Kubernetes klaszter telepítése

A telepített környezet három virtuális gépből áll:

* *kube01* (control plane): Almalinux 10 (x86_64), minimal install (VCPU: 2, RAM: 3 GB, DISK: 20 GB)
* *kube02* (worker): Almalinux 10 (x86_64), minimal install (VCPU: 2, RAM: 4 GB, DISK: 20 GB) *
kube03 (worker): Almalinux 10 (x86_64), minimal install (VCPU: 2, RAM: 4 GB, DISK: 20 GB)

A telepítéskor ne adjunk swap területet.

Telepítést követő lépések

Az alábbi utasításokat a klaszter összes gépén le kell futtatni.

SElinux megengedő módba kapcsolása

```
# sed -i 's/^SELINUX=.*/SELINUX=permissive/' /etc/selinux/config

# setenforce 0
```

Tűzfal szolgáltatás kikapcsolása és tiltása

```
# systemctl disable firewalld
Removed '/etc/systemd/system/multi-user.target.wants/firewalld.service'.
Removed '/etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service'.

# systemctl stop firewalld
```

Hosts állományok módosítása

```
# cat > /etc/hosts <<'EOF'
127.0.0.1   localhost localhost.localdomain localhost4
localhost4.localdomain4
::1         localhost localhost.localdomain localhost6
localhost6.localdomain6

192.168.110.161 kube01
192.168.110.162 kube02
192.168.110.163 kube03
EOF
```

Containerd repo telepítése

```
# curl -L -o /etc/yum.repos.d/docker-ce.repo
https://download.docker.com/linux/centos/docker-ce.repo
```

Kubernetes repo létrehozása

```
# cat > /etc/yum.repos.d/kubernetes.repo <<'EOF'
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.34/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.34/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
```

Containerd telepítése

```
# dnf install containerd
```

Containerd konfiguráció mentése

```
# cp -a /etc/containerd/config.toml /etc/containerd/config.toml.orig
```

Containerd konfiguráció készítése

```
# containerd config default > /etc/containerd/config.toml
```

Containerd konfiguráció módosítása

```
# grep pause:3 /etc/containerd/config.toml
    sandbox_image = "registry.k8s.io/pause:3.8"

# sed -i 's/pause:3.8/pause:3.10.1/' /etc/containerd/config.toml

# grep pause:3 /etc/containerd/config.toml
    sandbox_image = "registry.k8s.io/pause:3.10.1"

# grep SystemdCgroup /etc/containerd/config.toml
            SystemdCgroup = false

# sed -i 's/SystemdCgroup = false/SystemdCgroup = true/'
/etc/containerd/config.toml

# grep SystemdCgroup /etc/containerd/config.toml
            SystemdCgroup = true
```

Containerd engedélyezése és indítása

```
# systemctl --now enable containerd
Created symlink '/etc/systemd/system/multi-
user.target.wants/containerd.service' →
'/usr/lib/systemd/system/containerd.service'.
```

Kubernetes klaszterhez szükséges csomagok telepítése

```
# dnf --disableexludes=kubernetes install kubeadm kubectl kubelet
```

Kubernetes kubelet szolgáltatás engedélyezése

```
# systemctl enable kubelet
```

Control plane konfigurálása

Az alábbi utasításokat a control plane gépen kell futtatni

Klaszter init meghívása

```
# kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.34.1
[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 [kube01 kubernetes
kubernetes.default kubernetes.default.svc
kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 192.168.110.161]
[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 1.50097886s
[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://127.0.0.1:10257/healthz
[control-plane-check] Checking kube-scheduler at
https://127.0.0.1:10259/livez
[control-plane-check] kube-controller-manager is healthy after 3.507200493s
[control-plane-check] kube-scheduler is healthy after 4.632817046s
[control-plane-check] kube-apiserver is healthy after 11.004003859s
[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] Skipping phase. Please see --upload-certs
[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: is490j.gmk4mrbp5aum3q8y
[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/

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

kubeadm join 192.168.110.161:6443 --token is490j.gmk4mrbp5aum3q8y \
    --discovery-token-ca-cert-hash
sha256:2454cd136d590b724210551fcb95ac360a2761f18a43729fe043eaf8dc139027
```

