API-Gateways

MICROSERVICES

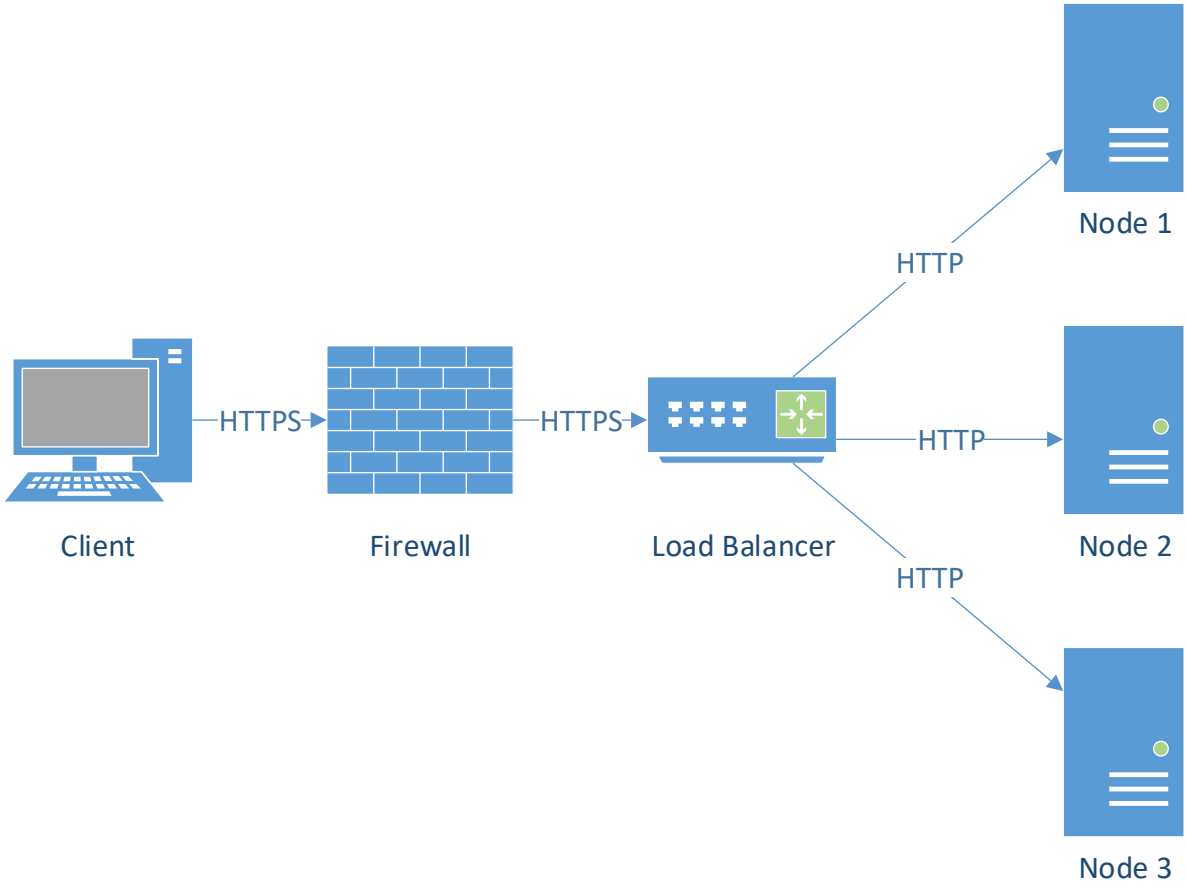
Content

- Load Balancing basics
- Load Balancing strategies
  - Round Robin
  - Least Connection
  - Chained Failover
  - …
- Failover techniques
  - Common address redundancy protocol (CARP)
  - Gateway Load Balancing Protocol (GLBP)
- API gateways
- Products
  - Traefik
  - HAProxy
  - …

Basics

- Load Balancers are needed to avoid single point of failures
- Load Balancers distribute calls sent to them to one or more instances
- Load Balancers keep track of their known backends to avoid errors when a services is no longer healthy
- Optionally they have additional features like *SSL termination*
  - Only one or a few servers where certificates have to be exchanged when a new certificate is created
  - No special handling required in the services behind the load balancer
  - Admins have to take care that communication between load balancer(s) and nodes is safe (e.g. VLANs)
- In Microservice environments it's essential that the load balancer(s) can be reconfigured dynamically (e.g. in combination with etcd or Consul)
- When you're able to scale your microservice instances but not the persistence layer of them you're just moving the single point of failure one layer backwards!

