# How to connect to kubernetes internal network using WireGuard

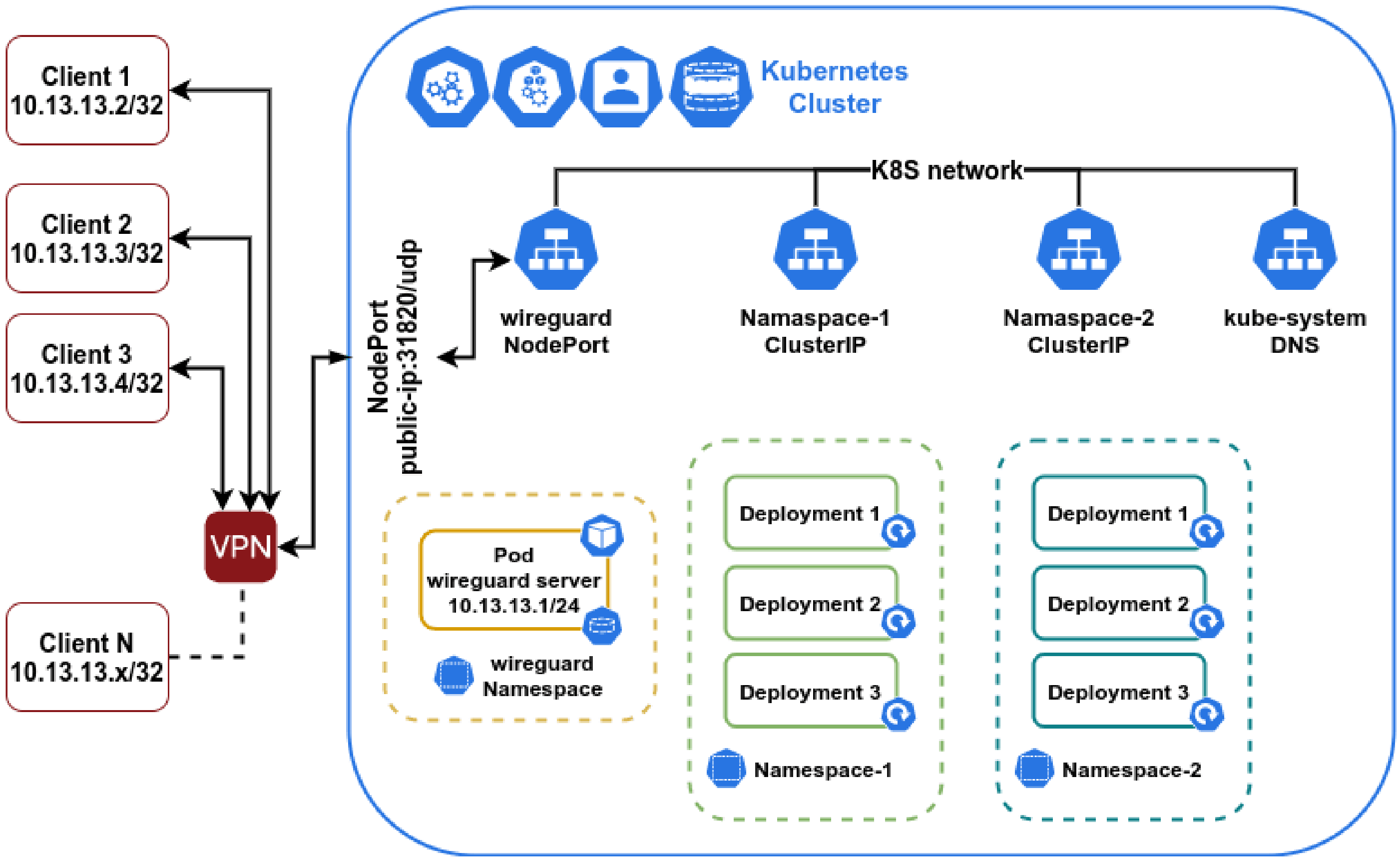
#kubernetes #docker #cloud #wireguard

Ivan Moreno Nov 13, 2020 · 4 min read

When you are testing your deployments in a kubernetes cluster on the cloud you have a few options to expose your services outside world, for example you can use a `NodePort` service, but also you need to configure the firewall rules for each `NodePort` service, the other type of service that you can use is `LoadBalancer` however each of them is billed by cloud provider. To solve this problem you can use a vpn running within your k8s cluster, this vpn can be exposed outside the cluster with a `NodePort` or `LoadBalancer` service. As client you can access to you kubernetes internal network using service `FQDN` in your local machine.

In this tutorial we gonna setup a pod that run wireguard server, this wireguard will be configured with the kube-dns service and generate cliente credentials automatically the diagram will be like this:

3 2 4



Assuming that you are in a testing k8s cluster in the cloud with multiple namespaces and services.

First we need to know the `kube-dns` IP address with the following command

```
$ kubectl -n kube-system get svc | grep kube-dns | awk '{print $3}'
# output example: 10.124.0.10
```

In order to isolate wireguard server from another apps, we need to create a wireguard namespace named `wireguard`

```
apiVersion: v1
kind: Namespace
metadata:
  name: wireguard
  labels:
    name: wireguard
```

To store wireguard config files, we need a persistent volume, in my case i'm using a gke managed service that provides me a storage class, so i'm gonna create a persistent volume claim to that storage class.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pv-claim-wireguard
```

3 2 4 •••

```
  storageClassName: "standard"
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10M
```

The next thing to configure is the environment variables of wireguard server, this will be do with a config map. The `kube-dns` IP from steps earlier will be set in `PEERDNS` field.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: wireguard-configmap
