Gestão de Infraestruturas de Computação

The News Project

Pedro Sobral, 98491
Ricardo Rodriguez, 98388
Tiago Coelho, 98385

15/06/2023

Index

1 - Introduction

The design and development of effective and scalable news platforms have become more crucial as online news platforms continue to be popular among readers. One of the main challenges in developing such a platform is managing the underlying computing infrastructure to guarantee high availability and constant performance. Managing huge web applications by using container orchestration technologies like Kubernetes has grown in popularity in this setting.

Our project's goal is to create a comprehensive news platform using Kubernetes to manage the underlying infrastructure. With four main user types - Administrators, Writers, Readers and Anonymous - the platform will serve a wide range of user demographics.

Administrators are in charge of editing postings, overseeing other users and their groups, creating and editing topics and also overseeing comments. Readers will be able to read and comment on postings, while Writers will develop and submit articles for approval. Anonymous are normal readers, not logged in, and can only read news.

The implementation and administration of the news platform utilizing Kubernetes will be covered in this study. We'll go through the platform's architecture, along with how it's deployed in a Kubernetes cluster and containerized. Along with a discussion of redundancy and failover procedures, we will also go through the platform's replication and recovery plan. In addition, we'll look at the bottlenecks that could occur during platform implementation, like resource limitations and network congestion. We will also go over the platform's monitoring, which is crucial for preserving high availability and reliability.

This report's overall objective is to give a thorough overview of the deployment and administration of a news platform utilizing Kubernetes. We seek to offer insights into the creation and deployment of a reliable and scalable news platform in a cloud infrastructure using Kubernetes by examining the many aspects of the project.

2 - Our Idea

Our concept offers a modern news online platform that is user-friendly using cutting-edge technology. Users will be able to browse news articles divided into sections including sports, politics, and technology. Additionally, they may comment and like them.

To implement our news web platform, we will be using several technologies that have been carefully selected to meet our requirements.

Firstly, we will use:

- Traefik for reverse proxy and load balancing, which will allow us to easily manage incoming requests and distribute them efficiently across multiple instances of our platform;
- Angular will be used for our frontend/web client to deliver a seamless and responsive user interface;
- For our backend, we have chosen Django, a robust and scalable web framework that provides built-in tools for authentication, authorization, and other core functionalities. The Django component consists of a RESTful API to perform CRUD operations over all the system's entities and connects to the Redis cache (to update frequently accessed news) and the MongoDB database (persistent storage), which will be explained below;
- Redis will be implemented as our cache, it's an in-memory key-value database with sub-millisecond latency which will be used to improve the platform performance by retrieving frequently accessed data (news) without the need to talk to the backend or the database;
- We will use MongoDB as our database to store all the system entities. When a news article is never accessed or isn't popular among users, the cache request for that specific article will be a miss, and, thus, the client will communicate to the API, which will, in turn, request the article to the MongoDB database for retrieval;
- Prometheus and Grafana will be used to monitor, store and visualize metrics, thus giving a real-time input on how well our platform is working;
- Kubernetes, a potent container orchestration engine that will assist and guarantee high availability and scalability of our news platform, to handle the deployment of our containers.

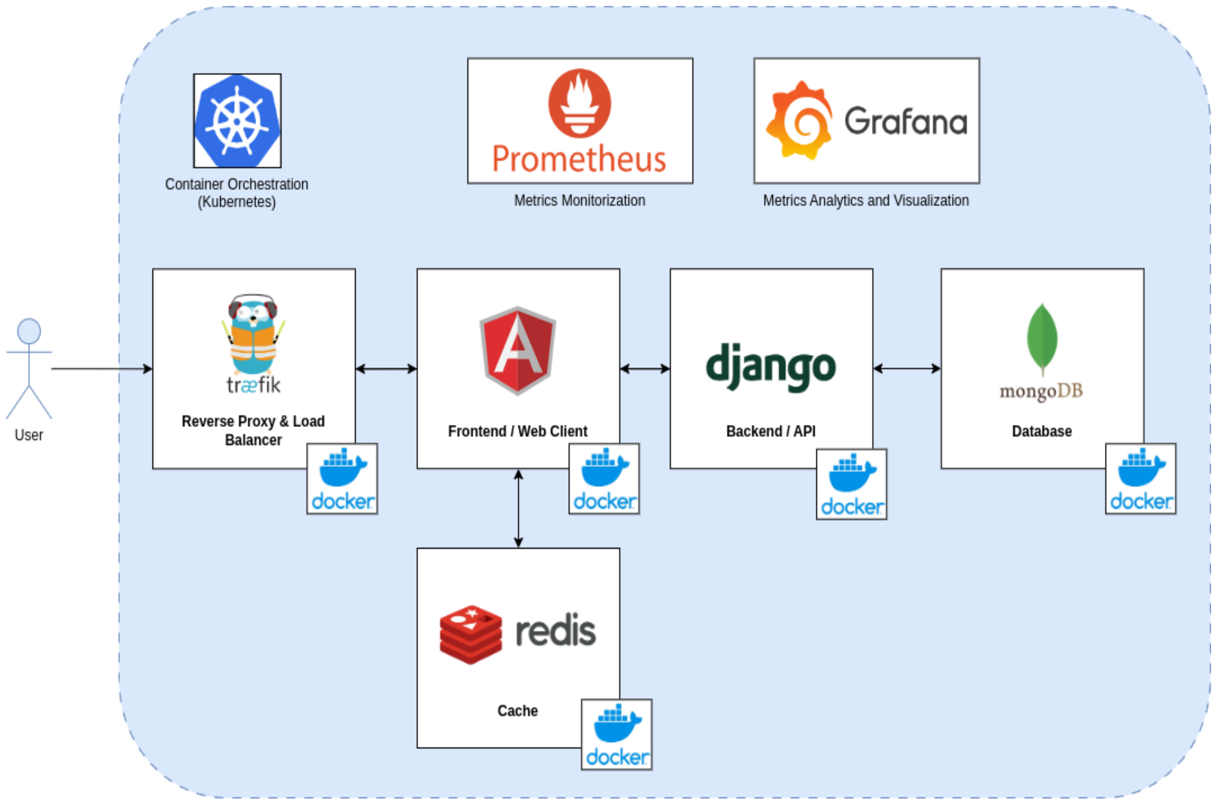


Figure 1 - Project's architecture and used technologies

Our platform's text editor, which authors may use to produce and submit their articles, is one of its primary advantages. This function will enable authors to produce more interesting, engrossing and formatted content for readers.

We have incorporated an approval process for publication submission, where authors submit their articles for review by administrators before they are made accessible for public viewing. This is done to ensure that the publications adhere to the highest standards. Thus, an article can have 3 possible states: approved (visible to everyone), pending (needs approval from higher-power entities) and archived (not visible but stored for archival purposes).

The administrator interface for the platform will make it possible to control all platform entities, such as publications, publication topics, and users. A dashboard will be available to the administrator, where they may view users, all of the articles, and other information.

