Optimization of Kubernetes resources for movie files processing

Large Scale Computing

Maja Szrubarczyk Jakub Skulski

1 Introduction

In this project we proposed a simple REST API application created with microservices architecture that provides the ability for users to convert a video file into mp3 file. Services are contenerized with use of Docker and orchestrated by Kubernetes. As the main purpose of the project was to optimize resources of k8s deployment, each component is responsible for a different functionality. This approach allows to keep system design simplicity and assure the independence in matter of resources management (e.g. scalability).

2 Architecture

Figure 1. shows the microservices architecture of the application.

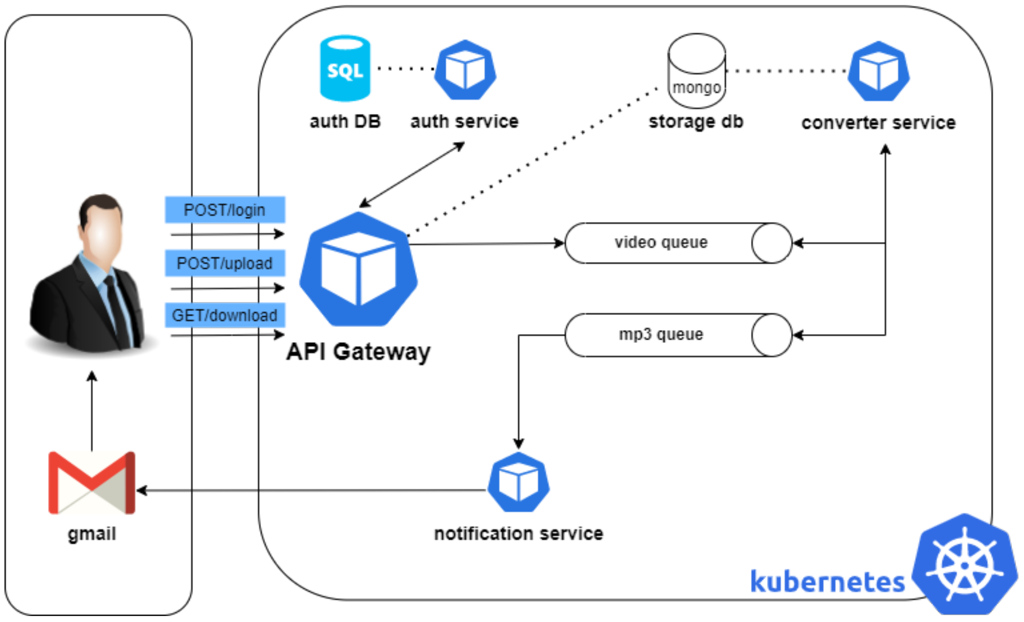


Figure 1: Application architecure

3 Communication

In order to guarantee better application performance the communication is synchronous only between Gateway and Authentication services. Spreading the informations among other services in an asynchronous way is significant for dealing with large files. Keeping uploading and downloading proccesses away is crucial in matter of user experience. API Gateway is the first service in communication flow and the only one exposed for external requests. Having received request with user credentials Gateway communicates with Authentication service to either generate a token or return an error. Assuming that, with use of token, user requested uploading a video file to the server, Gateway stores the file in Storage Database. At the same time it sends the message with user name and file ID to the queue system managed by Message Broker. Converter service creates mp3 files from video files uploaded by users. Having consuming the message from video queue it finds the proper file in Storage Database, converts it into mp3 file and notifies that task with given file ID was finished. As the last part of communication flow, Notification service, which is listening on mp3 queue, consumes the message and sends an email to the user that file with specified ID is ready to be download.

4 Components

Application components can be divided into two groups i.e. services that perform some functionalities (Gateway, Authentication, Converter, Notification) and services that are being used by the first group in order to process their tasks (databases, message broker). Both groups are orchestrated by Kubernetes but only resources of first type can be optimized when there is a need. Second group only provides its functionality and number of its replicas will always be equal to one. The logic of all services from first group was written with use of Python language and Flask framework. Message Broker was provided by RabbitMQ, and databases - authentication and storage respectively with mySQL and MongoDB.

4.1 Gateway service

Gateway service is a broker between user and application. Sharing endpoints described below is responsible for:

- POST /login
  - Sending received credentials to the Authentication service
- POST /upload
  - Uploading video file to the storage database
  - Sending message to the video queue with user name and unique file ID
- GET /download
  - Downloading mp3 file from storage database

4.2 Authentication service

Authentication service is responsible for user authorization. Having received user name and password, it checks if the provided credentials are equal to stored in authentication database. It shares endpoints only for Gateway service:

- POST /login
  - Comparing received credentials with those stored in authentication database
  - Creating user token and returning it to the Gateway service
- POST /validate
  - Checking if token provided by user is valid

4.3 Converter service

Converter service is listening on queue „video”. Having consumed a message with user name and file ID, it finds this video in storage database and converts it into mp3 file. After finishing this process, a message that file has been processed is being published on queue „mp3”.

4.4 Notification service

Notification service is listening on queue „mp3”. After consuming a message sends an email to the proper user that file with certain ID is ready to be download.

4.5 Message Broker

Message broker manages the queue system. It provides two queues – „video” and „mp3”.

4.6 Authentication database

Authentication database stores credentials of all registered users. It's data are available only for Authentication service.

4.7 Storage database

Storage database stores video and mp3 files for all users. It shares data only with Gateway and Converter services.

5 Deployment

Well considered deployment strategy is crucial for application performance and scalability. Having all microservices containerized and being aware that requests load generated by multiple users can affect the application efficiency, the decision to use Kubernetes as an orchestrator was made. In this section we present some of k8s functionalities and our approach to deploy the application within it.

5.1 Kubernetes

Kubernetes is a portable, extensible, open source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It provides e.g.:

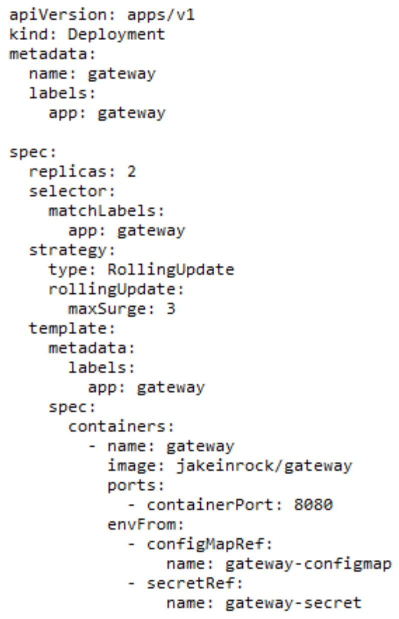
- Service discovery and load balancing
- Storage orchestration
- Automated rollouts and rollbacks
- Automatic bin packing
- Self-healing
- Secret and configuration management

5.2 Manifests

Each of microservices requires multiple resources to be created. In order to simplify the deployment process all information about these resources is stored in YAML configuration files often called manifests. Below are explained all configuration files used in our implementation.

