

Systems Automation

With Puppet

Ohad Levy
Senior Staff Engineer at
Infineon Technologies

Tel Aviv Linux Club
07/09/2008

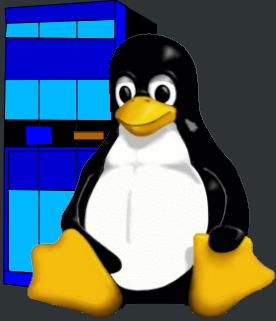


Warning

Many of the slides are copied! :)



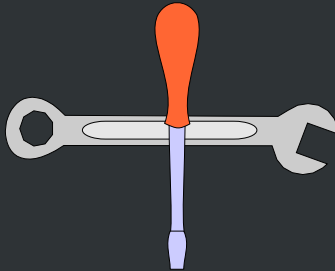
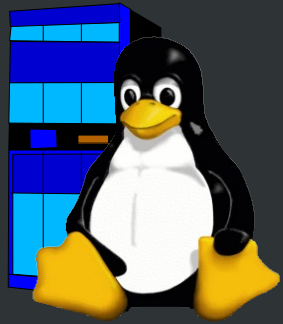
Typical System Life cycle



Installation



Typical System Life cycle

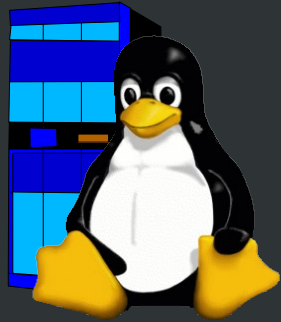


Installation

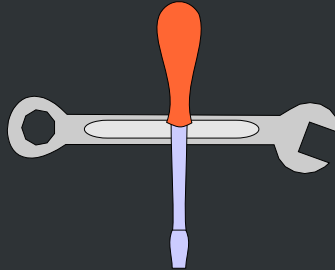
Initial
Configuration



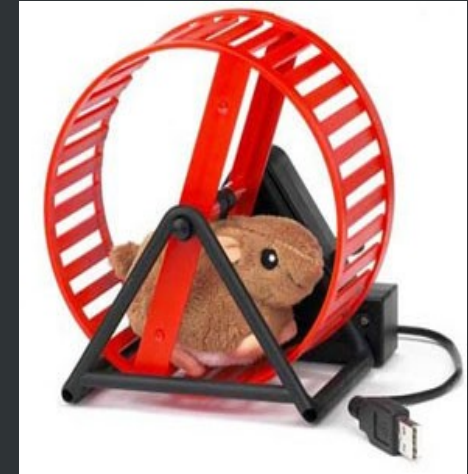
Typical System Life cycle



Installation



Initial
Configuration



Fixes
Updates
Audits



The Challenges

- Keep our systems "harmonized"
- Know what's going on on each system
- Replace a server if it dies or to be able to add another server that is exactly like it
- Similar Applications, different OS's
- Push out changes to all the servers that need a particular change
- Stop duplicating effort
- Go home early

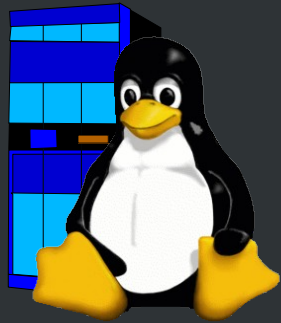


How to solve the problem?

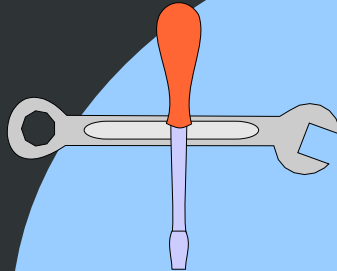
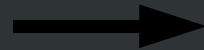
- The Manual way
 - Log in and do it
 - Thats OK only for a limited set of machines...
- Install time auto configure
 - Kickstart, jumpstart etc, with a post installation scripts
- But then what?
 - How to push a change?
 - No History of changes, audit etc...
- Or....



