Configuration Management
- Using Puppet

lctseng / Liang-Chi Tseng

Our Target

- Share several files
  - Configuration files (httpd.conf)
  - Regular data files (wallpapaer.jpg)
- System-wide configuration
  - Add/Modify users
  - Install applications
  - Enable / disable services
- Execute specific scripts on each host
- Manage large number of client hosts, but…
  - No human operation on each host
  - No expensive servers (means low-loading on servers)

Type of File Management

- Exported by some host (shared client-server)
  - Only one copy on server
  - NFS、NIS
  - Online shared
  - When server die: all functionality stopped
- Pushed by some host (deployment)
  - Eventually one copy for each client
  - Puppet, Salt-stack, Chef
  - Offline distributed
  - When server die: host still works fine
    - Although the information may not up-to-date

Puppet – Introduction (1)

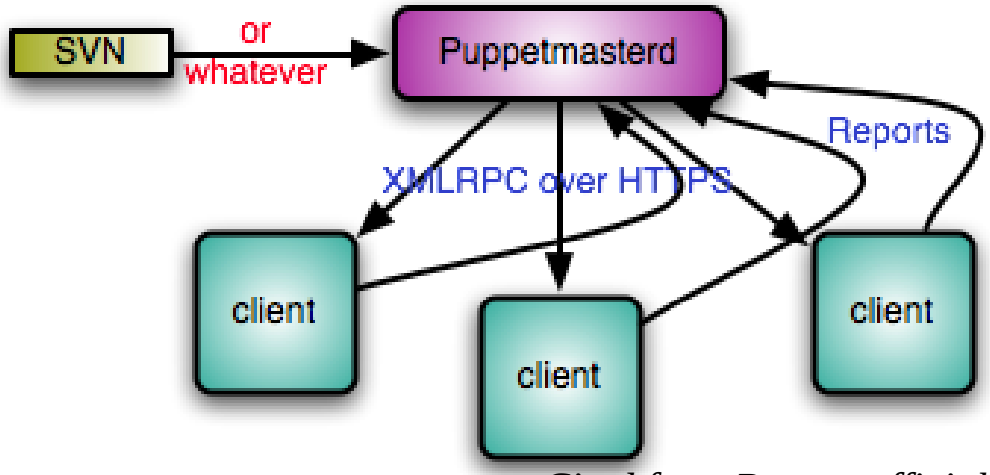
- A configuration management system implemented in Ruby
  - Published by PuppetLabs
  - Open source version is available
  - No need to know about Ruby, puppet has its own language
- Master-Agent architecture
- Cross platform agents
  - FreeBSD, Linux, even Windows
- Use 'push' model to share files
  - Every agent will receive one copy
- Low load on master
  - Nothing to do if all agents are up-to-date
  - A moderate server can serve hundreds of agents

Puppet – Introduction (2)

- Recipe-like design
  - Write ‘recipe’ to construct a host
  - Called ‘manifest’
  - Only care about ‘result’
    - Almost no system-specific command needed
    - E.g. Declared that user ‘lctseng’ must exist with passowd “uccu”
      - Don’t need to write ‘adduser’ command in FreeBSD
- Module design
  - Separate functionality info modules
  - Enable / Disable modules on the fly
- Many community recourses
  - Install variety of modules from “puppet forge”

Puppet – Architecture

- Master (server)
  - Write and keep the manifests
  - Passively wait for connection from agents
- Agent (client)
  - Fetch manifests from master (periodically or manually)
  - Compare and execute manifests if needed
  - Report status to master



