DGX Spark 4-Node AI Cluster Kurulum Rehberi

Bu doküman, 4 NVIDIA DGX Spark node'undan oluşan switch tabanlı bir AI cluster'ın kurulum ve konfigürasyon adımlarını baştan sona anlatmaktadır. Cluster, dağıtık AI iş yüklerini ve model çalıştırmayı yönetmek için **sparkrun** araç setini kullanır.

Doküman; management ve compute ağlarının hazırlanması, ConnectX-7/QSFP port yapılandırması, RoCEv2/RDMA ayarları, SSH erişimi, NCCL iletişim doğrulaması ve cluster sağlık kontrolü adımlarını kapsar.

1. Mimari Özet

Bileşen	Açıklama
DGX Spark × 4	Her birinde ConnectX-7 200GbE QSFP56 port bulunur
MikroTik CRS312	10GbE management ağı switch'i; NAS bonding portlarını da içerir
MikroTik CRS812	400G QSFP-DD compute ağı switch'i; breakout ile 4×200G sağlar
ASUSTOR AS6808T	8 diskli NAS; 2×10Gbps LACP ile CRS312'ye bağlı
sparkrun	Cluster yönetim, SSH mesh ve CX7 yapılandırma araç seti

2. Ön Koşullar

Donanım

- 4× NVIDIA DGX Spark sistemi
- 1× MikroTik CRS312-4C+8XS switch (management ağı)
- 1× MikroTik CRS812 switch (compute ağı)
- 1× ASUSTOR AS6808T NAS (8× HDD)
- 2× QSFP-DD → 2× QSFP56 pasif breakout kablo (compute ağı)
- Cat6 kablolar (management ağı ve NAS bağlantıları)

Yazılım ve İşletim Sistemi

- DGX OS (her Spark sisteminde kurulu)
- İnternet erişimi (paket indirmeleri ve güncellemeler için)

Bilgi ve Erişim

- MikroTik RouterOS web arayüzü ve CLI kullanım bilgisi
- Linux komut satırı temel bilgisi
- Tüm cihazların fiziksel erişimi (kablo bağlantıları için)

DGX Spark OS Kurulumu

DGX Spark OS kurulumu için aşağıdaki videoyu kullanabilirsiniz:

NVIDIA DGX Spark Kurulumu Part 1

3. DGX Spark Düğümlerin Hazırlığı

Fiziksel bağlantılar ve ağ yapılandırmalarına geçmeden önce tüm Spark sistemlerinin güncel yazılım ve firmware sürümlerini kullandığından emin olunmalıdır. Kurulum sırasında karşılaşılan performans problemlerinin önemli bir kısmı eski sürücüler, eksik güncellemeler veya firmware uyumsuzluklarından kaynaklanabilmektedir.

Aşağıdaki işlemler dört Spark sistemi üzerinde de uygulanmalıdır.

a. Sistem ve Firmware Güncellemeleri

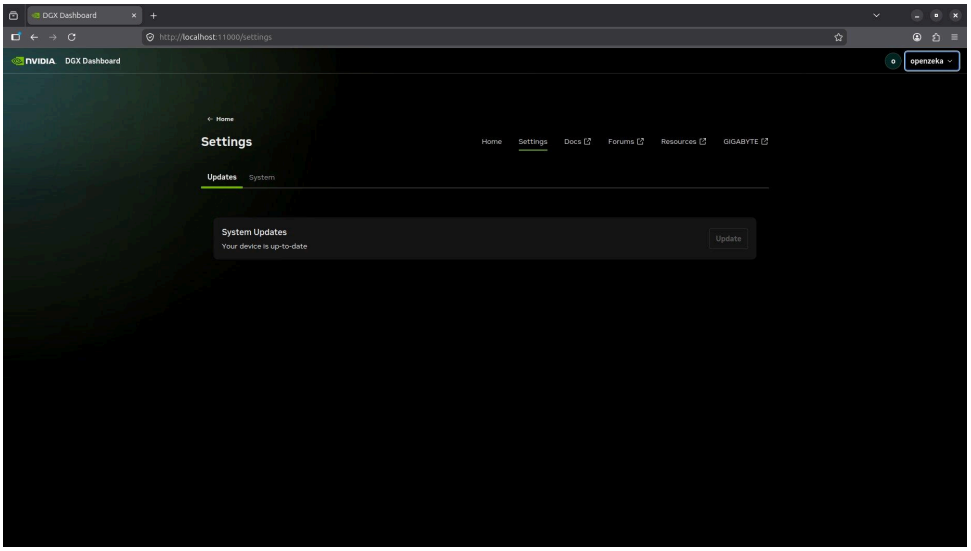
İlk olarak işletim sistemi paketleri güncellenir:

```shell
sudo apt update
sudo apt dist-upgrade
```

Daha sonra sistem firmware'leri güncellenir:

```shell
sudo fwupdmgr refresh --force
sudo fwupdmgr upgrade
```

DGX Dashboard üzerinden, herhangi bir güncelleme olmadığı kontrol edilir, eğer varsa yapılır:



Güncellemelerin tamamlanmasının ardından sistem yeniden başlatılır:

```
Shell
sudo reboot
```

Kurulum sırasında yapılan testlerde, güncel olmayan firmware sürümleri nedeniyle bağlantı performansının beklenen seviyeye ulaşmadığı gözlemlenmiştir. Bu nedenle kurulumun ilk adımı olarak tüm sistemlerin güncellenmesi önerilmektedir.

b. Docker Yapılandırması

Sonraki adımlarda kullanılacak container tabanlı araçların sudo gerektirmeden çalıştırılabilmesi için her bir Spark üzerinde Docker post-install işlemleri uygulanır.

Öncelikle mevcut kullanıcı Docker grubuna eklenir:

```
Shell
sudo groupadd docker
sudo usermod -aG docker $USER
newgrp docker
```

Yapılandırmanın başarılı olduğu aşağıdaki test ile doğrulanabilir:

```
Shell
docker run hello-world
```

```
openzeka@edgexpert-2e10:~$ sudo groupadd docker
[sudo] password for openzeka:
groupadd: group 'docker' already exists
openzeka@edgexpert-2e10:~$ sudo usermod -aG docker $USER
openzeka@edgexpert-2e10:~$ newgrp docker
openzeka@edgexpert-2e10:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
58dee6a49ef1: Pull complete
c3bdf82c34d1: Download complete
Digest: sha256:c3cbe1cc1aa588a64951ac6286e0df7b27fe2e6324b1001c619bb358770c0178
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (arm64v8)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

openzeka@edgexpert-2e10:~$
```

Komutun başarılı şekilde çalışması ve Docker'ın örnek container'ı başlatabilmesi, sonraki adımlarda kullanılacak container tabanlı araçlar için gerekli hazırlığın tamamlandığını göstermektedir.

Depolama Sürücüsünün Kontrolü

Ayrıca tüm Spark düğümlerinde Docker depolama sürücüsü kontrol edildi. `docker info` çıktısında Storage Driver değerinin `overlayfs` olduğu doğrulandı. `overlay2` veya farklı bir depolama sürücüsü tespit edilmesi durumunda Docker, containerd snapshotter (`overlayfs`) kullanacak şekilde yapılandırıldı ve Docker servisi yeniden başlatıldı. Böylece tüm düğümlerde aynı depolama altyapısı kullanılarak tutarlı bir çalışma ortamı sağlandı.

Öncelikle mevcut depolama sürücüsü aşağıdaki komut ile kontrol edildi:

Shell

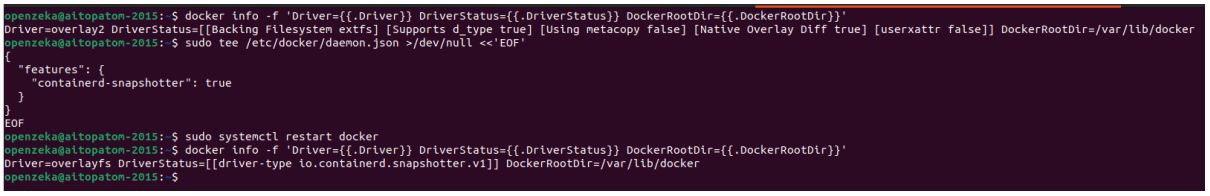
```
docker info -f 'Driver={{.Driver}} DriverStatus={{.DriverStatus}}
DockerRootDir={{.DockerRootDir}}'
```

Çıktıda sürücünün `overlay2` olarak görülmesi durumunda aşağıdaki yapılandırma uygulandı:

```shell
sudo tee /etc/docker/daemon.json >/dev/null <<'EOF'
{
  "features": {
    "containerd-snapshotter": true
  }
}
EOF

sudo systemctl restart docker
```

Yapılandırma sonrasında aynı kontrol komutu tekrar çalıştırıldı ve depolama sürücüsünün `overlayfs` olduğu doğrulandı.



4. Management Ağı (10GbE) Bağlantısı

Her bir DGX Spark'ın 10GbE Ethernet portu MikroTik CRS312 switch'in 10G portlarından birine Cat6 kablo ile bağlanır. Kablo takıldıktan sonra switch üzerindeki ilgili portun bağlantı göstergesinin yandığı teyit edilir.

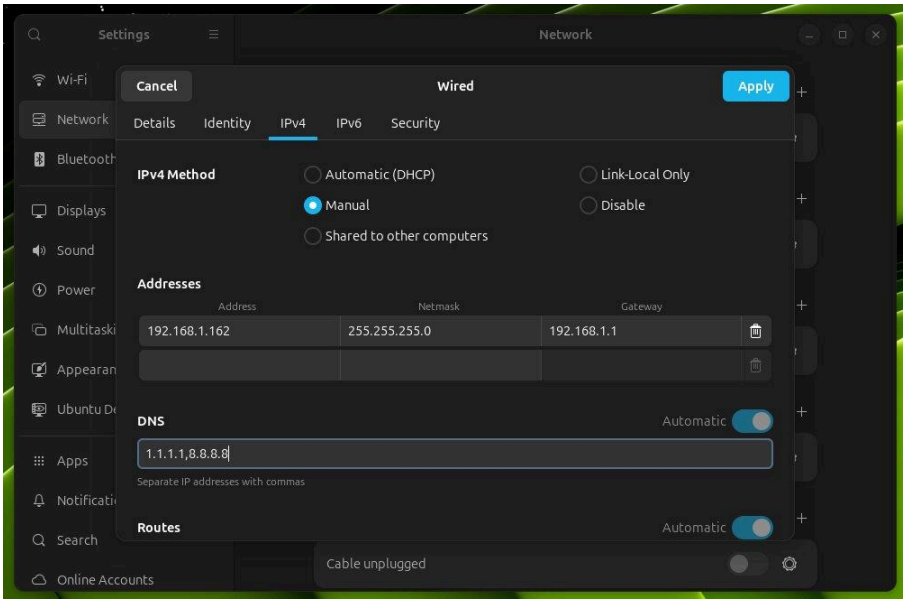
Spark masaüstünde terminal açılır ve cihazın IP adresi alıp almadığını kontrol edin:

```shell
ip addr show
```

```
openzeka@spark-77fb:~$ ip addr show
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: enP7s7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 4c:bb:47:82:77:fb brd ff:ff:ff:ff:ff:ff
    altname enP7p1s0
    inet 192.168.1.162/24 brd 192.168.1.255 scope global dynamic noprefixroute enP7s7
       valid_lft 599407sec preferred_lft 599407sec
    inet6 fe80::9249:4d6f:4b4e:7611/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: enp1s0f0np0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether 4c:bb:47:82:77:fc brd ff:ff:ff:ff:ff:ff
4: enp1s0f1np1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether 4c:bb:47:82:77:fd brd ff:ff:ff:ff:ff:ff
5: enP2p1s0f0np0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether 4c:bb:47:82:78:00 brd ff:ff:ff:ff:ff:ff
6: enP2p1s0f1np1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether 4c:bb:47:82:78:01 brd ff:ff:ff:ff:ff:ff
7: wlP9s9: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether f0:68:e3:31:a2:1e brd ff:ff:ff:ff:ff:ff
    altname wlP9p1s0
8: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 9a:82:cb:90:15:df brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
openzeka@spark-77fb:~$
```

Çıktıda, örnekte olduğu gibi 10GbE arayüzünde bir IP adresi görüyorsanız, management ağı üzerinden SSH erişimi sağlanabilir. IP adresi alınmamışsa, DGX OS masaüstü üzerinden manuel olarak atanır:

1. Sağ üst köşedeki ağ simgesine tıklayın → Wired Settings seçin
2. İlgili 10GbE bağlantısının yanındaki dişli (⚙) simgesine tıklayın
3. IPv4 sekmesine geçin
4. Method alanını Manual olarak değiştirin
5. Aşağıdaki bilgileri girin:
    - Address: 192.168.1.162 (her Spark için farklı ve kendi ağınıza uygun olarak—.163, .164, .165)
    - Netmask: 255.255.255.0
    - Gateway: 192.168.1.1 (varsa, yoksa boş bırakın)
    - DNS: 1.1.1.1,8.8.8.8
6. Apply butonuna basın ve bağlantıyı kapatıp tekrar açın



İnternet erişimi varsa 10GbE management bağlantısı hazırdır. Diğer üç Spark üzerinde de aynı adımları tekrarlayın ve her birine farklı bir IP adresi atayın.

Düğümler Arası Erişim Kontrolü

Management ağı üzerinden tüm Spark'ların birbirini görebildiğini teyit edin. Bir Spark üzerinden diğerlerine ping atın:

```shell
ping -c 4 192.168.1.157

ping -c 4 192.168.1.158

ping -c 4 192.168.1.161
```

Tüm ping'ler başarılıysa management ağı hazır ve tüm düğümler birbiriyle iletişim kurabiliyor demektir.

5. Compute Ağı (200GbE QSFP) Fiziksel Bağlantısı

DGX Spark Quad AI Cluster'da her Spark, ConnectX-7 QSFP

portu üzerinden 200GbE hızında MikroTik CRS812 switch'e bağlanır. CRS812'nin 400G QSFP-DD portları, pasif breakout kablolar ile ikişer 200G bağlantıya ayrılır.

Kablo Planı

CRS812 Port	Port Hızı	Breakout	Bağlanan Spark

QSFP-DD Port 1	400G	2 × 200G QSFP56	Spark 1 + Spark 2
QSFP-DD Port 2	400G	2 × 200G QSFP56	Spark 3 + Spark 4

Bağlantı Adımları

1. İlk breakout kablosunun QSFP-DD ucunu CRS812'nin 1 numaralı QSFP-DD portuna takın. Kablonun her iki ucundaki kilitleme mandalının tam oturduğundan emin olun.

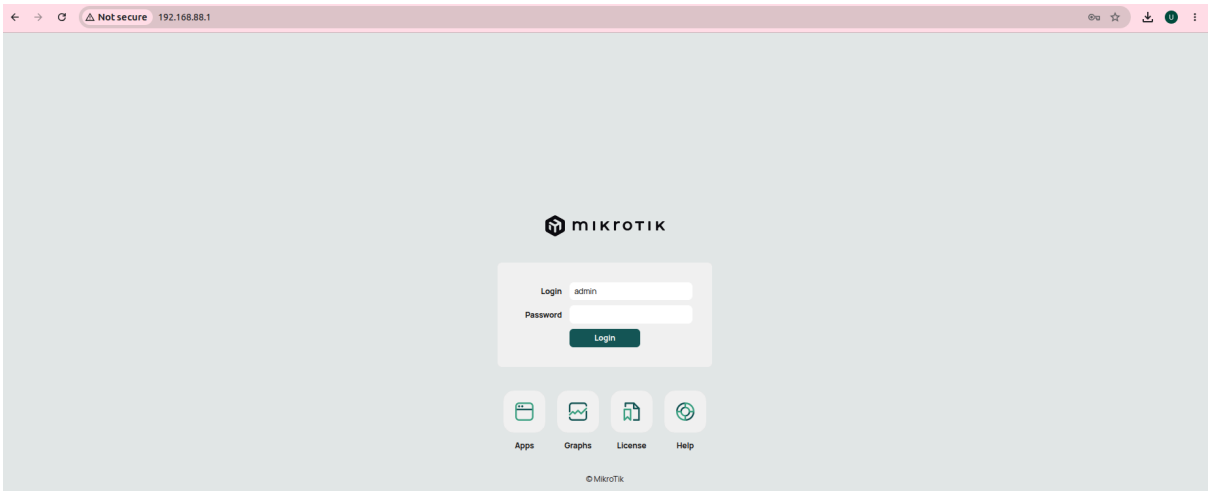
2. Aynı kablonun iki QSFP56 ucunu Spark 1 ve Spark 2 sistemlerinin en dışta bulunan ConnectX-7 portlarına takın.