5.2.1 Deployment

This file defines a desired state of a service running in a pod on any worker node. Deployment controller is able to change the actual state to the desired state when needed. Figure 2 presents deployment.yaml file for Gateway service.



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gateway
  labels:
    app: gateway

spec:
  replicas: 2
  selector:
    matchLabels:
      app: gateway
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 3
  template:
    metadata:
      labels:
        app: gateway
    spec:
      containers:
        - name: gateway
          image: jakeinrock/gateway
          ports:
            - containerPort: 8080
          envFrom:
            - configMapRef:
                name: gateway-configmap
            - secretRef:
                name: gateway-secret
```

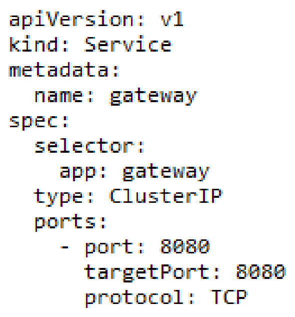
Figure 2: Gateway-deployment.yaml

Components of deployment manifest:

- **Metadata** – standard object's metadata
- **Spec** – specification of the desired behaviour of the Deployment
  - **Replicas** - number of desired pods
  - **Selector** - label selector for pods. Existing ReplicaSets whose pods are selected will be the ones affected by this deployment
  - **Strategy** - describes how to replace existing pods with new ones
    - **Rolling update** - incrementally replace pods with new ones, which are then scheduled on nodes with available resources
      - **MaxSurge** - the number of pods that can be created above the desired amount of pods during an update
  - **Template** - describes the template from which new pods will be created
    - **Spec** - template specifications
      - **Containers** - describes information about creating containers
      - **Image** - describes which docker image should be used to create container
      - **Ports** - describes on which port within the kubernetes cluster service should be exposed
      - **EnvFrom** - describes where to find configuration information like e.g. environmental variables, secrets

5.2.2 Service

Service manifest is an abstraction which defines a logical set of Pods and a policy by which to access them. It enables the pods in a deployment to be accessible from outside the cluster. Figure 3 presents service.yaml file for Gateway service.



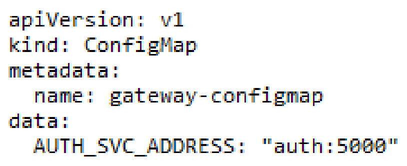
```
apiVersion: v1
kind: Service
metadata:
  name: gateway
spec:
  selector:
    app: gateway
  type: ClusterIP
  ports:
    - port: 8080
      targetPort: 8080
      protocol: TCP
```

Figure 3: Gateway-service.yaml

The specification creates a new Service object named ”gateway”, which targets TCP port 8080 on any Pod with the app=gateway label. Kubernetes assigns an IP address (sometimes called the ”cluster IP”) to the Service, which is used by its proxies. The controller for the Service selector continuously scans for Pods that match its selector, and then POSTs any updates to an Endpoint object also named ”gateway”.

5.2.3 ConfigMap

A ConfigMap is an API object used to store non-confidential data in key-value pairs. Pods can consume ConfigMaps as environment variables, command-line arguments, or as configuration files in a volume. Figure 4 presents ConfigMap manifest for Gateway service.



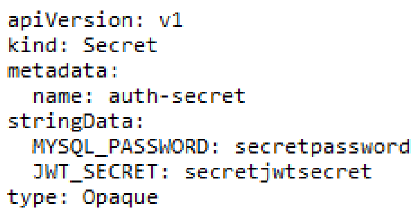
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: gateway-configmap
data:
  AUTH_SVC_ADDRESS: "auth:5000"
```

Figure 4: Gateway-configmap.yaml

This manifest contains an environmental variable AUTH SVC ADDRESS which is used by Gateway service in order to communicate with Authentication service.

5.2.4 Secret

A Secret is an object that contains a small amount of sensitive data such as a password, a token, or a key. Using a Secret means that there is no need to include confidential data in application code. Figure 4 presents Secret manifest for Authentication service.

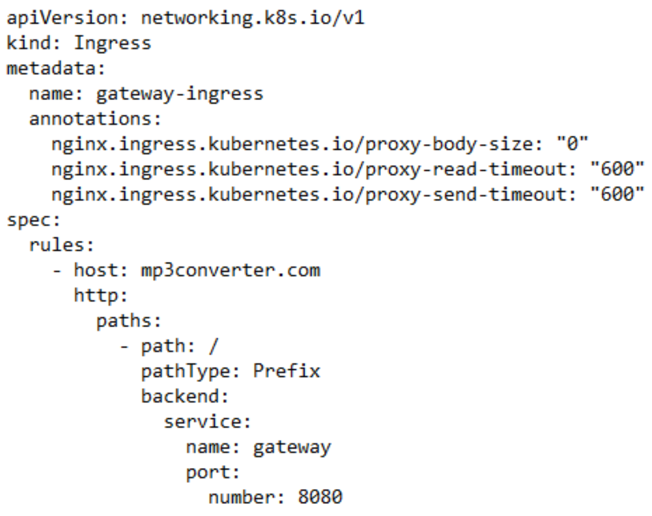


```
apiVersion: v1
kind: Secret
metadata:
  name: auth-secret
stringData:
  MYSQL_PASSWORD: secretpassword
  JWT_SECRET: secretjwtsecret
type: Opaque
```

Figure 5: Auth-secret.yaml

5.2.5 Ingress

Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster. Traffic routing is controlled by rules defined on the Ingress resource. Figure 5 presents ingress manifest for Gateway service.



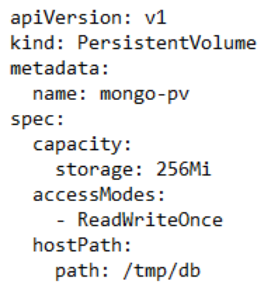
```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: gateway-ingress
  annotations:
    nginx.ingress.kubernetes.io/proxy-body-size: "0"
    nginx.ingress.kubernetes.io/proxy-read-timeout: "600"
    nginx.ingress.kubernetes.io/proxy-send-timeout: "600"
spec:
  rules:
    - host: mp3converter.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: gateway
                port:
                  number: 8080
```

Figure 6: Gateway-ingress.yaml

Nginx annotations define a timeout for e.g. reading a response from the proxied server. Service is available under host http://mp3converter.com.

