INFRAESTRUCTURA OFENSIVA

DISCLAIMER

PENTESTING != RED TEAM ENGAGEMENT

Sin embargo, la idea de la charla es armar una infraestructura flexible que sirva para distintos escenarios.

RESUMEN

1- Que? Definir requerimientos

2- Threat Model

3- Propuestas de infraestructura

4- Provisioning y Configuracion

5- Demo

QUE?

Diseño de la Infraestructura

- Que hacemos y que necesitamos?
  - Vamos a simular un adversario real, lo mas similar a un APT posible con nuestros recursos.
  - Vamos a diseñar una infraestructura con los requerimientos que evaluemos del mapeo/modelado de ataques reales

Diseño de la Infraestructura

- Como modelamos/mapeamos el ataque?
  - mitre ATT&CK
  - la **CYBER** kill chain

Diseño de la Infraestructura

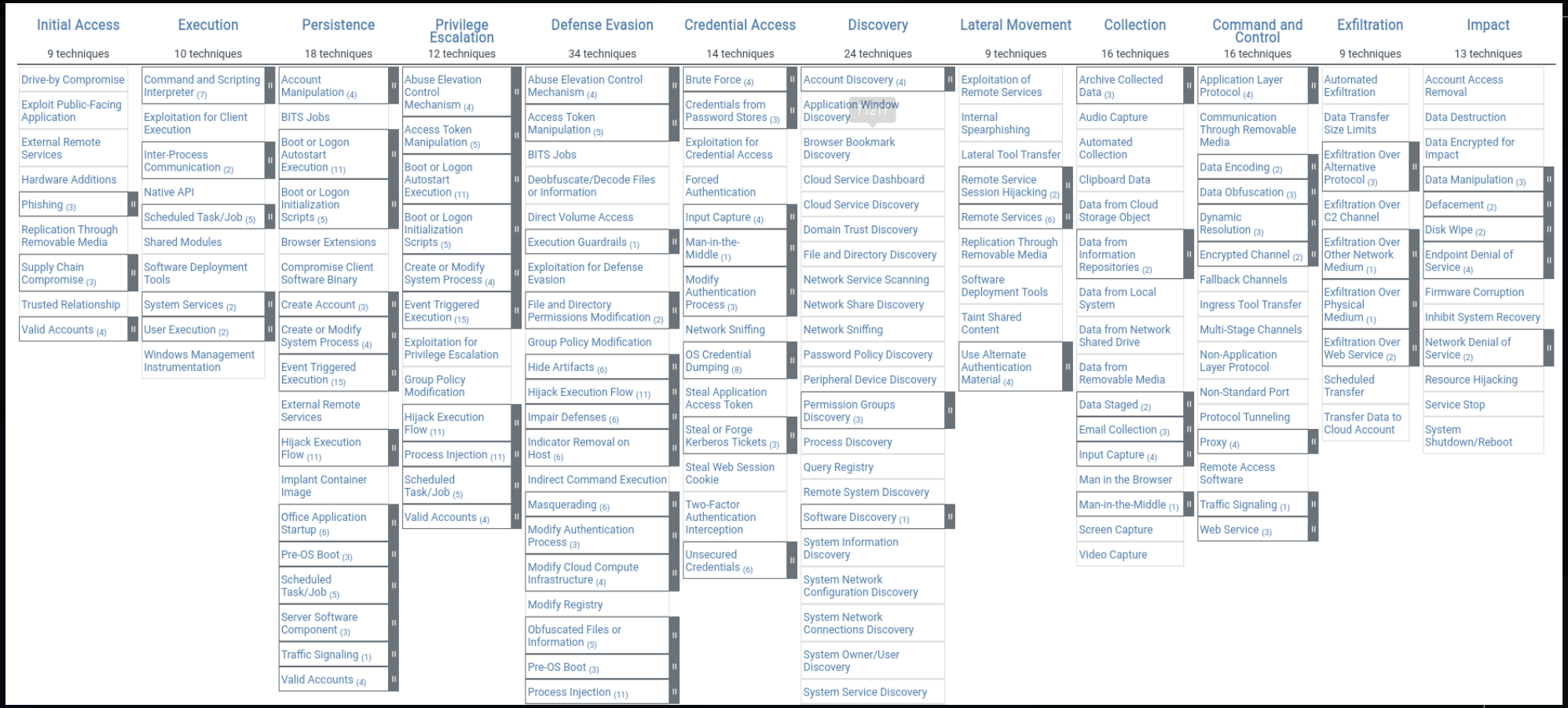
- MITRE ATTA&CK

  Es un conjunto de Tacticas y Tecnicas basadas en ataques reales, ej:

  - Tactica(TA0001): acceso inicial
  - Tecnica(T1190): explotar un servicio/app publico

  https://attack.mitre.org/

Mitre ATT&CK matrix



CYBER kill chain

Mapeando un Ataque

- Acceso Inicial
  - Exploit Public Facing App (T1190)
  - Phishing(T1566)
- Ejecucion
  - Powershell(T1059.001)
- Persistencia
  - Logon Script(T1037.001)

Mapeando un Ataque

- Escalar Privilegios
  - Access Token Manipulation (T1134)
- Defense Evasion
  - Trusted Developer Utilities, MSBuild (T1127.001)
- Discovery
  - Network Service Scanning (T1046)

Mapeando un Ataque

- Movimiento Lateral
  - RDP(T1021.001)
- Collection
  - Email Collection (TT1114)
- Command and Control
  - Proxy(T1090)

Requerimientos

- Poder hacer recon y phishing
- Poder explotar servicios publicos
- Que sea resiliente
- Recibir conexiones reversas
- C2 con proxies
- Payload delivery
- Poder establecer persistencia(ex: phone home)
- Storage
- Almacenar logs y timestamps para comunicarlos al blue team(No es un ctf !)
- Tunelear multiples protocolos

Threat Model

Threat Model tl;dr

- Identificar nuestros Assets(c2 location?)
- Identificar adversario/a(blue team maybe?)
- Identificar los recursos del adversario/a
- Identificar nuestros recursos

Team Server

- Es desde donde vamos a realizar todas las operaciones y donde vamos a configurar los distintos Command and Control.
- Vamos a usar Kubernetes!

Command and Control

Command and Control

- Vamos a separarlos en funcion a cada cuanto los agentes se conectan y en funcion a que protocolos usan

C2 Long y Short Haul

- Long Haul: los agentes se conectan al servidor una vez al dia/ cada x dias o por semana, este layer lo usamos para garantizar persistencia
- Short Haul: Los agentes se conectan al servidor cada 1/5 minutos, es desde donde podemos operar

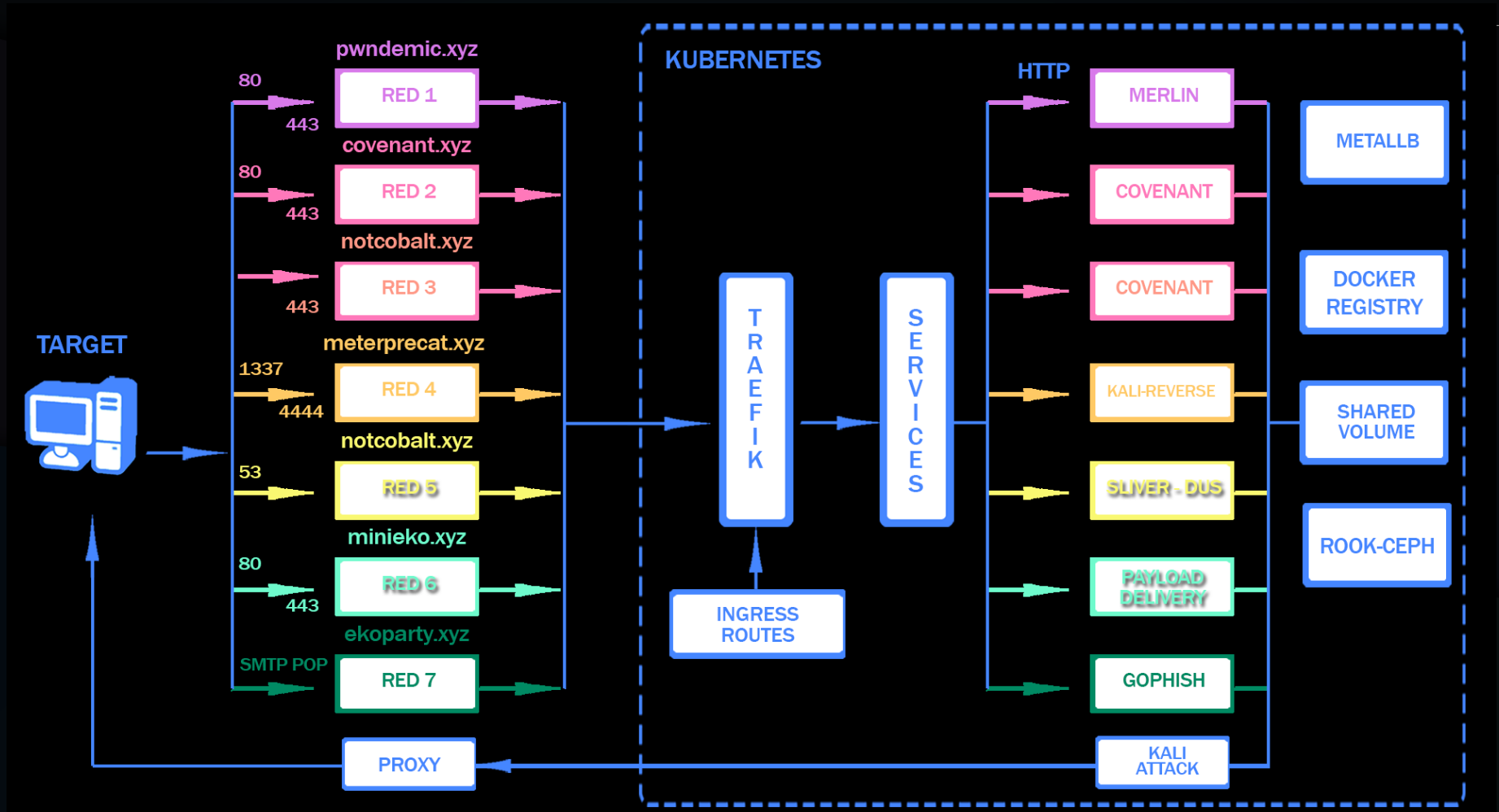
C2 Protocolos

- TCP: old school reverse shells, meterpreter/netcat
- DNS: Vamos a usar sliver para tener reverse shells por DNS.
- HTTP/HTTPS: vamos a usar merlin/covenant como C2s https

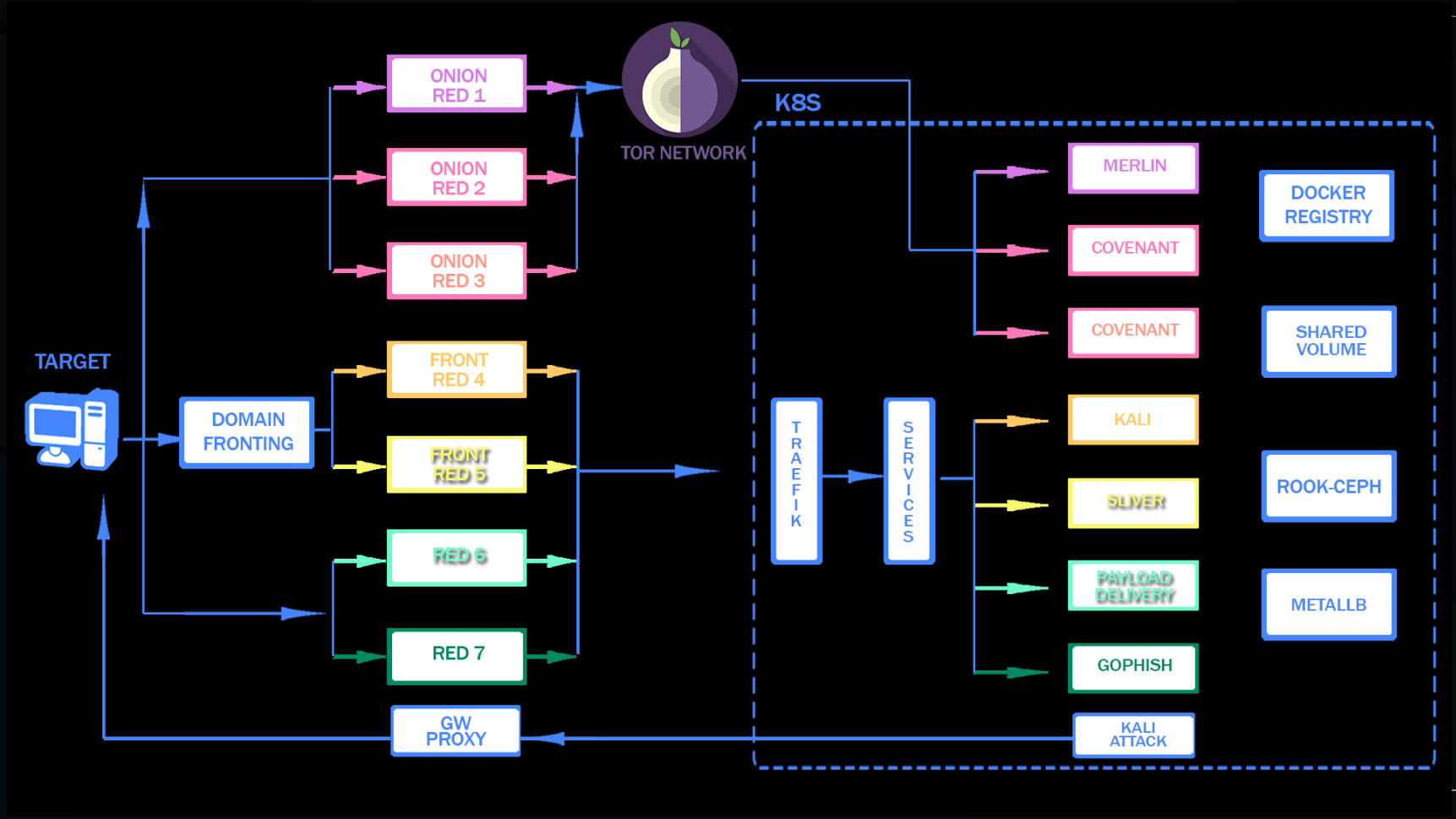
PHISHING

Vamos a usar gophish adentro del cluster y configurar “docker-mailserver” de tomav en un redirector con su propio dominio

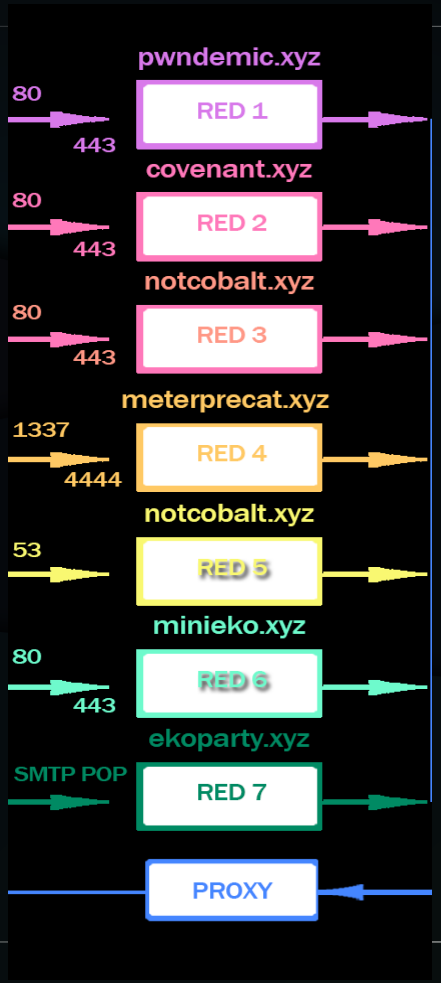
Infraestructura Propuesta 1



Infraestructura Propuesta 2



Redirectors



Redirectors

Son nuestros fronts, se encargan de redirigir el trafico para ofuscar la ubicación real de nuestra infra.

- Dumb pipe: Los vamos a usar para las reverse shells y DNS
- HTTPS: vamos a usarlo para payloads,phishing y los C2 que van por https

Redirectors

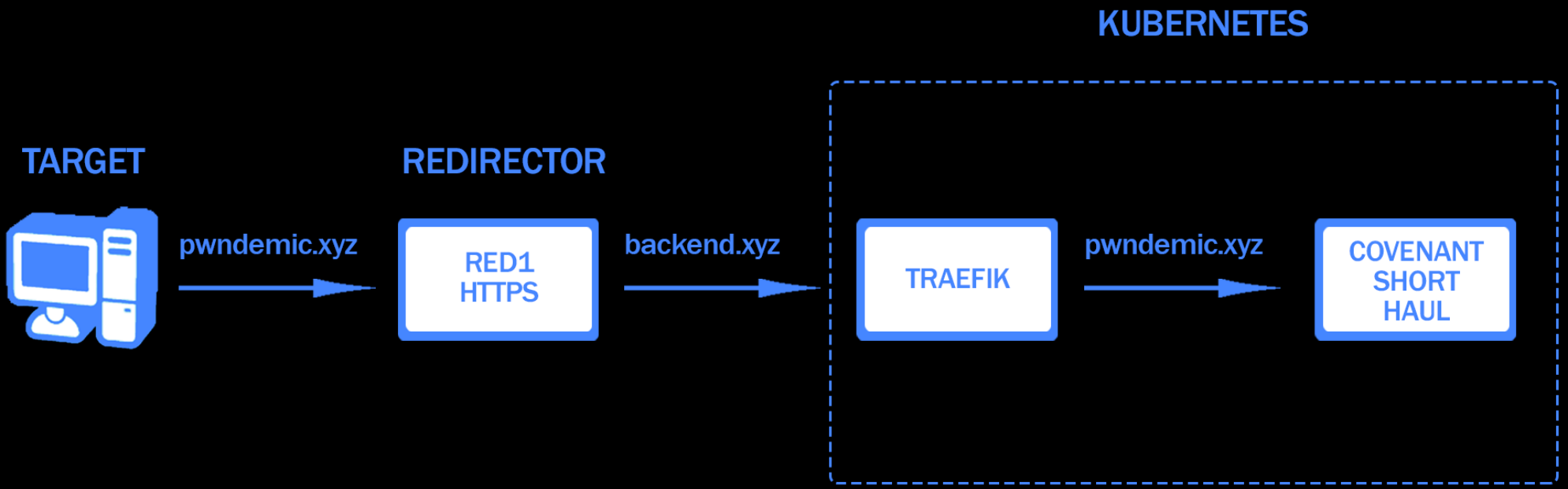
Dumb Pipe

```
socat TCP4-LISTEN:4444,fork TCP4:135.15.15.15:4444
```

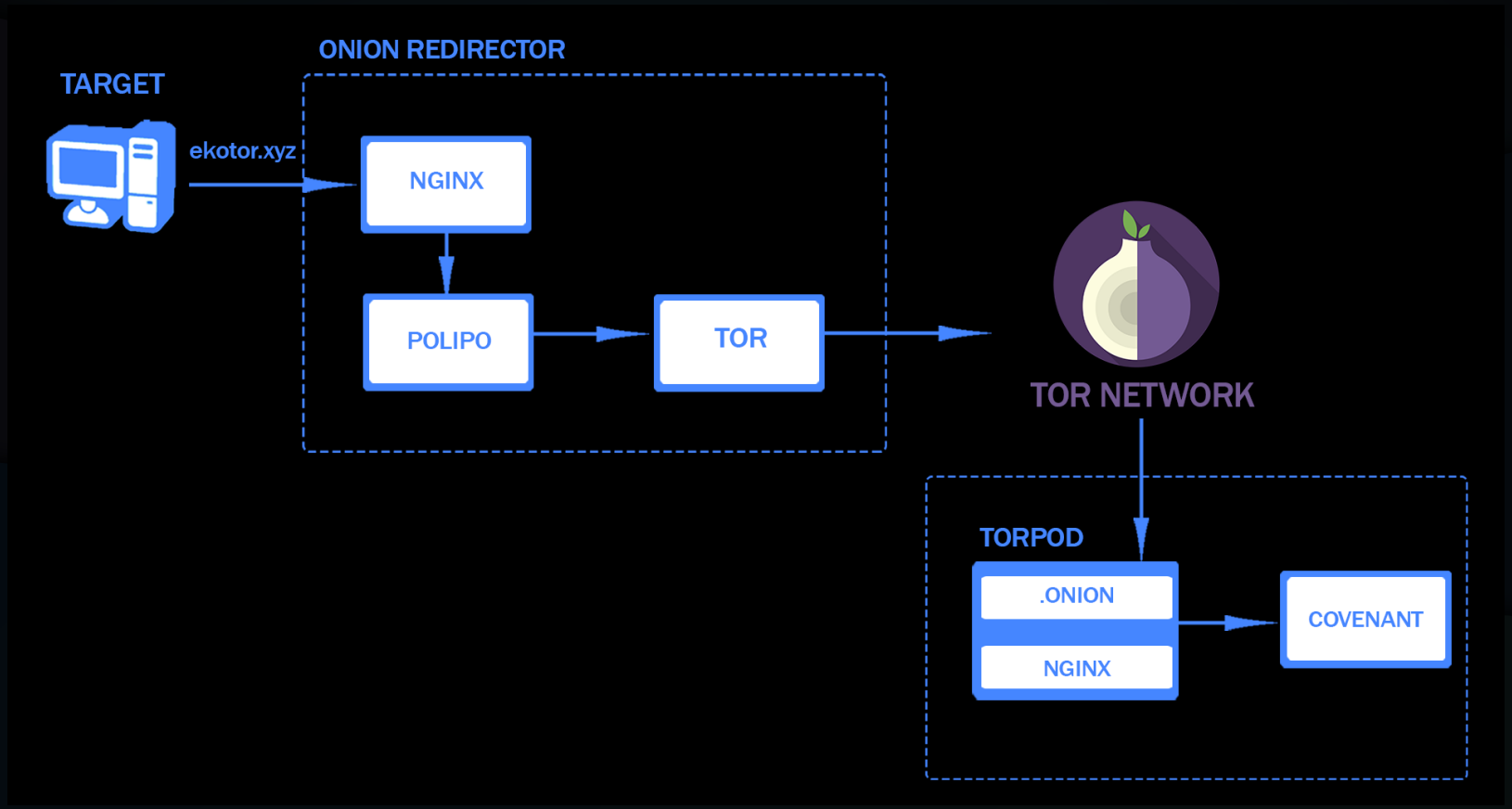
HTTPS

```
<VirtualHost *:443>
ServerName pwndemic.xyz
# Enable SSL
SSLEngine On
# Trust Self-Signed Certificates generated by Cobalt Strike
SSLProxyVerify none
SSLProxyCheckPeerCN off
SSLProxyCheckPeerName off

ProxyPreserveHost On
ProxyRequests Off
SSLCertificateFile      /etc/letsencrypt/live/pwndemic.xyz/cert.pem
SSLCertificateKeyFile   /etc/letsencrypt/live/pwndemic.xyz/privkey.pem
SSLCertificateChainFile /etc/letsencrypt/live/pwndemic.xyz/chain.pem
# Enable Proxy
SSLProxyEngine On
  ProxyPass /  https://backend.xyz/
  ProxyPassReverse /  https://backend.xyz/
</VirtualHost>
```