  namespace: wireguard
data:
  PUID: "1000"
  PGID: "1000"
  TZ: "America/Mexico_City"
  SERVERPORT: "31820"
  PEERS: "2"
  PEERDNS: "10.124.0.10"
  ALLOWEDIPS: "0.0.0.0/0, ::/0"
  INTERNAL_SUBNET: "10.13.13.0"
```

Now we can create the wireguard server pod, this pod needs to be privileged with `NET_ADMIN` and `SYS_MODULE` capabilities and needs to mount `/lib/modules` directory from the host. The image used is `ghcr.io/linuxserver/wireguard` from linuxserver.io

```
apiVersion: v1
kind: Pod
metadata:
  name: wireguard
  namespace: wireguard
  labels:
    app: wireguard
spec:
  containers:
  - name: wireguard
    image: ghcr.io/linuxserver/wireguard
    envFrom:
    - configMapRef:
        name: wireguard-configmap
    securityContext:
      capabilities:
```

3 2 4

```
        privileged: true
      volumeMounts:
        - name: wg-config
          mountPath: /config
        - name: host-volumes
          mountPath: /lib/modules
      ports:
      - containerPort: 51820
        protocol: UDP
      resources:
        requests:
          memory: "64Mi"
          cpu: "100m"
        limits:
          memory: "128Mi"
          cpu: "200m"
    volumes:
      - name: wg-config
        persistentVolumeClaim:
          claimName: pv-claim-wireguard
      - name: host-volumes
        hostPath:
          path: /lib/modules
          type: Directory
```

Finally to access to wireguard server, we need to create a service, this service could be a `NodePort` or `LoadBalancer`, in my case i used a `NodePort` service on port `31820`, take in mind that you probably need to configure a firewall rule to access at this service.

```
kind: Service
apiVersion: v1
metadata:
  labels:
    k8s-app: wireguard
  name: wireguard-service
  namespace: wireguard
spec:
  type: NodePort
  ports:
  - port: 51820
    nodePort: 31820
    protocol: UDP
    targetPort: 51820
  selector:
    app: wireguard
```

This configurations are in a single file `wireguard-pod.yaml` to execute just apply the file