We will also provide sections with the most recent publications and the most popular articles over a given time period in order to enhance the user experience. Readers will find it simpler to stay current on news and find popular articles thanks to this feature.

We have chosen technologies that will guarantee the platform's scalability and availability, such as Kubernetes for container orchestration. With Kubernetes, the platform can be managed and scaled quickly and reliably available during times of high demand.

Our goal is to build a News online platform that gives a pleasurable user experience while also providing sophisticated functionality to meet the needs of today's readers. We are certain that our choice of cutting-edge technology will enable us to achieve this goal while also assuring that the platform can scale, maintain availability, and offer reliable performance.

3 - Requirements gathering

Requirements gathering is a crucial stage in the development process, as it helps ensure that the end product meets the needs and expectations of its intended users. For our news web platform project, we gathered both functional and non-functional requirements.

In our case, the main functional requirements include that the platform must be web-based, must have a frontend, backend, and database, and must incorporate caching functionality.

Apart from the normal (frontend, backend, database) we determined that Redis would be an ideal choice for implementing the cache, to store frequently accessed data, such as recent and most viewed articles, to reduce database access times.

Hypothetically, if we wanted to deploy our app worldwide, we would need to take care to block traffic to countries that cannot receive certain content, for example, we need a cache for Portugal and another for China, to comply with local regulations and legal requirements, avoiding content with cultures, material critical of the Chinese Communist Party, socially or politically sensitive content, etc.

Other functional requirements are:

- User Registration and Authentication: To access the Platform services (likes, and comments), users must first register with the Platform and create an account. A user must be able to securely log in using their credentials and remain authenticated throughout her browser session.

- News Articles: The platform should allow writers to post news articles with headlines, full-length text and a topic. Users should be able to read, like and comment on articles.

- Search Functionality: Users should be able to search for articles by keyword, topic, author, and date.

- Commenting System: Users should be able to comment on articles and engage in discussions with other users. Commenting should be moderated to ensure appropriate content.

- Admin Panel: The administrators are in charge of making sure that all material and communication flow is of a high standard and is suitable. All news articles must be checked and assessed before publication to make sure they are factually accurate, objective, and compliant with the platform's standards and guidelines. Administrators should often review comments material to ensure there are no offensive or inappropriate remarks.

In our project, non-functional requirements include the need for the platform to be highly available, scalable, and reliable.

A strong infrastructure that can manage heavy traffic and sustain performance even during peak usage is one of the essential needs. To ensure that no server is overloaded, we must additionally create a load-balancing mechanism to split traffic across several servers. To address this, we've put Traefik, a reverse proxy and load balancer, into place, which can disperse traffic among several servers and keep any one server from becoming overloaded.

Another important factor is ensuring that the platform is fault tolerant. This means that it continues to function even if a component fails. To do this, we must build redundancy into our system, such as by using numerous replicas in the different components.

To do this, we chose Kubernetes for container orchestration. This allows us to set up and manage multiple copies of our service to improve availability and maintain functionality even if some component fails. To maintain data integrity and availability, we also deployed a Redis cache and a MongoDB database with built-in replication and clustering capabilities.

We must concentrate on monitoring and logging in addition to infrastructure and fault tolerance to swiftly detect and resolve any potential problems. Implementing technologies that can proactively monitor our systems and warn us of any possible problems before they become serious is necessary to achieve this.

Prometheus and Grafana, which offer real-time analytics and visualization for our systems, have been integrated as a result. This enables us to anticipate possible problems and deal with them before they become serious.

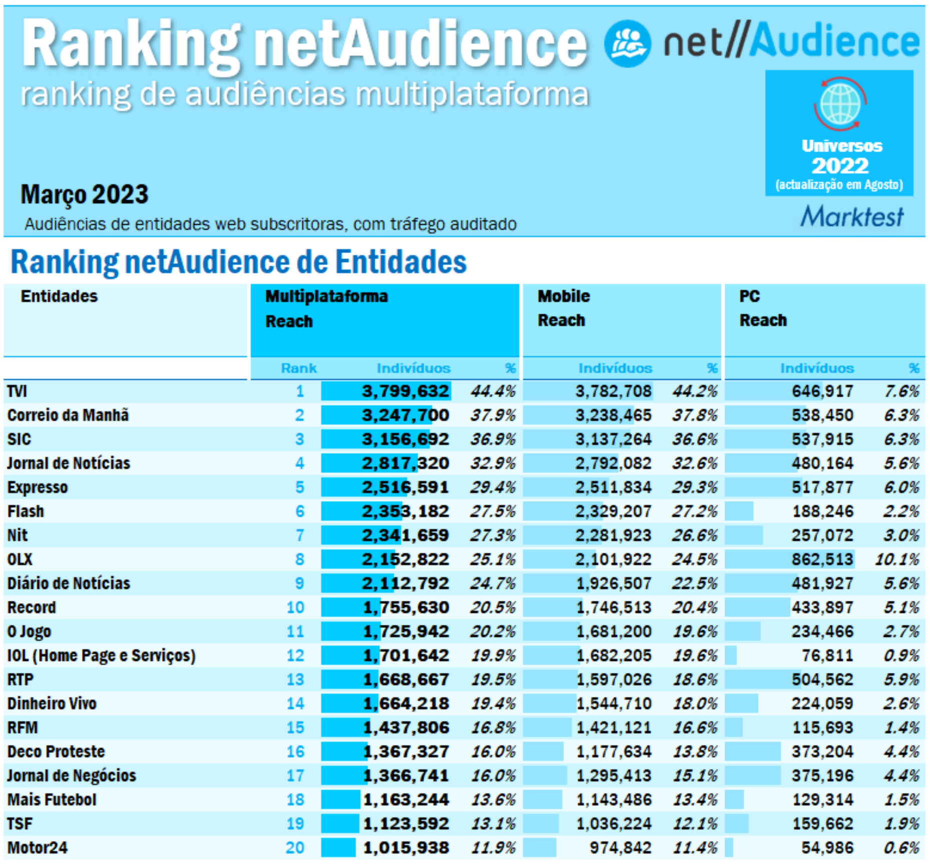
Finally, we must design a thorough backup and recovery strategy to guarantee that our platform is highly available and dependable. This includes a routine for backing up our caches and databases as well as a strategy for promptly restoring services in the case of an emergency.

We can quickly satisfy the demands for a highly accessible, scalable, and dependable platform by putting these technologies and approaches into practice. Our infrastructure and systems must be fault tolerant and highly available to handle heavy traffic and maintain performance during periods of high usage. This ensures our users have a positive experience on our platform and gives them the confidence they need to continue to provide a reliable service.

4 - Users and their management

User base is one of the most important factors to consider when building and growing your web platform. Understanding their requirements and expectations is key to creating a platform that not only meets your business goals, but also provides a great user experience.This chapter will go through the anticipated user base and how to manage them on our platform.

