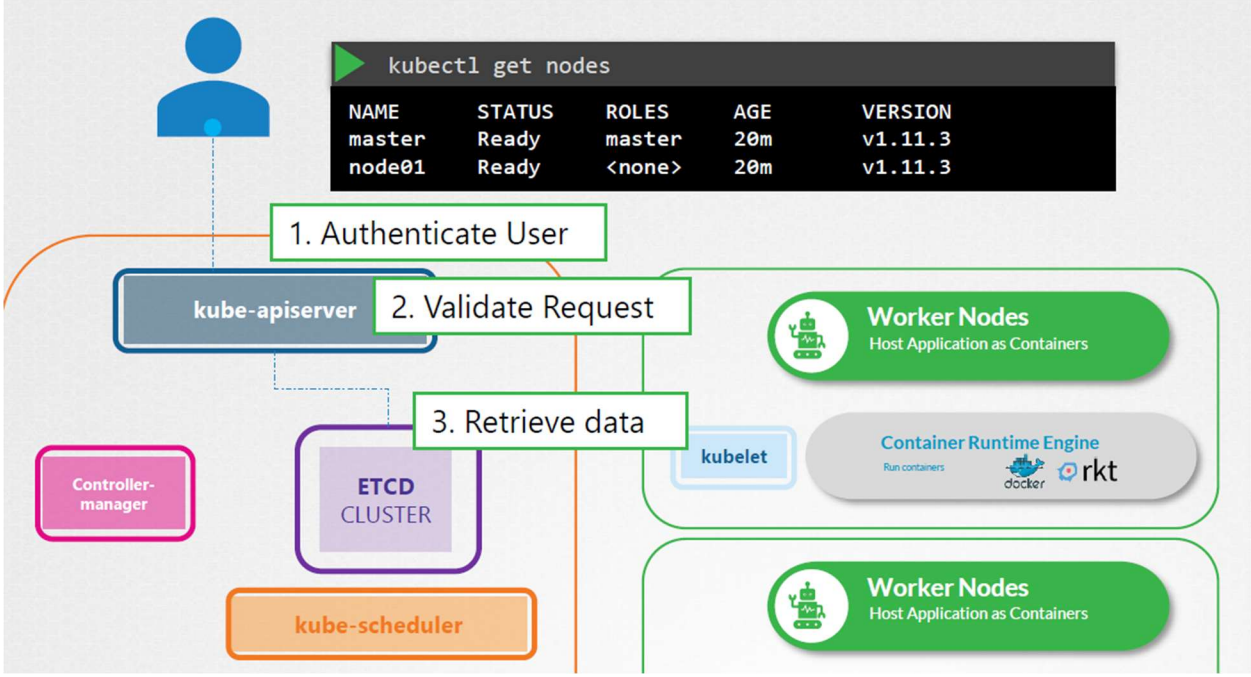
API – SERVER

In this lecture, we will talk about the kube-apiserver in Kubernetes. Earlier, we discussed that the kube-apiserver is the primary management component in Kubernetes. When you run a `kubectl` command, the `kubectl` utility is, in fact, reaching to the kube-apiserver. The kube-apiserver first authenticates the request and validates it. It then retrieves the data from the etcd cluster and responds back with the requested information.