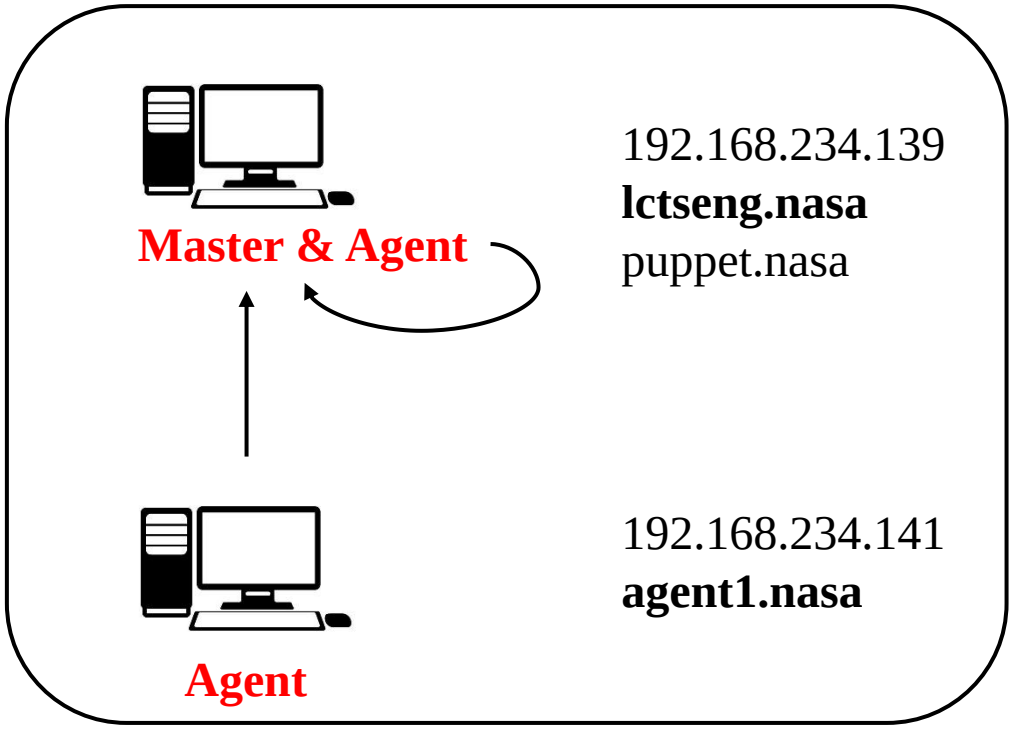
Cited from Puppet official site

In this course

- Time limitation
  - Only featured functionality is discussed
  - Basic deployment on both master and agent
- A simple project involved master and agent
  - Simple manifest and module building
  - Step by step introduction
  - May involved skills required in homework
- Reference
  - Puppet document: Quick Start
    - http://docs.puppetlabs.com/puppet/4.3/reference/quick_start.html
  - Puppet on FreeBSD
    - https://deuterion.net/sysadmin/2014/08/25/puppet-master-agent-installation-on-freebsd/

Simple Project - Overall Architecture

- 1 master, 2 agents
  - Master is also an agent of itself
  - Within VMWare virtual network, no DNS services, using /etc/hosts



Simple Project - Functionality

- Different manifests for different agents
- File distribution
  - Different from NFS
  - Everything store in local filesystem
  - After transfer, files can be access even agents are offline
- Offline user management
  - Different from NIS
  - Manage local user
- Service deployment
  - Install and activate services on all agents
- Command execution
  - Execute programs on all agents

Setup a puppet master (1) - Installation

- sysutils/puppet4
  - Port or pkg
  - Don’t install version 3(sysutils/puppet37), syntax is different!
- Enable in /etc/rc.conf
  - puppetmaster_enable="YES"
- Configuration directory
  - /usr/local/etc/puppet/
  - All configuration & manifest files

Setup a puppet master (2) - puppet.conf for master

- Generate a sample configuration file
  - puppet master --genconfig > /usr/local/etc/puppet/puppet.conf
  - Permission: save as another file and *sudo* rename it
- Edit puppet.conf
  - Only show different parts

```
[main]
  environmentpath = /usr/local/etc/puppet/environments
```