3. İkinci breakout kablosunun QSFP-DD ucunu CRS812'nin 2 numaralı QSFP-DD portuna takın.
4. Bu kablonun iki QSFP56 ucunu Spark 3 ve Spark 4 sistemlerinin ConnectX-7 portlarına takın.
5. Tüm bağlantıların fiziksel olarak oturduğunu ve mandalların kilitlendiğini kontrol edin.
6. CRS812 switch'in gücünü açın.

6. CRS312 MikroTik Switch Yapılandırması

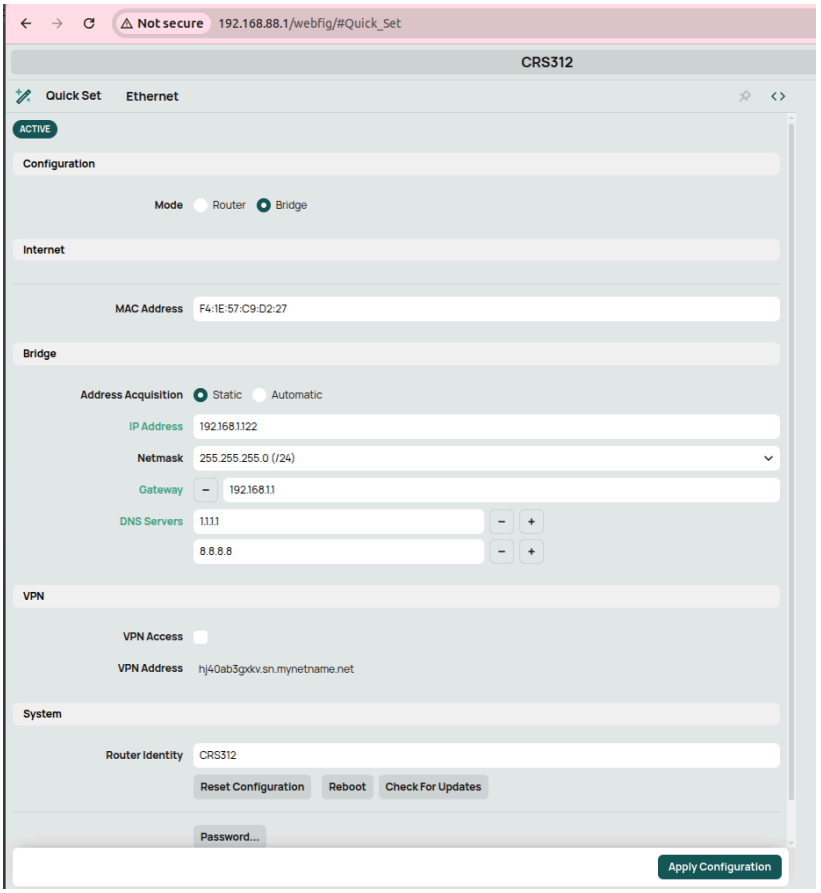
Switch açıldığında default IP adresi 192.168.88.1/24'tür. Bu adrese Ethernet-1 portundan erişebilirsiniz.

1. CRS312'nin Ethernet-1 portunu bilgisayarınıza bağlayın.
2. Bilgisayarınızın Ethernet arayüzüne 192.168.88.2/24 statik IP adresini atayın.
3. Tarayıcıdan http://192.168.88.1 adresine gidin
4. Kullanıcı adı admin, şifre ise cihazın altındaki etikette yazandır. Bu bilgilerle giriş yapın:



Yönetim IP Ataması

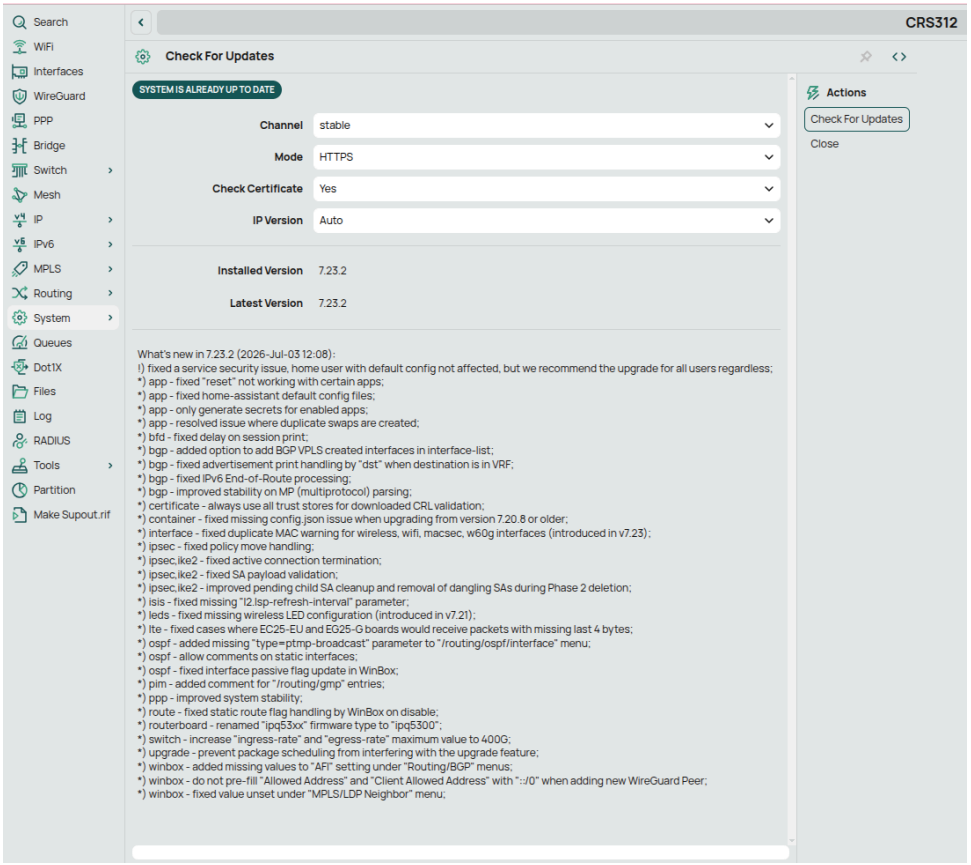
Giriş yaptıktan sonra şifrenizi değiştirmeniz istenen bir ekran gelecektir. Burada şifreyi değiştirdikten sonra açılan ekranda yönetim ile ilgili ip atamasını yapabilirsiniz. Burada örnek olarak 192.168.1.122/24 adresi verilmiştir, ağ ayarlarınıza göre uygun gateway ve DNS Sunucu adreslerini de girin ve “Apply Configuration” butonuna basın:



Bu ayarın uygulanmasıyla bağlantınız kopacaktır. Bu nedenle bağlantı için kullandığınız bilgisayarın ip adresini tekrar 192.168.1.0/24 ağında bir adrese alarak tarayıcıdan 192.168.1.122 adresini girerek switch arayüzüne ulaşabilirsiniz.

RouterOS Güncellemesi

Bu aşamada ilk olarak RouterOS yazılımının güncel olup olmadığı kontrol edilir. Bunu kontrol etmek için System - Packages - Check for Updates sayfasına gidilerek “Check for Updates” butonuna basılır ve güncelleme varsa yapılır:

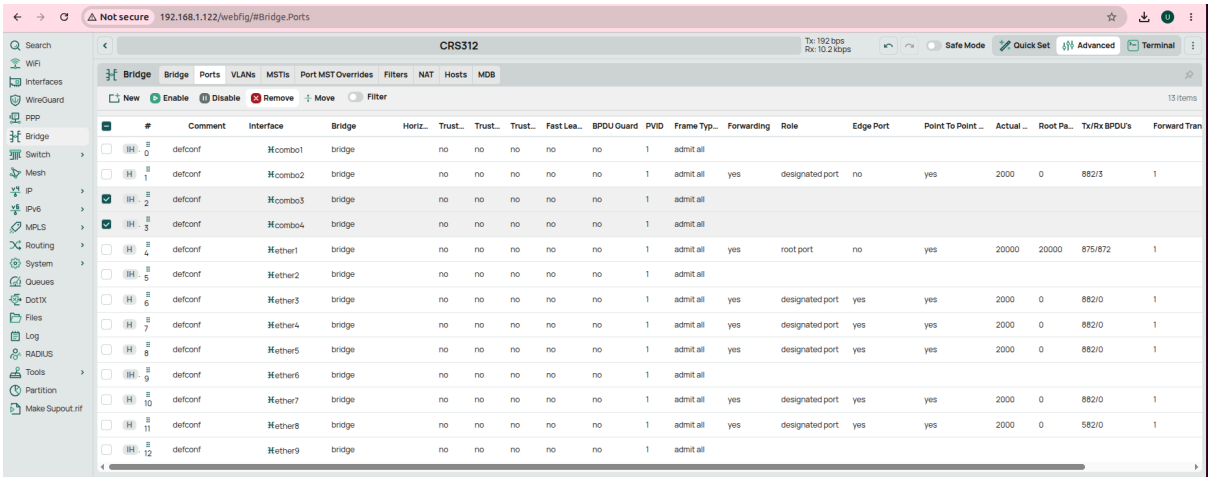


a. NAS Portlarının Bond Yapılması

Kullanılan NAS cihazının her iki 10Gbps portu bond yapılarak, switch tarafından bond yapılan Combo3 ve Combo4 portlarına bağlanarak toplamda 20Gbps boyutunda bir bant genişliği elde edilecektir.

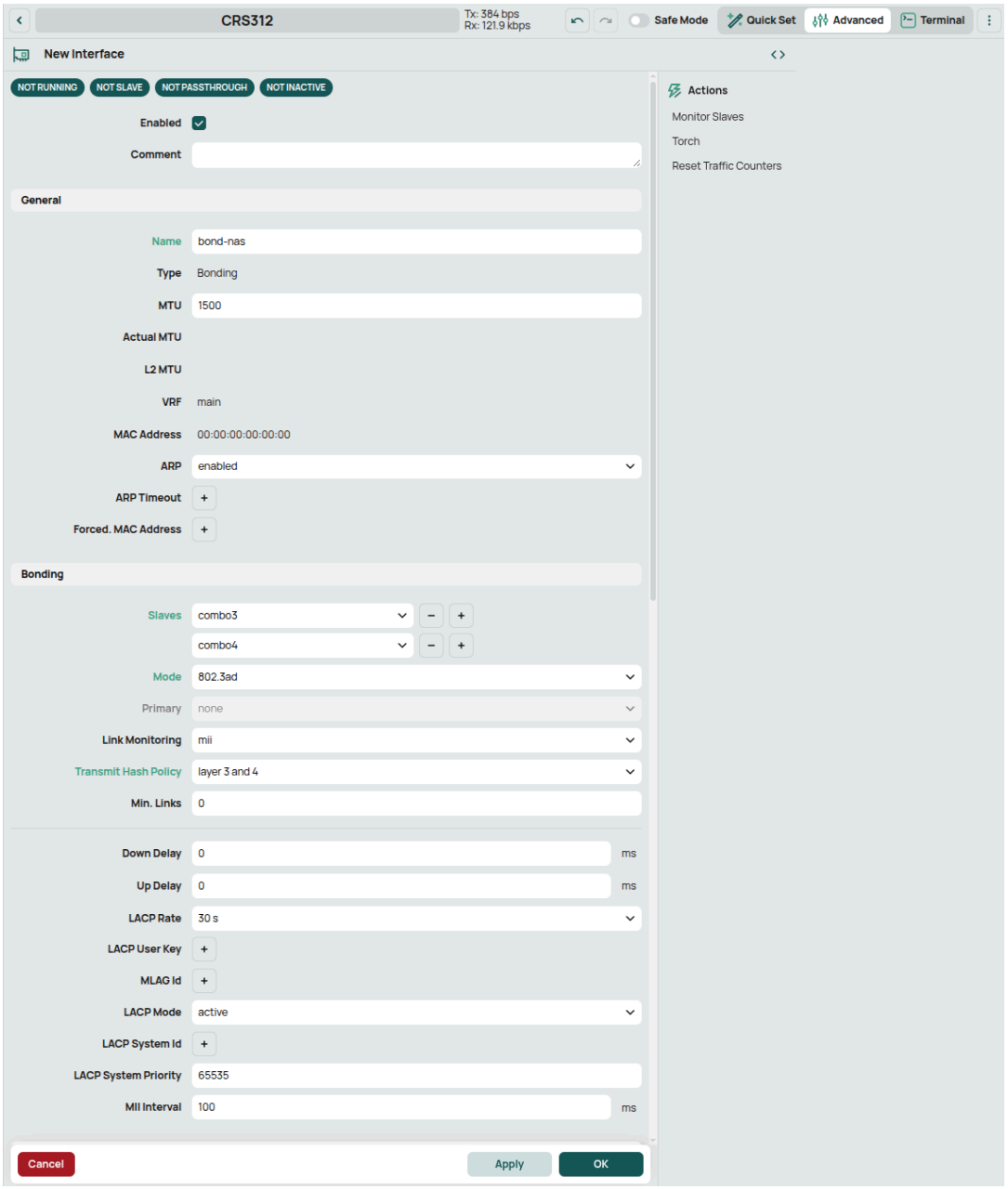
Combo3 ve Combo4 Portlarını Bridge'den Kaldırma

1. Bridge → Ports sekmesine gidin
2. combo3 ve combo4 portlarını bulun
3. Her birini seçip Remove (−) butonuna tıklayın



Bonding Oluşturma

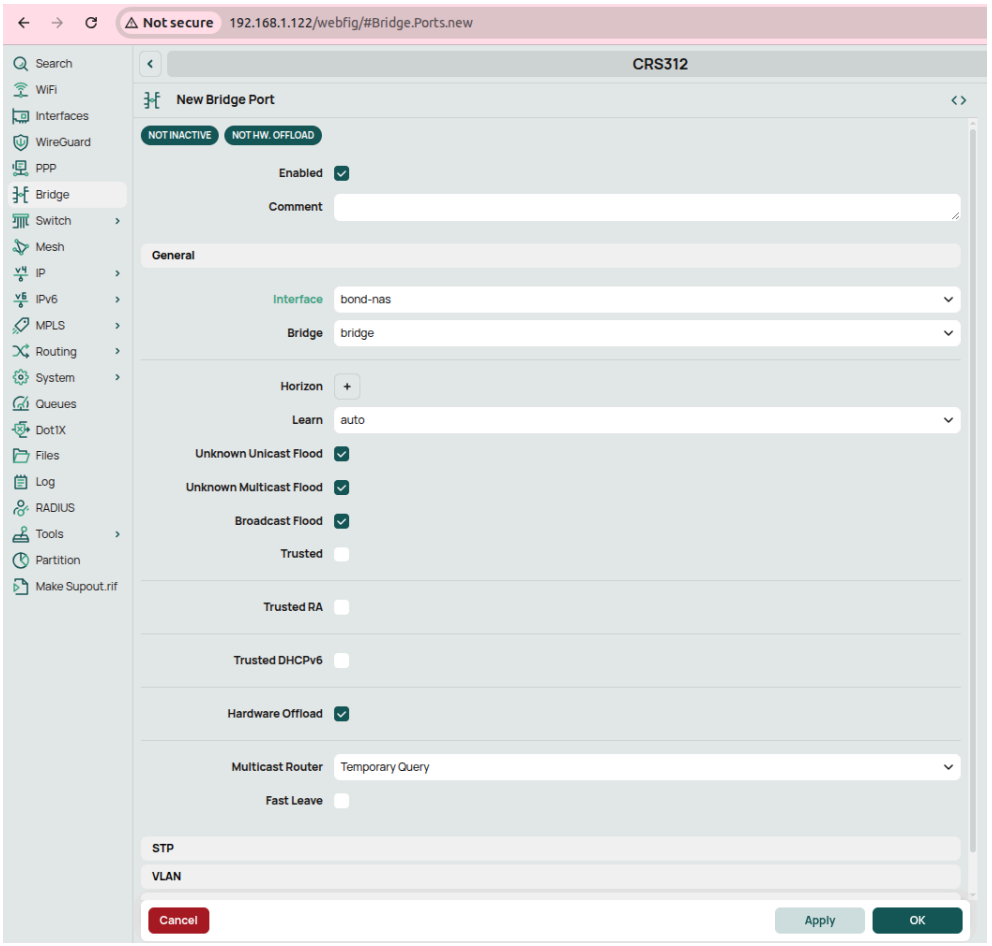
1. Switch arayüzünde Interfaces → Bonding sekmesine gidilir
2. “new” butonuna basılır
3. Aşağıdaki değerleri girin:
    a. Name: bond-nas
    b. Slaves: combo3, combo4
    c. Mode: 802.3ad
    d. Transmit Hash Policy: layer-3-and-4
4. Apply ve OK butonlarına basılır



