3 node-os K8S multi master klaszter

Környezet

- kube01 (192.168.110.161)
- kube02 (192.168.110.162)
- kube03 (192.168.110.163)

Kubernetes szolgáltatás cím kube.r-l.hu (192.168.110.160).

Elsődleges node inicializálása

Az elsődleges gépen el kell végezni a kube-vip szolgáltatás inicializálását. Ehhez három paramétert kell megadni.

VIP cím megadása

```
export VIP=192.168.110.160
```

Hálózati csatoló nevének megadása

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
UP group default qlen 1000
    link/ether 52:54:00:b7:4b:8d brd ff:ff:ff:ff:ff:ff
    altname enx525400b74b8d
    inet 192.168.110.161/24 brd 192.168.110.255 scope global noprefixroute
enp1s0
       valid_lft forever preferred_lft forever

export INTERFACE=enp1s0
```

Kube-VIP verzió

```
[root@kube01 ~]# curl -sL
https://api.github.com/repos/kube-vip/kube-vip/releases | jq -r ".[0].name"
```

```
v1.1.2

[root@kube01 ~]# export KVVERSION=$(curl -sL
https://api.github.com/repos/kube-vip/kube-vip/releases | jq -r ".[0].name")

[root@kube01 ~]# echo $KVVERSION
v1.1.2
```

Ellenőrzés

```
[root@kube01 ~]# echo $VIP
192.168.110.160

[root@kube01 ~]# echo $INTERFACE
enp1s0

[root@kube01 ~]# echo $KVVERSION
v1.1.2
```

Statikus POD-ok

```
[root@kube01 ~]# ctr image pull ghcr.io/kube-vip/kube-vip:$KVVERSION

[root@kube01 ~]# ctr image ls
REF                             TYPE
DIGEST                                                                  SIZE
PLATFORMS                                                          LABELS
ghcr.io/kube-vip/kube-vip:v1.1.2 application/vnd.oci.image.index.v1+json
sha256:840305b94ef2a89abb3b7fd2b09edfbde690d90052020da4dff90679fe892da2 17.7
MiB linux/amd64,linux/arm/v7,linux/arm64,linux/ppc64le,linux/s390x -
```

VIP konfiguráció létrehozása

A generált fájlban cserélni kell a `kubeconfig` volume `path: /etc/kubernetes/super-admin.conf` elérési útvonalát `path: /etc/kubernetes/super-admin.conf` értékre. Ez visszaállítható a `kubeadm init` futtatása után.

```
apiVersion: v1
kind: Pod
metadata:
  name: kube-vip
  namespace: kube-system
spec:
  containers:
  - args:
    - manager
    env:
```

```
    - name: vip_arp
      value: "true"
    - name: port
      value: "6443"
    - name: vip_nodename
      valueFrom:
        fieldRef:
          fieldPath: spec.nodeName
    - name: vip_interface
      value: enp1s0
    - name: vip_subnet
      value: "32"
    - name: dns_mode
      value: first
    - name: dhcp_mode
      value: ipv4
    - name: cp_enable
      value: "true"
    - name: cp_namespace
      value: kube-system
    - name: svc_enable
      value: "true"
    - name: svc_leasename
      value: plndr-svcs-lock
    - name: vip_leaderelection
      value: "true"
    - name: vip_leasename
      value: plndr-cp-lock
    - name: vip_leaseduration
      value: "15"
    - name: vip_renewdeadline
      value: "10"
    - name: vip_retryperiod
      value: "2"
    - name: address
      value: 192.168.110.160
    - name: prometheus_server
      value: :2112
    image: ghcr.io/kube-vip/kube-vip:v1.1.2
    imagePullPolicy: IfNotPresent
    name: kube-vip
    resources: {}
    securityContext:
      capabilities:
        add:
        - NET_ADMIN
        - NET_RAW
        drop:
        - ALL
    volumeMounts:
    - mountPath: /etc/kubernetes/admin.conf
```

```
      name: kubeconfig
  hostAliases:
  - hostnames:
    - kubernetes
    ip: 127.0.0.1
  hostNetwork: true
  volumes:
  - hostPath:
      path: /etc/kubernetes/admin.conf
    name: kubeconfig
status: {}

