

# Développer et déployer une API HTTP/3 en Go

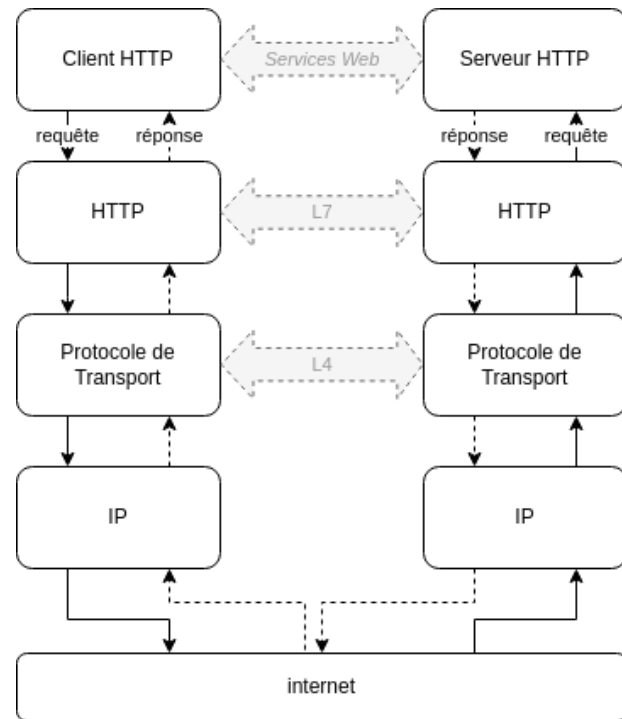
**Capitole du Libre**

16-17 novembre 2024

Thierry Beigbeder

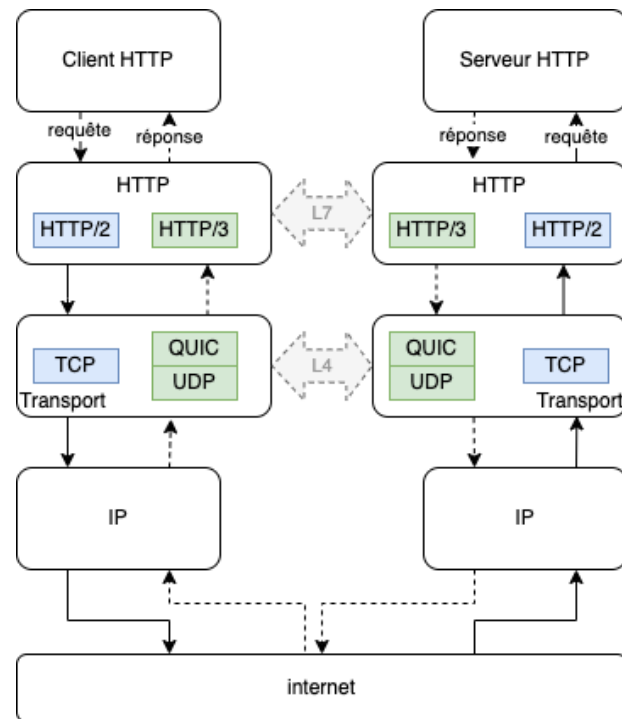
# HTTP/3 : continuité...

- HTTP = interagir avec des ressources Web
  - URLs
  - Méthodes: GET, POST...
- Transport = échange de données
  - Connexion durable (...)
  - Fiable (...)
- Réseau (IP) = envoi à un destinataire
  - Adressage, routage
  - Transmission de paquets
- Internet = interconnexion de réseaux



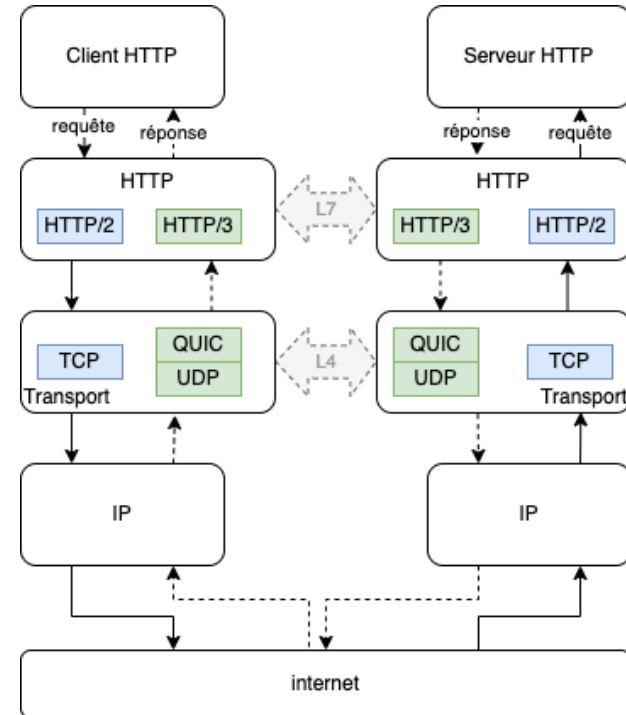
# HTTP/3 : ...ou rupture ?

