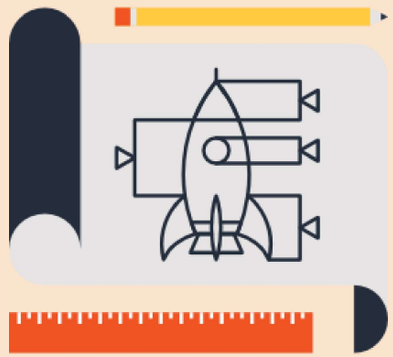
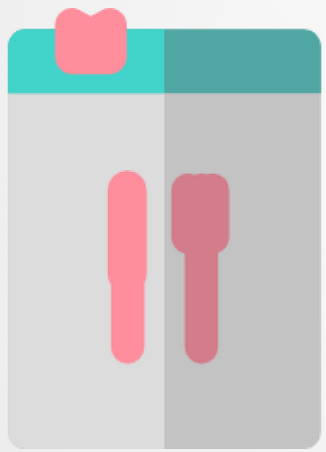


TOPICS

- MODULES
- CLASSES
- ORDERING
- NODE CLASSIFICATION
- NOTIFICATION AND HANDLERS

MODULES



- modules are the packages with manifests and supporting files
- have 1:1 mapping with the applications
- let you create a library of reusable code

ANATOMY

manifests

files

templates

data

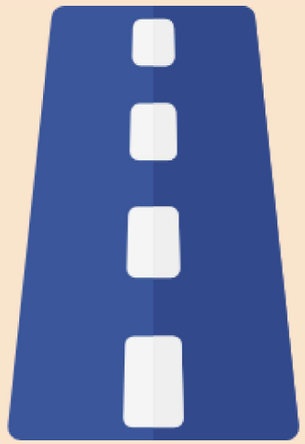
spec

examples

metadata.json

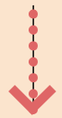
README.MD

MODULEPATH



- Modules are stored on Puppet Master
- Puppet Master has a **code** directory to store modules and rest of the configurations
- The default code dir needs to be changed in case want to point to a custom path.

```
master-code-dir: /etc/puppetlabs/code
```



```
master-code-dir:  /workspace/code
```

Copy existing structure to our workspace

```
cp -r /etc/puppetlabs/code /workspace/code
```

Edit File : /etc/puppetlabs/puppetserver/conf.d/puppetserver.conf

Change from

master-code-dir: /etc/puppetlabs/code

Change to

master-code-dir: /workspace/code

Restart Puppet Master

```
service puppetserver restart
```

GROUP EXERCISE

Problem Statement

You have been asked to deploy a java application server with tomcat. You have been tasked to create automation code with puppet to set it up.

TOMCAT

GROUP EXERCISE

APP SERVER MODULES

Solution:

- We would create a module to setup **tomcat** server and apply it to automate the task
- As a prerequisite, we also need to install **Java**. We will create a module to install openjdk.

GROUP EXERCISE

TASKS

1. Generate modules for java and tomcat
2. Create class to install java and
3. Create a node definition to apply the classes
4. Create classes to install tomcat and start the service, apply
5. Write classes to manage configuration files

AUTO GENERATING MODULES

```
root@puppet:/workspace# puppet help module


USAGE: puppet module <action> [--environment production

ACTIONS:
  build        Build a module release package.
  changes      Show modified files of an installed modul
  generate     Generate boilerplate for a new module.
  install      Install a module from the Puppet Forge or
  list         List installed modules
  search       Search the Puppet Forge for a module.
  uninstall    Uninstall a puppet module.
  upgrade      Upgrade a puppet module.
```

- **puppet module** is a utility which comes with code generator
- it can let you create, search install, upload modules (to and from puppet forge)

