

Configuration Management with Puppet

Introduction



What is Puppet



- is a configuration management system
- is a programming language
- offers a Client/Server architecture
- has a huge community
- widely used in the IT industry
- commercial support available if needed



What else is needed

- central software repositories
 - yum
 - apt
- Provisioning system
 - kickstart
 - preseed
- Version control
 - GIT
 - subversion

Contents



- Resources
- Manifests
- Ordering
- Variables, Conditionals, and Facts
- Classes
- Module

Resources



All elements of a node will be described as resources

- Files
- User
- Services
- ... (about 50)

Resources



- Abstraction layer to access the resources (**RAL**)
- Resources has attributes
 - a File has a Path.
- The RAL gives you os independence
 - BUT!! not all resources are available on every platform
- Resources well documented



Example Resource

```
user { 'dave':  
    ensure      => present,  
    uid         => '507',  
    gid         => 'admin',  
    shell       => '/bin/zsh',  
    home        => '/home/dave',  
    managehome => true,  
}
```

Ressource shell



You can interact with the RAL directly.

- `puppet resource user root`
- `puppet resource user dave \`
`ensure=present shell="/bin/zsh" \`
`home="/home/dave" managehome=true`

Resource Documentation



```
$ puppet describe -s user
```

```
user
```

```
====
```

Manage users. This type is mostly built to manage system users, so it is lacking some features useful for managing normal users.

This resource type uses the prescribed native tools for creating groups and generally uses POSIX APIs for retrieving information about them. It does not directly modify `/etc/passwd` or anything.

```
Parameters
```

```
-----
```

```
allowdupe, auth_membership, auths, comment, ensure, expiry, gid, groups,
home, key_membership, keys, managehome, membership, name, password,
password_max_age, password_min_age, profile_membership, profiles,
project, role_membership, roles, shell, uid
```

```
Providers
```

```
-----
```

```
directoryservice, hpuxuseradd, ldap, pw, user_role_add, useradd
```

Resource basic



- file vs. augeas
- yumrepo stages
- package ensure latest ?
- exec only if needed