- TCP vs UDP
- TCP
  - Connexion durable
  - Échange données fiable, ordonné
- UDP
  - Sans connexion
  - Sans garantie de transmission, d'ordre
- QUIC
  - Surcouche apporte connexion et fiabilité
  - HTTP/3 utilise QUIC



# HTTP/3 : confidentiel?

- QUIC
  - Développé chez Google en 2012
  - IETF draft 2015 (géants internet)
- HTTP/3
  - Chrome 2019
  - FF, Chromium 2021
  - Serveurs, passerelles depuis 2017
- [...]
- Nginx : experimental
- Apache → Litespeed « Open Source »



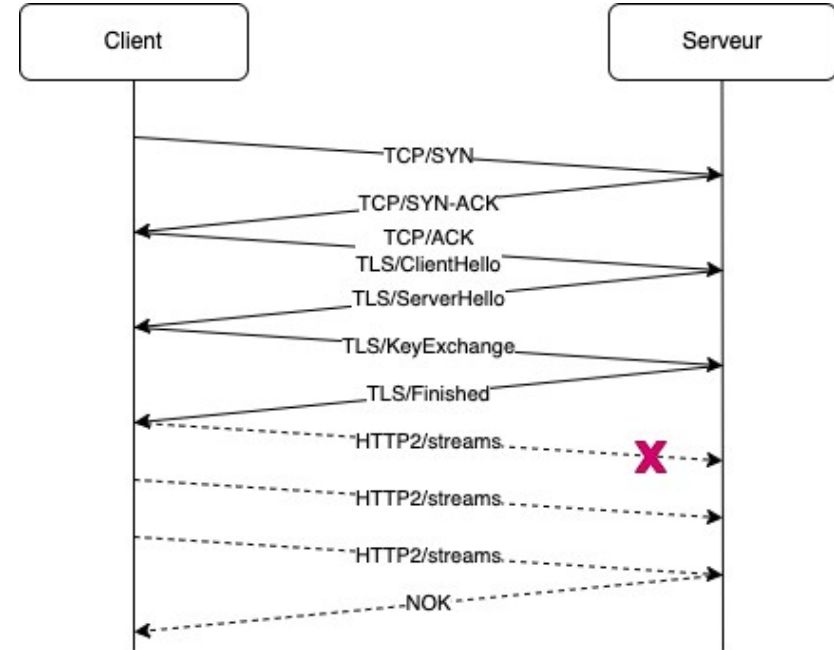
# La saga TCP/IP, TLS...

- [TCP conçu en 1974, standardisé en 1980]
- TCP handshake (3 way)
- TLS handshake (2 RTT)
  - Reprise de session
- TCP « head of line blocking »
- Erreurs, retransmissions + latence = cumul important
- TLS1.3 réduit les RTT



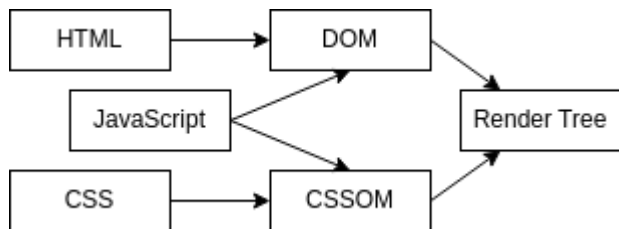
# La saga TCP/IP, TLS... et HTTP

- HTTP/1.1: fin des années 90
  - TCP keepalive
  - Paquetage des ressources HTML
- HTTP/2: 2015
  - Multiplexage et priorisation de flux
  - Push depuis le serveur
  - Compression en-têtes
- Relativiser
  - TCP HOL blocking perdure
  - Parallélisme vs logique des ressources page Web
  - Multiplexage/priorisation peu implémentés



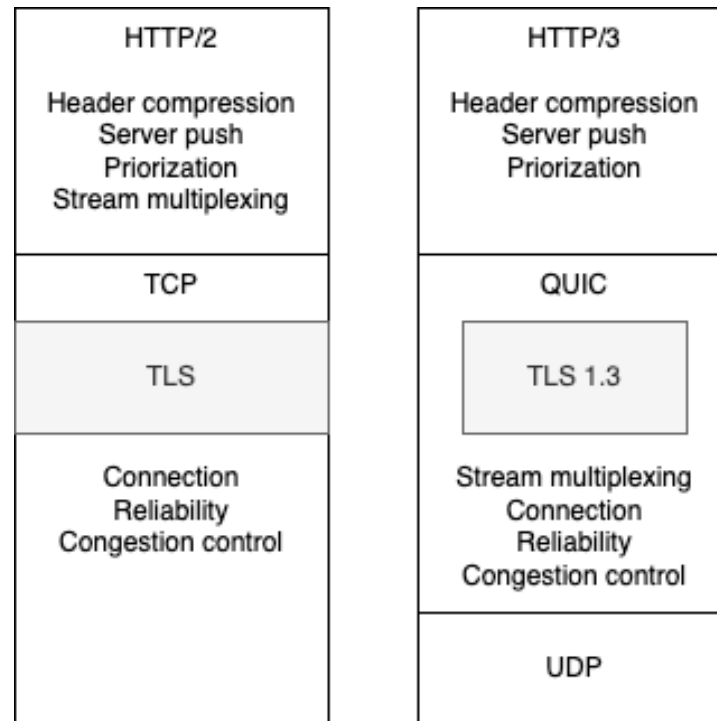
# Page Web et réseau

- Navigateur parallélise les téléchargements
    - HTTP/1.1 : souvent packages CSS/JS
    - HTTP/2 : préférable petits fichiers indépendants
  - JavaScript
    - Enrichit le DOM
    - Consulte le CSSDOM
- => comment paralléliser/prioriser ?



