High Available RKE2 Kubernetes Cluster için OpenStack Altyapısında Network ve Sanal Makine Yapılandırılması

OpenStack üzerinde kuracağımız Kubernetes Cluster'i için kullanacağımız yapı 3 Master, 3 Worker ve Master HA yapısını uygulayacağımız Bastion LoadBalancer VM'lerinden oluşmaktadır.
Bu kısımda ilk önemli olan yer Network yapılandırılmasıdır. Bu noktada kullanacağımız Kubernetes Cluster Dağıtıcısı olan RKE2'dir. Network tarafında yapacağımız işlemler öncesi RKE2 dokümantasyonu üzerinden ağ yapılandırılmasına dair gereksinim şu şekilde belirtilmiştir:

Inbound Network Rules

Port	Protocol	Source	Destination	Description
6443	TCP	All RKE2 nodes	RKE2 server nodes	Kubernetes API
9345	TCP	All RKE2 nodes	RKE2 server nodes	RKE2 supervisor API
10250	TCP	All RKE2 nodes	All RKE2 nodes	kubelet metrics
2379	TCP	RKE2 server nodes	RKE2 server nodes	etcd client port
2380	TCP	RKE2 server nodes	RKE2 server nodes	etcd peer port
2381	TCP	RKE2 server nodes	RKE2 server nodes	etcd metrics port
30000-32767	TCP	All RKE2 nodes	All RKE2 nodes	NodePort port range

CNI Specific Inbound Network Rules

Canal | Cilium | Calico | Flannel

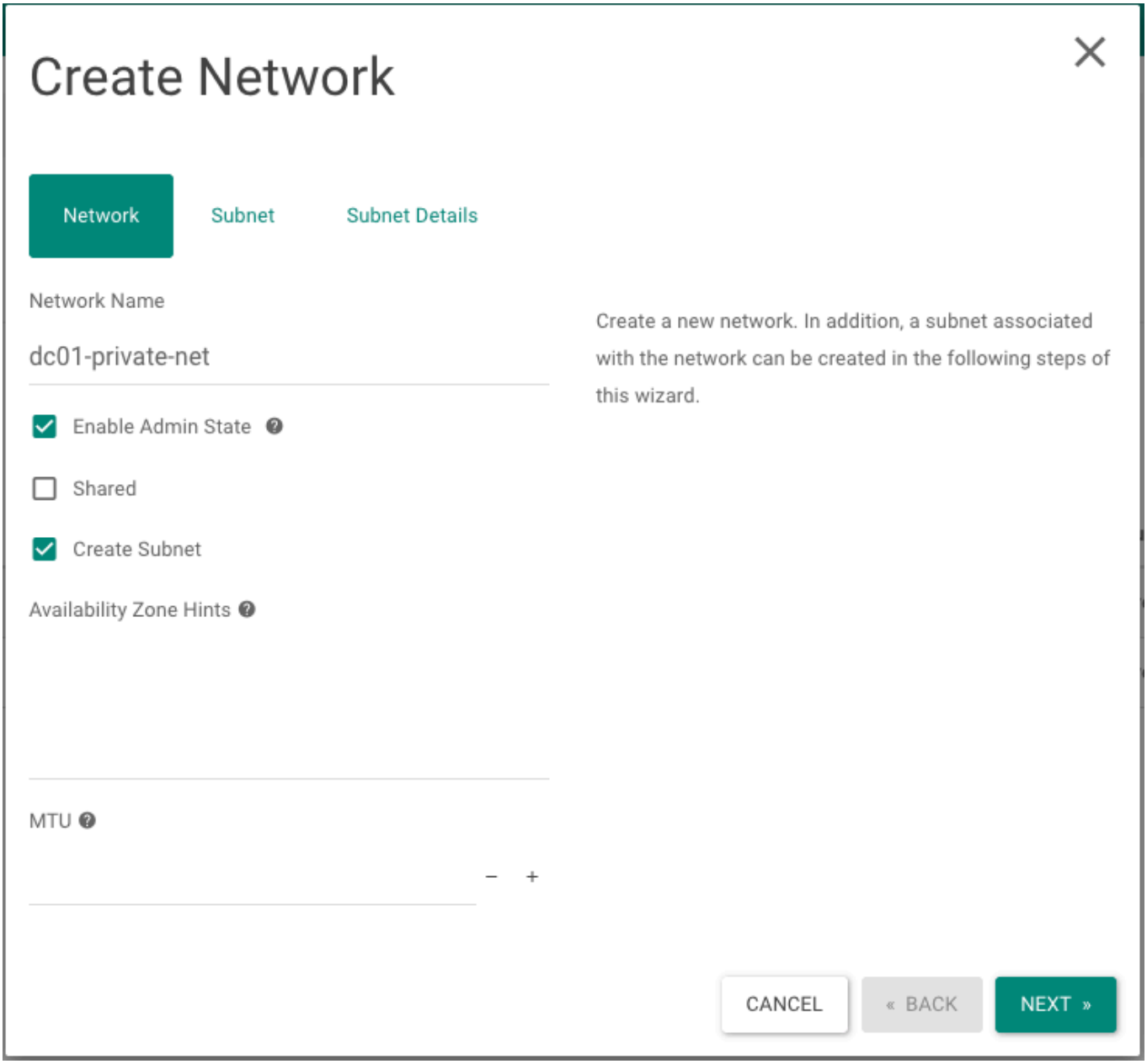
Port	Protocol	Source	Destination	Description
8472	UDP	All RKE2 nodes	All RKE2 nodes	Canal CNI with VXLAN
9099	TCP	All RKE2 nodes	All RKE2 nodes	Canal CNI health checks
51820	UDP	All RKE2 nodes	All RKE2 nodes	Canal CNI with WireGuard IPv4
51821	UDP	All RKE2 nodes	All RKE2 nodes	Canal CNI with WireGuard IPv6/dual-stack

Container Network Interface (CNI) kısmında birden fazla seçeneğimiz vardır. Bu rehberde Canal olarak devam edeceğiz. Bir sonraki kısımda OpenStack üzerinden cluster network ayarlanması üzerinden devam edebiliriz.

OpenStack üzerinden Network Yapılandırılması

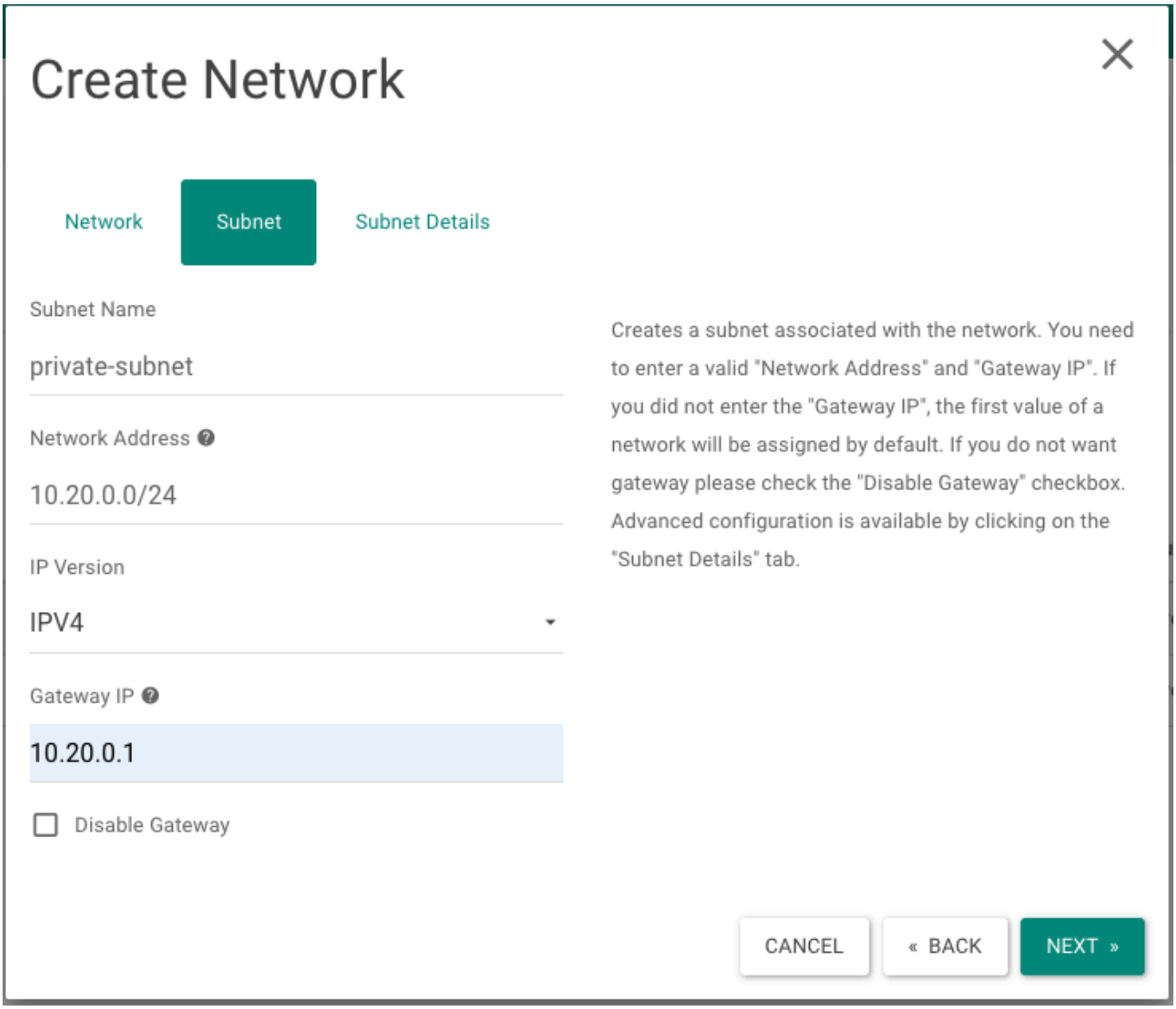
Yapılandırmanın tamamı arayüz üzerinden yapılmıştır aşağıdaki şekilde kendi yapı ve mimarinize uygun şekilde isimlendirmede değişikliliğe gidebilirsiniz. İlk önce Create Network

kısmından örnek datacenter üzerine kurulu olacak kubernetes clusteri ifade etmek için dc01 olarak gidilmiştir. Network ismi yerinde **dc01-private-net** yazarak devam edebiliriz.



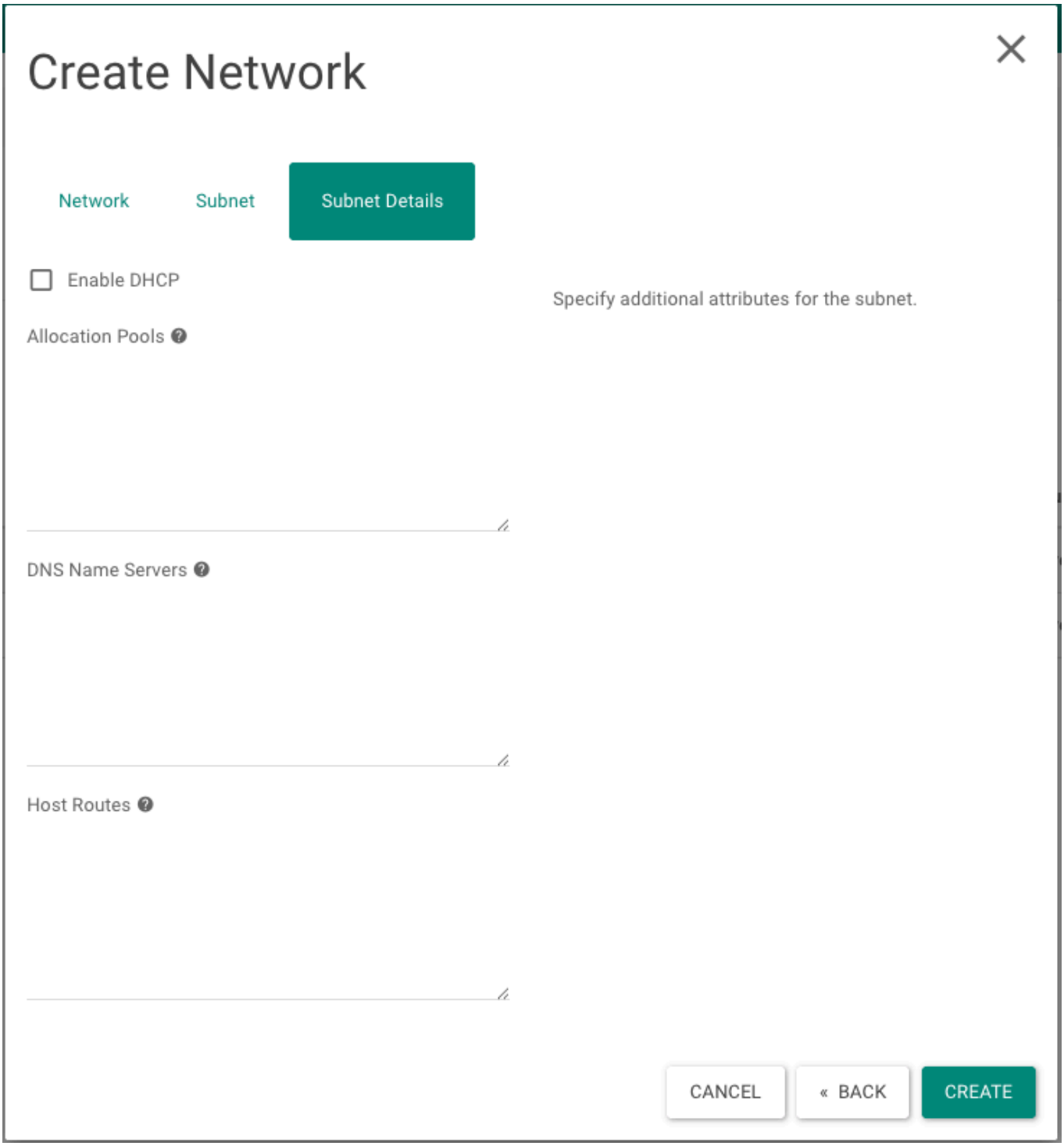
Oluşturulacak node'ların paylaşacağı ip subnet için **private-subnet** diyerek devam edebiliriz. Yapılandıracağımız kubernetes cluster yapısında /24 olarak kullanacağız. Biz **10.20.0.0/24** olarak devam edebiliriz. Gateway IP kısmını boş olarak geçebilirsiniz default olarak ip'yi altyapı otomatik olarak tahsis edecektir.

> Kubernetes service cidr ve pod cidr ile çakışmamasına dikkat ediniz.



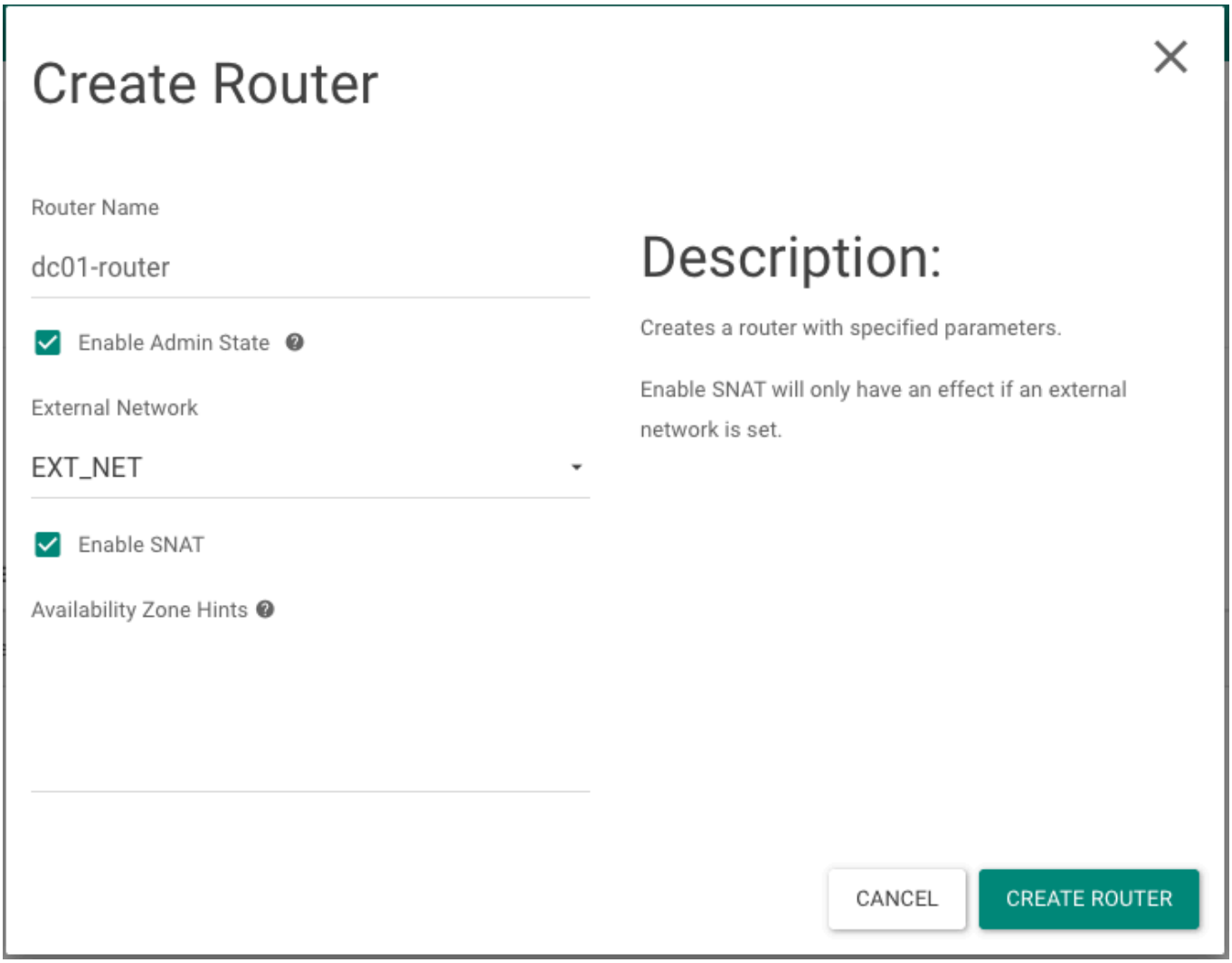
Burada sahip olduğumuz altyapıdan ötürü DHCP açık olarakta devam edebilirsiniz. Ama Kubernetes fixed ip'ler önerildiği ve ip değişikliliğinde yaşanacak platform dengesizlik problemi için DHCP'yi deaktif yapabilirsiniz.

> Bu dokümantasyonda DHCP aktif olarak devam edeceğiz.

