COLLEGE OF ELECTRICAL AND MECHANICAL ENGINEERING

DEPARTMENT OF SOFTWARE ENGINEERING

Software defined system.

Title: software defined system configuration tool puppet

Group members	Id
Kokebe Negalgn	ETS 0419/12
Kaleab Girma	ETS 0378/12
Mastewal Tesaye	ETS0435/12

Submitted to – Mr. Biniyam B.

Submission date April 30,2024

What is a Puppet?

Puppet is a tool that **helps you manage and automate the configuration of servers**. When you use Puppet, you define the desired state of the systems in your infrastructure that you want to manage.

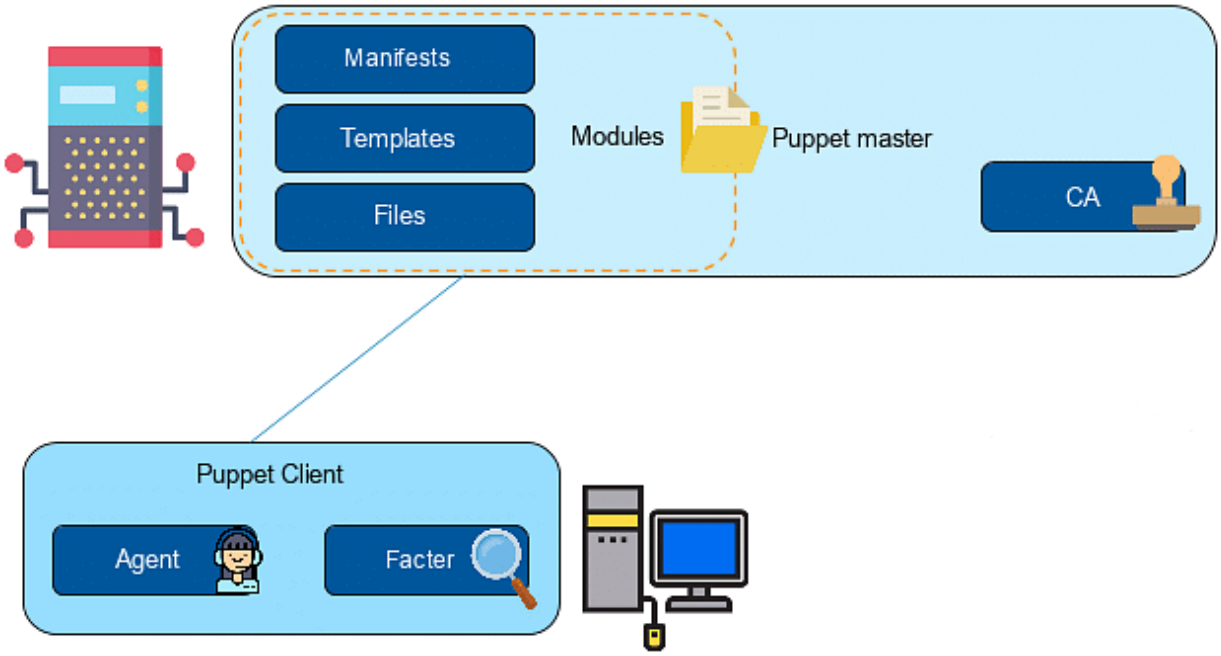
Puppet is a configuration management tool ensuring that all systems are configured to a desired and predictable state.

You can also use puppet as a development tool as it can automatically deploy software on the system. Puppet implements infrastructure as code, which means you can test the environment and ensure that it is deployed accurately.

Using a master-agent architecture, a Puppet master server controls and distributes configuration data to Puppet agents running on target nodes or servers. The Puppet master server stores configuration data in the form of manifests and modules that provide recommended setup and maintenance procedures for systems.

Puppet can help IT departments automate and expedite tasks including software installation, configuration management, user account management, and compliance auditing. With its centralized platform, it ensures consistency and reduces the likelihood of errors or configuration drift by managing configurations across several servers or nodes.

Components of Puppet



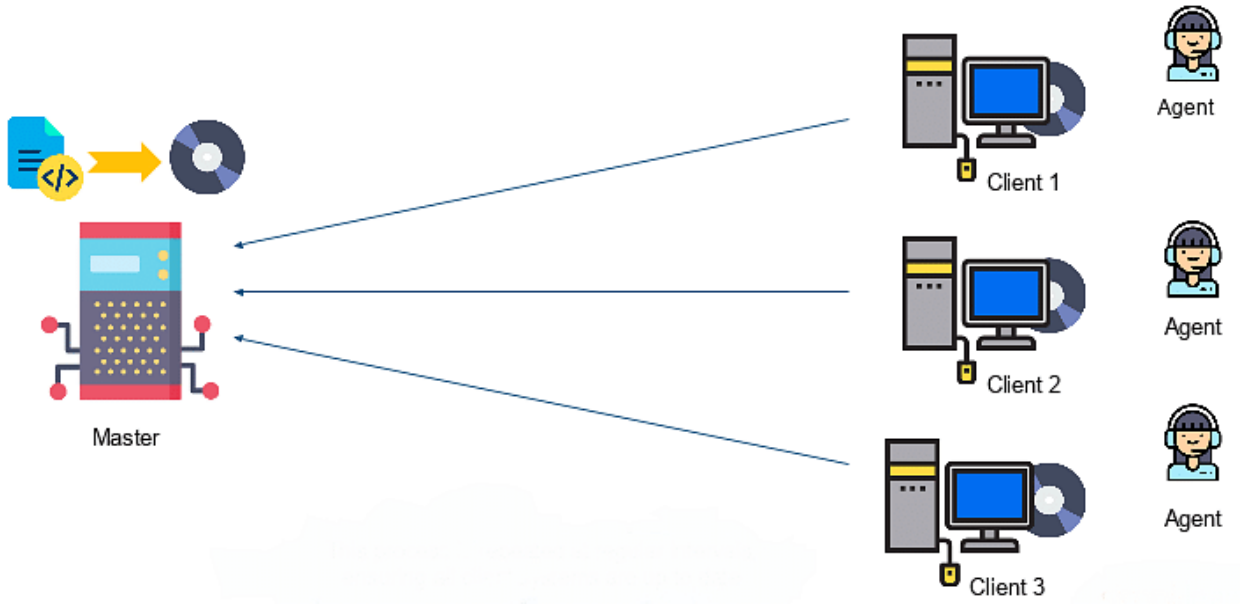
The Puppet environment can be broken down into the main server environment (shown above) and the client environment. In the main server environment, there is a Puppet master store which stores all configuration files.