A kapcsolódáshoz szükséges konfiguráció beállítása

```
# mkdir -p $HOME/.kube

# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

# sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Működés ellenőrzése

```
# kubectl get nodes
NAME     STATUS     ROLES           AGE     VERSION
kube01   NotReady   control-plane   11m     v1.34.1
```

Pod hálózat létrehozása (Flannel)

```
# 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
```

Rövid idő elteltével újabb ellenőrzés

```
# kubectl get nodes
NAME     STATUS   ROLES           AGE    VERSION
kube01   Ready    control-plane   2m     v1.34.1
```

Worker gépek csatlakoztatása

Az alábbi utasításokat a worker gépeken kell futtatni

```
# kubeadm join 192.168.110.161:6443 --token is490j.gmk4mrbp5aum3q8y --
discovery-token-ca-cert-hash
sha256:2454cd136d590b724210551fcb95ac360a2761f18a43729fe043eaf8dc139027
[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.
[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
[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 1.004029985s
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was
received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the
cluster.
```

Klaszter ellenőrzése

A klaszter ellenőrzését a control plane gépen végezzük el

```
# kubectl get nodes
```

```
NAME     STATUS   ROLES           AGE    VERSION
kube01   Ready    control-plane   12m    v1.34.1
kube02   Ready    <none>          4m5s   v1.34.1
kube03   Ready    <none>          112s   v1.34.1
```

Pod

Pod erőforás dokumentáció megjelenítése

```
# kubectl explain pod
```

Egyszerű pod létrehozása egy konténerrel parancssorból

```
# kubectl run nginx-pod --image=nginx:latest --restart=Never
```

Egyszerű pod létrehozása egy konténerrel yaml fájlból

```
# cat > egyszeru-pod-egy-kontenerrel.yaml <<EOF
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
spec:
  containers:
    - name: nginx
      image: nginx:latest
EOF
```

Konténer indítása

```
# kubectl apply -f egyszeru-pod-egy-kontenerrel.yaml
pod/nginx-pod created

# kubectl wait --for=condition=Ready pod/nginx-pod --timeout=90s
pod/nginx-pod condition met
```

Konténer ellenőrzése

```
# kubectl get pod/nginx-pod -o yaml
```

```
# kubectl describe pod/nginx-pod
Name:             nginx-pod
Namespace:        default
Priority:         0
Service Account:  default
Node:             worker01.r-logic.eu/185.207.251.233
Start Time:       Tue, 16 Sep 2025 04:33:07 +0200
Labels:           run=nginx-pod
```

```
Annotations:      <none>
Status:           Running
IP:               10.244.1.14
IPs:
  IP:  10.244.1.14
Containers:
  nginx-pod:
    Container ID:
containerd://406b1f5856e2bfaa9e91d391078458c56e64c2f9d068f9b65dbab4d3c0b44e8
b
    Image:          nginx:latest
    Image ID:
nginx@sha256:d5f28ef21aabddd098f3dbc21fe5b7a7d7a184720bc07da0b6c9b9820e97f25
e
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Tue, 16 Sep 2025 04:33:14 +0200
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-
f79p9 (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-f79p9:
    Type:                    Projected (a volume that contains injected data
from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    Optional:                false
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute
op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute
op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  53s   default-scheduler  Successfully assigned
default/nginx-pod to worker01.r-logic.eu
  Normal  Pulling    52s   kubelet            Pulling image "nginx:latest"
```

```
  Normal  Pulled     46s   kubelet            Successfully pulled image
"nginx:latest" in 5.622s (5.622s including waiting). Image size: 72319182
bytes.
  Normal  Created    46s   kubelet            Created container: nginx-pod
  Normal  Started    46s   kubelet            Started container nginx-pod
```

Konténer nevének kiolvasása

