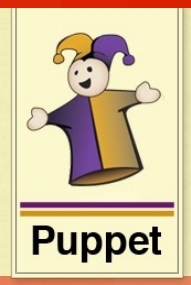
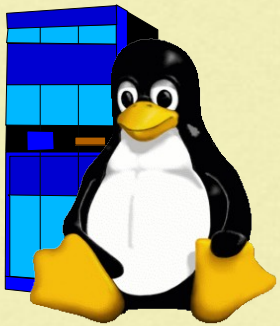
Puppet



Configuring your systems so you don't have to...

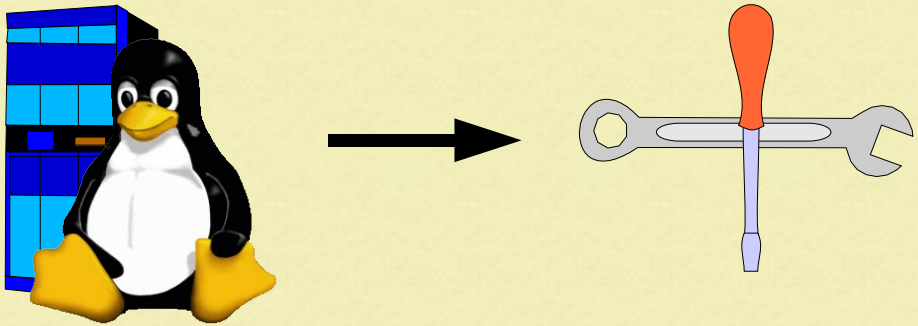
Frank Sweetser
Senior Network Engineer
WPI Network Operations and Security

Typical System Lifecycle



Installation

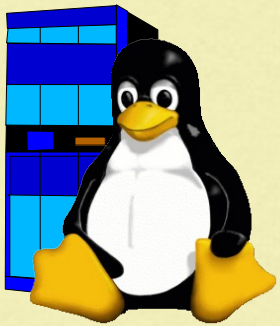
Typical System Lifecycle

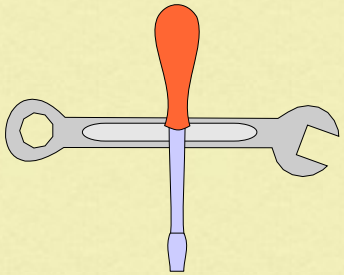


Installation

Initial
Configuration

Typical System Lifecycle



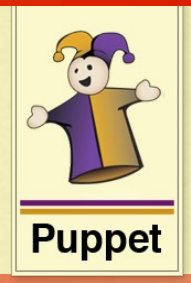


Installation

Initial
Configuration

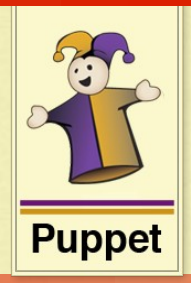
Fixes
Updates
Audits

The Problem



- Systems need to be configured
- The more systems you have, the more work it is to configure them all
- Bad configuration can be worse than no configuration
- How do you make sure all of your systems
  - Get properly configured?
  - Stay that way?

Manual System Configuration

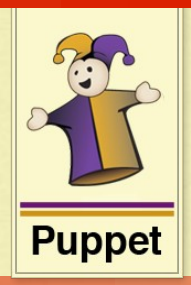


- Just log in and do it!
- Fine for very small number of systems – a *very* small number
- Attempting to scale brings severe risk of carpal tunnel
- Checklists can help... a little
- Settings you care about buried with everything else
- Missing:
  - Auditing
  - History
  - Reliable documentation

Install Time Auto-Configuration

- Kickstart (RedHat), preseed (Debian), Jumpstart (Solaris), etc
- Ensures new systems are brought into proper state as part of installation process
- Then what?
  - Changes in configuration policy
  - Configs derived from dynamic data
  - Validation that settings remain correct
  - Multiple operating systems with varying install support

SSH Keys and Shell Scripting



- Great for ad-hoc or one off items
- Can be pushed out via cron
- But do you *always* write scripts that are:
  - Concurrent safe
  - Testable
  - Idempotent
  - Reversible
  - Legible
  - Full of good logging
  - Portable

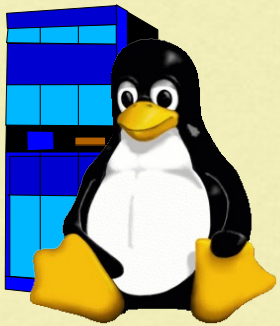
Or ...

