PARAMETERS

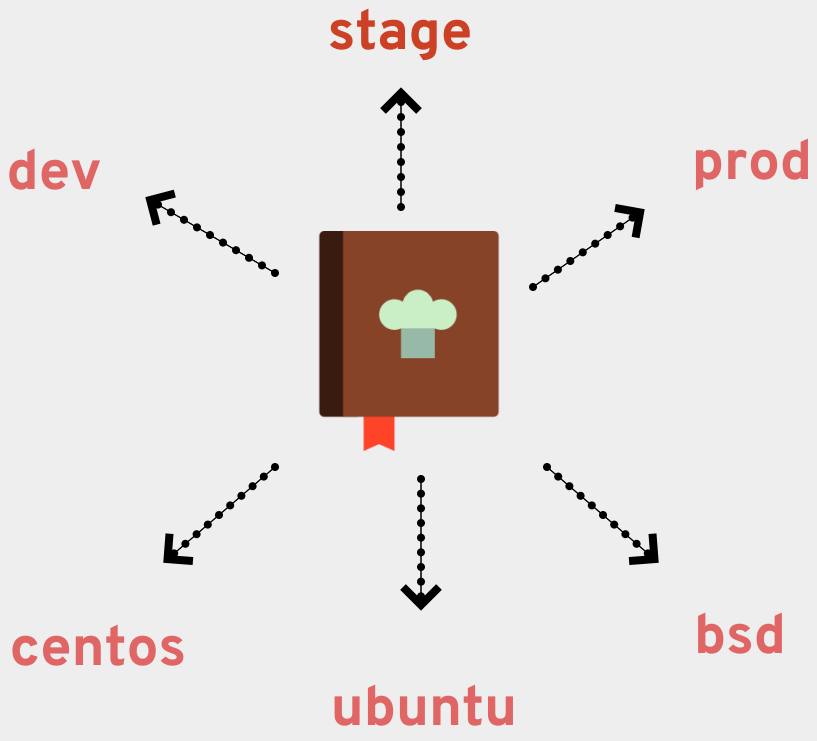
MANAGING PROPERTIES

TOPICS

- Code vs data
- Parameters
- Scoping
- Inheritance
- Facts

CODE vs DATA

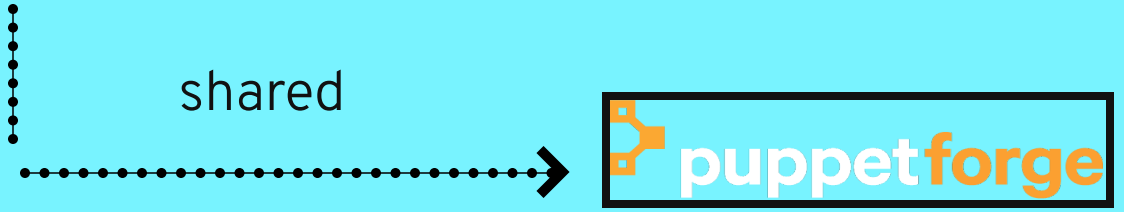
- one module

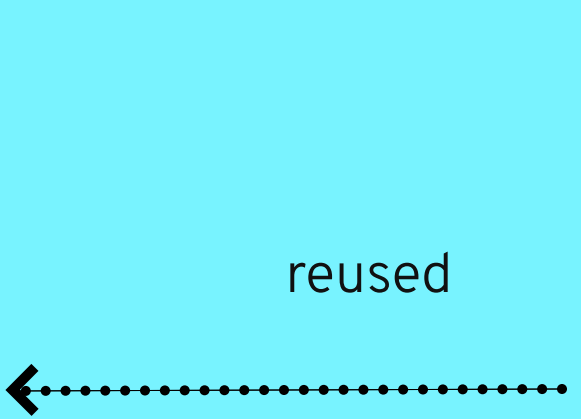


- different envionments
- different platforms

