Implementacja i testy skalowalności systemu wideokonferencyjnego

2025-11-30

Jakub Jędrzejczyk **Sebastian Kwaśniak** **Anna Berkowska**

Wdrożenie Jitsi

Konfiguracja Jitsi

jitsi-values.yaml

```yaml
publicURL: "jitsi.google.sex.pl"

jvb:
  useHostPort: true
  stunServers: 'turn:158.101.210.198:3478?transport=tcp,turn:158.101.210.198:3478?transport=udp'
  publicIPs:
    - '158.101.210.198'
  service:
    enabled: true
    type: ClusterIP
    externalTrafficPolicy: ""

prosody:
  transcriber:
    enabled: false
  extraEnvFrom:
    - secretRef:
        name: jitsi-turn-secret

web:
  extraEnvsFrom:
    - secretRef:
        name: jitsi-turn-secret
```

Uruchomienie Jitsi

turn-secret.yaml

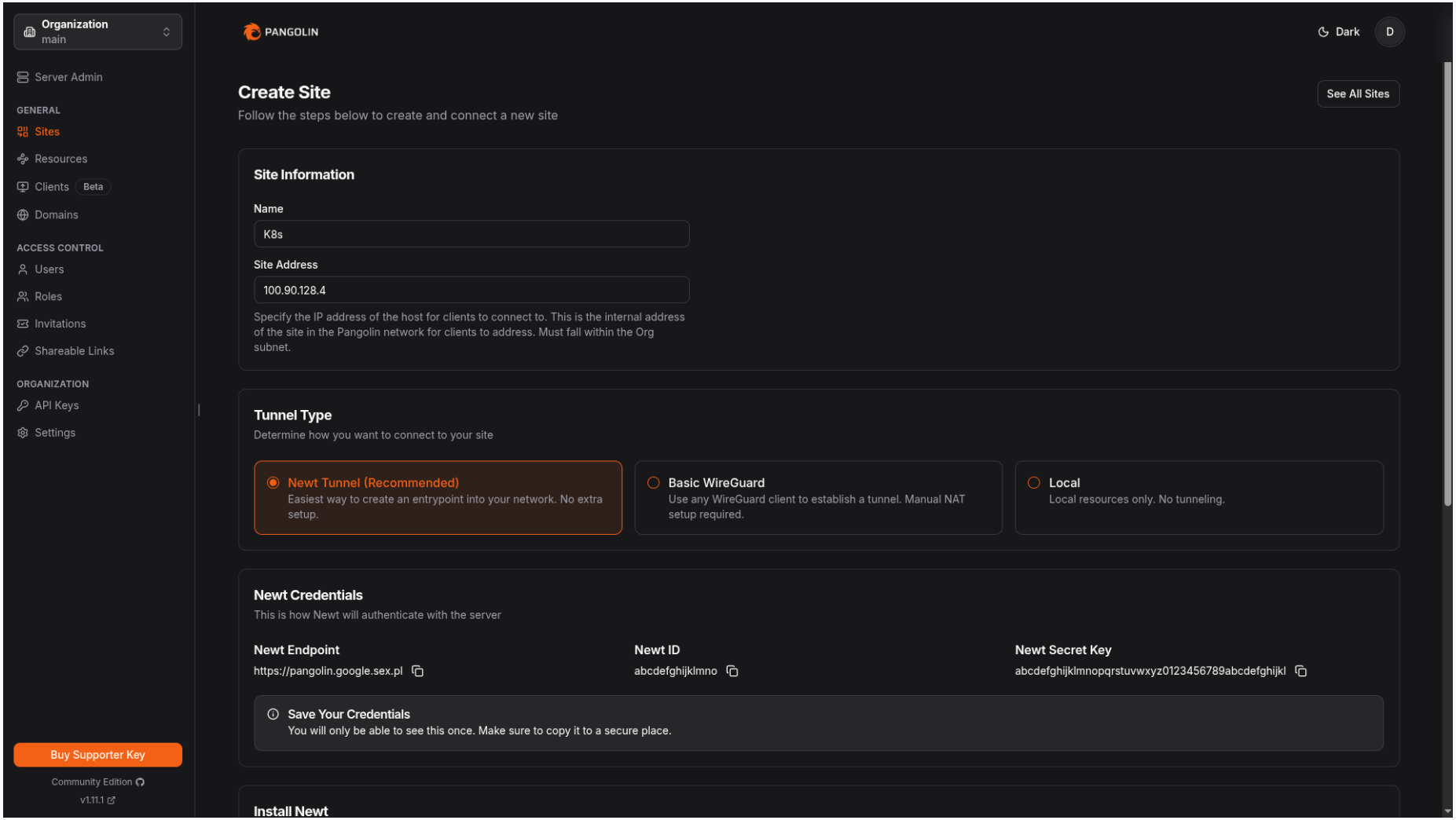
```yaml
apiVersion: v1
kind: Secret
metadata:
  name: jitsi-turn-secret
type: Opaque
stringData:
  TURN_CREDENTIALS: "abcdefghijklmnopqrstuvwxyz012345"
  TURN_HOST: "158.101.210.198"
  TURN_PORT: "3478"
```

```
# postawienie samego jitsi

kubectl apply -f turn-secret.yaml
helm install myjitsi jitsi/jitsi-meet --values values.yaml
```

Udostępnienie Jitsi bez publicznego adresu IP

Pangolin - Tworzenie zdalnego dostępu



Pangolin - Uruchomienie zdalnego dostępu

newt-cred.env

```
PANGOLIN_ENDPOINT=https://pangolin.google.sex.pl
NEWT_ID=abcdefghijkmnop
NEWT_SECRET=abcdefghijkmnopqrstuvwxyz0123456789abcdefghijkmn
```