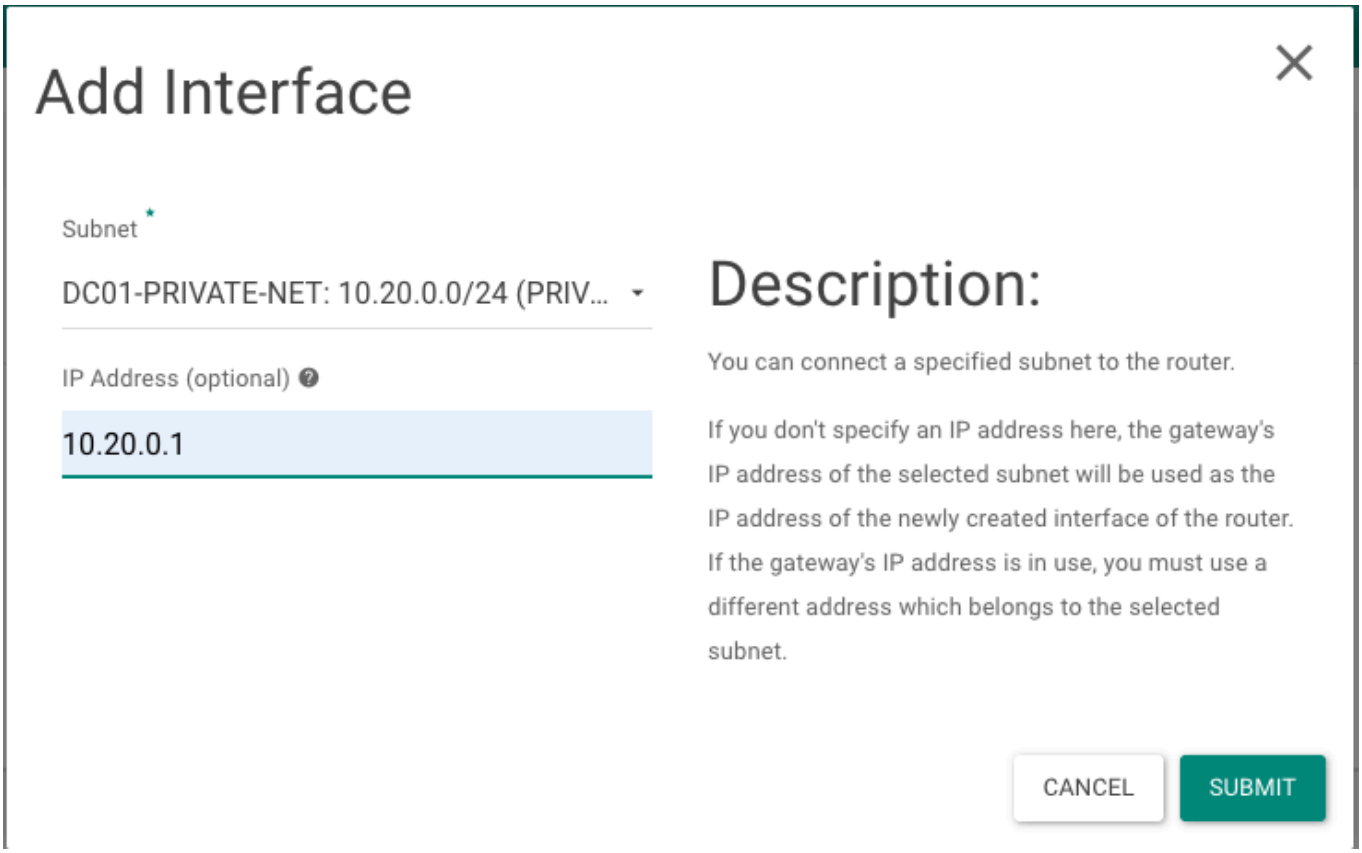


Router'in Oluşturulması

Kubernetes RKE2 kurulumu gerçekleştirebilmek için bazı paketlerin güncelliği ve argümanlara göre dinamik indirip ona göre paketlerin çekilebilmesi için cluster'imizin internete erişimi gerekmektedir bunun için bir router tanımlaması yapalım. İsimlendirme olarak **dc01-router** olarak devam edebiliriz.

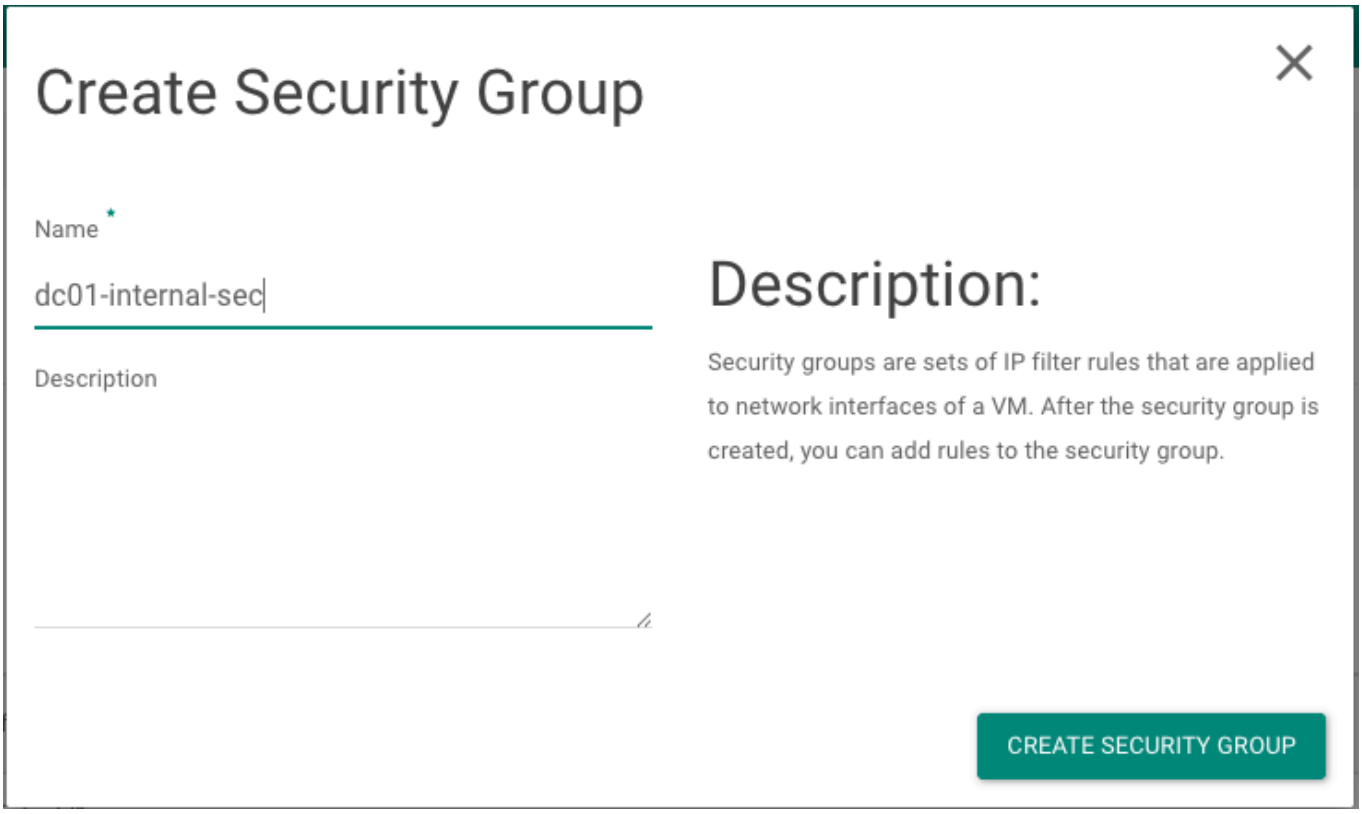


Subnet kısmında önceki başlıkta oluşturduğumuz subnet'i seçerek opsiyonel olarak ip'sini verebiliriz.

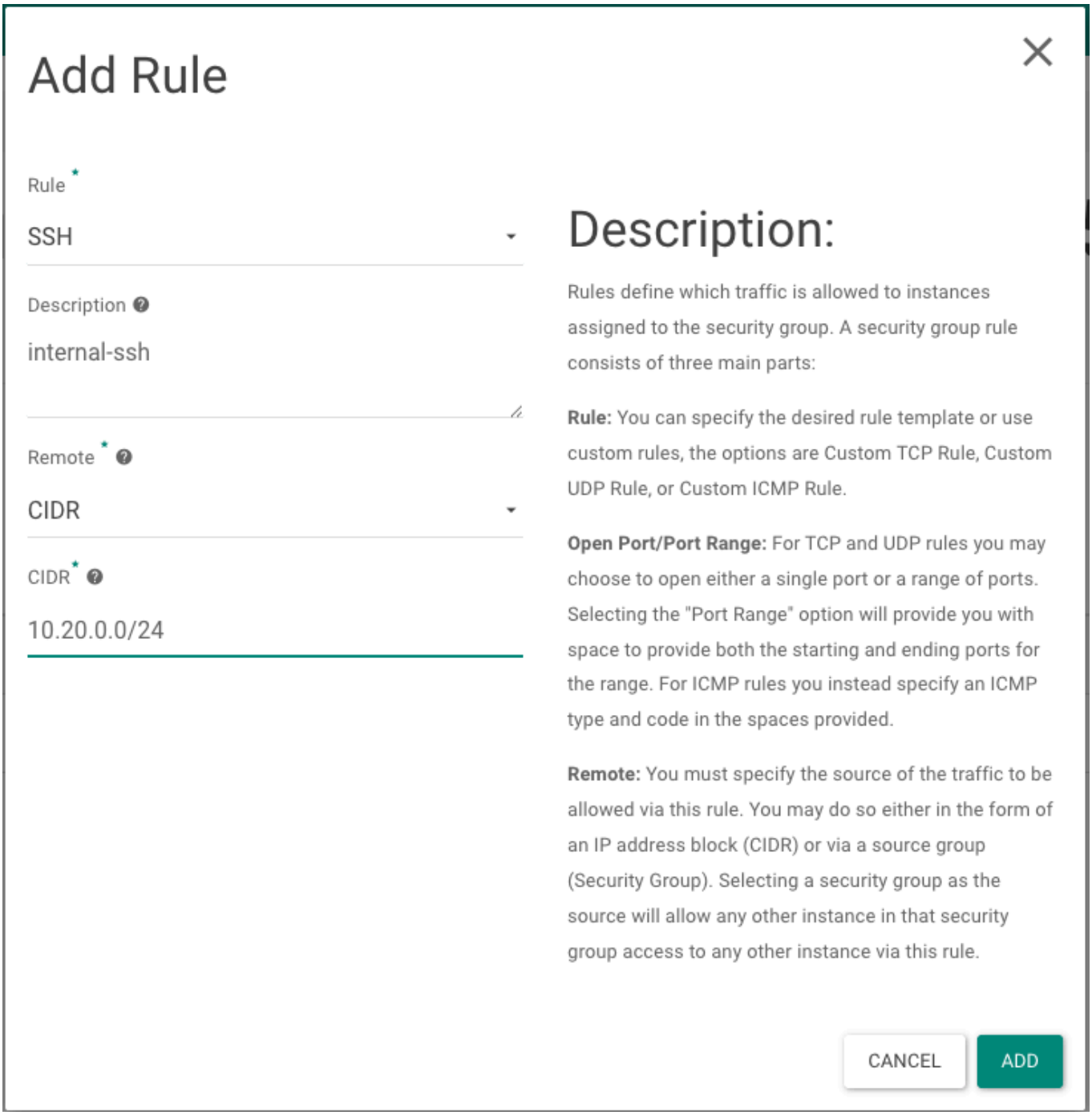


Security Group ile Cluster Internal Konfigürasyonu

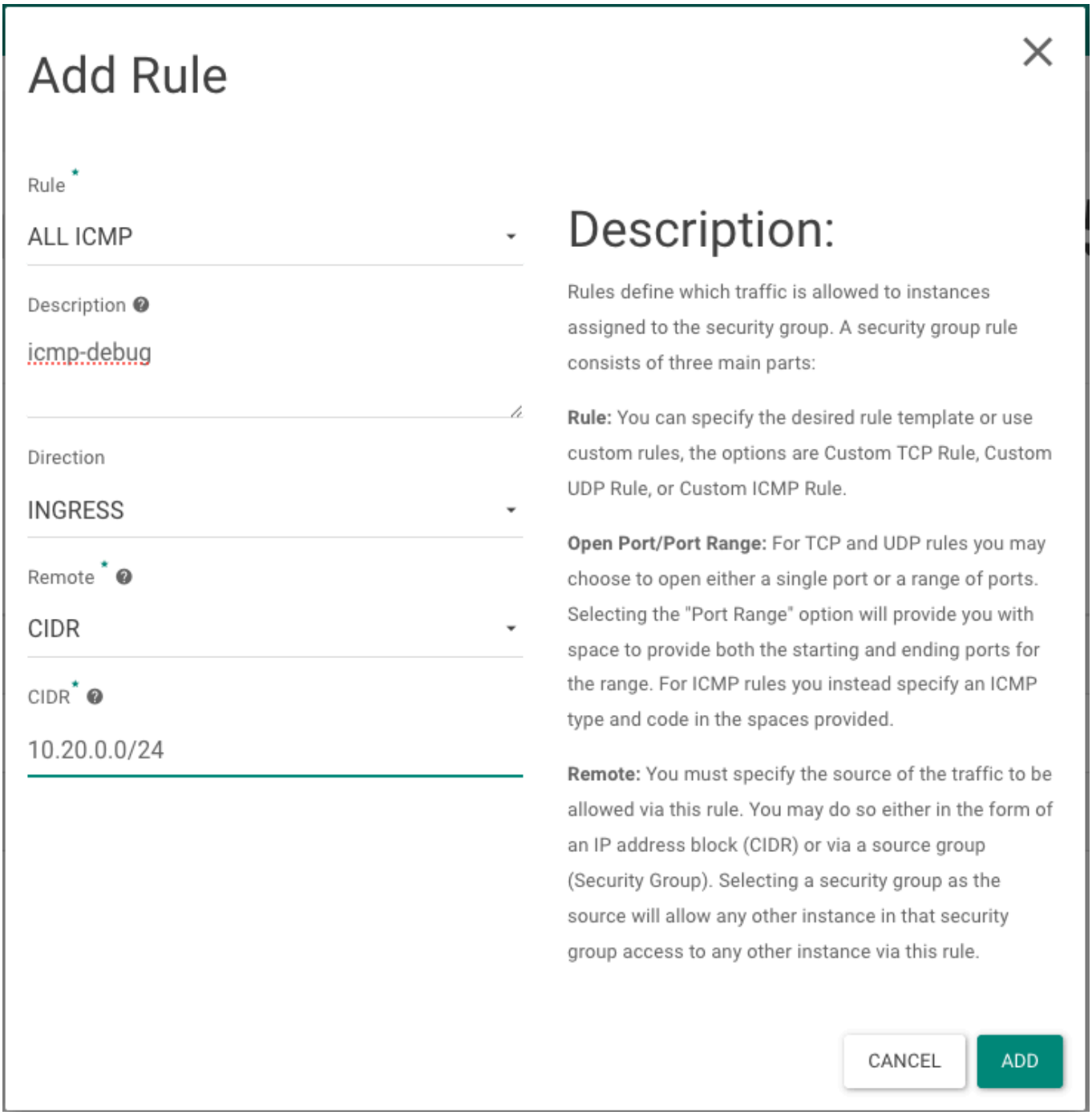
Bu kısımda iki farklı security group tanımlaması ile devam edeceğiz bunlardan ilki RKE2 cluster'imizin master ve worker node'lar üzerindeki RKE2 Gereksinimleri kısmından istenilen port ve protokollere izin vererek devam edeceğiz. İlk önce node-to-node yapısının anlaşılabilir olması için **dc01-internal-sec** yazarak Security Group işlemine başlayalım.



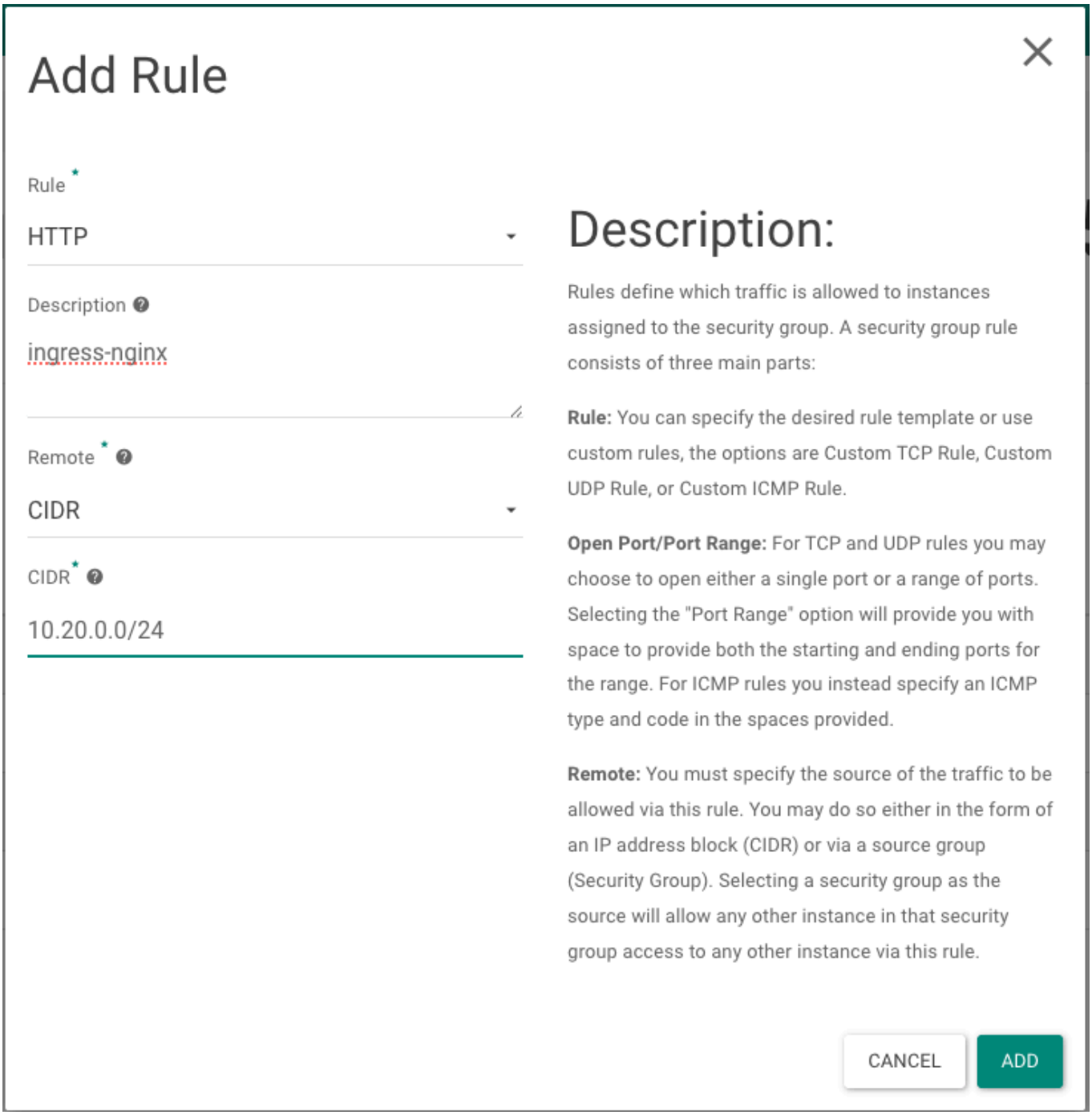
İlk önce oluşturacağımız LoadBalancer tarafından master ve worker node'lara bağlanabilmek için 10.20.0.0/24 altından SSH kuralı ekleyerek devam edelim.



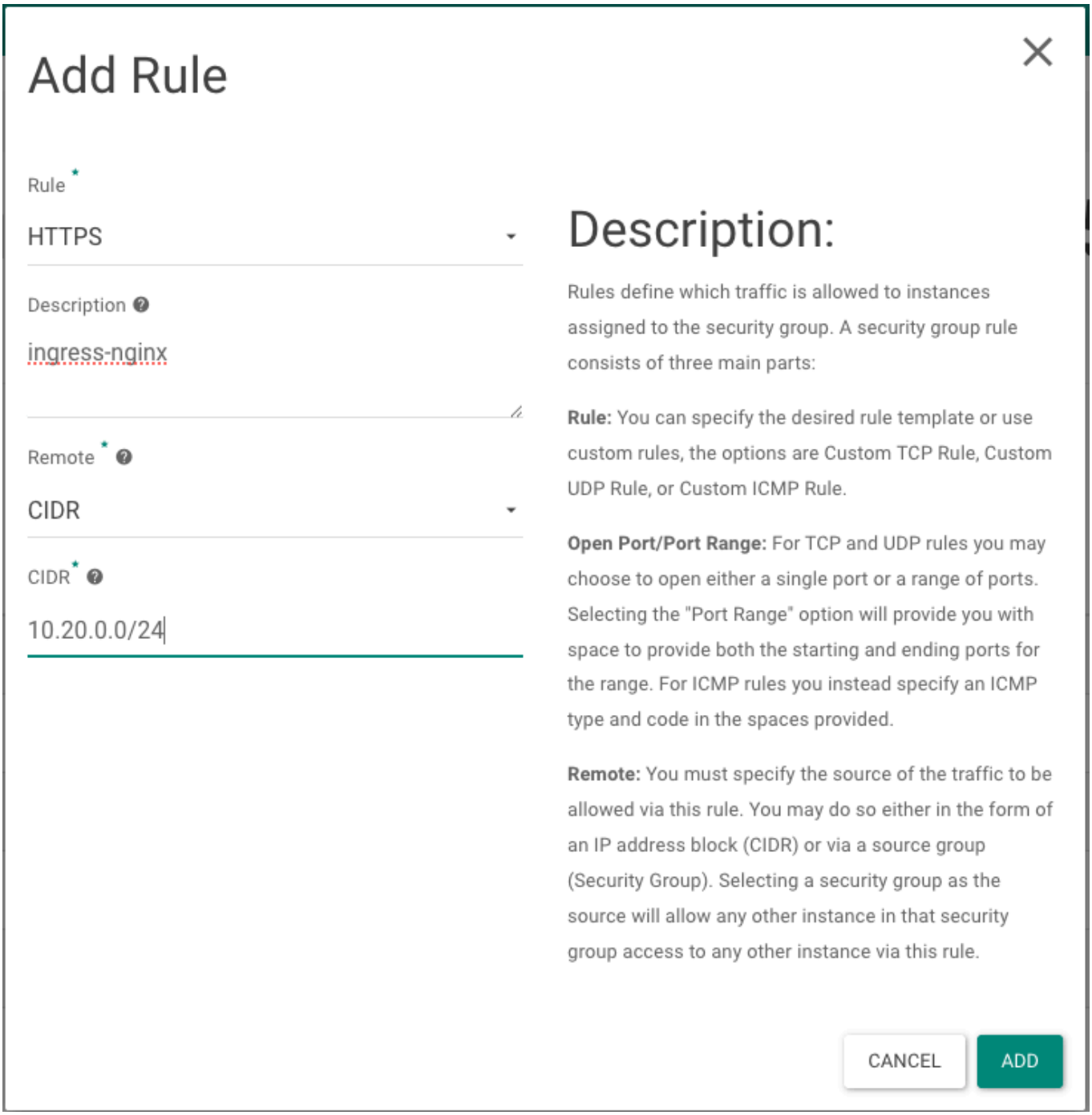
Node'lar arasındaki bağlantının test edilmesi ve bazı noktalardaki network debug işlemi için ping kullanılabilir bundan dolayı ICMP kuralı ekleyerek 10.20.0.0/24 cidr kısmını seçelim.