With an anticipated 100,000 users each day, our platform is built for a sizable user base. We must carefully evaluate how people are handled and interacted with if we want to provide a seamless experience to them. This number of 100,000 users per day is based on the March 2023 netAudience ranking published by Marktest, which contains the audiences of subscribing web entities, with audited traffic for that month. In this ranking (Figure 2) we can see that the main news organizations on the web have between 3 (Correio da Manhã, Jornal de Notícias) and 2 (Diário de Notícias, Flash) million users during the month. These values give between 70,000 to 100,000 users per day. Taking these values into account, we predict the value of 100,000 users per day for our news platform.



Ranking netAudience net//Audience

ranking de audiências multiplataforma

Universos 2022 (actualização em Agosto)

Março 2023

Audiências de entidades web subscritoras, com tráfego auditado

Marktest

Ranking netAudience de Entidades

Entidades	Multiplataforma Reach			Mobile Reach		PC Reach	
	Rank	Indivíduos	%	Indivíduos	%	Indivíduos	%
TVI	1	3,799,632	44.4%	3,782,708	44.2%	646,917	7.6%
Correio da Manhã	2	3,247,700	37.9%	3,238,465	37.8%	538,450	6.3%
SIC	3	3,156,692	36.9%	3,137,264	36.6%	537,915	6.3%
Jornal de Notícias	4	2,817,320	32.9%	2,792,082	32.6%	480,164	5.6%
Expresso	5	2,516,591	29.4%	2,511,834	29.3%	517,877	6.0%
Flash	6	2,353,182	27.5%	2,329,207	27.2%	188,246	2.2%
Nit	7	2,341,659	27.3%	2,281,923	26.6%	257,072	3.0%
OLX	8	2,152,822	25.1%	2,101,922	24.5%	862,513	10.1%
Diário de Notícias	9	2,112,792	24.7%	1,926,507	22.5%	481,927	5.6%
Record	10	1,755,630	20.5%	1,746,513	20.4%	433,897	5.1%
O Jogo	11	1,725,942	20.2%	1,681,200	19.6%	234,466	2.7%
IOL (Home Page e Serviços)	12	1,701,642	19.9%	1,682,205	19.6%	76,811	0.9%
RTP	13	1,668,667	19.5%	1,597,026	18.6%	504,562	5.9%
Dinheiro Vivo	14	1,664,218	19.4%	1,544,710	18.0%	224,059	2.6%
RFM	15	1,437,806	16.8%	1,421,121	16.6%	115,693	1.4%
Deco Proteste	16	1,367,327	16.0%	1,177,634	13.8%	373,204	4.4%
Jornal de Negócios	17	1,366,741	16.0%	1,295,413	15.1%	375,196	4.4%
Mais Futebol	18	1,163,244	13.6%	1,143,486	13.4%	129,314	1.5%
TSF	19	1,123,592	13.1%	1,036,224	12.1%	159,662	1.9%
Motor24	20	1,015,938	11.9%	974,842	11.4%	54,986	0.6%

Figure 2 - Ranking netAudience Março 2023

On our platform, there are four basic categories of users: Anonymous, Reader, Writer, and Admin. The following lists the many rights and capabilities that each type of user has:

- Anonymous:

Anonymous users are normal readers who are not logged into the platform. While they may browse and read news stories, they cannot interact with them in any way. They are unable to contribute, like, or comment on news stories.

- Reader:

Registered readers who have logged in to our platform have more capabilities than anonymous users. They may like and comment on news stories in addition to reading them. They can also search for news by title, author, date or topic.

- Writer:

Users who may post articles to our site are known as writers. Prior to beginning to post articles, individuals must first register as users and have admin approval. Once approved, writers can submit articles for review and publishing.

- Admin:

Admins have the highest level of permissions on our platform. They are responsible for managing news, writers, users, topics, and comments. Admins can approve or reject article submissions from writers, moderate comments, and manage user accounts. They can also create and manage topics, which are used to categorize news articles on our platform.

5 - Deployment architecture

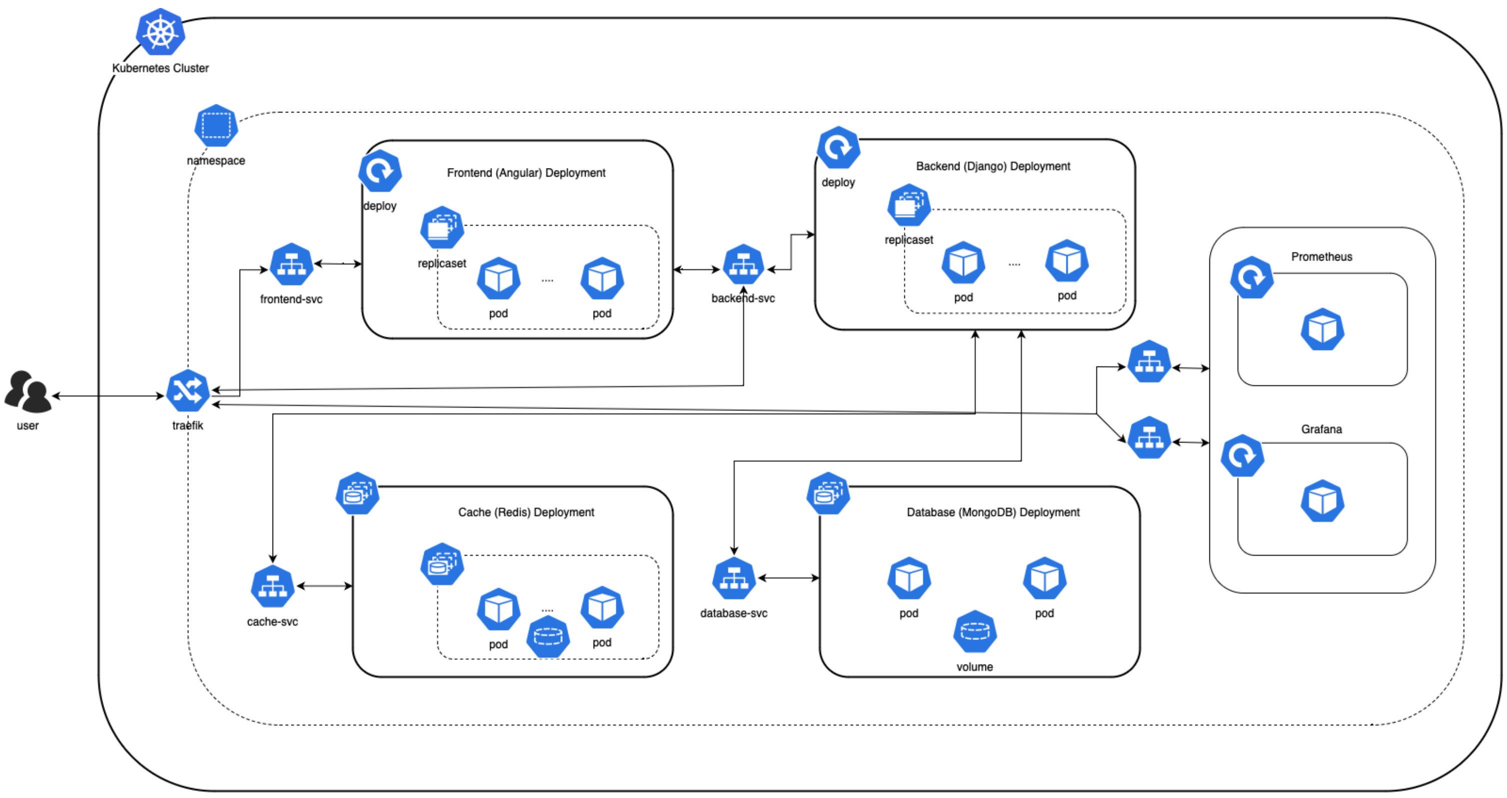


Figure 3 - Architecture

About our Deployment architecture, it is divided into 5 main modules: backend, frontend, cache, database, load balancer, and Prometheus/Grafana, all of them within our namespace: gic-group-6.