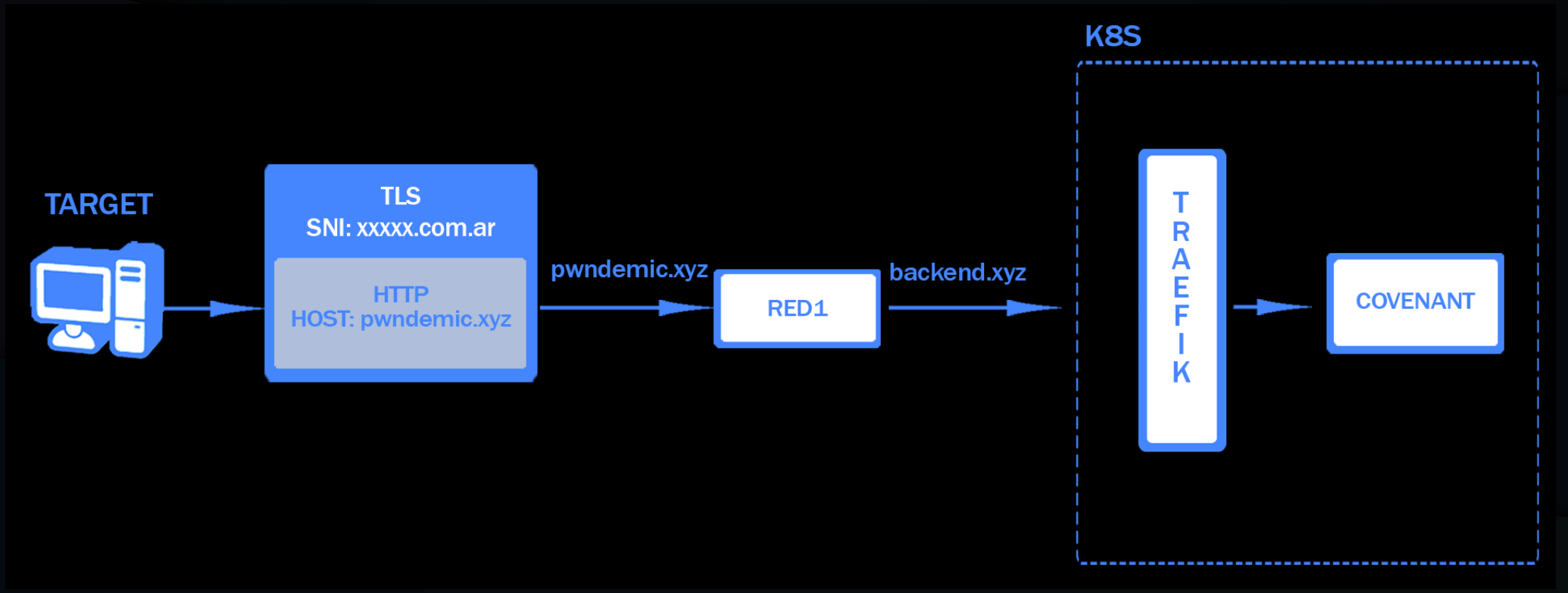
CASO 1


KUBERNETES
TARGET
REDIRECTOR
pwndemic.xyz
RED1
HTTPS
backend.xyz
TRAEFIK
pwndemic.xyz
COVENANT
SHORT
HAUL

Caso 2 .onion


ONION REDIRECTOR
TARGET
ekotor.xyz
NGINX
POLIPO
TOR
TOR NETWORK
TORPOD
.ONION
NGINX
COVENANT

Caso 3 domain fronting



2- PROVISIONING Y CONFIGURACION

STACK Baremetal

- Provisioning:
  - kubespray (K8s+ansible)
- Kubernetes:
  - Metallb (Baremetal load balancer)
  - Traefik (Reverse Proxy/LB)
  - Calico (CNI)
  - rook-ceph(Storage)
  - Docker (Containers runtime)

STACK Redirectors

- Provisioning/configuration:
  - Ansible
  - Terraform
- Tools:
  - Docker
  - Mailserver
  - Apache
  - Socat
  - Certbot (letsencrypt)

STACK Conexion

- En caso que tu threat model lo amerite lo importante es conectarse al bastion a travez de Tor. Me gustan 2 opciones

  1- tor+ ssh con proxychains

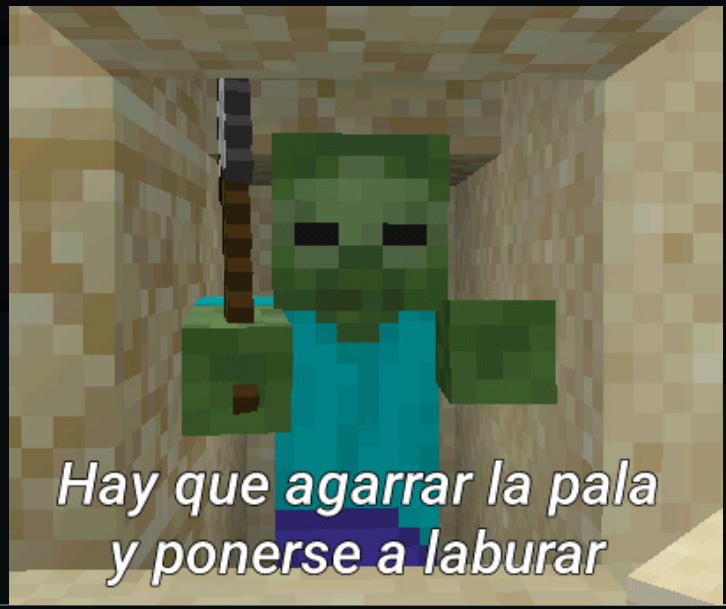
  2- whonix + VM

Mas info:

https://scottlinux.com/2015/09/01/use-kali-linux-through-tor-with-whonix-gateway/

https://medium.com/swlh/proxying-like-a-pro-cccdc177b081

Deployando y Configurando



Repositorios

https://github.com/notchxor/infraestructura-ofensiva-workshop

https://github.com/notchxor/infraestructura-ofensiva-cluster

https://github.com/notchxor/infraestructura-ofensiva-infra

Deployando kubernetes

Deployando kubernetes

PASOS:

- Deployar cluster
- Configurar firewall
- Deployar Metallb
- Deployar rook-ceph
- Deployar traefik
- Deployar docker-registry

Deployando kubernetes

offensive-cluster/

Vamos a editar /group_vars/k8s-cluster/k8s-cluster.yml

```
## Change this to use another Kubernetes version,
kube_version: v1.18.8
```

```
kube_network_plugin: calico
```

Deployando kubernetes

Correr el playbook:

```
root@molly:/home/mini/eko2020/offensive-cluster# ansible-playbook --private-key=/home/mini/.ssh/anoyo -i inventory/offensive-cluster/hosts.yaml  --become --become-user=root cluster.yml

PLAY [localhost] ***************************************************************************************************************************************************************************************
Tuesday 22 September 2020  18:49:22 -0300 (0:00:00.088)       0:00:00.088 *****

TASK [Check ansible version >=2.8.0] *******************************************************************************************************************************************************************
ok: [localhost] => {
    "changed": false,
    "msg": "All assertions passed"
}

PLAY [all] *********************************************************************************************************************************************************************************************
Tuesday 22 September 2020  18:49:22 -0300 (0:00:00.035)       0:00:00.124 *****

TASK [Set up proxy environment] ************************************************************************************************************************************************************************
ok: [node1]
```

Deployando kubernetes

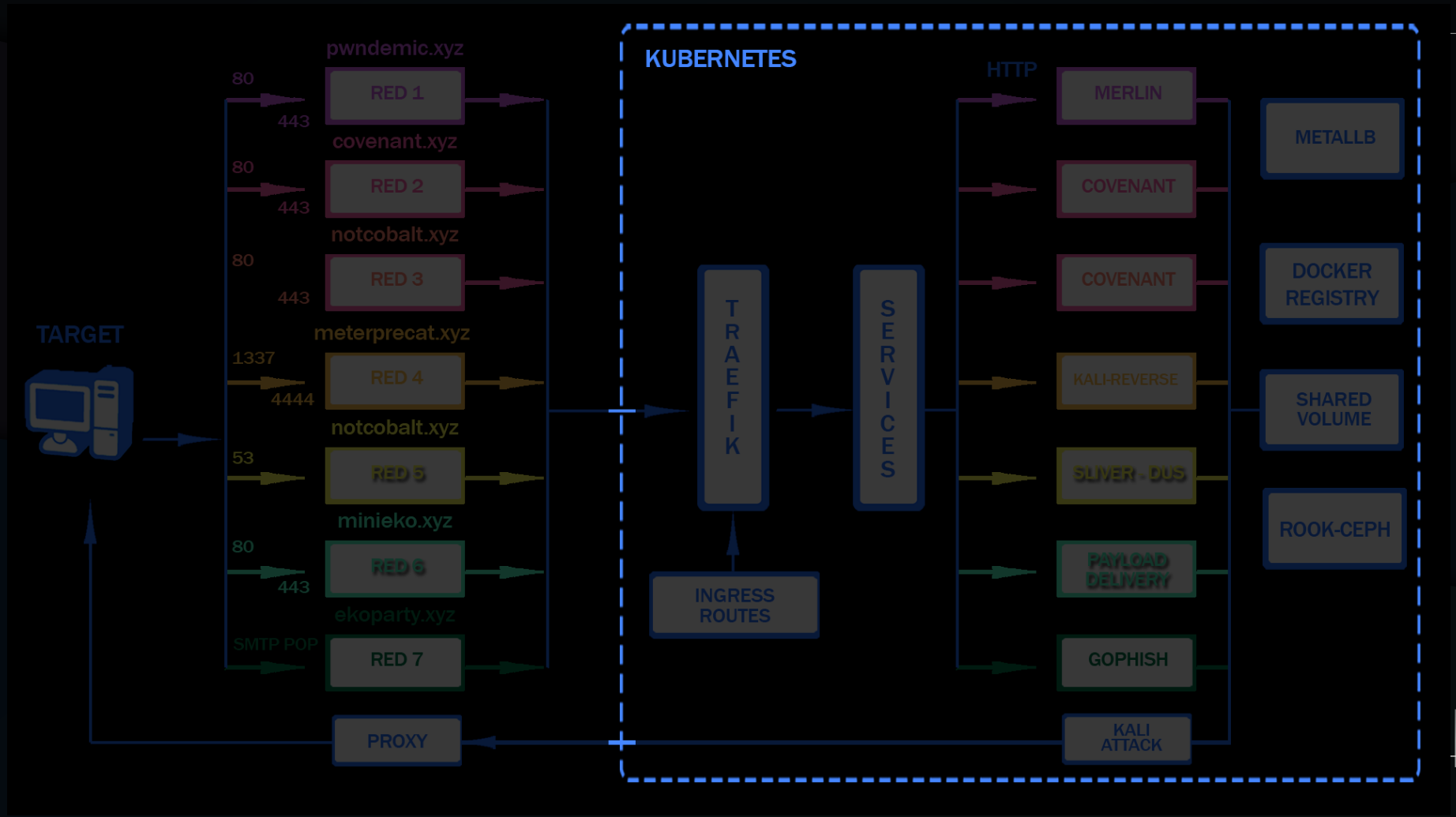
FIREWALL RULES:

- ufw allow from <redirector_ip>
- ufw allow from 10.233.0.0/16 (red interna, config en kubespray)
- Vim /etc/default/ufw

```
# Set the default input policy to ACCEPT, DROP, or REJECT. Please note that if
# you change this you will most likely want to adjust your rules.
DEFAULT_INPUT_POLICY="DROP"

# Set the default output policy to ACCEPT, DROP, or REJECT. Please note that if
# you change this you will most likely want to adjust your rules.
DEFAULT_OUTPUT_POLICY="ACCEPT"

# Set the default forward policy to ACCEPT, DROP or REJECT.  Please note that
# if you change this you will most likely want to adjust your rules
DEFAULT_FORWARD_POLICY="ACCEPT"

# Set the default application policy to ACCEPT, DROP, REJECT or SKIP. Please
# note that setting this to ACCEPT may be a security risk. See 'man ufw' for
# details
DEFAULT_APPLICATION_POLICY="SKIP"
```

Deployando kubernetes

USTED ESTA AQUI



Configurando kubernetes

Disclaimer: esto no es un workshop sobre buenas practicas de kubernetes y hardening, vamos a hacer lo minimo.

Acceso:

- Copiar config

```
root@mini:~# scp node1:/root/.kube/config  /root/eko2020/admin.kubeconfig
```

- Crear un alias en ~/.bashrc:

```
alias ko='kubectl –kubeconfig=/root/eko2020/admin.kubeconfig'
```

- ssh tunneling:

```
ssh -L 6443:127.0.0.1:6443 <TUSERVER>
```

Configurando kubernetes

Verificar:

```
mini@molly:~$ ko get pods -n kube-system
NAME                                       READY   STATUS
calico-kube-controllers-848ff59cd4-zmmm5   1/1     Running
calico-node-t227w                          1/1     Running
coredns-59dcc4799b-gnpn5                   1/1     Running
coredns-59dcc4799b-rxcgx                   0/1     Pending
dns-autoscaler-66498f5c5f-69lfl            1/1     Running
kube-apiserver-node1                       1/1     Running
kube-controller-manager-node1              1/1     Running
kube-proxy-tbcjl                           1/1     Running
kube-scheduler-node1                       1/1     Running
kubernetes-dashboard-57777fbdcb-jtlcr      1/1     Running
kubernetes-metrics-scraper-54fbb4d595-z6sxm   1/1     Running
nodelocaldns-7hmgj                         1/1     Running
```

Configurando kubernetes

Repaso MUY general

Kubectl: cli que nos deja interactuar con la API de kubernetes

Yaml: archivos de configuracion para los distintos resources del cluster

Pods: grupo de 1 o mas containers, tienen almacenamiento EFIMERO y acceso a la red

Servicios: es una abstraccion para exponer pods como servicios del cluster

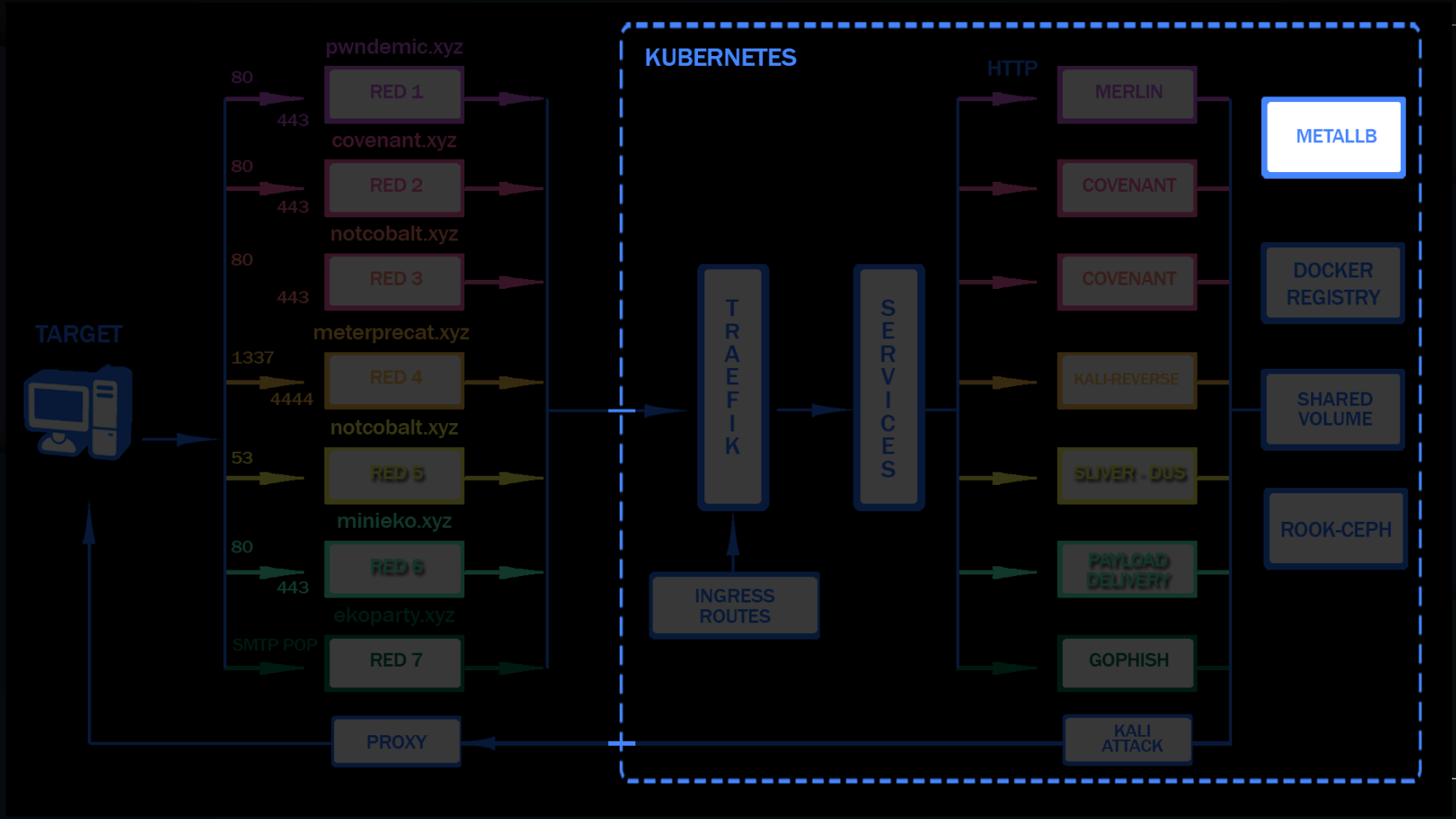
Ingress: es un objeto que administra el acceso externo a servicios del cluster

Ingress-controller: es quien maneja los Ingress :) (vamos a usar traefik)

Ingressroute: es la implementacion del router de traefik, vamos a usarlo para Configurar los routeos.

Statefulset: gestiona los pods para que tengan ESTADO

Usted esta aquí ->metallb



Configurando kubernetes/metallb

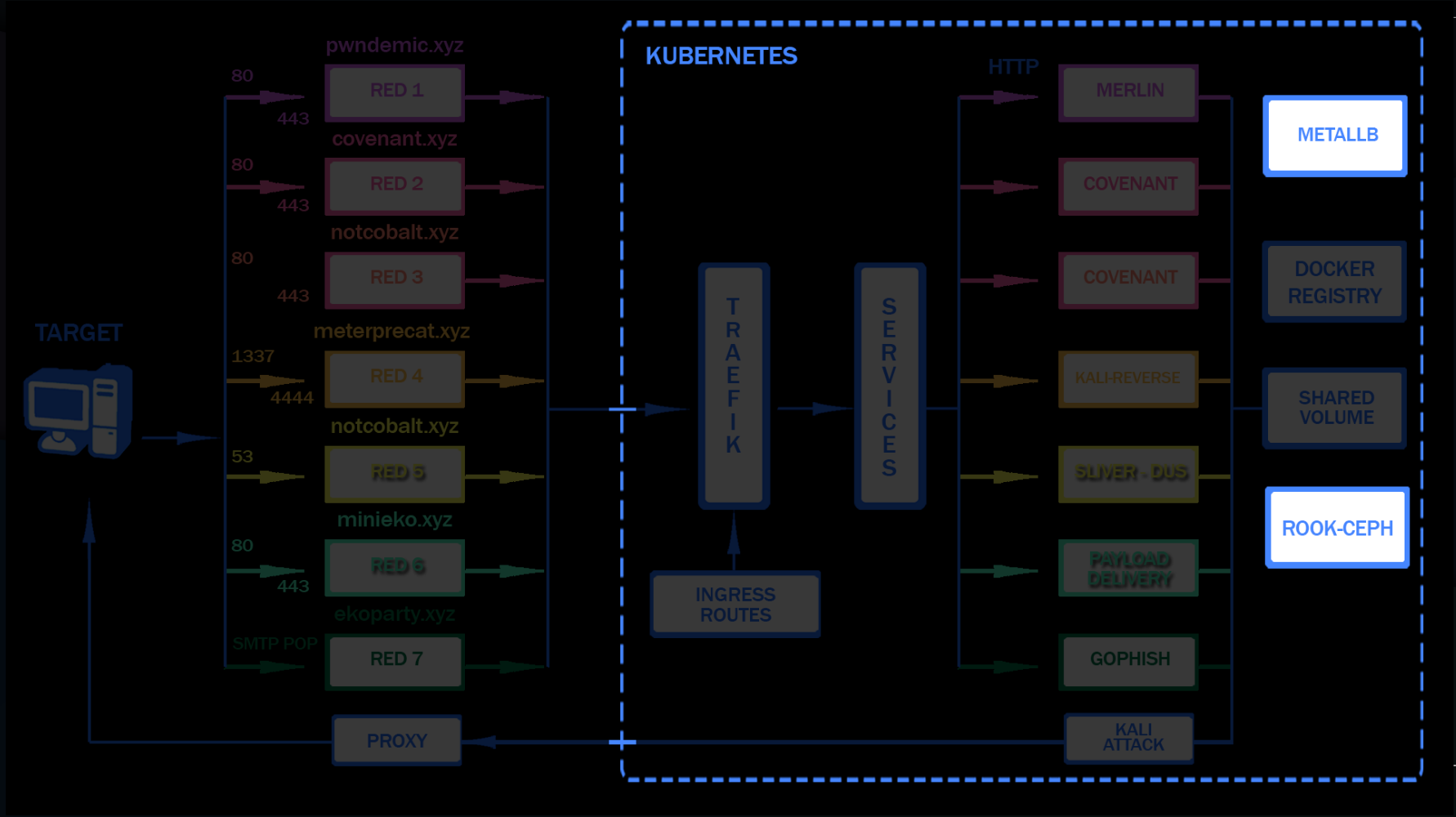
Metallb es un baremetal load balancer, nos va a dejar entre otras cosas, asignar ips publicas automaticamente a servicios de traefik en k8s.