- Manifests are the actual codes for configuring the clients
- Templates combine code and data to render a final document
- Files are the static content that can be downloaded by the clients
- Modules are a collection of manifests, templates, and files
- Certificate authority allows the master to sign the certificates sent by the client

The Puppet client is the machine that needs to be configured, consisting of Agent and Factor. The agent continuously interacts with the master server to ensure that the certificates are being updated appropriately. The factor collects the current state of the client that is used and communicates it back through the agent.

How Puppet Works?

Puppet has a primary-secondary node architecture.



The clients are distributed across the network and communicate with the primary-secondary environment where Puppet modules are present. The client agent sends a certificate with its ID to the server; the server then signs that certificate and sends it back to the client. This authentication allows for secure and verifiable communication between the client and the master.

The factor then collects the state of the clients and sends it to the master. Based on the fact sent, the master compiles the manifests into the catalogs, which are sent to the clients, and an agent executes the manifests on its machine. A report is generated by the client that describes any changes made and is sent to the master.

This process is repeated at regular intervals, ensuring all client systems are up to date. In the next section, let us find out about the various companies adopting Puppet as a part of our learning about what is Puppet.

Companies Adopting Puppet

Your interest in understanding what is Puppet would grow if you knew about the companies that have adopted it to manage their infrastructure. Some of them include:

- Spotify
- Google
- AT&T
- Staples
- AON
- The U.S. Air Force

The reasons why these companies adopted Puppet may vary. For example, Staples used Puppet as a configuration management tool to automate its private cloud management and IT operations to provide consistency, allowing their IT teams more time to innovate.

Writing Manifests in Puppet

Files written for configuring a node are called manifests, compiled into catalogs that are executed at the client. Each of the manifests is written in Ruby language with a. pp extension.

The five key steps to write a manifest are:

1. **Create** - Manifests are written by the system administrator.
2. **Compile** - Manifests are compiled into catalogs.
3. **Deploy** - Catalogs are deployed onto the clients.
4. **Execute** - Catalogs are run on the client by the agent.
5. **End** - Clients are configured to the desired state.

All manifests must have a common syntax:

Resourcetype { ‘title’ :

Attribute_name1 => attribute_value

Attribute_name2 => attribute_value

}

Here, the resource type can be a package, a file, or a service.

‘title’ refers to the name of the resource type.

Attribute_name1 - This is the feature whose value needs to be changed or set (for example: IP, ensure).

attribute_value - This is the new value for the attribute (for example: present, start)

A manifest can contain multiple resource types. The keyword ‘default’ applies manifest to all the clients. For example:

Node default {

file { ‘/etc/sample’ :

Content => “this is a sample manifest”

}

service { ‘httpd’ :

ensure => installed

}

}

Our first resource is a file path ‘/etc/sample.’

The specified content is written into the file. The file is created first if it does not already exist. Our next resource is the Apache service, which is to be installed on the client node. The manifest is deployed on the client machine. The client now has a file named ‘sample’ under ‘etc’ folder and the Apache server will be installed.

Advantages of puppet.

Automation: By eliminating the need for manual configurations and boosting productivity, Puppet enables the automation of repetitive processes.

Consistency: Puppet lowers the possibility of mistakes or inconsistencies by ensuring that configurations are consistent across all environments.

Scalability: Puppet can grow to handle settings for a big number of servers or nodes with ease.

Simple management: Puppet makes it simpler to monitor and control infrastructure by offering a centralized platform for managing configurations.

A puppet's extensibility allows it to be expanded with unique modules and plugins to satisfy certain needs.

The drawbacks of utilizing Puppet

Complexity: Puppet can be difficult to set up and configure, especially for novices, and has a steep learning curve.

Resource-intensive: Puppet may not be appropriate for smaller environments or situations with limited resources because it needs specialized resources to function effectively.

Dependency: Puppet depends on a stable network connection, and any disruptions to that connection could cause problems.

Restricted support: Puppet may not provide enough documentation or support for some configurations or unique needs.

Cost: A licensing charge for Puppet Enterprise, the commercial version of Puppet, may be necessary. This could be a drawback for smaller businesses or those with a limited budget.