Setup a puppet master (3) - Other configuration

- Create directories for manifests and modules
  - mkdir -p /usr/local/etc/puppet/environments/production/manifests
  - mkdir -p /usr/local/etc/puppet/environments/production/modules
- Create environment configuration file
  - /usr/local/etc/puppet/environments/production/environment.conf

```
modulepath = /usr/local/etc/puppet/modules:/usr/share/puppet/modules
manifest = /usr/local/etc/puppet/environments/production/manifests/site.pp
```

- Firewalling (Access control in Puppet)
  - Rename auth.conf-dist info auth.conf
  - Add your IP entry in the top of file to allow access

```
path ~ /puppet/v3/.*
auth no
allow_ip 192.168.0.0/16, 127.0.0.1
```

Setup a puppet master (4) - Manifest

- Recipe written in puppet DSL language (*.pp)
  - Domain Specific Language
- Describe a ‘result’
  - Don’t need to know about commands on specific system
- Main manifest
  - /usr/local/etc/puppet/environments/production/manifests/site.pp
  - The entry point
  - Sample configuration (create an empty file on agents)

```
node default {
  file { '/tmp/test-puppet':
    ensure      => present,
  }
}
```

  - For each(default) agent, create an empty file: /tmp/test-puppet

Setup a puppet master (5) - Start Master Daemon

- Start service via rc script or 'service'
  - service puppetmaster start
- There may be some warning messages
  - Warn for deprecation settings
  - Don't care in most cases

```
7:23pm lctseng@lctseng(192.168.234.139)[/tmp]
[W1] > sudo service puppetmaster start
Warning: Setting configtimeout is deprecated.
   (at /usr/local/lib/ruby/site_ruby/2.1/puppet/settings.rb:1142:in `issue_deprecation_warning
Performing sanity check of puppetmaster configuration: Warning: Setting configtimeout is depre
   (at /usr/local/lib/ruby/site_ruby/2.1/puppet/settings.rb:1142:in `issue_deprecation_warning
OK
Starting puppetmaster.
```

Setup a puppet master as agent (1)

- Configure master as agent of itself
  - Much easier than normal agents
- Add puppet_enable="YES" in /etc/rc.conf
- Add [agent] section in puppet.conf
  - Server name should be master's hostname
  - Find hostname by 'hostname' command

```
[main]
  …
[agent]
  server = puppet.nasa
[master]
  …
```

Setup a puppet master as agent (2)

- Edit /etc/hosts
  - Add your puppet master's hostname in 127.0.0.1 entry
  - Agents must have the same domain as master!

```
::1                     localhost localhost.my.domain
127.0.0.1               localhost localhost.my.domain puppet.nasa
```

- Start puppet agent daemon
  - service puppet start

Setup a puppet master as agent (3)

- Trigger puppet manually to fetch manifest
  - puppet agent –t
    - t : trigger

```
9:26pm lctseng@lctseng(192.168.234.139)[/usr/local/etc/puppet/manifests]
[W2] > sudo puppet agent -t
Warning: Setting configtimeout is deprecated.
   (at /usr/local/lib/ruby/site_ruby/2.1/puppet/settings.rb:1142:in `issue_deprecation_warnir
Info: Using configured environment 'production'
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Caching catalog for lctseng.nasa
Info: Applying configuration version '1450185992'
Notice: /Stage[main]/Main/File[/tmp/test-puppet]/ensure: created
Notice: Applied catalog in 0.02 seconds
9:26pm lctseng@lctseng(192.168.234.139)[/usr/local/etc/puppet/manifests]
[W2] > ls /tmp/test-puppet
/tmp/test-puppet
```

Setup a normal puppet agent (1) - Basic setup

- Install the same version of puppet
- Add puppet_enable="YES" in /etc/rc.conf
- Copy puppet agent config file
  - Rename puppet.conf-dist to puppet.conf
  - Edit [agent] section

```
[agent]
  server = puppet.nasa
```

(around line 980)

    - Fill master's hostname
- Add master's hostname in /etc/hosts
  - Need to find out master's IP first

```
192.168.234.139          puppet.nasa puppet
```