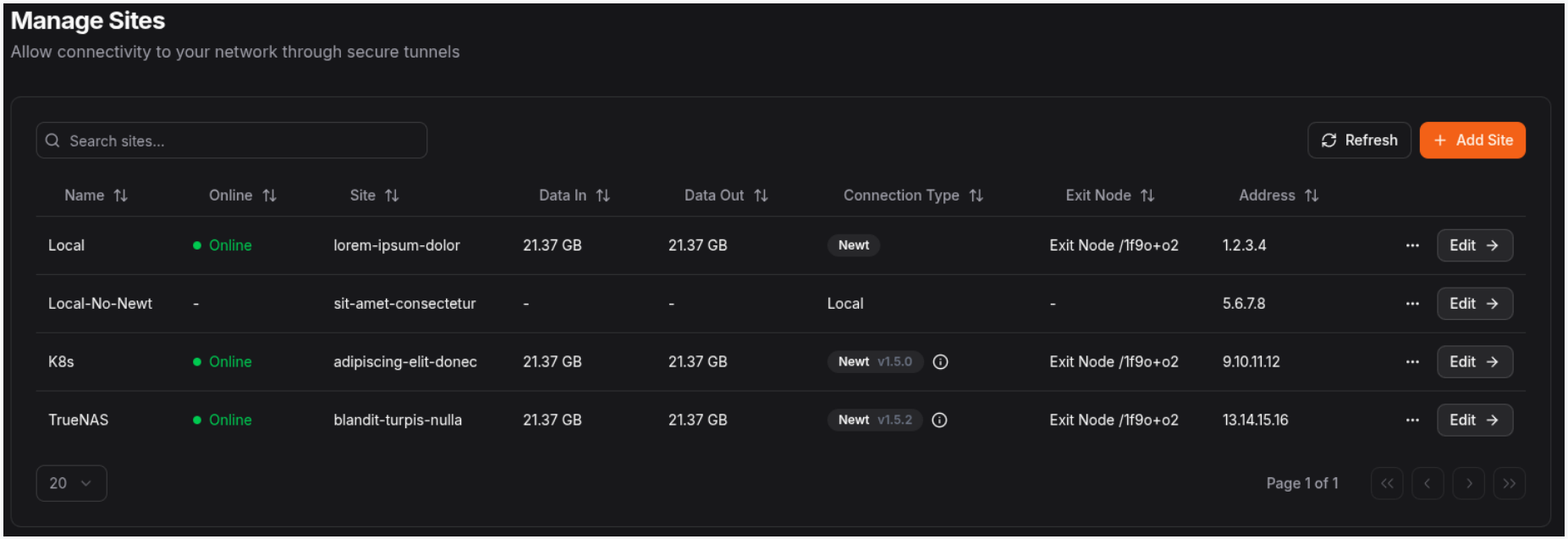
values-newt.yaml

```yaml
newtInstances:
  - name: main
    enabled: true
    auth:
      existingSecretName: newt-cred
      keys:
        endpointKey: PANGOLIN_ENDPOINT
        idKey: NEWT_ID
        secretKey: NEWT_SECRET
```

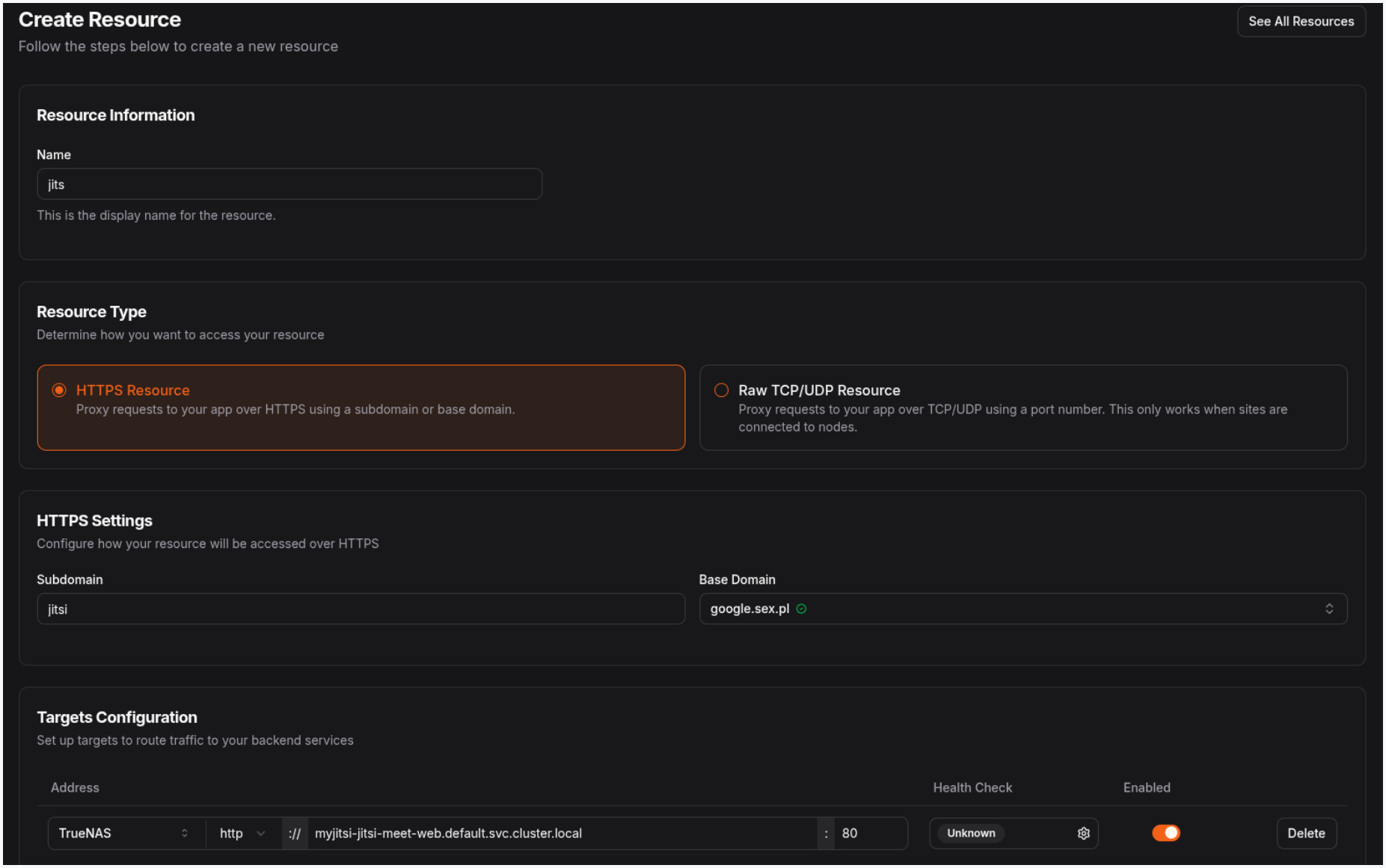
```
# dodanie repo

helm repo add fossorial https://charts.fossorial.io
```

