Less talk, more rock

Puppet

ALM Connect 2013
deepak

giridharagopal

deeak@puppetlabs.com
@grim_radical [github twitter freenode]
Let’s talk about...
Sorry NO CHANGE
Immutability is great!
Classes should be immutable unless there's a very good reason to make them mutable....If a class cannot be made immutable, limit its mutability as much as possible.
Immutability allows for invariants, which help you reason about correctness.
Immutability prevents spooky action at a distance
Immutability fosters modular, composable abstractions
(this shouldn’t be a tough sell to developers)
That’s great for development, but how about operations?
Immutability for infrastructure?

Because operations is in the same boat as development
Everyone who’s got their app running on a fleet of servers has experienced spooky action at a distance.
Known, good state is critical for reliable upgrades
A lack of predictability in your systems ruins automation and abstraction
The problem is that:

Systems are inherently mutable!

But ideally:

Systems should behave as though they weren’t!
façade of immutability
Computer systems are in many ways open systems, providing the keys to the vault if one is so inclined to grab them. But in order to foster an air of immutability in our own systems, it's of utmost importance to create a façade of immutability. Immutability requires that we layer over and abstract the parts of our system that provide unrestrained mutability.
Describe how you’d like your systems to look, and *Puppet* does all the hard work for you!
Example: Inquire a simple service for exposing system metrics
Example: Inquire

http://box/inquire/disk_usage
install apache
create apache user
create apache config file
cgi script for "df -h"
correct perms for script
start apache
restart if config changes!
Once you’ve got a spec for your service, you can see if a given machine is up to code
• **Packages:**
  • apache
• **Users:**
  • apache
• **Config files:**
  • apache’s httpd.conf
• **Services:**
  • apache should be running
  • restart if config file changes
• **Data files:**
  • /var/www/cgi-bin/disk_usage
  • executable, owned by apache
  • does “df -h”
class inquire_server {
    package { apache: ensure => installed }
    user { apache: uid => 1000, shell => "/bin/false" }
    service { apache: ensure => running }

    file {
        "/etc/httpd/httpd.conf":
            owner => root,
            mode => 644,
            source => "puppet://master-server/httpd.conf",
            notify => Service[apache];

        "/var/www/cgi-bin/disk_usage.sh":
            owner => apache,
            mode => 755,
            content => "/usr/bin/df -h";
    }
}
class inquire_bootstrap {
    package {  apache: ensure => installed }  
    user    {  apache: uid => 1000, shell => "'/bin/false'" }  
    service {  apache: ensure => running }  

    file {
        "'/etc/httpd/httpd.conf'":  
            owner => root,  
            mode => 644,  
            source => "puppet://master-server/httpd.conf",  
            notify => Service[apache];  
    }
}

class inquire_disk_usage {
    include inquire_bootstrap

    file {
        "'/var/www/cgi-bin/disk_usage.sh'":  
            owner => apache,  
            mode => 755,  
            content => "'/usr/bin/df -h'";
    }
}
class inquire_bootstrap {
    package { apache: ensure => installed }
    user    { apache: uid => 1000, shell => "/bin/false" }
    service { apache: ensure => running }

    file {
        "/etc/httpd/httpd.conf":
        owner => root,
        mode => 644,
        source => "puppet://master-server/httpd.conf",
        notify => Service[apache];
    }
}

define inquiry($command) {
    include inquire_bootstrap
    file {
        "/var/www/cgi-bin/$name.sh":
        owner => apache,
        mode => 755,
        content => $command;
    }
}
node "appserver.mydomain.com" {
  inquiry {
    "disk-usage": command => "df -h";
    "processes": command => "ps aux";
    "kernel-info": command => "uname -a";
  }
}
node "appserver1.mydomain.com" {
  inquiry {
    "disk-usage": command => "df -h";
    "processes": command => "ps aux";
    "kernel-info": command => "uname -a";
  }
}

node "appserver2.mydomain.com" {
  inquiry {
    "disk-usage": command => "df -h";
    "processes": command => "ps aux";
    "kernel-info": command => "uname -a";
  }
}
class instrumentation {
    inquiry {
        "disk-usage": command => "df -h";
        "processes": command => "ps aux";
        "kernel-info": command => "uname -a";
    }
}

node "appserver1.mydomain.com" {
    include instrumentation
}

node "appserver2.mydomain.com" {
    include instrumentation
}
Rich set of primitives, and make your own. Can use existing modules and abstractions, and make your own.
1. **Facts**
   The node sends normalized data about itself to the Puppet Master.

2. **Catalog**
   Puppet uses the Facts to compile a Catalog that specifies how the node should be configured.

3. **Report**
   The node reports back to Puppet indicating the configuration is complete, which is visible in the Puppet Dashboard.

4. **Report Collector**
   (Puppet or 3rd party tool)
   Puppet's open API can also send data to third party tools.

**SSL secure encryption on all data transport**
1. **Facts**
The node sends normalized data about itself to the Puppet Master.

