



safespring

Anders Bruvik

Infrastructure engineer at Safespring



@bruvik

Creating an open source load balancer for S3

www.safespring.com



Infrastructure company based in Sweden/Norway

Offers storage, compute and backup as a service from local data centres

Built with Open Source – Openstack/CEHP





What is **SUNET** object storage?

Private cloud developed by **Safespring**

A solution to data gravity

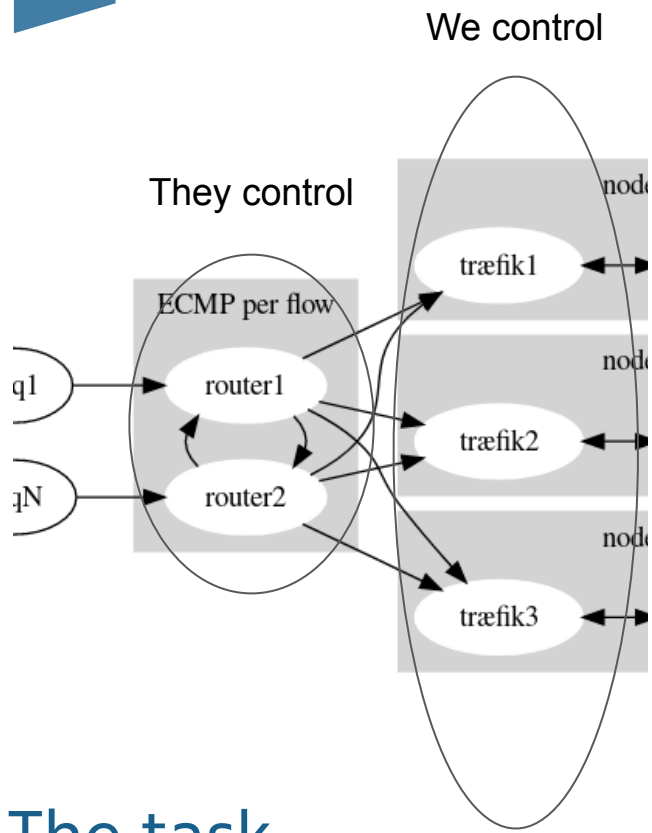
On premises

Data stay close to its users

Available

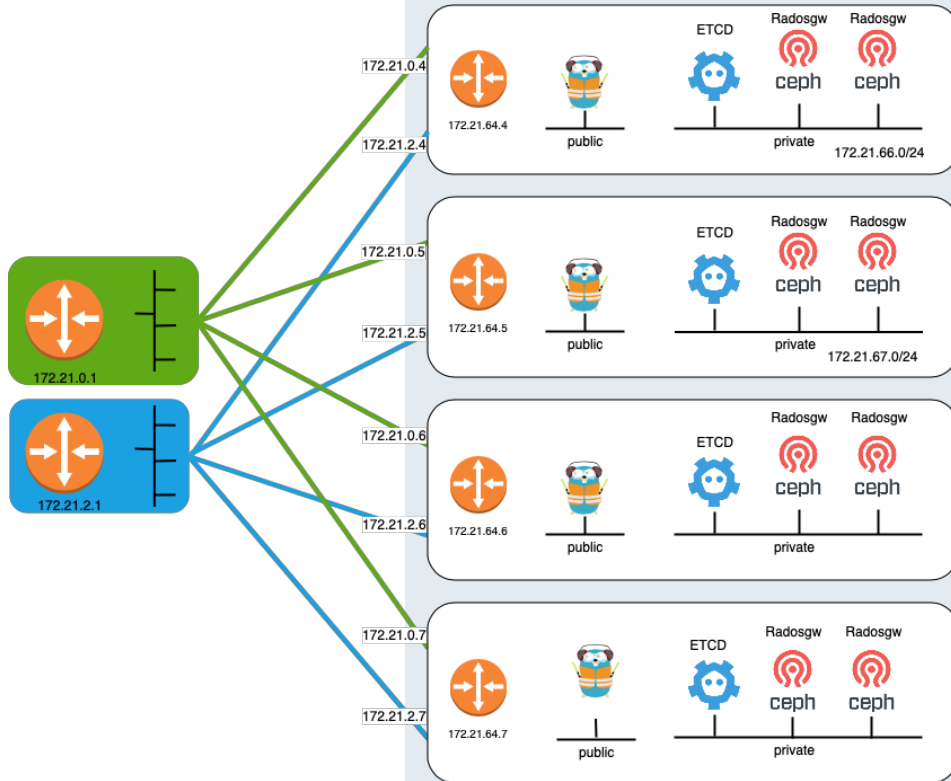
Expandable

Distributed



- Create a reusable load balancer built from open source components
- Support a hybrid cloud service - we want to deploy in customer datacenter
- Focus on simplicity, reliability, speed