SSL termination



Load Balancing strategies

- Round-robin
- Weighted round-robin
- Least connection
- Weighted least connection
- Agent Based Adaptive Load Balancing
- Chained Failover (Fixed Weighted)
- Weighted Response Time
- Source IP Hash
- …

(Weighted) Round-robin

- There are many Round-robin algorithms
- The simplest one is to use a FIFO queue to keep track of all available backens
  1. Dequeue
  2. Relay request
  3. Enqueue
- Results in max-min-fairness (the longest waiting requests gets the highest priority)
- The weighted round robin algorithm gives every backend a weight and the scheduler takes the weights into account to prefer servers with a higher weight before servers with a lower weight (e.g. used for quality of service (QoS))

(Weighted) Least connection

- In contrast to round-robin the least connection algorithm takes the load of every node in account.
- The least connection algorithm relays an incoming request always to the node which has the lowest count of active connections.
- This way nodes with a higher performance handle more requests than nodes with lower performance without the need to configure weights.
- The weighted least connection variant enables the administrator the give nodes a weight. These weights are considered when two nodes serve the same count of active connections and the node with the higher weight is considered first.

Agent Based Adaptive Load Balancing

- Every node has an local agent installed which reports real time data to the load balancer (e.g. CPU usage, memory allocation,…)
- Load balancers takes load of every node into to account when a new request has to be relayed
- Can be combined with weighted round-robin or weighted least connection algorithms
- Used e.g. in Windows Terminal Server (RDS role after Server 2003)

Chained Failover

- All backend nodes are in a predefined chain
- Whenever the first node can’t handle/accept another request the next node in the chain is taken into account and so on
- Not a real load balancing protocol!

Weighted Response Time

- Kind of health check done by the load balancer(s)
- Uses the response time of the health check to determine the fastest server currently available
- Whenever a node is under heavy load the response times will be longer than the response times of a node with least load.
- Avoid overload of nodes.

Source IP Hash

- Algorithm creates a hash of source and destination IP (unique hash key)
- Hash key is used to determine to which node the request should be forwarded
- When the same client sends another request the hash key can be regenerated and the client gets forwarded to the same node
- Useful for stateful services (**don’t do that in microservices!**) when nodes aren’t able to sync session information because a client always gets relayed to the same node (as long as its source IP does not change)

Network failover

- But wait! If I have **1** load balancer what happens if this load balancer fails?
- Possible solution: multiple load balancers with multiple DNS A/AAAA records to balance the load of the load balancers -> but DNS does not check for availability and there are the caches…
- Better solution: configure network based failover:
  - Common address redundancy protocol (CARP)
  - *Gateway Load Balancing Protocol (GLBP) (just for routers)*

Common address redundancy protocol (CARP)

- Enables multiple hosts in the same LAN to share a set of IP addresses
- Available on BSD and Linux based hosts
- Master-slave (or more polite active-passive) based
- One master per *group of redundancy*
- Each *group of redundancy* shares one *virtual* IP
- A server maybe member of multiple *groups of redundancy*
- Every server needs a second unique IP address for communication (best practice is to configure two unique IP addresses: one for LAN communication and one for the communication between all members of the *group of redundancy* e.g. heartbeats)
- Whenever the master fails a slave takes over and answers all incoming requests
- Can be combined with DNS round robin e.g. two *groups of redundancy* with two members each to ensure that always two load balancers are available