[root@kube01 ~]# sed -i 's/path: \/etc\/kubernetes\/admin.conf/path:
\/etc\/kubernetes\/super-admin.conf/' /etc/kubernetes/manifests/kube-
vip.yaml

[root@kube01 ~]# tail -n5 /etc/kubernetes/manifests/kube-vip.yaml
  - hostPath:
      path: /etc/kubernetes/super-admin.conf
    name: kubeconfig
status: {}
```

Kubernetes klaszter inicializálása

kubeadm-config.yaml létrehozása

Figyelni kell arra, hogy megadjuk a kapcsolódáshoz használt összes nevet. Érdemes a CA és a belső tanúsítványok érvényességét nagyon hosszú időre megadni.

```
[root@kube01 ~]# cat > /etc/kubernetes/kubeadm-config.yaml <<'EOF'
apiVersion: kubeadm.k8s.io/v1beta4
kind: InitConfiguration
localAPIEndpoint:
  advertiseAddress: 192.168.110.161
  bindPort: 6443
nodeRegistration:
  criSocket: unix:///run/containerd/containerd.sock
  imagePullSerial: true
  kubeletExtraArgs:
    - name: node-ip
      value: "192.168.110.161"
    - name: cgroup-driver
      value: "systemd"

---
apiVersion: kubeadm.k8s.io/v1beta4
kind: ClusterConfiguration
clusterName: kubernetes
```

```
kubernetesVersion: v1.36.0
controlPlaneEndpoint: "192.168.110.160:6443"
certificatesDir: /etc/kubernetes/pki
certificateValidityPeriod: 873648h
caCertificateValidityPeriod: 1747296h

networking:
  serviceSubnet: "10.96.0.0/12"
  podSubnet: "10.244.0.0/16"
  dnsDomain: "cluster.local"

apiServer:
  certSANs:
    - "192.168.110.160"
    - "192.168.110.161"
    - "192.168.110.162"
    - "192.168.110.163"
    - "127.0.0.1"
    - "kube.r-l.hu"
    - "kube"
    - "kube01"
    - "kube02"
    - "kube03"
    - "kubernetes"
    - "kubernetes.default"
    - "kubernetes.default.svc"
    - "kubernetes.default.svc.cluster.local"
  extraArgs:
    - name: authorization-mode
      value: Node,RBAC
    - name: enable-admission-plugins
      value: NodeRestriction

controllerManager:
  extraArgs:
    - name: bind-address
      value: "0.0.0.0"

scheduler:
  extraArgs:
    - name: bind-address
      value: "0.0.0.0"

etcd:
  local:
    dataDir: /var/lib/etcd
EOF
```

Kubernetes klaszter inicializálása

```
[root@kube01 ~]# kubeadm init --config /etc/kubernetes/kubeadm-config.yaml --upload-certs
W0503 11:27:01.592863    1427 validation.go:79] WARNING: certificateValidityPeriod: the value 873648h0m0s is more than the recommended default for certificate expiration: 8760h0m0s
W0503 11:27:01.592946    1427 validation.go:79] WARNING: caCertificateValidityPeriod: the value 1747296h0m0s is more than the recommended default for CA certificate expiration: 87600h0m0s
[init] Using Kubernetes version: v1.36.0
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [kube kube.r-l.hu kube01 kube02 kube03 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 192.168.110.161 192.168.110.160 192.168.110.162 192.168.110.163 127.0.0.1]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [kube01 localhost] and IPs [192.168.110.161 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [kube01 localhost] and IPs [192.168.110.161 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "super-admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[kubelet-start] Writing kubelet environment file with flags to file
```

