Reverse HTTP

Schwartz, Reddy, Boucadair, Tiesel
HTTPBIS @ IETF 118

Cloudflare Tunnel

Protect your web servers from direct attack

From the moment an application is deployed, developers and IT spend time locking it down — configuring ACLs, rotating IP addresses, and using clunky solutions like GRE tunnels.

There's a simpler and more secure way to protect your applications and web servers from direct attacks: Cloudflare Tunnel.

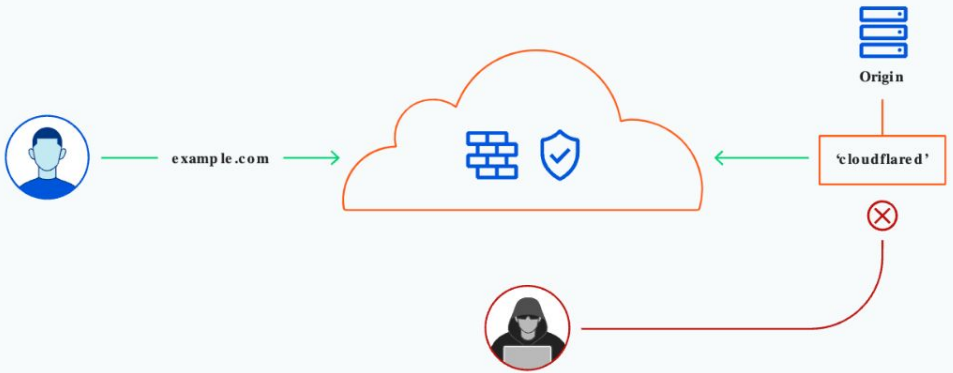
Ensure your server is safe, no matter where it's running: public cloud, private cloud, Kubernetes cluster, or even a Mac mini under your TV.

Protect web servers from direct attacks

Tunnel works with Cloudflare DDoS Protection and Web Application Firewall (WAF) to defend your web properties from attacks.

Once you deploy the Tunnel daemon and lock down your firewall, all inbound web traffic is filtered through Cloudflare's network.

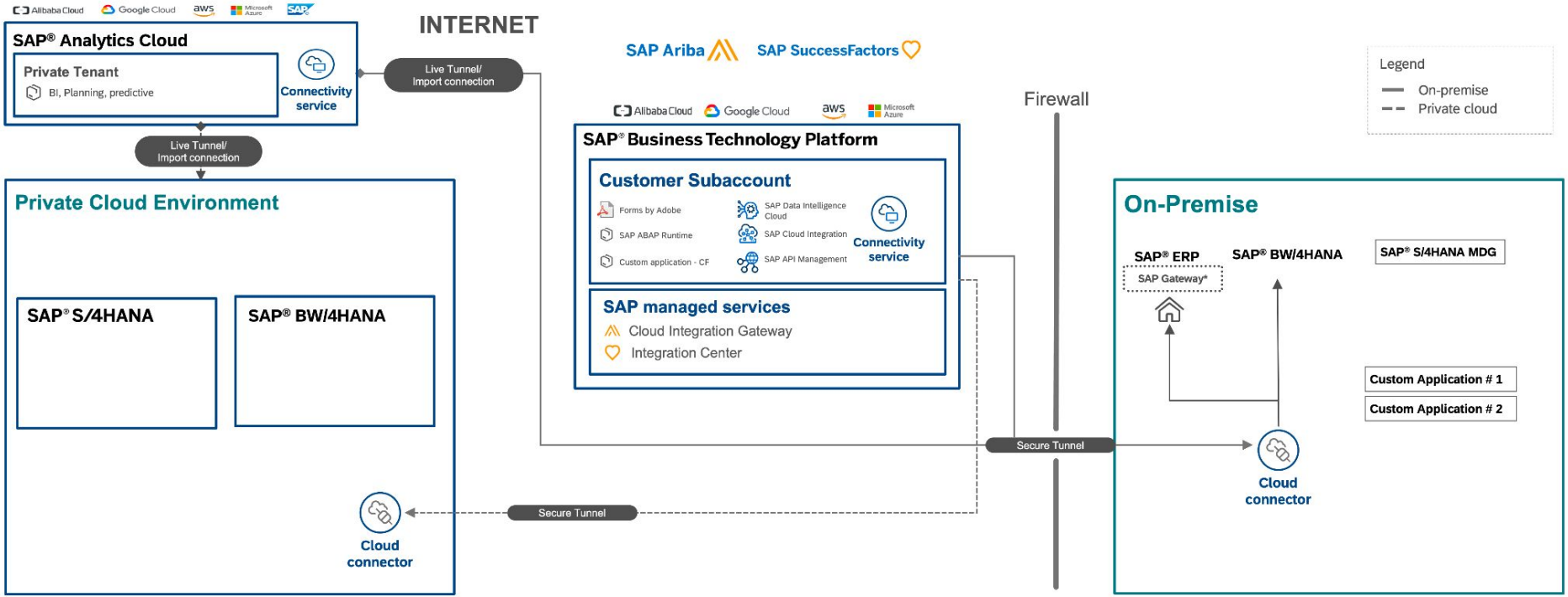
Now, your web server's firewall can block volumetric DDoS attacks and data breach attempts from reaching your application's origin servers.



https://www.cloudflare.com/products/tunnel/

SAP Cloud Connector / SAP BTP Connectivity Service

SAP Cloud Connector serves as a link between SAP BTP applications and on-premise systems,
so that SAP Cloud products can communicate securely with systems in a customer's on-premises/private cloud landscape.



https://blogs.sap.com/2022/02/03/cloud-connector-explained-in-simple-terms/

What are these things?

