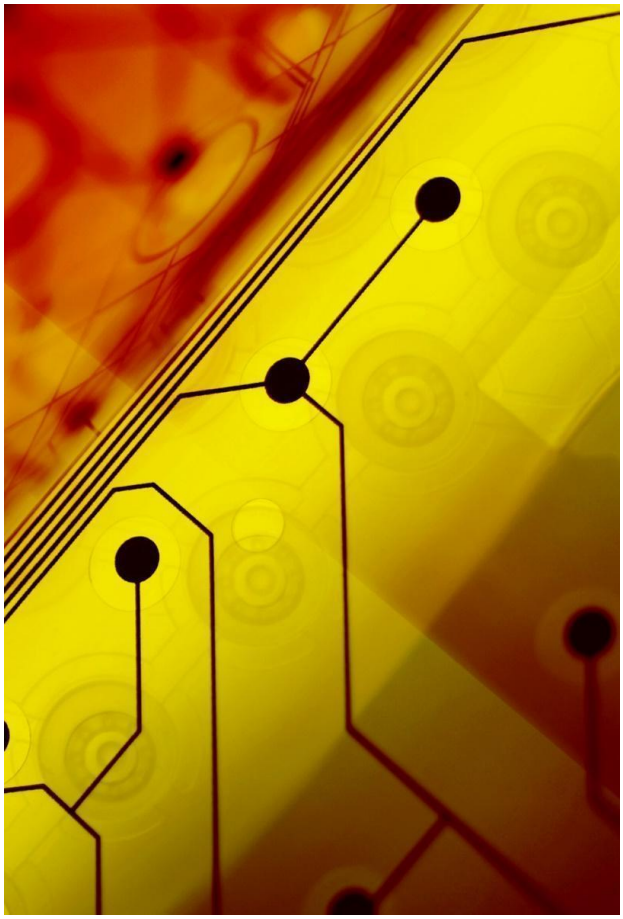


UNIVERSIDAD PANAMERICANA

App Web - montada en la nube - para Monitoreo de Portafolio de inversión

Desarrollo de una aplicación, montada en la nube, que sirva como herramienta de apoyo al monitoreo del valor de acciones de empresas de tecnología y banca en la Bolsa de Valores a través de predicciones del valor basado en el histórico de la acción y factores externos, principalmente noticias y actividad de actores clave para las acciones.

Muñoz, Guillermo

Ortíz-Monasterio, Pedro

Trillas, Josune

Cómputo en la nube

2do semestre

Profesor: Raúl Morales Salcedo

06 de octubre del 2021

App Web - montada en la nube - para Monitoreo de Portafolio de inversión

Comprensión del problema

Comprensión general de problema

En este proyecto trabajaremos con las acciones que conforman a un portafolio de inversión, seleccionado por un fondo de inversionistas, hecho a la medida para el perfil de riesgo e intereses de inversión de sus clientes.

Lo que un fondo de inversión busca es optimizar el dinero de sus clientes a través de la selección de acciones y porcentajes para la inversión que maximicen las ganancias para el dinero puesto en juego. El manejo de los portafolios de inversión tiene toda una ciencia detrás. Cuenta con distintos métodos para el armado de los portafolios según el nivel de riesgo que se quiere, el capital a invertir y el plazo en el que se espera retorno. Con base en lo anterior, se diseñan portafolios de inversión a la medida de los clientes, siempre buscando satisfacer sus requerimientos.

Una labor que los especialistas realizan una vez armado el portafolio es el monitoreo del valor de las acciones. El objetivo natural de un portafolio es contar con la volatilidad histórica de la acción al momento de armarlo, sin embargo, el monitoreo sirve para detectar cambios en el mercado que puedan afectar la rentabilidad y/o riesgo que puedan proyectarse al futuro más allá de lo contemplado en el diseño del portafolio. Cuando esto sucede, se recomienda al cliente mover su capital de una acción a otra o cambiar la mezcla de las acciones para salvaguardar el rendimiento y el riesgo del portafolio en su conjunto.

Objetivos comerciales de la solución

El presente trabajo busca desarrollar una aplicación, montada en la nube, que sirva como herramienta de apoyo al monitoreo de los valores de las acciones a través de predicciones del valor basado en el histórico de la acción y factores externos, principalmente noticias y actividad de actores clave para las acciones.

El objetivo de que la solución esté montada en la nube es evitar altas inversiones por parte del fondo de inversión para crear infraestructura que pueda soportar el análisis de todos sus portafolios. Así siempre podrán hacerse soluciones modulares, escalables y a medida que satisfagan las necesidades actuales del negocio.

Valoración de la situación

Actualmente no todos los fondos de inversión cuentan con herramientas sencillas que les permitan realizar la tarea del monitoreo del valor de acciones y el comparativo con eventos externos. Contar con una herramienta de este estilo les permitirá eficientar procesos y

tener más precisión al momento de tomar decisiones importantes relacionadas con las acciones y los portafolios.

Requisitos, supuestos y restricciones

Se tomará como base un portafolio con acciones de 10 empresas tecnológicas y bancarias que cotizan en el índice S&P 500 en Estados Unidos. Esto es porque este índice es uno de los más importantes a nivel mundial y es considerado como uno de los más representativos del mercado.

El portafolio contiene las empresas tecnológicas más importantes, las cuales se encuadran en el tipo de rendimiento y riesgo buscado, por su dinamismo y crecimiento sostenido, y pesar de su volatilidad. Estas empresas son:

- Apple Inc.
- Amazon Inc.
- NVIDIA Corp.
- Salesforce Inc.
- Oracle Corp.
- Microsoft Corp.
- Visa Inc.
- Bank of America Corp.
- Adobe Inc.
- Mastercard Inc.

Es necesario contar con datos históricos y actuales de las acciones elegidas, para lo cual se usarán datos públicos de Yahoo Finance. Además, las noticias serán extraídas también de los datos públicos en Google News.

Sabemos que las acciones de la bolsa de valores continuamente se mueven por noticias, comunicados de prensa de las compañías, publicaciones en redes sociales, y otros medios de comunicación, por lo cual cuantificamos y mostraremos datos de éstas noticias en la aplicación, de tal manera que se puedan usar para la toma de decisiones.

En determinado momento podríamos tener muchos usuarios del fondo de inversión accesando la información y corriendo los modelos al mismo tiempo, por lo cual necesitamos accesibilidad remota, en una arquitectura elástica y escalable.

Riesgos y contingencias

Desde el punto de vista de negocio, uno de los riesgos que siempre existe es la mala adopción de la aplicación para su uso inteligente. Será necesario capacitar a los usuarios para que entiendan el potencial que tiene la aplicación y los beneficios que pueden obtener al utilizarla en su trabajo.

Desde el punto de vista del desarrollo e implementación de la aplicación en la nube, el riesgo más importante a atacar es el de la seguridad. Habrá que cuidar que los permisos estén bien definidos y todo bien resguardado para que no haya riesgos de seguridad.

