
OpenShift 3 TLS Certificates and Proxy Deep Dive
Presenter: Ayush Garg
Red Hat

Why TLS Certificates are used?

Specific to OpenShift:

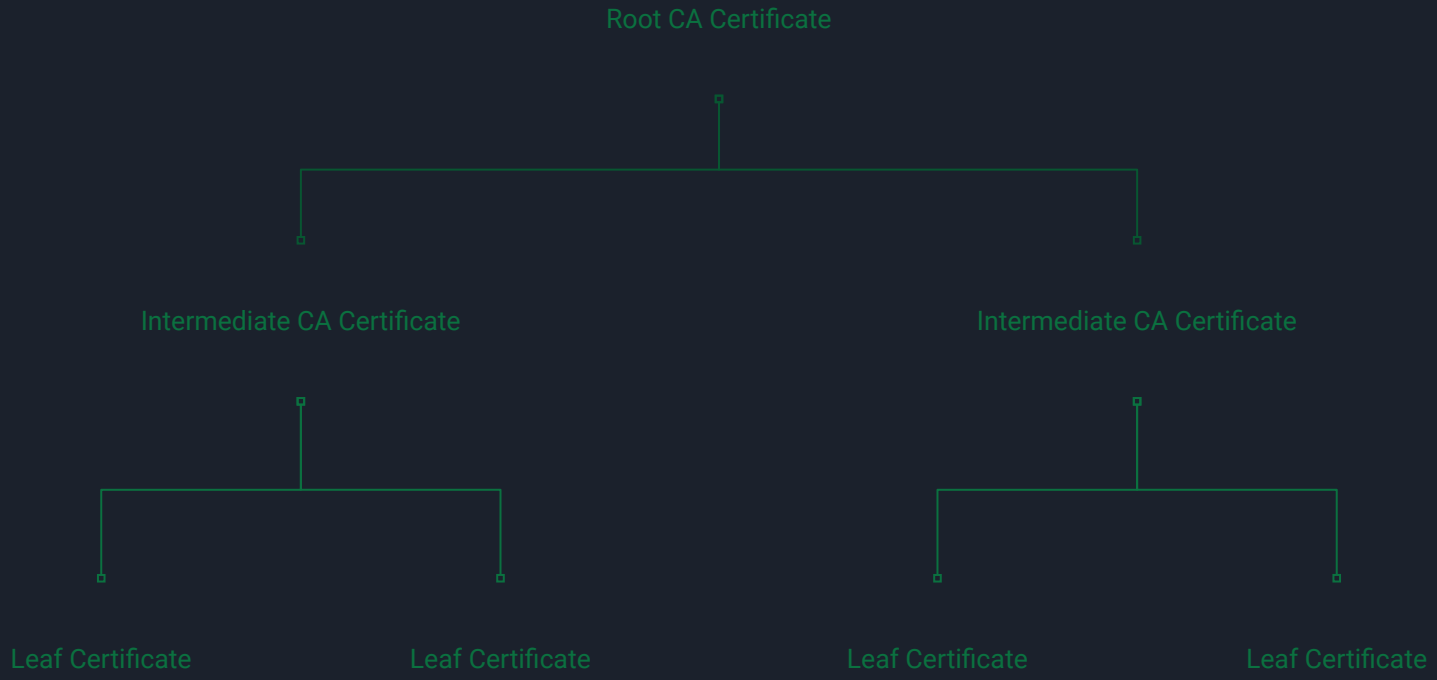
- Encrypts the sensitive data that gets transmitted
- Authentication
- Authorization

```
$ oc api-versions | grep -i certificates
certificates.k8s.io/v1beta1
```

Authentication:

- Server Authentication
- Client Authentication

Server Cert Date → Signed by a CA that client recognize → CA validate cert sign.→ DN of server



