Netmux

Paulo Simão - 2023

Agenda

- Histórico / Motivação
- Overview
- Benefícios
- Demos
- Principais características arquiteturais
- Detalhes de deployment / operação
- Organização do repositório
- Call for action

Histórico / Motivação

- Um belo dia, precisei depurar uma nova funcionalidade, usando uma lib nova, no cluster.
- Comportamento errático - na minha máquina funcionava - sabe como é...
- Não era tão simples - se tratava de um cluster kafka - 5 nós ouvindo na 9092 - quem nunca?
- Pessoal de segurança não me deixou usar qualquer ferramenta...
- Mas eu sou um DEV, não sou?... e foi aí que tudo começou... (pra dar uma ideia a 1a versão foi implementada usando GRPC).

[Pausa dramática - as câmeras dão ênfase em minhas olheiras, cabelos brancos e aspecto decrépito]

O que é o Netmux?

- Netmux é um conjunto de ferramentas (3) que permitem estabelecer um mesh controlado entre redes isoladas - multiplexando várias conexões dentro de uma só.
- Com ele, você acessa o cluster através de recursos locais, e o cluster acessa você também (permite que serviços no cluster acessem serviços rodando em sua máquina)!
- Ele coleta métricas das conexões e permite que você avalie o tráfego por serviço, por endpoint, etc... (Sim permite métricas e telemetria)

O que é o Netmux?

- Netmux é um conjunto de ferramentas que permitem estabelecer um mesh controlado entre redes isoladas - multiplexando várias conexões dentro de uma só.
- Com ele, você acessa o cluster através de recursos locais, e o cluster acessa você também (permite que serviços no cluster acessem serviços rodando em sua máquina)!
- Ele coleta métricas das conexões e permite que você avalie o tráfego por serviço, por endpoint, etc... Integrado com prometheus.

Benefícios

Em ambiente que permita tal abordagem (PROD não, né velho?), o Netmux promove os seguintes benefícios:

1. Redução do SDLC, os desenvolvedores podem ousar mais durante o desenvolvimento, e usar o depurador (debugger), ao invés de ficar injetando `log.Printf` pra tudo.
2. Tempo reduzido com troubleshooting - você pode rodar o serviço que falha em modo debugger direto em sua máquina e avaliar melhor o problema.
3. Simplificação para implantação de infra - deploy de ferramentas de telemetria, redundância (tipo xds), etc - você consegue testar do cluster direto na sua máquina.
4. Menor consumo de recursos na máquina do dev (não precisamos de 540 containers rodando localmente para testar uma nova funcionalidade).

   De forma prática: o dev ganha tempo, a empresa ganha em menor TCO e menor time -to-market. O custo é diminuído pelo tempo do dev, e pelo custo de execução da esteira (menos interações).

Demo

Métricas

Algumas métricas de nosso repositório

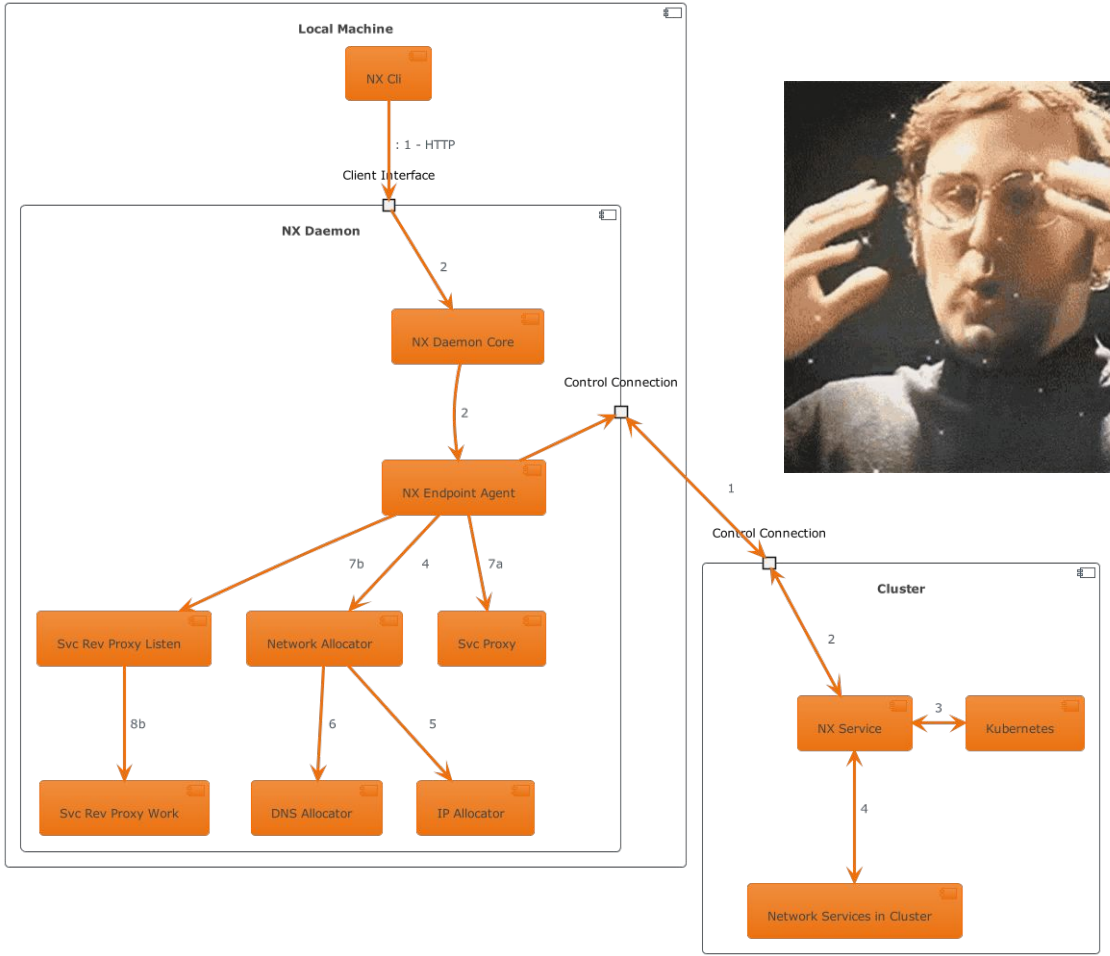
Language	Files	Lines	Blanks	Comments	Code	Complexity
Go	39	5363	1128	140	4095	834
YAML	24	785	13	29	743	0
Markdown	3	467	60	0	407	0
Dockerfile	2	15	4	0	11	0
JSON	1	789	0	0	789	0
License	1	201	32	0	169	0
Makefile	1	48	14	0	34	0
gitignore	1	26	3	9	14	0
Total	72	7694	1254	178	6262	834

