PRÁCTICA RECUPERACIÓN DE LA INFORMACIÓN

ELASTICSEARCH + KIBANA

Verónica Gómez Gómez, Carlos Grande Núñez y Pablo Olmos Martínez

UNIVERSIDAD REY JUAN CARLOS

[ÍNDICE]

01 Introducción a la práctica

Descripción

La práctica propuesta consiste en la carga de los datos de eventos en repositorios de Github relacionados con el proyecto de Cloud Nation Computing Foundation en la API de Elasticsearch y Kibana. Posteriormente los datos cargados se usaran para realizar un análisis de superviviencia y un estudio de la comunidad en general.

01

Objetivos

- Cargar los datos proporcionados en Elasticsearch para su estudio y monitorización.
- Realizar un análisis exploratorio previo sobre los datos mediante queries a elastic.
- Realizar un análisis de superviviencia sobre una muestra de los datos.
- Desarrollar un Dashboard para la monitorización de los datos.

02

Metodología:

1. Desarrollo de los scripts necesarios para la carga de datos en elasticsearch y creación de un mapping correcto para indexar los documentos. Incluyendo notebook explicativo en python.
2. Desarrollo de un análisis exploratorio para una mejor comprensión y estudio de los datos. Además Se realizarán qeries directamente a elastic para poder responder algunas preguntas que nos faciliten el estudio.

2.1

3. Desarrollo de un análisis de superviviencia sobre una muestra de 20.000 datos recogidos mediante una query de Elasticsearch en python.
4. Finalmente se van a realizar un dashboard en Kibana para simular una monitorización de los datos relevantes relacionados con la práctica.

2.2

2.3

2.4

A

02 Elaboración de la práctica

2.1 Carga de los datos en Elasticsearch

Para la carga de los datos en Elasticsearch se han desarrollado un script llamado *elastic_loader.py* con dos funciones. La primera función llamada *load_json()* carga los datos de json en una variable de python, por otro lado, la segunda función llamada *upload_to_index()* permite añadir los datos de python a Elasticsearch mediante la inserción de documentos en el índice creado previamente en Elastic.

Para la carga de datos se realizan las siguientes operaciones en el notebook.

1. Conexión con Elasticsearch
2. Carga de datos en python
3. Carga del mapping generado
4. Creación de un índice
5. Subida de documentos de python al índice de Elasticsearch

Script de carga de datos en Elasticsearch

Permite la carga de documentos desde python a un índice en elasticsearch

```
def upload_to_index(client, index_name, data):
    start = time.time()
    print('Starting upload...', '\n')
    for i, doc in enumerate(data):
        client.index(index=index_name, id=doc['_id'], body=doc['_source'])
    print('Data was uploaded succesfully!')
```

```
Loading json file...
100 percent completed: 491703 documents in 49.03s
json file was loaded succesfully!
```

Generación de mapping

Otro script interesante desarrollado dentro del notebook nos ha permitido extraer el mapping de un índice creado, lo que nos ha permitido modificar el mapping por defecto de Elastic y reimportar el índice con los formatos corregidos. También nos ha permitido comprobar la correcta subida de los datos.

Permite la extracción de un mapping aplicado en un índice

```
mapping = elastic.indices.get_mapping(idx) # extraemos el mapping
mapping_keys = mapping[idx]["mappings"].keys()
doc_type = list(mapping_keys)[0]
schema = mapping[idx]['mappings']['properties']
print(json.dumps(schema, indent=4))
```

```
{
    "Author_bot": {
        "type": "boolean"
    },
    "Author_domain": {
        "type": "keyword"
    },
    "Author_gender": {
        "type": "keyword"
    },
    "Author_gender_acc": {
        "type": "long"
    }, ...
```

2.2 Análisis exploratorio de los datos

En este notebook hemos desarrollado un pequeño análisis exploratorio de los datos, respondiendo a preguntas que nos surgían antes de realizar el análisis de superviviencia.

Para ello hemos extraido todos los campos de la base de datos con un total de 76 columnas muchas de ellas duplicadas o sin relevancia para la práctica por lo que mostraremos las variables usadas en las queries.

01

- Author_user_name
- Author_id
- author_date
- Commit_id
- commit_date
- Author_org_name
- files
- author_id
- hash
- repo_name
- commit_date_hour
- commit_date_weekday

02

¿Cuántos desarrolladores únicos hay en el proyecto CNCF?