Análisis de costes/beneficios

Los beneficios esperados de la adopción de una aplicación como la que se desarrolla para este trabajo son:

- Se producirán ahorros en tiempo de parte de los asesores de inversión. Esto provocará que cada asesor pueda atender a más clientes y generar más eficiencias.
- Se tendrá información a tiempo para la toma de decisiones inteligentes, lo que puede provocar una mejora significativa en los rendimientos de los portafolios. Este tipo de decisiones suele generar movimientos del +/- 5% del rendimiento de las inversiones.
- Es una herramienta que servirá como argumento de venta. Los potenciales clientes, al ver la tecnología que utiliza el fondo de inversión pueden sentirse más seguros y atraídos a invertir su dinero con esta institución.

Los costos que este desarrollo implica, son:

Para la estimación de los costos, utilizamos la herramienta de GCP la cual nos dió lo siguiente.

Estimated Monthly Cost: USD 197.85

1 x	e2-standard-4	730 total hours per month	USD 100.76
4 x	n1-standard-1	2920 total hours per month	USD 97.09
Total Estimated Monthly Cost			**USD 197.85**

Sin embargo tenemos que considerar que mucho del procesamiento y modelado se está haciendo en cuadernos o archivos python por parte de los desarrolladores y que los files procesados se están guardando en el storage de la instancia. Por lo que de requerir mayor almacenamiento el costo aumentaría.

Desarrollo de un plan de proyecto

Para llevar a cabo este proyecto, es necesario completar varias etapas:

1. Obtención de la información:

    a. Obtener los valores históricos de las acciones
    b. Obtener las noticias históricas de las acciones
    c. Generar un modelo de predicción de las acciones
2. Creación de modelos:
    a. Generar un modelo de predicción estándar para las acciones
    b. Generar un modelo de análisis de sentimientos sobre las noticias históricas.
3. Generar la web app donde los usuarios puedan interactuar con los modelos.
    a. Guardar en sistema de control de versiones
4. Crear la arquitectura en la nube donde cargaremos nuestra web app.
    a. Diseño de la arquitectura
    b. Desarrollo de API de forecasting
    c. Creación de imagen de contenedor
5. Cargar la web app en la arquitectura de la nube.
    a. Crear cluster de Kubernetes
    b. Crear registro de contenedores
    c. Construir Imagen de Contenedor
    d. Guardar contenedores en registro
    e. Desplegar contenedores en cluster
        i. Exponer servicios
    f. Creación de instancia de VM
    g. Configuración del ambiente y dependencias
    h. Configuración redes
        i. GCP
        ii. Instancia
    i. Despliegue de tablero

Valoración de datasets, herramientas y modelos a utilizar

Los datasets a utilizar son dos:

- Los datos de las acciones a evaluar.
- Los datos de las noticias con relación a las empresas.

Las herramientas principales que utilizaremos son:

- Streamlit
- Python
- Github
- GKE
- Docker
- GKE

Los modelos a utilizar son:

- Prophet de Facebook

- VADER (Valence Aware Dictionary for Sentiment Reasoning)

Diseño de la Arquitectura

Para el desarrollo de nuestro proyecto era necesario utilizar una arquitectura completamente en la nube, aprovechando todos los beneficios de esta. Necesitamos un sistema con una arquitectura escalable de acuerdo con nuestros requerimientos actuales, y fácil de administrar. Para satisfacer esto, se desarrolló la siguiente arquitectura sobre GCP:

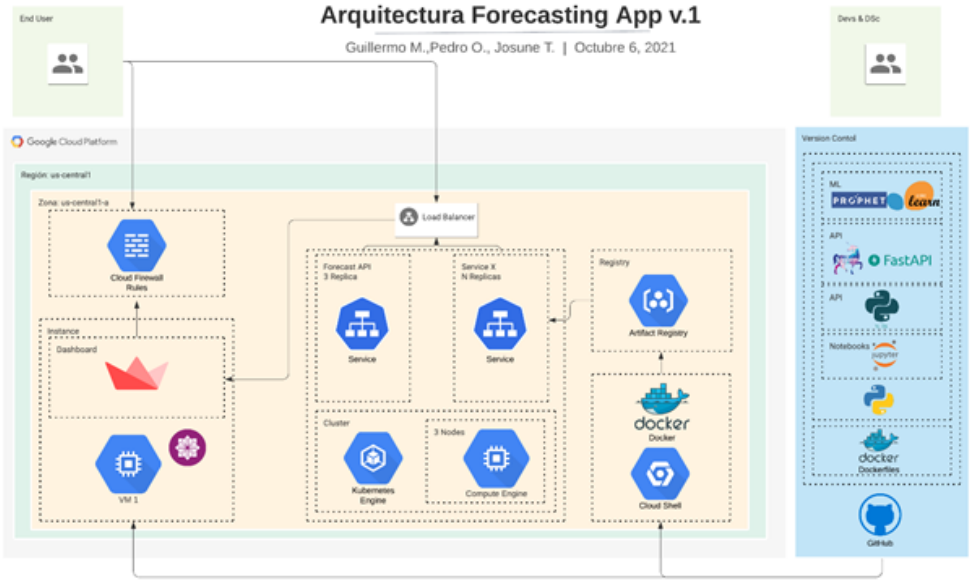


Imagen 1. Arquitectura de la aplicación

Nuestro sistema cuenta con dos secciones principales, el dashboard montado sobre una instancia de VM y nuestro API de predicciones desplegada sobre un contenedor en un clúster de GKE (Google Kubernetes Engine), esta arquitectura es modular y está pensada para desarrollarse basada en microservicios, por lo que, de querer agregarse cualquier otro servicio adicional, tenemos una fácil integración al sistema y sin tener que ser invasiva a servicios existentes. Además, trabajar con en un ambiente de este estilo nos permite:

- Tener despliegues de servicios independientes.
- Añadir servicios y funciones de forma incremental.
- Fácil actualización e integración de librerías y dependencias.
- Agilizar el ciclo de vida de los servicios.

Por otra parte, para seguir las buenas prácticas, todo el desarrollo del proyecto se trabajó utilizando Git como framework de control de versiones. Pudiendo así colaborar con facilidad, integrar el trabajo de los distintos miembros, así como compartir de manera sencilla. Esto también nos facilitó el despliegue de nuestros aplicativos en las distintas partes de la arquitectura.

API Forecasting

El API de forecasting se desplegó como un contenedor con un servicio Rest en Python donde se tenían los modelos entrenados con Prophet y así simplemente se pueden hacer llamadas para obtener predicciones estableciendo las acciones y días de predicción deseados.

Fast API y Uvicorn

Existen distintos frameworks para la construcción de APIs con Python, para este caso específico se decidió usar FastAPI, sus características principales son:

- Rápido y de alto rendimiento.
- Fácil de usar y aprender, por lo que reduce tiempos de desarrollo.
- Diseñado para tener servicios listos para producción, con documentación interactiva automática.
- Basados en los estándares abiertos para APIs: OpenAPI y JSON Schema.

Sin embargo FastAPI no cuenta con un servidor integrado como otros frameworks, pero esto se soluciona utilizando Uvicorn, un servidor ultraligero y rápido diseñado para microservicios. Este está basado en ASGI (Asynchronous Server Gateway Interface), destinado a proporcionar una interfaz estándar entre servidores web, frameworks y aplicaciones Python con capacidad asíncrona.