Why bother to go through the trouble of carefully writing scripts that are all those things when you could use someone else's?

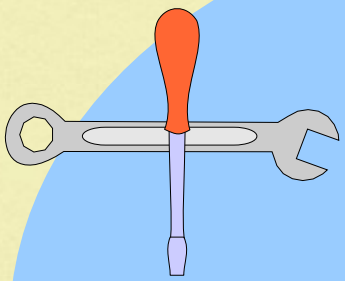
Puppet

- Guaranteed to be everything on that list of good stuff
- Written in Ruby
- Extensible
- Client defaults to polling server for latest config every 30 minutes
- Can listen to have push updates triggered
- Includes built-in file server

Puppet Lifecycle

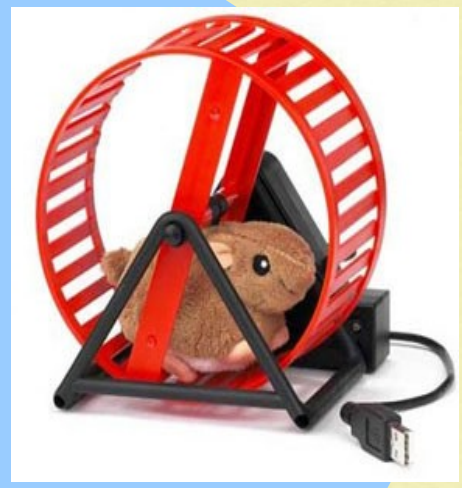


Installation



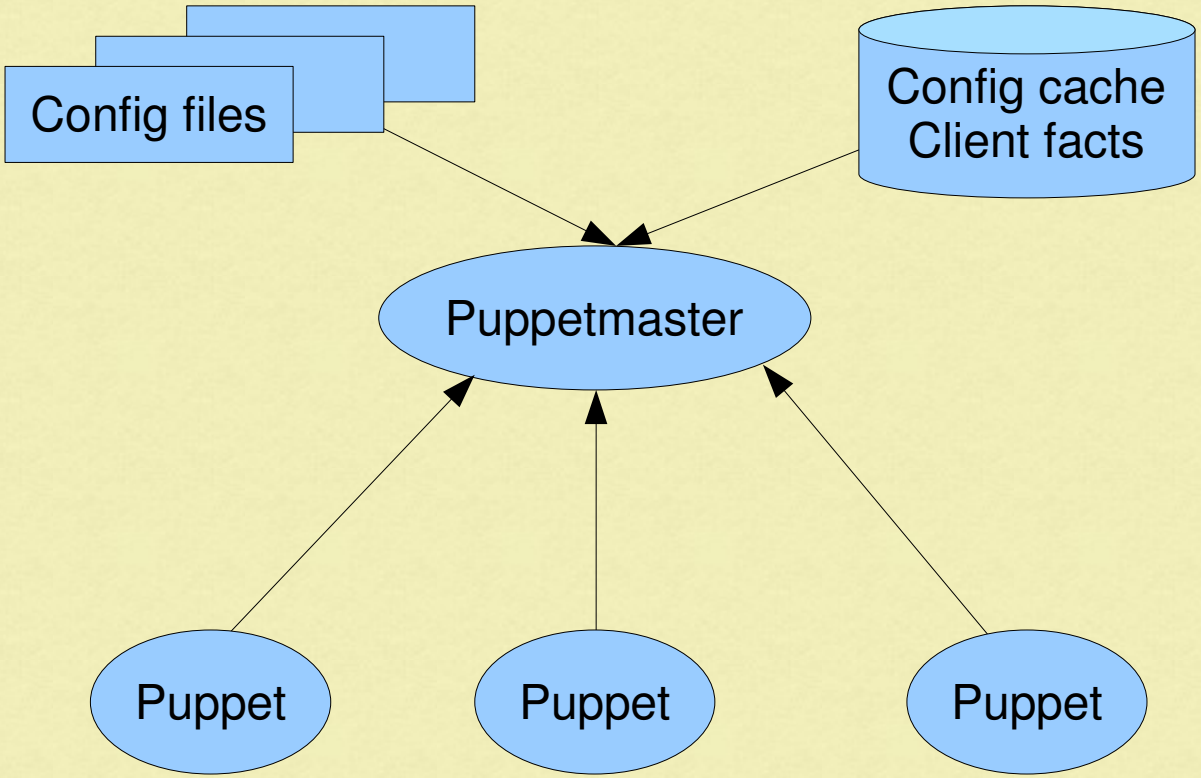
Initial
Configuration

Puppet

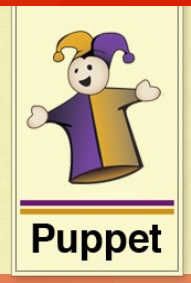


Fixes
Updates
Audits

Puppet Components



Configuration Elements

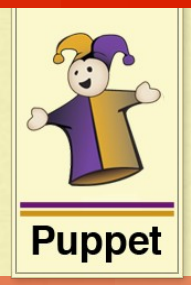


- Types
- Classes
- Nodes
- Templates
- Defines

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)

Example: Managing sudoers File

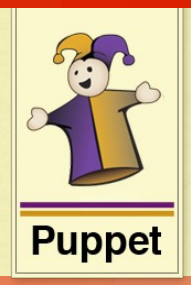



```
file { “/etc/sudoers”:
   ensure => file,
   owner  => root,
   group  => root,
   mode   => 600,
   source => “puppet://server/files/sudoer”
}
```

Native Types Library

- Cron
- Exec
- File
- Filebucket
- Group
- Host
- Mount
- Notify
- Package
- Resources
- Schedule
- Service
- Sshkey
- Tidy
- User
- Yumrepo
- Zone

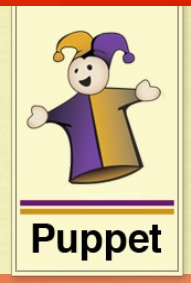
Filebucket



- Special resource for backing up changed files
- Provides complete record of all changes
- Can have multiple filebuckets

```
file { “/etc/sudoers”:
    ...
    backup => backup-bucket
}
```

Classes



