Securing Data-in-Transit with Calico & WireGuard

Wednesday, Sep 18, 2024

Prepared by Davide Sellitri,

Customer Success Engineer @ Tigera

Agenda

- Introduction
- The need for network encryption
- How it works
- Configuration Challenges
- Calico’s Way
- Lab Description
- Demo

TIGERA

Introduction

TIGERA

Introduction: Calico Open Source

8M+
Nodes

1M+
Clusters

1.4B+
Docker Pulls

50k+
Enterprises

>50%
of Fortune 100

166
Countries

Most adopted container networking and security solution

Introduction: Calico Open Source

Choice of Data Plane

- Pluggable Data Plane
- eBPF, Linux, Windows, VPP

Best in class performance

- Blazing fast performance
- Minimal CPU usage & occupancy
- Lower costs

Full Kubernetes Network policy support

- Full implementation Kubernetes network policies
- Additional support for policies across namespaces

Workload Interoperability

- Unified policy across hosts, bare-metal, VMs, and containers
- Mix and match workload types

Kubernetes Native Security Policy Model

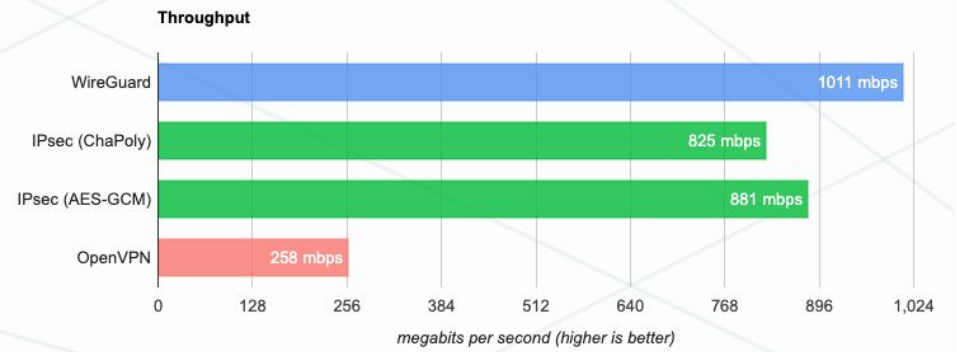
- Declarative security policies
- Unified model from host to application layers

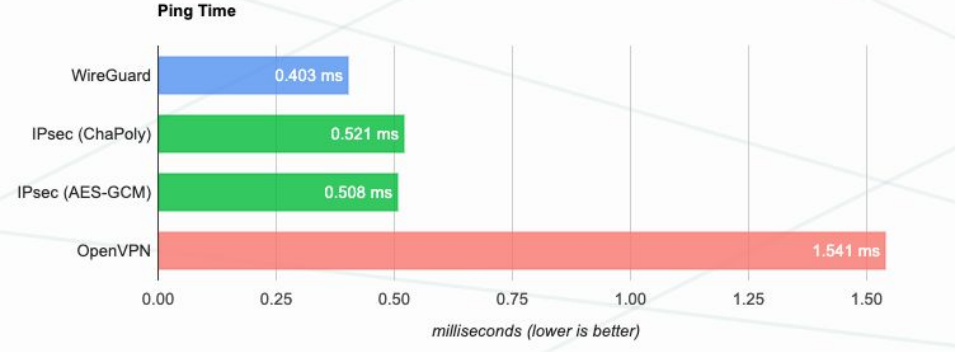
Scalable Networking with Encryption

- Exceptional scalability
- Advanced IP Address Management

Introduction: WireGuard Tunneling Protocol

- simple, lightweight, and performant
- UDP
- kernel VNI
- CryptoKey Routing
- Cross Platform