```
# kubectl get pod/nginx-pod -o jsonpath='{.spec.containers[*].name}'
```

Utasítások futtatása a konténerben

```
# kubectl exec -it pod/nginx-pod -c nginx -- sh
```

Naplók megtekintése

```
# kubectl logs pod/nginx-pod

# kubectl logs pod/nginx-pod -c nginx
```

Port tesztelése

```
# kubectl port-forward pod/nginx-pod 8080:80
```

Bővített pod definíció

```
# cat > bovitett-pod-egy-kontenerrel.yaml <<'EOF'
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod-advanced
  labels:
    app: nginx
spec:
  containers:
    - name: nginx
      image: nginx:1.25
      ports:
        - containerPort: 80
      resources:
        requests:
          cpu: "100m"
          memory: "128Mi"
        limits:
          cpu: "500m"
          memory: "256Mi"
      env:
        - name: NGINX_HOST
          value: "rl-hu"
```

```
        - name: NGINX_PORT
          value: "80"
      volumeMounts:
        - name: nginx-html
          mountPath: /usr/share/nginx/html
  volumes:
    - name: nginx-html
      emptyDir: {}
  nodeSelector:
    kubernetes.io/hostname: worker01.r-logic.eu
  restartPolicy: Always
EOF
```

Bővitett tartalom elemei

- **labels** → címkék, amelyekre később service-ek vagy deploymentek hivatkozhatnak
- **ports** → a konténeren belüli port meghatározása(TCP/80, HTTP)
- **resources** → CPU és memória foglalás minimum és maximum értékek
- **env** → környezeti változók beállítása
- **volumeMounts + volumes** → átmeneti tároló (emptyDir) csatolása a HTML tartalomnak
- **nodeSelector** → pod csak a worker01 gépen futhat
- **restartPolicy** → amennyiben megáll, újraindul

Egyszer használatos pod tesztelésekhez

```
# kubectl run debug-pod --rm -it --image=busybox:1.36 --restart=Never -- sh
```

Deployment

Létrehozás

Deployment létrehozása parancssorból

```
# kubectl create deployment nginx-deployment --image=nginx:latest && kubectl
wait --for=condition=Available deployment/nginx-deployment --timeout=90s
deployment.apps/nginx-deployment created
deployment.apps/nginx-deployment condition met
```

Deployment példányok módosítása

Frissítés és visszaállás

Deployment definíció

```
cat > nginx-deployment.yaml <<'EOF'
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  annotations:
    kubernetes.io/change-cause: "Initial deploy: nginx 1.25"
spec:
  replicas: 3
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.25
          ports:
            - containerPort: 80
EOF

kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
```

Frissítés 1.26-ra

```
# kubectl annotate deployment/nginx-deployment kubernetes.io/change-
cause="Upgrade to nginx 1.26" --overwrite
deployment.apps/nginx-deployment annotated

# kubectl set image deployment/nginx-deployment nginx=nginx:1.26
deployment.apps/nginx-deployment image updated

# kubectl rollout status deployment/nginx-deployment
Waiting for deployment "nginx-deployment" rollout to finish: 1 old replicas
are pending termination...
Waiting for deployment "nginx-deployment" rollout to finish: 1 old replicas
are pending termination...
deployment "nginx-deployment" successfully rolled out

# kubectl rollout history deployment/nginx-deployment
deployment.apps/nginx-deployment
REVISION  CHANGE-CAUSE
1         Upgrade to nginx 1.26
```

```
2         Upgrade to nginx 1.26
```

Frissítés 1.27-re

```
# kubectl rollout pause deployment/nginx-deployment

# kubectl annotate deployment/nginx-deployment kubernetes.io/change-
cause="Upgrade to nginx 1.27" --overwrite
deployment.apps/nginx-deployment annotated