# Le transport QUIC

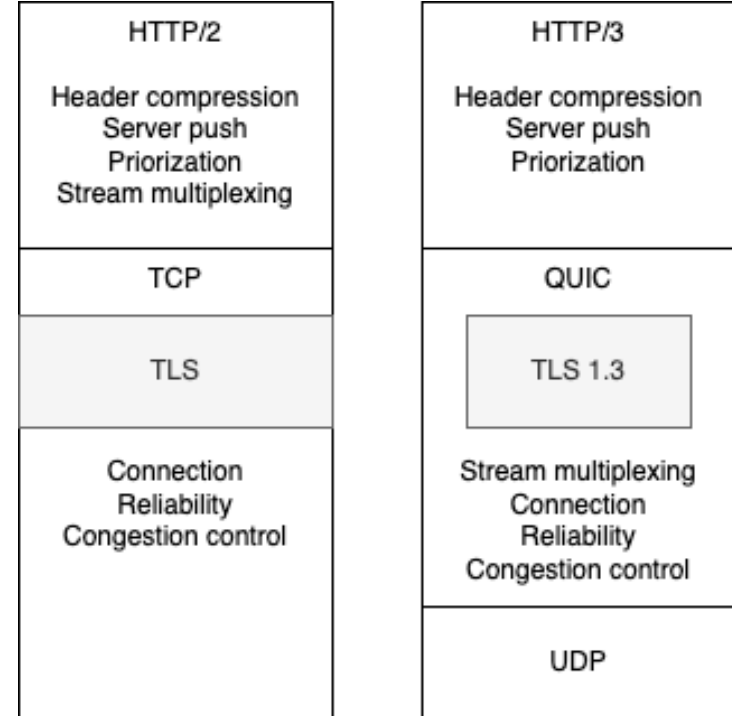
- UDP
  - Emission/réception de datagrammes de bout en bout
  - Ni fiable, ni ordonné
  - Très proche du réseau, très performant
- QUIC
  - Connexion, fiabilité, ordonnancement
    - Evaluation bande passante et adaptation
  - Contrôle de congestion
  - Intègre le chiffrement complet : en-têtes + charge utile
  - Intègre TLS de façon optimisée
  - Intègre le multiplexage des flux





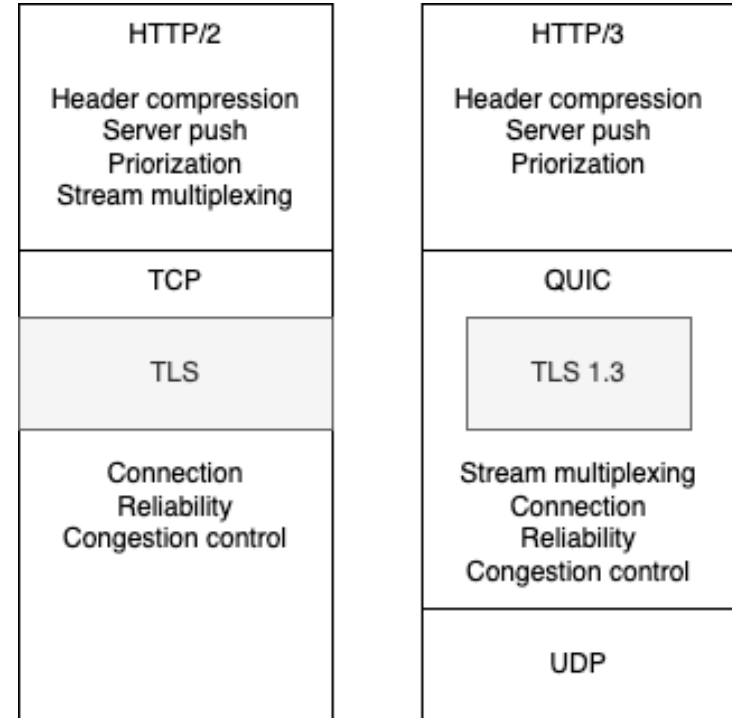
# QUIC et HTTP/3

- QUIC
  - Transport fiable, ordonné, chiffré
  - Intègre le multiplexage des flux
    - Datagramme  $\sim$  paquet = N trames
    - Un flux par trame
  - Migration de connexion
  - Réduit taille données
- HTTP/3
  - Reprend les fonctions restantes de HTTP/2
  - Priorisation s'appuie sur le multiplexage



