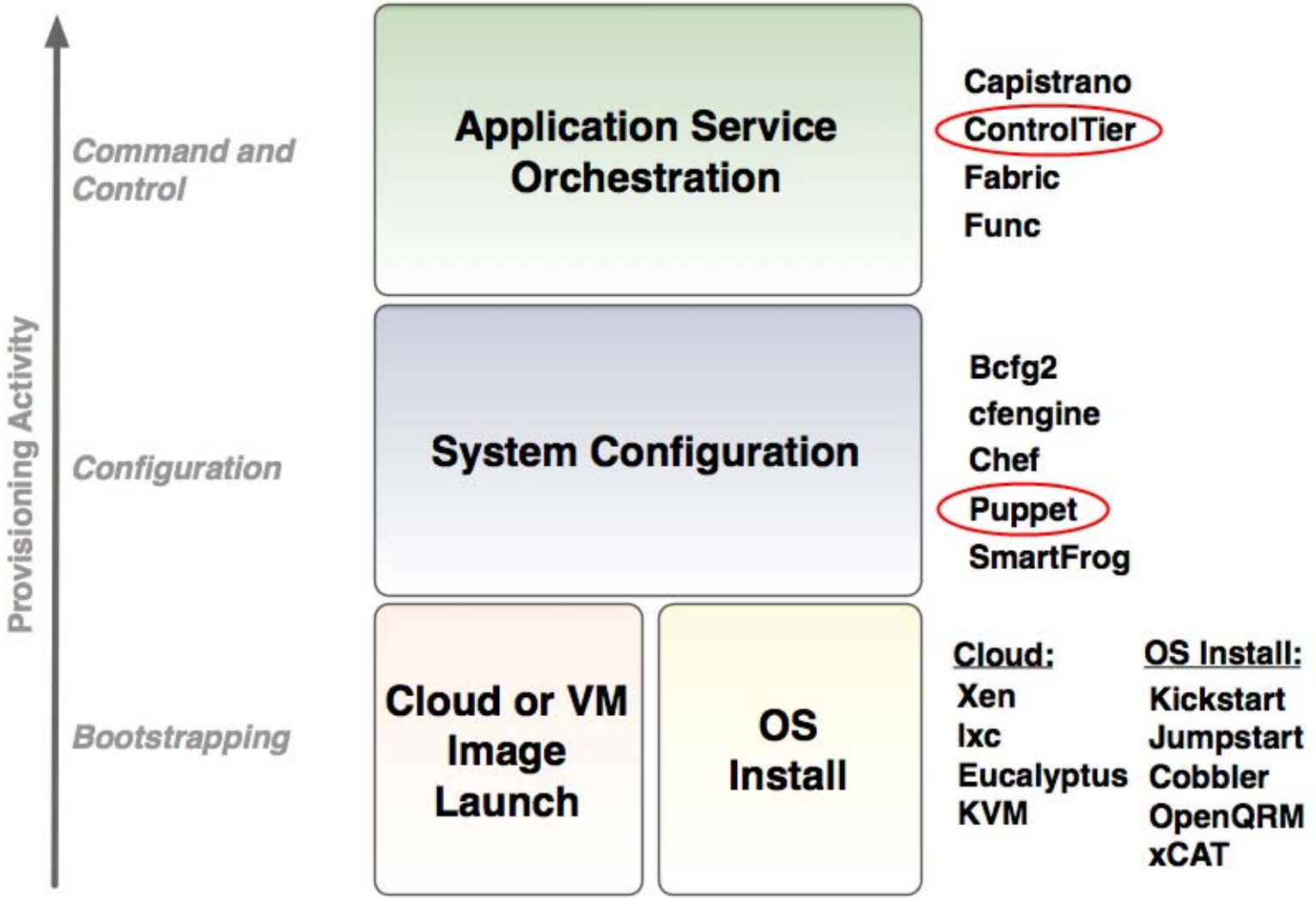
马哥教育

Operations

主讲：马永亮(马哥)
QQ:113228115
客服QQ：2813150558，1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com


Provisioning Activity
Command and Control
Configuration
Bootstrapping
Application Service Orchestration
System Configuration
Cloud or VM Image Launch
OS Install
Capistrano
ControlTier
Fabric
Func
Bcfg2
cfengine
Chef
Puppet
SmartFrog
Cloud:
Xen
lxc
Eucalyptus
KVM
OS Install:
Kickstart
Jumpstart
Cobbler
OpenQRM
xCAT

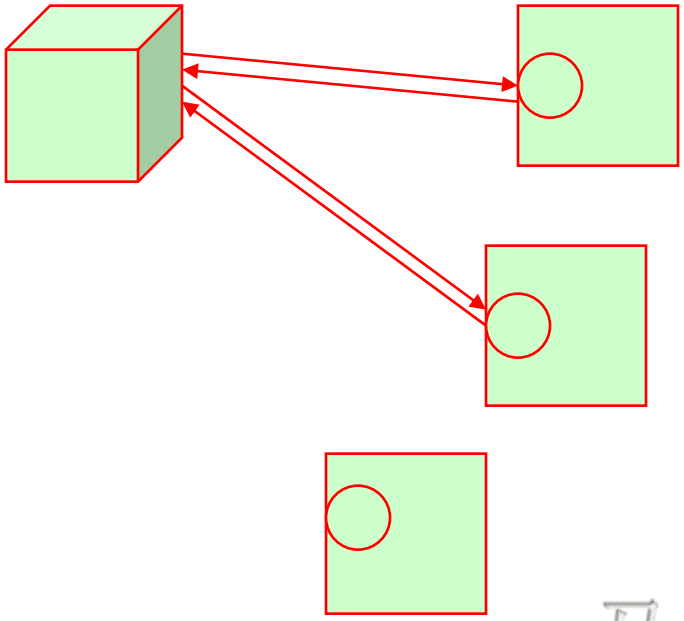
- ControlTier是基于"对目标进行操作（Activity）"的工具
- Puppet则是基于"定义目标状态"的工具

马哥教育

puppet

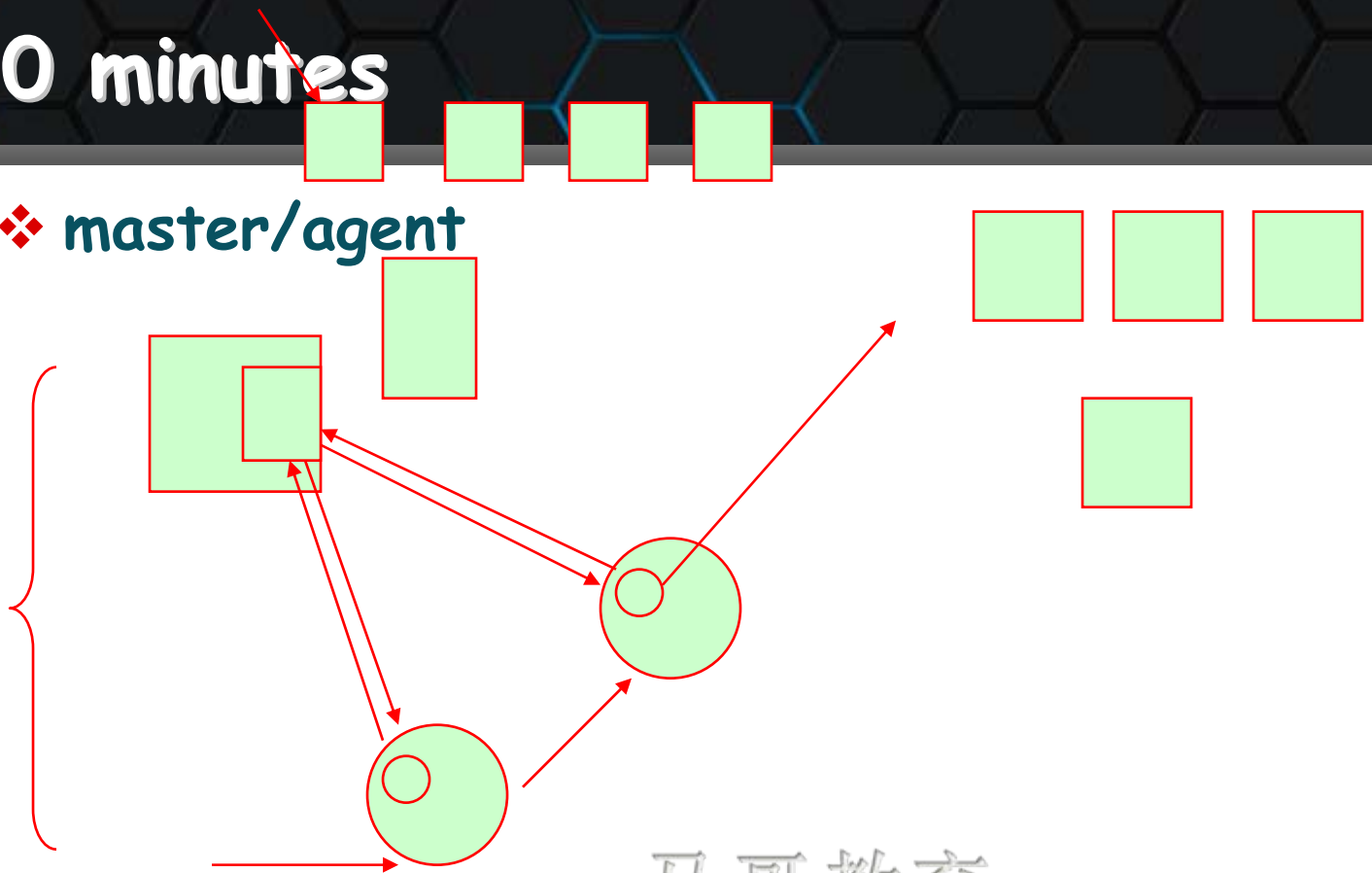
主讲：马永亮(马哥)
QQ:113228115
客服QQ：2813150558，1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

- ssl (https, xml-rpc, restful)



30 minutes

- master/agent



自动化运维技术—运维框架

自动化监控

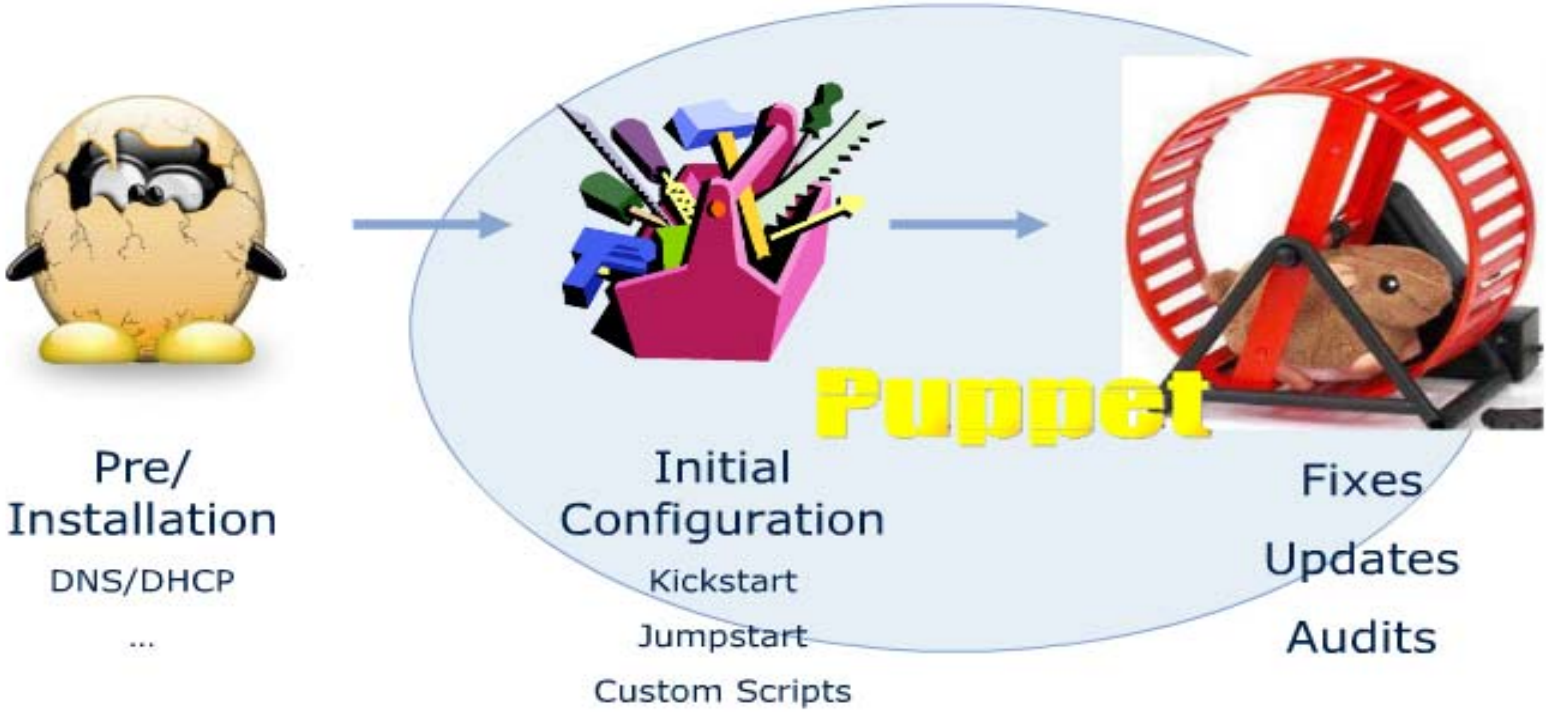

监控评估
数据采集（主动）
Client
公共插件
自定义脚本
服务状态探测（被动）
服务状态
程序状态
用户访问质量
第三方信息
公司内相关系统
数据处理
复杂计算
阈值判别
智能分析
报警与联动
报警策略
联动处理
报警跟踪
问题管理
API

内容概览

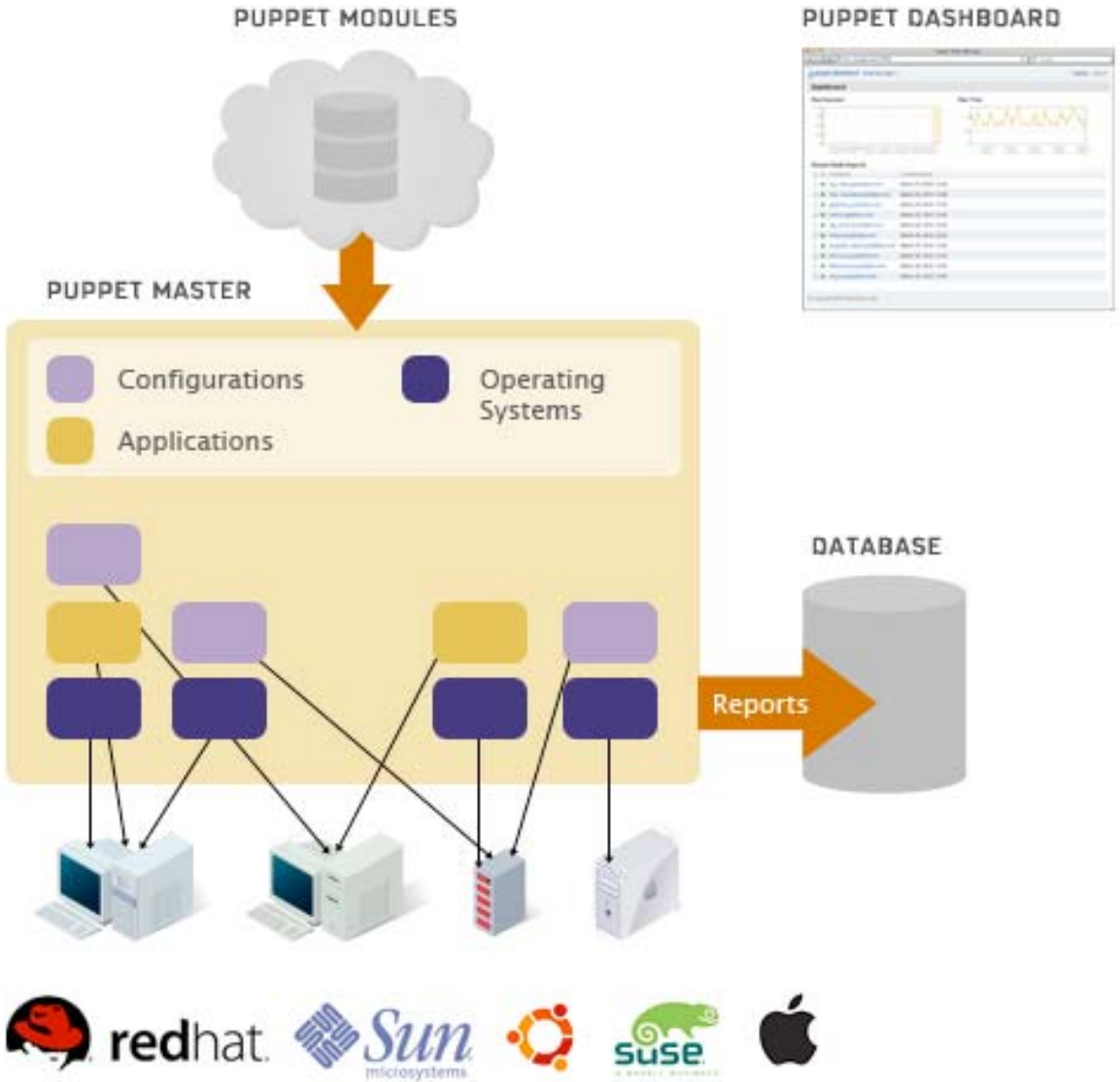
- puppet 介绍
  - 什么是puppet
  - puppet 能做什么
  - puppet 如何做到
  - puppet 目录结构介绍
- puppet 使用时注意事项
- puppet 应用案例
  - puppet 管理用户
  - puppet nginx 管理
  - puppet kick 介绍
  - puppet MCollective 介绍
- puppet 架构与集群
  - puppet 架构
  - puppet 集群方案
  - puppet 集群核心思想
- puppet性能测试方法