2.1

```
s = Search(using=elastic, index=idx)
s.aggs.metric('Number of developers', 'cardinality', field='Author_id')

result = s.execute()
result.to_dict()["aggregations"]
```

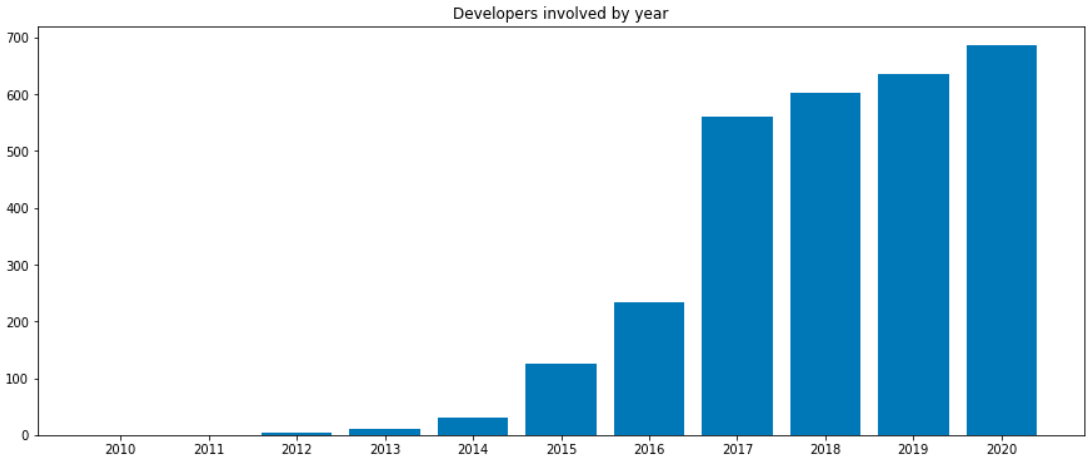
```
{'Number of developers': {'value': 16691}}
```

¿Cuántos desarrolladores se involucran cada año?

2.2

```
years = range(2010, 2021)
output = []
for year in years:
    s = Search(using=elastic, index=idx).filter('range', author_date={'gt': datetime(year-1, 12, 12), 'lt': datetime(year, 1, 1)})
    s.aggs.metric('developers', 'cardinality', field='author_id')
    result = s.execute()
    result = result.to_dict()["aggregations"]['developers']['value']
    output.append(result)
```

2.3



2.4

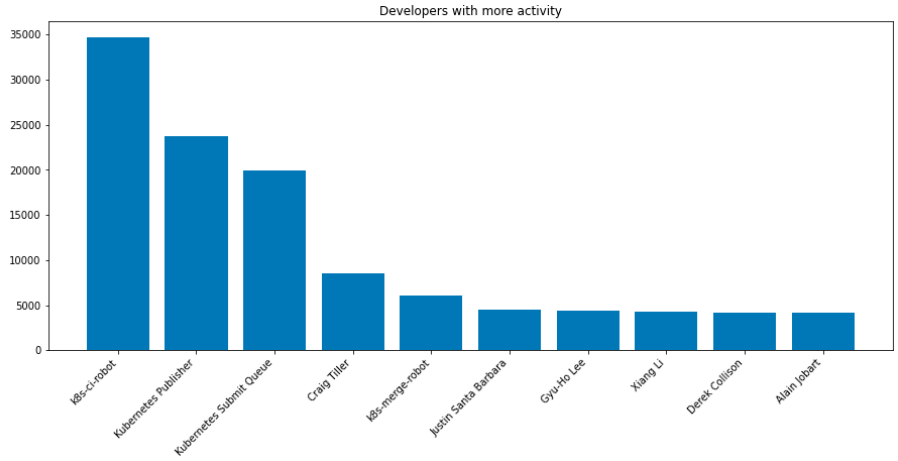
¿Cuales son los 10 desarrolladores más activos?

```
s = Search(using=elastic, index=idx)
s.aggs.metric('Number of developers', 'cardinality', field='Author_id')

result = s.execute()
result.to_dict()["aggregations"]
```

01

```
{'Number of developers': {'value': 16691}}
```



02

2.1

¿Cuántos commits se han realizado en la base de datos ignorando commits vacíos?

```
s = Search(using=elastic, index=idx).filter('range', files={'gt':0})
s.aggs.metric('commits', 'cardinality', field='hash')
result = s.execute()
result.to_dict()["aggregations"]
```

