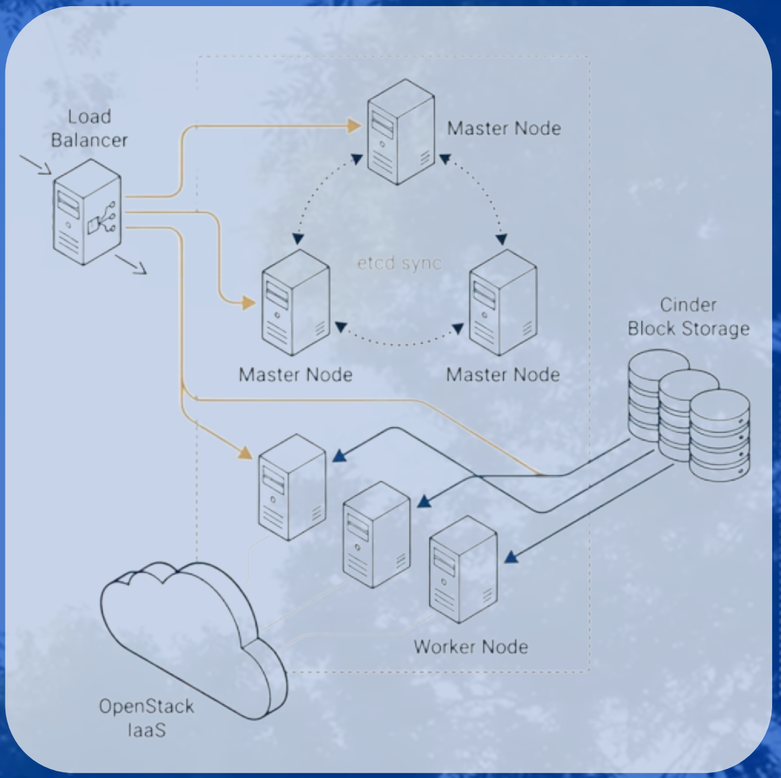
T.C. SANAYİ VE TEKNOLOJİ BAKANLIĞI

#MİLLİ TEKNOLOJİ HAMLESİ

TÜBİTAK

OpenStack Üzerinde Yüksek Erişilebilir RKE2 Kurulumu



openstack. kubernetes

TÜBİTAK ULAKBİM

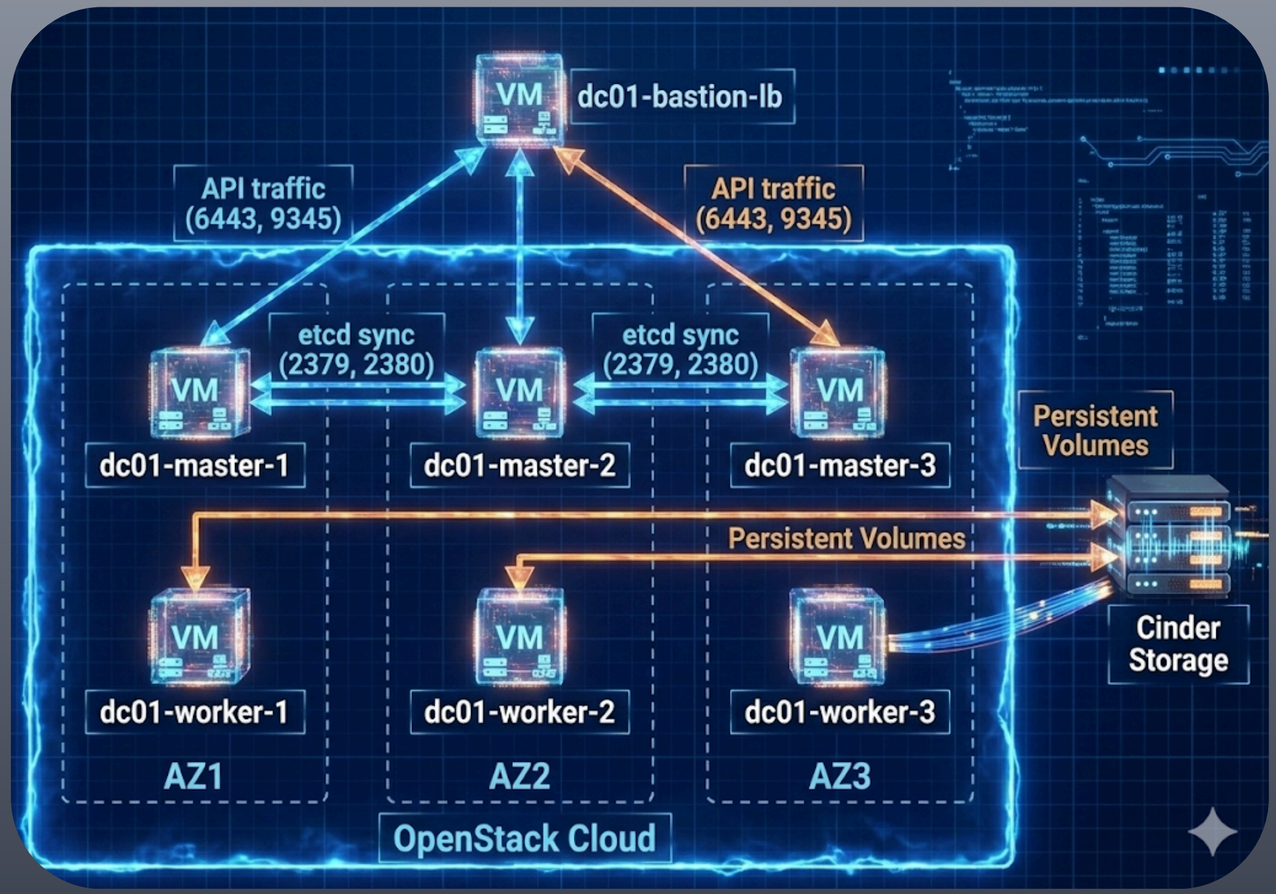
Dayanıklı, Ölçeklenebilir ve Durum Bilgili (Stateful)

OpenStack altyapısı üzerinde, tek bir hata noktası barındırmayan (no single point of failure), üretime hazır bir Kubernetes cluster'ı inşa edeceğiz. Mimarimiz üç ana katmandan oluşmaktadır:

Altyapı (IaaS): OpenStack üzerinde sanal makineler, ağlar ve güvenlik grupları.