- puppet是一个IT基础设施自动化管理工具，它能够帮助系统管理员管理基础设施的整个生命周期：供应(provisioning)、配置(configuration)、联动(orchestration)及报告(reporting)
  - 基于puppet，可实现自动化重复任务、快速部署关键性应用以及在本地或云端完成主动管理变更和快速扩展架构规模等
- 遵循GPL协议(2.7.0-),基于ruby 语言开发
  - 2.7.0以后使用(Apache 2.0 license)
- 对于系统管理员是抽象的，只依赖于ruby与facter
- 能管理多达40多种资源，例如：file、user 、group、host、package、service、cron、exec、yum repo等，适合整个软件生命周期管理

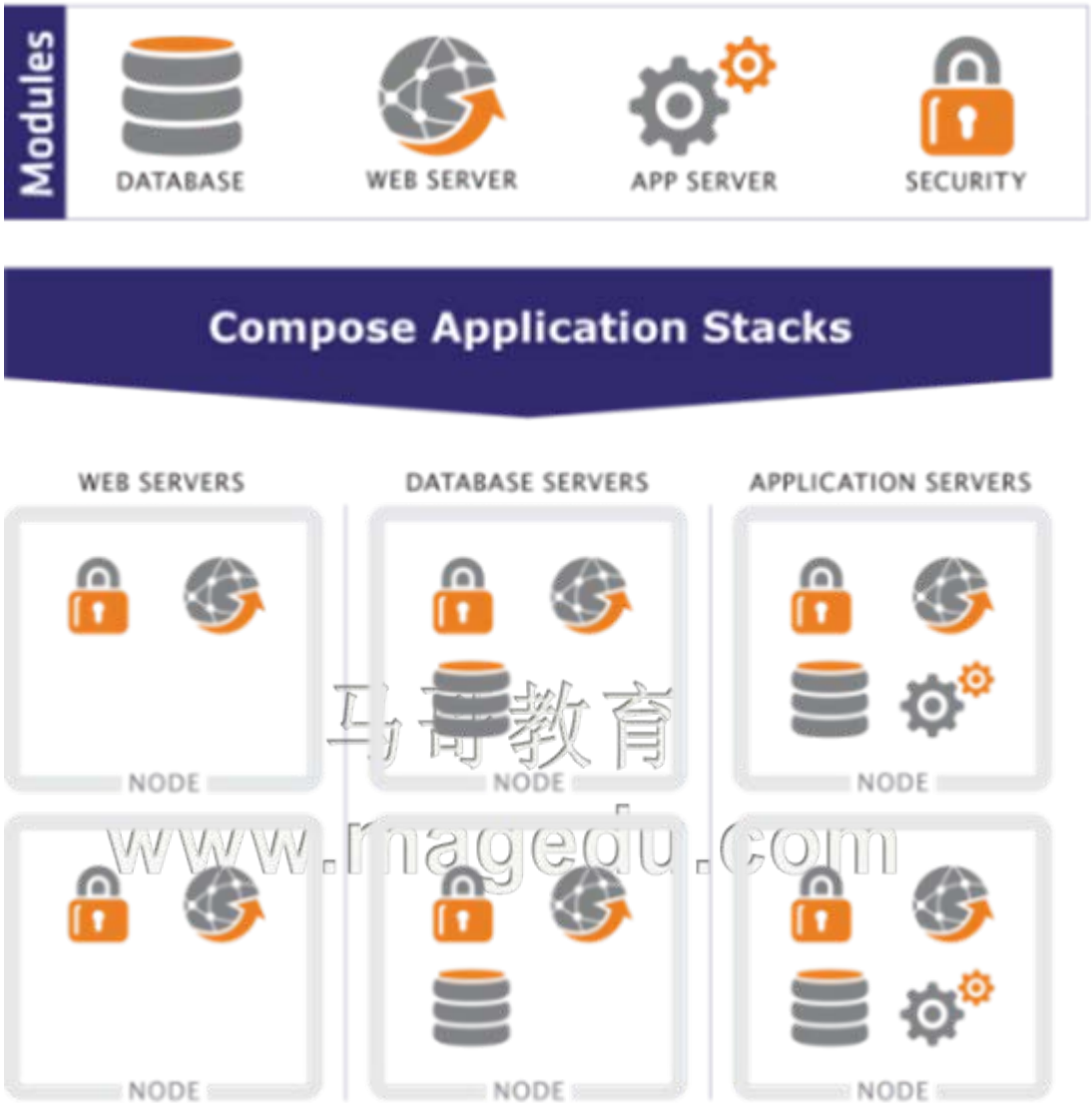
puppet适用的场景



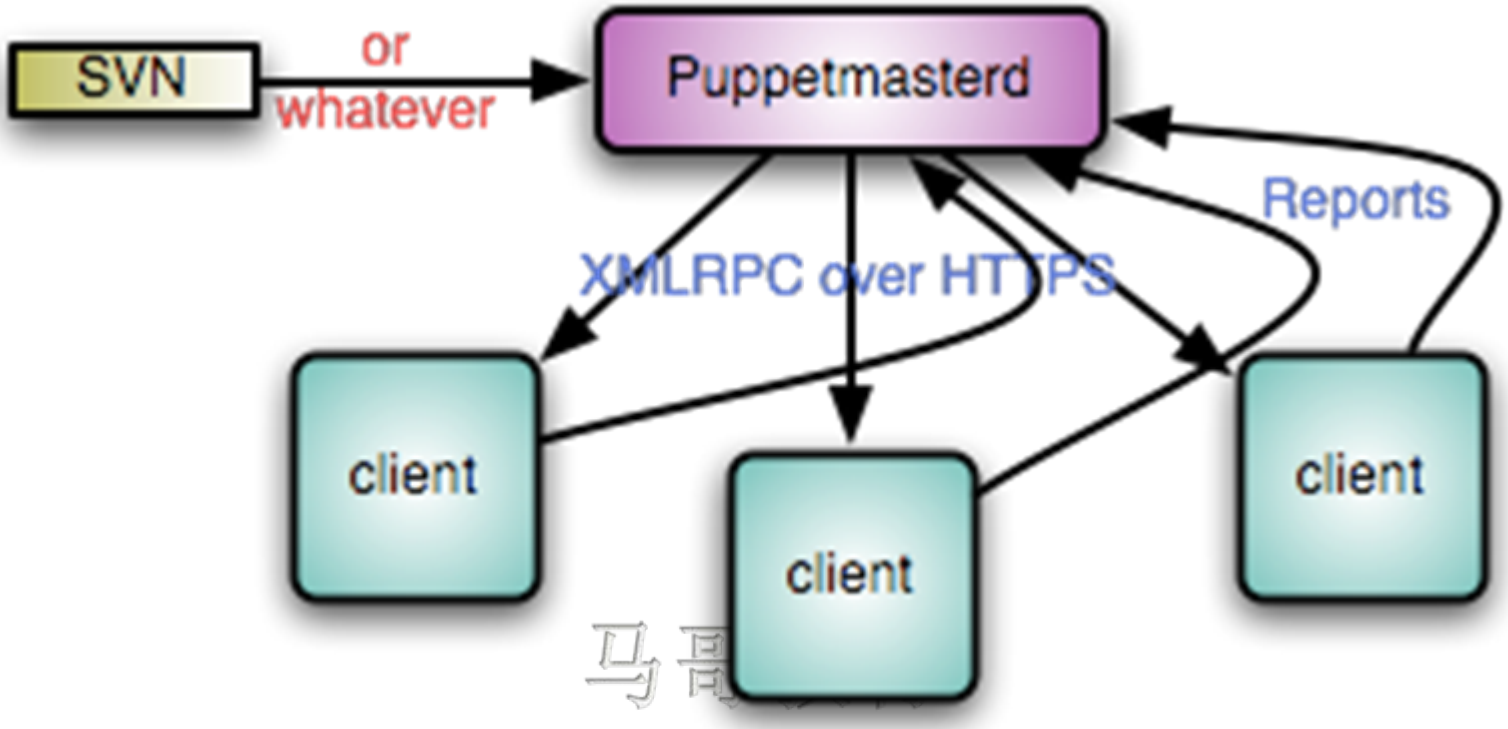
How does Puppet work?



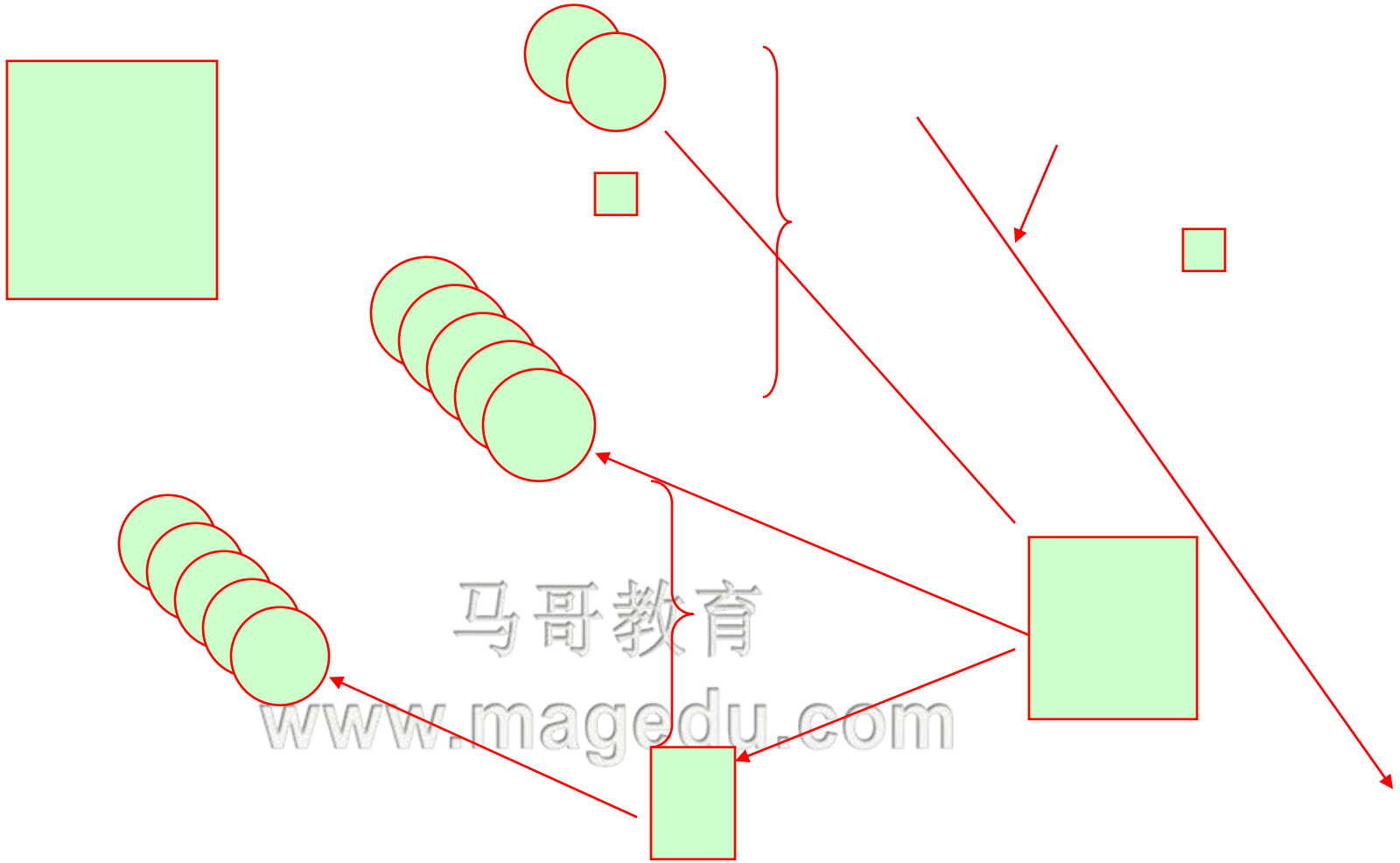
Define Reusable Configuration Modules



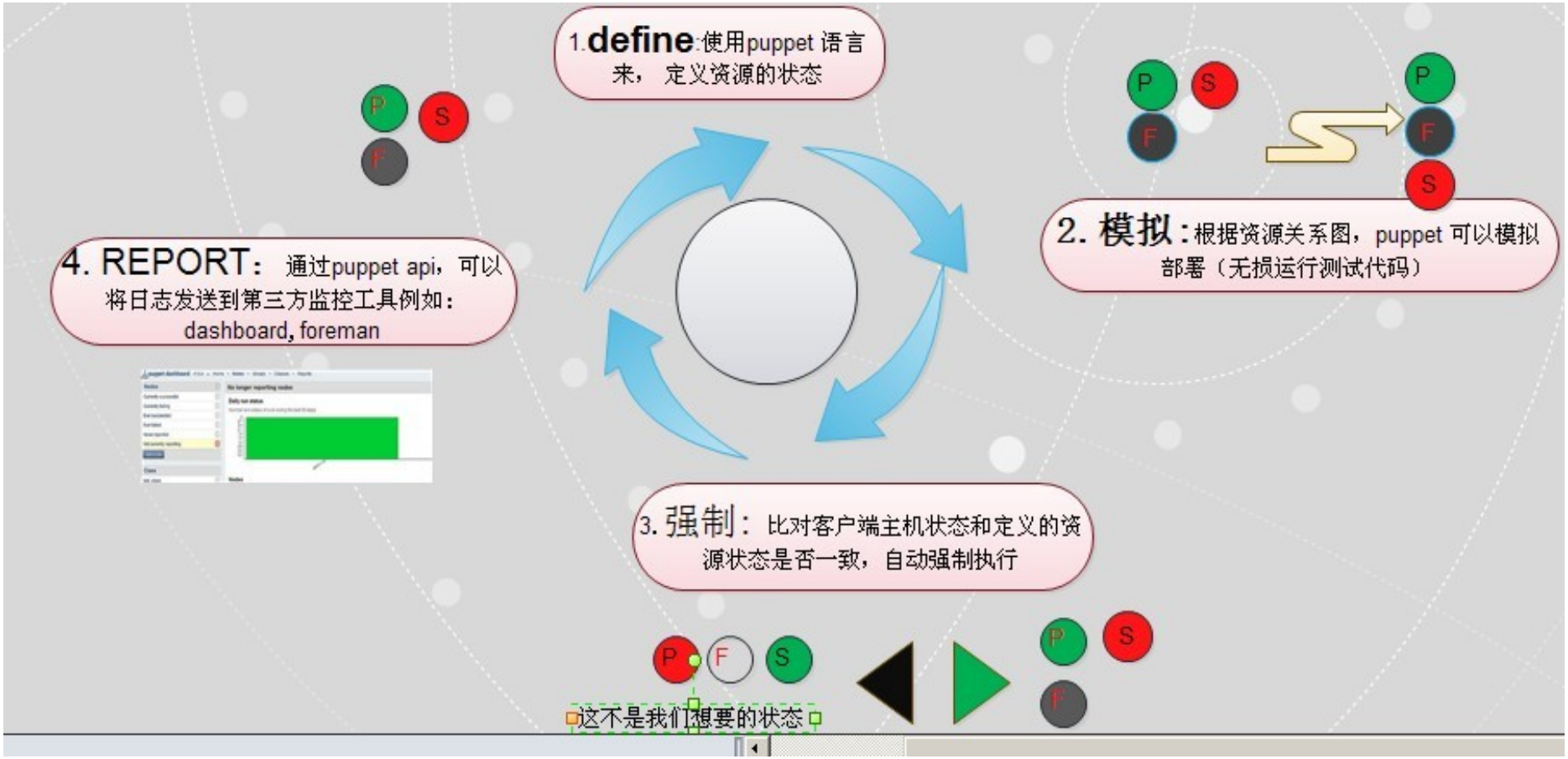
How does Puppet work?



