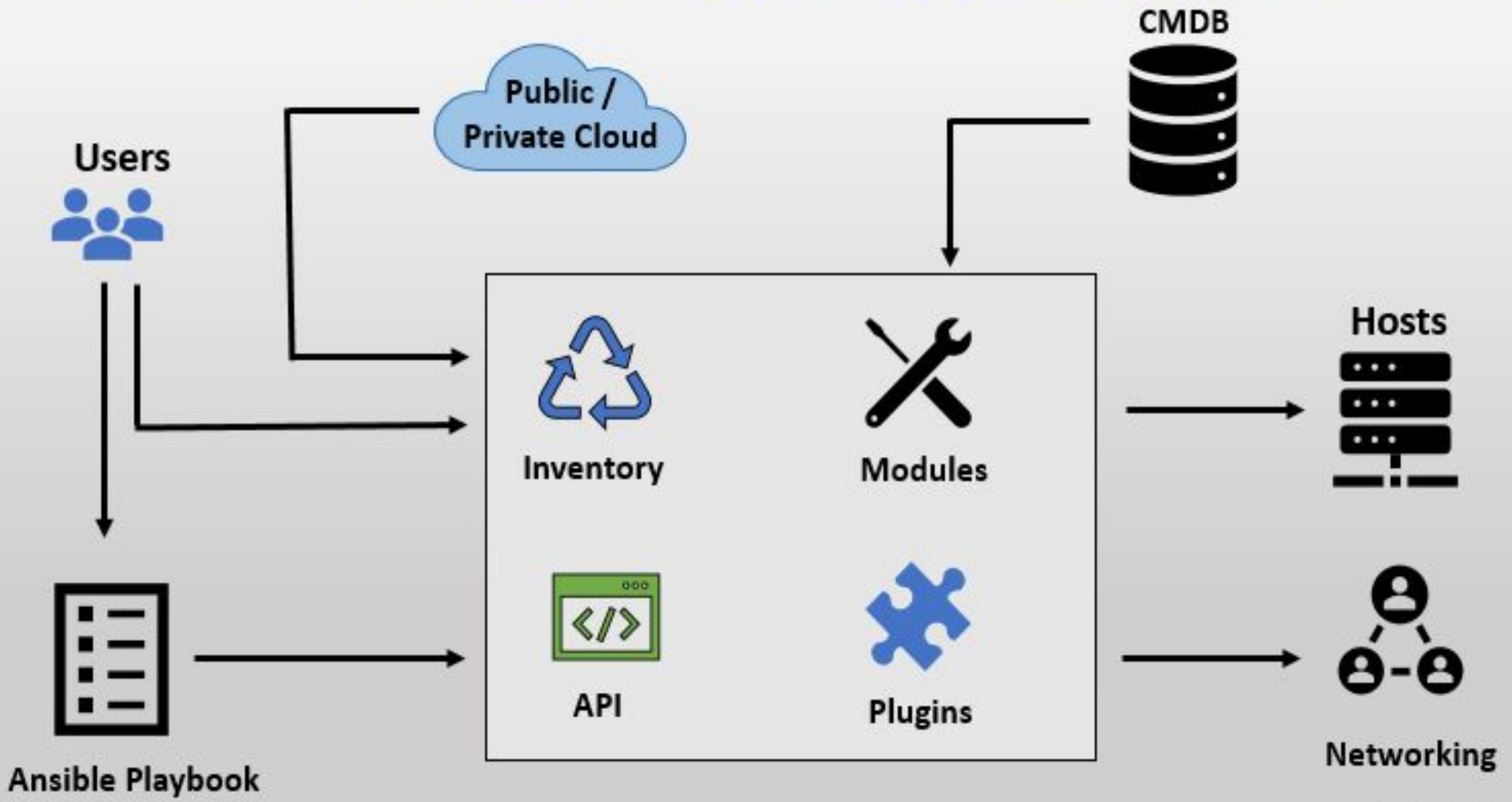
Ansible Intro

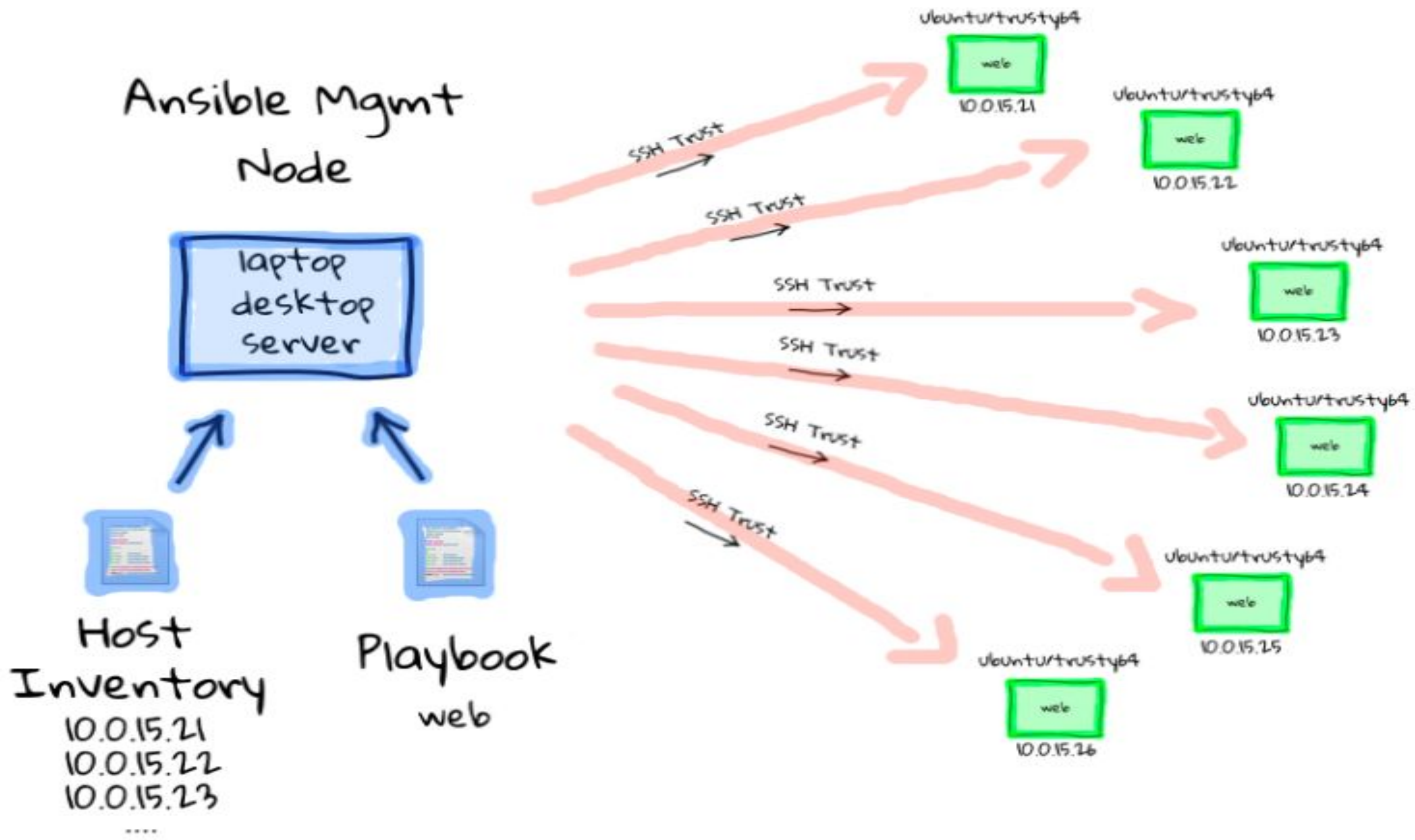
- Ansible is an open-source configuration management and provisioning tool, similar to Chef, Puppet or Salt.
- It uses SSH to connect to servers and run the configured Tasks. Ansible lets you control and configure nodes from a single machine.
- What makes it different from other management software is that Ansible uses SSH infrastructure. The project was founded in 2013 and bought by Red Hat in 2015.

Why Ansible?

- **No Agent-** As long as the box can be ssh'd into and it has python, it can be configured with Ansible.
- **Idempotent-** Ansible's whole architecture is structured around the concept of idempotency. The core idea here is that you only do things if they are needed and that things are repeatable without side effects.
- **Declarative Not Procedural-** Other configuration tools tend to be procedural do this and then do that and so on. Ansible works by you writing a description of the state of the machine that you want and then it takes steps to fulfill that description.
- **Tiny Learning Curve-** Ansible is quite easy to learn. It doesn't require any extra knowledge.

Ansible Architecture




Ansible Mgmt Node
laptop
desktop
server
Host Inventory
10.0.15.21
10.0.15.22
10.0.15.23
....
Playbook
web
SSH Trust
SSH Trust
SSH Trust
SSH Trust
SSH Trust
SSH Trust
ubuntu/trusty64
web
10.0.15.21
ubuntu/trusty64
web
10.0.15.22
ubuntu/trusty64
web
10.0.15.23
ubuntu/trusty64
web
10.0.15.24
ubuntu/trusty64
web
10.0.15.25
ubuntu/trusty64
web
10.0.15.26

Inventory

The Inventory is a description of the nodes that can be accessed by Ansible. By default, the Inventory is described by a configuration file, whose default location is in./etc/ansible/hosts The configuration file lists either the IP address or hostname of each node that is accessible by Ansible.