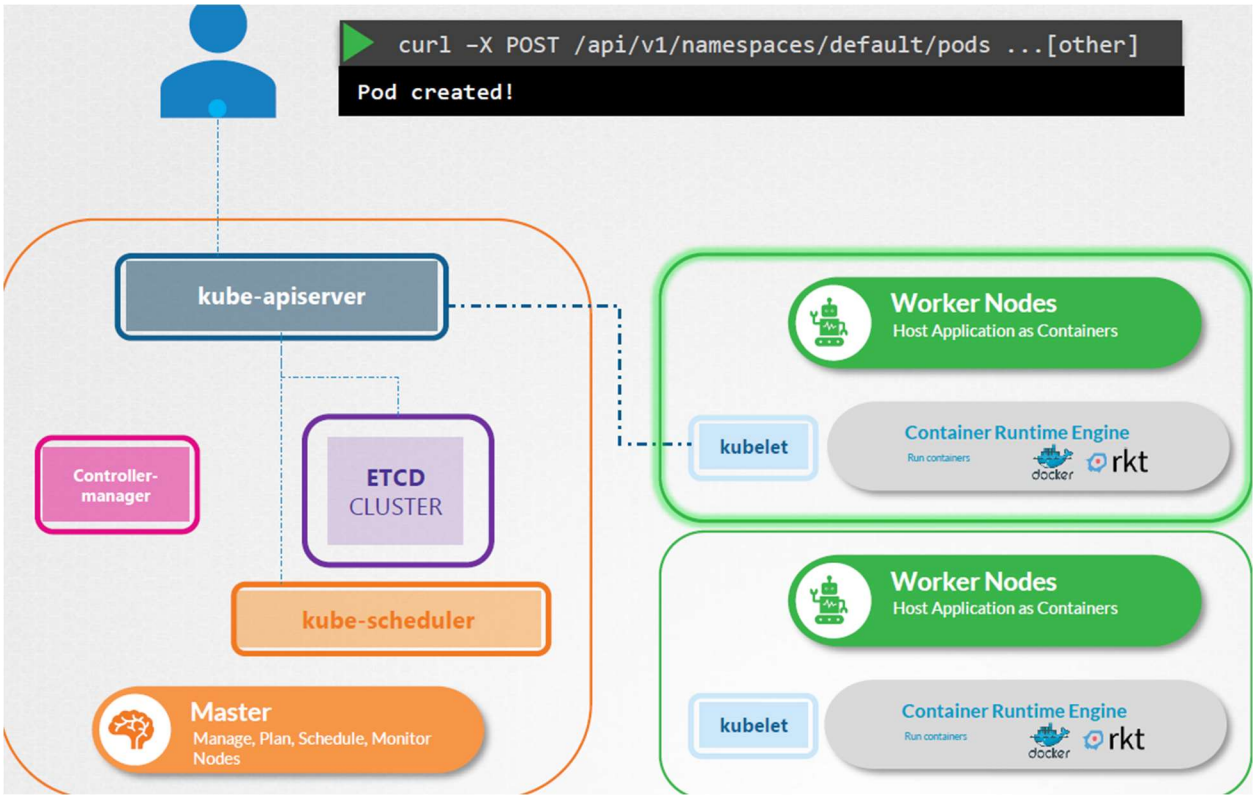


You don't really need to use the `kubectl` command line. Instead, you could also invoke the APIs directly by sending a POST request.

Let's look at an example of creating a pod. When you do that, as before, the request is authenticated first and then validated. In this case, the API server creates a pod object without assigning it to a node, updates the information in the etcd server, updates the user that the pod has been created.

The scheduler continuously monitors the API server and realizes that there is a new pod with no node assigned. The scheduler identifies the right node to place the new pod on and communicates that back to the kube-apiserver. The API server then updates the information in the etcd cluster.

The API server then passes that information to the kubelet in the appropriate worker node. The kubelet then creates the pod on the node and instructs the container runtime engine to deploy the application image. Once done, the kubelet updates the status back to the API server, and the API server then updates the data back in the etcd cluster.



A similar pattern is followed every time a change is requested. The kube-apiserver is at the center of all the different tasks that need to be performed to make a change in the cluster.

To summarize, the kube-apiserver is responsible for authenticating and validating requests, retrieving and updating data in the etcd data store. In fact, kube-apiserver is the only component that interacts directly with the etcd data store. The other components, such as the scheduler, kube-controller-manager, and kubelet, use the API server to perform updates in the cluster in their respective areas.