# kubectl set image deployment/nginx-deployment nginx=nginx:1.27
deployment.apps/nginx-deployment image updated

# kubectl rollout resume deployment/nginx-deployment

# kubectl rollout status deployment/nginx-deployment
Waiting for deployment "nginx-deployment" rollout to finish: 1 out of 3 new
replicas have been updated...
Waiting for deployment "nginx-deployment" rollout to finish: 1 out of 3 new
replicas have been updated...
Waiting for deployment "nginx-deployment" rollout to finish: 1 out of 3 new
replicas have been updated...
Waiting for deployment "nginx-deployment" rollout to finish: 2 out of 3 new
replicas have been updated...
Waiting for deployment "nginx-deployment" rollout to finish: 2 out of 3 new
replicas have been updated...
Waiting for deployment "nginx-deployment" rollout to finish: 2 out of 3 new
replicas have been updated...
Waiting for deployment "nginx-deployment" rollout to finish: 1 old replicas
are pending termination...
Waiting for deployment "nginx-deployment" rollout to finish: 1 old replicas
are pending termination...
deployment "nginx-deployment" successfully rolled out

kubectl rollout history deployment/nginx-deployment
deployment.apps/nginx-deployment
REVISION  CHANGE-CAUSE
1         Upgrade to nginx 1.26
2         Upgrade to nginx 1.27
3         Upgrade to nginx 1.27

kubectl get replicasets -o wide
NAME                          DESIRED   CURRENT   READY   AGE     CONTAINERS
IMAGES                               SELECTOR
nginx-deployment-6585597c84   0         0         0       6m35s   nginx
nginx:1.26   app=nginx,pod-template-hash=6585597c84
nginx-deployment-6ccb84987c   3         3         3       2m58s   nginx
nginx:1.27   app=nginx,pod-template-hash=6ccb84987c
nginx-deployment-7bdc5996d7   0         0         0       7m27s   nginx
nginx:1.25   app=nginx,pod-template-hash=7bdc5996d7
```

Visszaállás korábbi verzióra

```
# kubectl rollout undo deployment/nginx-deployment
deployment.apps/nginx-deployment rolled back

# kubectl get replicasets -o wide
NAME                          DESIRED   CURRENT   READY   AGE     CONTAINERS
IMAGES                               SELECTOR
nginx-deployment-6585597c84   3         3         3       12m     nginx
nginx:1.26   app=nginx,pod-template-hash=6585597c84
nginx-deployment-6ccb84987c   0         1         1       8m33s   nginx
nginx:1.27   app=nginx,pod-template-hash=6ccb84987c
nginx-deployment-7bdc5996d7   0         0         0       13m     nginx
nginx:1.25   app=nginx,pod-template-hash=7bdc5996d7
```

Java/SpringBoot alkalmazás kubernetesbe költöztetése

Alkalmazás elkészítése

```
# mkdir -p minimal-spring-k8s/src/main/java/com/example/demo

# cat > minimal-spring-
k8s/src/main/java/com/example/demo/DemoApplication.java <<'EOF'
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
  public class DemoApplication {
  public static void main(String[] args) {
    SpringApplication.run(DemoApplication.class, args);
  }
}
EOF

# cat > minimal-spring-
k8s/src/main/java/com/example/demo/HomeController.java <<'EOF'
package com.example.demo;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
```

```
@Controller
public class HomeController {

  @Value("${SITE_TITLE:Alkalmazás}")
  private String siteTitle;

  @Value("${SITE_MESSAGE:Hello Kubernetes!}")
  private String siteMessage;

  @GetMapping("/")
  public String index(Model model) {
    model.addAttribute("title", siteTitle);
    model.addAttribute("message", siteMessage);
    return "index"; // templates/index.html
  }
}
EOF

