





# How we run GraphQL APIs in production on our (own) Kubernetes cluster



numberly  
1000mercis group

# @ultrabug

Gentoo Linux developer  
PSF contributing member  
CTO at Numberly



Couldn't you have more buzz words in your talk title?







# Previous workflow and its limitations



# GitLab

Code repositories  
Configuration repositories  
Continuous Integration  
Code reviews





# GitLab

Code repositories  
Configuration repositories  
Continuous Integration  
Code reviews



YAML configuration file

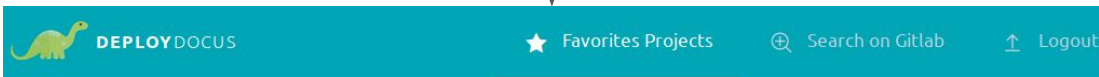


# GitLab

Code repositories  
Configuration repositories  
Continuous Integration  
Code reviews



YAML configuration file



Name	Path	Last Activity	Actions
> ansible	app-admin/ansible	an hour ago	
> trello_cleaner	app-admin/trello_cleaner	a day ago	



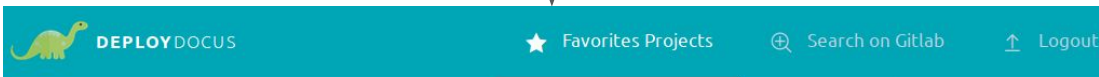


# GitLab

Code repositories  
Configuration repositories  
Continuous Integration  
Code reviews



YAML configuration file



Name	Path	Last Activity	Actions
> ansible	app-admin/ansible	an hour ago	
> trello_cleaner	app-admin/trello_cleaner	a day ago	



# GitLab

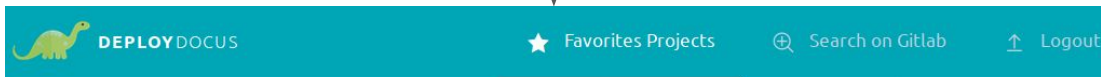
Code repositories  
Configuration repositories  
Continuous Integration  
Code reviews



YAML configuration file



ansible



Name	Path	Last Activity	Actions
> ansible	app-admin/ansible	an hour ago	
> trello_cleaner	app-admin/trello_cleaner	a day ago	



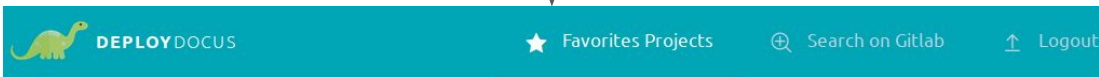


# GitLab

Code repositories  
Configuration repositories  
Continuous Integration  
Code reviews



YAML configuration file



Name	Path	Last Activity	Actions
> ansible	app-admin/ansible	an hour ago	
> trello_cleaner	app-admin/trello_cleaner	a day ago	



ansible



SSL offloading






# GitLab

Code repositories  
Configuration repositories  
Continuous Integration  
Code reviews



YAML configuration file

DEPLOYDOCUS

★ Favorites Projects

🔍 Search on Gitlab

🔗 Logout

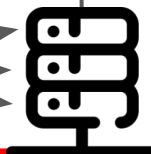
Name	Path	Last Activity	Actions
> ansible	app-admin/ansible	an hour ago	<div><div></div><div></div><div></div><div></div></div>
> trello_cleaner	app-admin/trello_cleaner	a day ago	<div><div></div><div></div><div></div><div></div></div>



ansible

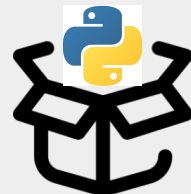


SSL offloading



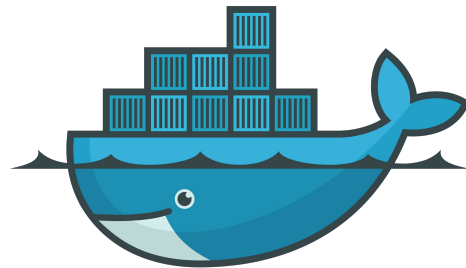
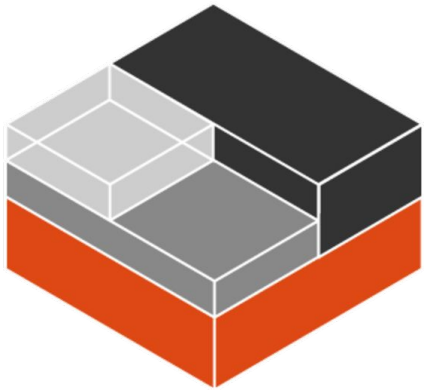
NGINX

