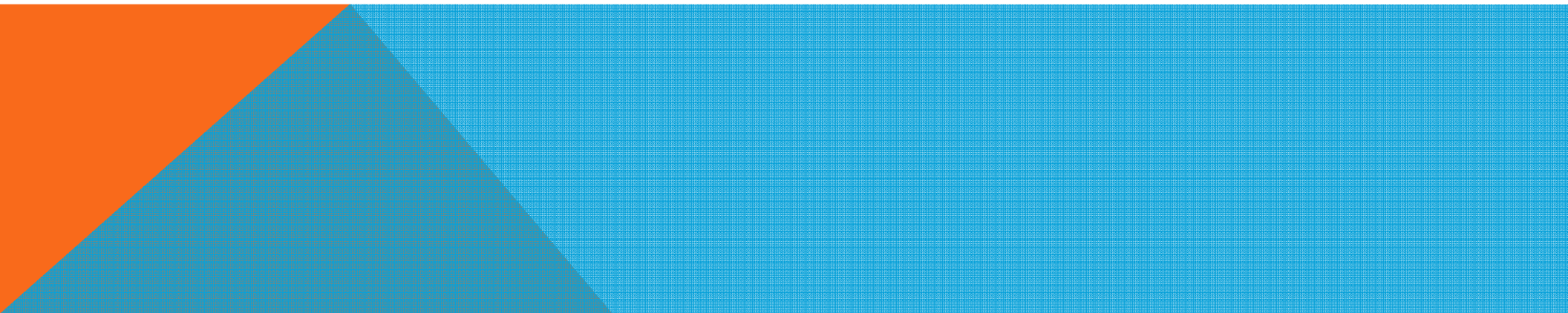


Now We know

- ✓ Need for Configuration Management
- ✓ Intro To Puppet
- ✓ Benefits of Puppet
- ✓ Puppet Basics - Various resources/attributes in Puppet
- ✓ Classes
- ✓ Use Case and Exercise



Requirement of Configuration Management

- Automated Infrastructure
- Saves time, Increases efficiency
- Allows for scalability
- Reduces economic impact
- Improves flexibility

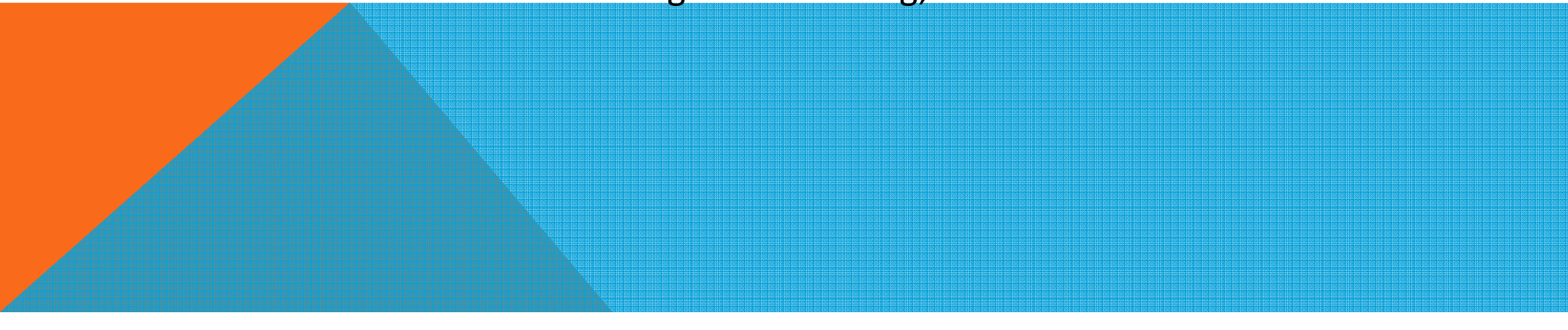


Puppet Ecosystem

- Started in 2005
- Written in Ruby
- released with an open source license Apache2.
- Open source and Enterprise versions

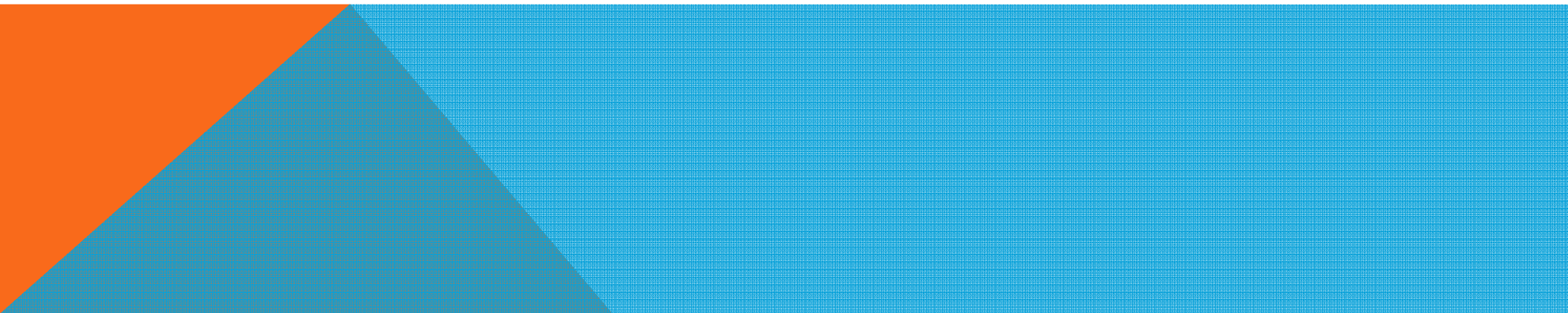


Puppet Basics

1. When Puppet is executed, it first runs `facter`, a companion application, which gathers facts about the system and send to Master.
 2. Facts are information about a puppet agent's current state in the form of variables that can be used in manifests to manage how and what resources to be provided to the clients.
 3. When Master receives a connection, it looks in its manifests (starting from `/etc/puppet/manifests/site.pp`) what resources have to be applied for that agent.
 4. Resources are various component of a system that can be managed using Puppet, for example, packages to install, services to start, files to manage, users to create, cronjobs and also custom resources such as Nagios monitoring, and so on.
- 

