Puppet Declarative config management

CONCEPTS

Puppet is declarative: you describe the end state and it only changes what differs, so runs are safe to repeat.

Always --noop first: it prints what would change without touching the system.

APPLY AND AGENT

`puppet apply` *`<file.pp>`*
Apply a manifest locally

`puppet apply --noop` *`<file.pp>`*
Dry run, show changes only

`puppet apply -e '`*`<code>`***`'`**
Run a one-line snippet

`puppet apply --detailed-exitcodes` *`<file.pp>`*
Exit code reports changes

`puppet agent -t`
One foreground run, verbose

`puppet agent -t --noop`
Preview what the server would change

INSPECT AND CHANGE

`puppet resource` *`<type>`*
List all resources of a type

`puppet resource` *`<type> <name>`*
Show one resource as code

`puppet describe` *`<type>`*
Docs for a resource type

`puppet help` *`<subcommand>`*
Help for a subcommand

`puppet --version`
Print Puppet version

FACTS AND CONFIG

`facter`
Print all facts

`facter` *`<fact>`*
One fact, e.g. os.family

FACTS AND CONFIG (cont.)

`puppet config print` *`<setting>`*
One setting, e.g. modulepath

HIERA AND LOOKUP

`puppet lookup` *`<key>`*
Look up a data value

`puppet lookup` *`<key>`* **`--explain`**
Show where the value came from

MODULES

`puppet module install` *`<author-name>`*
Install from the Forge

`puppet module list`
Installed modules

CHECK AND LINT

`puppet parser validate` *`<file.pp>`*
Syntax check only

`puppet-lint` *`<file.pp>`*
Style problems

MANIFEST SYNTAX

Resource

```
package { 'nginx':
  ensure => installed,
}
```

Service and file

```
service { 'nginx':
  ensure => running,
  enable => true,
}
file { '/etc/motd':
  ensure  => file,
  content => "Managed by Puppet\n",
}
```

Class

```
class web (
  String $port = '80',
) {
  package { 'nginx': ensure =>
installed }
}
include web
```