From DevOps to MLOps: Scaling ML models to 2 Million+ requests per day

Chinmay Naik

Chinmay Naik

@chinmay185

🏗️ Founder One2N (Backend and Reliability engineering)

📖 Writes stories on Pragmatic Software Engineering

❤️ Engineering 👨‍💻, Psychology 🧐, Percussion 🥁 and

Age of Empires 2

One2N

Agenda

What is MLOps

MLOps for DevOps practitioners

Real-world production case study walkthrough

What is MLOps

Operationalizing data science

Getting ML models to production

MLOps phases - Build, Manage, Deploy, Monitor

MLOps Steps

Data extraction
Data analysis
Data preparation
Model training
Model evaluation
Model validation

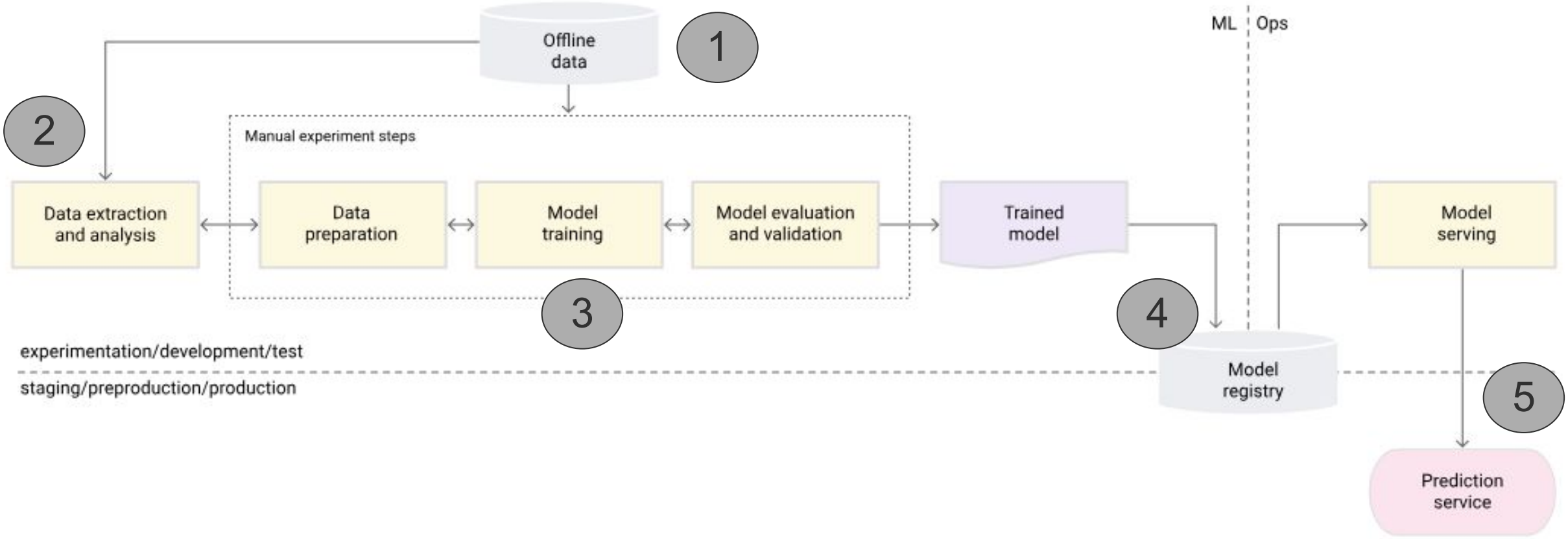
ML

Model serving
Model monitoring

Ops

Feedback loop
CI/CD/CT (Continuous Training)