The task





The components

- Bird
- Træfik
- Etc
- Letsencrypt
- Radosgw (Ceph)
- Prometheus

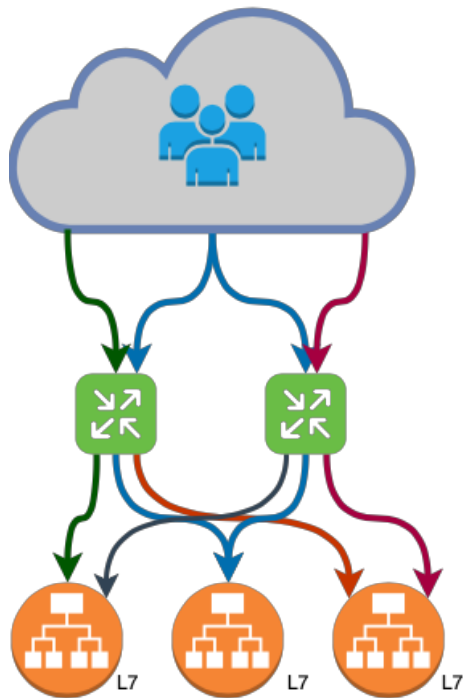


Internet as a design pattern for the datacenter cluster network ensure scalability and predictable performance.

No surprises now, no surprises in the future.

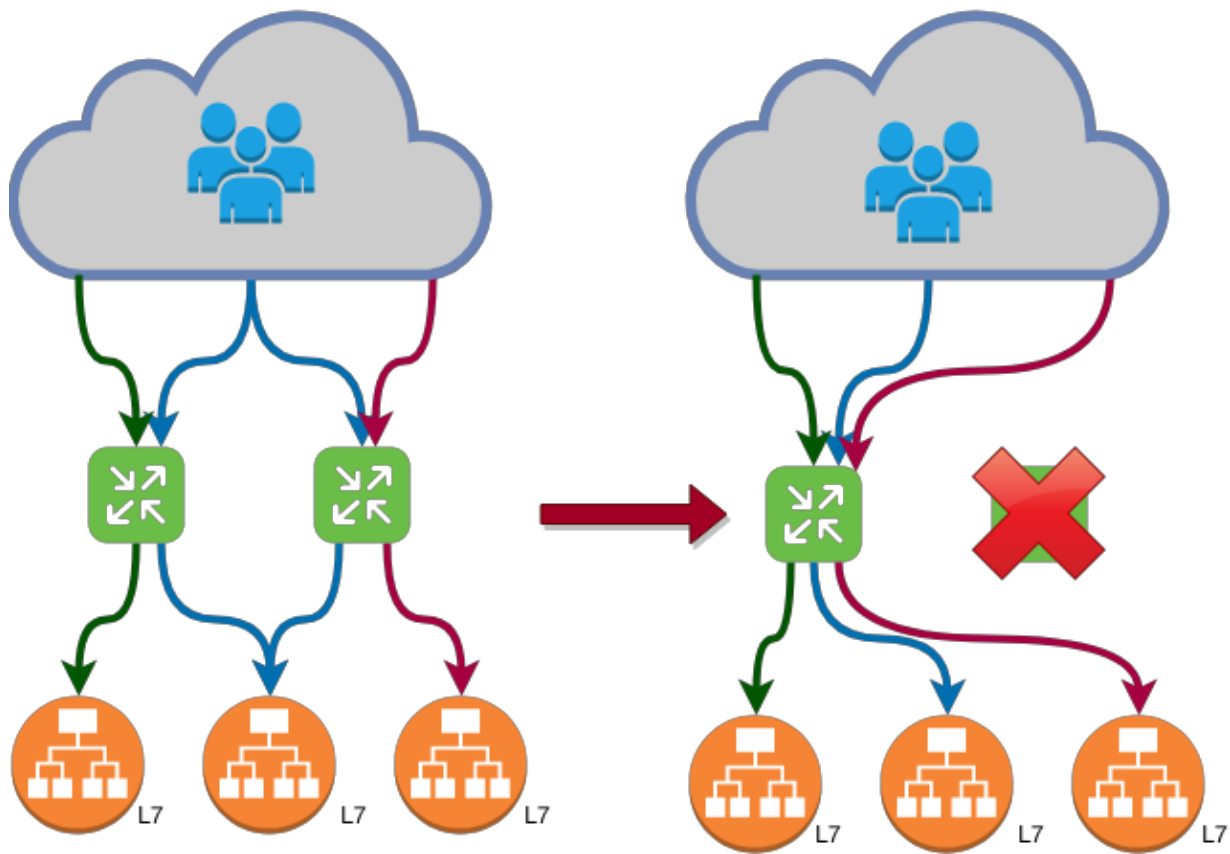
Technically, this means BGP is used everywhere, even for the last hop to each server node.