Estimated Cost to Develop (organic) $185,414
Estimated Schedule Effort (organic) 7.25 months
Estimated People Required (organic) 2.27

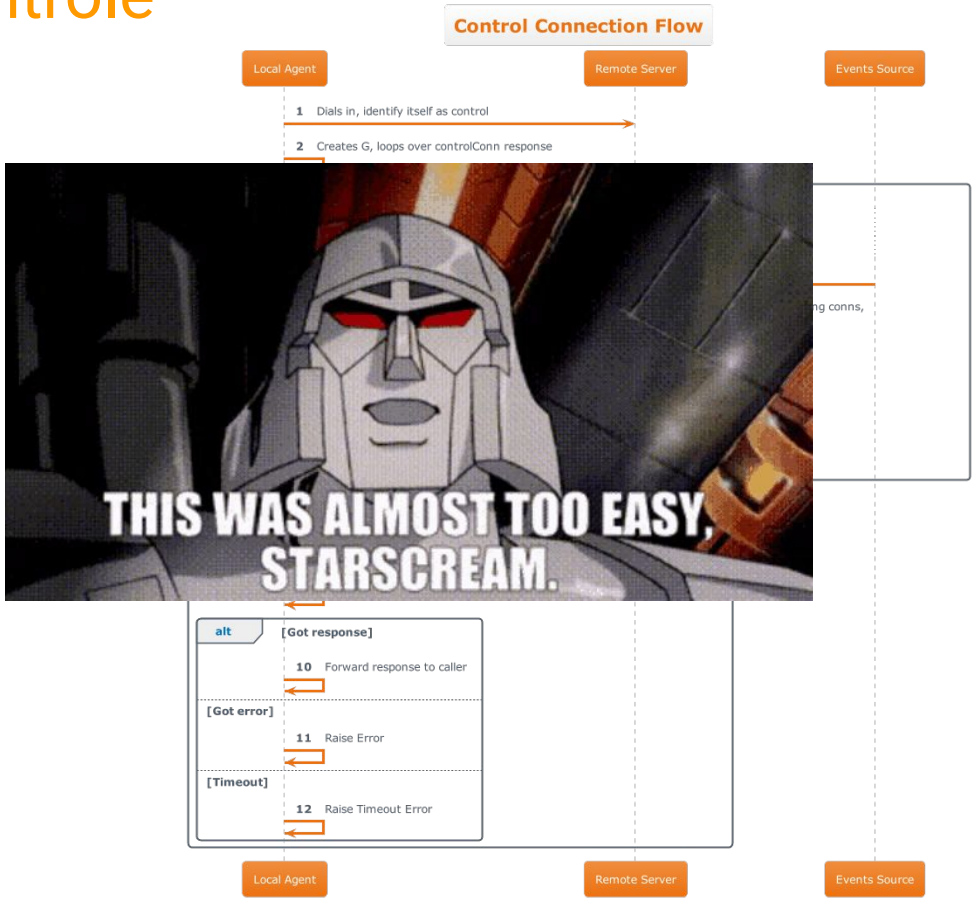
Processed 183653 bytes, 0.184 megabytes (SI)