Puppet Life cycle



Installation



Initial
Configuration



Puppet



Fixes
Updates
Audits

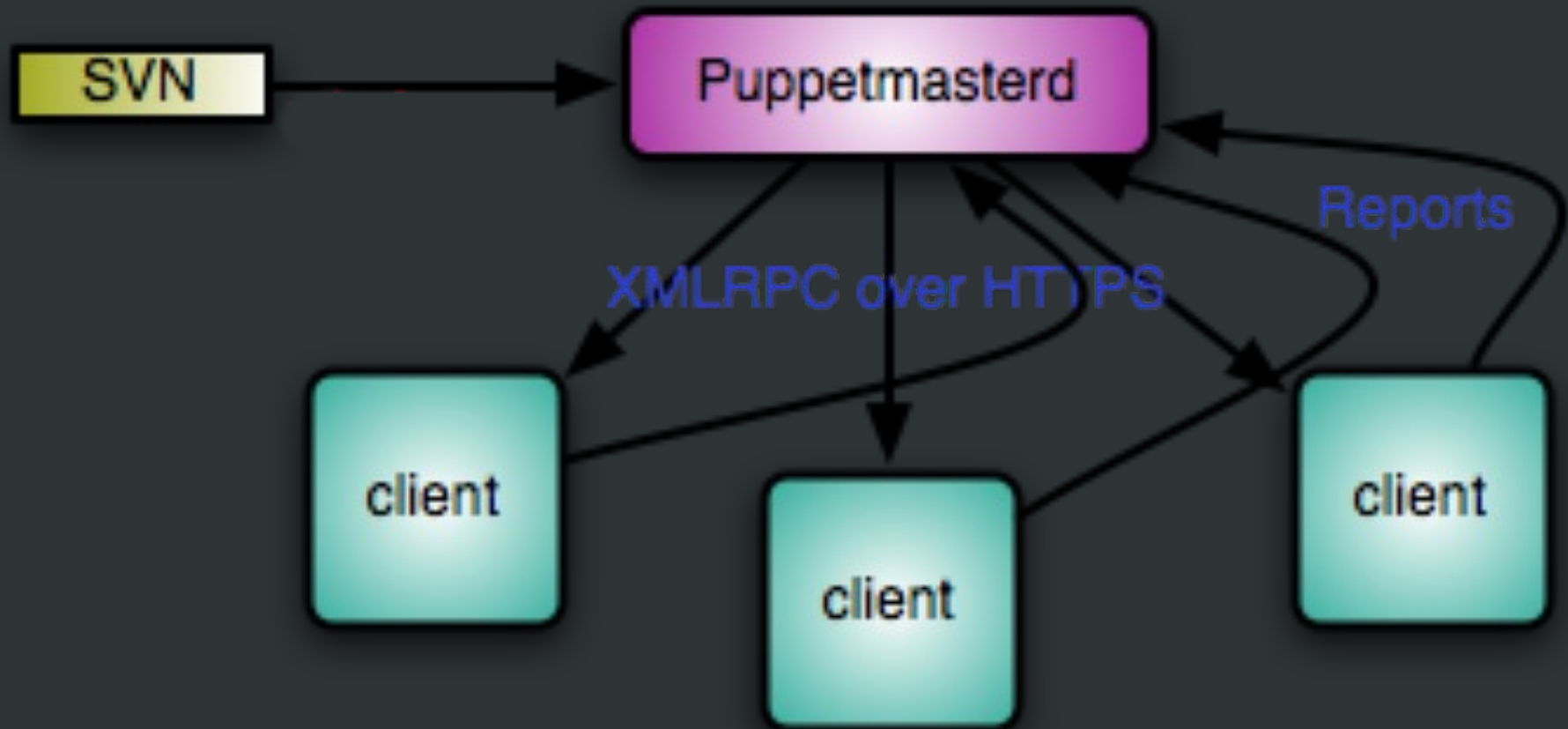


What is Puppet?