Starting with the backend, a deployment is performed where we retrieve the Docker image of our Django API from the registry. By default, we have one replica with a limit of half a CPU and 128 MB of memory. We also have a Service to allow communication between the backend and other modules, with the service exposed on port 8000.

For the frontend, a deployment is done where we also retrieve the Docker image of our Angular frontend. By default, we have one replica with a limit of half a CPU and 128 MB of memory. We have a Service to communicate with our Load Balancer, with the service exposed on port 4200.

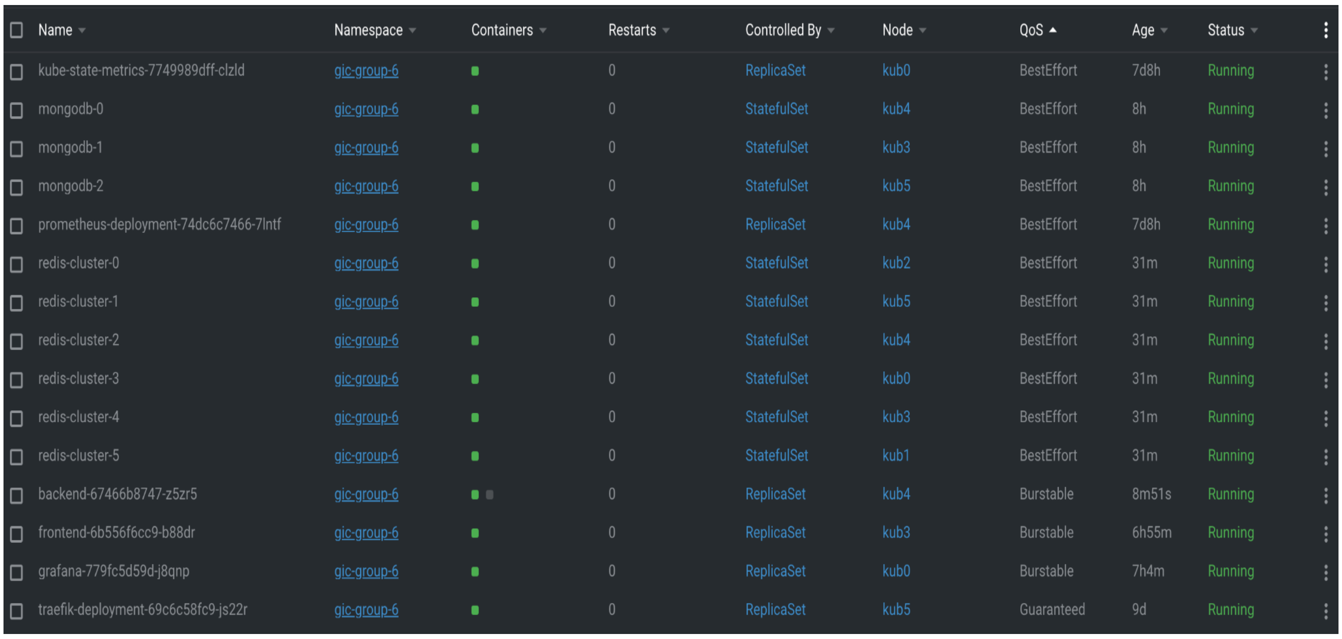
Regarding the cache, we create a StatefulSet and use an Alpine-based Redis Docker image. The goal is to create a Redis cluster with masters and slaves, following the cluster concept. Initially, we start with 6 replicas, 3 of them being masters and the other 3 being slaves. A Redis cluster uses data sharding, which means data is clustered across 3 partitions, each one having a master instance and a slave instance, in this specific case. However, if there’s a need to scale the number of Redis replicas, new slave instances can join the cluster and connect to a master node. The Pod volumes have a size of 1 GB, and the cluster is exposed through a Service on port 6379.

As for the database, we create a StatefulSet and use a Bitnami-provided MongoDB Docker image. This image allows for easier replication configuration to create a MongoDB cluster. The strategy here is to have one Primary and two Secondary nodes (by default), with the possibility to increase the number of Secondaries. All volumes have a size of 1 GB, and the service can be accessed through a Service on port 27017.

Both the database and cache are deployed using a StatefulSet, as this deployment type allows us to have a specific order of pod creation, and the Pod name and identifier are the same.

For load balancing and reverse proxy purposes, we are using Traefik, creating an Ingress to access the content within the cluster via a URL. All modules we want to access through the browser need to be configured in Traefik with redirection to the respective Service. We have two different hostname rules for Traefik. If the host has the http://gic-group-6.k3s/ prefix, the user will be redirected to the frontend/client service, and if the host prefix is http://django.gic-group-6.k3s/, the user will be redirected to the backend/API.

Finally, we are using Prometheus/Grafana for monitoring. The Grafana can be accessed through the following link http://grafana.group-6.k3s. The prometheus of the group can be accessed here: http://prometheus.group-6.k3s that will be used to take mainly latency metrics, and the http://prometheus.deti to take more metrics about the pods (CPU, RAM, network, etc).



Name	Namespace	Containers	Restarts	Controlled By	Node	QoS	Age	Status
kube-state-metrics-7749989dff-clzld	gic-group-6	🟩	0	ReplicaSet	kub0	BestEffort	7d8h	Running
mongodb-0	gic-group-6	🟩	0	StatefulSet	kub4	BestEffort	8h	Running
mongodb-1	gic-group-6	🟩	0	StatefulSet	kub3	BestEffort	8h	Running
mongodb-2	gic-group-6	🟩	0	StatefulSet	kub5	BestEffort	8h	Running
prometheus-deployment-74dc6c7466-7lntf	gic-group-6	🟩	0	ReplicaSet	kub4	BestEffort	7d8h	Running
redis-cluster-0	gic-group-6	🟩	0	StatefulSet	kub2	BestEffort	31m	Running
redis-cluster-1	gic-group-6	🟩	0	StatefulSet	kub5	BestEffort	31m	Running
redis-cluster-2	gic-group-6	🟩	0	StatefulSet	kub4	BestEffort	31m	Running
redis-cluster-3	gic-group-6	🟩	0	StatefulSet	kub0	BestEffort	31m	Running
redis-cluster-4	gic-group-6	🟩	0	StatefulSet	kub3	BestEffort	31m	Running
redis-cluster-5	gic-group-6	🟩	0	StatefulSet	kub1	BestEffort	31m	Running
backend-67466b8747-z5zr5	gic-group-6	🟩 ⬜	0	ReplicaSet	kub4	Burstable	8m51s	Running
frontend-6b556f6cc9-b88dr	gic-group-6	🟩	0	ReplicaSet	kub3	Burstable	6h55m	Running
grafana-779fc5d59d-j8qnp	gic-group-6	🟩	0	ReplicaSet	kub0	Burstable	7h4m	Running
traefik-deployment-69c6c58fc9-js22r	gic-group-6	🟩	0	ReplicaSet	kub5	Guaranteed	9d	Running