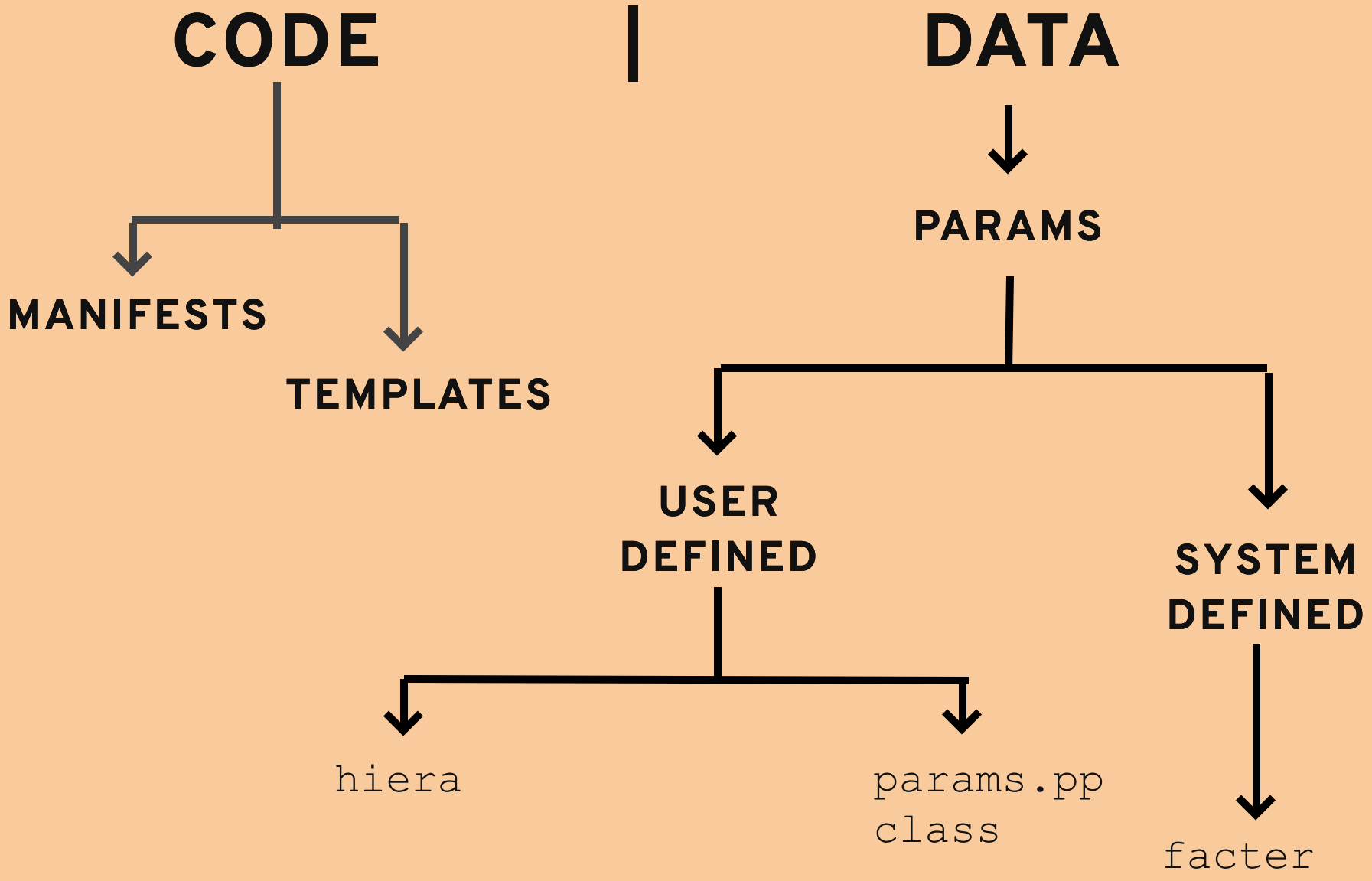
MODULES WITHOUT DATA

- Creates generic modules which can be



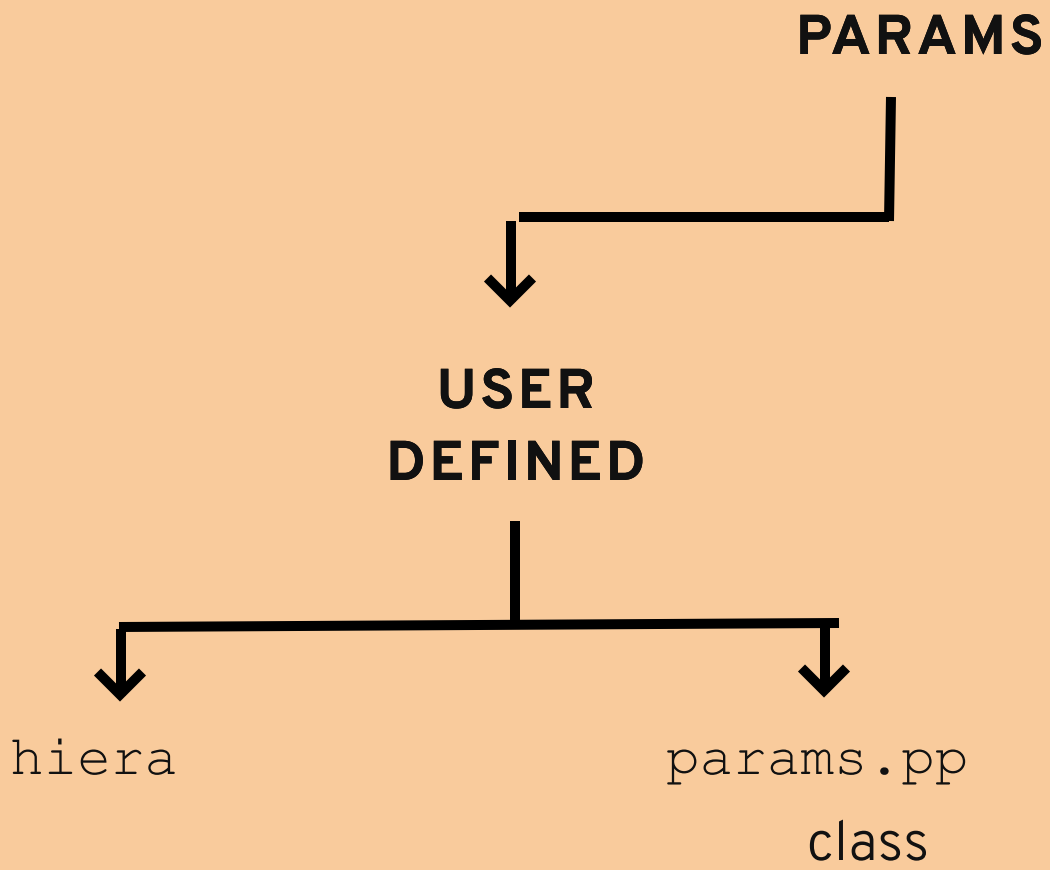


- save tonnes of time


CODE
DATA
MANIFESTS
TEMPLATES
PARAMS
USER DEFINED
SYSTEM DEFINED
hiera
params.pp
class
facter