Simplest MLOps flow

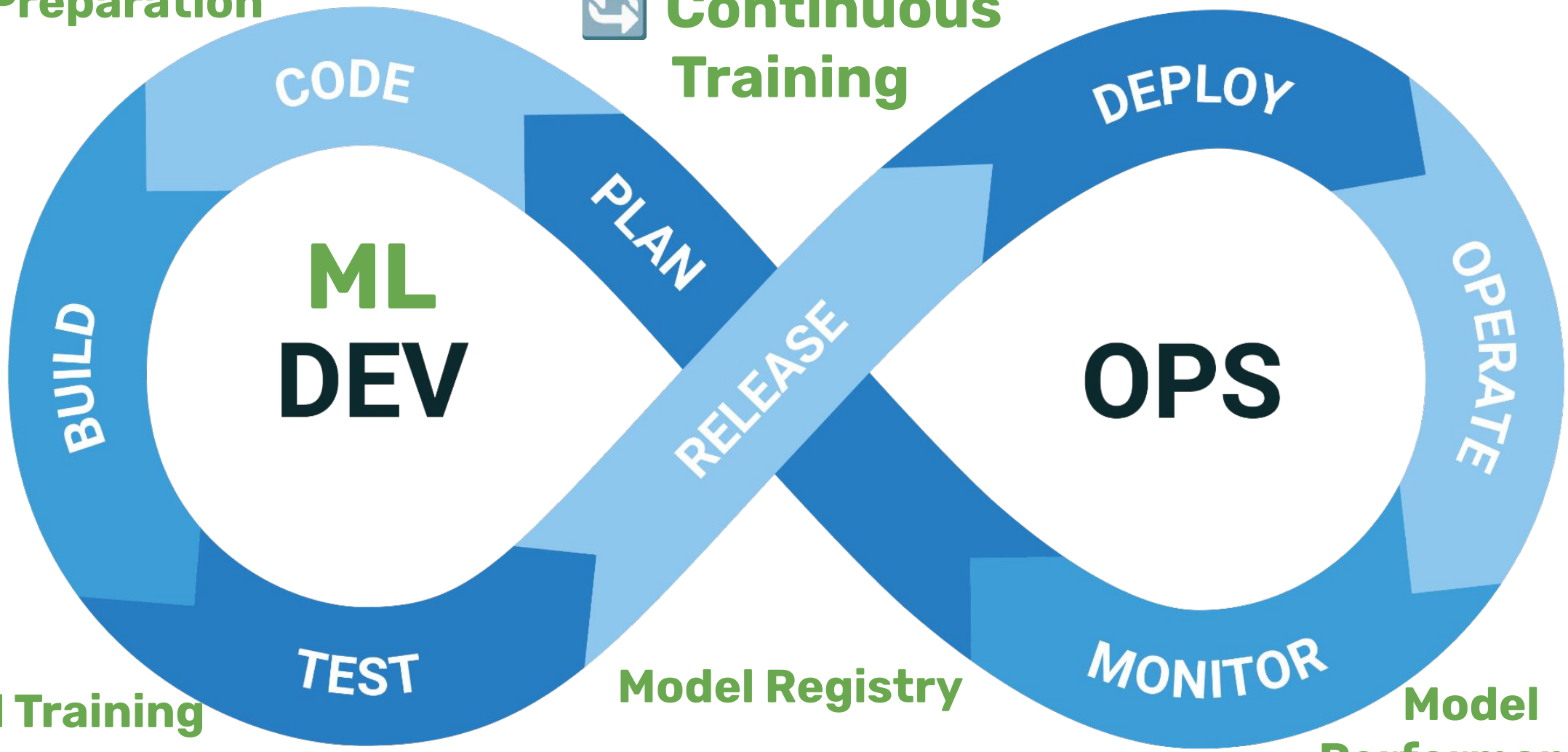


Ref: Google's ML Ops Guide

Data Preparation

Continuous Training

Model Deployment



Model Training

Model Registry

Model Performance

Production work ahead 🚧

One2N

Case study - eKYC SaaS APIs

The company has **eKYC SaaS APIs** accessible via mobile SDKs to its B2B customers. This needed to scale upto **2 Million+ API requests per day**

ML Model APIs

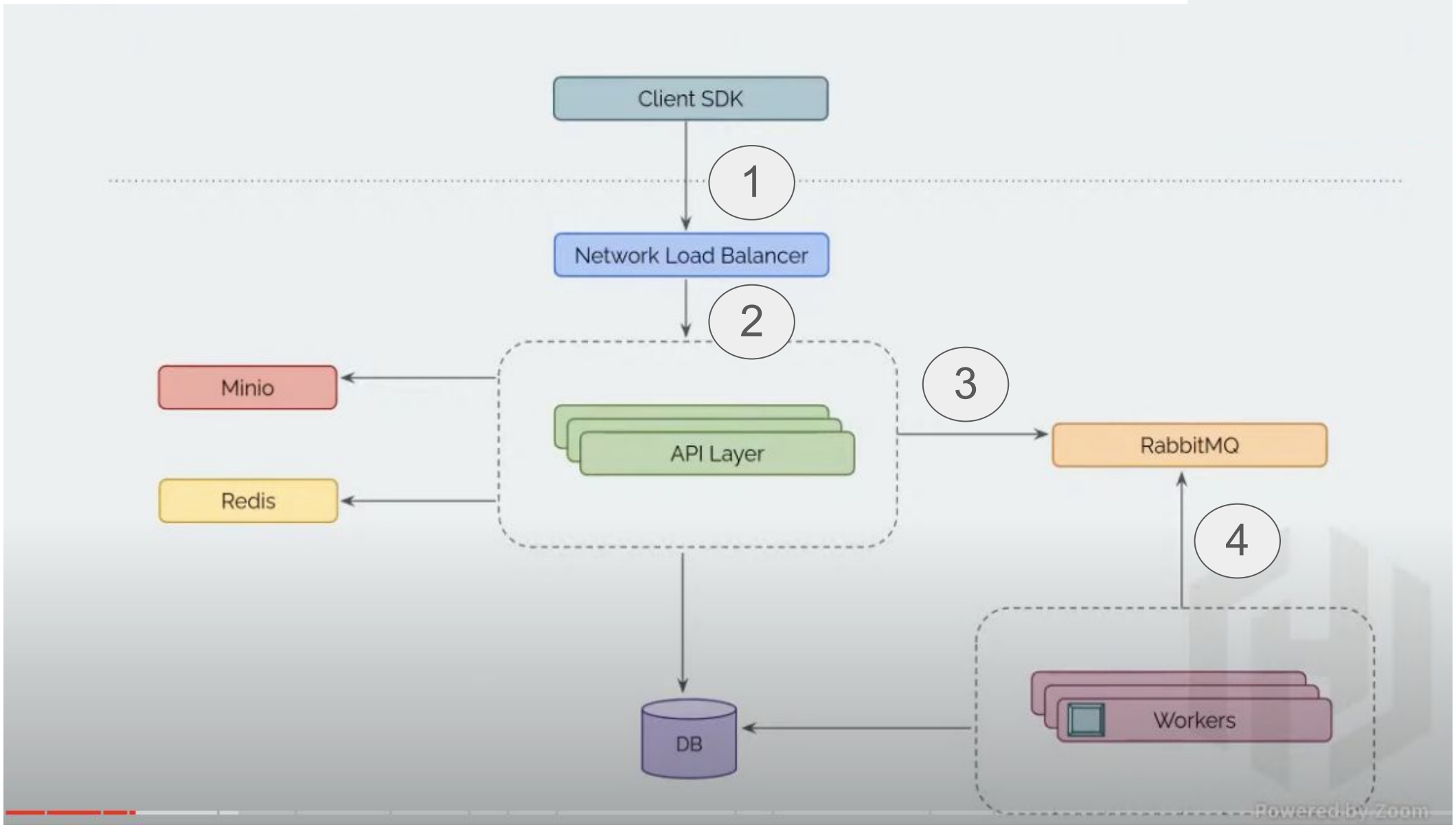
Face matching between images (score: 0 to 1)

Face Liveness detection

Optical character recognition from an image

etc..