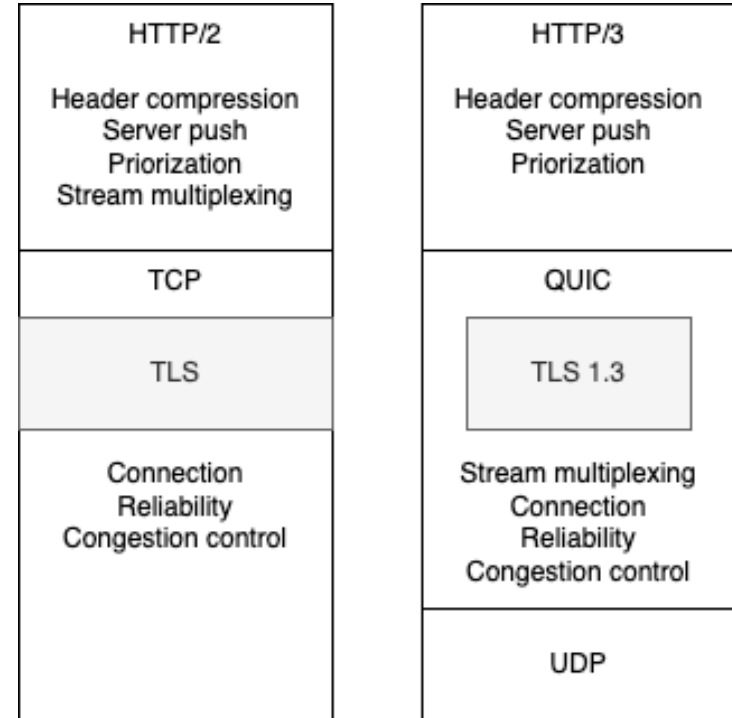
# HTTP/3: encore des promesses?

- HOL blocking
  - Adressé avec plus de réactivité
  - Perte affecte le(s) flux du seul paquet
  - Retransmission ponctuelle
- Handshake transport + TLS optimisé (1 RTT)
- Gains sensibles sur réseaux de faible qualité
- Relativiser
  - Assumer existant incompatible
  - Utilisation priorisation applications
  - Implémentations variées



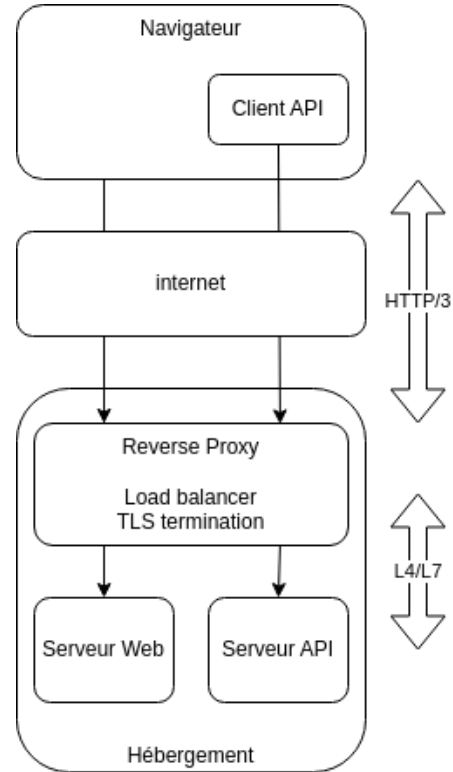
# HTTP/3: aspects plus techniques

- TCP dans le kernel / QUIC dans des bibliothèques
  - Cycles de développement/déploiement courts
    - Contrôle de congestion
    - Expérimentations
  - Sécurité plus délicate
- Consommation de ressources plus importante
  - Chiffrement
  - Allocation mémoire
- Chiffrement vs inspection des flux



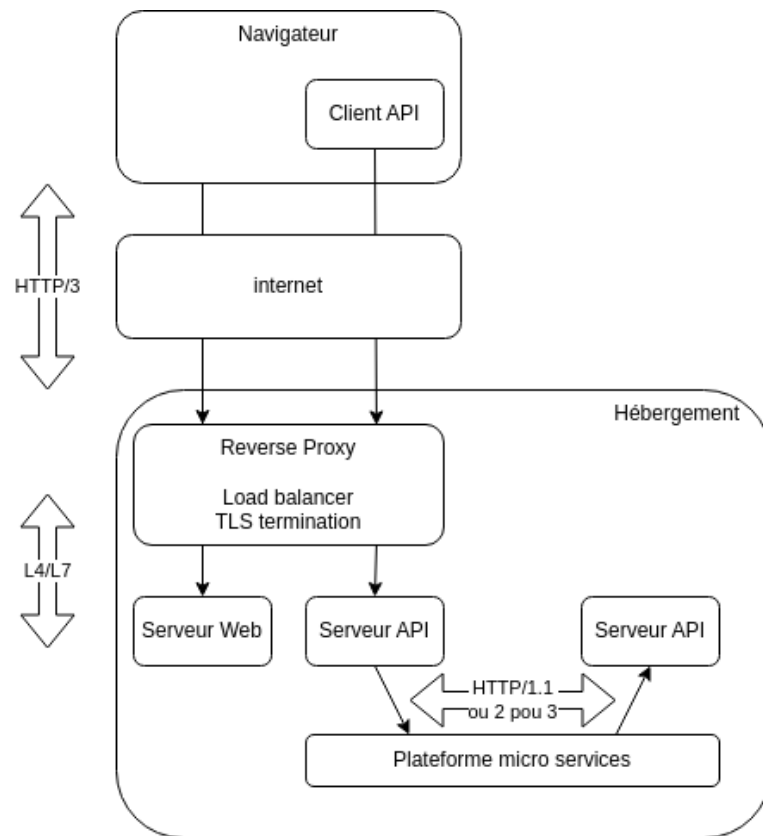
# API depuis le navigateur

- Réutilisation de connexions QUIC
- Rupture à l'arrivée [...]
- Peu de gains supplémentaires
  - APIs très réactives




