NaturalAntibody

Introduction

This document outlines how to create a setup to run dockers that are needed to run our application in your environment. All the examples are based on how we did it with our infrastructure, but it should be as much service-agnostic as possible. In your case, some resources might have to be adjusted.

We will provide you with an example of Kubernetes manifests. In this document we will sometimes reference names of those manifests when discussing applying it.
We are using Fargate and EC2 for this setup.

Third-party images

Assuming that your machines won't be able to connect externally, we recommend that you put the necessary third-party container images into your internal container registry. We assume that you are also using ECR Repositories.

Redis

We assume that Redis is running in the `naturalantibody-platform-redis` namespace, though you can adjust this to match your environment.

Example namespace definition:

```
apiVersion: v1
kind: Namespace
metadata:
 name: naturalantibody-platform-redis
```

The official Redis image is available at [redis - Official Image | Docker Hub]
We recommend using **Redis version 8.0**.

For deployment instructions, refer to the official Redis documentation.

Check if Redis is running:

```
$ kubectl get pods -n=naturalantibody-platform-redis
NAME                     READY   STATUS    RESTARTS   AGE
redis-67f8bff8b6-q4lct   1/1     Running   0          29m
```

RabbitMQ

We assume that RabbitMQ runs in the namespace naturalantibody-platform-rabbitmq and naturalantibody-platform-rabbitmq-system, but you can modify these to match your environment.

Example namespace definition:

```
apiVersion: v1
kind: Namespace
metadata:
 name: naturalantibody-platform-rabbitmq
```

The official RabbitMQ image is available at https://hub.docker.com/_/rabbitmq
We recommend using **RabbitMQ version 4.1.2**
For deployment instructions, refer to the official documentation

Check if RabbitMQ is running:

```
$ kubectl get pods -A | grep rabbitmq
naturalantibody-platform-rabbitmq          rabbitmq-server-0                                    1/1     Running   0          4d
```

MongoDB

We assume that MongoDB runs in the namespace naturalantibody-platform-mongo, but you can modify this to match your environment.

Example namespace definition:

```
apiVersion: v1
kind: Namespace
metadata:
 name: naturalantibody-platform-mongo
```

The official RabbitMQ image is available at https://hub.docker.com/_/mongo
We recommend using **RabbitMQ version 4.1.2**
For deployment instructions, refer to the official documentation

We are requiring a few databases called:

- mailer
- users
- search
- sequence_engineering
- project_management

And users for each database with read/write permissions

Example of how to create a user:
https://www.mongodb.com/docs/manual/reference/method/db.createUser/

Dumps for this part are stored in s3 bucket:
aws s3 cp s3://naturalantibody-public-dumps/search.tar.xz . --request-payer requester

Example uploading dump using mongorestore:

1. Install Mongo Database Tools
2. Copy dumps to ./dump on the machine that can connect to Mongo

3. `mongorestore --username root --password <password-for-mongodb-root> --authenticationDatabase=admin mongodb://<your-mongo-host>/search?ssl=false&authMechanism=SCRAM-SHA-256`

Natural Antibody Images

All of the following images can be pulled from the ECR repository in our organization that you have access to.

Users

Images required for this:

- 071253002612.dkr.ecr.eu-central-1.amazonaws.com/naturalantibody-users-repository-public:latest

We assume that Users run in the namespace `naturalantibody-platform`, but you can change it to fit your setup. Ensure that URLs are correct based on a namespace.

Change nodeSelector and tolerances to fit your setup.

Environment variables

Only variables that you need to change to match your setup.

- MONGODB_PASSWORD

  Example:

```
- name: MONGODB_PASSWORD
 valueFrom:
   secretKeyRef:
     key: password
     name: mongodb-users
```

  Value:

Password for the database that is used for the users microservice.

- MONGODB_URL

  Example:

```
- name: MONGODB_URL
 value:
mongodb://users:$(MONGODB_PASSWORD)@mongodb-svc.naturalantibody-platform-
mongo.svc.cluster.local/users?ssl=false&authSource=admin
```

  Value:

  Connect URL for MongoDB.

  mongodb://<user for this database>:$(MONGODB_PASSWORD)@<internal DNS record of MongoDB service>/<database name>?ssl=false&authSource=admin

- MAILER

  Example:

```
- name: MAILER
 value: MICROSERVICE
```

  Value:

  Possible values: [MICROSERVICE, DUMMY]

  For testing you can just use DUMMY which will print the mail content to logs.

  MICROSERVICE needs a running Mailer container.

- FORGOT_PASSWORD_URL

  Example:

```
- name: FORGOT_PASSWORD_URL
 value: https://open.naturalantibody.com/forgot-password/reset
```

  Value:

  External address of frontend + /forgot-password/reset

- MAILER_API_URL

Example:

```
- name: MAILER_API_URL
  value: http://mailer.naturalantibody-platform-mongo.svc.cluster.local
```

Value:

Internal DNS record of Mailer service.

If your MAILER value is DUMMY, this can be whatever you want.

- CHANGE_MAIL_URL

Example:

```
- name: CHANGE_MAIL_URL
  value: 'https://open.naturalantibody.com/reset-email'
```

Value:

External address of frontend + /reset-email

- REDIS_URL

Example:

```
- name: REDIS_URL
  value: redis.naturalantibody-platform-redis.svc.cluster.local
```

Value:

Internal DNS record of Redis.