# mkdir -p minimal-spring-k8s/src/main/resources/templates

# cat > minimal-spring-k8s/src/main/resources/templates/index.html <<'EOF'
<!doctype html>
<html lang="hu">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title th:text="${title}">Alkalmazás</title>
<style>
body { font-family: system-ui, -apple-system, Segoe UI, Roboto, Ubuntu,
Cantarell, 'Helvetica Neue', Arial, 'Noto Sans', 'Apple Color Emoji', 'Segoe
UI Emoji', 'Segoe UI Symbol';
margin: 0; padding: 2rem; background: #0f172a; color: #e2e8f0; }
.card { max-width: 680px; margin: 10vh auto; background: #111827; border-
radius: 16px; padding: 2rem; box-shadow: 0 10px 40px rgba(0,0,0,0.35); }
h1 { margin: 0 0 1rem; font-size: 2rem; }
p { margin: 0 0 1rem; font-size: 1.125rem; }
small { opacity: .7; }
code { background: #0b1020; padding: .25rem .4rem; border-radius: .4rem; }
</style>
</head>
<body>
<main class="card">
<h1 th:text="${title}">Alkalmazás</h1>
<p th:text="${message}">Hello Kubernetes!</p>
<small>Forrás: ConfigMap → env → @Value → Thymeleaf</small>
</main>
</body>
</html>
EOF

# cat > minimal-spring-k8s/src/main/resources/application.properties <<'EOF'
```

```
# Spring Boot alapbeállítások
server.port=${PORT:8080}
server.shutdown=graceful


# Actuator health végpont a kubernetes ellenőrzéshez
management.endpoints.web.exposure.include=health,info
management.endpoint.health.probes.enabled=true
EOF

# cat > minimal-spring-k8s/pom.xml  <<'EOF'
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>minimal-spring-k8s</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>minimal-spring-k8s</name>
  <description>Minimal Spring Boot app for Kubernetes with
ConfigMap</description>

  <properties>
    <java.version>21</java.version>
    <spring-boot.version>3.3.4</spring-boot.version>
  </properties>

  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-dependencies</artifactId>
        <version>${spring-boot.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
```

```
      <artifactId>spring-boot-starter-actuator</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
        <executions>
          <execution>
            <goals>
              <goal>repackage</goal>
            </goals>
          </execution>
        </executions>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
          <release>21</release>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
EOF
```

Alkalmazás fordítása a teszteléshez

```
# cd minimal-spring-k8s

# mvn clean package
[INFO] Scanning for projects...
...
[INFO] ------------------------------------------------------------------------
---
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
---
[INFO] Total time:  4.565 s
[INFO] Finished at: 2025-09-17T07:09:50+02:00
[INFO] ------------------------------------------------------------------------
---
```

Elkészült a **minimal-spring-k8s/target/minimal-spring-k8s-0.0.1-SNAPSHOT.jar** alkalmazás.

Konténer image készítése

Első megoldás: mindig friss alkalmazás készítése:

```
# cd minimal-spring-k8s

# cat > Dockerfile <<'EOF'
# Alkalmazás fordítása
FROM maven:3.9.8-eclipse-temurin-21-alpine AS build
WORKDIR /app
COPY pom.xml .
RUN mvn -q -e -B -DskipTests dependency:go-offline
COPY src ./src
RUN mvn -q -e -B -DskipTests package


# Konténer image készítés
FROM eclipse-temurin:21-jre-alpine
WORKDIR /app


# Spring Boot alkalmazás másolása
COPY --from=build /app/target/minimal-spring-k8s-*.jar app.jar


# A Spring Boot a PORT env változót ismeri
ENV PORT=8080
EXPOSE 8080


# JVM opciók konténeres környezethez
ENV JAVA_OPTS="-XX:+UseContainerSupport -XX:MaxRAMPercentage=75"
ENTRYPOINT ["sh", "-c", "java $JAVA_OPTS -jar app.jar"]
EOF
```

Másik megoldás: a már meglévő build használata

```
# cat > Dockerfile <<'EOF'
# Konténer image készítése
FROM eclipse-temurin:21-jre-alpine
WORKDIR /app