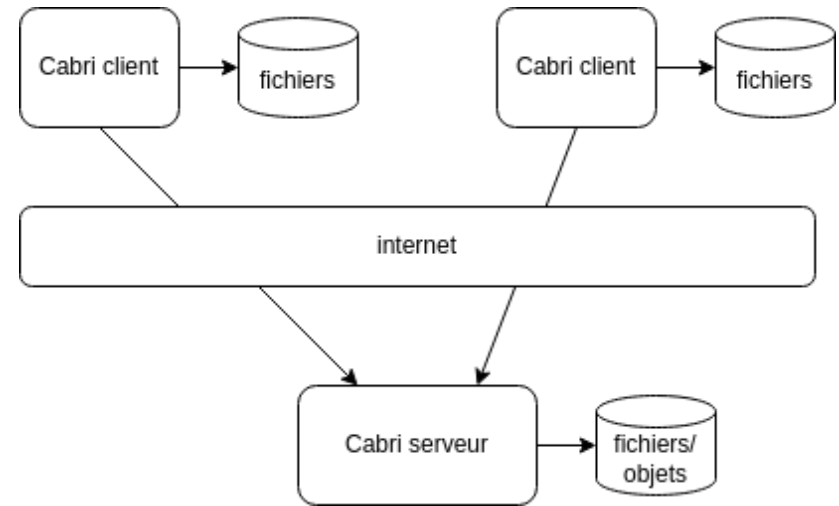
# APIs « backend »

- Latence minimale et qualité de réseau maximale
- Pas de gain sensible de performances
  - sauf APIs très réactives
- Gain de sécurité
- Encore peu de solutions [...]



# Exemple cabri

- Stockage et synchronisation de données 
- API HTTP
- Réconciliation de données => mélange:
  - Requêtes rapides: état d'une entrée
  - Requêtes lentes: calcul de checksum
  - Requêtes volumineuses : upload/download de données
- Parallélisme intensif
- Erreurs sporadiques sur certains hébergements
- Gains possibles?
  - Meilleur parallélisme
  - Meilleure réaction aux erreurs



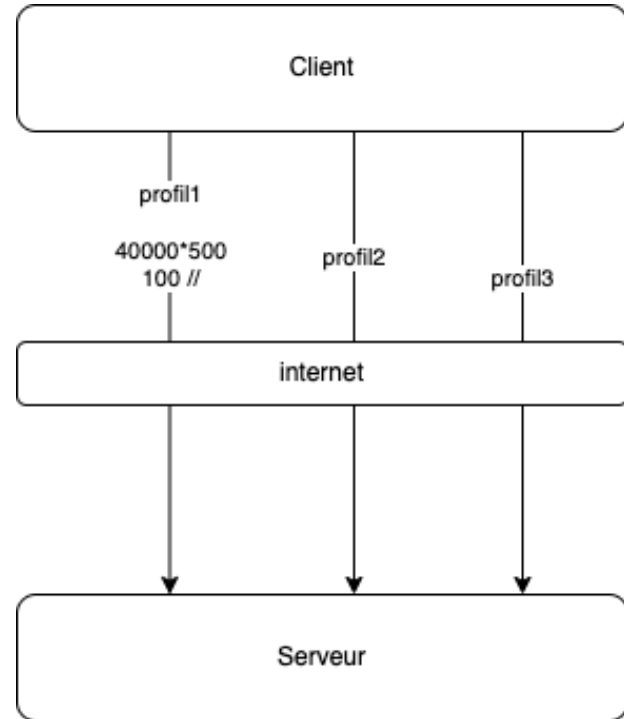
# Bibliothèque quic-go

- quic-go : implementation of the QUIC protocol in Go
  - 10k stars on github
  - support for HTTP/3
- En complément
  - Unreliable Datagram Extension
  - QUIC Version 2 (RFC 9369)
  - QUIC Event Logging using qlog
- webtransport-go : support for WebTransport



# Prototype simulateur de requêtes

- Bibliothèque quic-go pour HTTP/3
- Client et serveur HTTP/1.1 ou HTTP/3
- Jeu parallèle de N profils de charge
  - Nombre de requêtes
  - Taille des requêtes et des réponses
  - Nombre maximal de requêtes en cours
- Go: séparation claire http / transport
  - Impact HTTP/3 minimum pour une utilisation compatible HTTP/1.1