Puppet Basics

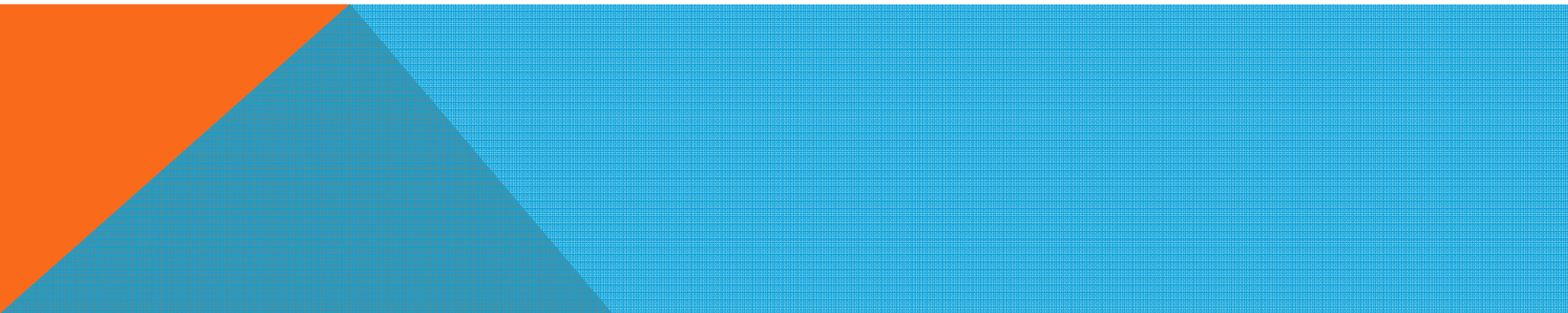
5. Puppet code are known as manifests, which are simple text files written in DSL with a .pp extension.
6. The Master parses all the DSL code and produces a catalog that is sent back to the client in the PSON format
7. Once the client receives the catalog, it starts to apply all the resources declared there.
8. Puppet can report the changes it makes on the system and audit the drift between the system's state and the desired state as defined in its catalog.



Puppet Drift

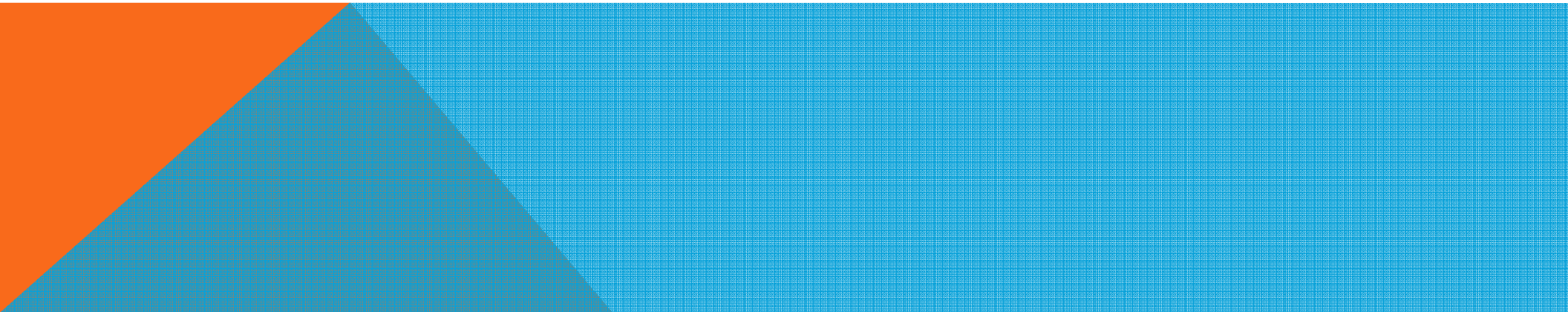
The “desired” state of the server aka puppet node is set by the Puppet-master. The computation of puppet agent from current state to desired state is called a – DRIFT.

So whenever a DRIFT is found on the puppet node. The node start working to achieve the desired state, as instructed by the Puppet Master.



Puppet is Idempotent

If same catalog is applied multiple times and if there exist any manual modification, they are reverted to the state defined by Puppet; if the system's resources are already at the desired state, nothing happens.



Puppet is Declarative

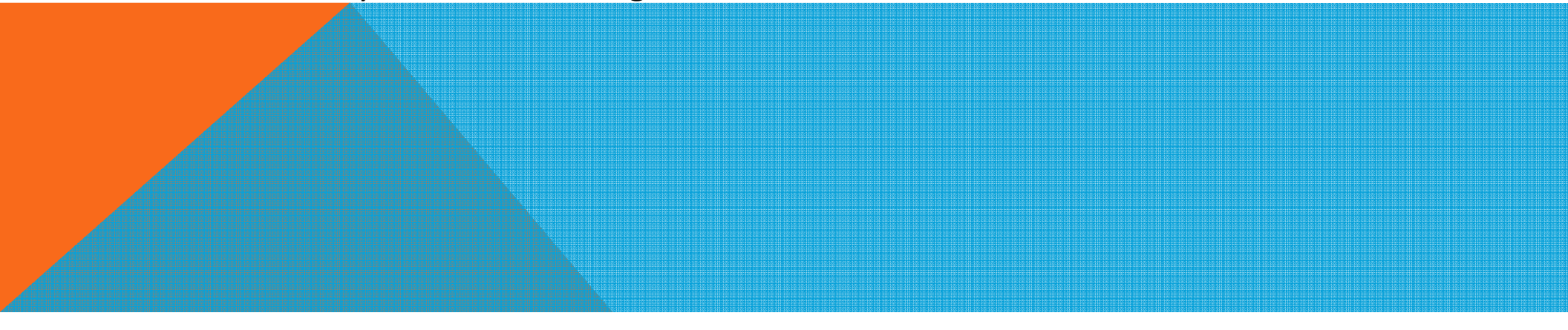
Puppet features a declarative Domain Specific Language (DSL).

It does not follow imperative approach of writing series of instructions, as in a shell script or a Ruby program.

```
user { 'sharad':  
  ensure => present,  
}
```

This is Puppet language for the declaration "The sharad user should be present."

Puppet manifest, for configurations is a set of declarations about what things should exist, and how they should be configured.