Cluster kurulumu sonrası kullanacağımız ingress-nginx node'larda hostPort olarak 80 portunu HTTP için açmaktadır bunun için HTTP kuralını da ekleyelim.

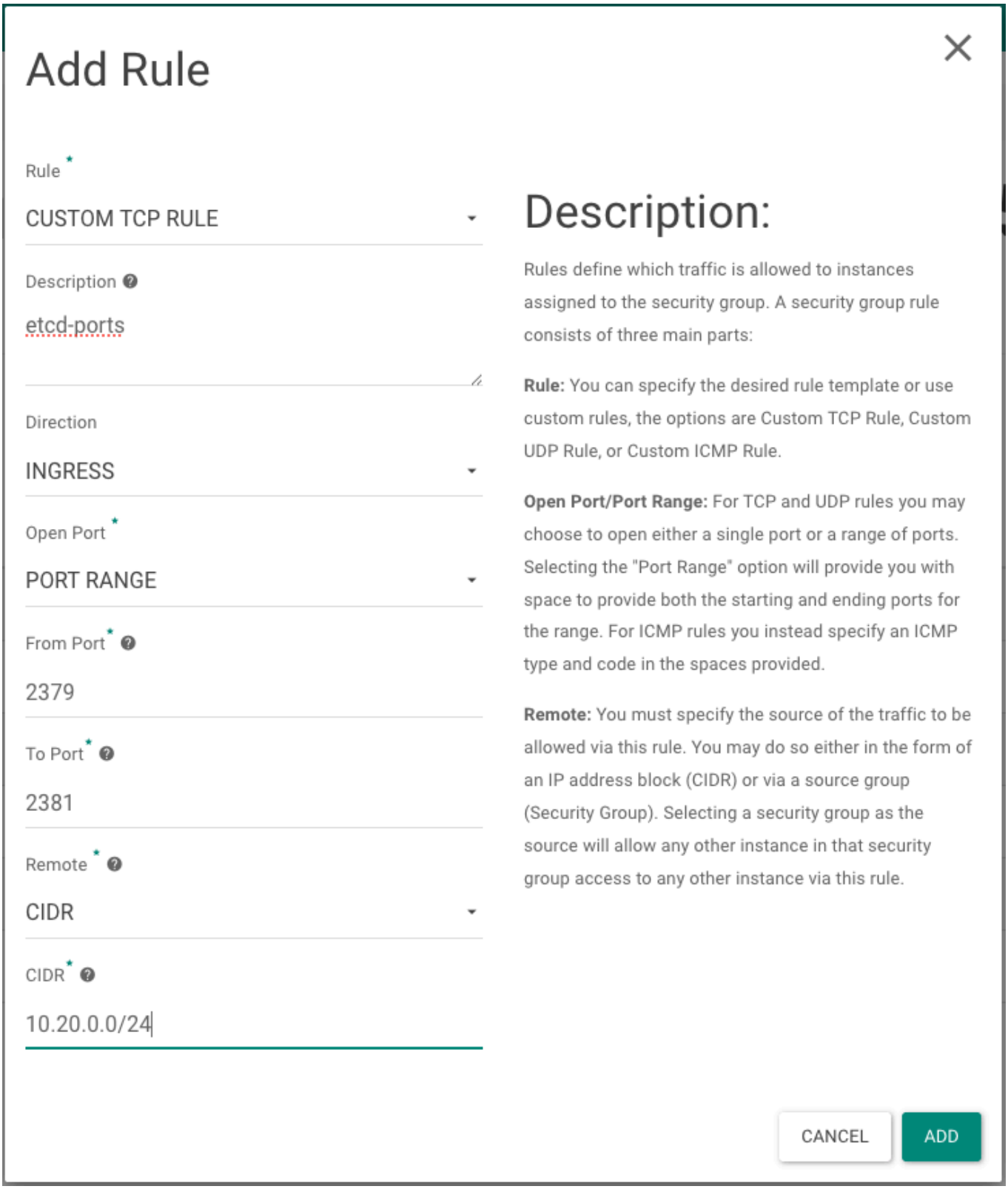


Ingress-nginx ayrıca HTTPS için 443 portunu da açmaktadır bunun için kural kısmından HTTPS seçerek private subnetimizi de ekleyerek devam edelim.

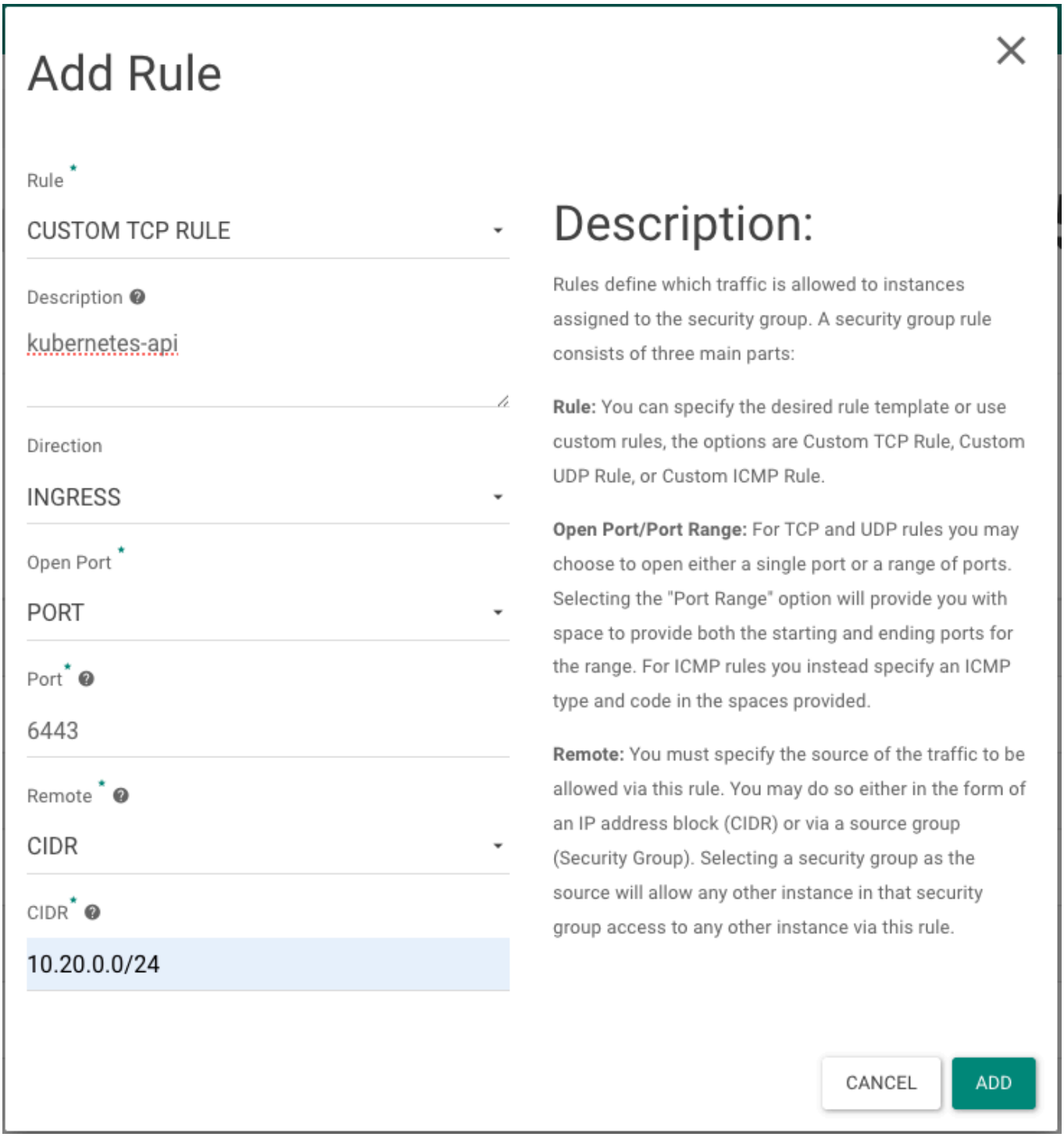


RKE2 cluster yapımızın sağlıklı çalışabilmesi için **etcd** bileşeninin iletişimini sağlamamız gerekiyor:

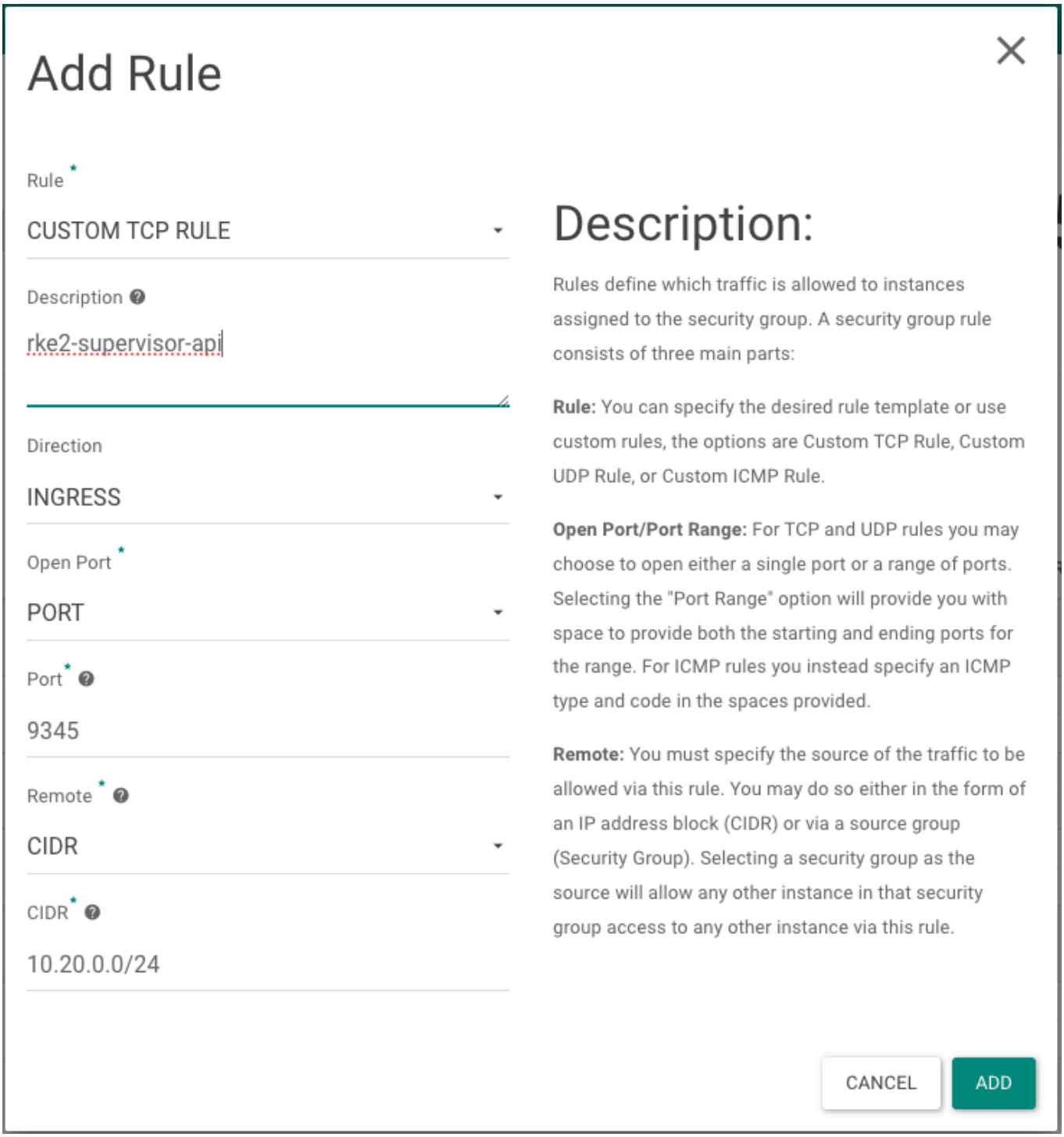
- Client Port (2379): **Kubernetes API Server'ın veri tabanına (etcd) yazabilmesi ve sorgu atabilmesi için bu yolu açık tutuyoruz.**
- Peer Port (2380): **Birden fazla server (master) node kullandığımızda, node'ların birbirlerinden haberdar olması ve veriyi kendi aralarında eşitlemesi (High Availability) için bu portu kullanıyoruz.**
- Health Metrics (2381): **Sistemimizin durumunu izlemek ve metrikleri toplamak için bu port üzerinden haberleşiyoruz.**



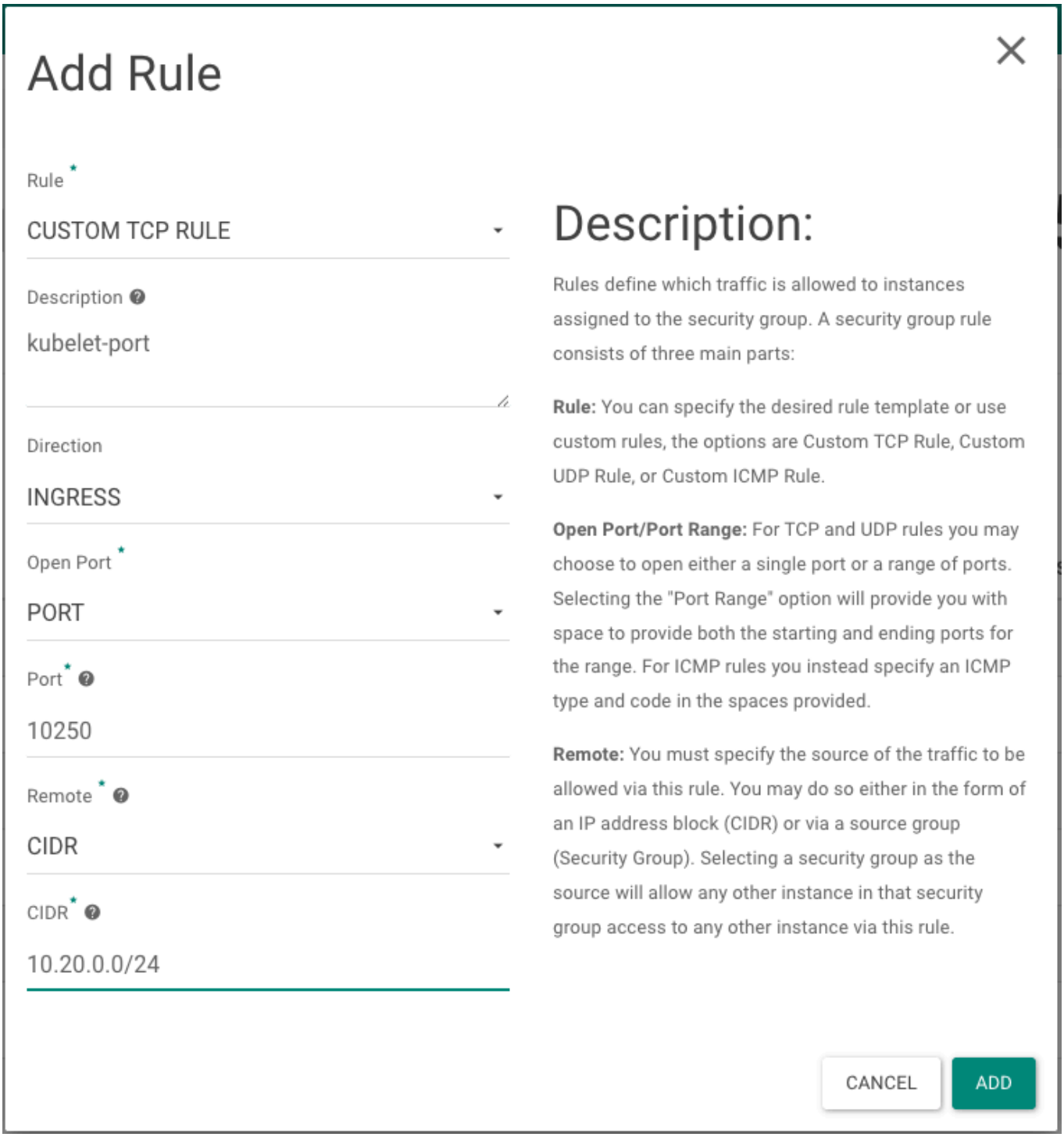
OpenStack üzerinde **6443 TCP** portunu, Kubernetes'in ana yönetim merkezi olan **API Server**'a erişim sağlamak için açıyoruz. Bu sayede hem `kubectl` komutlarımızla cluster'ı dışarıdan yönetebiliyor hem de worker node'larımızın master node'lar ile iletişim kurup cluster'a dahil olmasını sağlıyoruz.



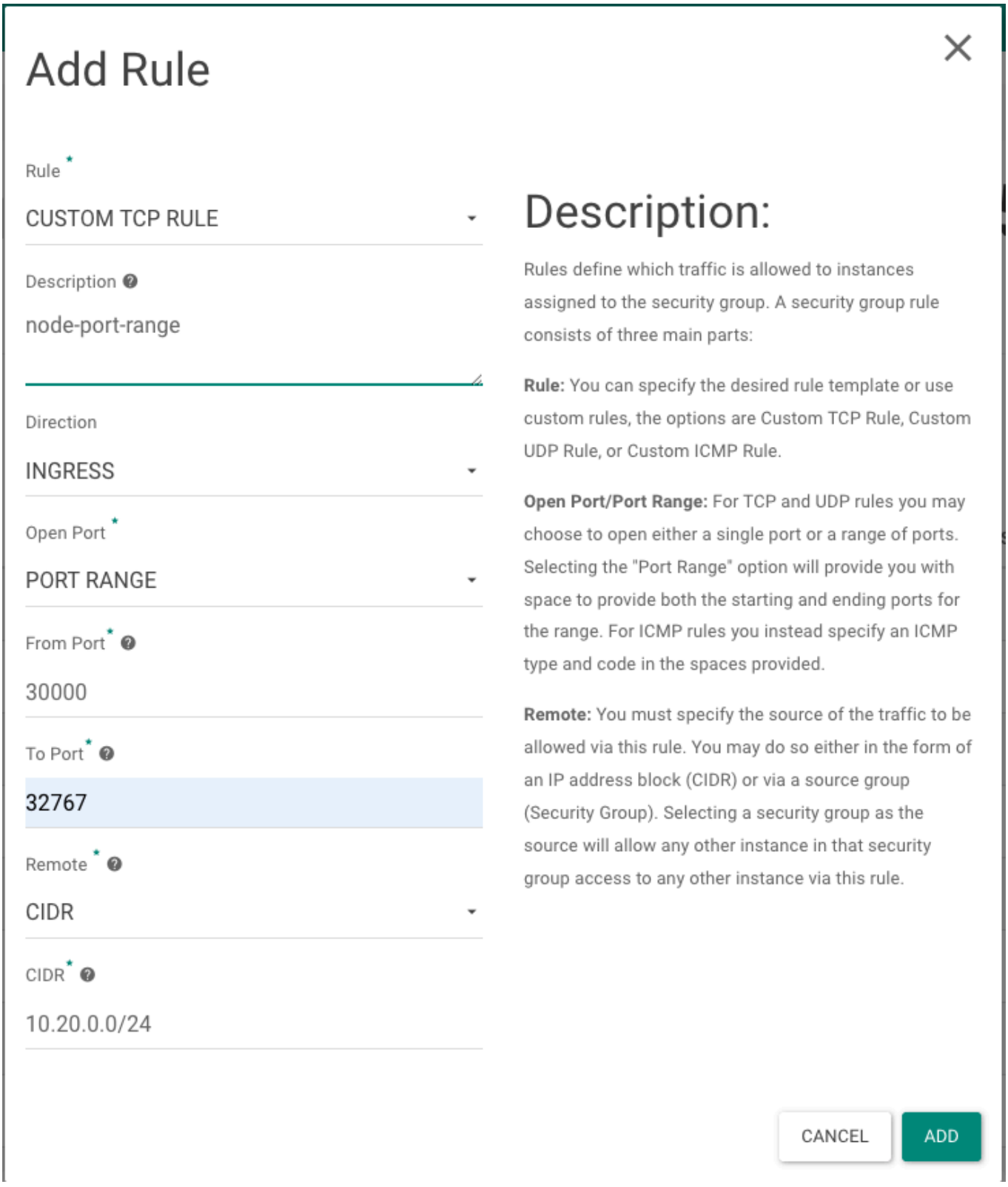
OpenStack üzerinde **9345 TCP** portunu, RKE2'ye özel **Supervisor API** servisi için açıyoruz; bu port sayesinde yeni eklenen node'ların cluster'a güvenli bir şekilde kayıt olmasını (registration) sağlıyor ve node'lar arasındaki TLS sertifikalarının dağıtımı ile güncellenmesini yönetiyoruz.