Red Hat OpenShift

Red Hat

Basics of TLS Certificates

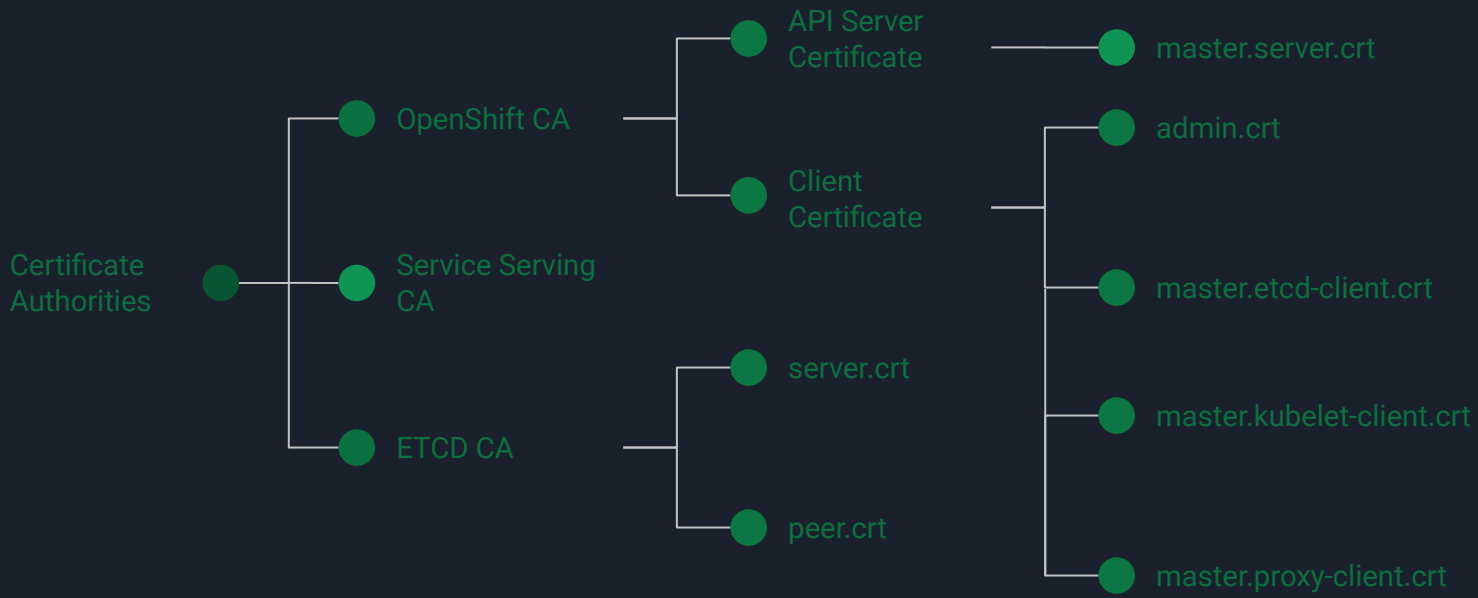
Types of TLS Certificates:

- Single Domain (/etc/origin/master/admin.crt)
- Wildcard (/etc/origin/master/openshift-router.crt)
- Multidomain (/etc/origin/master/master.server.crt)

X509v3 extensions:

- Key Usage
- Extended Key Usage
- Basic Constraints
- Subject Alternative Name
- Authority Key Identifier
- Subject Key Identifier

[man pages: x509, x509v3_config]



Red Hat OpenShift

Red Hat

Categorize by components

- Master
- ETCD
- Nodes
- Router
- Registry
- Service Serving Certificate

Master Certificates

OpenShift CA:

- The root CA for OpenShift can be found at "**/etc/origin/master/ca.crt**".
- Almost all of the component's certificate are signed by this CA.

Internal and External API:

- The "**master.server.crt**" gets used for internal API communication.
- The external API also uses the same certificate by default if custom is not specified.
  - Configure named certificate for Web-Console during or after deployment
  - Problems with masterURL, masterPublicURL, and Named Certificates in OpenShift

System:admin user:

- The "**/etc/origin/master/admin.crt**" is used for the admin access without any password.
- The users like system:admin can also be created.
  - Create-user-like-system-admin-OCP

Red Hat

etcdClientInfo:

- The “**etcdClientInfo**” parameter is present inside “master-config.yaml” file.
- It points to “**master.etcd-client.crt**” which is client certificate for the API server to talk to etcd.

kubeletClientInfo:

- This “**master.kubelet-client.crt**” certificate gets used while running the “**oc exec**”, “**oc rsh**” and “**oc logs**” command. The request first hits the API with certificate and then the node where the node’s “**/etc/origin/node/certificates/kubelet-server-current.pem**” gets served.
- If the certificate is commented out in “master-config.yaml”, then the above commands will fail.

```
$ oc logs ruby-ex-1-9gphs
Error from server (Forbidden): Forbidden (user=system:anonymous, verb=get, resource=nodes,
subresource=proxy) ( pods/log ruby-ex-1-9gphs)

$ oc exec ruby-ex-1-9gphs -- ls
error: unable to upgrade connection: Forbidden (user=system:anonymous, verb=create,
resource=nodes, subresource=proxy)
```

Controller:

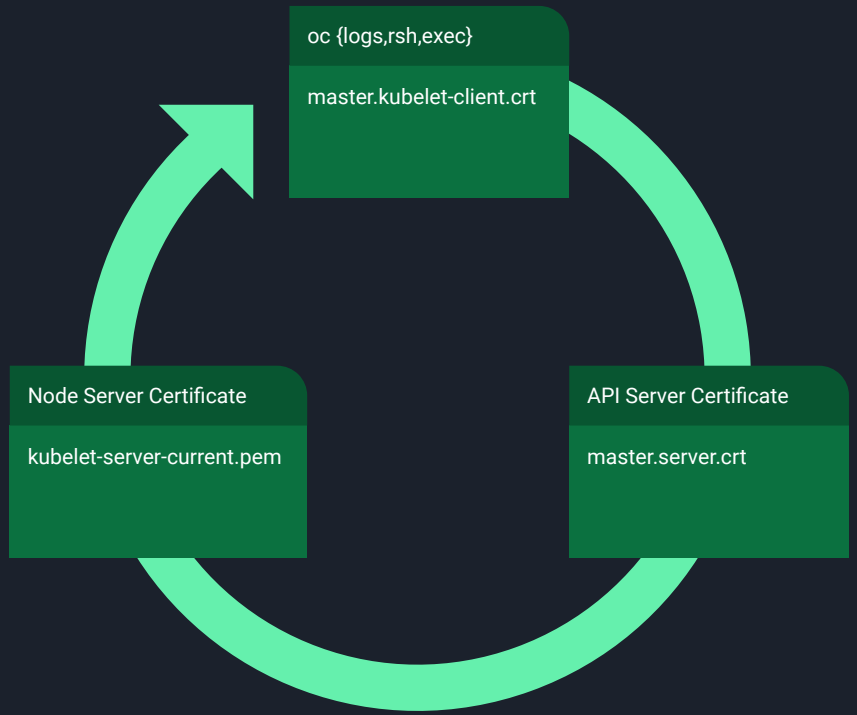
- All the CSRs are approved by the controller only.
- The "**controllerArguments**" option inside the "master-config.yaml" specifies the path to CA certificate and key for signing the CSRs.

Service Account Tokens:

- The tokens for service account as a secret are generated by controller using the "**serviceaccounts.private.key**" and "**serviceaccounts.public.key**".

openshift-master.crt/kubeconfig:

- The "**openshift-master.kubeconfig**" is used by the controllers service to talk to the master API so it can talk to the local master (loopback) rather than the load balanced master nodes since the controllers are going to talk to the API a lot. The "**openshift-master.kubeconfig**" contains the "**openshift-master.crt**" and "**openshift-master.key**" data in base64 encoded form and specified in "**master-config.yaml**" as "**openshiftLoopbackKubeConfig**".



Red Hat OpenShift

Red Hat

ETCD Certificates

ETCD CA:

- The root CA for ETCD can be found at "**/etc/etcd/ca.crt**".

Server Cert:

- The "**/etc/etcd/server.crt**" gets served by ETCD when API tries to communicate on port 2379.

Peer Cert:

- The ETCD members communicate with each other on port 2380 with "**/etc/etcd/peer.crt**".