2-configmap.yaml

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
 namespace: metallb-system
 name: config
data:
  config: |
    address-pools:
    - name: public
      protocol: layer2
      addresses:
      - 138.68.21.56
```

```
ko apply -f https://raw.githubusercontent.com/metallb/metallb/v0.9.3/manifests/namespace.yaml
ko apply -f https://raw.githubusercontent.com/metallb/metallb/v0.9.3/manifests/metallb.yaml

ko create secret generic -n metallb-system memberlist --from-literal=secretkey="$(openssl rand -base64 128)"

ko apply -f 2-configmap.yaml
```

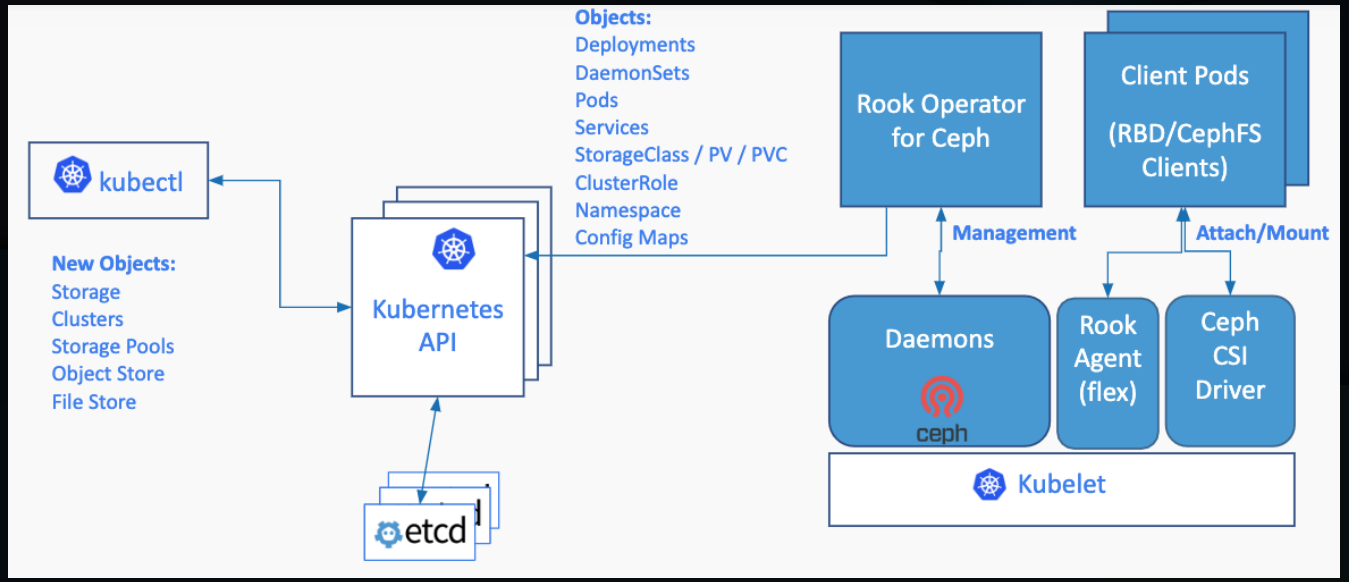
usted esta aquí -> rook-ceph



Configurando kubernetes/rook-ceph

Repo: infraestructura-ofensiva-infa/rook-ceph

Ceph es una solucion de storage distribuida permite usar block storage,object storage y filesystems compartidos, rook es un orquestador cloud de storage, que lo vamos a configurar para usar ceph

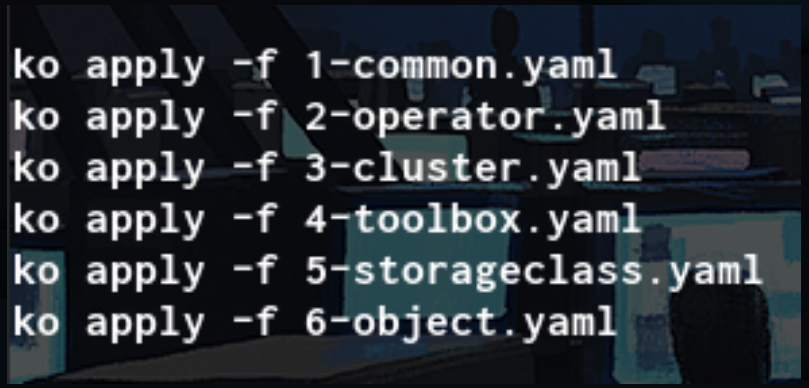


rook-ceph pros

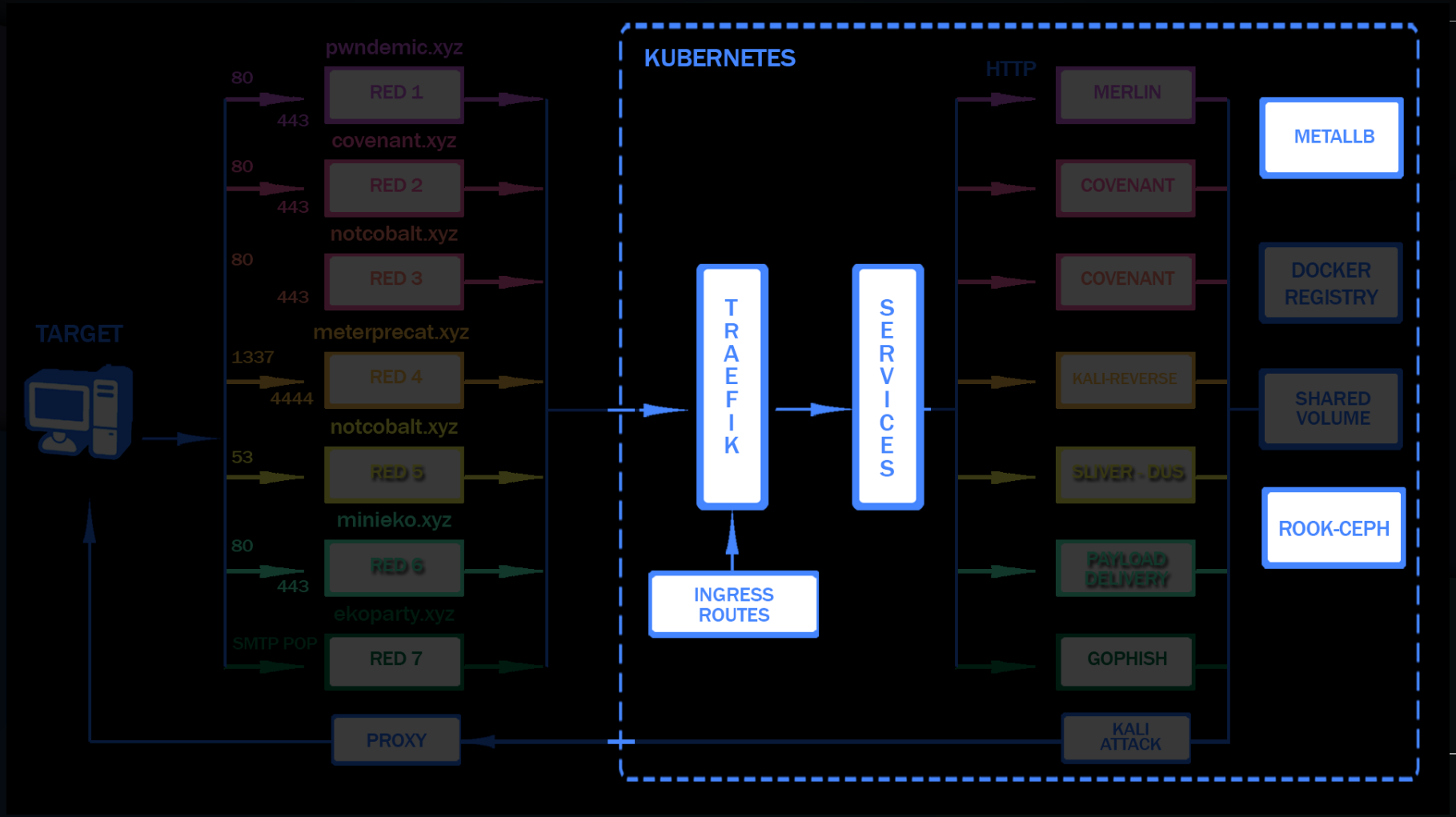
- Snapshots
- Permite block storage, object storage, filesystem compartido
- Health checks
- Automatic failover
- Facil de escalar
- interface

Configurando kubernetes/rook-ceph

Repo: offensive-infra/rook-ceph



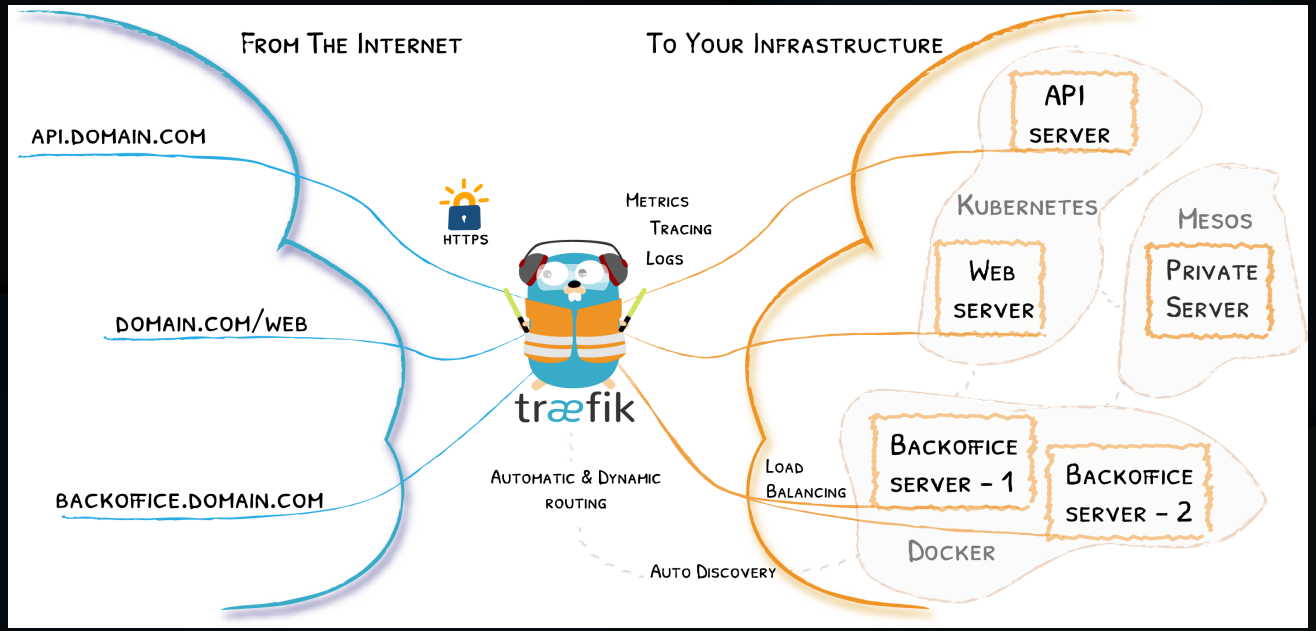
Usted esta aquí -> traefik



Configurando kubernetes/traefik

Repo: infraestructura-ofensiva-infra/traefik

Traefik es un load balancer o “Edge router” que nos va a permitir publicar nuestros servicios(payloads,c2,gophish, etc), una de las ventajas mas importantes es el auto discovery.



Configurando kubernetes/traefik

1- aplicar custom resource definitions y los RBAC

```
root@mini:~# ko apply -f 1.yaml
```

Configurando kubernetes/traefik

1- Crear los Servicios para traefik,

con ips publicas manejadas

por metallb

2- aplicar

Ko apply -f 2.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: traefik-svc
  annotations:
    metallb.universe.tf/address-pool: public
spec:
  ports:
    - protocol: TCP
      name: web
      port: 80
    - protocol: TCP
      name: https
      port: 443
    - protocol: TCP
      name: rev
      port: 1337
  selector:
    app: traefik
  type: LoadBalancer
---
apiVersion: v1
kind: Service
metadata:
  name: traefik-svc-udp
  annotations:
    metallb.universe.tf/address-pool: public
spec:
  ports:
    - protocol: UDP
      name: dns
      port: 53
  selector:
    app: traefik
  type: LoadBalancer
```

Configurando kubernetes/traefik

3- modificar y crear el deployment

ko apply -f 3.yaml

```yaml
ports:
  - name: web
    containerPort: 80
  - name: https
    containerPort: 443
  - name: rev
    containerPort: 1337
  - name: dns
    containerPort: 53
```

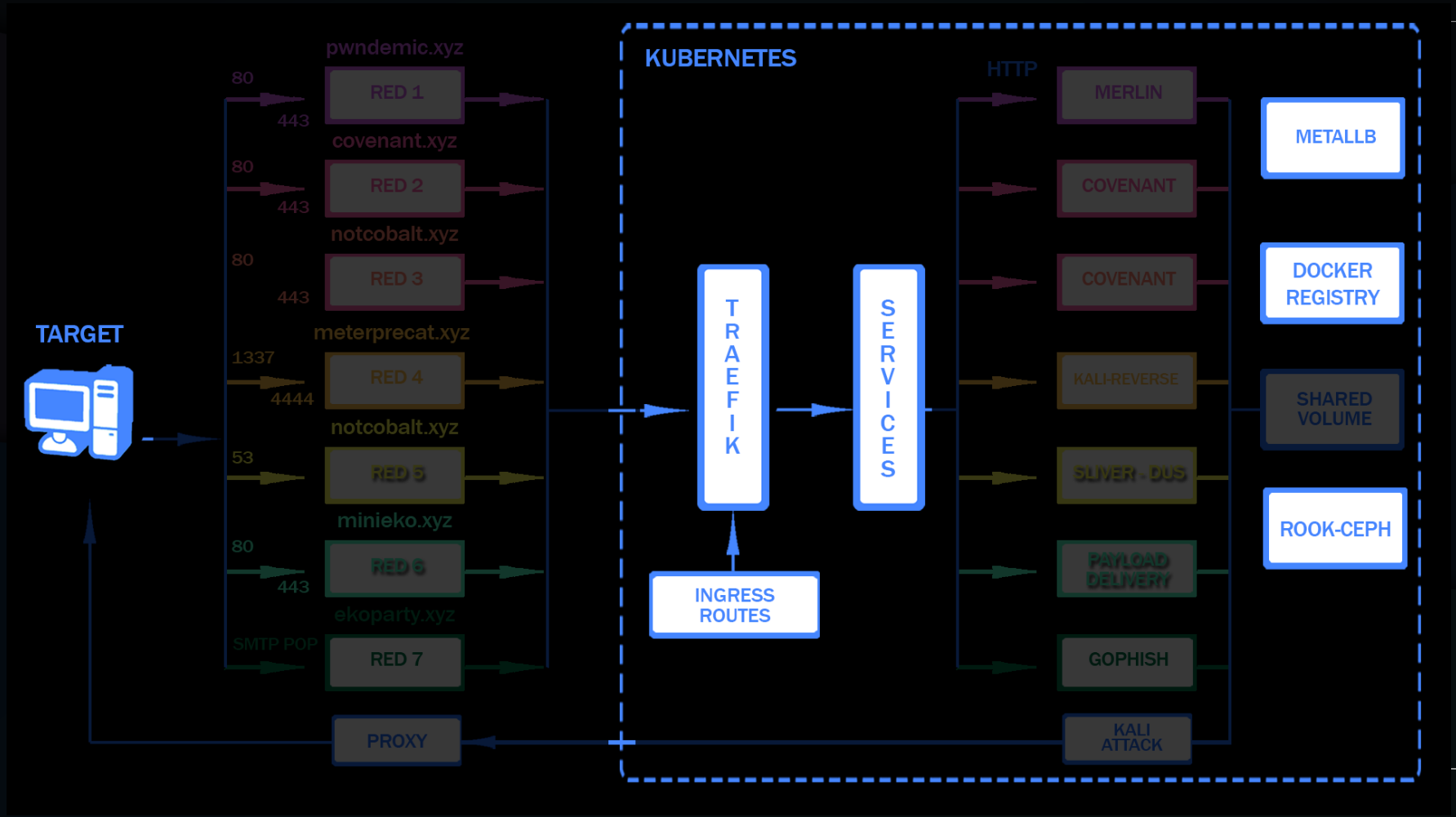
```yaml
kind: Deployment
apiVersion: apps/v1
metadata:
  namespace: default
  name: traefik
  labels:
    app: traefik
spec:
  replicas: 1
  selector:
    matchLabels:
      app: traefik
  template:
    metadata:
      labels:
        app: traefik
    spec:
      serviceAccountName: traefik-ingress-controller
      volumes:
        - name: data
          emptyDir: {}
        - name: tmp
          emptyDir: {}
      containers:
        - name: traefik
          image: traefik:v2.2
          volumeMounts:
            - name: data
              mountPath: /data
            - name: tmp
              mountPath: /tmp
          args:
            - --api.insecure=true
            - --serverstransport.insecureskipverify=true
            - --global.sendanonymoususage
            - --serverstransport.insecureskipverify=true
            - --providers.kubernetesingress
            - --accesslog
            - --entrypoints.web.address=:80
            - --entrypoints.rev.address=:1337
            - --entrypoints.https.Address=:443
            - --entrypoints.dns.Address=:443/udp
            - --providers.kubernetescrd
            - --providers.kubernetesingress
```

Configurando kubernetes/traefik

4- verificar

```
mini@molly:~/eko2020/offensive-infra/c2/merlin-lh$ ko get svc
NAME          TYPE           CLUSTER-IP      EXTERNAL-IP                PORT(S)                                AGE
dashboard     ClusterIP      10.233.57.100   <none>                     3333/TCP                               19h
kubernetes    ClusterIP      10.233.0.1      <none>                     443/TCP                                18d
traefik-svc   LoadBalancer   10.233.21.140   135.181.58.56,135.10.10.10   80:30480/TCP,443:31017/TCP,1337:32578/TCP   3d20h
```

Usted esta aquí -> docker-registry



Configurando kubernetes/docker-registry

Si queremos poder orquestrar containers, necesitamos un lugar de donde kubernetes pueda bajar las imagenes, recomiendo un registry privado.

Crear espacio

```
root@node1:~# mkdir -p /root/registry/{images,certs,auth}
```

K3S COMPATIBLE!

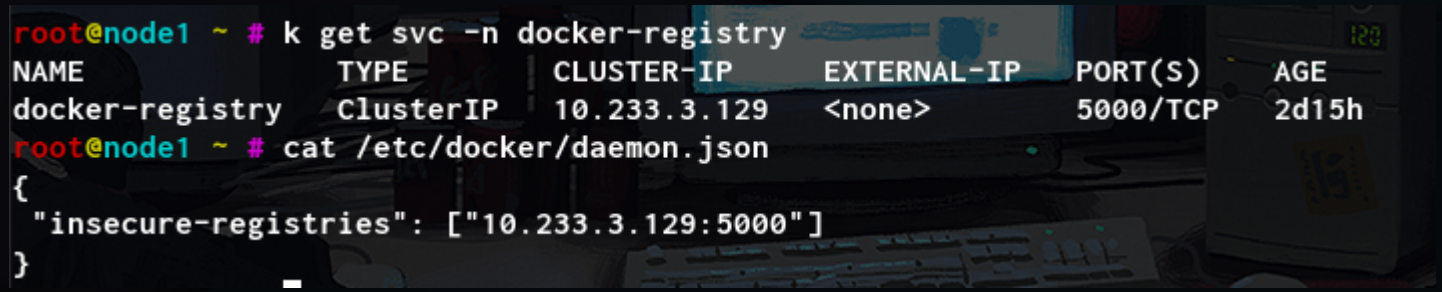
Configurando kubernetes/docker-registry

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: docker-registry
  namespace: docker-registry
spec:
  replicas: 1
  selector:
    matchLabels:
      app: docker-registry
  template:
    metadata:
      labels:
        app: docker-registry
    spec:
      containers:
        - name: docker-registry
          image: registry:2.6.2
          env:
            - name: REGISTRY_HTTP_ADDR
              value: ":5000"
            - name: REGISTRY_STORAGE_FILESYSTEM_ROOTDIRECTORY
              value: "/var/lib/registry"
          ports:
          - name: http
            containerPort: 5000
          volumeMounts:
          - name: image-store
            mountPath: "/var/lib/registry"
      volumes:
        - name: image-store
          hostPath:
            path: /root/registry/images
```

```yaml
kind: Service
apiVersion: v1
metadata:
  name: docker-registry
  namespace: docker-registry
  labels:
    app: docker-registry
spec:
  selector:
    app: docker-registry
  ports:
  - name: http
    port: 5000
    targetPort: 5000
```

Configurando kubernetes/docker-registry

Check docker service cluster-ip y agregala como insecure-registry #noshame



Testeando kubernetes/docker-registry

* ssh tunelTesteando el registry

    ssh -L 5000:<ip-docker-registry-svc>:5000 anoyo

* build a micro dockerfile

```
1 FROM debian:latest
2 CMD tail -f /dev/null
3
```

Kubernetes usa los exit codes del runtime, un hack es usar tail

```
mini@molly:~/eko2020/offensive-infra/docker-registry$ docker build -t 127.0.0.1:5000/microdebian .
Sending build context to Docker daemon  5.632kB
Step 1/2 : FROM debian:latest
```

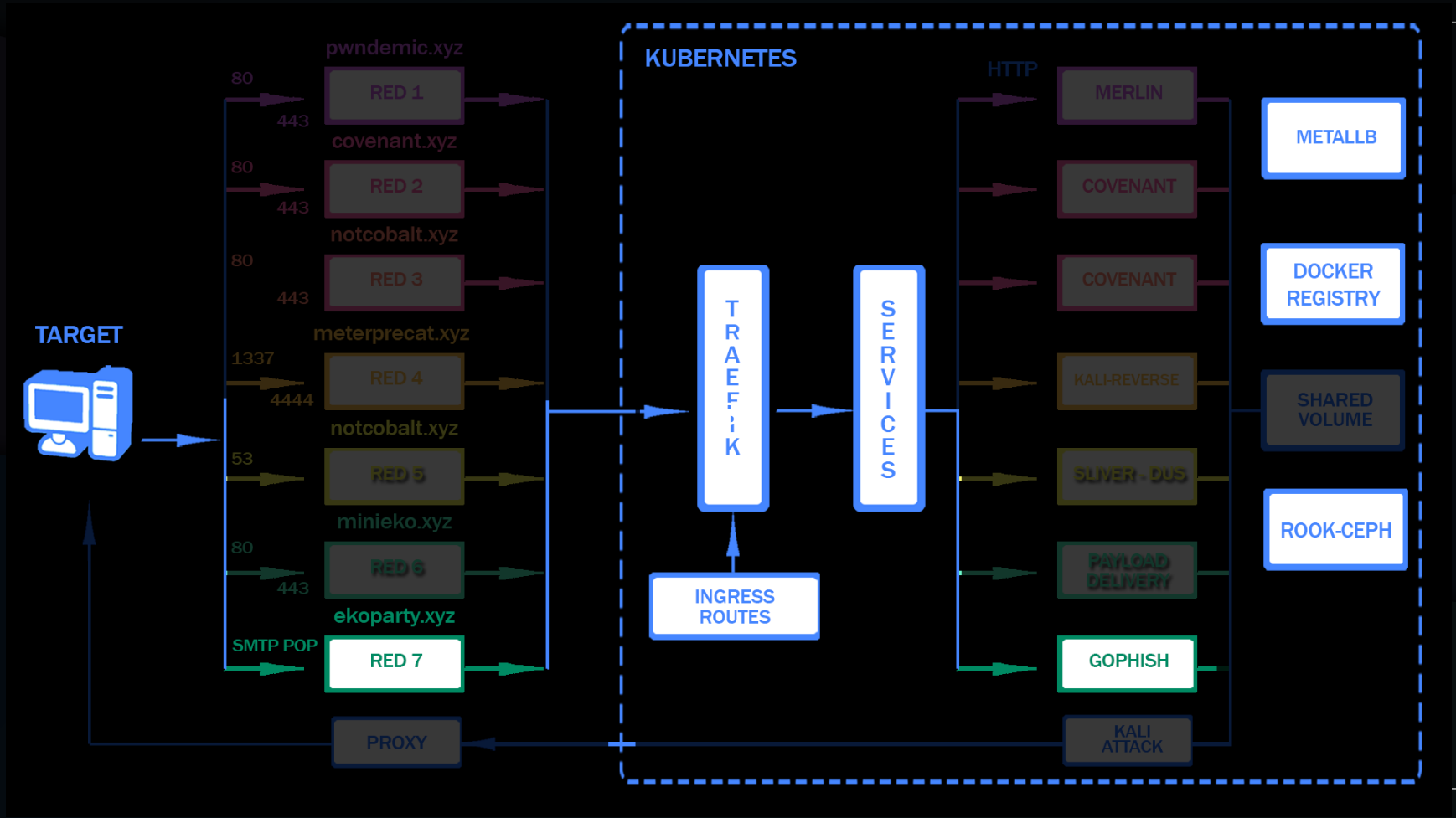
* push it

```
mini@molly:~$ docker push 127.0.0.1:5000/microdebian
The push refers to repository [127.0.0.1:5000/microdebian]
```

Testeando kubernetes/docker-registry