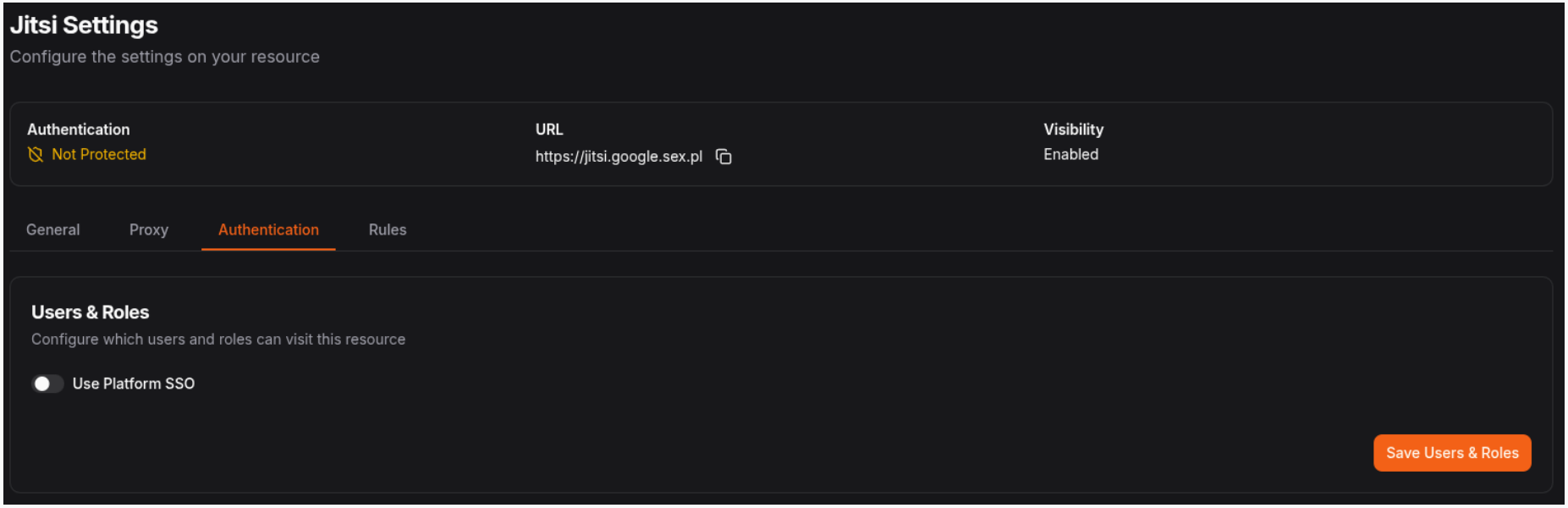
Pangolin - Zdalny Dostęp

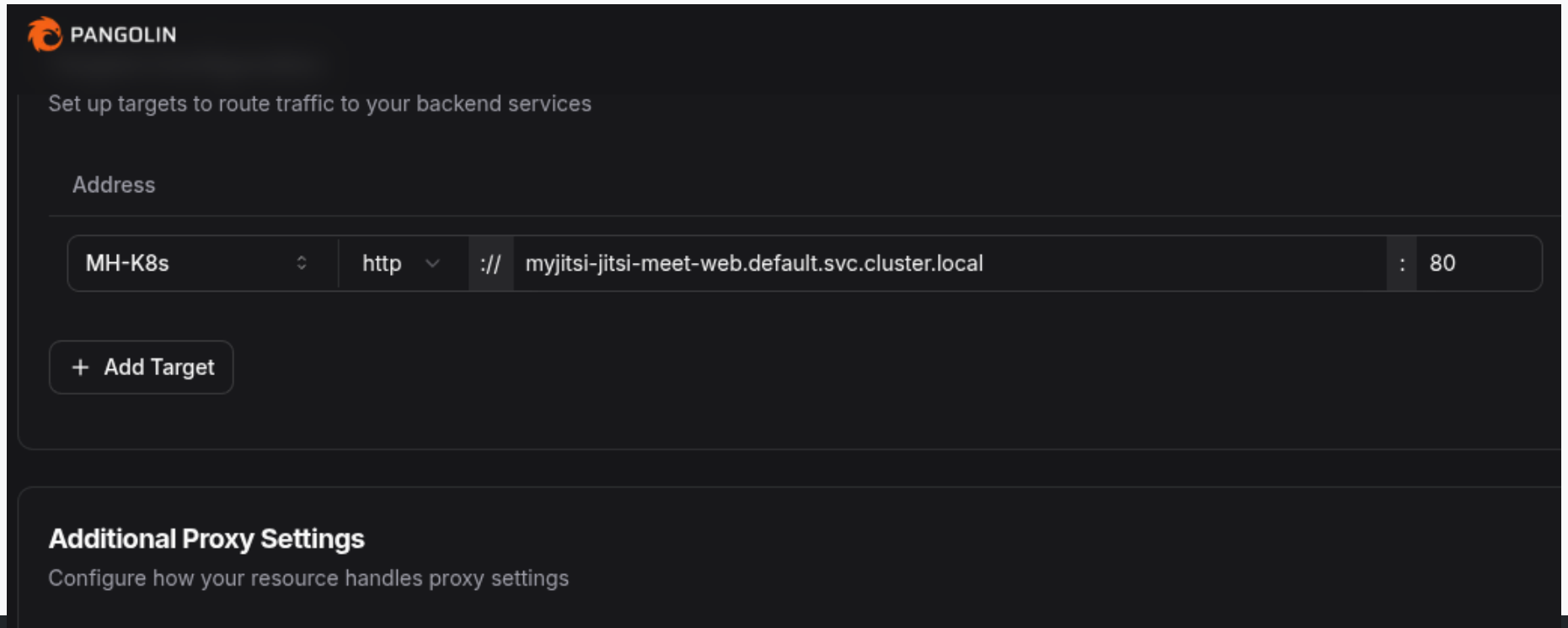


Pangolin - Tworzenie Zasobu