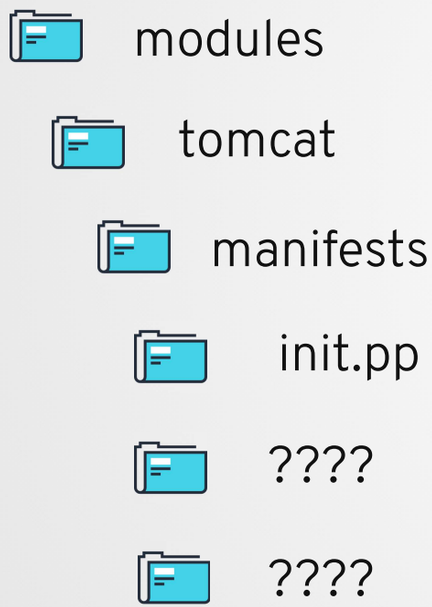
GROUP EXERCISE

GENERATING MODULES SKELETON

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

puppet module generate --skip-interview user-java

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

```
-- java
|   |-- Gemfile
|   |-- README.md
|   |-- Rakefile
|   |-- examples
|   |   `-- init.pp
|   |-- manifests
|   |   `-- init.pp
|   |-- metadata.json
|   `-- spec
|       |-- classes
|       |   `-- init_spec.rb
|       `-- spec_helper.rb
```