版本管理系统



How does Puppet work?

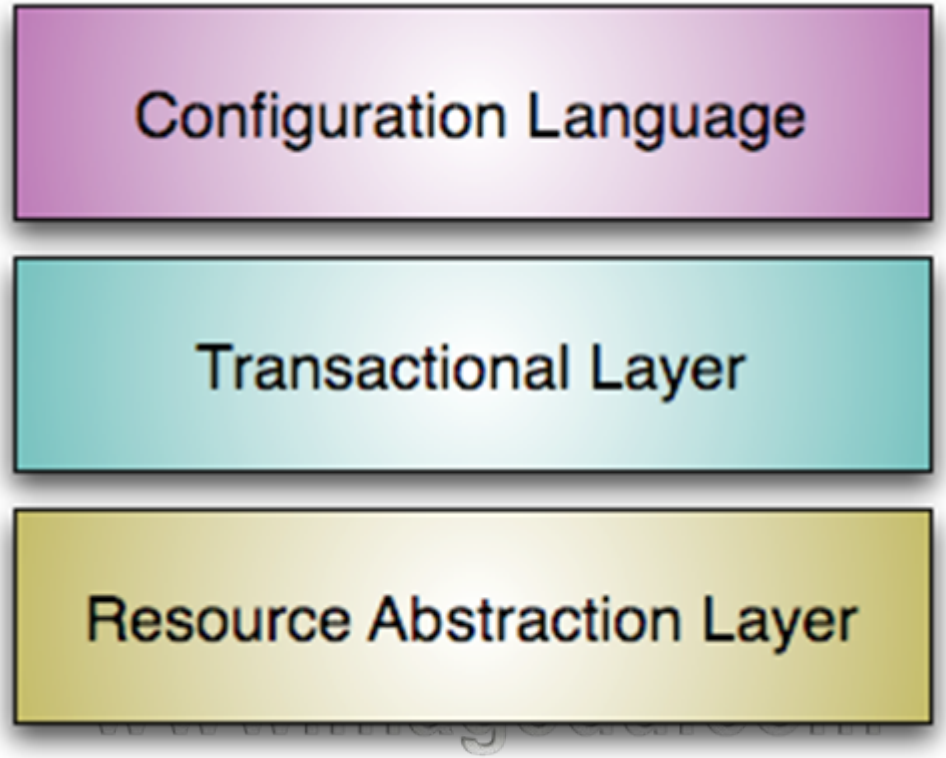


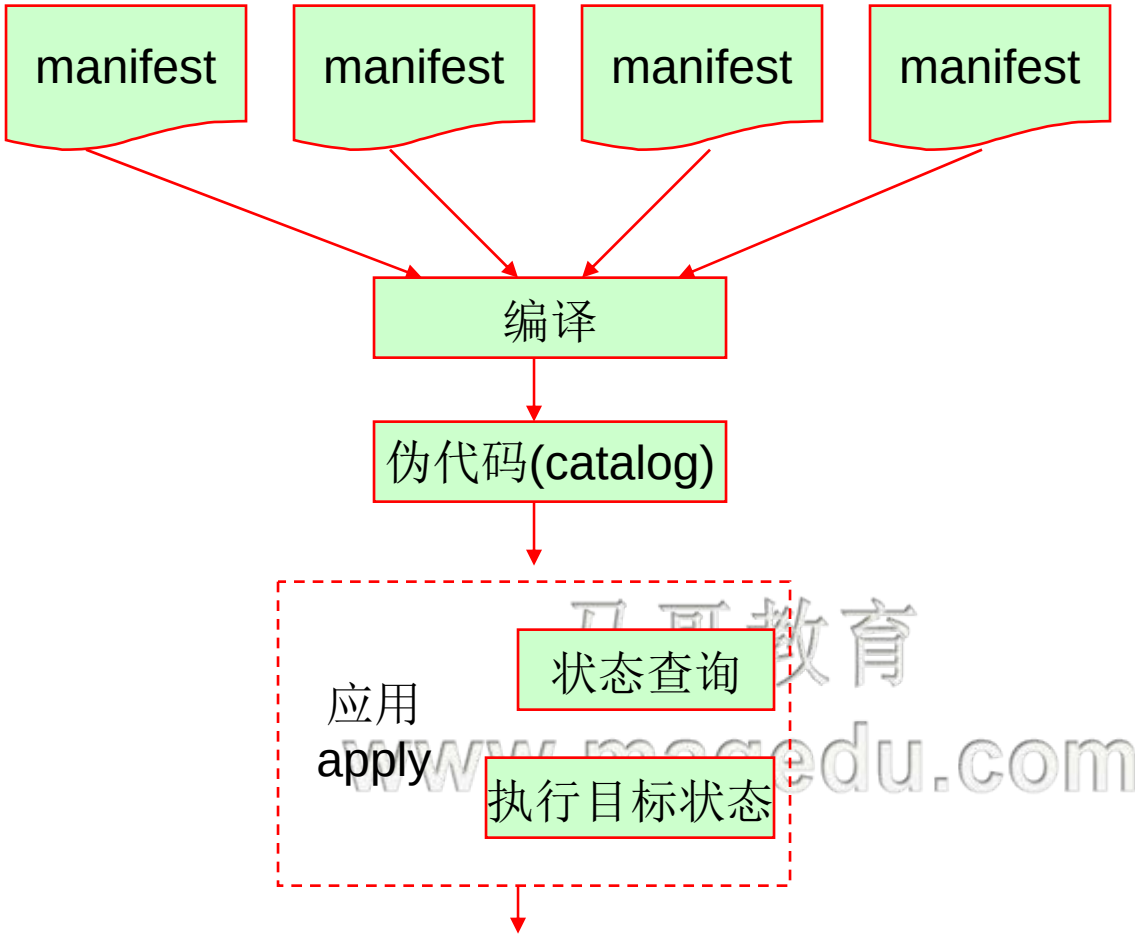
puppet的工作模型

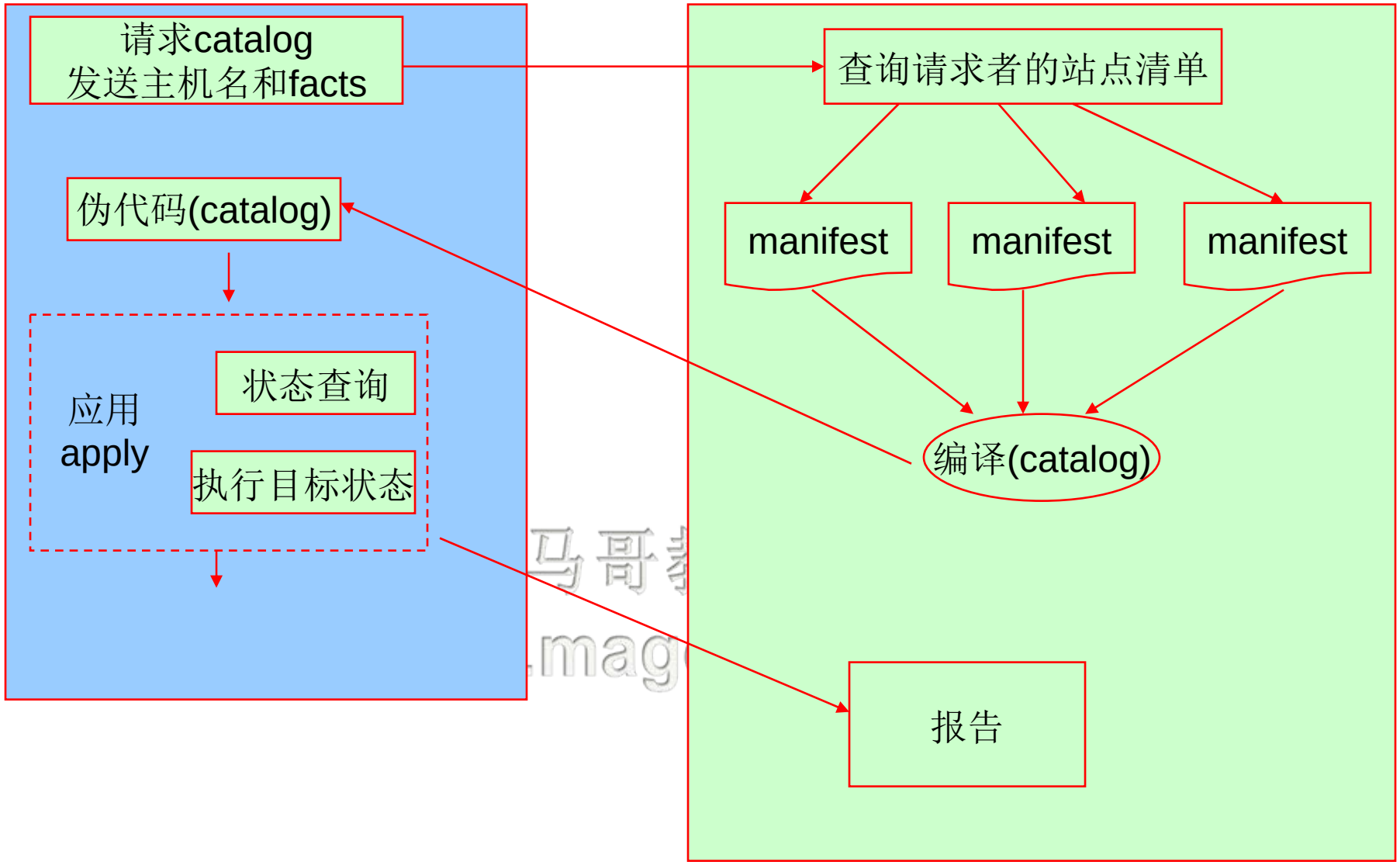
- puppet通过声明性、基于模型的方法进行IT自动化管理
  - 定义：通过puppet的声明性配置语言定义基础设置配置的目标状态；
  - 模拟：强制应用改变的配置之前先进行模拟性应用；dry-run
  - 强制：自动、强制部署达成目标状态，纠正任何偏离的配置；
  - 报告：报告当下状态及目标状态的不同，以及达成目标状态所进行的任何强制性改变；

Puppet Layers

- Puppet的三层模型






请求catalog
发送主机名和facts
伪代码(catalog)
应用
apply
状态查询
执行目标状态
查询请求者的站点清单
manifest
manifest
manifest
编译(catalog)
报告

- 抽象→事务→配置语言
- 资源申报(定义依赖关系)→类→模块
- 模块：类、文件、模板、使用样例

- 模块是通用的，调用类

- 定义节点：定义其要应用的类

- 资源定义时所在的文件：资源清单(manifest)
  - .pp

马哥教育

Puppet Installation and Configuration

主讲：马永亮(马哥)
QQ:113228115
客服QQ：2813150558，1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

Installing Puppet

PlatformPuppet	Server	Puppet Client
Debian	puppetmaster	puppet
Fedora	puppet-server	puppet
FreeBSD	puppet	NA
Gentoo	puppet	NA
OpenBSD	ruby-puppet	NA
Ubuntu	puppet	NA

Installing Puppet

PlatformFacter	Package Name
Debian	facter
Fedora	facter
FreeBSD	facter
Gentoo	facter
OpenBSD	ruby-facter
Ubuntu	facter

Installing Puppet

OS	Ruby	Ruby Libraries	Additional Package
Debian	ruby	libruby libopenssl-ruby	libxmlrpc-ruby
FreeBSD	ruby		
Gentoo	ruby		
Mandriva	ruby		
NetBSD	ruby		
OpenBSD	ruby		
Red Hat	ruby	ruby-libs	
SuSE	ruby		
Ubuntu	ruby	libruby libopenssl-ruby	libxmlrpc-ruby

马哥教育

使用puppet管理IT资源

主讲：马永亮(马哥)
QQ:113228115
客服QQ：2813150558，1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

两种使用模型

- 单机使用
  - puppet
- master/agent
  - master: puppet, puppet-server
  - agent: puppet

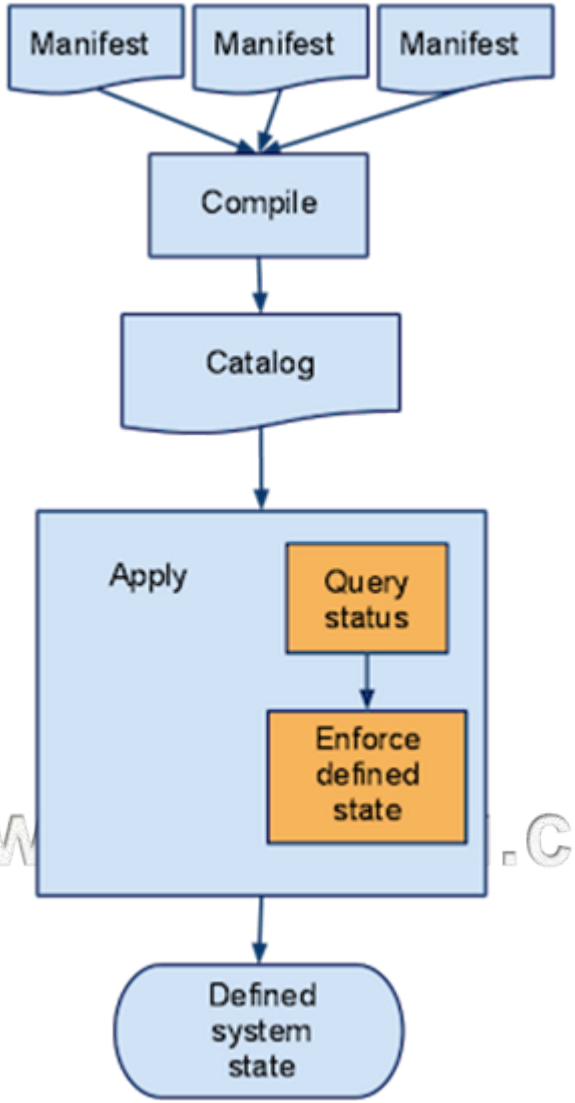
Agenda

- manifest
- puppet resource
- metaparameter
- variable
  - data type
  - scope
  - Facts and Built-In Variables
- condition statement
  - if
  - case
  - selector
- class
  - class
  - class declaration
  - parameter

Manifest

- puppet的程序文件称作"manifest(清单)"，以.pp作为文件名后缀
  - 如前所述，puppet语言的核心是"资源定义"，而定义一个资源的核心就在于描述其目标状态
  - 而manifest却还实现了常见的程序逻辑，如条件语句、资源集合及打印文件信息等功能
  - 简言之，也即resource定义在manifest文件中，或使用manifest文件定义resource
- "puppet apply"子命令能够将一个manifest中描述的目标状态强制实现，因此，它通常以manifest文件为参数

- puppet通过manifest同步资源时，不会直接应用manifest文件，而是要经过预先的编译过程，这是因为manifest文件中可能会包含条件判断、变量、函数及其它的程序逻辑
- 实际执行中，有可能会基于层次及包含关系存在多个manifest文件，这些文件会被编译进同一个称作"catalog"的文件
  - 编译完成后的catalog文件仅包含各个资源以及它们同步时的次序


Manifest
Manifest
Manifest
Compile
Catalog
Apply
Query status
Enforce defined state
Defined system state

- 在无puppet服务器的场景中仅于本地应用manifest时，前述过程似乎无特别之处，但在C/S架构的应用场景中其执行流程可能要复杂得多
  - 因为可以使用条件判断、变量及函数等程序逻辑，一个manifest文件可以同时适用于多个不同的系统；但catalog是为特定需求编译生成的执行文件，因此仅适用于一个系统；
  - 默认情况下，agent仅能从server端获取属于自己的catalog，它们无法看到适用于其它节点的信息；
  - 尽管catalog的应用机制明白无误，但还是可以使用"puppet agent --test --noop"在一个系统上模拟其执行过程而非强制应用；另外，还可以使用diff命令比较两个catalog的不同之处；