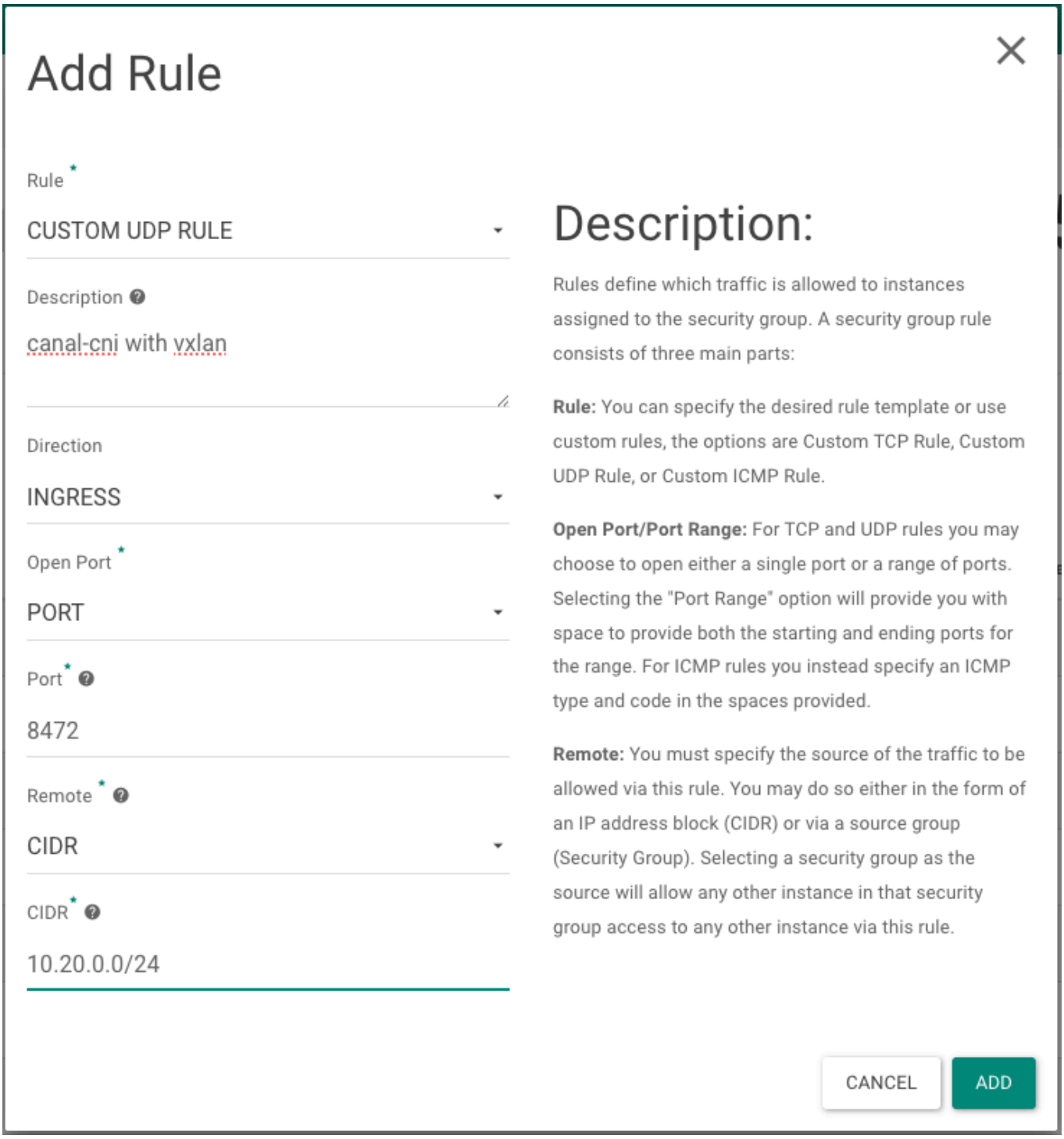
OpenStack üzerinde **10250 TCP** portunu, her node üzerinde çalışan **Kubelet** servisinin güvenli iletişimi için açıyoruz; bu sayede master node üzerindeki API Server'ın worker node'lara bağlanarak pod loglarını çekmesini, komut çalıştırmasını (`kubectl exec`) ve node üzerindeki kaynakları yönetmesini sağlıyoruz.



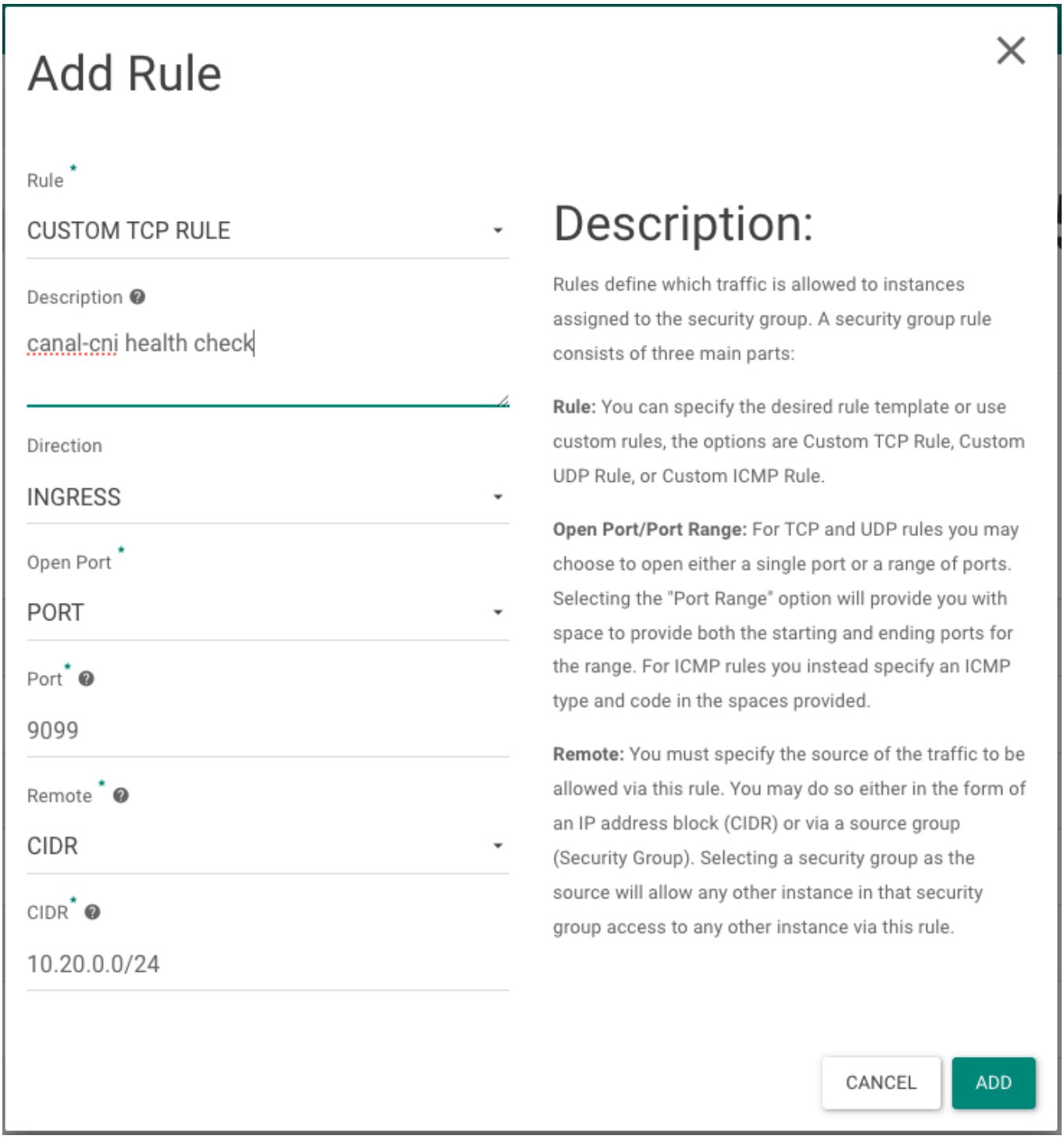
OpenStack üzerinde **30000-32767 TCP/UDP** port aralığını **NodePort** servisleri için açıyoruz; bu sayede Kubernetes üzerinde koşan uygulamalarımızı dış dünyaya açabiliyor ve gelen trafiğin herhangi bir node üzerinden doğrudan ilgili pod'lara yönlendirilmesini sağlıyoruz.



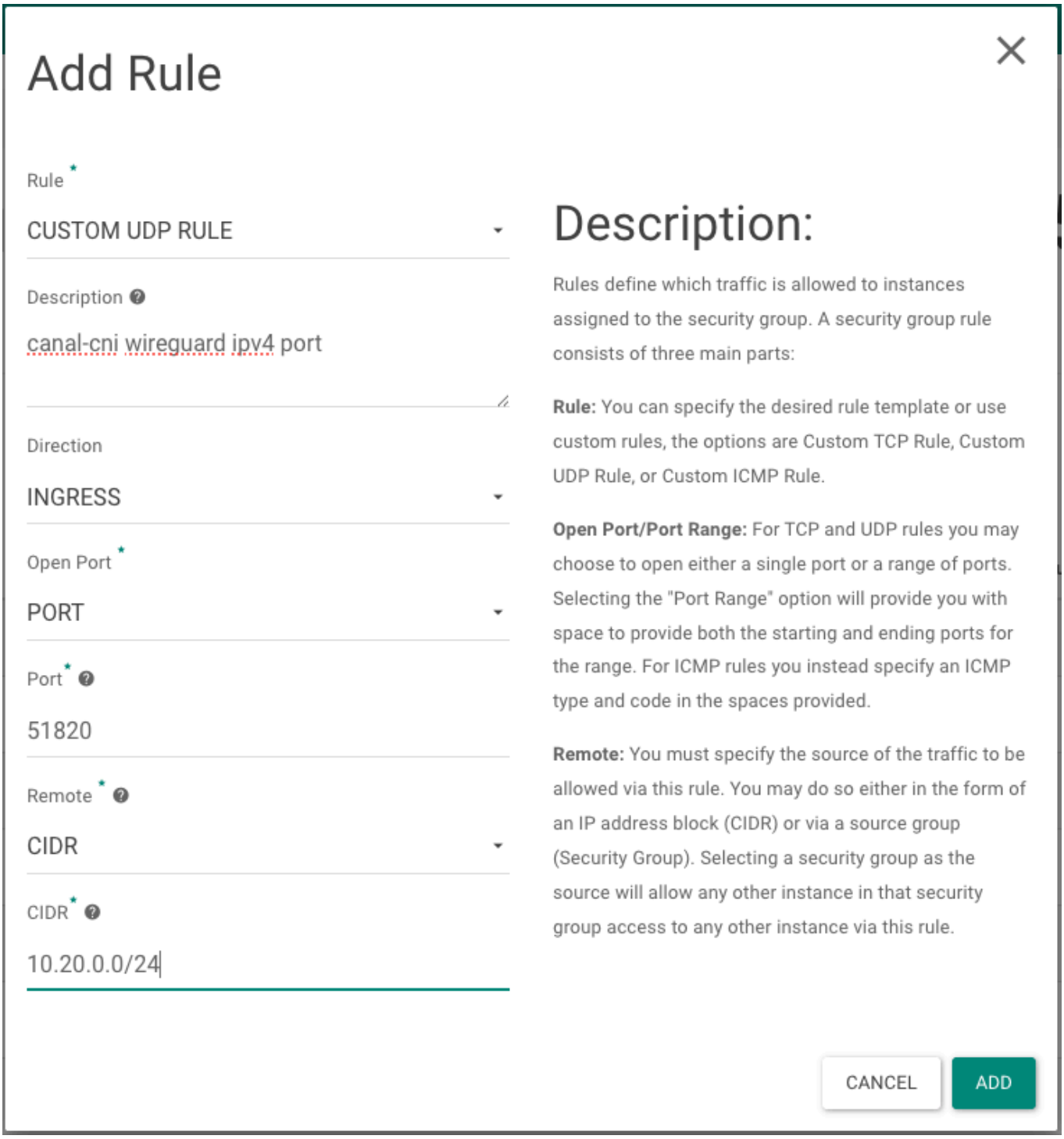
OpenStack üzerinde **8472 UDP** portunu, Kubernetes ağ trafiğini yöneten **Canal CNI** eklentisinin **VXLAN** katmanı için açıyoruz; bu sayede farklı node'lar üzerinde çalışan pod'ların birbirleriyle sanki aynı fiziksel ağdaymış gibi güvenli bir tünel üzerinden haberleşmesini sağlıyoruz.



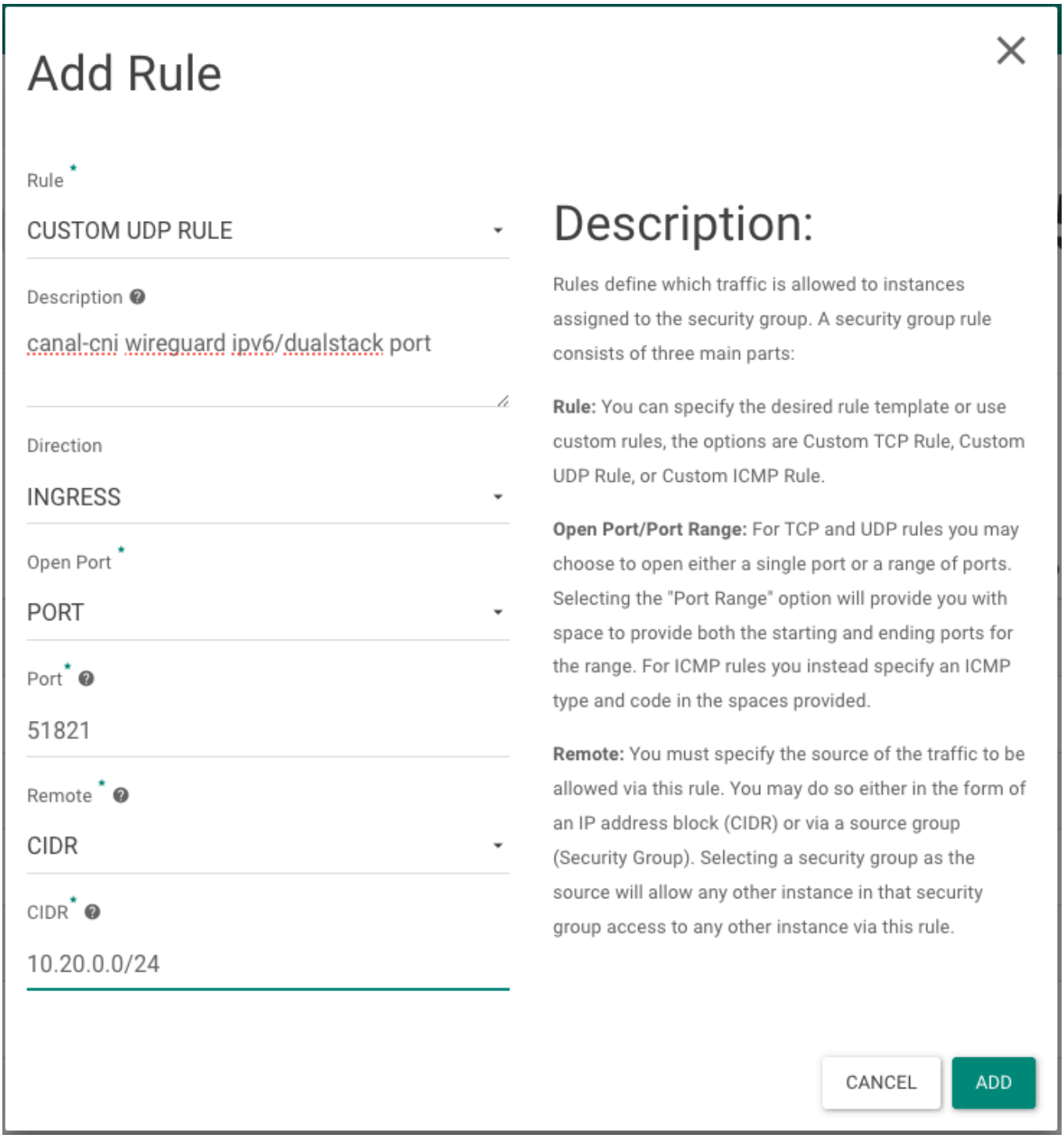
OpenStack üzerinde **9099 TCP** portunu, RKE2 içerisinde ağ trafiğini yöneten **Canal (Calico + Flannel)** bileşeninin sağlık kontrollerini (health checks) yapabilmesi için açıyoruz; bu sayede sistemin her bir node üzerindeki ağ sürücüsünün düzgün çalışıp çalışmadığını denetlemesine ve olası ağ hatalarını raporlamasına olanak tanıyoruz.



OpenStack üzerinde **51820 UDP** portunu, RKE2 içerisinde ağ trafiğini şifrelemek için kullanılan **WireGuard** protokolü için açıyoruz; bu sayede node'lar arasındaki tüm pod trafiğinin (CNI seviyesinde) yüksek performanslı ve güvenli bir tünel üzerinden şifrelenmiş olarak iletilmesini sağlıyoruz.