Orkestrasyon (Kubernetes): Yüksek erişilebilirlik (HA) için 3 Master ve 3 Worker node'dan oluşan RKE2 cluster'ı. Master node'lar farklı Availability Zone'lara (AZ) dağıtılmıştır.

Kalıcı Depolama (Storage): Cinder CSI eklentisi ile Kubernetes pod'ları için dinamik olarak sağlanan kalıcı block storage.



Temelin Atılması - Cluster Ağı ve Yönlendiricinin Oluşturulması

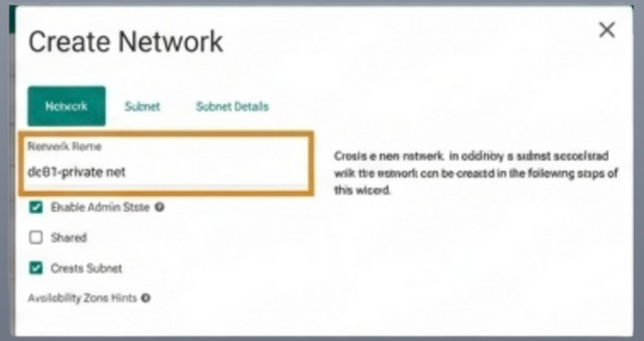
Ağ (Network) Oluşturma

Neden?

Node'ların kendi aralarında ve RKE2 kurulumu için gerekli paketleri indirebilmek amacıyla internet ile iletişim kurabilmesi için izole bir sanal ağ gereklidir.

Nasıl ?

OpenStack arayüzünde **`dc01-private-net`** adında bir ağ oluşturulur.



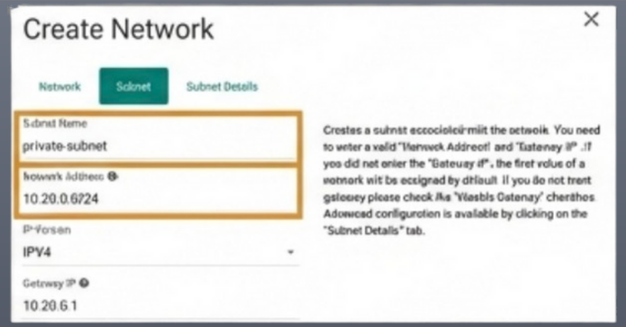
Alt Ağ (Subnet) Oluşturma

Nasıl ?

private-subnet adıyla **`10.20.0.0/24`** CIDR aralığı tanımlanır.

Kritik Not

Bu IP aralığı, Kubernetes Service CIDR ve Pod CIDR aralıkları ile çakışmamalıdır.



Yönlendirici (Router) Oluşturma

Neden?

private-net üzerindeki sanal makinelerin dış ağa (internete) erişimini sağlamak için.

Nasıl ?

dc01-router adında bir yönlendirici oluşturulur ve **private-subnet**'e bir arayüz olarak eklenir.

Güvenli İletişim - Cluster İçi Güvenlik Grubu (dc01-internal-sec)

Bu güvenlik grubu, Master ve Worker node'lar arasındaki tüm gerekli RKE2, etcd ve CNI trafiğine izin verir. Kurallar, en az yetki prensibine göre sadece cluster içi (10.20.0.0/24) iletişime izin verecek şekilde yapılandırılmıştır.

Port	Protokol	Kaynak	Açıklama
22	TCP	10.20.0.0/24	Yönetim için SSH erişimi
-	ICMP	10.20.0.0/24	Ağ sorunlarını giderme (Ping)
80, 443	TCP	10.20.0.0/24	Ingress-nginx tarafından kullanılan HTTP/S trafiği
2379-2381	TCP	10.20.0.0/24	etcd client, peer ve metrik portları (HA için kritik)
6443	TCP	10.20.0.0/24	Kubernetes API Server erişimi
9345	TCP	10.20.0.0/24	RKE2 Supervisor API (Node katılımı için)
10250	TCP	10.20.0.0/24	Kubelet API (API Server'dan node'a erişim)
30000-32767	TCP/UDP	10.20.0.0/24	NodePort servis aralığı
8472, 51820, 51821	UDP	10.20.0.0/24	Canal CNI (VXLAN, WireGuard) pod-to-pod ağı
9099	TCP	10.20.0.0/24	Canal CNI sağlık kontrolleri

Dış Erişim Kapısı - Load Balancer Kurulumu

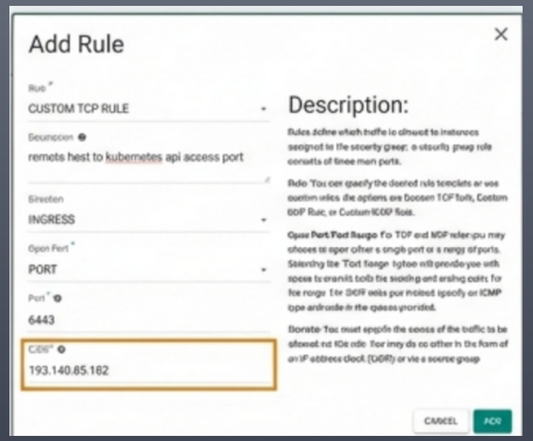
Bölüm 1: Güvenlik Grubu (dc01-sec)

Amaç

Dışarıdan yalnızca belirli IP adreslerinden gelen trafiğe izin vererek cluster'ı korumak.