Setup a normal puppet agent (2) - Basic setup

- Ensure your hostname is under the same domain as master
  - You can use “bsdconfig” command to change hostname
    - Networking Management → Hostname/Domain
  - For example, set agent’s hostname as ‘agent1.nasa’
    - While the master could be ‘puppet.nasa’
- Start puppet service
  - service puppet start

Setup a normal puppet agent (3) - Signing certificate

- We don't want strangers to access our manifest
  - Master needed to sign the new agent
- When first run 'puppet agent -t' on agents

```
8:44pm lctseng@agent1(192.168.234.141)[/usr/local/etc/puppet]
[W2] > sudo puppet agent -t
Warning: Setting configtimeout is deprecated.
   (at /usr/local/lib/ruby/site_ruby/2.1/puppet/settings.rb:1
Exiting; no certificate found and waitforcert is disabled
```

- We need to go back to master and sign for this agent

Setup a normal puppet agent (4) - Master signs for new agent

- On master, we can list all certificates
  - puppet cert --list --all

```
  "agent1.nasa"  (SHA256) 18:E9:A2:75:BD:2C:D…
+ "lctseng.nasa" (SHA256) FF:74:3C:85:44:88:D… (alt names: "DNS:puppet.nasa")
```

New agent

Master itself

  - "+" in the beginning : signed agents
- Then we can sign the new agent
  - puppet cert --sign *agent1.nasa*
- List the certificate again
  - Both signed

```
+ "agent1.nasa"  (SHA256) 18:E9:A2:75:BD:2C:D…
+ "lctseng.nasa" (SHA256) FF:74:3C:85:44:88:D… (alt names: "DNS:puppet.nasa")
```

- Add agent's IP into master/agent's /etc/hosts

```
192.168.234.141      agent1.nasa
```

Setup a normal puppet agent (5) - Trigger

- Agent trigger
  - puppet agent –t

```
10:12pm lctseng@agent1(192.168.234.141)[/usr/local/etc/puppet]
[W2] > sudo puppet agent -t
Warning: Setting configtimeout is deprecated.
   (at /usr/local/lib/ruby/site_ruby/2.1/puppet/settings.rb:1142:in `issue_deprecation_warni
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Caching catalog for 101amd64-quarterly-job-23.nyi.freebsd.org
Info: Applying configuration version '1450192859'
Notice: /Stage[main]/Main/Node[__node_regexp__agent1.nasa]/File[/tmp/test-puppet-agent]/ensu
Notice: Applied catalog in 0.02 seconds
```

Simple Project – Node definition(1)

- In previous site.pp, manifest applied for all agents
  - "node default" means all node without specific settings
- Node definition: make difference between agents
- Basic syntax

```
node <node-name1> {
  <manifests>
  <manifests>
  …
}
node <node-name2> {
  <manifests>
  <manifests>
  …
}
```

- Reference
  - https://docs.puppetlabs.com/puppet/latest/reference/lang_node_definitions.html

Simple Project – Node definition(2)

- For example, we want to push different files to different agents

```
# For all agents who doesn’t match
node default {
  file { '/tmp/test-puppet-default':
    ensure      => present,
  }
}
# for agent1.nasa
node /^agent1.nasa$/ {
  file { ‘/tmp/test-puppet-agent1':
    ensure      => present,
  }
}
```

Every option should ended with comma!