uWSGI

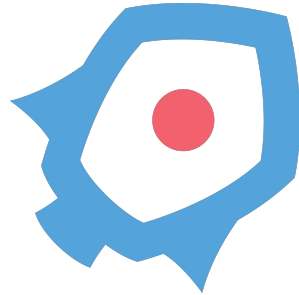




# Why Kubernetes?



docker





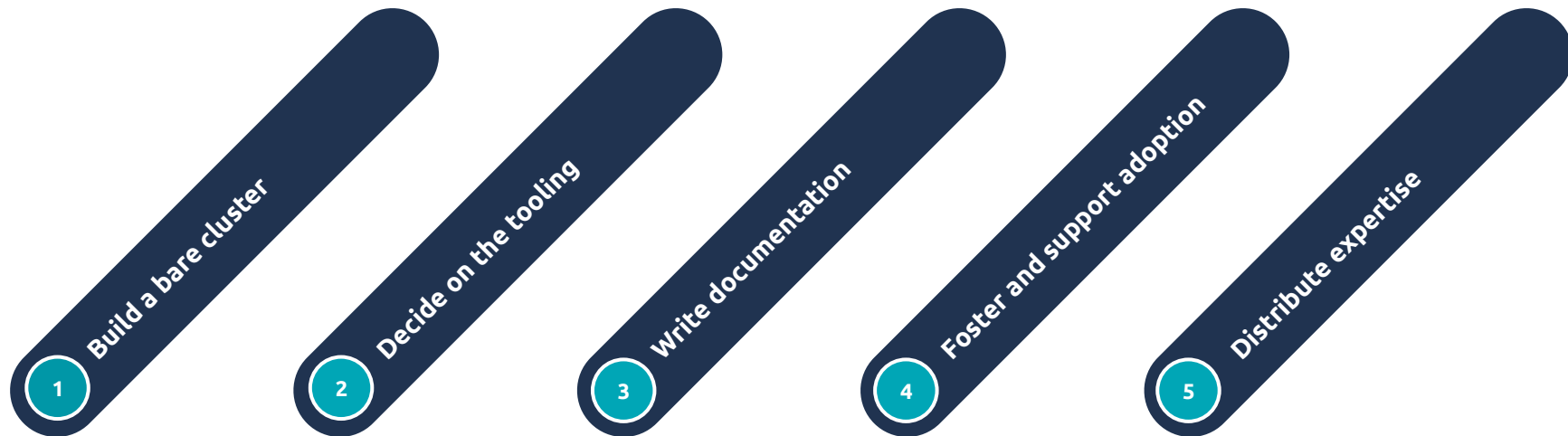
our own bare-metal Kubernetes cluster





# Methodology

---



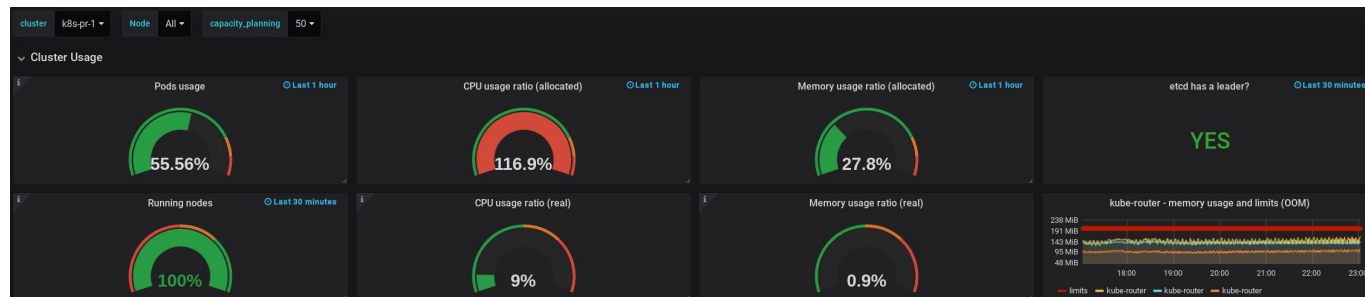
# A bare-metal Kubernetes cluster?



- Package it to deeply know what's it's made of and how it works



- Automate installation, configuration, provisioning... everything!





Developer-driven



# OpenID authentication

- **Developer goes to internal kubeconfig URL**
- **Login using usual Google Suite account (openID) + free MFA (Yubikey)**
- **Download Kubeconfig**
- **Welcome to Kubernetes!**

gangway

Logout

Welcome 113115567274234729608.

