# Section 2: Architecting Puppet for scalability, redundancy and performance

**Estimated Video Length: 35 mins**

For a large infrastructure to run seamlessly, it is very important that it be planned and architected properly by understanding the requirements. In this section, we will look at how to configure Puppet server infrastructures by using different methodologies and components, add features to achieve redundancy and optimize performance.

## PuppetServer

In this section we are going to go through the typical installation of a puppetserver machine in a standalone configuration. For convenience we will refer to this machine as puppetca.

Start with an empty machine. Create Vagrantfile, mention that any VM method will work, we just want a blank machine. Your VM needs to have at least 2.5GB of Memory

SSH into machine

go to browser, yum.puppet.com, find the package appropriate to your release, show that apt is available.

pick the one for our machine puppetlabs-release-pc1-el-7.noarch.rpm, copy the url

yum install the rpm on the machine

Show the yum repo method.

Install puppetserver.

Generate CA certificates with /opt/puppetlabs/bin/puppet cert list -a

We will generate a certificate for our master with multiple dns names to allow for load balancing which we will cover in another section.

puppet certificate generate --dns-alt-names puppet,puppet.example.com,puppet.dev.exmaple.com puppet.example.com --ca-location local

puppet cert sign puppet.example.com --allow-dns-alt-names

puppet certificate find puppet.example.com --ca-location local

systemctl start puppetserver

set the hosts file up. puppet puppet.example.com to /etc/hosts

lsof -i :8140

Run against puppetserver locally

Bring up another node, test it and show that we can't talk yet, needs firewall.

Open up firewall

firewall-cmd --add-port=8140/tcp

firewall-cmd --add-port=8140/tcp --permanent

This is a single puppetserver configuration at this point.

## Load Balancing / CA configuration

In this configuration we are going to have one puppetca and two puppetmaster machines. The puppetca will act as a load balancer for our configuration.

puppetca 192.168.50.100

* puppetmaster1 192.168.50.201
* puppetmaster2 192.168.50.202

We'll start from a functioning puppetserver machine (refer to the previous section if you do not have a machine).

edit /etc/puppetlabs/puppetserver/conf.d/webserver.conf

webserver: {

access-log-config: /etc/puppetlabs/puppetserver/request-logging.xml

client-auth: want

ssl-host: 0.0.0.0

ssl-port: 8141

host: 0.0.0.0

port: 18140

}

Change the memory requirements.

/etc/sysconfig/puppetserver

JAVA\_ARGS

restart puppetserver

### Create the load balancing configuration

install apache and mod\_ssl

yum install httpd mod\_ssl

create puppet\_lb.conf in /etc/httpd/conf.d

Listen 8140

<VirtualHost \*:8140>

ServerName puppet.example.com

SSLEngine on

SSLProtocol -ALL +TLSv1 +TLSv1.1 +TLSv1.2

SSLCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:-LOW:-SSLv2:-EXP

SSLCertificateFile /etc/puppetlabs/puppet/ssl/certs/puppet.example.

com.pem

SSLCertificateKeyFile /etc/puppetlabs/puppet/ssl/private\_keys/

puppet.example.com.pem

SSLCertificateChainFile /etc/puppetlabs/puppet/ssl/ca/ca\_crt.pem

SSLCACertificateFile

/etc/puppetlabs/puppet/ssl/ca/ca\_crt.pem

# If Apache complains about invalid signatures on the CRL, you can

try disabling

# CRL checking by commenting the next line, but this is not

recommended.

SSLCARevocationFile

/etc/puppetlabs/puppet/ssl/ca/ca\_crl.pem

SSLVerifyClient optional

SSLVerifyDepth 1

# The `ExportCertData` option is needed for agent certificate

expiration warnings

SSLOptions +StdEnvVars +ExportCertData

# This header needs to be set if using a loadbalancer or proxy

RequestHeader unset X-Forwarded-For

RequestHeader set X-SSL-Subject %{SSL\_CLIENT\_S\_DN}e

RequestHeader set X-Client-DN %{SSL\_CLIENT\_S\_DN}e

RequestHeader set X-Client-Verify %{SSL\_CLIENT\_VERIFY}e

ProxyPassMatch ^/(puppet-ca/v[123]/.\*)$ balancer://puppetca/$1

ProxyPass / balancer://puppetworker/

ProxyPassReverse / balancer://puppetworker

<Proxy balancer://puppetca>

BalancerMember http://127.0.0.1:18140

</Proxy>

<Proxy balancer://puppetworker>

BalancerMember http://192.168.50.201:18140

BalancerMember http://192.168.50.202:18140

</Proxy>

</VirtualHost>

our virtual host is going to send ssl information in headers, we have to make a change to puppetserver to allow it accept these headers.

Create /etc/puppetlabs/puppetserver/conf.d/master.conf

master: {

allow-header-cert-info: true

}

Now create site.pp files on each master

/etc/puppetlabs/code/environments/production/manifests/site.pp

node default {

notify { "compiled on puppetmaster1": }

}

setsebool httpd\_can\_network\_connect 1

run against, see that it's bouncing between the two.

done.

## Master of Master

create a repository to configure your masters, talk about schemes for sharing ssl certs

how to make this work. define your nodes, make a drawing of what is needed.

mastter of master is a term used in enterprise deployments. In large deployments, you may need to create a large number of puppetmaster machines to address load concerns. To make this manageable, you create a module or role and profile that you can use to configure your puppetmasters. You then make your puppetmaster machines be clients of a higher level master, the master of masters. In most configurations this master of master will be the CA machine, but it can be an entirely different machine. the main thing is to make sure that the master of Master or MoM is not dependent on other machines. The MoM should be able to create new master machines without relying on any other master. The theory is that you bring up the MoM, and use the MoM to deploy all your other machines.

In our example, we'll modify our configuration to make the puppetca machine accessible directly.

First, modify the webserver.conf

comment out the ssl-port and ssl-host lines.

create a new apache configuration that only points at this machine.

call that puppet\_mom.conf

(copy in from /root)

restart puppetserver (to release port 8141)

restart httpd

create a configuration that works for you on the CA, edit manifest to include a role

we'll create a role, role::puppetmaster, which will include the beginning of a puppetmaster configuration.

run puppet on the puppetmasters to show the configuration working.

## Performance

i don't know what they were thinking here.

puppetserver java args

ulimits