- Result
  - lctseng.nasa: /tmp/test-puppet-default
  - agent1.nasa: /tmp/test-puppet-agent1

Simple Project – Class definition(1)

- If we want to do similar settings on multiple nodes
- Define a class and put manifests in it

```
class <class-name> {
  <manifests>
  <manifests>
  …
}
```

- Then **include** it in node definition

```
node <node-name> {
  include <class-name1>
  include <class-name2>
  …
  <manifests>
  <manifests>
  …
}
```

- Reference
  - https://docs.puppetlabs.com/puppet/latest/reference/lang_classes.html

Simple Project – Class definition(2)

- For example, we want to create common files on both agents

```
class common_file {
  file { '/tmp/test-puppet-common':
    ensure      => present,
  }
}
```

- **include** it in node definition

```
node default {
  include common_file
  file { '/tmp/test-puppet':
    ensure      => present,
  }
}
```

Simple Project – Class definition(3)

- Classes can be nested

```
class common_file {
  file { '/tmp/test-puppet-common':
    ensure      => present,
  }
}

class basic_settings {
  include common_file
}

node default {
  include basic_settings
  file { '/tmp/test-puppet':
    ensure      => present,
  }
}
```

Simple Project – Module definition(1)

- Puppet has modular design
  - It not a good idea to put all things in site.pp
  - Easy to manage with VCS (Git, SVN)
- Split your classes into modules
- Module path
  - /usr/local/etc/puppet/modules/<module-name>/
- Two subdirectories
  - <module-name>/manifests
    - *.pp files
  - <module-name>/files
    - Files that needed to transfer to agents
- Reference
  - https://docs.puppetlabs.com/puppet/latest/reference/modules_fundamentals.html

Simple Project – Module definition(2)

- For example, rewrite ***class basic_settings*** into module
- Step 1: create module directories
  - Under /usr/local/etc/puppet/modules
  - mkdir -p basic_settings/manifests
  - mkdir -p basic_settings/files
- Step 2: write entry point manifest for this module
  - Create basic_settings/manifests/**init.pp**
  - Write your class definition inside

```
class basic_settings {
  include basic_settings::common_file
  <other-manifests>
  …
}
```

Simple Project – Module definition(3)

- Step 3. define submodule
  - Like namespace in C++
  - Create submodule file: basic_settings/manifests/common_file.pp

```
class basic_settings::common_file {
  file { '/tmp/test-puppet-common':
    ensure      => present,
  }
}
```

- Step 4. In node definition, only include **basic_settings**

```
node default {
  include basic_settings
  <other-manifests>
  …
}
```

Simple Project – File Distribution(1)

- Transfer files/directories from master to agents
  - Not just create an empty file!
  - Must write in modules, **not in site.pp**
- Reference
  - https://docs.puppetlabs.com/references/latest/type.html#file

Simple Project – File Distribution(2)

- Syntax
  - Assume your file/directory is <module-name>/**files**/<file-name>

```
file { '/agent/path/to/file'
  ensure => [present|absent|directory],
  source => 'puppet:///modules/<module-name>/<file-name>‘,
  recurse => true, # for directory copy
  owner       => <owner user>,
  group       => <owner group>,
  mode        => <permission>
}
```

Note: ‘files’ directory must not appear

  - ensure => directory : only for create empty directory
  - recurse:
    - For directory
    - If absent, only create an empty directory
  - mode
    - File permissions
    - Could be '0644' or 'ug+rwx'

Simple Project – File Distribution(3)

- Example
  - In basic_settings::commin_file
  - Send /etc/motd file and /usr/local/etc/apache24 directory
  - Copy them under basic_settings/files/

```
class basic_settings::common_file {
  file { '/etc/motd':
    ensure      => present,
    source      => 'puppet:///modules/basic_settings/motd',
    owner       => 'root',
    group       => 'wheel',
    mode        => '0644',
  }
  file { '/usr/local/etc/apache24':
    ensure      => present,
    source      => 'puppet:///modules/basic_settings/apache24',
    recurse     => true,
  }
}
```

Simple Project – User Management(1)

- Create/Modify users
- There are many features can applied
  - Only select several important options
- Reference
  - https://docs.puppetlabs.com/references/latest/type.html#user

Simple Project – User Management(2)

❑ Syntax

```
user { <user-name>:
     ensure    => [present|absent],
     uid       => <uid>,
     groups    => <array-of-groups>,
     home      => <home-directory>,
     password  => <hashed-password>,
     shell     => <full-path-of-shell>,
 }
```