The need for network encryption

Wireguard: How it works

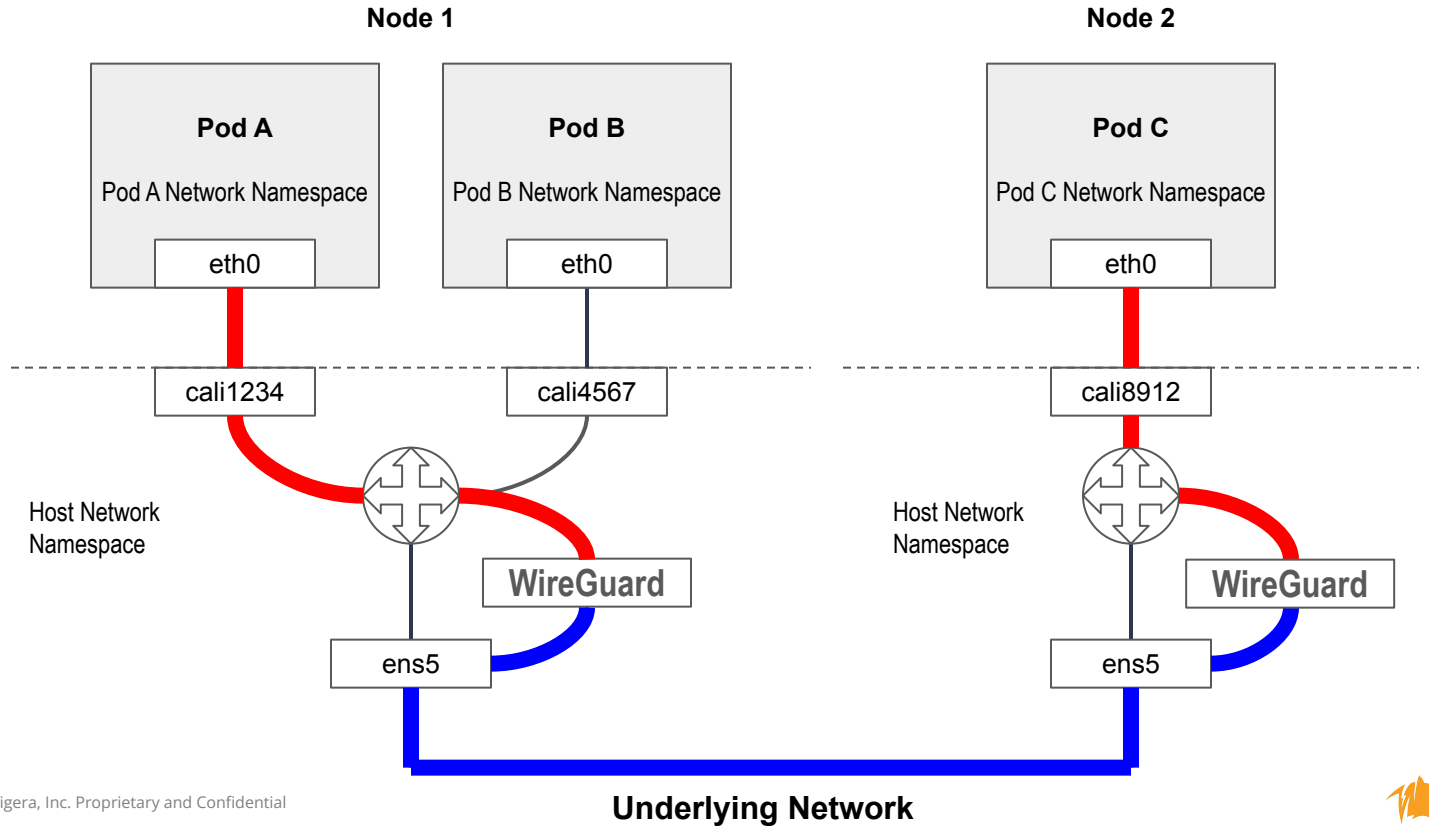
- Provides **wg, wg-quick,** and **wgctrl-go** to configure Wireguard and view information
- Establishes a secure tunnel between two peers
  - Manual **configurations** on each peer
  - **CryptoKey Routing** to associate public keys with IP addresses & endpoint
    - Routing table when sending packets
    - Access control list when receiving packets

```
ubuntu@ip-10-0-1-30:~$ sudo wg
interface: wireguard.cali
  public key: fpmFbBX4lMh7hH9vT4zgGnLypBcHz1XA81YFB7ZNAX8=
  private key: (hidden)
  listening port: 51820
  fwmark: 0x20000000

peer: 0srB5mLCPfZKyl9Fke/MAXuH0yjEtlHql3e9ZfjX0Hk=
  endpoint: 10.0.1.20:51820
  allowed ips: 10.48.115.64/26, 10.48.115.64/32, 10.48.115.71/32
  latest handshake: 15 seconds ago
  transfer: 17.88 KiB received, 154.35 KiB sent

peer: kw3cnxNy2CR/3c1LvVujhLpLI944rcyhkX2z6u3WuGk=
  endpoint: 10.0.1.31:51820
  allowed ips: 10.48.116.128/26, 10.48.116.128/32, 10.48.116.138/32
```