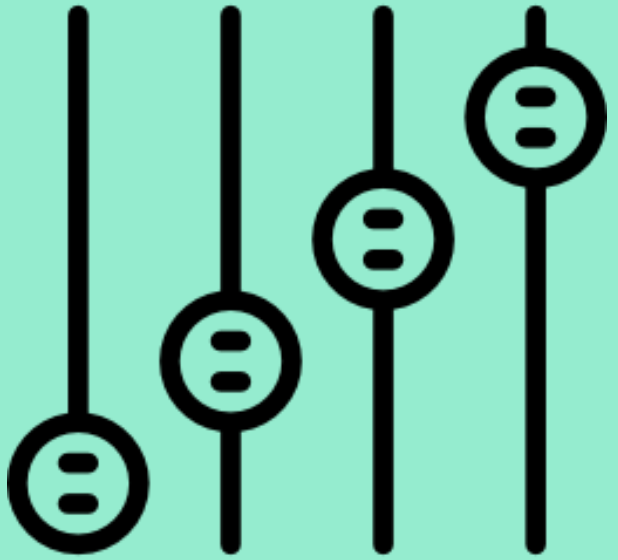
PARAMS

USER DEFINED



STORY OF DEFAULTS

- Modules define the default properties
- Defaults are overridden from outside the modules
- Overriding of properties happens either from node definitions, or external data sources
- This makes modules generic and sharable