```
"/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file
"/var/lib/kubelet/instance-config.yaml"
[patches] Applied patch of type "application/strategic-merge-patch+json" to
target "kubeletconfiguration"
[kubelet-start] Writing kubelet configuration to file
"/var/lib/kubelet/config.yaml"
[kubelet-start] Starting the kubelet
[wait-control-plane] Waiting for the kubelet to boot up the control plane as
static Pods from directory "/etc/kubernetes/manifests"
[kubelet-check] Waiting for a healthy kubelet at
http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 1.004560939s
[control-plane-check] Waiting for healthy control plane components. This can
take up to 4m0s
[control-plane-check] Checking kube-apiserver at
https://192.168.110.161:6443/livez
[control-plane-check] Checking kube-controller-manager at
https://0.0.0.0:10257/healthz
[control-plane-check] Checking kube-scheduler at https://0.0.0.0:10259/livez
[control-plane-check] kube-controller-manager is healthy after 8.004530664s
[control-plane-check] kube-scheduler is healthy after 9.972576107s
[control-plane-check] kube-apiserver is healthy after 15.501804074s
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config"
in the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config" in namespace kube-system
with the configuration for the kubelets in the cluster
[upload-certs] Storing the certificates in Secret "kubeadm-certs" in the
"kube-system" Namespace
[upload-certs] Using certificate key:
7b0abde00adadcc8c5d6ed2879f63eff27ed5027777027c48a6bf0fac77d343e
[mark-control-plane] Marking the node kube01 as control-plane by adding the
labels: [node-role.kubernetes.io/control-plane node.kubernetes.io/exclude-
from-external-load-balancers]
[mark-control-plane] Marking the node kube01 as control-plane by adding the
taints [node-role.kubernetes.io/control-plane:NoSchedule]
[bootstrap-token] Using token: 2559e9.vkgwdyge2y3f1crp
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC
Roles
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to
get nodes
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to
post CSRs in order for nodes to get long term certificate credentials
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller
automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] Configured RBAC rules to allow certificate rotation for
all node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public"
namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a
rotatable kubelet client certificate and key
```

```
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular
user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of control-plane nodes running the following
command on each as root:

  kubeadm join 192.168.110.160:6443 --token 2559e9.vkgwdyge2y3f1crp \
    --discovery-token-ca-cert-hash
sha256:838e04036d629fbf3e152fc6606bf9a143f8e1ca0b0bc00d747fe1caba5df8f1 \
    --control-plane --certificate-key
7b0abde00adadcc8c5d6ed2879f63eff27ed5027777027c48a6bf0fac77d343e

Please note that the certificate-key gives access to cluster sensitive data,
keep it secret!
As a safeguard, uploaded-certs will be deleted in two hours; If necessary,
you can use
"kubeadm init phase upload-certs --upload-certs" to reload certs afterward.

Then you can join any number of worker nodes by running the following on
each as root:

kubeadm join 192.168.110.160:6443 --token 2559e9.vkgwdyge2y3f1crp \
    --discovery-token-ca-cert-hash
sha256:838e04036d629fbf3e152fc6606bf9a143f8e1ca0b0bc00d747fe1caba5df8f1

[root@kube01 ~]# mkdir -p $HOME/.kube

[root@kube01 ~]# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

[root@kube01 ~]# sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Tanúsítvány lejáratok ellenőrzése

```
[root@kube01 ~]# openssl x509 -in /etc/kubernetes/pki/ca.crt -noout -dates
notBefore=May  3 09:22:01 2026 GMT
notAfter=Sep  1 09:27:01 2225 GMT

[root@kube01 ~]# openssl x509 -in /etc/kubernetes/pki/etcd/ca.crt -noout -
dates
notBefore=May  3 09:22:01 2026 GMT
notAfter=Sep  1 09:27:01 2225 GMT

[root@kube01 ~]# openssl x509 -in /etc/kubernetes/pki/apiserver.crt -noout -
datesnotBefore=May  3 09:22:01 2026 GMT
notAfter=Jan  1 09:27:01 2126 GMT
```

IP ellenőrzés

```
[root@kube01 ~]# ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
UP group default qlen 1000
    link/ether 52:54:00:b7:4b:8d brd ff:ff:ff:ff:ff:ff
    altname enx525400b74b8d
    inet 192.168.110.161/24 brd 192.168.110.255 scope global noprefixroute
enp1s0
       valid_lft forever preferred_lft forever
    inet 192.168.110.160/32 scope global deprecated enp1s0
       valid_lft forever preferred_lft forever
```