Site manifest

- 在puppet的C/S架构场景中，manifest的应用过程也与本地场景的方式有所不同
  - puppet agent通常运行为一个服务进程，其默认每隔半个小时向master发出一次连接请求
  - puppet agent并不直接访问任何manifest，而是向master请求一个预编译的catalog文件
  - puppet master会为发出请求的agent读取一个名为"site mainfest"的特殊manifest文件，并基于此编译一个catalog后发送给请求者
  - puppet agent在获取到catalog后应用于本地
- 因此，基于此种工作架构，仅在master端提供一个或少量几个manifest即可实现管理大量的节点，并能提供更加安全的工作过程

puppet resource

- 如果把OS的所有配置，如用户账号、特定的文件、文件所属的目录、运行的服务、程序包以及cron任务等，看作是许多独立原子单元的集合的话，这些所谓的“单元”就是“资源”
  - 不过，这些资源在其大小、复杂程度以及生命周期的跨度上等多个维度上可能会各不相同
- 通常来说，类属于同一种资源的属性是相近的，如文件都有其属主和属组，而用户账号则由用户名、UID、GID等组成
  - 不过，即便是同一种资源，其在不同OS上的实现方式却又可能各不相同，例如，在windows上和Linux上启动和停止服务的方式相去甚远，因此puppet必须对资源进行某种程度的抽象

puppet资源抽象象

- puppet从以下三个维度来对资源完成抽象
  - 相似的资源被抽象成同一种资源"类型"，如程序包资源、用户资源及服务资源等；
  - 将资源属性或状态的描述与其实现方式剥离开来，如仅说明安装一个程序包而不用关心其具体是通过yum、pkgadd、ports或是其它方式实现；
  - 仅描述资源的目标状态，也即期望其实现的结果，而不是其具体过程，如"确定nginx运行起来"而不是具体描述为"运行nginx命令将其启动起来"；
- 这三个也被称作puppet的资源抽象层(RAL)
  - RAL由type(类型)和provider(提供者，即不同OS上的特定实现)组成

resource declaration

- Resources are the fundamental unit for modeling system configurations
- The block of Puppet code that describes a resource is called a resource declaration
- Syntax
  - Every resource has a **type**, a **title**, and a set of **attributes**

```
type {'title':
   attribute => value,
}
```

- puppet有许多内置的资源类型，而通过安装插件还可以继续新增额外的类型
  - http://docs.puppetlabs.com/references/latest/type.html
  - puppet describe"命令来获取puppet当前所支持的类型列表及每种类型的详细信息

一个示例

```
user { 'magedu':
    ensure     => present,
    uid        => '601',
    gid        => '601',
    shell      => '/bin/bash',
    home       => '/home/magedu',
    managehome => true,
  }
```

- 在定义时，资源类型必须使用小写字符；
- 资源名称仅是一个字符串，但要求在同一个类型中其必须惟一
  - 例如，可以同时有名为nginx的“service”资源和“package”资源，但在“package”类型的资源中只能有一个名为“nginx”
- “puppet resource”命令可用于交互式查找及修改puppet资源

Resource Special Attributes

- Name/Namevar
  - Most types have an attribute which identifies a resource *on the target system*
  - This is referred to as the "namevar," and is often simply called "name"
  - Namevar values must be unique per resource type, with only rare exceptions (such as exec)
- Ensure
  - 它可见于大多数资源中，用于控制资源的存在性
    - ensure => file：存在且为一个普通文件；
    - ensure => directory：存在且为一个目录；
    - ensure => present：存在，可通用于描述上述三种；
    - ensure => absent：不存在；
- Metaparameters

资源间的次序

- Relationship Metaparameters
  - Puppet uses four metaparameters to establish relationships
    - Each of them can be set as an attribute in any resource
    - The value of any relationship metaparameter should be a resource reference (or array of references) pointing to one or more target resources
  - puppet提供了before、require、notify和subscribe四个元参数来定义资源间的相关性
    - 这四个元参数都以另外的其它资源或资源数组作为其值，这也称作资源引用
    - 资源引用要通过“Type['title']”的方式进行，如User['magedu']
      - 注意：资源引用时，其类型名的首字母要大写

Relationship Metaparameters

- before
  - Causes a resource to be applied before the target resource
- require
  - Causes a resource to be applied after the target resource
- notify
  - Causes a resource to be applied before the target resource
  - The target resource will refresh if the notifying resource changes
- subscribe
  - Causes a resource to be applied after the target resource
  - The subscribing resource will refresh if the target resource changes

before and require

- before和require用于定义资源间的依赖关系，从而也定义了资源应用时的次序

```
group { 'magedu':
    ensure  => present,
    gid     => 601,
    before  => User['magedu'],
}

user { 'magedu':
    ensure     => present,
    uid        => '601',
    gid        => '601',
    shell      => '/bin/bash',
    home       => '/home/magedu',
}
```

notify and subsribe

- 对于诸如service、exec和mount等类型资源，它们还支持“refreshed”——在应用改变后执行一个后续的操作
  - 例如，对于service类型的资源，其配置文件改变后通常还需要一个重启操作；对于exec类型的资源，在任何相关外部条件改变后还需要执行重新执行指定的外部脚本
- notify和subscribe元参数除了具有相应于before和require元参数的依赖性定义功能之外，还具备额外的通知功能

```
file { '/etc/nginx/nginx.conf':
    ensure => file,
    mode   => 644,
    source => 'puppet:///modules/nginx/nginx.conf',
}
service { 'nginx':
    ensure    => running,
    enable    => true,
    subscribe => File['/etc/nginx/nginx/conf'],
}
```

资源间的应用次序链

- “->”用于定义次序链，而“~>”用于定义通知链
- 它们既可以用于资源引用间，也可以用于资源申报之间
- Note: Arrays of references cannot be chained
  - Package['ntp'] -> File['/etc/ntp.conf'] ~> Service['ntpd']
  -

```
# first:
package { 'openssh-server':
  ensure => present,
} -> # and then:
file { '/etc/ssh/sshd_config':
  ensure => file,
  mode   => 600,
  source => 'puppet:///modules/sshd/sshd_config',
} ~> # and then:
service { 'sshd':
  ensure => running,
  enable => true,
}
```

puppet variable

- puppet的变量名称须以"$"开头，赋值操作符为"="
- 任何正常数据类型(非正则)的值都可以赋予puppet中的变量，如字符串、数值、布尔值、数组、hash以及特殊的undef值(即变量未被赋值)
- puppet的每个变量都有两个名字：简短名称和长格式完全限定名称(FQN)，完全限定名称的格式为"$scope::variable"

puppet data type

- puppet语言支持多种数据类型以用于变量和属性的值，以及函数的参数
  - 字符型
    - 非结构化的文本字符串，可以使用引号，也可以不用；
    - 单引号中的变量不会替换，而双引号中的能够进行变量替换；
    - 字符型值也支持使用转义符；
  - 数值型
    - 可为整数或浮点数，不过，puppet只有在数值上下文才把数值当数值型对待，其它情况下一律以字符型处理；
  - 数组
    - 数组值为中括号"[]"中的以逗号分隔的项目列表，最后一个项目后面可以有逗号；
    - 数组中的元素可以为任意可用数据类型，包括hash或其它数组；
    - 数组索引为从0开始的整数，也可以使用负数索引；

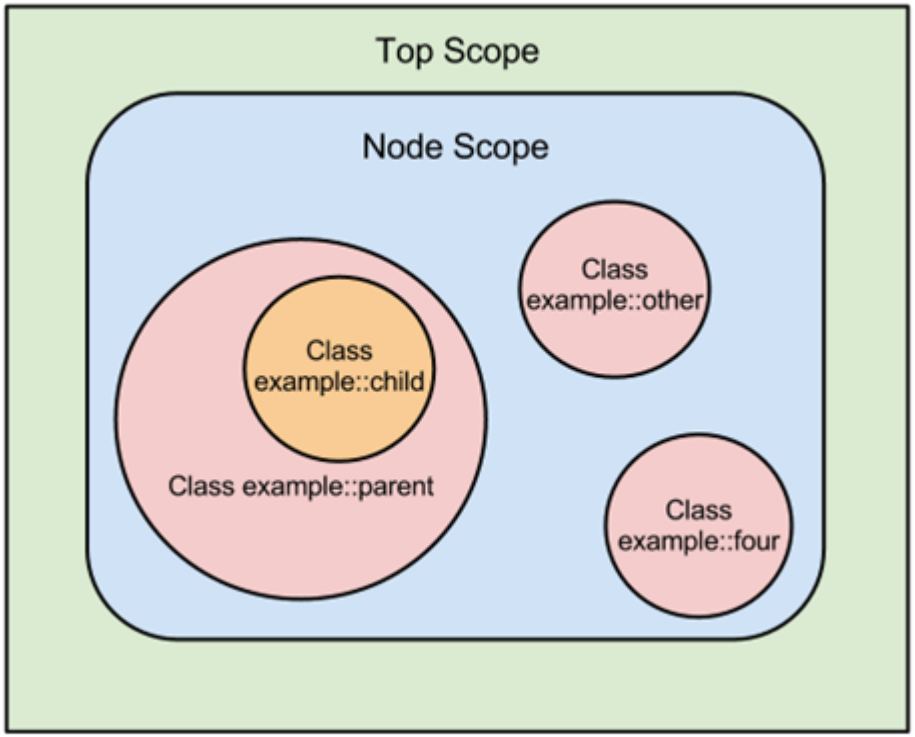
- 布尔型
  - true和false，不能加引号；
  - if语句的测试条件和比较表达式都会返回布尔型值；
  - 另外，其它数据类型也可自动转换为布尔型，如空字符串为false等；
- undef
  - 从未被声明的变量的值类型即为undef；
  - 也可手动为某变量赋予undef值，即直接使用不加引号的undef字符串；
- hash
  - 即为外键值数据类型，键和值之间使用"=>"分隔，键值对儿定义在"{}"中，彼此间以逗号分隔；
  - 其键为字符型数据，而值可以为puppet支持的任意数据类型；
  - 访问hash类型的数据元素要使用"键"当作索引进行

- 正则表达式
  - 属于puppet的非标准数据类型，不能赋值给变量，仅能用于有限的几个接受正则表达式的地方，即接受使用"=~"及"!~"匹配操作符的位置，通常包括case语句中的selector，以及节点名称匹配的位置；
  - 它们不能传递给函数或用于资源属性的定义；
  - puppet中的正则表达式支持使用(?<ENABLED OPTION>:<SUBPATTERN>)和(?-<DISABLED OPTION>:<SUBPATTERN>)两个特殊的符号
    - 例如下面的示例表示做正则表达式匹配时启用选项"i"(忽略字符大小写)，但不支持使用"m"(把.当作换行符)和"x"(忽略模式中的空白字符和注释)

```
$packages = $operatingsystem ? {
   /(?i-mx:ubuntu|debian)/         => 'apache2',
   /(?i-mx:centos|fedora|redhat)/ => 'httpd',
}
```

Scope

- Scope是一个特定的代码区域，用于同程序中的其它代码隔离开来
- 在puppet中，scope可用于限定变量及资源默认属性的作用范围，但不能用于限定资源名称及资源引用的生效范围



- 任何给定的scope都可以访问它自己的内容，以及接收来自于其父scope、节点scope及top scope的内容
  - 如图所示，top scope仅能访问自己变量和属性默认值；
  - 节点scope能访问自己的及top scope的变量和属性默认值；
  - example::parent，example::other和example::four能访问自己的以及节点scope和top scope的变量和默认值
- 如果要访问非当前scope中的变量，则需要通过完全限制名称进行，如$vhostdir = $apache::params::vhostdir
- 需要注意的是，top scope的名称为空，因此，如若要引用其变量，则需要使用类似"$::osfamily"的方式进行

Built-In Variables

- Puppet provides several built-in top-scope variables
  - Facts
    - puppet使用了一个称作Facter的工具来收集系统信息，规范化之后将其放进一系列变量中，并传递给puppet
    - Fact的各变量是top scope的变量，这意味着，可以在各manifest中直接通过${fact name}访问所需的fact变量
    - facter -p
  - Built-In Variables
    - Agent-Set Variables
      - $environment、$clientcert、$clientversion
    - Master-Set Variables
      - $servername、$serverip、$serverversion
    - Parser-Set Variables
      - $module_name

Operators

- Comparison Operators
  - == (equality)
  - != (non-equality)
  - < (less than)
  - > (greater than)
  - <= (less than or equal to)
  - >= (greater than or equal to)
  - =~ (regex match)
  - !~ (regex non-match)
  - in
- Boolean Operators
  - and
  - or
  - ! (not)
- Arithmetic Operators
  - + (addition)
  - - (subtraction)
  - / (division)
  - * (multiplication)
  - << (left shift)
  - >> (right shift)

Conditional Statements

- Puppet 2.7 supports "if" statements, case statements, and selectors
- puppet 3 新引入了unless语句
- 条件判断可以让puppet代码基于不同场景采取不同的行为，尤其是在通过fact或外部资源获取数据时最为有用

if statement

单分支：

```
if CONDITION {
    statement
    ...
}
```

双分支：

```
if CONDITION {
    statement
    ...
}
else {
    statement
    ...
}
```

多分支：

```
if CONDITION {
    statement
    ...
}
elsif CONDITION {
    statement
    ...
}
else {
    statement
    ...
}
```

- Conditions
  - The condition(s) of an "if" statement may be any fragment of Puppet code that resolves to a boolean value
    - Variables
    - Expressions, including arbitrarily nested and and or expressions
    - Functions that return values

Regex Capture Variables

- If you use the regular expression match operator in a condition, any captures from parentheses in the pattern will be available inside the associated code block as numbered variables ($1, $2, etc.), and the entire match will be available as $0

```
if $operatingsystem =~ /^(?i-mx:(centos|redhat))/ {
  notice("Welcome to $1 linux server")
}
```

Case Statements

- 类似if语句，case语句会从多个代码块中选择一个分支执行，这跟其它编程语言中的case语句功能一致
- case语句会接受一个控制表达式和一组case代码块，并执行第一个匹配到控制表达式的块
- Syntax

```
case CONTROL_EXPRESS {
    case1,...: { statement... }
    case2,...: { statement... }
    ... ...
    default: { statement... }
}
```

Case Statements

- Control Expressions
  - The control expression of a case statement may be any fragment of Puppet code that resolves to a normal value
    - Variables
    - Expressions
    - Functions that return values
- Cases
  - A literal value (remember to quote strings)
  - A variable
  - A function call that returns a value
  - A regular expression
  - The special bare word value default

```
case $operatingsystem {
  'Solaris':          { notice("Welcome to Solaris") }
  'RedHat', 'CentOS': { notice("Welcome to RedHat OSFamily") }
  /^(Debian|Ubuntu)$/:{ notice("Welcome to $1 linux") }
  default:            { notice("Welcome, alien *_*") }
}
```

Selectors

- Selector statements are similar to case statements, but return a value instead of executing a code block
- Selector只能用于期望出现直接值(plain value)的地方，这包括变量赋值、资源属性、函数参数、资源标题、其它selector的值及表达式
- selector不能用于一个已经嵌套于于selector的case中，也不能用于一个已经嵌套于case的case语句中
- Syntax

```
CONTROL_VARIABLE ? {
    case1 => value1
    case2 => value2
    ...
    default => valueN
}
```

```
$webserver = $operatingsystem ? {
    /(?i-mx:ubuntu|debian)/        => 'apache2',
    /(?i-mx:centos|fedora|redhat)/ => 'httpd',
}
```

Selectors 使用要点

- 整个selector语句会被当作一个单独的值，puppet会将控制变量按列出的次序与每个case进行比较，并在遇到一个匹配的case后，将其值作为整个语句的值进行返回，并忽略后面的其它case
- 控制变量与各case比较的方式与case语句相同，但如果没有任何一个case与控制变量匹配时，puppet在编译时将会返回一个错误，因此，实践中，其必须提供default case
- selector的控制变量只能是变量或有返回值的函数，切记不能使用表达式
- 其各case可以是直接值(需要加引号)、变量、能调用返回值的函数、正则表达式模式或default
- 但与case语句所不同的是，selector的各case不能使用列表
- selector的各case的值可以是一个除了hash以外的直接值、变量、能调用返回值的函数或其它的selector

Functions

- Functions are pre-defined chunks of Ruby code which run during compilation
- Most functions either return values or modify the catalog
- Puppet includes several built-in functions, and more are available in modules on the Puppet Forge, particularly the puppetlabs-stdlib module
- You can also write custom functions and put them in your own modules
- 不多做介绍