ETCD Security:

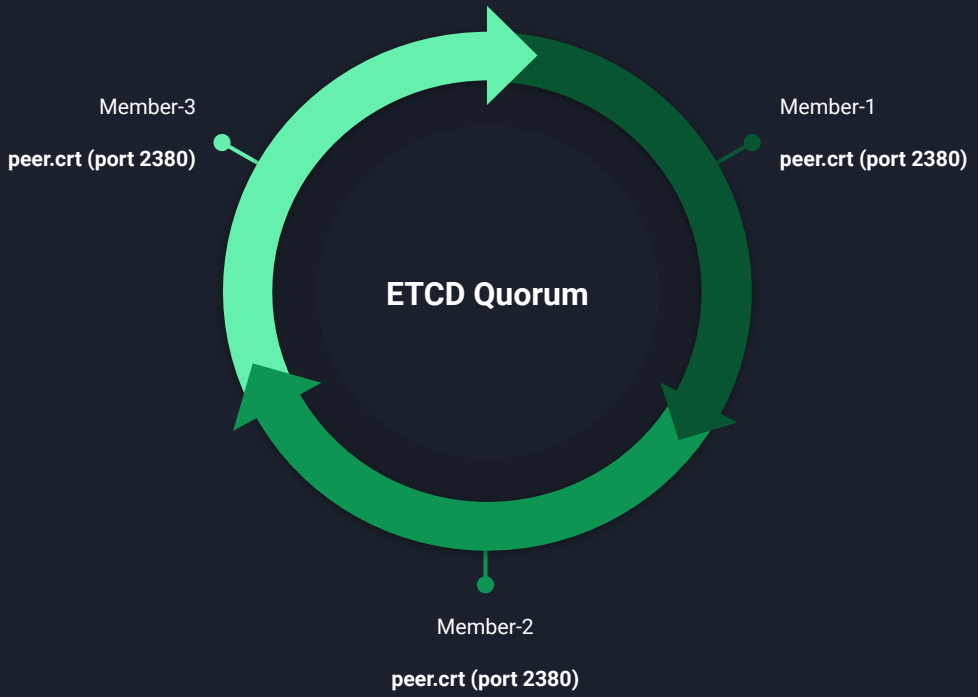
- $ cat /etc/etcd/etcd.conf | grep AUTH

  ETCD_CLIENT_CERT_AUTH=true

  ETCD_PEER_CLIENT_CERT_AUTH=true

$ curl -kv --cacert /etc/etcd/ca.crt https://10.x.x.x:2379/version

$ curl -kv --cacert /etc/etcd/ca.crt --cert /etc/etcd/server.crt --key /etc/etcd/server.key https://10.x.x.x:2379/version



Red Hat OpenShift

Red Hat

oc command

master.etcd-client.crt

ETCD Server Certificate

server.crt (port 2379)

API Server Certificate

master.server.crt

Red Hat OpenShift

Red Hat

Create ETCD Certificates Manually

- Create some environment variables
- Create the directory to store the configuration and certificates
- Create the server certificate request and sign it: (server.csr and server.crt)
- Create the peer certificate request and sign it: (peer.csr and peer.crt)
- Copy the current etcd configuration and ca.crt files from the current node as examples to modify later

https://docs.openshift.com/container-platform/3.11/admin_guide/assembly_replace-etcd-member.html#manually-adding-etcd-host_replace-etcd-member

Red Hat OpenShift

Red Hat

Node Certificates

Kubelet Client Cert:

- Kubelet uses client certificate "**/etc/origin/node/certificates/kubelet-client-current.pem**" to communicate with API for updating the node status.
  X509v3 Extended Key Usage:
  TLS Web Client Authentication

Kubelet Server Cert:

- The "**/etc/origin/node/certificates/kubelet-server-current.pem"** certificate gets served whenever an "**oc exec**", "**oc rsh**" and "**oc logs**" command executed over the pod running on the particular node.
  X509v3 Extended Key Usage:
  TLS Web Server Authentication

Red Hat OpenShift

Red Hat

Registry Certificate

Registry Certificate:

- The registry certificate always remain self-signed as it contains the IP address of the service as the common name.
- The custom certificate can be applied to the registry route.
- Default Registry certificate gets served by the route as the route remains in passthrough mode.

```
$ oc get secret registry-certificates -n default -o jsonpath='{.data.registry\.crt}' | base64 -d | openssl x509 -text -noout
```