Architecture



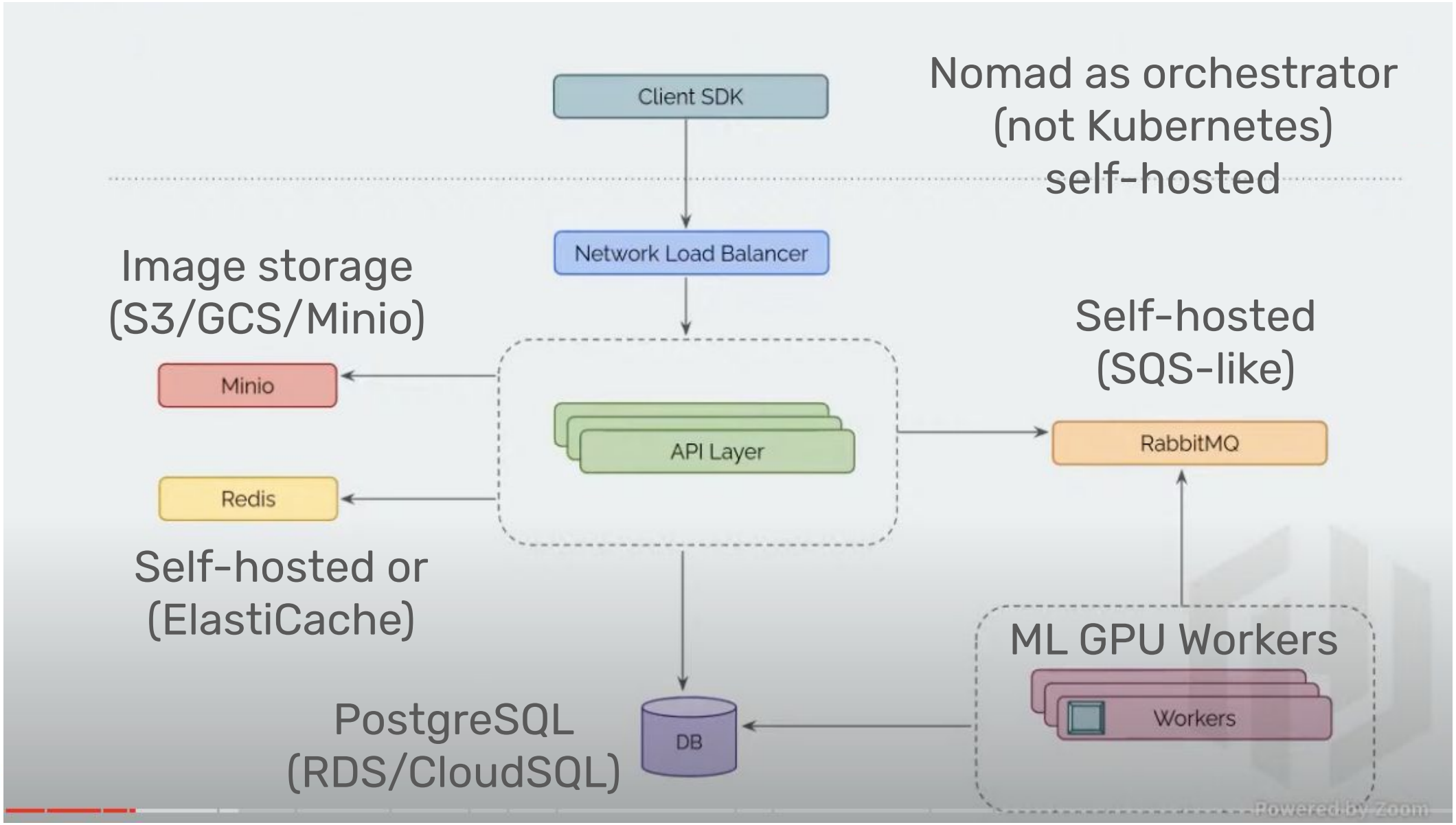
eKYC SaaS APIs - Requirements

99% availability

Cost optimizations

< 3 sec API latency for 95th percentile

Cloud Agnostic Architecture



Why Cloud Agnostic?