Figure 4 - Kubernetes Pods

6 - Deployment and replication strategy

Regarding the replication strategy, we take into consideration some aspects. We include RollingUpdate in the configuration files to ensure that updates happen gradually across the replicas, guaranteeing that the service remains functional.

Currently, it is not yet defined, but the goal is to establish criteria that allow for horizontal scaling of our application, increasing or decreasing the number of replicas for each service. It is also planned to implement health checks to gather more information about the state of each service at regular intervals.

To secure our platform, we use Kubernetes Secrets to avoid the exposure of confidential data, specially regarding the databases authentication credentials, in our application code.

This helps ensure that there is no single point of failure and that the pods are distributed evenly for optimal performance.

Another good practice is to use the *topologySpreadConstraints* strategy to ensure that pods are not scheduled on the same node in the Kubernetes cluster. Thus, this strategy spreads out pods across different nodes in a Kubernetes cluster, based on their labels and the topology of the nodes in the cluster. This, in turn, increases deployment redundancy, availability and helps ensure that there is no single point of failure and that the pods are distributed evenly for optimal performance.

This topic also encompasses monitoring metrics for the entire application using Prometheus and Grafana. Additionally, a system of alerts will be implemented in Grafana to notify developers in case of critical failures.

With all these implementations, we will have a scalable, resilient, and robust application, which is ultimately the desired outcome.

The components that offer the greatest replication capacity are the Redis and MongoDB clusters.

7 - Evaluate the cluster operation

We performed some load tests. We wrote a Python script that uses the requests library to make requests to the respective endpoints. To different pages in frontend, which consequently calls backend and cache resources. We ran the script in 5 different terminals, resulting in a total of approximately 100 requests per second (+/- 60 frontend, +/- 40 backend). The service in general responds well, as we now have autoscaling (will be discussed further). We also performed other tests using Lens, such as "killing" certain pods, and they automatically "resurrected." (As it shows in the section 15 - Demo).

8 - Redundancy and Recovery strategy

The main way we achieve redundancy in our application is through Pod replication. It is also an essential aspect of ensuring fault tolerance and increasing the overall availability of the service in a more general sense.
Regarding the recovery strategy, it is important to take action when, for example, a node or a set of Pods fails. In such cases, the failed components are recreated. This can be achieved by setting a minimum number of replicas for a particular service, which forces the system to create replicas to meet the predefined minimum.
Once again, monitoring the entire system is advantageous. By using Prometheus/Grafana, we can define rules to detect anomalies and set up alerts to receive notifications about any potential issues that may occur.

A good practice to reduce system or service failures is to conduct contingency tests. These tests simulate the failure of certain parts of the system to observe how the system as a whole reacts. This provides a broader understanding and helps identify and resolve any potential problems that may arise from the failure of specific systems. Implementing a system for regular backups is a great recovery strategy. In the event of a catastrophe, we would not lose all the information because it is stored in the backup service. These services primarily store Pods, configurations, and volumes. Velero is an example of such a tool for Kubernetes backups.

9 - Bottlenecks

Initially, we expected to encounter some bottlenecks, especially when it came to resource access, as the Kubernetes cluster is being used by multiple groups. This limited our ability to horizontally scale our application across its various services, as the performance of the cluster was severely affected throughout the project’s development. We notice that the latency of our services, especially the backend, has increased in the last weeks (we can notice it through Grafana, and our Alert Manager).

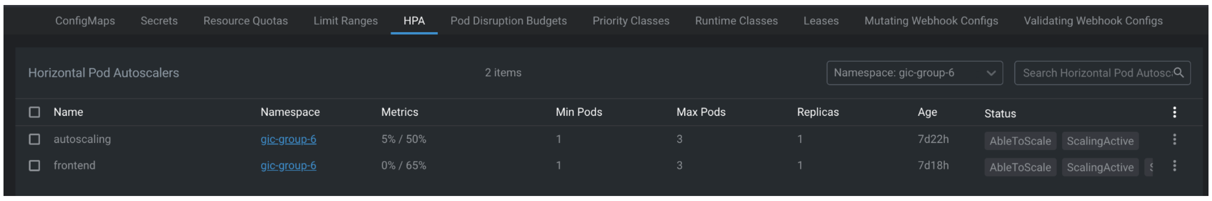
Additionally, we had to analyze and choose the best Redis solution for our case: Redis Sentinel or Redis Cluster. We’ve stuck to Redis Cluster, since it splits data among multiple nodes based on a hash slot, it’s highly scalable and supports automatic failover. We’ve also used a strategy to make new Redis pods join the cluster automatically when ready, avoiding the need to manually configure each pod to join the Redis Cluster.

10 - Autoscaling

Regarding Autoscaling, it is implemented in the Backend and Frontend, using HPA (Horizontal Pod Autoscaling).

In the case of the Backend, the autoscaling configuration is defined that the minimum of Pods is 1 and the maximum is 3, and a Pod is automatically created whenever the CPU is above 50% (of the CPU established in the configuration file in the Deployment from Backend). HPA reduces the number of Pods to a minimum of 1, when the CPU is 5 minutes below the established limit (50%).

In the case of Frontend, the autoscaling configuration is also defined for a minimum of 1 and a maximum of 3 Pods, and in this case a new Pod is created whenever the CPU is above 65% utilization (the CPU established in the file Frontend Deployment Configuration Setup). The Autoscaling configuration has defined that if the CPU is 5 minutes below the 65% threshold, the number of Pods will decrease to the established minimum, which in this case is 1.



Horizontal Pod Autoscalers 2 items

Name	Namespace	Metrics	Min Pods	Max Pods	Replicas	Age	Status
autoscaling	gic-group-6	5% / 50%	1	3	1	7d22h	AbleToScale ScalingActive
frontend	gic-group-6	0% / 65%	1	3	1	7d18h	AbleToScale ScalingActive

Figure - 5 - Horizontal Pod Autoscaling

Not being exactly autoscaling based on a predefined metric, we like to mention that our databases, in this case our mongodb and redis clusters, when desired, increase or decrease the number of replicas, they connect and disconnect automatically to their respective masters, continuing with all the integrity of the clusters, both mongodb and redis.