马哥教育

Puppet Classes and Modules

主讲：马永亮(马哥)
QQ:113228115
客服QQ：2813150558，1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

Classes

- Classes are named blocks of Puppet code which are not applied unless they are invoked by name
- They can be stored in modules for later use and then declared (added to a node's catalog) with the include function or a resource-like syntax
- Class是用于通用目标的一组资源，因此，它是命名的代码块，在某位置创建之后可在puppet全局使用
- 类似于其它编程语言中的类的功能，puppet的Class可以被继承，也可以包含子类
- Syntax


```
class my_class {
        ... puppet code ...
}
```

- 一个例子

```
class apache {
    package { httpd:
        ensure => installed
    }
    file { 'httpd.conf':
    path    => '/etc/httpd/conf/httpd.conf',
    ensure  => file,
    require => Package['httpd'],
    }
    service { httpd:
        ensure => running,
        require => Package["httpd"],
        subscribe => File['httpd.conf'],
    }
}
```

- 类的名称只能以小写字母开头，可以包含小字字母、数字和下划线
- 每个类都会引入一个新的变量scope，这意味着在任何时候访问类中的变量时，都得使用其完全限定名称
  - 不过，在本地scope可以重新为top scope中的变量赋予一个新值

Declaring a Class

- 在manifest文件中定义的类不会直接被执行，它们需要事先声明后才能被执行
- Declaring a class adds all of the code it contains to the catalog
  - Declaring a Class With include
  - Declaring a Class with require
  - Declaring a Class Like a Resource
  - Declaring a Class With an ENC

- Declaring a Class With include
  - Classes can be declared with the include function
    - include base::linux
  - Can safely use include multiple times on the same class and it will only be declared once
  - The include function can accept a single class name or a comma-separated list of class names
    - include base::linux, apache

Declaring a Class with require

- The require function acts like include, but also causes the class to become a dependency of the surrounding container

```
define apache::vhost ($port, $docroot, $servername, $vhost_name) {
  require apache
  ...
}
```

Declaring a Class Like a Resource

- Classes can also be declared like resources, using the special "class" resource type

```
# Declaring a class with the resource-like syntax
class {'apache':
  version => '2.2.21',
}
# With no parameters:
class {'base::linux':}
```

- A class can only be declared this way once
- The resource-like syntax should not be mixed with include for a given class

带参数的类

- 同一个类在不同的OS上可能会略有不同，因此需要通过获取相应系统的fact来实现有区别对待
- 然而，万一相应的OS没有输出类所期望的fact或者是类依赖于非fact因素时，此机制将无法满足需求
- 此时就需要使用带参数的类来完成此类功能，同时，在声明类时为其参数传递相应的值即可完成传参功能
- 在定义在带参数的类时，需要将参数声明在类名后的小括号"()"，参数可以有默认值；如果使用多个参数，彼此间要使用逗号分隔
- 在类的内部使用参数的方式同使用本地变量


```
class my_class (para1='val1', para2='val2'){
        ... puppet code ...
}
```

- 在声明类时，可向其传递参数；其方式如同定义资源的属性，因此，其也称为"资源属性风格的类声明"，其语法格式如下

```
class {'class_name':
  para1 => value1,
  para2 => value2,
  ...
}
```

```
class mysql ($user = 'mysql', $port = 3306) {
  ...
}

class {'mysql':
  user => mysqlserver,
}
```

- 不能在使用include声明类时向其传递参数
- 对于带参数的类来说，如果使用其参数的默认值，仍然可以使用include声明类，否则就必须使用"资源属性风格的类声明"
- 另外，如果在使用不同的参数值将某个类声明了多次，最后生效的声明将很难判定

Class Inheritance

- Classes can be derived from other classes using the inherits keyword
- This allows you to make special-case classes that extend the functionality of a more general "base" class
  - Note: Puppet 2.7 does not support using parameterized classes for inheritable base classes. The base class must have no parameters.
- Inheritance causes three things to happen
  - When a derived class is declared, its base class is automatically declared first
  - The base class becomes the parent scope of the derived class, so that the new class receives a copy of all of the base class's variables and resource defaults
  - Code in the derived class is given special permission to override any resource attributes that were set in the base class

- Overriding Resource Attributes
  - The attributes of any resource in the base class can be overridden with a **reference** to the resource you wish to override, followed by a set of curly braces containing attribute => value pairs
- Appending to Resource Attributes
  - hen overriding attributes in a derived class, you can add to the existing values instead of replacing them by using the +> ("plusignment") keyword instead of the standard => hash rocket


```
class apache {
  service {'apache':
    require => Package['httpd'],
  }
}

class apache::ssl inherits apache {
  Service['apache'] {
    require +> [ File['httpd.pem'], File['httpd.conf'] ],
  }
}
```

Modules概述

- 到目前为止，资源申报、定义类、声明类等所有功能都只能在一个manifest文件中实现，但这却非最有效的基于puppet管理IT基础架构的方式
- 实践中，一般需要把manifest文件分解成易于理解的结构，例如将类文件、配置文件甚至包括后面将提到的模块文件等分类存放，并且通过某种机制在必要时将它们整合起来
- 这种机制即"模块"，它有助于以结构化、层次化的方式使用puppet，而puppet则基于"模块自动装载器"完成模块装载
- 从另一个角度来说，模块实际上就是一个按约定的、预定义的结构存放了多个文件或子目录的目录，目录里的这些文件或子目录必须遵循其命名规范
- puppet会按此种规范在特定位置查找所需的模块文件，不过，这些特定目录也可以通过puppet的配置参数modulepath定义

Module Layout

- On disk, a module is simply a directory tree with a specific, predictable structure

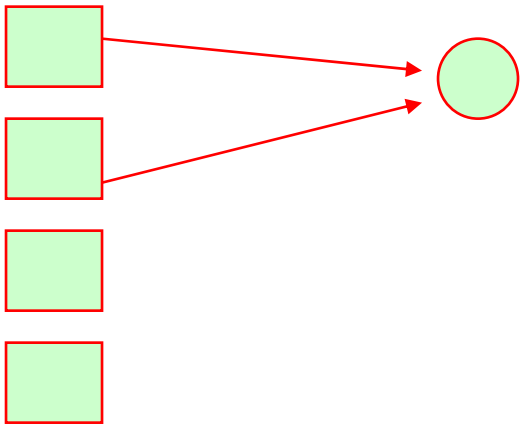
```
MODULE NAME
    manifests
        init.pp
    files
    templates
    lib
    tests
    spec
```

Module Layout

- MODULE NAME：模块名称，也即模块目录名称；模块名称只能以小写字母开头，可以包含小写字母、数字和下划线，但不能使用“main”或“settings”作为模块名；
  - manifests目录：包含当前模块的所有manifest文件；每个manifest文件必包含一个类或一个定义的类，此文件访问路径格式为“ModuleName::[SubDirectoryName::]ManifestFileName”，注意manifiest文件名不需要其后缀.pp；
    - init.pp：只能包含一个单独的类定义，且类的名称必须与模块名称相同；
  - files目录：包含了一组静态文件，这些文件可被节点下载使用；每个文件的访问路径遵循puppet:///modules/MODULE_NAME/filename路径格式；

Module Layout

- lib目录：插件目录，常用于自定义fact及自定义资源类型等；
- templates目录：存储了manifest用到的模板文件，其访问路径遵循template('ModuleName/TemplateName')格式；
- tests目录：当前模块的使用帮助或使用范例文件，类似如何声明当前模块中的类及定义的类型等；
- spec目录：类似于tests目录的功能，只不过，其是为lib目录中定义的各插件提供使用范例的；



一个示例

- mkdir -pv /etc/puppet/modules/nginx/{files,manifests}

```
[root@node2.magedu.com ~]#cat /etc/puppet/modules/nginx/manifests/init.pp
class nginx {
}

[root@node2.magedu.com ~]#cat /etc/puppet/modules/nginx/manifests/nginxsrv.pp
class nginx::nginxsrv inherits nginx {
        package {'nginx':
                ensure => installed,
        }

        service {'nginx':
                ensure  => running,
                require => Package['nginx'],
        }
}

[root@node2.magedu.com ~]#cat /etc/puppet/manifests/site.pp
node 'node2.magedu.com' {
        include nginx::nginxsrv
}
```

- puppet apply -v -d /etc/puppet/manifests/site.pp

马哥教育

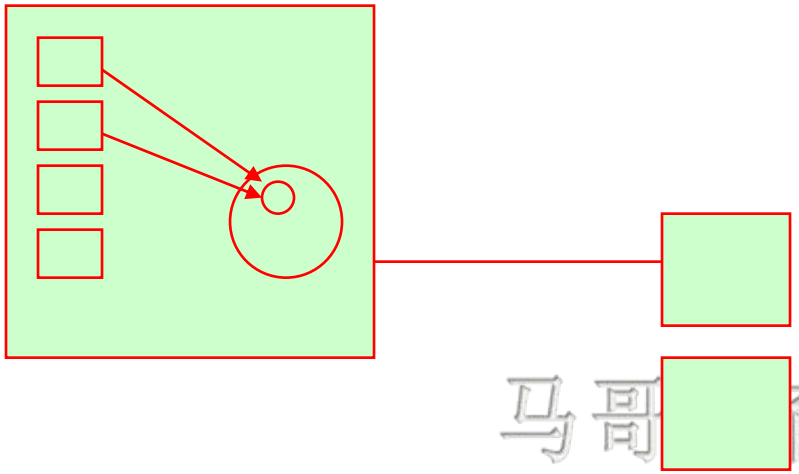
puppet agent/master

主讲：马永亮(马哥)
QQ:113228115
客服QQ：2813150558，1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

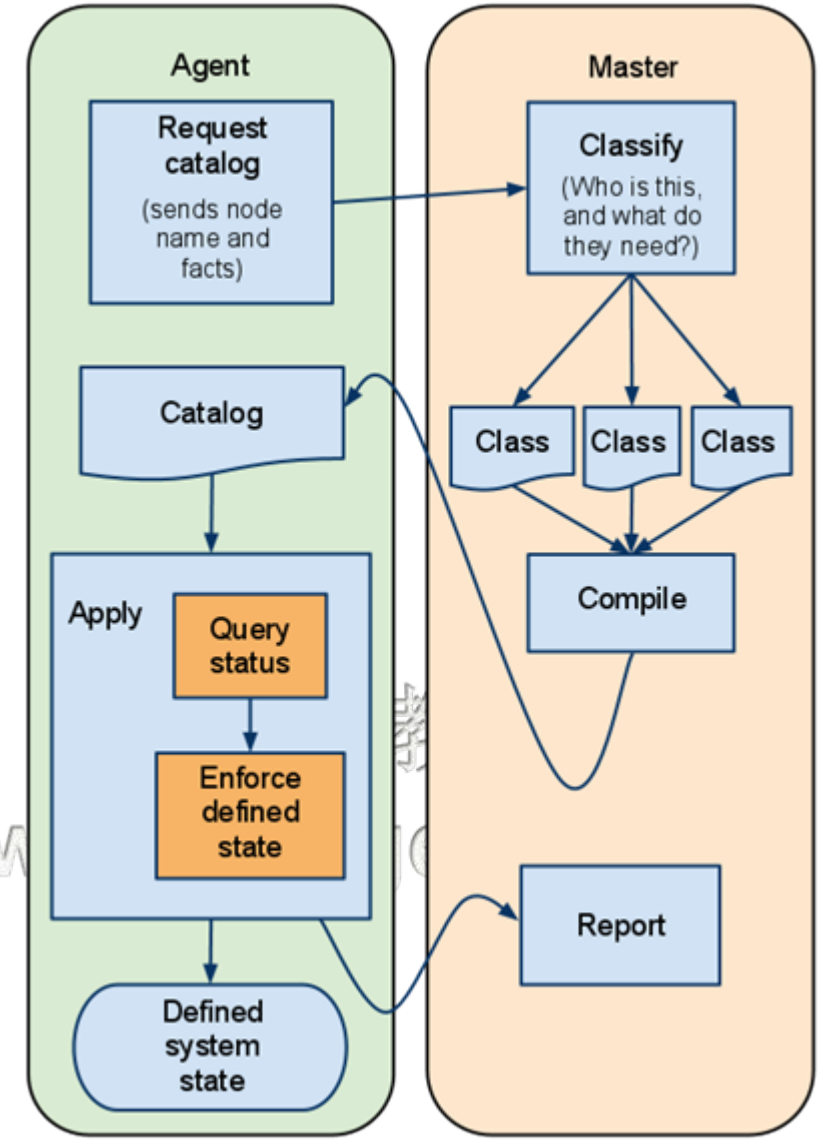
puppet master

- 站点清单
  - /etc/puppet/manifests/site.pp
    - node 'FQDN' {
      - include CLASS
      - include
    - }
  - /etc/puppet/modules/
    - manifests
      - init.pp
      - *.pp
    - files
      - puppet:///modules/mod_name/filename
    - lib
    - tests
    - spec
    - templates

- puppet.conf
- [main]
- [agent]
- runinterval = 1800



puppet agent/master



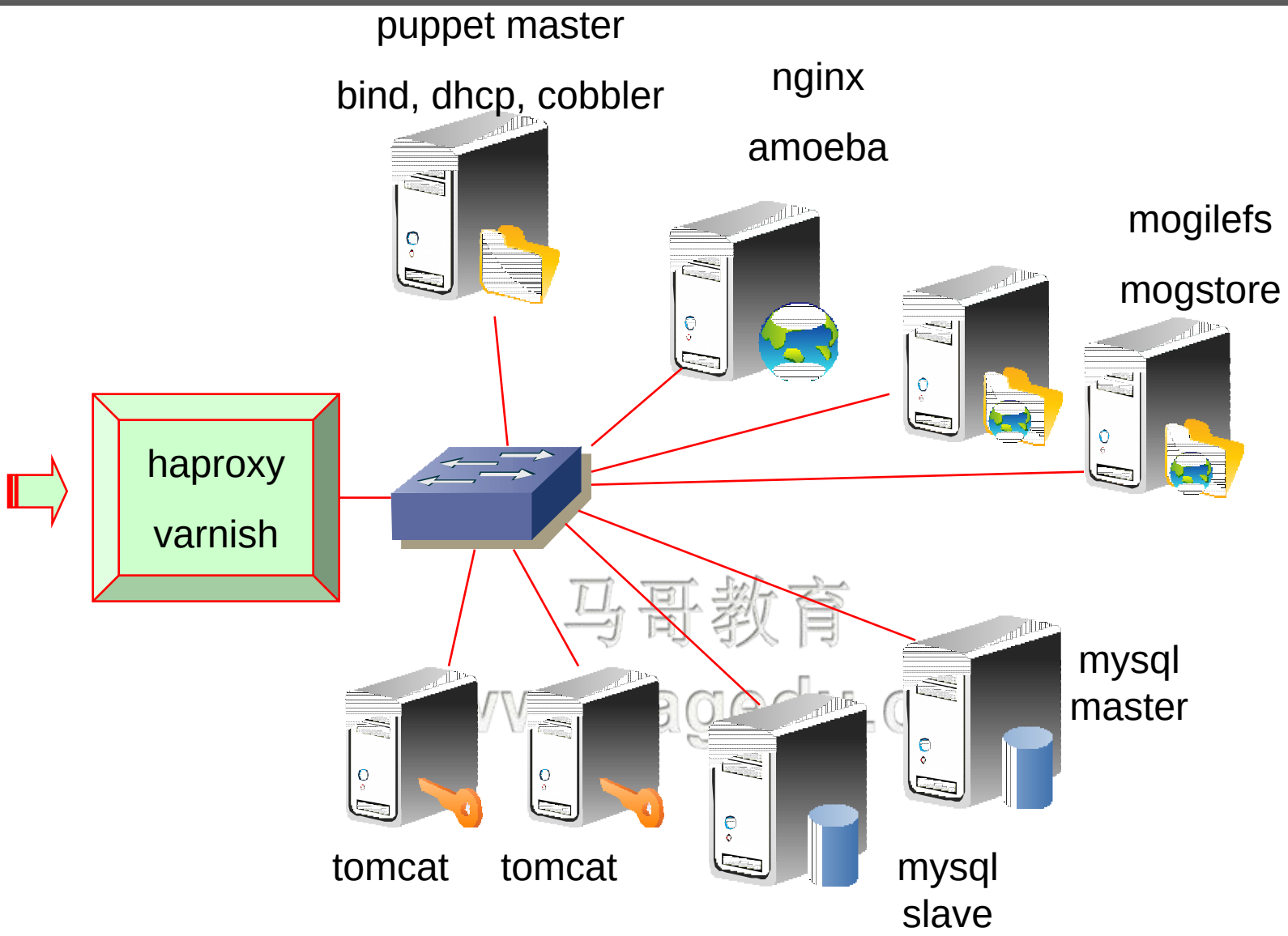
Site.pp

- 实践中，在master上通常会存放有许多的类文件，但当某特定的agent来请求其catalog时，master如何完成指派？
  - 本地工作模式的puppet只需要读取本地定义的manifest文件即可完成某种特定的应用，而在agent/master模型中也不例外，只不过，所有的agent都将请求同一个manifest文件，即位于manifests目录中site.pp，
  - manifests目录的位置可以通过puppet的配置参数$manifestdir指定，默认通常为/etc/puppet/manifests

site.pp

- 为了让不同的agent仅能按需访问指派给其的资源，需要在site.pp文件中定义节点来实现
- 定义的格式非常简单，而且其也不像类一样需要声明，因为puppet会自动声明每一个节点
- 此外，关联至同一个catalog上的node也只能有一个，这表示一个catalog专用于一个特定的节点
- 节点名称同其证书中的名称，一般应该为FQDN格式，当然，此名称也可以修改
- 节点声明的格式如下所示