Usar estas tecnologías para desplegar nuestros servicios nos permite tener de manera fácil y rápida aplicaciones que siguen siendo competitivas, dentro de los estándares de la industria y listas para ir a un ambiente productivo.

Contenedores

Cuando vamos a desarrollar una arquitectura basada en microservicios, lo hacemos utilizando contenedores. Un contenedor es una unidad de software estándar que empaqueta el código y todas sus dependencias para que la aplicación se ejecute de forma rápida y fiable de un entorno informático a otro. Una imagen de contenedor Docker es un paquete de software ligero, independiente y ejecutable que incluye todo lo necesario para ejecutar una aplicación: código, runtimes, herramientas del sistema, bibliotecas y configuraciones.

Las imágenes de contenedores se convierten en contenedores en tiempo de ejecución. El software en contenedores se ejecutará siempre igual, independientemente de la infraestructura. Los contenedores aíslan el software de su entorno y garantizan que funcione de manera uniforme a pesar de las diferencias, por ejemplo, entre el desarrollo y la puesta en escena.

Algunos de los beneficios de los contenedores:

- Estándar: Los contenedores siguen distintos estándares para que pudieran ser portátiles en cualquier lugar.
- Ligeros: Los contenedores comparten el núcleo del sistema operativo de la máquina y, por lo tanto, no requieren un sistema operativo por aplicación, lo que impulsa una mayor eficiencia de los servidores y reduce los costes de estos y de las licencias.
- Seguros: Las aplicaciones son más seguras en los contenedores proporcionan las capacidades de aislamiento por defecto más fuertes de la industria.

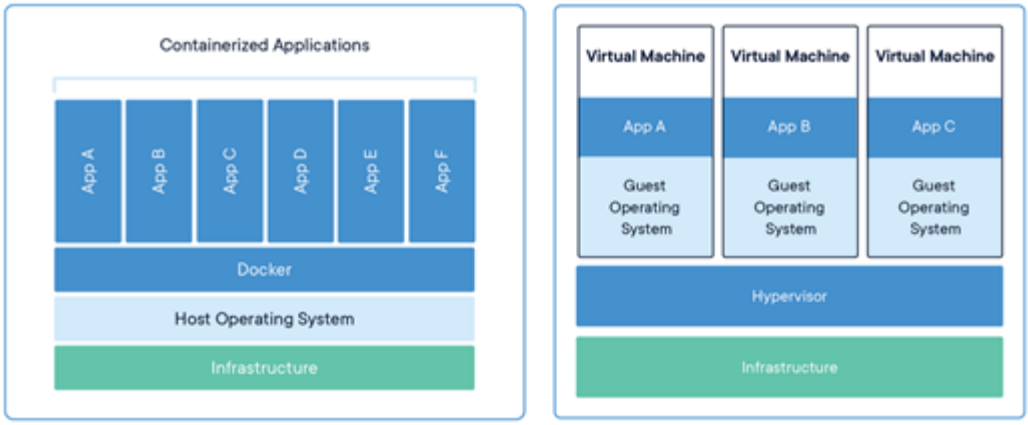


Imagen 2. Contenedores

Docker

Para el manejo de los contenedores utilizamos Docker. La tecnología Docker usa el kernel de Linux y las funciones de este, como Cgroups y namespaces, para segregar los procesos, de modo que puedan ejecutarse de manera independiente. El propósito de los contenedores es esta independencia: la capacidad de ejecutar varios procesos y aplicaciones por separado para hacer un mejor uso de su infraestructura y, al mismo tiempo, conservar la seguridad que tendría con sistemas separados.

Las herramientas del contenedor, como Docker, ofrecen un modelo de implementación basado en imágenes. Esto permite compartir una aplicación, o un conjunto de servicios, con todas sus dependencias en varios entornos. Docker también automatiza la implementación de la aplicación (o conjuntos combinados de procesos que constituyen una aplicación) en este entorno de contenedores.

Estas herramientas desarrolladas a partir de los contenedores de Linux, lo que hace a Docker fácil de usar y único, otorgan a los usuarios un acceso sin precedentes a las aplicaciones, la capacidad de implementar rápidamente y control sobre las versiones y su distribución.

El enfoque Docker para la creación de contenedores se centra en la capacidad de tomar una parte de una aplicación, para actualizarla o repararla, sin necesidad de tomar la aplicación completa. Además de este enfoque basado en los microservicios, puede compartir procesos entre varias aplicaciones de la misma forma que funciona la arquitectura orientada al servicio (SOA).

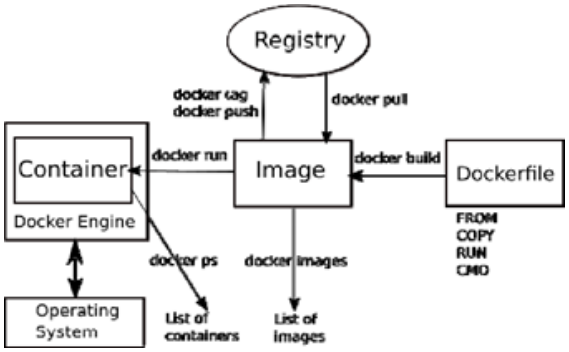


Imagen 3. - Flujo contenedores

Artifact Registry

Para administrar las imágenes de contenedores de nuestros servicios, utilizamos Artifact Registry. Un registro de contenedores es un repositorio o grupo de repositorios que se utiliza para almacenar imágenes de contenedores.

La imagen de un contenedor es una copia de este (con todos los archivos y elementos que contiene y que conforman una aplicación), y se utiliza para reproducirla con el fin de expandir rápidamente o trasladarla a otros sistemas, según sea necesario. Una vez que se crea esa imagen, se forma una especie de plantilla que luego se puede usar para crear aplicaciones nuevas o para ajustar una aplicación actual.

Cuando trabaja con imágenes de contenedores, necesita un lugar para guardarlas y acceder a ellas a medida que se generen. Ahí entra en juego el registro de contenedores. Básicamente funciona como un lugar para almacenarlas y compartirlas a través de un proceso de carga y descarga.

Como se está trabajando sobre GCP, utilizaremos Artifact Registry. Esta herramienta permite administrar imágenes de contenedor y paquetes de lenguajes (como Maven y npm) en un solo lugar. Está completamente integrado a las herramientas y entornos de ejecución de Google Cloud. Así podremos mantener el control de nuestras imágenes y versionarlas de manera fácil, así como integrarlas a todo el ecosistema de nuestra aplicación

GKE

En sí mismo, Docker es una excelente herramienta para la gestión de contenedores individuales. Al comenzar a utilizar cada vez más contenedores y aplicaciones en

contenedores, divididas en cientos de piezas, la gestión y la organización se pueden tornar muy difíciles. Finalmente, debe retroceder y agrupar los contenedores para ofrecer servicios, como redes, seguridad, telemetría, etc., en todos sus contenedores. Es aquí donde aparece Kubernetes. Kubernetes es una plataforma que automatiza las operaciones de los contenedores de Linux. Elimina muchos de los procesos manuales involucrados en la implementación y escalabilidad de las aplicaciones en contenedores. En otras palabras, puede crear un clúster de grupos de hosts que ejecutan contenedores de Linux, y Kubernetes lo ayuda a administrar con facilidad y eficacia esos clústeres.