```
root@node1 ~ # k run microtest --image 10.233.3.129:5000/microdebian
pod/microtest created
root@node1 ~ # k get pods
NAME                        READY   STATUS    RESTARTS   AGE
gophish-0                   1/1     Running   0          20h
microtest                   1/1     Running   0          4s
traefik-dd46cf49b-zqwxf     1/1     Running   0          21h
root@node1 ~ # k exec microtest -it -- bash
root@microtest:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run
root@microtest:/#
```

Usted esta aquí -> gophish



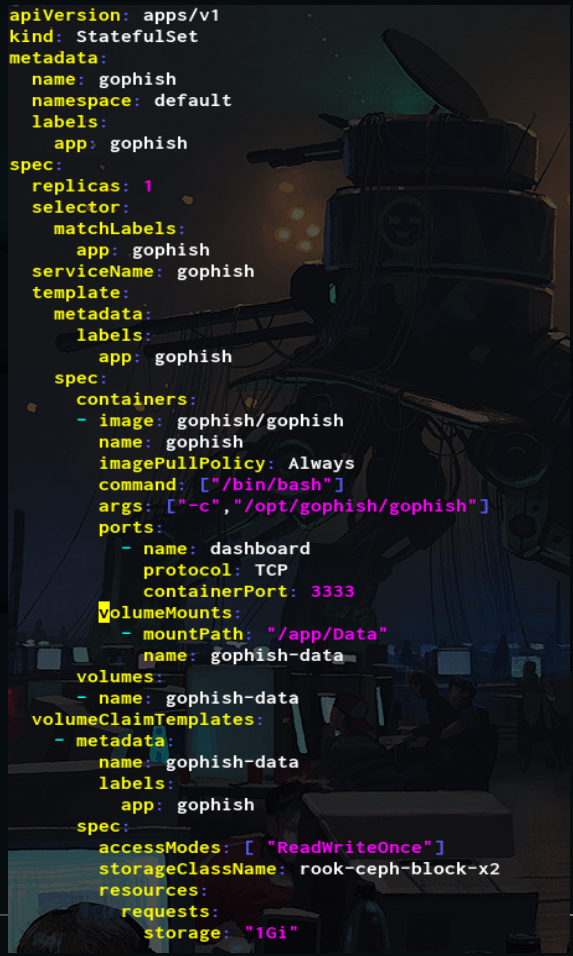
Deployando gophish

Para gophish vamos a necesitar:

- Levantar gophish server en un Statefulset
- Levantar gophish service para usar el dashboard
- Levantar el redirector con el servidor mail

Gophish Statefulset

ko apply -f gophish-statefulset.yaml



```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: gophish
  namespace: default
  labels:
    app: gophish
spec:
  replicas: 1
  selector:
    matchLabels:
      app: gophish
  serviceName: gophish
  template:
    metadata:
      labels:
        app: gophish
    spec:
      containers:
      - image: gophish/gophish
        name: gophish
        imagePullPolicy: Always
        command: ["/bin/bash"]
        args: ["-c","/opt/gophish/gophish"]
        ports:
          - name: dashboard
            protocol: TCP
            containerPort: 3333
        VolumeMounts:
          - mountPath: "/app/Data"
            name: gophish-data
      volumes:
      - name: gophish-data
  volumeClaimTemplates:
    - metadata:
        name: gophish-data
        labels:
          app: gophish
      spec:
        accessModes: [ "ReadWriteOnce"]
        storageClassName: rook-ceph-block-x2
        resources:
          requests:
            storage: "1Gi"
```

Gophish Service

ko apply -f gosphish-svc.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: dashboard
  namespace: default
  labels:
    app: gophish
spec:
  ports:
  - port: 3333
    name: dashboard
    targetPort: 3333
  selector:
    app: gophish
```

Gophish Redirector

- Ansible pseudo-codigo provisioning

```
- hosts: redirector1
  gather_facts: no
  vars:
    username: tomav
    repo_name: docker-mailserver

  tasks:

  - name: install base packages
    apt: pkg={{item}} state=present update_cache=yes cache_valid_time=604800
    with_items:
      - git
      - tmux
      - cerbot
      - python3-certbot-apache

  - name: Create and Install Cert Using {{ certbot_plugin }} Plugin
    command: "certbot --apache -d ekoparty.xyz -m tu@mail.com --no-eff-email --agree-tos "


  - name: Checkout The Code From Github Using Ansible.
    git:
     repo: 'https://github.com/{{ username }}/{{ repo_name }}.git'
     dest: /root/docker-mailserver
```

Docker-mailserver

1. mkdir -p /var/ds/mail.ekoparty.xyz
2. cd /var/ds/mail.ekoparty.xyz
3. wget https://raw.githubusercontent.com/tomav/docker-mailserver/master/setup.sh
4. chmod a+x ./setup.sh

Docker-mailserver

5. Crear docker-compose.yaml

```yaml
version: '2'
services:
  mail:
    image: tvial/docker-mailserver:latest
    hostname: ${HOSTNAME}
    domainname: ${DOMAINNAME}
    container_name: ${CONTAINER_NAME}
    ports:
    - "25:25"
    - "143:143"
    - "587:587"
    - "993:993"
    volumes:
    - maildata:/var/mail
    - mailstate:/var/mail-state
    - maillogs:/var/log/mail
    - ./config/:/tmp/docker-mailserver/
    env_file:
    - .env
    - env-mailserver
    cap_add:
    - NET_ADMIN
    - SYS_PTRACE
    restart: always
volumes:
  maildata:
    driver: local
  mailstate:
    driver: local
  maillogs:
    driver: local
```

Docker-mailserver

6. UFW!!!
    - ALLOW 25 587 465 143 993
7. Crear un mail

    ./setup.sh email add test@ekoparty.xyz cecinestpasunepassword
8. Crear DKIM

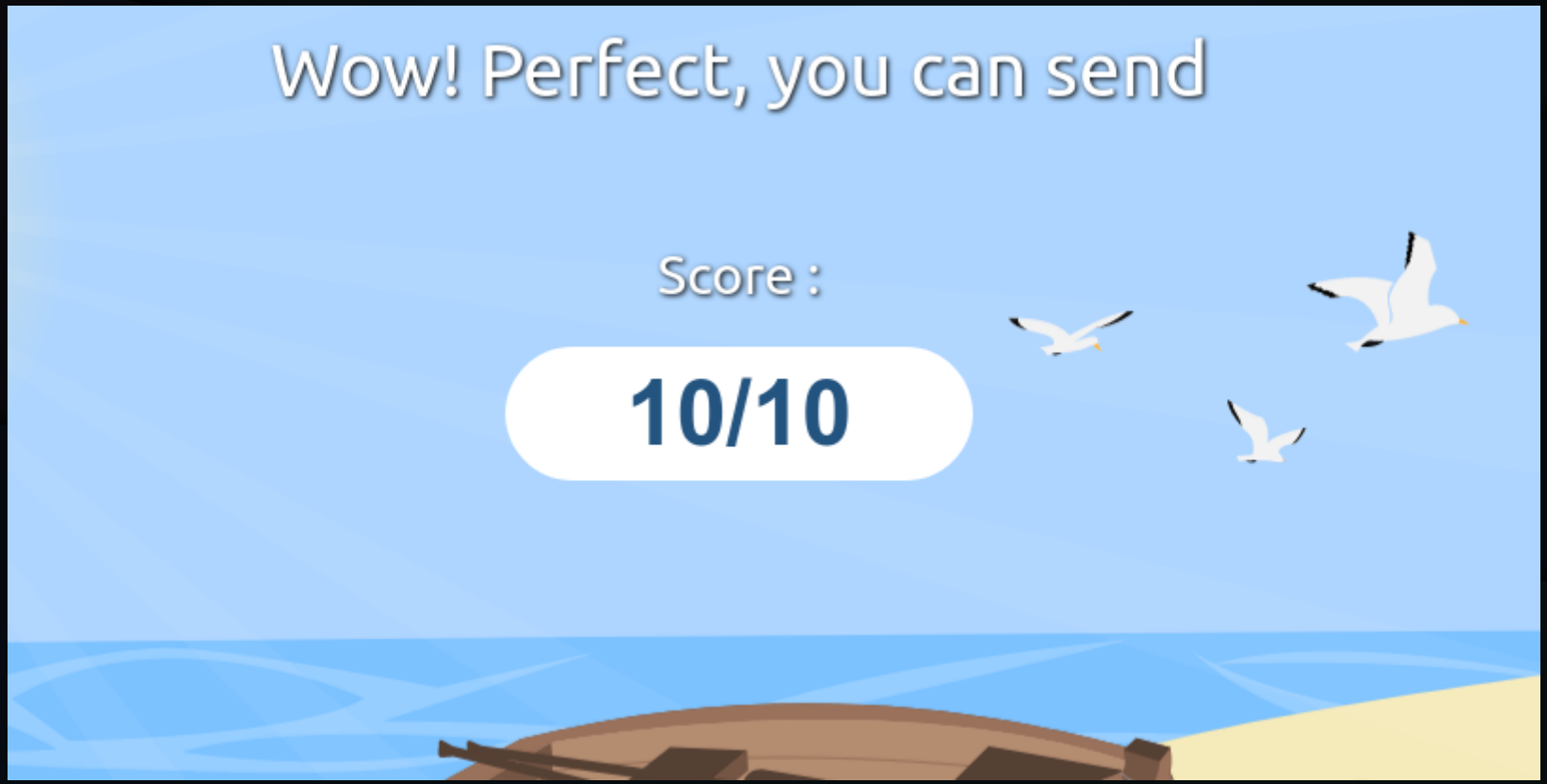
    ./setup.sh config dkim

9. Configurar reverse lookup , spf,dmarc,ptr, etc configurar un servidor/dominio para phishing es una charla en si misma.

Mas info: https://github.com/obscuritylabs/RAI/tree/master/PhishingServer

Docker-mailserver

https://www.mail-tester.com



Gophish round 2

Gophish

- Identificar ip servicio:

  ko get svc

- Identificar password

  ko logs gophish-0

```
OK    20200619000000_0.11.0_password_policy.sql
OK    20200730000000_0.11.0_imap_ignore_cert_errors.sql
time="2020-09-23T02:52:10Z" level=info msg="Please login with the username admin and the password 33388ed56e6caa34"
time="2020-09-23T02:52:10Z" level=info msg="Starting phishing server at http://0.0.0.0:80"
time="2020-09-23T02:52:10Z" level=info msg="Background Worker Started Successfully - Waiting for Campaigns"
time="2020-09-23T02:52:10Z" level=info msg="Creating new self-signed certificates for administration interface"
time="2020-09-23T02:52:10Z" level=info msg="Starting IMAP monitor manager"
time="2020-09-23T02:52:10Z" level=info msg="Starting new IMAP monitor for user admin"
```

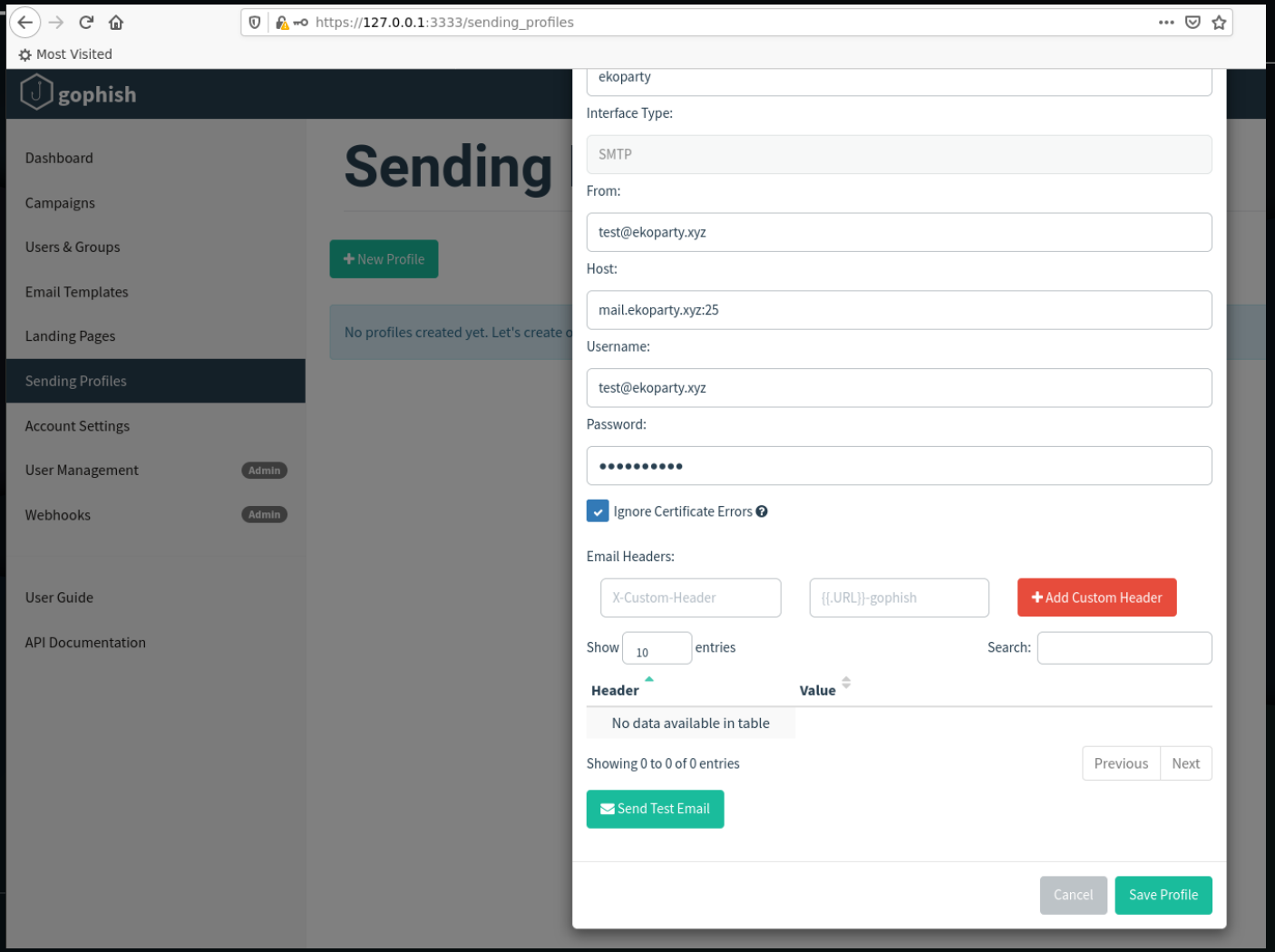
- Por que hacer el dashboard publico si existen los tuneles?

  ssh -L 3333:gophish_cluster_ip:3333 node1

Gophish

1-

Crear sending profile y probar enviar un test email

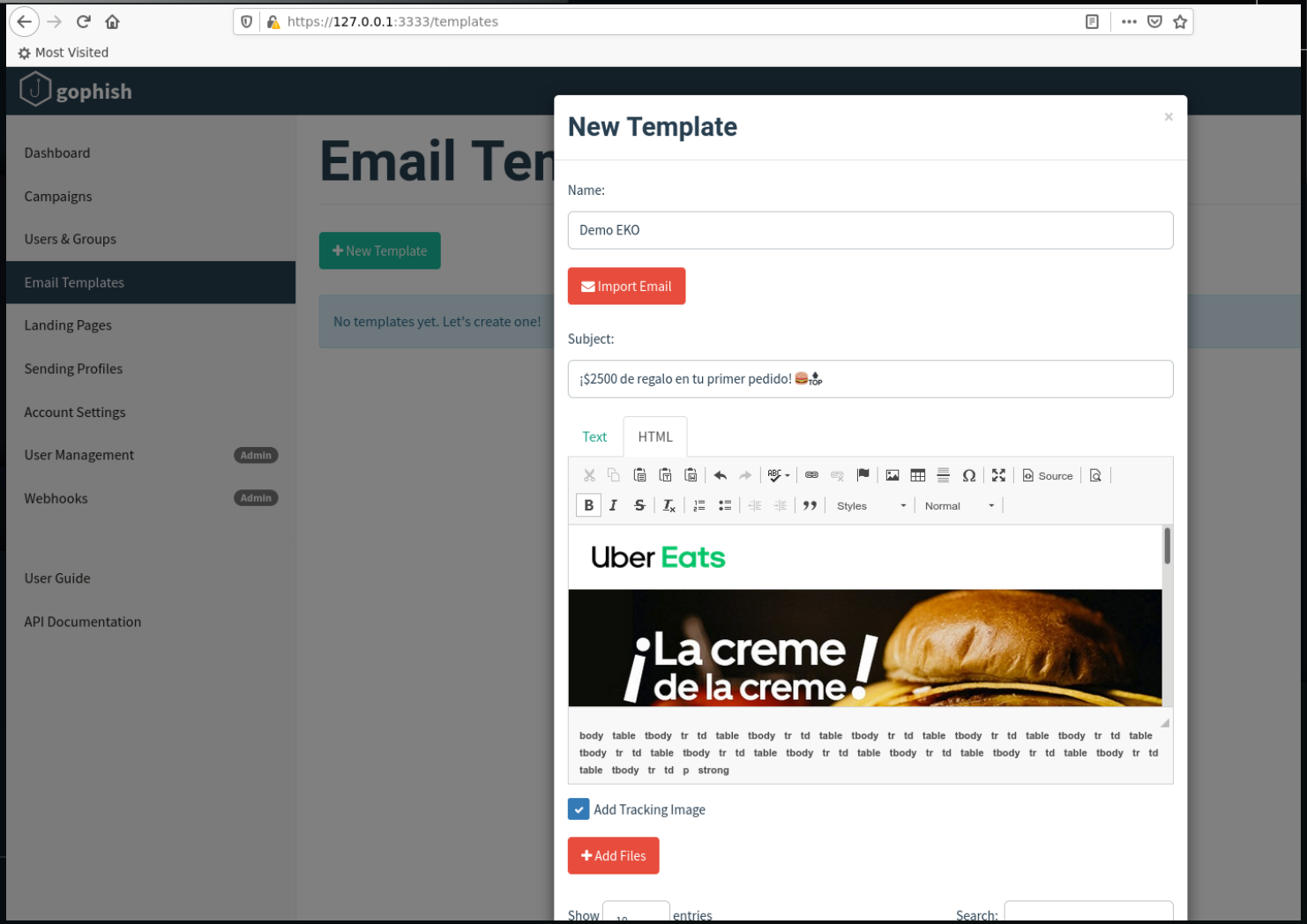


Gophish

2-

Creamos un template

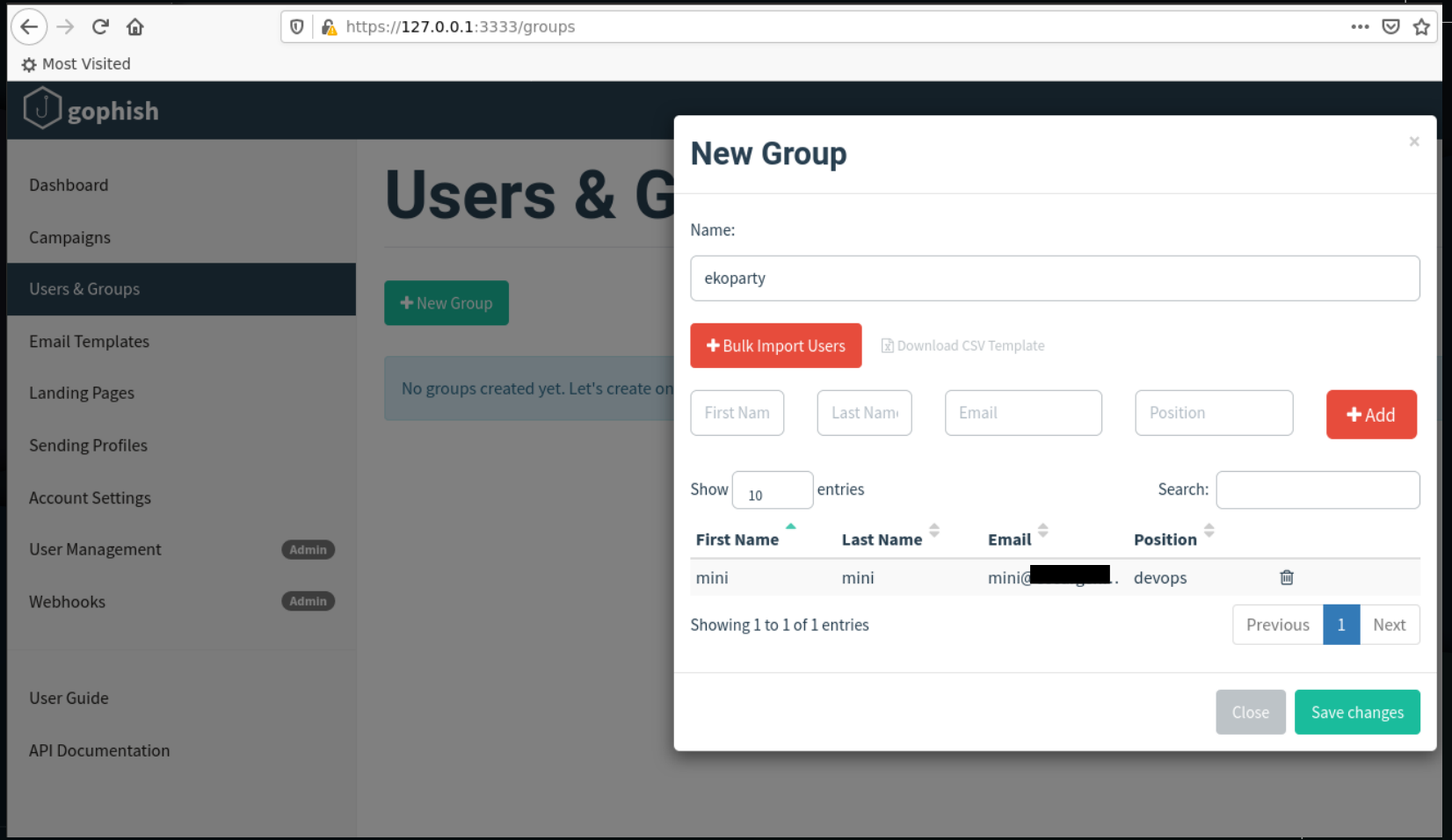
DEMO :)