Puppetmaster

Central Management Server, from where all the agents are managed. /etc/hosts

Services Name : puppetmaster

Port: tcp/8140

Configuration file: /etc/puppet/puppet.conf

Certificates: /var/lib/puppet/ssl/

Puppet default manifest file: /etc/puppet/manifests/site.pp

Puppet default modules directory: /etc/puppet/modules



Puppet Configuration File

[main]

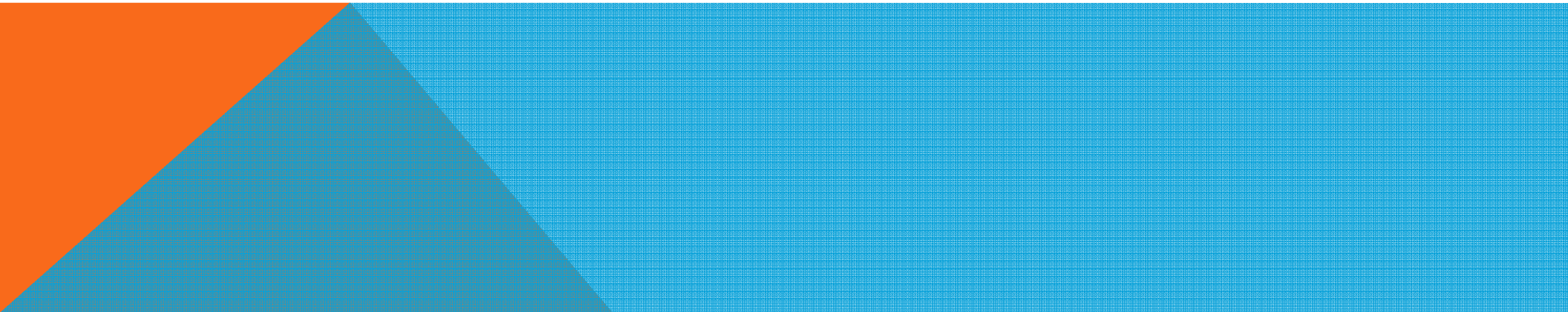
```
logdir=/var/log/puppet
vardir=/var/lib/puppet
ssldir=/var/lib/puppet/ssl
rundir=/var/run/puppet
factpath=$vardir/lib/facter
#templatedir=$confdir/templates
```

[agent]

```
Server = <puppetmaster>
```

[master]

```
# These are needed when the puppetmaster is run
by passenger
# and can safely be removed if webrick is used.
ssl_client_header = SSL_CLIENT_S_DN
ssl_client_verify_header = SSL_CLIENT_VERIFY
```



Configuring Puppet Agent

1. Edit `/etc/puppet/puppet.conf`. Under [main] section,

`server=puppet.example.com`

2. Edit `/etc/default/puppet`

Set `start = yes`

3. Start the puppet service on node

`Service puppet restart`

GO TO PUPPET MASTER:

1. Check the Certificate request and sign it

`puppet cert --list`

`puppet cert --sign`

`puppetagent.example.com`

GO TO PUPPET AGENT:

`Puppet agent -t`

Agent authentication at Puppetmaster

puppet cert list # List the unsigned clients certificates

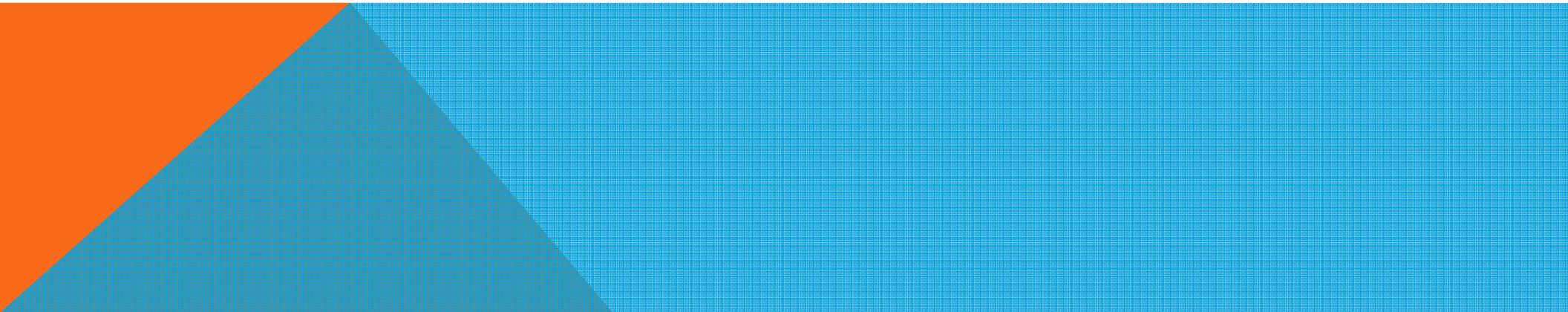
puppet cert list --all # List all certificates

puppet cert sign <certname> # Sign the given certificate

Puppet cert revoke <certname> #to revoke a certificate

Puppet cert clean <certname> #to clean a certificate request from

puppet master

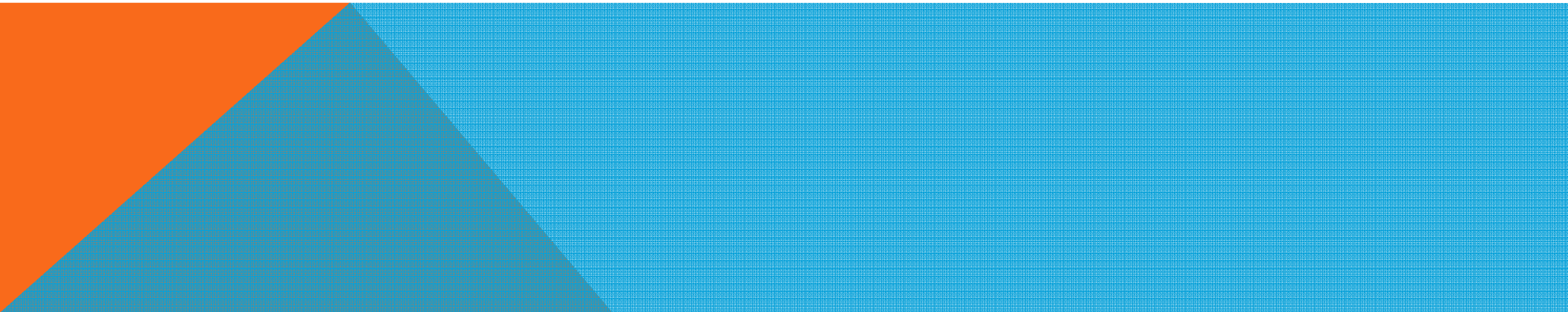


Puppet Resources

File Resource:

```
file{'data4':  
  ensure => file,  
  path => '/tmp/monday.txt',  
  content => "Anything\n",  
}
```

```
file{'data4':  
  ensure => directory,  
  path => '/tmp/monday',  
}
```