One2N

Scaling Journey

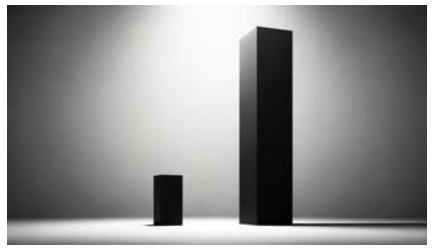
Eliminate single points of failure

Capacity Planning

Cost optimization and Autoscaling

Rolling deployments, Observability, etc

Production incidents

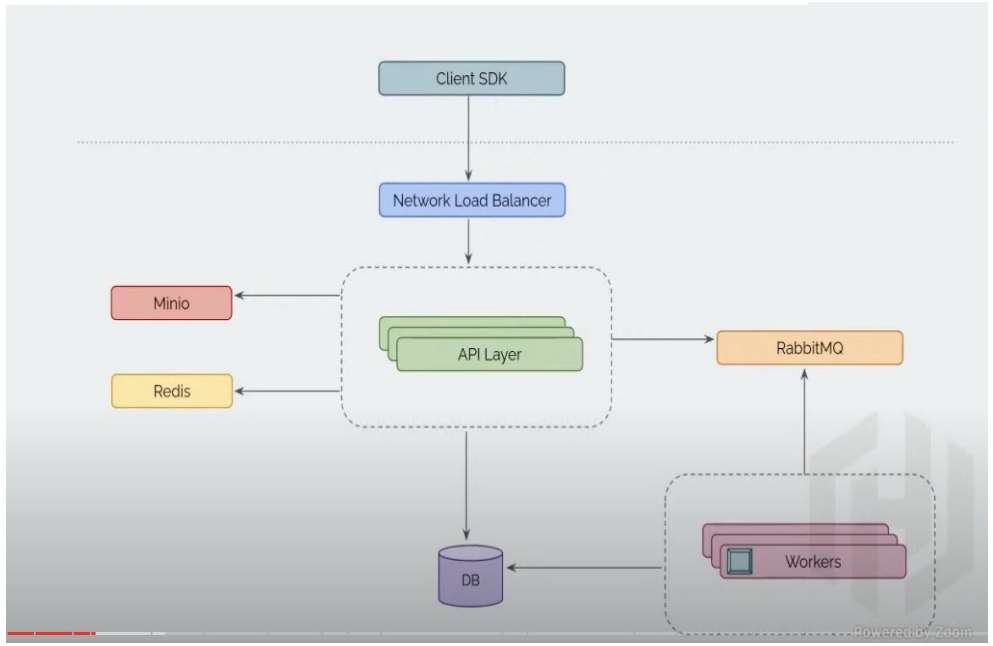


❌

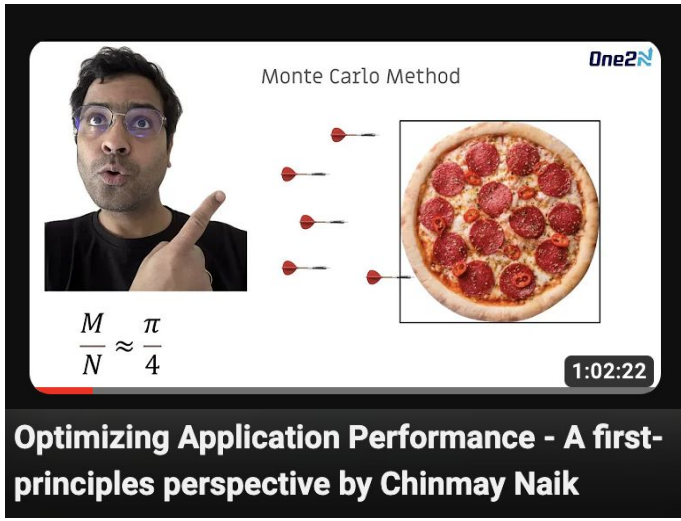
✅

Eliminate single points of failure

- High Availability mode for RabbitMQ
  - Queue replication
  - Cross AZ deployment
  - (alternative was SQS, etc.)
- Split ML workers across AZs
  - Fix Autoscaling
  - Fix Deployment automation
- SaaS offering for Redis and Database
  - Stateless = easy, Stateful = hard



Capacity Planning



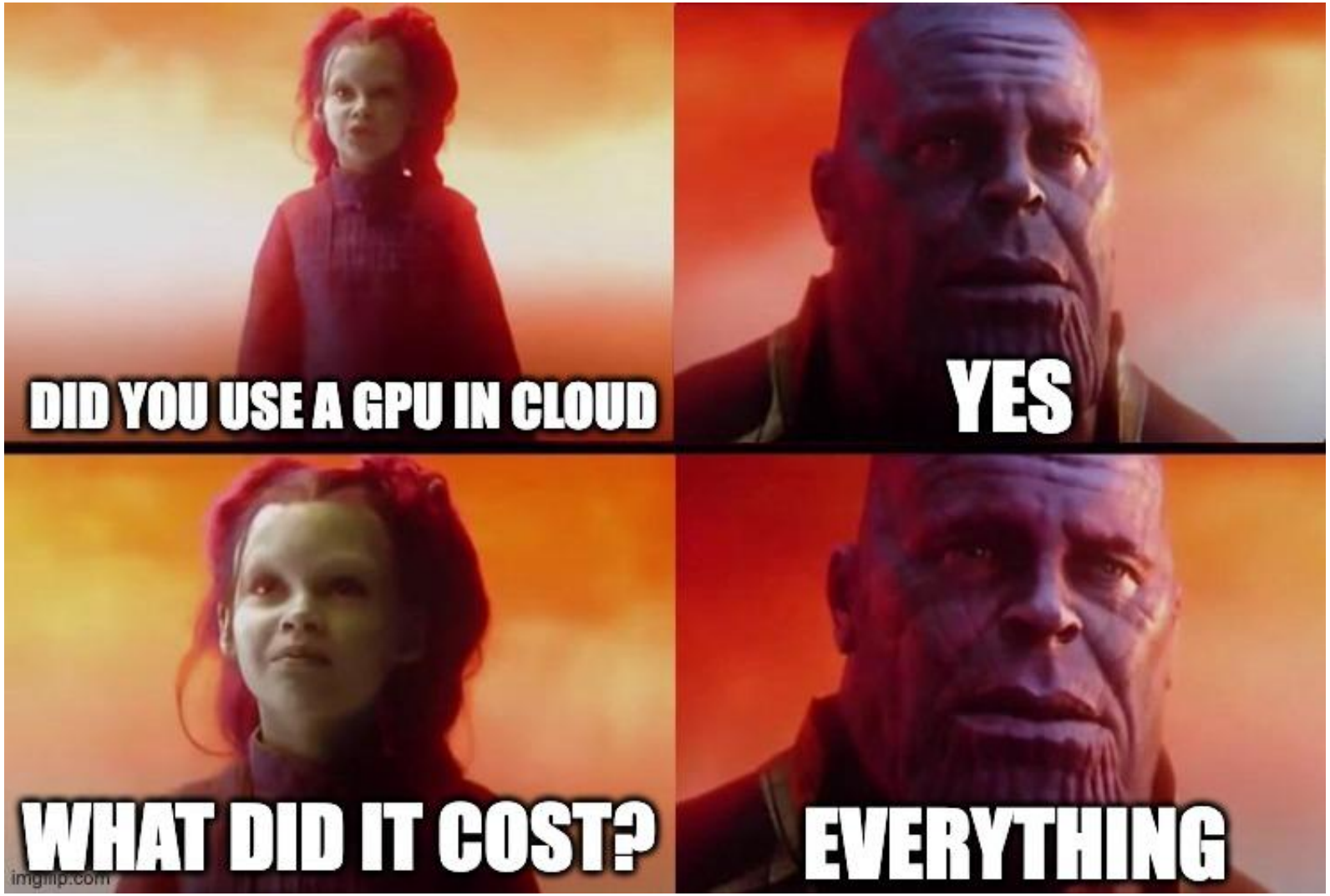
Where's the bottleneck?