Bond'u Bridge'e Ekleme

Son olarak oluşturulan bond, bridge üzerine eklenir:

1. Bridge → Ports sekmesine gidin
2. “new” butonuna basılır
3. Interface olarak “bond-nas” seçilir
4. Apply ve OK butonlarına basılır

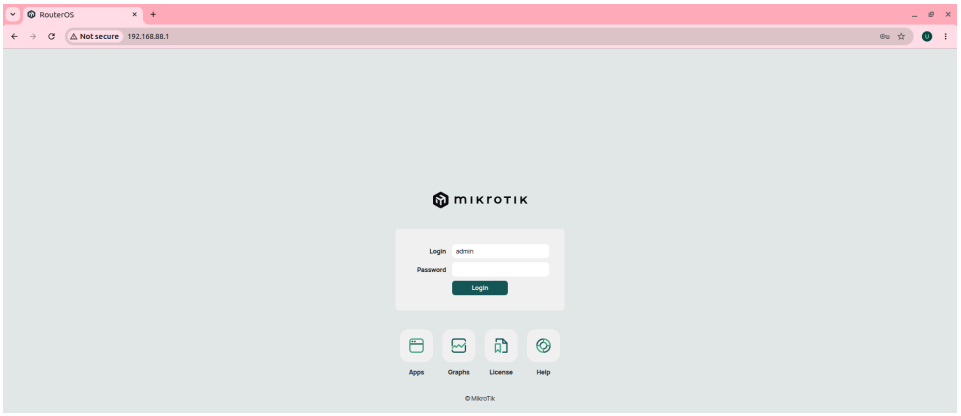


7. CRS812 MikroTik Switch Yapılandırması

a. Switch Hazırlığı:

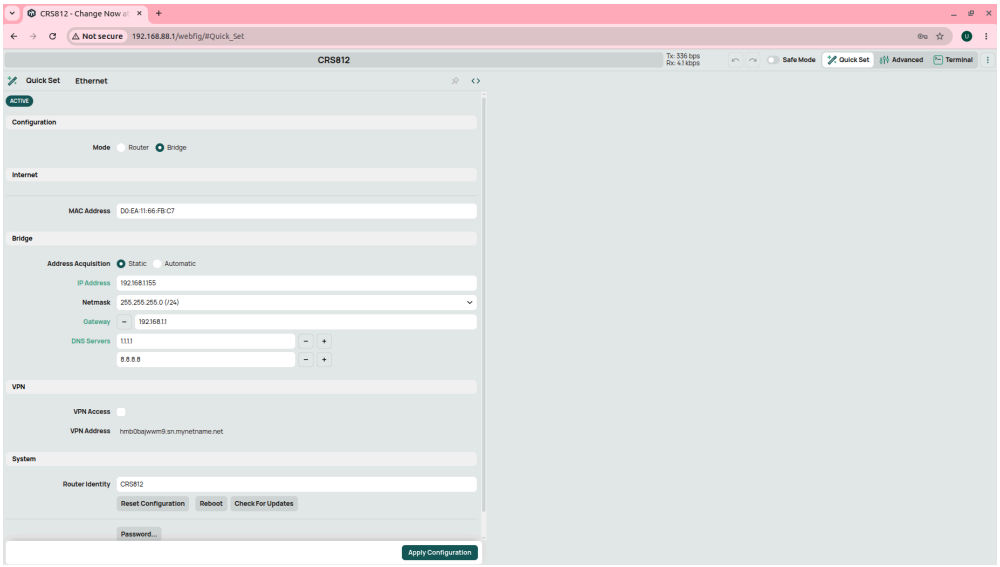
Switch açıldığında default IP adresi 192.168.88.1/24'tür. Bu adrese MGMT-1 portundan erişebilirsiniz:

1. CRS812'nin MGMT-1 portunu bilgisayarınıza bağlayın.
2. Bilgisayarınızın Ethernet arayüzüne 192.168.88.2/24 statik IP adresini atayın.
3. Tarayıcıdan http://192.168.88.1 adresine gidin
4. Kullanıcı adı admin, şifre ise cihazın altındaki etikette yazandır. Bu bilgilerle giriş yapın:



Yönetim IP Ataması

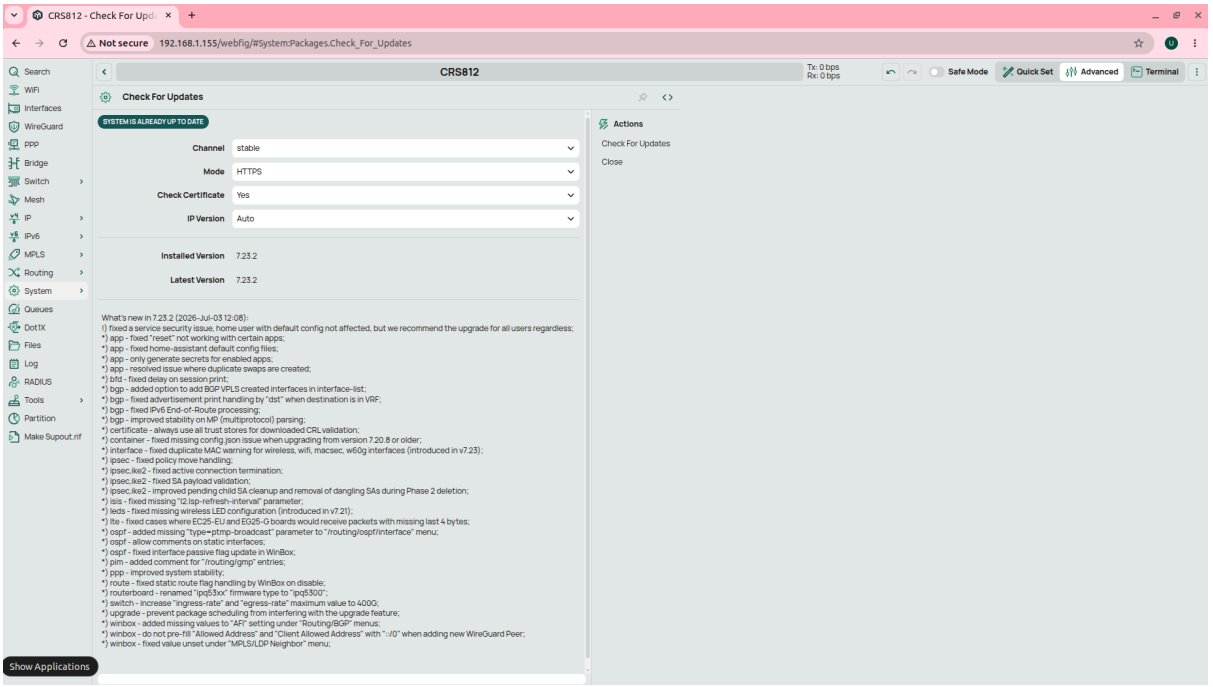
Giriş yaptıktan sonra şifrenizi değiştirmeniz istenen bir ekran gelecektir. Burada şifreyi değiştirdikten sonra açılan ekranda yönetim ile ilgili ip atamasını yapabilirsiniz. Burada örnek olarak 192.168.1.155/24 adresi verilmiştir, ağ ayarlarınıza göre uygun gateway ve DNS Sunucu adreslerini de girin ve “Apply Configuration” butonuna basın:



Bu ayarın uygulanmasıyla bağlantınız kopacaktır. Bu nedenle bağlantı için kullandığınız bilgisayarın ip adresini tekrar 192.168.1.0/24 ağında bir adrese alarak tarayıcıdan 192.168.1.155 adresini girerek switch arayüzüne ulaşabilirsiniz.

RouterOS Güncellemesi

Bu aşamada ilk yapılacak işlem RouterOS yazılımının en güncel versiyonda olduğudur. Bunu kontrol etmek için System - Packages - Check for Updates sayfasına gidilerek “Check for Updates” butonuna basılır ve güncelleme varsa yapılır:



b. Kurulum Öncesi Envanter ve Yedek

Switch arayüzüne management IP üzerinden SSH ile bağlanın. Tüm yapılandırma adımları terminal üzerinden yapılacaktır.

```shell
ssh admin@192.168.1.155
```

RoCEv2 yapılandırmasına başlamadan önce mevcut config'i yedekleyin:

```shell
/export file=before-roce
/system/backup/save name=before-roce
/file/print
```

```
[admin@CRS812] > export file=before-roce
[admin@CRS812] > system/backup/save name=before-roce
Saving system configuration
Configuration backup saved
[admin@CRS812] > file/print
Columns: NAME, TYPE, SIZE, LAST-MODIFIED
# NAME                      TYPE       SIZE      LAST-MODIFIED
0 before-roce.rsc           script     3853      2026-07-20 15:41:43
1 auto-before-reset.backup  backup     74.5KiB   1970-01-01 03:00:03
2 skins                     directory              1970-01-01 03:00:02
3 crs812-pfc-200g.rif       rif        558.3KiB  2026-06-25 09:02:14
4 console-dump.txt          .txt file  65.7KiB   2026-06-26 15:44:26
5 before-roce.backup        backup     96.6KiB   2026-07-20 15:42:00
```

c. QSFP-DD Portlarını 2×200G Breakout Yapılandırma

CRS812'nin iki QSFP-DD fiziksel portu her biri varsayılan olarak 400G'dir. Her 400G port, pasif breakout kablo ile iki adet 200G bağlantıya ayrılır. QSFP-DD portları 8 alt arayüze sahiptir; 2×200G modunda 200G ana arayüzler -1 ve -5'tir. Diğer alt arayüzler (-2, -3, -4, -6, -7, -8) kapatılmaz, etkin kalır ancak yapılandırma gerektirmez.

Breakout kabloları takıldıktan sonra, her 200G ana arayüzü zorlanmış hız ve kapalı auto-negotiation ile yapılandırın:

```shell
/interface/ethernet
set qsfp56-dd-1-1 auto-negotiation=no speed=200G-baseCR4
set qsfp56-dd-1-5 auto-negotiation=no speed=200G-baseCR4
set qsfp56-dd-2-1 auto-negotiation=no speed=200G-baseCR4
set qsfp56-dd-2-5 auto-negotiation=no speed=200G-baseCR4
```

```
[admin@CRS812] /interface/ethernet> set qsfp56-dd-1-1 auto-negotiation=no speed=200G-baseCR4
[admin@CRS812] /interface/ethernet> set qsfp56-dd-1-5 auto-negotiation=no speed=200G-baseCR4
[admin@CRS812] /interface/ethernet> set qsfp56-dd-2-1 auto-negotiation=no speed=200G-baseCR4
[admin@CRS812] /interface/ethernet> set qsfp56-dd-2-5 auto-negotiation=no speed=200G-baseCR4
```

d. Jumbo Frame ve MTU Yapılandırması

Spark'lara bağlı QSFP-DD portlarında hem L2MTU hem de MTU değerlerini ayarlayın:

```shell
/interface/ethernet
set qsfp56-dd-1-1 l2mtu=9500 mtu=9000
set qsfp56-dd-1-5 l2mtu=9500 mtu=9000
```

```
set qsfp56-dd-2-1 l2mtu=9500 mtu=9000
set qsfp56-dd-2-5 l2mtu=9500 mtu=9000
```

```
[admin@CRS812] /interface/ethernet> set qsfp56-dd-1-1 l2mtu=9500 mtu=9000
[admin@CRS812] /interface/ethernet> set qsfp56-dd-1-5 l2mtu=9500 mtu=9000
[admin@CRS812] /interface/ethernet> set qsfp56-dd-2-1 l2mtu=9500 mtu=9000
[admin@CRS812] /interface/ethernet> set qsfp56-dd-2-5 l2mtu=9500 mtu=9000
[admin@CRS812] /interface/ethernet>
```

e. RoCEv2 Trafik Sınıflandırması

QoS Profilleri

Shell

```
/interface/ethernet/switch/qos/profile
add name=roce dscp=26 traffic-class=3
add name=cnp dscp=48 traffic-class=6
```

```
[admin@CRS812] /interface/ethernet> switch/qos/profile/
[admin@CRS812] /interface/ethernet/switch/qos/profile> add name=roce dscp=26 traffic-class=3
[admin@CRS812] /interface/ethernet/switch/qos/profile> add name=cmp dscp=48 traffic-class=6
```

Tx Queue, ETS, ECN ve CNP Önceliği

Shell

```
/interface/ethernet/switch/qos/tx-manager/queue
set 1 schedule=high-priority-group weight=1
set 3 schedule=high-priority-group weight=1 ecn=yes
set 6 schedule=strict-priority
```

TC1 ve TC3 ETS grubunda eşit ağırlıkla (1:1) çalışır; TC3 üzerinde ECN işaretlemesi açıktır; CNP kontrol paketleri TC6 strict priority ile öncelik alır. TC1 boşsa TC3 portun tamamını kullanabilir — bu komut kalıcı 100G/100G rate-limit oluşturmaz.

```
[admin@CRS812] /interface/ethernet/switch/qos> tx-manager/queue/
[admin@CRS812] /interface/ethernet/switch/qos/tx-manager/queue> set 1 schedule=high-priority-group weight=1
[admin@CRS812] /interface/ethernet/switch/qos/tx-manager/queue> set 3 schedule=high-priority-group weight=1 ecn=yes
[admin@CRS812] /interface/ethernet/switch/qos/tx-manager/queue> set 6 schedule=strict-priority
```

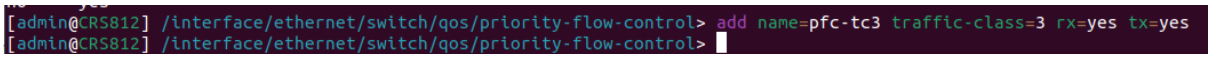
PFC Profili

Shell

```
/interface/ethernet/switch/qos/priority-flow-control
```

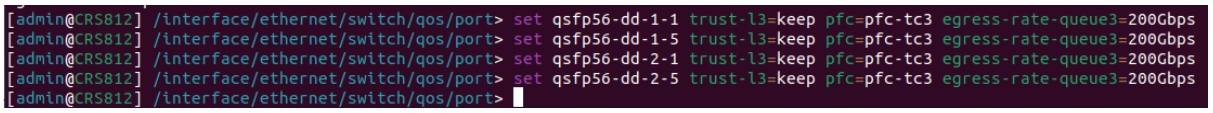
```
add name=pfc-tc3 traffic-class=3 rx=yes tx=yes
```

TC3 için iki yönlü Priority-based Flow Control profili oluşturur. tx=yes switch'in komşuya TC3 için XOFF/XON frame göndermesine; rx=yes komşudan gelen TC3 PFC frame'lerine uymasına izin verir.



