1. Maven logo.svg

Phát triển bởi Apache Software Foundation.

Maven là một công cụ biên dịch tự động chủ yếu dùng cho các dự án Java, nhưng nó cũng có thể được dùng để xây dựng và quản lý các dự án dùng C#, Ruby, Scala hay ngôn ngữ khác.

Maven dùng khái niệm Project Object Model (POM) để mô tả việc build project, các thành phần phụ thuộc và các module. Nó định nghĩa trước các target cho việc khai báo task, trình biên dịch, đóng gói và thứ tự hoạt động để mọi việc diến ra tốt nhất.

Maven tải các thư viện, plug-in từ 1 hoặc nhiều repositories. Maven cung cấp việc hỗ trợ tải file từ repository trung tâm của Maven 2 và các repositories khác. Cũng có thể upload các sản phẩm lên repository sau khi đã build xong.

Maven được xây dựng dựa trên kiến trúc plugin - cho phép sử dụng bất kỳ ứng dụng điều khiển nào thông qua đầu vào tiêu chuẩn. Về mặt lý thuyết, điều này cho phép bất kỳ ai cũng có thể viết các plugin để giao tiếp với các build tool (trình biên dịch, unit test tool, etc) cho bất kỳ ngôn ngữ nào khác. Nhưng trong thực tế, hỗ trợ và sử dụng cho các ngôn ngữ khác đã bị thu hẹp. Hiện tại thì plugin cho.Net đã có và vẫn được duy trì, và một plugin cho C/C++ cũng được duy trì cho Maven 1

Từ phiên bản 2.76 trở đi Jenkins đã mặc định cho phép tạo dự dán theo kiểu Maven, người dùng chỉ cần cài đặt Maven (phiên bản 3.3.9 hoặc mới hơn) và thiết lập vị trí của Maven cho Jenkins.

1. 

Puppet nổi tiếng trong việc cấu hình hàng loạt đi kèm với ngôn ngữ riêng – ngôn ngữ cấu hình (configuration language).

[Puppet](http://www.gocit.vn/bai-viet/tag/puppet/) là 1 phần mềm quản lý cấu hình dựa trên Ruby, và được phân phối dưới bản quyền Apache 2.0, nó có thể chạy trên 2 chế độ là client-[server](http://www.gocit.vn/bai-viet/tag/server/) hoặc stand-alone.

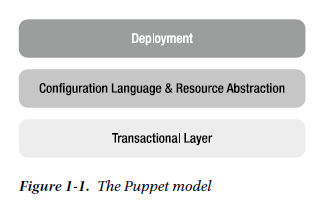
Puppet được phát triển bởi Luke Kanies và giờ được phát triển bởi công ty của anh ấy, Puppet Labs. Kanies đã làm việc với [Unix](http://www.gocit.vn/bai-viet/tag/unix/) và vai trò quản trị hệ thống từ năm 1997 và phát triển Puppet dựa trên kinh nghiệm đó.

Puppet có 2 phiên bản: phiên bản mã nguồn mở (open source) và phiên bản doanh nghiệp (Enterprise). Phiên bản Enterprise bao gồm  1 bộ cài đặt tự động, 1 giao diện quản lý web, và thông tin liên lạc hỗ trợ.

Puppet có thể được dùng để quản lý cấu hình trên các nền tảng Unix (bao gồm cả Mac OS X), Linux và [Microsoft](http://www.gocit.vn/bai-viet/tag/microsoft/) [Windows](http://www.gocit.vn/bai-viet/tag/windows/).

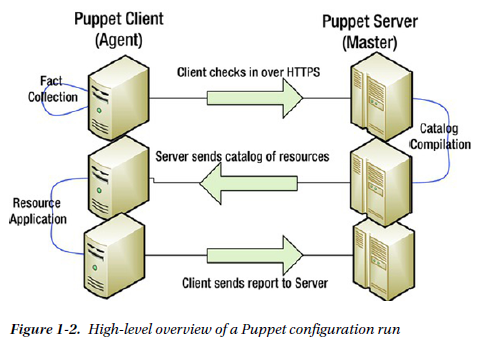
Mô hình hoạt động của Puppet gồm 3 thành phần:

* Tầng triển khai (Deployment Layer)
* Ngôn ngữ cấu hình và tầng trừu tượng tài nguyên. (Configuration language and Resource Abstraction Layer)
* Tầng giao dịch (Transactional Layer)



**1.       Deployment Layer**

Puppet thường được triển khai theo mô hình client-server. Trong đó Puppet server được gọi là *Puppet Master*, còn Puppet client được gọi là *Agent*, và mỗi host được định nghĩa như là 1*node*.