További control-plane gépek csatlakoztatása

```
[root@kube02 ~]# kubeadm join 192.168.110.160:6443 --token
2559e9.vkgwdyge2y3f1crp \
        --discovery-token-ca-cert-hash
sha256:838e04036d629fbf3e152fc6606bf9a143f8e1ca0b0bc00d747fe1caba5df8f1 \
        --control-plane --certificate-key
7b0abde00adadcc8c5d6ed2879f63eff27ed5027777027c48a6bf0fac77d343e
[preflight] Running pre-flight checks
[preflight] Reading configuration from the "kubeadm-config" ConfigMap in
namespace "kube-system"...
[preflight] Use 'kubeadm init phase upload-config kubeadm --config your-
config-file' to re-upload it.
```

```
[preflight] Running pre-flight checks before initializing the new control
plane instance
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your
internet connection
[preflight] You can also perform this action beforehand using 'kubeadm
config images pull'
[download-certs] Downloading the certificates in Secret "kubeadm-certs" in
the "kube-system" Namespace
[download-certs] Saving the certificates to the folder:
"/etc/kubernetes/pki"
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [kube02 localhost]
and IPs [192.168.110.162 127.0.0.1 ::1]
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [kube02 localhost]
and IPs [192.168.110.162 127.0.0.1 ::1]
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [kube kube.r-l.hu
kube01 kube02 kube03 kubernetes kubernetes.default kubernetes.default.svc
kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 192.168.110.162
192.168.110.160 192.168.110.161 192.168.110.163 127.0.0.1]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Valid certificates and keys now exist in "/etc/kubernetes/pki"
[certs] Using the existing "sa" key
[kubeconfig] Generating kubeconfig files
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[check-etcd] Checking that the etcd cluster is healthy
[kubelet-start] Writing kubelet configuration to file
"/var/lib/kubelet/instance-config.yaml"
[patches] Applied patch of type "application/strategic-merge-patch+json" to
target "kubeletconfiguration"
[kubelet-start] Writing kubelet configuration to file
"/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file
"/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[etcd] Announced new etcd member joining to the existing etcd cluster
[etcd] Creating static Pod manifest for "etcd"
[etcd] Waiting for the new etcd member to join the cluster. This can take up
```

```
to 40s
[kubelet-check] Waiting for a healthy kubelet at
http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 2.12497ms
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap
[mark-control-plane] Marking the node kube02 as control-plane by adding the
labels: [node-role.kubernetes.io/control-plane node.kubernetes.io/exclude-
from-external-load-balancers]
[mark-control-plane] Marking the node kube02 as control-plane by adding the
taints [node-role.kubernetes.io/control-plane:NoSchedule]
[control-plane-check] Waiting for healthy control plane components. This can
take up to 4m0s
[control-plane-check] Checking kube-apiserver at
https://192.168.110.162:6443/livez
[control-plane-check] Checking kube-controller-manager at
https://0.0.0.0:10257/healthz
[control-plane-check] Checking kube-scheduler at https://0.0.0.0:10259/livez
[control-plane-check] kube-scheduler is healthy after 5.314373ms
[control-plane-check] kube-apiserver is healthy after 10.398496ms
[control-plane-check] kube-controller-manager is healthy after 10.876078ms

This node has joined the cluster and a new control plane instance was
created:

* Certificate signing request was sent to apiserver and approval was
received.
* The Kubelet was informed of the new secure connection details.
* Control plane label and taint were applied to the new node.
* The Kubernetes control plane instances scaled up.
* A new etcd member was added to the local/stacked etcd cluster.

To start administering your cluster from this node, you need to run the
following as a regular user:

    mkdir -p $HOME/.kube
    sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
    sudo chown $(id -u):$(id -g) $HOME/.kube/config

Run 'kubectl get nodes' to see this node join the cluster.

[root@kube02 ~]# mkdir -p $HOME/.kube

[root@kube02 ~]# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

[root@kube02 ~]# sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

```
[root@kube03 ~]# kubeadm join 192.168.110.160:6443 --token
2559e9.vkgwdyge2y3f1crp \
        --discovery-token-ca-cert-hash
sha256:838e04036d629fbf3e152fc6606bf9a143f8e1ca0b0bc00d747fe1caba5df8f1 \
        --control-plane --certificate-key
```