- A GPL Open Source Project written in Ruby
- A declarative language for expressing system configuration
- A Client and server
- A library to realize the configuration
- Puppet is the abstraction layer between the system administrator and the system
- Puppet requires only Ruby and Facter
- Client runs every 30 minutes by default



Puppet components



Puppet Types

A Type is a particular element that Puppet knows how to configure

- Files (content, permissions, ownership)
- Packages (ensure installed or absent)
- Services (enabled/disabled, running/stopped)
- Exec (run commands)
- Full List: cron, exec, file, filebucket, group, host, interface, k5login, mailalias, maillist, mount, nagios*, package, service, sshkey, tidy, user, yumrepo, zone



Example: Managing sudoers file

```
file { “/etc/sudoers”:  
    ensure => file,  
    owner  => root,  
    group  => root,  
    mode   => 600,  
    source => “puppet://server/files/sudoer”  
}
```



Dependencies

“require” and “before” / “after” settings ensures that types are applied in the correct order

```
file { "/etc/sudoers":  
    ...  
    require => Package[sudo]  
}  
  
package { "sudo":  
    ensure => present,  
    before => File[ "/etc/sudoers" ]  
}
```



Dependencies - continued

- “notify” and “subscribe” settings can trigger cascaded updates
- Particularly useful in services, exec

```
file { "/etc/ssh/sshd_conf":  
    ...  
    notify => Service["sshd"]  
}
```

```
service { "sshd":  
    subscribe => File["/etc/ssh/sshd_conf"]  
}
```



What is Facter?

- Facter gathers information about the client, which can be used as variables within puppet.
- You can add custom facts as needed.

```
package {"sshhd":  
  
  ensure => installed,  
  
  name => $operatingsystem ? {  
    solaris => "IFKLssh",  
    default => "openssh-server"  
  }  
}
```



Example Facts

```
$ sudo facter
```

```
architecture => amd64
```

```
domain => sin.infineon.com
```

```
facterversion => 1.3.8
```

```
fqdn => sinn1636.sin.infineon.com
```

```
hardwaremodel => x86_64
```

```
hostname => sinn1636
```

```
ipaddress => 172.20.88.132
```

```
kernel => Linux
```

```
kernelrelease => 2.6.24-16-generic
```

```
lsbdistcodename => hardy
```

```
lsbdistdescription => Ubuntu 8.04
```

```
lsbdistid => Ubuntu
```

```
lsbdistrelease => 8.04
```

```
macaddress => 00:1c:25:14:26:ab
```

```
manufacturer => LENOVO
```

```
memorysize => 1.94 GB
```

```
processorcount => 2
```

```
puppetversion => 0.24.4
```

```
rubysitedir =>  
  /usr/local/lib/site_ruby/1.8
```

```
rubyversion => 1.8.6
```



What is a Class?

- A named collection of type objects
- Can include or inherit from other classes

```
class sudo_class {  
    include foo_class  
    file { "/etc/sudoers":  
        ...  
    }  
    package{ "sudo":  
        ...  
    }  
}
```



Class inheritance

```
class afile {  
  file { "/tmp/foo":  
    ensure => file  
    source => "/src/versionA"  
  }  
}  
  
class another_file inherits afile {  
  File["/tmp/foo"] {  
    source => "/src/versionB"  
  }  
}
```



What is a Node ?

- A configuration block matching a client
- Can contain types, classes
- “default” node matches any client without a node block

```
node "ohad.myself" {  
    include sudo_class  
    include other_class  
}
```



External Node

- Node definitions can be defined outside of puppet - LDAP, external script
- Ideal for sites with too many nodes to bother pre-creating