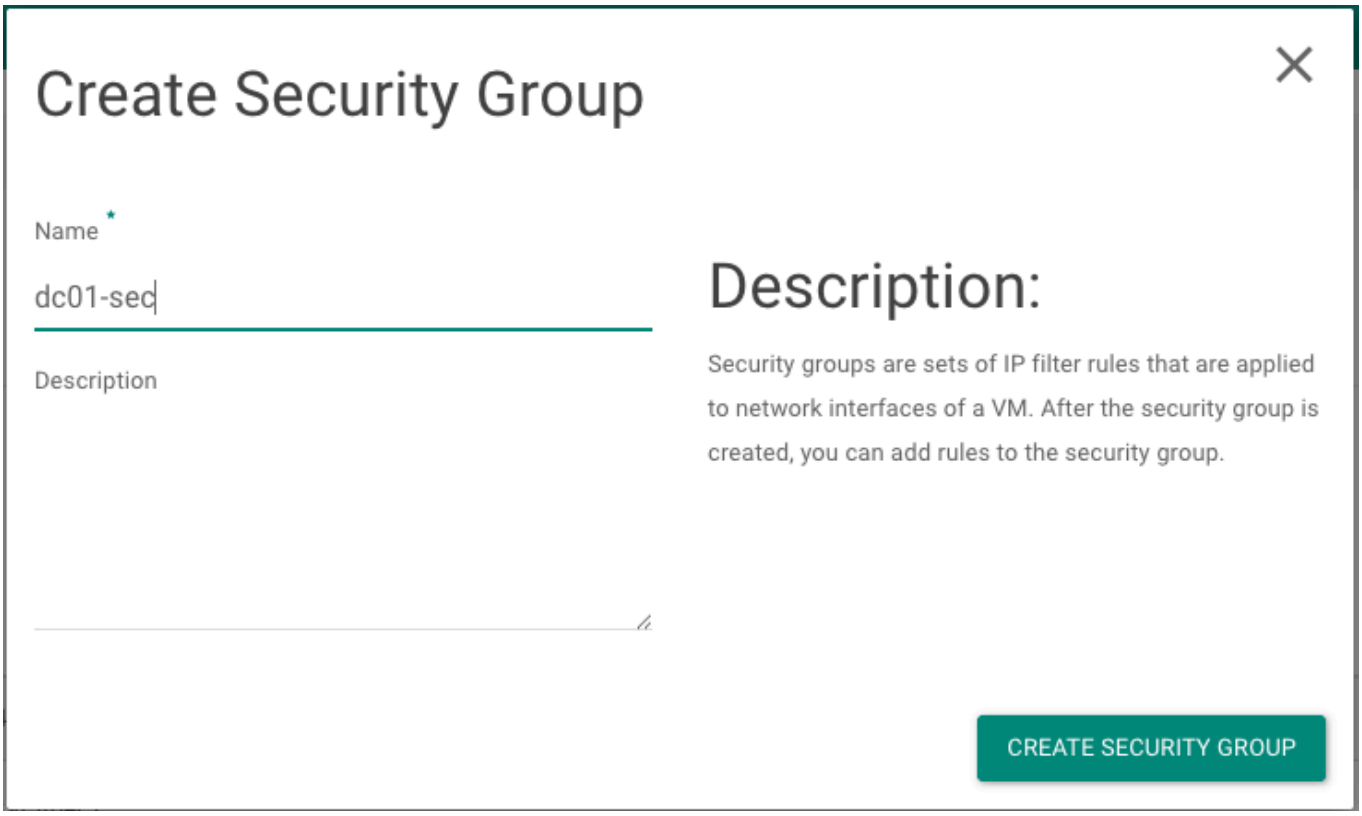


OpenStack üzerinde **51821 UDP** portunu, RKE2'de **WireGuard** protokolü ile ağ şifrelemesi (IPv6 desteği dahil) yaparken kullanılan ikinci veri tüneli için açıyoruz; bu sayede node'lar arasındaki güvenli haberleşmenin kesintisiz devam etmesini ve şifrelenmiş trafiğin alternatif yollar üzerinden de akabilmesini sağlıyoruz.

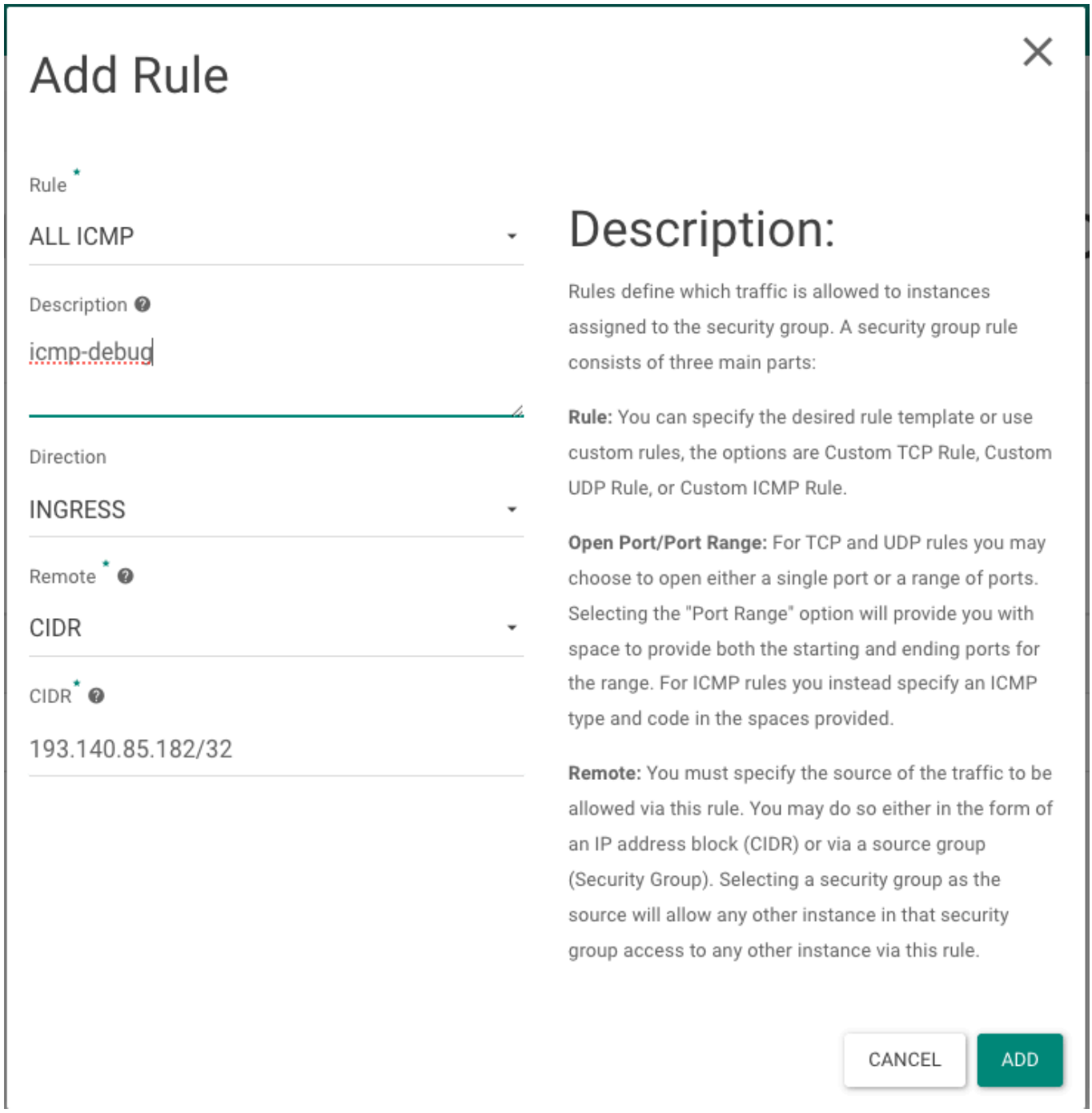


Security Group ile LoadBalancer Konfigürasyonu

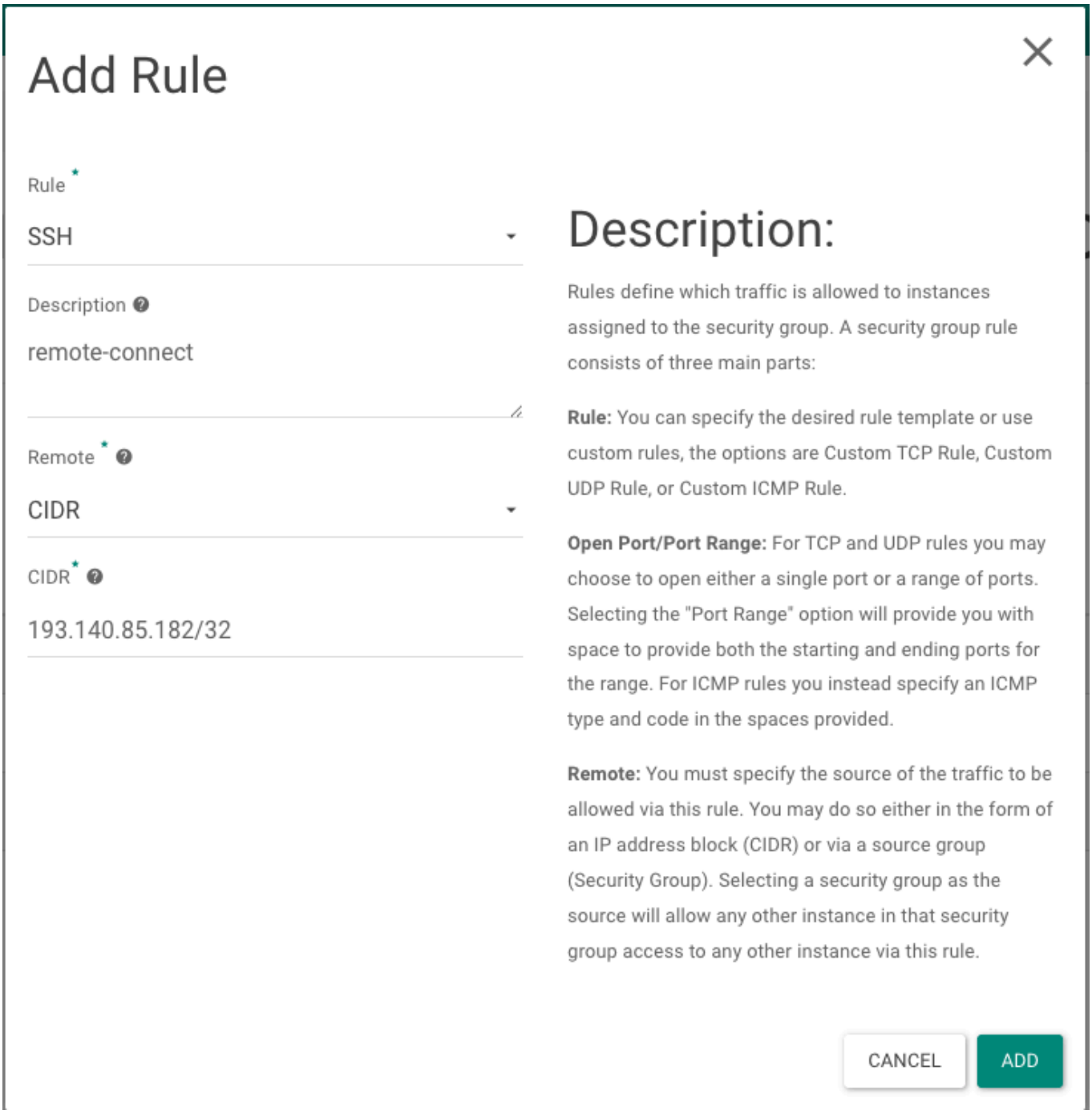
Dışardan clusterimize bir geçiş kapısı ve HA fonksiyonalitesini sağlayabilmek için aşağıdaki şekilde security group yapılandırılmasına ihtiyacımız vardır. **dc01-sec** diyerek security group oluşturmaya başlayabiliriz.



Güvenli olabilmesi adına https://icanhazip.com/ adresinden public ip adresimizi kontrol edip ICMP paketlerine olan izni açabiliriz. public ip miz ve 32 subnet mask vererek devam edelim.



Kendi public ip miz ile loadbalancer sanal makinemize erişmek için SSH iznini de tanımlayalım.



HTTP yönlendirilmesi bizim web üzerinden kubernetes cluster'a yaptığımız deploy'ları inceleyebilmek ve kullanabilmek adına önemlidir. Bunun için sırasıyla HTTP ve HTTPS kurallarını kendi public ip mizi belirterek ekleyelim.

Add Rule

Rule *

HTTP

Description

nginx-access

Remote *

CIDR

CIDR *

193.140.85.182/32

Description:

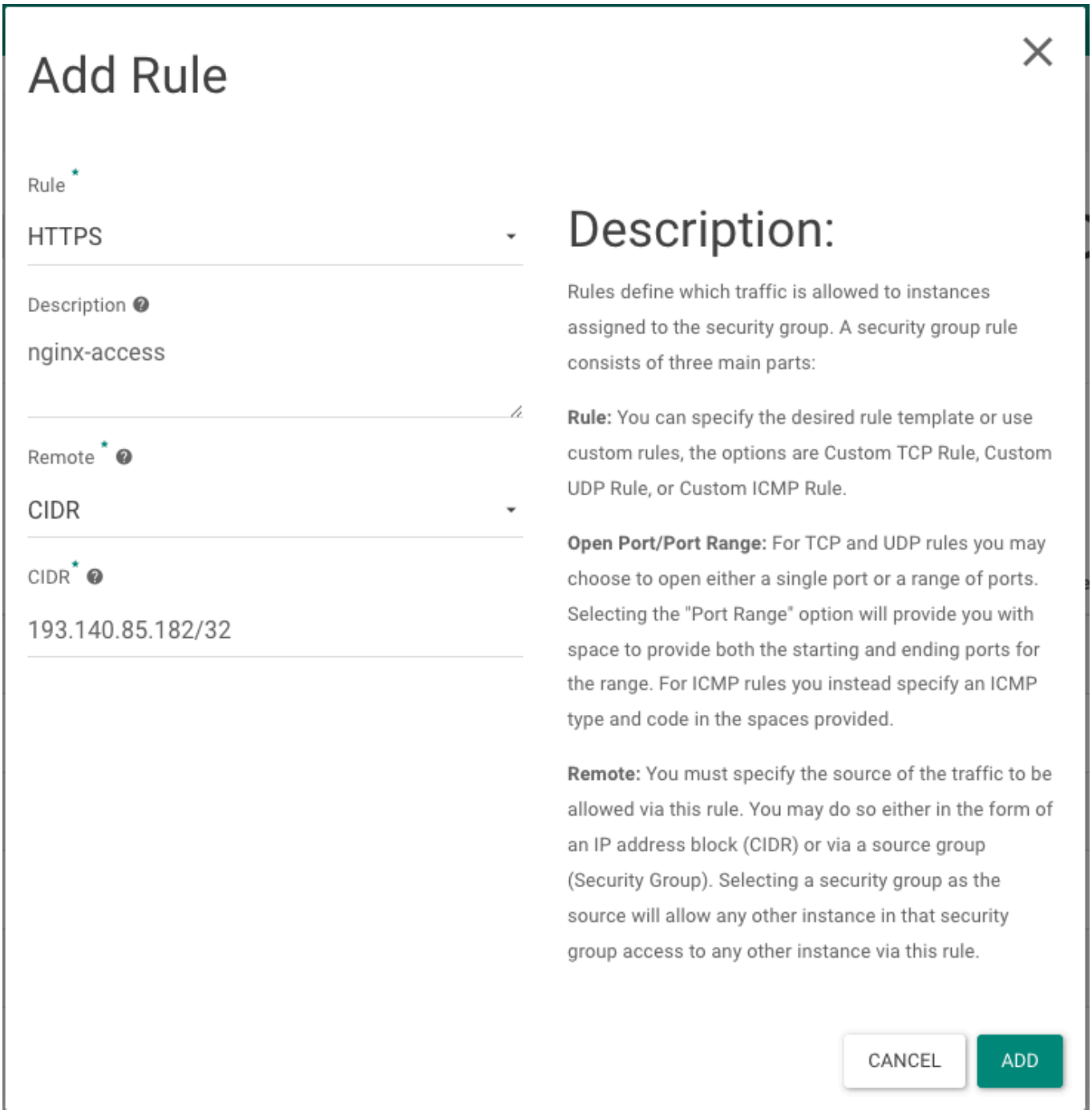
Rules define which traffic is allowed to instances assigned to the security group. A security group rule consists of three main parts:

Rule: You can specify the desired rule template or use custom rules, the options are Custom TCP Rule, Custom UDP Rule, or Custom ICMP Rule.

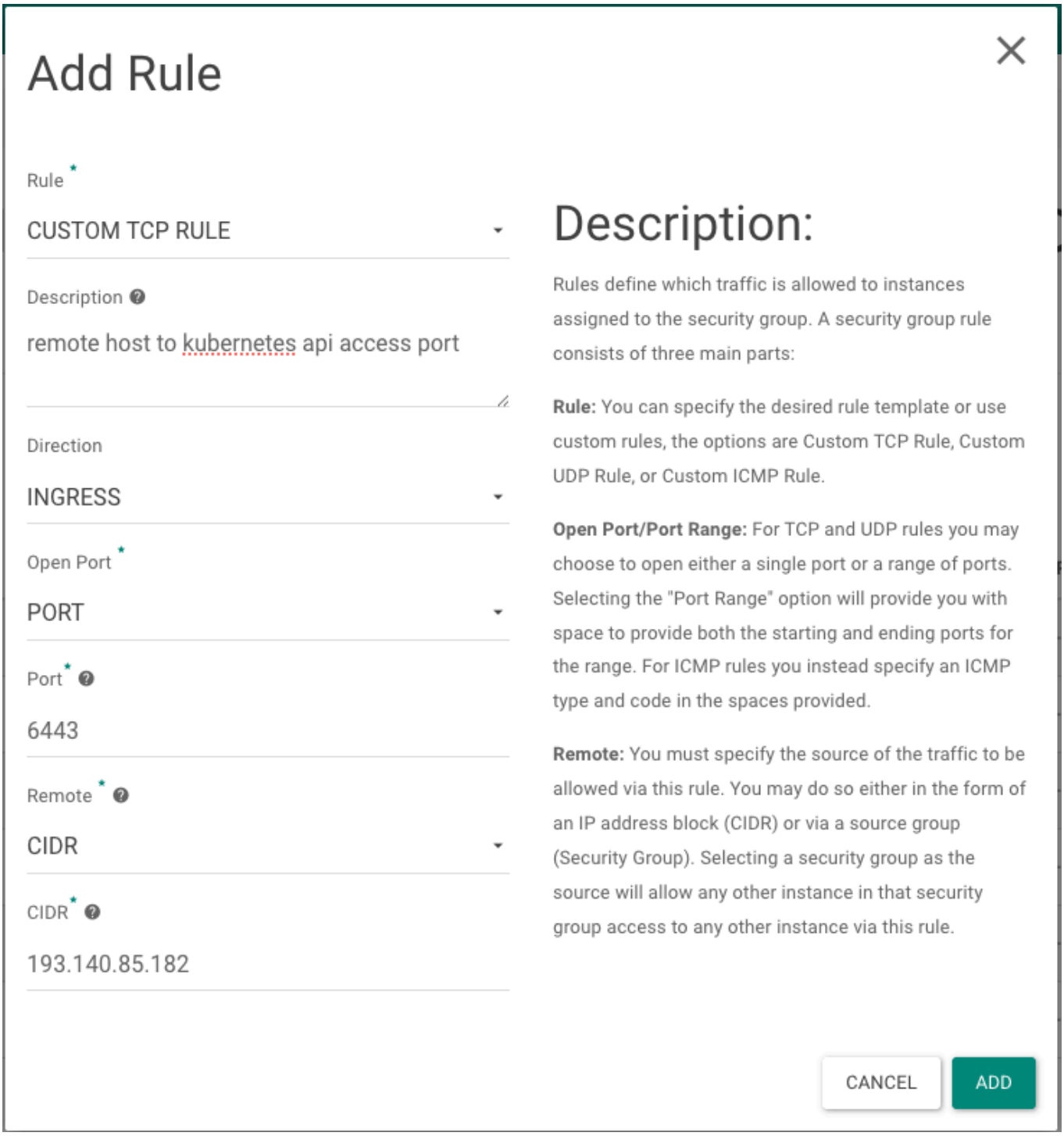
Open Port/Port Range: For TCP and UDP rules you may choose to open either a single port or a range of ports. Selecting the "Port Range" option will provide you with space to provide both the starting and ending ports for the range. For ICMP rules you instead specify an ICMP type and code in the spaces provided.

Remote: You must specify the source of the traffic to be allowed via this rule. You may do so either in the form of an IP address block (CIDR) or via a source group (Security Group). Selecting a security group as the source will allow any other instance in that security group access to any other instance via this rule.

CANCEL ADD



İleride Cluster yönetim için bir Lens veya Portainer gibi bir tool kullanacaksak onun içinde kubernetes api portuna da izin verelim.



LoadBalancer Sanal Makinesinin Oluşturulması

LoadBalancer sanal makinesini oluşturma aşamasına adını kullanacağımız kubernetes node'larıyla bağlantılı olduğunu belirtmek adın dc01-bastion-lb diyerek kuruluma devam edebiliriz.