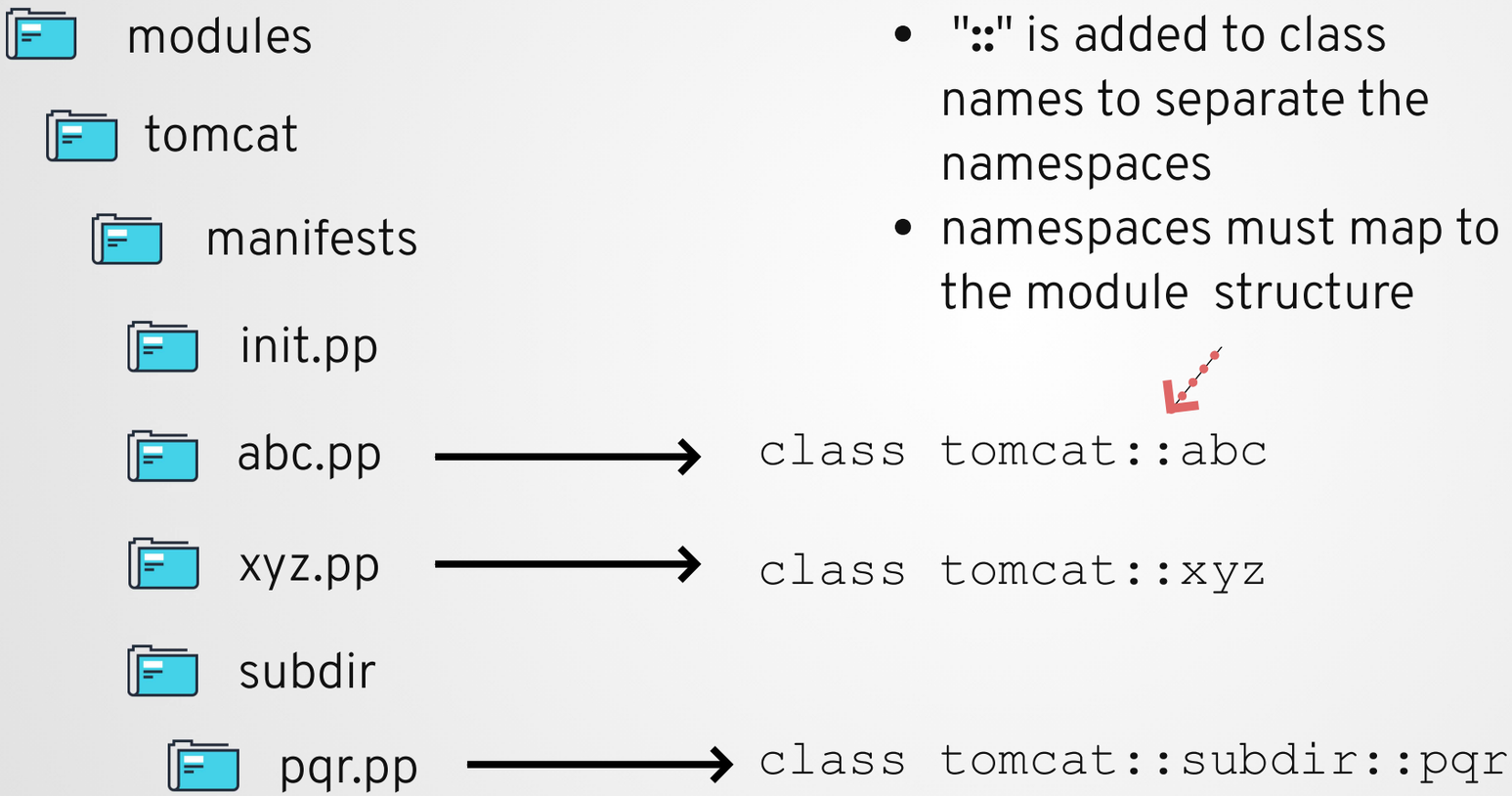
WRITING MANIFESTS

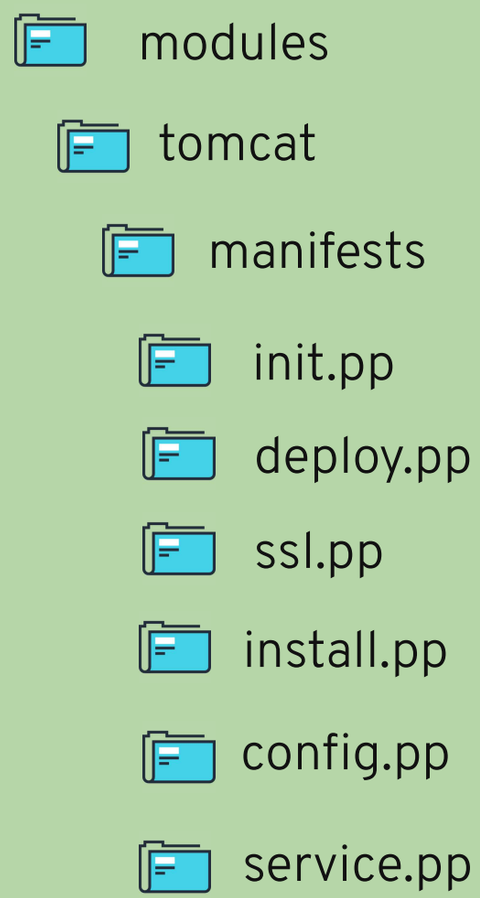


- puppet module generator creates a scaffolding required to write manifests
- it also generates the default manifest (init.pp)
- additional manifests can be added to modules/xxx/manifest directory

CLASSES

- each manifest contains a **class**, a named container, which encompasses one or more resources
- "**::**" is added to class names to separate the namespaces
- namespaces must map to the module structure





STRATEGIES

- every feature gets its own class, and in turns a manifest
- create a separate classes for each phase of application lifecycle. This provides more granular control

GROUP EXERCISE

TOMCAT EXAMPLE

phases		classes
• install packages to setup web server application and pre requisites	→	`tomcat::install`
• manage configurations for web servers	→	`tomcat::config`
• start/stop service	→	`tomcat::service`

GROUP EXERCISE

JAVA MODULE

phases		classes
• install packages to setup openjdk	→	`java::install`

GROUP EXERCISE

JAVA MANIFEST

file: modules/java/manifests/install.pp

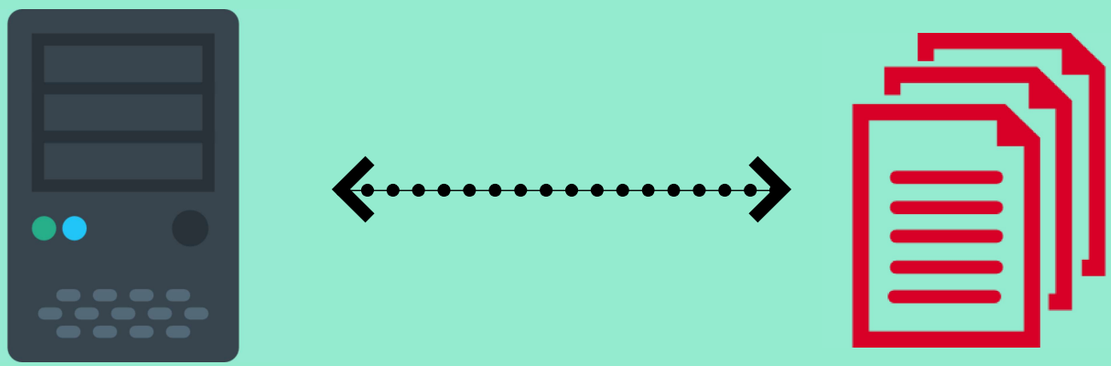
```
class java::install {

  package { [ 'epel-release', 'java-1.7.0-openjdk'] :
    ensure => installed,
  }

}
```