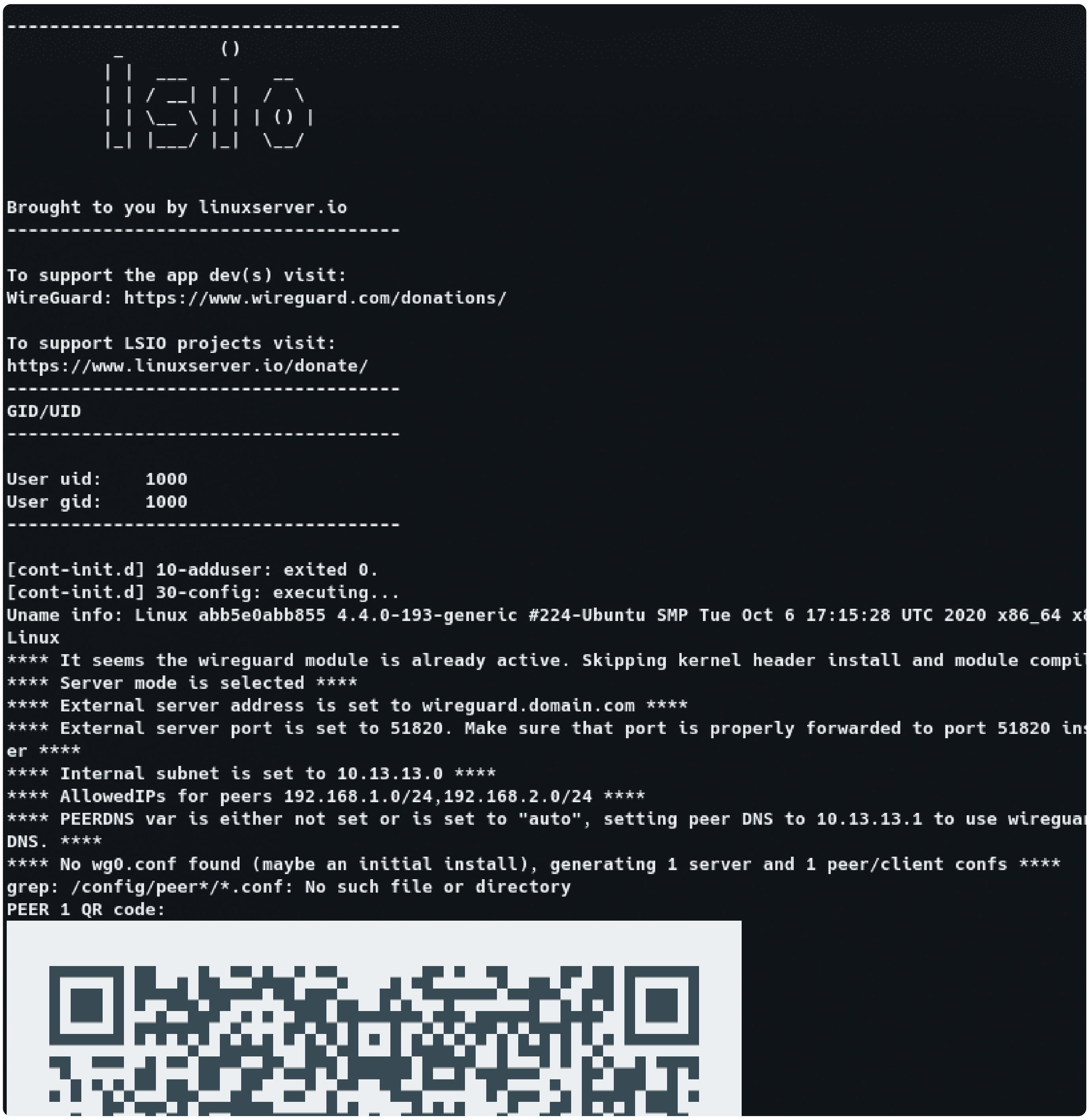
3 2 4 •••

```
$ kubectl apply -f wireguard-pod.yaml
```