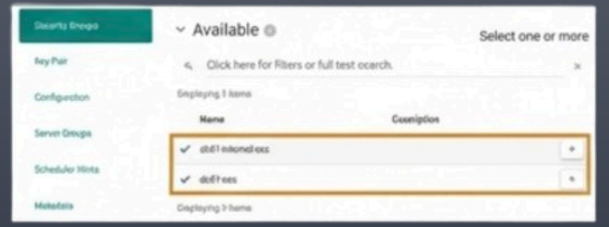
Kurallar

SSH (22), HTTPS (443), HTTP (80) ve Kubernetes API (6443) portlarına sadece **kendi genel IP adresinizden** (/32) erişim izni verilir.



Bölüm 2: Sanal Makine (dc01-bastion-lb)

- **Özellikler**: Ubuntu 22.04, 4 vCPU, 8 GB RAM
- **Ağ**: dc01-private-net ağına bağlanır.
- **Güvenlik**: Hem dış erişim için dc01-sec hem de cluster içi iletişim için dc01-internal-sec gruplarına atanır.
- **Erişim**: Bir Key Pair atanır ve dış dünyadan erişim için bir Floating IP ilişkilendirilir.

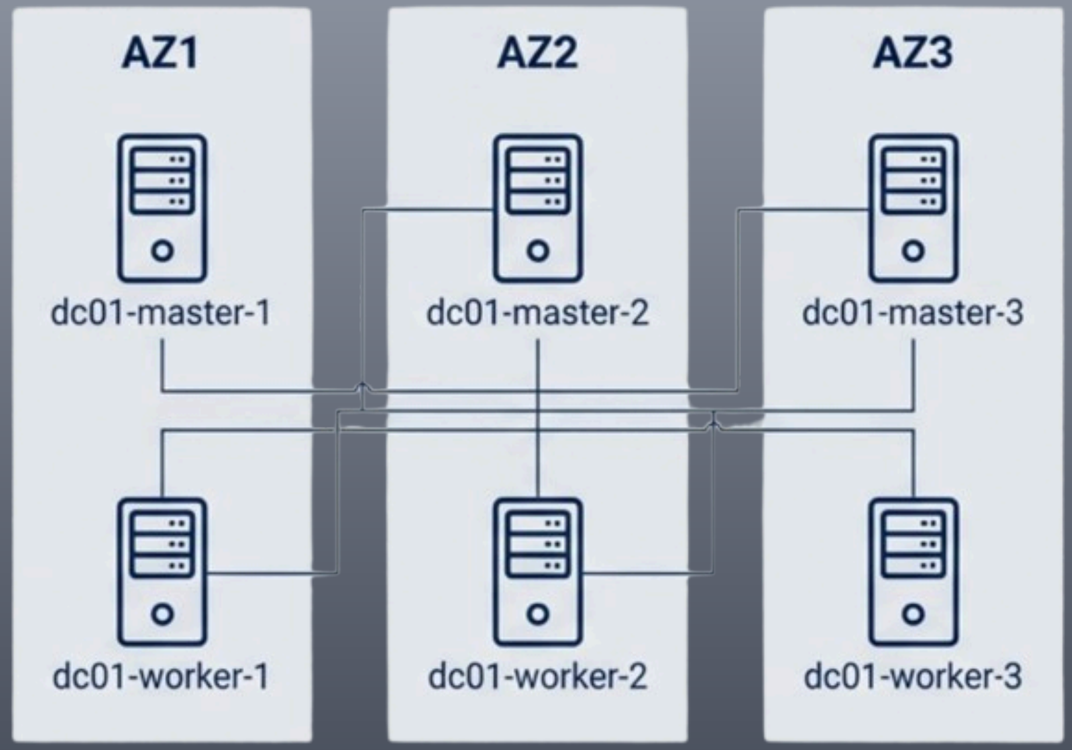


Yüksek Erişilebilirlik - Master ve Worker Node'ların Dağıtımı

Strateji: Availability Zone (AZ) Dağılımı

Neden?

Tek bir AZ'de yaşanacak donanım veya ağ arızasının tüm cluster'ı etkilemesini önlemek için Master ve Worker node'lar farklı fiziksel konumlara (AZ'lere) dağıtılır.



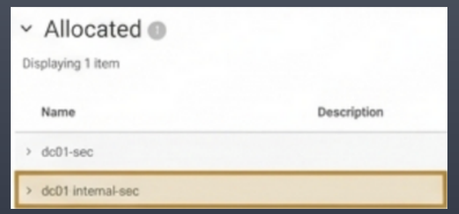
Özellikler (Flavor)

8 vCPU, 8 GB RAM, 100 GB Disk. Bu kaynaklar; etcd kararlılığı, RKE2 bileşenlerinin performansı ve gelecekteki ölçeklenebilirlik için seçilmiştir.

Name	VCPUS	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
fkm.c8m8.d100	8	8 GB	100 GB	100 GB	0 GB	Yes

Temel Yapılandırma

Tüm node'lar `dc01-private-net ağına ve `dc01-internal-sec güvenlik grubuna atanır. Load node'lar'cm Balancer'dan güvenli SSH erişimi için özel Keiy Pair ('dc01-bastion-ssh') kullanılır.



Trafik Yönetimi - Yüksek Erişilebilirlik için HAProxy Yapılandırması

Amaç

HAProxy, gelen istekleri sağlıklı Master ve Worker node'lara yönlendirerek Kubernetes API ve uygulamalar için kesintisiz bir erişim noktası sağlar.