Spark Portlarına Trust, PFC ve Queue Hız Referansı

```shell
/interface/ethernet/switch/qos/port
set qsfp56-dd-1-1 trust-l3=keep pfc=pfc-tc3
egress-rate-queue3=200Gbps
set qsfp56-dd-1-5 trust-l3=keep pfc=pfc-tc3
egress-rate-queue3=200Gbps
set qsfp56-dd-2-1 trust-l3=keep pfc=pfc-tc3
egress-rate-queue3=200Gbps
set qsfp56-dd-2-5 trust-l3=keep pfc=pfc-tc3
egress-rate-queue3=200Gbps
```



Lossless traffic class ve buffer havuzu

```shell
/interface/ethernet/switch/qos/settings
set lossless-traffic-class=3 lossless-buffers=auto
```

QoS hardware offload

```shell
/interface/ethernet/switch
set switch1 qos-hw-offloading=yes
```

```
[admin@CRS812] /interface/ethernet/switch> set switch1 qos-hw-offloading=yes
[admin@CRS812] /interface/ethernet/switch>
```

LLDP DCBX ilanı

```shell
/ip/neighbor/discovery-settings
set lldp-dcbx=yes
```

```
[admin@CRS812] /interface/ethernet/switch> /ip/neighbor/discovery-settings/
[admin@CRS812] /ip/neighbor/discovery-settings> set lldp-dcbx=yes
[admin@CRS812] /ip/neighbor/discovery-settings>
```

8. Spark'lara Sparkrun Yüklenmesi

a. Kullanıcı ve SSH Yapılandırması

Ağ yapılandırması tamamlandıktan sonra Spark sistemlerinin birbirleriyle şifresiz olarak haberleşebilmesi için tüm node'larda ortak bir kullanıcı oluşturulmalıdır. sparkrun, bu kullanıcı üzerinden tüm node'lara SSH ile bağlanır ve cluster yönetim işlemlerini gerçekleştirir.

Hostname Ayarlama
Her Spark'a benzersiz bir hostname verin. Bu, SSH known_hosts yönetimi, log analizi ve cluster node takibi için gereklidir:

```shell
# Spark 1 üzerinde:
sudo hostnamectl set-hostname spark1

# Spark 2 üzerinde:
sudo hostnamectl set-hostname spark2

# Spark 3 üzerinde:
sudo hostnamectl set-hostname spark3

# Spark 4 üzerinde:
sudo hostnamectl set-hostname spark4
```

Ortak Kullanıcı Oluşturma
Tüm Spark sistemlerinde aynı kullanıcı adı oluşturulmalıdır. Bu dokümanda nvidia kullanıcı adı kullanılacaktır. Aşağıdaki komutlar dört Spark sistemi üzerinde de çalıştırılır:

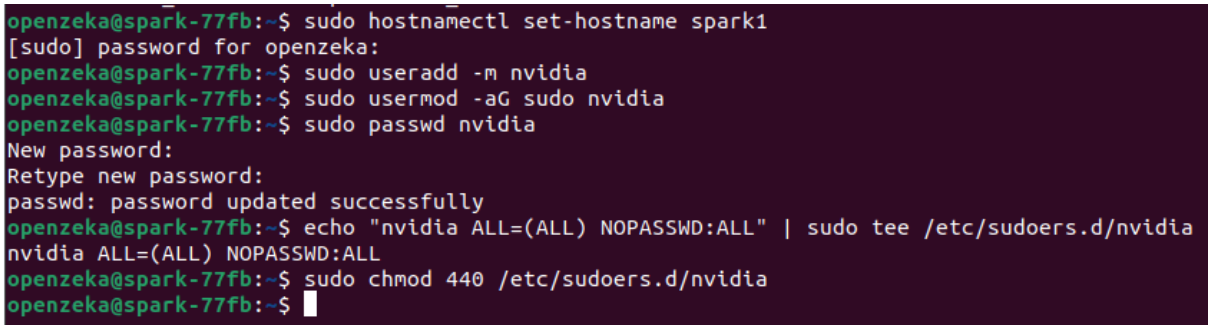
```shell
sudo useradd -m nvidia
sudo usermod -aG sudo nvidia
sudo passwd nvidia
```

Tüm sistemlerde aynı şifreyi kullanın — yönetim süreçlerini kolaylaştırır. sparkrun SSH mesh kurulumu sırasında ilk bağlantıda bu şifre sorulur, sonrasında anahtar tabanlı kimlik doğrulamaya geçilir.

Passwordless Sudo Yapılandırması

sparkrun, CX7 ağ yapılandırması sırasında sudo ile komutlar çalıştırır. Her seferinde şifre sormaması için passwordless sudo ayarlanmalıdır:

```shell
echo "nvidia ALL=(ALL) NOPASSWD:ALL" | sudo tee
/etc/sudoers.d/nvidia
sudo chmod 440 /etc/sudoers.d/nvidia
```



b. sparkrun Kurulumu

Kurulum nvidia kullanıcısı olan hesapta yapılır.

```shell
su - nvidia
```

Önce uv paketi kurulur:

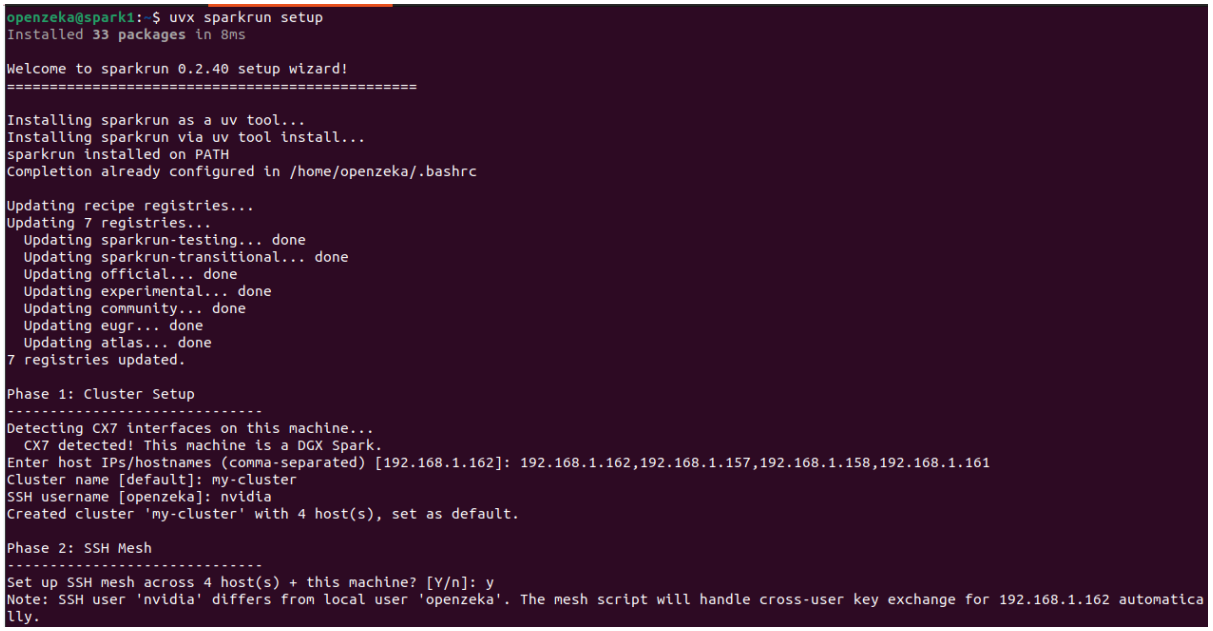
```shell
curl -LsSf https://astral.sh/uv/install.sh | sh
source ~/.bashrc
```

Sonra Sparkrun kurulur:

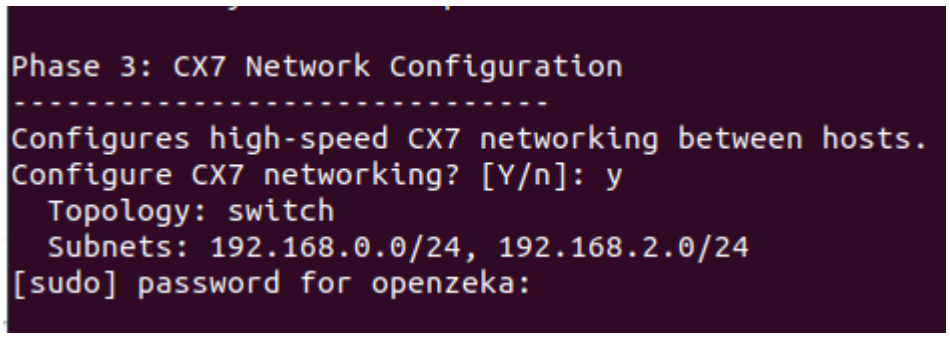
```
Shell
uvx sparkrun setup
```

Kurulum esnasında sorulan sorulara uygun cevaplar verilir:

1. öncelikle cihazların ip adresleri girilir:
2. cluster için bir isim verilir
3. SSH kullanıcı adı olarak önceki adımda oluşturulan `nvidia` ismi girilir.
4. MESH kurulumu için Y seçilir



5. Configure CX7 networking? sorusuna Y denir, burada sorduğu nvidia değil orijinal kullanıcı şifresidir:



6. Add 'nvidia' to the docker group on all hosts? sorusuna Y seçilir
7. Install sudoers entries? sorusuna “Y” seçilir
8. Install earlyoom? sorusuna “Y” seçilir
9. Setup complete mesajı geldiğinde kurulum başarıyla tamamlanmış demektir

```
Setup Complete!
================================================

  Cluster:    my-cluster (4 hosts, set as default)
  SSH mesh:   OK
  CX7:        configured (switch)
  SSH remesh: OK
  Docker:     OK (4/4)
  Sudoers:    installed (fix-permissions, clear-cache)
  earlyoom:   installed

Next steps:
  sparkrun list                          # Browse available recipes
  sparkrun show qwen3-1.7b-vllm          # Recipe details + VRAM estimate
  sparkrun run qwen3-1.7b-vllm --dry-run  # Preview a launch
  sparkrun run qwen3-1.7b-vllm            # Launch inference

openzeka@spark1:~$
```

c. DCB (Data Center Bridging) Konfigürasyonu

Switch tarafında PFC ve ECN yapılandırıldı, ancak Spark tarafında da ConnectX-7 arayüzleri için DSCP etiketleme ve PFC etkinleştirilmelidir. Bu adım sparkrun tarafından yapılmaz — her Spark'ta manuel uygulanmalıdır.

Aktif CX7 Arayüzlerini Tespit Etme
Her Spark'ta aktif CX7 arayüzlerini tespit edin:

```shell
ip link show | grep -E 'enp.*np|enP.*np'
```

MTU 9000 ve UP,LOWER_UP durumundaki arayüzler aktif olanlardır. Bu dokümanda her Spark'ta iki aktif arayüz kullanılmaktadır:

- enp1s0f1np1 — Subnet 1
- enP2p1s0f1np1 — Subnet 2

```
openzeka@spark1:~$ ip link show | grep -E 'enp.*np|enP.*np'
3: enp1s0f0np0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN mode DEFAULT group default qlen 1000
4: enp1s0f1np1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc mq state UP mode DEFAULT group default qlen 1000
5: enP2p1s0f0np0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN mode DEFAULT group default qlen 1000
6: enP2p1s0f1np1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc mq state UP mode DEFAULT group default qlen 1000
openzeka@spark1:~$
```

DSCP → Traffic Class Eşlemesi
ConnectX-7 arayüzlerine gelen RoCEv2 paketlerinin DSCP 26 değerini traffic class 3'e eşlemesi için dcb app komutu kullanılır:

```shell
sudo dcb app add dev enp1s0f1np1 dscp-prio 26:3
```

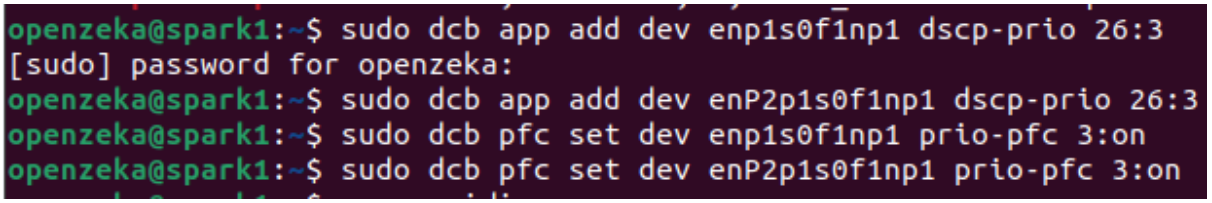
```
sudo dcb app add dev enP2p1s0f1np1 dscp-prio 26:3
```

Bu komut, DSCP 26 ile işaretlenmiş paketlerin priority 3 (TC3) kuyruğuna yönlendirilmesini sağlar. Switch tarafındaki TC3 yapılandırmasıyla aynı traffic class kullanılır.

PFC Etkinleştirme
TC3 için lossless iletişim sağlamak üzere PFC'yi etkinleştirin:

```
Shell
sudo dcb pfc set dev enp1s0f1np1 prio-pfc 3:on
sudo dcb pfc set dev enP2p1s0f1np1 prio-pfc 3:on
```

prio-pfc 3:on — yalnızca priority 3 için PFC frame'leri gönder ve alın. Diğer priority'ler etkilenmez.



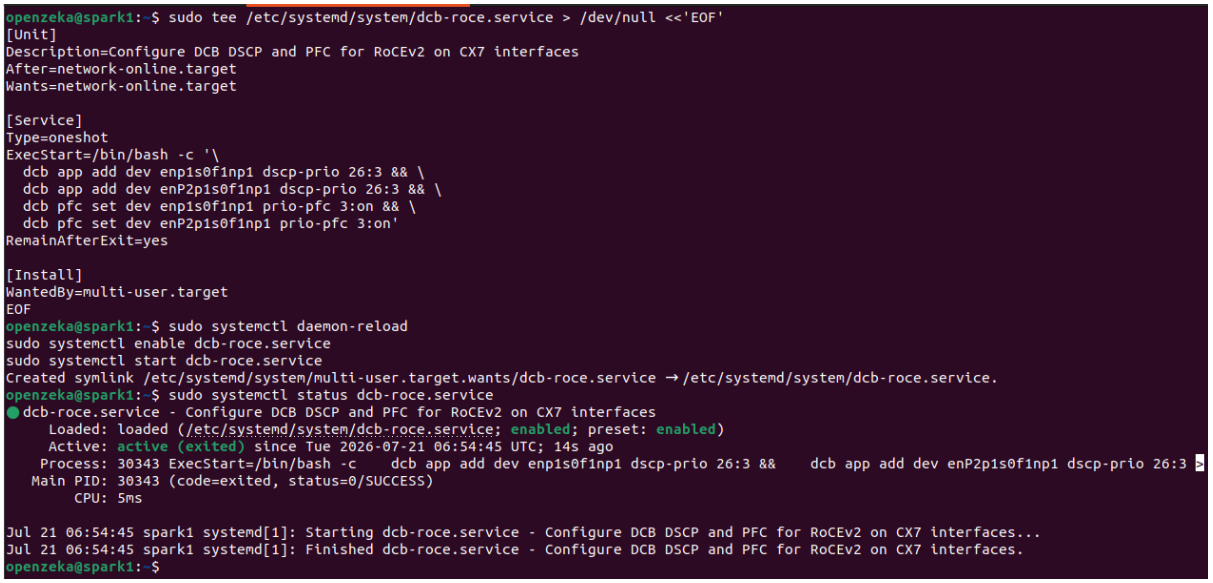
Kalıcılaştırma (systemd Servisi)
DCB komutları reboot sonrası kaybolur. Otomatik uygulanması için systemd servis dosyası oluşturun. Her Spark'ta:

```
Shell
sudo tee /etc/systemd/system/dcb-roce.service > /dev/null <<'EOF'
[Unit]
Description=Configure DCB DSCP and PFC for RoCEv2 on CX7
interfaces
After=network-online.target
Wants=network-online.target

[Service]
Type=oneshot
ExecStart=/bin/bash -c '\
  dcb app add dev enp1s0f1np1 dscp-prio 26:3 && \
  dcb app add dev enP2p1s0f1np1 dscp-prio 26:3 && \
  dcb pfc set dev enp1s0f1np1 prio-pfc 3:on && \
  dcb pfc set dev enP2p1s0f1np1 prio-pfc 3:on'
RemainAfterExit=yes
```

```
[Install]
WantedBy=multi-user.target
EOF

sudo systemctl daemon-reload
sudo systemctl enable dcb-roce.service
sudo systemctl start dcb-roce.service
```



9. Hız ve RDMA Testleri

Bu adımda, compute ağının doğru çalıştığını ve RoCEv2 üzerinden RDMA iletişiminin beklendiği gibi performans verdiğini doğrulayacağız. Testler iki Spark arasında yapılır; 4 node için çapraz testler benzer şekilde tekrarlanabilir.