- SECRET_KEY

Example:

```
- name: SECRET_KEY
  valueFrom:
    secretKeyRef:
      key: password
      name: token-secret
```

Value:

Secret that contains value used for creating and unpacking authorization tokens.

Other variables

- image

  Example:
  ```
  image: <aws account id>.dkr.<url for ECR repository for users>:v1.0.0
  ```

  Value:
  URL to Users microservice image.

- namespace

  Example:
  ```
  namespace: naturalantibody-platform
  ```

  Value:
  Name of main namespace used for services.

Steps to apply manifests for users:
`kubectl apply -f users/deployment.yaml`

API Gateway

Images required for this:
- 071253002612.dkr.ecr.eu-central-1.amazonaws.com/naturalantibody-api-gateway-repository-public:latest

We assume that API Gateway runs in the namespace `naturalantibody-platform`, but you can change it to fit your setup.

Change nodeSelector and tolerances to fit your setup.

Environment variables

Only variables that you need to change to match your setup.

- SECRET_KEY

  Example:

```
- name: SECRET_KEY
  valueFrom:
    secretKeyRef:
      key: password
      name: token-secret
```

  Value:

  A secret that contains value used for creating and unpacking authorization tokens.

- REDIS_URL

  Example:

```
- name: REDIS_URL
  value: redis.naturalantibody-platform-redis.svc.cluster.local:6379
```

  Value:

  Internal DNS record of Redis + port.

- SEARCH_API_URL

  Example:

```
- name: SEARCH_API_URL
  value: http://search-api.naturalantibody-platform.svc.cluster.local
```

  Value:

  Internal DNS record of the Antibody Databases service.

- USERS_API_URL

  ```
  - name: USERS_API_URL
   value: http://users.naturalantibody-platform.svc.cluster.local
  ```

  Value:

  Internal DNS record of the Users service.

- SEQUENCE_ENGINEERING_API_URL

  Example:

  ```
  - name: SEQUENCE_ENGINEERING_API_URL
   value:
  http://sequence-engineering-api.naturalantibody-platform.svc.cluster.loca
  l
  ```

  Value:

  Internal DNS record of the Sequence Engineering service.

- PROJECT_MANAGEMENT_API_URL

  Example:

  ```
  - name: PROJECT_MANAGEMENT_API_URL
   value:
  http://project-management-api.naturalantibody-platform.svc.cluster.local
  ```

  Value:

  Internal DNS record of the Project Management service.

- COOKIE_DOMAIN

  Example:

  ```
  - name: COOKIE_DOMAIN
   value: .naturalantibody.com
  ```

  Value:

  Domain part of the external URL for the frontend.

- UI_URL

  Example:
  ```
  - name: UI_URL
   value: "https://open.naturalantibody.com"
  ```

  Value:

  External URL for frontend.

Other variables

- image

  Example:
  ```
  image: <aws account id>.dkr.<url for ECR repository for api
  gateway>:v1.0.0
  ```

  Value:

  URL to API Gateway microservice image.

- namespace

  Example:
  ```
  namespace: naturalantibody-platform
  ```

  Value:

  Name of the main namespace used for services.

Steps to apply manifests for users:

```
kubectl apply -f api-gateway/deployment.yaml
```

Frontend

Images required for this:

- 071253002612.dkr.ecr.eu-central-1.amazonaws.com/naturalantibody-frontend-repository-public:latest

We assume that Frontend runs in the namespace `naturalantibody-platform`, but you can change it to fit your setup.

Change nodeSelector and tolerances to fit your setup.

Environment variables

Only variables that you need to change to match your setup.

- REDIS_URL

  Example:
  ```
  - name: REDIS_URL
   value: redis.naturalantibody-platform-redis.svc.cluster.local:6379
  ```
- GRAPHQL_ENDPOINT

  Example:
  ```
  - name: GRAPHQL_ENDPOINT
   value:
  http://api-gateway.naturalantibody-platform.svc.cluster.local/graphql
  ```

  Value:

  Internal DNS record of API Gateway service + /graphql.

- APP_SERVER_URL
  ```
  - name: APP_SERVER_URL
   value: http://frontend.naturalantibody-platform.svc.cluster.local
  ```

  Value:

  Internal DNS record of Frontend service

- USERS_API_URL

```
- name: USERS_API_URL
 value: http://users.naturalantibody-platform.svc.cluster.local
```

Value:

Internal DNS record of the Users service.

- SEQUENCE_ENGINEERING_API_URL

Example:

```
- name: SEQUENCE_ENGINEERING_API_URL
 value:
http://sequence-engineering-api.naturalantibody-platform.svc.cluster.loca
l
```

Value:

Internal DNS record of the Sequence Engineering service.

- SEARCH_API_URL

Example:

```
- name: SEARCH_API_URL
 value: http://search-api.naturalantibody-platform.svc.cluster.local
```

Value:

Internal DNS record of the Antibody Databases service.

- PROJECT_MANAGEMENT_API_URL

Example:

```
- name: PROJECT_MANAGEMENT_API_URL
 value:
http://project-management-api.naturalantibody-platform.svc.cluster.local
```

Value:

Internal DNS record of the Project Management service.