Kurulum

```
apt install -y haproxy
```

Yapılandırma (`/etc/haproxy/haproxy.cfg`)

```
# Kubernetes API (6443) - Master'lara yönlendirir
frontend k8s_api
    bind 10.20.0.251:6443 <-- LB'nin özel IP'si kullanılır
    mode tcp
    default_backend kBs_masters
backend kBs_masters
    mode top
    balance roundrobin
    server master1 10.20.0.27:6443 check <-- Tüm master node IP'leri eklenir
    # ... diğer masterlar

# RKE2 Server Join (9345) - Master'lara yönlendirir
frontend rke2_server
    bind 10.20.0.251:9345
    # ...

# Ingress HTTP/HTTPS (80, 443) - Worker'lara yönlendirir
frontend ingress_https
    bind *:443
    # ...
```

Aktivasyon

```
systemctl restart haproxy
```

Cluster'ın Başlatılması - Node Hazırlığı ve İlk Master Bootstrap

Bölüm 1: Node Hazırlığı (Tüm Master ve Worker Node'larda)

1. Tüm node'lara Load Balancer üzerinden SSH ile bağlanılır ve `root kullanıcısına geçilir.
2. Swap devre dışı bırakılır:

```
swapoff -a
```

```
(ve /etc/fstab düzenlenir)
```

3. IP yönlendirme etkinleştirilir:

```
sysctl -w net.ipv4.ip_forward=1
```

Bölüm 2: İlk Master Kurulumu (`dc01-master-1`)

Yapılandırma (`/etc/rancher/rke2/config.yaml)

```
token: super-secret-token
# server: https://10.20.0.251:6443
tls-san:
  - 10.20.0.251    # HAProxy IP
node-name: dc01-master-1
```

Bootstrap için bu satır yorumlu kalmalı

Kurulum ve Başlatma

```
curl -sfL https://get.rke2.io | sh -
systemctl enable --now rke2-server
```

Doğrulama

```
export KUBECONFIG=/etc/rancher/rke2/rke2.yaml
/var/lib/rancher/rke2/bin/kubectl get nodes
```

```
NANE          STATUS   ROLES                       AGE   VERSION
dc01-master-1 Ready    control-plane,etcd,master   2m    v1.24.4+rke2r1
```

Cluster'ı Büyütme - Diğer Node'ların Katılımı

Bölüm 1: Diğer Master'ların Katılımı (`dc01- master-2`, `dc01-master-3`)

Farklılık: config.yaml dosyasında, cluster'a katılmak için server parametresi aktif hale getirilir ve Load Balancer'ın 9345 portunu işaret eder.

Yapılandırma (config.yaml)

```
server: https://10.20.0.251:9345
token: super-secret-token
tls-san:
  - 10.20.0.251
node-name: dc01-master-2 # Her node için güncellenir
```

Kurulum, ilk master ile aynı komutlarla (**systemctl start rke2-server**) yapılır.

Bölüm 2: Worker Node'ların Katılımı (`dc01-worker-1/2/3`)

Farklılık: Kurulum, INSTALL_RKE2_TYPE="agent" parametresi ile yapılır ve rke2-agent servisi başlatılır.

Yapılandırma (config.yaml)

Master katılımıyla aynıdır, sadece `node-name` güncellenir.

Kurulum ve Başlatma

```
curl -sfL https://get.rke2.io | INSTALL_RKE2_TYPE="agent" sh -
systemctl enable --now rke2-agent
```

Merkezi Yönetim - Load Balancer'dan `kubectl` Erişimi

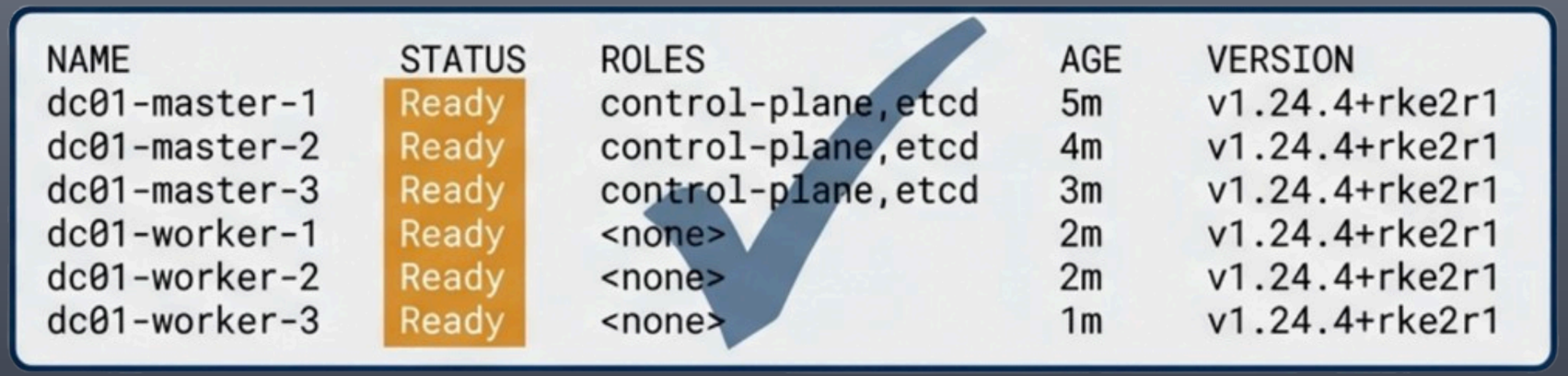
Cluster'ı yönetmek için her seferinde bir master node'a bağlanmak yerine, tüm işlemleri merkezi ve yüksek erişilebilir bir nokta olan Load Balancer üzerinden yapmak.

Adımlar

- **Kubeconfig Kopyalama**: Master node'lardan birindeki`/etc/rancher/rke2/rke2.yaml` dosyası, Load Balancer VM'ine `scp`ile kopyalanır.
- **IP Güncelleme**: Kopyalanan `kubeconfig` dosyasında `server:`adresi 127.0.0.1'den Load Balancer'ın özel IP'sine (10.20.0.251`) değiştirilir.
- **kubectl Kurulumu**: Load Balancer VM'ine `kubectl` binary'si indirilir ve kurulur.