How it works: End Result



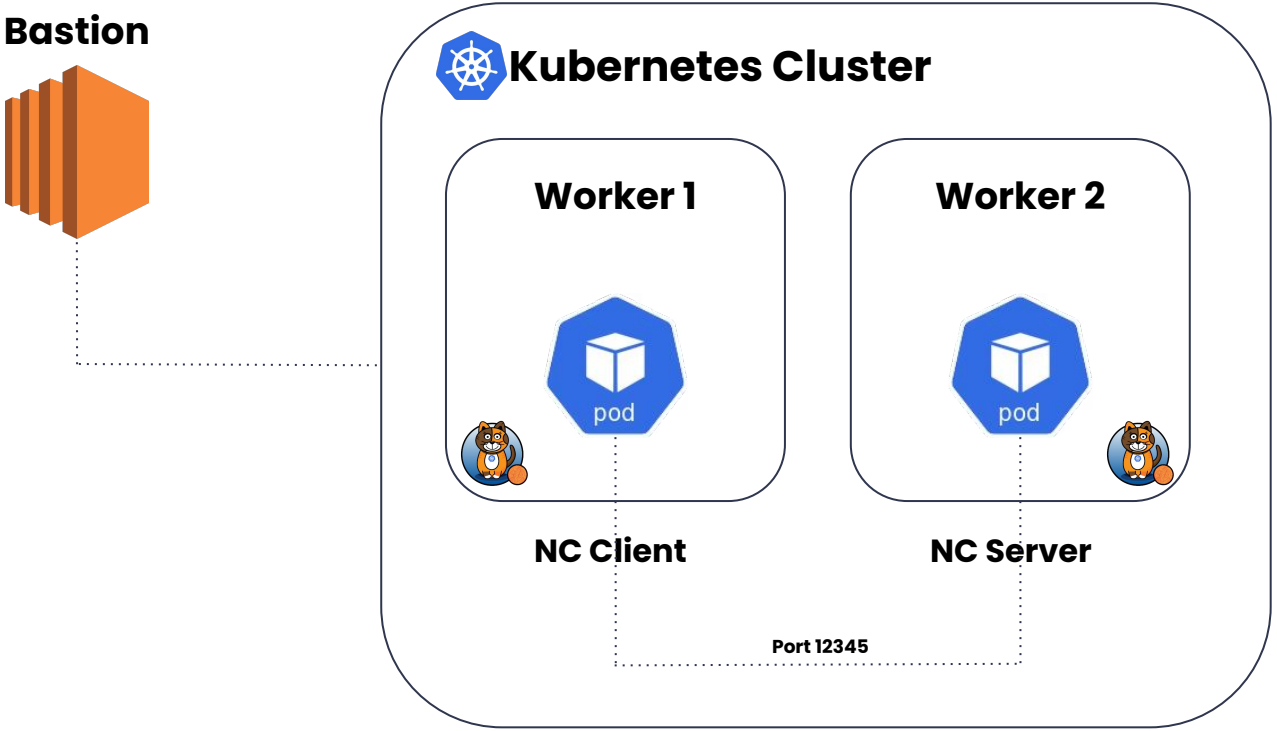
Wireguard Configuration Challenges

- A manual process
  - Create a key pair
  - Create tunnel virtual interface, allocate an IP address, and bring the interface online
  - Configure the endpoint address
  - Configure and continuously update the allowed IPs
  - No in-band key exchange: distribute public keys securely out–of-band
  - Keys are stored in memory and are lost upon node restarts

Calico's Way

```
kubectl patch felixconfiguration default
--type='merge' -p '{"spec":{"wireguardEnabled":true}}'
```

Lab Description



Demo

Q & A

GitHub

TIGERA

Follow us on:

Thank you!