DEFAULTS INSIDE MODULES

PARAMS.PP

- old way
- design pattern, mixed with existing classes
- can get complex with increasing properties and platform specific logic

HIERA

- new way, with puppet 4
- dedicated place to store data
- well though after solution, can scale, simplified data management

params.pp

the story of sane defaults

params.pp

Sane Defaults

- **dedicated class** to define data/properties/params
- can be also used to write logic/conditionals to separate **system specific** properties

GROUP LAB

DEFINING PARAMS WITH PARAMS.PP

```
puppet module generate prefs --skip-interview user-prefs
```

prefs::params

```
class prefs::params {

  $color = 'white'
  $car   = 'ford'

}
```

- lets create a params class for prefs module
- define two params $color and $car

GROUP LAB

WHERE TO USE PARAMS

- resource names
- parameter value
- template variable

```
file { $::tomcat::config_path:
    owner     => $::tomcat::user,
    group     => $::tomcat::group,
    mode      => '0644',
  }
```

```
<%= @max_connections %>
```

GROUP LAB

- lets write a class to consume the params defined earlier

class prefs::config

```
class prefs::config {

   notify{"print the prefs":
     message => "

        FAVOURITE COLOR :  ${color}
        FAVOURITE CAR   :  ${car}

     "
   }

}
```

GROUP LAB

- to apply this class on a node, it needs to be called from the init.pp of prefs module

class prefs

```
class prefs {

  include prefs::config

}
```

- prefs class has to be called from the node definition in order for it to be applied

app.pp

```
node node1 {

  include tomcat
  include prefs

}
```

INHERITANCE

GRANDFATHER

FATHER

SON

PARAMS.PP

INIT.PP

inherits xyz::params

CONFIG.PP

inherits xyz

GROUP LAB

prefs::params

```
class prefs inherits prefs::params{

  include prefs::config

}
```

class prefs

prefs::config

```
class prefs::config inherits prefs{



   notify{"print the scope":
     message => "

        FAVOURITE COLOR :  ${color}
        FAVOURITE CAR   :  ${car}

     "
   }

}
```

QUALIFIED PARAMS

```
$color                  $::tomcat::params::color

                        $::tomcat::color
```

LAB EXERCISE

PARAMETERIZING TOMCAT CONFIGS

GROUP EXERCISE

DEFINE PARAMS

tomcat::params

```
class tomcat::params {


  $user  = 'tomcat'
  $group = 'tomcat'
  $config_path  = '/etc/tomcat/tomcat.conf'
  $packages  = [ 'tomcat', 'tomcat-webapps' ]
  $service_name = 'tomcat'
  $service_state = running

}
```

GROUP EXERCISE

DEFINE ATTRIBUTES

```
cd myapp
chef generate attribute cookbooks/tomcat default
```

file: cookbooks/tomcat/attributes/default.rb

```
default['tomcat']['user'] = 'tomcat'
default['tomcat']['group'] = 'tomcat'
default['tomcat']['config'] = '/etc/tomcat/tomcat.conf'
default['tomcat']['user_config'] = '/etc/tomcat/tomcat-users.
default['tomcat']['packages'] = [ 'tomcat', 'tomcat-webapps'
default['tomcat']['service'] = 'tomcat'
```

class tomcat

```
class tomcat inherits tomcat::params{

  include tomcat::scope
  include tomcat::install
  include tomcat::config
  include tomcat::service
}
```

tomcat::install

```
class tomcat::install inherits tomcat{

    include java

    package { $::tomcat::packages:
      ensure   => installed,
      require  => Package['epel-release']
    }

}
```

tomcat::config

```
class tomcat::config inherits tomcat{

  file { $::tomcat::config_path:
    source     => 'puppet:///modules/tomcat/tomcat.conf',
    owner      => $::tomcat::user,
    group      => $::tomcat::group,
    mode       => '0644',
    notify     => Service['tomcat']
  }

}
```

tomcat::service

```
class tomcat::service inherits tomcat{

   service { $::tomcat::service_name:
     ensure    => $::tomcat::service_state,
     enable    => true,
     require   => Class['tomcat::install'],
   }
}
```

APPLY

GROUP EXERCISE

HANDLE PLATFORM SPECIFIC CONFIGS

“ lets convert base.pp recipe into a module which can be applied to all linux nodes to perform common tasks.....

