Offensive Operations as Code: Personal Journey in Innovating and Automating Cloud Infrastructure

By Vito "Trust_No_One" De Laurentis

HackInBo® Winter 2024 Edition
23ª EDIZIONE

Vito "Trust_No_One" De Laurentis è un esperto di sicurezza informatica con 20 anni di esperienza nel settore della cybersecurity. Programmatore dal 1998, inizia la sua carriera nel 2004 come sistemista, specializzandosi nel 2005 come istruttore di "Penetration Testing" e successivamente di "Red Teaming". Da sempre sostenitore della filosofia hacking come vera e propria arte, Vito ha lavorato instancabilmente per affermare la cultura della cybersecurity, diventando figura di riferimento nella Pubblica Amministrazione nella quale ha esercitato la sua opera di divulgazione.
Attualmente ricopre il ruolo di ricercatore nel ramo dell'incident response e del threat hunting, continuando a contribuire alla sicurezza delle organizzazioni attraverso l'identificazione e la mitigazione delle minacce avanzate.

Questa presentazione ripercorre il mio personale viaggio nell'innovazione delle offensive cyber operations, utilizzando infrastrutture cloud con Terraform e il paradigma Infrastructure as Code (IaC). Le architetture introdotte, sfruttando il linguaggio HCL (HashiCorp Configuration Language), si integrano con strumenti avanzati di comando e controllo, strumenti di networking e anonimizzazione, riuscendo a garantire scalabilità e sicurezza. L'automazione della configurazione e la gestione delle risorse cloud possono consentire una migliore efficacia delle operazioni offensive complesse.

WHOAMI

Vito "Trust_No_One" De Laurentis (a.k.a. Ev1lman)
Hacking Enthusiast
Professional Procrastinator
Red Teamer Student
Techno & DJing addicted

WHOAMI /ALL

delaurentis@live.it
https://www.linkedin.com/in/vito-de-laurentis-00a20410

ROADMAP

Introduzione:

- Offensive Operations
- Offensive Infrastrucrture

Parte 1: Terraform

- Concetti IaC
- Cos’è Terraform e il linguaggio HCL
- Moduli
- Provisioner
- Risorse
- Variabili, Random e Condition

Parte 2: Architettura

- Rete Servizi: Vpn, C2, Store, Gateway
- Redirectors: Caddy, Tor, Socat
- Security: UFW vs VPC
- Geolocation & Maps

Offensive Operations

La definizione data dal "National Institute of Standards and Technology" (NIST) descrive le "Offensive Cyberspace Operations" come operazioni svolte nel cyberspazio che hanno lo scopo di esercitare influenza o potere, spesso attraverso l'uso della forza, sia nel cyberspazio che tramite esso.

Concetto evidenziato dalla pubblicazione "Joint Cyberspace Operations 3-12" del 2013, che rappresenta la dottrina dei "Joint Chiefs of Staff" per le operazioni nel dominio cibernetico.

DCO

Infatti, sono definite le implementazioni delle cosiddette “Operazioni Cyber Difensive” (DCO), ovvero azioni di risposta difensive nel cyberspazio.

Offensive Infrastructure

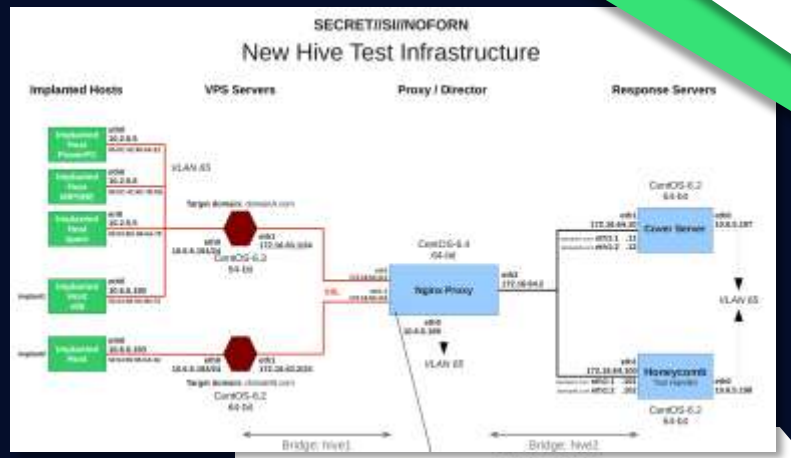
Il progetto Hive di Vault 7 è stato rivelato da WikiLeaks nel 2017, e riguarda una serie di strumenti di hacking sviluppati dalla CIA.

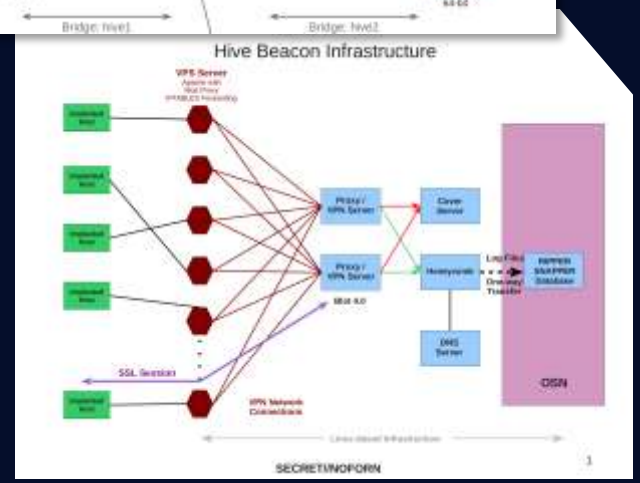
Vault 7: è il nome in codice del fascicolo di documenti riservati riguardanti le capacità di hacking dell'agenzia statunitense, inclusi software, malware, virus, trojan e altre armi cibernetiche.

Hive: è una delle componenti chiave di queste capacità. Si tratta infatti di una piattaforma di controllo e comando (C2) utilizzata dalla CIA per gestire i malware e le operazioni di hacking su larga scala.

Secondo i documenti Hive permette(?) alla CIA di controllare a distanza i dispositivi compromessi e raccogliere informazioni da essi.

Caratteristiche :
1.Infrastruttura di controllo
2.Comunicazioni crittografate
3.Mascheramento del traffico
4.Plugin modulari



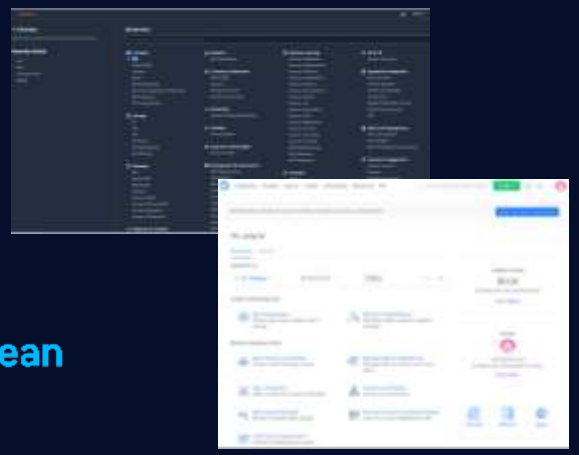


"Trust_No_One"

Ma COME realizzarla?

Cosa si intende per Infrastructure as Code (Iac)

L'Infrastructure as Code (IaC) è un approccio alla gestione dell'infrastruttura IT che prevede la scrittura e l'esecuzione di codice per definire, distribuire, aggiornare e distruggere una o più infrastrutture.