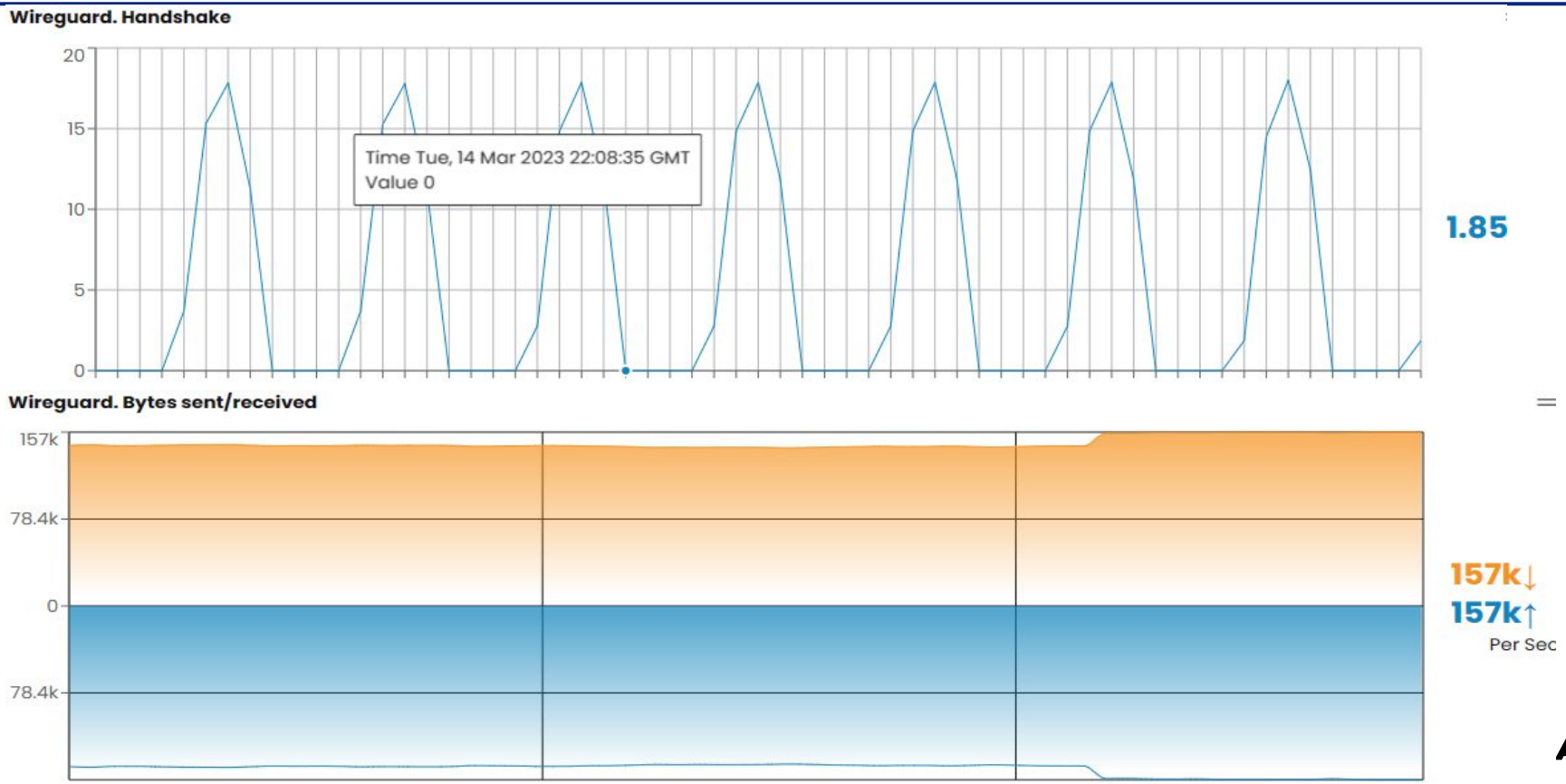
Feedback

TIGERA

Follow us on:

Additional Slides

Calico Wireguard Widget in Calico Cloud or Calico Enterprise



Calico - Wireguard Encryption Support

- Provides in-cluster pod traffic encryption
- Supported platforms:

The following platforms using only IPv4:

- Kubernetes, on-premises
- EKS using Calico CNI
- EKS using AWS CNI
- AKS using Azure CNI
- AKS using Calico CNI

All platforms listed above will encrypt pod-to-pod traffic. Additionally, when using AKS or EKS, host-to-host traffic will also be encrypted, including host-networked pods.

Implementation ...

- Only on nodes that have WireGuard installed
- Rules from local workload to the wireguard table

```
$ ip rule
0:     from all lookup local
99:    from all fwmark 0x0/0x80000 lookup 1
32766: from all lookup main
32767: from all lookup default
```

- FWMark is used to prevent routing loops
  - First pass routes packets to wireguard (in table 1)
  - Second pass routes wireguard encrypted UDP packets to the other host (in main table)

Implementation ...