```
7b0abde00adadcc8c5d6ed2879f63eff27ed5027777027c48a6bf0fac77d343e
[preflight] Running pre-flight checks
[preflight] Reading configuration from the "kubeadm-config" ConfigMap in
namespace "kube-system"...
[preflight] Use 'kubeadm init phase upload-config kubeadm --config your-
config-file' to re-upload it.
[preflight] Running pre-flight checks before initializing the new control
plane instance
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your
internet connection
[preflight] You can also perform this action beforehand using 'kubeadm
config images pull'
[download-certs] Downloading the certificates in Secret "kubeadm-certs" in
the "kube-system" Namespace
[download-certs] Saving the certificates to the folder:
"/etc/kubernetes/pki"
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [kube03 localhost]
and IPs [192.168.110.163 127.0.0.1 ::1]
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [kube03 localhost]
and IPs [192.168.110.163 127.0.0.1 ::1]
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [kube kube.r-l.hu
kube01 kube02 kube03 kubernetes kubernetes.default kubernetes.default.svc
kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 192.168.110.163
192.168.110.160 192.168.110.161 192.168.110.162 127.0.0.1]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Valid certificates and keys now exist in "/etc/kubernetes/pki"
[certs] Using the existing "sa" key
[kubeconfig] Generating kubeconfig files
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[check-etcd] Checking that the etcd cluster is healthy
[kubelet-start] Writing kubelet configuration to file
"/var/lib/kubelet/instance-config.yaml"
[patches] Applied patch of type "application/strategic-merge-patch+json" to
target "kubeletconfiguration"
[kubelet-start] Writing kubelet configuration to file
"/var/lib/kubelet/config.yaml"
```

```
[kubelet-start] Writing kubelet environment file with flags to file
"/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[etcd] Announced new etcd member joining to the existing etcd cluster
[etcd] Creating static Pod manifest for "etcd"
{"level":"warn","ts":"2026-05-03T11:34:09.973018+0200","logger":"etcd-
client","caller":"v3@v3.6.8/retry_interceptor.go:65","msg":"retrying of
unary invoker failed","target":"etcd-
endpoints://0x1ceadd3e5c20/192.168.110.162:2379","method":"/etcdserverpb.Clu
ster/MemberPromote","attempt":0,"error":"rpc error: code =
FailedPrecondition desc = etcdserver: can only promote a learner member
which is in sync with leader"}
[etcd] Waiting for the new etcd member to join the cluster. This can take up
to 40s
[kubelet-check] Waiting for a healthy kubelet at
http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 3.012325ms
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap
[mark-control-plane] Marking the node kube03 as control-plane by adding the
labels: [node-role.kubernetes.io/control-plane node.kubernetes.io/exclude-
from-external-load-balancers]
[mark-control-plane] Marking the node kube03 as control-plane by adding the
taints [node-role.kubernetes.io/control-plane:NoSchedule]
[control-plane-check] Waiting for healthy control plane components. This can
take up to 4m0s
[control-plane-check] Checking kube-apiserver at
https://192.168.110.163:6443/livez
[control-plane-check] Checking kube-controller-manager at
https://0.0.0.0:10257/healthz
[control-plane-check] Checking kube-scheduler at https://0.0.0.0:10259/livez
[control-plane-check] kube-controller-manager is healthy after 13.773855ms
[control-plane-check] kube-scheduler is healthy after 17.595626ms
[control-plane-check] kube-apiserver is healthy after 635.031961ms

This node has joined the cluster and a new control plane instance was
created:

* Certificate signing request was sent to apiserver and approval was
received.
* The Kubelet was informed of the new secure connection details.
* Control plane label and taint were applied to the new node.
* The Kubernetes control plane instances scaled up.
* A new etcd member was added to the local/stacked etcd cluster.