11 - Global Scaling

As our news platform grows and gains popularity, the need for global scaling becomes a significant consideration. Scaling our platform worldwide allows us to reach a larger audience and cater to users from different countries and regions. However, expanding globally brings forth various challenges related to load balancing, data replication and storage, and content compliance. In this chapter, we will explore the key aspects we should focus on when scaling our news platform globally and discuss the measures we need to take to ensure a seamless and compliant user experience.

To handle the increased traffic and ensure optimal performance, implementing a robust load balancing strategy is crucial. Load balancers distribute incoming requests across multiple servers, preventing any single server from becoming overwhelmed. When scaling globally, we need to consider setting up load balancers in multiple geographical regions to minimize latency and provide a smooth user experience. By strategically distributing user requests, load balancing helps to optimize resource utilization and enhance the scalability of our platform.

When it comes to global scaling, we also need to consider the importance of L3 (network layer) and L4 (transport layer) optimizations. Facebook has shown the significance of optimizing these layers to ensure efficient routing and network performance. By implementing cutting-edge techniques such as Anycast routing and leveraging the capabilities of Content Delivery Networks (CDNs), it is possible to effectively mitigate latency, alleviate network congestion, and enhance the overall speed and reliability of our news platform. These optimization strategies enable efficient content delivery to users across diverse geographical regions, thereby ensuring a seamless and highly responsive user experience regardless of their location. Through the judicious implementation of these advancements, our news platform can achieve significant performance improvements, facilitating the timely dissemination of news content to a wide audience while upholding the highest standards of quality and user satisfaction.

When scaling globally, data replication and storage play a vital role in maintaining data integrity and reducing latency. We ensure that data is accessible and available even in the case of localized outages or failures by replicating our databases across many areas. We may also give customers from different locations low-latency access by storing data in geographically dispersed data centers or using cloud-based storage solutions. We may increase data availability, reduce reaction times, and improve platform performance by intelligently replicating and storing our data.

As we expand our platform globally, it is crucial to comply with local regulations and legal requirements concerning content. Different countries have varying restrictions and sensitivities regarding the type of content that can be accessed or distributed. For example, certain countries may have restrictions on content critical of their government or culturally sensitive material. To address this, we need to implement measures to block traffic to countries that cannot receive certain content. Hypothetically, we could establish separate caches for different regions, such as having a cache for Portugal and another for China, ensuring compliance with local regulations and legal obligations. By adhering to content compliance guidelines, we can avoid potential legal issues, maintain a positive reputation, and provide a tailored user experience for each region.

Global scaling of our news platform presents opportunities for growth and reaching a wider audience. However, it also brings challenges related to load balancing, data replication and storage, and content compliance. By implementing a robust load balancing strategy, ensuring efficient data replication and storage, and complying with local content regulations, we can successfully scale our platform worldwide. This allows us to provide a seamless and compliant user experience, regardless of the user's geographical location. By addressing these aspects of global scaling, we can position our news platform for success on a global scale.

12 - Monitoring

As said before, the monitoring part was done through Prometheus (metric collection) and Grafana (metric visualization, and alerts). Two Dashboards were created, one on a more general context about metrics like CPU, RAM, network, and total number of Pods that are active.

The deployment of these services should be in a separate cluster, or in a different structure, because if in a moment of catastrophe something happens with the monitoring services in the same "place" as the rest of the system, it is not possible to have information about anything, as well as having these Non-isolated services end up consuming resources intended for the system itself, so it is good practice to have this type of service isolated from the rest of the system.

It is possible to have a long-term perception, since it is possible to change the timescale of the data that is being visualized. The grafana credentials are: user: admin, and password: admin (it asks to reset the password, so type admin again).

The dashboard can be accessed through this link: http://grafana.group-6.k3s/d/ca10a4b6-eef3-4a0f-b14a-ec53eabc4109/main-dashboard?orgId=1, reading a more general view of the number of CPU, RAM, and network that all the Pods are using, and then a more specific look at the Pods components like backend, frontend, databases. This dashboard consumes data from http://prometheus.deti.

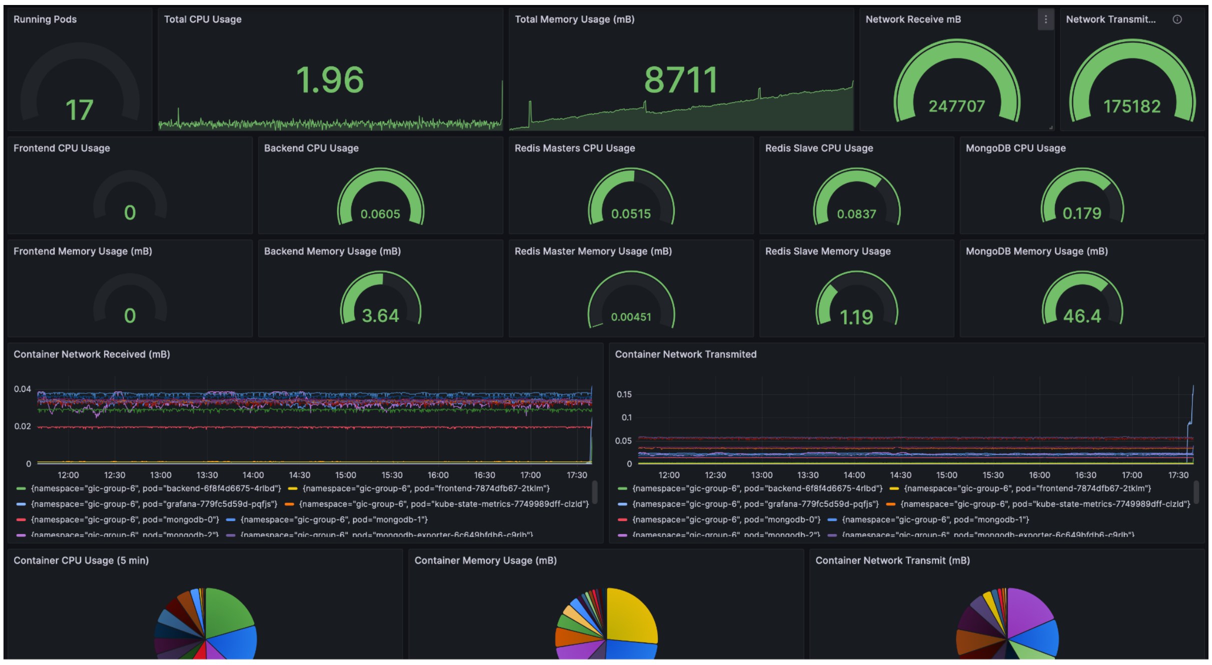


Figure 6- Grafana Main Dashboard

The second dashboard more connected to the latency part, and types of responses that the backend and frontend offer, can be accessed through the following link: http://grafana.group-6.k3s/d/qPdAviJmz/traefik-frontend- and-backend?orgId=1, here we can see the Frontend and Backend latency, the amount of response codes (200, 300, 400, 500), have a perception of the average response time over time, and the number of requests that each component is receiving, with the minimum, maximum, average and current value being displayed. This dashboard consumes data from http://prometheus.group-6.k3s.