Classes and definitions

- Classes are groups of resources.
- Definitions are similar to classes, but they can be instantiated multiple times with different arguments on the same node.

```
class apache2 {  
    define simple-vhost ( $admin = "webmaster", $aliases, $docroot) {  
        file { "/etc/apache2/sites-available/$name":  
            mode      => "644",  
            require => [ Package["apache2"], Service["apache2"] ],  
            content => template("apache/vhost.conf"),  
        } } }  
}
```

```
node debiantest {  
    include apache2  
    apache2::simple-vhost { "debian.example.com": docroot =>  
        "/var/www/debian"}  
    apache2::simple-vhost { "test.example.com": docroot =>  
        "/var/www/test"}  
}
```



vhost.conf template

- Puppet uses Ruby's ERB template system:

```
<VirtualHost *>
    ServerAdmin      <%= admin %>
    DocumentRoot     <%= docroot %>
    ServerName       <%= name %>
    <% aliases.each do |al| -%>
        ServerAlias  <%= al %>
    <% end -%>
    ErrorLog         "/usr/bin/cronolog /var/log/apache/<%=
name %>/%Y-%m/error-%d"
    CustomLog        "/usr/bin/cronolog /var/log/apache/<%=
name %>/%Y-%m/access %d" sane
</VirtualHost>
```



Templates output

```
# more /etc/apache2/sites-available/debian.example.com
<VirtualHost *>
    ServerAdmin      system@example.com
    DocumentRoot    /var/www/debian
    ServerName       debian.example.com
    ServerAlias      debiantest.example.com
    ServerAlias      debian

    ErrorLog        "|/usr/bin/cronolog
/var/log/apache/debian.example.com/%Y-%m/error-%d"
    CustomLog       "|/usr/bin/cronolog
/var/log/apache/debian.example.com/%Y-%m/access-%d" sane

</VirtualHost>
```



OS API - It also works the other way around:

```
$ rals user levyo
```

```
user { 'levyo':
```

```
  password => 'absent',
```

```
  shell => '/bin/bash',
```

```
  ensure => 'present',
```

```
  uid => '49960',
```

```
  gid => '49960',
```

```
  home => '/home/levyo',
```

```
  comment => 'Ohad Levy',
```

```
  groups =>
```

```
  ['adm','dialout','fax','cdrom','floppy','tape','audio','dip','plugdev','scanner','fuse','lp  
  admin','admin']
```



Getting Started

- Install puppet (yum/apt-get install puppet) or install ruby, gem install facter/puppet.
- Setup the puppet server (puppetmaster) – use version control!
- Write a manifest for your node.
- Start puppet master on the server
- Run puppetd on the client



Next steps - modules

- Modules allow you to group both the logic and the files for an application together.
- Puppet automatically searches its module path to find modules.
- Modules can contain four types of files, each of which must be stored in a separate subdirectory:
 - Manifests - must be stored in manifests/, and if you create manifests/init.pp then that file will be loaded if you import the module name directly, e.g. import "mymodule".
 - Templates - must be stored in templates/, and the module name must be added to the template name: `template("mymodule/mytemplate.erb")`
 - Files - stored in files/, these are available from the file server under `modules/<module name>/files/<file name>`.
 - Plugins – additional types, providers or facts.



File server and File Bucket

- Puppet also includes a file server which you can use for transferring files from the server to the client.
- If you configure it, puppet can also save a backup of each file that is changed on the client to the server. The backups go in a filebucket and can be retrieved later.



Some more info

- Puppet uses SSL for all communications, therefore it includes a CA, you can automatically sign CSR or use puppetca tool to manage them.
- Storeconfig, option to save machine states(facts, configuration runs) and special facts (e.g. SSH keys) in a database.
- Reporting, puppet has a few reporting options, most common are emails with changes, RRD files, yaml files and puppetshow web interface.
- Puppet HA, load balancing etc.



Conclusions

- We're all stuck on the hamster wheel
- Makes easy stuff easy, hard stuff possible
- Similar projects
 - cfengine
 - bcfg2
- Additional Resources
 - <http://reductivelabs.com/trac/puppet>
 - <http://reductivelabs.com/trac/puppet/wiki/LanguageTutorial>
 - <http://reductivelabs.com/trac/puppet/wiki/CompleteConfiguration>
 - #puppet on irc.freenode.org

