2020-10-15

CLOUDFLARE

How Nomad and Consul are used at Cloudflare

Thomas Lefebvre

CLOUDFLARE

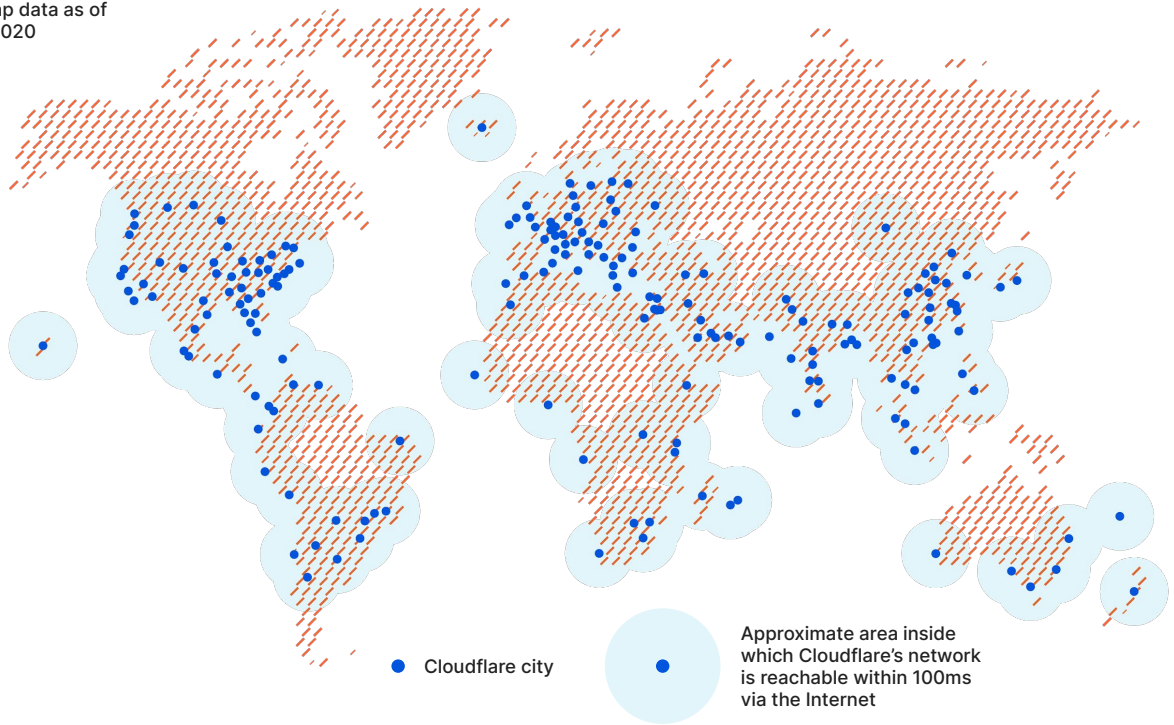
A bit about me

- Edge Platform team
- SRE at Cloudflare for 4 years
- Role:
  - Production reliability
  - Projects:
    - Provisioning
    - Release management
    - Task scheduling

CLOUDFLARE

Cloudflare's network operates at massive scale

Note: map data as of Jan 15, 2020



25M+

Internet properties

200+

cities and 100+ countries

72B

cyber threats blocked each day in Q2'20

99%

of the Internet-connected population in the developed world is located within 100 milliseconds of our network

CLOUDFLARE

Use case walkthrough

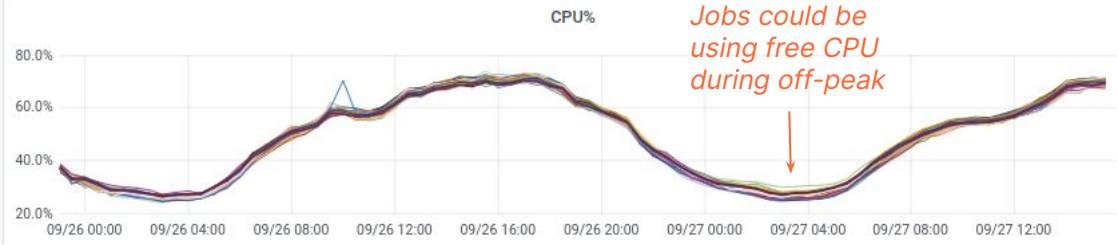
CLOUDFLARE

Some history

- At Cloudflare's Edge, all servers run all software
- Easy and works extremely well for most customer facing services