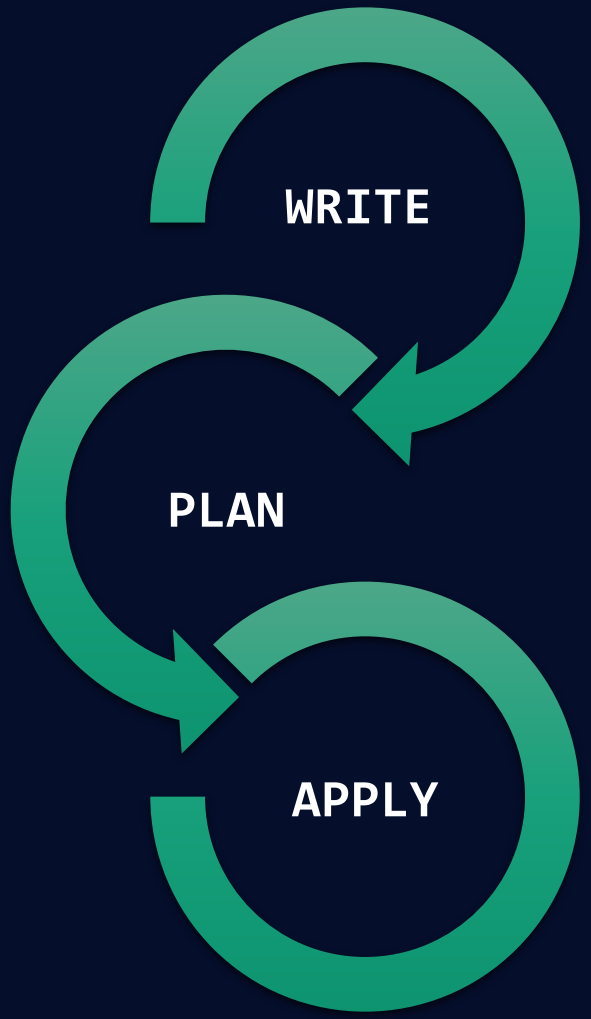


```
## ec2.tf
provider "aws" {
    access_key = "<aws_access_key"
    secret_key = "<secret_key>"
    region = "<aws_region>"
}
resource "aws_instance" "example" {
    count = 5
    ami = "ami-v1"
    instance_type = "t2.micro"
```

Infrastructure espressa come codice Multicloud

Cos'è Terraform e il linguaggio HCL

Terraform Workflow



Scrivi elemento

Piano di Azione

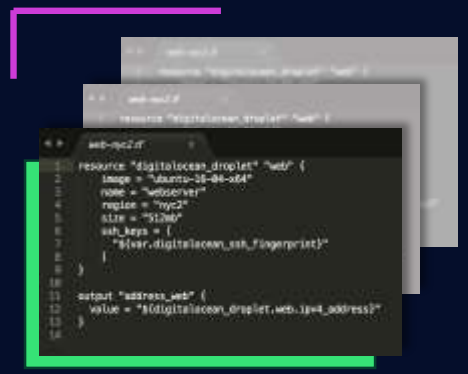
- Check dipendenze
- Check infrastruttura già esistenteù
  - [tfstate]

Applica le Operazioni

- Provisioning Gerarchie

Architettura: Cos'è Terraform e il linguaggio HCL

Terraform è un tool che permette di definire le risorse informatiche in un ambiente cloud o onpremise in un file di configurazione leggibile dall'uomo (secondo il paradigma IaC).



```
## ec2.tf
provider "aws" {
    access_key = "<aws_access_key"
    secret_key = "<secret_key>"
    region = "<aws_region>"
}
resource "aws_instance" "example" {
    count = 5
    ami = "ami-v1"
    instance_type = "t2.micro"
```

./main.tf
./providers.tf

Providers.tf

- Versione
- Autenticazione
- Dettagli Progetto

./terraform init
registry.terraform.io

RPC

Plugins
DO_Provider

Plugins
AWS_Provider

API
DigitalOcean
FIREWALL

API
AWS
AWS Vpc
AWS Sec
EC2

Client Library

4396 providers, 17249 modules & counting

"Trust_No_One"

Terraform: Concetti Base [File e Moduli]

- Providers
- Risorse
- Variabili
- Output
- Moduli

Modulo ‘nome_modulo’

Providers.tf

```
terraform {
  required_providers {
    digitalocean = {
      source  = "digitalocean/digitalocean"
      version = "~> 2.0«
...

provider "digitalocean" {
    token = var.do_token
}
```

main.tf

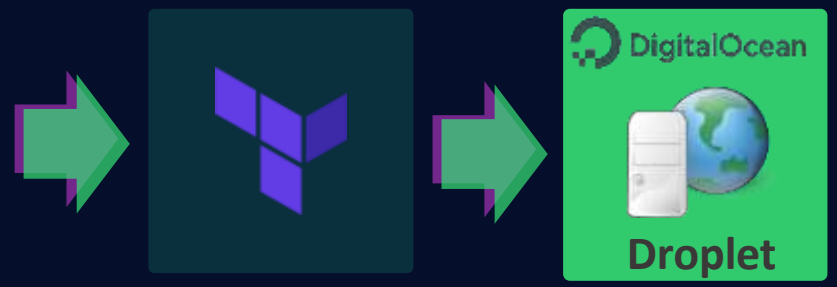
```
resource "digitalocean_droplet" "Service2" {

    name = "Service${local.index1}"
    image = "ubuntu-23-10-x64"
    region = var.Transport_nation
    size = "s-4vcpu-8gb"
    #depends_on = [module.SpectreCORE]
    ipv6 = false
    ...
```

variables.tf

```
variable "Transport_nation" {
            description = "Location Datacenter"
            type        = string
}
```

output.tf

```
output "service_ip" {
  value = digitalocean_droplet.Service2.ipv4_address
}
```



./terraform apply
./terraform destroy

Terraform: Risorse

Le risorse sono gli elementi cardine nel codice nel linguaggio Terraform.

Attraverso le risorse è possibile creare codice in grado di comunicare a Terraform quali oggetti di infrastruttura si prevede di creare, eliminare o modificare, come istanze di calcolo, reti virtuali o componenti di livello più avanzato, come applicazioni web e database, aws lambda, DO function etc.

Quando si inserisce un blocco di codice risorsa nel file di configurazione, esso inizia con il nome del provider, per esempio aws_instance, azurerm_subnet o google_app_engine_application

```
resource "digitalocean_droplet" "SpectreC4" {
image = "ubuntu-23-10-x64"
name = "SpectreC4"
region = var.c4_nation
size = "s-1vcpu-1gb"
ipv6 = false

ssh_keys = [var.ssh_fingerprint]
connection {
      host = self.ipv4_address
      user = "root"
      type = "ssh"
      private_key = "${file(var.pvt_key)}"
      timeout = "2m"      }

. . .
```

```
resource "aws_security_group" "http-redirector-security" {  name
= "http-redirector-security"
      vpc_id = "${aws_vpc.default.id}"
      provider = aws.region

ingress {
      from_port = 22
      to_port = 22
      protocol = "tcp"
      cidr_blocks = ["0.0.0.0/0"]    }

ingress {
      from_port = 25
      to_port = 25
      protocol = "tcp"
cidr_blocks = ["0.0.0.0/0"]    }
. . .
```

Terraform: Variabili

Terraform supporta anche variabili che possono essere definite nel codice di configurazione.
L'unica differenza tra le variabili di altri linguaggi di programmazione e le variabili di Terraform è che, nelle variabili di Terraform, si suppone che tu debba definire i valori di input quando vuoi eseguire il tuo codice.

```
variable "do_token" {
      description = "Token API di DigitalOcean"
      type        = string
      default     = "dop_v1_bda00839b86…a95"
}


variable "pub_key" {
      description = "public"
      type        = string
      default     = "/home/.ssh/id_rsa.pub"
}


variable "pvt_key" {
      description = "private"
      type        = string
      default     = "/home/.ssh/id_rsa"
}

…

variable "c4_nation" {
      description = "Location DigitalOcean"
      type        = string
}
```

Terraform: Output

I valori di output sono i valori restituiti da una risorsa, modulo o dati di Terraform e hanno molteplici utilizzi:

- L'output di una risorsa, modulo o dati può essere richiamato in altre risorse, moduli o dati se c'è una dipendenza dalla prima risorsa. Ad esempio, se si vuole creare una subnet Azure e si è già creata una rete virtuale Azure, è possibile utilizzare l'output della rete virtuale per fornire il riferimento alla subnet.