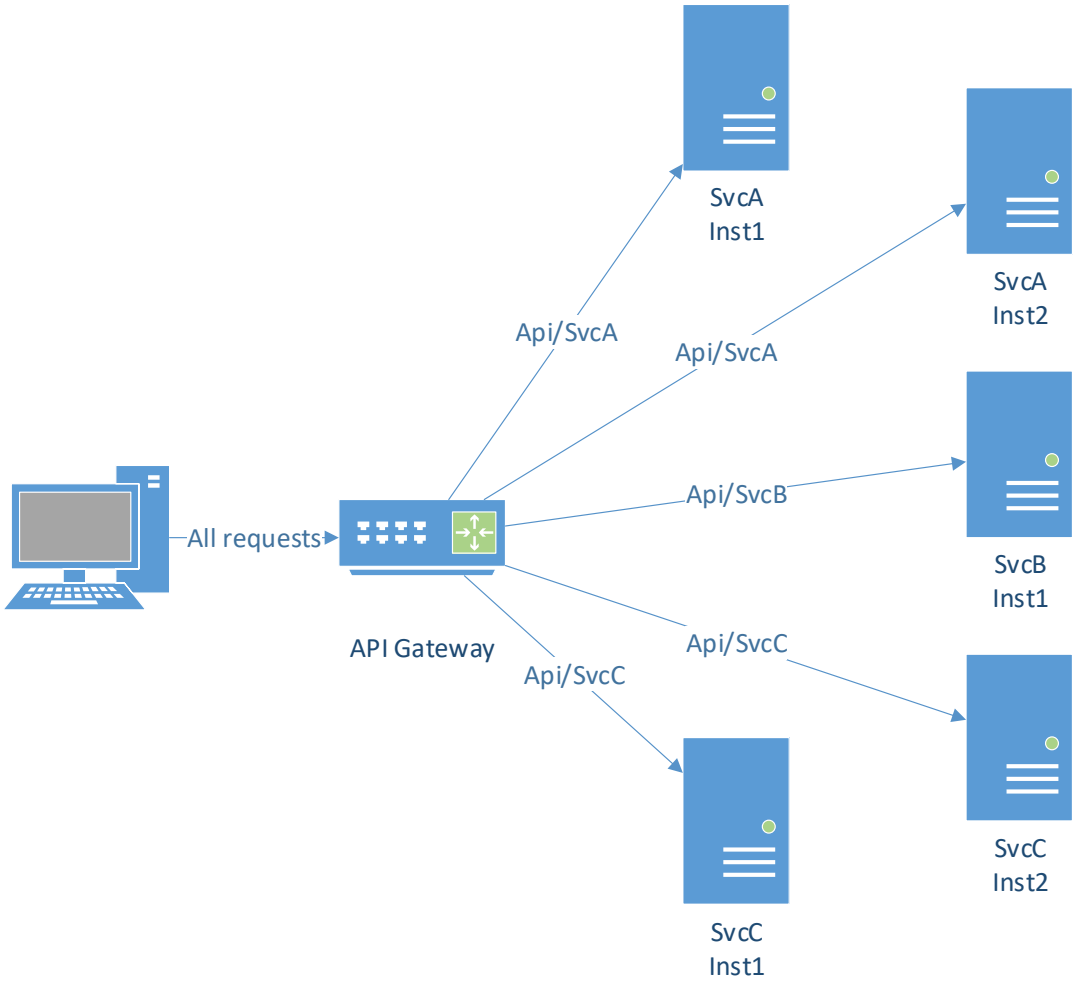
Gateway Load Balancing Protocol (GLBP)

- Proprietary protocol created by Cisco for redundant routers
- Allows weighting parameter to be set
- Based on the weights (in a virtual router group) ARP requests will be answered with MAC addresses pointing to different routes
- Balances in round-robin fashion by default
- Elects one *Active Virtual Gateway (AVG)* for each group
- AVG assigns every listener (and itself) virtual MAC addresses which enables *Active Virtual Forwarders (AVF)*
- Each AVF is responsible to forward packages sent to its virtual MAC address

API Gateways

- API Gateways are a crucial part of every microservices environment
- API Gateways enable a microservices environment to scale by implement load balancing (e.g. round-robin based)
- API Gateways are the single entry point for your whole application
- Access to specific services is managed by:
  - DNS-Host per Service (A/AAAA/CName e.g. ServiceA.my-domain.com)
  - Virtual Routes (e.g. gateway.my-domain.com/ServiceA)
- API Gateways are a kind of server-side service discovery (usually just for client apps but it's possible to use it also for cross service calls)
- They also hide implementation details by optionally aggregating all internal APIs to one (or more in case of *Backends for frontends*) in the point of view of the client app(s)
- In the case of custom API Gateways the gateway may also execute calls to multiple services and aggregate the responses answering to the client request