- API
- DB (Redis/RabbitMQ/PostgreSQL/Minio)
- ML workers
- Something else?

How many ML model requests can a single node handle?

Talk link: https://www.youtube.com/watch?v=HRKUd0pDh8w

Cost Optimization and Autoscaling



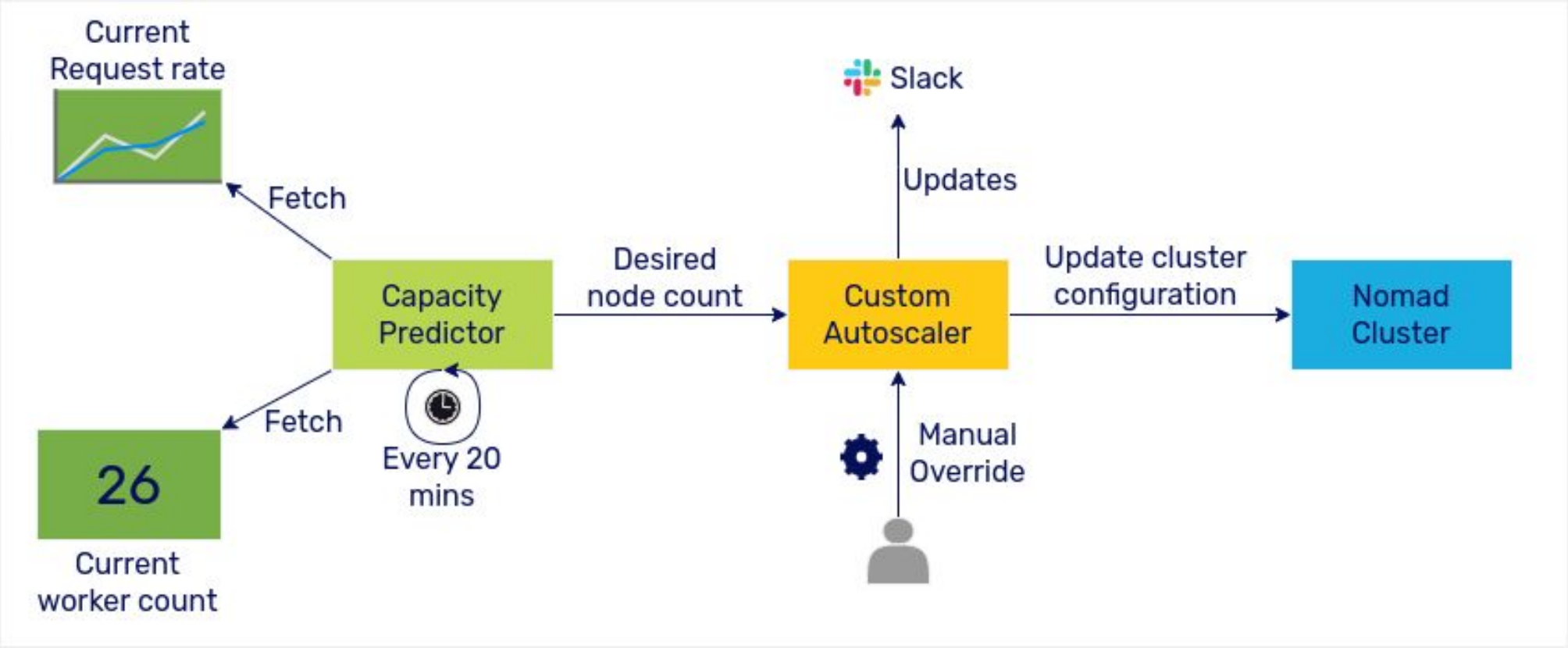
Cost Optimization and Autoscaling

What metric should we autoscale on?

- CPU/GPU utilization
- Memory utilization
- Incoming requests
- Queue depth
- Something else?

