How to securely deploy Elasticsearch 6 with TLS certificates from Puppet 6.

Introduction

The motivation for this paper is to guide the reader through the accurate steps to quickly deploy Elasticsearch version 6.4.2 with TLS certificates using Puppet version 6.

Requirements

If you see the following $ symbol on a command line to execute, what that means is that the command is executed as a regular user, i.e. the Ubuntu user. Ignore the leading $ and execute the rest of the command.

```
$ command to execute as a regular user
```

If you see a command line lead with the # symbol, then that means that the command is executed as the root user. This implies you need to elevate to the root user before running the command, e.g. with: `sudo su - root`.

```
# command to execute as the root user
```

VirtualBox

Go to: https://www.virtualbox.org/wiki/Downloads and download VirtualBox.

The author is running on Ubuntu 17.04, so following to this URL:
https://www.virtualbox.org/wiki/Linux_Downloads

For Ubuntu, double click on the .deb file, i.e. virtualbox-5.2_5.2.0-118431-Ubuntu-zesty_amd64.deb, and install VirtualBox on your local workstation.

Clean VirtualBox Networking

Run these two commands from a Terminal:

```
VBoxManage list natnetworks
VBoxManage list dhcpservers
```

```
Output:
NetworkName:    192.168.139-NAT
IP:             192.168.139.1
Network:        192.168.139.0/24
IPv6 Enabled:   No
IPv6 Prefix:    fd17:625c:f037:a88b::/64
DHCP Enabled:   Yes
Enabled:        Yes
loopback mappings (ipv4)
        127.0.0.1=2

NetworkName:    192.168.139-NAT
IP:             192.168.139.3
```

```
NetworkMask:    255.255.255.0
lowerIPAddress: 192.168.139.101
upperIPAddress: 192.168.139.254
Enabled:        Yes

NetworkName:    HostInterfaceNetworking-vboxnet0
IP:             172.20.0.3
NetworkMask:    255.255.255.0
lowerIPAddress: 172.20.0.101
upperIPAddress: 172.20.0.254
Enabled:        Yes

NetworkName:    HostInterfaceNetworking-vboxnet1
IP:             0.0.0.0
NetworkMask:    0.0.0.0
lowerIPAddress: 0.0.0.0
upperIPAddress: 0.0.0.0
Enabled:        No
```

Now, delete ALL of the pre-installed VirtualBox networks (one at a time following the syntax below):

```
VBoxManage natnetwork remove --netname <NetworkName_from_above>
VBoxManage natnetwork remove --netname 192.168.139-NAT
# repeat as many times as necessary to delete all of them.
```

```
VBoxManage dhcpserver remove --netname <DHCP_Server_NetworkName_from_above>
VBoxManage dhcpserver remove --netname 192.168.139-NAT
# repeat as many times as necessary to delete all of them.
```

Add VirtualBox Networking

Now, add the new VirtualBox networks so the that this guide work.

```
VBoxManage natnetwork add \
  --netname 192.168.139-NAT \
  --network "192.168.139.0/24" \
  --enable --dhcp on

VBoxManage dhcpserver add \
  --netname 192.168.139-NAT \
  --ip 192.168.139.3 \
  --lowerip 192.168.139.101 \
  --upperip 192.168.139.254 \
  --netmask 255.255.255.0 \
  --enable
```

```
VBoxManage hostonlyif create

VBoxManage hostonlyif ipconfig vboxnet0 \
  --ip 172.20.0.1 \
  --netmask 255.255.255.0

VBoxManage dhcpserver add \
  --ifname vboxnet0 \
  --ip 172.20.0.3 \
  --lowerip 172.20.0.101 \
  --upperip 172.20.0.254 \
  --netmask 255.255.255.0

VBoxManage dhcpserver modify \
  --ifname vboxnet0 \
  --enable
```

Vagrant

Go to: https://www.vagrantup.com/downloads.html, follow the appropriate link to your OS and 32 or 64 bit version representing your local workstation. Download.