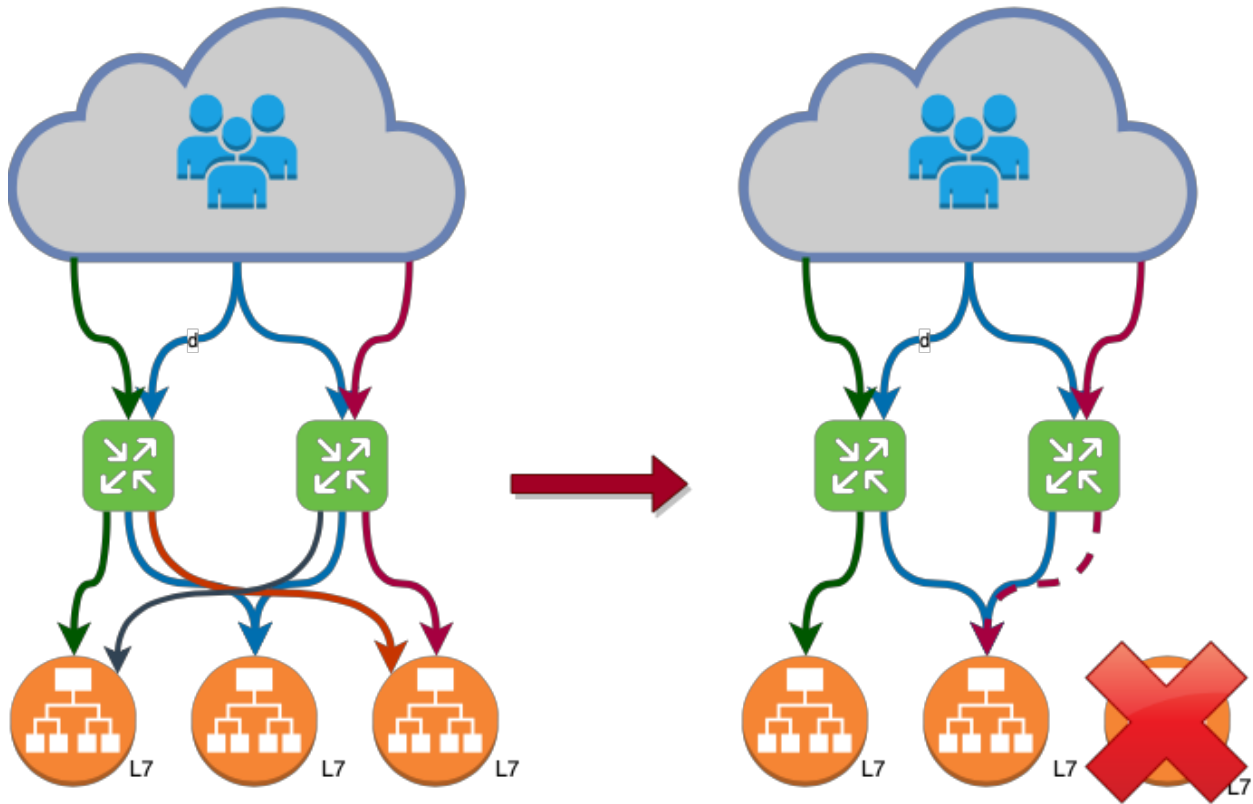
BGP everywhere

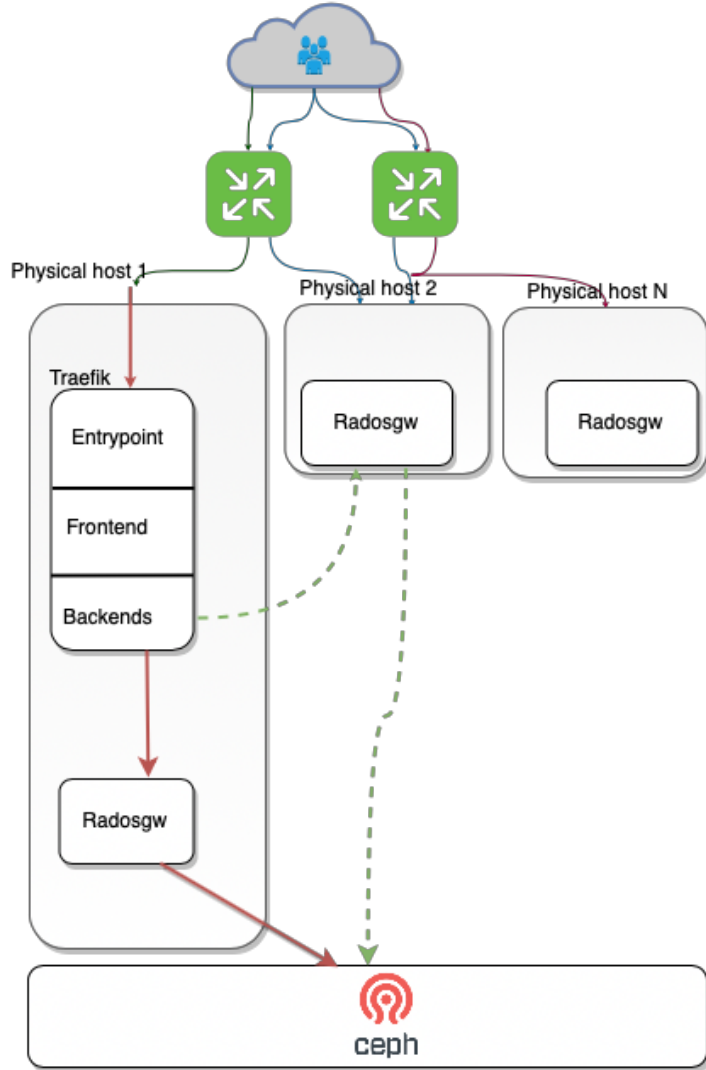


- We chose ECMP as routing strategy
- Ensures maximum capacity is used
- Achieve redundancy
- Per flow ECMP