TASKS

1. Generate modules for java and tomcat
2. Create class to install java and
3. Create a node definition to apply the classes
4. Create classes to install tomcat and start the service, apply
5. Write classes to manage configuration files

“ now that we have written the class, lets learn how to apply it, in a client server model

NODE DEFINITION



node

list of classes

NODE DEFINITION

```
node 'nodename' {

    include class1
    include class2
    include class3, class4

    class { 'class3':

       param1 => val1,
       param2 => val2,
       param3 => val3,


    }

}

node 'app1.example.io', 'app2.example.io', 'app3.example.io' {
  include class1
  include class2
}


node /^(app|web)\.blr\d+\.example\.io$/ {
  include class1
  include class2
}
```

simple

groups

regex

CLASSIFYING NODES

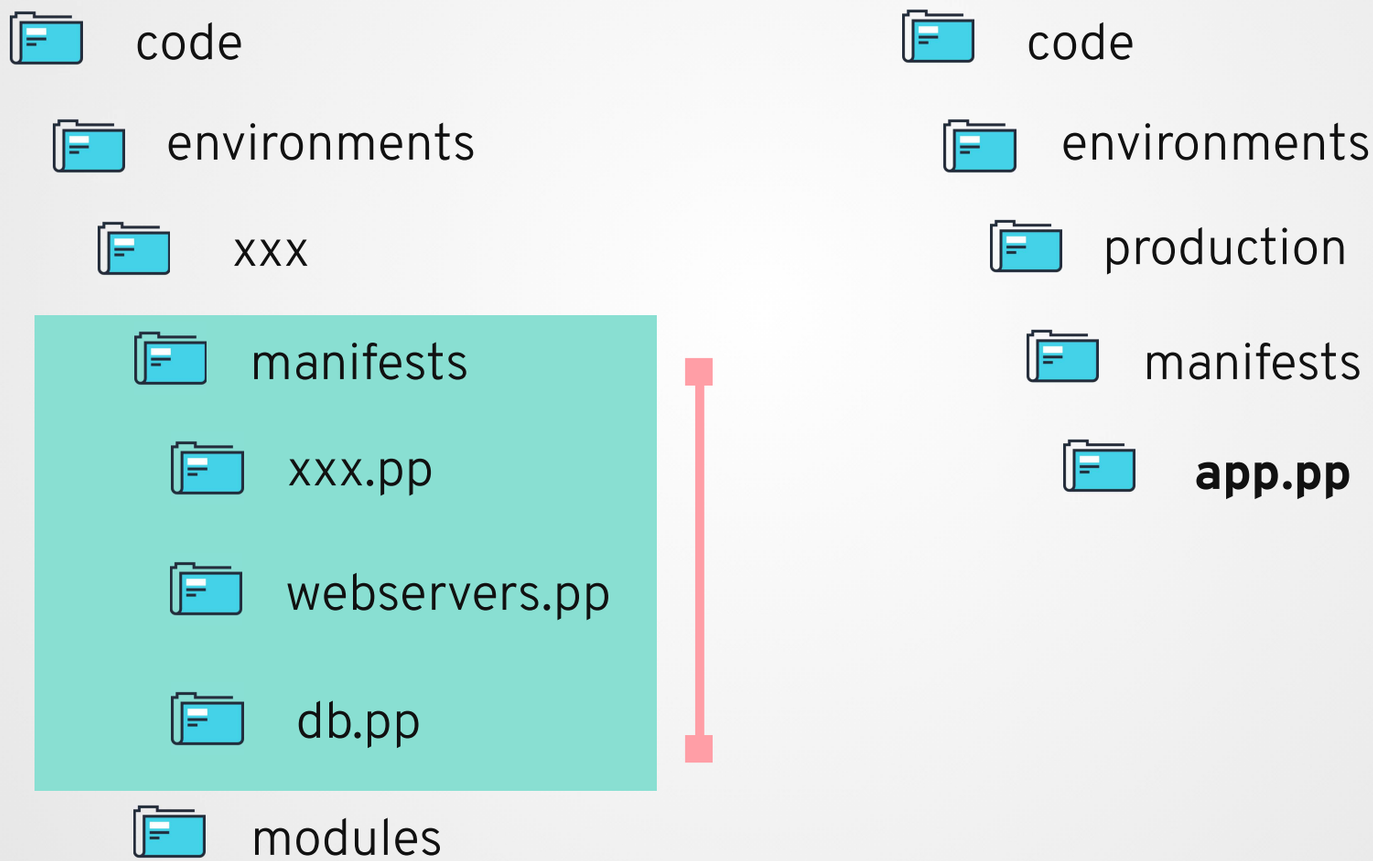
ENC

Puppet Entperprise