IP Ataması Referansı
sparkrun wizard tarafından CX7 arayüzlerine atanan IP adresleri aşağıda göründüğü gibidir, sizin senaryoda bu adresler yerine kendi adreslerinizi kullanmanız gerekir:

Spark	Management (enP7s7)	CX7 Subnet 1 (enp1s0f1np1)	CX7 Subnet 2 (enP2p1s0f1np1)
Spark 1	192.168.1.162	192.168.0.162	192.168.2.162
Spark 2	192.168.1.157	192.168.0.157	192.168.2.157
Spark 3	192.168.1.158	192.168.0.158	192.168.2.158

Spark 4	192.168.1.161	192.168.0.161	192.168.2.161

Testler için iki CX7 subnet'i kullanılır:

- Subnet 1: 192.168.0.0/24 → enp1s0f1np1
- Subnet 2: 192.168.2.0/24 → enP2p1s0f1np1

IP ve MTU Testi

Spark 1'den Spark 2'ye ping ile bağlantı ve jumbo frame testi yapın:

```shell
# Spark 1 üzerinde:
ping -c 4 192.168.0.157
ping -M do -s 8972 -c 4 192.168.0.157
```

İlk ping normal bağlantıyı, ikinci ping 9000 byte MTU'yu test eder. -M do fragmentation'ı engeller — paket düşmezse MTU 9000 uçtan uca çalışıyor demektir.

İkinci subnet için de tekrarlayın:

```shell
ping -c 4 192.168.2.157
ping -M do -s 8972 -c 4 192.168.2.157
```

```
openzeka@spark1:~$ ping -c 4 192.168.0.157
PING 192.168.0.157 (192.168.0.157) 56(84) bytes of data.
64 bytes from 192.168.0.157: icmp_seq=1 ttl=64 time=1.22 ms
64 bytes from 192.168.0.157: icmp_seq=2 ttl=64 time=0.962 ms
64 bytes from 192.168.0.157: icmp_seq=3 ttl=64 time=0.890 ms
64 bytes from 192.168.0.157: icmp_seq=4 ttl=64 time=0.938 ms

--- 192.168.0.157 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 0.890/1.002/1.219/0.127 ms
openzeka@spark1:~$ ping -M do -s 8972 -c 4 192.168.0.157
PING 192.168.0.157 (192.168.0.157) 8972(9000) bytes of data.
8980 bytes from 192.168.0.157: icmp_seq=1 ttl=64 time=0.909 ms
8980 bytes from 192.168.0.157: icmp_seq=2 ttl=64 time=1.00 ms
8980 bytes from 192.168.0.157: icmp_seq=3 ttl=64 time=0.994 ms
8980 bytes from 192.168.0.157: icmp_seq=4 ttl=64 time=0.992 ms

--- 192.168.0.157 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 0.909/0.974/1.002/0.037 ms
openzeka@spark1:~$ ping -c 4 192.168.2.157
PING 192.168.2.157 (192.168.2.157) 56(84) bytes of data.
64 bytes from 192.168.2.157: icmp_seq=1 ttl=64 time=2.19 ms
64 bytes from 192.168.2.157: icmp_seq=2 ttl=64 time=0.625 ms
64 bytes from 192.168.2.157: icmp_seq=3 ttl=64 time=0.581 ms
64 bytes from 192.168.2.157: icmp_seq=4 ttl=64 time=1.17 ms

--- 192.168.2.157 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3045ms
rtt min/avg/max/mdev = 0.581/1.141/2.189/0.647 ms
openzeka@spark1:~$ ping -M do -s 8972 -c 4 192.168.2.157
PING 192.168.2.157 (192.168.2.157) 8972(9000) bytes of data.
8980 bytes from 192.168.2.157: icmp_seq=1 ttl=64 time=0.580 ms
8980 bytes from 192.168.2.157: icmp_seq=2 ttl=64 time=1.09 ms
8980 bytes from 192.168.2.157: icmp_seq=3 ttl=64 time=0.988 ms
8980 bytes from 192.168.2.157: icmp_seq=4 ttl=64 time=0.877 ms

--- 192.168.2.157 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3015ms
rtt min/avg/max/mdev = 0.580/0.883/1.088/0.190 ms
openzeka@spark1:~$
```

TCP Throughput Testi (iperf3)

Ethernet/IP katmanı üzerinden temel bant genişliğini ölçün. Bu test RDMA değildir — TCP üzerinden CPU involvement'lı transferdir.

Shell

```
# Spark 2 üzerinde (sunucu):
iperf3 -s
# Spark 1 üzerinde (istemci):
iperf3 -c 192.168.0.157 -P 8 -t 30
```

-P 8 sekiz paralel akış, -t 30 otuz saniye test süresi. Beklenen sonuç: ~100-120 Gbps toplam throughput.

Not: iperf3 kurulu değilse yükleyin:

```
Shell
sudo apt install iperf3
```

```
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-30.00  sec  63.9 GBytes  18.3 Gbits/sec    0             sender
[  5]   0.00-30.00  sec  63.9 GBytes  18.3 Gbits/sec                  receiver
[  7]   0.00-30.00  sec  33.7 GBytes  9.63 Gbits/sec    1             sender
[  7]   0.00-30.00  sec  33.7 GBytes  9.63 Gbits/sec                  receiver
[  9]   0.00-30.00  sec  64.1 GBytes  18.3 Gbits/sec    0             sender
[  9]   0.00-30.00  sec  64.1 GBytes  18.3 Gbits/sec                  receiver
[ 11]   0.00-30.00  sec  64.3 GBytes  18.4 Gbits/sec    0             sender
[ 11]   0.00-30.00  sec  64.3 GBytes  18.4 Gbits/sec                  receiver
[ 13]   0.00-30.00  sec  33.5 GBytes  9.59 Gbits/sec    0             sender
[ 13]   0.00-30.00  sec  33.5 GBytes  9.59 Gbits/sec                  receiver
[ 15]   0.00-30.00  sec  62.7 GBytes  18.0 Gbits/sec    0             sender
[ 15]   0.00-30.00  sec  62.7 GBytes  18.0 Gbits/sec                  receiver
[ 17]   0.00-30.00  sec  35.0 GBytes  10.0 Gbits/sec    0             sender
[ 17]   0.00-30.00  sec  35.0 GBytes  10.0 Gbits/sec                  receiver
[ 19]   0.00-30.00  sec  30.7 GBytes  8.79 Gbits/sec    0             sender
[ 19]   0.00-30.00  sec  30.7 GBytes  8.79 Gbits/sec                  receiver
[SUM]   0.00-30.00  sec   388 GBytes   111 Gbits/sec    1             sender
[SUM]   0.00-30.00  sec   388 GBytes   111 Gbits/sec                  receiver

iperf Done.
openzeka@spark1:~$
```

RDMA Cihazlarını Belirleme
RDMA cihaz adlarını öğrenin:

```
Shell
ibdev2netdev
```

Örnek çıktı:

rocep1s0f1 port 1 ==> enp1s0f1np1 (Up)
roceP2p1s0f1 port 1 ==> enP2p1s0f1np1 (Up)

```
openzeka@spark1:~$ ibdev2netdev
roceP2p1s0f0 port 1 ==> enP2p1s0f0np0 (Down)
roceP2p1s0f1 port 1 ==> enP2p1s0f1np1 (Up)
rocep1s0f0 port 1 ==> enp1s0f0np0 (Down)
rocep1s0f1 port 1 ==> enp1s0f1np1 (Up)
openzeka@spark1:~$
```

RDMA Write Testi (ib_write_bw)
RoCEv2 üzerinden RDMA write işleminin bant genişliğini ölçün. Bu test CPU involvement olmadan doğrudan bellek transferini test eder.

Subnet 1 (enp1s0f1np1 → rocep1s0f1):
Spark 2 üzerinde (sunucu):

```
Shell
ib_write_bw -d rocep1s0f1 -F --report_gbits
```

Spark 1 üzerinde (istemci):

```
Shell
ib_write_bw -d rocep1s0f1 -F --report_gbits 192.168.0.157
```

Beklenen sonuç: ~100-111 Gbps.

```
openzeka@spark1:~$ ib_write_bw -d rocep1s0f1 -F --report_gbits 192.168.0.157
---------------------------------------------------------------------------------------
                    RDMA_Write BW Test
 Dual-port       : OFF          Device         : rocep1s0f1
 Number of qps   : 1            Transport type : IB
 Connection type : RC           Using SRQ      : OFF
 PCIe relax order: ON
 ibv_wr* API     : ON
 TX depth        : 128
 CQ Moderation   : 1
 Mtu             : 4096[B]
 Link type       : Ethernet
 GID index       : 3
 Max inline data : 0[B]
 rdma_cm QPs     : OFF
 Data ex. method : Ethernet
---------------------------------------------------------------------------------------
 local address: LID 0000 QPN 0x0228 PSN 0x9bd50e RKey 0x1c03ec VAddr 0x00e83c995cd000
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:00:162
 remote address: LID 0000 QPN 0x0228 PSN 0x3fcb66 RKey 0x1c03ec VAddr 0x00f01cc1cf4000
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:00:157
---------------------------------------------------------------------------------------
 #bytes     #iterations    BW peak[Gb/sec]    BW average[Gb/sec]   MsgRate[Mpps]
 65536      5000             109.27             109.25             0.208383
---------------------------------------------------------------------------------------
```

Subnet 2 (enP2p1s0f1np1 → roceP2p1s0f1):
Spark 2 üzerinde (sunucu):

```
Shell
ib_write_bw -d roceP2p1s0f1 -F --report_gbits
```

Spark 1 üzerinde (istemci):

```
Shell
ib_write_bw -d roceP2p1s0f1 -F --report_gbits 192.168.2.157
```

Beklenen sonuç: ~100-111 Gbps.

```
openzeka@spark1:~$ ib_write_bw -d roceP2p1s0f1 -F --report_gbits 192.168.2.157
---------------------------------------------------------------------------------------
                    RDMA_Write BW Test
 Dual-port       : OFF          Device         : roceP2p1s0f1
 Number of qps   : 1            Transport type : IB
 Connection type : RC           Using SRQ      : OFF
 PCIe relax order: ON
 ibv_wr* API     : ON
 TX depth        : 128
 CQ Moderation   : 1
 Mtu             : 4096[B]
 Link type       : Ethernet
 GID index       : 3
 Max inline data : 0[B]
 rdma_cm QPs     : OFF
 Data ex. method : Ethernet
---------------------------------------------------------------------------------------
 local address: LID 0000 QPN 0x02a8 PSN 0xcb63e RKey 0x1e03ec VAddr 0x00fbee0f4ad000
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:02:162
 remote address: LID 0000 QPN 0x02a8 PSN 0x8b67bf RKey 0x1e03ec VAddr 0x00f3671aaed000
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:02:157
---------------------------------------------------------------------------------------
 #bytes     #iterations    BW peak[Gb/sec]    BW average[Gb/sec]   MsgRate[Mpps]
 65536      5000             109.14             109.13             0.208140
---------------------------------------------------------------------------------------
openzeka@spark1:~$
```

İki arayüz de ~100 Gbps veriyorsa, her Spark arasında toplam ~200 Gbps RDMA bant genişliği mevcuttur.

RDMA Read Testi (ib_read_bw)
RDMA read işleminin bant genişliğini ölçün:

Subnet 1 (enp1s0f1np1 → rocep1s0f1):
Spark 2 üzerinde (sunucu):

```shell
ib_read_bw -d rocep1s0f1 -F --report_gbits
```

Spark 1 üzerinde (istemci):

```shell
ib_read_bw -d rocep1s0f1 -F --report_gbits 192.168.0.157
```

Beklenen sonuç: ~95-110 Gbps.

```
openzeka@spark1:~$ ib_read_bw -d rocep1s0f1 -F --report_gbits 192.168.0.157
---------------------------------------------------------------------------------------
                    RDMA_Read BW Test
 Dual-port       : OFF          Device         : rocep1s0f1
 Number of qps   : 1            Transport type : IB
 Connection type : RC           Using SRQ      : OFF
 PCIe relax order: ON
 ibv_wr* API     : ON
 TX depth        : 128
 CQ Moderation   : 1
 Mtu             : 4096[B]
 Link type       : Ethernet
 GID index       : 3
 Outstand reads  : 16
 rdma_cm QPs     : OFF
 Data ex. method : Ethernet
---------------------------------------------------------------------------------------
 local address: LID 0000 QPN 0x022a PSN 0x231671 OUT 0x10 RKey 0x1c0300 VAddr 0x00fabc0871d000
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:00:162
 remote address: LID 0000 QPN 0x022a PSN 0x7ce117 OUT 0x10 RKey 0x1c03ee VAddr 0x00f6b4b38ed000
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:00:157
---------------------------------------------------------------------------------------
 #bytes     #iterations    BW peak[Gb/sec]    BW average[Gb/sec]   MsgRate[Mpps]
 65536      1000             109.16             109.14             0.208163
---------------------------------------------------------------------------------------
```

Subnet 2 (enP2p1s0f1np1 → roceP2p1s0f1):
Spark 2 üzerinde (sunucu):

Shell

```
ib_read_bw -d roceP2p1s0f1 -F --report_gbits
```

Spark 1 üzerinde (istemci):

Shell

```
ib_read_bw -d roceP2p1s0f1 -F --report_gbits 192.168.2.157
```

```
openzeka@spark1:~$ ib_read_bw -d roceP2p1s0f1 -F --report_gbits 192.168.2.157
---------------------------------------------------------------------------------------
                    RDMA_Read BW Test
 Dual-port       : OFF          Device         : roceP2p1s0f1
 Number of qps   : 1            Transport type : IB
 Connection type : RC           Using SRQ      : OFF
 PCIe relax order: ON
 ibv_wr* API     : ON
 TX depth        : 128
 CQ Moderation   : 1
 Mtu             : 4096[B]
 Link type       : Ethernet
 GID index       : 3
 Outstand reads  : 16
 rdma_cm QPs     : OFF
 Data ex. method : Ethernet
---------------------------------------------------------------------------------------
 local address: LID 0000 QPN 0x02a9 PSN 0xa923c1 OUT 0x10 RKey 0x1e0300 VAddr 0x00eb80a9abd000
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:02:162
 remote address: LID 0000 QPN 0x02a9 PSN 0x1695ab OUT 0x10 RKey 0x1e0300 VAddr 0x00e2141659d000
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:02:157
---------------------------------------------------------------------------------------
 #bytes     #iterations    BW peak[Gb/sec]    BW average[Gb/sec]   MsgRate[Mpps]
 65536      1000             109.16             109.14             0.208168
---------------------------------------------------------------------------------------
```

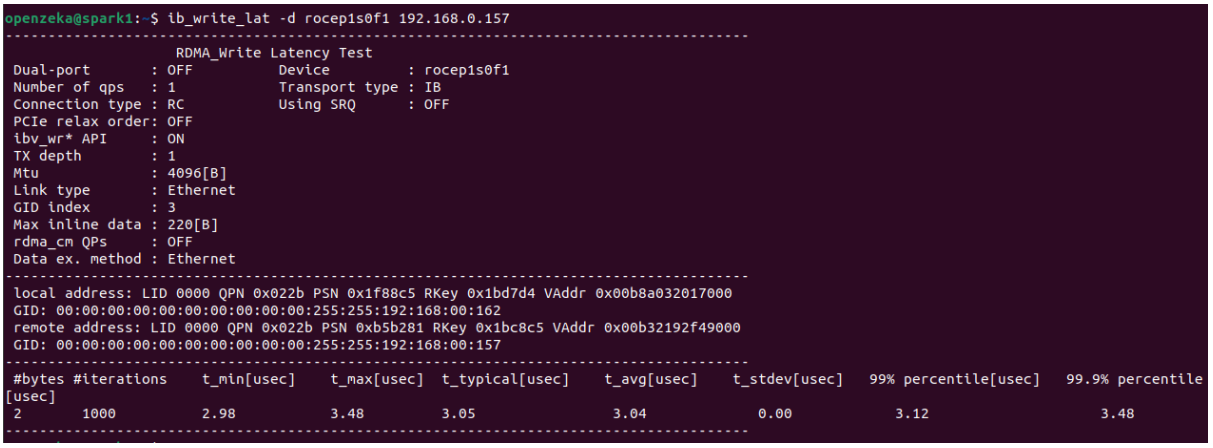
Beklenen sonuç: ~95-110 Gbps.