Fonte: scc

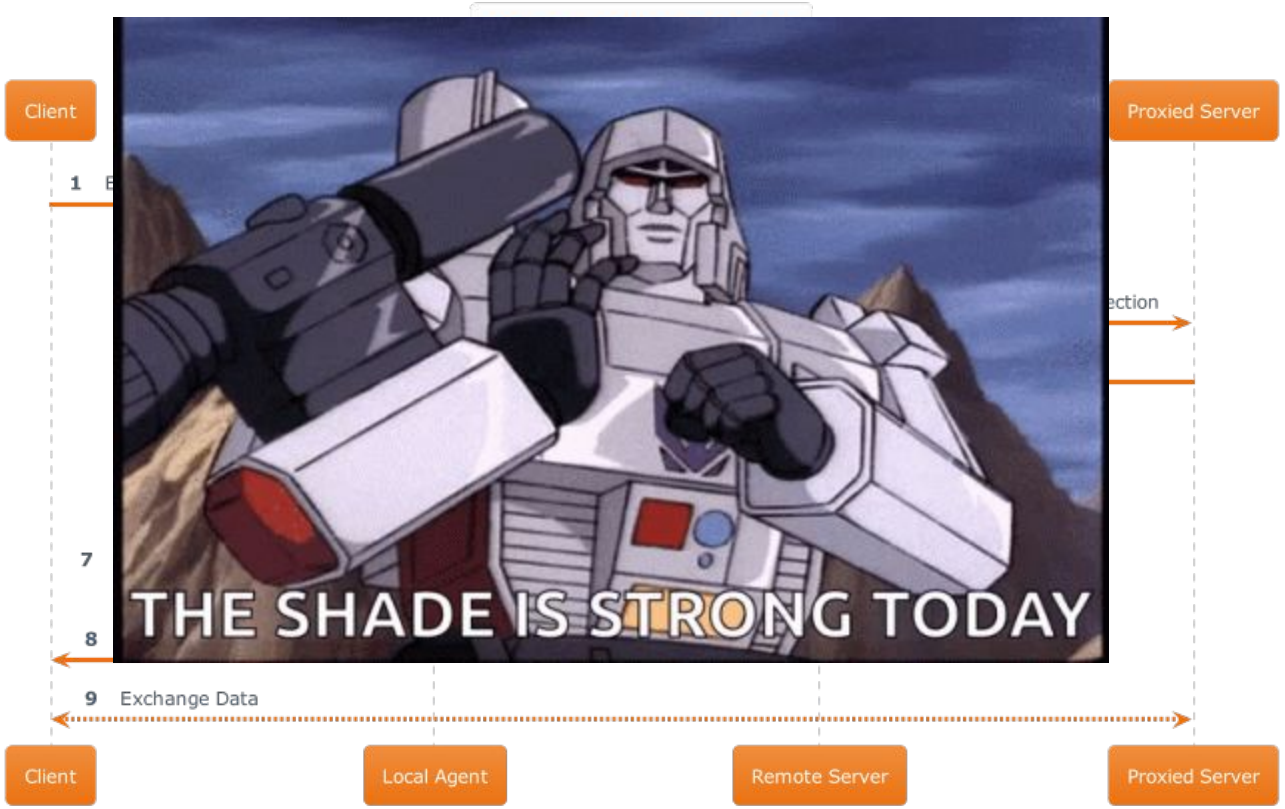
Arquitetura



Conexão de Controle

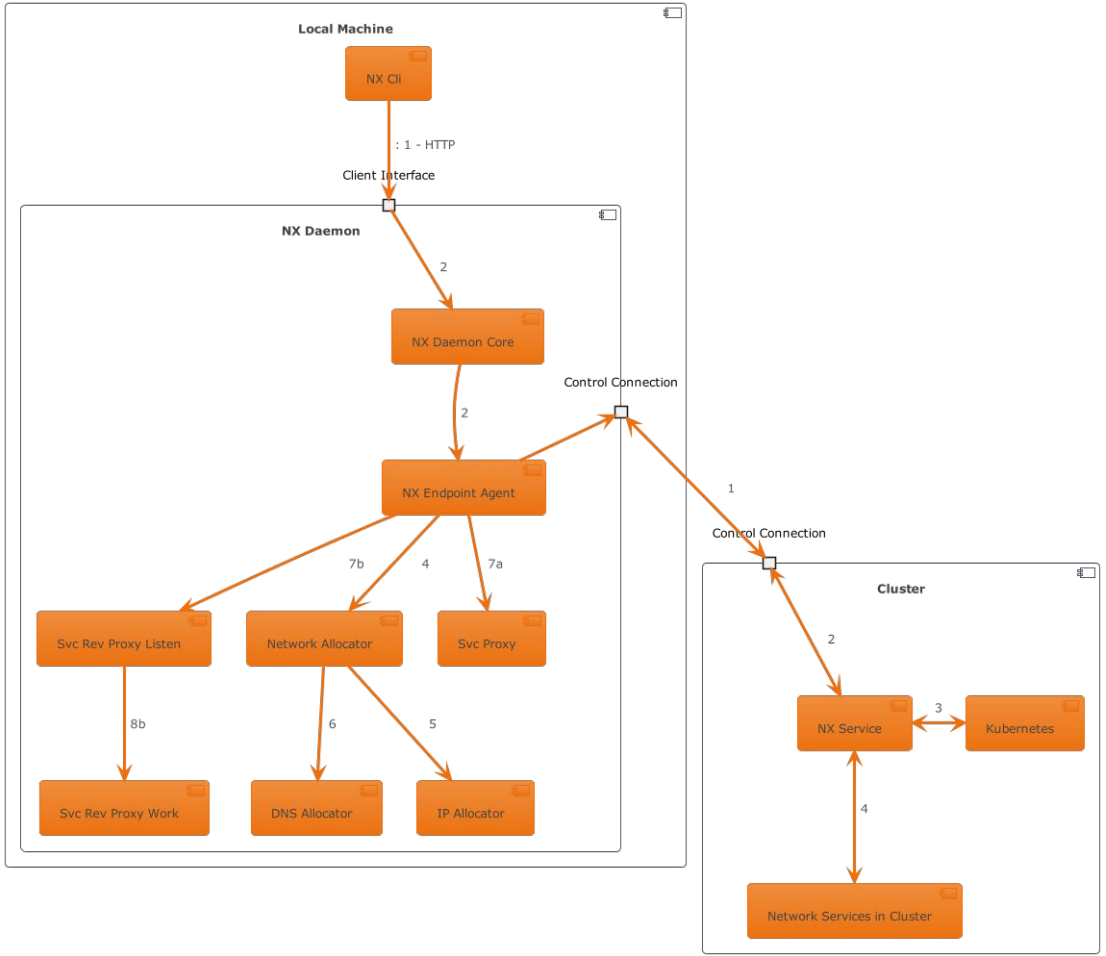


Proxy Direto

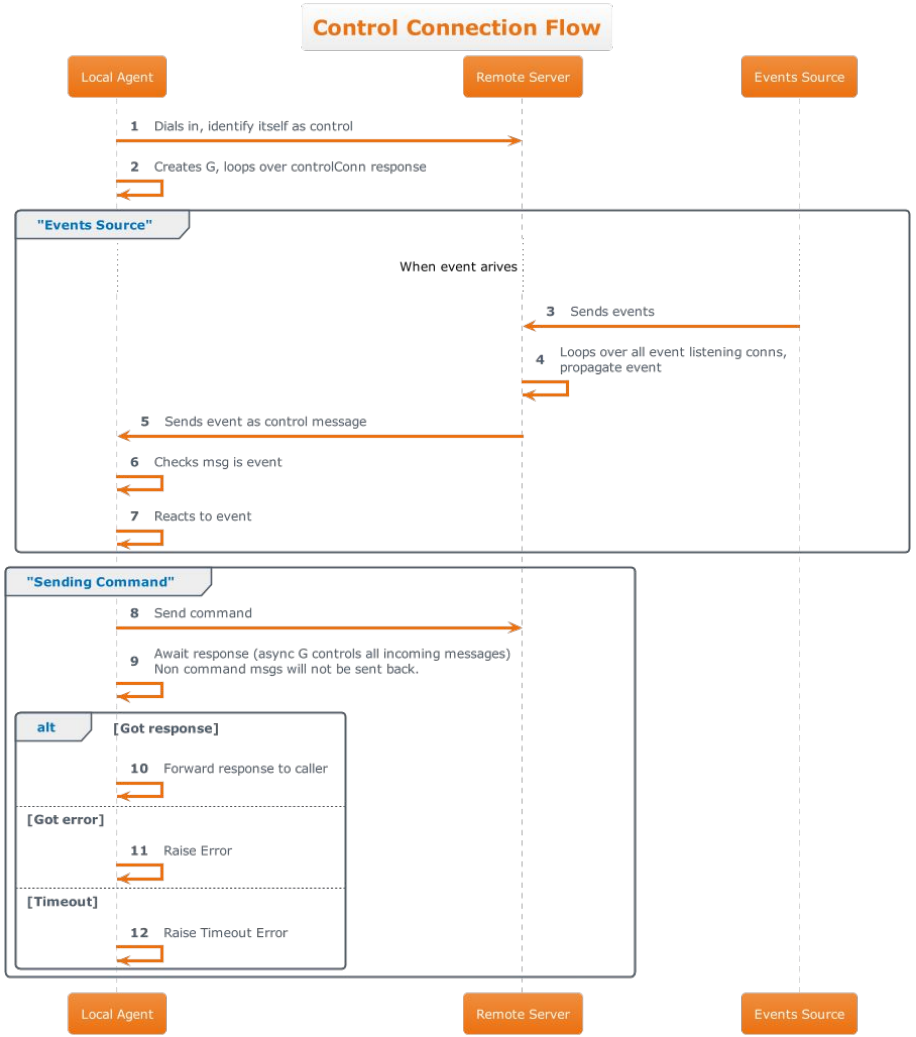


Proxy Reverso

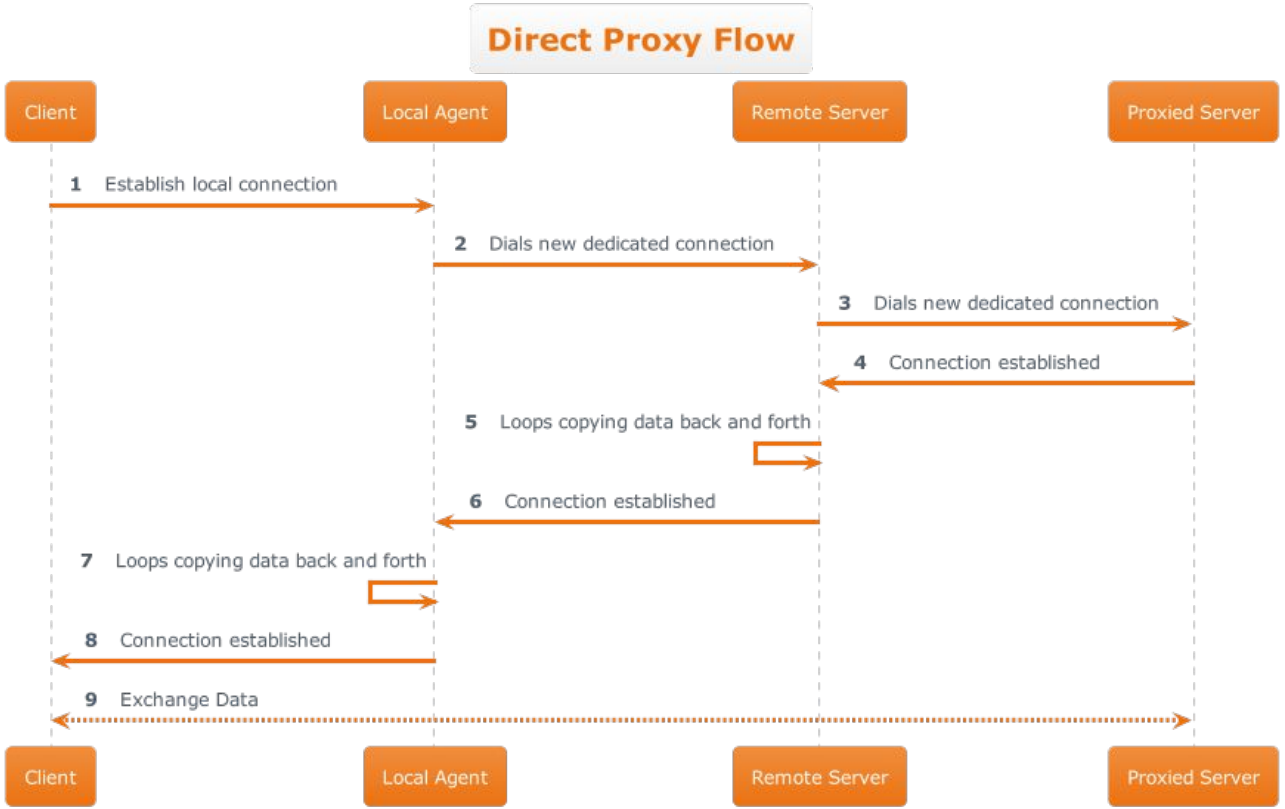
Arquitetura



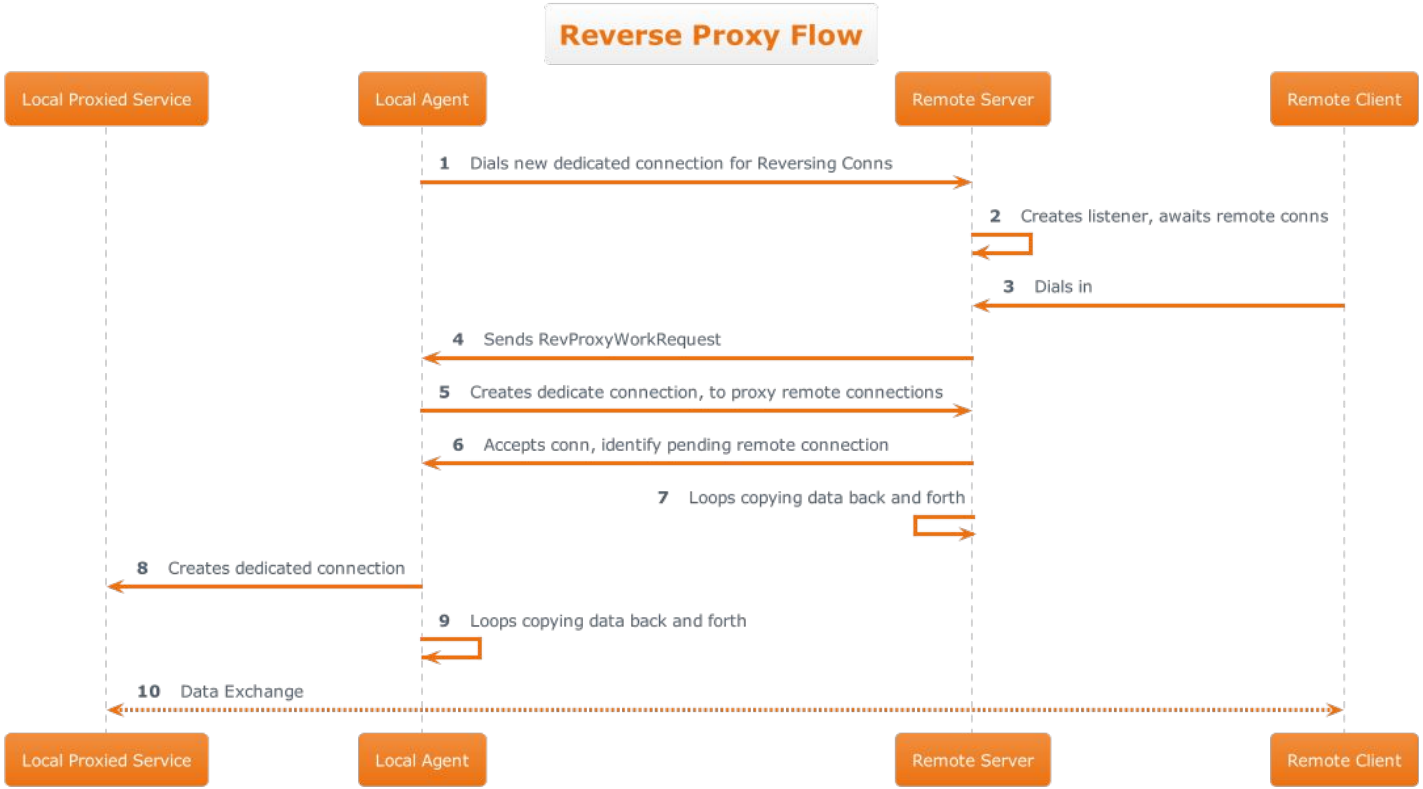
Conexão de Controle



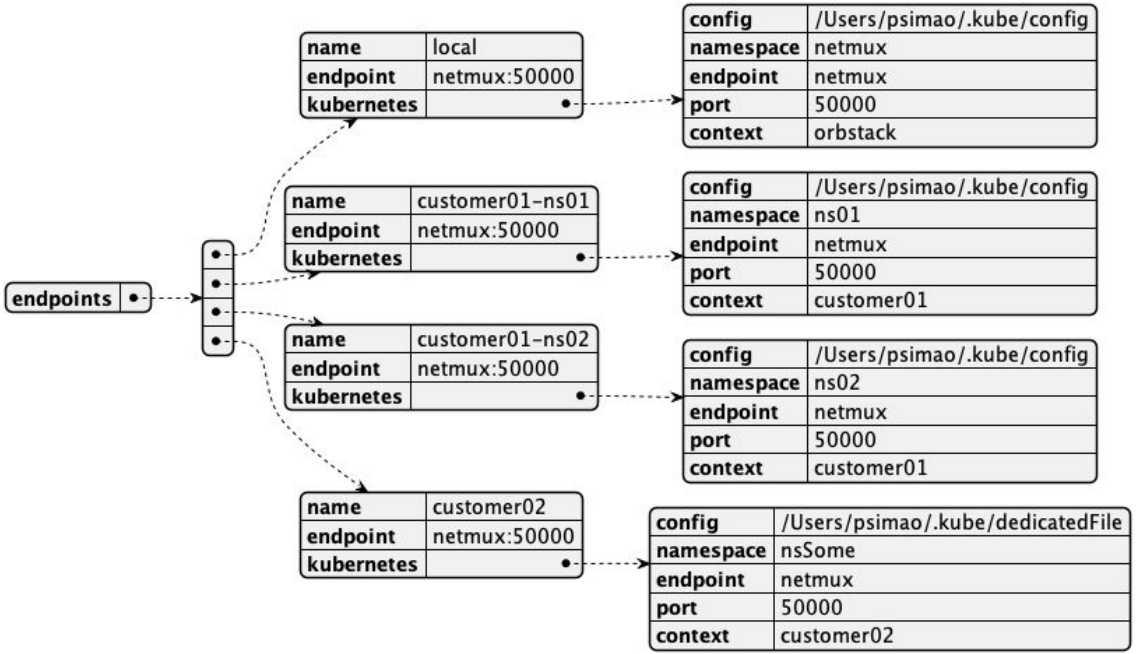
Proxy Direto



Proxy Reverso



Configuração Local



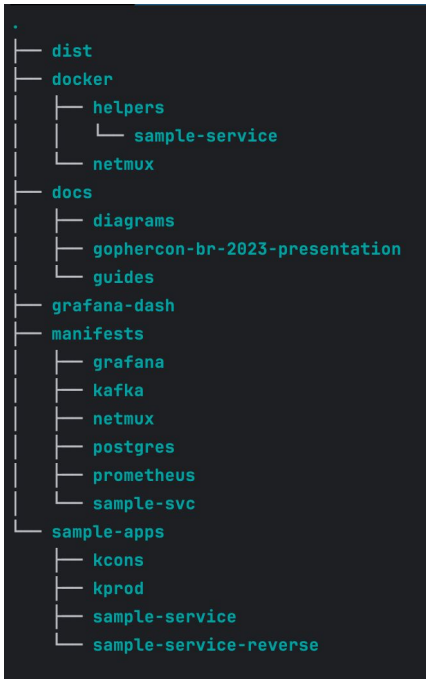
Organização do Repo

- **app**: Executáveis (injeção de dependências e *main*)
  - **nx-cli**: linha de comando
  - **nx-daemon**: serviço local
  - **nx-server**: serviço remoto
- **business**: Componentes com Regras de Negócio
  - **caroot:**gestão de certificados
  - **netmux**: core das regras de negócio
  - **networkallocator**: gestão de rede, aloca ips e nomes
  - **portforwarder**: manipulação de port-forward para k8s