# Spring Boot alkalmazás másolása
COPY target/minimal-spring-k8s-*.jar app.jar
```

```
# A Spring Boot a PORT env változót ismeri
ENV PORT=8080
EXPOSE 8080


# JVM opciók konténeres környezethez
ENV JAVA_OPTS="-XX:+UseContainerSupport -XX:MaxRAMPercentage=75"
ENTRYPOINT ["sh", "-c", "java $JAVA_OPTS -jar app.jar"]
EOF
```

Konténer image készítése

```
# podman build -t minimal-spring-k8s:0.0.1 .
STEP 1/7: FROM eclipse-temurin:21-jre-alpine
STEP 2/7: WORKDIR /app
--> 45811f6fd665
STEP 3/7: COPY target/minimal-spring-k8s-*.jar app.jar
--> 9d027583908b
STEP 4/7: ENV PORT=8080
--> d8b7374f93ea
STEP 5/7: EXPOSE 8080
--> 5468f35be894
STEP 6/7: ENV JAVA_OPTS="-XX:+UseContainerSupport -XX:MaxRAMPercentage=75"
--> bec2bb2e08e7
STEP 7/7: ENTRYPOINT ["sh", "-c", "java $JAVA_OPTS -jar app.jar"]
COMMIT minimal-spring-k8s:0.0.1
--> f0d688f68506
Successfully tagged localhost/minimal-spring-k8s:0.0.1
f0d688f685065441108f94b6460d7ca3917c7f444d2a07a9993ac7f561a4f4e3
```

Image előkészítése és a registry-be töltése

```
# podman image tag localhost/minimal-spring-k8s:0.0.1 REGISTRY_URL/minimal-
spring-k8s:0.0.1

# podman push REGISTRY_URL/minimal-spring-k8s:0.0.1
Getting image source signatures
Copying blob cba3fb5670d7 done   |
Copying blob a6af48261b3d done   |
Copying blob 27d41fb27db9 done   |
Copying blob 4ac76939e813 done   |
Copying blob df603300ccbc done   |
Copying blob a5048fc1ae11 done   |
Copying config f0d688f685 done   |
Writing manifest to image destination
```

Kubernetes configmap, deployment, service definíciók elkészítése

```
# mkdir k8s
```

```
# cat >k8s/configmap.yaml <<'EOF'
apiVersion: v1
kind: ConfigMap
metadata:
  name: minimal-spring-config
  labels:
    app: minimal-spring-k8s
data:
  SITE_TITLE: "Kubernetesből jövő cím"
  SITE_MESSAGE: "Ez az üzenet ConfigMap-ból érkezik."
EOF

# cat > k8s/deployment.yaml <<'EOF'
apiVersion: apps/v1
kind: Deployment
metadata:
  name: minimal-spring-k8s
  labels:
    app: minimal-spring-k8s
spec:
  replicas: 2
  selector:
    matchLabels:
      app: minimal-spring-k8s
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 0
      maxSurge: 1
  template:
    metadata:
      labels:
        app: minimal-spring-k8s
    spec:
      terminationGracePeriodSeconds: 30
      containers:
        - name: app
          image: REGISTRY_URL/minimal-spring-k8s:0.0.1
          imagePullPolicy: IfNotPresent
          ports:
            - name: http
              containerPort: 8080
          envFrom:
            - configMapRef:
                name: minimal-spring-config
          startupProbe:
            httpGet:
              path: /actuator/health/liveness
              port: 8080
            failureThreshold: 30
            periodSeconds: 2
```

```
          readinessProbe:
            httpGet:
              path: /actuator/health/readiness
              port: 8080
            initialDelaySeconds: 5
            periodSeconds: 10
            timeoutSeconds: 2
            failureThreshold: 3
          livenessProbe:
            httpGet:
              path: /actuator/health/liveness
              port: 8080
            initialDelaySeconds: 10
            periodSeconds: 20
            timeoutSeconds: 2
            failureThreshold: 3
          resources:
            requests:
              cpu: "100m"
              memory: "128Mi"
            limits:
              memory: "512Mi"
EOF