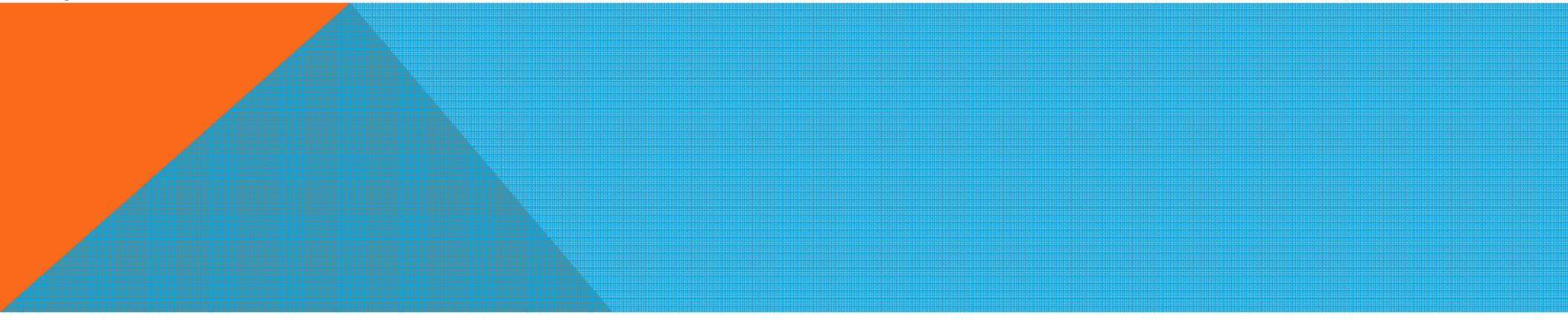
Puppet Resources

Package Resource:

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

Service Resource:

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



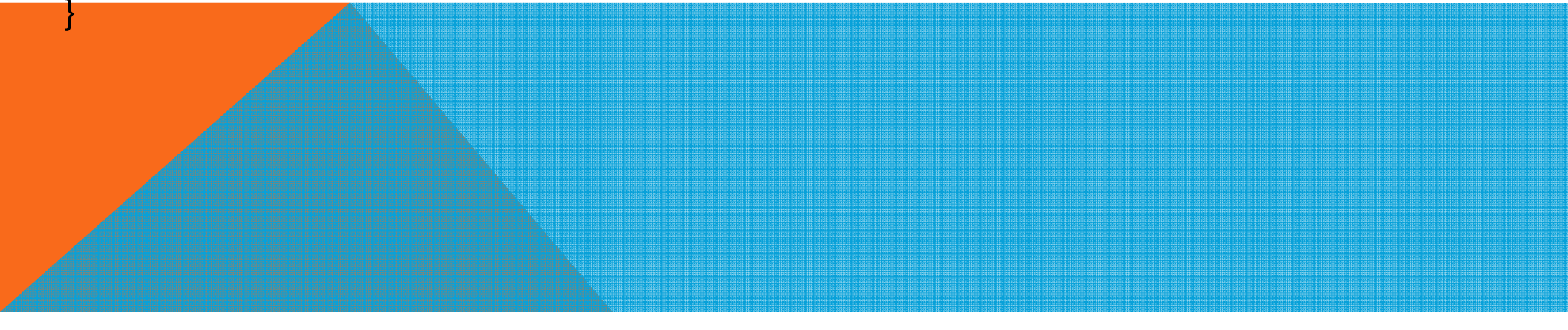
Puppet Resources

Exec Resource:

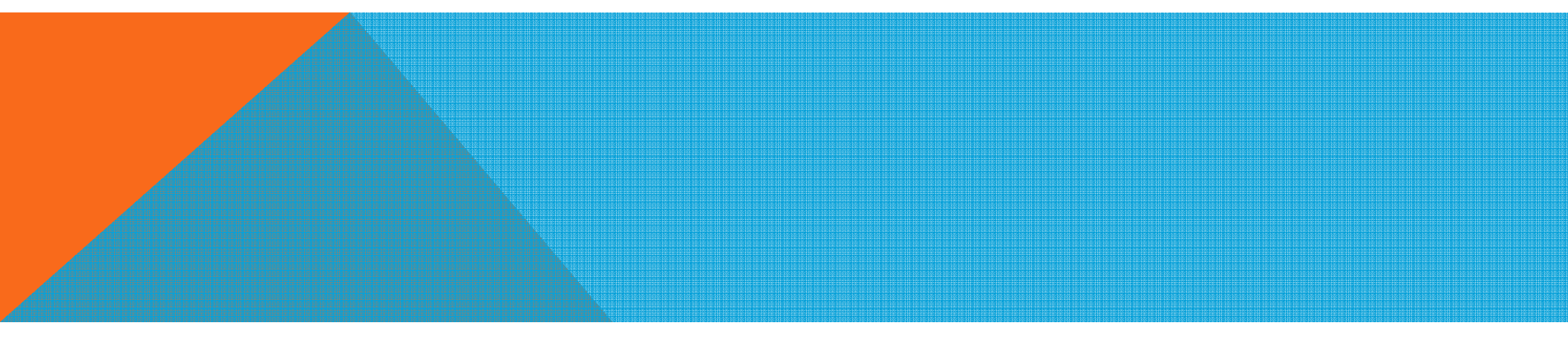
```
exec { 'command1':  
  command => '/bin/touch /tmp/sharadfile',  
}
```

Cron Resource:

```
cron {'filesharad':  
  ensure => present,  
  user => root,  
  command => 'touch /tmp/sharad',  
  minute => '12',  
  hour => '10',  
}
```



Metaparameters:

- ❑ **Before** is used in earlier resources and list resources that depends on it.
 - ❑ **Require** is used in later resources and list resources that it depends on.
 - ❑ **Notify** used to notify other resource about the drift
 - ❑ **Subscribe** is sued by those resources that should be refreshed after some incident in another resource. (Eg. Service, mount)
 - ❑ **Require** ()
- 