Gophish

3-

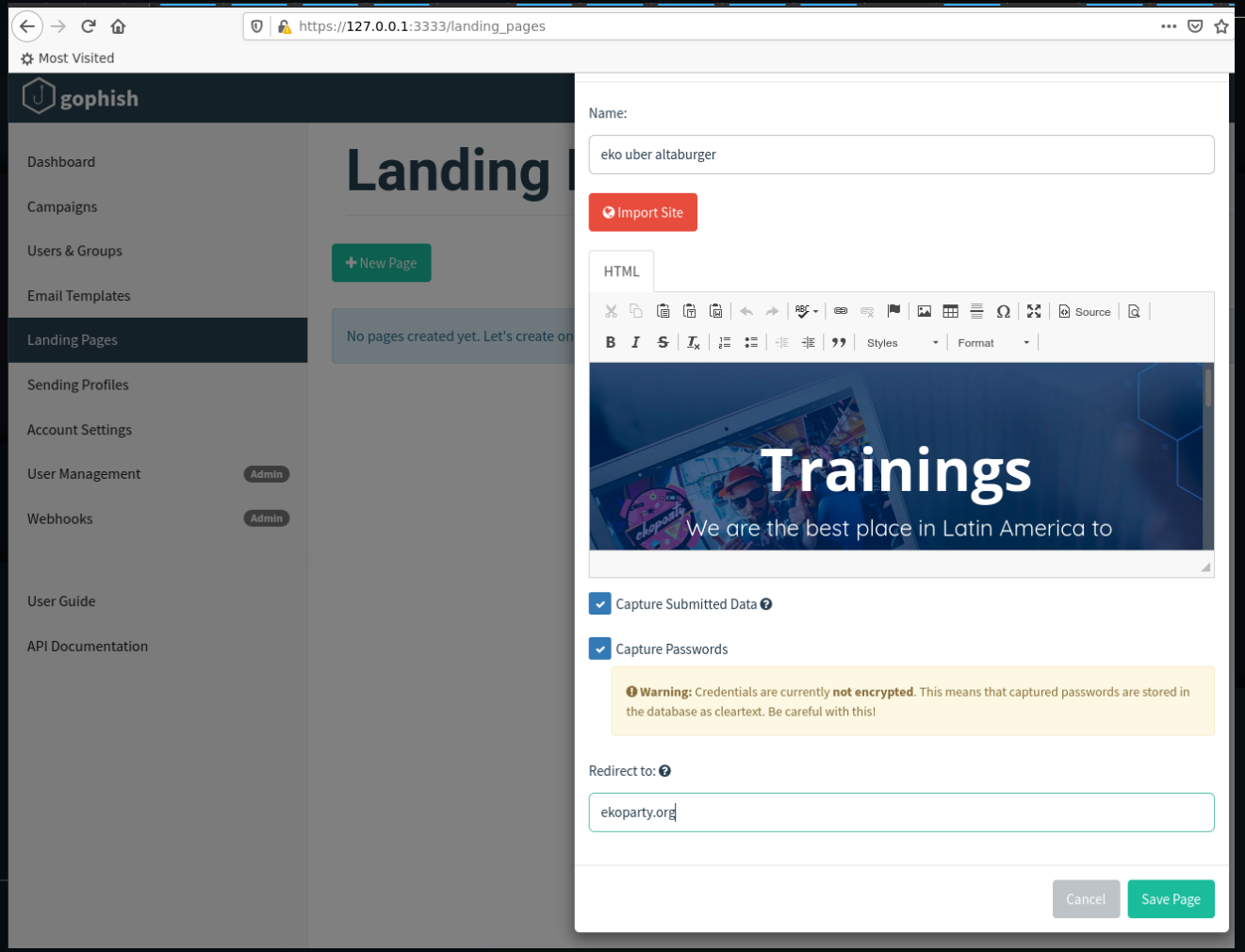
Crear grupo



Gophish

4-

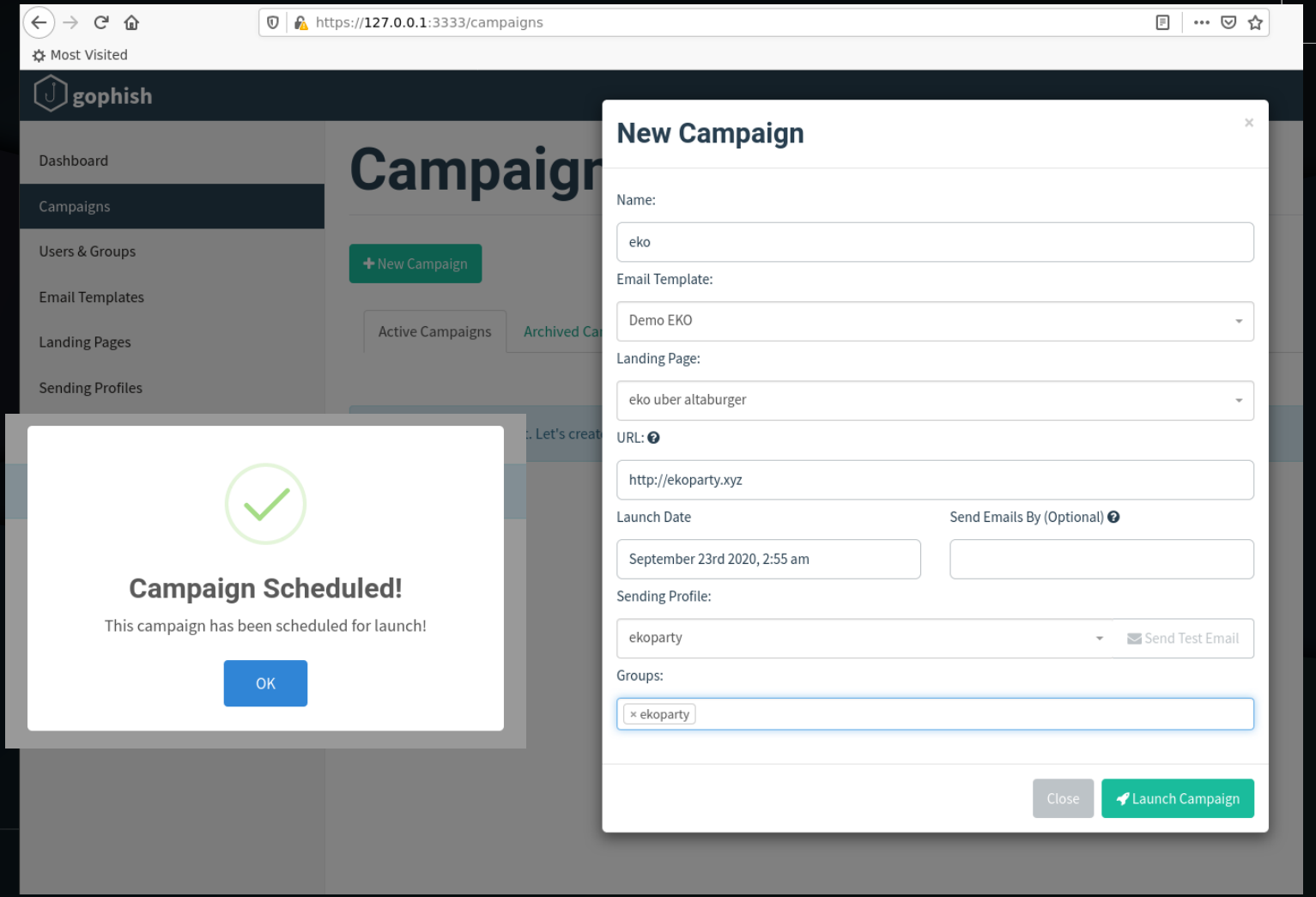
Crear landing



Gophish

5-

Crear campaña



Gophish

From Me <test@ekoparty.xyz>
Subject **¡$2500 de regalo en tu primer pedido!**
To Me
Date We
essage ID
elivered-To mi
Received by

Uber Eats

¡Tus sabores preferidos a un precio especial!

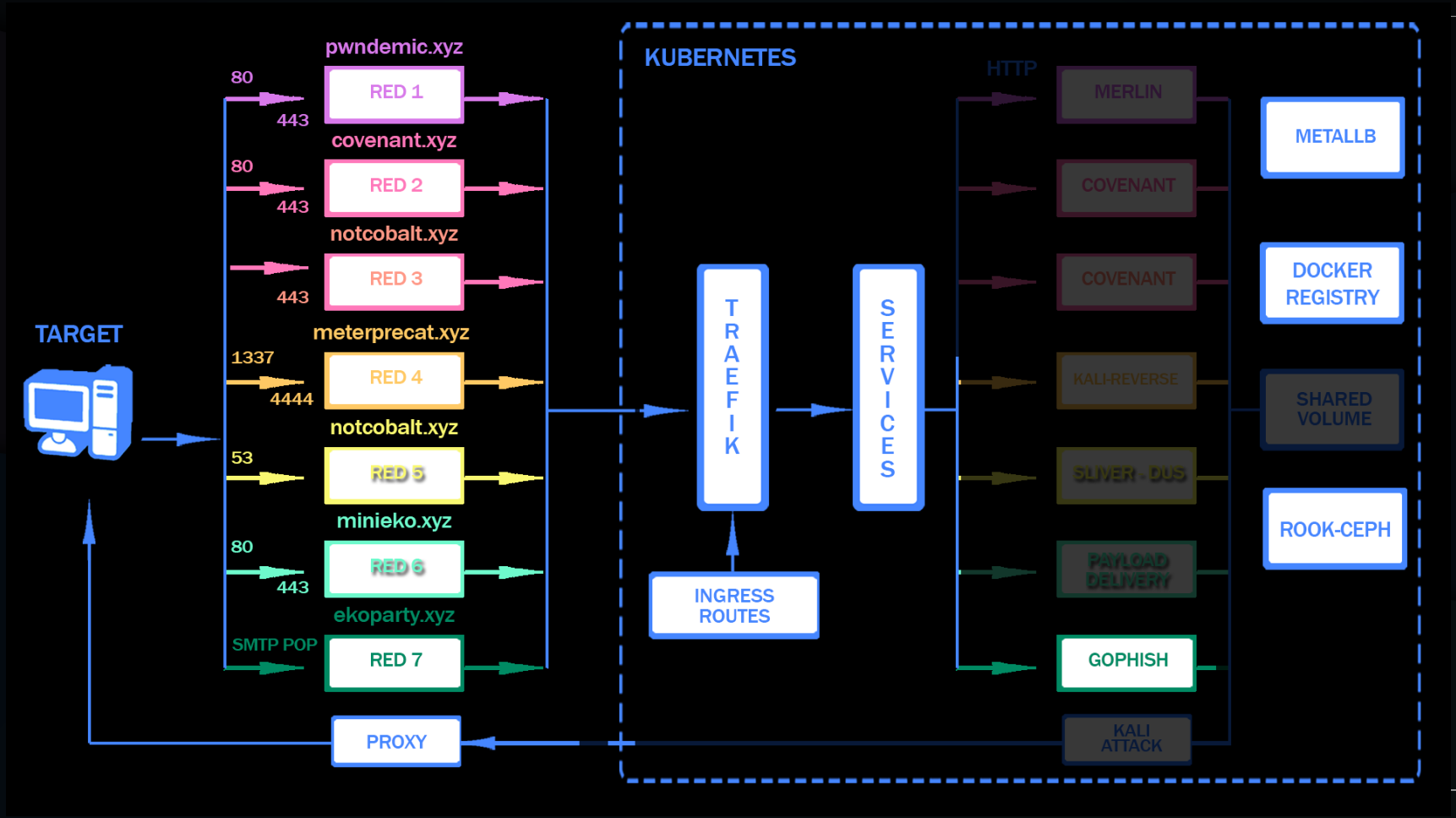
¡Siempre al top!

nicolas, descubrí los platos mejor rankeados de nuestra app icon descuentos increíbles! Esta semana, aprovechá **hasta 50% OFF en las opciones con mejores reseñas en Uber Eats.**

ARG2500EATS

Además, **te regalamos $2500 OFF en tu primer pedido** hasta el domingo

Usted esta aquí -> Redirectors



Deployando redirectors/ansible

```yaml
---
- hosts: all
  become: true
  vars:
    ansible_user: root
    socat_ip: [some-ip]
    ansible_python_interpreter: /usr/bin/python3
  roles:
    - setup
    - adduser
    - docker-installer
    - apache
    - certbot
    - socat_script
```

```yaml
---
- shell: "(socat TCP4-LISTEN:{{ port }},fork TCP4:{{ ip }}:{{ port }} >/dev/null 2>&1 &)"
  async: 10
  poll: 0
```

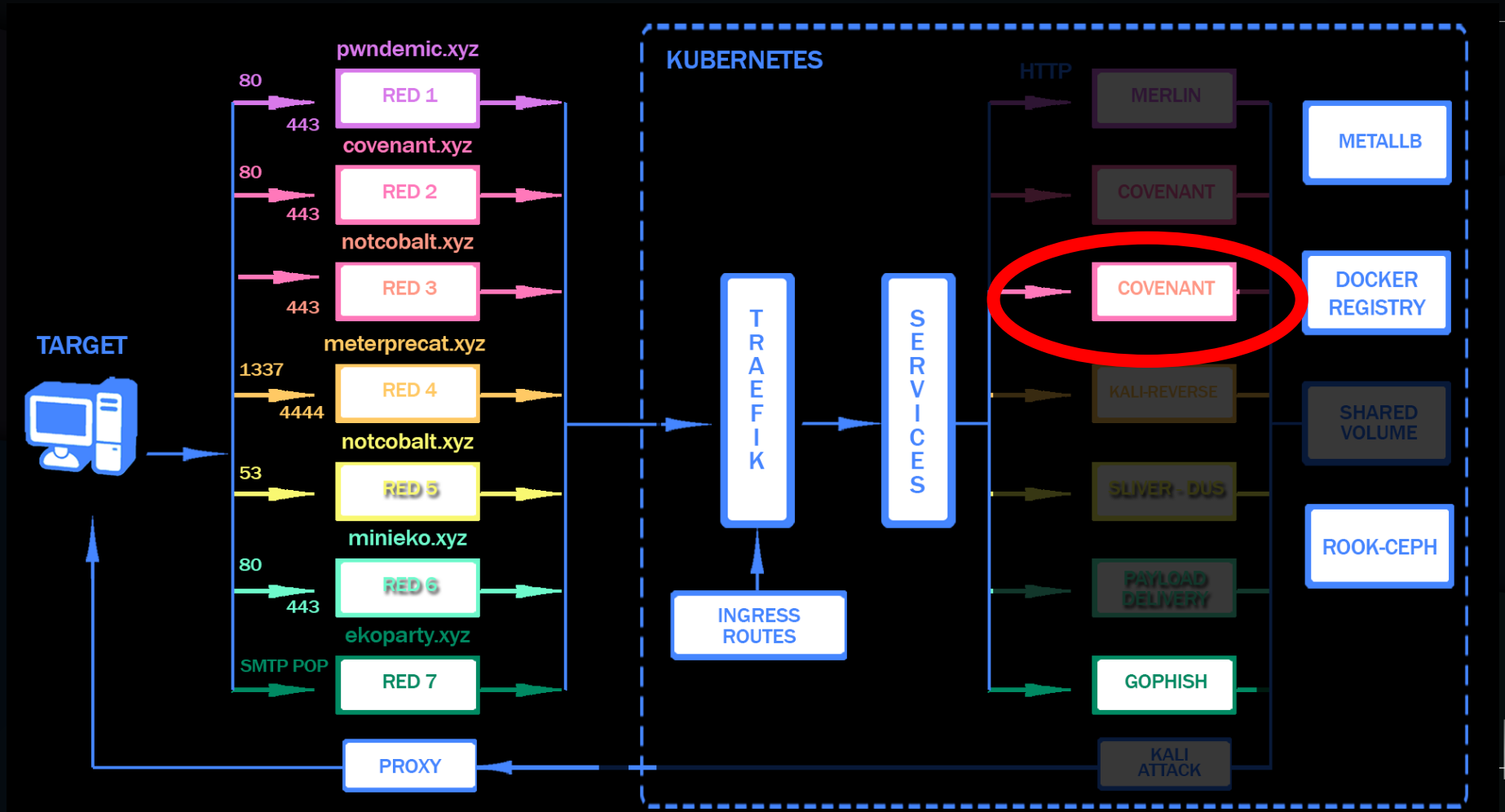
Deployando redirectors/ansible

```yaml
- name: Ensure a locale exists
  locale_gen:
    name: "{{ item }}"
    state: present
  with_items:
    - es_AR.UTF-8
    - en_US.UTF-8

- name: cache
  apt:
    update_cache: yes
    cache_valid_time: 3600
- name: upgrade
  apt: upgrade=full
- name: clean
  apt:
    autoclean: yes
- name: remove
  apt:
    autoremove: yes

```

```yaml

- name: install dependencies base with apt
  apt:
    name: [
     htop,
     telnet,
     vim,
     socat,
     apt-transport-https,
     ntp,
     tmux,
     zsh,
     rsync,
     iftop,
     iotop,
     bzip2,
     screen,
     ncdu,
     dnsutils,
     mc,
     mlocate,
     fail2ban,
     wget,
     curl,
     pv,
     ]
    state: latest
```

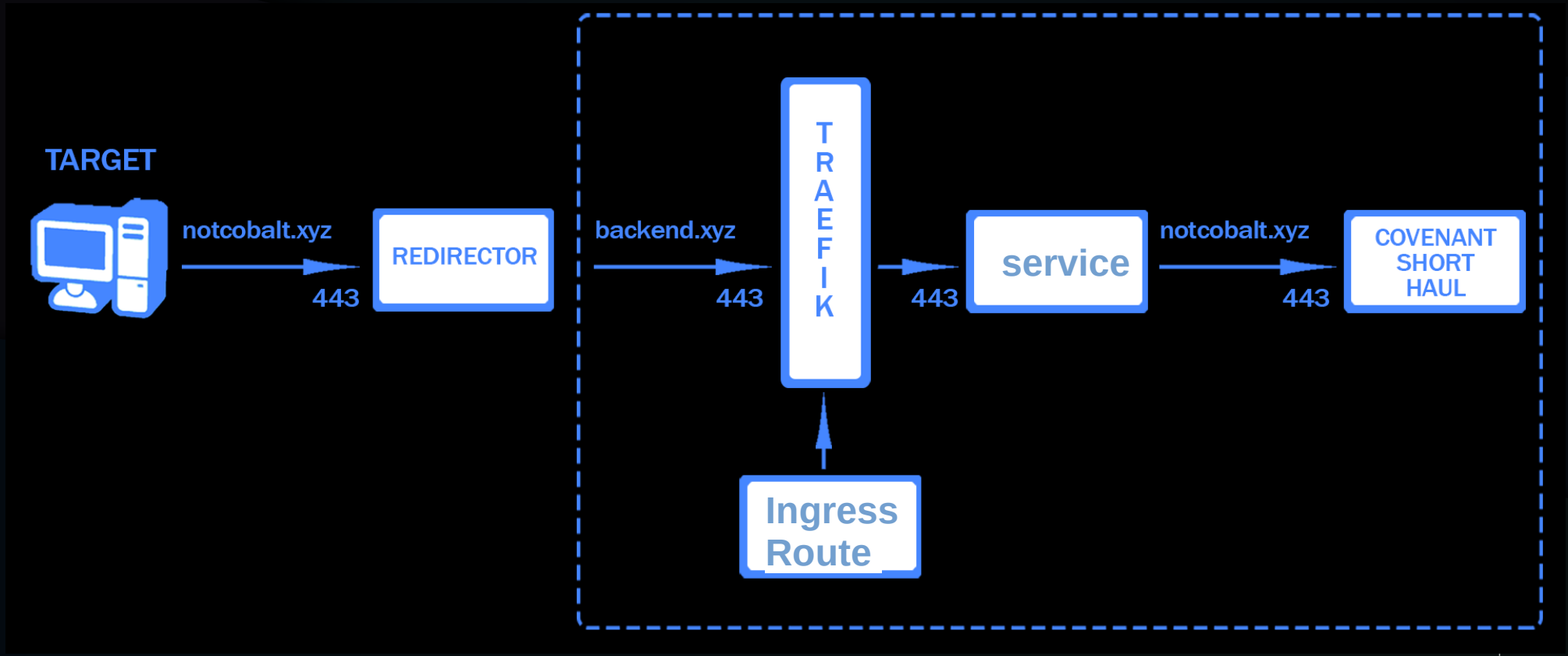
root@mini:~# ansible-playbook playbook.yml -i /inventory/redirectors/hosts.yaml

Usted esta aquí -> covenant-sh




#
COVENANT

COVENANT


TARGET
notcobalt.xyz
443
REDIRECTOR
backend.xyz
443
TRAEFIK
443
service
notcobalt.xyz
443
COVENANT SHORT HAUL
Ingress Route

COVENANT/dominio+certs

Dns a redirector

A Record @ notcobalt.xyz 138.68.21.56

Generar certificado

cerbot -d notcobalt.xyz --apache

COVENANT/Redirector

```
<VirtualHost *:443>
ServerName notcobalt.xyz
# Enable SSL
SSLEngine On
SSLProxyVerify none
SSLProxyCheckPeerCN off
SSLProxyCheckPeerName off

ProxyPreserveHost On
ProxyRequests Off
SSLCertificateFile      /etc/letsencrypt/live/notcobalt.xyz/cert.pem
SSLCertificateKeyFile   /etc/letsencrypt/live/notcobalt.xyz/privkey.pem
SSLCertificateChainFile /etc/letsencrypt/live/notcobalt.xyz/chain.pem
# Enable Proxy
SSLProxyEngine On
  ProxyPass /  https://backend.xyz/
  ProxyPassReverse /  https://backend.xyz/
</VirtualHost>
```

COVENANT/Certs

1- Pasar certificados de letsencrypt (redirector) a traefik
- privkey.pem == tls.key
- cert.pem == tls.cert

2- add as secret
ko create secret tls notcobalt-tls --key=privkey.pem --cert=cert.pem -n c2

3- convert cert de .pem a PKCS#12

openssl pkcs12 -export -out certificate.pfx -inkey privkey.pem -in cert.pem -certfile chain.pem

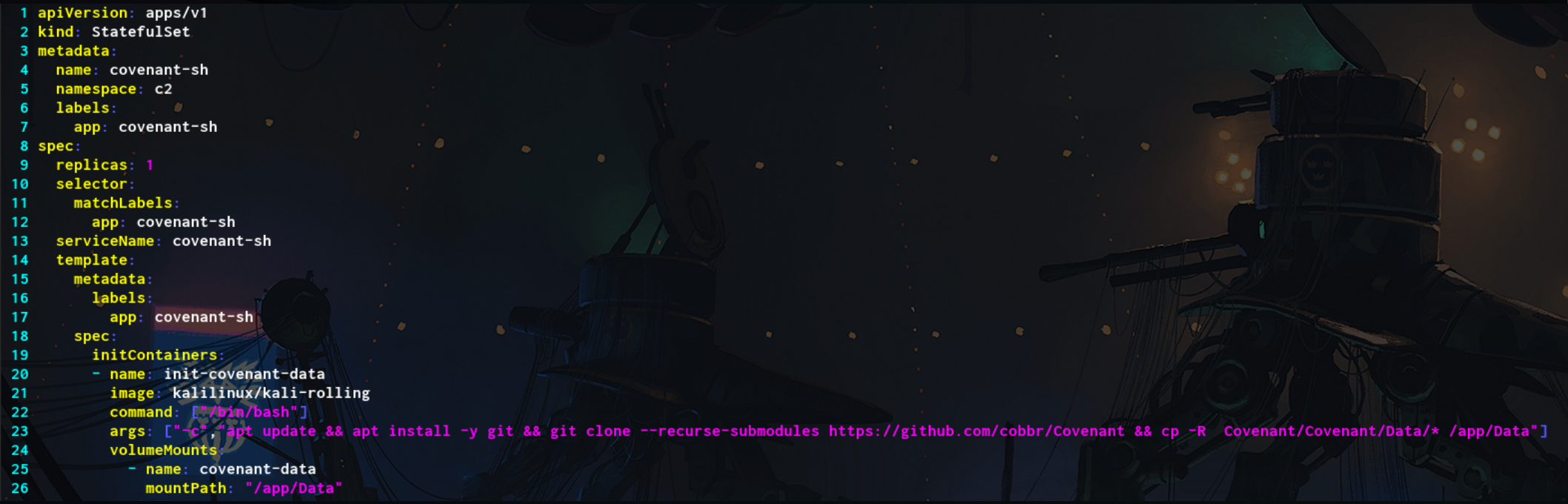
COVENANT/Ingressroute

```yaml
apiVersion: traefik.containo.us/v1alpha1
kind: IngressRoute
metadata:
  name: cov-sh-http
  namespace: c2
spec:
  entryPoints:
    - web
  routes:
  - kind: Rule
    match: Host(`notcobalt.xyz`)
    services:
    - kind: Service
      name: cov-sh-web
      port: 80
---
```

```yaml
---
apiVersion: traefik.containo.us/v1alpha1
kind: IngressRoute
metadata:
  name: cov-sh-https
  namespace: c2
spec:
  entryPoints:
    - https
  routes:
  - kind: Rule
    match: Host(`notcobalt.xyz`)
    services:
    - kind: Service
      name: cov-sh-https
      port: 443
  tls:
    secretName: notcobalt-tls
```

ko apply -f covenant-sh.yaml

COVENANT/statefulset



```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: covenant-sh
  namespace: c2
  labels:
    app: covenant-sh
spec:
  replicas: 1
  selector:
    matchLabels:
      app: covenant-sh
  serviceName: covenant-sh
  template:
    metadata:
      labels:
        app: covenant-sh
    spec:
      initContainers:
      - name: init-covenant-data
        image: kalilinux/kali-rolling
        command: ["/bin/bash"]
        args: ["-c","apt update && apt install -y git && git clone --recurse-submodules https://github.com/cobbr/Covenant && cp -R  Covenant/Covenant/Data/* /app/Data"]
        volumeMounts:
          - name: covenant-data
            mountPath: "/app/Data"
```

COVENANT/statefulset

ko apply -f covenant-statefullset.yaml

```
27      containers:
28      - image: 10.233.3.129:5000/covenant
29        name: covenant-sh
30        imagePullPolicy: Always
31        command: ["dotnet"]
32        args: ["/app/Covenant.dll"]
33        ports:
34          - name: cov-sh-web
35            protocol: TCP
36            containerPort: 80
37          - name: cov-sh-https
38            protocol: TCP
39            containerPort: 443
40        volumeMounts:
41          - mountPath: "/app/Data"
42            name: covenant-data
43          - mountPath: "/shared-pvc"
44            name: shared-storage
45      volumes:
46      - name: covenant-data
47      - name: shared-storage
48        persistentVolumeClaim:
49          claimName: shared-pvc
50  volumeClaimTemplates:
51    - metadata:
52        name: covenant-data
53        labels:
54          app: covenant-sh
55      spec:
56        accessModes: [ "ReadWriteOnce"]
57        storageClassName: rook-ceph-block-x2
58        resources:
59          requests:
60            storage: "16Gi"
```