5.2.6 PersistentVolume

PersistentVolume is a piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using e.g. Storage Classes. It's lifecycle is independent of any individual Pod that uses the PV. The PersistentVolume subsystem provides an API for users and administrators that abstracts details of how storage is provided from how it is consumed. Figure 6 presents PV manifest for storage database service.



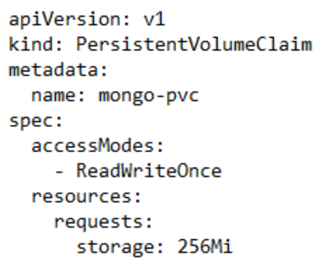
```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mongo-pv
spec:
  capacity:
    storage: 256Mi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /tmp/db
```

Figure 7: mongodb-pv.yaml

This manifests provides a storage volume of 256 MB that is available for mongodb service. The content of this volume persist, even if the MongoDB pod is deleted or moved to a different node. The ReadWriteOnce access mode restricts volume access to a single node, which means it is possible for multiple pods on the same node to read from and write to the same volume.

5.2.7 PersistentVolumeClaim

A PersistentVolumeClaim (PVC) is used to claim/obtain the storage created by PersistentVolume. Figure 7 presents PVC manifest for storage database service.

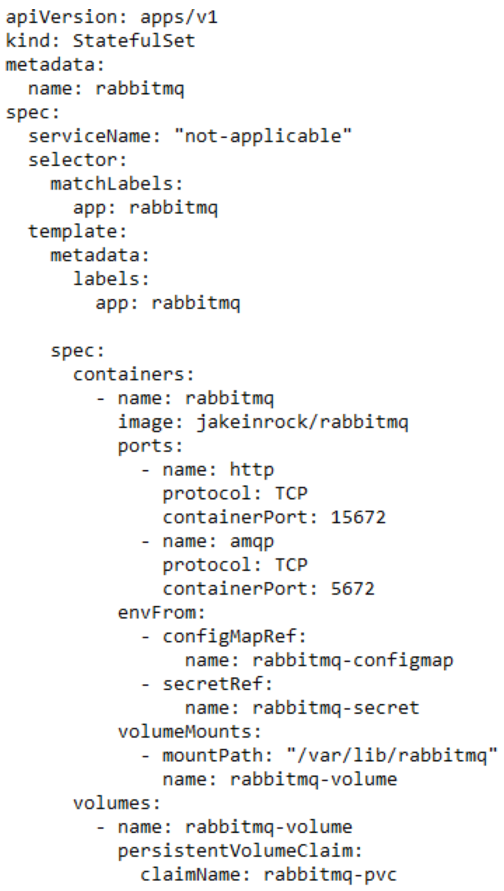


```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mongo-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 256Mi
```

Figure 8: mongodb-pvc.yaml

5.2.8 StatefulSet

StatefulSet manages Pods that are based on an identical container spec. Unlike Deployment, a StatefulSet maintains a sticky identity for each of their Pods. These pods are created from the same spec, but are not interchangeable: each has a persistent identifier maintained across any rescheduling. The persistent Pod identifiers make it easier to match existing volumes to the new Pods that replace any that have failed. FIgure 8 presents StatefulSet manifest for Message Broker service.



```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: rabbitmq
spec:
  serviceName: "not-applicable"
  selector:
    matchLabels:
      app: rabbitmq
  template:
    metadata:
      labels:
        app: rabbitmq

    spec:
      containers:
        - name: rabbitmq
          image: jakeinrock/rabbitmq
          ports:
            - name: http
              protocol: TCP
              containerPort: 15672
            - name: amqp
              protocol: TCP
              containerPort: 5672
          envFrom:
            - configMapRef:
                name: rabbitmq-configmap
            - secretRef:
                name: rabbitmq-secret
          volumeMounts:
            - mountPath: "/var/lib/rabbitmq"
              name: rabbitmq-volume
      volumes:
        - name: rabbitmq-volume
          persistentVolumeClaim:
            claimName: rabbitmq-pvc
```

Figure 9: rabbitmq-statefulset.yaml

Vast majority of manifests content has been already described before. A Volume represents a directory with data that is accessible across multiple containers in a Pod.

6 Implementation

Having explained all types of manifests used in our kubernetes deployment, there is no need to explore all details among all configuration files for each service. The differences concern mostly naming, environmental variables and deployment details. Number of services replicas are presented below:

- Gateway service – 2
- Authentication service – 2
- Converter service – 4
- Notification service – 2
- Message Broker – 1
- Authentication database – 1
- Storage database – 1

Number of replicas varies because of different functionality. Converter service tasks usually last longer than other services because of video files that need to be converted into mp3 file. Gateway, Authentication and Notification services are dealing with less time consuming tasks like generating a token, authorization or sending emails. The rest of services only provides functionality for other services so their number of replicas is equal to one. Figure 8 presents example of service deployment (Gateway).

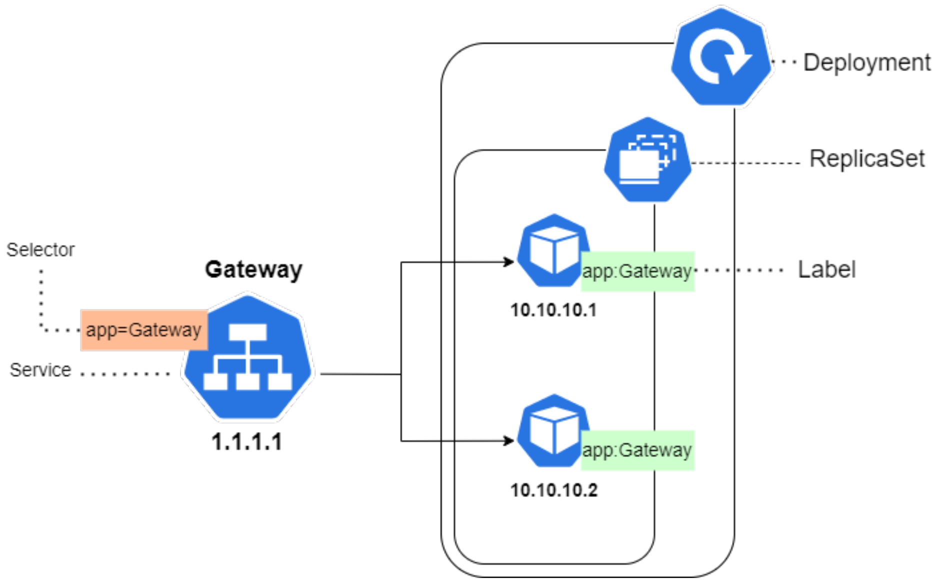


Figure 10: Gateway deployment

7 End-to-end tests

End-to-end testing is a methodology to test the functionality and performance of an application under product-like circumstances and data to replicate live settings. The goal is to simulate what a real user scenario looks like from start to finish.