Recently:

- Internal asks for more on-demand workloads
- PoP management services do not need to run on all machines



CLOUDFLARE

Two options for new services

Options	Pros	Cons
Run on all machines	Easy scalability	Uses more resources than it needs Increased risk of noisy neighbor
Run on a static list of machines	Less wasteful of resources	**Each software has to engineer their reliability strategy** Suboptimal reliability to hardware failures

CLOUDFLARE

A new (and better) option

A new layer of dynamic task scheduling in every PoP

Objectives:

- Abstracting “my service should be running at all times”
- Unique place to add resource isolation and security
- Improved reliability to hardware failures
- Service only uses the resources it needs
- Minimize operational cost of the new system

CLOUDFLARE

Why Nomad

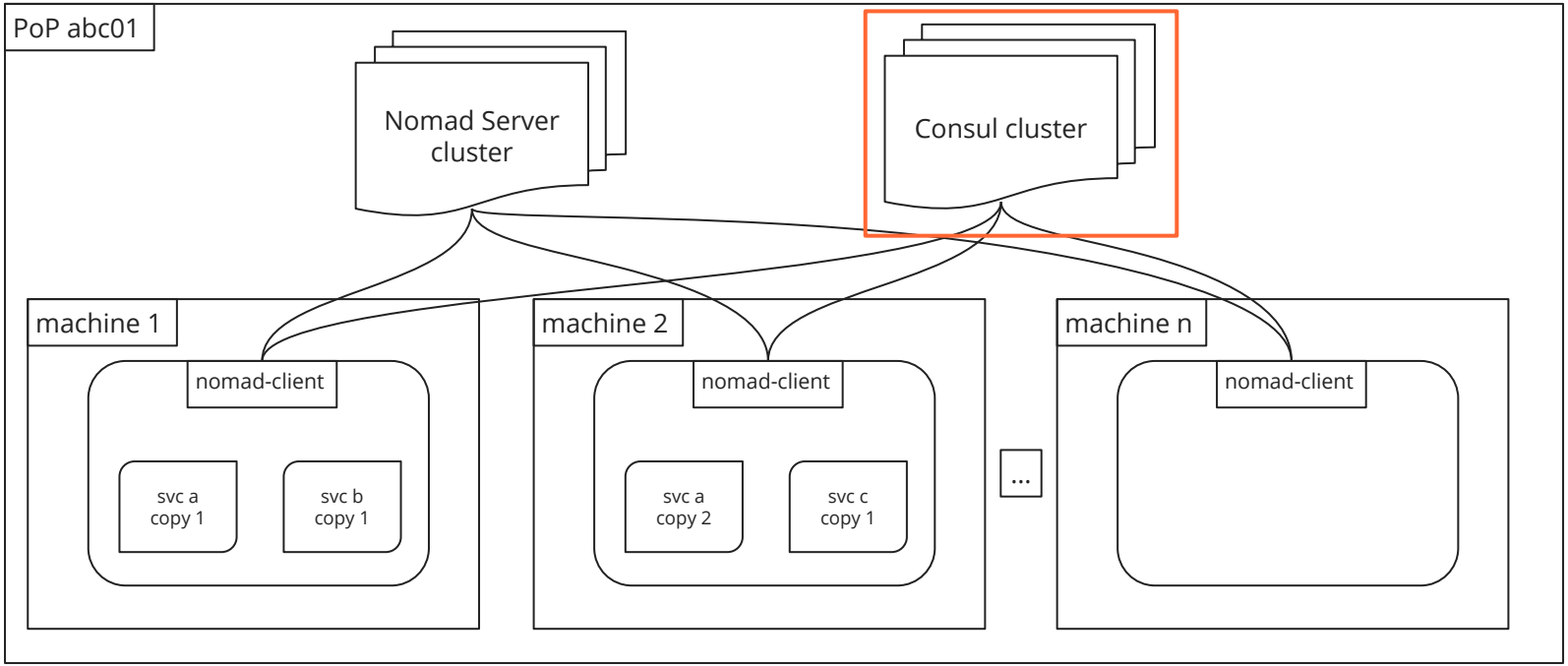
- Lightweight, simple and satisfied our initial requirement:
  - Reliably running a single instance of a management service in every location
- Few dependencies:
  - Consul which we already have deployed