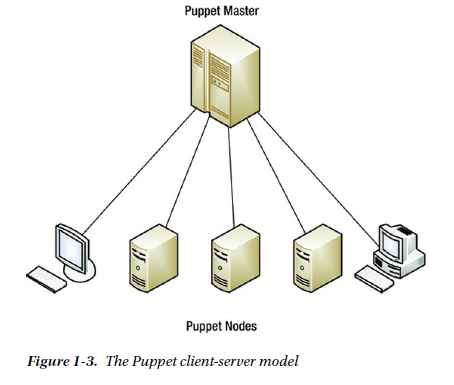
[](http://www.gocit.vn/wp-content/uploads/2015/01/puppet_configuration_run.png)

*Puppet agent* kết nối đến *Puppet master* qua 1 kết nối được mã hoá và chứng thực sử dụng SSL, để lấy về hoặc “đẩy lên” những cấu hình cần được kiểm tra và áp dụng trên agent.

Nếu *Puppet agent* không được cấu hình hoặc đã có cấu hình được yêu cầu, Puppet sẽ không thực hiện gì.

Puppet sẽ chỉ thay đổi môi trường của bạn nếu cần thiết. Thuộc tính này được gọi là*idempotency* và là tính năng then chốt của Puppet. Toàn bộ tiến trình trên được gọi là*configuration run*.

Mặc định, *Puppet agent* sẽ kiểm tra các cấu hình mới hoặc được thay đổi trên *Puppet Master*mỗi 30 phút 1 lần.



Bên cạnh mô hình triển khai theo client-server, chúng ta có thể triển khai Puppet theo mô hình stand-alone, khi đó không cần đến sự có mặt của Puppet master. Các cấu hình được cài đặt cục bộ trên host và các tập tin nhị phân của puppet sẽ chạy để thực thi và áp dụng các cấu hình này.

**2.      Configuration Language and Resource Abstraction Layer**

Puppet sử dụng 1 ngôn ngữ đặt tả, gọi là ngôn ngữ Puppet, để định nghĩa các cấu hình, mà Puppet gọi là *resource*. Đây là 1 sự khác nhau quan trọng giữa Puppet và nhiều công cụ cấu hình khác.

Sự khác biệt nằm ở chỗ là ngôn ngữ đặc tả mô tả về trạng thái của cấu hình – vd: nó mô tả là 1 gói phần mềm sẽ được cài đặt hoặc 1 dịch vụ sẽ được chạy. Phần lớn các công cụ cấu hình, như Shell script hoặc [Perl](http://www.gocit.vn/bai-viet/tag/perl/), quan tâm đến qui trình triển khai cấu hình, mà ít quan tâm đến trạng thái của cấu hình trên máy tính.

**2.1.            Configuration Language**

**Chuyện gì sẽ xảy ra nếu không có Configuration Language?**

Giả sử chúng ta cần cài đặt ứng dụng [Vim](http://www.gocit.vn/bai-viet/tag/vim/) trên 3 máy tính (host) chạy 3 hệ điều hành khác nhau là Microsoft Windows, Red Hat Enterprise Linux và [Ubuntu](http://www.gocit.vn/bai-viet/tag/ubuntu/), thì chúng ta cần phải viết kịch bản (script) cài đặt cho từng hệ điều hành, trong trường hợp này là 3 kịch bản, thực hiện các việc sau:

* Kết nối đến host tương ứng (bao gồm cả việc xử lý mật khẩu hoặc key).
* Kiểm tra ứng dụng Vim đã được cài đặt chưa?
* Nếu chưa, sử dụng các lệnh tương ứng cho từng nền tảng để cài đặt Vim – vd: sử dụng lệnh [yum](http://www.gocit.vn/bai-viet/tag/yum/) trong RHEL, apt-get trong Ubuntu …
* Tiếp nhận các thông báo trả về cho từng nền tảng để đảm bảo ứng dụng đã được cài thành công.