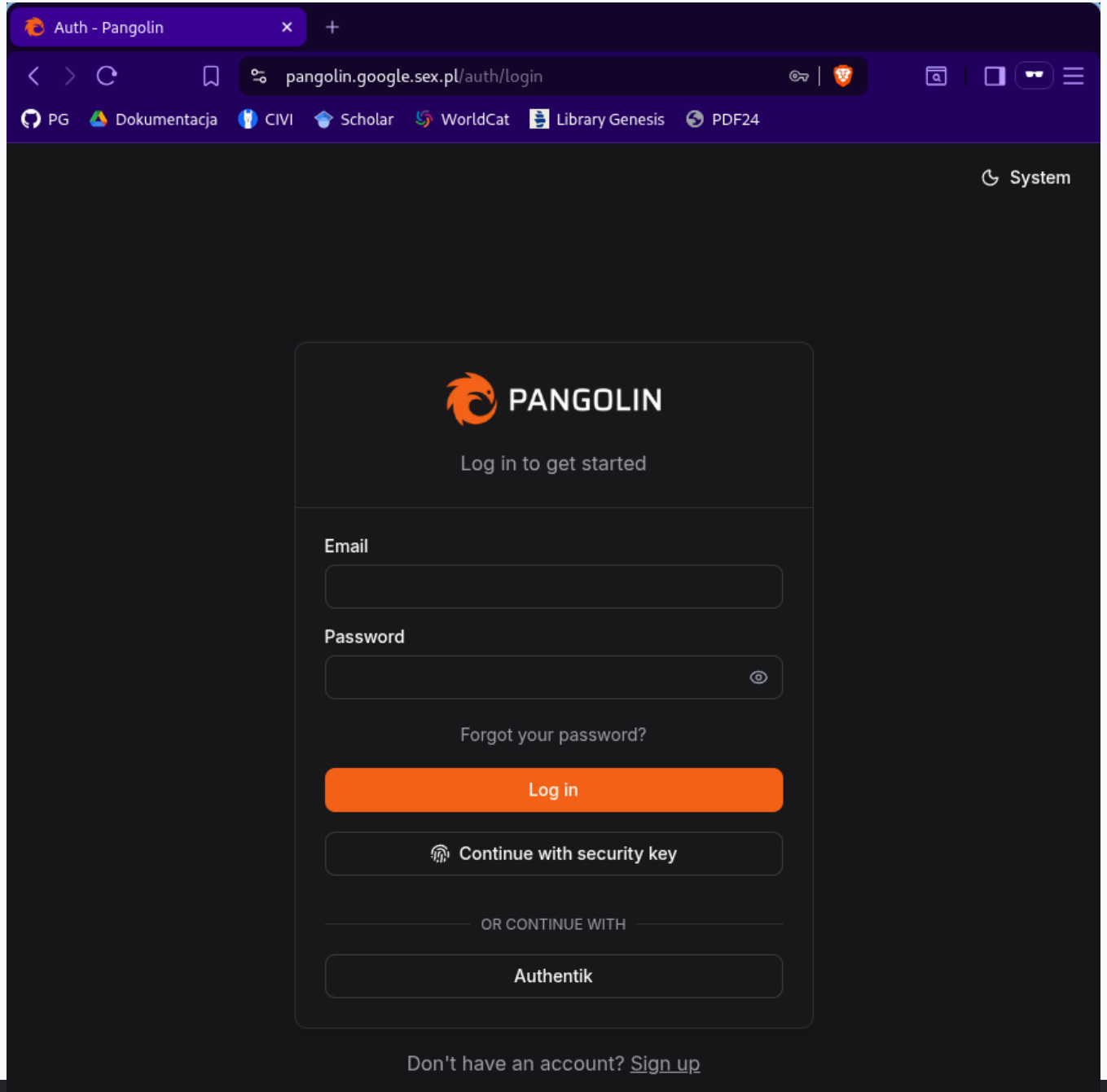


Pangolin - SSL + Wyłączenie SSO





Tailscale - Czemu wyłączyć SSO?