- COOKIE_DOMAIN

Example:

```
- name: COOKIE_DOMAIN
 value: .naturalantibody.com
```

Value:

Domain part of the external URL for the frontend.

- TWOFA_TRANSITION_TOKEN_SECRET

```
- name: TWOFA_TRANSITION_TOKEN_SECRET
 valueFrom:
   secretKeyRef:
     name: abstudio-secrets
     key: TWOFA_TRANSITION_TOKEN_SECRET
```

Value:

A secret that is used as salt for creating sessions.

- AUTH_SECRET

```
- name: AUTH_SECRET
valueFrom:
 secretKeyRef:
   name: abstudio-secrets
   key: AUTH_SECRET
```

Value:

Secret used for Auth0-based login. Secret of the Octa organization.

- CSRF_SECRET

```
- name: CSRF_SECRET
 valueFrom:
   secretKeyRef:
     name: abstudio-secrets
     key: CSRF_SECRET
```

Value:

A secret that is used as salt for tokens that prevents CSRF attacks.

- GRAPHQL_DOC_HOST

Example:

```
- name: GRAPHQL_DOC_HOST
 value: api-docs.open.naturalantibody.com
```

Value:

External address of API Docs Service.

- DOCS_HOST

  Example:

  ```
  - name: DOCS_HOST
   value: docs.open.naturalantibody.com
  ```

  Value:

  External address of Docs Service.

- NEXT_PUBLIC_DOCS_URL

  Example:

  ```
  - name: NEXT_PUBLIC_DOCS_URL
   value: https://docs.open.naturalantibody.com
  ```

  Value:

  https:// + external address of Docs Service.

- NEXT_PUBLIC_GRAPHQL_CLIENT_ENDPOINT

  Example:

  ```
  - name: NEXT_PUBLIC_GRAPHQL_CLIENT_ENDPOINT
   value: https://platform-api.open.naturalantibody.com/graphql
  ```

  Value:

  External address of API Gateway service + /graphql.

- TOKEN_SECRET

Example:

```
- name: TOKEN_SECRET
 valueFrom:
   secretKeyRef:
     name: token-secret
     key: password
```

Value:

A secret that contains value used for creating and unpacking authorization tokens.

- NEXTAUTH_URL

```
- name: NEXTAUTH_URL
 value: https://open.naturalantibody.com
```

Value:

External DNS record of Frontend service

- AUTH0_ISSUER

```
- name: AUTH0_ISSUER
 value: https://organization.eu.auth0.com
```

Value:

Url to Octa app.

- AUTH0_CLIENT_ID

```
- name: AUTH0_CLIENT_ID
 value: 8fs7d8gf87sdgsd8fg7s
```

Value:

ID of Octa app.

- AUTH0_CLIENT_SECRET

```
- name: AUTH0_CLIENT_SECRET
 valueFrom:
   secretKeyRef:
     name: abstudio-secrets
     key: AUTH0_CLIENT_SECRET
```

Value:

Secret of Octa app.

Other variables

- image

  Example:
  ```
  image: <aws account id>.dkr.<url for ECR repository for frontend>:v1.0.0
  ```

  Value:

  URL to Frontend microservice image.

- namespace

  Example:
  ```
  namespace: naturalantibody-platform
  ```

  Value:

  Name of the main namespace used for services.

Steps to apply manifests for users:

`kubectl apply -f frontend/deployment.yaml`

Ingress

We assume that Ingress is placed in the namespace `naturalantibody-platform`, but you can change it to fit your setup.

Change hosts to match your setup. In all deployments in this part, you have to change `<your-domain-url>` and `<your-api-domain-url>` so it will match the hosts.

This is an example ingress definition to run on an EKS Cluster.

Steps to apply manifest:
`kubectl apply -f ingress/ingress.yaml`

Load Balancer

If you want to use AWS ALB, you can check our setup in `aws-load-balancer-controller` and `aws-pca-issuer`. It's created based on this tutorial from AWS:

- ○ https://docs.aws.amazon.com/eks/latest/userguide/lbc-manifest.html.

Sequence Engineering

Images required for this:

- ○ 071253002612.dkr.ecr.eu-central-1.amazonaws.com/naturalantibody-sequence-engineering-repository:latest

We assume that Sequence Engineering runs in the namespace `naturalantibody-platform`, but you can change it to fit your setup.

Change nodeSelector and tolerances to fit your setup.

Environment variables

Only variables that you need to change to match your setup.

- MONGODB_PASSWORD

  Example:

```
- name: MONGODB_PASSWORD
 valueFrom:
   secretKeyRef:
     name: mongodb-sequence-engineering
     key: password
```

Value:
Password for the database that is used for the Sequence Engineering microservice.

- MONGODB_URL

  Example:

```
- name: MONGODB_URL
 value:
mongodb://sequence-engineering:$(MONGODB_PASSWORD)@mongodb-svc.naturalant
ibody-platform-mongo.svc.cluster.local/sequence_engineering?authSource=ad
min&ssl=false
```

- SECRET_KEY

  Example:

```
- name: SECRET_KEY
 valueFrom:
   secretKeyRef:
     key: password
     name: token-secret
```

Value:
A secret that contains value used for creating and unpacking authorization tokens.