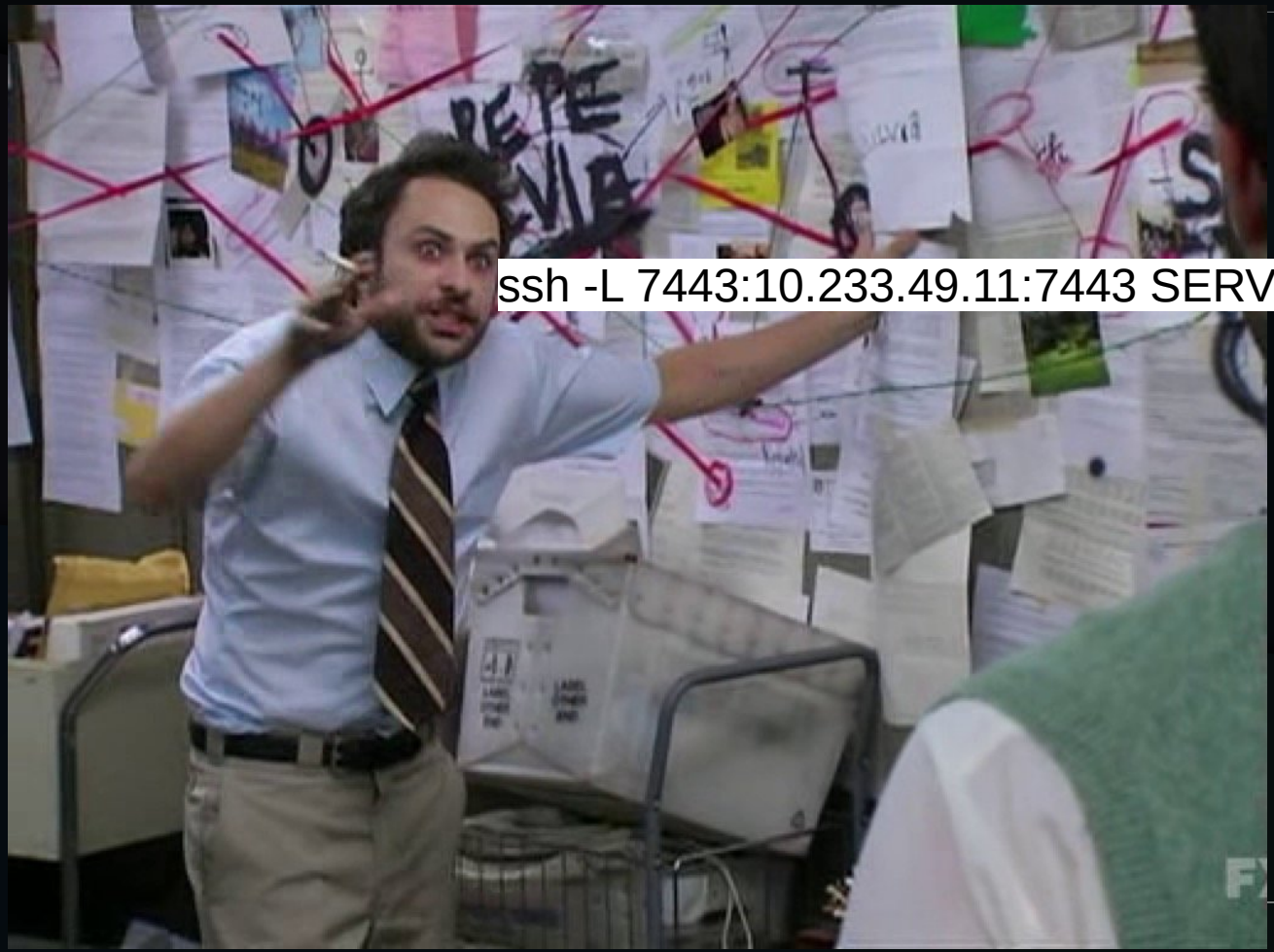
COVENANT/svc

```yaml
apiVersion: v1
kind: Service
metadata:
  name: cov-sh-web
  namespace: c2
  labels:
    app: covenant-sh
spec:
  ports:
  - port: 80
    name: cov-sh-web
    targetPort: 80
  selector:
    app: covenant-sh
---
apiVersion: v1
kind: Service
metadata:
  name: cov-sh-dashboard
  namespace: c2
  labels:
    app: covenant-sh
spec:
  ports:
  - port: 7443
    name: cov-sh-dashboard
    targetPort: 7443
  selector:
    app: covenant-sh
---
```

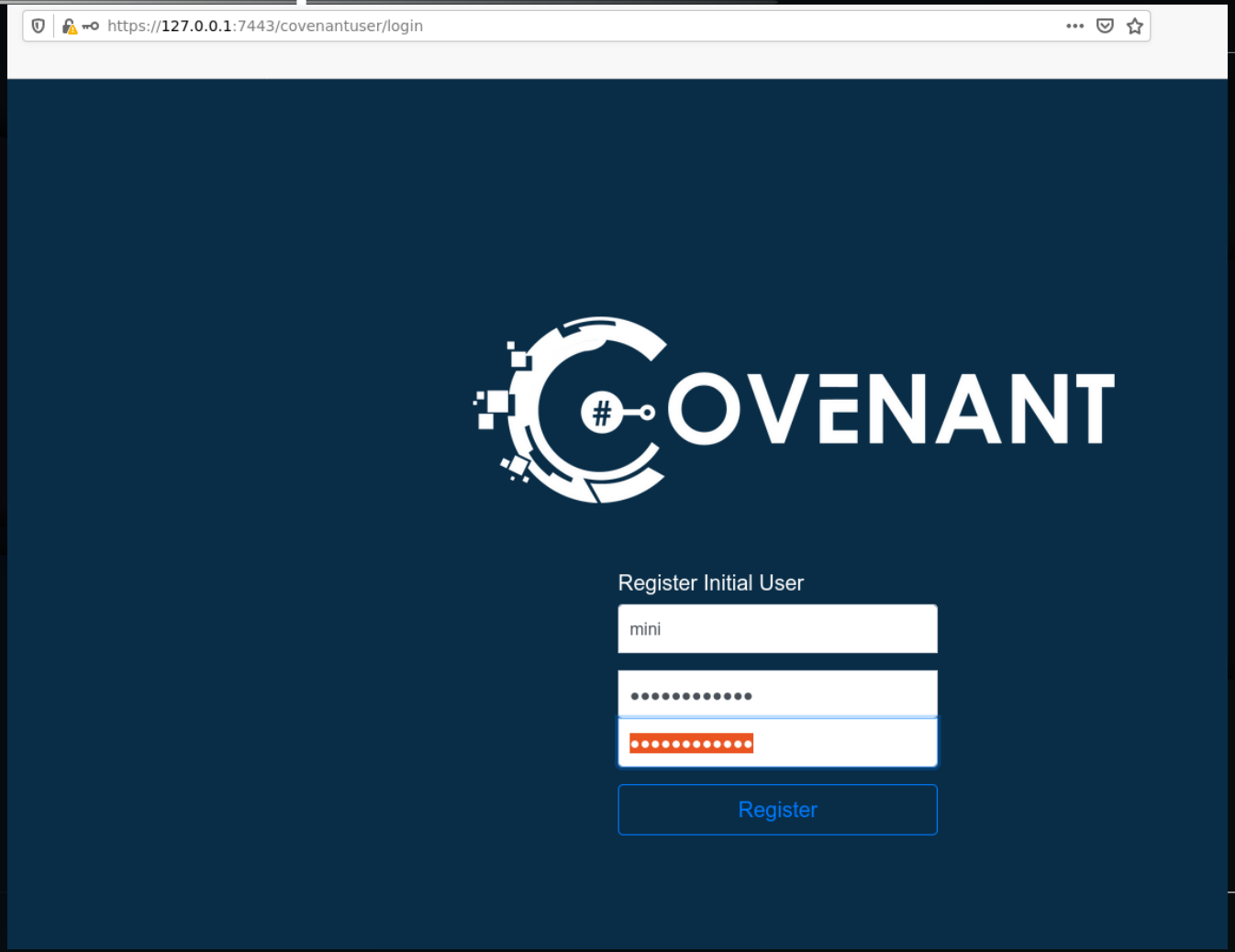
```yaml
---
apiVersion: v1
kind: Service
metadata:
  name: cov-sh-https
  namespace: c2
  labels:
    app: covenant-sh
spec:
  ports:
  - port: 443
    name: cov-sh-https
    targetPort: 443
  selector:
    app: covenant-sh
```