Red Hat OpenShift

Red Hat

Router Certificate

Router Certificate:

- A self-signed wild-card router certificate gets created by default as a secret if the custom certificate is not specified at the time of deployment.

```
$ oc get secret router-certs -n default -o jsonpath='{.data.tls\.crt}' | base64 -d | openssl x509 -text -noout
```

Secured Routes:

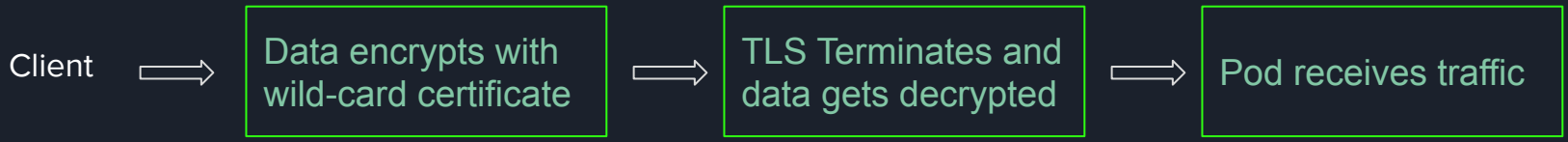
- Edge Termination
- Passthrough Termination
- Re-encryption Termination

https://docs.openshift.com/container-platform/3.11/architecture/networking/routes.html#secured-routes

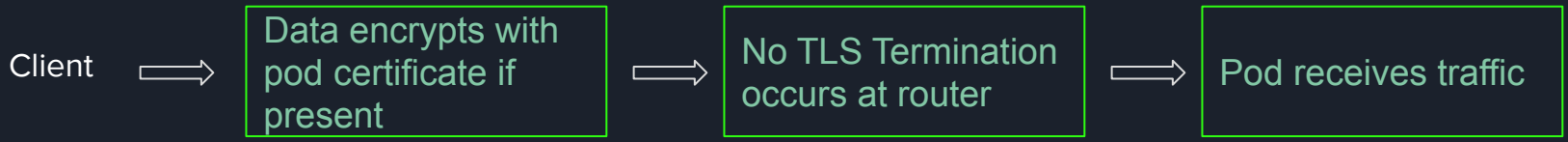
Red Hat OpenShift

Red Hat

Edge Termination:



Passthrough Termination:



Re-encryption Termination:

Red Hat OpenShift

Red Hat

Renew and Configure Router Certificates

Redeploy Default Router Cert:

- To redeploy router certificates, change to the playbook directory and run the following playbook, specifying your inventory file:

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook -i <inventory_file> playbooks/openshift-hosted/redeploy-router-certificates.yml
```

Configure Custom Router Cert:

- Edit the Ansible inventory file to set the openshift_master_overwrite_named_certificates=true
- Specify the path to the certificate using the openshift_hosted_router_certificate parameter.

```
openshift_hosted_router_certificate={"certfile": "/path/on/host/to/app-crt-file", "keyfile": "/path/on/host/to/app-key-file", "cafile": "/path/on/host/to/app-ca-file"}
```

- Run the same playbook to deploy the certificate.

Red Hat OpenShift

Red Hat

Router and Registry Cert Documentation

- REDEPLOYING REGISTRY CERTIFICATES ONLY
- REDEPLOYING ROUTER CERTIFICATES ONLY
- REDEPLOYING REGISTRY CERTIFICATES MANUALLY
- REDEPLOYING ROUTER CERTIFICATES MANUALLY
- Retrofit Custom Router Certificates into a Cluster

Red Hat OpenShift

Red Hat

Service Serving Certificate

Service Serving Certificate Secrets:

- Service serving certificate secrets are intended to support complex middleware applications that need out-of-the-box certificates. It has the same settings as the server certificates generated by the administrator tooling for nodes and masters.
- This certificate secret present by default in openshift-console, default, openshift-web-console, etc namespaces.
- The service needs to be annotated with "**service.alpha.openshift.io/serving-cert-secret-name**" in order to create the secret.
- All the certificates in the secret gets signed by "**/etc/origin/master/service-signer.crt**" as specified in "master-config.yaml" with "**serviceServingCert**" parameter.

  Service Serving Certificate Secrets

Red Hat OpenShift

Red Hat

Create API Certificate Manually

master.server.crt:

- Obtain the "**CN**" and "**X509v3 Subject Alternative Name**" first.

