JEUDI, 30 MARS 2017 18:33

Building the Neo4j Sandbox: AWS, ECS, Docker, Python, Neo4j, ++

Ryan Boyd, 30.03.2017

http://ptat.ch/neo4j-sandbox-tech-overview *http://ptat.ch/neo4j-sandbox-tech-overview*
https://www.youtube.com/watch?v=2XbNhAJ9wh0 *https://www.youtube.com/watch?v=2XbNhAJ9wh0*

Goals

- Fast on-boarding experience – to have good conversion rates
- Data that users care about
- Guided experience thru that data and the Neo4j product
- Fast time to first line of code
- Isolated environment – no impact when mutating data on other users

Key stakeholders

- End-users
- Neo4j CEO – wants everyone to use the product
- Marketing – wants to be able to reach the users (send messages about events for instance, or new edition of the O'Reilly book, etc.)
- Developer relations: half-engineering, marketing – wants to be able to reach users also (get their feedback, go to events and build the community)

Risks

« 130 people launching a sandbox in the last hour »

- Costs – *fixed costs* for capacity (machines ready to serve users), *variables costs* for demand (more machines with demands)
- Low data quality: not asking for name, company, country, state, biz e-mail – but OAuth2-provided and derived data, much better quality
- Lower user «stickiness» vs downloads: encourage downloads
- Operational nightmares – server maintenance and security

(patches, outages, memory/CPU management), scaling challenges (spikes in load, drops in load): we're only a handful of persons (1-3 persons)

Techology stack

- Docker Images – provides the desired CPU allocation, as well as memory and storage isolation
- Amazon EC2 Container Services (Amazon ECS) – every Neo4j sandbox is a task on the ECS, that has different docker images in it
- AWS Lambda – serverless computing: similar to Google AppEngine; scales indefinitely, while allowing to use any library (a restriction Google AppEngine had), such as Python library with compiled C packages; has support for launching Docker images
- Neo4j Server
- Python – every Lambda function is written in Python
- Auth0 and JWT (Jason Web Tokens) – to authenticate on the server, a cryptographically signed set of data, without calling external APIs
- Bootstrap and jQuery for the front-end
- Data providers, namely to retrieve a validated e-mail address:
  - Google, LinkedIn, GitHub, Twitter provide user data
  - sent to FullContact, to lookup by e-mail for complements;
  - also using MaxMind, to lookup by IP the country and state the user comes from

Features

- Social Sign-in: launched with Twitter, GitHub, Google; LinkedIn later
- Code snippets
- Hidden Sandboxes
- Streaming Twitter Sandbox (~ 36min.)
- Nginx proxy for SSL – using high ports, for those corporate firewalls
- Single Sign-on to *Neo4j Browser* app: a custom plug-in validates a JWT (JSON auth-token) sent by Auth0
- E-mail Nurturing

Architecture

voir le diagramme vers ~ 39min.

- S3 Static hosting + CloudFlare CDN
- Amazon EC2 Container Services (ECS) machines, running 16 Docker Neo4j images – among them 3 Nginx proxy, distributed on 3 of the machines
- Amazon Lambda functions for provisioning
- Amazon EC2 Autoscaling Group for autoscaling
- Amazon Elastic Load Balancers (ELB)
- NGINX proxy for SSL

Lessons learned

- Quotas :-(
- Balancing load spikes smartly
- CSS Hell – namespacing especially, using tooling
- Coding is rewarding

What's next

- More use cases: BYO (bring-your-own data), data journalism, API data (GitHub data), enterprise use cases (fraud detection)
- Sharing with colleagues and friends – JWT and Auth0 challenge

Q&A

- How long to build? 6-9 months on part time, for infrastructure side
- How many people to operate? 1 person, almost zero issue
- How many people to build? 1 person, the presenter
- How much memory and CPU? 16MB for each sandbox; 768MB for each machine; ¼ CPU-core per sandbox, being allowed to spike up to how much it needs.

See also

- Neo4j Docker Image *http://hub.docker.com/_/neo4j*
- Neo4j GraphGists *https://neo4j.com/graphgists/* *teaching tools which allow you to explore how data in a particular domain would be modeled as a graph and see some example queries of that graph data*

PLANÈTE CHARMILLES, GENÈVE, GENÈVE, SUISSE • 20° MOSTLY SUNNY