Combination of these

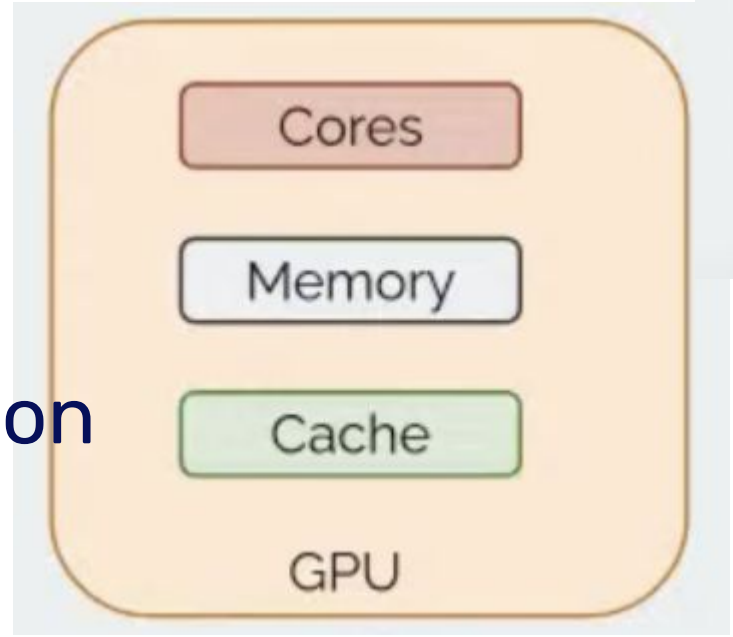
Cost Optimization and Autoscaling



https://one2n.in/case-studies/auto-scaling-machine-learning-ekyc-workloads/

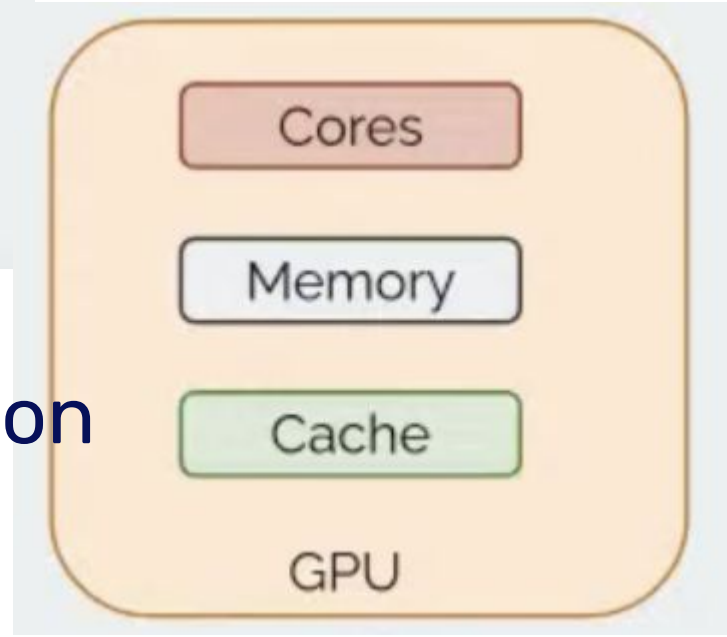
Production Issue 1 - GPU utilization in Nomad