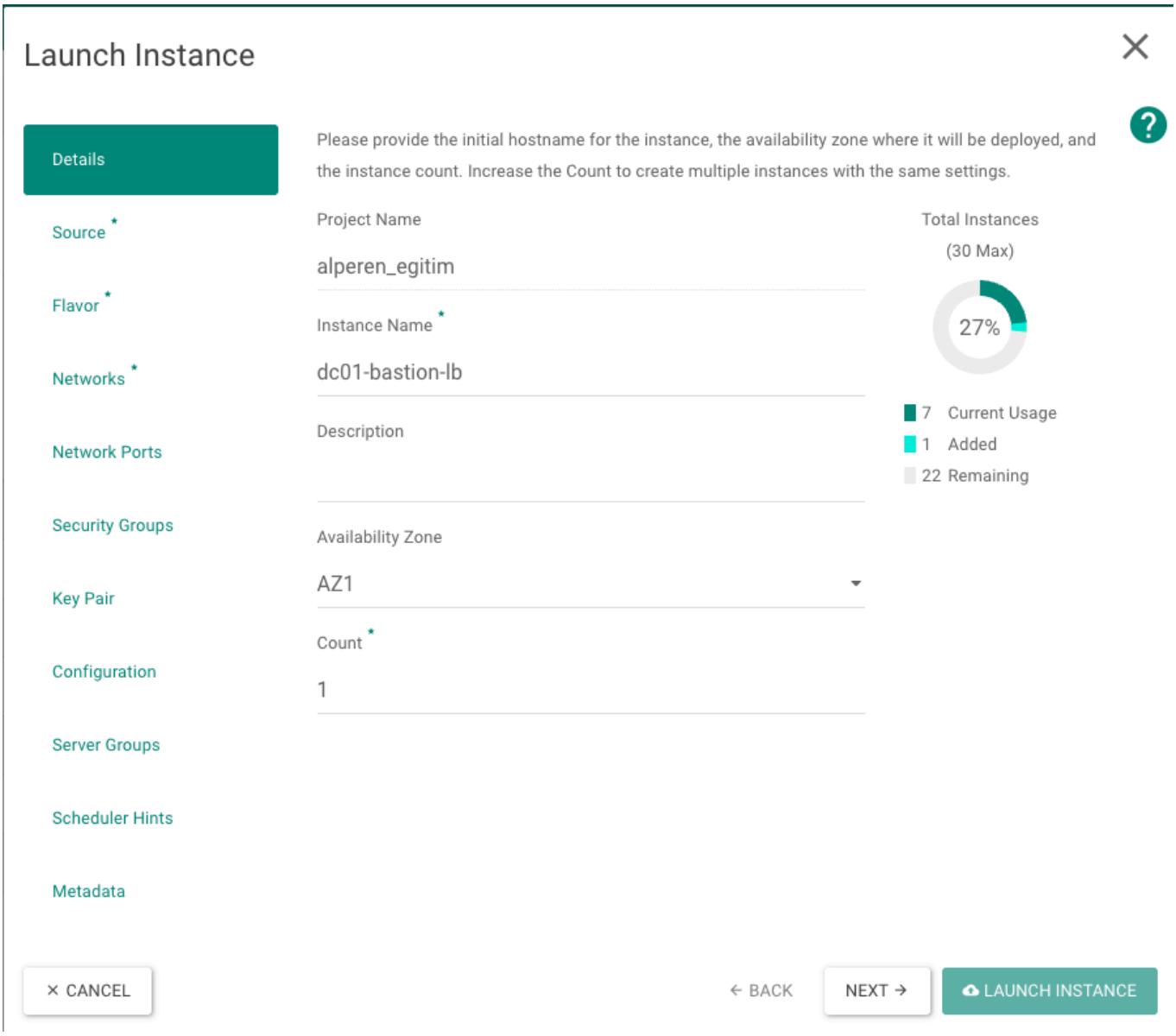
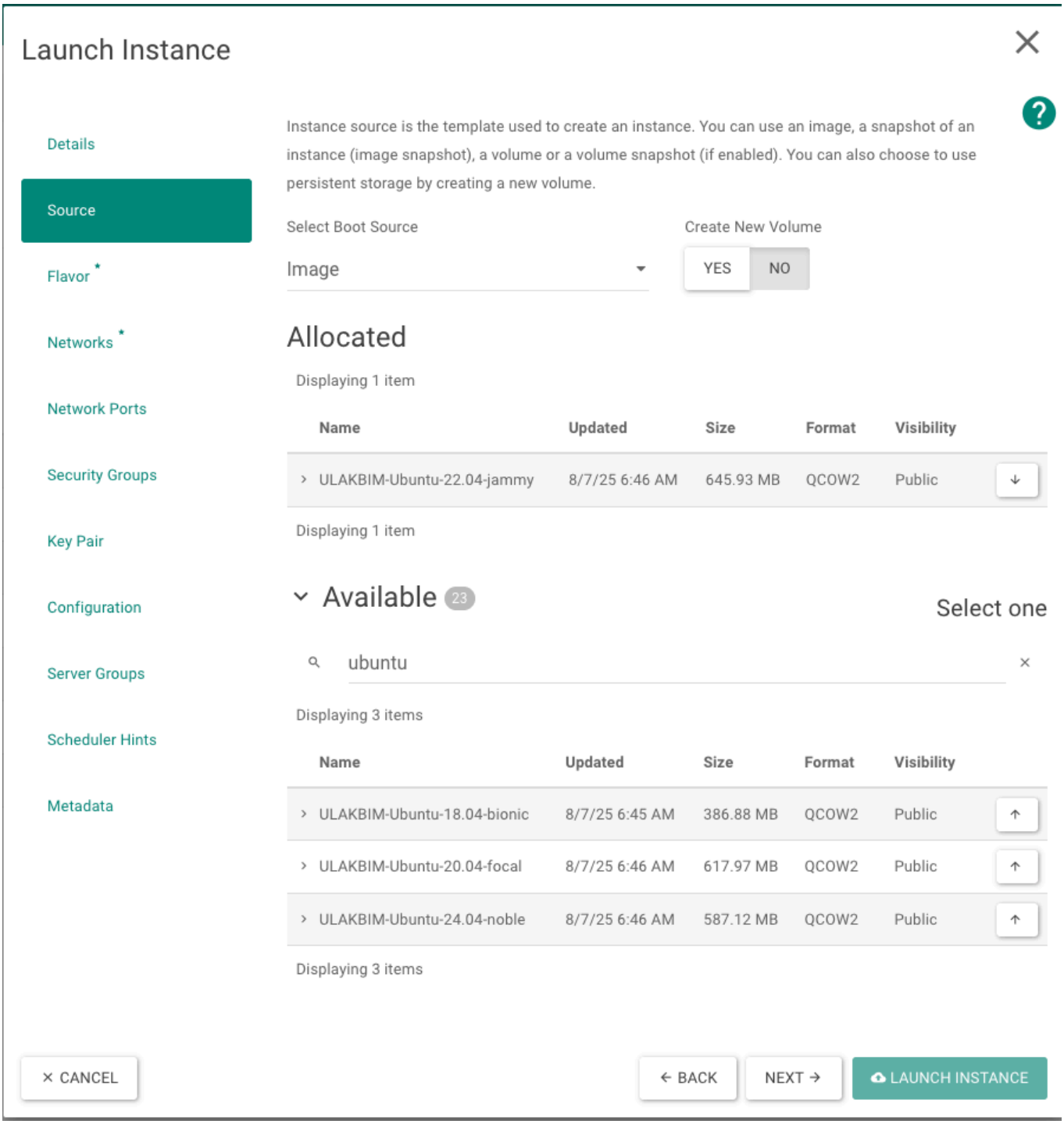
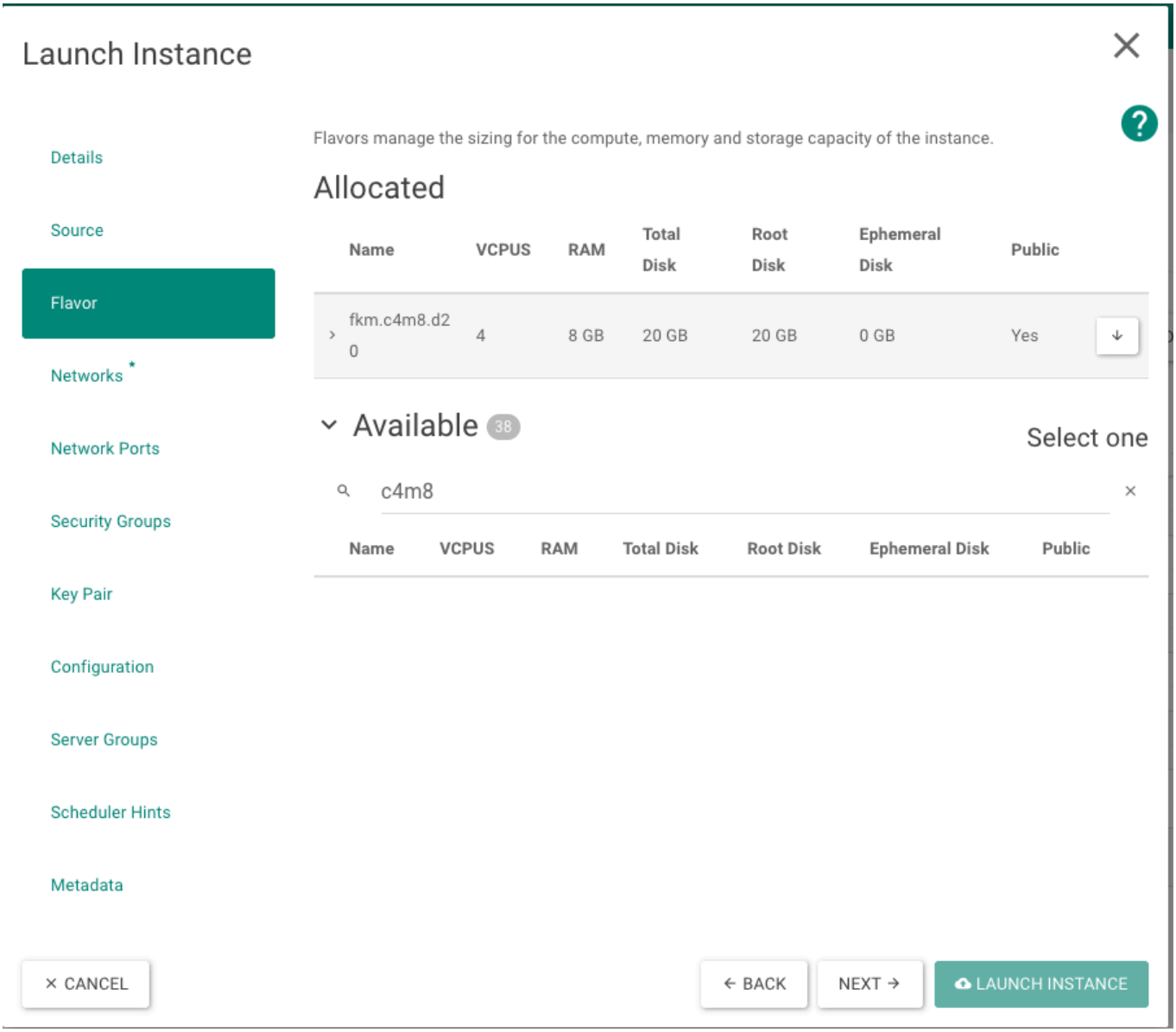


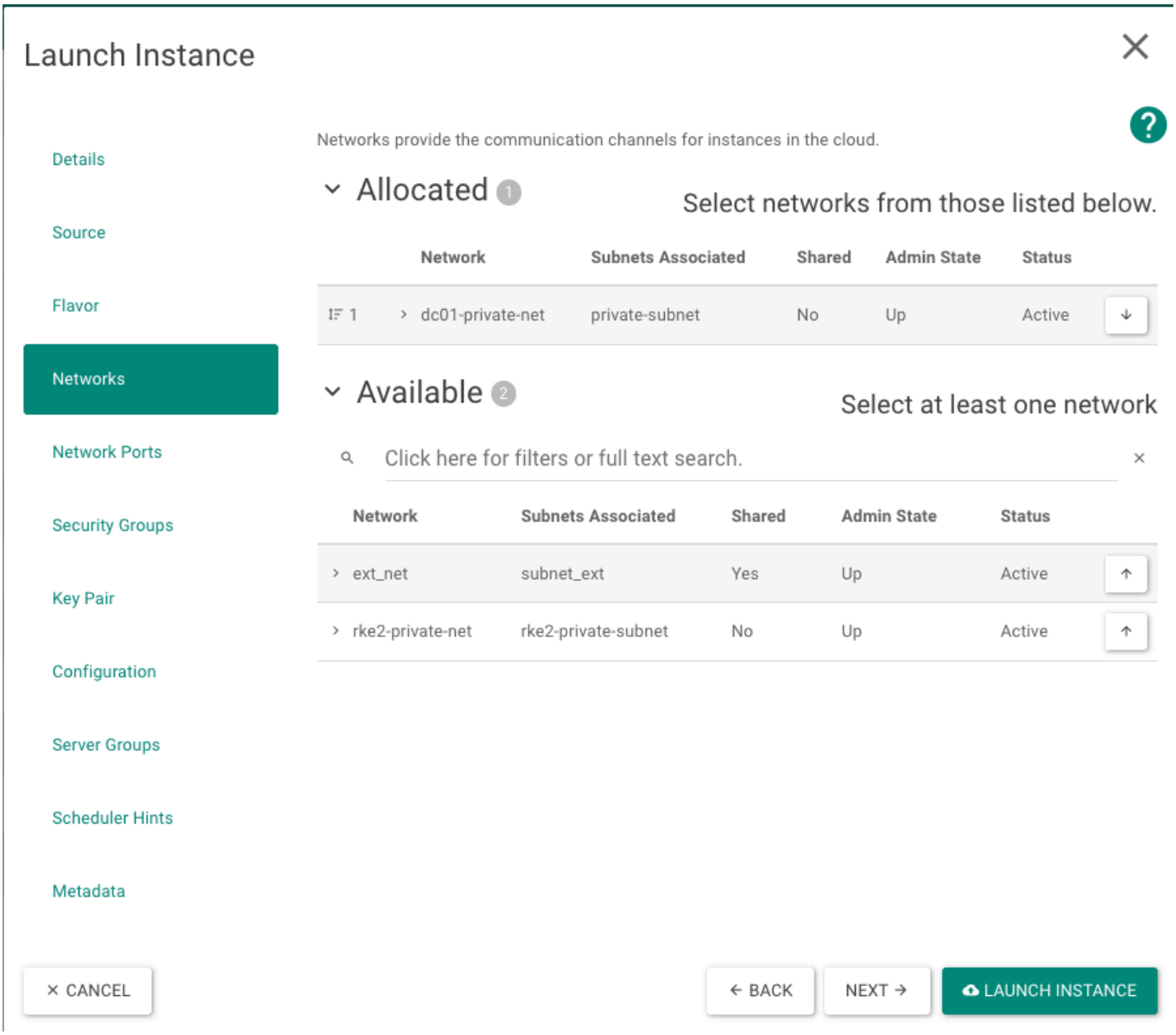
Image seçiminde ise Ubuntu 22-04 üzerinden ilerleyeceğiz. Hem load balancer hem de kubernetes node'ları bu şekilde oluşturulacak.



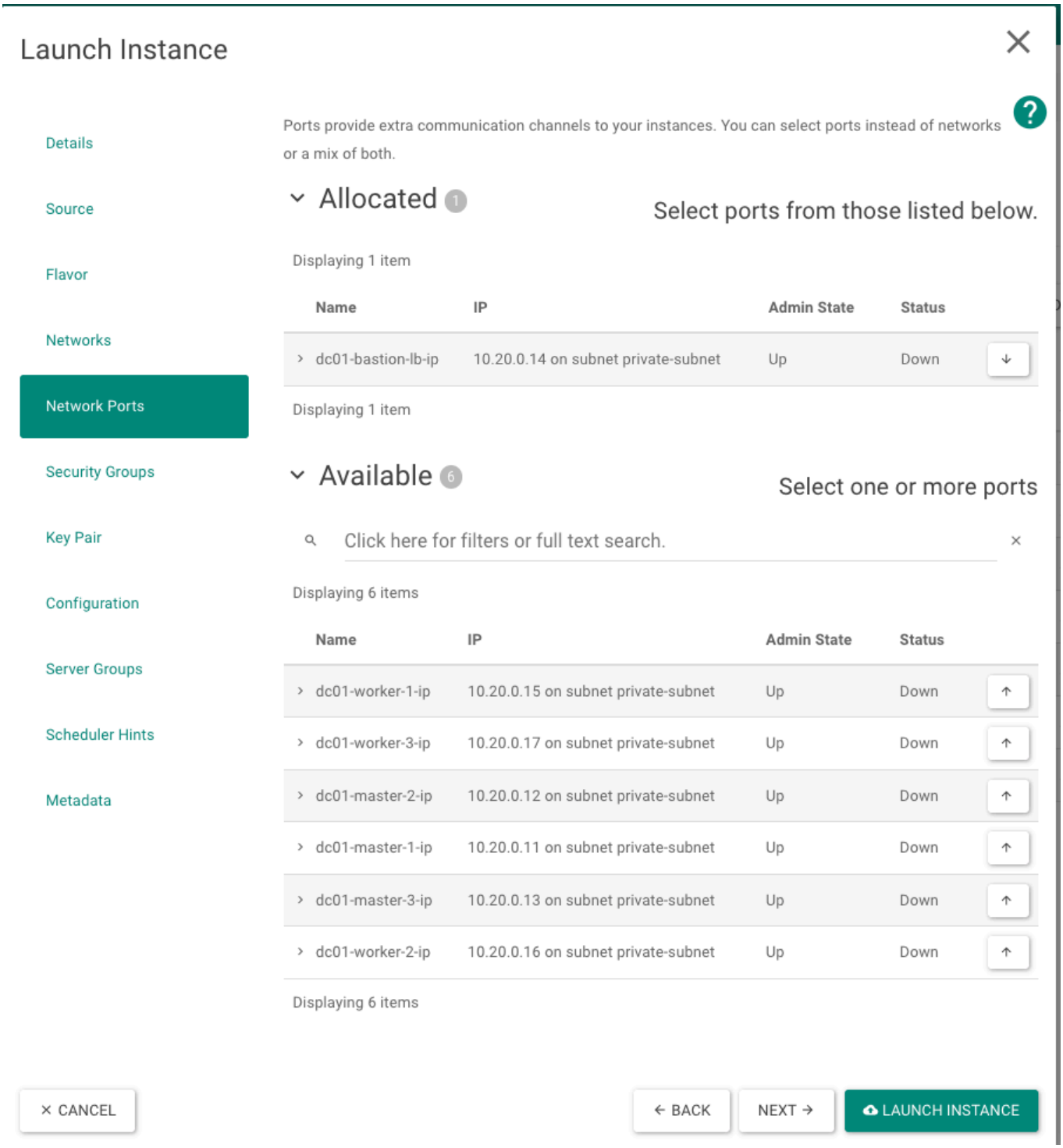
Load balancer sanal makinemiz için 4 CPU 8 RAM yeterli olacaktır. **fkm.c4m8.d20** seçerek kuruluma devam edelim.



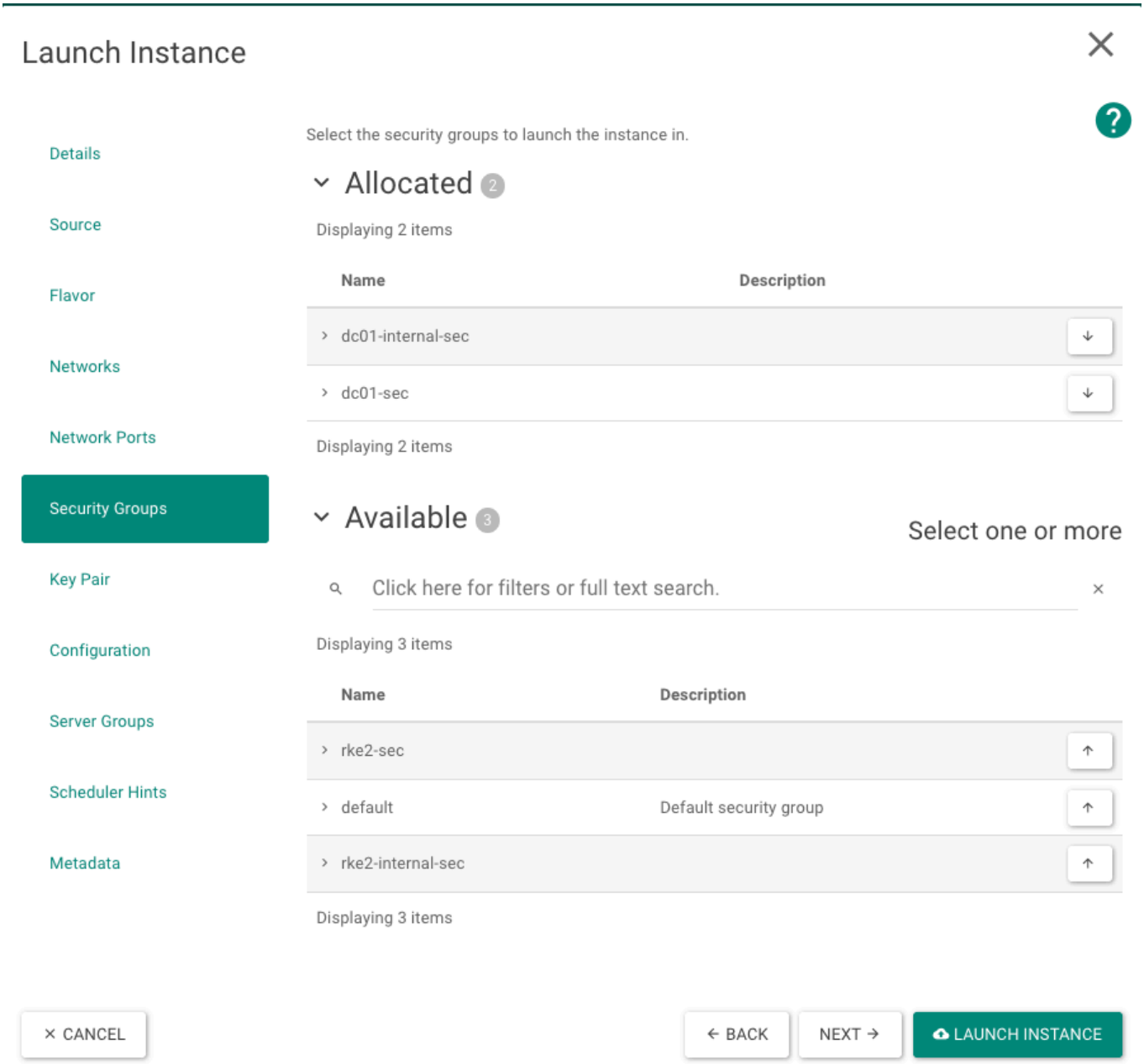
Oluşturmuş olduğumuz dc01-private-net'i burada seçerek buradan ip almasını ve yönlendirilmesini sağlayalım.