  - **shell**: comandos para SO
- **foundation**: Componentes de Base
  - **buildinfo**: gestão de versão
  - **memstore**: cache genérico thread-safe (usado em vários lugares)
  - **metrics**: abstração para coleta de métricas
  - **pipe**: cópia de dados gerenciada full duplex
  - **wire**: Protocolo de baixo nível

Organização do Repo

- **app**: Executáveis (injeção de dependências e *main*)
  - **nx-cli**: linha de comando
  - **nx-daemon**: serviço local
  - **nx-server**: serviço remoto
- **business**: Componentes com Regras de Negócio
  - **caroot:**gestão de certificados
  - **netmux**: core das regras de negócio
  - **networkallocator**: gestão de rede, aloca ips e nomes
  - **portforwarder**: manipulação de port-forward para k8s
  - **shell**: comandos para SO
- **foundation**: Componentes de Base
  - **buildinfo**: gestão de versão
  - **memstore**: cache genérico thread-safe (usado em vários lugares)
  - **metrics**: abstração para coleta de métricas
  - **pipe**: cópia de dados gerenciada full duplex
  - **wire**: Protocolo de baixo nível

Organização do Repo II - Zarf



- **`dist`**: Onde salvamos binários - .gitignore
- **`docker`**: Dockerfile
- **`docs`**: Documentação em geral
- **`grafana-dash`**: Exemplo de dash grafana para acompanhar o uso do netmux
- **`manifests`**: múltiplos arquivos manifest para fazer o deploy do netmux e demais exemplos. Use **`kustomize`**!
- **`sample-apps`**: Aplicativos complementares
  - **`kcons`**: consumidor kafka
  - **`kprod`**: produtor kafka
  - **`sample-service`**: serviço exemplo
  - **`sample-service-reverse`**: serviço exemplo para conexão reversa

Deploy no Cluster

- Role
- Service Account
- Role Binding
- Service
- Deployment

Annotations

Para maior flexibilidade, você pode configurar a última milha de seu serviço.

No exemplo ao lado estamos publicando o cluster kafka inteiro (vide demo kafka)

```yaml
apiVersion: v1
kind: Service
metadata:
  name: kafka
  annotations:
    nx: |-
      - name: kafka
      - name: kafka-0
        localAddr: kafka-0.kafka.netmux.svc.cluster.local
        containerAddr: kafka-0.kafka.netmux.svc.cluster.local
      - name: kafka-1
        localAddr: kafka-1.kafka.netmux.svc.cluster.local
        containerAddr: kafka-1.kafka.netmux.svc.cluster.local
      - name: kafka-2
        localAddr: kafka-2.kafka.netmux.svc.cluster.local
        containerAddr: kafka-2.kafka.netmux.svc.cluster.local
  labels:
    app: kafka-app
spec:
  ports:
    - name: '9092'
      port: 9092
      protocol: TCP
      targetPort: 9092
  selector:
    app: kafka-app
```

Call for Action - Hey oh, lets go!