- There is a new wireguard route table (index 1)

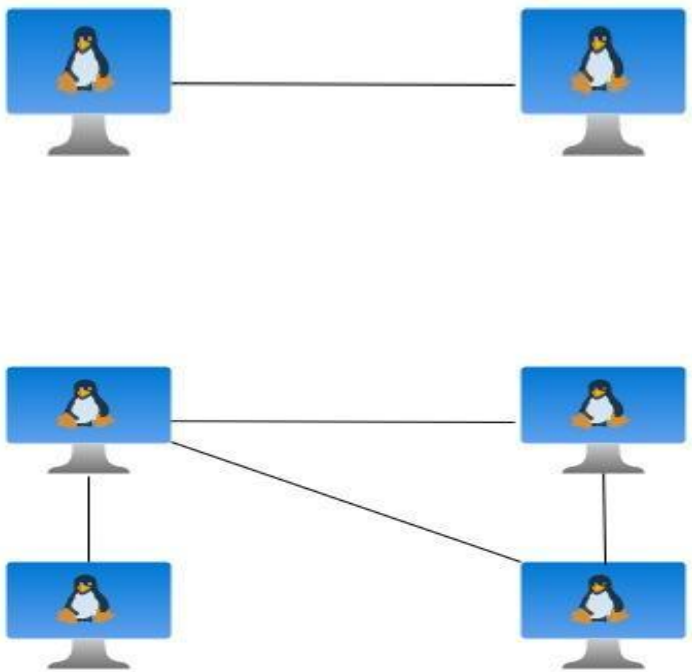
```
$ ip route show table 1
throw 192.168.6.128/26
192.168.24.128/26 dev wireguard.cali  scope link
192.168.42.64/26 dev wireguard.cali  scope link
```

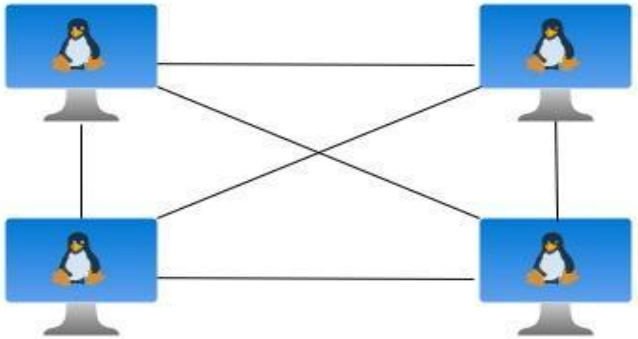
  - Routes to pods on hosts that have wireguard installed via wireguard interface
  - Routes to local workloads, or to pods on hosts that do not have wireguard installed revert back to previous routing (throw routes)

Implementation …

- Supported config options in FelixConfig:
  - wireguardInterfaceName
  - wireguardListeningPort
  - wireguardMTU
  - wireguardRoutingRulePriority
  - wireguardKeepAlive

Wireguard Topologies





Wireguard Configuration - Calico's Way

- A felix configuration parameter
- Uses **wgctrl-go** to configure Wireguard
- Configures a wireguard virtual interface named **wireguard.cali**
- Key pairs are stored in memory and regenerated upon the node restart
- Stores public key in the datastore and securely distribute them between the nodes
- Uses the default Wireguard port 51820
- Configures the peers by accessing the required information in the datastore
  - Public key
  - Endpoint address - node API resource
  - Allowed IPs - Pod IP addresses and IPAM blocks
- Configures ip rule and special routing table
- Configures iptables/ebpf rules to allow Wireguard traffic
- Encrypts pods traffic across different nodes
  - Intra-node pod connectivity is not encrypted by Wireguard

Wireguard Configuration

- Load Wireguard kernel module 1

Peer1 2

```
wg genkey | sudo tee /etc/wireguard/private.key
sudo cat /etc/wireguard/private.key | wg pubkey | sudo tee /etc/wireguard/public.key
sudo ip link add dev wg0 type wireguard
sudo ip address add dev wg0 10.16.1.220/24
sudo ip link set up dev wg0
```

Peer2 2

```
wg genkey | sudo tee /etc/wireguard/private.key
sudo cat /etc/wireguard/private.key | wg pubkey | sudo tee /etc/wireguard/public.key
sudo ip link add dev wg0 type wireguard
sudo ip address add dev wg0 10.16.1.221/24
sudo ip link set up dev wg0
```