Con Kubernetes podemos:

- Orquestar contenedores en uno o múltiples hosts.
- Hacer un mejor uso del hardware para maximizar los recursos necesarios para ejecutar sus aplicaciones empresariales.
- Controlar y automatizar las implementaciones y actualizaciones de las aplicaciones.
- Montar y añadir almacenamiento para ejecutar aplicaciones con estado.
- Escalar las aplicaciones en contenedores y sus recursos sobre la marcha.
- Comprobaciones de estado y autoregeneración de aplicaciones y escalamiento automáticos.

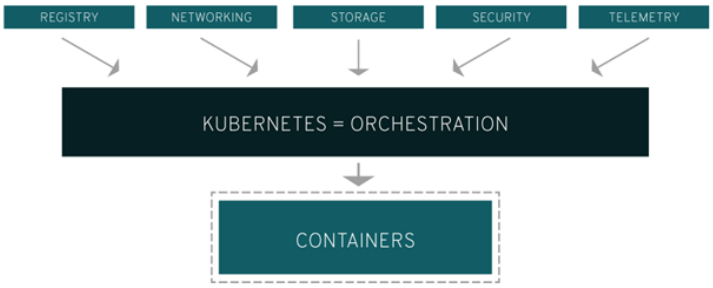


Imagen 4. Orquestación de Kubernetes

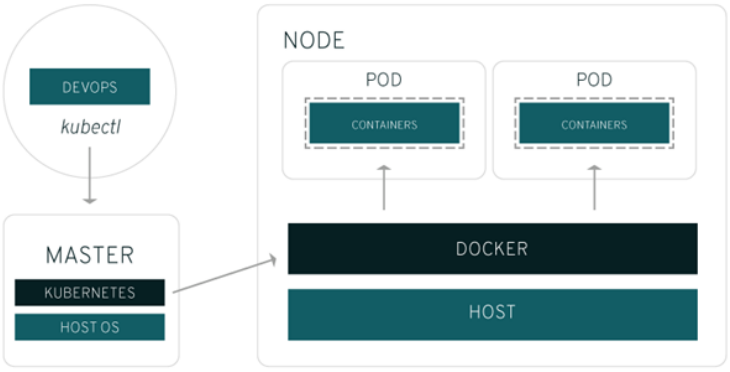


Imagen 5. Flujo Kubernetes

En este caso de uso particular usamos la distribución de Kubernetes GKE. Google Kubernetes Engine (GKE) proporciona un entorno gestionado para desplegar, gestionar y escalar sus aplicaciones en contenedores utilizando la infraestructura de Google. El entorno GKE consta de varias máquinas (concretamente, instancias de Compute Engine) agrupadas para formar un clúster.

En este cluster, desplegamos las imágenes guardadas en Artifact Registry, y así podemos exponerla como un API pública y GKE se encargará del escalamiento, balanceo de cargas y todo lo demás.

Control De Versiones

El control de versiones es un sistema que registra los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo para poder recuperar versiones específicas más adelante. Te permite revertir archivos seleccionados a un estado anterior, revertir todo el proyecto a un estado anterior, comparar los cambios a lo largo del tiempo, ver quién fue el último en modificar algo que podría estar causando un problema, quién introdujo un problema y cuándo, y mucho más. El uso de un VCS también significa, en general, que, si se estropean las cosas o se pierden archivos, se puede recuperar fácilmente. Además, todo esto se consigue con muy pocos gastos generales.

Tener un repositorio de GitHub facilita el seguimiento de los proyectos: todos los archivos necesarios para determinados análisis pueden mantenerse. A medida que los proyectos se desarrollan. Cada archivo en GitHub tiene un historial, lo que facilita la exploración de los cambios que se produjeron en él en diferentes momentos. Puedes revisar el código de otras personas, añadir comentarios a ciertas líneas o al documento en general, y sugerir cambios. GitHub permite asignar tareas a diferentes usuarios, dejando claro quién es responsable de cada parte del proyecto.

Aprovechando esta herramienta, el proyecto está planeado para que cada miembro del equipo vaya desarrollando lo que le corresponde en el entorno de su preferencia y finalmente centralizar todo en el repositorio. Por otra parte, el uso de este repositorio nos facilitó el despliegue de nuestro código a las distintas partes de nuestra arquitectura. Tanto el Dashboard en la máquina virtual, como la creación y despliegue de contenedores utilizando Docker y GKE.

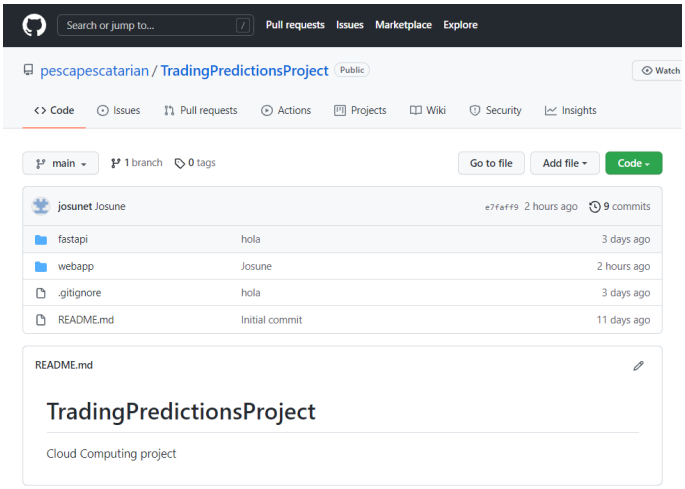


Imagen 5. Repositorio de Github

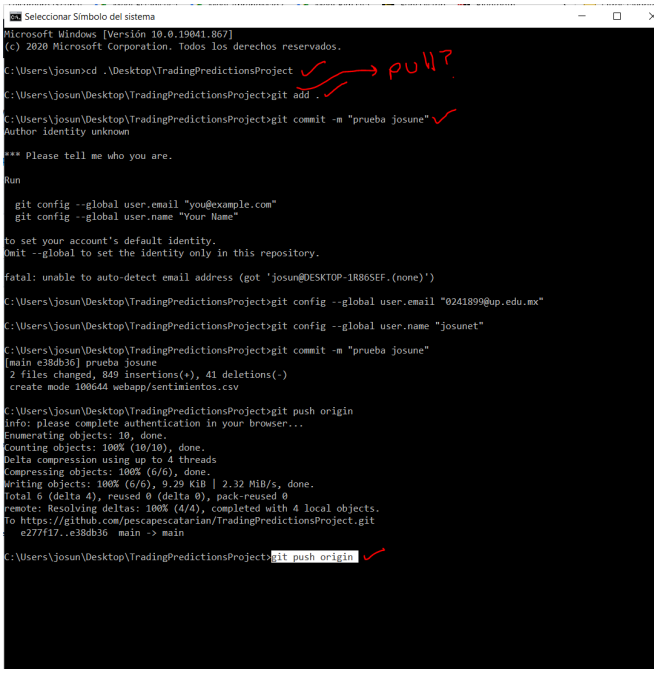


Imagen 6. Enviar cambios en archivos al repositorio de github

Dashboard

Para la parte del dashboard utilizamos una instancia de VM en Compute Engine, ya que para nuestro dashboard trabaja con datos vivos y algunos de los procesos son un poco más costosos computacionalmente, por lo que podrían afectar el desempeño de los contenedores. Sobre esto te montamos un servidor web de Python utilizando la herramienta Streamlit la cual nos permite crear reportes completamente personalizables y desplegarlos como un sitio web.