1. Authenticate User
2. Validate Request
3. Retrieve data
4. Update ETCD
5. Scheduler
6. Kubelet

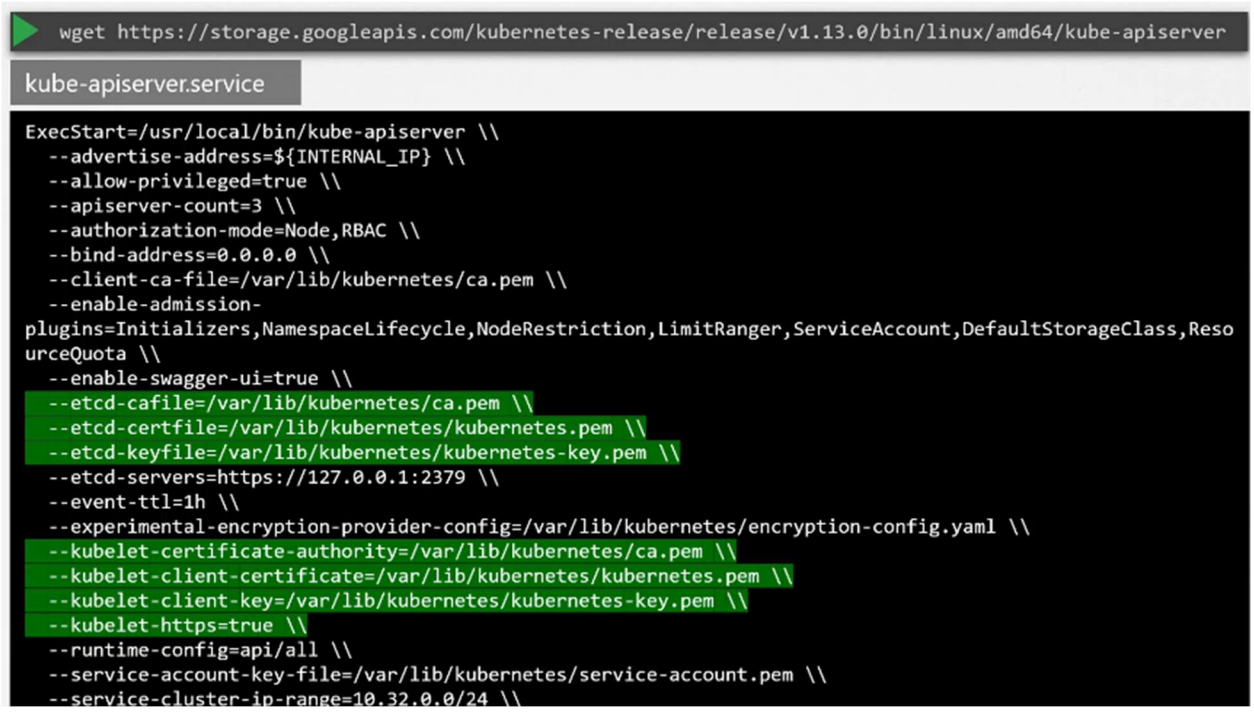
If you bootstrapped your cluster using the kubeadmin tool, then you don't need to know this. But if you're setting up the hardway, then the kube-apiserver is available as a binary on the Kubernetes release page. Download it and configure it to run as a service on your Kubernetes master node.

The kube-apiserver is run with a lot of parameters, as you can see here. Throughout this section, we're going to take a peek at how to install and configure these individual components of the Kubernetes architecture. You don't have to understand all of the options right now. But I think having a high-level understanding of some of these now will make it easier later when we configure the whole cluster and all of its components from scratch.

The Kubernetes architecture consists of a lot of different components working with each other, talking to each other in many different ways. So they all need to know where the other components are. There are different modes of authentication, authorization, encryption, and security. And that's why you have so many options. When we go through the relevant section in the course, we will pull up this file and look at the relevant options.

For now, we will look at a few important ones. A lot of them are certificates that are used to secure the connectivity between different components. We'll look at these certificates in more detail when we go through the SSL/TLS certificates lecture later in this course. There is a whole section just for it. So we'll get rid of them for now. But just remember, all of the various components we're going to look at in this section will have certificates associated with them.

The option `etcd servers` is where you specify the location of the etcd servers. This is how the kube-apiserver connects to the etcd servers.