7.1 Test flow

Test scenario consists of following steps made by user:

- Login to the server
- Uploading movie to the server using generated token
- Downloading file from server using file ID received in email

The python code for test flow:

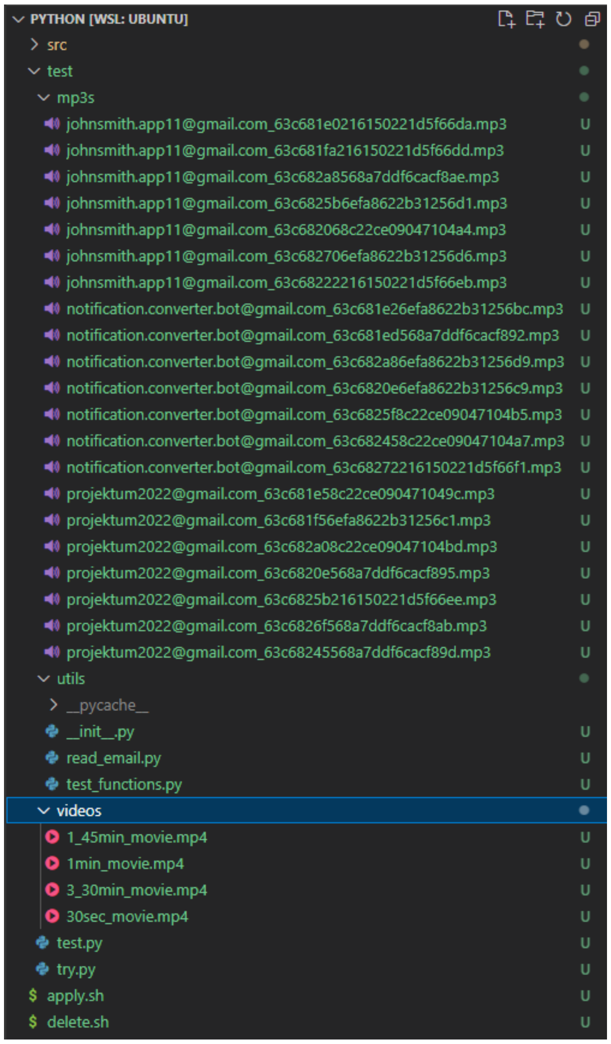
```python
def test_flow(user: str, password: str, g_pass: str):
    u = User(user, password, g_pass)
    token = login(u.name, u.password)
    upload_movie(token, get_video(user), user)

    download_link = None

    while download_link is None:
        download_link = get_download_link(u.name, u.gmail_pass)

    download_mp3(u.name, token, download_link)
    print(f'{user} has successfully downloaded the file.')
```

To make our scenario more demanding for Kubernetes deployment we decided to run our test parallel to replicate live settings when multiple users are sending requests to the application. The code is shown below.

```python
while len([name for name in os.listdir('mp3s')]) < 21:
    with Pool() as pool:
        users = [
            ('notification.converter.bot@gmail.com', os.getenv('PASS_1'),
                os.getenv('GMAIL_PASS_1')),
            ('johnsmith.app11@gmail.com', os.getenv('PASS_2'), os.getenv('GMAIL_PASS_2')),
            ('projektum2022@gmail.com', os.getenv('PASS_3'), os.getenv('GMAIL_PASS_3'))
            ]
        pool.starmap(test_flow, users)
```

7.2 Results

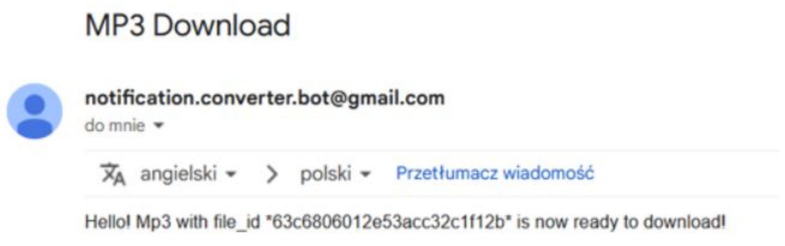
The test was running until the number of files downloaded by all user was at least 20. Users were randomly choosing from four video files with different duration time (30, 60, 105, 210 seconds). Figure 9 presents test results.



```
PYTHON [WSL: UBUNTU]
> src
v test
  v mp3s
    johnsmith.app11@gmail.com_63c681e0216150221d5f66da.mp3          U
    johnsmith.app11@gmail.com_63c681fa216150221d5f66dd.mp3          U
    johnsmith.app11@gmail.com_63c682a8568a7ddf6cacf8ae.mp3          U
    johnsmith.app11@gmail.com_63c6825b6efa8622b31256d1.mp3          U
    johnsmith.app11@gmail.com_63c682068c22ce09047104a4.mp3          U
    johnsmith.app11@gmail.com_63c682706efa8622b31256d6.mp3          U
    johnsmith.app11@gmail.com_63c68222216150221d5f66eb.mp3          U
    notification.converter.bot@gmail.com_63c681e26efa8622b31256bc.mp3  U
    notification.converter.bot@gmail.com_63c681ed568a7ddf6cacf892.mp3  U
    notification.converter.bot@gmail.com_63c682a86efa8622b31256d9.mp3  U
    notification.converter.bot@gmail.com_63c6820e6efa8622b31256c9.mp3  U
    notification.converter.bot@gmail.com_63c6825f8c22ce09047104b5.mp3  U
    notification.converter.bot@gmail.com_63c682458c22ce09047104a7.mp3  U
    notification.converter.bot@gmail.com_63c68272216150221d5f66f1.mp3  U
    projektum2022@gmail.com_63c681e58c22ce090471049c.mp3            U
    projektum2022@gmail.com_63c681f56efa8622b31256c1.mp3            U
    projektum2022@gmail.com_63c682a08c22ce09047104bd.mp3            U
    projektum2022@gmail.com_63c6820e568a7ddf6cacf895.mp3            U
    projektum2022@gmail.com_63c6825b216150221d5f66ee.mp3            U
    projektum2022@gmail.com_63c6826f568a7ddf6cacf8ab.mp3            U
    projektum2022@gmail.com_63c68245568a7ddf6cacf89d.mp3            U
  v utils
    > __pycache__
    __init__.py                                                    U
    read_email.py                                                  U
    test_functions.py                                              U
  v videos
    1_45min_movie.mp4                                              U
    1min_movie.mp4                                                 U
    3_30min_movie.mp4                                              U
    30sec_movie.mp4                                                U
  test.py                                                          U
  try.py                                                           U
$ apply.sh                                                         U
$ delete.sh                                                        U
```