In order to get command-line access to the k8s-pr-1 Kubernetes cluster, you will need to configure OpenID Connect (OIDC) authentication for your client.

The Kubernetes command-line utility, kubectl, may be installed like so:

```
$ curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt`/bin/${uname}
$ chmod +x ./kubectl
$ sudo mv ./kubectl /usr/local/bin/kubectl
```

DOWNLOAD KUBECONFIG

Once kubectl is installed, you may execute the following:

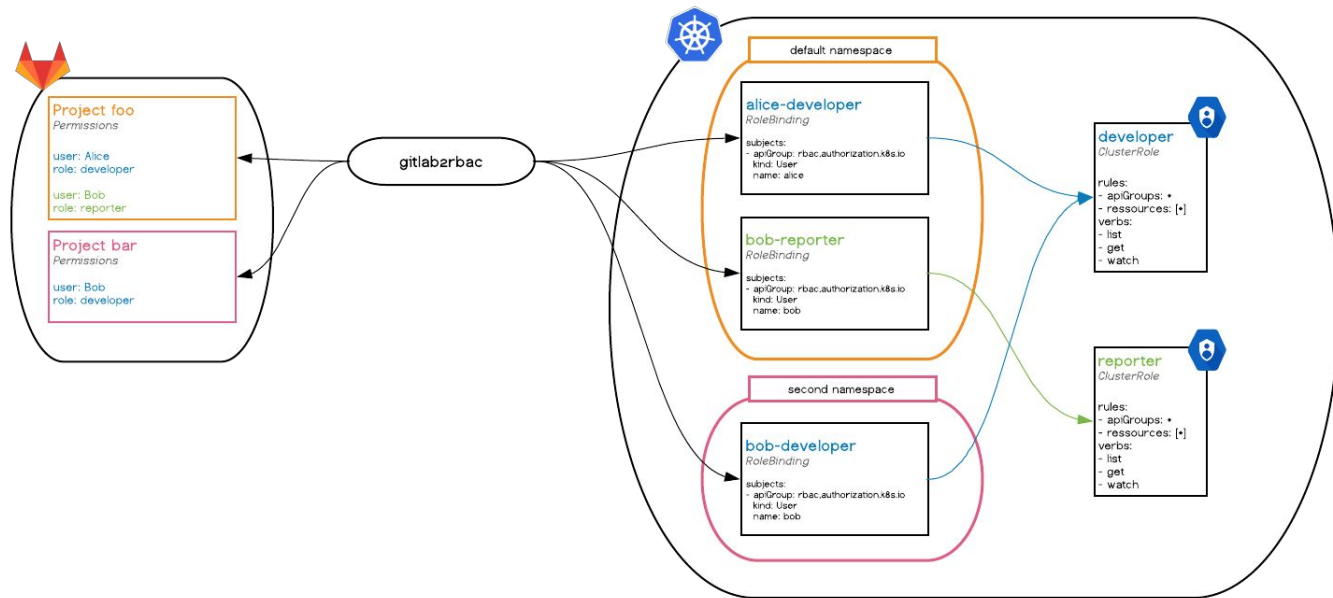
```
echo "-----BEGIN CERTIFICATE-----
MIIF2DCCAB8CgAwIBAgIURYS08NUPk3kH8mJyFpCoI8090swDQYJKoZIhvcNAQEN
BQAwcJELMAKGA1UEBhMCRLlxdjAMBGNVBAcTBVBhcnZMRQwEgYDVQQKEwtrOHMT
cHItMS1jMTENMCUGA1UECgMeQ2x1c3RlcjBrdWJlc5ldGVzIGs4cy1wc10xLWMx
```





# Gitlab based authorization

- **Gitlab based RBAC + Pod Security Policy since day 1**
  - 1 namespace = 1 team
- Open sourced **gitlab2rbac**: <https://github.com/numberly/gitlab2rbac>



# Cluster capabilities and choices

- **Gitlab registry for our Docker containers**
  - Ensure only whitelisted images can be deployed
- **runAsNonRoot + strict Network Policies enforced**
- **Ingress using nginx-ingress with fully automated LetsEncrypt certificate lifecycle**
- **Multi-tenant cluster supporting all environments (production, staging, development)**
- **Special “sandbox” namespace to test things:**
- **No distributed persistent storage yet**