Compute Engine

Google Compute Engine (GCE) es la infraestructura global que sustenta el servicio de Google Cloud Platform. Dentro de la plataforma, podemos seleccionar la máquina adecuada para nuestras necesidades: tenemos opciones tanto de máquina pre definida como personalizada. Las máquinas predefinidas vienen con combinaciones de uso general de CPU y memoria. Pero si éstas no satisfacen nuestras necesidades, podemos elegir los tipos de máquinas personalizadas, que nos permiten seleccionar el número exacto de CPU y la cantidad de memoria que necesitamos para las cargas de trabajo.

Comprensión de los datos y Preparación

Primer set de datos

Los datos a utilizar son el valor de las acciones, proveniente de Yahoo finance. Estos datos se actualizan de manera diaria, y nos dan información suficiente para poder generar una predicción generalizada. Los datos se obtienen de acuerdo al símbolo de la acción o al "Ticker", es decir el símbolo es un input para obtener lo siguiente:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2015-01-02	5.0325	5.0700	4.9525	5.0325	4.840064	11368000
1	2015-01-05	5.0325	5.0475	4.9250	4.9475	4.758314	19795200
2	2015-01-06	4.9550	4.9600	4.7925	4.7975	4.614049	19776400
3	2015-01-07	4.8325	4.8750	4.7700	4.7850	4.602027	32180800
4	2015-01-08	4.8400	4.9950	4.8375	4.9650	4.775145	28378000

Imagen 7. Datos obtenidos de Yahoo Finance para cada acción

- ***Date***: La fecha en la que sucedieron las operaciones de compra y venta.
- ***Open***: El valor de la acción al momento de la apertura de mercados
- ***Close***: El valor de la acción al cierre de los mercados
- ***High***: El valor más alto al cual se vendió la acción durante ese día
- ***Low***: El valor más bajo al cual se vendió la acción ese día
- ***Adj Close***: Valor de la acción al considerar otros factores como dividendos, divisiones de acciones y nuevas ofertas de acciones.

De este set, los datos de interés son la fecha (Date) y el precio al cierre (Close). Una bondad adicional de este set de datos es que ya está limpio, y no fue necesario amputar datos, sino que solamente seleccionamos el rango de fechas que necesitamos para la extracción.

```
START = "2015-01-01"
TODAY = date.today().strftime("%Y-%m-%d")
selected_stock = "NVDA"
```

```
def load_data(ticker):
        data = yf.download(ticker, START, TODAY)
        data.reset_index(inplace=True)
        return data
```

Imagen 8. Código muestra de selección de información para análisis

Exploración de los datos:

```
data1 = load_data(selected_stock)
data1.info()
```

```
[*********************100%***********************]  1 of 1 completed
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1700 entries, 0 to 1699
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       1700 non-null   datetime64[ns]
 1   Open       1700 non-null   float64
 2   High       1700 non-null   float64
 3   Low        1700 non-null   float64
 4   Close      1700 non-null   float64
 5   Adj Close  1700 non-null   float64
 6   Volume     1700 non-null   int64
dtypes: datetime64[ns](1), float64(5), int64(1)
memory usage: 93.1 KB
```

Imagen 9. Exploración de datos 1 - datos de acciones

```
data1 = load_data(selected_stock)
data1.describe()
```

```
[*********************100%***********************]  1 of 1 completed
```

	Open	High	Low	Close	Adj Close	Volume
count	1700.000000	1700.000000	1700.000000	1700.000000	1700.000000	1.700000e+03
mean	56.854347	57.728059	55.900732	56.869446	56.632343	4.706964e+07
std	51.707017	52.442252	50.854795	51.723552	51.771037	2.850861e+07
min	4.812500	4.875000	4.735000	4.785000	4.602027	5.244800e+06
25%	15.726250	15.811250	15.468750	15.700625	15.475104	2.913770e+07
50%	43.656250	44.375000	43.010000	43.558750	43.376083	3.935480e+07
75%	65.938751	67.160002	65.078751	66.332502	65.823236	5.748800e+07
max	228.330002	230.429993	225.509995	228.429993	228.429993	3.692928e+08

Imagen 10. Exploración de datos 2 - datos de acciones

Segundo set de datos

El segundo set de datos son las noticias de las empresas seleccionadas, obtenidas de Google News mediante un web scrapper. Se puede ver y/o descargar el código completo del web scrapper en la siguiente liga:

https://github.com/pescapescatarian/TradingPredictionsProject/blob/main/analisis_sentimientos/googe_news_scrapping.py

Los datos se ven así:

	Unnamed: 0	title	link	date	Stock	Source
0	0	When To Buy Apple	https://seekingalpha.com/article/4411287-apple-when-to-buy	2021-03-04 08:00:00+00:00	AAPL	Seeking Alpha
1	1	Where Will Apple Stock Be In 5 Years? Know When To Hold 'Em And When To Fold 'Em	https://seekingalpha.com/article/4438479-apple-stock-5-years	2021-07-09 07:00:00+00:00	AAPL	Seeking Alpha
2	2	Apple: An Opportunity To Buy Shares As Apple Buys Shares During Consolidation	https://seekingalpha.com/article/4443021-apple-an-opportunity-to-buy-shares	2021-08-02 07:00:00+00:00	AAPL	Seeking Alpha
3	3	Will Apple Stock Reach $200? In Our Opinion, Not If As Is	https://seekingalpha.com/article/4437503-will-apple-stock-reach-200	2021-07-02 07:00:00+00:00	AAPL	Seeking Alpha
4	4	Where Will Apple Stock Be In 10 Years? What To Consider	https://seekingalpha.com/article/4432703-apple-stock-in-10-years	2021-06-03 07:00:00+00:00	AAPL	Seeking Alpha

Imagen 11. Datos obtenidos de la extracción de noticias de Google

Diccionario de datos:

- ***title***: el título de la noticia
- ***link***: la dirección url donde se encuentra el contenido completo de la noticia
- ***date***: la fecha en la cual se publicó la noticia
- ***Stock***: el símbolo o ticker de la acción de la empresa
- ***Source***: La fuente original que publicó la noticia

Exploración de los datos:

```
[7]  data = pd.read_csv('google_news.csv')

     data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Unnamed: 0   1000 non-null   int64
 1   date         1000 non-null   object
 2   title        1000 non-null   object
 3   link         1000 non-null   object
 4   Stock        1000 non-null   object
 5   Source       1000 non-null   object
 6   Unnamed: 6   0 non-null      float64
 7   Unnamed: 7   0 non-null      float64
 8   Unnamed: 8   0 non-null      float64
 9   Unnamed: 9   0 non-null      float64
 10  Unnamed: 10  0 non-null      float64
dtypes: float64(5), int64(1), object(5)
memory usage: 86.1+ KB
```

Imagen 12. Exploración de datos 3 - Noticias de Google

Limpieza de datos

Para la limpieza de este set de datos, tuvimos que pasarlo por un proceso de limpieza con dos sencillos pasos:

- Convertir el campo de fecha de objeto a fecha.