In HTTP terms:

- The origin server is not externally accessible as a transport server.
  - It's behind a firewall and/or at an unpredictable IP address, to avoid attacks.
- The origin server acts as a transport client to reach the HTTP query source.
  - That could be an HTTP client or an intermediary.

Why not just punch a hole in the firewall?

- Configuring firewalls is complicated and requires everything to be static.
- Firewalls don't prevent large volumetric attacks on small origin uplinks.

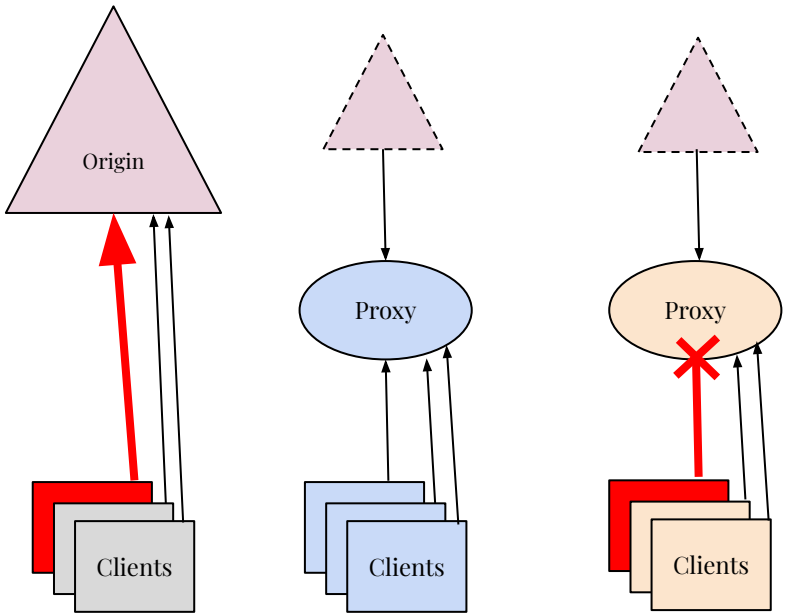
Why standardize?

Proprietary reverse origin protocols are working fine, with limitations:

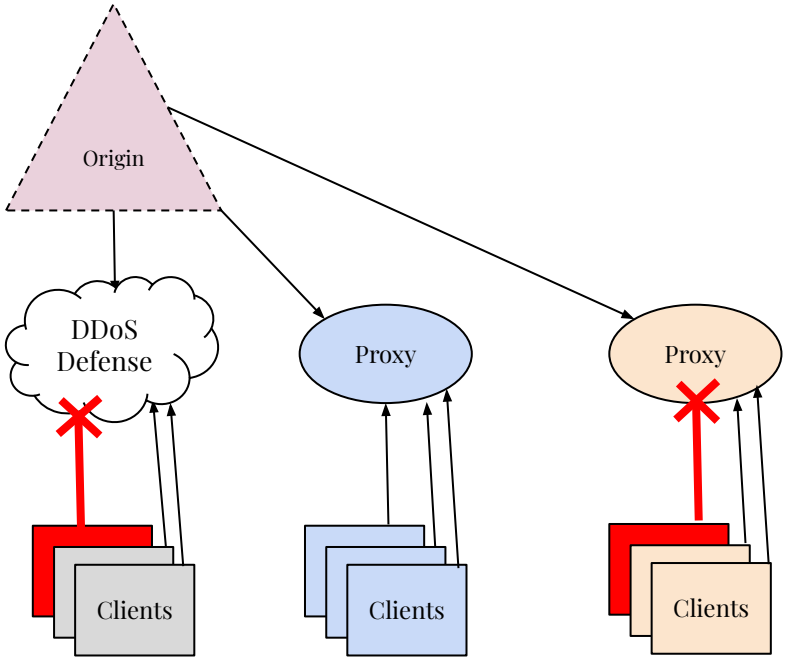
- INTEGRATION: Origins have to run software created by their cloud provider.
- MIGRATION: They don’t support easy mobility between providers.
- EVOLUTION: The protocols don’t keep up with all HTTP features (e.g. QUIC).

New Possibilities: Novel Architectures with Large Scale Proxies

Origin servers partitioned for attack resilience



DDoS defense bypass for cost reduction



-00 Protocol

- ALPN = “h2-reverse” or “h3-reverse”
- Totally standard* HTTP but with the roles reversed.
  - HTTP/3 requests flow on “ QUIC server-initiated” bidirectional streams.
- TLS CertificateRequest and “Client” authentication is mandatory.
  - TLS Server authentication is also mandatory, but that only identifies the HTTP client.
- Immediate ORIGIN frame is mandatory
  - *wildcards are permitted.
- “Via” headers are the same as in forward HTTP.
- Intermediaries can advertise support by SVCB at _http-reverse.$NAME
  - Encrypted ClientHello is allowed

Questions

- Do we need bidirectional multiplexing?
  - Provides fate-sharing for bidirectional communication patterns.
  - Could cover a broader range of Origin-to-Origin use cases.
  - Probably much more complicated than the current draft.
- What can we say about connection management?
  - Current text: *“Very large intermediaries SHOULD ensure that transitions to Reverse HTTP are gradual, so that large origins have time to establish multiple connections.”*
    - This is pretty vague…
- What do you think?