Every host is assigned to a group such as web servers, db servers etc. The inventory file can be in one of many formats such as yaml, INI etc

Playbook

Playbooks are simple YAML files. These files are descriptions of the desired state of your systems. Ansible then does the hard work of getting your systems to that state no matter what state they are currently in. Playbooks make your installations, upgrades and day-to-day management repeatable and reliable.

Playbooks are simple to write and maintain. Playbooks are written in a natural language so they are very easy to evolve and edit.

Playbook contains Plays.

Plays contain tasks.

tasks call modules.

Modules

There are over 1000 modules provided by Ansible to automate every part of the environment. Modules are like plugins that do the actual work in Ansible, they are what gets executed in each playbook task.

Each module is mostly standalone and can be written in a standard scripting language (such as Python, Perl, Ruby, Bash, etc.). One of the guiding properties of modules is idempotency, which means that even if an operation is repeated multiple times, it will always place the system into the same state.

Examples of Modules

There are lots of modules such as :

Service, file, copy, iptables etc.

Any Module can be used as :

```
ansible 127.0.0.1 -m service -a "name=httpd state=started"
ansible localhost -m ping
```

Roles

Roles are a way to group tasks together into one container. We could have a role for setting up MySQL, another one for configuring iptables etc.

Roles makes it easy to configure hosts. Any role can be performed on any host or group of hosts such as:

```
- hosts: all
  roles:
     - role_1
     - role_2
```

Thank You