ECMP





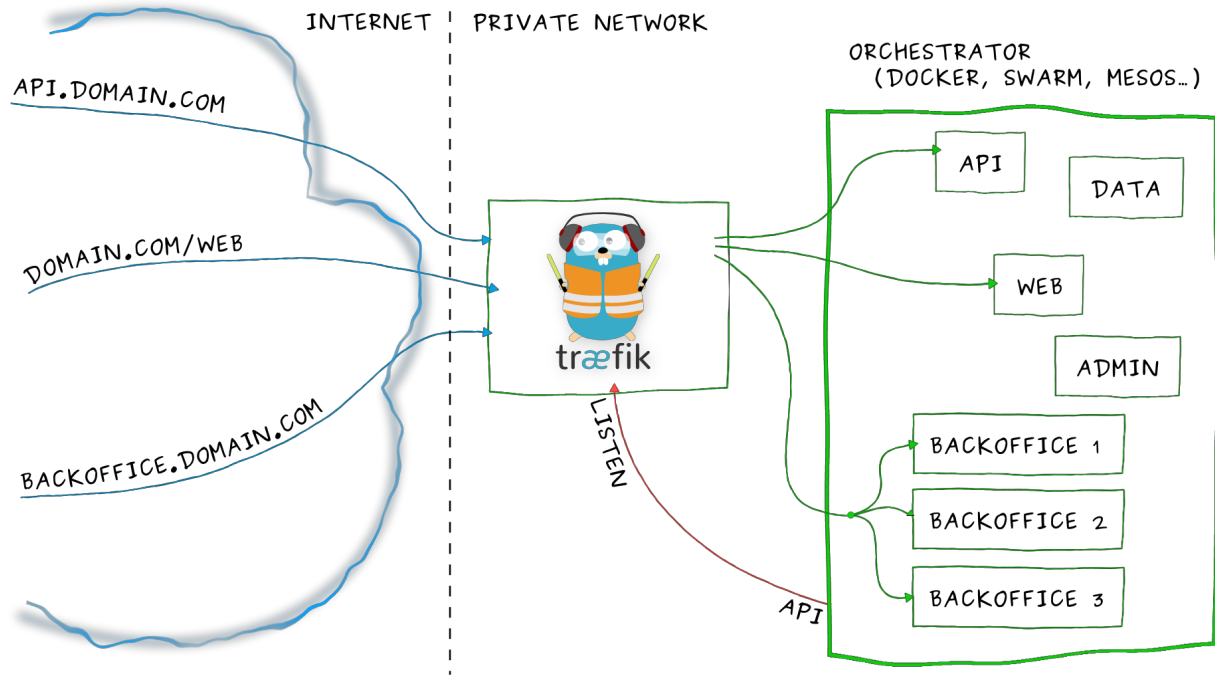


Fast path design



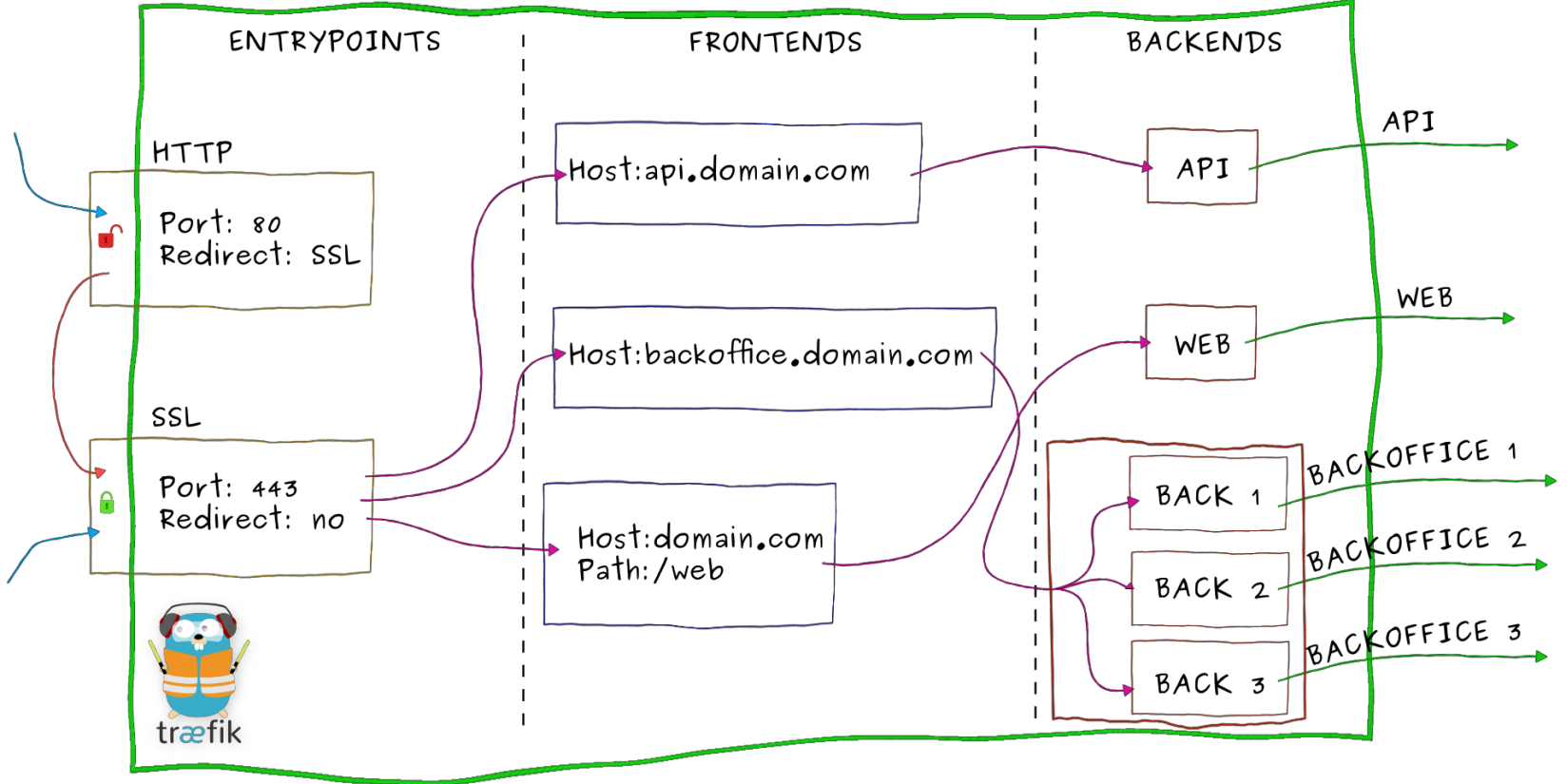
traefik

- Open Source Load balancer/Reverse proxy written in GO
- Lightweight, fast, easy configurable
- Can dynamically load configuration changes