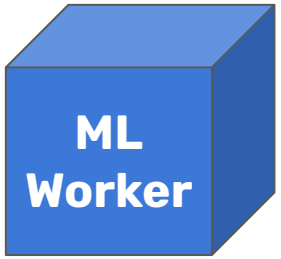
~20% utilization



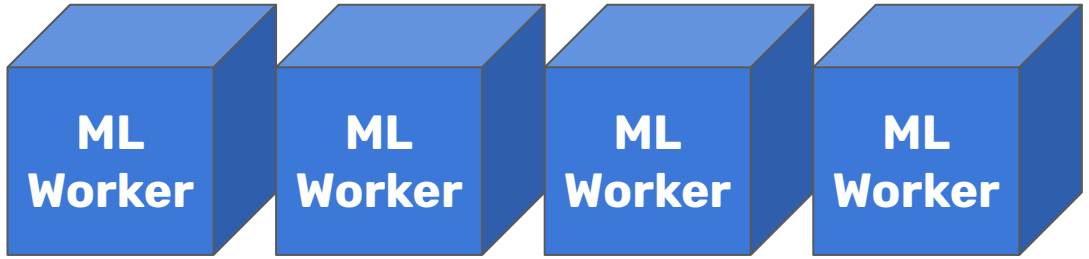
4x $

~80% utilization



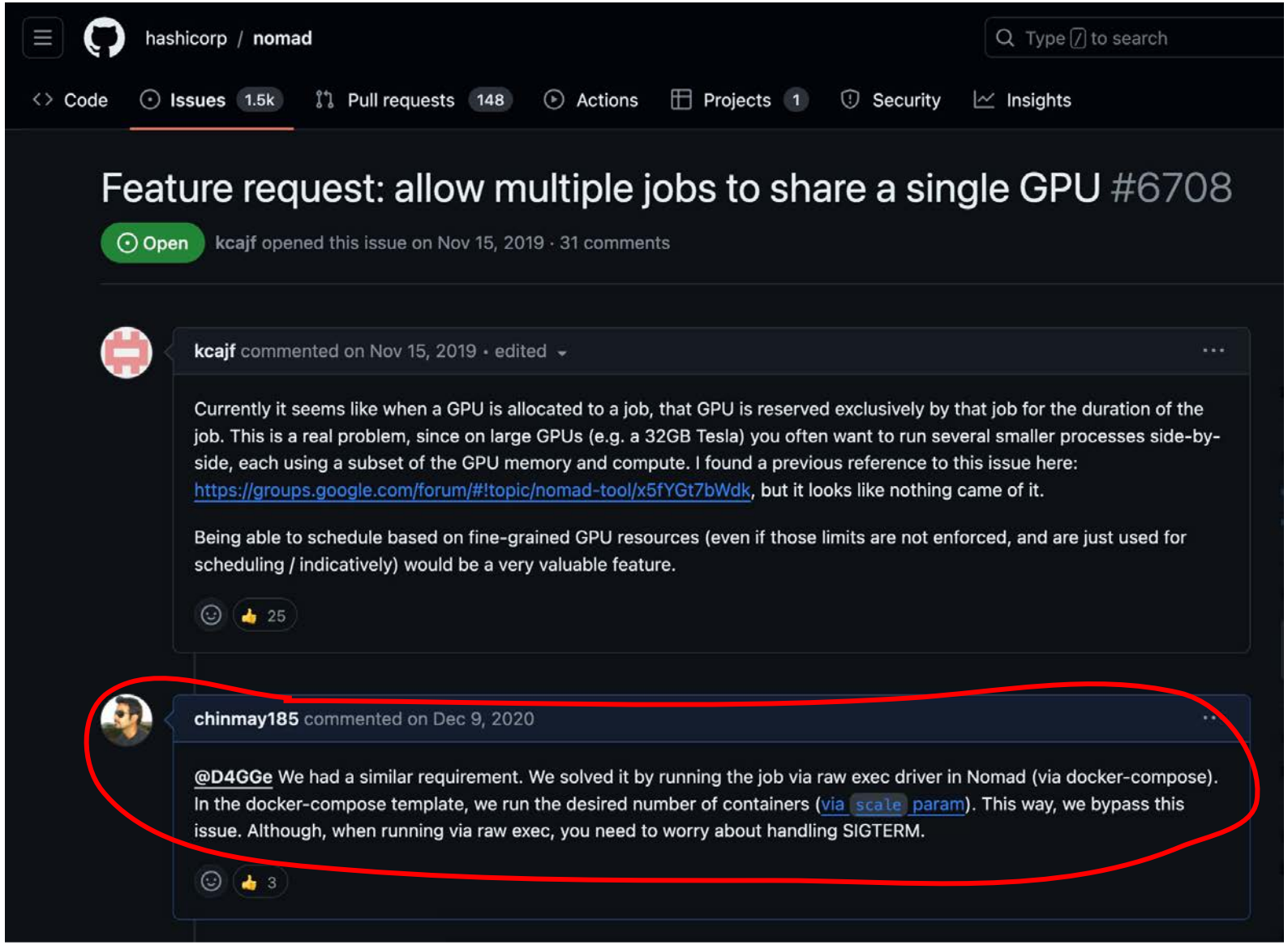


Docker Driver



Raw-exec Driver

Production Issue 1 - GPU utilization in Nomad



hashicorp / nomad

Type / to search

Code | Issues 1.5k | Pull requests 148 | Actions | Projects 1 | Security | Insights

Feature request: allow multiple jobs to share a single GPU #6708

Open kcajf opened this issue on Nov 15, 2019 · 31 comments

kcajf commented on Nov 15, 2019 · edited

Currently it seems like when a GPU is allocated to a job, that GPU is reserved exclusively by that job for the duration of the job. This is a real problem, since on large GPUs (e.g. a 32GB Tesla) you often want to run several smaller processes side-by-side, each using a subset of the GPU memory and compute. I found a previous reference to this issue here: https://groups.google.com/forum/#!topic/nomad-tool/x5fYGt7bWdk, but it looks like nothing came of it.

Being able to schedule based on fine-grained GPU resources (even if those limits are not enforced, and are just used for scheduling / indicatively) would be a very valuable feature.