The container generate a `QR` code for each peer, these `QR` appears in the logs of the pod, to see just type the following command

```
$ kubectl -n wireguard logs wwireguard
```

The output will be like this



```
-------------------------------------
          _         ()
         | |  ___   _    __
         | | / __| | |  /  \
         | | \__ \ | | | () |
         |_| |___/ |_|  \__/


Brought to you by linuxserver.io
-------------------------------------

To support the app dev(s) visit:
WireGuard: https://www.wireguard.com/donations/

To support LSIO projects visit:
https://www.linuxserver.io/donate/
-------------------------------------
GID/UID
-------------------------------------

User uid:    1000
User gid:    1000
-------------------------------------

[cont-init.d] 10-adduser: exited 0.
[cont-init.d] 30-config: executing...
Uname info: Linux abb5e0abb855 4.4.0-193-generic #224-Ubuntu SMP Tue Oct 6 17:15:28 UTC 2020 x86_64 x
Linux
**** It seems the wireguard module is already active. Skipping kernel header install and module compi
**** Server mode is selected ****
**** External server address is set to wireguard.domain.com ****
**** External server port is set to 51820. Make sure that port is properly forwarded to port 51820 in
er ****
**** Internal subnet is set to 10.13.13.0 ****
**** AllowedIPs for peers 192.168.1.0/24,192.168.2.0/24 ****
**** PEERDNS var is either not set or is set to "auto", setting peer DNS to 10.13.13.1 to use wiregua
DNS. ****
**** No wg0.conf found (maybe an initial install), generating 1 server and 1 peer/client confs ****
grep: /config/peer*/*.conf: No such file or directory
PEER 1 QR code:
```

In order to connect to wireguard server download mobile app of install in your local machine. See [wireguard.com](https://www.wireguard.com)

You can scan the code with the mobile app or copy the config file in your computer at

Now you can utilize the config file to activate the vpn. With `NetworManager` you can import the config file

```
$ nmcli connection import type wireguard file ~/peer1.conf
```

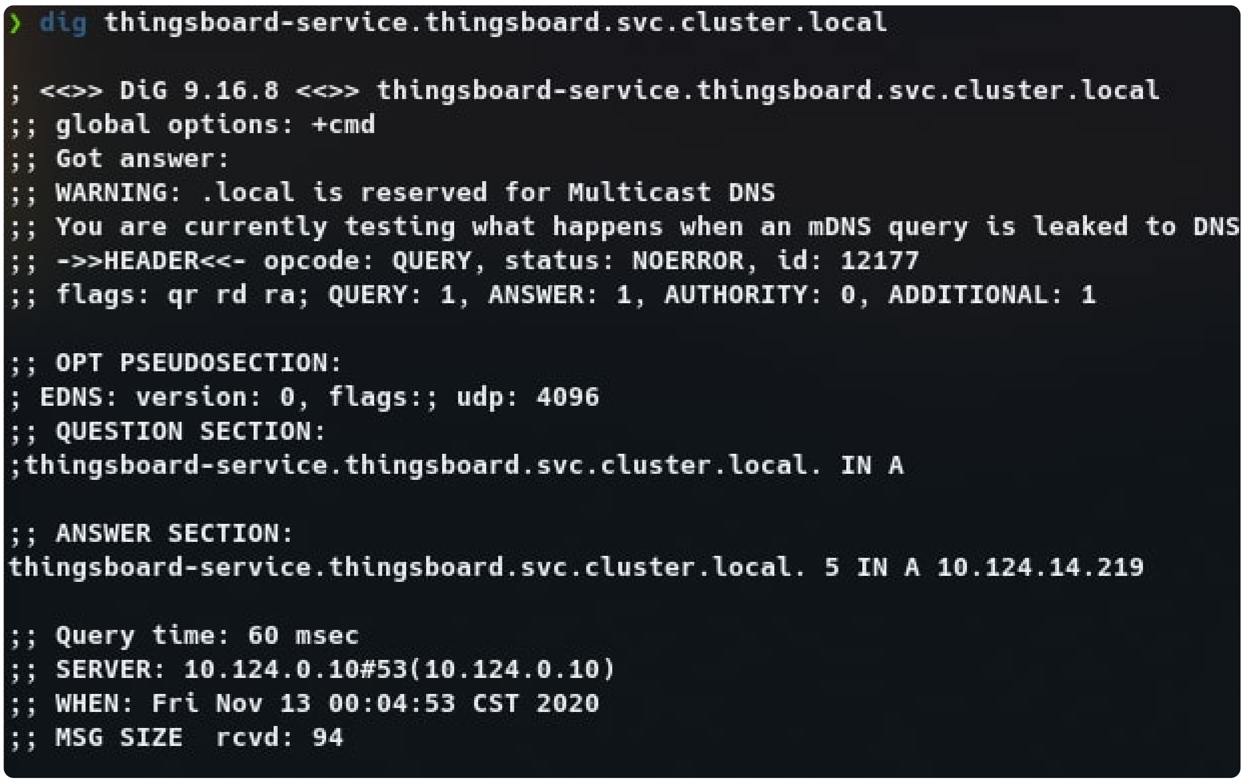
And activate or deactivate the connection

```
$ nmcli connection up peer1
$ nmcli connection down peer1
```