Træfik

architecture





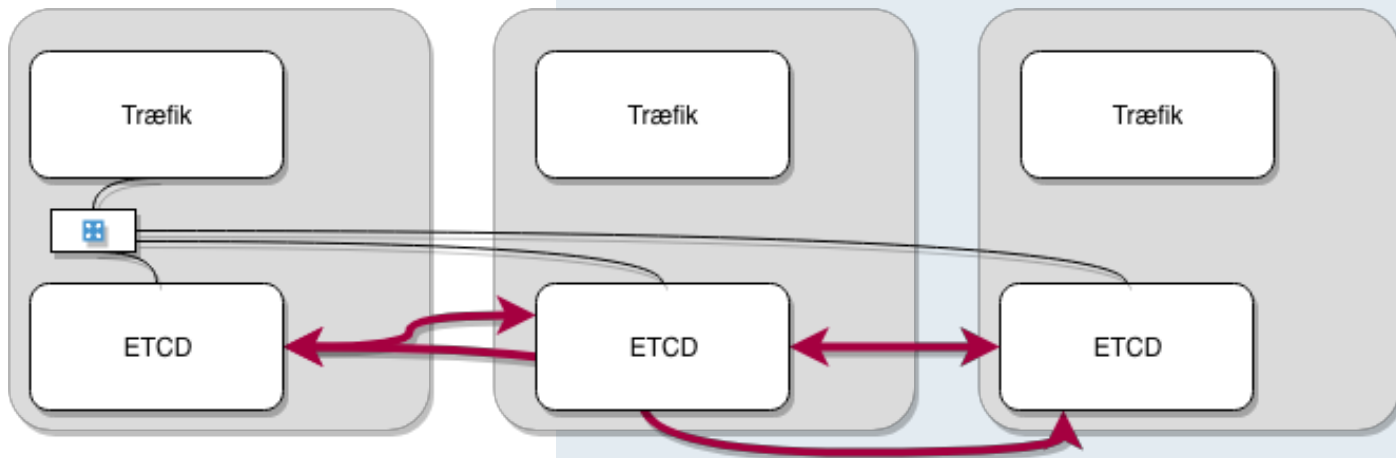
```
[entryPoints]
[entryPoints.http]
  address = ":80"
  [entryPoints.http.redirect]
    entryPoint = "https"
[entryPoints.https]
  address = ":443"
  [entryPoints.https.tls]
[entryPoints.api]
  address = ":8080"
  [entryPoints.api.whitelist]
    sourceRange = ["::1", "127.0.0.1"]
[entryPoints.traefik]
  address = "{{ ansible_default_ipv4.address }}:9091"
```

Multiple Traefik instances sharing same configuration

Static configuration as files

Dynamic configuration stored in etcd cluster

Traefik configuration



- Distributed key value store
- Used for keeping dynamic Træfik configuration synchronised

ETCD



```

# use etcd provider
[etcd]
endpoint = "127.0.0.1:23790"
watch = true
prefix = "/traefik"
useAPIV3 = true
  [etcd.tls]
  ca = "/etc/etcd/certs/ca.pem"
  cert = "/etc/etcd/certs/etcd-client.pem"
  key = "/etc/etcd/certs/etcd-client.key"
  insecureSkipVerify = true
```



```
[acme]
email = "contact@service.com"
storage = "/traefik/acme/account"
onHostRule = true
entryPoint = "https"
  [acme.httpChallenge]
    entryPoint = "http"
[[acme.domains]]
  main = "my.s3.service.com"
```

- TLS is terminated at the load balancer
- Allows us to use letsencrypt certificates - free and easy
- Træfik supports ACME, data stored in KV-store

Letsencrypt



○ ○ ○

```
#[backends]
[backends.radosgw]
  [backends.radosgw.servers.server1]
    url = "https://10.11.12.13:7480"
    weight = 10
  [backends.radosgw.servers.server2]
    url = "https://10.11.12.14:7480"
    weight = 1
  [backends.radosgw.circuitbreaker]
    expression = "NetworkErrorRatio() > 0.5"
```

Each Træfik instance has multiple backends

Using weighting to default to backend on same physical host

Monitoring status using systemd service

Remove route - restart services

Handling failure



```
[metrics]
#...
# To enable Traefik to export internal metrics to Prometheus
[metrics.prometheus]
# Name of the related entry point
#
# Optional
# Default: "traefik"
#
entryPoint = "traefik"
```