Wireguard Configuration

```
root@peer1:~#vi /etc/wireguard/wg0.conf
[Interface]
PrivateKey = yB8I/xYvMRCe5pjb7CFv3zbespTqNLNcX0FR0S1FmFg=
ListenPort = 51820

[Peer]
PublicKey = q9uKGNlQOSwAD+/4EA+9A11p1lqyHtEN/qOyCf8mGms=
Endpoint = 172.16.1.221:51820
AllowedIPs = 10.16.1.221/32
```

3

4

```
sudo wg setconf wg0 wg0.conf
```

```
root@peer2:~#vi /etc/wireguard/wg0.conf
[Interface]
PrivateKey = GJzf73I/fdKmIAozFF9Qd1LGLBG52HWyj2zIAVgogG0=
ListenPort = 51820

[Peer]
PublicKey = lNFJiC37+WwGAYK8LjJ5PGB+LbraoRlwOlOdCtS3Rk4=
Endpoint = 172.16.1.220:51820
AllowedIPs = 10.16.1.220/32
```

3

4

```
sudo wg setconf wg0 wg0.conf
```


TIGERA

Troubleshooting

- Tcpdump on the wireguard interface（wireguard.cali)
  - Should see packets routed to wireguard and decrypted response
- Tcpdump on the main host interface
  - Should see UDP packets between the wireguard supported nodes
- Wireguard shares public key information through the Node resource status
  - Check that the node has a public key assigned
  - If wireguard is enabled and installed, but there is no public key, check Felix logs.

```
metadata:
  annotations:
    projectcalico.org/WireguardPublicKey:
jlkVyQYooZYzI2wFfNhSZez5eWh44yfq1wKVjLvSX
gY=
```

Wireguard Configuration - Enable Wireguard

- `kubectl patch felixconfiguration default --type='merge' -p '{"spec":{"wireguardEnabled":true}}'`
- `kubectl patch felixconfiguration default --type='merge' -p '{"spec":{"wireguardEnabledV6":true}}'`
- `kubectl patch felixconfiguration default --type='merge' -p '{"spec":{"wireguardEnabled":true,"wireguardEnabledV6":true}}'`

Wireguard Configuration Validation

```
# kubectl get nodes -o yaml | grep -i wireguard

      projectcalico.org/IPv4WireguardInterfaceAddr: 192.168.0.106
      projectcalico.org/WireguardPublicKey:
G/syC2zarrAsAJp6iF9+g73ym+m8Wb+limuqpnqoDWI=

      projectcalico.org/IPv4WireguardInterfaceAddr: 192.168.1.232
      projectcalico.org/WireguardPublicKey:
3hwASDwR/miQ6JLB53j2xx+3ZbqJsVtW0OLj0mb8+ns=

      projectcalico.org/IPv4WireguardInterfaceAddr: 192.168.1.108
      projectcalico.org/WireguardPublicKey:
Vz6OTLji5oRN46EMHrU649TZj07fzGLhT3FIgfWd0XA=
```

IP Route Configurations

```
# ip rule list lookup 1
99:     not from all fwmark 0x200000/0x200000 lookup 1

----

# ip route list table 1
throw 192.168.0.96/28
192.168.1.96 dev wireguard.cali scope link
192.168.1.96/28 dev wireguard.cali scope link
192.168.1.108 dev wireguard.cali scope link
192.168.1.112/28 dev wireguard.cali scope link
192.168.1.208 dev wireguard.cali scope link
192.168.1.208/28 dev wireguard.cali scope link
192.168.1.224/28 dev wireguard.cali scope link
192.168.1.232 dev wireguard.cali scope link
```

IPTables Configuration

```
# iptables-save | grep -i wireguard

-A cali-INPUT -p udp -m comment --comment "cali:w1xlp5qwD97H6wH1" -m
comment --comment "Allow incoming IPv4 Wireguard packets" -m multiport
--dports 51820 -m addrtype --dst-type LOCAL -j ACCEPT

-A cali-OUTPUT -p udp -m comment --comment "cali:XjqWdMh-cno69n2T" -m
comment --comment "Allow outgoing IPv4 Wireguard packets" -m multiport
--dports 51820 -m addrtype --src-type LOCAL -j ACCEPT

-A cali-POSTROUTING -o wireguard.cali -m comment --comment
"cali:kgfCOPW4UKtzMAmO" -m addrtype ! --src-type LOCAL --limit-iface-out
-m addrtype --src-type LOCAL -j MASQUERADE --random-fully
```