- RABBITMQ_DEFAULT_USER

  Example:

```
- name: RABBITMQ_DEFAULT_USER
 valueFrom:
   secretKeyRef:
     name: rabbitmq-default-user
     key: username
```

Value:

Username of the user used in the RabbitMQ service.

- RABBITMQ_DEFAULT_USER_PASSWORD

  Example:

  ```
  - name: RABBITMQ_DEFAULT_USER_PASSWORD
   valueFrom:
     secretKeyRef:
       name: rabbitmq-default-user
       key: password
  ```

  Value:

  Password for the user used in the RabbitMQ service.

- RABBITMQ_HOST

  Example:

  ```
  - name: RABBITMQ_HOST
   valueFrom:
     secretKeyRef:
       name: rabbitmq-default-user
       key: host
  ```

  Value:

  Host of internal DNS for the RabbitMQ service.

- USERS_API_URL

  ```
  - name: USERS_API_URL
   value: http://users.naturalantibody-platform.svc.cluster.local
  ```

  Value:

  Internal DNS record of the Users service.

- REDIS_URL

  Example:

```
- name: REDIS_URL
  value: redis.naturalantibody-platform-redis.svc.cluster.local
```

Value:

Internal DNS record of Redis.

Other variables

- image

  Example:

  ```
  image: <aws account id>.dkr.<url for ECR repository for sequence
  engineering>:v1.0.0
  ```

  Value:

  URL to Sequence Engineering microservice image.

- namespace

  Example:

  ```
  namespace: naturalantibody-platform
  ```

  Value:

  Name of the main namespace used for services.

- fileSystemId

  Example:

  ```
  fileSystemId: fs-6rv6b86b5665b
  ```

  Value:

  ID of the file system created for persistent volumes.

For services based on the Sequence Engineering image (Sequence Engineering and some workers), you will need to have a disk with models mounted in these containers. Models should be available in ***/data***. In our infrastructure, we use file systems that you can see in persistent-volume-claims.yaml. You don't have to use EFS for that, if you want to use other file systems change it to fit your setup.

You can download all required data from S3:
aws s3 cp s3://naturalantibody-public-dumps/naturalantibody-models.tar.xz . --request-payer requester
Files and directories should be like

```
root@<seqeunce-engineering-pod>:/app# ls /data
ABDB        deepdiver          docking           immunogenicity     nanobert
naturalness developability     germlines.json    liabilities        paratope
profiles     structure_prediction
```

Steps to apply manifests for Sequence Engineering:

```
kubectl apply -f sequence-engineering/persistent-volume-claims.yaml
kubectl apply -f sequence-engineering/deployment.yaml
kubectl apply -f sequence-engineering/batch-grained-tasks.yaml
kubectl apply -f sequence-engineering/batch-tasks.yaml
kubectl apply -f sequence-engineering/docking-tasks.yaml
kubectl apply -f sequence-engineering/sequence-engineering-tasks.yaml
kubectl apply -f sequence-engineering/structural-modeling-tasks.yaml
```

Antibody Databases

Images required for this:

- 071253002612.dkr.ecr.eu-central-1.amazonaws.com/naturalantibody-antibody-databases-repository-public:latest

Change nodeSelector and tolerances to fit your setup.
We assume that Antibody Databases runs in the namespace naturalantibody-platform, but you can change it to fit your setup.

Environment variables

Only variables that you need to change to match your setup.

- MONGODB_PASSWORD

  Example:
  ```
  - name: MONGODB_PASSWORD
   valueFrom:
     secretKeyRef:
       name: mongodb-search
       key: password
  ```

  Value:

  Password for the database that is used for the Antibody Databases microservice.

- DB_HOST

  Example:
  ```
  - name: DB_HOST
   value:
  mongodb://search:$(MONGODB_PASSWORD)@mongodb-svc.naturalantibody-platform
  -mongo.svc.cluster.local/search?ssl=false&authSource=admin
  ```

  Value:

  Connect URL for MongoDB.

  mongodb://<user for this database>:$(MONGODB_PASSWORD)@<internal DNS record of MongoDB service>/<database name>?ssl=false&authSource=admin

- RABBITMQ_DEFAULT_USER

  Example:
  ```
  - name: RABBITMQ_DEFAULT_USER
   valueFrom:
     secretKeyRef:
       name: rabbitmq-default-user
       key: username
  ```

Value:

Username of the user used in the RabbitMQ service.

- RABBITMQ_DEFAULT_USER_PASSWORD

  Example:

```
- name: RABBITMQ_DEFAULT_USER_PASSWORD
 valueFrom:
   secretKeyRef:
     name: rabbitmq-default-user
     key: password
```

  Value:

  Password for the user used in the RabbitMQ service.

- RABBITMQ_HOST

  Example:

```
- name: RABBITMQ_HOST
 valueFrom:
   secretKeyRef:
     name: rabbitmq-default-user
     key: host
```

  Value:

  Host of internal DNS for the RabbitMQ service.

- SECRET_KEY

  Example:

```
- name: SECRET_KEY
 valueFrom:
   secretKeyRef:
     key: password
     name: token-secret
```

  Value:

A secret that contains value used for creating and unpacking authorization tokens.