- node definition provides a mapping between nodes and a list of classes to apply
- node definition resides on the puppet master
- its the simplest way of classifying nodes
- there are more options available for node classification

NODE DEFINITION TERRITORY

NODE DEFINITION TERRITORY



GROUP EXERCISE

CREATE A NODE DEFINITION

file: environments/production/manifests/app.pp

```
node 'node1' {

   include java::install

}
```

GROUP EXERCISE

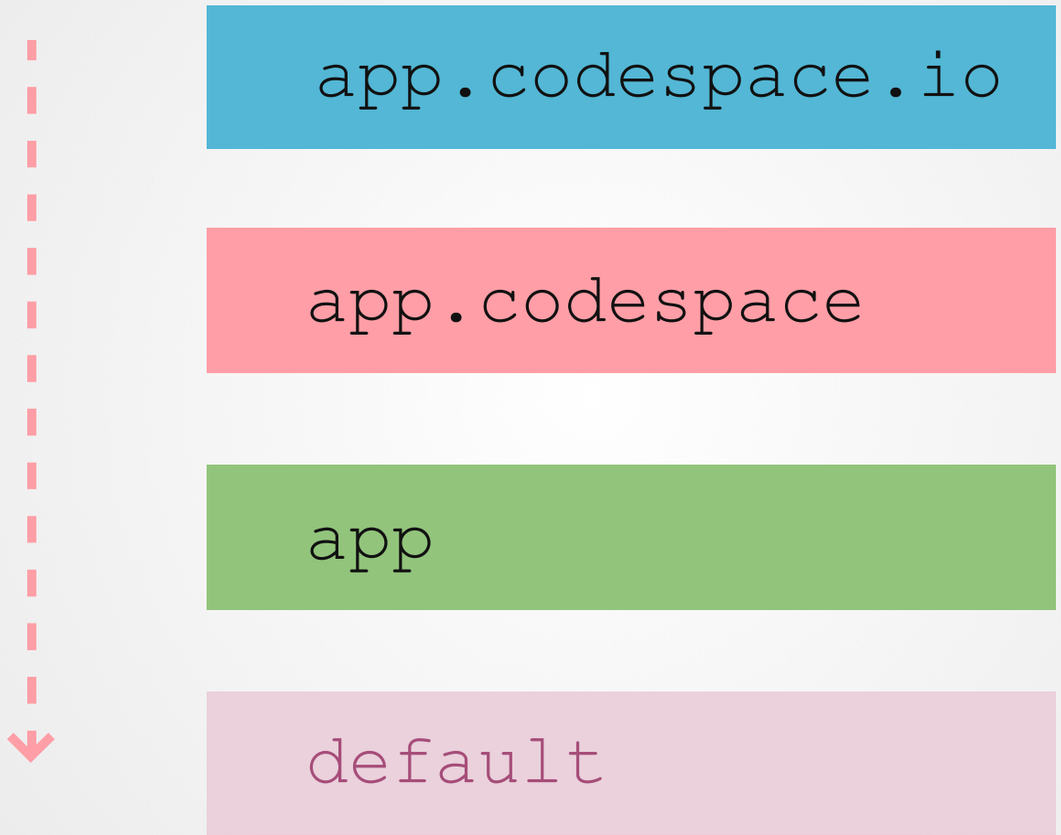
PULL AND APPLY FROM NODE 1

```
ssh devops@node1

sudo su

puppet agent -t
```

GROUP EXERCISE

APPLY ON NODE 2

```
ssh devops@node2

sudo su

puppet agent -t
```


app.codespace.io
app.codespace
app
default

GROUP EXERCISE

DEFAULT BLOCK

file: production/manifests/site.pp

```
node default {

  notify{'checkpoint_1':

    message => '

        CHECKPOINT_1

        DEFAULT BLOCK APPLIED
        Looks like there is no node definition for this host

    '
  }
}
```