For Ubuntu, double click on the .deb file, i.e. vagrant_2.0.1_x86_64.deb, and install Vagrant on your local system.

Two Servers, one Vagrantfile

The author highly recommends to create a directory structure that is easy to navigate and find your code. As an example, you could use something similar to:
${HOME}/Source_Code/Education/vagrant-machines/puppet_with_elasticsearch/

Go ahead and make this structure with the following command (inside a Terminal):

```
$ mkdir -p ${HOME}/Source_Code/Education/vagrant-machines/puppet_with_elasticsearch/
```

Inside of the puppet_with_elasticsearch directory, populate a new file with the exact name, "Vagrantfile". Case matters, uppercase the "V".

Vagrantfile:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
servers=[
  {
    :hostname => "puppetmaster",
    :ip => "192.168.139.50",
    :box => "centos/7",
    :box_ver => "1809.01",
    :ram => 4096,
    :cpu => 3,
    :local_folder_name => "puppet_master_guest/"
  },
  {
    :hostname => "esd001",
    :ip => "192.168.139.75",
    :box => "centos/7",
    :box_ver => "1809.01",
    :ram => 3072,
    :cpu => 2,
    :local_folder_name => "elasticsearch_guest/"
  }
]

Vagrant.configure(2) do |config|
  servers.each do |machine|
    config.vm.define machine[:hostname] do |node|
      node.vm.hostname = machine[:hostname]
      node.vm.box = machine[:box]
      node.vm.box_version = machine[:box_ver]
      # internal traffic only:
      node.vm.network "private_network", bridge: "HostInterfaceNetworking-vboxnet1", ip:
machine[:ip], :auto_config => "false", :netmask => "255.255.255.0"
      # external traffic:
      node.vm.network "public_network", bridge: "en0: Wi-Fi (AirPort)", auto_config: true
      node.vm.provision :shell, path: "bootstrap.sh"
```

```
      # originally I thought I wanted /etc/puppet, but in hindsight,
      # I realized I needed a folder for saves that my changes wouldn't affect.
      node.vm.synced_folder machine[:local_folder_name], "/var/tmp/puppet", create: true
      node.vm.provider "virtualbox" do |vb|
        vb.name = machine[:hostname]
        vb.memory = machine[:ram]
        vb.cpus = machine[:cpu]
        vb.gui = true
        vb.customize ["modifyvm", :id, "--cpuexecutioncap", "95"]
        vb.customize ["modifyvm", :id, "--vram", "32"]
        vb.customize ["modifyvm", :id, "--accelerate3d", "on"]
        vb.customize ["modifyvm", :id, "--ostype", "Debian_64"]
        vb.customize ["modifyvm", :id, "--boot1", "dvd"]
        vb.customize ["modifyvm", :id, "--boot2", "disk"]
        vb.customize ["modifyvm", :id, "--audio", "none"]
        vb.customize ["modifyvm", :id, "--clipboard", "hosttoguest"]
        vb.customize ["modifyvm", :id, "--draganddrop", "hosttoguest"]
        vb.customize ["modifyvm", :id, "--paravirtprovider", "kvm"]
      end
    end
  end
end
```

Save and write this file.

From a Terminal, change directory to:

```
$ cd ${HOME}/Source_Code/Education/vagrant-machines/puppet_with_elasticsearch/
```

Then run (inside the directory puppet_with_elasticsearch):

```
$ vagrant up
```

This will download the appropriate CentOS image and start both virtual machines.

Once running, through the VirtuaBox GUI, login as root. The password is “vagrant”. Edit the following file:

```
/etc/ssh/sshd_config
```

And change the line:

```
#PermitRootLogin prothibit-password
```

To:

```
PermitRootLogin yes
```

Then restart the ssh daemon:

```
# kill –HUP $(pgrep sshd)
```

Notice, you are on a Bridged adapter, this will open the instance to allow root to ssh in with the most unsecure password in the world. Only make this change (allowing root to login via SSH) if you require root SSH access. You can change the root user’s password, which is highly recommended.

Configure the Puppetmaster

One of tricks I learned was to use puppet ssh with the machine name, so run:

```
$ vagrant ssh "puppetmaster"
```

Elevate to root

```
$ sudo –i
```

Setup your /etc/hosts and /etc/hostname file:

```
# printf "puppetmaster.fortress.lan" > /etc/hostname
```

Create the Puppetlabs PC1 repo

```
# cat <</etc/yum.repos.d/puppetlabs-pc1.repo>EOF
[puppetlabs-pc1]
name=Puppet Labs PC1 Repository el 7 - $basearch
baseurl=http://yum.puppetlabs.com/el/7/PC1/$basearch
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-puppetlabs-PC1
       file:///etc/pki/rpm-gpg/RPM-GPG-KEY-puppet-PC1
enabled=1
gpgcheck=1

[puppetlabs-pc1-source]
name=Puppet Labs PC1 Repository el 7 - Source
baseurl=http://yum.puppetlabs.com/el/7/PC1/SRPMS
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-puppetlabs-PC1
       file:///etc/pki/rpm-gpg/RPM-GPG-KEY-puppet-PC1
failovermethod=priority
enabled=0
gpgcheck=1
EOF
```

Install puppetserver

```
# yum clean all
# yum install puppetserver -y
```

Install modules for dependencies

```
# puppet module install puppetlabs-stdlib --version 5.1.0
# puppet module install richardc-datacat --version 0.6.2
# puppet module install puppetlabs-java --version 3.2.0
# puppet module install elastic-elastic_stack --version 6.2.4
# puppet module install elastic-logstash --version 6.1.4
# puppet module install elastic-kibana --version 6.3.1
# puppet module install pcfens-filebeat --version 3.3.2
# puppet module install elastic-elasticsearch --version 6.3.3
```

Populate your puppet.conf file

```
# cat <</etc/puppetlabs/puppet/puppet.conf>EOF
# This file can be used to override the default puppet settings.
# See the following links for more details on what settings are available:
# - https://docs.puppetlabs.com/puppet/latest/reference/config_important_settings.html
# - https://docs.puppetlabs.com/puppet/latest/reference/config_about_settings.html
# - https://docs.puppetlabs.com/puppet/latest/reference/config_file_main.html
# - https://docs.puppetlabs.com/puppet/latest/reference/configuration.html
[master]
vardir = /opt/puppetlabs/server/data/puppetserver
```

```
logdir = /var/log/puppetlabs/puppetserver
rundir = /var/run/puppetlabs/puppetserver
pidfile = /var/run/puppetlabs/puppetserver/puppetserver.pid
codedir = /etc/puppetlabs/code
dns_alt_names=puppetmaster.fortress.lan,puppetmaster,puppet

[main]
certname = puppetmaster.fortress.lan
server = puppetmaster.fortress.lan
environment = production
runinterval = 1h
EOF
```

Populate your manifest

```
# cat << /etc/puppetlabs/code/environments/production/manifests/site.pp>EOF
node 'esd001.fortress.lan' {
 class { 'java' :
    package => 'java-1.8.0-openjdk-devel',
  }


  class { 'elasticsearch':
    version => '6.4.2',
    autoupgrade => true,
    restart_on_change => false,
    api_host => 'localhost',
    api_port => 9200,
    api_protocol => 'https',
    api_timeout => 20,
    security_plugin => 'x-pack',
    validate_tls => true
  }