- REDIS_URL

  Example:
  ```
  - name: REDIS_URL
    value: redis.naturalantibody-platform-redis.svc.cluster.local:6379
  ```

  Value:
  Internal DNS record of Redis + port.

- USERS_API_URL
  ```
  - name: USERS_API_URL
    value: http://users.naturalantibody-platform.svc.cluster.local
  ```

  Value:
  Internal DNS record of the Users service.

Other variables

- image

  Example:
  ```
  image: <aws account id>.dkr.<url for ECR repository for antibody
  databases>:v1.0.0
  ```

  Value:
  URL to the Antibody Databases microservice image.

- namespace

  Example:
  ```
  namespace: naturalantibody-platform
  ```

  Value:
  Name of the main namespace used for services.

Steps to apply manifests for Antibody Databases:

```
kubectl apply -f antibody-databases/deployment.yaml
```

Mailer

Images required for this:

- 071253002612.dkr.ecr.eu-central-1.amazonaws.com/naturalantibody-mailer-repository-repository-public:latest

Change nodeSelector and tolerances to fit your setup.
We assume that Mailer runs in the namespace `naturalantibody-platform`, but you can change it to fit your setup.

Environment variables

Only variables that you need to change to match your setup.

- MONGODB_PASSWORD

  Example:

  ```
  - name: MONGODB_PASSWORD
   valueFrom:
     secretKeyRef:
       name: mongodb-mailer
       key: password
  ```

  Value:
  Password for the database that is used for the Mailer microservice.

- MONGODB_URL

  Example:

  ```
  - name: MONGODB_URL
  ```

```
value:
mongodb://mailer:$(MONGODB_PASSWORD)@mongodb-svc.naturalantibody-platform
-mongo.svc.cluster.local/mailer?ssl=false&authSource=admin
```

Value:

Connect URL for MongoDB.

`mongodb://<user for this database>:$(MONGODB_PASSWORD)@<internal DNS record of MongoDB service>/<database name>?ssl=false&authSource=admin`

- RABBITMQ_DEFAULT_USER

Example:

```
- name: RABBITMQ_DEFAULT_USER
 valueFrom:
   secretKeyRef:
     name: rabbitmq-default-user
     key: username
```

Value:

Username of the user used in the RabbitMQ service.

- RABBITMQ_DEFAULT_USER_PASSWORD

Example:

```
- name: RABBITMQ_DEFAULT_USER_PASSWORD
 valueFrom:
   secretKeyRef:
     name: rabbitmq-default-user
     key: password
```

Value:

Password for the user used in the RabbitMQ service.

- RABBITMQ_HOST

Example:

```
- name: RABBITMQ_HOST
 valueFrom:
   secretKeyRef:
     name: rabbitmq-default-user
     key: host
```

Value:

Host of internal DNS for the RabbitMQ service.

- SECRET_KEY

  Example:

```
- name: SECRET_KEY
 valueFrom:
   secretKeyRef:
     key: password
     name: token-secret
```

  Value:

  A secret that contains value used for creating and unpacking authorization tokens.

- REDIS_URL

  Example:

```
- name: REDIS_URL
 value: redis://redis.naturalantibody-platform-redis.svc.cluster.local
```

  Value:

  redis:// + internal DNS for Redis service.

- USERS_API_URL

  Example:

```
- name: USERS_API_URL
 value: http://users.naturalantibody-platform.svc.cluster.local
```

  Value:

  Internal DNS record of the Users service.

- USE_SSL

  Example:

  ```
  - name: USE_SSL
   Value: true
  ```

  Value:

  Boolean. If true, the SMTP server will use SSL; if false, it will use TLS.

- SMTP_SERVER

  Example:

  ```
  - name: SMTP_SERVER
   value: smtp.gmail.com
  ```

  Value:

  SMTP provider.

- PORT

  Example:

  ```
  - name: PORT
   value: '587'
  ```

  Value:

  Port for SMTP connection.

- USERNAME

  Example:

  ```
  - name: USERNAME
   valueFrom:
     secretKeyRef:
       name: client-configuration
       key: username
  ```

  Value:

Username for the account used to send emails.

- PASSWORD

  Example:
  ```
  - name: PASSWORD
   valueFrom:
     secretKeyRef:
       name: client-configuration
       key: password
  ```

  Value:

  Password for the account used to send emails.

- SENDER_EMAIL

  Example:
  ```
  - name: SENDER_EMAIL
   value: no-reply@naturalantibody.com
  ```

  Value:

  What address will be displayed when sending email

Other variables

- image

  Example:
  ```
  image: <aws account id>.dkr.<url for ECR repository for mailer>:v1.0.0
  ```

  Value:

  URL to Mailer microservice image.

- namespace

Example:

```
namespace: naturalantibody-platform
```

Value:

Name of the main namespace used for services.

Steps to apply manifests for Mailer:

`kubectl apply -f mailer/deployment.yaml`

`kubectl apply -f mailer/mailer-tasks.yaml`

Project Management

Images required for this:

- 071253002612.dkr.ecr.eu-central-1.amazonaws.com/naturalantibody-project-management-repository-public:latest

Change nodeSelector and tolerances to fit your setup.
We assume that Project Management runs in the namespace `naturalantibody-platform`, but you can change it to fit your setup.

Environment variables

Only variables that you need to change to match your setup.