Doğrulama Kontrol Noktası

kubectl get nodes

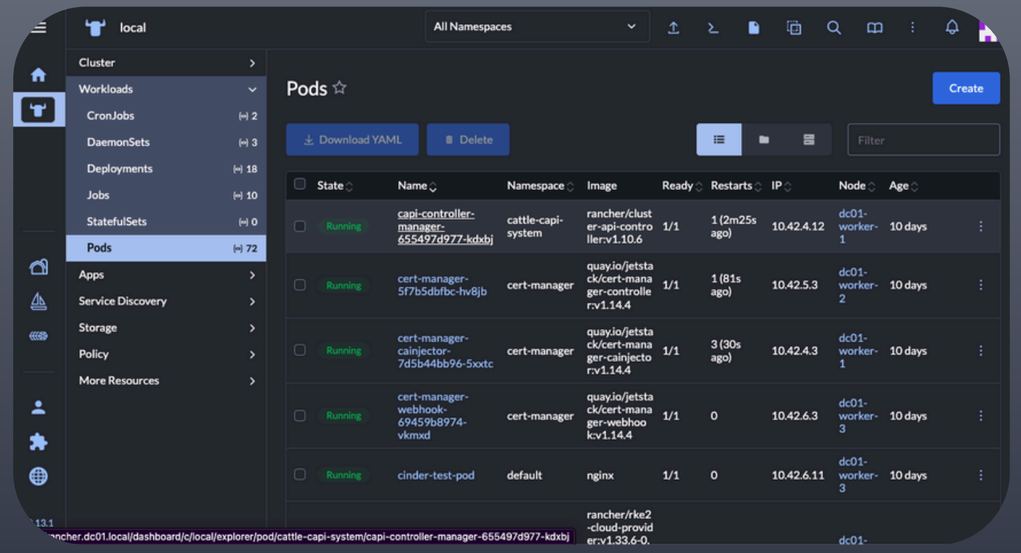


Beklenen Çıktı: 3'ü `control-plane,etcd` ve 3'ü <none> rolünde olan toplam 6 node'un `Ready durumunda listelendiği gösterilir. Bu, HA RKE2 cluster kurulumunun başarıyla tamamlandığını doğrular.

Kubernetes Yönetimini UI ile Basitleştirme

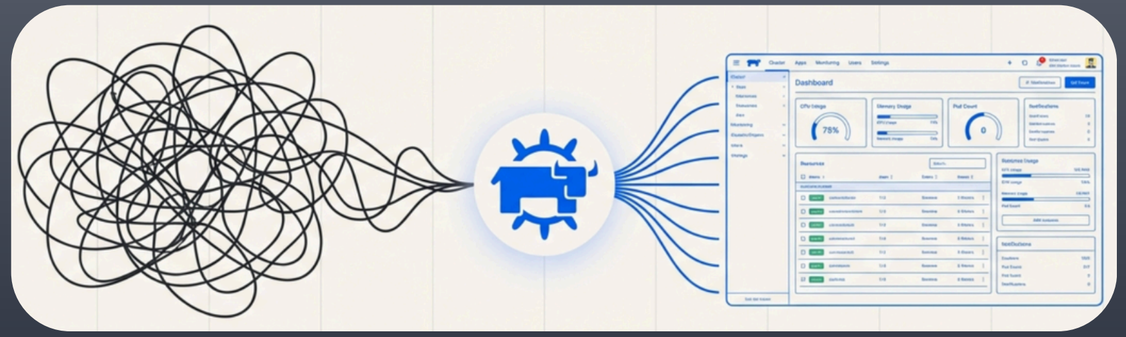
`kubectl` ile Cluster Yönetimi Bazen Yorucu Olabilir

`kubectl, Kubernetes için güçlü bir komut satırı aracıdır. Ancak, karmaşık cluster'ları yalnızca terminal üzerinden yönetmek, zamanla verimliliği düşürebilir ve yorucu hale gelebilir.



Cluster'ınız İçin Sezgisel Bir Kontrol Paneli: Rancher UI

Rancher Ul ile cluster'ınız üzerinde tam anlamıyla görsel bir kontrol sağlayın. Dağıtımları, kaynakları ve konfigürasyonları tek bir merkezi arayüzden yönetin.



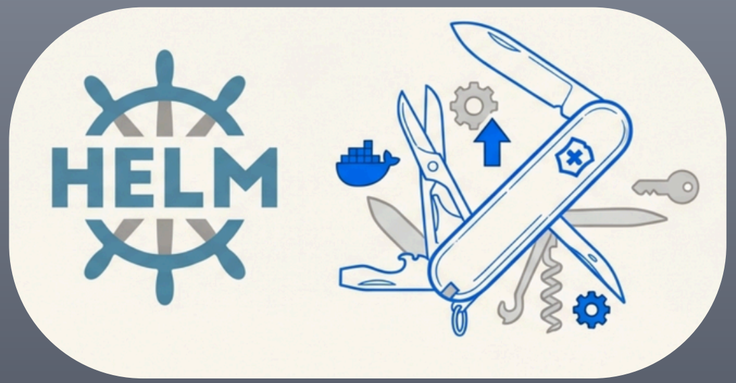
Karmaşık Kubernetes Yönetimi

Basitleştirilmiş Rancher UI

Rancher UI için Ön Koşullar

Kurulumdaki Yardımcımız: Helm Paket Yöneticisi

Helm, Kubernetes uygulamalarını kolayca dağıtmanızı, yükseltmenizi ve yönetmenizi sağlayan bir paket yöneticisidir. Rancher kurulumunu bizim için otomatikleştirecek.



Helm Kurulumu

```
curl -fsSL -o get_helm.sh
https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-4
chmod 700 get_helm.sh
./get_helm.sh
```

Cert-Manager'ı Kurun

Rancher, kendi sertifikalarını yönetmek için Cert-Manager CRD'lerine (Custom Resource Definitions) ihtiyaç duyar. Bu nedenle kuruluma başlamadan önce bu bağımlılığı eklemeliyiz.

```bash
kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/v1.14.4/
cert-manager.yaml
```

Rancher Kurulumuna Hazırlık (Repo Ekleme)

```
helm repo add rancher-latest https://releases.rancher.com/server-charts/latest
helm repo update
```

Rancher UI Dağıtım ve Erişim

RancherUI Helm ile Deploy

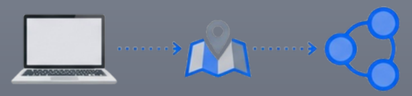
```
kubectl create namespace cattle-system

helm install rancher rancher-latest/rancher \
  --namespace cattle-system \
  --set hostname=rancher.dc01.com
```

ÖNEMLİ: Bu adresi Rancher arayüzüne erişeceğiniz kendi alan adınızla değiştirin.