- I valori di output sono anche visualizzabili tramite CLI di Terraform eseguendo "terraform apply". In questo modo è possibile verificare i valori di output delle risorse, dei moduli e dei dati durante l'esecuzione.

```
output "out_spectreC4_ip" {
value = module.SpectreCORE.spectreC4_ip}

output "out_transport1_ip" {
value = module.Transport.transport_ip }

output "out_aws_tor_ip" {
value = module.aws_tor_node.aws_public_ip }

output "out_aws_net_ip" {
value = module.aws_net_node.aws_public_ip }

output "CloudFront_Distribution_Url_Net" {
value = module.CloudfrontNet.disturl }

output "CloudFront_Distribution_Url_Tor" {
value = module.CloudfrontTor.disturl }
```

Terraform: Output

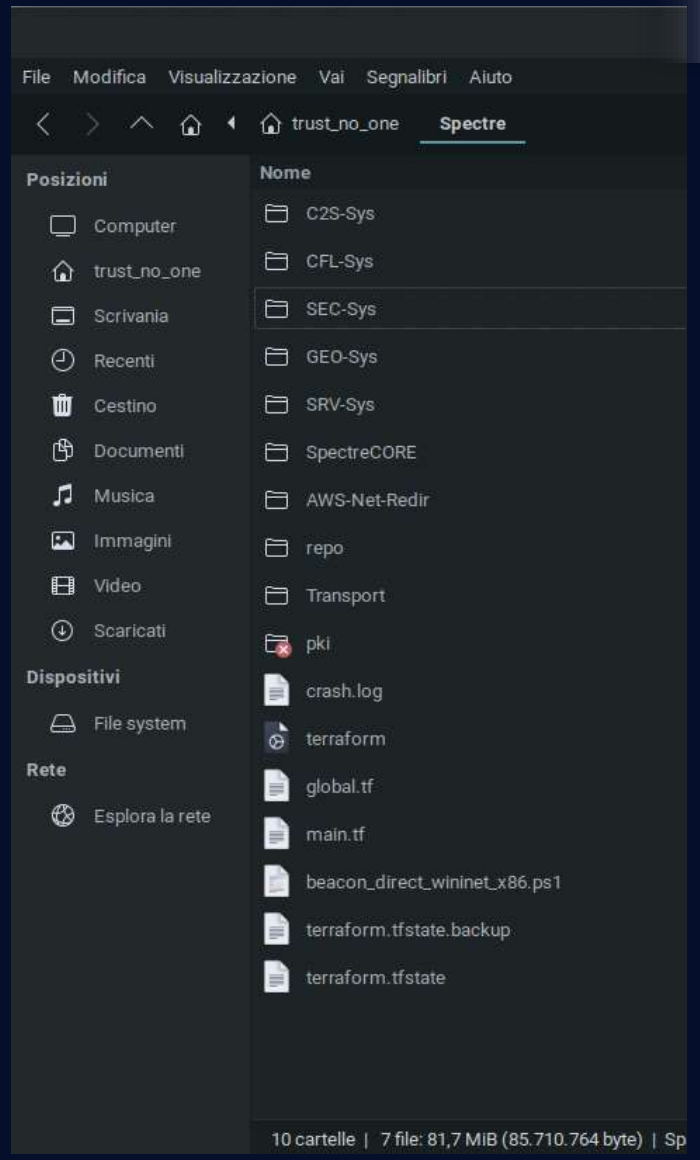
```
./main.tf

module "SpectreCORE" {
source = "./SpectreCORE"
c4_nation = "SYD1"
dns_name = var.dns_name
}


module "Transport" {
source = "./Transport"
spectreC4_ip = module.SpectreCORE.spectreC4_ip
Transport_nation= "TOR1"
}
```

```
./SpectreCORE/main.tf

resource "digitalocean_droplet" "SpectreC4" {
image = "ubuntu-23-10-x64"
name = "SpectreC4"
region = var.c4_nation
size = "s-1vcpu-1gb"    ipv6 = false

…
output "spectreC4_ip" {
value = digitalocean_droplet.SpectreC4.ipv4_address}
```

```
./Transport/main.tf

provisioner "remote-exec" {

inline = [
"sed -i ‘s/XXX/${var.spectreC4_ip}/g’ /etc/openvpn/client/client.conf",

"systemctl enable openvpn-client@client.service",
"systemctl start openvpn-client@client.service"
]
      connection {
            type = "ssh"
            user = "root"
            private_key = file(var.pvt_key)
      }
}
```

Terraform: Moduli

In Terraform, un modulo è un insieme di risorse e dati correlati, trattati come un'entità unica. Un modulo consiste in uno o più file di configurazione che descrivono un gruppo di risorse specifiche. L'utilizzo dei moduli permette di strutturare il codice in modo organizzato, suddividendo le configurazioni in blocchi logici, riutilizzabili e facili da gestire. Grazie a questa struttura, è possibile definire un modulo una sola volta e applicarlo in diversi contesti, facilitando la gestione dell'infrastruttura..



Terraform: Moduli

./main.tf

```
module "SpectreCORE" {
source = "./SpectreCORE"
c4_nation = "SYD1"
dns_name = var.dns_name    }

module "Transport" {
source = "./Transport"
spectreC4_ip = module.SpectreCORE.spectreC4_ip
Transport_nation= "TOR1"     }


module "aws_tor_node" {
source = "./AWS-Net-Redir"
transport_ip = module.Transport.transport_ip
Reflector_zone =  "us-east-1"
ami_key = "ami-0ea09f73e400d0c98"
name_connection = "tor"
# tipologia
mail_record = true
name_record = true
ns_record = false
index ="1"
dns_name_aws= "sky-warp.net"
dns_subname_aws = "torx"
dns_redirsite_aws = "www.repubblica.it"
useragent = "Mozilla/5.0 …"}

module "aws_extra_node" {
source = "./AWS-Net-Redir"  }
```

./AWS-Net-Redir/main.tf

```
resource "aws_instance" "http-rdir" {
provider = aws.region
ami = var.ami_key
instance_type = "t2.micro"
key_name = "${aws_key_pair.localkey.key_name}"
vpc_security_group_ids = ["${aws_..snip..id}"]
subnet_id = "${aws_subnet.default.id}"
associate_public_ip_address = true
private_ip = "10.0.0.1${var.index}"
…


module "CloudFlareName" {
# modiuficato
execution = var.name_record
source = "../CloudFlare-DNS/CloudFlare-NAME"
aws_public_ip = aws_instance.http-rdir.public_ip
zone_id =  "37ddcc … a106aaa89f3"
index = "1"
dns_name_cf= var.dns_name_aws
dns_subname_cf= var.dns_subname_aws
CDN_PROXY_1 = "false«
}
```

./CloudFlare-DNS/CloudFlare-NAME/main.tf

```
resource "cloudflare_record" "a-name" {
count = var.execution == true ? 1 : 0
name = "${var.dns_subname_cf}"
zone_id  = var.zone_id
value  = var.aws_public_ip
type   = "A"
ttl    = 1
proxied = var.CDN_PROXY_1}
```

Infrastruttura: Layer

Intranet Virtuale

./main.tf

```
module "SpectreCORE" {
source = "./SpectreCORE"
c4_nation = "SYD1"  dns_name = var.dns_name}

module "Store" {
source = "./Store"
spectreC4_ip = module.SpectreCORE.spectreC4_ip
Transport_nation= "TOR1"    }
```

Terraform

Client

Vpn personale

DigitalOcean

Internet

Store - DigitalOcean

C2 – DigitalOcean Teamserver

Target

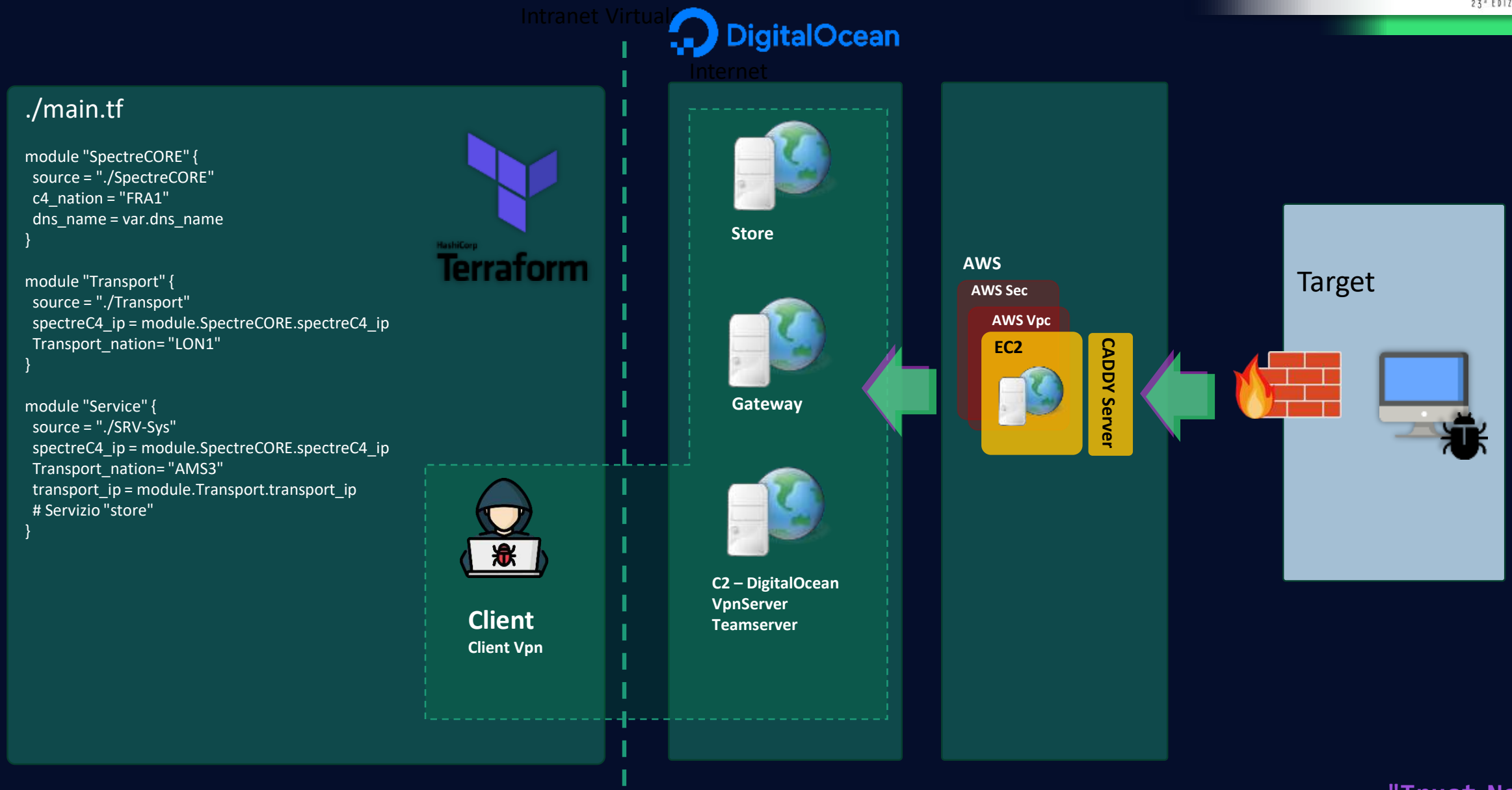
"Trust_No_One"