Figure 11: Test results

Figure 10 presents sample email send to user via Notification service when file was successfully converted.



MP3 Download

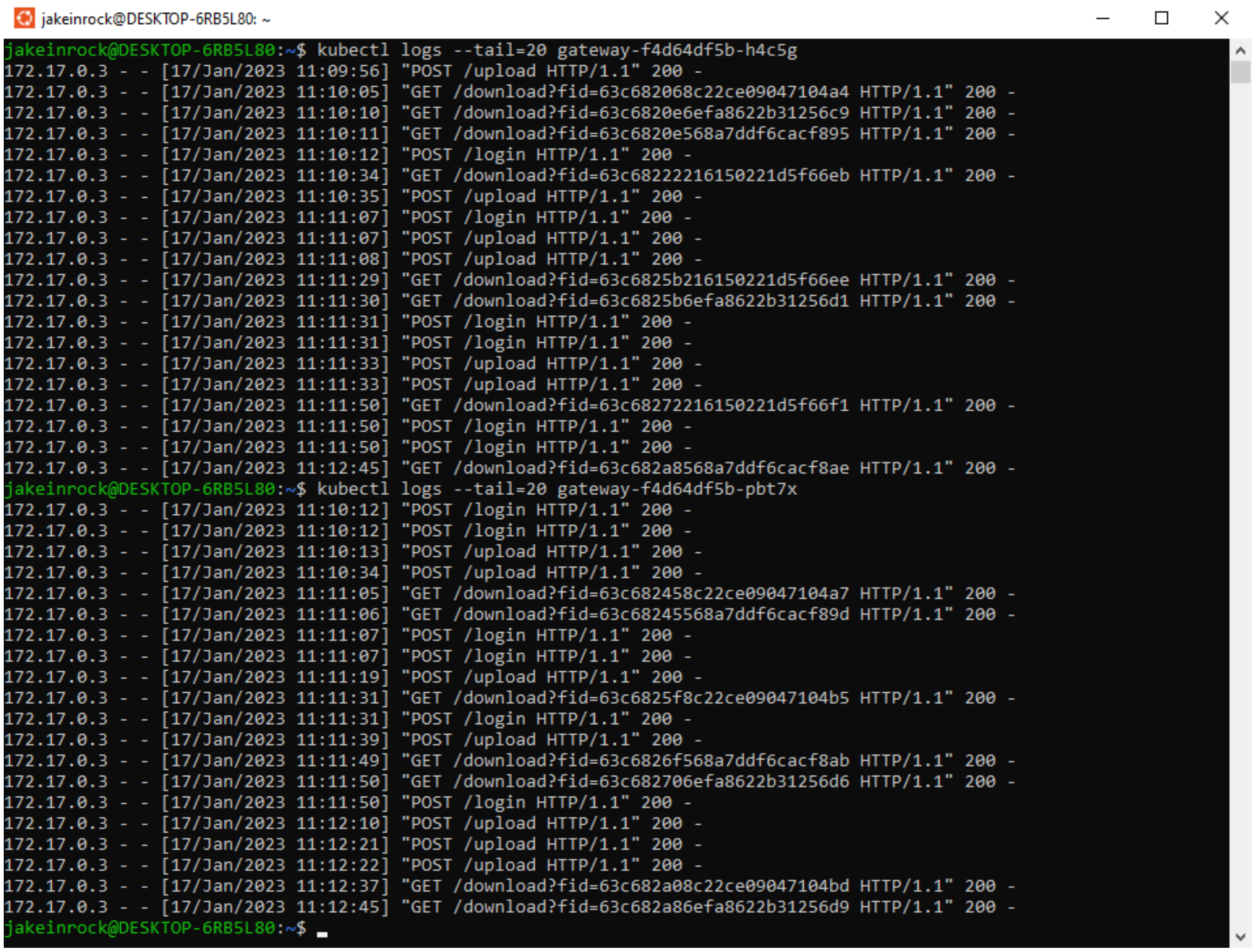
notification.converter.bot@gmail.com
do mnie

Hello! Mp3 with file_id "63c6806012e53acc32c1f12b" is now ready to download!