Prometheus is used for monitoring
Traefik has a built in metric exporter

Monitoring



```
193.11.89.166:9091/metrics

# HELP go_gc_duration_seconds A summary of the GC invocation durations.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 1.1436e-05
go_gc_duration_seconds{quantile="0.25"} 1.4413e-05
go_gc_duration_seconds{quantile="0.5"} 2.3251e-05
go_gc_duration_seconds{quantile="0.75"} 3.6378e-05
go_gc_duration_seconds{quantile="1"} 7.7003e-05
go_gc_duration_seconds_sum 0.000439975
go_gc_duration_seconds_count 15
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 99
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 8.114656e+06
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 5.5949688e+07
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.46661e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 845114
# HELP go_memstats_gc_cpu_fraction The fraction of this program's available CPU time used by the GC since the program started.
# TYPE go_memstats_gc_cpu_fraction gauge
go_memstats_gc_cpu_fraction 8.548869226104672e-05
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 2.392064e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 8.114656e+06
# HELP go_memstats_heap_idle_bytes Number of heap bytes waiting to be used.
# TYPE go_memstats_heap_idle_bytes gauge
go_memstats_heap_idle_bytes 5.4960128e+07
# HELP go_memstats_heap_inuse_bytes Number of heap bytes that are in use.
# TYPE go_memstats_heap_inuse_bytes gauge
go_memstats_heap_inuse_bytes 1.198464e+07
# HELP go_memstats_heap_objects Number of allocated objects.
# TYPE go_memstats_heap_objects gauge
go_memstats_heap_objects 54669
# HELP go_memstats_heap_released_bytes Number of heap bytes released to OS.
# TYPE go_memstats_heap_released_bytes gauge
go_memstats_heap_released_bytes 5.40672e+07
# HELP go_memstats_heap_sys_bytes Number of heap bytes obtained from system.
# TYPE go_memstats_heap_sys_bytes gauge
go_memstats_heap_sys_bytes 6.615852e+07
# HELP go_memstats_last_gc_time_seconds Number of seconds since 1970 of last garbage collection.
# TYPE go_memstats_last_gc_time_seconds gauge
go_memstats_last_gc_time_seconds 1.548424824791637e+09
# HELP go_memstats_lookups_total Total number of pointer lookups.
# TYPE go_memstats_lookups_total counter
go_memstats_lookups_total 0
# HELP go_memstats_mallocs_total Total number of mallocs.
```



Enable query history

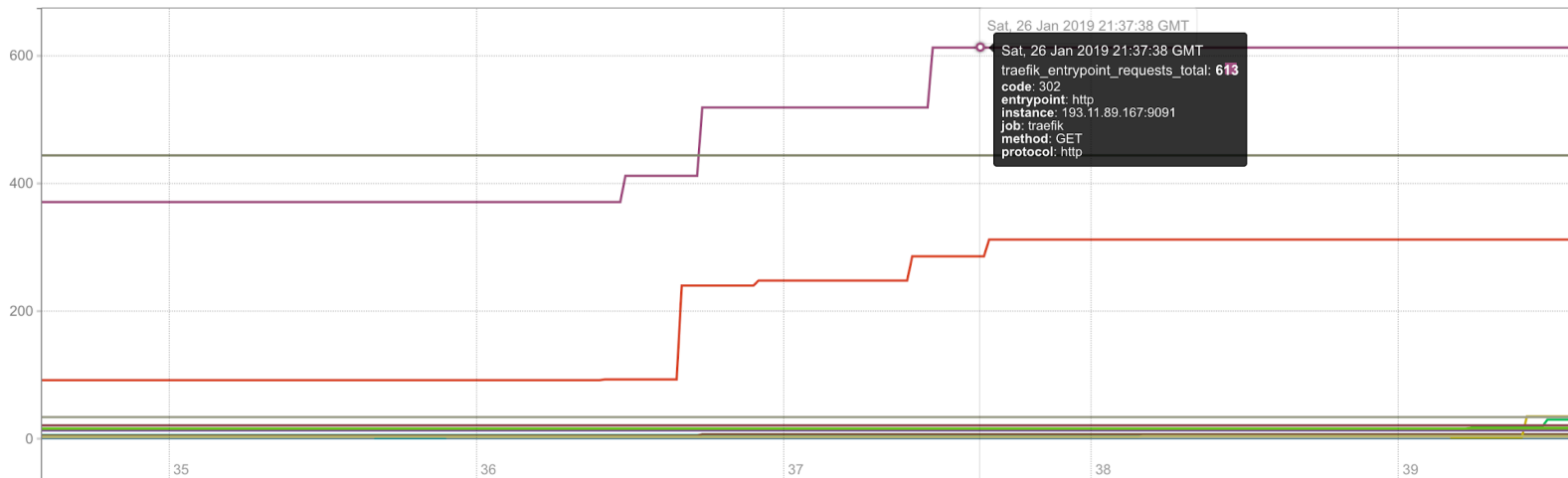
traefik_entrypoint_requests_total{status!~"4.."}

Load time: 73ms
Resolution: 1s
Total time series: 17

Execute

Graph **Console**

5m 2019-01-26 21:39 Res. (s) stacked



Sat, 26 Jan 2019 21:37:38 GMT
Sat, 26 Jan 2019 21:37:38 GMT
traefik_entrypoint_requests_total: **613**
code: 302
entrypoint: http
instance: 193.11.89.167:9091
job: traefik
method: GET
protocol: http

traefik_entrypoint_requests_total{code="499",entrypoint="https",instance="193.11.89.167:9091",job="traefik",method="GET",protocol="http"}
 traefik_entrypoint_requests_total{code="499",entrypoint="https",instance="193.11.89.166:9091",job="traefik",method="GET",protocol="http"}
 traefik_entrypoint_requests_total{code="404",entrypoint="https",instance="193.11.89.167:9091",job="traefik",method="POST",protocol="http"}