- Instaladores: `curl https://duxthemux.io/netmux/install.sh | sh`
- Portar para uso com docker e bare metal (lembra do runtime?)
- Ajustar o uso para outros mecanismos de autenticação k8s (específico dos cloud providers)
- Adicionar magic numbers ao protocolo e melhorar a tolerância a erros.
- Adaptar uso de interfaces TUN
- Trocar abordagem do arquivo **`hosts`** por um servidor dns local, embarcado
- Empregar TLS direto - para conexões diretas sem port forwarding
- Testes poderiam ter cobertura melhor
- Autenticação e Autorização
- Regras mais refinadas para alocação de endereços / pré-alocação e roteamento
- O que mais o coração mandar

Contatos / Repositório

Paulo (P.O.): paulo@digitalcircle.com.br

Github:

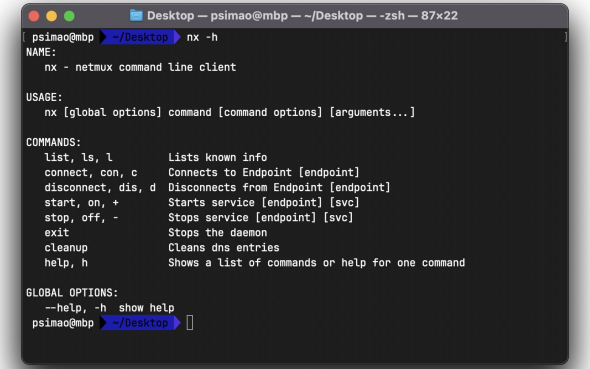
https://github.com/duxthemux/netmux

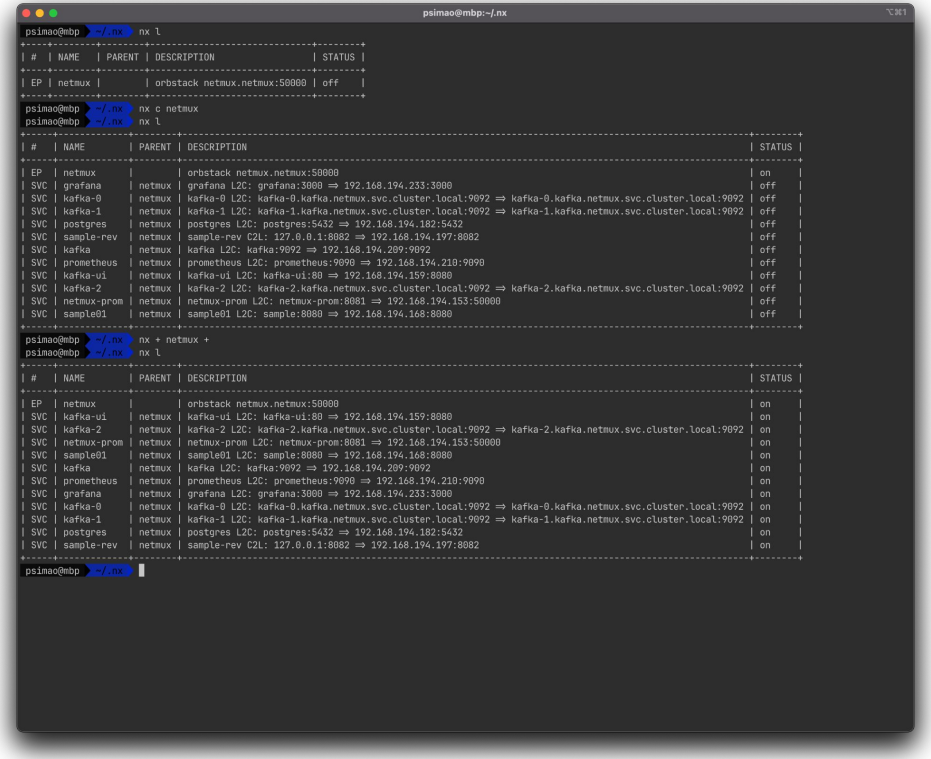
Dockerhub:

https://hub.docker.com/repository/docker/duxthemux/netmux/general

Backup Slides

CLI


Desktop — psimao@mbp — ~/Desktop — -zsh — 87×22
psimao@mbp ~/Desktop nx -h
NAME:
nx - netmux command line client
USAGE:
nx [global options] command [command options] [arguments...]
COMMANDS:
list, ls, l Lists known info
connect, con, c Connects to Endpoint [endpoint]
disconnect, dis, d Disconnects from Endpoint [endpoint]
start, on, + Starts service [endpoint] [svc]
stop, off, - Stops service [endpoint] [svc]
exit Stops the daemon
cleanup Cleans dns entries
help, h Shows a list of commands or help for one command
GLOBAL OPTIONS:
--help, -h show help
psimao@mbp ~/Desktop