Figure 12: Sample email received by user

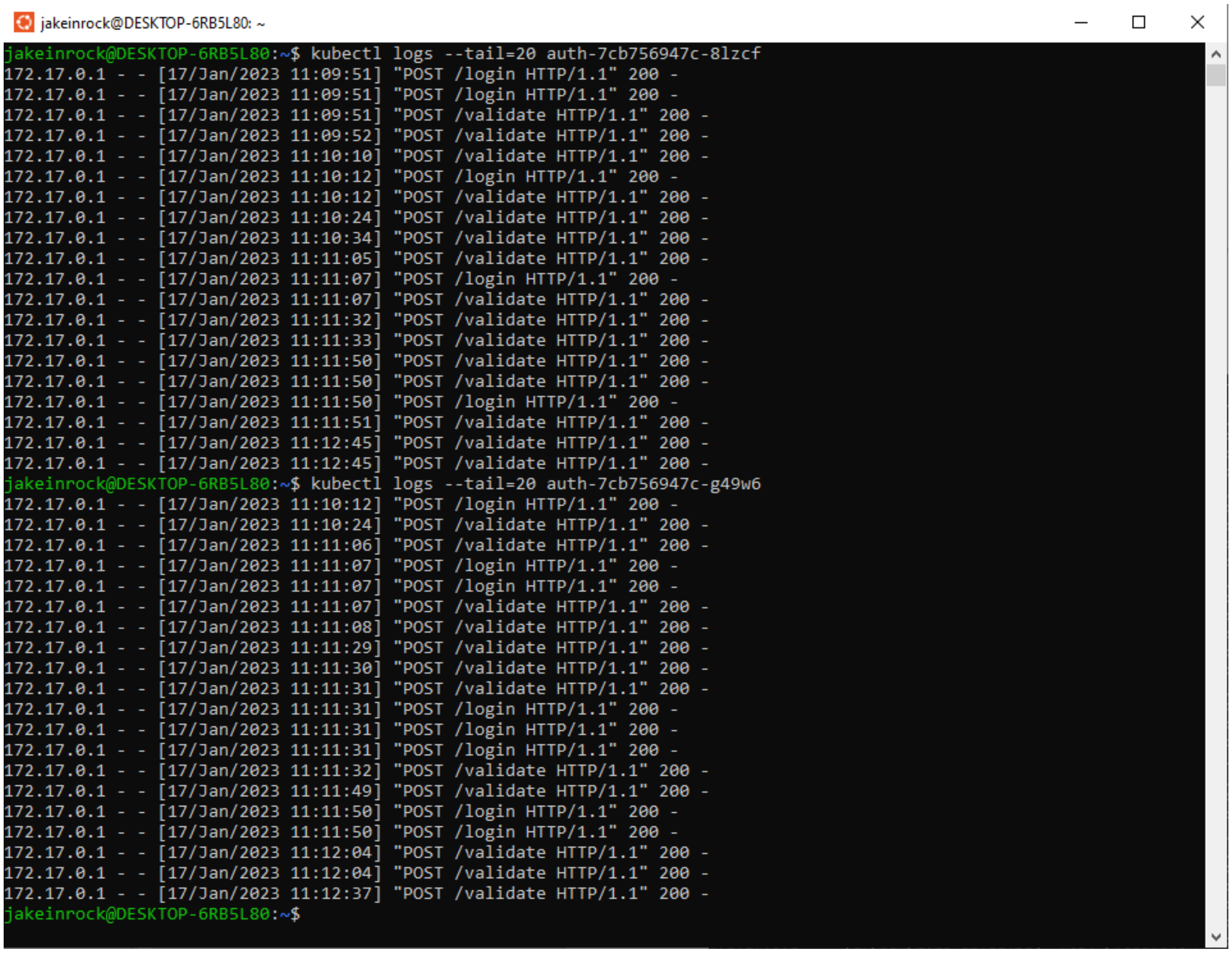
7.2.1 Logs

Below are presented logs from all pods from all services. As can be seen load balancer works properly and all pods are executing tasks.



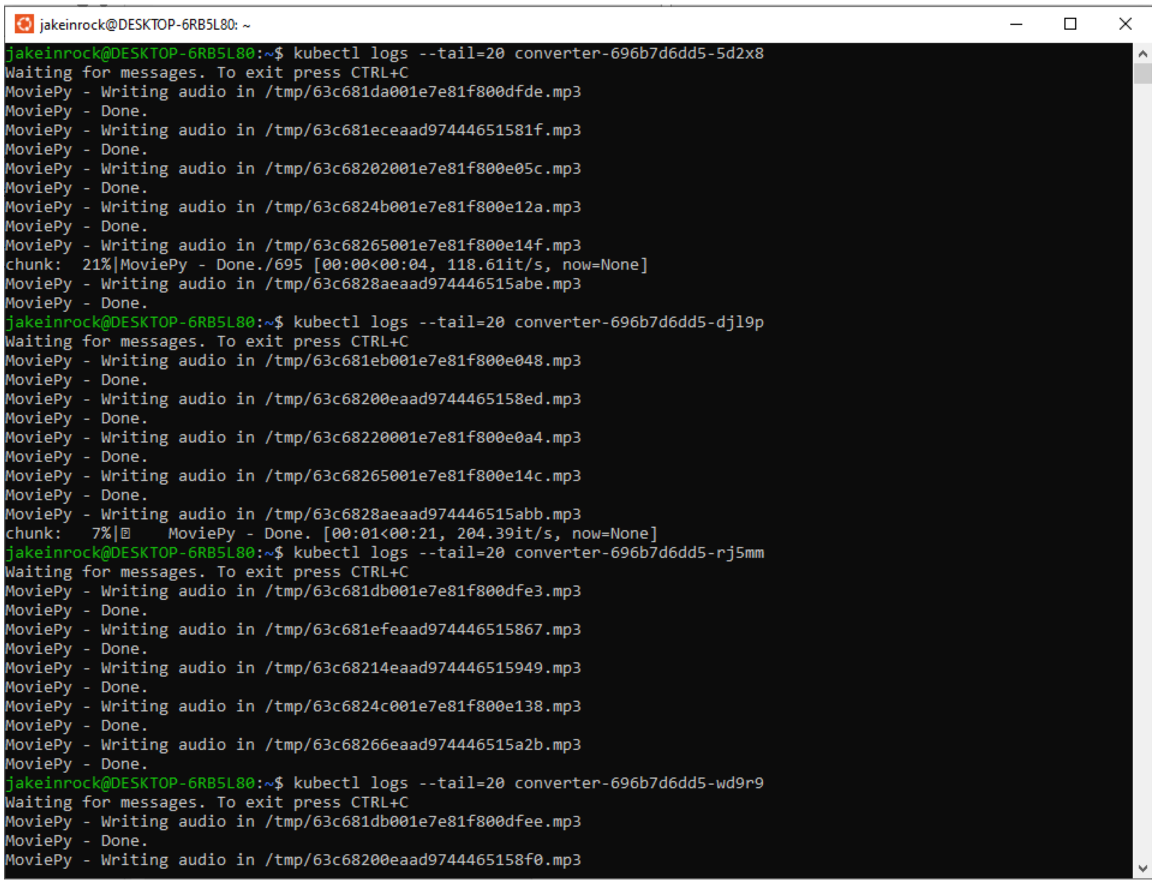
```
jakeinrock@DESKTOP-6RB5L80: ~
jakeinrock@DESKTOP-6RB5L80:~$ kubectl logs --tail=20 gateway-f4d64df5b-h4c5g
172.17.0.3 - - [17/Jan/2023 11:09:56] "POST /upload HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:10:05] "GET /download?fid=63c682068c22ce09047104a4 HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:10:10] "GET /download?fid=63c6820e6efa8622b31256c9 HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:10:11] "GET /download?fid=63c6820e568a7ddf6cacf895 HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:10:12] "POST /login HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:10:34] "GET /download?fid=63c68222216150221d5f66eb HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:10:35] "POST /upload HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:11:07] "POST /login HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:11:07] "POST /upload HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:11:08] "POST /upload HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:11:29] "GET /download?fid=63c6825b216150221d5f66ee HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:11:30] "GET /download?fid=63c6825b6efa8622b31256d1 HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:11:31] "POST /login HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:11:31] "POST /login HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:11:33] "POST /upload HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:11:33] "POST /upload HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:11:50] "GET /download?fid=63c68272216150221d5f66f1 HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:11:50] "POST /login HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:11:50] "POST /login HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:12:45] "GET /download?fid=63c682a8568a7ddf6cacf8ae HTTP/1.1" 200 -
jakeinrock@DESKTOP-6RB5L80:~$ kubectl logs --tail=20 gateway-f4d64df5b-pbt7x
172.17.0.3 - - [17/Jan/2023 11:10:12] "POST /login HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:10:12] "POST /login HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:10:13] "POST /upload HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:10:34] "POST /upload HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:11:05] "GET /download?fid=63c682458c22ce09047104a7 HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:11:06] "GET /download?fid=63c68245568a7ddf6cacf89d HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:11:07] "POST /login HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:11:07] "POST /login HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:11:19] "POST /upload HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:11:31] "GET /download?fid=63c6825f8c22ce09047104b5 HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:11:31] "POST /login HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:11:39] "POST /upload HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:11:49] "GET /download?fid=63c6826f568a7ddf6cacf8ab HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:11:50] "GET /download?fid=63c682706efa8622b31256d6 HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:11:50] "POST /login HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:12:10] "POST /upload HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:12:21] "POST /upload HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:12:22] "POST /upload HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:12:37] "GET /download?fid=63c682a08c22ce09047104bd HTTP/1.1" 200 -
172.17.0.3 - - [17/Jan/2023 11:12:45] "GET /download?fid=63c682a86efa8622b31256d9 HTTP/1.1" 200 -
jakeinrock@DESKTOP-6RB5L80:~$
```