ko apply -f covenant-service.yaml

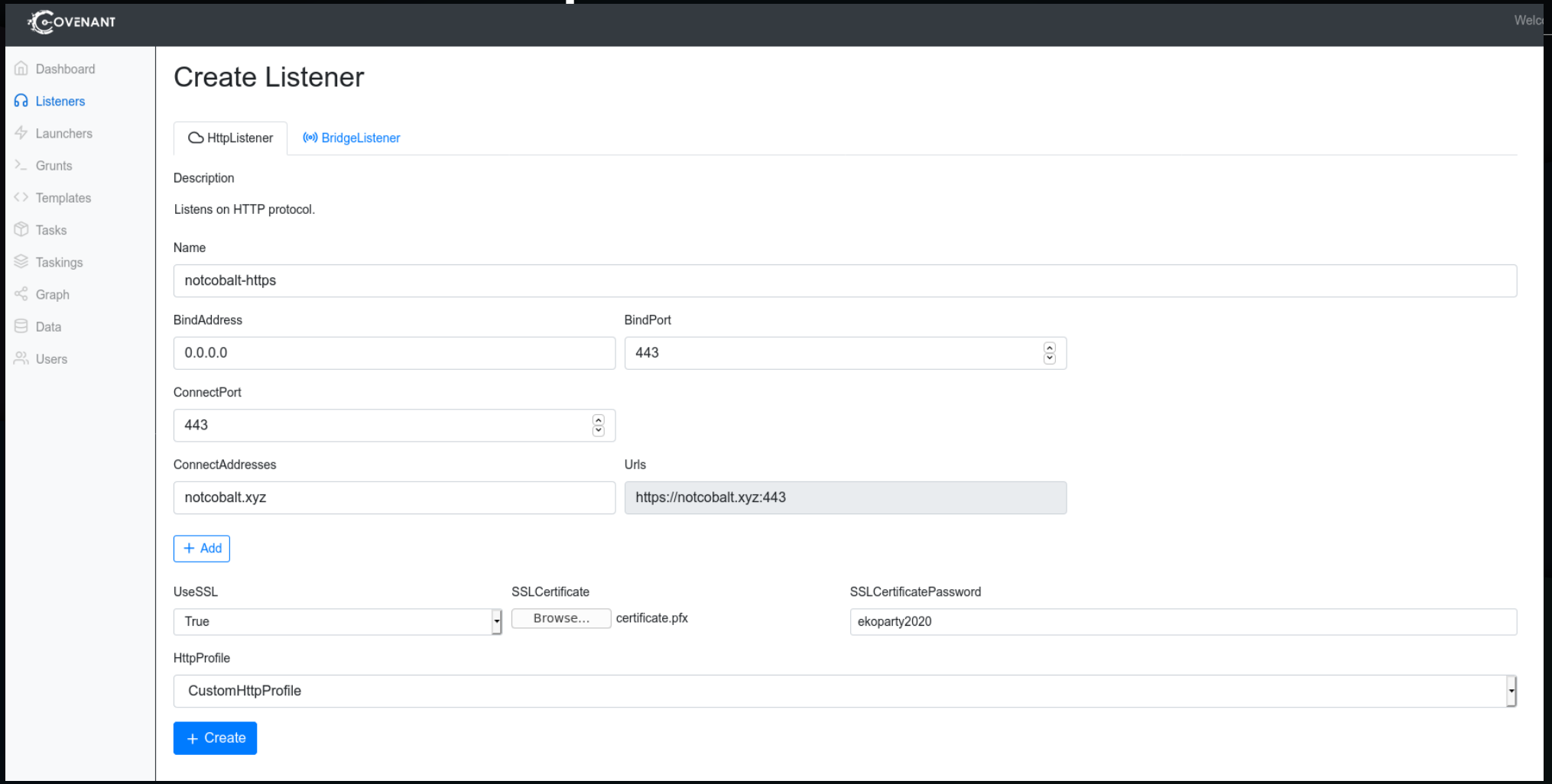
COVENANT/tunel



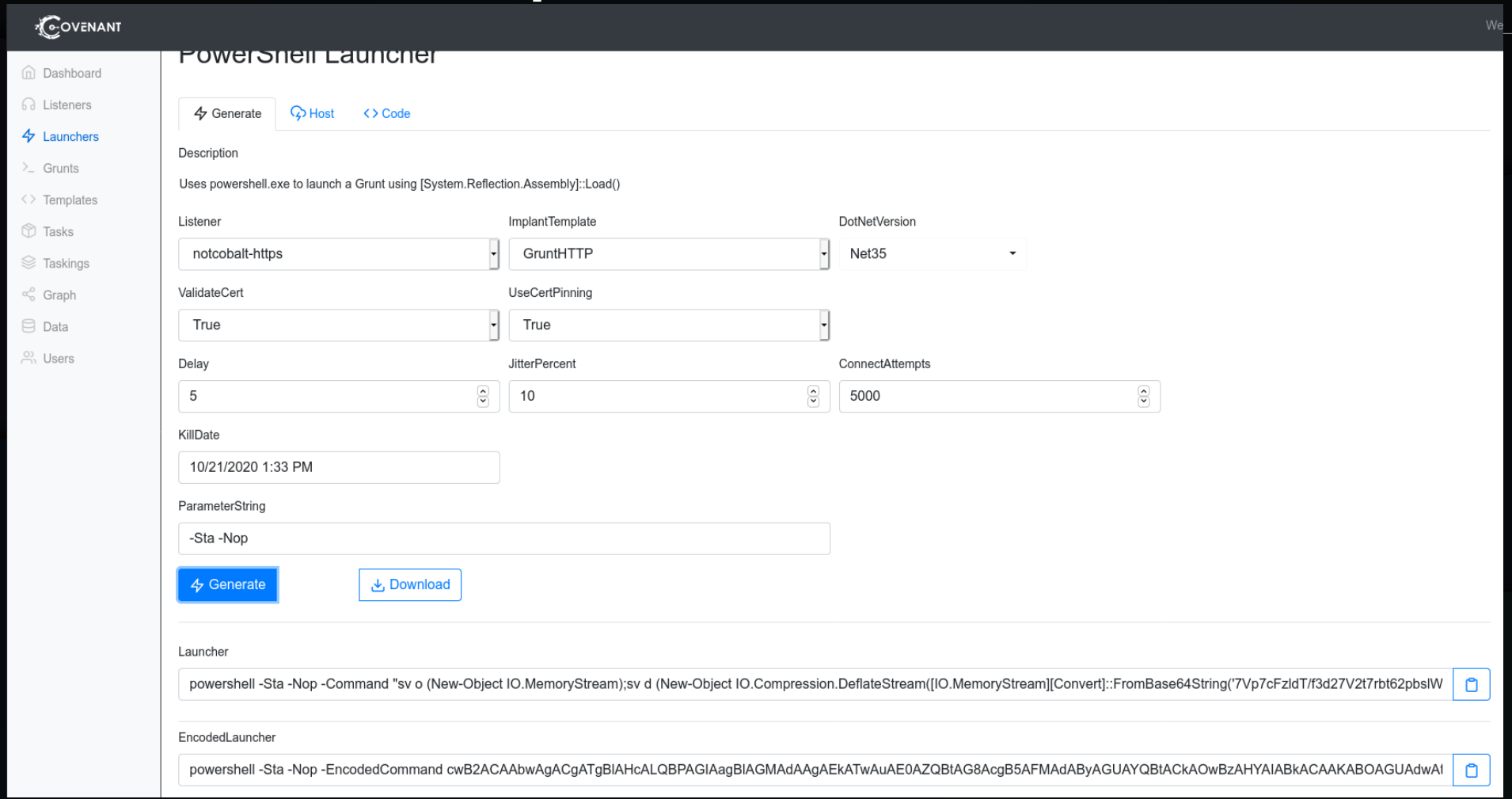
COVENANT/ pseudo DEMO



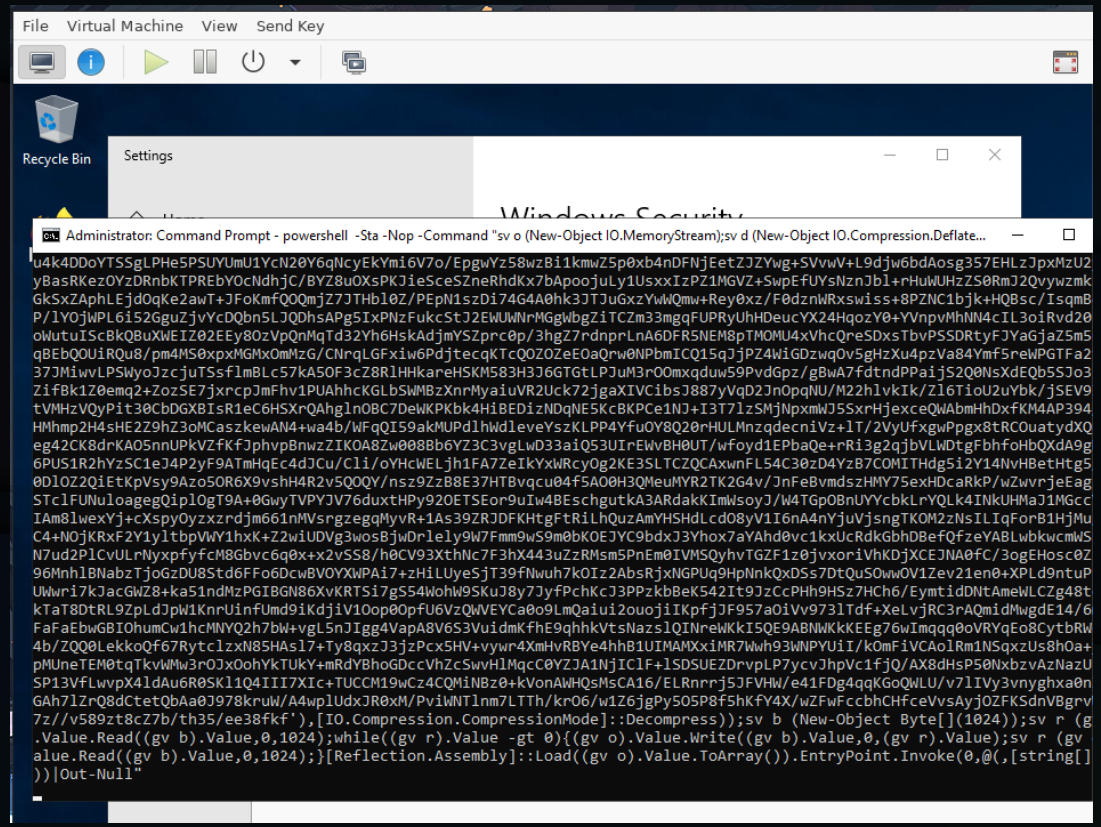
COVENANT/ pseudo DEMO



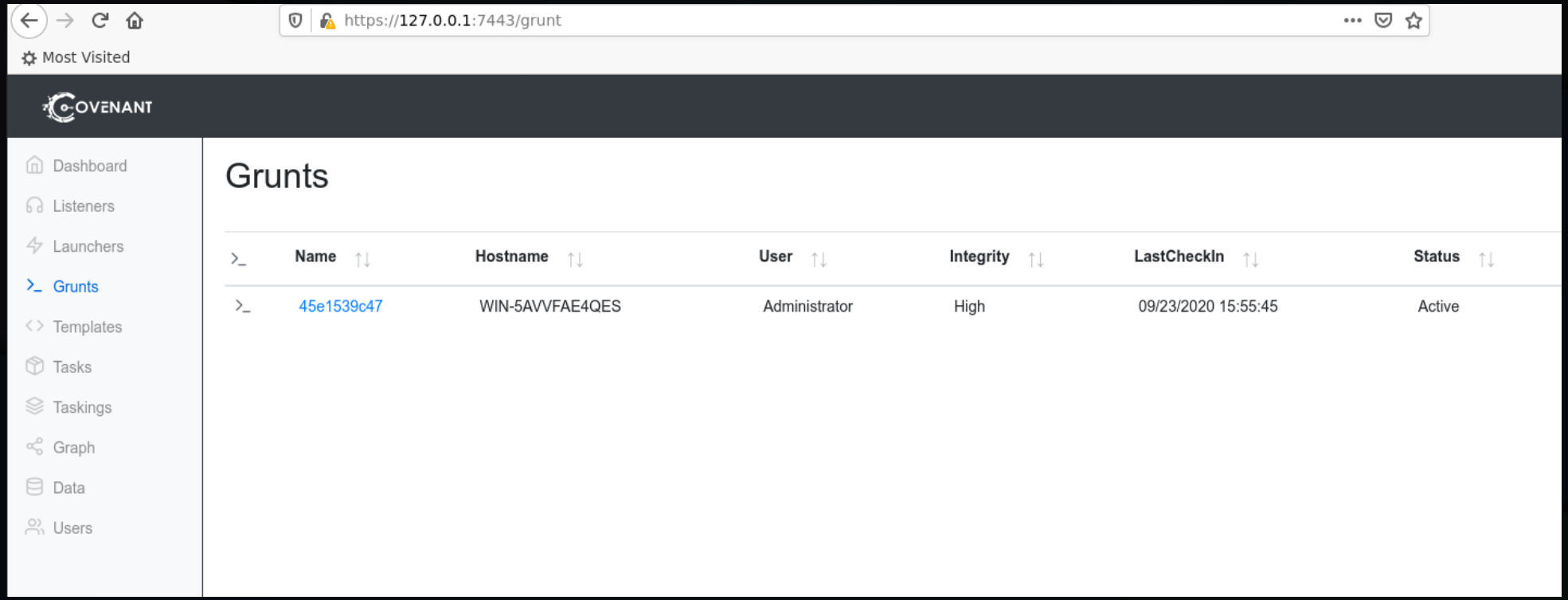
COVENANT/ pseudo DEMO



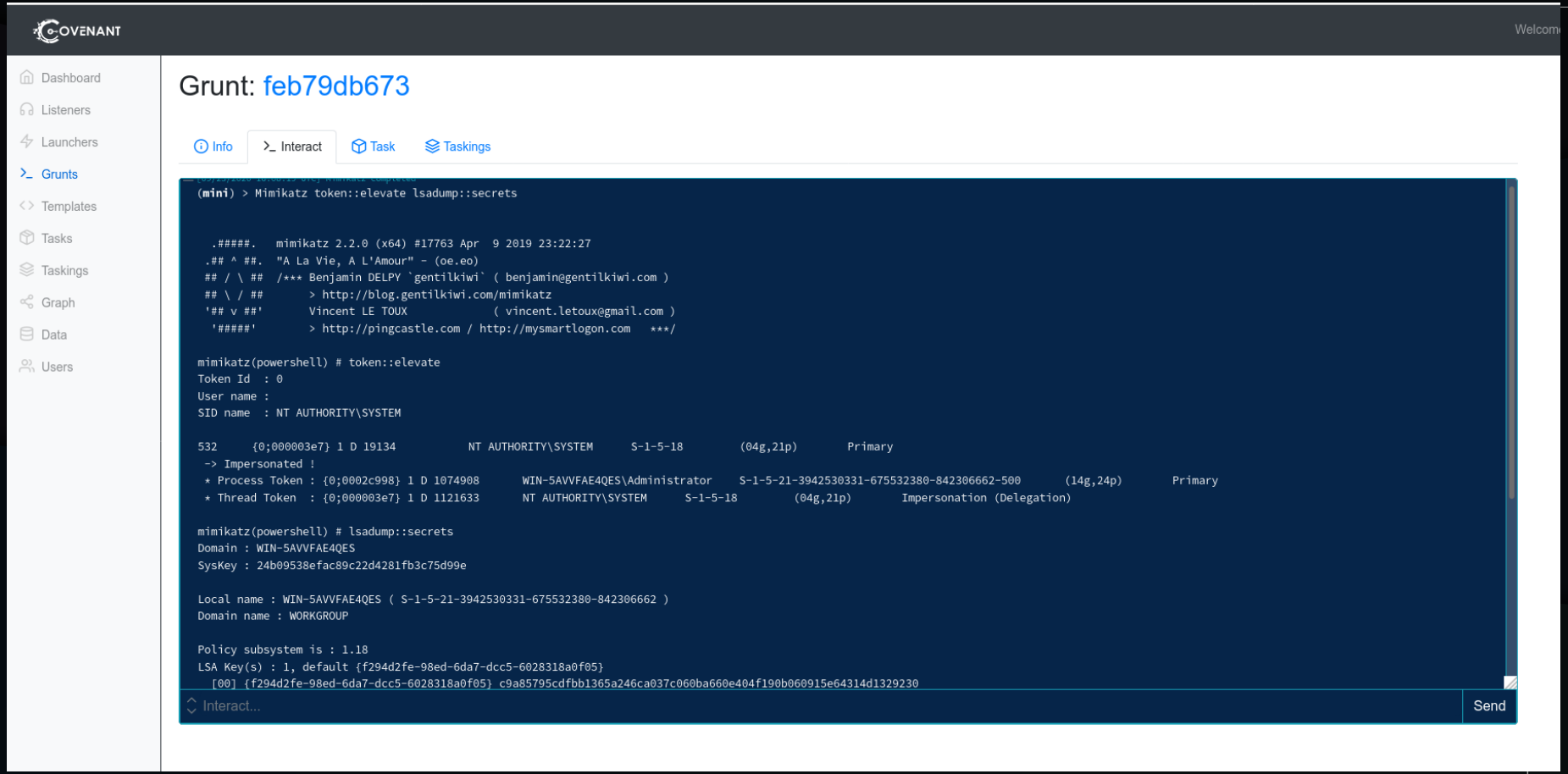
COVENANT/ pseudo DEMO



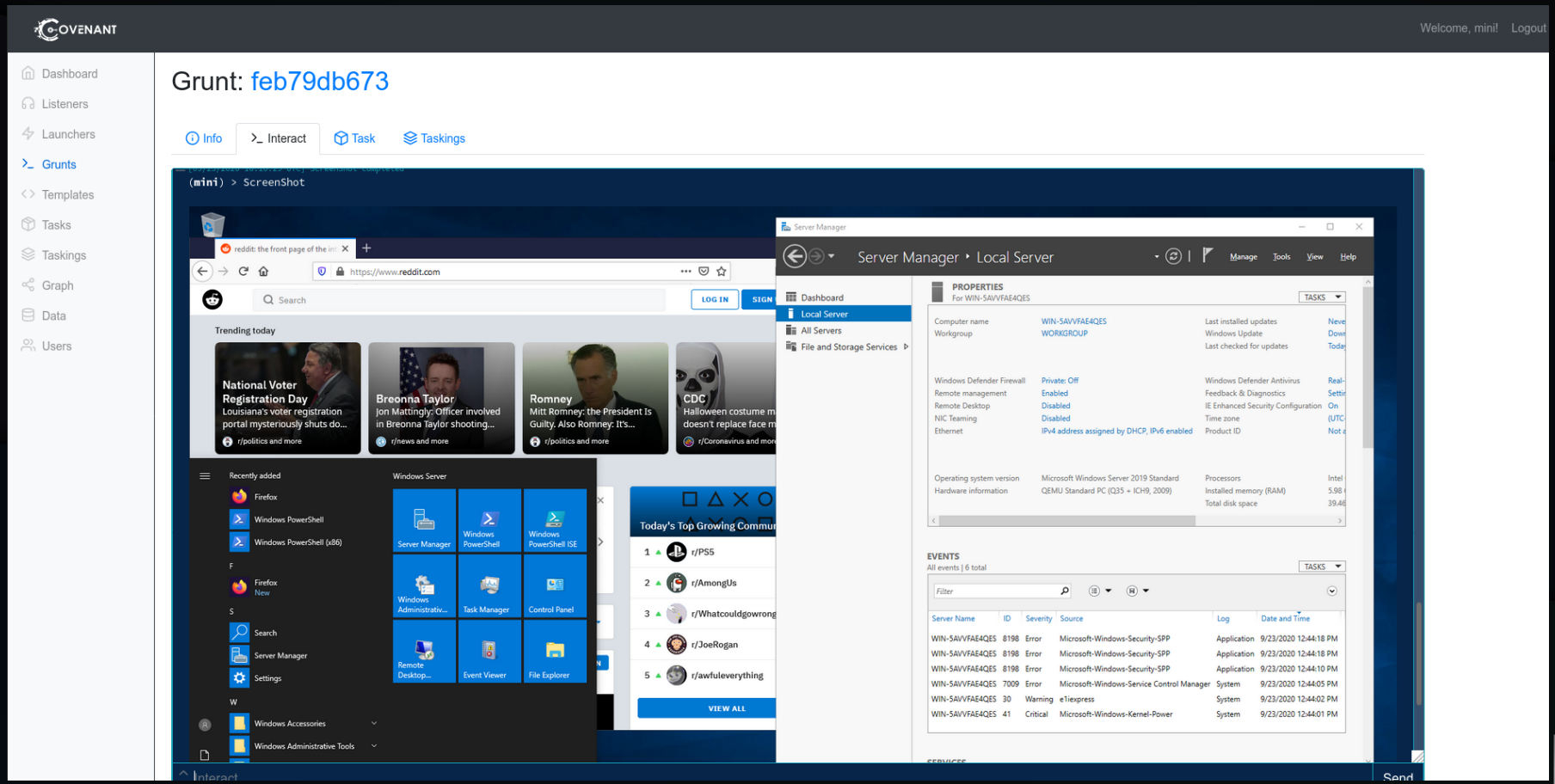
COVENANT/ pseudo DEMO



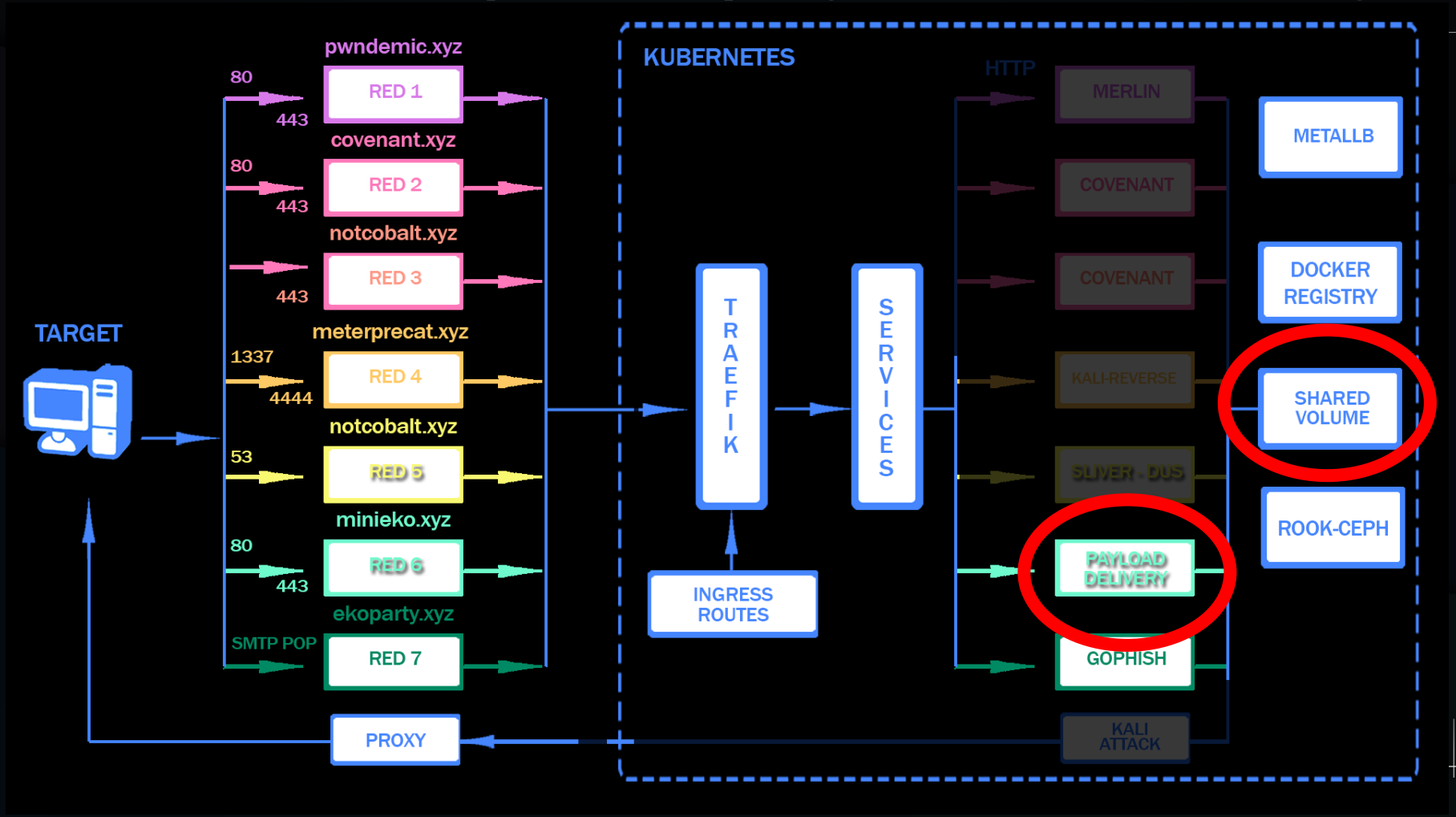
COVENANT/ pseudo DEMO



COVENANT/ pseudo DEMO



Usted esta aquí -> payload delivery



PAYLOAD DELIVERY

- Vamos a necesitar 2 cosas, un servicio con deployment corriendo un webserver y un persisten volume de kubernetes que nos deje compartir el recurso entre distintos pods.

- Persisten Volume (pv) es storage, en lugar de ceph, para este storage vamos a usar un recurso en node1

- Persitent volume claims(pvc) es un pedido de volumen al sistema de storage, como va a ser compartido y queremos que todos los pods lean y escriban va a ser RWX(read write many)

- Como webserver se puede usar nginx,apache con mod_rewrite o lo que gusten :) apache con mod_rewrite esta bueno, quien quiera investigar mas, esta super recomendado.

PAYLOAD DELIVERY/pv y pvc

~: # mkdir /mnt/data

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: shared-pv
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/mnt/data"
```

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: shared-pvc
  namespace: c2
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 3Gi
```

```
ko apply -f 1-shared-pv.yaml
ko apply -f 2-shared-pvc.yaml
```

PAYLOAD DELIVERY/deployment

ko apply -f deploy.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: payloads
  namespace: c2
  labels:
    app: payloads
spec:
  replicas: 1
  selector:
    matchLabels:
      app: payloads
  template:
    metadata:
      labels:
        app: payloads
    spec:
      containers:
      - image: nginx
        ports:
          - containerPort: 80
            protocol: TCP
            name: payloads
        name: payloads
        imagePullPolicy: Always
        volumeMounts:
          - mountPath: "/usr/share/nginx/html"
            name: shared-storage
      volumes:
      - name: shared-storage
        persistentVolumeClaim:
          claimName: shared-pvc
```

PAYLOAD DELIVERY/deployment

ko apply -f svc.yaml

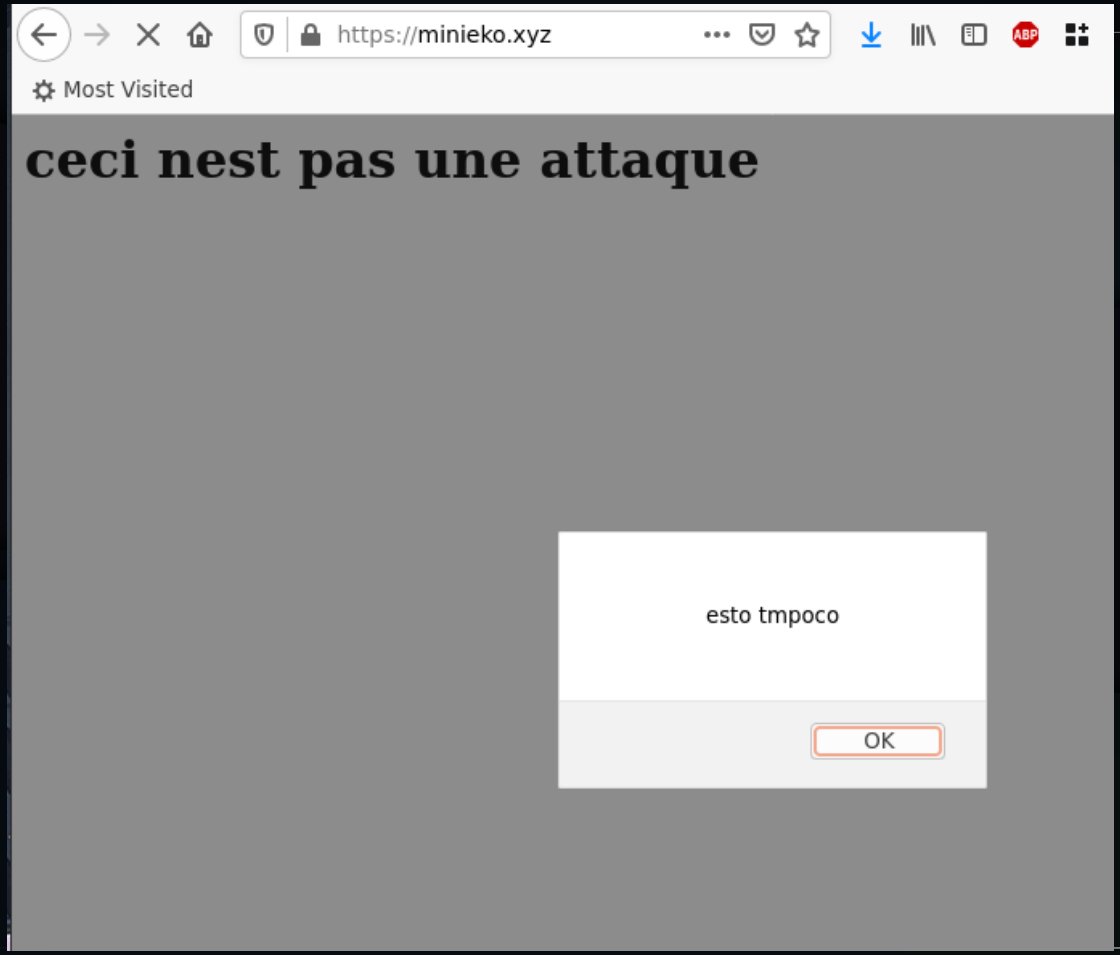
```yaml
apiVersion: v1
kind: Service
metadata:
  name: payloads
  namespace: c2
  labels:
    app: payloads
spec:
  ports:
    - port: 80
      name: payloads
      targetPort: 80
  selector:
    app: payloads
```

PAYLOAD DELIVERY/deployment

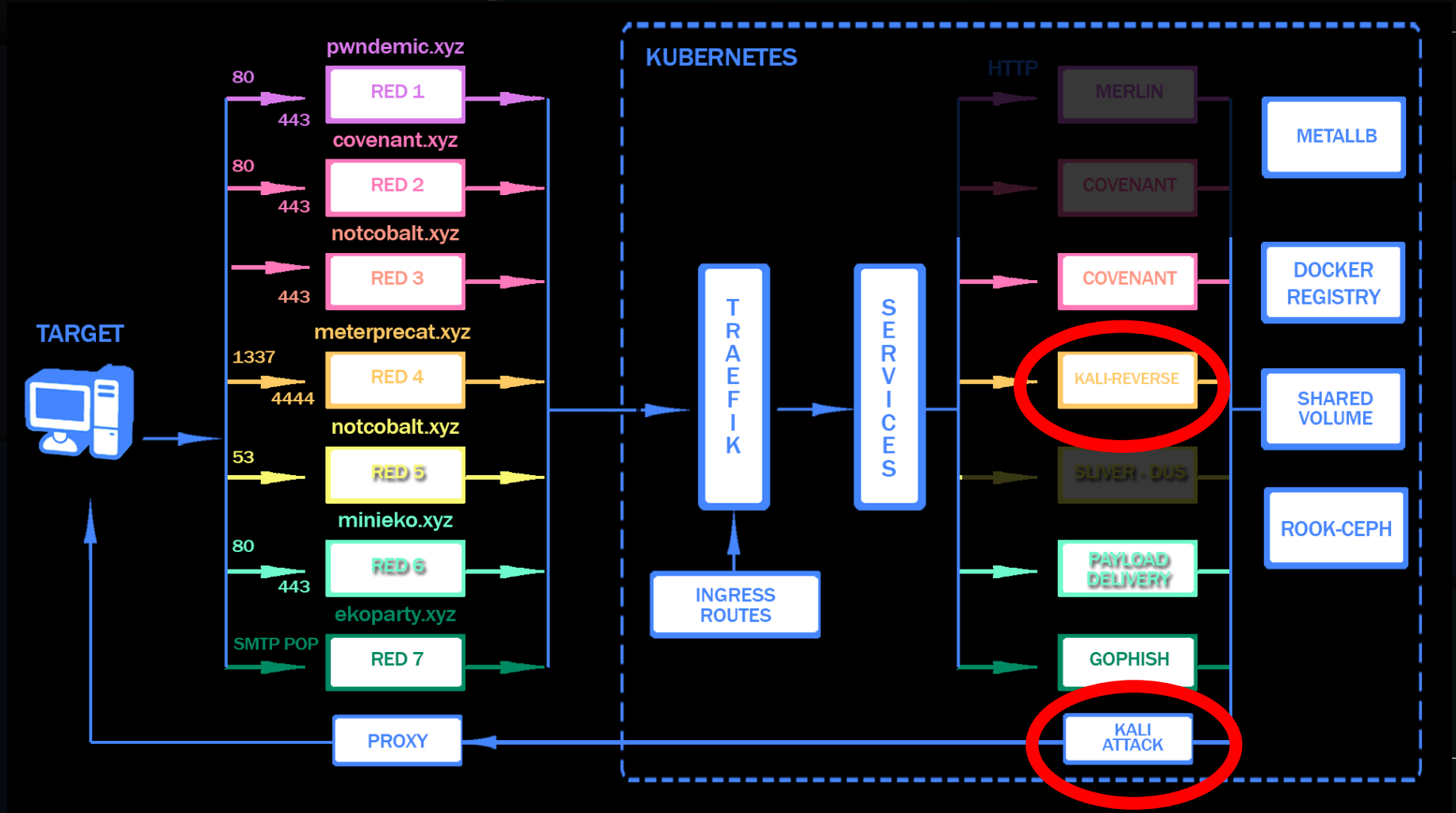
ko apply -f ingress.yaml

```yaml
apiVersion: traefik.containo.us/v1alpha1
kind: IngressRoute
metadata:
  name: payload-https
  namespace: c2
spec:
  entryPoints:
    - https
  routes:
  - kind: Rule
    match: Host(`minieko.xyz`)
    services:
    - kind: Service
      name: payloads
      port: 80
  tls:
    secretName: minieko-tls
```

PAYLOAD DELIVERY/test



Usted esta aquí -> kali



KALI/meterpreter-netcat-oldschool

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: kali-sh
  namespace: c2
  labels:
    app: kali-sh
spec:
  replicas: 1
  selector:
    matchLabels:
      app: kali-sh
  serviceName: kali-sh
  template:
    metadata:
      labels:
        app: kali-sh
    spec:
      containers:
      - image: kalilinux/kali-rolling
        name: kali-sh
        imagePullPolicy: Always
        command: ["/bin/bash"]
        args: ["-c","tail -f /dev/null"]
        ports:
          - name: kali-sh-web
            protocol: TCP
            containerPort: 1337
        volumeMounts:
          - mountPath: "/kali_data"
            name: kali-data
          - mountPath: "/shared-pvc"
            name: shared-storage
      volumes:
      - name: kali-data
      - name: shared-storage
        persistentVolumeClaim:
          claimName: shared-pvc
```

```yaml
  volumeClaimTemplates:
    - metadata:
        name: kali-data
        labels:
          app: kali-sh
      spec:
        accessModes: [ "ReadWriteOnce"]
        storageClassName: rook-ceph-block-x2
        resources:
          requests:
            storage: "16Gi"

```

```
ko apply -f kali-deployment.yaml
```

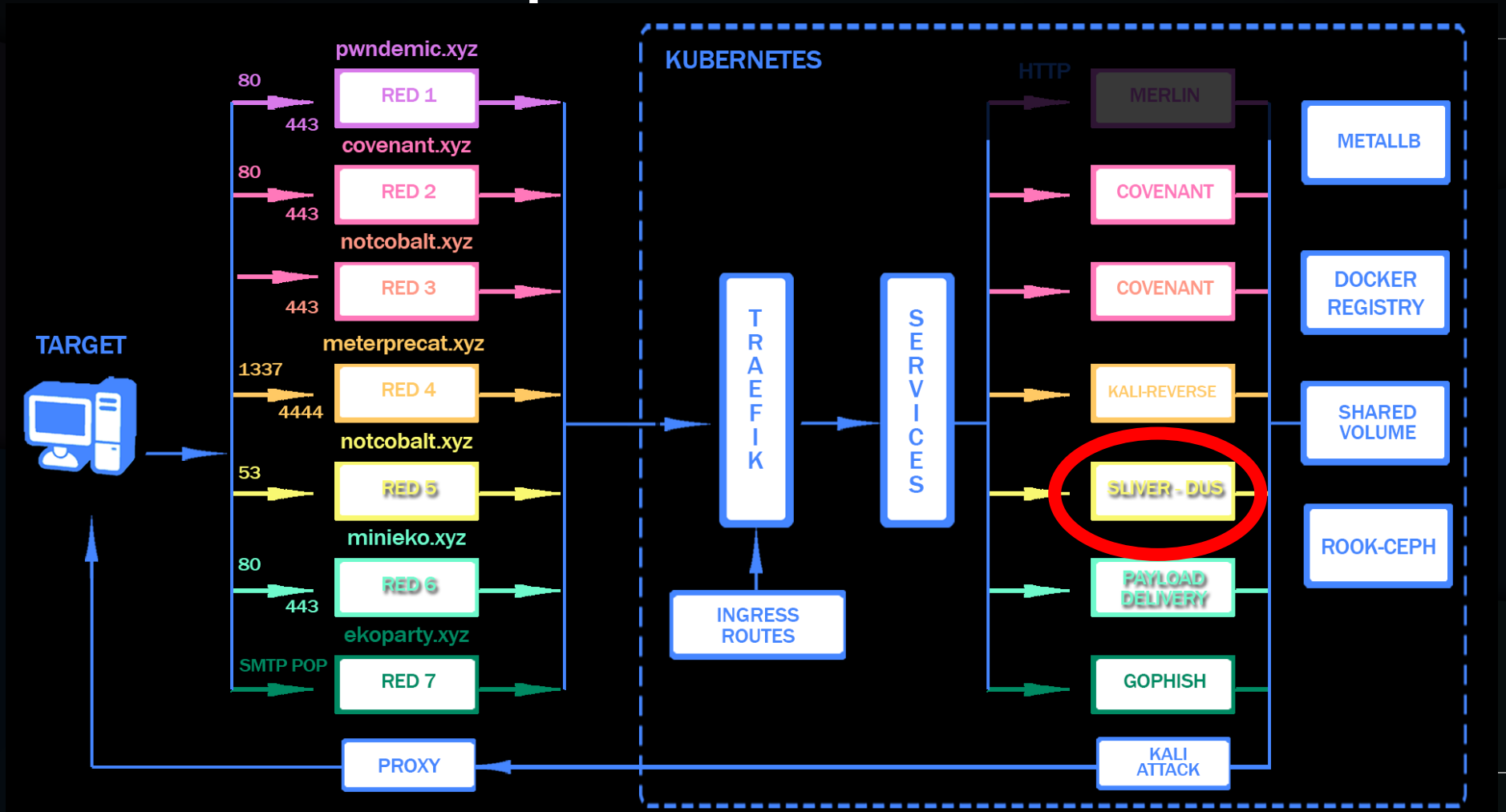
KALI/meterpreter-netcat-oldschool

En lugar de usar traefik, vamos a crear un servicio y que metallb le asigne una ip publica, esto agiliza el proceso de abrir nuevos puertos para futuras reverse shells