To start administering your cluster from this node, you need to run the
following as a regular user:

    mkdir -p $HOME/.kube
    sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
    sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

```
Run 'kubectl get nodes' to see this node join the cluster.

[root@kube03 ~]# mkdir -p $HOME/.kube

[root@kube03 ~]# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

[root@kube03 ~]# sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Node ellenőrzés

```
[root@kube01 ~]# kubectl get nodes
NAME     STATUS     ROLES           AGE     VERSION
kube01   NotReady   control-plane   11m     v1.36.0
kube02   NotReady   control-plane   5m38s   v1.36.0
kube03   NotReady   control-plane   4m56s   v1.36.0
```

CNI plugin telepítése

Ebben a konfigurációban a `flannel` CNI plugin telepítése történik

```
[root@kube01 ~]# kubectl apply -f
https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.
yml
namespace/kube-flannel created
serviceaccount/flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
```

Klaszter állapot ellenőrzés

Türelem!

```
[root@kube01 ~]# kubectl get nodes
NAME     STATUS     ROLES           AGE     VERSION
kube01   NotReady   control-plane   14m     v1.36.0
kube02   NotReady   control-plane   8m35s   v1.36.0
kube03   Ready      control-plane   7m53s   v1.36.0

[root@kube01 ~]# kubectl get nodes
NAME     STATUS   ROLES           AGE     VERSION
kube01   Ready    control-plane   14m     v1.36.0
kube02   Ready    control-plane   8m40s   v1.36.0
kube03   Ready    control-plane   7m58s   v1.36.0
```

A klaszter kész.

Kubernetes klaszter koniguráció törlése

Az alábbi utasítások törlik a kubernetes klaszter állományait a fájlrendszerből.

Szolgáltatások leállítása

```
[root@kube01 ~]# systemctl stop kubelet

[root@kube01 ~]# systemctl stop containerd

[root@kube02 ~]# systemctl stop kubelet

[root@kube02 ~]# systemctl stop containerd

[root@kube03 ~]# systemctl stop kubelet

[root@kube03 ~]# systemctl stop containerd
```

```
[root@kube01 ~]# kubeadm reset -f
[reset] Reading configuration from the "kubeadm-config" ConfigMap in
namespace "kube-system"...
[reset] Use 'kubeadm init phase upload-config kubeadm --config your-config-
file' to re-upload it.
[preflight] Running pre-flight checks
[reset] Deleted contents of the etcd data directory: /var/lib/etcd
[reset] Stopping the kubelet service
[reset] Unmounting mounted directories in "/var/lib/kubelet"
W0503 10:40:36.411120   54725 cleanupnode.go:105] [reset] Failed to remove
containers: failed to create new CRI runtime service: validate service
connection: validate CRI v1 runtime API for endpoint
"unix:///var/run/containerd/containerd.sock": rpc error: code = Unavailable
desc = connection error: desc = "transport: Error while dialing: dial unix
/var/run/containerd/containerd.sock: connect: no such file or directory"
[reset] Deleting contents of directories: [/etc/kubernetes/manifests
/var/lib/kubelet /etc/kubernetes/pki]
[reset] Deleting files: [/etc/kubernetes/admin.conf /etc/kubernetes/super-
admin.conf /etc/kubernetes/kubelet.conf /etc/kubernetes/bootstrap-
kubelet.conf /etc/kubernetes/controller-manager.conf
/etc/kubernetes/scheduler.conf]

The reset process does not perform cleanup of CNI plugin configuration,
network filtering rules and kubeconfig files.

For information on how to perform this cleanup manually, please see:
    https://k8s.io/docs/reference/setup-tools/kubeadm/kubeadm-reset/
```