```
$ openssl x509 -in /etc/origin/master/master.server.crt -text -noout
```

- Create the certificate by specifying the obtained values.

```
$ oc adm ca create-server-cert --signer-cert=/etc/origin/master/ca.crt \
--signer-key=/etc/origin/master/ca.key --signer-serial=/etc/origin/master/ca.serial.txt \
--hostnames='<all-the-hostnames>' --cert=/path/to-save/master.server.crt \
--key=/path/to-save/master.server.key
```

- Run the same playbook to deploy the certificate.

Red Hat OpenShift

Red Hat

Why Proxy?

- Network Security
- Deny Direct Internet Access
- Proxying Docker Pull
- Configuring S2I Builds for Proxies
- Git Repository Access

Proxy Variables for Inventory

openshift_http_proxy:

- This variable specifies the **HTTP_PROXY** environment variable for masters and the Docker daemon. openshift_http_proxy=http://10.74.254.232:3128

openshift_https_proxy

- This variable specifies the **HTTPS_PROXY** environment variable for masters and the Docker daemon. openshift_https_proxy=http://10.74.254.232:3128

openshift_no_proxy:

- This variable is used to set the NO_PROXY environment variable for masters and the Docker daemon. Provide a comma-separated list of host names, domain names, or wildcard host names that do not use the defined proxy. By default, this list is augmented with the list of all defined OpenShift Container Platform host names. openshift_no_proxy='<node-IP>,<node-hostname>'

  Configuring Global Proxy Options

Red Hat OpenShift

Red Hat

Configuration Files for Proxy

master-config.yaml

```
admissionConfig:
  pluginConfig:
    BuildDefaults:
      configuration:
        apiVersion: v1
        env:
        - name: HTTP_PROXY
          value: http://10.74.254.232:3128
        - name: HTTPS_PROXY
          value: http://10.74.254.232:3128
        - name: NO_PROXY
          value: .cluster.local,.svc,10.74.176.208,10.74.176.47,10.74.177.203,10.74.177.208,10.74.177.87,10.74.178.107,10.74.178.148,10.74.178.19,127.0.0.1,16
9.254.169.254,172.30.0.0/16,172.30.0.1,infra-0.aygargocp311.lab.pnq2.cee.redhat.com,infra-1.aygargocp311.lab.pnq2.cee.redhat.com,infra-2.aygargocp311.lab.pnq2
.cee.redhat.com,lb-0.aygargocp311.lab.pnq2.cee.redhat.com,localhost,master-0.aygargocp311.lab.pnq2.cee.redhat.com,master-1.aygargocp311.lab.pnq2.cee.redhat.co
m,master-2.aygargocp311.lab.pnq2.cee.redhat.com,node-0.aygargocp311.lab.pnq2.cee.redhat.com,openshift.internal.aygargocp311.lab.pnq2.cee.redhat.com
        - name: http_proxy
          value: http://10.74.254.232:3128
        - name: https_proxy
          value: http://10.74.254.232:3128
        - name: no_proxy
          value: .cluster.local,.svc,10.74.176.208,10.74.176.47,10.74.177.203,10.74.177.208,10.74.177.87,10.74.178.107,10.74.178.148,10.74.178.19,127.0.0.1,16
9.254.169.254,172.30.0.0/16,172.30.0.1,infra-0.aygargocp311.lab.pnq2.cee.redhat.com,infra-1.aygargocp311.lab.pnq2.cee.redhat.com,infra-2.aygargocp311.lab.pnq2
.cee.redhat.com,lb-0.aygargocp311.lab.pnq2.cee.redhat.com,localhost,master-0.aygargocp311.lab.pnq2.cee.redhat.com,master-1.aygargocp311.lab.pnq2.cee.redhat.co
m,master-2.aygargocp311.lab.pnq2.cee.redhat.com,node-0.aygargocp311.lab.pnq2.cee.redhat.com,openshift.internal.aygargocp311.lab.pnq2.cee.redhat.com
        gitHTTPProxy: http://10.74.254.232:3128
        gitHTTPSProxy: http://10.74.254.232:3128
        gitNoProxy: .cluster.local,.svc,10.74.176.208,10.74.176.47,10.74.177.203,10.74.177.208,10.74.177.87,10.74.178.107,10.74.178.148,10.74.178.19,127.0.0.1
,169.254.169.254,172.30.0.0/16,172.30.0.1,infra-0.aygargocp311.lab.pnq2.cee.redhat.com,infra-1.aygargocp311.lab.pnq2.cee.redhat.com,infra-2.aygargocp311.lab.p
nq2.cee.redhat.com,lb-0.aygargocp311.lab.pnq2.cee.redhat.com,localhost,master-0.aygargocp311.lab.pnq2.cee.redhat.com,master-1.aygargocp311.lab.pnq2.cee.redhat
.com,master-2.aygargocp311.lab.pnq2.cee.redhat.com,node-0.aygargocp311.lab.pnq2.cee.redhat.com,openshift.internal.aygargocp311.lab.pnq2.cee.redhat.com
```

/etc/sysconfig/atomic-openshift-node:

```
# populated by openshift-ansible
OPTIONS=
DEBUG_LOGLEVEL=2
IMAGE_VERSION=v3.11
HTTP_PROXY=http://10.74.254.232:3128
HTTPS_PROXY=http://10.74.254.232:3128
NO_PROXY=.cluster.local,.svc,10.74.176.208,10.74.176.47,10.74.177.203,10.74.177.208,10.74.177.87,10.74.178.107,10.74.178.148,10.74.178.19,127.0.0.1,169.254.169.254,172.30.0.0/16,172.30.0.1,infra-0.aygargocp311.lab.pnq2.cee.redhat.com,infra-1.aygargocp311.lab.pnq2.cee.redhat.com,infra-2.aygargocp311.lab.pnq2.cee.redhat.com,lb-0.aygargocp311.lab.pnq2.cee.redhat.com,localhost,master-0.aygargocp311.lab.pnq2.cee.redhat.com,master-1.aygargocp311.lab.pnq2.cee.redhat.com,master-2.aygargocp311.lab.pnq2.cee.redhat.com,node-0.aygargocp311.lab.pnq2.cee.redhat.com,openshift.internal.aygargocp311.lab.pnq2.cee.redhat.com,172.30.0.0/16,10.128.0.0/14
KUBECONFIG=/etc/origin/node/bootstrap.kubeconfig
BOOTSTRAP_CONFIG_NAME=node-config-master
```

/etc/sysconfig/docker:

```
HTTP_PROXY='http://10.74.254.232:3128'
HTTPS_PROXY='http://10.74.254.232:3128'
NO_PROXY='.cluster.local,.svc,10.74.176.208,10.74.176.47,10.74.177.203,10.74.177.208,10.74.177.87,10.74.178.107,10.74.178.148,10.74.178.19,127.0.0.1,169.254.169.254,172.30.0.0/16,172.30.0.1,infra-0.aygargocp311.lab.pnq2.cee.redhat.com,infra-1.aygargocp311.lab.pnq2.cee.redhat.com,infra-2.aygargocp311.lab.pnq2.cee.redhat.com,lb-0.aygargocp311.lab.pnq2.cee.redhat.com,localhost,master-0.aygargocp311.lab.pnq2.cee.redhat.com,master-1.aygargocp311.lab.pnq2.cee.redhat.com,master-2.aygargocp311.lab.pnq2.cee.redhat.com,node-0.aygargocp311.lab.pnq2.cee.redhat.com'
```

Few Resources

- Master KCS - Understanding Openshift Container Platform 3.x Certificates
- How to list all OpenShift TLS certificate expire date?
- How to enable OpenShift Node TLS Bootstrapping auto-approver
- Creating New Configuration Files

Red Hat OpenShift

Red Hat

Thank You!