Burada önceden oluşturulmuş olan ip'ler üzerinden gidebiliriz fakat biz DHCP Enable yaptığımız için Allocated kısmını boş bırakarak devam edelim.



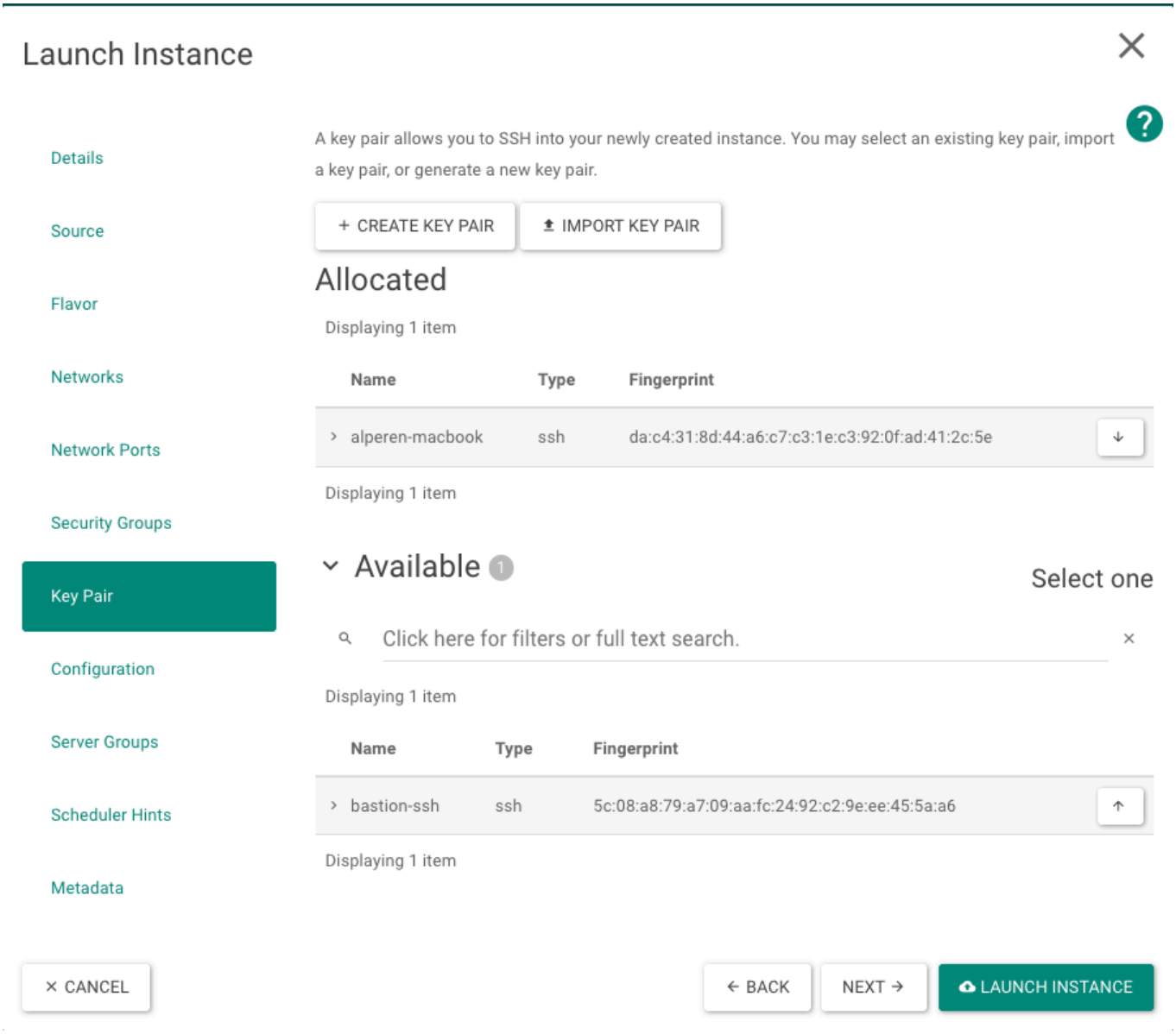
Load balancer hem dış taraftan erişeceğimiz ve kubernetes cluster'imiz arasındaki iletişime sahip olacağı için hem **dc01-sec** hem de **dc01-internal-sec** security group'larını atayalım.



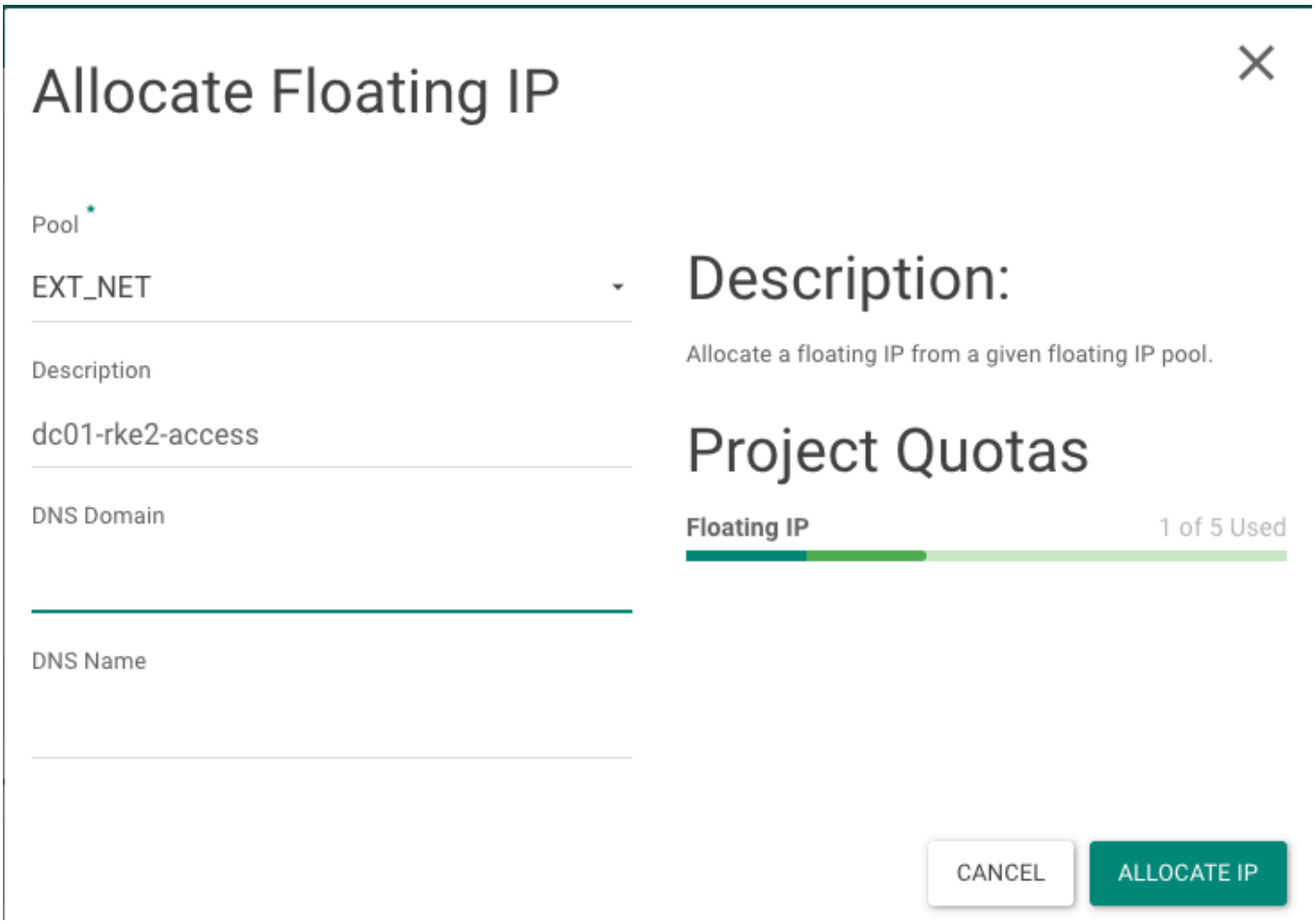
Burada ise eğer cluster'de kullanmak üzere kendi lokal bilgisayarımızdan oluşturduğumuz bir key-pair varsa onu seçerek devam edelim. Eğer yoksa kısa bir şekilde oluşturmak için. Bir terminal açıp şu komutları yazınız:

```
ssh-keygen -t rsa -b 2048
# Onaylamalarla devam edelim ardından
cat ~/.ssh/id_rsa.pub
```

Komutun çıktısını Import Key Pair tarafını seçerek key type SSH Key olacak şekilde public key kısmına terminal çıktımızda olan public key'i yapıştırarak oluşturabilirsiniz.



Launch Instance diyerek load balancer sanal makinemizi oluşturalım. Ardından sanal makinelerin olduğu menüden load balancer sanal makinemizin yanındaki actions kısmından Associate Floating IP verelim.



Kubernetes Cluster Node'lerinin Oluşturulması

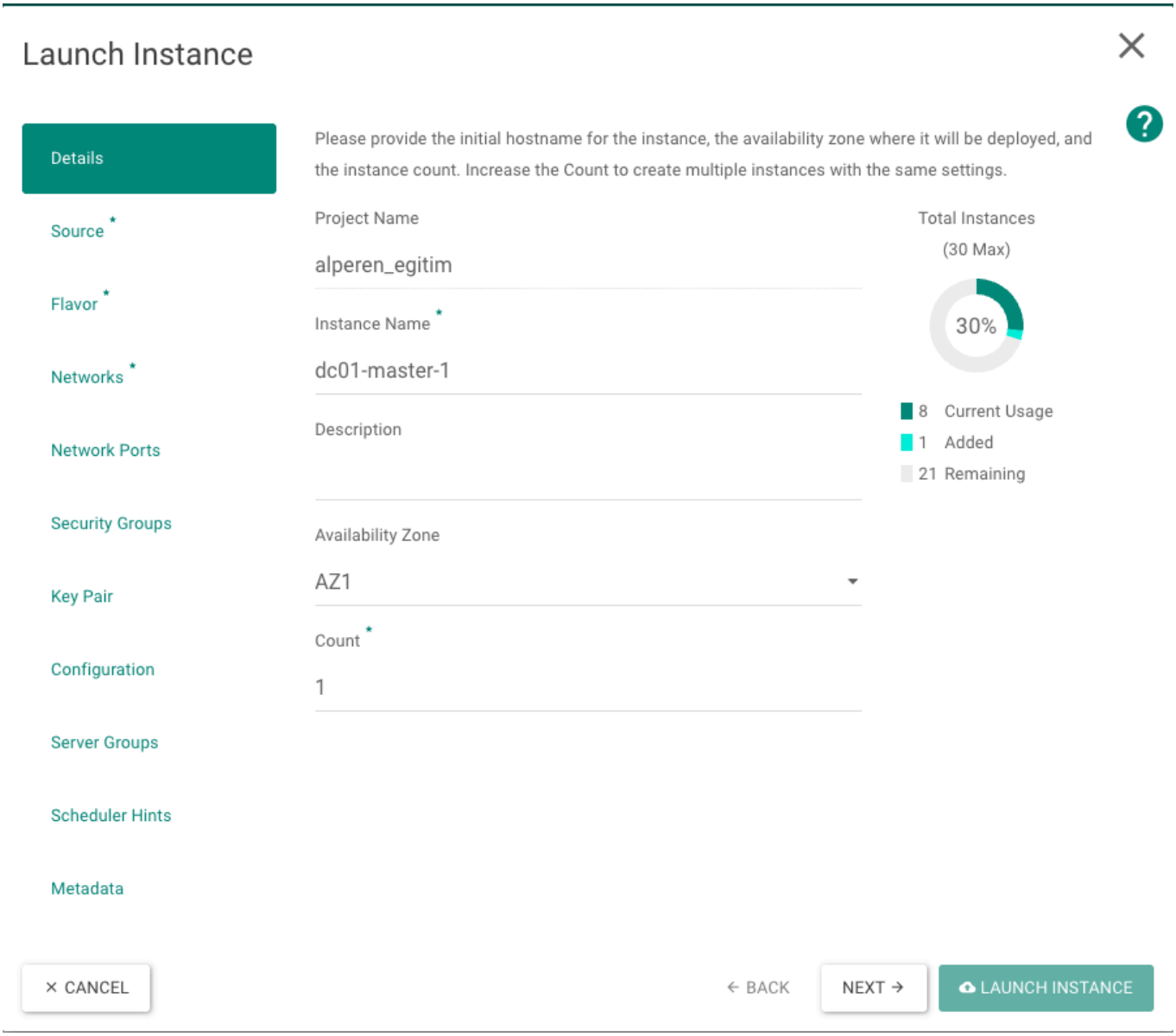
Cluster'ın yüksek erişilebilirliğini (High Availability) sağlamak amacıyla, node'larımızı OpenStack üzerindeki farklı Availability Zone'lara (AZ) dağıtarak oluşturuyoruz. Bu sayede bir bölgede (AZ) oluşabilecek donanım veya enerji kesintisinden tüm cluster'ın etkilenmesini engelliyoruz.

- **Master (Control Plane) Node'lar:** Toplam 3 adet master node; her biri sırasıyla **AZ1, AZ2 ve AZ3** bölgelerinde yaratılacak. Bu yapı, etcd veri tutarlılığı ve yönetim katmanının kesintisizliği için kritiktir.
- **Worker Node'lar:** Toplam 3 adet worker node; iş yüklerinin dengeli dağılması için yine sırasıyla **AZ1, AZ2 ve AZ3** bölgelerine dağıtılacak.

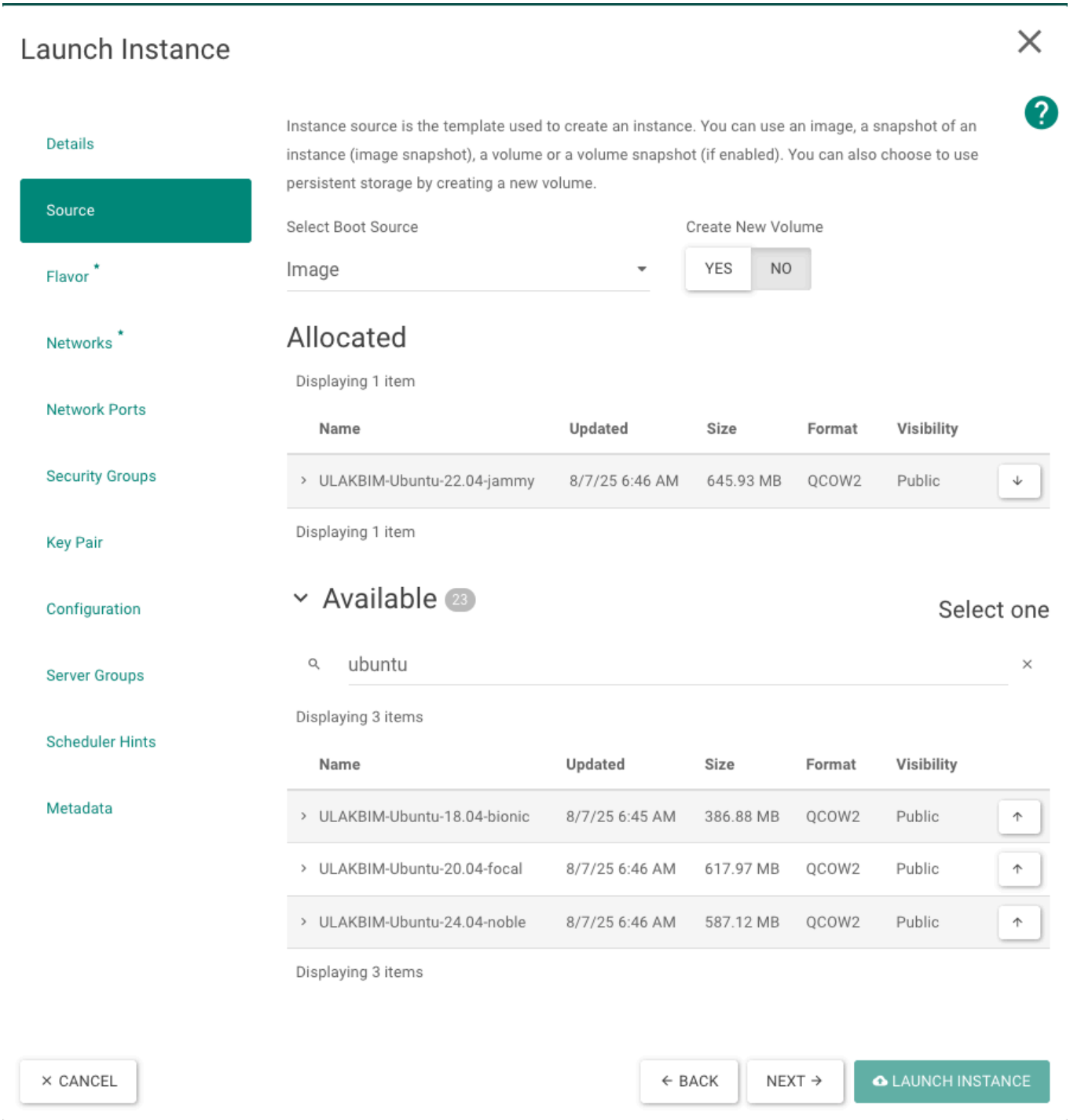
Aşağıda belirtilen tüm adımlar her bir node için tekrarlanacaktır.