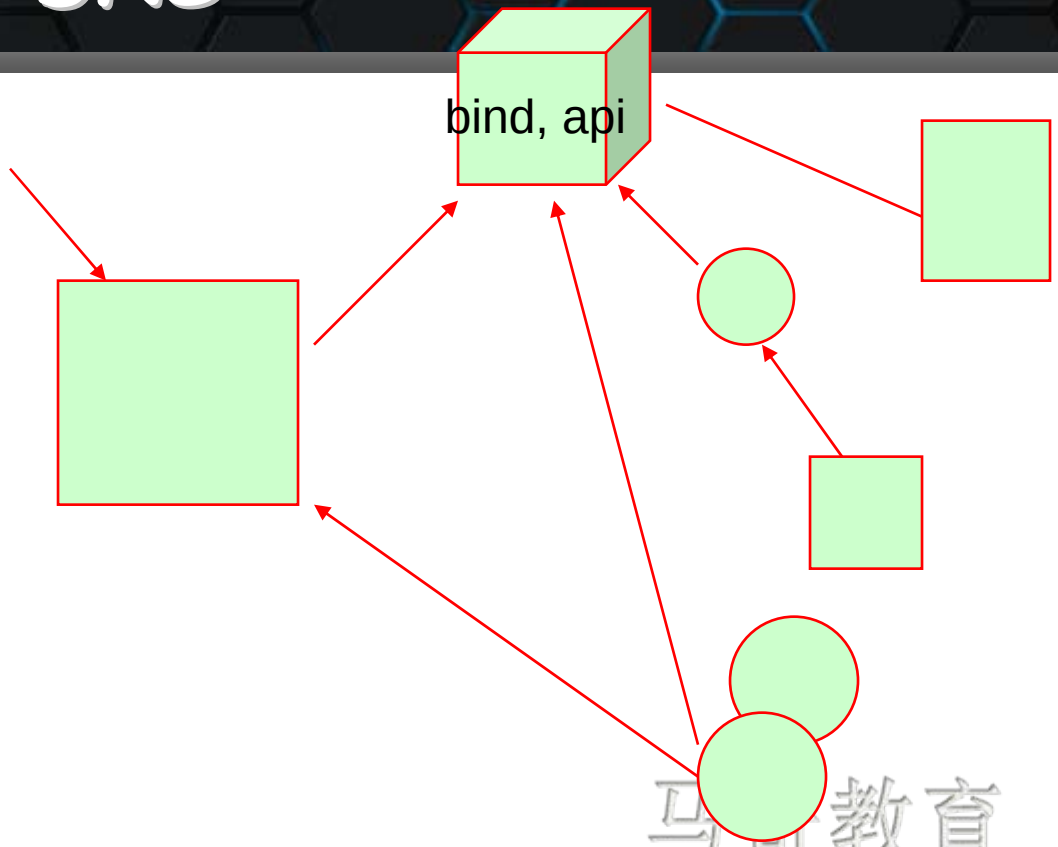
```
node 'NODE FQDN' {
        ...
    }
```



马哥教育

puppet扩展应用

主讲：马永亮(马哥)
QQ:113228115
客服QQ：2813150558，1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com



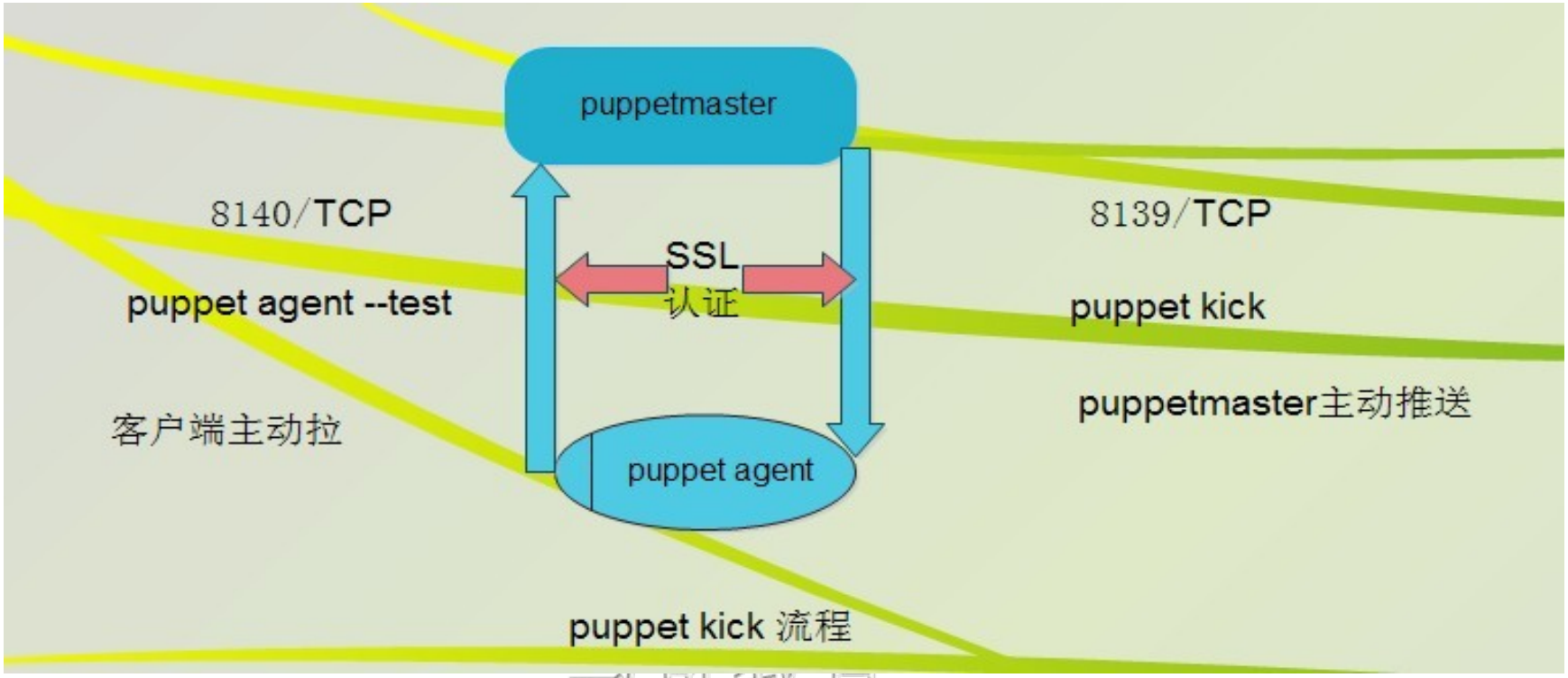
puppet使用要点

- 主机名使用FQDN格式命名，需要DNS或者hosts解析
- C/S两端时间要同步，SSL认证依赖于时间
- ssl认证过后，请不要修改主机名
  - 如果不得不改，master要吊销原有证书，甚至还需要清理
    - puppet cert {revoke|clean}
  - /var/lib/puppet/ssl/*
- Usecachefaiure
- tag 和tagged
- 节点可以继承，但不能多重继承
- 类可以继承与覆盖，子类可以向父类添加额外属性值
- puppet 支持多环境部署（分：开发、测试、线上）
- 选择正确的版本

Puppet应用案例

- Puppet 管理用户(关于:管理用户的几点说明)
  - puppet 支持ldap以及nis集中认证
  - puppet 支持用户密码管理，但需要以''(单引号)括起来
  - puppet 建用户的默认是不建家目录的，需要使用managerhome=>true

puppet+kick



- puppet kick：主动强制客户端运行puppet agent.
- 注意:puppet kick 并不关心客户端puppet agent 有没有执行错误，它成功连接到agent即返回退出0.

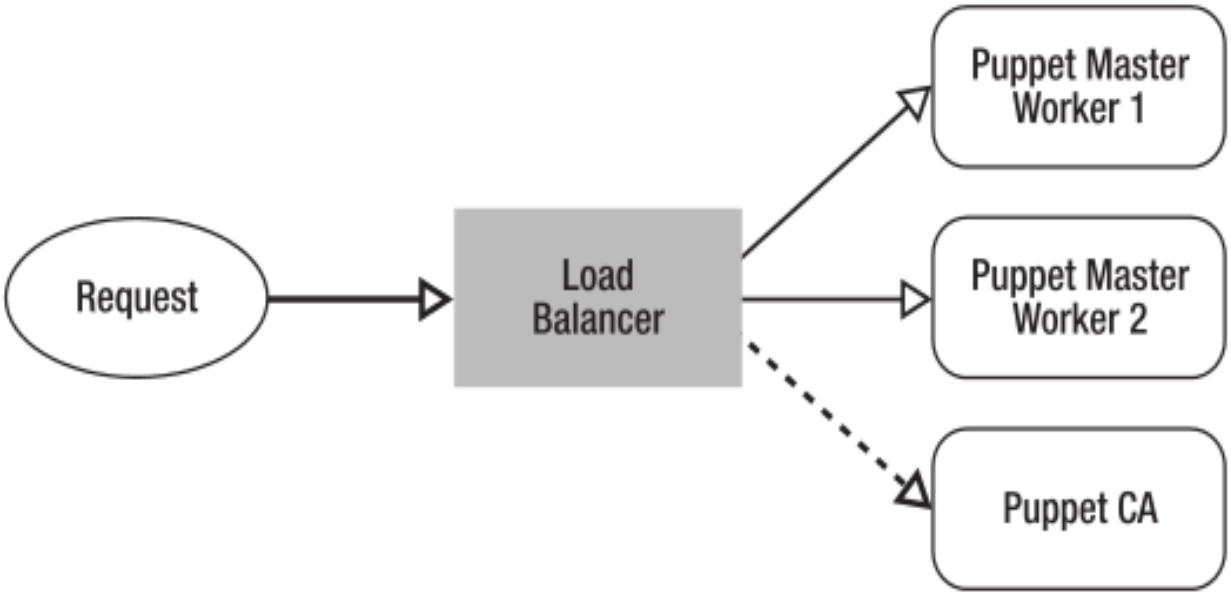
MCollective简介

- MCollective就是一个调度器,可以解决多个puppet agent同时向master提出请求造成性能,速度下降的问题;
  - 它可以根据不同的属性对节点进行分类,对不同分类执行不同的任务;
  - 它是一个控制终端,可以使用它控制客户端和服务器,因此不需要puppet agent定时运行了
- MCollective亦是C/S架构,而且client和server使用Midware(中间件)进行通信， 需要java以及activemq支持

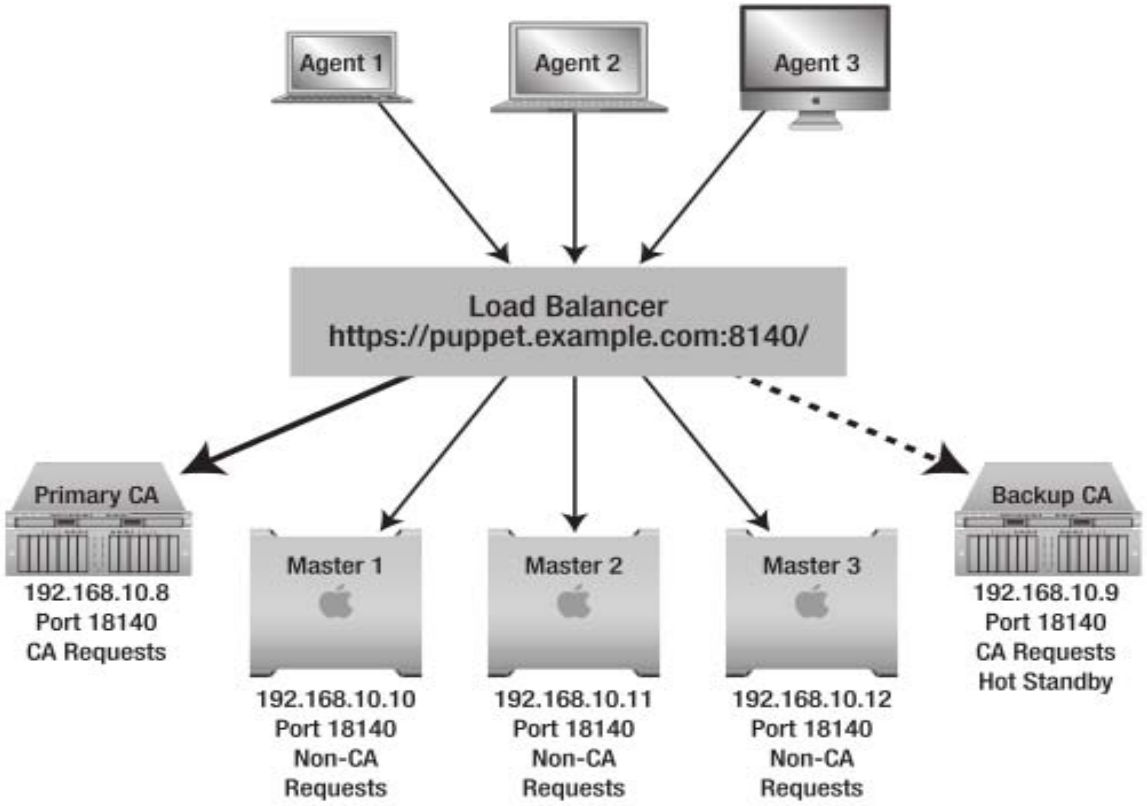
Puppet架构与集群

- Puppet通常部署为C/S架构，当agent过多时会面临性能问题
- Puppet常见的集群方案
  - Puppet + nginx
  - Puppet + passenger + apache
- Puppet集群的构建机制
  - puppetmaster集群
    - Active/Active模式高可用集群，分摊puppetmaster上来自于agent的请求压力
    - 反向代理模式，将针对于8140的端口请求分散到多台puppetmaster
  - puppet ssl证书集群