# Prototype client

```
import (  
    "net/http"  
    "github.com/quic-go/quic-go"  
    "github.com/quic-go/quic-go/http3"  
)
```

```
if !config.IsHttp2 {  
    roundTripper := &http3.RoundTripper{  
        TLSClientConfig: tc,  
        QUICConfig:      &quic.Config{},  
    }  
    hc = &http.Client{  
        Transport: roundTripper,  
    }  
} else {  
    hc = &http.Client{  
        Transport: &http.Transport{  
            TLSClientConfig: tc,  
        },  
    }  
}  
// ...  
hc.Post(url, "application/octet-stream", data)
```

# Prototype serveur

```
func RunServer(config *Config,
    logger *slog.Logger) error {

    mux := http.NewServeMux()

    mux.HandleFunc("/",

        func(writer http.ResponseWriter,
            request *http.Request) {
            bytesRead, err :=
                io.Copy(io.Discard,
                    request.Body)
```

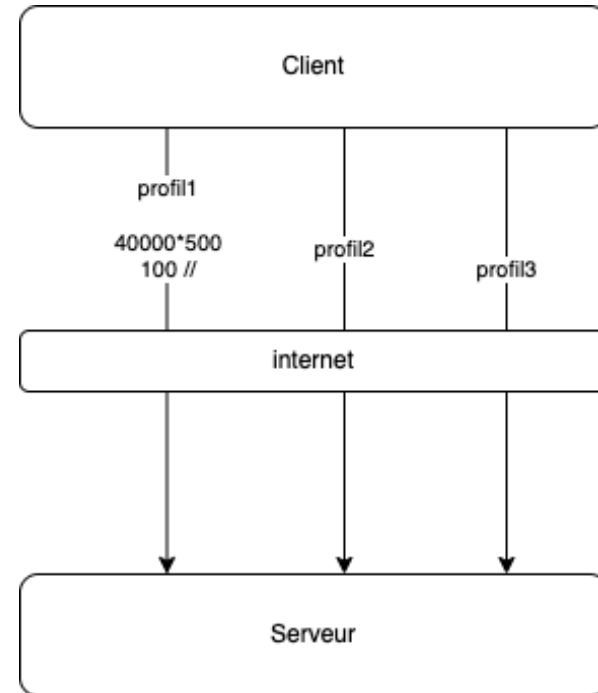
```
        if !config.IsHttp2 {
            err = http3.ListenAndServeQUIC(addr,
                config.CertFile, config.KeyFile, mux)

        } else {
            err = http.ListenAndServeTLS(addr,
                config.CertFile, config.KeyFile, mux)
        }
    }
```

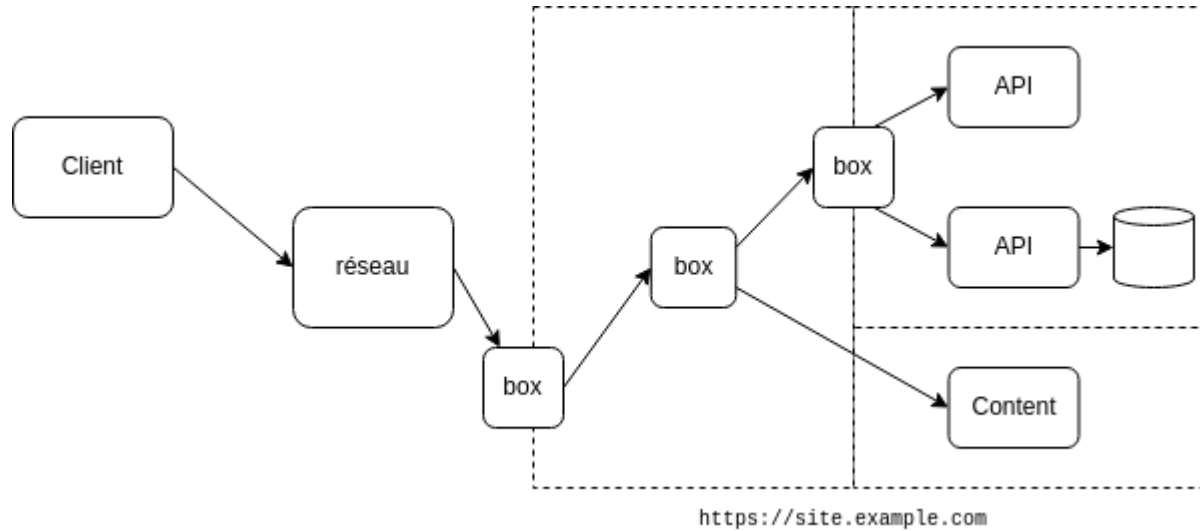
# Résultats HTTP/3 vs HTTP/1.1

- Profil de requêtes avec petits volumes
  - Gain de temps d'exécution jusqu'à 50%
  - Gain marginal de volume de données
- Pas de gain sensible pour des données volumineuses
- Pas d'erreur liée à la charge
- Pas d'erreur liée à la qualité du réseau