Terraform: Provisioners [Local-Exec & File]

```
provisioner "local-exec" {
command = <<EOT
        hostname
        rm -rf pki
        rm -rf /etc/openvpn/client
        repo/easy-rsa/easyrsa3/easyrsa init-pki --batch
        EASYRSA_BATCH='yes' repo/easy-rsa/easyrsa3/easyrsa build-ca nopass
        repo/easy-rsa/easyrsa3/easyrsa gen-dh --batch
        openvpn --genkey secret pki/ta.key
        EASYRSA_BATCH='yes' repo/easy-rsa/easyrsa3/easyrsa build-server-full ${digitalocean_droplet.SpectreC4.name}   nopass --batch
        EASYRSA_BATCH='yes' repo/easy-rsa/easyrsa3/easyrsa build-client-full local-server nopass
        repo/easy-rsa/easyrsa3/easyrsa gen-crl
        … snip …
        sed -i 's/openvpnserverplaceholder/${digitalocean_droplet.SpectreC4.ipv4_address}/g'   /etc/openvpn/client/client.conf
        … snip …
        systemctl stop openvpn-client@client.service
        systemctl start openvpn-client@client.service
        sysctl net.ipv4.ip_forward=1
EOT    }
```

```
provisioner "file" {#chiave pubblica CA
        source = "pki/ca.crt"
        destination = "/etc/openvpn/server/ca.crt"
        connection {
                type = "ssh"
                user = "root"
                private_key = file(var.pvt_key)
        }
}
```

Infrastruttura: Redirector

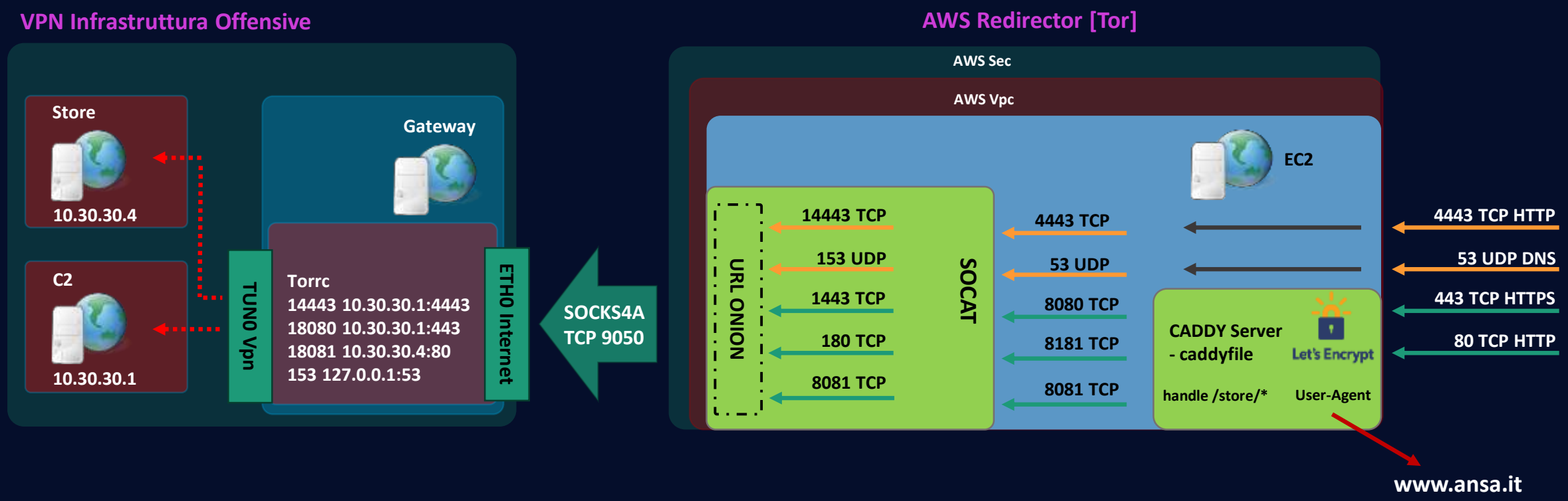


"Trust_No_One"

Terraform + TOR + Socat + Caddy + LetsEncrypt Certificate = ❤️_

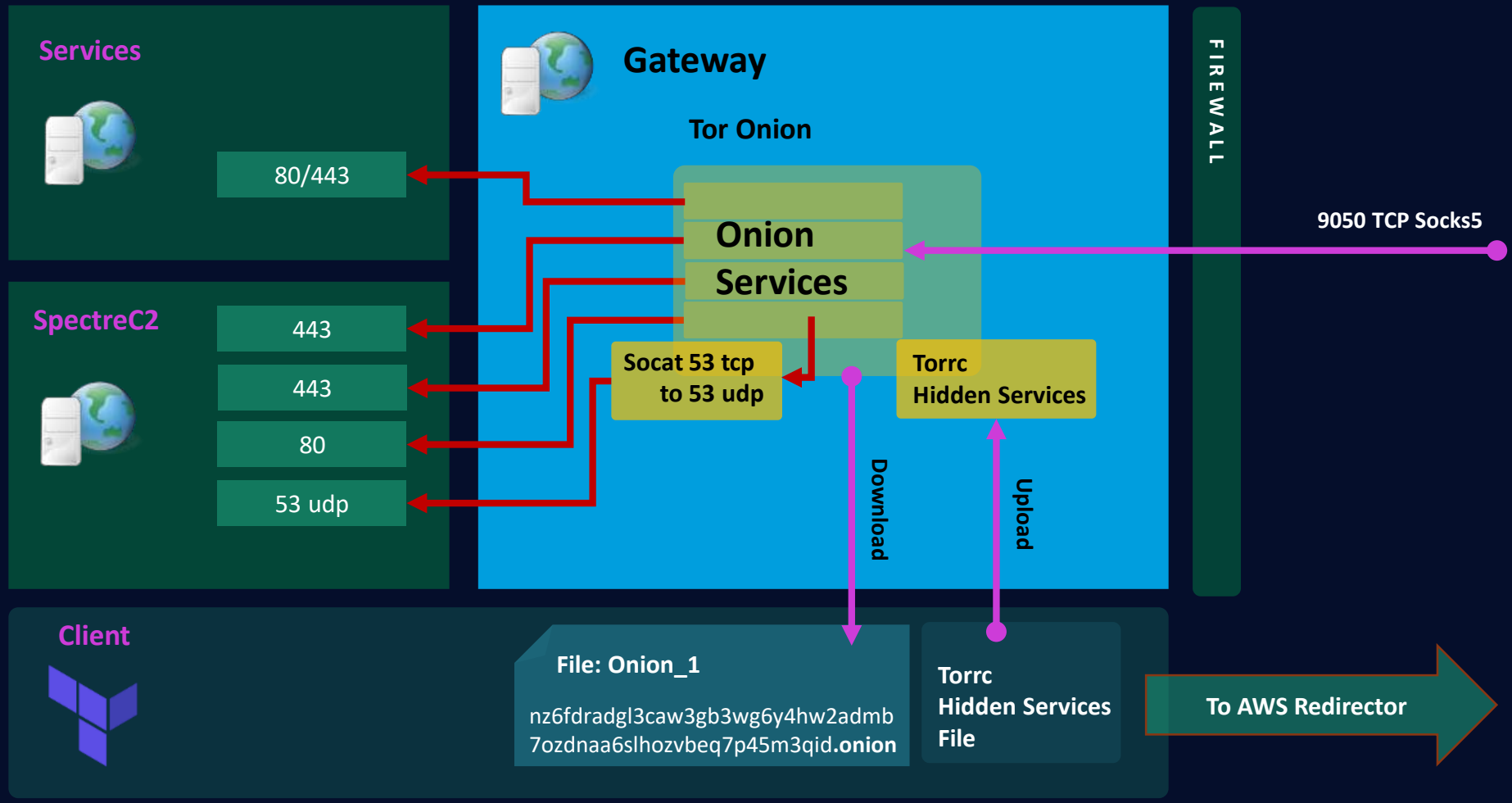

```
module "aws_tor_node" {
  source = "./AWS-Net-Redir"

  transport_ip = module.Transport.transport_ip

  ssh_port = "9443"

  Reflector_zone =  "eu-south-1"
  ami_key = "ami-0a196592622fa712c"
  InstanceType= "t3.micro"

  name_connection = "tor" # tipologia : net o tor

  mail_record = true
  name_record = true
  ns_record = false

  front_execution = true
  front_region = "eu-central-1"

  index ="1"
  dns_name_aws= "sky-warp.net"
  dns_subname_aws = "tor-${module.Rand_Tor.random_lowercase_string}"
  dns_redirsite_aws = "www.repubblica.it"
  useragent = "Mozilla/5.0 (Windows NT 6.3.1; Trident/7.0; rv:11.1.0) like Gecko"
}
```