**CIS** Center for Internet Security®



```
1 apiVersion: batch/v1beta1
2 kind: CronJob
3 metadata:
4   name: sandbox-cleaner
5   namespace: kube-system
6   labels:
7     managed-by: ansible
8 spec:
9   schedule: "0 0 * * * # every day at 00:00
10  successfulJobsHistoryLimit: 1
11  failedJobsHistoryLimit: 1
12  jobTemplate:
13    spec:
14      template:
15        spec:
16          securityContext:
17            runAsUser: 65534 # nobody
18            serviceAccount: sandbox-cleaner
19          containers:
20            - name: sandbox-cleaner
21              image: roffe/kubectl
22              args:
23                - -c
24                - >
25                - kubectl version
26                - && kubectl api-resources --verbs="delete" --namespaced -o name
27                - | grep -v role
28                - | grep -v limitrange
29                - | xargs -n1 kubectl -n sandbox delete --all
30          restartPolicy: OnFailure
```

# A workflow-oriented documentation

---

## Labs

---

To help you with Kubernetes at Numberly, here are a few sheets about various subjects, from its basic usage to more advanced topics.

Kubernetes features are very broad; these labs will help you to find your way around, but are not meant to cover the whole subject and will not replace either [Kubernetes'](#) or [Docker's documentation](#).

You can either go across them one by one, or pick the ones that concern you when you need them. If you do not know what Kubernetes and Docker are, it is recommended that you follow the first three ones, at least.

### Beginner

- [Hello Kubernetes!](#) (~15 min.)
- [Hello Docker!](#) (~15 min.)
- [Orchestrate your first application](#) (~30 min.)
- [Et voilà, on a les droits!](#) (~15 min.)

### Intermediate

- [Release the Kraken! \(part 1\)](#) (~20 min.)
- [Release the Kraken! \(part 2\)](#) (~30 min.)

### Advanced

- [Few ways \(not\) to get hacked](#) (~10 min.)
- [Best practices](#) (~N min.)
- [Schedule me](#) (~10 min.)

## FAQ

---

- [My Python container does not log anything, what can I do?](#)
- [Migrate your code to the new GitLab](#)
- [How to use Kerberos?](#)
- [How to `docker login` with 2FA enabled on GitLab?](#)
- What about...
  - `auto_conf`?
  - [Deploydocus \(a.k.a Koufar\)](#)?

## Cheatsheets

---

Don't remember how to do that particular one thing that is very useful? Check our cheatsheets:

- [Example Dockerfiles](#)
- [Example Kubernetes manifest](#)
- [Useful annotations](#)
- [Useful `kubectl` commands](#)

# Foster and scale Kubernetes adoption

---

## We created an internal Kubernetes Certification

- To make sure that **in every team someone can help** with Kubernetes
- To help everyone **identify who can support** them when they need a Kubernetes expert
- To **value the expertise** of members of our teams

### Certification

---

#### Why

---

This certification exists to make sure that in every team, someone can help with Kubernetes. Once they pass the assessment, those persons are clearly identified so that everyone knows who to ask for help.

#### Details

---

There are 30 questions, half about Docker and the other half about Kubernetes. Each question is one point worth, and the point is only granted if all correct answers are ticked. A minimum score of 15 is required to obtain the certification.

- Time limit: 20 minutes
- Authorized resources: Docker and Kubernetes documentations, a shell

Tke Away



# T<sub>ke</sub> Away

- **Gitlab** for **RBAC** and image registry + **Kubernetes** = **gitlab2rbac**
- **Balance security vs freedom: not opposed all the time!**
- **Enforce security and QA rules from the start**
  - TODO: work on admission controller to enforce whitelisted images only
- **Ops concentrate on features that are immediately available to all devs**
  - TODO: automate F5 ingress SSL setup for public services
- **Practical and useful docs are key**
- **Spread expertise to foster and scale adoption**
  - TODO: create more certification levels

ur Kubernetes workflow



# GitLab

Code repositories  
~~Configuration repositories~~  
Continuous Integration  
Code reviews  
**Users roles = k8s RBAC**  
**Groups = k8s namespaces**  
**Docker image registry**

Moved to k8s secrets



YAML kubernetes  
deployment

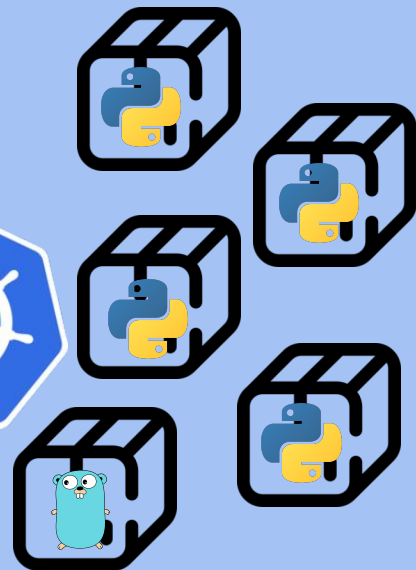
Needs Dockerfile

```
1 $ kubectl apply -f deployments/development.yaml
2 $ kubectl get pods
3 NAME
4 trello-to-graphql-development-bdcf5d96c-jwbk6
```