Manifests

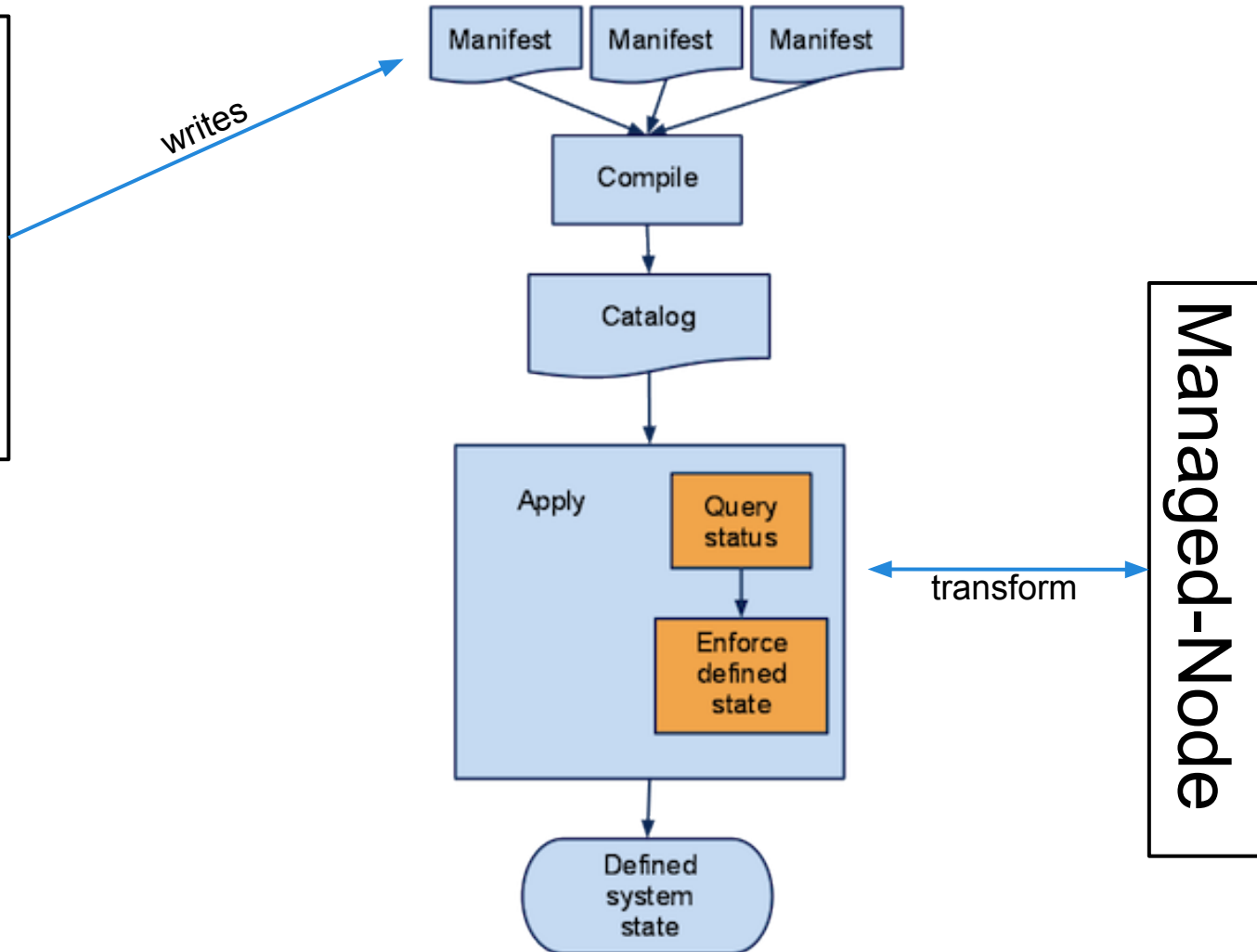


- *manifests* are puppet programs
- puppet programs
 - declare resources
 - define conditions
 - group resources
 - generate text
 - link other manifests
 - define ordering

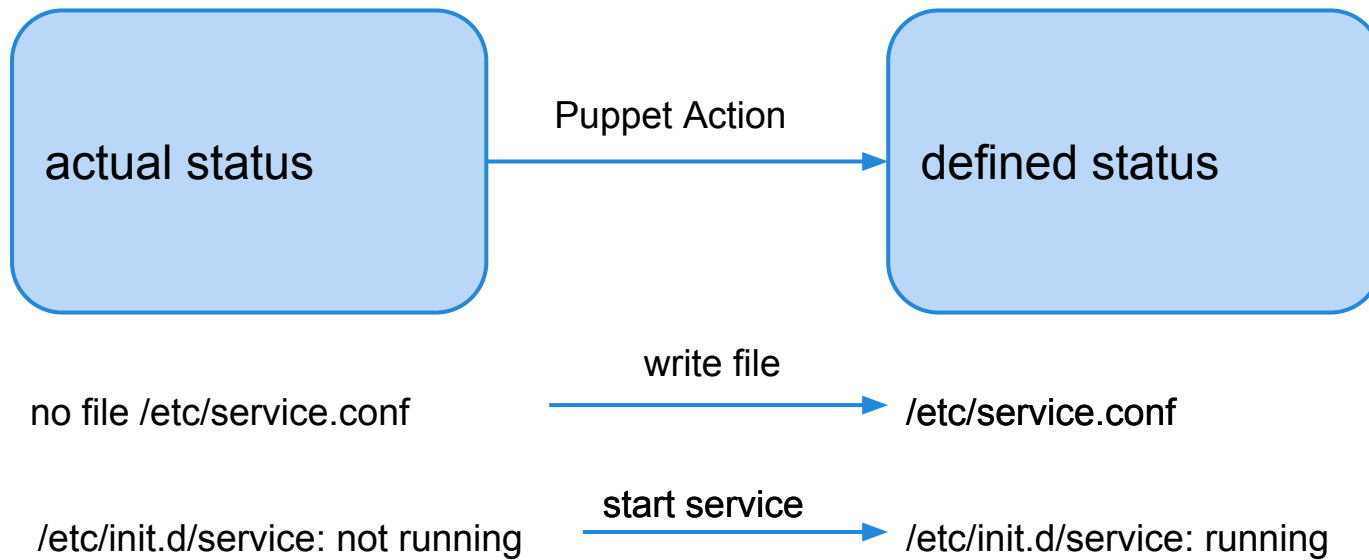
Manifests Compile



Puppet-Dev



Manifests Run





Manifests example!

```
file {'/tmp/test1':  
  ensure => present,  
  content => "Hi.",  
}  
  
file {'/tmp/test2':  
  ensure => directory,  
  mode   => 0644,  
}  
  
file {'/tmp/test3':  
  ensure => link,  
  target => '/tmp/test1',  
}  
  
notify {"I'm notifying you." :}  
notify {"So am I!" :}
```