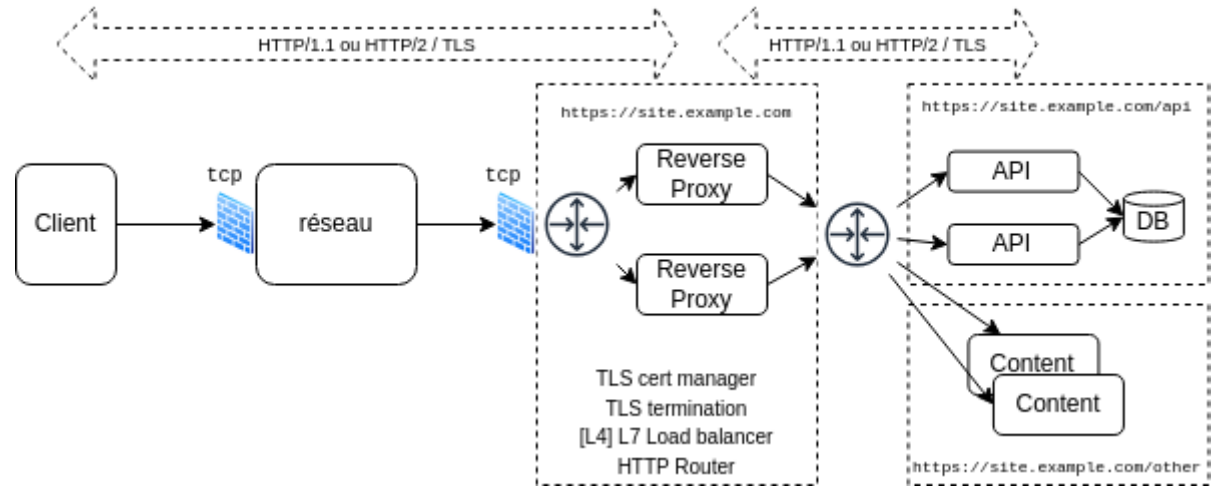
=> investiguer l'hébergement de l'API Cabri



# Hébergement

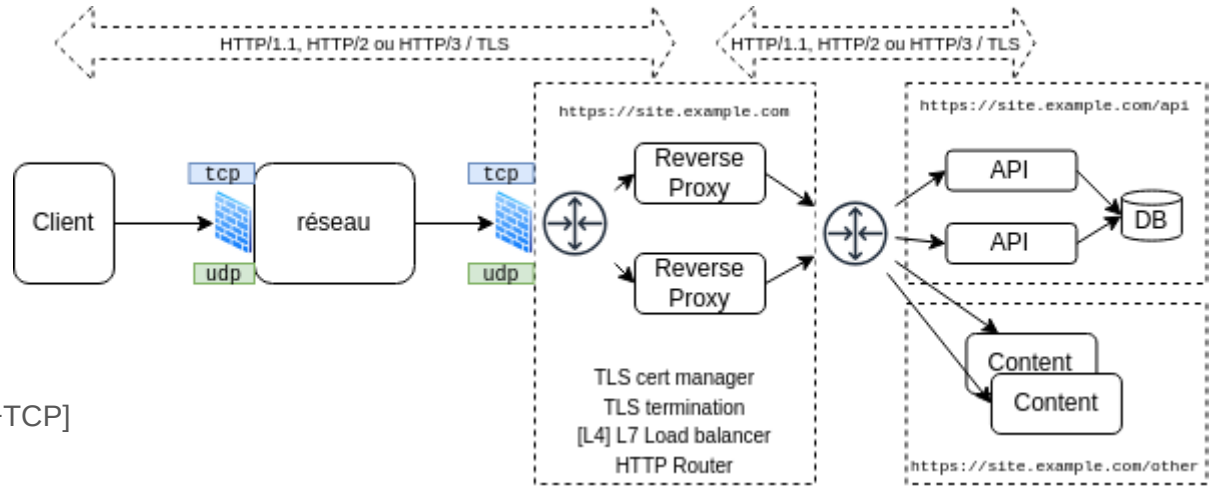


# Exemple avec reverse proxy



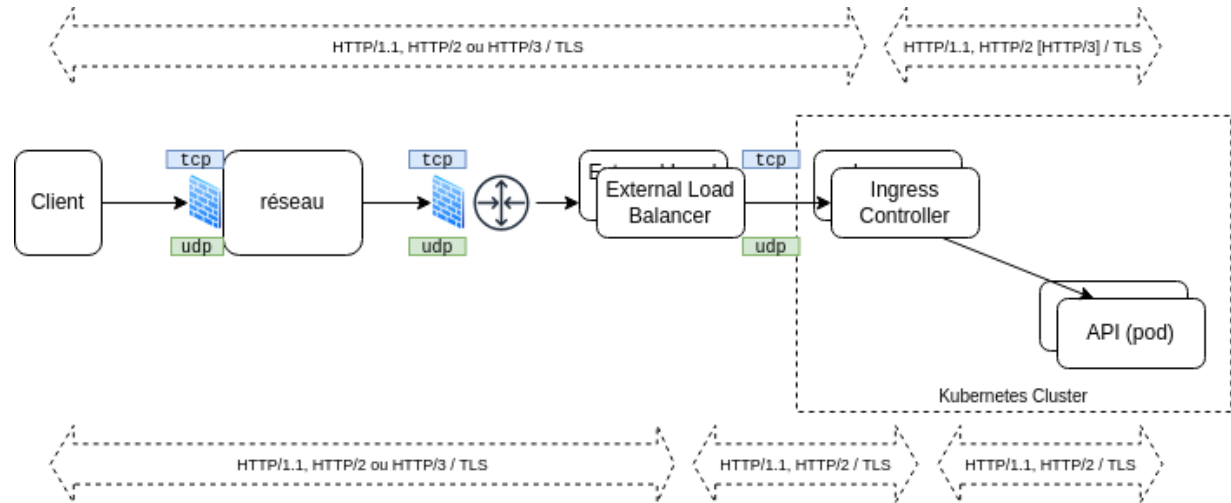
- Besoins
  - Sécurité : TLS [+ authentication]
  - Disponibilité
- Contraintes
  - CORS : DNS unique
    - => Routage
  - TLS : gestion certificat (eg ACME LetsEncrypt)
  - Equilibrage de charge : HTTP [ou TCP]