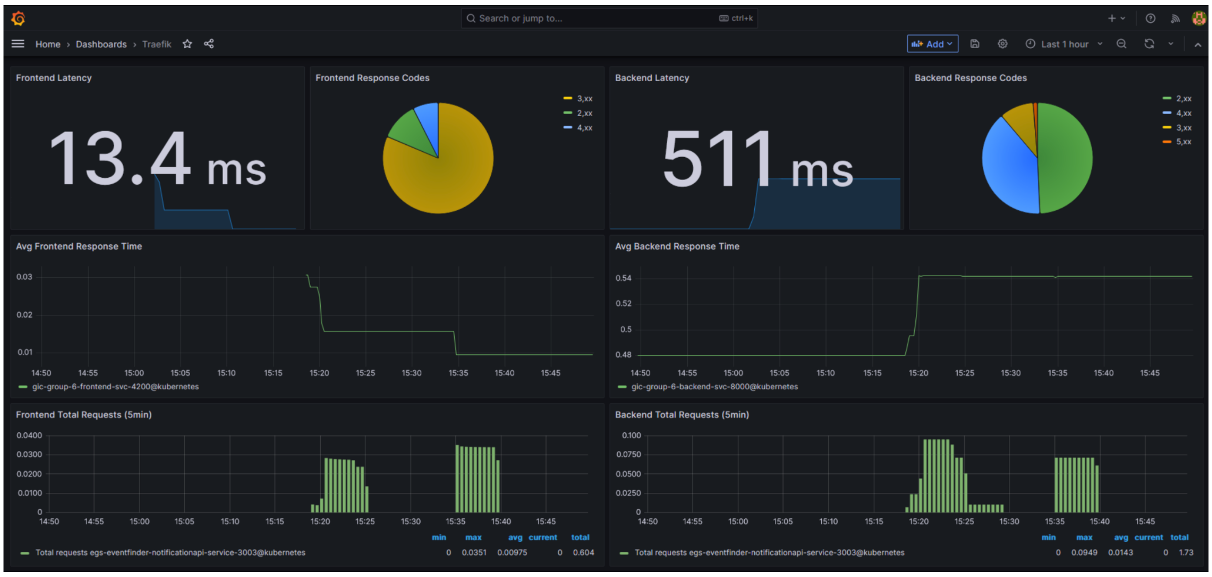
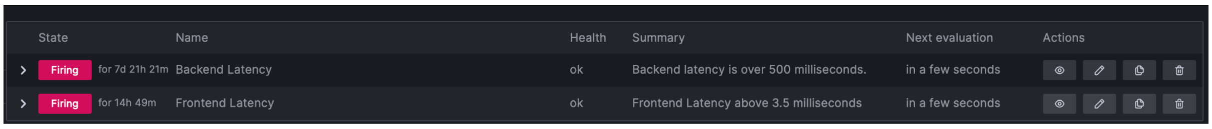

Home › Dashboards › Traefik
Frontend Latency
13.4 ms
Frontend Response Codes
3,xx
2,xx
4,xx
Backend Latency
511 ms
Backend Response Codes
2,xx
4,xx
3,xx
5,xx
Avg Frontend Response Time
gic-group-6-frontend-svc-4200@kubernetes
Avg Backend Response Time
gic-group-6-backend-svc-8000@kubernetes
Frontend Total Requests (5min)
Total requests egs-eventfinder-notificationapi-service-3003@kubernetes
min max avg current total
0 0.0351 0.00975 0 0.604
Backend Total Requests (5min)
Total requests egs-eventfinder-notificationapi-service-3003@kubernetes
min max avg current total
0 0.0949 0.0143 0 1.73

Figure 7 - Latency and Response Codes of Backend and Frontend

13 - Alert Manager

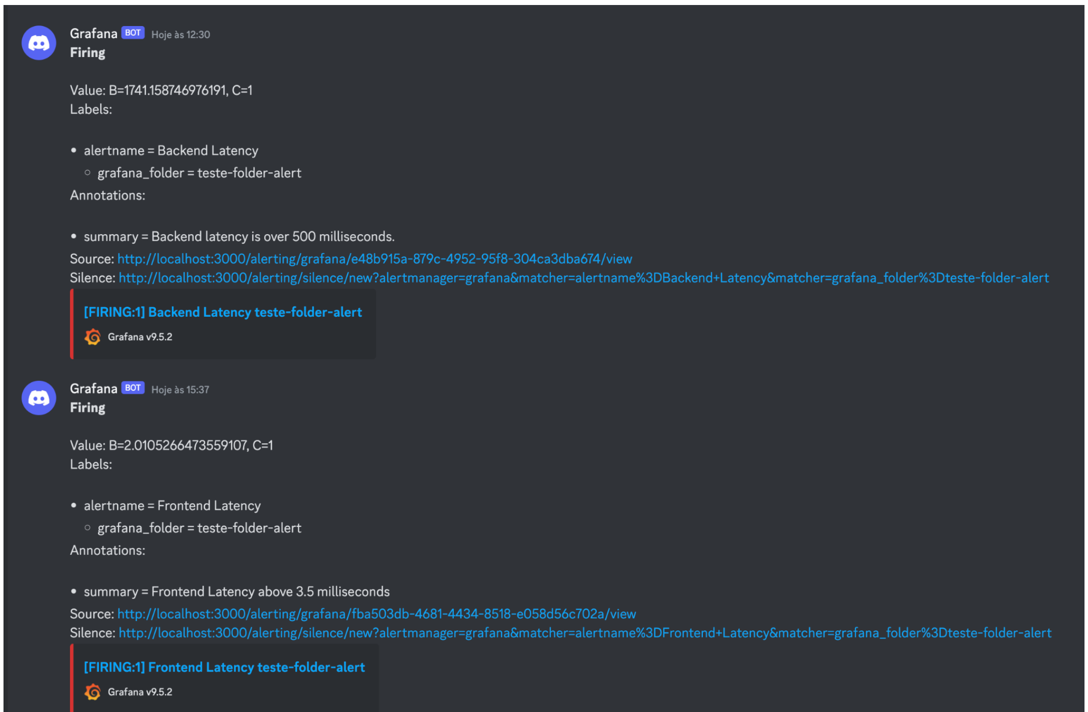
An Aler Manager is also configured in Grafana, for a Discord server, in this case, alerts are defined regarding latency that are triggered in the case of the backend when it is above 500 milliseconds, and in the case of the frontend when it is above of 3.5 milliseconds.



State		Name	Health	Summary	Next evaluation	Actions
Firing	for 7d 21h 21m	Backend Latency	ok	Backend latency is over 500 milliseconds.	in a few seconds	
Firing	for 14h 49m	Frontend Latency	ok	Frontend Latency above 3.5 milliseconds	in a few seconds	

Figure 8 - Alerts in Grafana

As said, alerts are received on a Discord server, and notification/messages are received whenever the alert is triggered.