PRO TIP

notify

PRO TIP

```
notify{'checkpoint_1':

    message => '

        CHECKPOINT_1

        DEFAULT BLOCK APPLIED
        Looks like there is no node definition for this host

    '
}
```

```
notify{'checkpoint_1':}
```

- notify is a resource and a metaparameter (will talk about this later)
- as a resource, can be used for checkpointing
- prints a message in the run log
- useful to check if a particular block of code is being called or not

LAB EXERCISE

- Create a node definition for **node2**
- Include the same **java** class that you applied earlier for node1
- Apply and validate on node2

DISCUSSION

What would this code do?

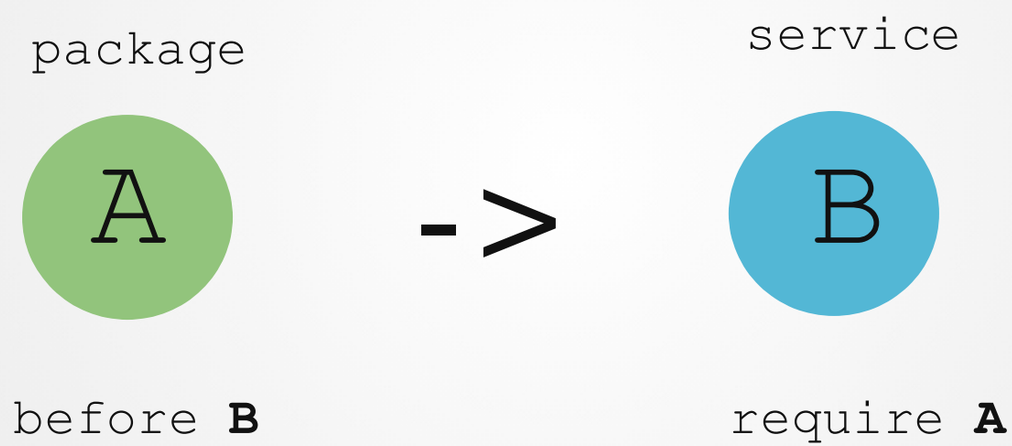
```
package { 'httpd':
  ensure => absent,
}

package {'nginx':
  ensure => $nginx_version,
  require => Package['httpd'],
}
```

ORDERING

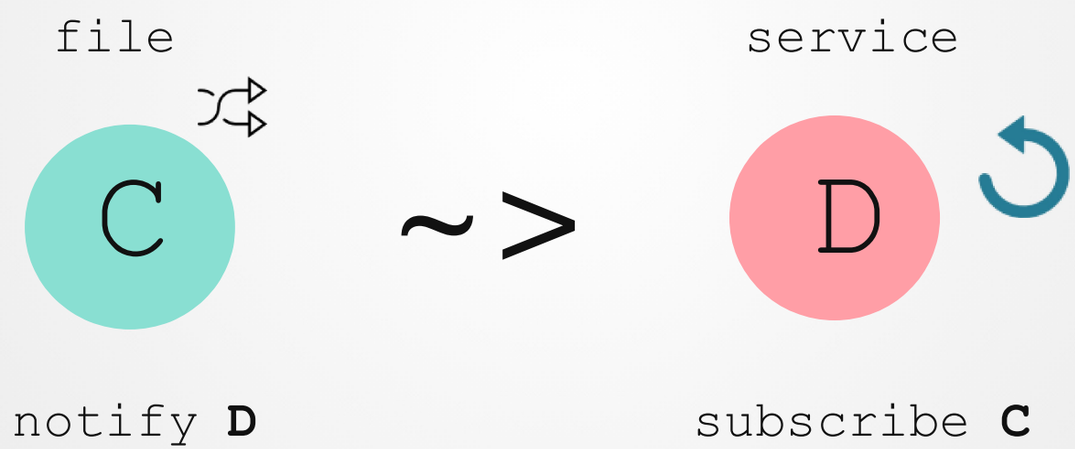
why ?

how ?