Finally to access a `ClusterIP` service within k8s cluster just use the IP of `ClusterIP` service or use the `FQDN` of the service using the following rule

```
<clusterip-service>.<namespace>.svc.cluster.local
```

Check the output of `dig` in a `FQDN` inside a remote k8s cluster, note that the query is answered by `kube-dns` IP inside the k8s cluster.



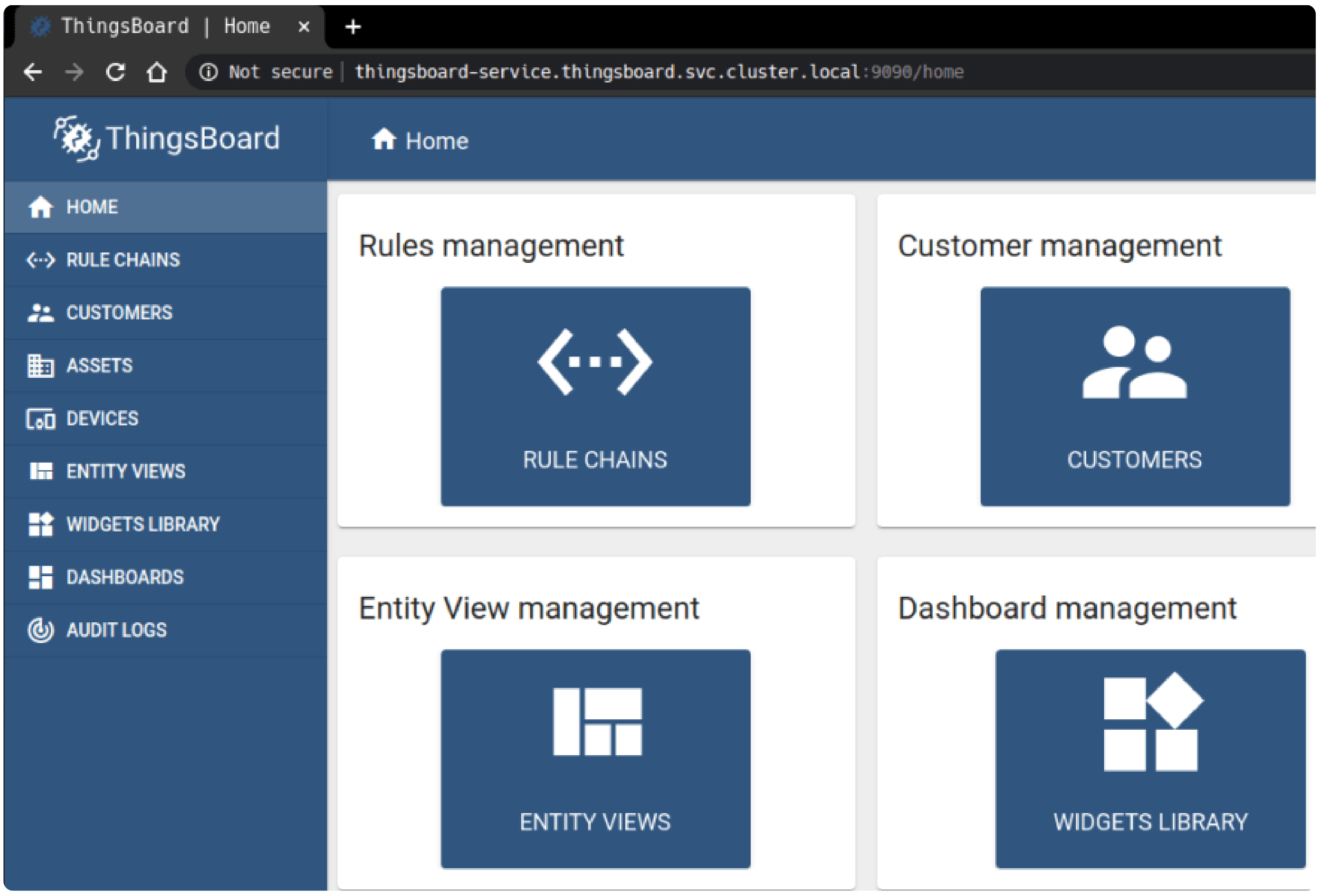
```
❯ dig thingsboard-service.thingsboard.svc.cluster.local

; <<>> DiG 9.16.8 <<>> thingsboard-service.thingsboard.svc.cluster.local
;; global options: +cmd
;; Got answer:
;; WARNING: .local is reserved for Multicast DNS
;; You are currently testing what happens when an mDNS query is leaked to DNS
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12177
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;thingsboard-service.thingsboard.svc.cluster.local. IN A

;; ANSWER SECTION:
thingsboard-service.thingsboard.svc.cluster.local. 5 IN A 10.124.14.219

;; Query time: 60 msec
;; SERVER: 10.124.0.10#53(10.124.0.10)
;; WHEN: Fri Nov 13 00:04:53 CST 2020
;; MSG SIZE  rcvd: 94
```

For example to access a `ClusterIP` service named `thingsboard-service` in the namespace `thingsboard` at `9090` port from our local machine through wireguard vpn:

```
http://thingsboard-service.thingsboard.svc.cluster.local:9090
```

And the output in our local environment

3 2 4



## Conclusion

This method is very useful for a managed kubernetes service in the cloud in a development environment because we can test our services without configure a nodeport for each service and his respectively firewall rule.

**WARNING: Only use this method in a development environment, don't use in a production environment**

Source Code

## Discussion

Subscribe

DEV

Add to the discussion

Justin Gauthier • Jan 17

Hey, awesome write up!

3 2 4

I've found github.com/linuxserver/docker-wire..., however it seems to not be working. Any ideas?

2

Ivan Moreno • Jan 21

Did you enable the ip_forward in the host?

1

Justin Gauthier • Jan 21

Yea, I had to enable sysctls on the host, thanks.

2

Code of Conduct • Report abuse

**Ivan Moreno**

Engineer || MSc student || DevOps in progress

Follow

**LOCATION**
Mexico city

**EDUCATION**
Communication and electronics Engineer

**JOINED**
Apr 15, 2020

## More from Ivan Moreno

How to bypass a firewall using a VPS on AWS
#vpn #linux #cloud #openvpn

How to install and configure you own VPN server in GCP with Wireguard

3 2 4

3 2 4