25

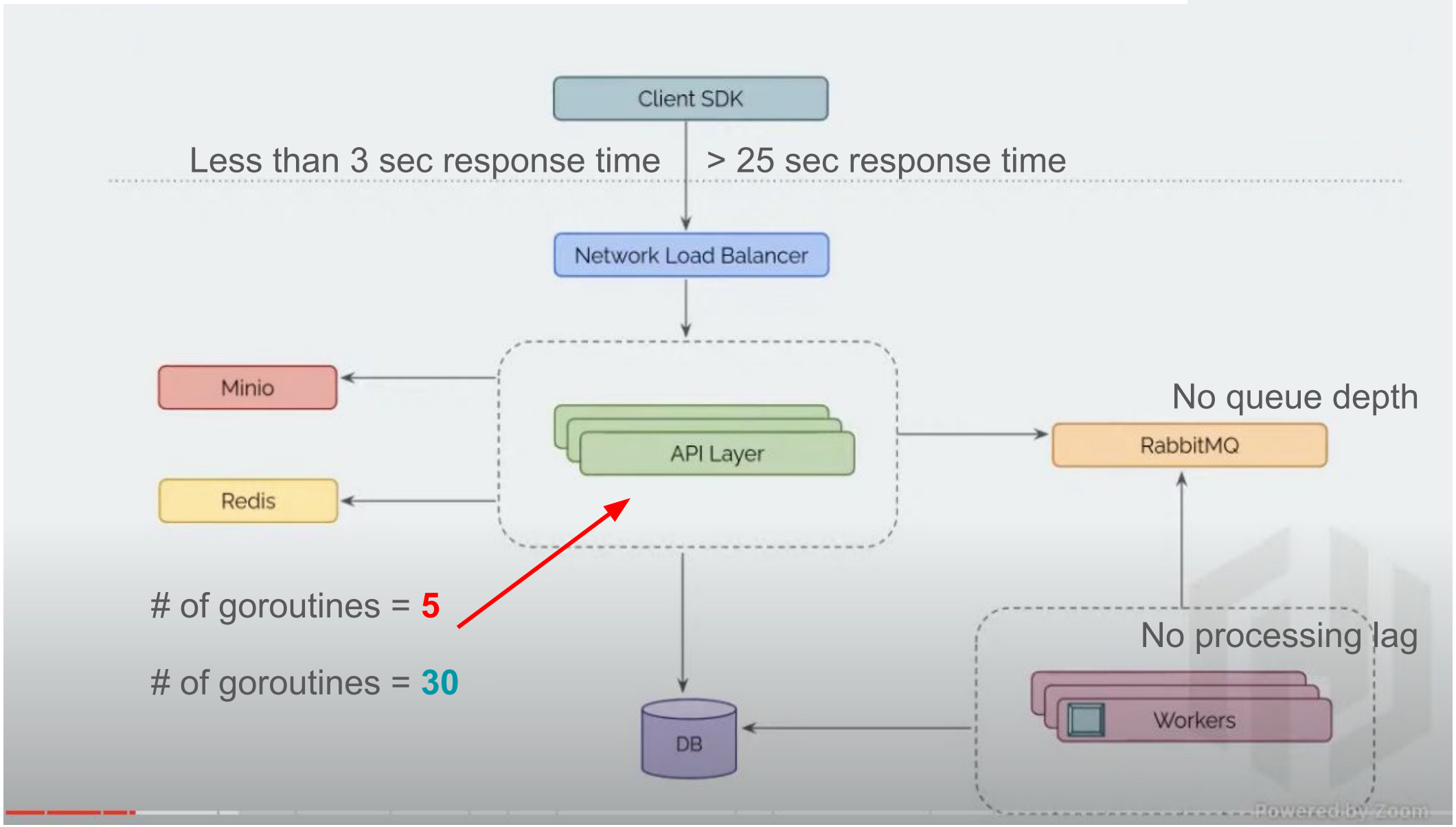
chinmay185 commented on Dec 9, 2020

@D4GGe We had a similar requirement. We solved it by running the job via raw exec driver in Nomad (via docker-compose). In the docker-compose template, we run the desired number of containers (via `scale` param). This way, we bypass this issue. Although, when running via raw exec, you need to worry about handling SIGTERM.

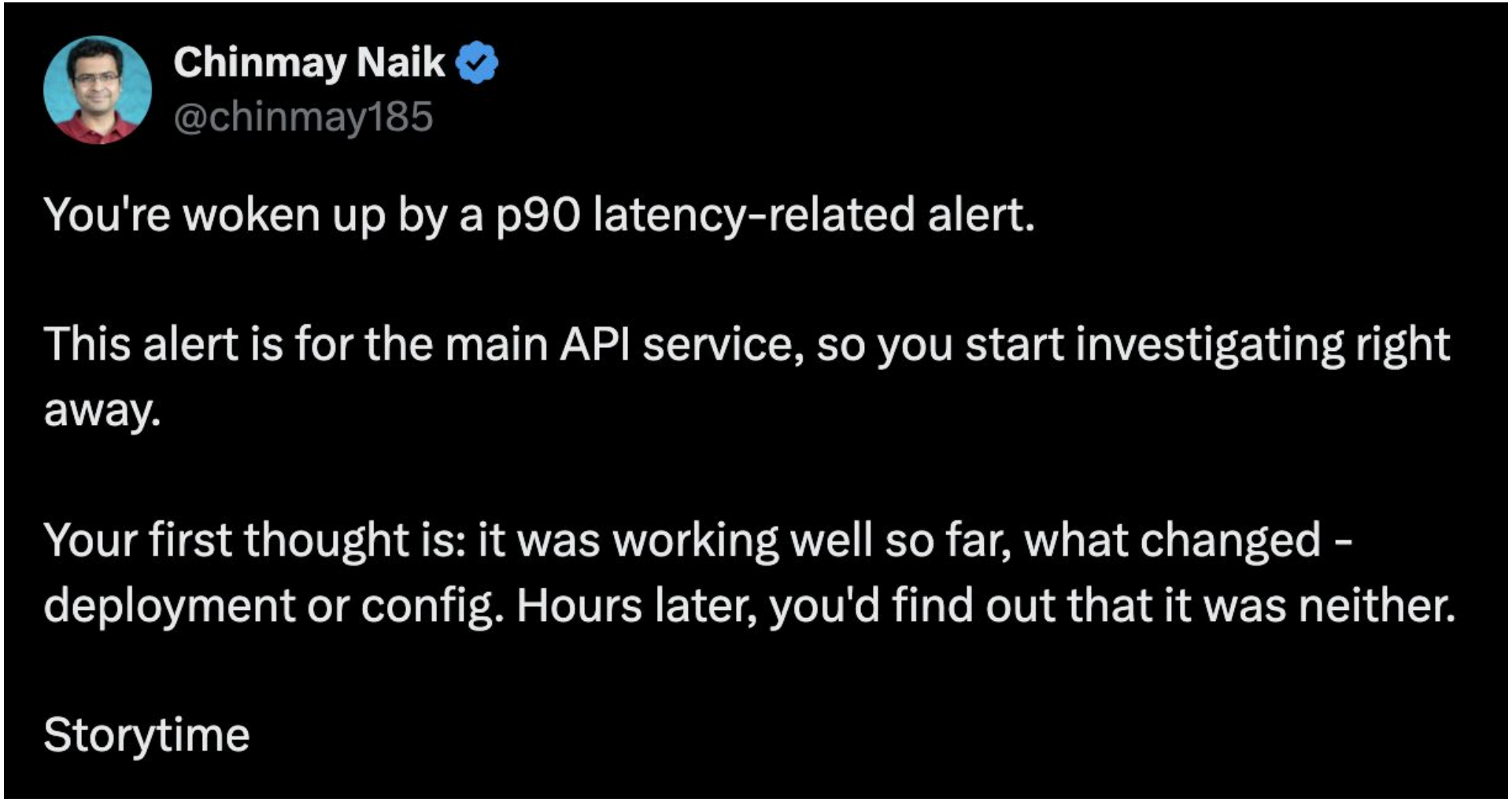
3

Issue: https://github.com/hashicorp/nomad/issues/6708

Production Issue 2 - High latency issue



Production Issue 2 - High latency issue



Story link (Twitter): https://twitter.com/chinmay185/status/1729911773405352030

Lessons

- Data quality and training is super important
- Cloud agnostic architecture FTW
- Treat Operations-work as first class citizen
- Ensure close collaboration across Data Scientist, Backend engineers, Business teams and SREs

Keep learning

Connect with me @chinmay185 on Twitter, LinkedIn, etc.

Check out Go and SRE bootcamps along with the Pragmatic Software Engineering stories

https://playbook.one2n.in