Puppet Introduction

Systems Administration

Otago Polytechnic
Dunedin, New Zealand

The problem

- Configuring systems one at a time is too slow.
- One at a time configuration can lead to inconsistencies.
- Information about how your systems are configured winds up scattered across your network.

The solution: configuration management systems

We'll store all of the configuration information on a central server that will push configurations out to client machines. This will

- Get all of our configuration information in one place.
- Ensure that configuration is consistently and promptly applied to all systems.
- Save time!

Puppet

In this paper we will use *Puppet* for configuration management.

- It is mature and powerful.
- It is widely used.
- It is reasonably cross-platform.

Puppet overview

Our `mgmt` servers will manage Puppet for us. In Puppet terms, these servers will be *puppetmasters*.

The client machines (including the `mgmt` servers) will be *agents*. They periodically contact the puppetmaster to get new configuration information.

Some Key Terms

Manifest Any bit of Puppet code stored in a file that ends with the .pp extension. These sit on the puppetmaster.

Node A collection of resources in a manifest that will be applied to a particular agent.

Catalog The puppetmaster reads the manifests and compiles a catalog for each host. A catalog is a set of resources to be used on an agent system.

Resource A unit of puppet configuration. A resources has a *type*, a *title*, and one or more *attributes*.

Some Types

Puppet supports many standard types, and it is possible to define your own. Some important types include:

- Package
- File
- Service
- User
- Group
- Exec
- Cron

Nodes

A node is basically a host you want to configure.

```
node 'www.sqrawler.com' {

}

node 'db1.sqrawler.com', 'db2.sqrawler.com' {

}
```

The Default Node

If you specify a *default* node, its configuration will be applied to any node that does not have a specific node definition.

```
node default {

}
```

Node Inheritance

A node can inherit from another node. For example 'www.sqrawler.com' gets all of the configuration from the base node, plus it's own specific configuration.

```
node base {

}

node 'www.sqrawler.com'  inherits base {

}
```

Note that Puppet best practice is to be careful to avoid overusing inheritance. If you find yourself going back and modifying base classes or adding and removing inheritance relationships often, that is a sign that you shouldn't be using inheritance.

Modules

A collection of related resources can be organised into a *module*. For example, we may want to install the bacula-fd package, its configuration file, and set up a working directory for restores. We can create a module that incorporates all of these things, and then use the module in a node:

```
node 'foo.sqrawler.com' {
  include bacula-client
}
```

We will create and apply a module in today's lab.