2. **Catalog**
Puppet uses the Facts to compile a Catalog that specifies how the node should be configured.

3. **Report**
The node reports back to Puppet indicating the configuration is complete, which is visible in the Puppet Dashboard.

4. **Report Collector**
(Puppet or 3rd party tool)
Puppet’s open API can also send data to third party tools.
Node

1. **Facts**
The node sends normalized data about itself to the Puppet Master.

2. **Catalog**
Puppet uses the Facts to compile a Catalog that specifies how the node should be configured.

3. **Report**
The node reports back to Puppet indicating the configuration is complete, which is visible in the Puppet Dashboard.

4. **Report Collector**
(Puppet or 3rd party tool)

- **SSL secure encryption on all data transport**
- **Report**
Puppet's open API can also send data to third party tools.
1. **Facts**
   The node sends normalized data about itself to the Puppet Master.

2. **Catalog**
   Puppet uses the Facts to compile a Catalog that specifies how the node should be configured.

3. **Report**
   The node reports back to Puppet indicating the configuration is complete, which is visible in the Puppet Dashboard.

4. **Report Collector**
   (Puppet or 3rd party tool)
   - Puppet’s open API can also send data to third party tools.

**SSL secure encryption on all data transport**
netmask_lo: 255.0.0.0
augeasversion: 0.10.0
fqdn: pe-debian6.localdomain
manufacturer: "VMware, Inc."
processorcount: "1"
productname: VMware Virtual
Platform
physicalprocessorcount: 1
facterversion: 1.6.7
boardproductname: 440BX Desktop
Reference Platform
kernelmajversion: "2.6"
hardwareisa: unknown
timezone: PDT
puppetversion: 2.7.12 (Puppet Enterprise 2.5.1)
lsbdistcodename: squeeze
is_virtual: "true"
operatingsystemrelease: 6.0.2
virtual: vmware
type: Other
domain: localdomain
hostname: pe-debian6
selinux: "false"
kernel: Linux
kernelrelease: 2.6.32-5-686
ipaddress: 172.16.245.128
processor0: Intel(R) Core(TM) i7-2635QM CPU @ 2.00GHz
lsbdistrelease: 6.0.2
uniqueid: 007f0101
hardwaremodel: i686
kernelversion: 2.6.32
operatingsystem: Debian
architecture: i386
lsbdistdescription: Debian GNU/Linux 6.0.2 (squeeze)
lsbmajdistrelease: "6"
interfaces: "eth0,lo"
ipaddress_lo: 127.0.0.1
uptime_days: 0
lsbdistid: Debian
rubysitedir: /opt/puppet/lib/site_ruby/1.8
rubyversion: 1.8.7
osfamily: Debian
memorytotal: &id001 502.57 MB
memorysize: *id001
boardmanufacturer: Intel
Corporation
path: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/usr/games:/bin
1. **Facts**
   The node sends normalized data about itself to the Puppet Master.

2. **SSL secure encryption on all data transport**

3. **Report**
   The node reports back to Puppet indicating the configuration is complete, which is visible in the Puppet Dashboard.

4. **Catalog**
   Puppet uses the Facts to compile a Catalog that specifies how the node should be configured.

5. **Report Collector**
   (Puppet or 3rd party tool)

   - **Report**
     Puppet’s open API can also send data to third party tools.
**Node**

1. **Facts**
   - The node sends normalized data about itself to the Puppet Master.

2. **Catalog**
   - Puppet uses the Facts to compile a Catalog that specifies how the node should be configured.

3. **Report**
   - The node reports back to Puppet indicating the configuration is complete, which is visible in the Puppet Dashboard.

4. **Report Collector**
   - (Puppet or 3rd party tool)
     - Report
       - Puppet’s open API can also send data to third party tools.
file { "/etc/issue":
    content => "Got an issue? Here’s a tissue!",
}

file { "/etc/motd":
    content => template("Welcome to $hostname!"),
}
file { "/etc/sudoers":
  owner => root,
  group => root,
  mode  => 440,
  source => "puppet:///modules/sudo/sudoers"
}

class ntp {
  package { 'ntp':
    ensure => installed,
  }

  service { 'ntpd':
    ensure => running,
    enable => true,
    subscribe => File['/etc/ntp.conf'],
  }

  file { '/etc/ntp.conf':
    ensure => file,
    require => Package['ntp'],
    source => "puppet:///modules/ntp/ntp.conf",
  }
}

node "webservcer.mydomain.com" {
    include ntp
}

node "appserver.mydomain.com" {
    include ntp
}

node "database.mydomain.com" {
    include ntp
}
class ssh {

    @@sshkey { $hostname:
        type => dsa,
        key => $sshdsakey
    }

    Sshkey <<| |>>

}
1. **Facts**
   - The node sends normalized data about itself to the Puppet Master.