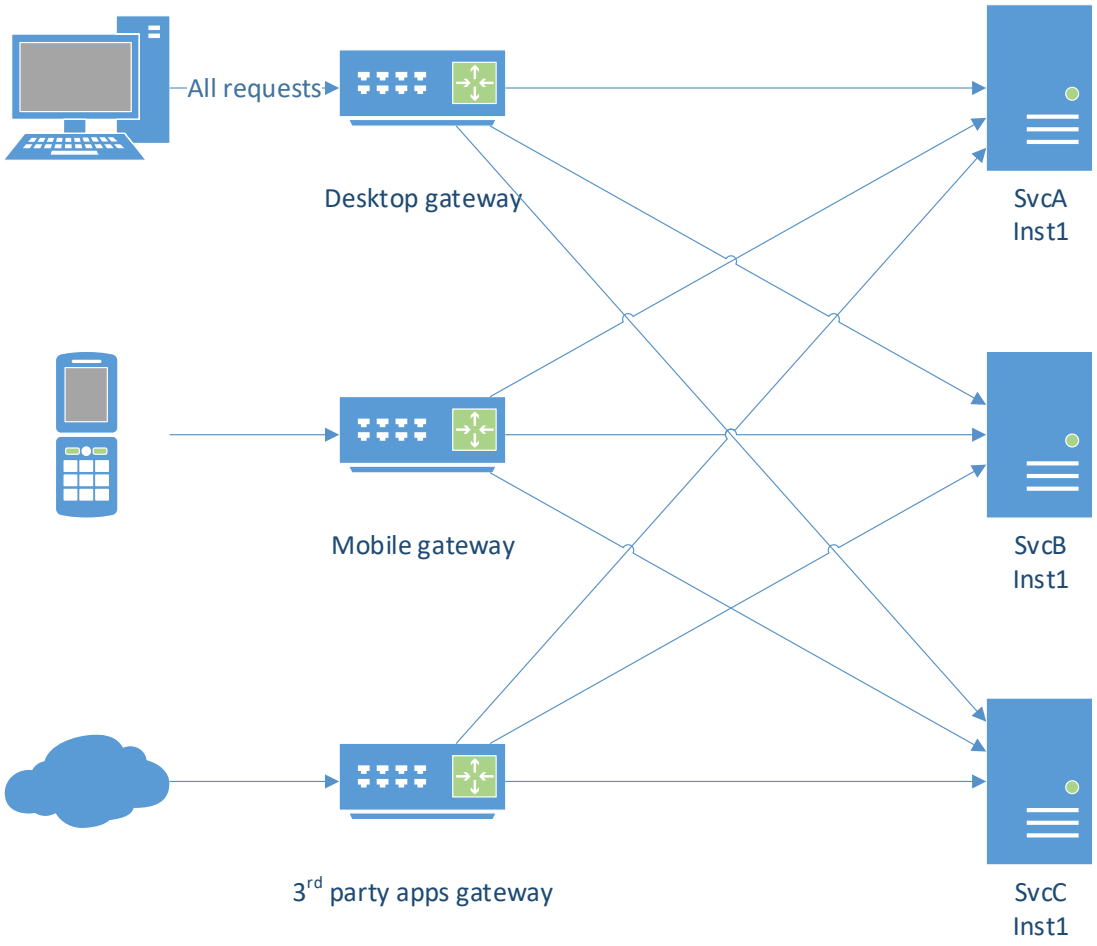
API Gateways



Variation: Backends for frontends

- Configure a API Gateway per kind of frontend e.g.
  - One for your web app
  - One for your mobile app
  - One for all 3rd party applications (public API)
- The backends for frontends-pattern ensures the *optimal* API for every kind of application (e.g. gRPC gateway for desktop apps but RESTful API for web apps)

Backends for frontends



Products

- General load balancers
  - HAProxy
  - Nginx
- Created for microservices
  - Traefik
  - Fabio

HAProxy

- Open source (GPLv2) high availability load balancer and proxy written by Willy Tarreau (core contributor of the Linux kernel)
- Initial release 2001
- Can be configured dynamically e.g. with consul-template
- Can be used for SSL termination
- Very good performance (dual-core servers are able to serve up to 40k requests/s and are able to saturate 2Gbit/s)
- Used by GitHub, Bitbucket, Stack Overflow, Reddit,…

StackOverflow architecture

Nginx

- Open source (BSD-like) web server which can also be used as reverse proxy, load balancer and HTTP cache
- Developed by Igor Sysoev (company founded 2011)
- Initial release 2004
- Can be dynamically reconfigured e.g. with consul-template
- According to w³Techs 35.9% off **all** websites in the internet are hosted by Nginx (according to Netcraft 19%)
- Performance compared to HAProxy is a little lower (~33.5k requests/s) but still 4 times better than Apache