  elasticsearch::instance { 'esd001.fortress.lan':
    config => {
      'bootstrap.memory_lock' => false,
      'cluster.name' => 'fortress_lan',
      'discovery.zen.minimum_master_nodes' => 1,
      'discovery.zen.ping.unicast.hosts' => [ 'esd001.fortress.lan' ],
      'gateway.expected_data_nodes' => '1',
      'gateway.recover_after_time' => '10m',
      'http.cors.allow-origin' => '*',
      'http.cors.enabled' => true,
      'indices.fielddata.cache.size' => '30%',
      'network.bind_host' => '0.0.0.0',
      'network.host' => [ "192.168.139.75", '127.0.0.1' ],
      'network.publish_host' => '192.168.139.75',
      'node.attr.environment' => 'production',
      'node.data' => true,
      'node.ingest' => true,
      'node.master' => true,
      'node.ml' => false,
      'node.name' => "$hostname",
      'path.data' => [ "/elast/data0/$hostname" ],
      'path.logs' => "/var/log/elasticsearch/$hostname",
      'xpack.ml.enabled' => false,
      'xpack.security.enabled' => true,
      'xpack.security.http.ssl.client_authentication' => 'none',
      'xpack.security.http.ssl.enabled' => true,
      'xpack.security.http.ssl.key' => "/etc/elasticsearch/$fqdn/certs/dev.fortress.lan.key",
      'xpack.security.http.ssl.certificate' =>
"/etc/elasticsearch/$fqdn/certs/dev.fortress.lan.crt",
      'xpack.security.http.ssl.certificate_authorities' =>
"/etc/elasticsearch/$fqdn/certs/signing-ca-chain.pem",
      'xpack.security.transport.ssl.enabled' => true,
      'xpack.security.transport.ssl.verification_mode' => 'certificate',
      'xpack.ssl.client_authentication' => 'none',
      'xpack.ssl.verification_mode' => 'none'
    },
```

```
    jvm_options => [
      '-Xms1g',
      '-Xmx1g',
      '-Des.scripting.exception_for_missing_value=true'
    ],
    status => 'enabled'
  }


  ## x-pack comes pre-installed with this version of Elasticsearch 6.x.
  # if you try to install it, it will give a lot of Warnings.
  # elasticsearch::plugin { 'x-pack': }
  elasticsearch::plugin { 'repository-s3':
    ensure => 'present',
  }


  elasticsearch::template { 'templatename':
    content => '{ "index_patterns":"*", "settings": { "number_of_replicas": 1 }}'
  }

  elasticsearch::user { 'monitoring_account':
    password => 'fubar',
    roles    => ['logstash', 'kibana'],
  }

  class { 'kibana' :
    config => {
      'elasticsearch.url' => 'https://127.0.0.1:9200',
      'elasticsearch.username' => 'kibana',
      'elasticsearch.password' => 'kibanapassword',
      'elasticsearch.ssl.certificate' => "/etc/kibana/certs/dev.fortress.lan.crt",
      'elasticsearch.ssl.key' => "/etc/kibana/certs/dev.fortress.lan.key",
      'elasticsearch.ssl.certificateAuthorities' => "/etc/kibana/certs/signing-ca-chain.pem",
      'elasticsearch.ssl.keyPassphrase' => 'changeit',
      'elasticsearch.ssl.verificationMode' => 'certificate',
      'logging.dest' => '/var/log/kibana/kibana.stdout',
      'logging.quiet' => true,
      'logging.verbose' => false,
      'ops.interval' => '15000',
      'server.host'  => '192.168.139.75',
      'server.name' => 'kibana001',
      'server.port'  => '5601',
      'server.defaultRoute' => '/app/monitoring',
      'server.ssl.enabled' => true,
      'server.ssl.key' => "/etc/kibana/certs/dev.fortress.lan.key",
      'server.ssl.certificate' => "/etc/kibana/certs/dev.fortress.lan.crt",
      'xpack.security.encryptionKey' => 'something_at_least_32_characters_if_not_longer',
      'xpack.security.sessionTimeout' => '600000'
    },
    status => 'enabled'
  }
}
EOF
```

Now that puppetmaster is up, go ahead and follow my previous guide, ***How to setup an enterprise level Certificate Authority using openssl***, and create your wildcard cert for, CN=*.fortress.lan. Be mindful, you can change this to your domain.

Setup the Elasticsearch data node

Now, login to your second node, esd001. I named it this for “Elasticsearch Data 001”. It could be anything you wish to change it to. Also, this one node will be setup as an all in one elasticsearch node, meaning, it will be the master, coordinator and data node.