Terraform: TOR Connector

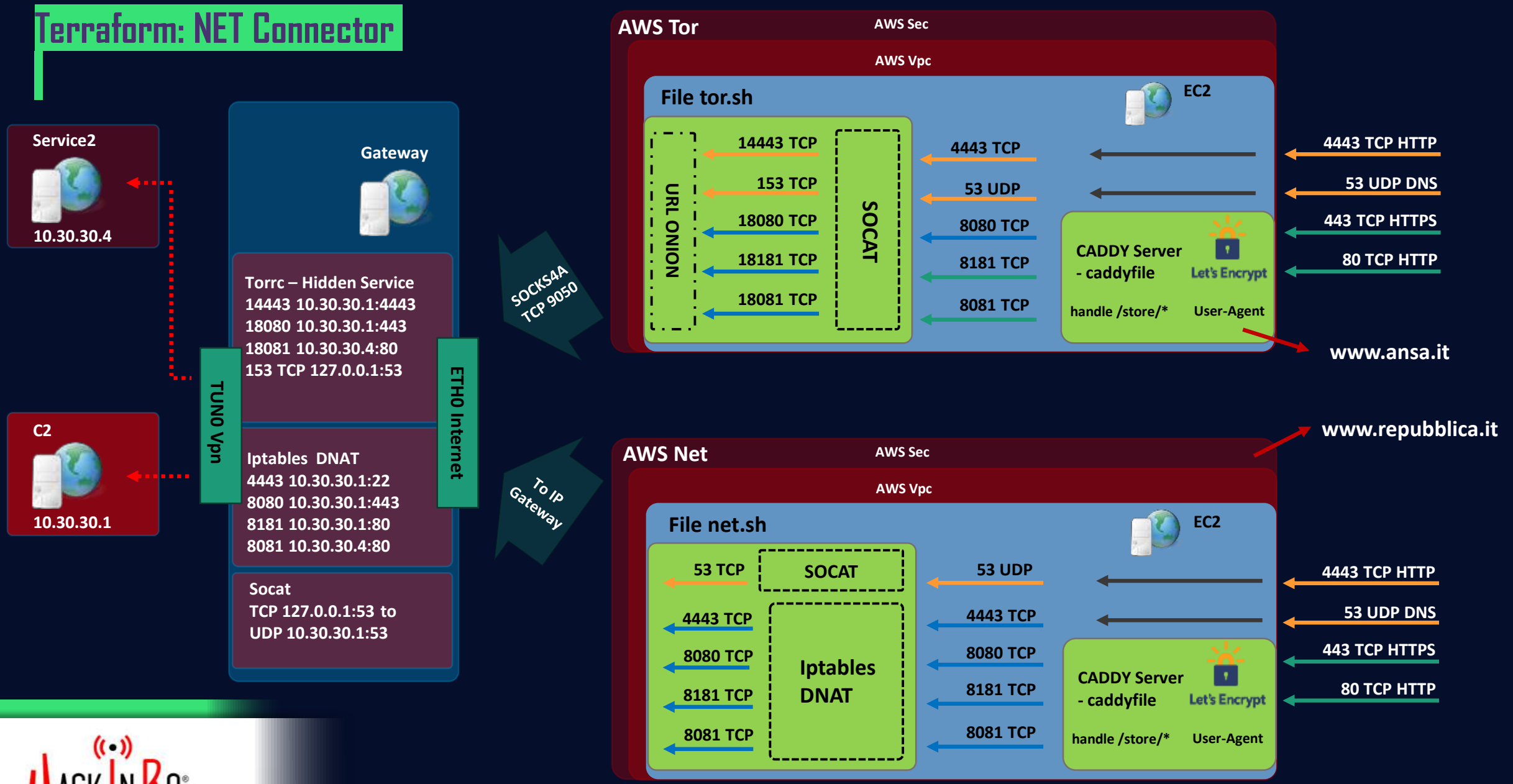


"Trust_No_One"