# HTTP/3 avec reverse proxy



- Adaptation directe avec rupture TLS
  - Equilibrage de charge : HTTP [ou UDP+TCP]
  - Latence induite par la rupture TLS
  - Reverse proxies qui gèrent le HTTP/3
    - HAProxy : inbound + outbound
    - Nginx expérimental : inbound
    - Traefik : inbound, TCP+UDP
    - Caddy : inbound
- NB : ACME challenge HTTP01 nécessite http:// => HTTP/1.1 ou HTTP/2

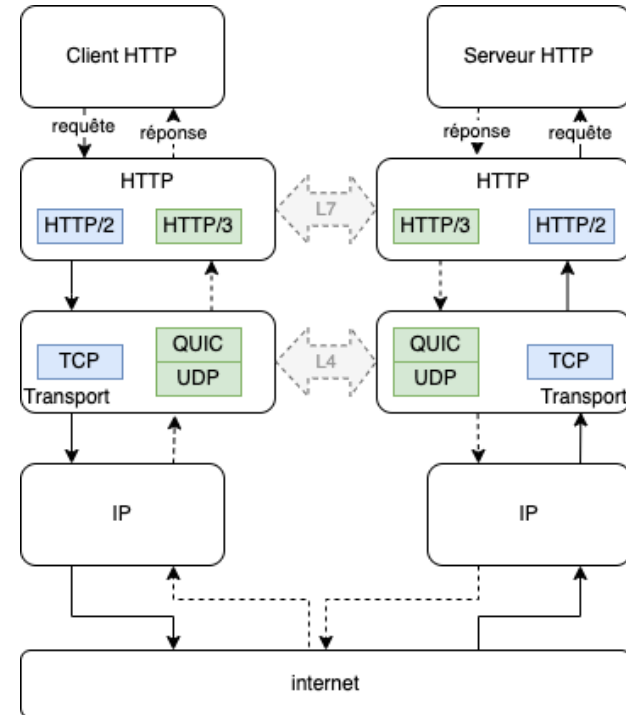
# Exemple avec Kubernetes



- Besoins
  - Sécurité : TLS [+ authentication]
  - Disponibilité
- Equilibrage de charge
  - External : TCP/UDP [ou HTTP]
  - Ingress : HTTP/1.1 ou HTTP/2
    - [ou Gateway API TCP/UDP]

# HTTP/3 : action ou veille?

- Bénéfices
  - Navigation fluide sur mauvais réseau
  - Performances pour APIs interactives
  - Comprendre et savoir
- Contraintes
  - Implémentation « dual-stack »
  - Solutions d'hébergement en devenir
- Opportunités
  - WebTransport over HTTP/3
  - QUIC :
    - RPC vs API
    - Expérimentations XMPP, SMB, streaming





Merci à  **Toulibre**, aux bénévoles et à l' 

Questions, remarques ?

# Références

- QUIC et HTTP/3 en profondeur
  - Analyse de la référence
    - <https://www.andy-pearce.com/blog/posts/2023/Mar/http3-in-practice-quic/>
    - <https://www.andy-pearce.com/blog/posts/2023/Apr/http3-in-practice-http3/>
  - Très sympa
    - <https://www.smashingmagazine.com/2021/08/http3-core-concepts-part1/>
    - <https://www.smashingmagazine.com/2021/08/http3-performance-improvements-part2/>
    - <https://www.smashingmagazine.com/2021/09/http3-practical-deployment-options-part3/>
- HTTP, TLS, performance
  - Background très complet (2013): <https://hpbn.co/>
  - HOL blocking: <https://github.com/rmarx/holblocking-blogpost>
- Study <https://quicwg.org/ops-drafts/draft-ietf-quic-applicability.html>
- Application (multimedia/video)  
<https://engineering.zalando.com/posts/2024/06/next-level-customer-experience-with-http3-traffic-engineering.html>
- Code :)
  - A QUIC implementation in pure Go: <https://github.com/quic-go/quic-go>
  - Mockup: [https://github.com/t-beigbeder/otvl\\_devops\\_tools/tree/main/src/go/ht3mock](https://github.com/t-beigbeder/otvl_devops_tools/tree/main/src/go/ht3mock)
  - Cabri : [https://github.com/t-beigbeder/otvl\\_cabri](https://github.com/t-beigbeder/otvl_cabri)