- puppetmaster 集群架构图



- puppetca集群架构图：



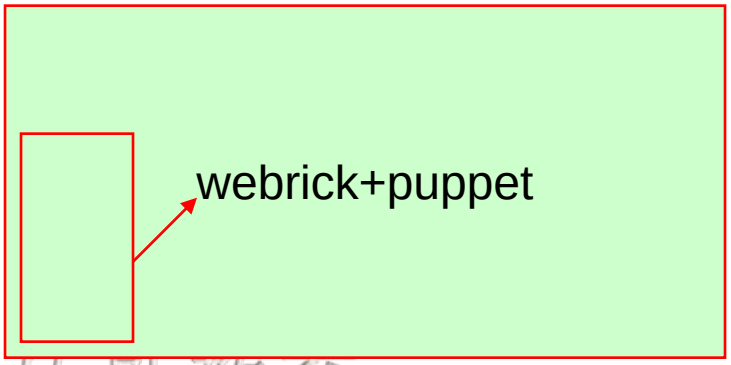
puppetmaster并发性能测试

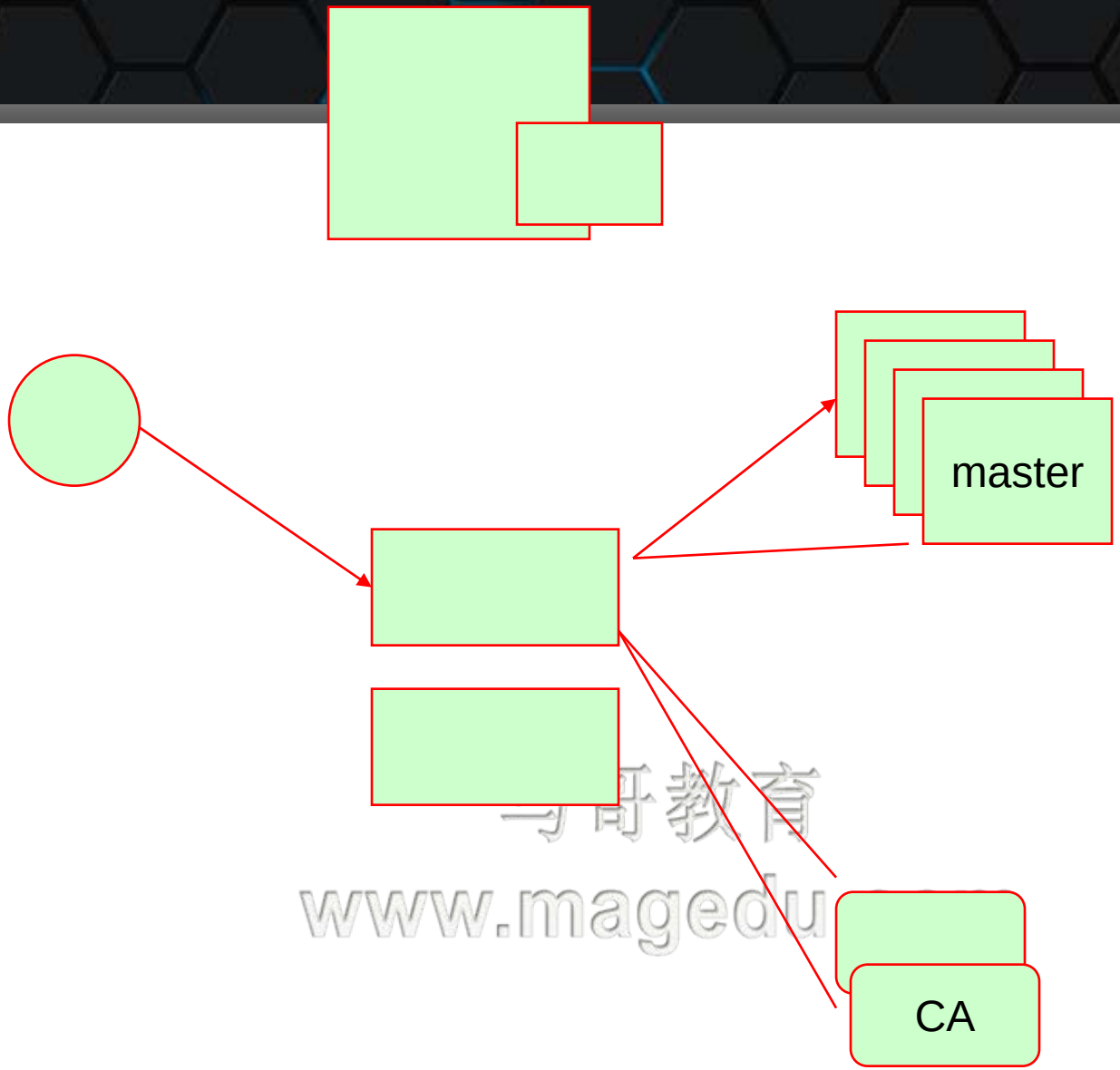
- 测试脚本
  - puppet-load.rb：/usr/share/puppet/ext目录中
- 测试puppetmaster性能：
- /usr/share/puppet/ext/puppet-load.rb - --debug --node node2.magedu.com --server ppmaster.magedu.com --factsdir=/var/lib/puppet/yaml/facts --concurrency 1 --repeat 1 --cert /var/lib/puppet/ssl/certs/puppet1.pem --key /var/lib/puppet/ssl/private_keys/puppet1.pem
- 更多信息请参考
  - http://www.masterzen.fr/2010/10/18/benchmarking-puppetmaster-stacks/