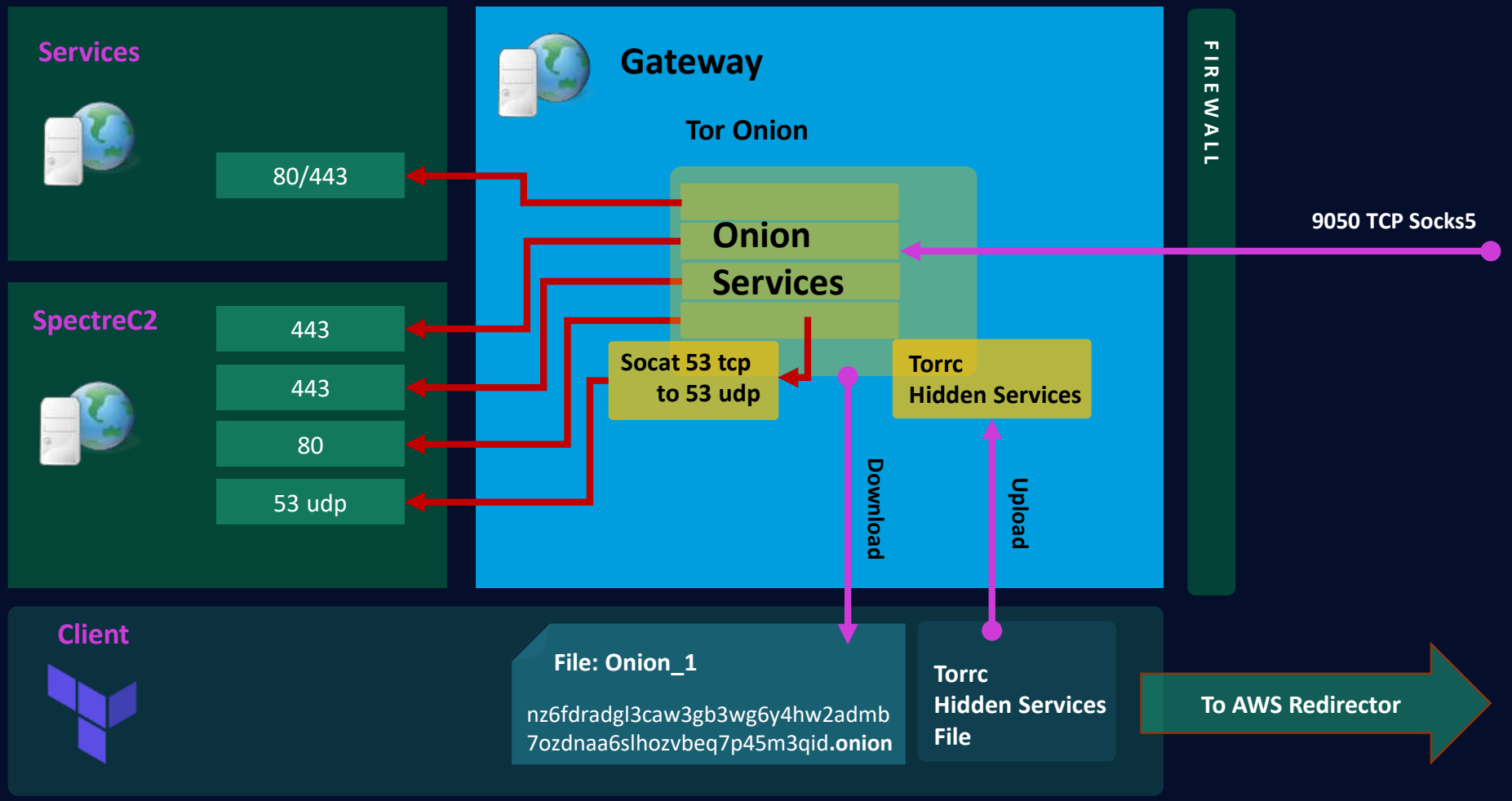
Terraform: TOR Connector



Terraform: NET Connector



Terraform: NET Connector



Terraform: Caddy Redirector

```
resource "aws_instance" "http-rdir" {
        provider = aws.region
        ami = var.ami_key
        instance_type = var.InstanceType
        key_name = "${aws_key_pair.localkey.key_name}"
        vpc_security_group_ids = ["${aws_security_group.http-redirector-security.id}"]
        subnet_id = "${aws_subnet.default.id}"
        associate_public_ip_address = true
        private_ip = "10.0.0.1${var.index}"

provisioner "file" {
        source = "repo/Redirector/Caddyfile"
        … snip … } }

provisioner "remote-exec" {
    inline = [
        "echo ${var.transport_ip}",
        … snip …
        "sudo apt install -y apt-transport-https",
        … snip …
        "sudo apt install -y tor deb.torproject.org-keyring",
        "sudo apt install -y tor",
        "sudo apt install -y daemonize",
        "sudo apt-get update",
        "sudo apt install -y socat",
        "sudo apt install -y daemonize",
        "sudo apt install -y tor",
        "sudo cp /tmp/caddy.service /etc/systemd/system/caddy.service",
        "echo \"sleep 30s\" > socat.sh"
        ]
```

```
{       admin off
        acme_ca https[:]//acme-v02.api.letsencrypt.org/directory [TEST]
        # https://acme-staging-v02.api.letsencrypt.org/directory        }

# file: https[:]//github.com/rsmudge/Malleable-C2-Profiles  webbug_getonly.profile
# User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 … etc …

#####.##-##-## {
  log
  @user-agent-whitelist {
    header User-Agent "MYUSERAGENT"
  }

        header {
                -Server
                 +X-Robots-Tag "noindex, nofollow, nosnippet, noarchive"
                +X-Content-Type-Options "nosniff"}

        handle /store/* {
        reverse_proxy @user-agent-whitelist http://127.0.0.1:8081 {}}

        handle /* {
                reverse_proxy @user-agent-whitelist https://127.0.0.1:8080 {
                transport http {tls_insecure_skip_verify}}}

        handle /* {
                reverse_proxy https://---------- {
                header_up Host "----------"}}}
```

Terraform: CloudFlare

HACK IN BO Winter 2024 Edition 23ª EDIZIONE

./Spectre/main.tf

```
module "aws_net_node" {
  source = "./AWS-Net-Redir"
  // Altri parametri…
  mail_record = false
  name_record = true
  ns_record = true
  // Altri parametri...
}
module "aws_tor_node" {
  source = "./AWS-Net-Redir"
  // Altri parametri…
  mail_record = true
  name_record = true
  ns_record = false
  // Altri parametri...
}
```

«Richiama la creazione di 2 istanze AWS-Net-Redir»

./Spectre/AWS-Net-Redir/main.tf

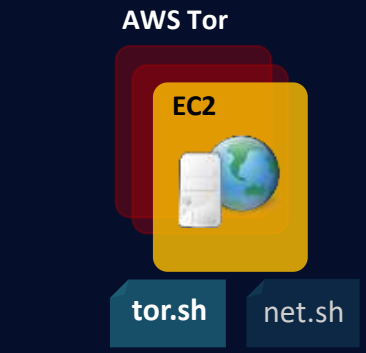
```
resource "aws_instance" "http-rdir" {
 //Codice Istanza EC2 AWS
}
module "CloudFlareName" {
  execution = var.name_record
  source = "../CFL-Sys/CloudFlare-NAME"
  // Codice
  dns_name_cf= var.dns_name_aws
  dns_subname_cf= var.dns_subname_aws
}
module "CloudFlareMail" {
  execution = var.mail_record
  source = "../CFL-Sys/CloudFlare-Mail"
  // Codice
  mail_record_cf = "mx"
  dns_name_cf= var.dns_name_aws
  dns_subname_cf= var.dns_subname_aws
}
module "CloudFlareNS" {
 execution = var.ns_record
  source = "../CFL-Sys/CloudFlare-NS"
  // Codice
  ns_record_cf = "nameserver"
  dns_name_cf= var.dns_name_aws
  dns_subname_cf= var.dns_subname_aws
}
```

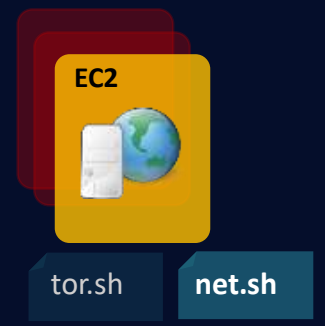
./Spectre/CFL-Sys/Clouflare-Name

```
resource "cloudflare_record" "a-name" {
count = var.execution == true ? 1 : 0
    name = "${var.dns_subname_cf}"
    zone_id  = var.zone_id
    value  = var.aws_public_ip
    type   = "A"
    ttl    = 1
    proxied = var.CDN_PROXY_1
}
```

./Spectre/CFL-Sys/Clouflare-NS

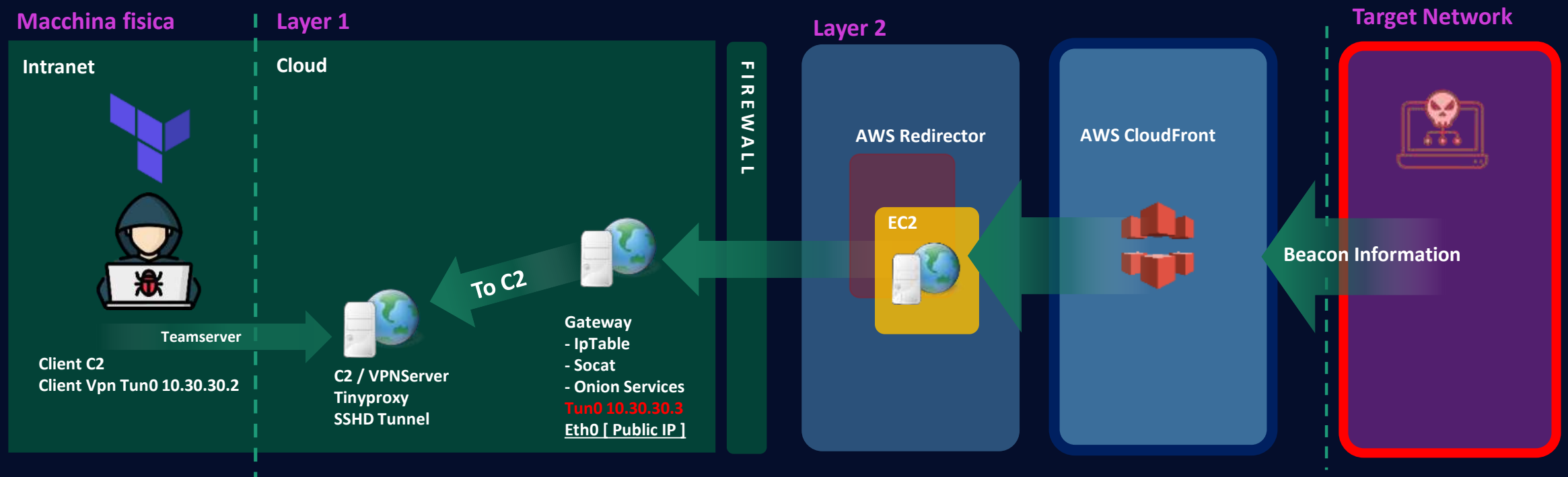
```
resource "cloudflare_record" "dns-ns1" {
count = var.execution == true ? 1 : 0
    zone_id  = var.zone_id
    name   = "ns1"
    value  = var.aws_public_ip
    type   = "A"
    ttl    = 300
}

resource "cloudflare_record" "dns-a" {
count = var.execution == true ? 1 : 0
    zone_id  = var.zone_id
    name   = var.ns_record_cf
    value  = "ns1.${var.dns_name_cf}"
    type   = "NS"
    ttl    = 300
}
```