READY	STATUS	RESTARTS	AGE
1/1	Running	0	16s



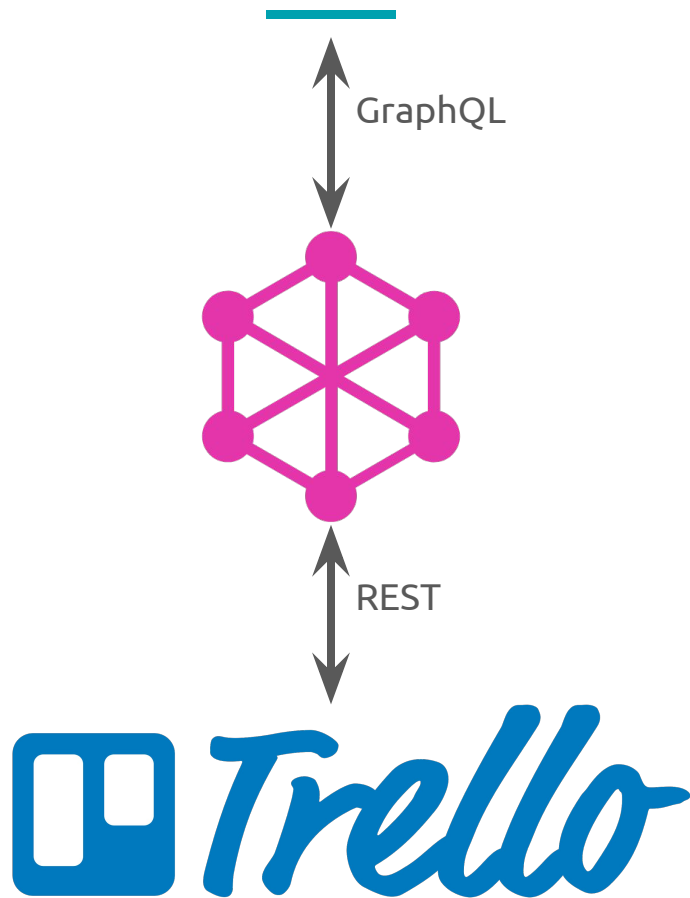
ingress-**NGINX**



Let's build a GraphQL app on Kubernetes!