Recomendación: abrir bocha de puertos antes :)

```yaml
---
apiVersion: v1
kind: Service
metadata:
  name: kali-sh-web
  namespace: c2
  annotations:
    metallb.universe.tf/address-pool: public
  labels:
    app: kali-sh
spec:
  ports:
    - protocol: TCP
      name: nc-rev-1
      port: 1337
  selector:
    app: kali-sh
  type: LoadBalancer

```

Usted esta aquí -> SLIVER



SLIVER/dns

Setea los dns con las ip del redirector!
Host del NS record, puede ser cualquier cosa

Type	Host	Value
A Record	@	172.105.248.184
A Record	ns1	172.105.248.184
NS Record	1	ns1.notcobalt.xyz.

sliver/dns, svc, udp ingress

```yaml
apiVersion: v1
kind: Service
metadata:
  name: sliver-sh-dns
  namespace: c2
  labels:
    app: sliver-sh
spec:
  ports:
  - port: 53
    protocol: UDP
    name: sliver-sh-dns
    targetPort: 53
  selector:
    app: sliver-sh
```

```yaml
apiVersion: traefik.containo.us/v1alpha1
kind: IngressRouteUDP
metadata:
  name: ingressrouteudp.crd
  namespace: c2

spec:
  entryPoints:
    - dns
  routes:
    - match: HostSNI(`*`)
      services:
        - name: sliver-sh-dns
          port: 53
```

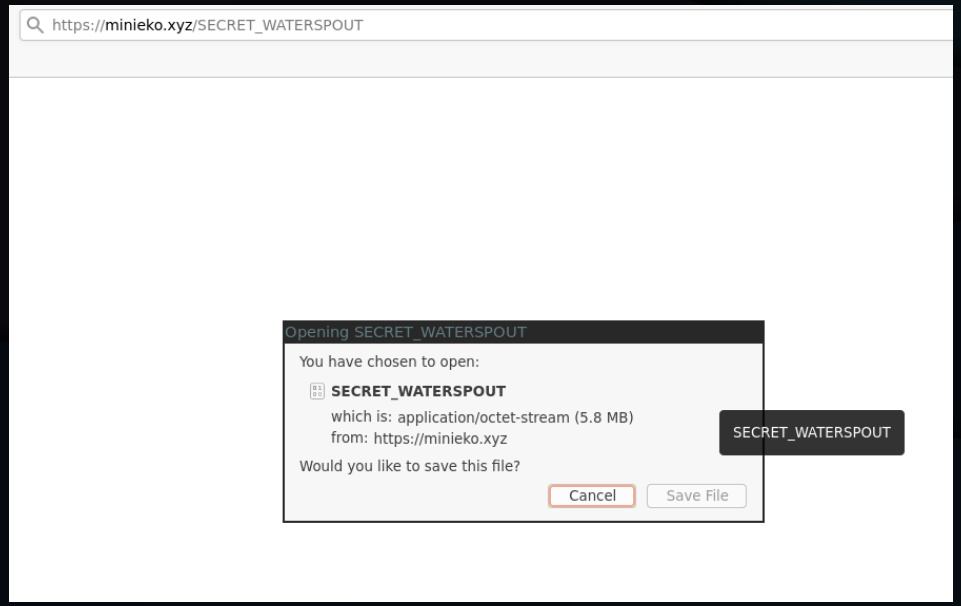
sliver/dns, statefulset

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: sliver-sh
  namespace: c2
  labels:
    app: sliver-sh
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sliver-sh
  serviceName: sliver-sh
  template:
    metadata:
      labels:
        app: sliver-sh
    spec:
      containers:
      - image: 10.233.3.129:5000/sliver
        name: sliver-sh
        imagePullPolicy: Always
        command: ["/bin/bash"]
        args: ["-c","apt update && apt install -y tmux && tail -f /dev/null"]
        ports:
          - name: sliver-sh-web
            protocol: TCP
            containerPort: 80
          - name: sliver-sh-https
            protocol: TCP
            containerPort: 443
        volumeMounts:
          - mountPath: "/app/Data"
            name: sliver-sh-data
          - mountPath: "/shared-storage"
            name: shared-storage
      volumes:
      - name: sliver-sh-data
      - name: shared-storage
        persistentVolumeClaim:
          claimName: shared-pvc
  volumeClaimTemplates:
    - metadata:
        name: sliver-sh-data
        labels:
          app: sliver-sh
      spec:
        accessModes: [ "ReadWriteOnce"]
        storageClassName: rook-ceph-block-x2
        resources:
          requests:
            storage: "16Gi"
```

sliver/dns, pseudo demo

sliver/dns, pseudo demo

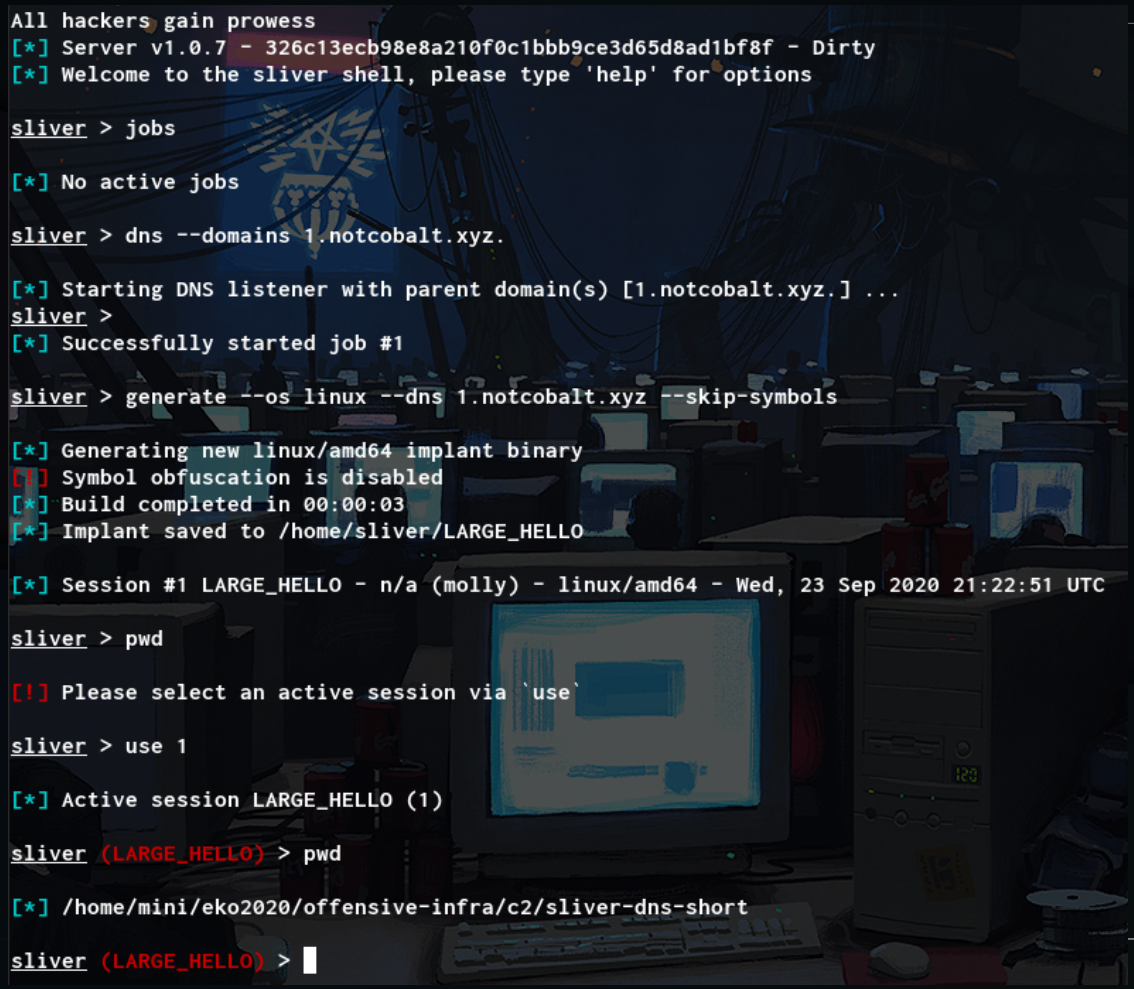
```
root@sliver-sh-0:/home/sliver# cp /home/sliver/SECRET_WATERSPOUT /shared-storage/
root@sliver-sh-0:/home/sliver#
```



target

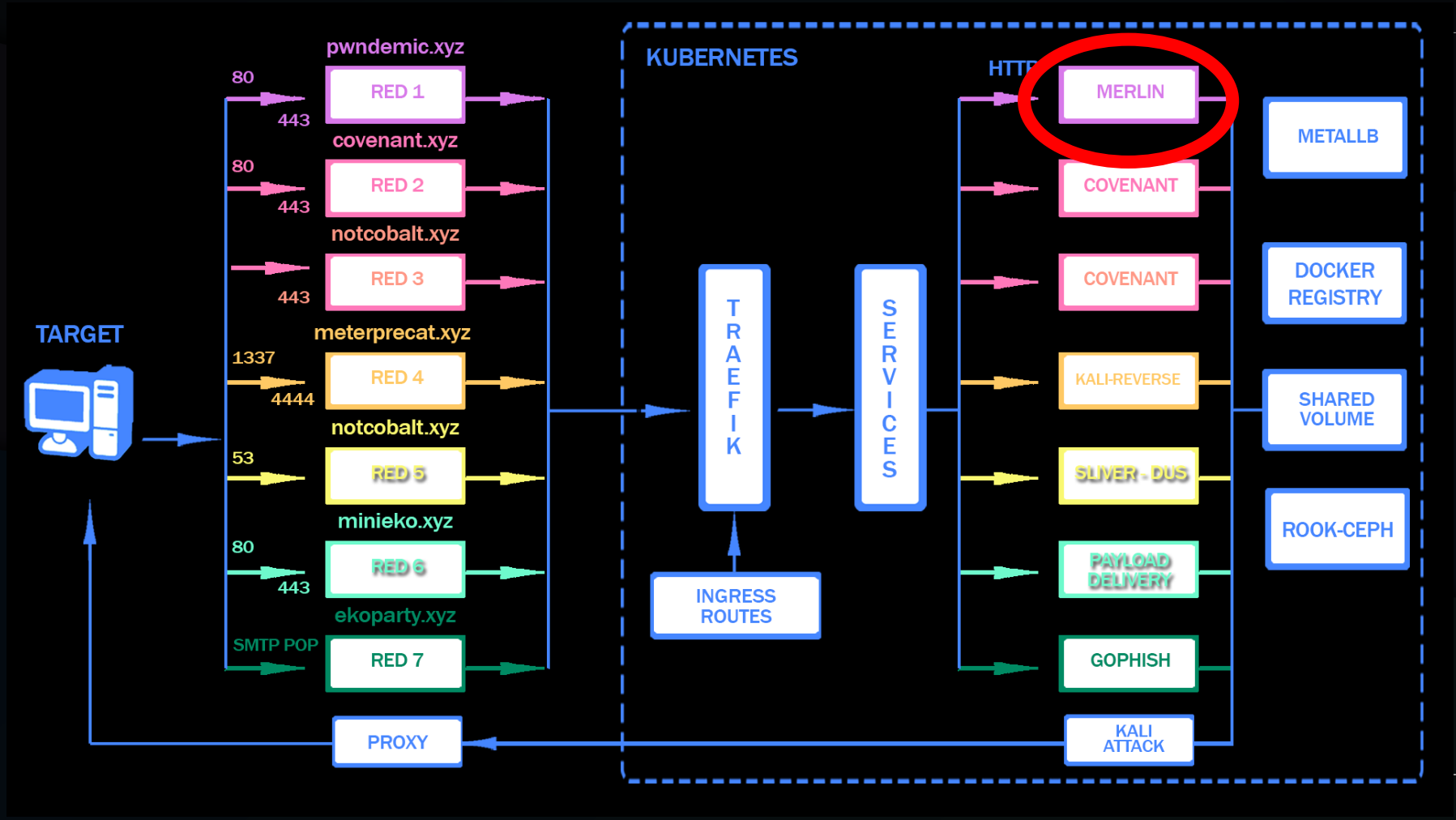
```
root@localhost:~# ./SECRET_WATERSPOUT
```

sliver/dns, pseudo demo



MERLIN

Usted esta aquí -> Merlin



MERLIN/statefulset

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: merlin-lh
  namespace: c2
  labels:
    app: merlin-lh
spec:
  replicas: 1
  selector:
    matchLabels:
      app: merlin-lh
  serviceName: merlin-lh
  template:
    metadata:
      labels:
        app: merlin-lh
    spec:
      containers:
      - image: 10.233.3.129:5000/merlin
        name: merlin-lh
        imagePullPolicy: Always
        command: ["/bin/bash"]
        args: ["-c","tail -f /dev/null"]
        ports:
          - name: melrin-lh-web
            protocol: TCP
            containerPort: 80
          - name: merlin-lh-https
            protocol: TCP
            containerPort: 443
        volumeMounts:
          - mountPath: "/app/Data"
            name: merlin-lh-data
          - mountPath: "/shared-pvc"
            name: shared-storage
      volumes:
      - name: merlin-lh-data
      - name: shared-storage
        persistentVolumeClaim:
          claimName: shared-pvc
  volumeClaimTemplates:
    - metadata:
        name: merlin-lh-data
        labels:
          app: merlin-lh
      spec:
        accessModes: [ "ReadWriteOnce"]
        storageClassName: rook-ceph-block-x2
        resources:
          requests:
            storage: "16Gi"
```

ko apply -f merlin-stateful.yaml

MERLIN/services

```yaml
apiVersion: v1
kind: Service
metadata:
  name: merlin-lh-web
  namespace: c2
  labels:
    app: merlin-lh
spec:
  ports:
  - port: 80
    name: merlin-lh-web
    targetPort: 80
  selector:
    app: merlin-lh
---
```

```yaml
---
apiVersion: v1
kind: Service
metadata:
  name: merlin-lh-https
  namespace: c2
  labels:
    app: merlin-lh
spec:
  ports:
  - port: 443
    name: merlin-lh-https
    targetPort: 443
  selector:
    app: merlin-lh
```

ko apply -f merlin-svc.yaml

MERLIN/ingressroute

```yaml
apiVersion: traefik.containo.us/v1alpha1
kind: IngressRoute
metadata:
  name: merlin-lh-http
  namespace: c2
spec:
  entryPoints:
    - web
  routes:
  - kind: Rule
    match: Host(`pwndemic.xyz`)
    services:
    - kind: Service
      name: merlin-lh-web
      port: 80
---
```

```yaml
---
apiVersion: traefik.containo.us/v1alpha1
kind: IngressRoute
metadata:
  name: merlin-lh-https
  namespace: c2
spec:
  entryPoints:
    - https
  routes:
  - kind: Rule
    match: Host(`pwndemic.xyz`)
    services:
    - kind: Service
      name: merlin-lh-https
      port: 443
  tls:
    secretName: pwndemic-tls
```

ko apply -f merlin-ingressroutes.yaml

DEMO

Recap

Recap

1- Usamos Mitre ATT&CK y la **CYBER** kill chain para mapear un adversario

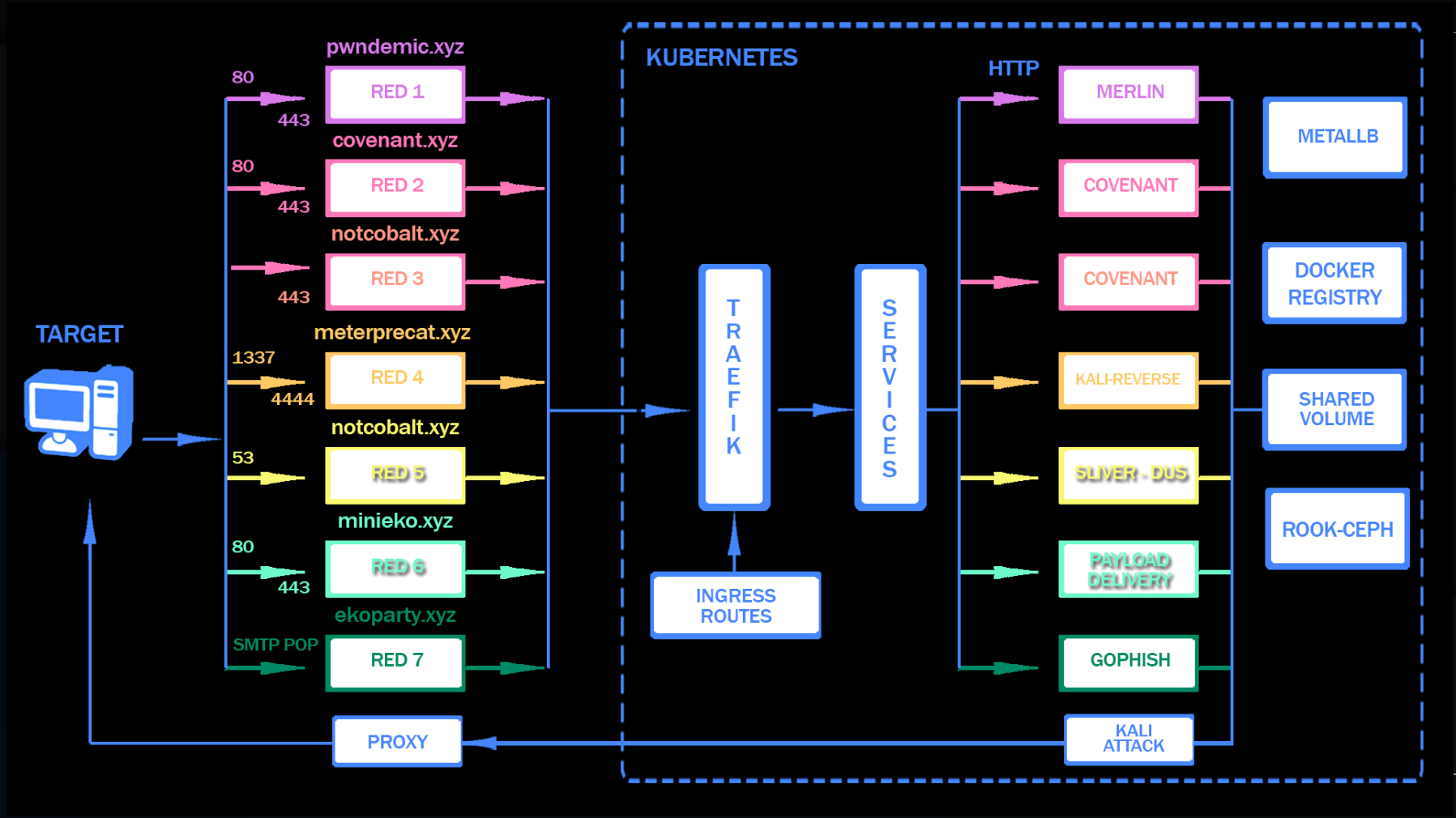
2- En funcion a este adversario abstracto definimos requerimientos para la infra

3- Elegimos nuestro stack (kubernetes + vps) y las herramientas para provisionar y configurar (Terraform, ansible)

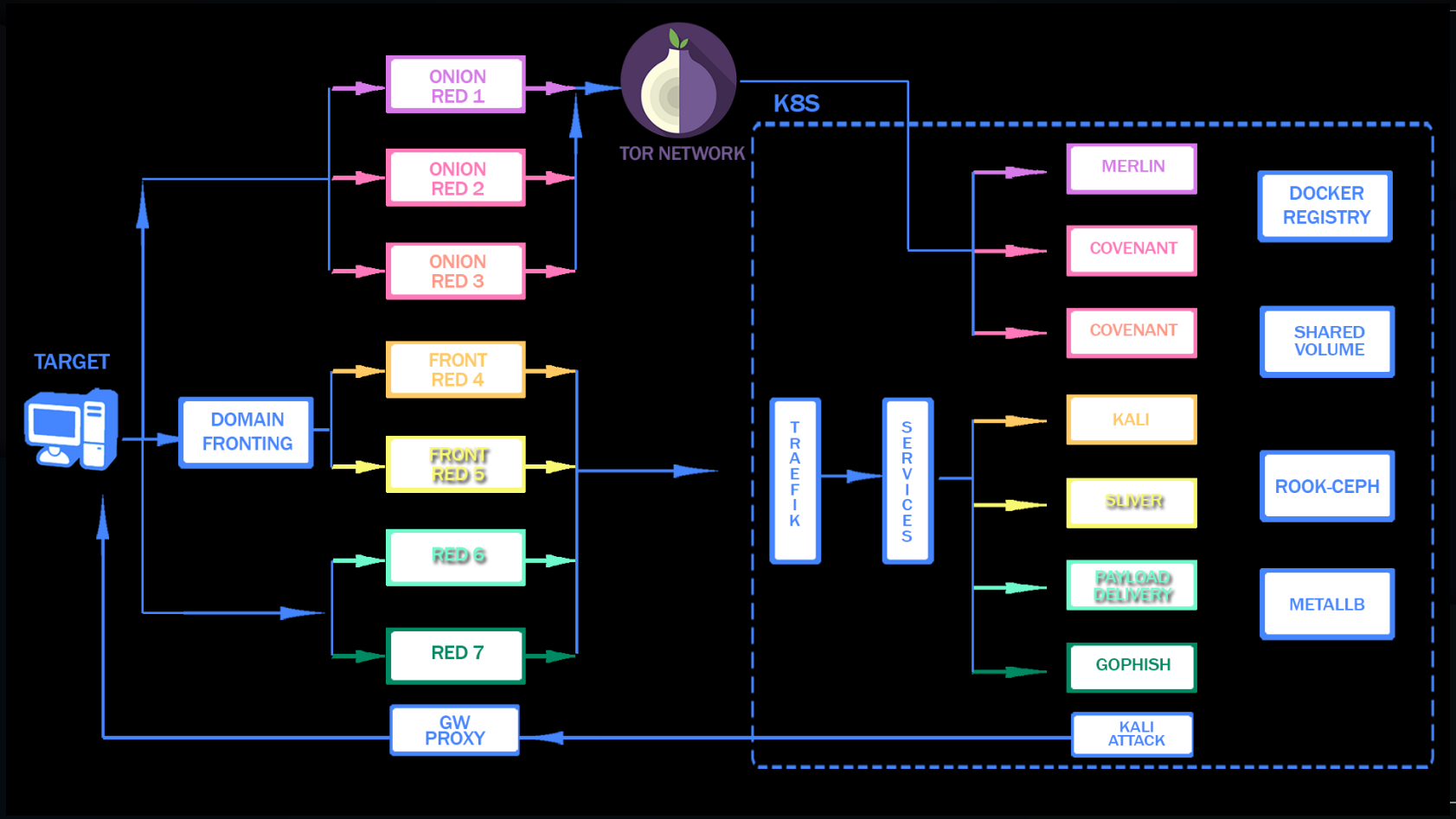
4- Provisionamos las tools que nos gustaban (covenant,sliver,kali,merlin)

5- Testeamos que todo funcionara

Recap infra 1



Recap infra 2



GRACIAS!

usen barbijo

mini@kaktheplanet.xyz