"Trust_No_One"

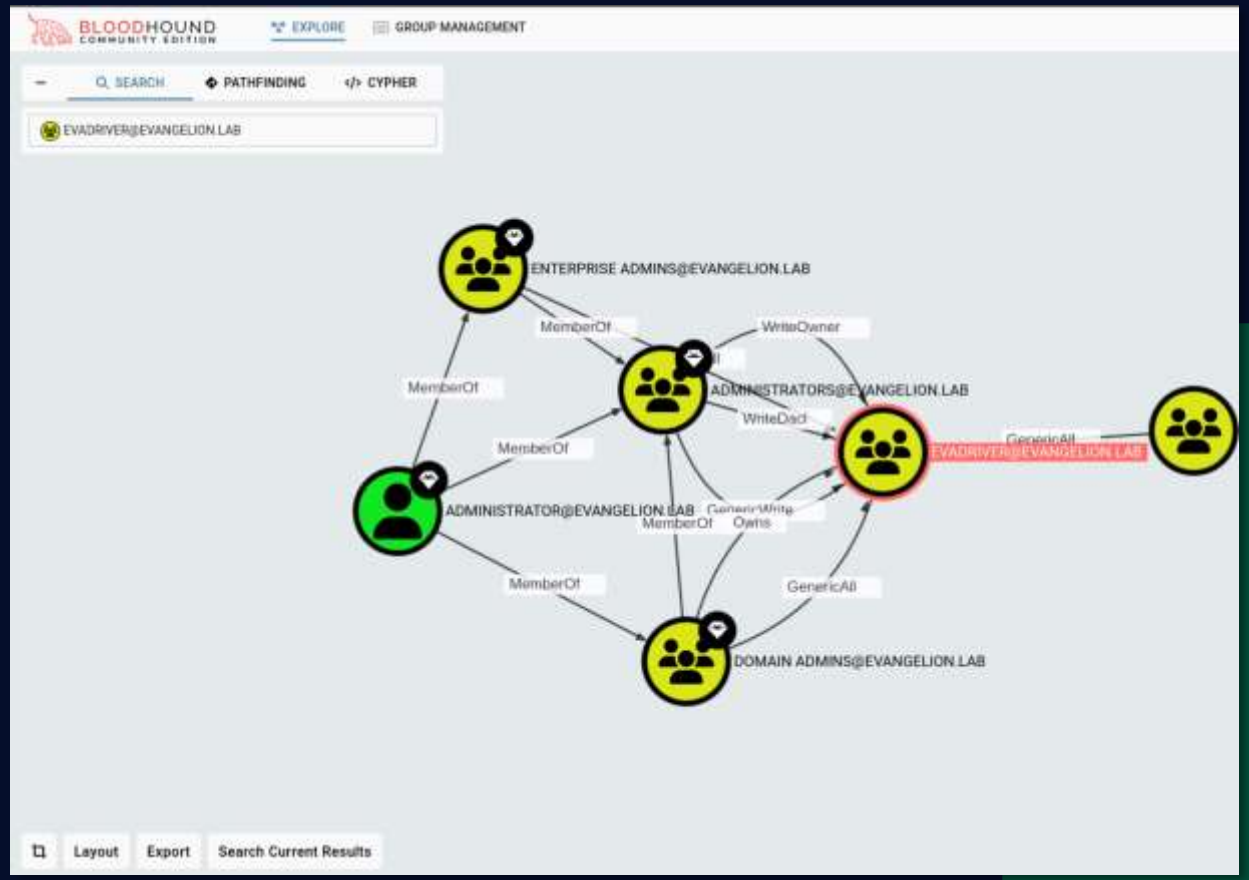
Terraform: CloudFront

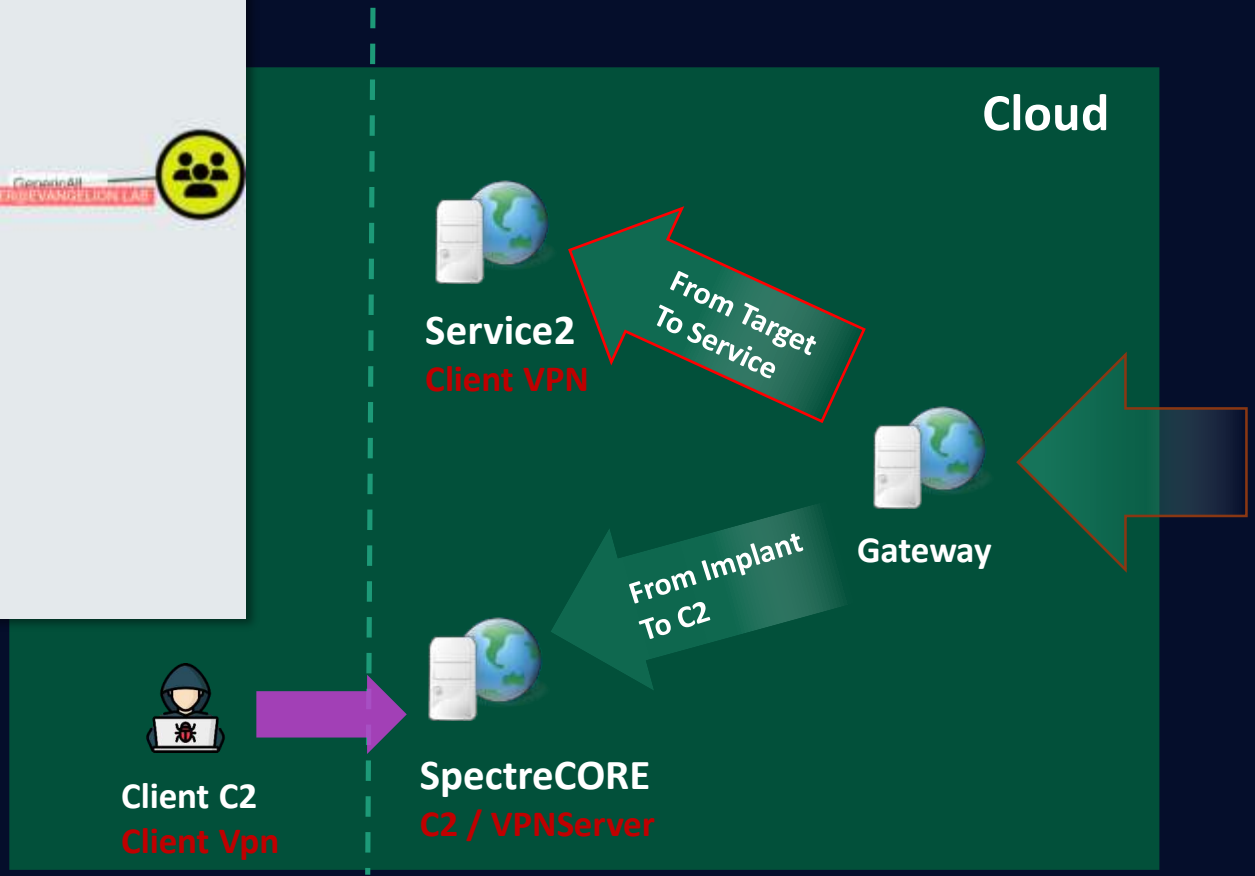


Terraform: CloudFront


```
resource "aws_cloudfront_distribution" "example_distribution" {
count = var.execution == true ? 1 : 0
provider = aws.region
origin {
        domain_name = "${var.dns_subname_aws}.${var.dns_name_aws}" # Il tuo server HTTP/HTTPS
        origin_id   = "custom-example-com"
        custom_origin_config {
        http_port             = 80  # Porta del tuo server HTTP, usa 443 per HTTPS
        https_port            = 443 # Porta del tuo server HTTPS
        origin_protocol_policy = "match-viewer" # Può essere "http-only", "https-only", "match-viewer"
        origin_ssl_protocols   = ["TLSv1.2"] # Protocolli SSL supportati
   }
 }
enabled = true
default_cache_behavior {
        allowed_methods  = ["DELETE", "GET", "HEAD", "OPTIONS", "PATCH", "POST", "PUT"]
        cached_methods   = ["GET", "HEAD"]
        target_origin_id = "custom-example-com"
        cache_policy_id = "4135ea2d-6df8-44a3-9df3-4b5a84be39ad"
        origin_request_policy_id = "acba4595-bd28-49b8-b9fe-13317c0390fa"
        viewer_protocol_policy = "redirect-to-https" # Forza HTTPS
        min_ttl               = 0
        default_ttl           = 3600
        max_ttl               = 86400
}
 restrictions {geo_restriction {restriction_type = "none"}}
 viewer_certificate {cloudfront_default_certificate = true}
}
```