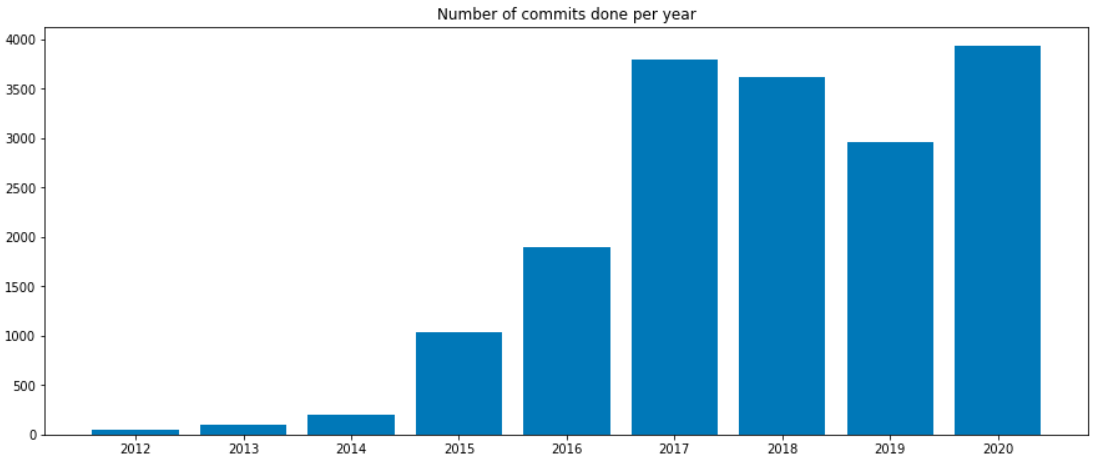
```
{'commits': {'value': 364481}}
```

2.2

¿Cuántos commits se realizan por año?

```
for year in years:
    s = Search(using=elastic, index=idx).filter('range', author_date={'gt': datetime(year-1, 12, 12), 'lt': datetime(year, 1, 1)})
    s.aggs.metric('developers', 'cardinality', field='hash')
    result = s.execute()
    result = result.to_dict()["aggregations"]['developers']['value']
    output.append(result)
```

2.3



2.4

A

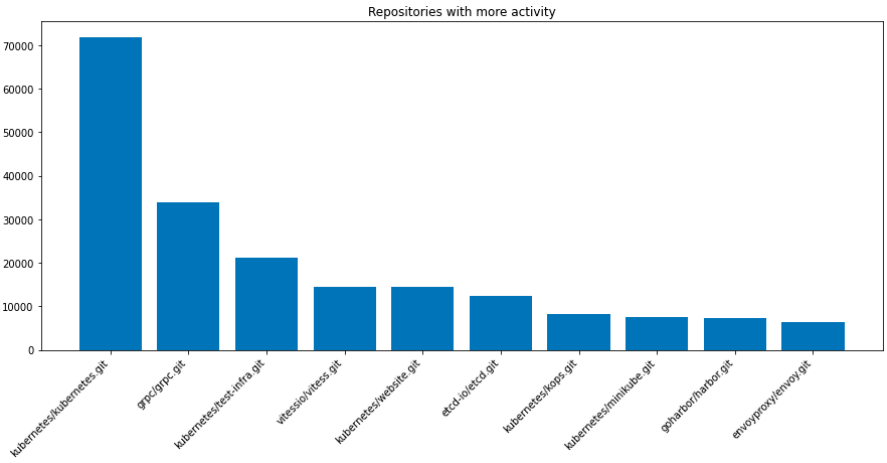
¿Cuales son los 10 repositorios más activos?

```
s = Search(using=elastic, index=idx).filter('range', files={'gt':0})
s.aggs.bucket('by_repo', 'terms', field='repo_name', size = 10).metric('n_commits', 'cardinality', field='hash')
result = s.execute()
result = result.to_dict()["aggregations"]['by_repo']['buckets']
```

01

	repositories	doc_count	n_commits
0	kubernetes/kubernetes.git	71194	71849
1	grpc/grpc.git	33941	33945
2	kubernetes/test-infra.git	21041	21071
3	vitessio/vitess.git	14689	14510
4	kubernetes/website.git	14363	14537
5	etcd-io/etcd.git	12581	12345
6	kubernetes/kops.git	8161	8122
7	kubernetes/minikube.git	7611	7570
8	goharbor/harbor.git	7340	7351
9	envoyproxy/envoy.git	6384	6379

02



¿Cuántas organizaciones hay en la base de datos?