RDMA Latency Testi (ib_write_lat)
Spark 2 üzerinde (sunucu):

```
Shell
ib_write_lat -d rocep1s0f1
```

Spark 1 üzerinde (istemci):

```
Shell
ib_write_lat -d rocep1s0f1 192.168.0.157
```



Beklenen sonuç: ~1-3 microsecond latency.

10. sparkrun ile Model Çalıştırma

Bu adımda, sparkrun üzerinden çok node'lu bir inference iş yükü çalıştırarak cluster'ın uçtan uca çalıştığını doğrulayacağız.

Model ve Recipe
Bu testte glm-5.2-int4 modeli kullanılır. Model, 4 node üzerinde tensor parallelism ile çalıştırılır. Bu model için özel container imajı ve recipe YAML dosyası OpenZeka tarafından sağlanmaktadır. Bu modeli kullanmak istiyorsanız OpenZeka ile irtibata geçin. Ya da kendi imaj ve modelinizi kullanabilirsiniz.

Modelin Çalıştırılması
Örnek olarak çalıştırılacak model için komut ve yaml dosyasının içeriği aşağıdadır:

```
Shell
sparkrun run glm52-qt-dcp4-4spark.yaml
```

Çalıştırılan yaml dosyası içeriği şu şekildedir:

```shell
model: QuantTrio/GLM-5.2-Int4-Int8Mix
runtime: vllm-ray
container: vllm-zatz-dcp:probe

min_nodes: 4
max_nodes: 4

metadata:
  description: GLM-5.2 Int4-Int8Mix TP4 DCP4 655K MTP
(tonyd2wild, cudagraph NONE for GB10 host-staged NCCL)
  maintainer: local
  model_params: 744B-MoE-40B-active
  model_dtype: int4-int8mix

defaults:
  port: 8210
  host: 0.0.0.0
  tensor_parallel: 4
  pipeline_parallel: 1
  decode_context_parallel: 4
  gpu_memory_utilization: 0.80
  max_model_len: 32000
  max_num_seqs: 16
  max_num_batched_tokens: 4096
  kv_cache_dtype: fp8_ds_mla
  kv_cache_memory_bytes: 9000000000
  load_format: auto
  served_model_name: glm-5.2
  quantization: compressed-tensors
  reasoning_parser: glm45
  tool_call_parser: glm47

env:
  NCCL_IB_TC: "106"
  HF_HUB_OFFLINE: "1"
  TRANSFORMERS_OFFLINE: "1"
  SAFETENSORS_FAST_GPU: "1"
```

```
  CUDA_DEVICE_ORDER: "PCI_BUS_ID"
  CUDA_DEVICE_MAX_CONNECTIONS: "32"
  CUTE_DSL_ARCH: "sm_121a"
  TORCH_CUDA_ARCH_LIST: "12.1a"
  VLLM_ALLOW_LONG_MAX_MODEL_LEN: "1"
  VLLM_RPC_TIMEOUT: "1800000"
  NCCL_MAX_NCHANNELS: "4"
  NCCL_MIN_NCHANNELS: "4"
  PYTORCH_CUDA_ALLOC_CONF: "expandable_segments:True"
  VLLM_WORKER_MULTIPROC_METHOD: "spawn"
  VLLM_USE_FLASHINFER_SAMPLER: "1"
  VLLM_USE_V2_MODEL_RUNNER: "1"
  VLLM_USE_B12X_SPARSE_INDEXER: "1"
  VLLM_DCP_GLOBAL_TOPK: "1"
  VLLM_DCP_SHARD_DRAFT: "1"
  VLLM_KZ_TRIM_AFTER_LOAD: "1"
  VLLM_USE_B12X_MOE: "0"
  VLLM_USE_B12X_FP8_GEMM: "0"
  VLLM_DISABLE_TP_MQ_BROADCASTER: "1"
  VLLM_ENABLE_PCIE_ALLREDUCE: "0"
  USES_B12X: "True"
  FLASHINFER_DISABLE_VERSION_CHECK: "1"
  RAY_memory_usage_threshold: "0.99"
  RAY_memory_monitor_refresh_ms: "0"

command: |
  vllm serve {model} \
      --served-model-name {served_model_name} \
      --trust-remote-code \
      --load-format {load_format} \
      --quantization {quantization} \
      --distributed-executor-backend ray \
      --tensor-parallel-size {tensor_parallel} \
      --pipeline-parallel-size {pipeline_parallel} \
      --decode-context-parallel-size {decode_context_parallel} \
      --dcp-comm-backend ag_rs \
      --dcp-kv-cache-interleave-size 1 \
      --gpu-memory-utilization {gpu_memory_utilization} \
```

```
      --max-model-len {max_model_len} \
      --max-num-seqs {max_num_seqs} \
      --max-num-batched-tokens {max_num_batched_tokens} \
      --kv-cache-dtype {kv_cache_dtype} \
      --kv-cache-memory-bytes {kv_cache_memory_bytes} \
      --generation-config vllm \
      --hf-overrides
'{"use_index_cache":true,"index_topk_pattern":"FFFSSSFSSSFSSSFSSS
FSSSFSSSFSSSFSSSFSSSFSSSFSSSFSSSFSSSFSSSFSSSFSSSFSSSFSSSFSSS"}' \
      --port {port} \
      --host {host} \
      --no-enable-log-requests \
      --compilation-config '{"cudagraph_mode":"NONE"}' \
      --attention-backend B12X_MLA_SPARSE \
      --moe-backend flashinfer_cutlass \
      --reasoning-parser {reasoning_parser} \
      --tool-call-parser {tool_call_parser} \
      --enable-auto-tool-choice \
      --speculative-config
'{"model":"QuantTrio/GLM-5.2-Int4-Int8Mix","method":"mtp","num_sp
eculative_tokens":3,"moe_backend":"flashinfer_cutlass","draft_att
ention_backend":"B12X_MLA_SPARSE","draft_sample_method":"probabil
istic"}' \
      --long-prefill-token-threshold 2048 \
      --async-scheduling
```

SSH Authorization Hatası Çözümü

Eğer sparkrun komutu çalıştırıldıktan sonra kendine bağlanırken authorization ile ilgili bir hata alınırsa aşağıdaki komut kullanılır ve sparkrun tekrar çalıştırılır:

Shell

```
cat ~/.ssh/id_ed25519.pub >> ~/.ssh/authorized_keys
```

```
[1/6] Preparing
  done (0.0s)
[2/6] Building image — skipped (no builder)
[3/6] Distributing resources
  SSH script <- 192.168.1.162 FAILED rc=255 (0.1s): nvidia@192.168.1.162: Permission denied (publickey,password).
  Distributing image vllm-zatz-dcp:probe to 4 host(s)
  SSH cmd <- 192.168.1.162 FAILED rc=255 (0.1s): nvidia@192.168.1.162: Permission denied (publickey,password).
  Container image stale on 4 of 4 host(s), syncing
  Pipeline <- 192.168.1.162 FAILED rc=255 (0.3s): nvidia@192.168.1.162: Permission denied (publickey,password).
Image distribution failed on hosts: ['192.168.1.162']
Error: Image distribution failed on: 192.168.1.162
```

Modelin Hazır Olduğunun Doğrulanması

Model başladığında “Application startup complete.” şeklinde bir mesaj alırsınız, artık model kullanıma hazırdır:

```
(APIServer pid=879) INFO 07-21 09:35:14 [api_server.py:576] Starting vLLM server on http://0.0.0.0:8210
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:37] Available routes are:
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /openapi.json, Methods: HEAD, GET
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /docs, Methods: HEAD, GET
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /docs/oauth2-redirect, Methods: HEAD, GET
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /redoc, Methods: HEAD, GET
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /load, Methods: GET
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /version, Methods: GET
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /health, Methods: GET
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /metrics, Methods: GET
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /tokenize, Methods: POST
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /detokenize, Methods: POST
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /v1/models, Methods: GET
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /ping, Methods: GET
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /ping, Methods: POST
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /invocations, Methods: POST
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /v1/chat/completions, Methods: POST
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /v1/chat/completions/batch, Methods: POST
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /v1/responses, Methods: POST
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /v1/responses/{response_id}, Methods: GET
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /v1/responses/{response_id}/cancel, Methods: POST
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /v1/completions, Methods: POST
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /v1/messages, Methods: POST
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /v1/messages/count_tokens, Methods: POST
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /generative_scoring, Methods: POST
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /inference/v1/generate, Methods: POST
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /scale_elastic_ep, Methods: POST
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /is_scaling_elastic_ep, Methods: POST
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /v1/chat/completions/render, Methods: POST
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /v1/completions/render, Methods: POST
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /v1/chat/completions/derender, Methods: POST
(APIServer pid=879) INFO 07-21 09:35:14 [launcher.py:46] Route: /v1/completions/derender, Methods: POST
(APIServer pid=879) INFO:     Started server process [879]
(APIServer pid=879) INFO:     Waiting for application startup.
(APIServer pid=879) INFO:     Application startup complete.
```

Benchmark Sonuçları

Bu şekilde ayağa kalkan modelde linkte verdiğimiz benchmark aracıyla yapılan testler sonucunda elde edilen ortalama değerler şu şekildedir:

Eşzamanlı İstek	TTFT (ms)	Token/sn	Gecikme (sn)	Verim (RPS)
1	565	19.47	6.60	0.15
2	1419	16.90	7.91	0.13
4	1263	12.75	10.39	0.10
8	1928	8.65	15.38	0.07
16	1729	6.85	18.83	0.05

11. NAS Yapılandırmasının Yapılması (ASUSTOR AS6808T)

Bu yapıda kullanılan NAS cihazının kutu açılışından itibaren kurulumunun yapılması, bonding yapılarak CRS312 switch'e 2×10Gbps LACP ile bağlanması, paylaşılan bir klasör oluşturulması, izinlerinin verilmesi ve bu dizinin bir Spark'a mount edilmesi adımları anlatılacaktır.

Fiziksel Kurulum

1. 8 adet HDD diski cihazınıza takın.
2. NAS arkasında bulunan iki adet 10Gbps portu CRS312 cihazının Combo3 ve Combo4 portlarına Cat6 kablo ile bağlayın.
3. Güç bağlantısını yapın.

NAS'ı İlk Açılış ve Fabrika Ayarlarına Alma

Power düğmesine 1-2 saniye basılı tutarak cihazı açın. Açılış sırasında LCD panelde sırasıyla şu mesajlar görünecektir:

1. "Starting system please wait" — sistem başlıyor
2. "booting-service" / "booting-network" — servisler ve ağ başlatılıyor
3. "Initialize NAS?" — ilk kurulum/fabrika ayarı sorusu

Ekranda "Initialize NAS?" sorusu geldiğinde, YES seçili iken LCD panelin sağ alt köşesindeki Confirm (enter) düğmesine basın.

Ardından "Delete all data?" sorusu çıkacaktır. Tüm veriler silineceği için eminseniz, YES seçili iken tekrar Confirm düğmesine basın.

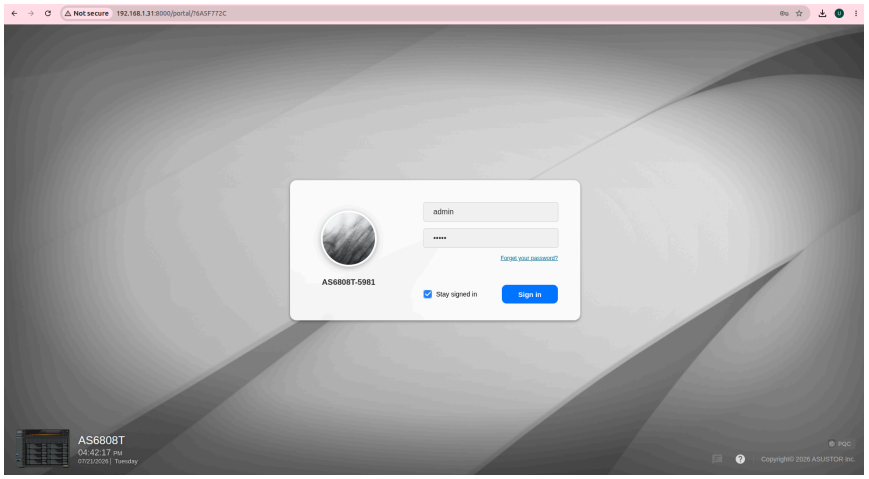
Ekranda "Initializing please wait" ifadesi görünür ve işlem başlar. Bir süre sonra işlem tamamlanır ve LCD panelde NAS'a atanmış IP adresi görünür.

NAS IP Adresini Bulma

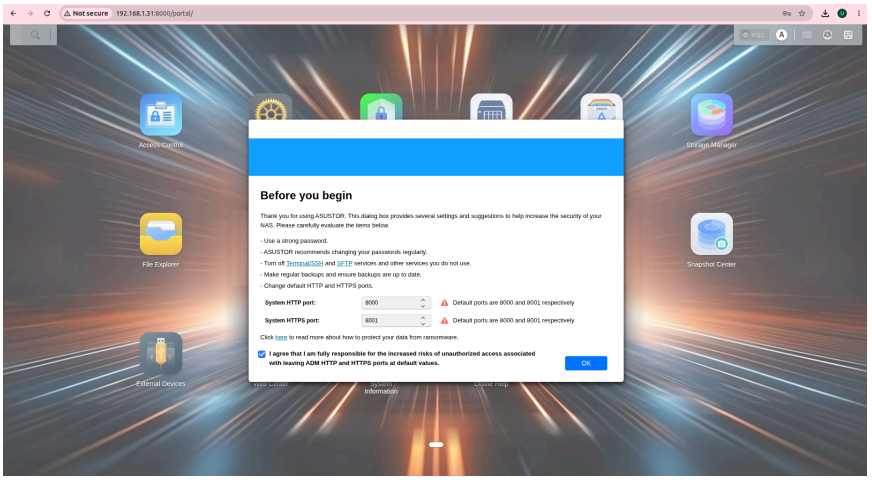
NAS fabrika ayarlarında DHCP ile IP alır. Ağınızda DHCP sunucu varsa, IP adresi LCD panelde otomatik görünür.