# cat > k8s/service.yaml <<'EOF'
apiVersion: v1
kind: Service
metadata:
  name: minimal-spring-k8s
  labels:
    app: minimal-spring-k8s
spec:
  type: NodePort
  selector:
    app: minimal-spring-k8s
  ports:
    - name: http
      nodePort: 30001
      port: 80
      targetPort: 8080
EOF
```

Kubernetes műveletek

```
# kubectl get all
NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   43h

# kubectl apply -f k8s/configmap.yaml
configmap/minimal-spring-config created
```

```
# kubectl apply -f k8s/deployment.yaml
deployment.apps/minimal-spring-k8s created

# kubectl apply -f k8s/service.yaml
service/minimal-spring-k8s created

# kubectl get all,cm
NAME                                        READY   STATUS    RESTARTS   AGE
pod/minimal-spring-k8s-6d956c4c9f-n6rb2   1/1     Running   0          4m33s
pod/minimal-spring-k8s-6d956c4c9f-vj8tc   1/1     Running   0          4m33s

NAME                           TYPE        CLUSTER-IP     EXTERNAL-IP
PORT(S)        AGE
service/kubernetes             ClusterIP   10.96.0.1      <none>
443/TCP        43h
service/minimal-spring-k8s   NodePort    10.106.11.23   <none>
80:30001/TCP   4m25s

NAME                                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/minimal-spring-k8s   2/2     2            2           4m33s

NAME                                              DESIRED   CURRENT   READY
AGE
replicaset.apps/minimal-spring-k8s-6d956c4c9f   2         2         2
4m33s

NAME                                DATA   AGE
configmap/kube-root-ca.crt          1      43h
configmap/minimal-spring-config   2      4m40s
```

Módosítások a configmap tartalmában

```
# cat > k8s/configmap.yaml <<'EOF'
apiVersion: v1
kind: ConfigMap
metadata:
  name: minimal-spring-config
  labels:
    app: minimal-spring-k8s
data:
  SITE_TITLE: "Kubernetesből jövő új cím"
  SITE_MESSAGE: "Ez az üzenet az űrből érkezett."
EOF

# kubectl apply -f k8s/configmap.yaml
configmap/minimal-spring-config configured

# kubectl rollout restart deployment/minimal-spring-k8s
deployment.apps/minimal-spring-k8s restarted

# kubectl get all
```

```
NAME                                        READY   STATUS    RESTARTS   AGE
pod/minimal-spring-k8s-5d757fcb88-km5bn   1/1     Running   0          5m26s
pod/minimal-spring-k8s-5d757fcb88-rtfrv   1/1     Running   0          5m41s

NAME                          TYPE        CLUSTER-IP      EXTERNAL-IP
PORT(S)        AGE
service/kubernetes            ClusterIP   10.96.0.1       <none>
443/TCP        47h
service/minimal-spring-k8s    NodePort    10.98.248.233   <none>
80:30001/TCP   33m

NAME                                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/minimal-spring-k8s   2/2     2            2           33m

NAME                                               DESIRED   CURRENT   READY
AGE
replicaset.apps/minimal-spring-k8s-57d696db7c   0         0         0
33m
replicaset.apps/minimal-spring-k8s-5d757fcb88   2         2         2
5m41s
```

Kubernetes natív LB megoldás

A konfigurálást a control plane gépen végezzük el

MetalLB telepítése

```
# kubectl apply -f
https://raw.githubusercontent.com/metallb/metallb/v0.13.12/config/manifests/
metallb-native.yaml
namespace/metallb-system created
customresourcedefinition.apiextensions.k8s.io/addresspools.metallb.io
created
customresourcedefinition.apiextensions.k8s.io/bfdprofiles.metallb.io created
customresourcedefinition.apiextensions.k8s.io/bgpadvertisements.metallb.io
created
customresourcedefinition.apiextensions.k8s.io/bgppeers.metallb.io created
customresourcedefinition.apiextensions.k8s.io/communities.metallb.io created
customresourcedefinition.apiextensions.k8s.io/ipaddresspools.metallb.io
created
customresourcedefinition.apiextensions.k8s.io/l2advertisements.metallb.io
created
serviceaccount/controller created
serviceaccount/speaker created
role.rbac.authorization.k8s.io/controller created
role.rbac.authorization.k8s.io/pod-lister created
clusterrole.rbac.authorization.k8s.io/metallb-system:controller created
clusterrole.rbac.authorization.k8s.io/metallb-system:speaker created
```