2.1

```
s = Search(using=elastic, index=idx)
s.aggs.metric('Organizations', 'cardinality', field='author_org_name')
result = s.execute()
result.to_dict()["aggregations"]
```

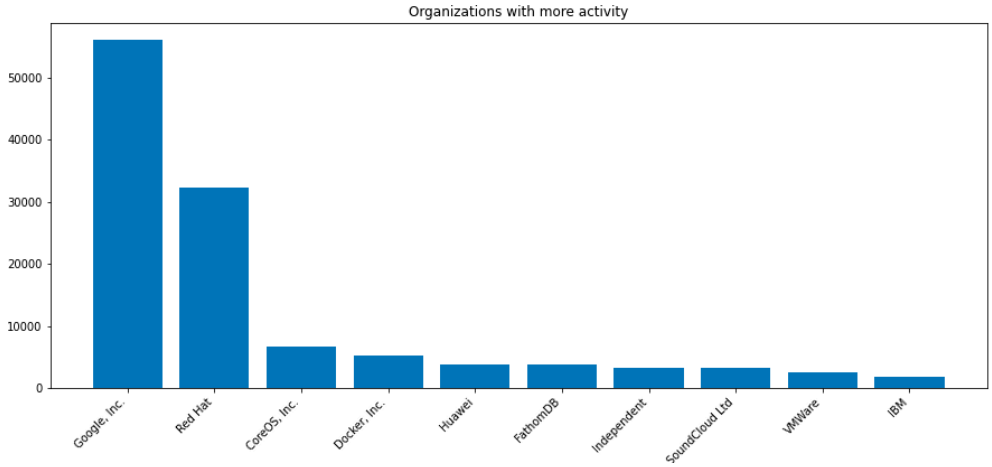
{'Organizations': {'value': 161}}

¿Cuales son las 10 organizaciones más activas?

2.2

```
s = Search(using=elastic, index=idx)
s.aggs.bucket('organizations', 'terms', field='author_org_name', size = 11)
result = s.execute()
```

2.3



2.4

2.3 Análisis de superviviencia

Para el análisis de supervivencia se ha realizado un tercer notebook explicando todo el proceso seguido para el estudio duración hasta que un desarrollador abandona un proyecto. Para poder realizar este análsis se han seguido los siguientes pasos.

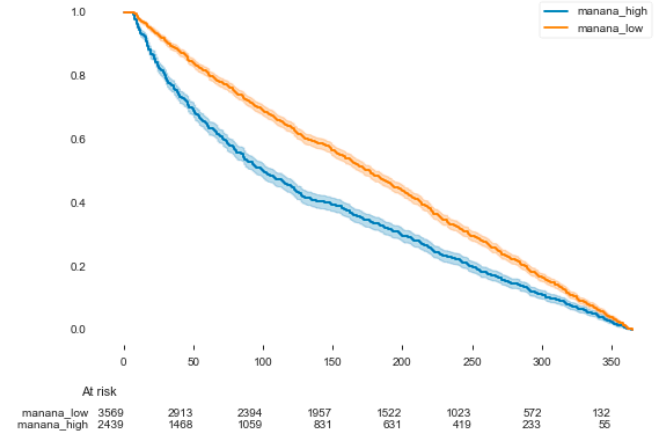
1. Generación de dataframe y recogida de la muestra de 20.000 elementos.
2. Transformación de variables
3. Unión de tablas de datos
4. Obtención de las tasas de abandono
5. Análisis de supervivencia mediante el estimador de Kaplan-Meier

El límite del evento para el análisis de supervivencia elegido es de una semana, pasada una semana sin actividad damos por abandonado un proyecto por parte del desarrollador. Este parámetro podría ser parametrizado permitiendo elegir el límite del evento y estudiar sus variaciones sobre los datos.

Por otro lado el rango temporal elegido para el estudio es de un año, el cual también podría también podría modificarse para la comprensión de los sucesos a lo largo de la temporalidad de la base de datos.