- MONGODB_PASSWORD

Example:

```
- name: MONGODB_PASSWORD
 valueFrom:
   secretKeyRef:
     name: mongodb-project-management
     key: password
```

Value:

Password for the database that is used for the Project Management microservice.

- MONGODB_URL

  Example:
  ```
  - name: MONGODB_URL
    value:
  mongodb://project-management:$(MONGODB_PASSWORD)@mongodb-svc.naturalantib
  ody-platform-mongo.svc.cluster.local/project_management?ssl=false&authSou
  rce=admin
  ```

  Value:

  Connect URL for MongoDB.
  `mongodb://<user for this database>:$(MONGODB_PASSWORD)@<internal DNS record of MongoDB service>/<database name>?ssl=false&authSource=admin`

- RABBITMQ_DEFAULT_USER

  Example:
  ```
  - name: RABBITMQ_DEFAULT_USER
    valueFrom:
      secretKeyRef:
        name: rabbitmq-default-user
        key: username
  ```

  Value:
  Username of the user used in the RabbitMQ service.

- RABBITMQ_DEFAULT_USER_PASSWORD

  Example:
  ```
  - name: RABBITMQ_DEFAULT_USER_PASSWORD
    valueFrom:
      secretKeyRef:
        name: rabbitmq-default-user
        key: password
  ```

Value:

Password for the user used in the RabbitMQ service.

- RABBITMQ_HOST

  Example:

```
- name: RABBITMQ_HOST
 valueFrom:
   secretKeyRef:
     name: rabbitmq-default-user
     key: host
```

  Value:

  Host of internal DNS for the RabbitMQ service.

- SECRET_KEY

  Example:

```
- name: SECRET_KEY
 valueFrom:
   secretKeyRef:
     key: password
     name: token-secret
```

  Value:

  A secret that contains value used for creating and unpacking authorization tokens.

- REDIS_URL

  Example:

```
- name: REDIS_URL
 value: redis.naturalantibody-platform-redis.svc.cluster.local:6379
```

  Value:

  Internal DNS record of Redis + port.

- SEARCH_API_URL

  Example:
```
- name: SEARCH_API_URL
 value: http://search-api.naturalantibody-platform.svc.cluster.local
```

  Value:

  Internal DNS record of the Antibody Databases service.

- USERS_API_URL

```
- name: USERS_API_URL
 value: http://users.naturalantibody-platform.svc.cluster.local
```

  Value:

  Internal DNS record of the Users service.

Other variables

- Image

  Example:
```
image: <aws account id>.dkr.<url for ECR repository for project
management>:v1.0.0
```

  Value:

  URL to Project Management microservice image.

- Namespace

  Example:
```
namespace: naturalantibody-platform
```

  Value:

  Name of the main namespace used for services.

Steps to apply manifests for Project Management:

```
kubectl apply -f project-management/deployment.yaml
kubectl apply -f project-management/project-management-tasks.yaml
```

API Docs

Images required for this:

- 071253002612.dkr.ecr.eu-central-1.amazonaws.com/naturalantibody-api-docs-repository-public:latest

Change nodeSelector and tolerances to fit your setup.
We assume that Docs runs in the namespace naturalantibody-platform, but you can change it to fit your setup.

Environment variables

Only variables that you need to change to match your setup.

- TOKEN_SECRET

  Example:

  ```
  - name: TOKEN_SECRET
   valueFrom:
     secretKeyRef:
       name: token-secret
       key: password
  ```

  Value:
  A secret that contains value used for creating and unpacking authorization tokens.

- UI_URL

  Example:

  ```
  - name: UI_URL
   value: https://platform.naturalantibody.com
  ```

  Value:

External URL for frontend.

- APP_URL

  Example:
  ```
  - name: APP_URL
   value: https://api-docs.platform.naturalantibody.com
  ```

  Value:

  External URL for this service.

Other variables

- Image

  Example:
  ```
  image: <aws account id>.dkr.<url for ECR repository for api-docs>:v1.0.0
  ```

  Value:

  URL to Docs microservice image.

- namespace

  Example:
  ```
  namespace: naturalantibody-platform
  ```
  Value:

  Name of the main namespace used for services.

Steps to apply manifests for API docs:

```
kubectl apply -f api-docs/deployment.yaml
```

Docs

Images required for this:

- 071253002612.dkr.ecr.eu-central-1.amazonaws.com/naturalantibody-docs-repository-public:latest

Change nodeSelector and tolerances to fit your setup.
We assume that Docs runs in the namespace naturalantibody-platform, but you can change it to fit your setup.

Environment variables

Only the variables that you need to change to match your setup.

- TOKEN_SECRET

  Example:
  ```
  - name: TOKEN_SECRET
   valueFrom:
     secretKeyRef:
       name: token-secret
       key: password
  ```

  Value:
  A secret that contains value used for creating and unpacking authorization tokens.

- NEXT_PUBLIC_UI_URL

  Example:
  ```
  - name: NEXT_PUBLIC_UI_URL
   value: https://platform.naturalantibody.com
  ```

  Value:
  External URL for frontend.

- APP_URL

  Example:
  ```
  - name: APP_URL
   value: https://api-docs.platform.naturalantibody.com
  ```

  Value:

  External URL for this service.