- Home directory will not be created
  - Use ‘file’ statement mentioned previously
- Password must in hashed format
  - What you see in master.passwd
  - E.g.
    $6$8psV5ze69dPWnyeU$eWBb5x1O80mFD6984ZQ/oiB/y1oK89eZakQCnBQ
    LptO3MBeqTJvJ3gV31XQbUmKfYZMdn3/vYniMTKrGX643a/

Hashed password of ‘5566’

Simple Project – User Management(3)

- Example

```
user { 'ppuser':
     ensure    => present,
     uid       => 15566,
     groups    => ['wheel','staff'],
     home      => '/home/ppuser',
     password  => '$6$8psV5ze…MTKrGX643a/',
     shell     => '/bin/sh',
}
```

- Create an empty home directory for him

```
file { '/home/ppuser':
     ensure => directory,
 }
```

Simple Project – Service Deployment - Install application(1)

- Install application on agents
- Different systems have different management tools
  - pkg, pacman, yum, apt …
  - Must specify which tool to use
    - But there are default settings for each OS
    - Puppet already handles it, don't worry ☺
- Reference
  - https://docs.puppetlabs.com/references/latest/type.html#package

Simple Project – Service Deployment - Install application(2)

- Syntax
  - Simpest one

```
package { <sofware-name>:
     ensure    => [present|absent],
     provider  => <provider>,
}
```

  - For FreeBSD, provider can use ‘pkgng’ or ‘ports’
    - ‘pkgng’ is the default value for FreeBSD agent
      - The pkg command
    - Note: provider ‘pkg’ is for OpenSolaris

Simple Project – Service Deployment - Install application(3)

- Example
  - Install apache24 from pkgng (the default provider)

```
package { 'apache24':
     ensure    => present
}
```

  - When puppet agent -t, wait for very long time
  - Apache24 is installed!!

```
1:48am lctseng@agent1(192.168.234.141)[/usr/local/etc/puppet]
[W2] > sudo puppet agent -t
Warning: Setting configtimeout is deprecated.
   (at /usr/local/lib/ruby/site_ruby/2.1/puppet/settings.rb:1142:in `issue_deprecation_
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Caching catalog for 101amd64-quarterly-job-23.nyi.freebsd.org
Info: Applying configuration version '1450255257'
Notice: /Stage[main]/Basic_settings::Common_service/Package[apache24]/ensure: created
Notice: Applied catalog in 87.32 seconds
```

```
1:48am lctseng@agent1(192.168.234.141)[/tmp]
[W4] > pkg info | grep apache
apache24-2.4.16_1              Version 2.4.x of Apache web server
```

Simple Project – Service Deployment - Service activation(1)

- After installation, we need to activate the service
- Syntax

```
service { <service-name>:
   ensure => [running|stopped],
   enable => [true|false],
 }
```

  - ensure : start the service
    - service <service-name> onestart
  - enable : start the service at boot time
    - Modify /etc/rc.conf.d/<service-name>
    - service <service-name> start
- Reference
  - https://docs.puppetlabs.com/references/latest/type.html#service

Simple Project – Service Deployment - Service activation(2)

❑ Example

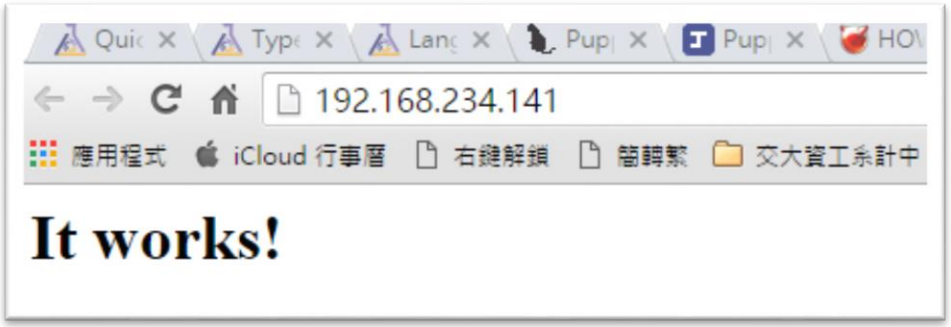
- Start apache24 service and start it when booting

```
service { 'apache24':
   ensure => running,
   enable => true,
}
```

- Puppet adds new file: /etc/rc.conf.d/apache24

```
# Added by Puppet
apache24_enable="YES
```

- It works!