Sanal Makine Adı	Rol	Availability Zone
dc01-master-1	Control Plane / etcd	AZ1
dc01-master-2	Control Plane / etcd	AZ2
dc01-master-3	Control Plane / etcd	AZ3
dc01-worker-1	Worker	AZ1

Sanal Makine Adı	Rol	Availability Zone
dc01-worker-2	Worker	AZ2
dc01-worker-3	Worker	AZ3



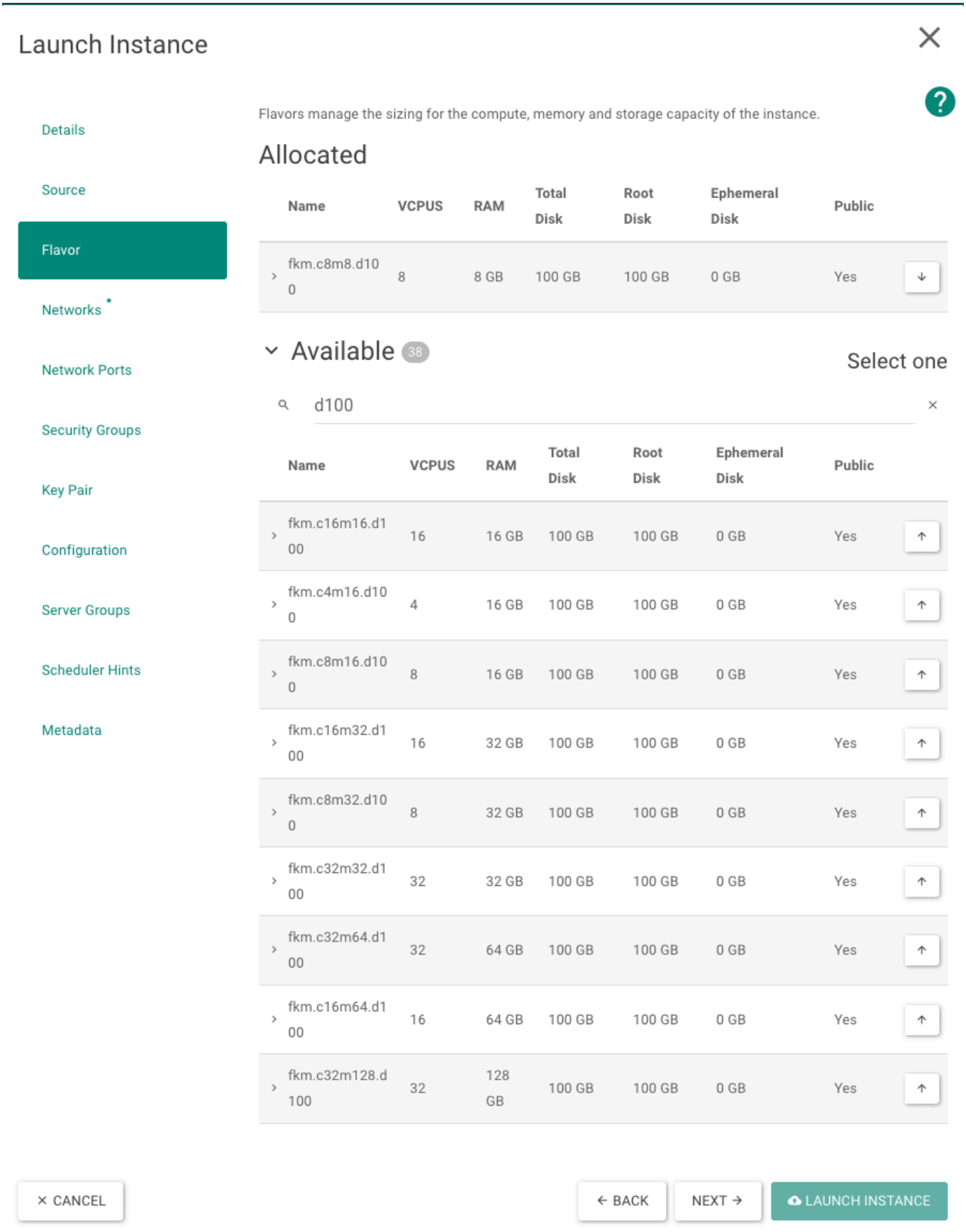
Sanal makine imajımız için de Ubuntu 22.04 üzerinden seçerek devam edebiliriz.



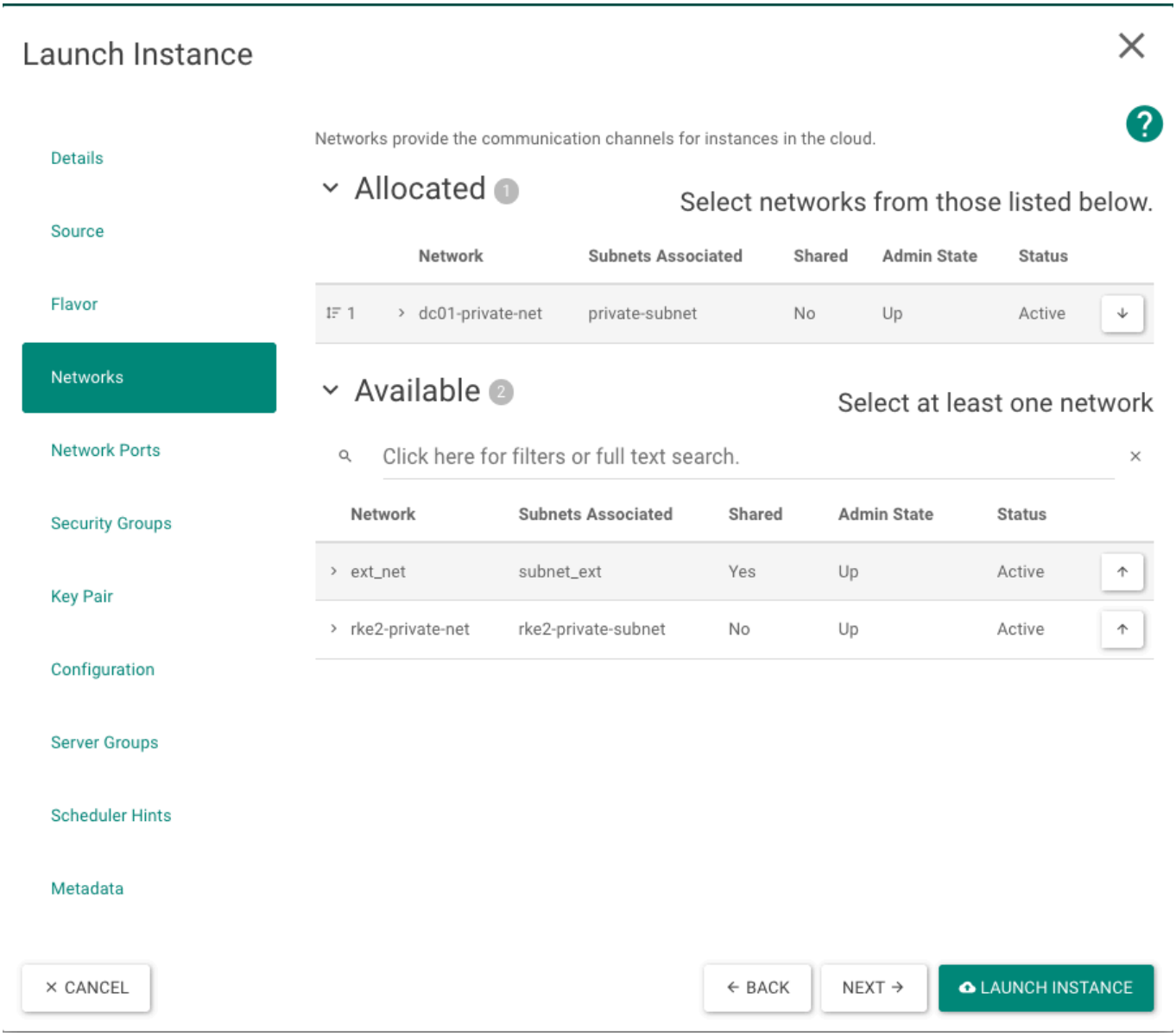
Cluster Node'larının RKE2'nin sistem gereksinimleri seviyesinde vermekte yeterli olabilir. Flavor olarak 8 CPU 8 RAM 100GB Root Disk olarak seçerek ilerlememizdeki sebepler şunlardır:

- **etcd Kararlılığı:** RKE2'nin kalbi olan etcd veri tabanının gecikme yaşamadan (latency) çalışması için yüksek işlemci gücü şart; 8 VCPU seçerek yönetim katmanında oluşabilecek darboğazların önüne geçiyoruz.
- **Servislerin Sürekliliği:** 8 GB RAM tercihiyle, hem işletim sisteminin hem de RKE2 bileşenlerinin (API Server, Kubelet vb.) aynı anda kararlı çalışmasını sağlıyor, bellek yetersizliği kaynaklı çökmeleri engelliyoruz.

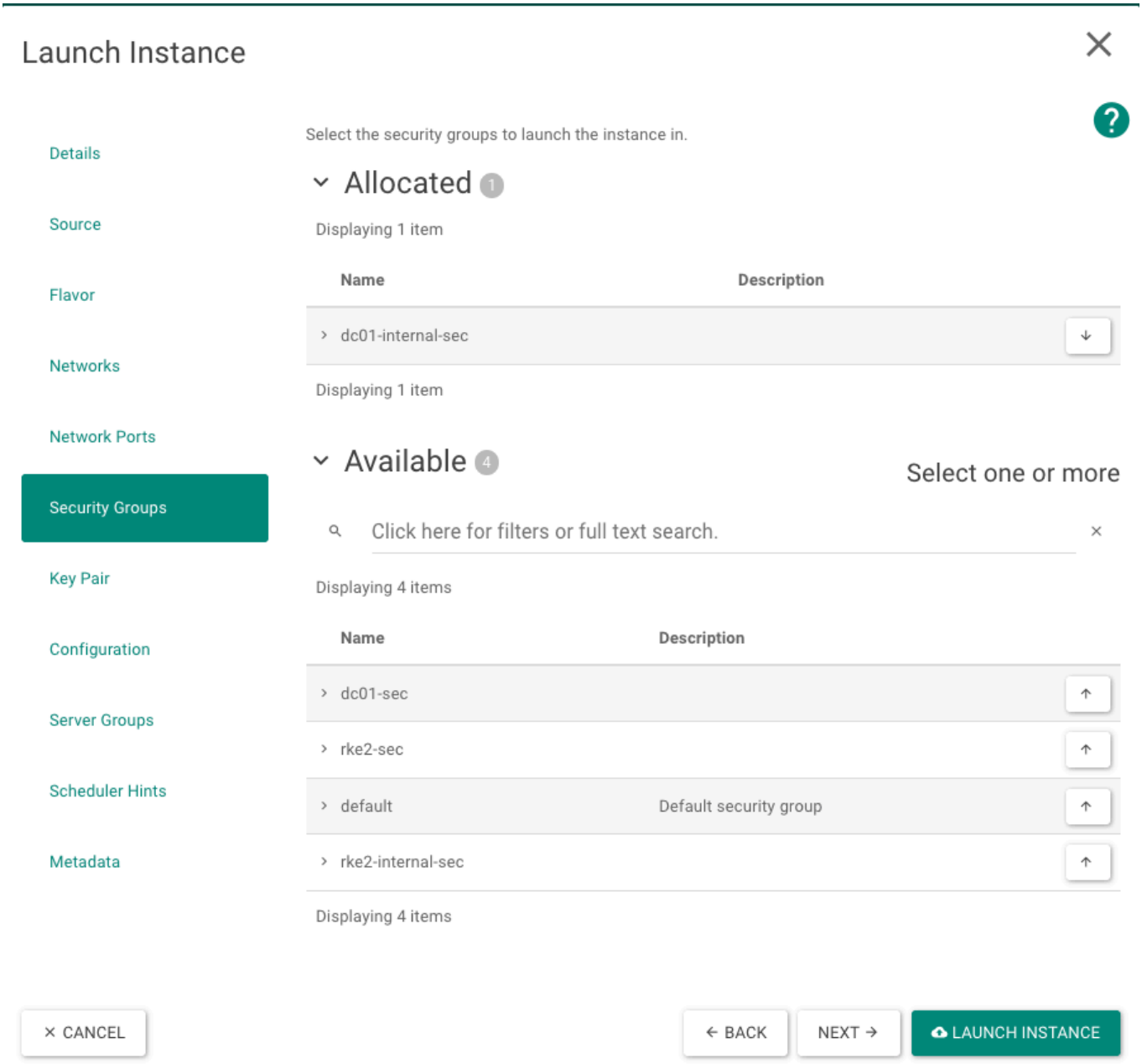
- **Disk Güvenliği:** 100 GB root disk alanı ile container logları ve imajları için geniş bir yer ayırıyoruz; böylece disk dolması sebebiyle node'un servis dışı kalmasını (Node Pressure) engelliyoruz.
- **Ölçeklenebilir Yapı:** Bu donanım seviyesiyle, cluster üzerine eklenecek olan izleme araçları (Prometheus, Grafana vb.) ve diğer sistem servisleri için gerekli kaynak rezervini en baştan hazır tutuyoruz.



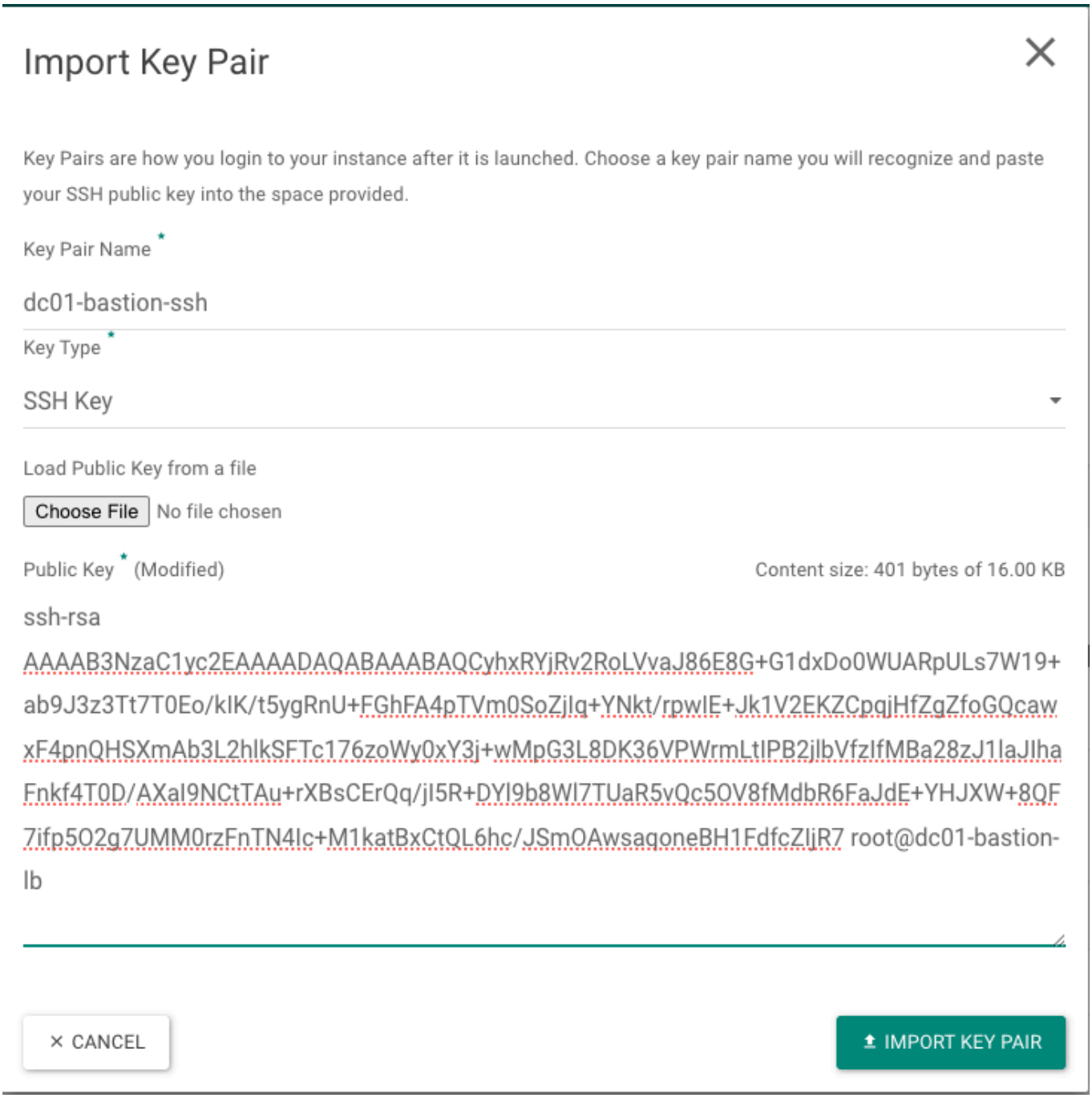
Clusterimiz için oluşturduğumuz network'ü sanal makinelerimize verelim. **dc01-private-net**



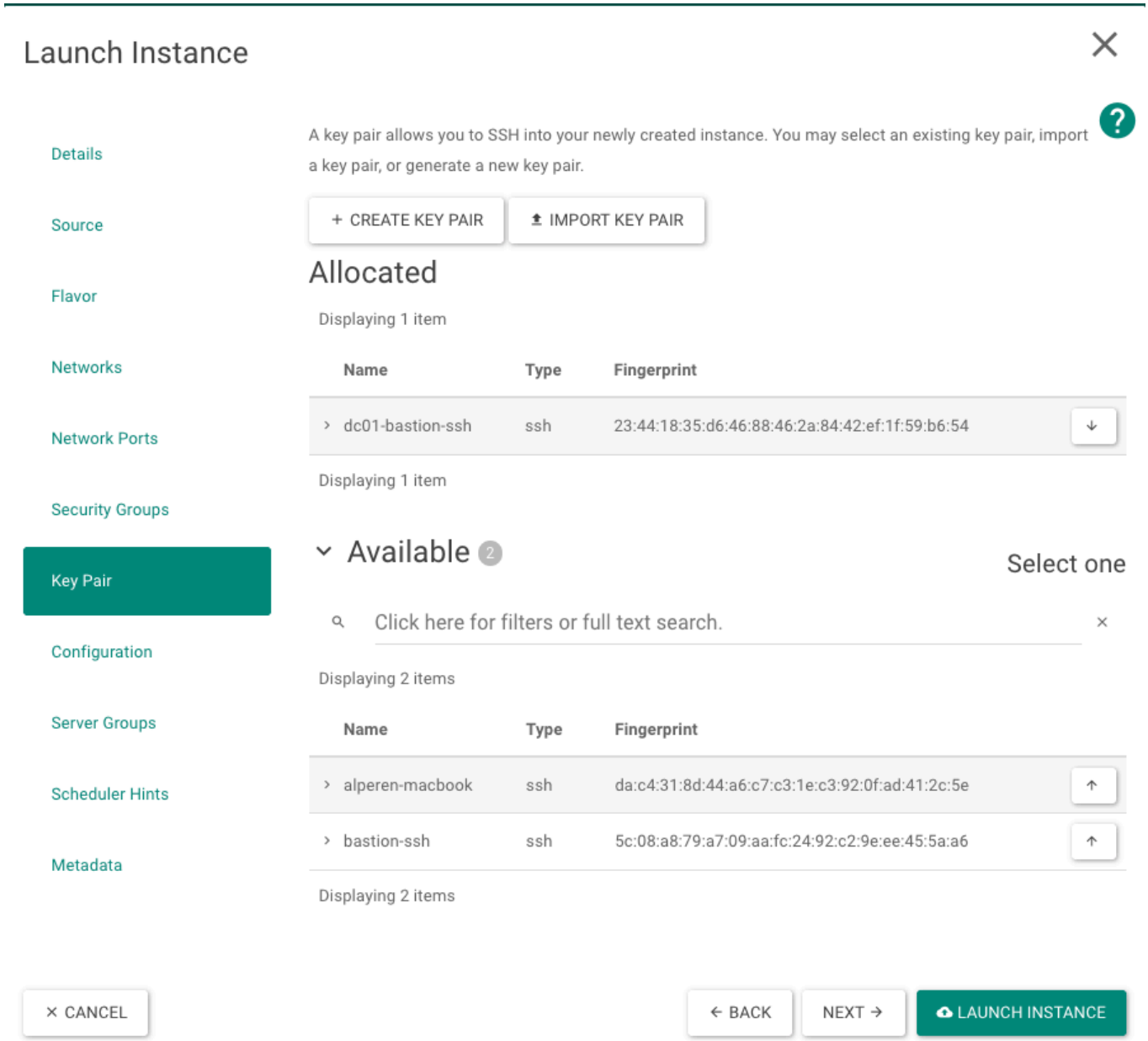
Node'larımızın kendi aralarında güvenli bir şekilde konuşabilmesi için **dc01-internal-sec** 'i tanımlıyoruz. Bu grup, cluster içindeki servislerin birbirine erişimini garanti altına alıyor.



LoadBalancer VM'imizden yine ssh key pair oluşturarak bu VM'imiz üzerinden master ve worker'lara erişmesini sağlamamız gerekmektedir. Kubernetes node'larımızı bizzat dışarı açmayarak güvenliğini sağlamak amacıyla bu adım önemlidir.



Ardından bu ssh key pairimizi sanal makinemize ekleyerek devam edebiliriz.



Yukarıdaki bütün aşamaları tamamladıktan sonra Kubernetes Clusterimizi kurabilmek adına OpenStack altyapısının bütün komponentleriyle birlikte kurmuş olduk. Buradan sonraki kısımda LoadBalancer tarafında Haproxy kurulumu ve Node'larımız üzerine RKE2 HA kurulumunu gerçekleştireceğiz.