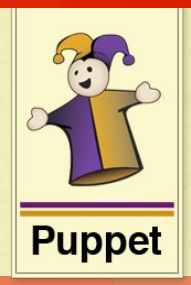
- A named collection of type objects
- Can include or inherit from other classes

```
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”
  }
}
```

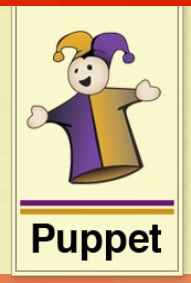
Nodes



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

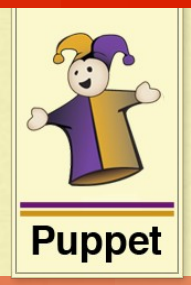
```
node “erwin.wpi.edu” {
    include sudo_class
    include other_class
}
```

External Nodes



- Node definitions can be defined outside of puppet
- Script definition
  - Takes in node name
  - Returns node configuration
- Node configuration in YAML format
  - List of node specific variables
  - List of classes to include
- Ideal for sites with too many nodes to bother pre-creating

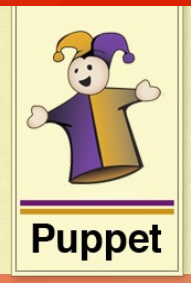
Dependencies



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

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

Subscribes



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

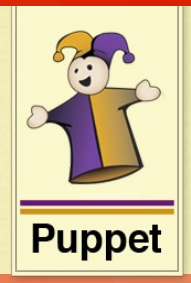
```
file { “/etc/ssh/sshd_conf”:
    ...
    notify => Service[“sshd”]
}

service { “sshd”:
    subscribe => File[“/etc/ssh/sshd_conf”
}
```

Notifications

- Results can be sent to different log sources
  - syslog, email, YAML dump, graphs
- Custom report methods are possible
- Simplest is email

```
From: report@puppethost
Subject: Puppet Report for clienthost


Tue Nov 13 16:52:42 -0500
  //clienthost/server_base/File[/etc/syslog.conf]/ensur
  e (notice): created
```