- Eliminar las fuentes de los títulos de las noticias (mediante expresiones regulares) para no ensuciar el modelo de análisis de sentimientos.

En un escenario normal pudo no haber sido necesario convertir la fecha, pero en esta caso ese campo será la llave para unir los datos de noticias con los datos de precios de las acciones.

La limpieza está en el mismo cuaderno de analisis de sentimientos y se puede consultar aqui:
https://github.com/pescapescatarian/TradingPredictionsProject/blob/main/analisis_sentimientos/AnalisisSentimientosPortafolio.ipynb

Descarga de datos

Pensando en que muchas veces a los asesores quisieran ver los datos de los valores de las acciones con mayor detalle para realizar algún tipo de reporte sencillo cara a sus clientes, habilitamos la opción para que desde la misma aplicación se pueda visualizar y/o descargar el set de datos del histórico del valor de las acciones al día de la consulta.

Modelado

Primer modelo: predicción del valor de las acciones a futuro

Para la predicción del valor futuro de las acciones se decidió utilizar un procedimiento de predicción desarrollado por Facebook: Prophet. Este procedimiento es verdaderamente útil para pronosticar datos provenientes de series de tiempo donde no necesariamente hay tendencias lineales. Trabaja muy bien cuando hay estacionalidad o grandes cantidades de datos históricos. También tiene formas de trabajar con datos faltantes, cambios bruscos en tendencias y maneja bien los *outliers*.

La librería de prophet además nos permite ver los componentes del pronóstico, usando el método Prophet.plot_components. Con el cual vemos la tendencia de toda la serie de tiempo, la estacionalidad anual y la estacionalidad semanal de la serie. Otra característica de valor de Prophet es que proyecta 3 líneas de tendencia, una central, una alta y otra baja, proporcionando un rango de optimismo y pesimismo a la proyección de futuro, estas 3 líneas de tendencia las proyecta también hacia el pasado, de tal manera que se puede observar lo que hubiera sido la predicción de haberla generado en el pasado y compararla con los movimientos reales en la historia.

Se utilizó la librería de Streamlit para montar este modelo en específico en una aplicación web, que permita al usuario interactuar con el modelo, obedeciendo a las necesidades definidas para el proyecto. Streamlit es una biblioteca de Python de código abierto especialmente diseñada para crear y compartir aplicaciones web personalizadas de aprendizaje automático y ciencia de datos. Streamlit permite alojar las aplicaciones en su propio Streamlit Cloud, en un nivel comunitario gratuito, en un nivel empresarial, o alojarlo

directamente con cualquier proveedor de hosting que soporte Python (en nuestro caso se utilizó Google Cloud Services).

El modelo fue creado para tomar algunos parámetros por default: el periodo histórico que se usará, y el listado limitado de símbolos de acciones para usar (las acciones que conforman el portafolio), y el modelo de Prophet no se puede cambiar. Sin embargo el usuario puede interactuar con él, ya que hay algunos otros parámetros que se pueden definir por el usuario: el símbolo de la acción que quiere predecir y el número de semanas para las cuales quiere hacer la predicción.

Posteriormente, mostramos un cuadro con las noticias de la acción (empezando por las más recientes), después del cual el usuario puede indicar si quiere afectar la predicción del modelo de manera positiva o negativa. El modelo entonces recalcula la predicción

El código del modelo se puede encontrar en el siguiente link:

https://github.com/pescapescatarian/TradingPredictionsProject/blob/a7aae75d2e09bb77367cc24c3103add249e5f43a/webapp/pages/main.py

Segundo modelo: Análisis de sentimientos de las noticias

Para poder cuantificar las noticias, en la línea de tiempo de tal forma que los datos sean comparables con los precios de la acción, se decidió usar sobre los encabezados de las noticias un modelo de procesamiento de lenguaje natural para el análisis de sentimientos: Vader Sentiment analysis.

VADER (Valence Aware Dictionary for Sentiment Reasoning) es un modelo utilizado para el análisis de sentimientos de texto que es sensible tanto a la polaridad (positiva / negativa) como a la intensidad de la emoción. Este modelo forma parte de la librería NLTK y se puede aplicar directamente a datos de texto sin etiquetar.

El análisis de sentimientos de VADER se basa en un diccionario que asigna intensidades emocionales conocidas como puntuaciones de sentimiento a características léxicas del texto. La puntuación de sentimiento de un texto se obtiene sumando la intensidad de cada palabra en el texto. VADER es sensible al contexto de las palabras, y procesa el énfasis de mayúsculas y puntuación.

Al modelo de análisis de sentimientos le pasamos el texto de los encabezados de las noticias, ya limpio, pero conservando puntuación y mayúsculas, las cuales son útiles para el modelo. El modelo nos arroja 4 puntuaciones para cada noticia: puntuación positiva, puntuación negativa, puntuación neutral y puntuación compuesta. La última, es la suma de positivos y negativos. Una puntuación compuesta mayor que 1 significa una noticia positiva, una noticia con puntuación menor que 1 significa una noticia negativa, y una puntuación cero implica una noticia completamente neutral. Esta puntuación compuesta será la que usaremos para comparar con el valor de las acciones.

```
{'compound': 0.25, 'neg': 0.0, 'neu': 0.889, 'pos': 0.111},
{'compound': 0.4404, 'neg': 0.0, 'neu': 0.818, 'pos': 0.182},
{'compound': 0.0772, 'neg': 0.053, 'neu': 0.886, 'pos': 0.061},
{'compound': 0.4019, 'neg': 0.0, 'neu': 0.863, 'pos': 0.137},
{'compound': -0.631, 'neg': 0.257, 'neu': 0.743, 'pos': 0.0},
{'compound': 0.7351, 'neg': 0.0, 'neu': 0.592, 'pos': 0.408},
{'compound': 0.4019, 'neg': 0.0, 'neu': 0.701, 'pos': 0.299},
```

Imagen 13. Muestra de resultados de análisis de sentimientos

Los resultados obtenidos se deben unir a la tabla de datos original para saber a qué noticias corresponden. En este punto se debe resolver una problemática adicional para llevarnos los datos finales de sentimiento: podemos tener más de 1 noticia en 1 día por cada acción, por lo cual, sacamos el promedio de la puntuación de sentimientos compuesta de cada acción por día, para tener una sola puntuación por acción por día:

	date	Stock	compound	neg	neu	pos
0	2020-11-02	AAPL	0.7430	0.000	0.656	0.344
1	2020-11-24	AAPL	0.0000	0.000	1.000	0.000
2	2020-12-24	AAPL	0.3818	0.000	0.698	0.302
3	2020-12-27	AAPL	0.0000	0.000	1.000	0.000
4	2021-01-08	AAPL	-0.3400	0.375	0.625	0.000

Imagen 14. Muestra del promedio de puntuación por acción por día del análisis de sentimientos.