Yerel Makineden Erişimi Yapılandırma

Rancher arayüzüne tarayıcınızdan erişebilmek için, bir önceki adımda belirttiğiniz alan adını cluster'ınızın IP adresiyle eşleştirmeniz gerekir. Bunun için yerel bilgisayarınızdaki`hosts` dosyasını düzenleyin.

```
# Örnek:
[CLUSTER_IP_ADRESINIZ] rancher.dc01.com
```



Yönetici Parolasını Alma

İlk kurulumda Rancher, geçici bir yönetici parolası oluşturur. Bu parolayı cluster içerisindeki bir 'secret' objesinden alarak ilk girişi yapabilirsiniz.

```
kubectl get secret --namespace cattle-system
bootstrap-secret -o
go-template='{{.data.bootstrapPassword|base64decode}}'
```

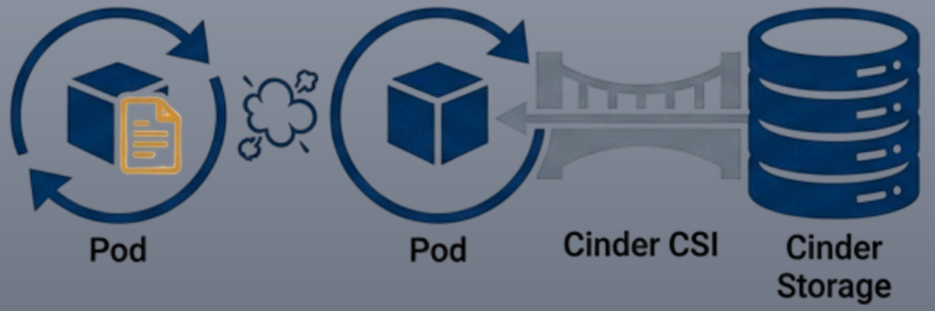
Cluster'e Erişim ve Kontrol Sizde

Kurulum tamamlandı. Artık belirlediğiniz adrese tarayıcınızdan gidip, az önce aldığınız parola ile giriş yapabilirsiniz.

Kalıcı Depolama Entegrasyonu - OpenStack Cinder CSI Eklentisi

Problem

Kubernetes pod'ları ve onların dosya sistemleri geçicidir. Bir pod yeniden başladığında içindeki veriler kaybolur. Veritabanları gibi durum bilgili (stateful) uygulamalar için verilerin kalıcı olarak saklanması gerekir.



Çözüm: Cinder CSI (Container Storage Interface):

CSI, Kubernetes'in harici depolama sistemleriyle konuşmasını sağlayan standart bir arayüzdür. OpenStack Cinder CSI eklentisi, Kubernetes'in OpenStack'in block storage servisi olan Cinder'ı kullanarak dinamik olarak kalıcı diskler (Persistent Volume) oluşturmasına ve bunları pod'lara bağlamasına olanak tanır.

Sürece Genel Bakış:

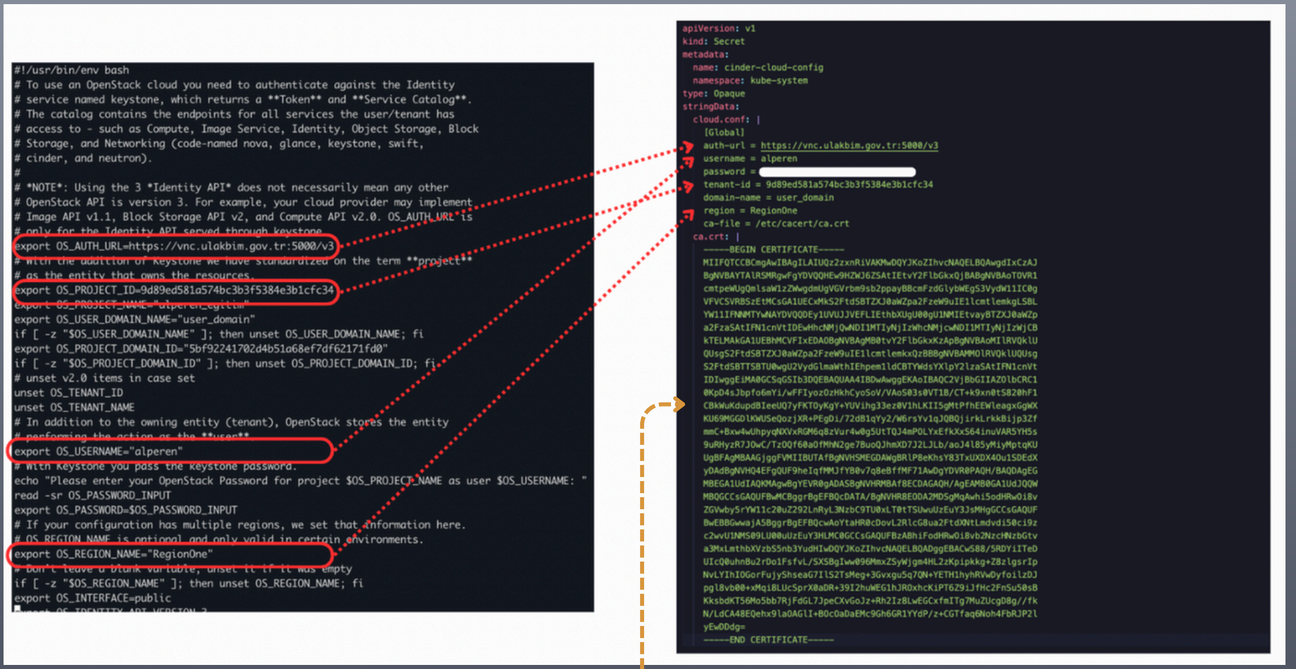
- Manifest dosyalarını indir
- OpenStack kimlik bilgileri için bir Kubernetes Secret oluştur
- Manifest dosyalarını güncelle ve eklentiyi kur
- Dinamik depolamayı test et

Güvenli Bağlantı - Cinder CSI Secret Yapılandırması

Kubernetes cluster'ının, Cinder volume'leri oluşturmak için OpenStack API'si ile güvenli bir şekilde kimlik doğrulaması yapmasını sağlamak.