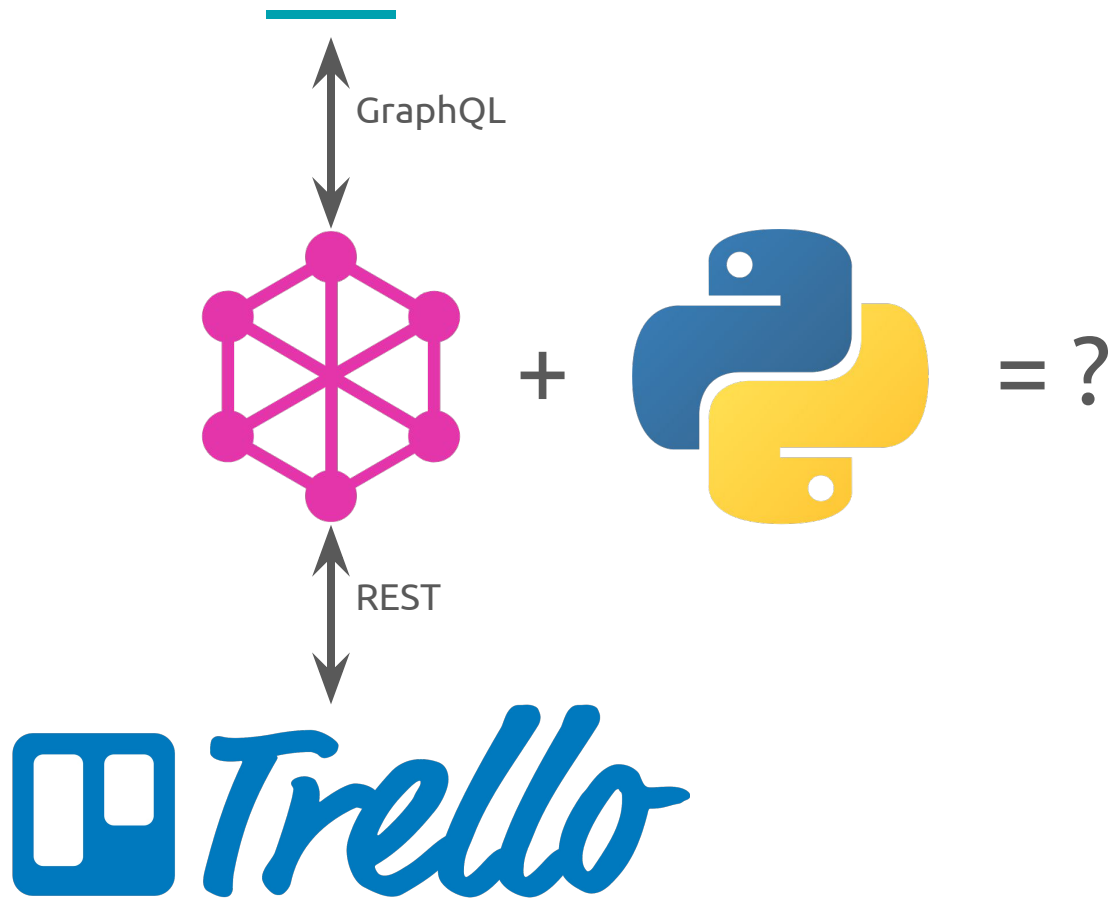


# Demo app: Trello REST API to GraphQL



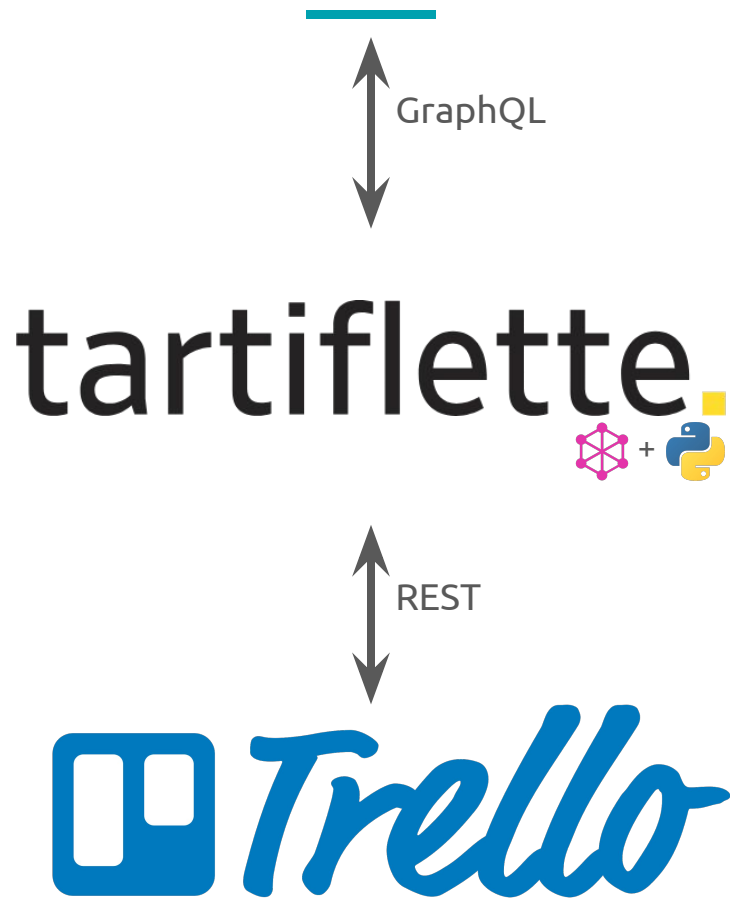


# Demo app: Trello REST API to GraphQL





# Demo app: Trello REST API to GraphQL



# Tartiflette main features

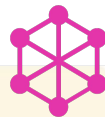
---

- **Python 3.6+**
- **Schema First (SDL)**
- **Built on AsyncIO**
- **aiohttp integration**
- **Embedded GraphQL development web interface**
- **Tastes even better than it smells (AKA developer friendly)**



# Schema Definition Language

---



```
1 type Query {  
2   member(id: String! = me): Member!  
3 }  
4  
5 # https://developers.trello.com/reference#member-object  
6 type Member {  
7   avatarUrl: String  
8   bio: String!  
9   boards: [Board!]  
10  confirmed: Boolean!  
11  email: String  
12  fullName: String!  
13  id: ID!  
14  organizations: [Organization!]  
15  initials: String!  
16  loginTypes: [String!]  
17  memberReferrer: Member  
18  memberType: String!  
19  url: String!  
20  username: String!  
21 }
```