Figure 13: Logs for gateway service



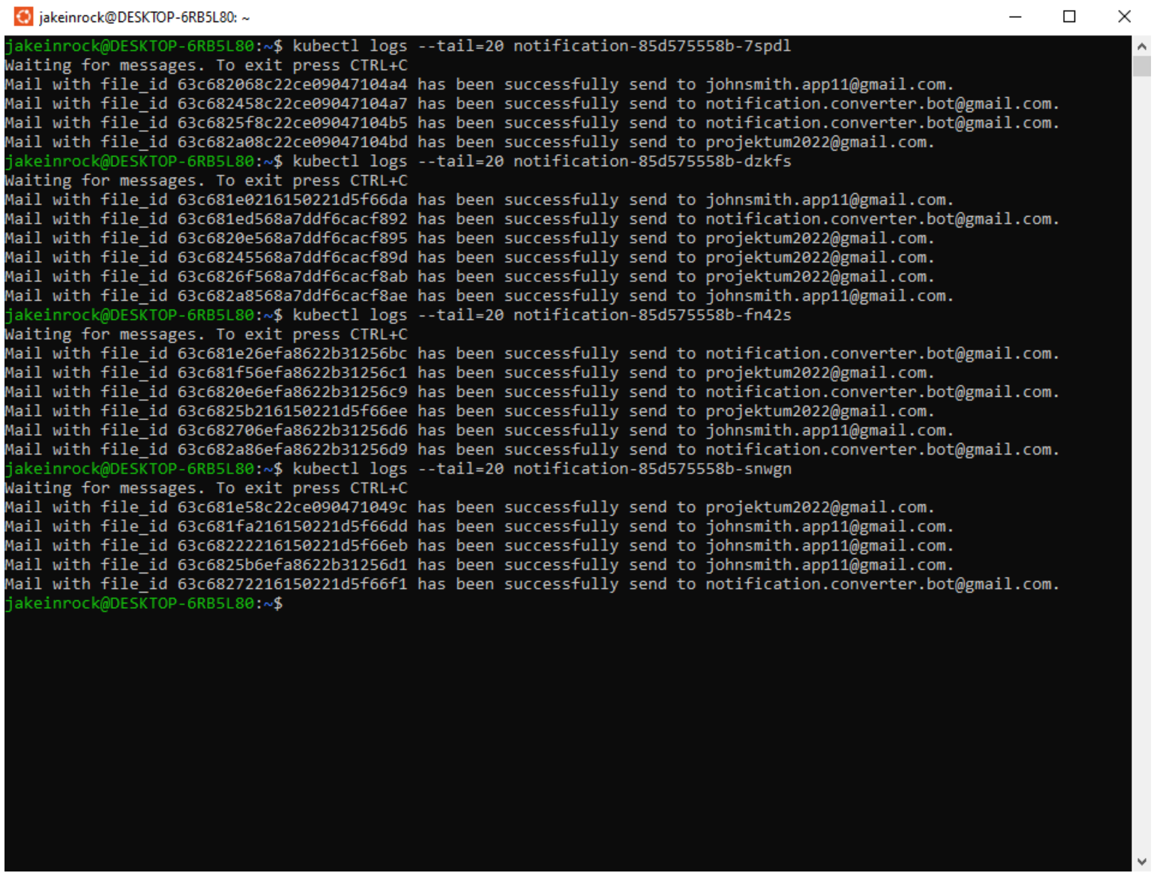
```
jakeinrock@DESKTOP-6RB5L80: ~
jakeinrock@DESKTOP-6RB5L80:~$ kubectl logs --tail=20 auth-7cb756947c-8lzcf
172.17.0.1 - - [17/Jan/2023 11:09:51] "POST /login HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:09:51] "POST /login HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:09:51] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:09:52] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:10:10] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:10:12] "POST /login HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:10:12] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:10:24] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:10:34] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:05] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:07] "POST /login HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:07] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:32] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:33] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:50] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:50] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:50] "POST /login HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:51] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:12:45] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:12:45] "POST /validate HTTP/1.1" 200 -
jakeinrock@DESKTOP-6RB5L80:~$ kubectl logs --tail=20 auth-7cb756947c-g49w6
172.17.0.1 - - [17/Jan/2023 11:10:12] "POST /login HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:10:24] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:06] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:07] "POST /login HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:07] "POST /login HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:07] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:08] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:29] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:30] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:31] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:31] "POST /login HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:31] "POST /login HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:31] "POST /login HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:32] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:49] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:50] "POST /login HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:11:50] "POST /login HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:12:04] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:12:04] "POST /validate HTTP/1.1" 200 -
172.17.0.1 - - [17/Jan/2023 11:12:37] "POST /validate HTTP/1.1" 200 -
jakeinrock@DESKTOP-6RB5L80:~$
```