2. **SSL secure encryption on all data transport**

3. **Catalog**
   - Puppet uses the Facts to compile a Catalog that specifies how the node should be configured.

4. **Report Collector**
   - (Puppet or 3rd party tool)

   **Report**
   - Puppet’s open API can also send data to third party tools.

   **Report**
   - The node reports back to Puppet indicating the configuration is complete, which is visible in the Puppet Dashboard.
1. **Facts**: The node sends normalized data about itself to the Puppet Master.

2. **Catalog**: Puppet uses the Facts to compile a Catalog that specifies how the node should be configured.

3. **Report**: The node reports back to Puppet indicating the configuration is complete, which is visible in the Puppet Dashboard.

4. **Report Collector** (Puppet or 3rd party tool):
   - **Report**: Puppet’s open API can also send data to third party tools.

---

SSL secure encryption on all data transport.
File "/tmp/foo/bar"
User "deepak"
Dir "/tmp/foo"
Dir "/tmp"
Dir “/tmp”

User “deepak”

Dir “/tmp/foo”

File “/tmp/foo/bar”
1. **Facts**
The node sends normalized data about itself to the Puppet Master.

2. **Catalog**
Puppet uses the Facts to compile a Catalog that specifies how the node should be configured.

3. **Report**
The node reports back to Puppet indicating the configuration is complete, which is visible in the Puppet Dashboard.

4. **Report Collector**
(Puppet or 3rd party tool)

   - **Report**
Puppet’s open API can also send data to third party tools.

SSL secure encryption on all data transport
1. **Facts**
The node sends normalized data about itself to the Puppet Master.

2. **Catalog**
Puppet uses the Facts to compile a Catalog that specifies how the node should be configured.

3. **Report**
The node reports back to Puppet indicating the configuration is complete, which is visible in the Puppet Dashboard.

4. **Report**
Puppet’s open API can also send data to third party tools.
package {  'ntp':
    ensure => installed,
}

service {  'ntpd':
    ensure => running,
    enable => true,
    subscribe => File['/etc/ntp.conf'],
}

file {  '/etc/ntp.conf':
    ensure => file,
    require => Package['ntp'],
    source => "puppet:///modules/ntp/ntp.conf",
}
package { 'ntp':
    ensure => installed,
}

service { 'ntpd':
    ensure => running,
    enable => true,
    subscribe => File['/etc/ntp.conf'],
}

file { '/etc/ntp.conf':
    ensure => file,
    require => Package['ntp'],
    source => "puppet:///modules/ntp/ntp.conf",
}
package { 'ntp':
    ensure => installed,
}

service { 'ntpd':
    ensure   => running,
    enable   => true,
    subscribe => File['/etc/ntp.conf'],
}

file { '/etc/ntp.conf':
    ensure   => file,
    require  => Package['ntp'],
    source   => "puppet:///modules/ntp/ntp.conf",
}
package {'ntp':
   ensure => installed,
}

service {'ntpd':
   ensure => running,
   enable => true,
   subscribe => File[('/')[- etc/ntp.conf'],
}

file {'/etc/ntp.conf':
   ensure => file,
   require => Package['ntp'],
   source => "puppet://modules/ntp/ntp.conf",
}
1. **Facts**
The node sends normalized data about itself to the Puppet Master.

2. **Catalog**
Puppet uses the Facts to compile a Catalog that specifies how the node should be configured.

3. **Report**
The node reports back to Puppet indicating the configuration is complete, which is visible in the Puppet Dashboard.

4. **Report Collector**
(Puppet or 3rd party tool)

- **SSL secure encryption on all data transport**
- **Report**
Puppet’s open API can also send data to third party tools.
1. **Facts**
   - The node sends normalized data about itself to the Puppet Master.

2. **Catalog**
   - Puppet uses the Facts to compile a Catalog that specifies how the node should be configured.

3. **Report**
   - The node reports back to Puppet indicating the configuration is complete, which is visible in the Puppet Dashboard.

4. **Report Collector**
   - (Puppet or 3rd party tool)
   - Report
     Puppet’s open API can also send data to third party tools.

**SSL secure encryption on all data transport**
1. **Facts**
The node sends normalized data about itself to the Puppet Master.

2. **Catalog**
Puppet uses the Facts to compile a Catalog that specifies how the node should be configured.

3. **Report**
The node reports back to Puppet indicating the configuration is complete, which is visible in the Puppet Dashboard.

4. **Report**
Puppet’s open API can also send data to third party tools.
Idempotent, and only does what’s necessary
Compensates for the inherent mutability of systems
Combats spooky action at a distance with automatic repair
Brings **predictability** to your systems
A foundation of predictability and reliability lets you perform higher-level operations on your infrastructure.
Code all the way down
Software-defined infrastructure is...just software.
Infrastructure as code is...just code.
Thus, you can treat it like the other code in your application.
Scary, but liberating!
Maintaining the state of your systems is the **foundation** upon which everything rests.
Start small, and start somewhere. :)
We’re hiring!