Eğer ağınızda DHCP sunucu yoksa, NAS link-local adresi (169.254.x.x) alır. Bu durumda bilgisayarınızı aynı link-local aralığına getirerek erişebilirsiniz.

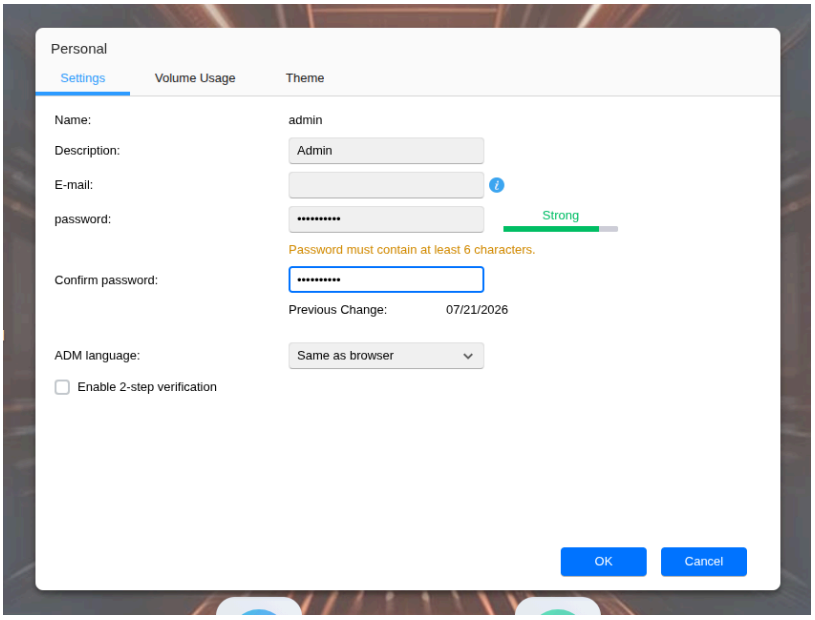
IP adresini öğrendikten sonra tarayıcıdan http://<nas-ip>:8000 adresine gidin. Örneğin: http://192.168.1.31:8000



İlk kurulum ekranında varsayılan kullanıcı adı ve şifre admin / admin'dir. Giriş yaptıktan sonra default olan port yapılandırmasını gösterecektir, değiştirebilir veya default ayarlarla devam edebilirsiniz:



Şifre değiştirmeniz istenirse şifrenizi değiştirin, eğer istenmezse sağ üstte bulunan A isimli simgeye tıklayıp buradan Personal seçeneğini seçerek açılan ekranda şifrenizi değiştirebilirsiniz:



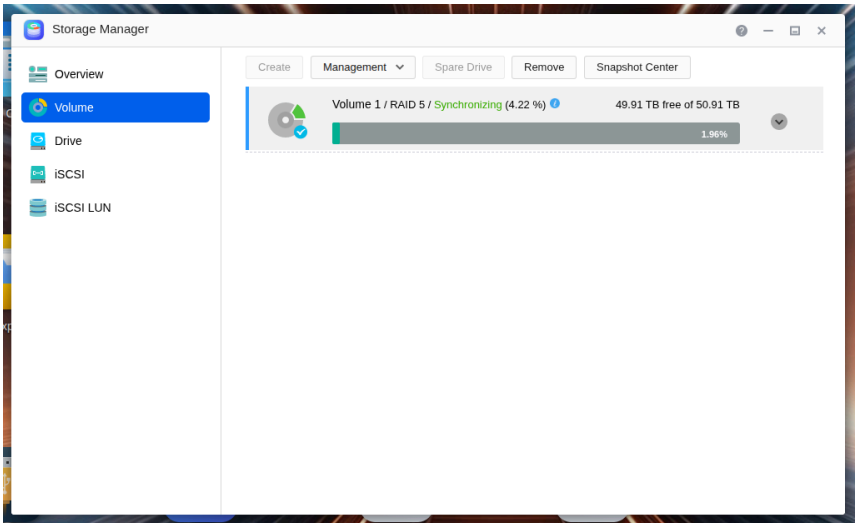
RAID Yapılandırması
NAS fabrika ayarlarında 8 disk ile RAID 5 yapılandırılmış olarak gelir. RAID 5, 1 disk toleransı sağlar ve ~51 TB kullanılabilir alan sunar.

RAID	Disk Toleransı	Kullanılabilir Alan	Hız	Önerilen Kullanım
RAID 0	Yok	58 TB	En Hızlı	Maksimum performans, veri güvenliği önemli değil
RAID 5	1 Disk	51 TB	Orta	Varsayılan, dengeli kullanım
RAID 6	2 Disk	44 TB	Düşük	Yüksek veri güvenliği

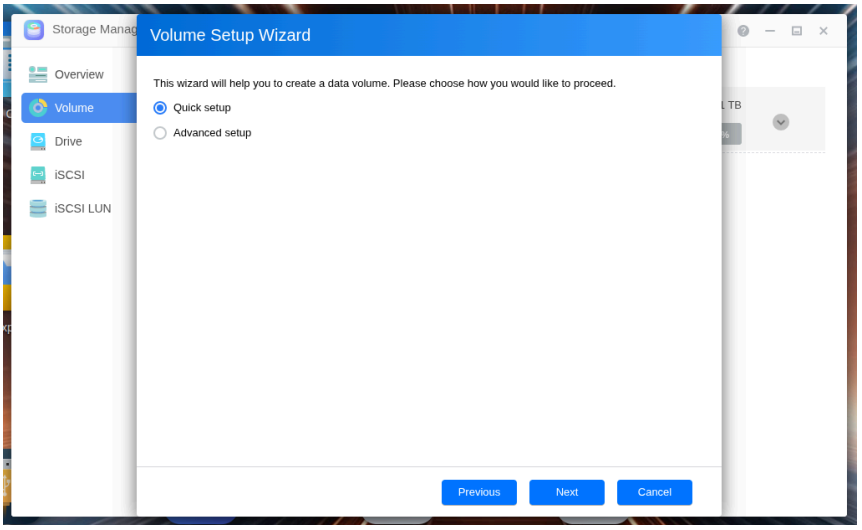
Not: Disk etiketlerinde 8 TB yazmasına rağmen, NAS arayüzünde 7.28 TB olarak görünür. Bunun sebebi disk üreticilerinin kapasiteyi ondalık sistemde belirtmesi, işletim sistemlerinin ise ikili sistemde hesaplamasıdır. 8 TB (ondalık) ≈ 7.28 TB (ikili) olarak görünür. Aşağıdaki değerler ikili sisteme göre hesaplanmıştır.

Bu dokümanda hızlı olması için ama yedeklikten feragat ederek RAID 0 yapılandırılması anlatılacaktır. Default olarak RAID 5 ile yapılandırılmış gelen sistemi kullanmak isterseniz bu adımı atlayabilirsiniz. Eğer 2 disk yedeği istiyorsanız RAID 6 tercih edebilirsiniz, fakat bu durumda hızınız ve depolama alanınız azalır. İhtiyacınıza göre seçebilirsiniz.

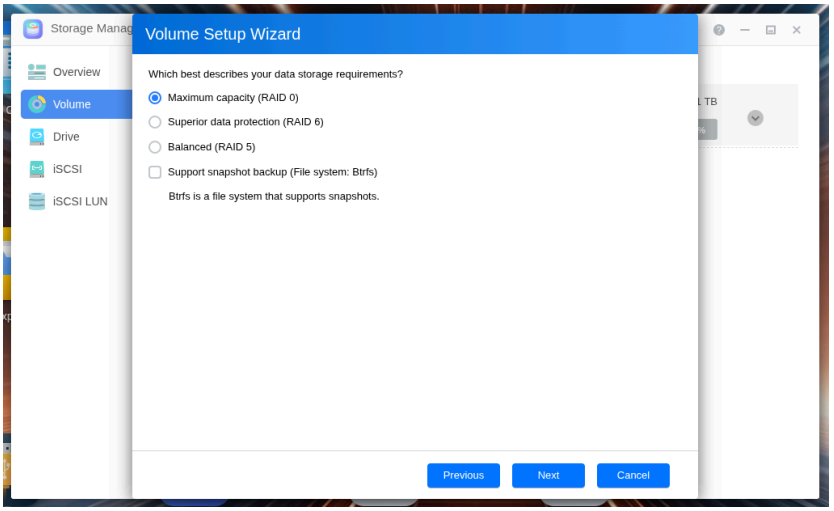
RAID 0 Yapılandırma Adımları:
1. NAS web arayüzünde Storage Manager → Volume sayfasına gidin
2. Mevcut RAID 5 volume'u seçin
3. Remove butonuna tıklayın



4. Gelen ekranda Quick Setup seçeneğini seçin



5. Sonraki ekranda RAID 0 seçeneğini seçin

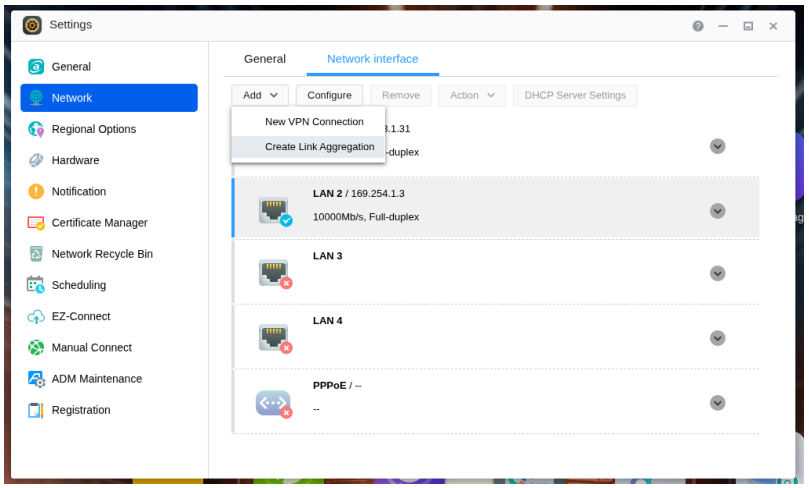


6. Finish butonuna basın

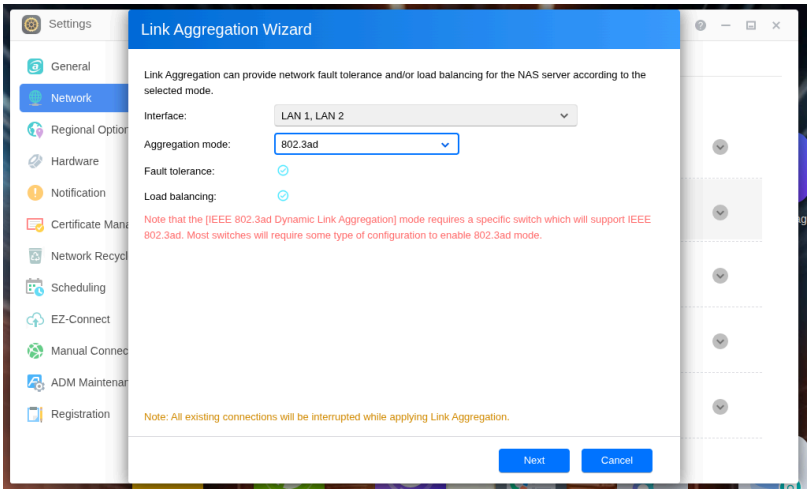
İşlem tamamlandığında 8 diskin oluşturduğu RAID 0 volume hazır olacaktır. Toplam kullanılabilir alan ~58 TB olacaktır.

Ağ Yapılandırması — LACP Bonding
NAS'ın iki adet 10Gbps Ethernet portunu tek bir bonding interface altında birleştirerek 2×10Gbps LACP bağlantısı sağlayın:

1. NAS web arayüzünde Settings → Network → Network Interface sayfasına gidin
2. Add → Create Link Aggregation seçeneğine tıklayın

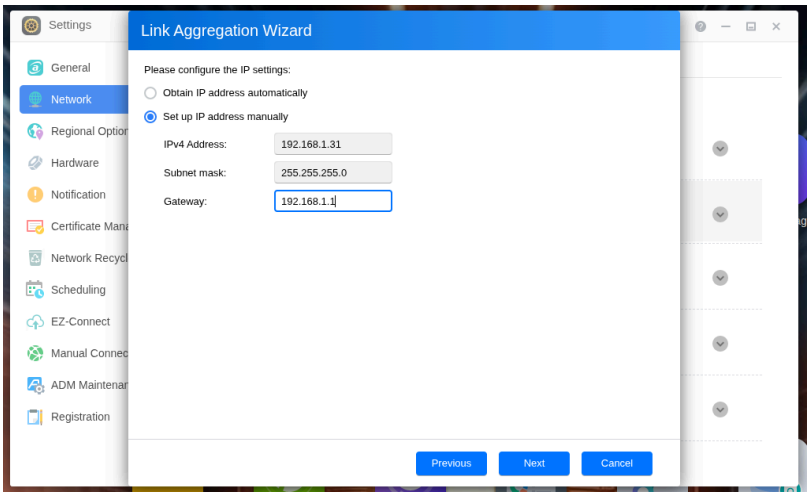


3. Interface alanında LAN 1 ve LAN 2'yi seçin
4. Aggregation Mode: 802.3ad seçin
5. Next butonuna basın



6. Set up IP address manually seçeneğini işaretleyin
7. Aşağıdaki bilgileri girin:
    a. IP Address: 192.168.1.31
    b. Subnet Mask: 255.255.255.0
    c. Gateway: 192.168.1.1

8. Next butonuna basın

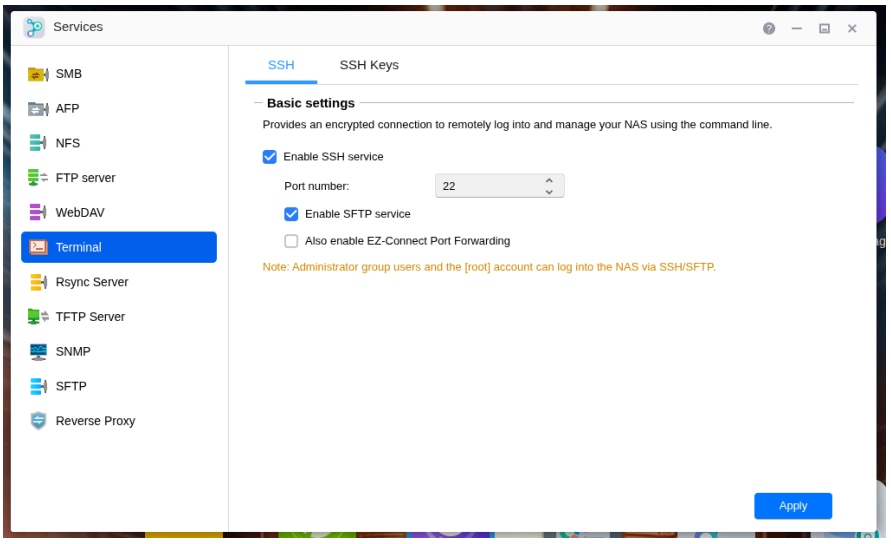


9. Özet ekranını kontrol edin ve Finish butonuna basın

İşlem tamamlandığında NAS, iki Ethernet portunu tek bir bonding interface olarak kullanmaya başlayacaktır. CRS312 switch tarafındaki LACP bonding ile birlikte 2×10Gbps aggregate throughput sağlanır.

SSH Erişimini Etkinleştirme
Performance tuning script'ini yüklemek için NAS'a SSH ile bağlanmak gereklidir. SSH varsayılan olarak kapalıdır.