ANSIBLE

Configuration management

We use Ansible for configuration management

Heavily investing in Ansible roles to make configuration reusable

Separating configuration data and configuration

Building tooling to convert Ansible roles to images, docker images or deploy directly

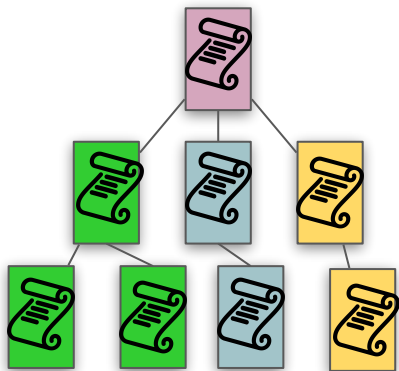


Safespring DevOps

- Workflow

What is needed?

- A mechanism to build (**Smie** - forge).
- A place to store artefacts - could be image, container or binary (**Naust** - boat house)
- Mechanism for deployment (**Seter** - settlement) that could describe different runtime environments

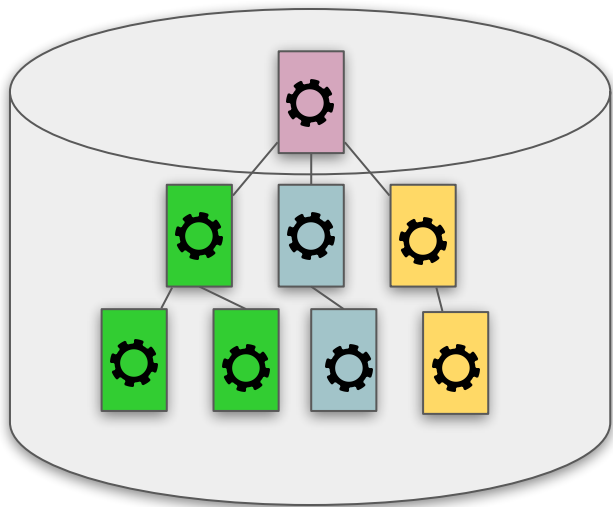


Safespring

DevOps - Smie

What is Smie?

- Wrapper around Packer (Hashicorp)
- Produces artefacts
- All artefacts can be built separately
- Role: service, endpoint or component

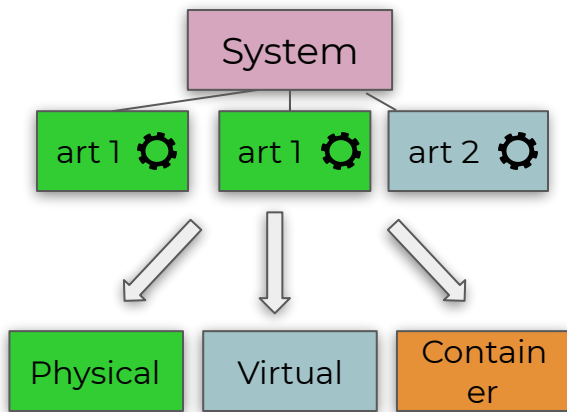


Safespring

DevOps - Naust

What is Naust?

- Both source and destination for Smie (cut dependencies to Internet repos)
- Full control over everything built for production
- Protocols:
 - HTTPS/file, S3, Docker Registry
- Protocols depend on target systems
- Everything built get an URI with metadata (type,version, date)



Safespring

DevOps - Seter

What is Seter?

- Wrapper around Ansible and Terraform provisioning mechanism
- Describes a system which is a set of artefacts in order to get a component running
- e.g Ceph Object Storage backend needs a set of OSD and RadosGW role images
- Also describes target: physical node, virtual node or container



Safespring DevOps

- Advantages

- Update systems faster
 - Lower barrier to changes
- Reproduce systems as needed
 - Build everything with as few dependencies as possible
- Add or change easily
 - Target the affected nodes easily
- Verify that software works as intended
- Scales better with many operators



Does it work?

Yes!



Actually deploy this :)

Test failure scenarios - chaos engineering

Metrics – getting useful

Move components to docker images

Test and profile different load balance strategies

Future work



QA

Follow us

[linkedin.com/company/safespring](https://www.linkedin.com/company/safespring)



twitter.com/safespring



twitter.com/bruvik