Fabio

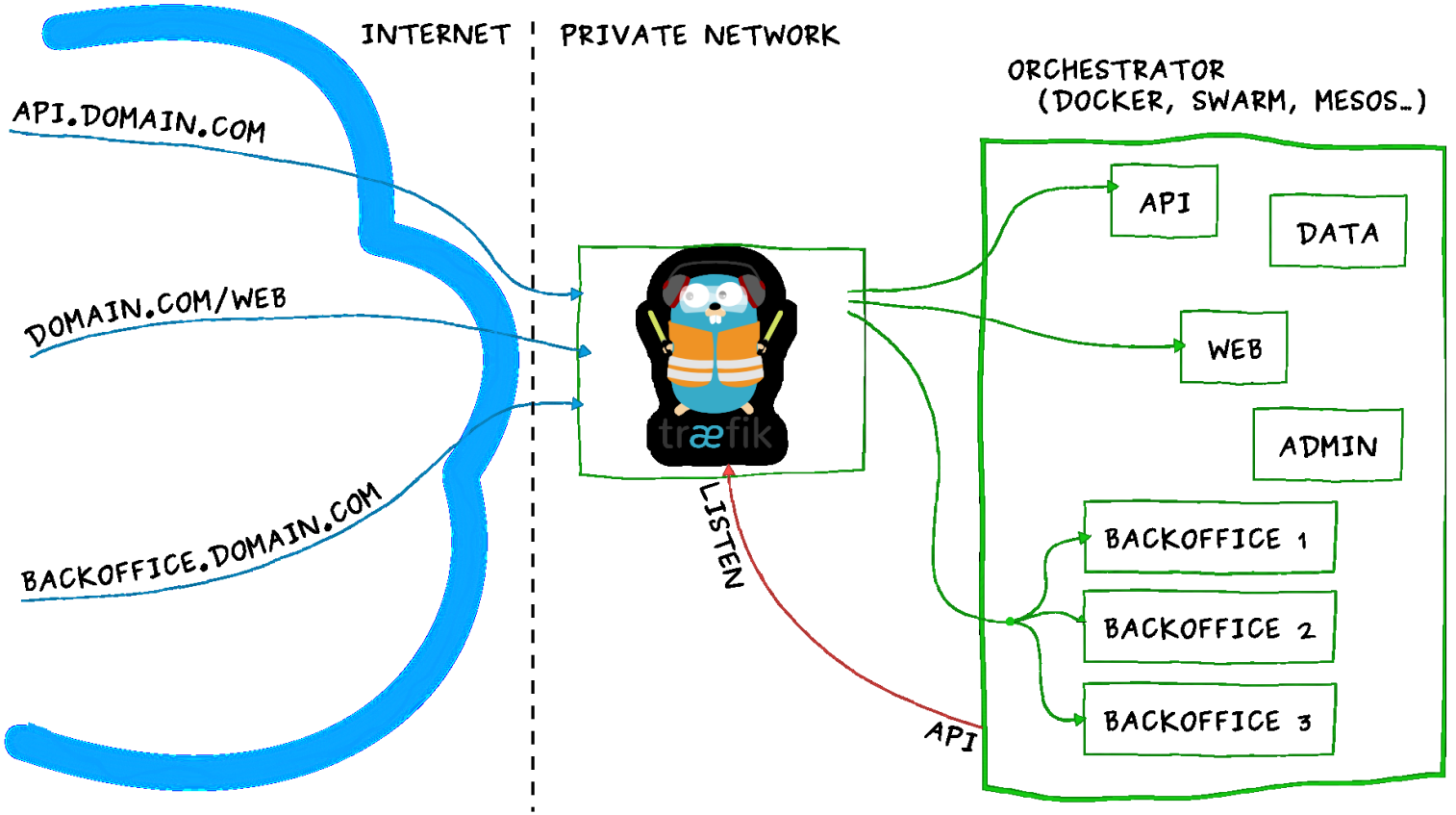
- Open source (MIT) zero-conf load balancer
- Developed by ebay.nl
- 1st GitHub release in 2015
- Written in Go
- Built to deploy microservices managed by Consul
- Performance is a little bit lower than Nginx (~23k requests/s) but still a young project
- Powers already some large websites (markplaats.nl, gumtree.com.au, www.kijiji.it)

Fabio

Traefik

- Open source (MIT) reverse proxy and load balancer developed with microservices and containers in mind
- 1st GitHub release in 2015
- Written in Go
- Built-in dynamic configuration with many supported Backend (Docker, Swarm mode, Kubernetes, Marathon, Consul,…)
- Performance compared to HAProxy and Nginx is a little bit lower (~28.4k requests per second) but it's still a young project
- Part of the Mantl stack

Traefik



Source: https://docs.traefik.io/basics/

Supported strategies

Product	Round robin	WRR	Least connection	Sticky session
Traefik	X	X		X
HAProxy	X		X	X
Nginx	X		X	X
Fabio	X	X		