Package, File and Service

```
package{'vsftpd':  
    ensure => present,  
    Before => File['vsftpf.conf'],  
}  
file{'vsftpd.conf':  
    ensure => file,  
    source => '/root/vsftpd.conf',  
    path => '/etc/vsftpd.conf',  
    owner => root,  
    group => root,  
    require => Package['vsftpd'],  
    notify => Service['vsftpd'],  
}
```

```
service{'vsftpd':  
    ensure => running,  
    enable => true,  
    subscribe => File['vsftpd.conf'],  
}
```

Puppet Classes

A block of code referred with a name. It helps in code reusability

```
Class ftpserver{  
    package{'vsftpd':  
        ensure => present,  
        Before => File['vsftpf.conf'],  
    }  
    file{'vsftpd.conf':  
        ensure => file,  
        source => '/root/vsftpd.conf',  
        owner => root,  
        group => root,  
        require => Package['vsftpd'],  
        notify => Service['vsftpd'],  
    }  
}
```

```
service{'vsftpd':  
    ensure => running,  
    enable => true,  
    subscribe => File['vsftpd.conf'],  
}
```

Puppet Manifests

Puppet programs are called “manifests,” and they use the .pp file extension.

The core of the Puppet language is the resource declaration. A resource declaration describes a desired state for one resource.

A resource declaration looks like this:

```
RESOURCE { NAME:  
  ATTRIBUTE => VALUE,  
  ...  
}
```

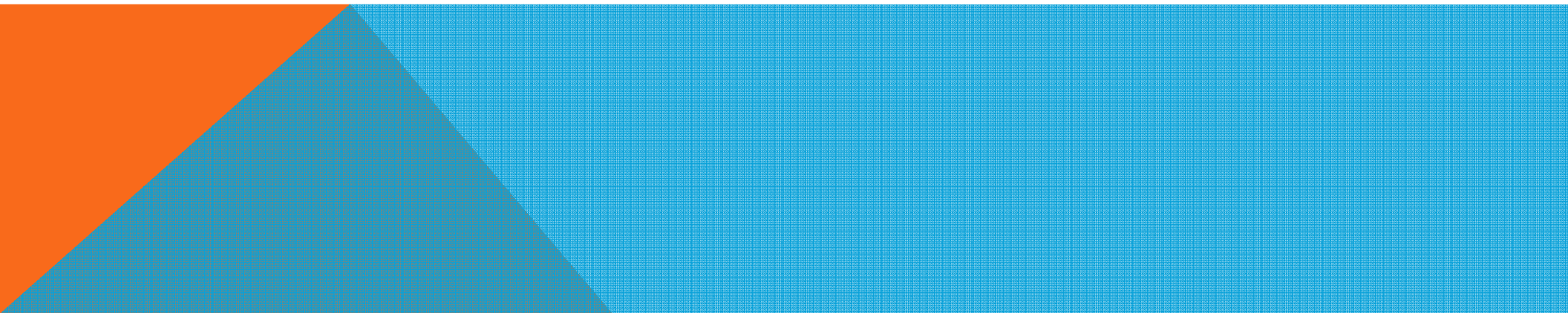
To apply a manifest:

```
# puppet apply filename.pp {To test master and agent on single node}
```



Puppet Ecosystem

- **Facter** is a required tool for Puppet as it is executed on each agent node and gathers information (facts) in key/value pairs that are used by Puppet.
facter , facter os, facter ipaddress, facter osfamily, facter uptime
- **PuppetDB** is a database backend that stores all the data gathered and generated by Puppet.
- **Puppet Enterprise** is the commercial solution to manage Puppet, Mcollective, and PuppetDB via a web frontend.
- Puppet Forge



Puppet Help

Puppet describe <resourcename>

#to view info for a particular resource

Puppet resource -types

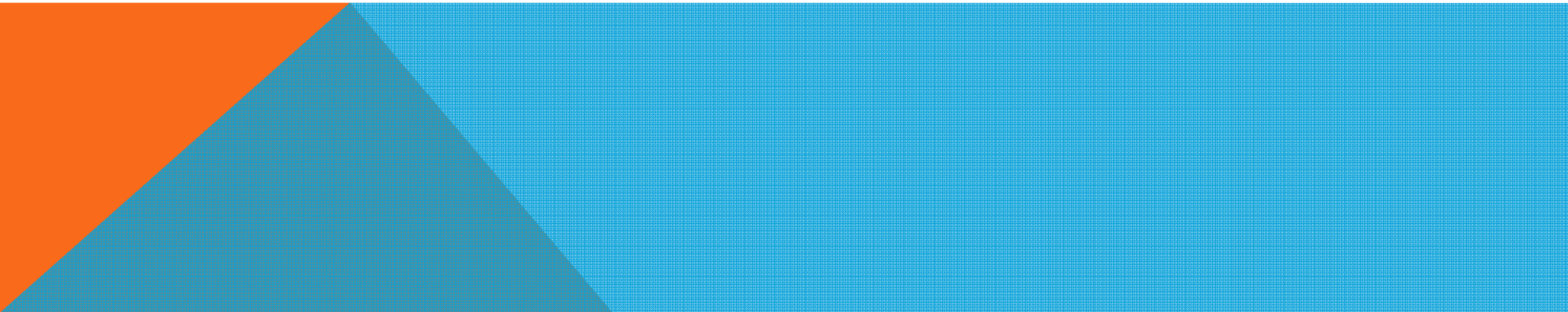
to view all resource managed by puppet

puppet master -configprint

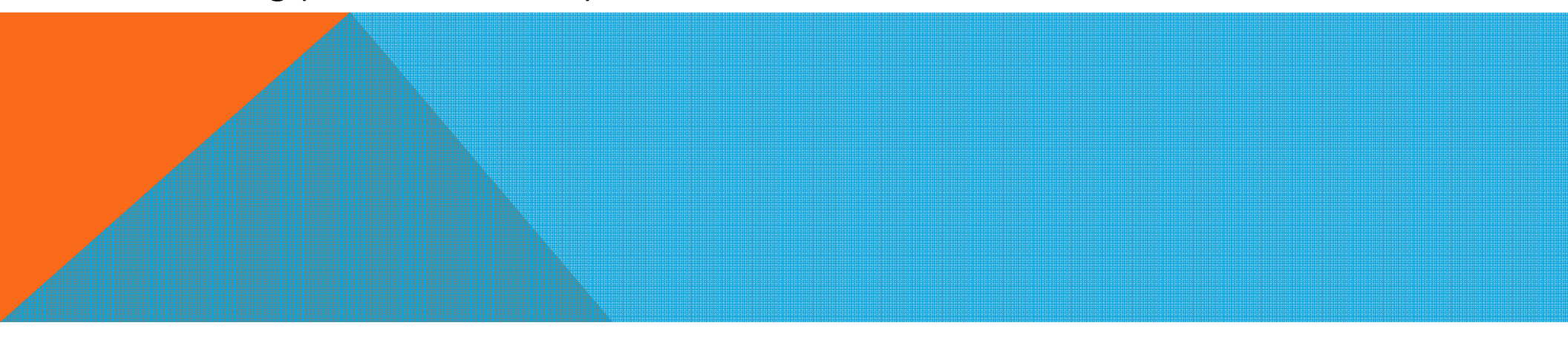
To print puppetmaster configurations

puppet config print all

#specifies all parameters that can configured in puppet.conf



Ensure Attribute:

- As a normal file (ensure => file)
 - As a directory (ensure => directory)
 - As a symlink (ensure => link)
 - As a package (ensure => installed)
 - As a package (with latest version)(ensure => latest)
 - As any of the above (ensure => present)
 - As nothing (ensure => absent)
- 

Package, File and Service

```
package{'vsftpd':  
    ensure => present,  
    Before => File['vsftpf.conf'],  
}  
file{'vsftpd.conf':  
    ensure => file,  
    source => '/root/vsftpd.conf',  
    path => '/etc/vsftpd.conf',  
    owner => root,  
    group => root,  
    require => Package['vsftpd'],  
    notify => Service['vsftpd'],
```

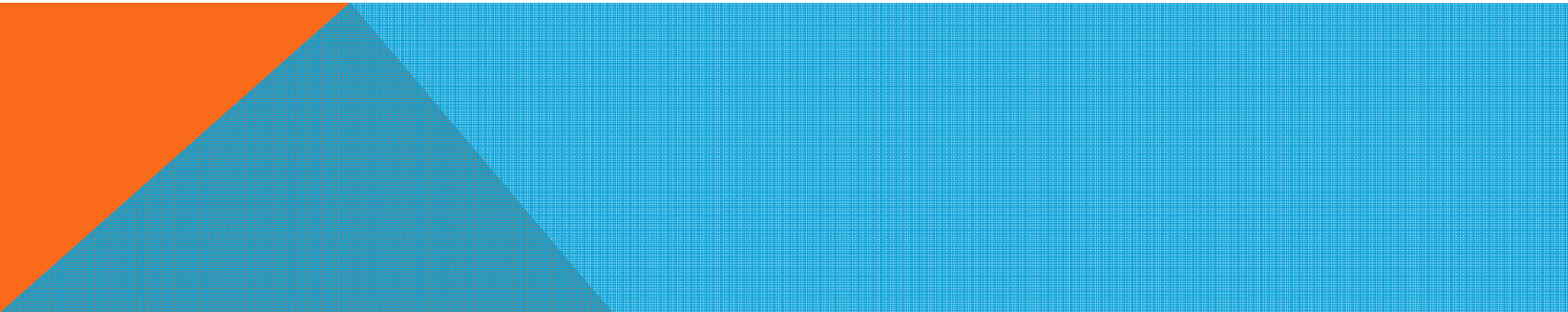
```
}  
service{'vsftpd':  
    ensure => running,  
    enable => true,  
    subscribe => File['vsftpd.conf'],  
}
```

nodes.pp

Site.pp is applied on all the nodes but nodes.pp is used for configuring changes on specific nodes.

```
node 'puppetagent.example.com' {  
    file{'/tmp/agentdir'  
    ensure => 'directory'  
    }  
  
    user{'sharad':  
    ensure => present,  
    }  
}
```

Edit site.pp and simply add,
import 'nodes.pp'

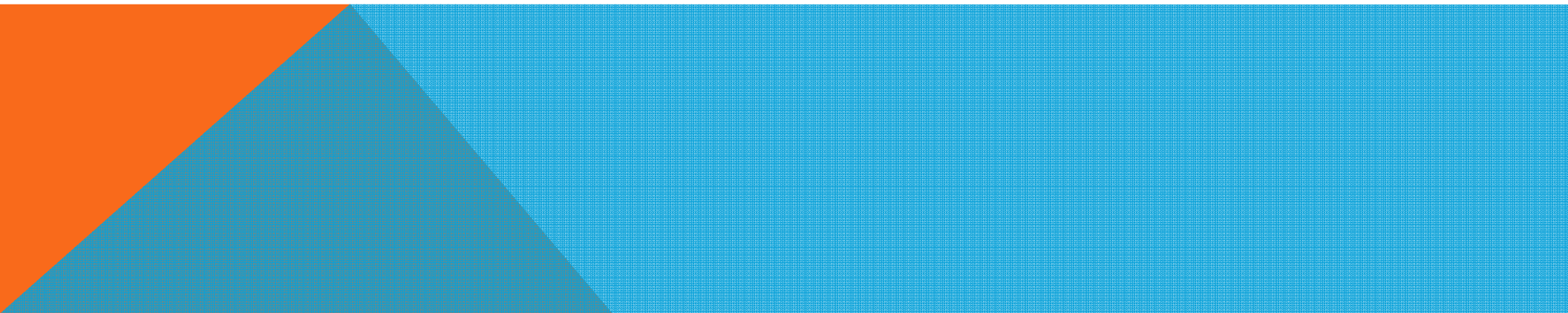


Node Inheritance

Site.pp is applied on all the nodes but nodes.pp is used for configuring changes on specific nodes.

```
Node 'lampnginx' {  
  include lamp  
  Include nginx  
}
```

```
node 'puppetagent.example.com' inherits 'lampnginx'{  
}
```



Regex Expression

- Regular expressions (regex) can be used to make groups of node to have same configuration changes.
- This is another method for writing a single node statement that matches multiple nodes.

```
node /^ip-10-0-0-\d+$/ {           # would match names like
include common                     ip-10-0-0-43,
}
```

ip-10-0-0-67 etc.

```
node /^(pc1|pc2)\.example\.com$/ { # hostnames like
include common                     pc1.example.com,
}
```

pc2.example.com etc.