CLOUDFLARE

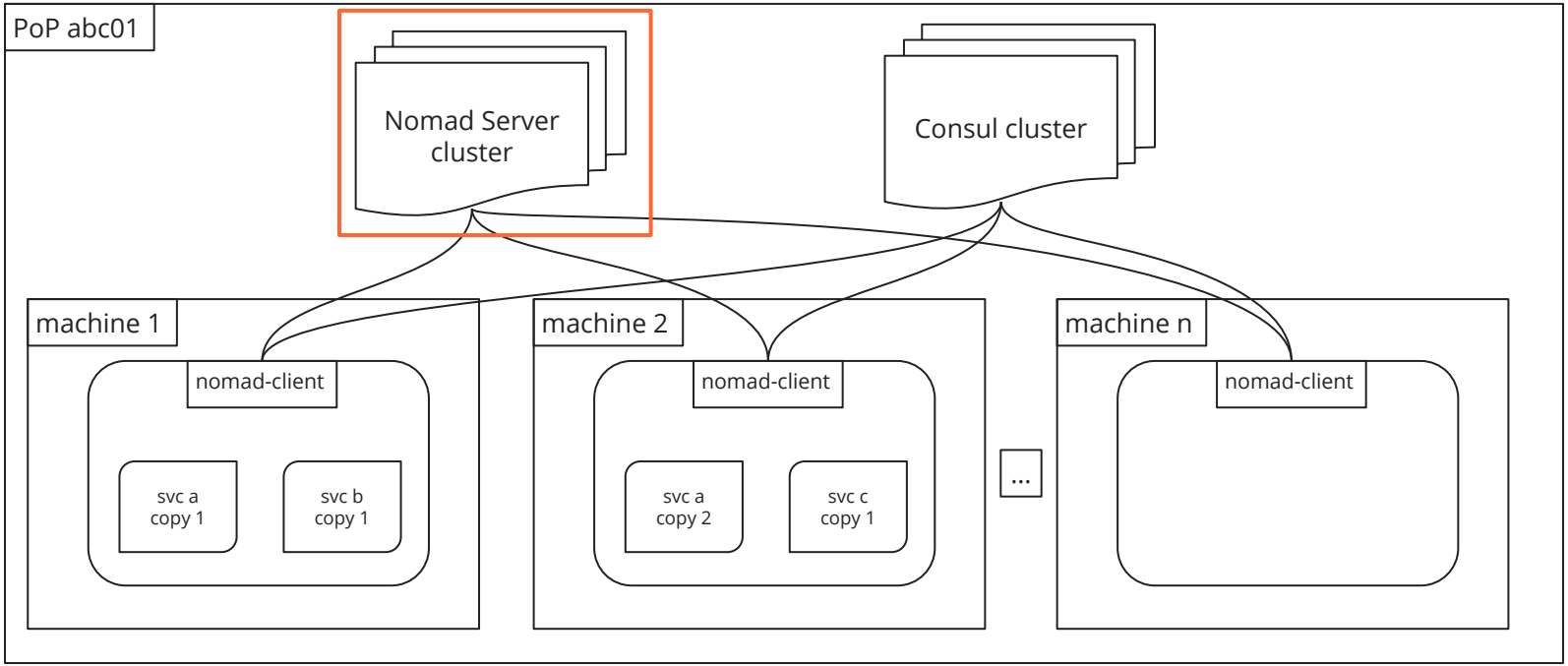
Nomad design and implementation

Architecture and operationalization

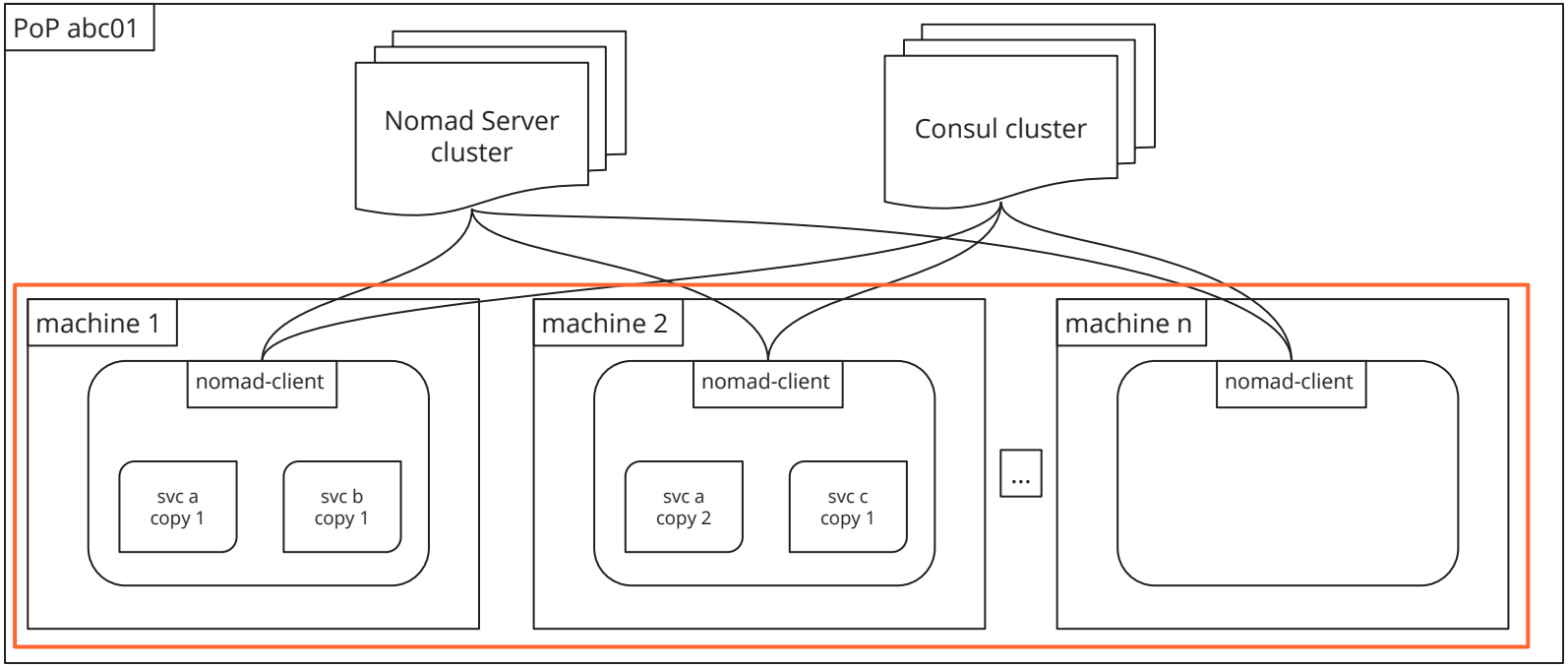
Cluster architecture



Cluster architecture



Cluster architecture



CLOUDFLARE

Nomad Server cluster lifecycle

CLOUDFLARE

Nomad Server cluster bootstrap

Edge machines are stateless, they PXE boot a minimal image

Objective: during new colo provisioning, how to bootstrap a Nomad cluster without manual intervention?

Solution:

- Orchestration already runs and bootstraps Consul cluster in the colo
- Set bootstrap_expect and get Nomad Server to use Consul to automatically cluster nodes

CLOUDFLARE

Graceful Nomad Server upgrades

- Nomad version driven through our orchestration system
- Runs periodically on all machines

Objective: how to achieve zero downtime upgrade when multiple Nomad Server instances can restart at once?

Solution:

- Orchestration holds a lock in Consul during the upgrade process
- leave_on_interrupt option set in Nomad Server configuration to have it leave the cluster gracefully on SIGINT

CLOUDFLARE

Nomad jobs provisioning

CLOUDFLARE

Shared ownership

Layer	Owner
Software running on top of Nomad	Product teams
Interface with Nomad cluster and observability	Edge Platform SRE team
Nomad cluster	Edge Platform SRE team

CLOUDFLARE

Jobs provisioning and management

Currently

Commit	Build	Deploy
Nomad jobs are committed to git repositories, respectively owned by teams	CI builds a .deb package rolled out in stages to the Edge by orchestration	Orchestration also ensures jobs are provisioned in Nomad and are in the correct state