ORDERING



ORDERING WITH NOTIFICATION



SAMPLE

sample ordering.pp

```
package { 'nginx':
  ensure  => installed,
  before  => Service["nginx"],
}

file { 'nginx.conf':
  ensure  => file,
  mode    => '0644',
  notify  => Service["nginx"],
}

service { 'nginx':
  ensure      => running,
  enable      => true,
  hasrestart  => true,
  hasstatus   => true,
  require     => [ Package["nginx"], File["nginx.conf"] ]
  subscribe   => File["nginx.conf"],
}

Package["nginx"] -> File["nginx.conf"] ~> Service["nginx"]
```

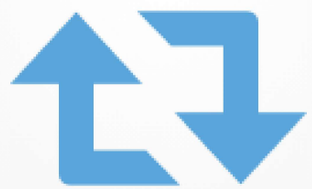
Type['title']

CAPS

META - PARAMETERS

before

require



notify

subscribe

```
puppet describe -sm package
```

```
Meta Parameters
---------------
    alias, audit, before, consume, export,
    loglevel, noop, notify, require,
    schedule, stage, subscribe, tag
```

TASKS

1. Generate modules for java and tomcat
2. Create class to install java and
3. Create a node definition to apply the classes
4. Create classes to install tomcat and start the service, apply
5. Write classes to manage configuration files

LAB EXERCISE

LAB EXERCISE

Create and apply the following recipes for tomcat

INSTALL

create a tomcat::install recipe to install tomcat along with example apps. Packages to install are

- **tomcat**
- **tomcat-webapps**

SERVICE

create a tomcat::service recipe to start and enable **tomcat** service

Service should depends on package **tomcat**

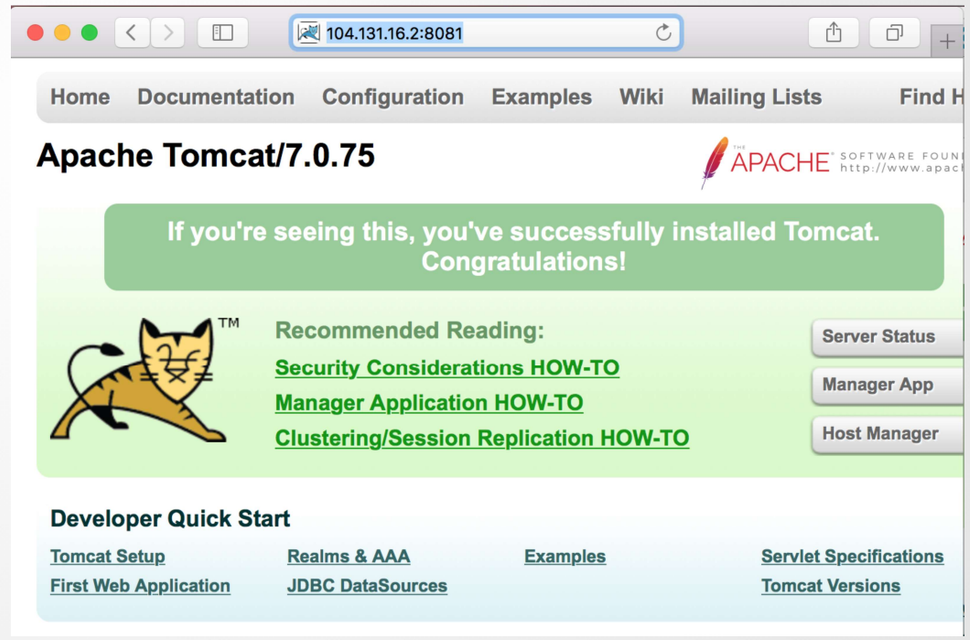
LAB EXERCISE

LAB

EXPECTED OUTPUT

- You should be able to browse to http://ipaddress:8081 port in the browser
- Tomcat home page is the expected output

http://IPADDR:8081



GROUP EXERCISE

SIMPLIFY NODE DEFINITION

Lets call all other manifests from init.pp

```
class tomcat {

  include java::install
  include tomcat::install
  include tomcat::service

}
```

```
node 'node1' {

  include tomcat

}


node 'node2' {

   include tomcat

}
```

GROUP EXERCISE

DEPENDENCIES

- we included java cookbook in tomcat init.pp
- We would also add dependency in metadata.json of tomcat module that depends on java

```
{
  "name": "user-tomcat",
  "version": "0.1.0",
  "author": "user",
  "summary": null,
  "license": "Apache-2.0",
  "source": "",
  "project_page": null,
  "issues_url": null,
  "dependencies": [
    {"name":"puppetlabs-stdlib","version_requirement":">= 1.0.0"}
  ],
  "data_provider": null
}
```