Manifest Dependency(1)

- Wait !! We need to install apache24 before we start service
- Also, we want to transfer apache24 configuration after we installed apache24
- But puppet manifest is unordered by default
  - Order of codes is meaningless!!
- To specify order

```
service { 'apache24':
   require => Package['apache24'],
   ensure => running,
   enable => true,
}
package { 'apache24':
   ensure => present,
}
```

- Reference
  - https://docs.puppetlabs.com/puppet/latest/reference/lang_relationships.html

Manifest Dependency(2)

- Another example: multiple dependency
  - Note: will not refresh apache24 even files in etc/apache24 changed
    - This script only ensures that etc/apache24 exists before starting service

```
package { 'apache24':
   ensure => present,
 }

package { 'mod_php5':
   ensure => present,
 }

file { '/usr/local/etc/apache24':
   ensure      => present,
   source      => 'puppet:///modules/basic_settings/apache24',
   recurse     => true,
 }

service { 'apache24':
   require => [File['/usr/local/etc/apache24'],Package['apache24','mod_php5']],
   ensure => running,
   enable => true,
 }
```

Simple Project – Command execution(1)

- Execute commands and programs on agents
- Syntax

```
exec { <unique-name>:
  command  => <command-to-execute>,
  cwd      => <current-working-directory>,
  path     => <array-of-path-to-search>,
  user     => <run-as-whom>,
  provider => [posix|shell],
}
```

  - Provider
    - posix : execute binary directly (default, much safer)
    - shell : pass to /bin/sh
- Reference
  - https://docs.puppetlabs.com/references/latest/type.html#exec

Simple Project – Command execution(2)

- Example
  - Record the time when puppet triggers

```
exec { "record date":
  command => "echo `date` >> trigger-time",
  cwd     => "/tmp",
  path    => ["/bin","/sbin","/usr/bin","/usr/sbin"]
}
```

  - Archive /var/log/message* when puppet triggers

```
exec { "archive log":
   command => "tar -cf `date '+%Y-%m-%d_%H:%M:%S.tar'` messages*",
   cwd     => "/var/log",
   path    => ["/bin","/sbin","/usr/bin","/usr/sbin"],
   user    => 'root',
 }
```

Simple Project – Command execution(3)

❑ Example : transfer a script/program and then execute

```
file { '/root/hello.out':
  ensure    => present,
  mode      => '0755',
  source    => 'puppet:///modules/basic_settings/hello.out',
}

exec { 'run hello':
  require   => File['/root/hello.out'],
  command   => "hello.out >> trigger-time",
  cwd       => "/tmp",
  path      => "/root",
  user      => 'root',
}
```

- Note: execute whenever triggered, not only once

Simple Project - Review

- Different manifests for different agents
  - site.pp
- File distribution
  - File: /etc/motd
  - Directory: /usr/local/etc/apache24
- Offline user management
  - Local user: ppuser
  - Create/update information(password)
- Service deployment
  - Apache24: installed via ‘pkg’ and run with ‘service’
- Command execution
  - Built-in: echo, tar
  - Custom: hello.out

We are running out of time

- Still many to be discussed…
  - Variables
  - Expressions
  - Conditions
  - Class parameters
  - Templates
  - Advanced ordering and relationship
  - …

- Go and find it by yourself! If you're still interested in it
  - http://docs.puppetlabs.com/puppet/latest/reference/