Ordering example!

```
# /root/learning-manifests/break_ssh.pp, again
file { '/etc/ssh/sshd_config':
  ensure => file,
  mode   => 600,
  source => '/root/learning-manifests/sshd_config',
}

service { 'sshd':
  ensure    => running,
  enable    => true,
  subscribe => File['/etc/ssh/sshd_config'],
}
```



Facts example!

```
host {'self':
  ensure      => present,
  name        => $fqdn,
  host_aliases => ['puppet', $hostname],
  ip          => $ipaddress,
}

file {'motd':
  ensure => file,
  path   => '/etc/motd',
  mode   => 0644,
  content => "Welcome to ${hostname},\na ${operatingsystem}
island in the sea of ${domain}.\n",
}
```




Conditionals a first example!

```
if $is_virtual == 'true' {
  service {'ntp':
    ensure => stopped,
    enable => false,
  }
}
else {
  service { 'ntp':
    name      => 'ntp',
    ensure    => running,
    enable    => true,
    hasrestart => true,
    require  => Package['ntp'],
  }
}
```



Structure of Configuration

Site

Node dbserv02.desy.de:

Node webserv01.desy.de:

Module: SSH

ssh

ssh::install

ssh::config

ssh::params

ssh::service

Module: apache

apache

apache::install

apache::config

apache::params

apache::service

Structure of a Module



Module: foo

manifests

files

templates

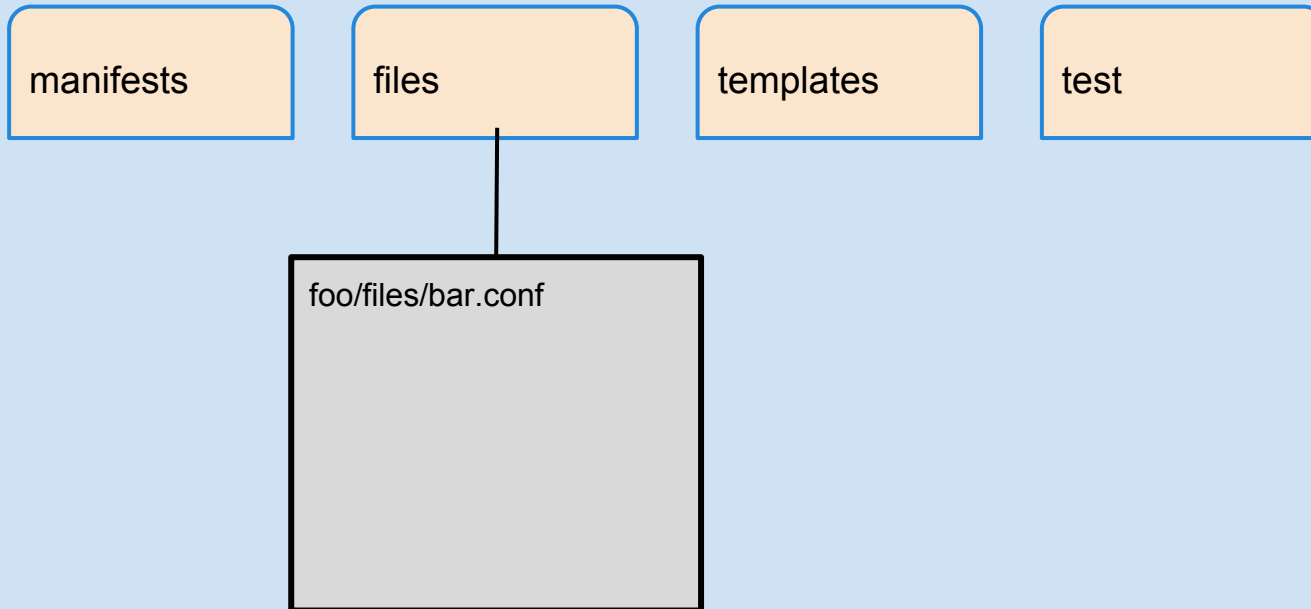
test

foo/manifests/init.pp
foo/manifests/params.pp
foo/manifests/config.pp
foo/manifests/install.pp
foo/manifests/service.pp