Debido a que el análisis de sentimientos no requiere ninguna interacción con el usuario final, este modelo se corrió en un cuaderno de python, que no será montado en la arquitectura final de nuestra aplicación.

El cuaderno de python completo, donde se llevó a cabo este análisis de sentimientos se puede encontrar en:

https://github.com/pescapescatarian/TradingPredictionsProject/blob/main/analisis_sentimientos/AnalisisSentimientosPortafolio.ipynb

Comprobación de modelos aplicados en la solución

Para el modelo de predicción de valor de las acciones corrimos las métricas que nos proporciona el mismo modelo de Facebook Prophet, de las cuales tomamos MAPE (Error Porcentual Absoluto Medio , por sus siglas en inglés) el cual nos permite comparar el error en diferentes acciones, ya que es porcentual. Lo que podemos concluir es que este modelo es bastante bueno para predecir el valor de las acciones en horizontes de tiempo cortos (menores a 100 días) aunque depende mucho de la volatilidad de la acción.

	MAPE for Prediction Horizon				
Stock	37 days	100 days	200 days	300 days	365 days
AAPL	8.3%	17.1%	15.5%	23.1%	20.2%
ADBE	6.1%	11.0%	12.6%	19.2%	20.0%
AMZN	8.5%	13.2%	16.6%	29.0%	32.0%
BAC	11.0%	15.9%	21.5%	22.4%	22.8%
CRM	7.1%	9.0%	13.8%	15.5%	22.3%
MA	6.5%	9.1%	10.1%	12.3%	9.0%
MSFT	4.0%	8.7%	6.7%	12.3%	10.4%
NVDA	18.2%	34.2%	39.6%	63.3%	58.4%
ORCL	2.8%	5.5%	13.4%	16.0%	20.5%
V	4.3%	7.4%	7.2%	10.1%	7.6%
Average	7.7%	13.1%	15.7%	22.3%	22.3%

Imagen 15.Error porcentual medio absoluto para diferentes horizontes de tiempo para el portafolio

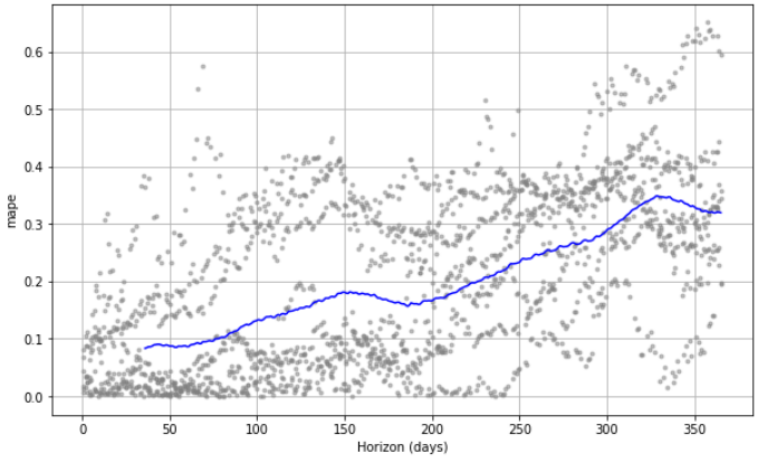


Imagen 16. Error porcentual absoluto medio para diferentes horizontes de predicción de AMZN

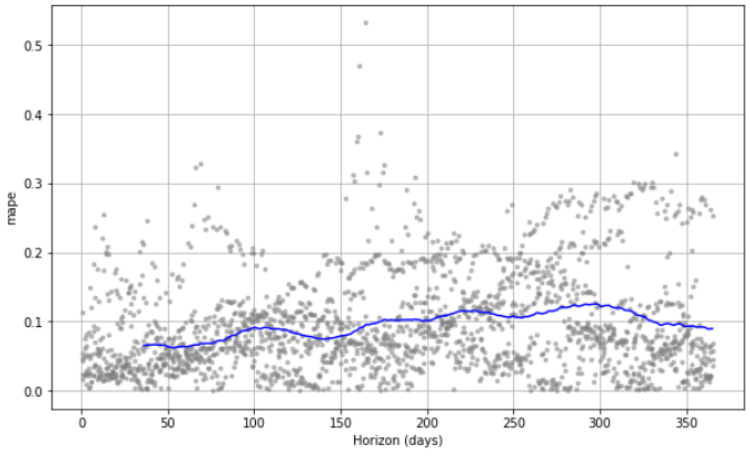


Imagen 17. Error porcentual absoluto medio para diferentes horizontes de predicción de Visa

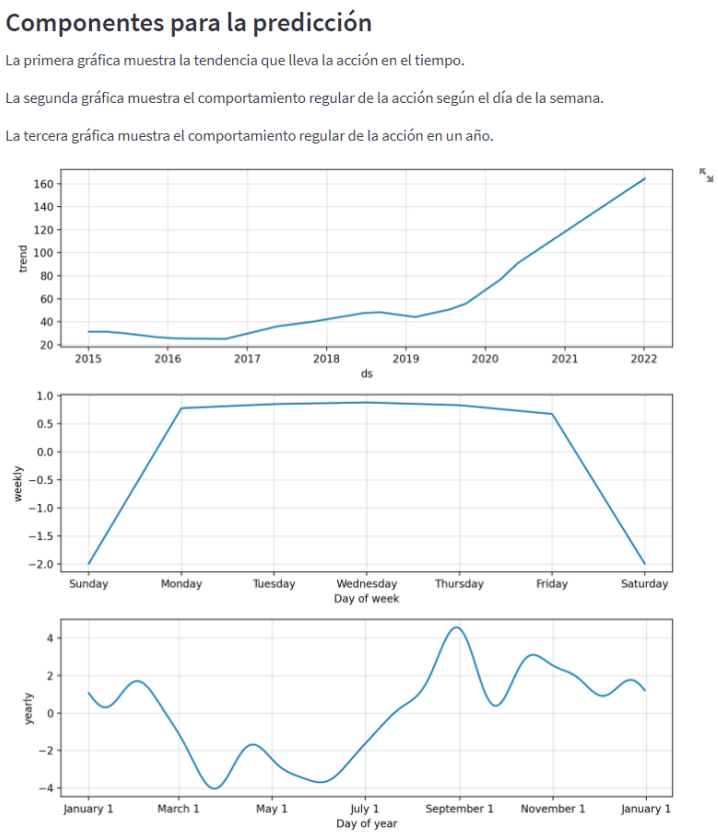
Se puede consultar el cuaderno de python completo y todas las métricas del modelo de cada una de las acciones en la siguiente carpeta: https://github.com/pescapescatarian/TradingPredictionsProject/tree/main/MetricasPredictProphet

Visualización de resultados

La visualización de los resultados se realizó a través de la aplicación web desarrollada con Streamlit. Este servicio cuenta con herramientas que permiten, desde el código, generar gráficas que muestran los resultados deseados. En particular, por el tipo de datos a mostrar, utilizamos gráficas de líneas para los datos históricos y de dispersión para las predicciones.