Puppet Modules

- ❑ A module is a directory at `/etc/puppet/modules/`.
- ❑ The module's name must be same as the name of the directory.
- ❑ The manifests directory should always contain an `init.pp` file.
- ❑ It contains a manifests directory, which can contain any number of `.pp` files.
- ❑ This file must contain a single class definition. The class name must be same as module's name.

`/etc/puppet/modules/`

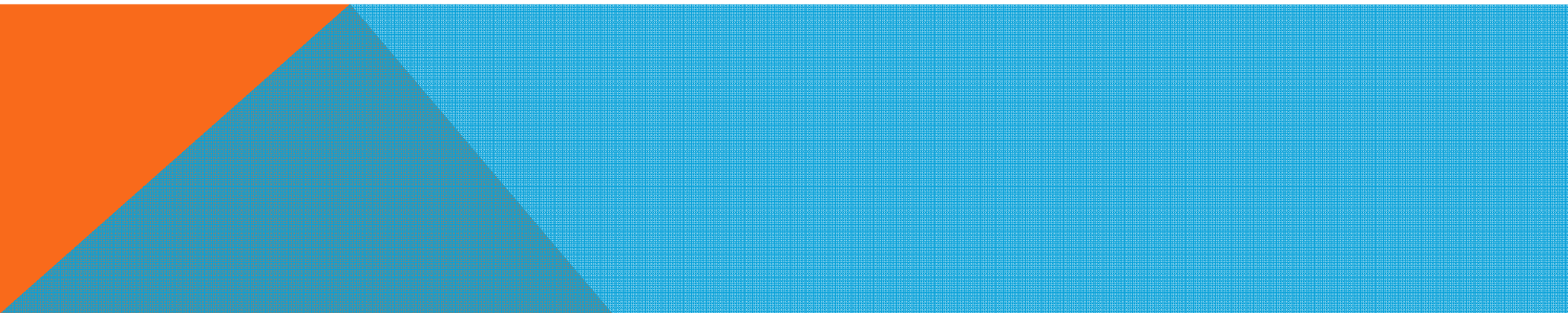
```
├── lamp
│   ├── files
│   └── manifests
│       └── init.pp
└── ntp
    ├── files
    └── manifests
        └── init.pp
```

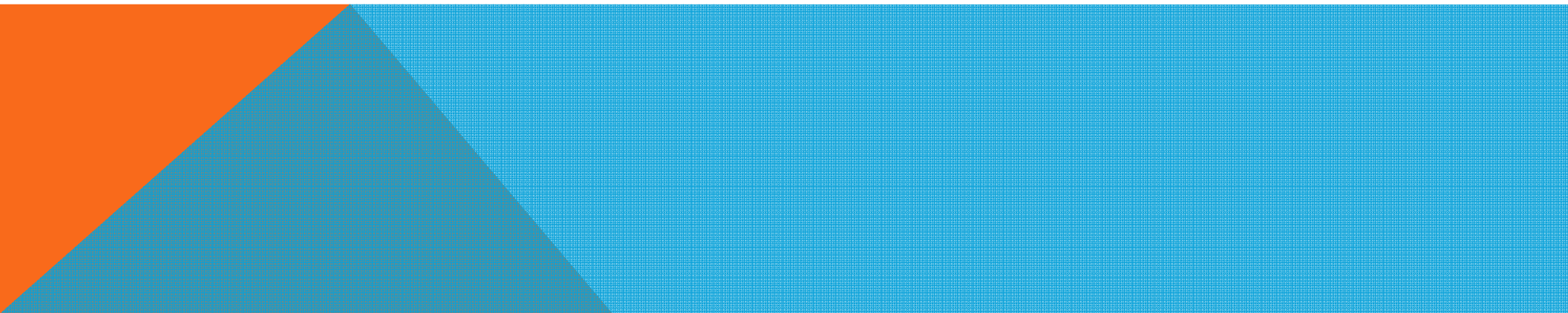
Fileserver

Puppet can create a file store from where files or directories can be copied as and when required. We can either uncomment following block from add following block in the puppet.conf file

```
#      [files]
# path /etc/puppet/files
# allow *
```

```
file { '/tmp':                                (copy files from /etc/puppet/files to agents /tmp dir)
    ensure => 'file',
    mode => 0644,
    source => "puppet:///files/hosts",
}
```



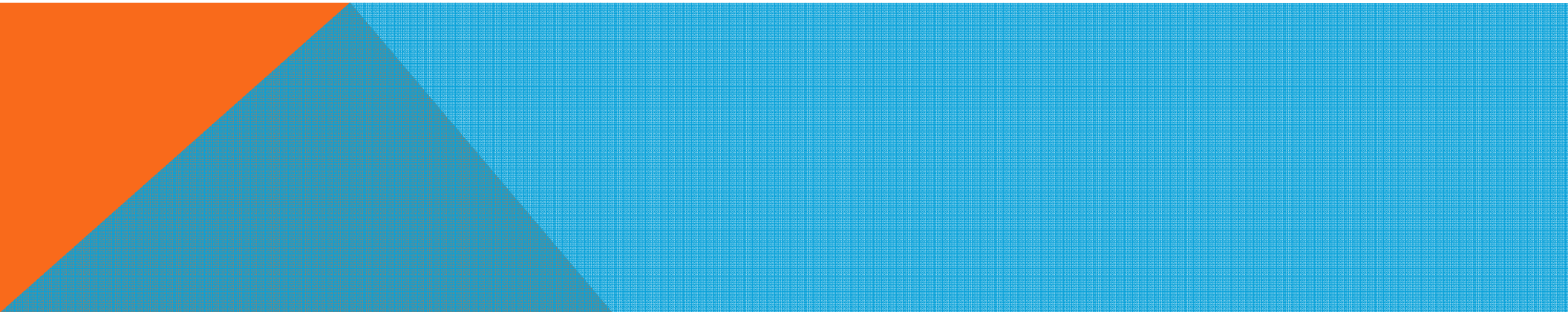


Puppet Ecosystem

Few more works of Puppet labs and others and that integrates well with Puppet.

