🐾 Puppet – Cheatsheet Completo

📌 ¿Qué es Puppet?

- Herramienta de **Configuration Management**.
- Permite **automatizar tareas**: instalación de paquetes, configuración de archivos, servicios, usuarios, permisos, etc.
- Usa un **lenguaje declarativo**: vos decís *"quiero este estado"* y Puppet lo garantiza.

📦 Conceptos fundamentales

1. Manifests

Archivo `.pp` con configuraciones en Puppet. Ejemplo:

```
file { '/tmp/test.txt':
  ensure => present,
  content => "Hola Axel",
}
```

---

2. Recursos (Resources)

Bloques de configuración que definen un objeto a gestionar.

Estructura:

```
<tipo> { '<título>':
  atributo => valor,
  atributo => valor,
}
```

Ejemplos de tipos comunes:

🗂 file

```
file { '/etc/motd':
  ensure  => present,
  content => "Bienvenido!",
  mode    => '0644',
}
```

📦 package

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

🔁 service

```
service { 'nginx':
  ensure => running,
  enable => true,
}
```

👤 user

```
user { 'axel':
  ensure => present,
  shell  => '/bin/bash',
}
```

🔗 3. Dependencies (Require / Before / Notify / Subscribe)

Sirven para **ordenar** la ejecución.

👉 require

"No hagas esto hasta que lo otro exista."

```
service { 'nginx':
  ensure => running,
  require => Package['nginx'],
}
```

👉 before

Hago esto primero.

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

```
  before => Service['nginx'],
}
```

👉 notify

Si cambia, notifica al otro recurso.

```
file { '/etc/nginx/nginx.conf':
  notify => Service['nginx'],
}
```

👉 subscribe

Se reinicia si se modifica el archivo.

```
service { 'nginx':
  subscribe => File['/etc/nginx/nginx.conf'],
}
```

🗀 4. Clases

Permiten organizar código en bloques reutilizables.

Definición:

```
class nginx {
  package { 'nginx': ensure => installed }
  service { 'nginx': ensure => running }
}
```

Uso:

```
include nginx
```

📦 5. Módulos

Estructura estándar para empaquetar clases, archivos, templates, etc.

Estructura típica:

```
mimodulo/
 ├── manifests/
 │    └── init.pp
 ├── files/
 ├── templates/
 └── README.md
```

📝 6. Plantillas (Templates)

Usadas para generar archivos dinámicos.

Template ERB:

```
ServerName <%= @hostname %>
```

Llamado desde Puppet:

```
file { '/etc/httpd/conf/httpd.conf':
  content => template('mimodulo/httpd.erb'),
}
```

🛠️ 7. Facter

Herramienta que da **facts** (datos del sistema):

Ejemplos:

- `osfamily`
- `hostname`
- `ipaddress`

Uso:

```
notify { $facts['os']['family']: }
```

🔍 8. Tipos de ejecución

Agent–Master

- Puppet Server centralizado.
- Los nodos descargan configuraciones.

Standalone (puppet apply)

Ejecuta un manifest local:

```
puppet apply test.pp
```

(Este es el modo que usa tu curso).

🧪 9. Comandos comunes

Comando	Descripción
`puppet apply file.pp`	Ejecutar un manifest local
`puppet lint file.pp`	Analiza errores de sintaxis
`puppet resource user root`	Ver recurso en formato puppet
`facter`	Muestra facts del sistema
`puppet module install <nombre>`	Instalar módulo

🚦 10. idempotencia

Puppet **solo hace cambios si algo está fuera del estado deseado**. Es un pilar clave de un sistema declarativo.

🧩 Ejemplo completo de un manifest

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

  file { '/var/www/html/index.html':
    ensure  => present,
    content => 'Hola desde Puppet!',
    notify  => Service['nginx'],
  }

  service { 'nginx':
```

```
    ensure  => running,
    enable  => true,
  }
}

include webserver
```