psimao@mbp:~/.nx
psimao@mbp ~/.nx nx l
| # | NAME | PARENT | DESCRIPTION | STATUS |
| EP | netmux | | orbstack netmux.netmux:50000 | off |
psimao@mbp ~/.nx nx c netmux
psimao@mbp ~/.nx nx l
#	NAME	PARENT	DESCRIPTION	STATUS
EP	netmux		orbstack netmux.netmux:50000	on
SVC	grafana	netmux	grafana L2C: grafana:3000 ⇒ 192.168.194.233:3000	off
SVC	kafka-0	netmux	kafka-0 L2C: kafka-0.kafka.netmux.svc.cluster.local:9092 ⇒ kafka-0.kafka.netmux.svc.cluster.local:9092	off
SVC	kafka-1	netmux	kafka-1 L2C: kafka-1.kafka.netmux.svc.cluster.local:9092 ⇒ kafka-1.kafka.netmux.svc.cluster.local:9092	off
SVC	postgres	netmux	postgres L2C: postgres:5432 ⇒ 192.168.194.182:5432	off
SVC	sample-rev	netmux	sample-rev C2L: 127.0.0.1:8082 ⇒ 192.168.194.197:8082	off
SVC	kafka	netmux	kafka L2C: kafka:9092 ⇒ 192.168.194.209:9092	off
SVC	prometheus	netmux	prometheus L2C: prometheus:9090 ⇒ 192.168.194.210:9090	off
SVC	kafka-ui	netmux	kafka-ui L2C: kafka-ui:80 ⇒ 192.168.194.159:8080	off
SVC	kafka-2	netmux	kafka-2 L2C: kafka-2.kafka.netmux.svc.cluster.local:9092 ⇒ kafka-2.kafka.netmux.svc.cluster.local:9092	off
SVC	netmux-prom	netmux	netmux-prom L2C: netmux-prom:8081 ⇒ 192.168.194.153:50000	off
SVC	sample01	netmux	sample01 L2C: sample:8080 ⇒ 192.168.194.168:8080	off
psimao@mbp ~/.nx nx + netmux +
psimao@mbp ~/.nx nx l
#	NAME	PARENT	DESCRIPTION	STATUS
EP	netmux		orbstack netmux.netmux:50000	on
SVC	kafka-ui	netmux	kafka-ui L2C: kafka-ui:80 ⇒ 192.168.194.159:8080	on
SVC	kafka-2	netmux	kafka-2 L2C: kafka-2.kafka.netmux.svc.cluster.local:9092 ⇒ kafka-2.kafka.netmux.svc.cluster.local:9092	on
SVC	netmux-prom	netmux	netmux-prom L2C: netmux-prom:8081 ⇒ 192.168.194.153:50000	on
SVC	sample01	netmux	sample01 L2C: sample:8080 ⇒ 192.168.194.168:8080	on
SVC	kafka	netmux	kafka L2C: kafka:9092 ⇒ 192.168.194.209:9092	on
SVC	prometheus	netmux	prometheus L2C: prometheus:9090 ⇒ 192.168.194.210:9090	on
SVC	grafana	netmux	grafana L2C: grafana:3000 ⇒ 192.168.194.233:3000	on
SVC	kafka-0	netmux	kafka-0 L2C: kafka-0.kafka.netmux.svc.cluster.local:9092 ⇒ kafka-0.kafka.netmux.svc.cluster.local:9092	on
SVC	kafka-1	netmux	kafka-1 L2C: kafka-1.kafka.netmux.svc.cluster.local:9092 ⇒ kafka-1.kafka.netmux.svc.cluster.local:9092	on
SVC	postgres	netmux	postgres L2C: postgres:5432 ⇒ 192.168.194.182:5432	on
SVC	sample-rev	netmux	sample-rev C2L: 127.0.0.1:8082 ⇒ 192.168.194.197:8082	on
psimao@mbp ~/.nx

Curiosidade - TAP & TUN

- TAP
  - Simula dispositivo de rede L2 (Ethernet packet)
  - MacOS não suporta tem um tempo
  - Drivers no Windows
- TUN
  - Simula dispositivo de rede L3 (IP packet)
  - Funciona em todos OSs
- Código exemplo - Não use, adaptei da fonte só como exemplo!!!

```go
func main() {
    file, err := os.OpenFile( name: "/dev/net/tun", os.O_RDWR, perm: 0)
    if err != nil {
        panic(fmt.Errorf( format: "error os.Open(): %v\n", err))
    }

    ifr := make([]byte, 18)

    copy(ifr, []byte("tun0"))

    ifr[16], ifr[17] = 0x01, 0x10

    _, _, errno := syscall.Syscall(
        syscall.SYS_IOCTL, uintptr(file.Fd()),
        uintptr(0x400454ca), uintptr(unsafe.Pointer(&ifr[0])))
    if errno != 0 {
        panic(fmt.Errorf( format: "error syscall.Ioctl(): %v\n", errno))
    }

    cmd := exec.Command( name: "/sbin/ifconfig", arg...: "tun0", "192.168.7.1", "pointopoint", "192.168.7.2", "up")

    if err := cmd.Start(); err != nil {
        panic(fmt.Errorf( format: "error running command: %v\n", err))
    }

    //------NEXT SLIDE---///
```

SRC: https://gist.githubusercontent.com/glacjay/585620/raw/3e685b9e9b035c360afc08621a7802e16bc7add4/ping-linux.go

Curiosidade - TAP & TUN

```
    //------NEXT SLIDE---///

    for {
        buf := make([]byte, 2048)
        read, err := file.Read(buf)
        if err != nil {
            panic(fmt.Errorf( format: "error os.Read(): %v\n", err))
        }

        for i := 0; i < 4; i++ {
            buf[i+12], buf[i+16] = buf[i+16], buf[i+12]
        }
        buf[20], buf[22], buf[23] = 0, 0, 0

        var checksum uint16
        for i := 20; i < read; i += 2 {
            checksum += uint16(buf[i])<<8 + uint16(buf[i+1])
        }

        checksum = ^(checksum + 4)

        buf[22], buf[23] = byte(checksum>>8), byte(checksum&((1<<8)-1))

        _, err = file.Write(buf)
        if err != nil {
            panic(fmt.Errorf( format: "error os.Write(): %v\n", err))
        }
    }
}
```

Referências interessantes

- Wireguard Go: https://github.com/WireGuard/wireguard-go
- GVisor: https://github.com/google/gvisor
- K8S: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/
- Docker: https://docs.docker.com/engine/api/v1.43/#tag/System/operation/SystemEvents