```
[root@kube02 ~]# kubeadm reset -f
[reset] Reading configuration from the "kubeadm-config" ConfigMap in
namespace "kube-system"...
[reset] Use 'kubeadm init phase upload-config kubeadm --config your-config-
file' to re-upload it.
W0503 10:40:57.271472   51194 reset.go:141] [reset] Unable to fetch the
kubeadm-config ConfigMap from cluster: failed to get config map: Get
"https://192.168.110.160:6443/api/v1/namespaces/kube-system/configmaps/kubea
dm-config?timeout=10s": net/http: request canceled (Client.Timeout exceeded
while awaiting headers)
[preflight] Running pre-flight checks
W0503 10:40:57.271708   51194 removeetcdmember.go:105] [reset] No kubeadm
config, using etcd pod spec to get data directory
[reset] Deleted contents of the etcd data directory: /var/lib/etcd
[reset] Stopping the kubelet service
[reset] Unmounting mounted directories in "/var/lib/kubelet"
W0503 10:40:57.285204   51194 cleanupnode.go:105] [reset] Failed to remove
containers: failed to create new CRI runtime service: validate service
connection: validate CRI v1 runtime API for endpoint
"unix:///var/run/containerd/containerd.sock": rpc error: code = Unavailable
desc = connection error: desc = "transport: Error while dialing: dial unix
/var/run/containerd/containerd.sock: connect: no such file or directory"
[reset] Deleting contents of directories: [/etc/kubernetes/manifests
/var/lib/kubelet /etc/kubernetes/pki]
[reset] Deleting files: [/etc/kubernetes/admin.conf /etc/kubernetes/super-
admin.conf /etc/kubernetes/kubelet.conf /etc/kubernetes/bootstrap-
kubelet.conf /etc/kubernetes/controller-manager.conf
/etc/kubernetes/scheduler.conf]

The reset process does not perform cleanup of CNI plugin configuration,
network filtering rules and kubeconfig files.

For information on how to perform this cleanup manually, please see:
    https://k8s.io/docs/reference/setup-tools/kubeadm/kubeadm-reset/

[root@kube03 lib]# kubeadm reset -f
[reset] Reading configuration from the "kubeadm-config" ConfigMap in
namespace "kube-system"...
[reset] Use 'kubeadm init phase upload-config kubeadm --config your-config-
file' to re-upload it.
W0503 10:41:07.848963   51644 reset.go:141] [reset] Unable to fetch the
kubeadm-config ConfigMap from cluster: failed to get config map: Get
"https://192.168.110.160:6443/api/v1/namespaces/kube-system/configmaps/kubea
dm-config?timeout=10s": net/http: request canceled while waiting for
connection (Client.Timeout exceeded while awaiting headers)
[preflight] Running pre-flight checks
W0503 10:41:07.849157   51644 removeetcdmember.go:105] [reset] No kubeadm
config, using etcd pod spec to get data directory
[reset] Deleted contents of the etcd data directory: /var/lib/etcd
[reset] Stopping the kubelet service
[reset] Unmounting mounted directories in "/var/lib/kubelet"
```

```
W0503 10:41:07.867639   51644 cleanupnode.go:105] [reset] Failed to remove
containers: failed to create new CRI runtime service: validate service
connection: validate CRI v1 runtime API for endpoint
"unix:///var/run/containerd/containerd.sock": rpc error: code = Unavailable
desc = connection error: desc = "transport: Error while dialing: dial unix
/var/run/containerd/containerd.sock: connect: no such file or directory"
[reset] Deleting contents of directories: [/etc/kubernetes/manifests
/var/lib/kubelet /etc/kubernetes/pki]
[reset] Deleting files: [/etc/kubernetes/admin.conf /etc/kubernetes/super-
admin.conf /etc/kubernetes/kubelet.conf /etc/kubernetes/bootstrap-
kubelet.conf /etc/kubernetes/controller-manager.conf
/etc/kubernetes/scheduler.conf]

The reset process does not perform cleanup of CNI plugin configuration,
network filtering rules and kubeconfig files.

For information on how to perform this cleanup manually, please see:
    https://k8s.io/docs/reference/setup-tools/kubeadm/kubeadm-reset/
```

Felesleges állományok és könyvtárak törlése

```
[root@kube01 ~]# rm -rf /etc/cni/net.d/* /root/.kube/ /var/lib/cni/
/var/lib/containerd/

[root@kube02 ~]# rm -rf /etc/cni/net.d/* /root/.kube/ /var/lib/cni/
/var/lib/containerd/

[root@kube03 ~]# rm -rf /etc/cni/net.d/* /root/.kube/ /var/lib/cni/
/var/lib/containerd/
```

From:
https://wiki.r-l.hu/ - **Reverse-Logic wiki**

Permanent link:
https://wiki.r-l.hu/doku.php?id=kubernetes:multi-master

Last update: **2026/05/03 09:49**