Structure of a Module



Module: foo



Structure of a Module



Module: foo

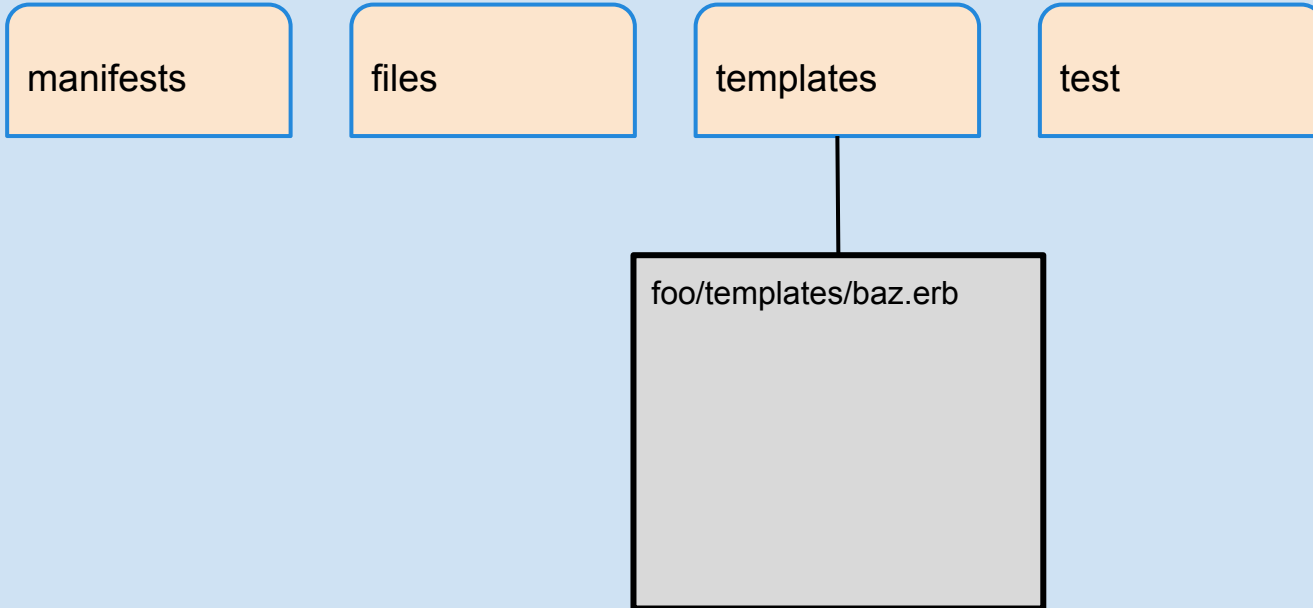
manifests

files

templates

test

foo/templates/baz.erb



Class a first example!



/ntp/manifests/init.pp

```
class ntp inherits ntp::params{  
    include ntp::install  
    include ntp::service  
}
```

Class a first example!



/ntp/manifests/params.pp

```
class ntp::params{
    $service_name = 'ntpd'
    $conf_file    = 'ntp.conf.el'
}
```

Class a first example!



/ntp/manifests/install.pp

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


Class a first example!



/ntp/manifests/service.pp

```
class ntp::service {
  service { 'ntp':
    name      => $service_name,
    ensure    => running,
    enable    => true,
    subscribe=> File['ntp.conf'],
  }
}
```

Puppet process - Step 1

registration

Example: foreman, satellite, spacewalk ...

Creates: Kickstart File and Puppet Node definition

Puppet process - Step 2

provisioning

Tool: Redhat anaconda, Ubuntu Fai, cobbler, preseed

Input:

anaconda configured by kickstart file

preseed config file

...

Result: Installing minimal linux and puppet, and start

Puppet after reboot

Puppet process - Step 3

configuration


Tool: puppet

Input:

node definition

Result: puppet defined system state

Organization of the modules



```
Node {  
  include  
  desktop}  
}
```

modules / features	ntp	afs	apache
it_desktop	x	x	
xfel_webserver	x		x

It's the end my friend!