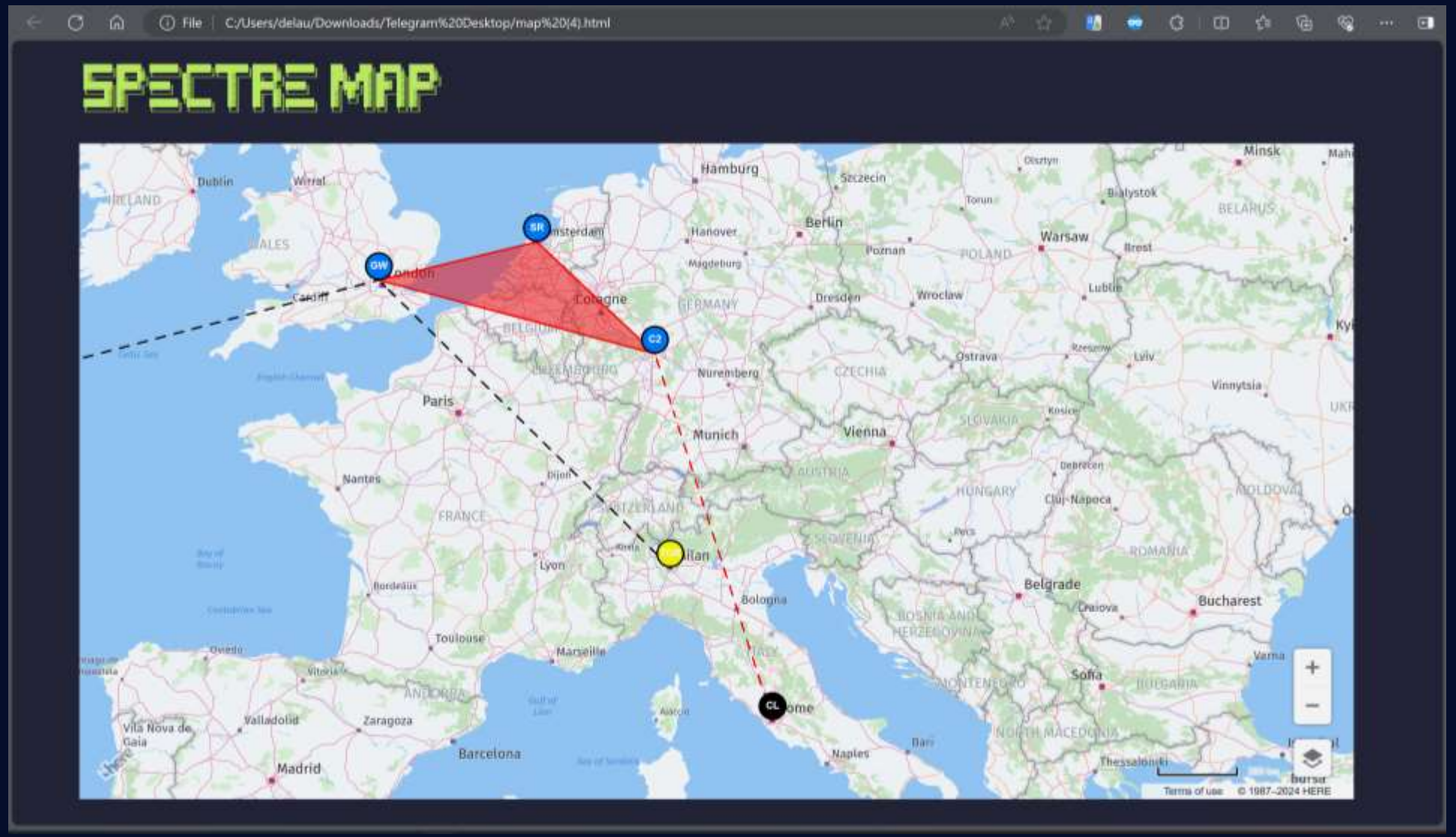
Terraform: Scriptare Bloodhound CE (Docker)





"Trust_No_One"

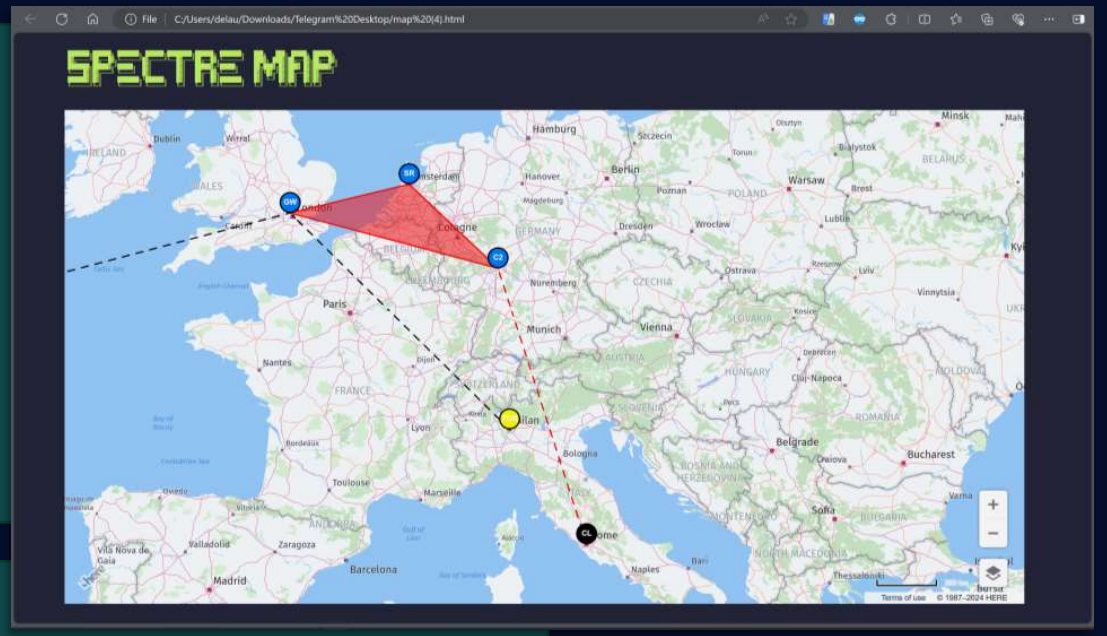
Terraform: TFSTATE $ Maps



"Trust_No_One"

Terraform: TFSTATE & Maps

```
resource "null_resource" "geo" {  # Imposta dipendenza dall'istanza AWS
provisioner "local-exec" {
command = <<-EOT
        apt -y install jq ;
        curl -k  'https://api.ipgeolocation.io/ipgeo?apiKey=${var.geo_api}' | jq -r  '.country_name, .city, ….
        curl -k  'https://api.ipgeolocation.io/ipgeo?apiKey=${var.geo_api}&ip=${var.geo_transport_ip}'
        …
        sed -i "s/sr-lon/$(sed -n '6p' ./repo/Here-Maps/cache/service.txt)/g" ./repo/Here-Maps/map.html ;
        sed -i "s/cl-lat/$(sed -n '5p' ./repo/Here-Maps/cache/my.txt)/g" ./repo/Here-Maps/map.html ;
        sed -i "s/cl-lon/$(sed -n '6p' ./repo/Here-Maps/cache/my.txt)/g" ./repo/Here-Maps/map.html ;
…
…
EOT  }
```




```
var gwMarker = createMarker({lat:gw-lat, lng:gw-lon}, "GW", "#007bff"); // Italy
var c2Marker = createMarker({lat:c2-lat, lng:c2-lon}, "C2", "#007bff"); // London
var srMarker = createMarker({lat:sr-lat, lng:sr-lon}, "SR", "#007bff"); // Netherlands
…
drawDashedLine(torMarker.getGeometry(), gwMarker.getGeometry(), 'black');
drawDashedLine(netMarker.getGeometry(), gwMarker.getGeometry(), 'black');
drawDashedLine(clMarker.getGeometry(), c2Marker.getGeometry(), 'red');
drawPolygon();
```

HACK IN BO
Winter 2024 Edition
23ª EDIZIONE

DOMANDE ?