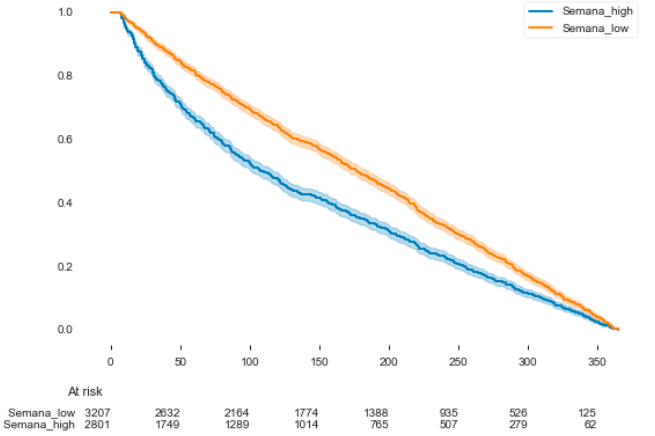
Dado que para el estudio se ha realizado un análisis de supervivencia en función de las diferentes variables situándolas como eventos. No vamos a mostrar los resultados del estudio en su totalidad, por lo que nos centraremos en las visualizaciones más interesantes.

¿Supervivencia de mañana frente a tarde?



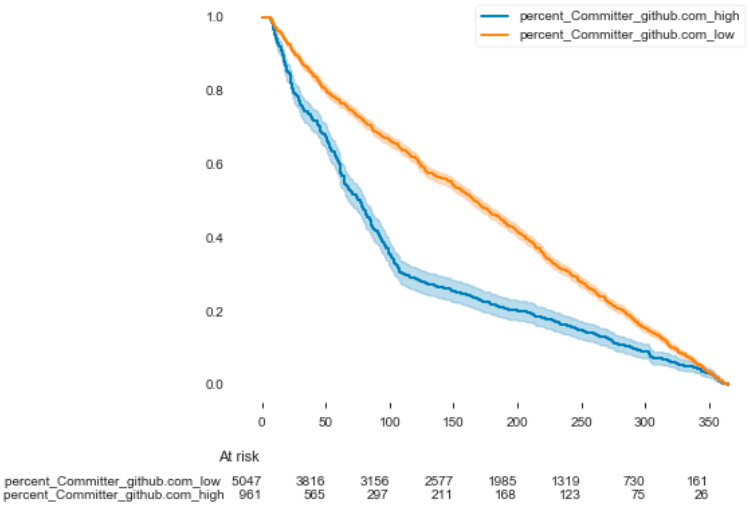
En este caso se puede observar la estimación de supervivencia de los desarrolladores que publican por las mañanas frente a los que publican por las tardes.

¿Supervivencia de fin de semana y entre semana ?



En este otro caso se puede ver el contraste en el abandono entre la gente que publica entre semana y los que publican los fines de semana.

¿Supervivencia por commits en Github?



Aquí podemos ver otro ejemplo sobre la estimación de supervivencia de los desarrolladores que han realizado commits en Github, habiendo una gran diferencia en las tasas de supervivencia.

2.4 Generación del dashboard en kibana

Para poder monitorizar los datos de la comunidad de desarrolladores a raiz de los estudios previos realizados, se han seleccionado una serie de visualizaciones que nos pueden ayudar a tener una visión general de lo que está sucediento en tiempo real. Para ello se han propuesto las siguientes visualizaciones en el dashboard.