Tailscale - Pliki konfiguracyjne

tailscale-secret.yaml

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: tailscale-auth
stringData:
  TS_AUTHKEY: tskey-auth-abcdefghijklmnopq-rstuvwxyz1234567890abcdefghijklmn
```

Tailscale - pliki konfiguracyjne cz. 2

tailscale-rbac.yaml

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: tailscale

---

apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: tailscale
rules:
  - apiGroups: [""]
    resourceNames: ["tailscale-auth"]
    resources: ["secrets"]
    verbs: ["get", "update", "patch"]

---

apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: tailscale
subjects:
  - kind: ServiceAccount
    name: tailscale
roleRef:
```

Tailscale - Pliki konfiguracyjne cz. 3

tailscale-proxy.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: tailscale-proxy
spec:
  serviceAccountName: tailscale
  initContainers:
    - name: sysctler
      image: busybox:latest
      securityContext:
        privileged: true
      command: ["/bin/sh"]
      args:
        - -c
        - sysctl -w net.ipv4.ip_forward=1 net.ipv6.conf.all.forwarding=1
  containers:
    - name: tailscale
      image: ghcr.io/tailscale/tailscale:latest
      env:
        - name: TS_KUBE_SECRET
          value: tailscale-auth
        - name: TS_AUTHKEY
          valueFrom:
            secretKeyRef:
              name: tailscale-auth
              key: TS_AUTHKEY
        - name: TS_USERSPACE
          value: "false"
        - name: TS_DEST_IP
          value: 10.108.40.240
      securityContext:
        privileged: true
```

Tailscale - Dlaczego Takie IP?

```
root@jitsi-1:/home/student/tailscale# kubectl get services
NAME                            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                                  AGE
kubernetes                      ClusterIP   10.96.0.1        <none>        443/TCP                                  26d
myjitsi-jitsi-meet-jvb          ClusterIP   10.108.40.240    <none>        10000/UDP                                31h
myjitsi-jitsi-meet-jvb-metrics  ClusterIP   10.108.15.188    <none>        9888/TCP                                 31h
myjitsi-jitsi-meet-web          ClusterIP   10.106.197.43    <none>        80/TCP                                   31h
myjitsi-prosody                 ClusterIP   10.108.213.244   <none>        5280/TCP,5281/TCP,5347/TCP,5222/TCP,5269/TCP   31h
root@jitsi-1:/home/student/tailscale#
```

Tailscale - uruchomienie

```
19  # dodanie repo
20
21  helm repo add tailscale https://pkgs.tailscale.com/helmcharts
22  helm repo update
23
24  # postawienie tailscale
25
26  kubectl apply -f tailscale-secret.yaml
27  kubectl apply -f tailscale-rbac.yaml
28  kubectl apply -f tailscale-proxy.yaml
```

Tailscale - Przekierowanie z VPS

MACHINE	ADDRESSES	VERSION	LAST SEEN
tailscale-proxy	100.105.219.104	1.90.9 Linux 6.12.48+deb13-amd64	Connected

```
30  # routing w iptables
31
32  iptables -t nat -A PREROUTING -p udp --dport 10000 -j DNAT --to-destination 100.105.219.104:10000
33  iptables -t nat -A POSTROUTING -j MASQUERADE
```

TURN - konfiguracja

turnserver.conf