Adım 1: Kimlik Bilgilerini Edinme

OpenStack arayüzünden, projeye özel 'OpenStack RC File` indirilir. Bu dosya, 'auth_url, `project_id, 'username' ve 'password gibi bilgileri içerir.



```
#!/usr/bin/env bash
# To use an OpenStack cloud you need to authenticate against the Identity
# service named keystone, which returns a **Token** and **Service Catalog**.
# The catalog contains the endpoints for all services the user/tenant has
# access to - such as Compute, Image Service, Identity, Object Storage, Block
# Storage, and Networking (code-named nova, glance, keystone, swift,
# cinder, and neutron).
#
# *NOTE*: Using the 3 *Identity API* does not necessarily mean any other
# OpenStack API is version 3. For example, your cloud provider may implement
# Image API v1.1, Block Storage API v2, and Compute API v2.0. OS_AUTH_URL is
# only for the Identity API served through keystone
export OS_AUTH_URL=https://vnc.ulakbim.gov.tr:5000/v3
# With the addition of Keystone we have standardized on the term **project**
# as the entity that owns the resources.
export OS_PROJECT_ID=9d89ed581a574bc3b3f5384e3b1cfc34
export OS_PROJECT_NAME="alperen_egitim"
export OS_USER_DOMAIN_NAME="user_domain"
if [ -z "$OS_USER_DOMAIN_NAME" ]; then unset OS_USER_DOMAIN_NAME; fi
export OS_PROJECT_DOMAIN_ID="5bf92241702d4b51a68ef7df62171fd0"
if [ -z "$OS_PROJECT_DOMAIN_ID" ]; then unset OS_PROJECT_DOMAIN_ID; fi
# unset v2.0 items in case set
unset OS_TENANT_ID
unset OS_TENANT_NAME
# In addition to the owning entity (tenant), OpenStack stores the entity
# performing the action as the **user**.
export OS_USERNAME="alperen"
# With Keystone you pass the keystone password.
echo "Please enter your OpenStack Password for project $OS_PROJECT_NAME as user $OS_USERNAME: "
read -sr OS_PASSWORD_INPUT
export OS_PASSWORD=$OS_PASSWORD_INPUT
# If your configuration has multiple regions, we set that information here.
# OS_REGION_NAME is optional and only valid in certain environments.
export OS_REGION_NAME="RegionOne"
# Don't leave a blank variable, unset it if it was empty
if [ -z "$OS_REGION_NAME" ]; then unset OS_REGION_NAME; fi
export OS_INTERFACE=public
```

```
apiVersion: v1
kind: Secret
metadata:
  name: cinder-cloud-config
  namespace: kube-system
type: Opaque
stringData:
  cloud.conf: |
    [Global]
    auth-url = https://vnc.ulakbim.gov.tr:5000/v3
    username = alperen
    password = [redacted]
    tenant-id = 9d89ed581a574bc3b3f5384e3b1cfc34
    domain-name = user_domain
    region = RegionOne
    ca-file = /etc/cacert/ca.crt
  ca.crt: |
    -----BEGIN CERTIFICATE-----
    MIIFQTCCBCmgAwIBAgILAIUQz2zxnRiVAKMwDQYJKoZIhvcNAQELBQAwgdIxCzAJ
    BgNVBAYTAlRSMRgwFgYDVQQHEw9HZWJ6ZSAtIEtvY2FlbGkxQjBABgNVBAoTOVR1
    cmtpeWUgQmlsaW1zZWwgdmUgVGVrbm9sb2ppayBBcmFzdGlybWEgS3VydW1lIC0g
    VFVCSVRBSzEtMCsGA1UECxMkS2FtdSBTZXJ0aWZpa2FzeW9uIEilcmtlemkgLSBL
    YWl1IFNMMTYwNAYDVQQDEy1UVUJJVEFLIEthbXUgU0dQU1NMIEtvayBTZXJ0aWZp
    a2FzaSAtIFN1cnVIDEwHhcNMjQwNDI1MTIyNjIzWhcNMjcwNDI1MTIyNjIzWjCB
    kTELMAkGA1UEBhMCVFIxEDAOBgNVBAgMB0tvY2FlbGkxKzApBgNVBAoMIlRVQklU
    QUsgS2FtdSBTZXJ0aWZpa2FzeW9uIE1lcmtlemkxQzBBBgNVBAMMOlRVQklUQUsg
    S2FtdSBTTSBTU0wgU2VydGlmaWthIEhpem1ldCBTYWdsYXlpY2lzaSAtIFN1cnVt
    IDIwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC2VjBbGIIAZOlbCRC1
    0KpD4sJbpfo6mYi/wFFIyozOzHkhCyoSoV/VAoS03s0VT1B/CT+k9xn0tSB20hF1
    CBkWuKdupd8IeeUQ7yFKTOyKgY+YUVihg33ez0VlhLKII5gMtPfhEEWleagxGgWX
    KU69MGGDlKWUSeQozjXR+PEgDi/72dB1qYy2/W6rsYv1qJQBQjirkLrkkBijp3Zf
    mmC+Bxw4wUhpyqNXVxRGM6qBzVur4w0g5UtTQJ4mPOLYxEfkXx564inuVAR5YH5s
    9uRHyzRTJOwC/TzOQf60aOfMhN2ge7BuoQJhmXD7J2LJLb/aoJ4l85yMiyMptqKU
    UgBFAgMBAAGjggFVMIIBUTAfBgNVHSMEGDAWgBRlP8eKhsY83TxUXDX4Ou1SDEdX
    yDAdBgNVHQ4EFgQUF9heIqfMMJfYB0v7q8eBffMF71AwDgYDVR0PAQH/BAQDAgEG
    MBEGA1UdIAQKMAgwBgYEVR0gADASBgNVHRMBAf8ECDAGAQH/AgEAMB0GA1UdJQQW
    MBQGCCsGAQUFBwMCBggrBgEFBQcDATA/BgNVHR8EODA2MDSgMqAwhi5odHRwOi8v
    ZGVwby5rYW11c20uZ29zLnRyL3NzbC9TU0xLT0tTSUwdzEuY3JsMGgGCCsGAQUF
    BwEBBGowajA5BggrBgEFBQcwAoYtaHR0cDovL2RlcGBua2FtdXNtLmdvdi50ci9z
    c2wvU1NMS09LU00uUzEuY3HLMC0GCCsGAQUFBzABhiFodHRwOi8vb2NzcHNzbGtv
    a3MxLmthbXVzb55nb3YudHIwDQYJKoZIhvcNAQELBQADggEBACwSB8/5RDYiITeD
    UIcQ0uhnBu2rDo1FsfvL/SXSBgIww096MmxZSyWjgm4HL2zKpipkkg+Z8zlgsrIp
    NvLYIhIOGorFujyShseaG7IlS2TsMeg+3Gvxgu5q7QN+YETH1hyhRVwDyfoilzDJ
    pgl8vb08+xMqi8LUcSprX0aDR+39I2huWEG1hJROxhcKiPT6Z9iJfHc2FnSu50sB
    KksbdKT56Mo5bb7RjFdGL7JpeCXvGoJz+Rh2Iz8LwEGCxfmITg7MuZUcgD8g//fk
    N/LdCA4BEQehx9laOAGlI+B0cOaDaEMc9Gh6GR1YYdP/z+CGTfaq6NoH4FbRJP2l
    yEwDdg=
    -----END CERTIFICATE-----
```