Other variables

- image

  Example:
  ```
  image: <aws account id>.dkr.<url for ECR repository for docs>:v1.0.0
  ```

  Value:

  URL to Docs microservice image.

- namespace

  Example:
  ```
  namespace: naturalantibody-platform
  ```

  Value:

  Name of the main namespace used for services.

Steps to apply manifests for Docs:

`kubectl apply -f docs/deployment.yaml`

Creating the first user

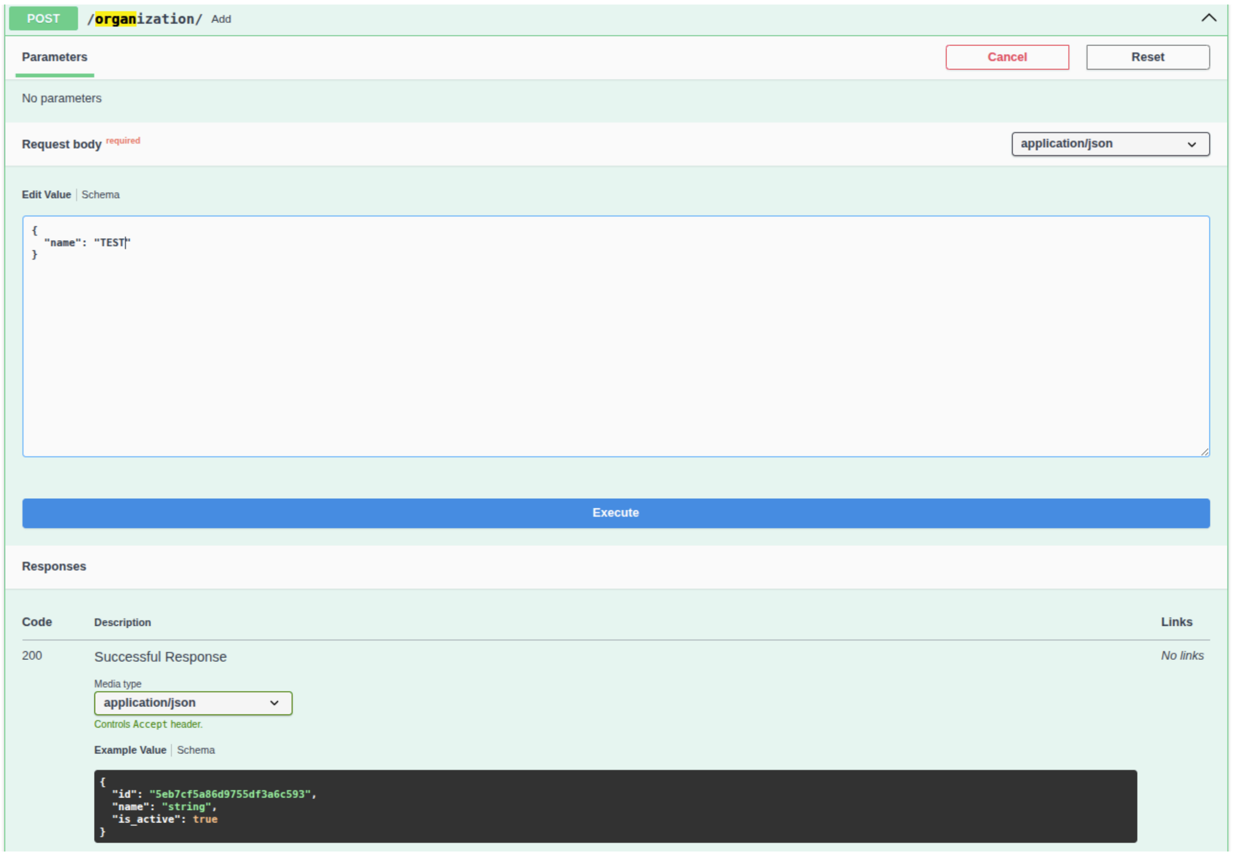
First user has to be created via the API of the users container

Connecting to the users container:

`kubectl port-forward service/users -n naturalantibody-platform 3000:80`

Then open your browser and go to http://localhost:3000/docs#

1. Create an organization using the POST /organization/ endpoint



POST /organization/ Add

Parameters

Cancel Reset

No parameters

Request body required

application/json

Edit Value | Schema

```
{
  "name": "TEST"
}
```

Execute

Responses

Code	Description	Links
200	Successful Response	No links

Media type: application/json

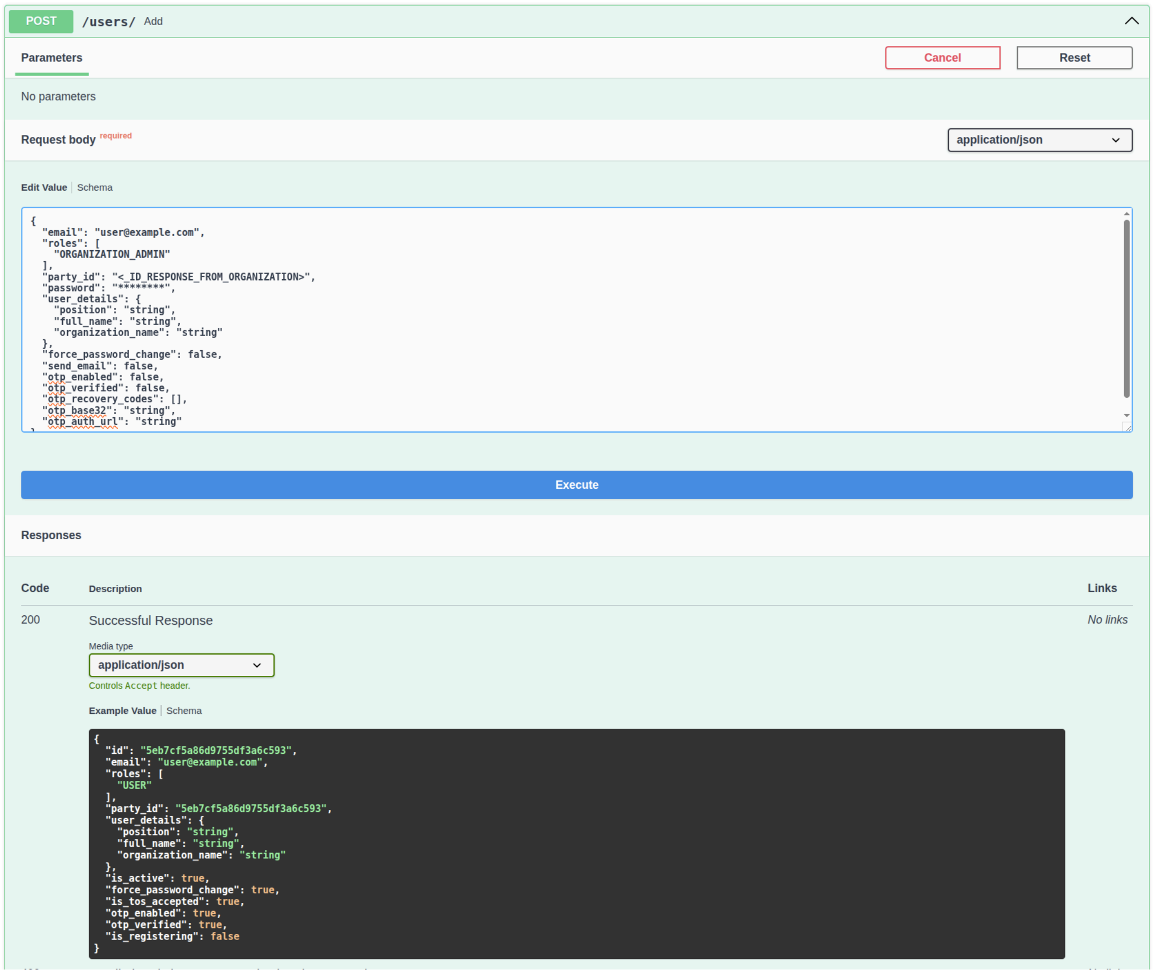
Controls Accept header.

Example Value | Schema

```
{
  "id": "5eb7cf5a86d9755df3a6c593",
  "name": "string",
  "is_active": true
}
```

Save _id from the response for further steps

2 . Create a user using POST /users/ endpoint



No parameters

Request body required

application/json

Edit Value | Schema

```
{
  "email": "user@example.com",
  "roles": [
    "ORGANIZATION_ADMIN"
  ],
  "party_id": "<_ID_RESPONSE_FROM_ORGANIZATION>",
  "password": "********",
  "user_details": {
    "position": "string",
    "full_name": "string",
    "organization_name": "string"
  },
  "force_password_change": false,
  "send_email": false,
  "otp_enabled": false,
  "otp_verified": false,
  "otp_recovery_codes": [],
  "otp_base32": "string",
  "otp_auth_url": "string"
```

Code	Description	Links
200	Successful Response	No links

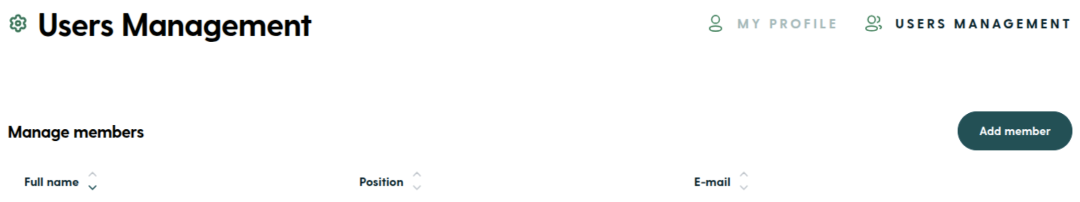
Media type: application/json

Controls Accept header.

Example Value | Schema

```
{
  "id": "5eb7cf5a86d9755df3a6c593",
  "email": "user@example.com",
  "roles": [
    "USER"
  ],
  "party_id": "5eb7cf5a86d9755df3a6c593",
  "user_details": {
    "position": "string",
    "full_name": "string",
    "organization_name": "string"
  },
  "is_active": true,
  "force_password_change": true,
  "is_tos_accepted": true,
  "otp_enabled": true,
  "otp_verified": true,
  "is_registering": false
}
```

3. Log in to the frontend with the credentials that you have created
4. To add more users, use the Form that is available at https://<your_application_domain>/user/users-management


Users Management
MY PROFILE
USERS MANAGEMENT
Manage members
Add member
Full name
Position
E-mail