```
realm=google.sex.pl
external-ip=158.101.210.198
fingerprint
lt-cred-mech

static-auth-secret=abcdefghijklmnopqrstuvwxyz012345
```

TURN - docker-compose

docker-compose.yaml

Run All Services

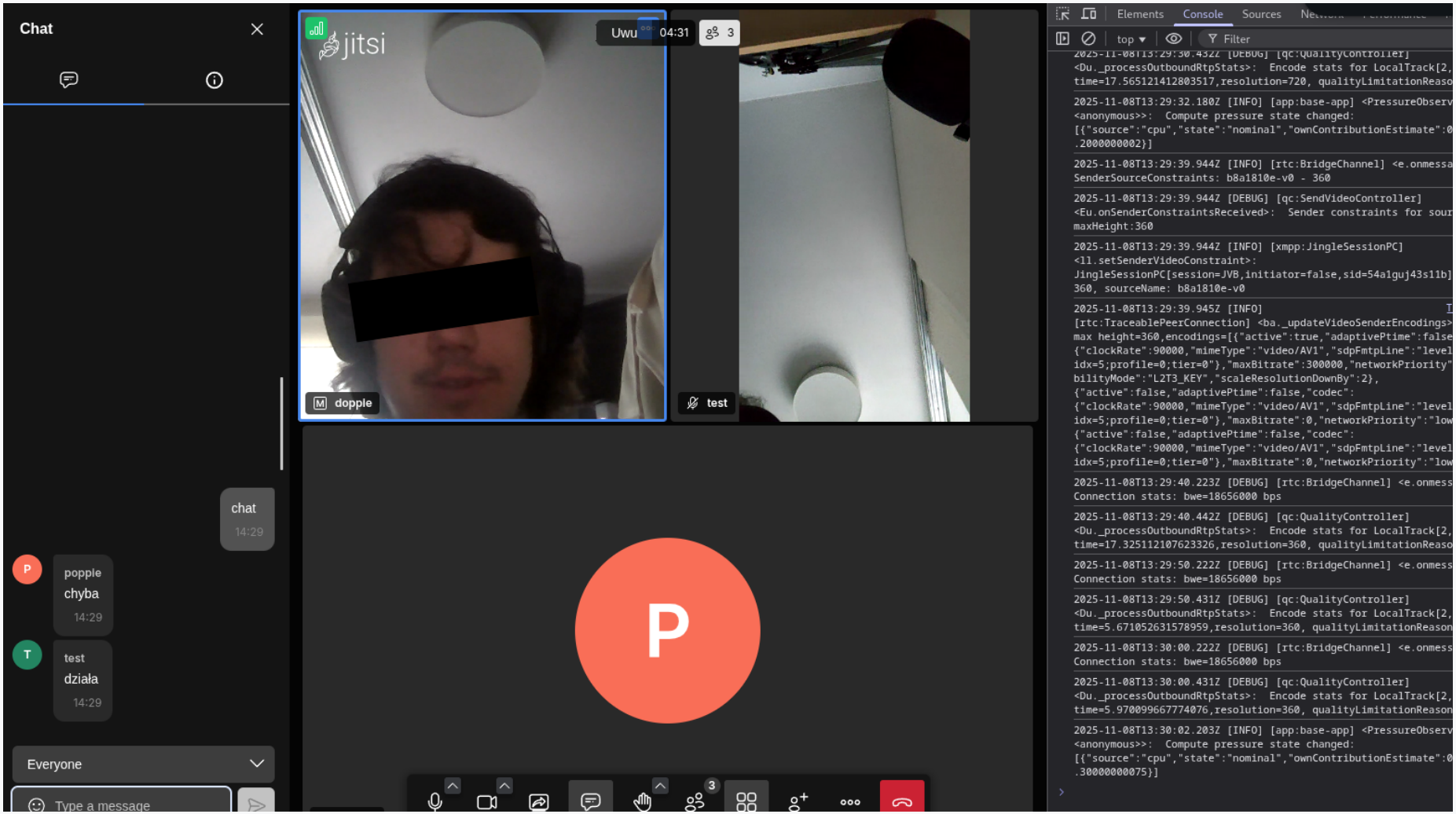
```yaml
services:
```

Run Service

```yaml
  coturn:
    image: coturn/coturn:4.6.3      # pin to the current stable tag
    container_name: coturn
    restart: unless-stopped
    ports:
      # STUN/TURN
      - "3478:3478"
      - "3478:3478/udp"
      # TLS-TURN
      - "5349:5349"
      - "5349:5349/udp"
      # RTP/RTCP relays (adjust range if you need fewer ports)
      - "49000-49020:49000-49020/udp"
    environment:
      # Secure long-term credentials (generate once; keep secret)
      STATIC_AUTH_SECRET: "${TURN_STATIC_AUTH_SECRET}"
      EXTERNAL_IP: "158.101.210.198"
    volumes:
      # Bind your custom conf + persistent database & logs
      - "/gdzie/tylko/chcesz/coturn/turnserver.conf:/etc/coturn/turnserver.conf:ro"
      - "coturn-data:/var/lib/coturn"
      - "coturn-logs:/var/log"
    # Use host networking if you prefer not to publish individual ports
    # network_mode: "host"

volumes:
  coturn-data:
```

Efekt - Działające (nierozproszone) Jitsi



Skalowanie jvb

Co zostało dodane do values.yaml?

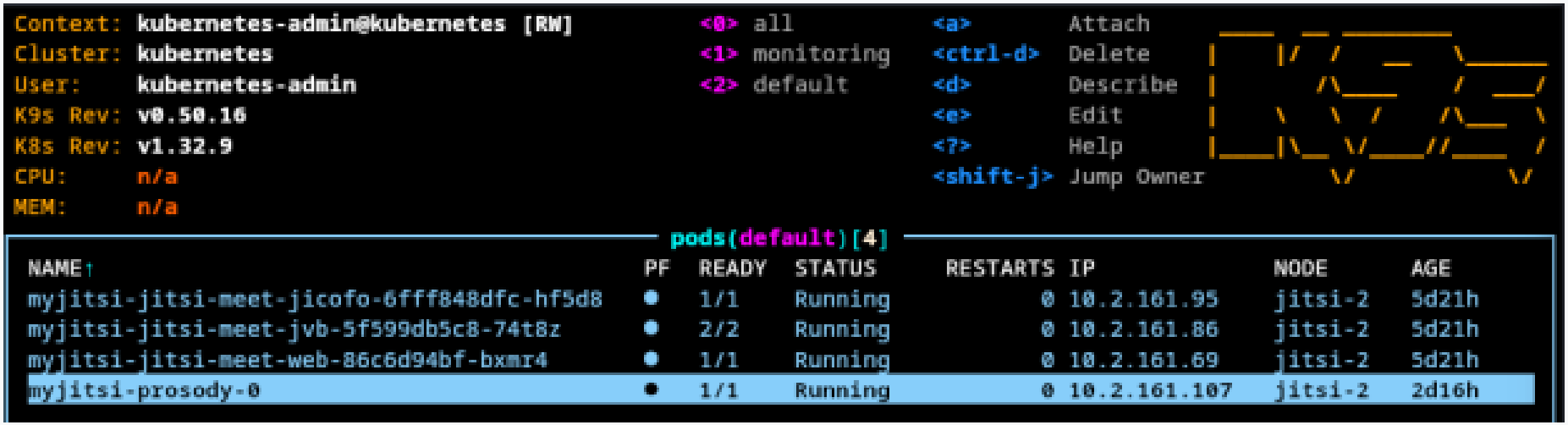
- Aby przeskalować jvb, następujące wartości zostały dodane:

```
jvb:
  ## Set JVB instance count:
  replicaCount: 2
  ## Expose JVB interface port to the outside world
  #  only on nodes that actually have it:
  useHostPort: true
  ## Make every JVB pod announce its Node's external
  #  IP address and nothing more:
  useNodeIP: true
octo:
  ## Enable OCTO support for both JVB and Jicofo:
  enabled: true
```

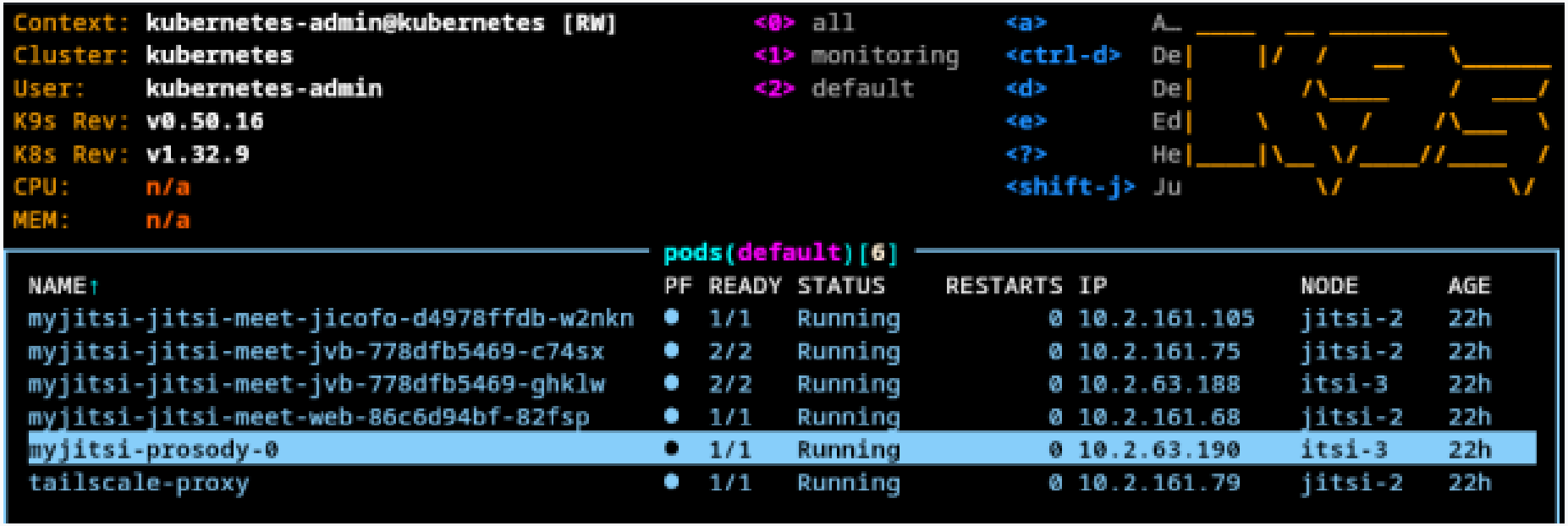
OCTO - co to?

- OCTO to mechanizm multi-bridge routing, który pozwala rozproszyć uczestników jednego spotkania między wiele instancji Jitsi Videobridge (jvb).
- Bez OCTO każde spotkanie musi być obsługiwane przez jeden jvb; gdy osiągnie limit CPU/bitrate - jakość spada.
- Z OCTO spotkanie może używać kilku jvb jednocześnie.

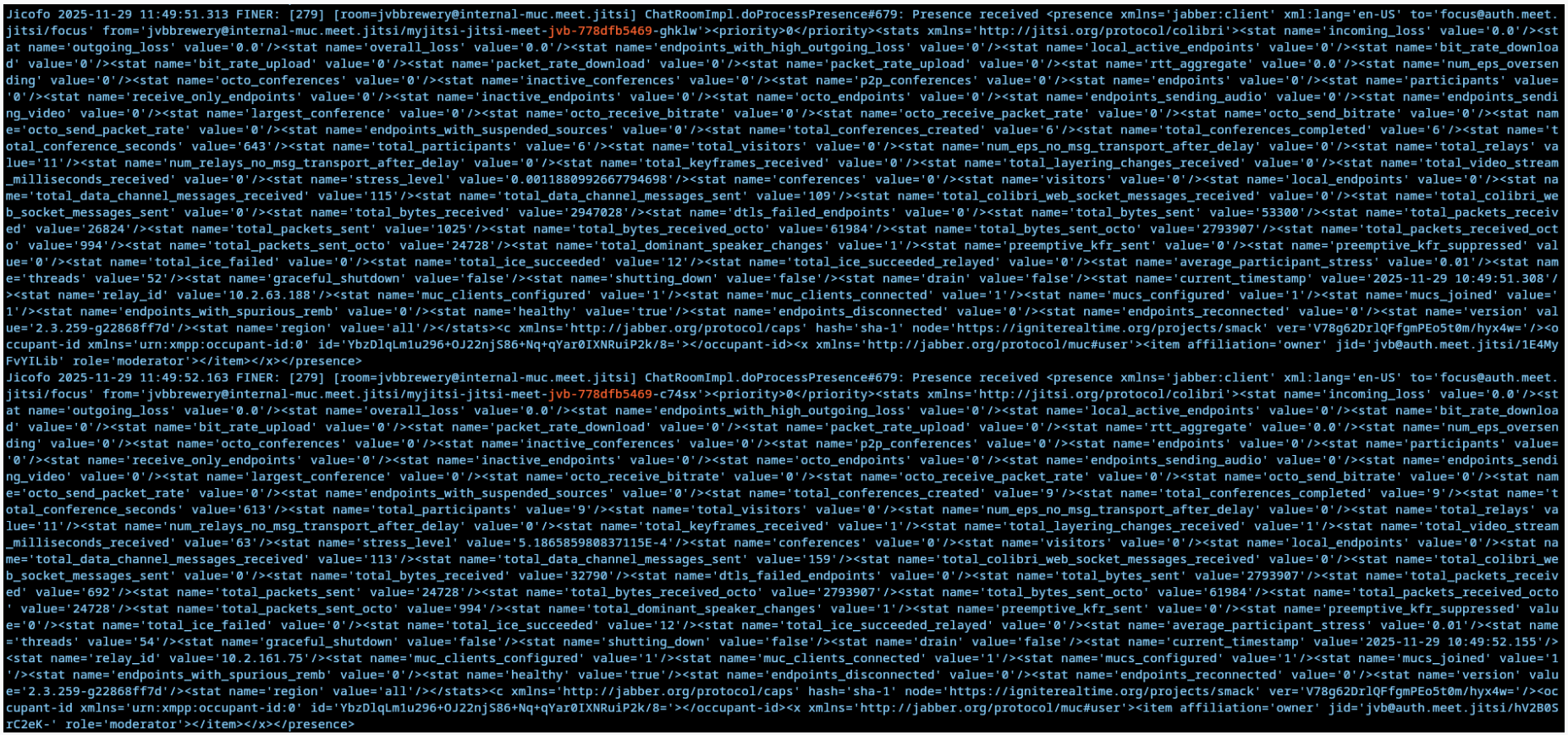
Przed skalowaniem jvb



Po skalowaniu jvb



Jak to wygląda w logach jicofo



Przetestowanie skalowania

Szybkie przypomnienie

- Master Node - maszyna której rolą jest zarządzanie Kubernetes
- Worker Node - maszyna której rolą jest trzymać aplikacje (w podach) użytkownika

Kubernetes

Co się okazuje:

- Master nie obsługuje ruchu użytkowników.
- Master nie uruchamia Podów aplikacyjnych.
- Master nie streamuje danych, nie przetwarza requestów HTTP, nie robi downloadów/uploadów.

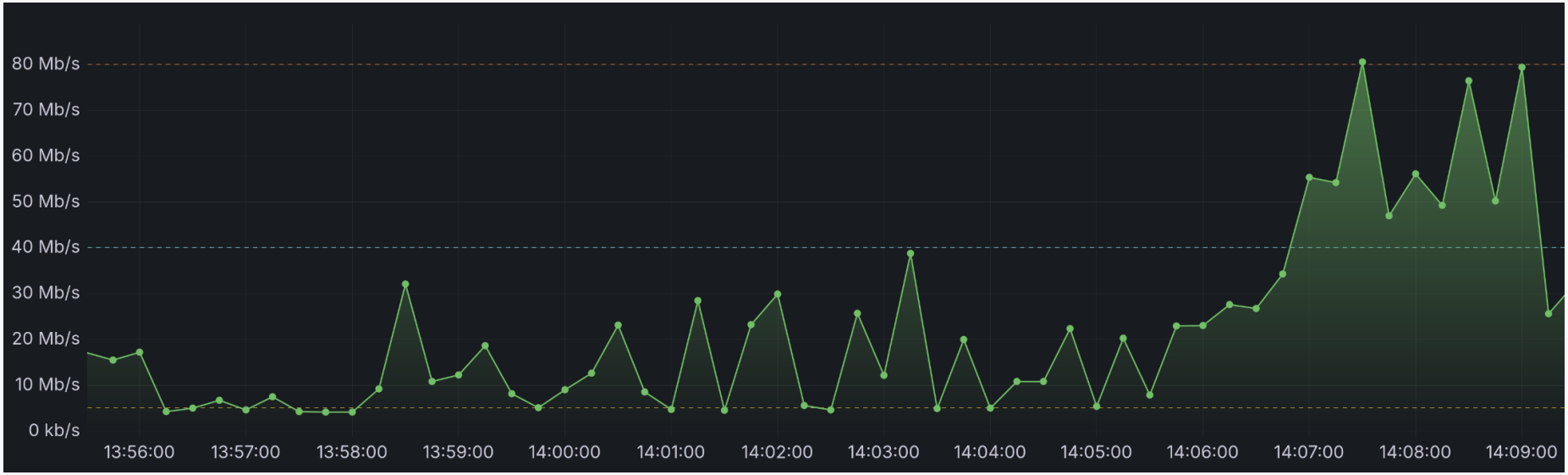
Kubernetes 2

Nawet jeśli master ma wolny internet (np. 1 Mbps), nie wpływa to na:

- prędkość odpowiedzi aplikacji,
- throughput,
- czas obsługi zapytań HTTP/Websocket/GRPC,
- szybkość pobierania danych przez aplikację.

Kubernetes 3

To w trakcie stress testowania jitsi, gdy master ma 128kbit/s:



Czemu nie uzylismy narzedzi zewnetrznych

Webrtcperf dla jitsi jest całkowicie zepsute.

No to teraz jak przetestować jitsi...

Jitsi bardzo dobrze się skaluje. Według oficjalnych pomiarów:

- Dla 1056 strumieni wideo z bitrate 550mbit/s zużycie CPU to tylko 20% przy czterordzeniowym procesorze.
- Dla 1056 strumieni wideo Zużycie RAMu nie przekroczyło 1.5GB

Czemu tak się dzieje?

Jitsi Videobridge jest tylko przekaźnikiem, bez żadnego transkodowania. Nie tworzy skomplikowanych reguł, ani nie weryfikuje nic.

Działa trochę jak router.

Demo

Dziękujemy za uwagę