# 1 GraphQL request = x REST requests



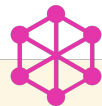
## Member Object

SUGGEST EDITS

<b>id</b> string	The ID of the member
<b>idBoards</b> array of strings	<u>An array of board IDs this member is on</u>
<b>idBoardsPinned</b> array of strings	deprecated
<b>idOrganizations</b> array of strings	<u>An array of organization IDs this member is in</u>

These edges will resolve in multiple REST API calls  
1 GraphQL call = multiple REST calls

```
1 type Query {
2   member(id: String! = me): Member!
3 }
4
5 # https://developers.trello.com/reference#member-object
6 type Member {
7   avatarUrl: String
8   bio: String!
9   boards: [Board!]
10  confirmed: Boolean!
11  email: String
12  fullName: String!
13  id: ID!
14  organizations: [Organization!]
15  initials: String!
16  loginTypes: [String!]
17  memberReferrer: Member
18  memberType: String!
19  url: String!
20  username: String!
21 }
22
23 # https://developers.trello.com/reference#board-object
24 type Board {
25   closed: Boolean!
26   desc: String
27   id: ID!
28   name: String!
29   url: String!
30 }
31
32 # https://developers.trello.com/reference#organization-object
33 type Organization {
34   desc: String
35   displayName: String!
36   id: ID!
37   name: String!
38   url: String!
39   website: String
40 }
```



```

1 query me {
2   member(id: "me") {
3     id
4     fullName
5     boards {
6       name
7       closed
8       url
9     }
10    organizations {
11      name
12      displayName
13    }
14  }
15 }
16

```

```

{
  "data": {
    "member": {
      "id": "5d1f33ca812dce738a69deec",
      "fullName": "ultrabug_ep2019",
      "organizations": [],
      "boards": [
        {
          "name": "Production Workflow",
          "closed": false,
          "url": "https://trello.com/b/pS9c6FIp/production-workflow"
        },
        {
          "name": "Europython 2019 Talk",
          "closed": false,
          "url": "https://trello.com/b/PB2UqOC8/europython-2019-talk"
        }
      ]
    }
  }
}

```

resolved edge with full objects

1x

```

1 type Query {
2   member(id: String! = me): Member!
3 }
4
5 # https://developers.trello.com/reference#member-object
6 type Member {
7   avatarUrl: String!
8   bio: String!
9   boards: [Board!]
10  confirmed: Boolean!
11  email: String!
12  fullName: String!
13  id: ID!
14  organizations: [Organization!]
15  initials: String!
16  loginTypes: [String!]
17  memberReferrer: Member
18  memberType: String!
19  url: String!
20  username: String!
21 }
22
23 # https://developers.trello.com/reference#board-object
24 type Board {
25   closed: Boolean!
26   desc: String!
27   id: ID!
28   name: String!
29   url: String!
30 }
31
32 # https://developers.trello.com/reference#organization-object
33 type Organization {
34   desc: String!
35   displayName: String!
36   id: ID!
37   name: String!
38   url: String!
39   website: String!
40 }

```

2x



Member Object

SUGGEST EDITS

<b>id</b>	The ID of the member
<b>boards</b>	An array of board IDs this member is on
<b>boardsPinned</b>	deprecated
<b>organizations</b>	An array of organization IDs this member is in

'idBoards': ['5d1f33e746ea0a8020560465', '5d1f341e82d5a37d0efb97b1']

# Show me some code: aiohttp app definition

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 import os
5
6 from aiohttp import web
7 from tartiflette import Engine
8 from tartiflette_aiohttp import register_graphql_handlers
9
10 import trello_to_graphql.resolvers
11
12 engine = Engine(
13     [os.path.dirname(os.path.abspath(__file__)) + "/trello_to_graphql/sdl/queries.sdl"]
14 )
15 ctx = {}
16
17 web.run_app(
18     register_graphql_handlers(
19         app=web.Application(),
20         engine=engine,
21         executor_context=ctx,
22         executor_http_endpoint="/graphql",
23         executor_http_methods=["POST", "GET"],
24         graphql_enabled=True,
25     )
26 )
```

Generic SDL

Resolver functions



# Show me some code: GraphQL resolvers