CLOUDFLARE

Jobs provisioning and management

Currently

Commit	**Build**	Deploy
Nomad jobs are committed to git repositories, respectively owned by teams	**CI builds a .deb package rolled out in stages to the Edge by orchestration**	Orchestration also ensures jobs are provisioned in Nomad and are in the correct state

CLOUDFLARE

Jobs provisioning and management

Currently

Commit	Build	**Deploy**
Nomad jobs are committed to git repositories, respectively owned by teams	CI builds a .deb package rolled out in stages to the Edge by orchestration	**Orchestration also ensures jobs are provisioned in Nomad and are in the correct state**

CLOUDFLARE

Jobs provisioning and management

Future

- Looking into CI pushing jobs directly to the Edge perhaps using Terraform

CLOUDFLARE

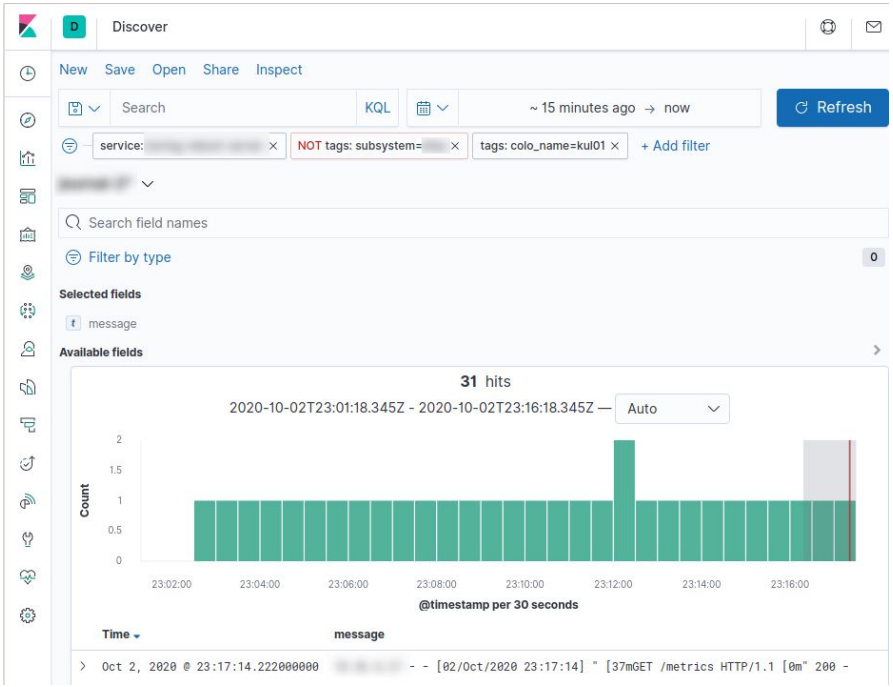
Nomad jobs observability

Logs and metrics

CLOUDFLARE

Logging

1. Jobs write to stdout/stderr
2. Task driver forwards to local syslog
3. Local syslog ships to central syslog and ElasticSearch
4. Logs available in Kibana



CLOUDFLARE

Metrics: Nomad jobs setup

Services defined in Nomad jobs:

- Get automatically added to Consul Directory
- Expose a **enable-prometheus-scraping** tag

```
service {
  name = "maintenance"
  port = "http"
  tags = ["enable-prometheus-scraping"]
}
```

CLOUDFLARE

Metrics: prometheus setup

Prometheus:

- Uses consul_sd_config
- Matches on the **enable-prometheus-scraping**

```
- job_name: nomad_jobs_from_consul_sd
  consul_sd_configs:
    - server: 'localhost:8500'
      tags:
        - enable-prometheus-scraping
  relabel_configs:
    - action: keep
      regex: nomad
      source_labels:
[__meta_consul_service_metadata_external_source]
    - source_labels: [__meta_consul_service]
      target_label: consul_service
    - source_labels: [__meta_consul_node]
      target_label: instance
```

Metrics: dashboards and alerting

Once those service-specific metrics are in Prometheus, each team can:

- Create Grafana dashboards
- Set up alerting based on the metrics their Nomad job exposes

CLOUDFLARE

Conclusion

CLOUDFLARE

Status

- New workloads are being added to Nomad every quarter
  - Legacy ones are being migrated
  - New ones are taking advantage of the dynamic scheduling mechanism in every colo

CLOUDFLARE

Questions?

Thank you