```
rolebinding.rbac.authorization.k8s.io/controller created
rolebinding.rbac.authorization.k8s.io/pod-lister created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:controller
created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:speaker created
configmap/metallb-excludel2 created
secret/webhook-server-cert created
service/webhook-service created
deployment.apps/controller created
daemonset.apps/speaker created
validatingwebhookconfiguration.admissionregistration.k8s.io/metallb-webhook-
configuration created
```

Publikus IP tartomány megadása

```
# cat > ~/metallb-l2.yaml <<'EOF'
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: pool-l2
  namespace: metallb-system
spec:
  addresses:
    - 192.168.110.170-192.168.110.179
---
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: l2adv
  namespace: metallb-system
spec:
  ipAddressPools:
    - pool-l2
EOF
```

MetalLB podjainak ellenőrzése

```
# kubectl -n metallb-system get pods
NAME                          READY   STATUS    RESTARTS   AGE
controller-7dbf649dcc-w4frr   1/1     Running   0          2m17s
speaker-4nkqt                 1/1     Running   0          2m17s
speaker-q4h2p                 1/1     Running   0          2m17s
speaker-vxp69                 1/1     Running   0          2m17s
```

Konfiguráció alkalmazása (amennyiben a pod-ok Ready/Running állapotban vannak)

```
# kubectl apply -f metallb-l2.yaml
ipaddresspool.metallb.io/pool-l2 created
l2advertisement.metallb.io/l2adv created
```

A metallb-system névtér ellenőrzése

```
# kubectl get all -n metallb-system
NAME                                 READY   STATUS    RESTARTS   AGE
pod/controller-7dbf649dcc-w4frr   1/1     Running   0          49m
pod/speaker-4nkqt                 1/1     Running   0          49m
pod/speaker-q4h2p                 1/1     Running   0          49m
pod/speaker-vxp69                 1/1     Running   0          49m

NAME                      TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
AGE
service/webhook-service   ClusterIP   10.104.247.76   <none>        443/TCP
49m

NAME                      DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE
NODE SELECTOR             AGE
daemonset.apps/speaker    3         3         3       3            3
kubernetes.io/os=linux    49m

NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/controller    1/1     1            1           49m

NAME                                     DESIRED   CURRENT   READY   AGE
replicaset.apps/controller-7dbf649dcc    1         1         1       49m
```

Teszt deployment létrehozása és ellenőrzése

```
# kubectl create deploy nginx --image=nginx:stable --port=80
deployment.apps/nginx created

# kubectl expose deploy nginx --type=LoadBalancer --port=80 --target-port=80
service/nginx exposed

# kubectl get svc nginx
NAME    TYPE           CLUSTER-IP     EXTERNAL-IP       PORT(S)        AGE
nginx   LoadBalancer   10.109.25.42   192.168.110.170   80:31422/TCP   13s
```

Amennyiben megjelent az EXTERNAL-IP oszlopban a definiált tartomány egyik IP címe, akkor tesztelhető a szolgáltatás

```
# curl -I http://192.168.110.170
HTTP/1.1 200 OK
Server: nginx/1.28.0
Date: Thu, 25 Sep 2025 17:17:34 GMT
Content-Type: text/html
Content-Length: 615
Last-Modified: Wed, 23 Apr 2025 11:48:54 GMT
Connection: keep-alive
ETag: "6808d3a6-267"
Accept-Ranges: bytes
```

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

Permanent link:
https://wiki.r-l.hu/doku.php?id=kubernetes:gyakorlatok&rev=1758820699

Last update: **2025/09/25 17:18**