**Và khi sử dụng Puppet và Configuration Language?**

Chúng ta thực hiện công việc trên bằng việc định nghĩa 1 tài nguyên cấu hình (configuration resource) cho gói phần mềm vim như bên dưới:

*package { ‘vim’:*

*ensure => present,*

*}*

Cú pháp để định nghĩa 1 resource trong Puppet là:

*type* { *title*:

*attribute* => *value*,

}

Trong đó:

* **type**: dạng tài nguyên (resource) sẽ được quản lý như: *package*, *service* hoặc *cron jobs*.
* **title**: tên của tài nguyên (trong ví dụ trên là tên package cần cài đặt).
* Và 1 danh sách các cặp khoá **attribute => value** của resource đó, với ***attribute*** là tên thuộc tính của resource và ***value*** là giá trị cần có của thuộc tính đó.

Danh sách đầy đủ các loại *resource* mà Puppet quản lý (và các thuộc tính của chúng) có thể xem tại:

<http://docs.puppetlabs.com/references/stable/type.html>

*Resource* trên sau đó sẽ được triển khai xuống các *Puppet agent*, chúng ta không cần quan tâm đến sự khác nhau giữa các nền tảng cũng như các kết quả trả về khác nhau, Puppet sẽ giúp chúng ta thực hiện việc này.

**2.2.            Resource Abstraction Layer**

Mỗi loại *resource* được định nghĩa trong Puppet có nhiều nhà cung cấp (*provider*) tương ứng với nó. Ví dụ với loại *resource* là *package*, tuỳ theo từng nền tảng và hệ điều hành, mà có hơn 20 *provider* bao hàm nhiều loại công cụ khác nhau, bao gồm *yum*, *aptitude*, *pkgadd*,*ports*, và *emerge*.

Khi 1 *agent* kết nối, Puppet sử dụng 1 công cụ gọi là **Facter** để trả về thông tin về *agent* đó, bao gồm cả thông tin về hệ điều hành mà *agent* đó đang chạy. Puppet sau đó chọn nhà cung cấp gói phần mềm tương ứng cho hệ điều hành đó và sử dụng nhà cung cấp đó để kiểm tra gói vim đã được cài đặt chưa.Vd: trên Red Hat, nó sẽ thực thi yum, trên Ubuntu là aptitude, và trên [Solaris](http://www.gocit.vn/bai-viet/tag/solaris/) thì lệnh pkgadd được sử dụng. Nếu gói phần mềm chưa được cài, Puppet sẽ cài nó. Nếu gói phần mềm cài rồi, Puppet không làm gì cả.

Puppet sau đó sẽ báo cáo kết quả thành công hoặc thất bại cho *Puppet master*.

**3.      The Transactional Layer**

Tầng giao dịch là đông cơ của Puppet, thực hiện việc hoàn thành tiến trình cấu hình từng máy tính trong hệ thống, bao gồm những bước sau:

* Phiên dịch và biên dịch cấu hình.
* Truyền tải cấu hình được biên dịch đến *agent*.
* Áp dụng cấu hình trên lên *agent*.
* Báo cáo kết quả áp dụng cấu hình đến *Puppet master*.

Đầu tiên Puppet sẽ phân tích cấu hình của chúng ta và tính toán như  thế nào để áp dụng nó lên *agent*. Để làm được điều này Puppet tạo 1 biểu đồ thể hiện tất cả *resource*, với mối quan hệ của chúng với nhau và với từng *agent*. Điều này cho phép Puppet làm việc theo thứ tự, dựa trên những mối quan hệ đã tạo, để áp dụng từng *resource* đến máy tính trong hệ thống. Mô hình này là 1 trong những tính năng mạnh nhất của Puppet.

Sau đó Puppet lấy *resource* và biên dịch chúng vào 1 *catalog* cho từng *agent*. *Catalog* này được gửi đến máy tính và được áp dụng bởi *Puppet agent*. Kết quả của việc áp dụng này sau đó được gửi về *Puppet master*.

Tầng giao dịch cho phép các cấu hình được tạo và được áp dụng lặp lại trên 1 máy tính.

Puppet không phải là giao dịch đầy đủ, các giao dịch của chúng ta không được ghi nhận lại, và do đó chúng ta không thể quay ngược lại các giao dịch  (*roll back*) như chúng ta có thể làm với các hệ cơ sở dữ liệu.

Nguồn: <http://www.gocit.vn/bai-viet/puppet-gioi-thieu/>