GROUP EXERCISE

Generate base module

```
cd /workspace/code/environments/production/modules

puppet module generate --skip-interview user-base

mv /workspace/base.pp base/manifests/init.pp
```

GROUP EXERCISE

class base

```
class base{

    user {"deploy" :
      ensure     => present,
      uid        => 5001,
      password   => '$1$WD98.uaZ$cxx30x/K3FXQrljxsvBIu/',
      home       => '/home/deploy'
    }

    user {"dojo" :
      ensure   => absent,
    }

    package { [ "tree", "unzip", "git", "ntp", "wget"   ]:

      ensure   => installed

    }

    service { "ntp":
      ensure   => running,
      enable   => true,

    }
```

GROUP EXERCISE

Add the base class to node definition for node2

```
node 'node2' {

   include base

   $color = 'green'
   include tomcat

}
```

DISCUSSION

Question : What happens when you apply this class on node2 by running puppet agent ??

note: change this

Add default recipe from base to run list

```
knife node run_list add app1 "recipe[base]"
```

Apply

```
sudo chef-client
```

DISCUSSION

Failes because service "ntp" is not available on centos. It was created for ubuntu.

	package	service
ubuntu	`ntp`	`ntp`
centos	`ntp`	`ntpd`

GROUP EXERCISE

create base::params class

file: modules/base/manifests/params.pp

```
class base::params {

  case $::os['family'] {
    'Debian': {
      $ntp_service = 'ntp'
    }
    'RedHat': {
      $ntp_service = 'ntpd'
    }
  }


}
```

EXERCISE

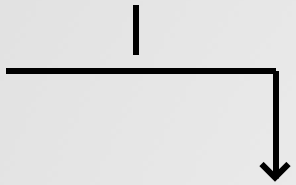
Exercise

Refactor the code to use the parameter just defined in base::params. You will need to,