So how do you view the kube-apiserver options in an existing cluster? It depends on how you set up your cluster. If you set it up with a kubeadm tool, the kubeadm deploys the kubeadm-apiserver as a pod in the kube-system namespace on the master node. You can see the options within the pod definition file, located at etc/kubernetes/manifest folder.

```
kubectl get pods -n kube-system
```

```
NAMESPACE     NAME                              READY     STATUS    RESTARTS   AGE
kube-system   coredns-78fcdf6894-hwrq9          1/1       Running   0          16m
kube-system   coredns-78fcdf6894-rzhjr          1/1       Running   0          16m
kube-system   etcd-master                       1/1       Running   0          15m
kube-system   kube-apiserver-master             1/1       Running   0          15m
kube-system   kube-controller-manager-master    1/1       Running   0          15m
kube-system   kube-proxy-lzt6f                  1/1       Running   0          16m
kube-system   kube-proxy-zm5qd                  1/1       Running   0          16m
kube-system   kube-scheduler-master             1/1       Running   0          15m
kube-system   weave-net-29z42                   2/2       Running   1          16m
kube-system   weave-net-snmdl                   2/2       Running   1          16m    -
```

```
cat /etc/kubernetes/manifests/kube-apiserver.yaml
```

```
spec:
  containers:
  - command:
    - kube-apiserver
    - --authorization-mode=Node,RBAC
    - --advertise-address=172.17.0.32
    - --allow-privileged=true
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --disable-admission-plugins=PersistentVolumeLabel
    - --enable-admission-plugins=NodeRestriction
    - --enable-bootstrap-token-auth=true
    - --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
    - --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt
    - --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key
    - --etcd-servers=https://127.0.0.1:2379
    - --insecure-port=0
    - --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.crt
    - --kubelet-client-key=/etc/kubernetes/pki/apiserver-kubelet-client.key
    - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
    - --proxy-client-cert-file=/etc/kubernetes/pki/front-proxy-client.crt
    - --proxy-client-key-file=/etc/kubernetes/pki/front-proxy-client.key
    - --requestheader-allowed-names=front-proxy-client
    - --requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt
    - --requestheader-extra-headers-prefix=X-Remote-Extra-
    - --requestheader-group-headers=X-Remote-Group
```

In a non-kubeadmin setup, you can inspect the options by viewing the kube-apiserver service located at etc/systemd/system/kube-apiserver.service.

```
cat /etc/systemd/system/kube-apiserver.service
```

```
[Service]
ExecStart=/usr/local/bin/kube-apiserver \\
  --advertise-address=${INTERNAL_IP} \\
  --allow-privileged=true \\
  --apiserver-count=3 \\
  --audit-log-maxage=30 \\
  --audit-log-maxbackup=3 \\
  --audit-log-maxsize=100 \\
  --audit-log-path=/var/log/audit.log \\
  --authorization-mode=Node,RBAC \\
  --bind-address=0.0.0.0 \\
  --client-ca-file=/var/lib/kubernetes/ca.pem \\
  --enable-admission-
plugins=Initializers,NamespaceLifecycle,NodeRestriction,LimitRanger,ServiceAccount,Defa
ultStorageClass,ResourceQuota \\
  --enable-swagger-ui=true \\
  --etcd-cafile=/var/lib/kubernetes/ca.pem \\
  --etcd-certfile=/var/lib/kubernetes/kubernetes.pem \\
  --etcd-keyfile=/var/lib/kubernetes/kubernetes-key.pem \\
  --etcd-
servers=https://10.240.0.10:2379,https://10.240.0.11:2379,https://10.240.0.12:2379 \\
  --event-ttl=1h \\
  --experimental-encryption-provider-config=/var/lib/kubernetes/encryption-config.yaml
\\
  --kubelet-certificate-authority=/var/lib/kubernetes/ca.pem \\
  --kubelet-client-certificate=/var/lib/kubernetes/kubernetes.pem \\
```

You can also see the running process and the effective options by listing the process on the master node and searching for kube-apiserver.

```
ps -aux | grep kube-apiserver
```

```
root      2348  3.3 15.4 399040 315604 ?       Ssl  15:46   1:22 kube-apiserver --authorization-mode=Node,RBAC --
advertise-address=172.17.0.32 --allow-privileged=true --client-ca-file=/etc/kubernetes/pki/ca.crt --disable-
admission-plugins=PersistentVolumeLabel --enable-admission-plugins=NodeRestriction--enable-bootstrap-token-
auth=true --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-
client.crt --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key --etcd-servers=https://127.0.0.1:2379 --
insecure-port=0 --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.crt --kubelet-client-
key=/etc/kubernetes/pki/apiserver-kubelet-client.key --kubelet-preferred-address-
types=InternalIP,ExternalIP,Hostname --proxy-client-cert-file=/etc/kubernetes/pki/front-proxy-client.crt --proxy-
client-key-file=/etc/kubernetes/pki/front-proxy-client.key--requestheader-allowed-names=front-proxy-client --
requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt --requestheader-extra-headers-prefix=X-Remote-
Extra- --requestheader-group-headers=X-Remote-Group --requestheader-username-headers=X-Remote-User --secure-
port=6443 --service-account-key-file=/etc/kubernetes/pki/sa.pub --service-cluster-ip-range=10.96.0.0/12 --tls-
cert-file=/etc/kubernetes/pki/apiserver.crt --tls-private-key-file=/etc/kubernetes/pki/apiserver.key
```

s