- **Hiera** is a key-value lookup tool that is the current choice of reference for storing data related to your Puppet infrastructure. It basically help in replacing hardcoded data in manifests with variables wherever possible that helps in code reusability.
- **MCollective** is an orchestration framework that allows parallel execution of tasks on multiple servers. It is a separate project by Puppet Labs, which works well with Puppet.
- **Puppet Dashboard** is an open source web console for Puppet

□

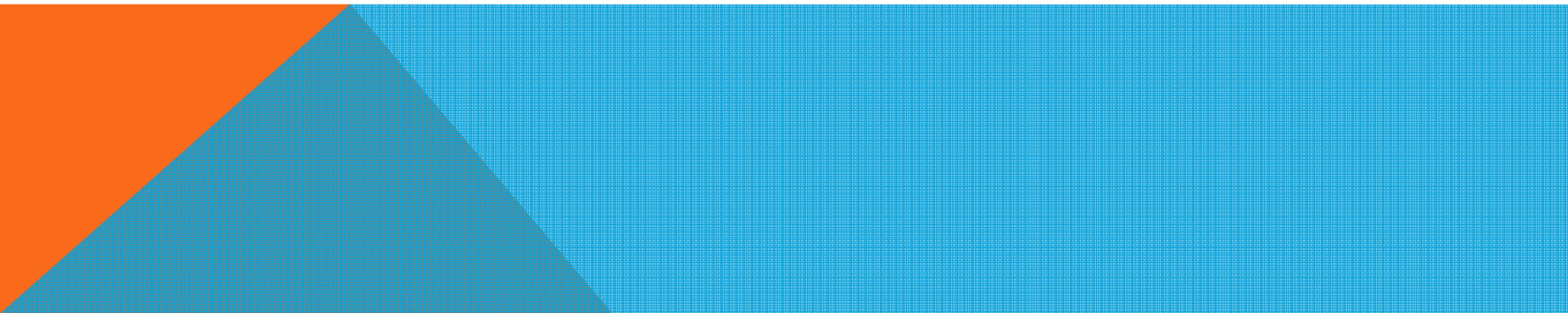


Puppet Community

The community has produced other tools and resources; the most noticeable ones are the following

- ▣ **The Foreman** is a systems lifecycle management tool that integrates perfectly with Puppet
- ▣ **Puppetboard** is a web frontend for PuppetDB
- ▣ **Kermit** is a web frontend for Puppet and Mcollective

A lot of community codes are also released as modules, which are reusable components that allow the management of any kind of application and software via Puppet.



Installing Puppet Master

Hostname: puppetmaster.example.com

```
apt-get update -y
```

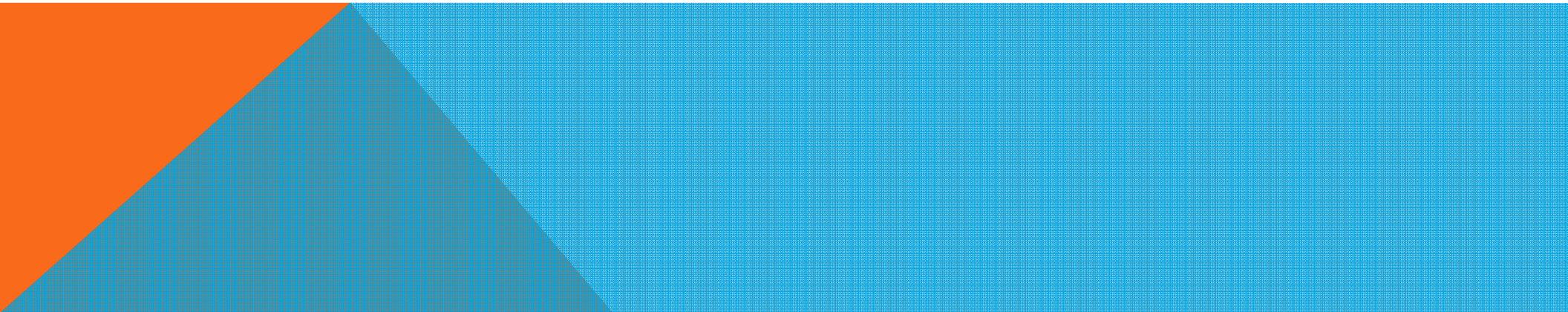
```
apt-get upgrade -y
```

```
wget http://apt.puppetlabs.com/puppetlabs-release-trusty.deb
```

```
dpkg -i puppetlabs-release-trusty.deb
```

```
apt-get update -y
```

```
apt-get -y install puppetmaster
```



Configuring Puppet Master

Hostname: puppet.example.com

1. Edit /etc/puppet/puppet.conf
[master]

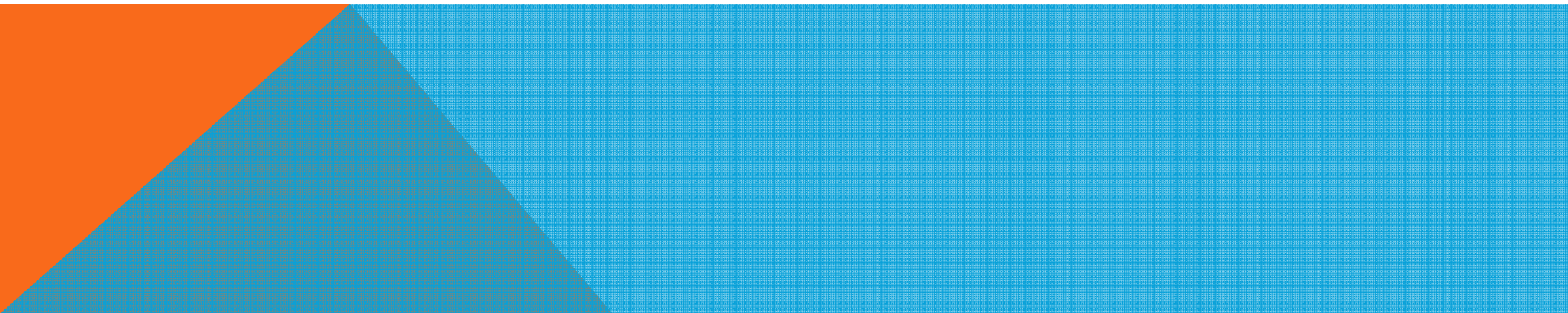
certname=puppet.example.com

2. Create a blank file

touch /etc/puppet/manifests/site.pp

3. Start the Puppet Master service

service puppetmaster start



Installing Puppet Agent

Hostname: puppetagent.example.com

```
apt-get update -y
```

```
apt-get upgrade -y
```

```
wget http://apt.puppetlabs.com/puppetlabs-release-trusty.deb
```

```
dpkg -i puppetlabs-release-trusty.deb
```

```
apt-get update -y
```

```
apt-get -y install puppet
```