- inherit base::params in base class (init.pp)
- parameterize the service name

Reference to the solution

ATTRIBUTES

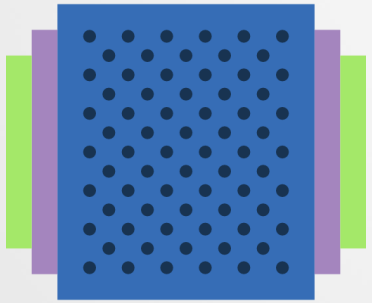


system

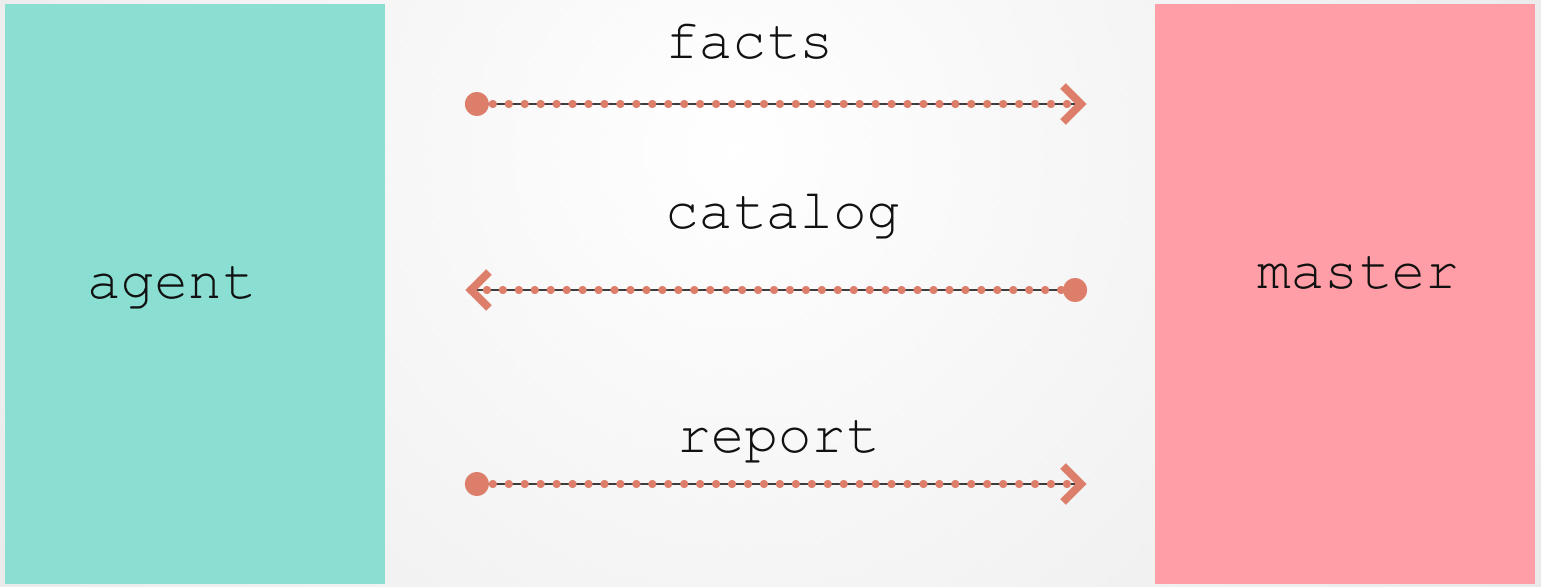
defined

FACTER

- System Profiler
- Installed with puppet
- Provides facts



- Platform
- Network
- Kernel
- Memory
- CPU
- Virtualization
- Disk Mounts
- Other system info


agent
facts
catalog
report
master

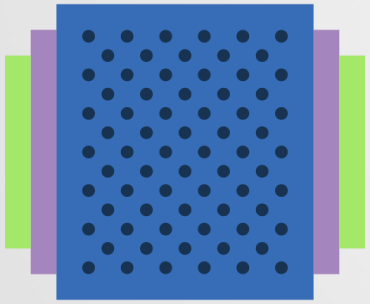
ATTRIBUTES

system

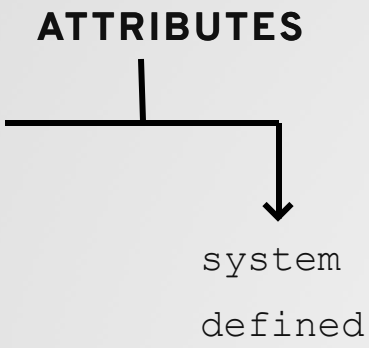
defined

FACTER

Finding facts
about a Node with facter



```
facter
facter ipaddress
facter hostname
facter memory
facter memory.system
facter memory.system.total
facter processors
facter processors.count
```



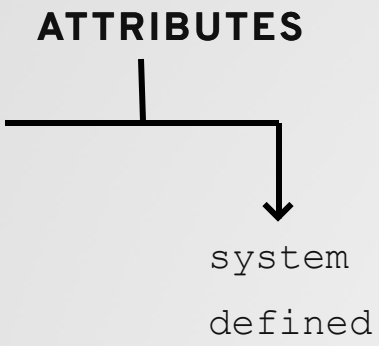
REFERENCING FACTS

```
$hostname

${hostname}

${::hostname}

$facts['hostname']
```



REFERENCING FACTS

```
facter
facter ipaddress
facter hostname
facter memory
facter memory.system
facter memory.system.total
facter processors
facter processors.count
```

```
${::ipaddress}
${::hostname}
${::memory}
${::memory['system']}
${::processors['count']}
```

GROUP EXERCISE

Update base class

file: modules/base/manifests/init.pp

```
file { '/etc/motd':
  ensure   => file,
  owner    => 'root',
  content  => "

     This server is a property of XYZ Inc.

     SYSTEM INFO
     ===========

     Hostname      : ${::fqdn}
     IP Address    : ${::ipaddress}
     Memory        : ${::memory['system']['total']}
     Cores         : ${::processors['count']}
     OS            : ${::os['distro']['description']}


   "

}
```