1. NAS web arayüzünde Services → Terminal sayfasına gidin
2. Enable SSH Service seçeneğini işaretleyin
3. Apply butonuna basın



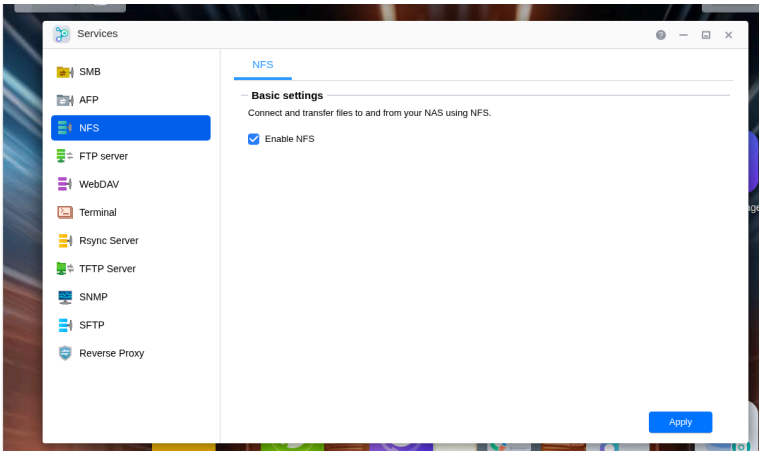
Artık NAS'a SSH ile bağlanabilirsiniz:

```
Shell
ssh admin@192.168.1.31
```

Kullanıcı adı admin, şifre kurulum sırasında belirlediğiniz şifredir.

NFS Servisini Etkinleştirme

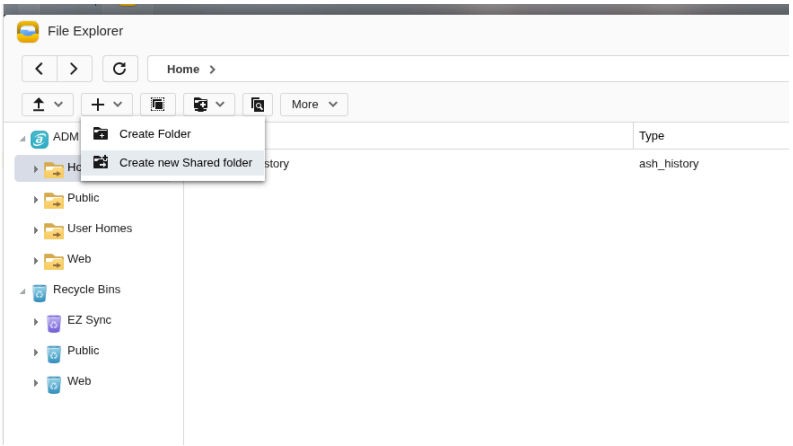
1. NAS web arayüzünde Services → NFS sayfasına gidin
2. Enable NFS Service seçeneğini işaretleyin
3. Apply butonuna basın
4. NFS servisinin tam olarak başlaması için NAS'ı yeniden başlatın



Önemli: NFS servisini enable yaptıktan sonra NAS'ı yeniden başlatmanız gerekir. Aksi takdirde NFS export eklerken "unknown error (ref. 5052)" hatası alabilirsiniz.

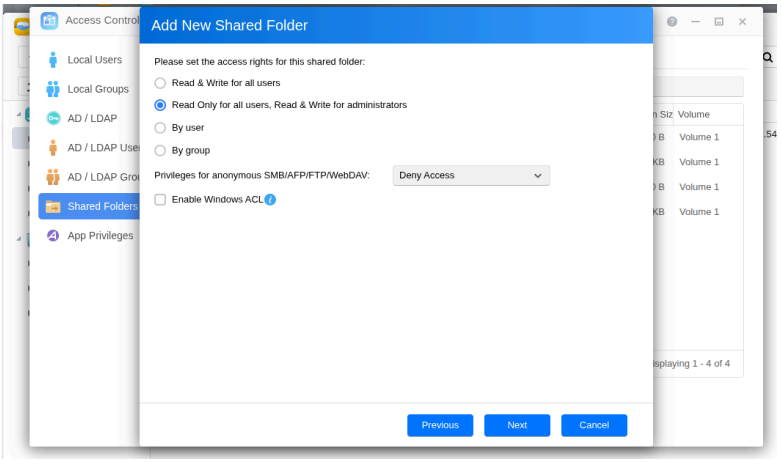
Paylaşılan Klasör Oluşturma

1. NAS web arayüzünde File Explorer → + (Create New Shared Folder) butonuna tıklayın

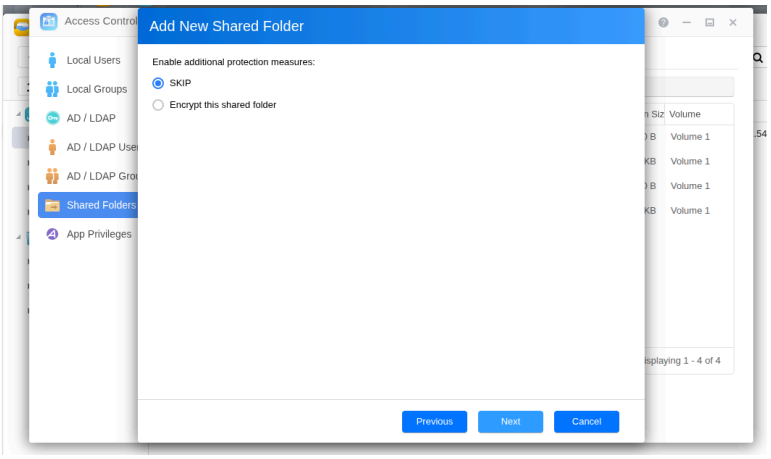


2. Add butonuna tıklayın
    a. Name: bir isim yazın
3. Next butonuna basın

4. Access Rights ekranında:
    a. Read and Write for all users seçeneğini seçin (herkesin okuma/yazma yetkisi olur)
    b. Veya default olan Read and Write for admins seçeneğini bırakabilirsiniz



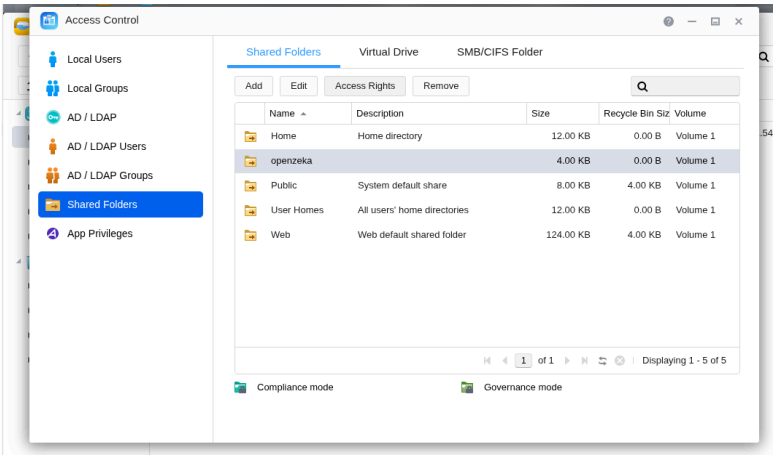
5. Next butonuna basın
6. Additional protection measures: Skip seçeneğini seçin
    a. Encrypt this shared folder: İsterseniz seçebilirsiniz, bu dokümanda seçilmemiştir



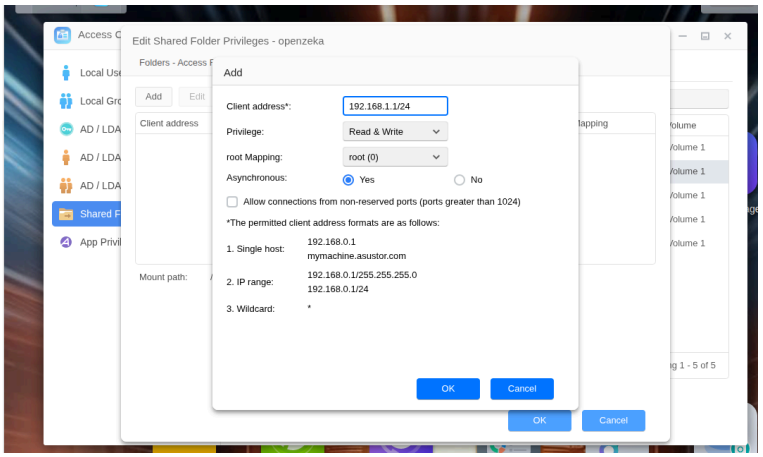
7. Next → Finish butonuna basın

NFS İzinlerini Verme

1. NAS web arayüzünde Access Control → Shared Folders sayfasına gidin
2. İlgili klasörü seçin
3. Access Rights butonuna tıklayın



4. NFS Privileges sekmesine geçin
5. Add butonuna tıklayın
6. Aşağıdaki bilgileri girin:
    a. Client Address: 192.168.1.1/24 (kendi ağ aralığınızı girin, /24 maskesi tüm alt ağı kapsar)
    b. Privilege: Read and Write
    c. Root Mapping: root (0)
7. OK butonuna basın



Performance Tuning Script

NAS'a SSH ile bağlanın ve performance tuning script'ini oluşturun. Bu script her reboot'ta otomatik çalışır ve bonding hash policy'sini layer3+4'e değiştirir:

```shell
ssh admin@192.168.1.31
```

```shell
sudo tee /usr/local/etc/init.d/S98nfstune > /dev/null <<'EOF'
```

```
#!/bin/sh
# NFS + RAID0 performance tuning - persistent across reboots
case "$1" in
  start)
    # wait for NFS and network to be ready
    sleep 30

    # RAID0 read-ahead 8MB
    blockdev --setra 8192 /dev/md1 2>/dev/null
    # per-disk read-ahead 8MB
    for d in a b c d e f g h; do blockdev --setra 8192 /dev/sd$d
2>/dev/null; done
    # nfsd threads 64
    echo 64 > /proc/fs/nfsd/threads 2>/dev/null
    # RPC slot table
    echo 128 > /proc/sys/sunrpc/tcp_slot_table_entries
2>/dev/null
    echo 128 > /proc/sys/sunrpc/tcp_max_slot_table_entries
2>/dev/null
    # VM writeback cache
    echo 40 > /proc/sys/vm/dirty_ratio
    echo 5 > /proc/sys/vm/dirty_background_ratio
    echo 6000 > /proc/sys/vm/dirty_expire_centisecs
    # bond xmit_hash_policy layer3+4
    sh -c 'echo layer3+4 >
/sys/class/net/bond0/bonding/xmit_hash_policy' 2>/dev/null

    # exports: async + wdelay (NO no_wdelay) - allow write
batching
    [ -f /volume0/etc/exports ] && sed -i "s/no_wdelay,//"
/volume0/etc/exports 2>/dev/null
    /volume0/usr/builtin/sbin/exportfs -ra 2>/dev/null
    ;;
  stop)
    ;;
esac
exit 0
EOF
```

```shell
sudo chmod +x /usr/local/etc/init.d/S98nfstune
sudo /usr/local/etc/init.d/S98nfstune start
```

Script çalıştırıldıktan sonra hash policy'nin değiştiğini doğrulayın:

```shell
cat /sys/class/net/bond0/bonding/xmit_hash_policy
# Beklenen: layer3+4
```

Spark Node'lara NAS Mount
Her Spark'ta NFS mount noktası oluşturun ve NAS'a bağlayın:

```shell
sudo mkdir -p /mnt/asustor
sudo mount -t nfs \
  -o vers=4.2,nconnect=8,rsize=1048576,wsize=1048576,hard,noatime
\
  192.168.1.31:/volume1/openzeka /mnt/asustor
```

Basit Yazma/Okuma Testi
NAS mount edildikten sonra temel yazma ve okuma performansını test edin. Yazma Testi:

```shell
dd if=/dev/zero of=/mnt/asustor/testfile bs=1M count=10240
conv=fdatasync
```

Okuma Testi:

```shell
# Cache boşalt:
sync; echo 3 | sudo tee /proc/sys/vm/drop_caches

# Okuma:
dd if=/mnt/asustor/testfile of=/dev/null bs=1M
```

Temizlik:

```shell
rm /mnt/asustor/testfile
```

12. Sonuç ve Doğrulama

Bu dokümandaki adımları takip ederek aşağıdaki bileşenlerden oluşan tam fonksiyonel bir DGX Spark AI cluster'ı kurulmuş olur:

Bileşen	Durum	Doğrulama Yöntemi
Management Ağı (10GbE)	Hazır	Düğümler arası ping başarılı
Compute Ağı (200GbE QSFP)	Hazır	`ib_write_bw` ~100-111 Gbps
RoCEv2 / RDMA	Hazır	`ib_write_lat` ~1-3 µs
`sparkrun` Cluster	Hazır	sparkrun setup tamamlandı
DCB / PFC	Hazır	dcb pfc show çıktısı TC3:on
NAS (NFS)	Hazır	`dd` yazma/okuma testi
Model Servisi	Hazır	"Application startup complete." mesajı

Cluster Sağlık Kontrolü Özeti

Kurulumun tamamlandığını doğrulamak için aşağıdaki kontrolleri gerçekleştirebilirsiniz:

1. **Management ağı:** Tüm düğümler birbirini ping edebiliyor mu?
2. **Compute ağı:** `ib_write_bw` testinde her subnet ~100 Gbps veriyor mu?
3. **MTU:** `ping -M do -s 8972` testi paket düşmeden geçiyor mu?
4. **sparkrun mesh:** Tüm node'lara şifresiz SSH erişimi çalışıyor mu?
5. **DCB servisi:** `systemctl status dcb-roce.service` active görünüyor mu?
6. **NAS mount:** `df -h /mnt/asustor` mount noktasını gösteriyor mu?
7. **Model:** Benchmark sonuçları yukarıdaki tabloyla tutarlı mı?

Tüm bu kontroller başarılıysa cluster AI iş yükleri için hazırdır.

13. Sorun Giderme

Docker Storage Driver `overlay2` Görünüyor

`docker info` çıktısında `overlay2` görüyorsanız `/etc/docker/daemon.json` dosyasına `containerd-snapshotter` feature'ını ekleyin ve Docker'ı yeniden başlatın (bkz. Docker Yapılandırması).

ping -M do ile Jumbo Frame Testi Başarısız

- Switch tarafında QSFP-DD portlarında `l2mtu=9500 mtu=9000` ayarlarının yapıldığından emin olun.
- Spark tarafında CX7 arayüzlerinin MTU değerlerinin 9000 olduğundan emin olun: `ip link show`.

RDMA Bant Genişliği Düşük (~100 Gbps Altında)

- Tüm sistem ve firmware güncellemelerinin yapıldığından emin olun (bkz. Sistem ve Firmware Güncellemeleri).
- Switch tarafında QoS hardware offloading'in açık olduğundan emin olun: `qos-hw-offloading=yes`.
- DCB ayarlarının uygulandığını doğrulayın: `dcb pfc show dev enp1s0f1np1`.
- `dcb-roce.service` servisinin çalıştığını kontrol edin: `systemctl status dcb-roce.service`.

sparkrun SSH Authorization Hatası

sparkrun çalıştırıldığında authorization hatası alınırsa:

```
Shell
cat ~/.ssh/id_ed25519.pub >> ~/.ssh/authorized_keys
```

Komutunu çalıştırın ve sparkrun'u tekrar çalıştırın.

NFS Export "unknown error (ref. 5052)" Hatası

NFS servisini enable yaptıktan sonra NAS'ı yeniden başlatın. NFS servisi tam olarak başlamadan export eklemeye çalışırsanız bu hatayı alırsınız.

NAS Bond Hash Policy layer2+4 Görünüyor

Performance tuning script'ini çalıştırın ve doğrulayın:

```
Shell
sudo /usr/local/etc/init.d/S98nfstune start

cat /sys/class/net/bond0/bonding/xmit_hash_policy
```

```
# Beklenen: layer3+4
```

CRS812 Breakout Portları Link Up Olmuyor

- Breakout kablolarının mandallarının tam oturduğunu kontrol edin.
- `auto-negotiation=no` ve `speed=200G-baseCR4` ayarlarının doğru portlara uygulandığından emin olun.
- Kabloyu çıkarıp tekrar takın ve port durumunu kontrol edin: `/interface/ethernet/print`.