```
$ vagrant ssh “esd001”
```

Eleavate to root

```
$ sudo –i
```

Setup your /etc/hosts and /etc/hostname file:

```
# printf “esd001.fortress.lan” > /etc/hostname
```

```
# cat <</etc/hosts>EOF
127.0.0.1   localhost   localhost.localdomain   localhost4  localhost4.localdomain4
::1         localhost   localhost.localdomain   localhost6  localhost6.localdomain6
##
#
192.168.139.50         puppetmaster.fortress.lan     puppetmaster   puppet
192.168.139.75         esd001.fortress.lan
EOF
```

Create the Puppetlabs PC1 repo

```
# cat <</etc/yum.repos.d/puppetlabs-pc1.repo>EOF
[puppetlabs-pc1]
name=Puppet Labs PC1 Repository el 7 - $basearch
baseurl=http://yum.puppetlabs.com/el/7/PC1/$basearch
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-puppetlabs-PC1
       file:///etc/pki/rpm-gpg/RPM-GPG-KEY-puppet-PC1
enabled=1
gpgcheck=1

[puppetlabs-pc1-source]
name=Puppet Labs PC1 Repository el 7 - Source
baseurl=http://yum.puppetlabs.com/el/7/PC1/SRPMS
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-puppetlabs-PC1
       file:///etc/pki/rpm-gpg/RPM-GPG-KEY-puppet-PC1
failovermethod=priority
enabled=0
gpgcheck=1
EOF
```

Install the puppet agent

```
# yum install –y puppet-agent
```

Run puppet

```
# puppet agent –tv
```

On the puppetmaster, you might have to sign the SSL cert for esd001.fortress.lan

```
# puppet cert list –a
# puppet cert sign esd001.fortress.lan
```

Now, manually create the directory and copy your 3 TLS certificates to said directory.

```
# mkdir –p /etc/elasticsearch/$(hostname)/certs
```

Copy the TSL certs to the above directory. Then run:

```
# chmod 0750 /etc/elasticsearch/$(hostname)/certs
# chmod 0644 /etc/elasticsearch/$(hostname/certs/*
# chown –R elasticsearch: /etc/elasticsearch/$(hostname)/certs
# mkdir –p /elast/data0
# chown –R elasticsearch: /elast/data0
# systemctl restart elasticsearch-$(hostname).service
```

Monitor the log file for errors on esd001.

```
# tail -n 50 -f /var/log/elasticsearch/$(hostname)/fortress_lan.log
```

At this point, you should be done.

To do:

- Create puppet module to push certs to esd001 and all nodes.
- Add in Roles and Profiles concept to puppetserver.

Conclusion

By following this guide, the reader has setup two nodes. One with a puppet server, and the other with a full elasticsearch stack. The reader can now further this and expand to have more nodes from the site.pp manifest file. There is a lot going on here. From the Elasticsearch side, both the transport and management of data is encrypted with the TLS certificate. From Kibana, the web service and communication to the Elasticsearch server is encrypted with the TLS certificate. My whole motivation for putting this together in this fashion is so that I only have to manage three certificates for an entire domain. E.g. 10+ coordinator nodes, 3 master nodes, and 22 data nodes. There is always more to be done, and I still need to create a separate module to push the correct three certs out to all required nodes. The other cool concept with this is, I can use the same certificates on all of my application servers, and they can talk directly to my Elasticsearch stack without failure because they all trust the same Root certificate. I don't remember when the change was put in to allow the wildcard certs with the Common Name (CN), but it was one of the best decisions ever made regarding PKI certificates.