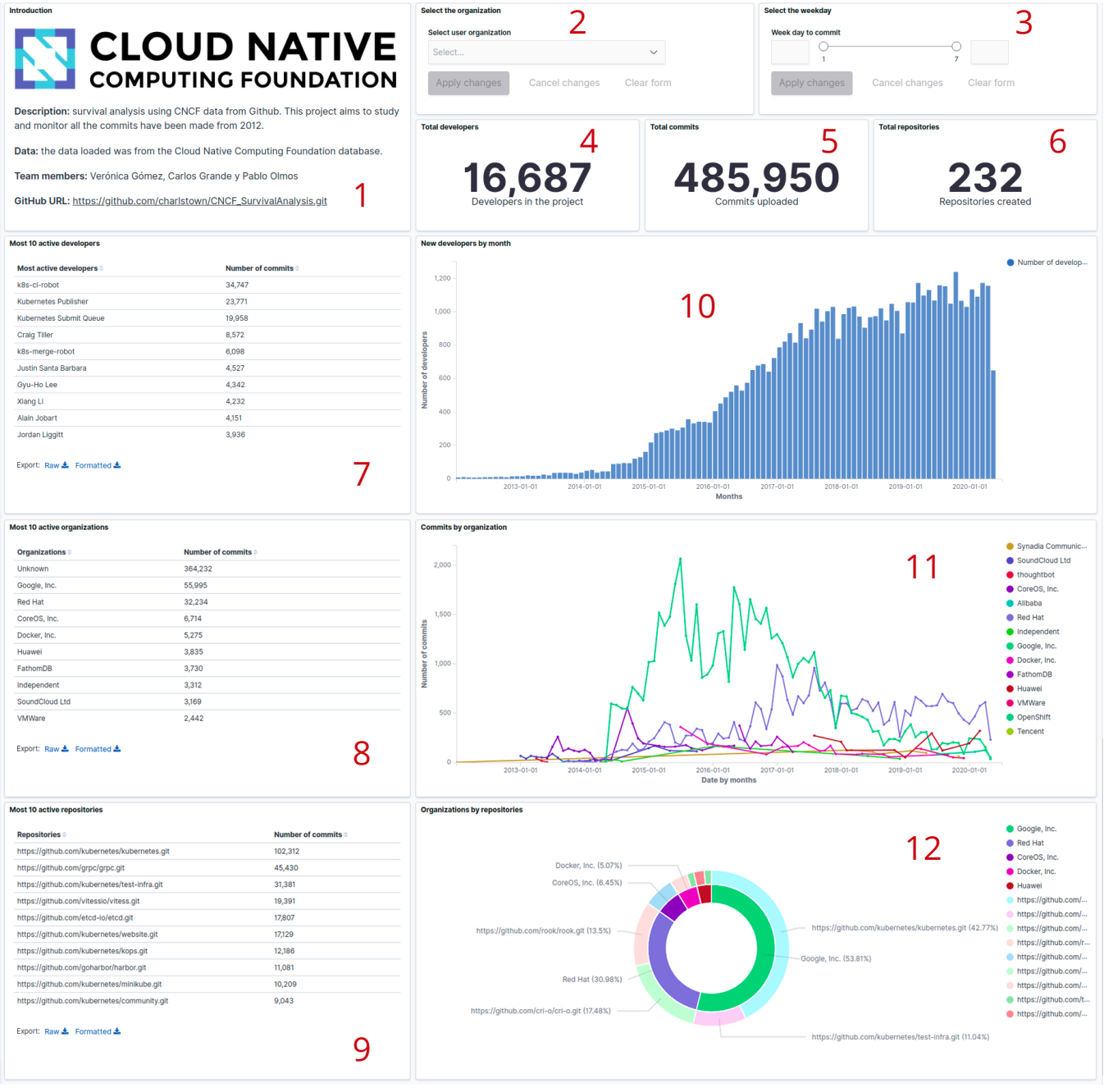
01

1. Markdown de introducción explicativo.
2. Panel de control de la organización a la que pertenecen los desarrolladores.
3. Panel de control con un slider para limitar los días de la semana.
4. Número total de desarrolladores en el proyecto hasta la fecha.
5. Número total de commits realizados hasta la fecha.
6. Numero total de repositorios creados hasta la fecha.
7. Los 10 desarrolladores más activos.
8. Las 10 compañías más activas.
9. Los 10 repositorios más activos
10. Histograma con el número de nuevos desarrolladores cada mes.

02

11. Grafico de líneas con la actividad temporal de cada empresa.
12. Gráfico con las empresas y los principales repositiorios en los que publican.

2.1



2.2

2.3

2.4

A

ANEXO: LIBRERÍAS Y FUNCIONES USADAS

Librerías usadas para la práctica

```
01 import warnings
   warnings.filterwarnings('ignore')
   import pandas as pd
   import pandas_profiling as pf
   import json
   from datetime import datetime
   from pandas.io.json import json_normalize
   import matplotlib.pyplot as plt
   import seaborn as sns
02 import datetime
   import os
   import numpy as np
   from lifelines.plotting import add_at_risk_counts
   from elasticsearch import Elasticsearch
   from elasticsearch_dsl import Search
```

Otras funciones usadas para la práctica

```
2.1 def load_json(path):
        path_open = open(path, 'r').readlines()
        file = []
        try:
            for i, line in enumerate(path_open):
                file.append(json.loads(line))
        except Exception as e:
2.2         print(e)
        return file

    def search_to_pandas(search):
        results = search['hits']['hits']
        df_documents = pd.DataFrame([doc['_source'] for doc in results])
        return df_documents

2.3 def documents_to_pandas(documents):
        df_documents = pd.DataFrame(documents)
        return df_documents
```

2.4