```
1 from tartiflette import Resolver
2
3 from trello_to_graphql.managers import send_request
4
5
6 @Resolver("Query.member")
7 async def query_member(parent, args, ctx, info):
8     # print("resolve member", parent, args)
9     endpoint = "members/" + args.pop("id")
10    return await send_request(endpoint, args)
11
12
13 @Resolver("Member.boards")
14 async def query_boards(parent, args, ctx, info):
15     # print("resolve boards", parent, args)
16     all_boards = []
17     for idBoard in parent.get("idBoards", []):
18         endpoint = "boards/" + idBoard
19         all_boards.append(await send_request(endpoint, args))
20    return all_boards
```

Root query resolver

Edge resolver



#shipit

# Dockerfile: multi-stage build

Full python3.7 build image

```
1 FROM python:3.7 AS build
2
3 RUN apt-get update
4 RUN apt-get install -y cmake
5
6 WORKDIR /tmp
7 ADD https://github.com/graphql/libgraphqlparser/archive/v0.7.0.tar.gz .
8 RUN tar -xzf v0.7.0.tar.gz \
9     && cmake -S libgraphqlparser-0.7.0 \
10    && make \
11    && rm v0.7.0.tar.gz
12
13 WORKDIR /app
14 COPY requirements.txt .
15 RUN pip install -r requirements.txt
16 COPY trello_to_graphql trello_to_graphql
17 COPY run.py .
18
19
```

Slim python3.7 run image

```
20 FROM python:3.7-slim
21
22 COPY --from=build /usr/local/lib/python3.7 /usr/local/lib/python3.7
23 COPY --from=build /tmp/libgraphqlparser.so /usr/lib/
24
25 USER nobody
26
27 WORKDIR /app
28 COPY --from=build /app .
29 CMD ["python", "-u", "run.py"]
```

# Build + Image tag = git branch + Upload to Gitlab registry

---

## Git branch workflow

- development
- staging
- master + git tag = production

```
1 #!/bin/bash
2
3 BRANCH=$(grep ref .git/HEAD | sed 's@.*\/\(.*\)\@l@g')
4 echo "pushing ${BRANCH} image"
5
6 docker build -t registry.numberly.in/ajm/trello-to-graphql:${BRANCH} . && \
7 docker push registry.numberly.in/ajm/trello-to-graphql:${BRANCH}
```

# To Kubernetes!

```
1 apiVersion: apps/v1
2 kind: Deployment
3
4 ...
5
6 spec:
7   securityContext:
8     runAsUser: 65534 # nobody
9   containers:
10    - name: trello-to-graphql-development
11      image: registry.numberly.in/ajm/trello-to-graphql:development
12    env:
13      - name: TRELLO_API_KEY
14        valueFrom:
15          secretKeyRef:
16            name: trello-to-graphql-development
17            key: trello-api-key
18      - name: TRELLO_TOKEN
19        valueFrom:
20          secretKeyRef:
21            name: trello-to-graphql-development
22            key: trello-token
23    ports:
24      - containerPort: 8080
25    resources:
26      limits:
27        cpu: 10m
28        memory: 32Mi
29      requests:
30        cpu: 10m
31        memory: 32Mi
```

**Security**



```
1 apiVersion: extensions/v1beta1
2 kind: Ingress
3 metadata:
4   name: trello-to-graphql-development
5   annotations:
6     kubernetes.io/tls-acme: "true"
7     nginx.ingress.kubernetes.io/rewrite-target: /
8     nginx.ingress.kubernetes.io/ssl-redirect: "false"
9   namespace: sandbox
10 spec:
11   rules:
12     - host: trello-to-graphql-development.sandbox.numberly.in
13       http:
14         paths:
15           - backend:
16               serviceName: trello-to-graphql-development
17               servicePort: 80
18   tls:
19     - hosts:
20         - trello-to-graphql-development.sandbox.numberly.in
21       secretName: tls-trello-to-graphql-development
```



Automated  
Let's Encrypt SSL



Quick demo



Take Away



# Take Away

- **GraphQL removes friction** by normalizing how data is addressed between teams
- **Schema Definition Language** lets you **concentrate on the data**, not the code
- **Tartiflette** is a modern, fast and efficient way of doing Python + GraphQL
- **Workflow** for environment deployment **based on git branches**
  - TODO: challenge environment multi-tenancy of the cluster later
- **Kubernetes secrets** + environment variables to store and access secrets
  - TODO: generalize vault
- **Kubectl is powerful: give that power to developers!**
  - TODO: allow some abstraction tools when adoption is higher if needed

# Thanks!

@ultrabug

---

<https://github.com/ultrabug/ep2019>

numberly  
boomeris group