Figure 14: Logs for authentication service



```
jakeinrock@DESKTOP-6RB5L80:~$ kubectl logs --tail=20 converter-696b7d6dd5-5d2x8
Waiting for messages. To exit press CTRL+C
MoviePy - Writing audio in /tmp/63c681da001e7e81f800dfde.mp3
MoviePy - Done.
MoviePy - Writing audio in /tmp/63c681eceaad97444651581f.mp3
MoviePy - Done.
MoviePy - Writing audio in /tmp/63c68202001e7e81f800e05c.mp3
MoviePy - Done.
MoviePy - Writing audio in /tmp/63c6824b001e7e81f800e12a.mp3
MoviePy - Done.
MoviePy - Writing audio in /tmp/63c68265001e7e81f800e14f.mp3
chunk:  21%|MoviePy - Done./695 [00:00<00:04, 118.61it/s, now=None]
MoviePy - Writing audio in /tmp/63c6828aeaad974446515abe.mp3
MoviePy - Done.
jakeinrock@DESKTOP-6RB5L80:~$ kubectl logs --tail=20 converter-696b7d6dd5-djl9p
Waiting for messages. To exit press CTRL+C
MoviePy - Writing audio in /tmp/63c681eb001e7e81f800e048.mp3
MoviePy - Done.
MoviePy - Writing audio in /tmp/63c68200eaad9744465158ed.mp3
MoviePy - Done.
MoviePy - Writing audio in /tmp/63c68220001e7e81f800e0a4.mp3
MoviePy - Done.
MoviePy - Writing audio in /tmp/63c68265001e7e81f800e14c.mp3
MoviePy - Done.
MoviePy - Writing audio in /tmp/63c6828aeaad974446515abb.mp3
chunk:   7%|▯    MoviePy - Done. [00:01<00:21, 204.39it/s, now=None]
jakeinrock@DESKTOP-6RB5L80:~$ kubectl logs --tail=20 converter-696b7d6dd5-rj5mm
Waiting for messages. To exit press CTRL+C
MoviePy - Writing audio in /tmp/63c681db001e7e81f800dfe3.mp3
MoviePy - Done.
MoviePy - Writing audio in /tmp/63c681efeaad974446515867.mp3
MoviePy - Done.
MoviePy - Writing audio in /tmp/63c68214eaad974446515949.mp3
MoviePy - Done.
MoviePy - Writing audio in /tmp/63c6824c001e7e81f800e138.mp3
MoviePy - Done.
MoviePy - Writing audio in /tmp/63c68266eaad974446515a2b.mp3
MoviePy - Done.
jakeinrock@DESKTOP-6RB5L80:~$ kubectl logs --tail=20 converter-696b7d6dd5-wd9r9
Waiting for messages. To exit press CTRL+C
MoviePy - Writing audio in /tmp/63c681db001e7e81f800dfee.mp3
MoviePy - Done.
MoviePy - Writing audio in /tmp/63c68200eaad9744465158f0.mp3
```

Figure 15: Logs for converter service


jakeinrock@DESKTOP-6RB5L80: ~

```
jakeinrock@DESKTOP-6RB5L80:~$ kubectl logs --tail=20 notification-85d575558b-7spdl
Waiting for messages. To exit press CTRL+C
Mail with file_id 63c682068c22ce09047104a4 has been successfully send to johnsmith.app11@gmail.com.
Mail with file_id 63c682458c22ce09047104a7 has been successfully send to notification.converter.bot@gmail.com.
Mail with file_id 63c6825f8c22ce09047104b5 has been successfully send to notification.converter.bot@gmail.com.
Mail with file_id 63c682a08c22ce09047104bd has been successfully send to projektum2022@gmail.com.
jakeinrock@DESKTOP-6RB5L80:~$ kubectl logs --tail=20 notification-85d575558b-dzkfs
Waiting for messages. To exit press CTRL+C
Mail with file_id 63c681e0216150221d5f66da has been successfully send to johnsmith.app11@gmail.com.
Mail with file_id 63c681ed568a7ddf6cacf892 has been successfully send to notification.converter.bot@gmail.com.
Mail with file_id 63c6820e568a7ddf6cacf895 has been successfully send to projektum2022@gmail.com.
Mail with file_id 63c68245568a7ddf6cacf89d has been successfully send to projektum2022@gmail.com.
Mail with file_id 63c6826f568a7ddf6cacf8ab has been successfully send to projektum2022@gmail.com.
Mail with file_id 63c682a8568a7ddf6cacf8ae has been successfully send to johnsmith.app11@gmail.com.
jakeinrock@DESKTOP-6RB5L80:~$ kubectl logs --tail=20 notification-85d575558b-fn42s
Waiting for messages. To exit press CTRL+C
Mail with file_id 63c681e26efa8622b31256bc has been successfully send to notification.converter.bot@gmail.com.
Mail with file_id 63c681f56efa8622b31256c1 has been successfully send to projektum2022@gmail.com.
Mail with file_id 63c6820e6efa8622b31256c9 has been successfully send to notification.converter.bot@gmail.com.
Mail with file_id 63c6825b216150221d5f66ee has been successfully send to projektum2022@gmail.com.
Mail with file_id 63c682706efa8622b31256d6 has been successfully send to johnsmith.app11@gmail.com.
Mail with file_id 63c682a86efa8622b31256d9 has been successfully send to notification.converter.bot@gmail.com.
jakeinrock@DESKTOP-6RB5L80:~$ kubectl logs --tail=20 notification-85d575558b-snwgn
Waiting for messages. To exit press CTRL+C
Mail with file_id 63c681e58c22ce090471049c has been successfully send to projektum2022@gmail.com.
Mail with file_id 63c681fa216150221d5f66dd has been successfully send to johnsmith.app11@gmail.com.
Mail with file_id 63c68222216150221d5f66eb has been successfully send to johnsmith.app11@gmail.com.
Mail with file_id 63c6825b6efa8622b31256d1 has been successfully send to johnsmith.app11@gmail.com.
Mail with file_id 63c68272216150221d5f66f1 has been successfully send to notification.converter.bot@gmail.com.
jakeinrock@DESKTOP-6RB5L80:~$
```

Figure 16: Logs for notification service