TASKS

1. Generate modules for java and tomcat
2. Create class to install java and
3. Create a node definition to apply the classes
4. Create classes to install tomcat and start the service, apply
5. Write classes to manage configuration files

MANGING FILES

- We will need to manage configurations eg. tomcat.conf
- since chef is a centralized configuration management system, we will keep the files centrally in cookbooks, which will then be copied to all managed nodes

LAB EXERCISE

MANGING FILES

- modules
  - tomcat
    - manifests
      - init.pp
      - install.pp
      - service.pp
      - config.pp
    - files
      - tomcat.conf

- Create tomcat.conf file in tomcat modules' files directory.

- add tomcat::config class to copy these files to the relevant locations on destination hosts

link to tomcat.conf source

destination path in nodes:
/etc/tomcat/tomcat.conf

LAB EXERCISE

MANGING FILES

- cookbooks
  - tomcat
    - recipes
      - default.rb
      - install.rb
      - service.rb
      - config.rb
    - files
      - tomcat.conf
      - tomcat-users.xml

- Generate tomcat.conf using chef generate file in tomcat cookbook directory.

- add tomcat::config recipe to copy these files to the relevant locations on destination hosts

link to tomcat.conf source
path: /etc/tomcat/tomcat.conf

link to tomcat-users.xml source
path: /etc/tomcat/tomcat-users.xml

GROUP EXERCISE

TOMCAT.CONF

file: cookbooks/tomcat/files/default/tomcat.conf

```
TOMCAT_CFG_LOADED="1"

JAVA_HOME="/usr/lib/jvm/jre"
JAVA_OPTS="-Xms64m -Xmx128m -XX:MaxPermSize=128M -Djava.security.egd=file:/dev/./urand

CATALINA_BASE="/usr/share/tomcat"
CATALINA_HOME="/usr/share/tomcat"
JASPER_HOME="/usr/share/tomcat"
CATALINA_TMPDIR="/var/cache/tomcat/temp"


TOMCAT_USER="tomcat"

SECURITY_MANAGER="false"

SHUTDOWN_WAIT="30"

SHUTDOWN_VERBOSE="false"

CATALINA_PID="/var/run/tomcat.pid"
```

GROUP EXERCISE

TOMCAT::CONFIG

file: modules/tomcat/manifests/config.rb

```
class tomcat::config {

  file { '/etc/tomcat/tomcat.conf':
    source    => 'puppet:///modules/tomcat/tomcat.conf',
    owner     => 'tomcat',
    group     => 'tomcat',
    mode      => '0644'
  }

}
```

TOMCAT::CONFIG

file: cookbooks/tomcat/recipes/config.rb

```
cookbook_file '/etc/tomcat/tomcat.conf' do
  source 'tomcat.conf'
  owner 'tomcat'
  group 'tomcat'
  mode  0644
  action :create
end

cookbook_file '/etc/tomcat/tomcat-users.xml' do
  source 'tomcat-users.xml'
  owner 'tomcat'
  group 'tomcat'
  mode  0644
  action :create
end
```

REFRESHING SERVICE

file: modules/tomcat/manifests/config.rb

```
class tomcat::config {

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

TOMCAT::CONFIG

Update file: cookbooks/tomcat/recipes/config.rb

```
cookbook_file '/etc/tomcat/tomcat.conf' do
  source 'tomcat.conf'
  owner 'tomcat'
  group 'tomcat'
  mode  0644
  action :create
  notifies :restart, 'service[tomcat]', :delayed
end
```

Note: Add config.rb recipe to default.rb

TOMCAT::CONFIG

file: cookbooks/tomcat/recipes/config.rb

```
cookbook_file '/etc/tomcat/tomcat.conf' do
  source 'tomcat.conf'
  owner 'tomcat'
  group 'tomcat'
  mode  0644
  action :create
  notifies :restart, 'service[tomcat]', :delayed
end

cookbook_file '/etc/tomcat/tomcat-users.xml' do
  source 'tomcat-users.xml'
  owner 'tomcat'
  group 'tomcat'
  mode  0644
  action :create
  notifies :restart, 'service[tomcat]', :delayed
end
```