OpenStack RC File

csi-secret-cinderplugin.yaml

Adım 2: Kubernetes Secret Oluşturma

Cinder CSI manifestleri içindeki 'csi-secret-cinderplugin.yaml dosyası düzenlenir. RC dosyasındaki bilgiler, YAML dosyasındaki ilgili alanlara (base64 ile encode edilerek) girilir.

Adım 3: Kök Sertifika Ekleme (CA)

Güvenli TLS iletişimi için OpenStack ortamının kök sertifikası alınır ve Secret dosyasındaki 'ca.crt' alanına eklenir.

Eklentiyi Kurma ve Doğrulama

Bölüm 1: Manifestleri Sonlandırma

`cinder-csi-controller-plugin.yaml've cinder-csi- nodeplugin.yaml' dosyalarında, önceki adımda oluşturulan cinder-cloud-config Secret'ının ve sertifikanın volume olarak mount edildiğinden emin olunur.

```
# volumeMounts kısmı:
- name: secret-cinderplugin
  mountPath: /etc/config
- name: cacert
  mountPath: /etc/cacert

# volumes kısmı:
- name: secret-cinderplugin
  secret:
    secretName: cinder-cloud-config
- name: cacert
  secret:
    secretName: cinder-cloud-config
  items:
    - key: ca.crt
      path: ca.crt
```

Bölüm 2: Kurulum

Manifest dosyalarının bulunduğu dizinde `kubectl apply -f.`komutu çalıştırılır.

```
kubectl apply -f .
```

Bölüm 3: Doğrulama Kontrol Noktası

```
kubectl get pod -n kube-system | grep -i cinder

NAME                                READY   STATUS    RESTARTS   AGE
cinder-csi-controller-abcd1         3/3     Running   0          2m
cinder-csi-nodeplugin-efgh2         2/2     Running   0          2m
cinder-csi-nodeplugin-ijkl3         2/2     Running   0          2m
```

Bu, eklentinin başarıyla kurulduğunu ve OpenStack ile iletişim kurabildiğini doğrular.

Canlı Test - Dinamik Depolama Sağlama

TÜBİTAK

Bir uygulama pod'u için Cinder'dan otomatik olarak bir disk alanı talep edip kullanmak.

Adım 1: Node'ları etiketleme

Worker node'lar, Cinder'ın hangi AZ'den volume oluşturacağını bilmesi için etiketlenir:

```
kubectl label node <WORKER>  topology.cinder.csi.openstack.org/zone=nova
                                                --overwrite
```

Adım 2: StorageClass Oluşturma

Kullanıcıların depolama taleplerini karşılamak için Cinder CSI provisioner'ını kullanan bir `StorageClass'` tanımlanır.

Adım 3: PVC ve Pod ile Test

Sağ taraftaki YAML ile 1 GiB'lık bir `PersistentVolumeClaim (PVC) ve bu PVC'yi kullanan bir Nginx `Pod'u oluşturulur. Kubernetes, bu PVC talebini gördüğünde Cinder CSI aracılığıyla OpenStack'te otomatik olarak bir Cinder volume oluşturur ve pod'a bağlar.

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-cinder-sc
provisioner: cinder.csi.openstack.org
parameters:
  availability: nova
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: csi-cinder-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-cinder-sc
---
apiVersion: v1
kind: Pod
metadata:
  name: cinder-test-pod
spec:
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: csi-cinder-pvc
  containers:
  - name: nginx
    image: nginx
    volumeMounts:
    - mountPath: "/var/www/html"
      name: data
```

T.C. SANAYİ VE TEKNOLOJİ BAKANLIĞI

#MİLLİ TEKNOLOJİ HAMLESİ

TÜBİTAK

TEŞEKKÜRLER

TÜBİTAK | TÜRKİYE BİLİMSEL VE TEKNOLOJİK ARAŞTIRMA KURUMU

bulut@ulakbim.gov.tr