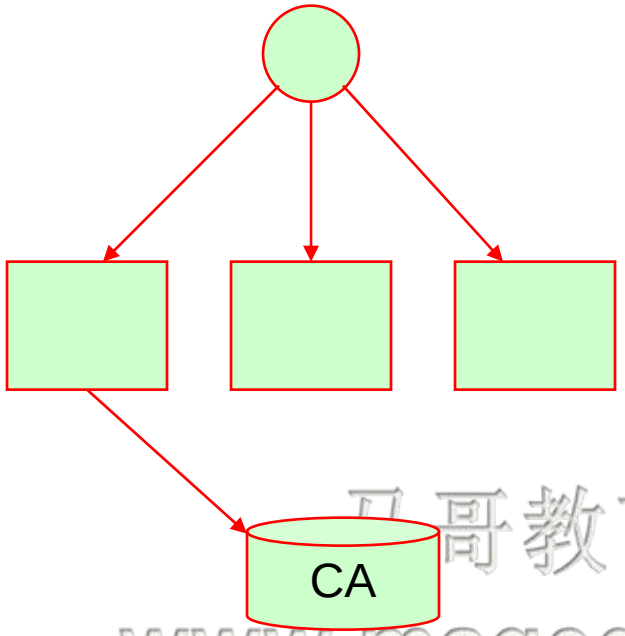
马哥教育

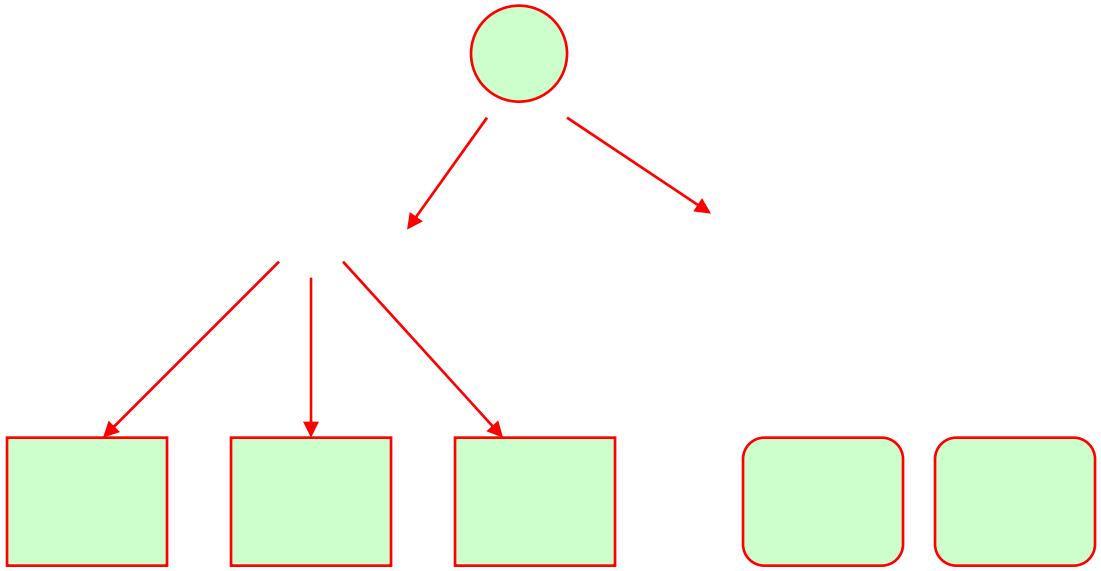
补充资料

主讲：马永亮(马哥)
QQ：113228115，1661815153
http://www.magedu.com

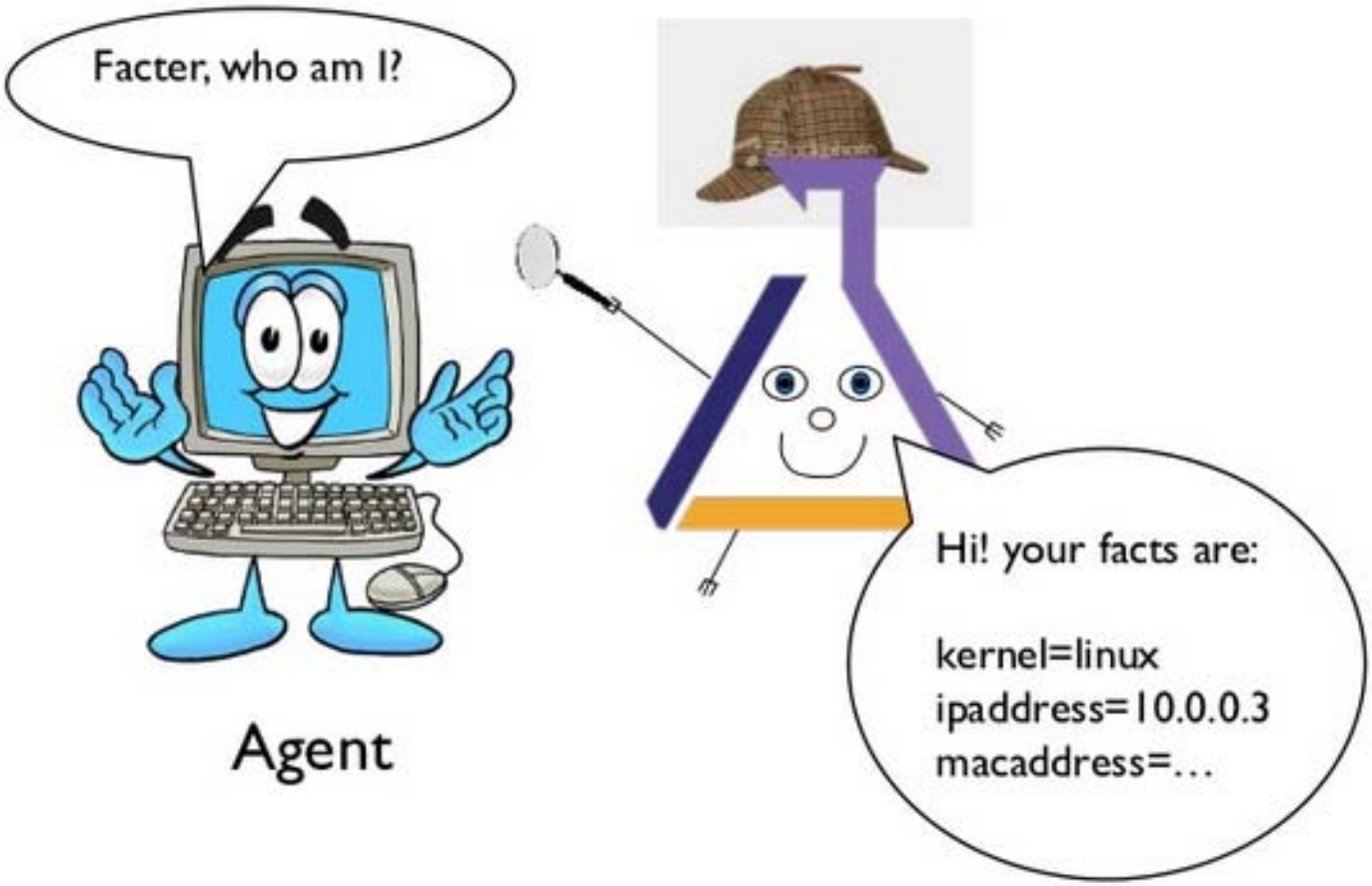




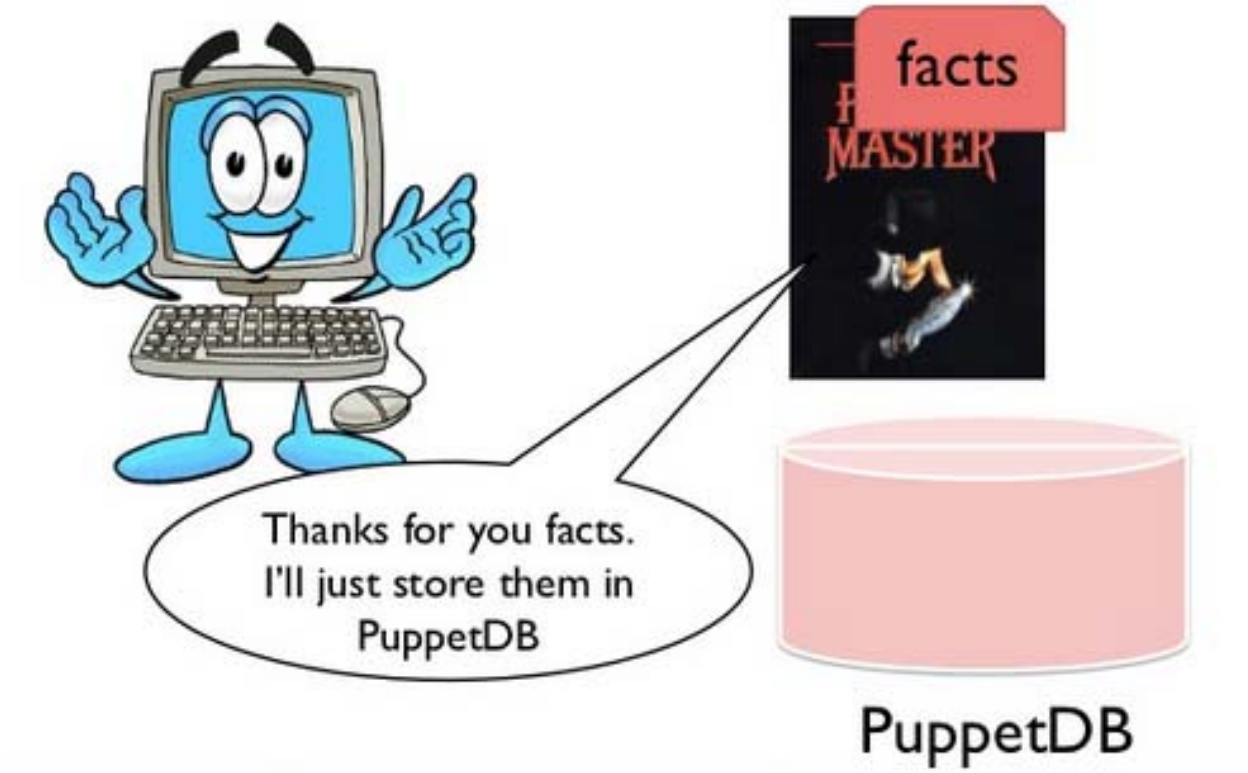


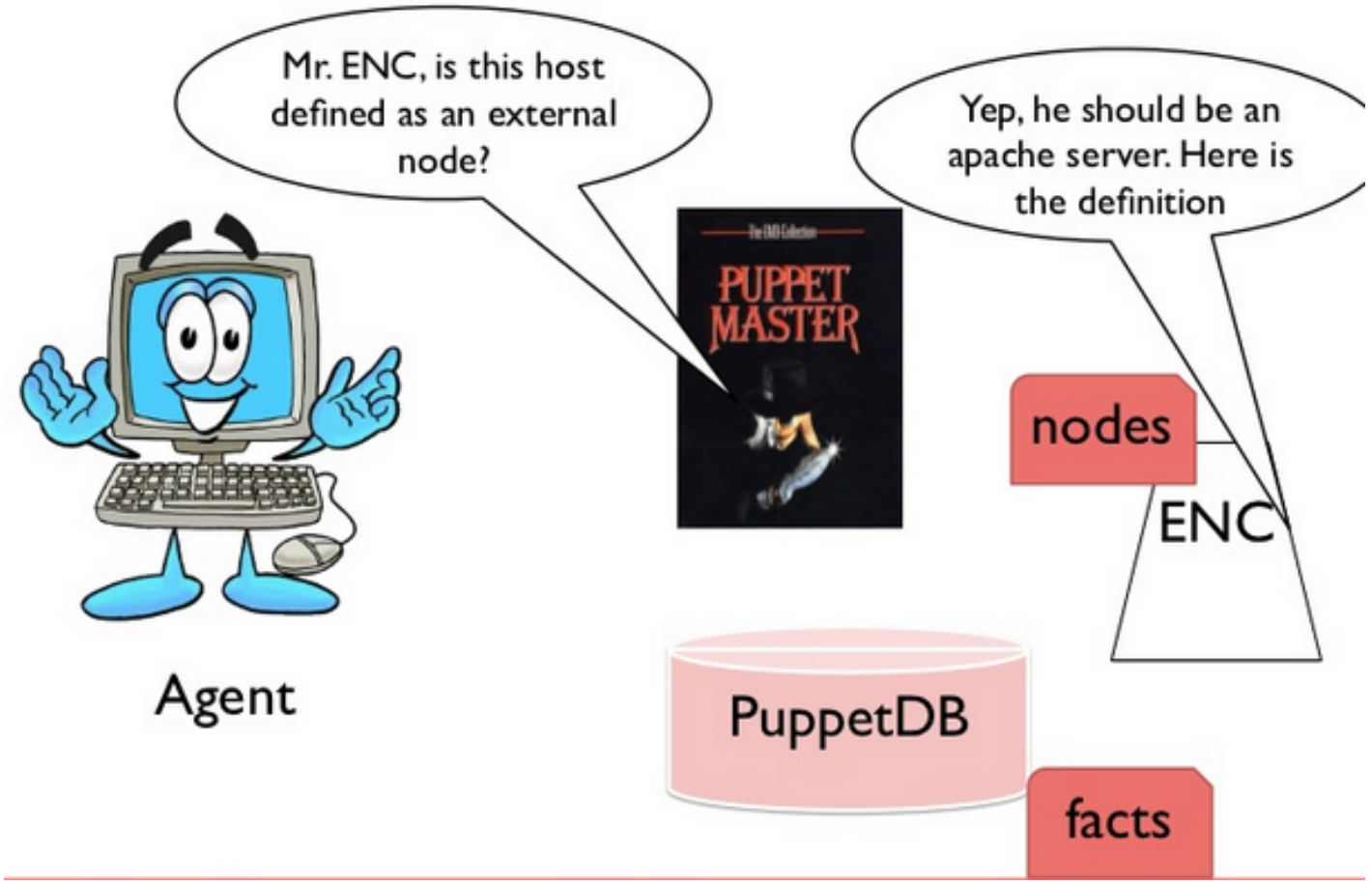


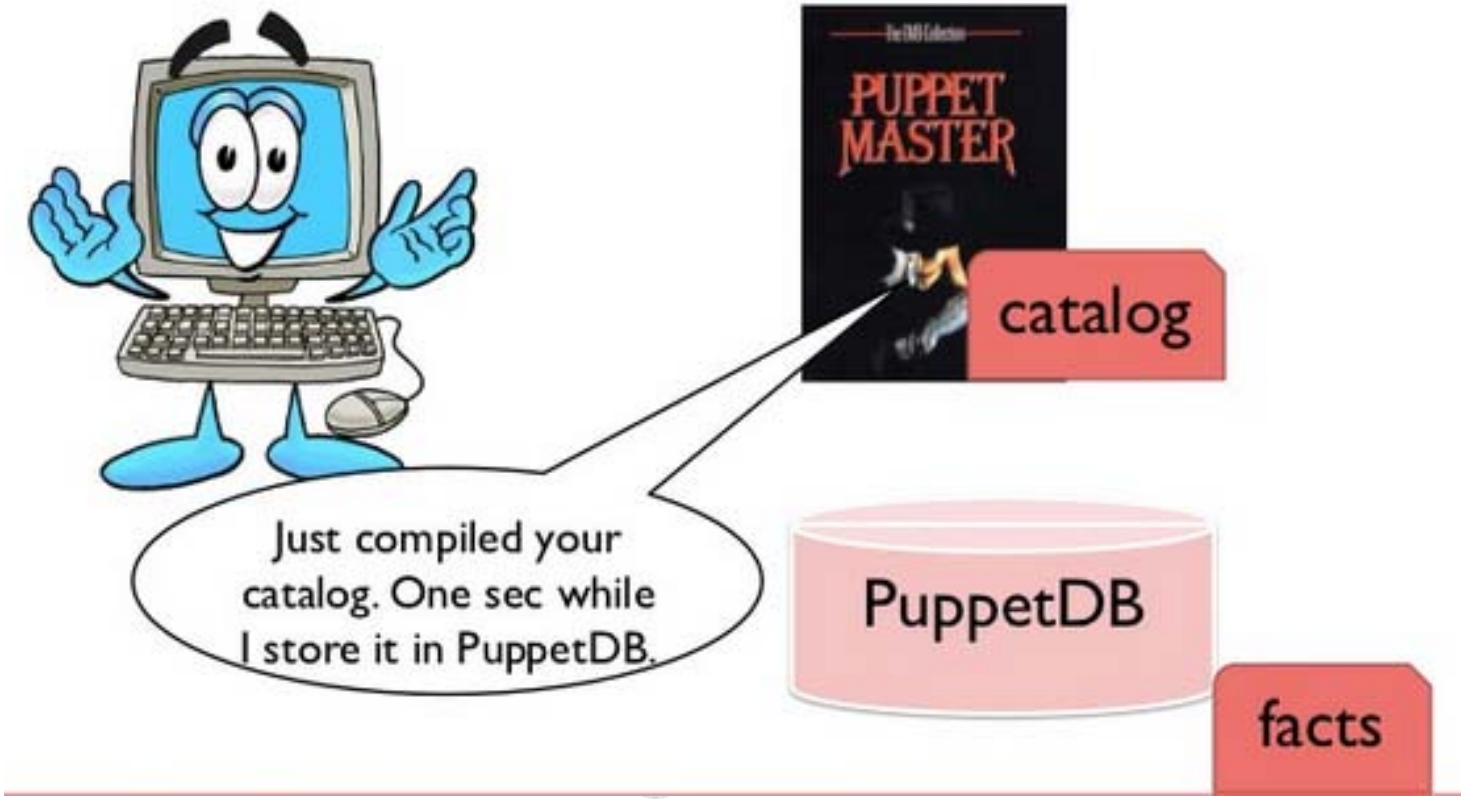
Facter


Facter, who am I?
Agent
Hi! your facts are:
kernel=linux
ipaddress=10.0.0.3
macaddress=…

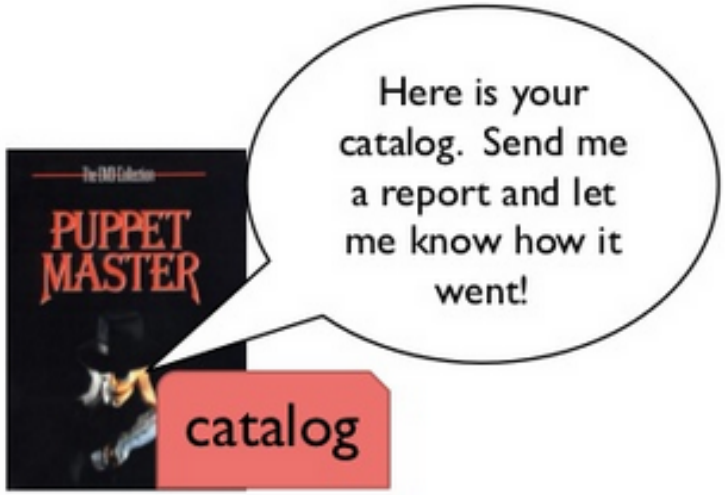
http://www.dgcomputers.org/testimonials.php

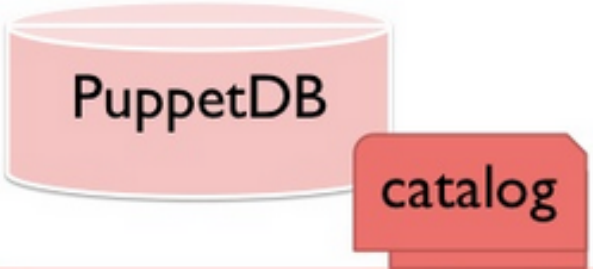

facts
MASTER
Thanks for you facts.
I'll just store them in
PuppetDB
PuppetDB

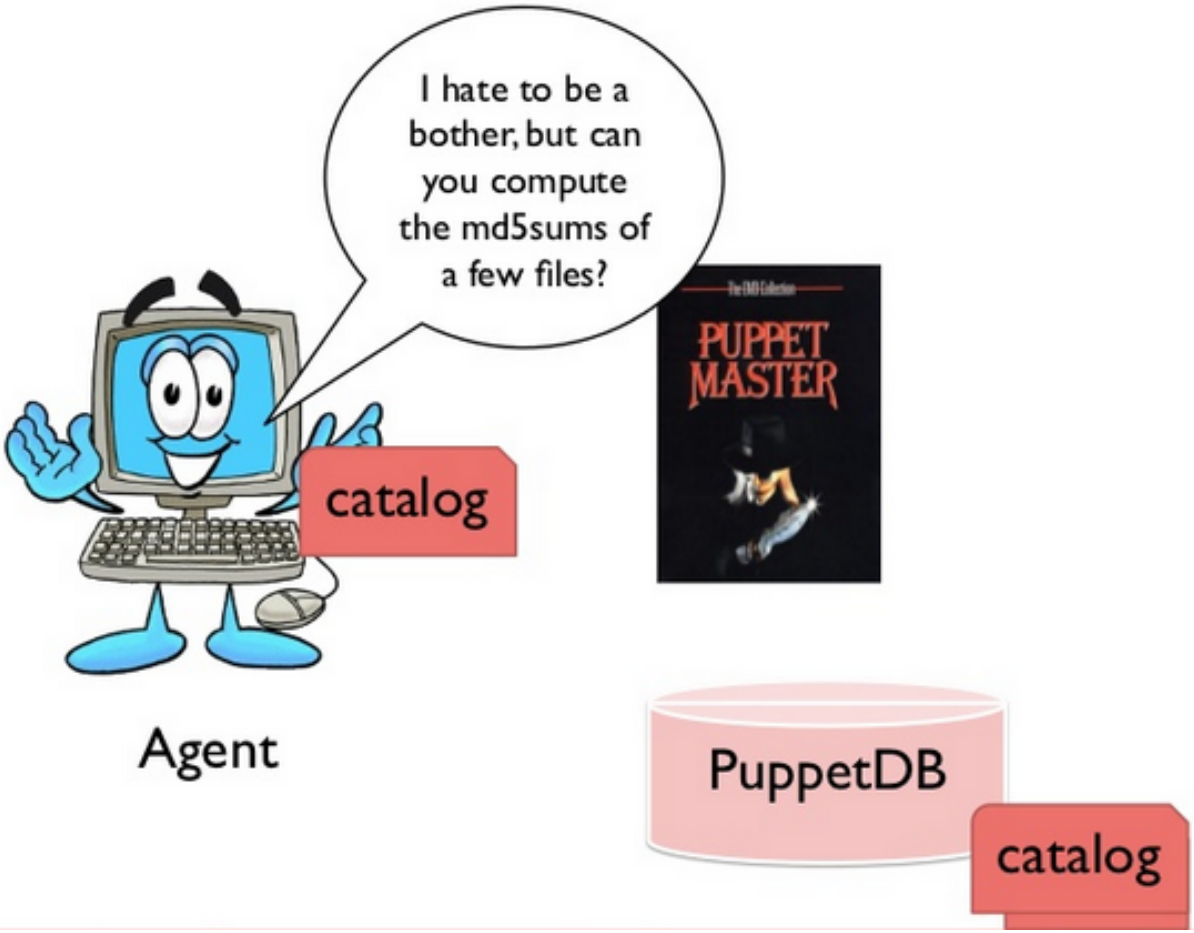

Mr. ENC, is this host defined as an external node?
Yep, he should be an apache server. Here is the definition
PUPPET MASTER
nodes
ENC
Agent
PuppetDB
facts


PUPPET MASTER
catalog
Just compiled your catalog. One sec while I store it in PuppetDB.
PuppetDB
facts

Agent

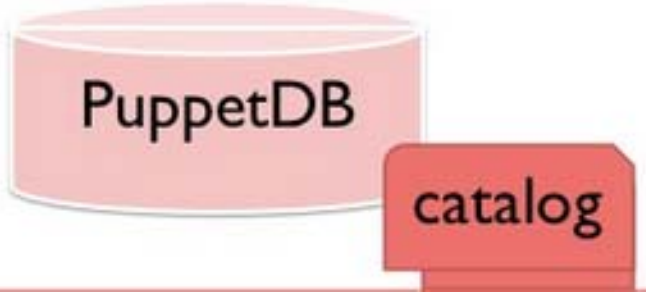

Here is your catalog. Send me a report and let me know how it went!
PUPPET MASTER
catalog


PuppetDB
catalog

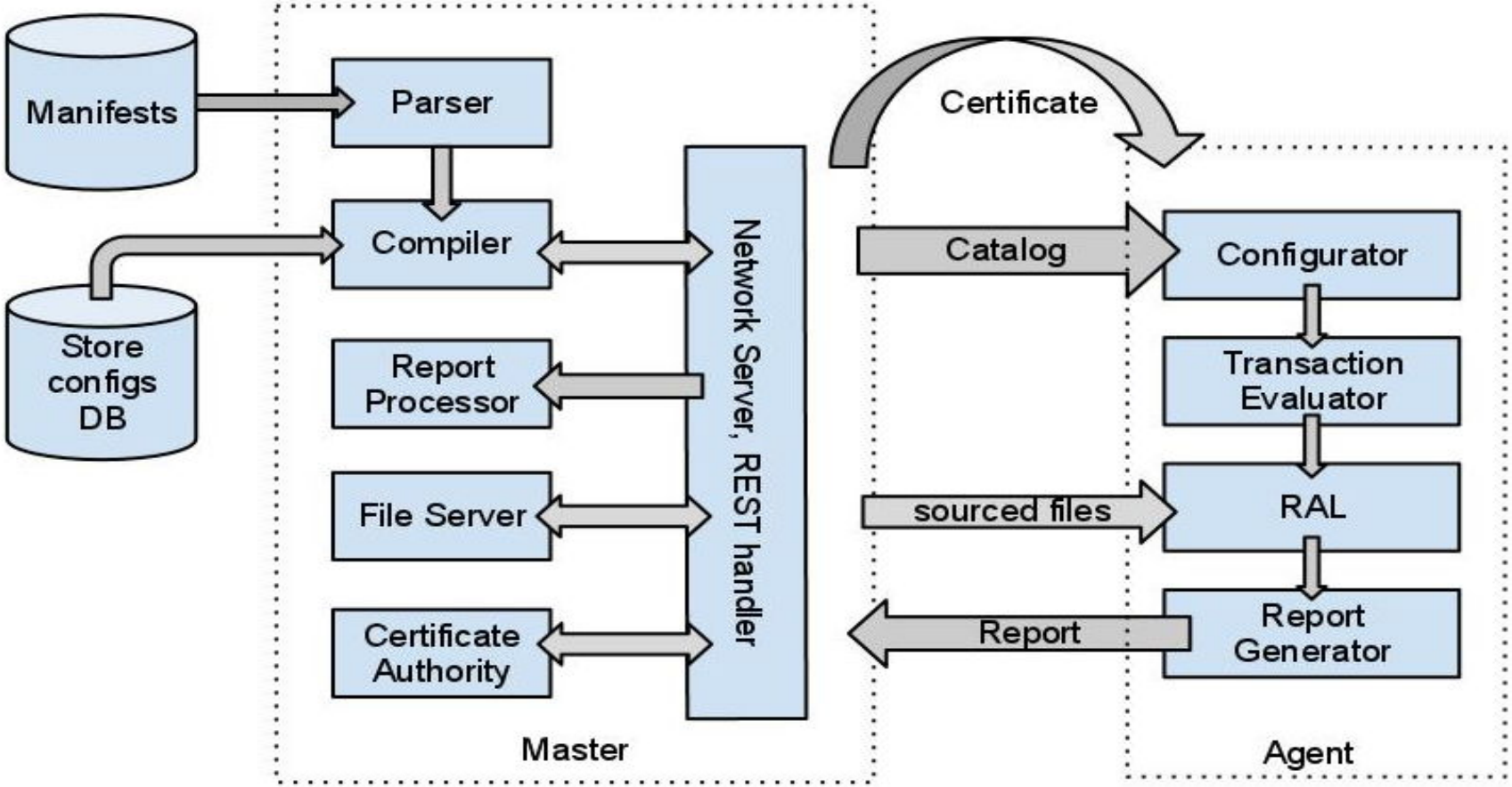

I hate to be a bother, but can you compute the md5sums of a few files?
catalog
PUPPET MASTER
Agent
PuppetDB
catalog


report
Just finished applying.
Here are the results.


PUPPET
MASTER


PuppetDB
catalog

puppet工作流程


Manifests
Parser
Compiler
Store configs DB
Report Processor
File Server
Certificate Authority
Network Server, REST handler
Master
Certificate
Catalog
Configurator
Transaction Evaluator
sourced files
RAL
Report
Report Generator
Agent

关于马哥教育

- 博客：http://mageedu.blog.51cto.com
- 主页：http://www.magedu.com
- QQ：2813150558，1661815153，113228115
- QQ群：203585050，279599283

马哥教育

Thank You!