Templates



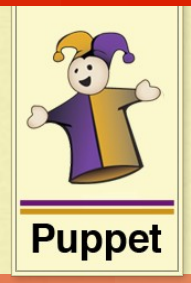
- Apply code and variable substitution
- Can be applied to files or strings
- Uses ERB, same templating engine as the famous Ruby on Rails

```
file { “/etc/hosts”:
    content => template(“127.0.0.1  $hostname”)
    ...
}
```

... might produce:

```
127.0.0.1    erwin
```

Facter



- Where all of those cool variables you use in templates come from
- Default fact library includes OS type, version, arch, IP...

```
$ facter
kernel => Linux
kernelrelease => 2.6.22.9-91.fc7
lsbdistcodename => Moonshine
lsbdistdescription => Fedora release 7 (Moonshine)
...
```

Using Facts

- Custom facts can be added to systems
- Conditional behavior based on facts

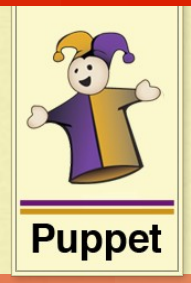
```
service { ntpd:
    ...
    enable => $operatingsystem ? {
        fedora  => true,
        default => false
    }
}
```

Central Facts Repository

- Server can store client facts in SQL database
- Great for running custom reports

```
+------------------+----------+
| name             | value    |
+------------------+----------+
| londo.wpi.edu    | 71KHT71  |
| noc1.wpi.edu     | F3K8M51  |
| avocent.wpi.edu  | 3KTD351  |
| delenn.wpi.edu   | JDYBSC1  |
| gkar.wpi.edu     | DV6KT71  |
+------------------+----------+
```

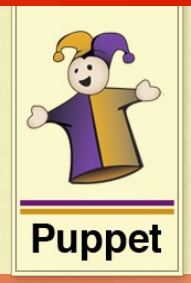
Defines



- Create simple functions that can be treated like classes
- Can instantiate multiple type definitions

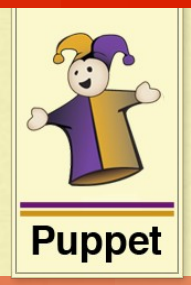
```
define svn_repo($path) {
    file { “$path”:
        ensure => directory
    }
    exec { "/usr/bin/svnadmin create $path/$title":
        unless  => "/bin/test -d $path",
        require => File[$path]
    }
}
svn_repo { puppet: path => "/var/svn" }
```

Writing Custom Types



- Programmed in Ruby
- Simple ones are simple, harder ones are possible
- Types are split into two components
  - Type: defines type interface
  - Provider: a specific implementation of backend
- Types can have multiple providers
  - package: apt, yum, emerge...
  - user: useradd, pw, netinfo
- Puppet fileserver can be used to push new code out to clients

Simple Sysctl Type

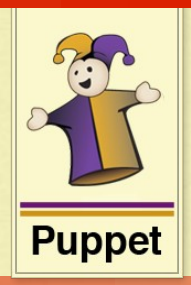


- /etc/sysctl.conf is a simple “key = val” format file containing multiple kernel parameter tweaks
- Example line:

```
net.ipv4.ip_forward = 0
```

- Sample code only edits file
- Uses ParsedFile helper library

Type Code




```
module Puppet
        newtype(:sysctl) do
                ensurable
                newparam(:name, :namevar => true) do
                end

                newproperty(:val) do
                end

                newproperty(:target) do
                end
        end
end
```

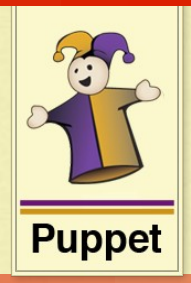
Provider Code


```
require 'puppet/provider/parsedfile'
conffile = "/etc/sysctl.conf"

Puppet::Type.type(:sysctl).provide(:parsed,
            :parent => Puppet::Provider::ParsedFile,
            :default_target => conffile,
            :filetype => :flat) do

        text_line :comment, :match => /^#/;
        text_line :blank,   :match => /^\s*$/;
        record_line :parsed, :fields => %w{name val},
            :joiner => ' = ', :separator => /\s*=\s*/;
end
```

Sysctl Usage


```
sysctl { “net.ipv4.ip_local_port_range”:
   val = “50000 65535”
}

sysctl { “net.ipv4.ip_forward”:
   val = “0”
}
```

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