Grafana BOT Hoje às 12:30
Firing

Value: B=1741.158746976191, C=1
Labels:

- alertname = Backend Latency
  - grafana_folder = teste-folder-alert

Annotations:

- summary = Backend latency is over 500 milliseconds.

Source: http://localhost:3000/alerting/grafana/e48b915a-879c-4952-95f8-304ca3dba674/view
Silence: http://localhost:3000/alerting/silence/new?alertmanager=grafana&matcher=alertname%3DBackend+Latency&matcher=grafana_folder%3Dteste-folder-alert

[FIRING:1] Backend Latency teste-folder-alert
Grafana v9.5.2

Grafana BOT Hoje às 15:37
Firing

Value: B=2.0105266473559107, C=1
Labels:

- alertname = Frontend Latency
  - grafana_folder = teste-folder-alert

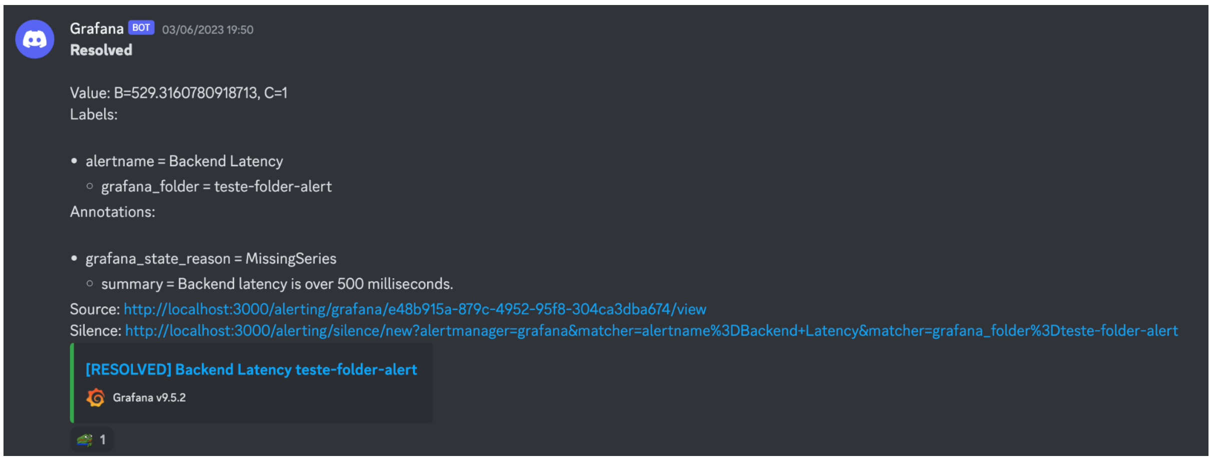
Annotations:

- summary = Frontend Latency above 3.5 milliseconds

Source: http://localhost:3000/alerting/grafana/fba503db-4681-4434-8518-e058d56c702a/view
Silence: http://localhost:3000/alerting/silence/new?alertmanager=grafana&matcher=alertname%3DFrontend+Latency&matcher=grafana_folder%3Dteste-folder-alert

[FIRING:1] Frontend Latency teste-folder-alert
Grafana v9.5.2

When the condition that triggered the alert is no longer verified, we also receive a notification saying that it is now resolved.



```
Grafana BOT 03/06/2023 19:50
Resolved

Value: B=529.3160780918713, C=1
Labels:

• alertname = Backend Latency
  ○ grafana_folder = teste-folder-alert
Annotations:

• grafana_state_reason = MissingSeries
  ○ summary = Backend latency is over 500 milliseconds.
Source: http://localhost:3000/alerting/grafana/e48b915a-879c-4952-95f8-304ca3dba674/view
Silence: http://localhost:3000/alerting/silence/new?alertmanager=grafana&matcher=alertname%3DBackend+Latency&matcher=grafana_folder%3Dteste-folder-alert
  [RESOLVED] Backend Latency teste-folder-alert
  Grafana v9.5.2
1
```

Figure 10 - Alert Resolved (Message on Discord)

14 - Auto Deployment

The auto deployment strategy for the news platform involves utilizing Kubernetes to automate the deployment process, providing scalability, availability, and reliability. This strategy eliminates the need for manual intervention and enables rapid platform upgrades and scalability.

Continuous Integration and Continuous Deployment (CI/CD) pipelines play a crucial role in automating the building, testing, and deployment of application code. By implementing CI/CD, all code changes undergo rigorous testing and quality checks before being deployed. This ensures a smooth and efficient deployment process.

In our case we can not implement a pipeline because the cluster is accessed through a VPN, so we were not able to run the pipelines.

Infrastructure as Code (IaC) allows for the automated provisioning and management of the underlying infrastructure, including Kubernetes clusters and networking. With IaC, the entire infrastructure setup can be defined in code, making it easy to replicate and deploy across different environments.

Deployment templates and configuration management tools, such as Kubernetes YAML files are used to define the desired state of the application and its dependencies. These templates capture all the necessary information for deployment, including container images, resource requirements, environment variables, and service configurations. Version control systems help track and apply changes and updates to these templates.

Rolling updates and blue/green deployments are supported by Kubernetes, enabling zero-downtime deployments and seamless updates. Rolling updates gradually update pods in a controlled manner, while blue/green deployments involve deploying a new version of the application alongside the existing version and gradually switching traffic to the new version.

Having a well-defined rollback and recovery strategy is essential in case of issues during updates or scaling operations. This involves maintaining multiple versions of container images, proper backups of databases and caches, and clear procedures for handling unexpected scenarios.

By implementing this autodeployment strategy, the news platform can benefit from streamlined deployments, scalability, availability, and reliability. Automation reduces human error, facilitates quick updates, and enables seamless scaling based on user demand.

15 - Demo

A video demo of the project can be accessed here: https://youtu.be/C2yAi3COBeA
It had 3 parts, the presentation and how the site works, the auto recovery working, and the autoscaling working.

16 - Conclusion

Through this project's course, we've learned how to use Kubernetes, and many of its best practices, to manage the underlying infrastructure and scale a system according to our specific case requirements. It's very important to understand how the system will work in order to choose the best technologies and the best deployment strategy.

In retrospect, we solved all of the initial bottlenecks we faced in the initial phases of the project, since we've configured everything to run, scale and communicate automatically, which reduces the burden of the developer/operator and is very useful for disaster recovery and scalability.

Thus, throughout the project's course we've consolidated several aspects, strategies and best practices to scale a specific system using Kubernetes, ensuring the service of our system to users, depending on the current demand.

17 - References

- https://www.marktest.com/wap/a/n/id~29a2.aspx - Ranking netAudience Março 2023, Marktest
- https://velero.io/ - Velero - Backups on Kubernetes
- https://kubernetes.io/docs/home/ - Kubernetes Documentation
- https://medium.com/geekculture/redis-cluster-on-kubernetes-c9839f1c14b6 - Redis Cluster on Kubernetes
- https://www.split.io/glossary/blue-green-deployment/ - Blue/Green Deployment