Imagen 18. Gráfico de líneas para datos históricos

Imagen 19. Gráfico de dispersión para predicciones



Componentes de Series de Tiempo de FB Prophet

Generación de reportes de visualización en línea

La aplicación fue construida de tal forma que el usuario tuviera la oportunidad de, en tiempo real, ir jugando con los datos y solicitudes y que se fueran generando los resultados y las visualizaciones a demanda. De cierta manera, la aplicación y el reporte es una sola cosa.

Adicional a la visualización de las predicciones, creamos una sección en la que los asesores de inversión podrían monitorear otro tipo de recursos como criptomonedas, forex, índices y futuros. Toda la información se obtiene de Yahoo finance y se despliega utilizando las librerías de Streamlit.

Implementación de la arquitectura de nube

API

Antes de comenzar el despliegue de nuestros servicios necesitamos hacer algunas cosas. Primero comenzamos por crear un repositorio de imágenes de contenedores, esto lo haremos en GCP por medio de Artifact Registry.

Imagen 20. Creación de registro

Posteriormente necesitamos crear el cluster que albergará nuestros contenedores, por lo que creamos uno utilizando GKE

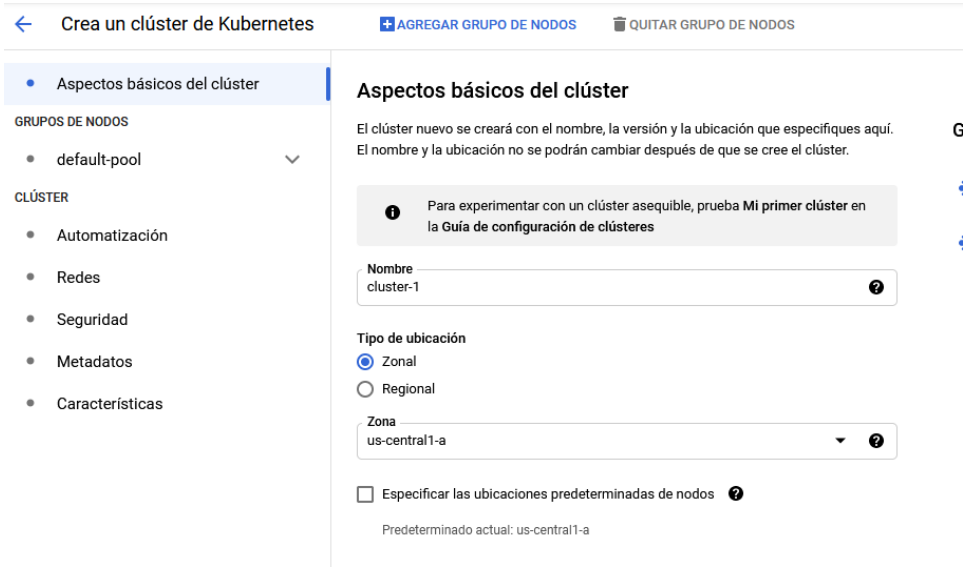


Imagen 21.Creación Cluster

Una vez listo nuestro cluster deberíamos de ver nuestro cluster de esta forma:

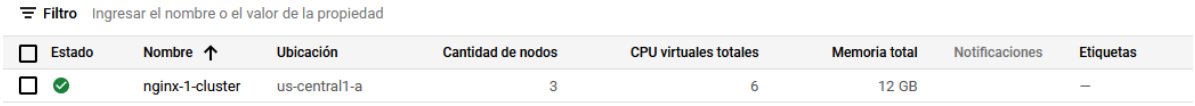


Imagen 22. Validación estado cluster

Después, una vez desarrollada nuestra API de predicciones con el modelo, lo primero que debemos hacer para su despliegue es crear un Dockerfile donde definiremos el contenedor. En este caso tomamos la imagen de Python 3.8 como base y sobre esta instalamos las dependencias necesarias y establecemos los comandos de que se correrán al despliegue que son el entrenamiento de los modelos y levantar el servlet .

```
FROM python:3.8

WORKDIR /app

RUN apt-get -y update  && apt-get install -y \
  python3-dev \
  apt-utils \
  python-dev \
  build-essential \
&& rm -rf /var/lib/apt/lists/*

RUN pip install --upgrade setuptools
RUN pip install \
    cython==0.29.24 \
    numpy==1.21.1 \
    pandas==1.3.1 \
    pystan==2.19.1.1

RUN pip install convertdate==2.1.2 lunarcalendar==0.0.9 holidays==0.10.3
RUN pip install tqdm --upgrade
RUN pip install gunicorn
COPY requirements.txt .
RUN pip install -r requirements.txt

COPY . .

RUN python model_setup.py
CMD gunicorn -w 3 -k uvicorn.workers.UvicornWorker main:app --bind 0.0.0.0:$PORT
```

Imagen 23. Dockerfile API

Una vez todos nuestros archivos estén listos los mandamos a nuestro repositorio de Guthub.

```
(env) [slappy@fedora TradingPredictionsProject]$ git add .
(env) [slappy@fedora TradingPredictionsProject]$ git commit -m "API READY"
[main 91123f0] API READY
 4 files changed, 33 insertions(+), 7 deletions(-)
 create mode 100644 fastapi/model_setup.py
(env) [slappy@fedora TradingPredictionsProject]$ git push origin
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 16 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 1023 bytes | 1023.00 KiB/s, done.
Total 7 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To https://github.com/pescapescatarian/TradingPredictionsProject
   e7faff9..91123f0  main -> main
```

Imagen 24. Version Control API

Para facilitar el despliegue sin tener que configurar nuestro ambiente local aprovecharemos que GCP nos proporciona una máquina con docker instalado en CloudShell, por lo que simplemente importamos nuestro repositorio de github y configuramos docker para la región de nuestro proyecto con el siguiente comando.

```
portizmm@cloudshell:~ (proyectofinal-326802)$ gcloud auth configure-docker us-central1-docker.pkg.dev
WARNING: Your config file at [/home/portizmm/.docker/config.json] contains these credential helper entries:

{
  "credHelpers": {
    "gcr.io": "gcloud",
    "us.gcr.io": "gcloud",
    "eu.gcr.io": "gcloud",
    "asia.gcr.io": "gcloud",
    "staging-k8s.gcr.io": "gcloud",
    "marketplace.gcr.io": "gcloud"
  }
```

Imagen 25. Setum Cloud Shell

Construimos la imagen de nuestro contenedor con Docker y el Dockerfile de nuestro repositorio.

```
portizmm@cloudshell:~/TradingPredictionsProject/fastapi (proyectofinal-326802)$ docker build -t api-forecast .
Sending build context to Docker daemon  8.704kB
Step 1/12 : FROM python:3.8
3.8: Pulling from library/python
df5590a8898b: Pulling fs layer
705bb4cb554e: Pull complete
519df5fceacd: Pull complete
ccc287cbeddc: Pull complete
e3f8e6af58ed: Pull complete
aebed27b2d86: Pull complete
c98d5c4ff928: Pull complete
bf21952cbf7e: Pull complete
6f80715d8390: Pull complete
Digest: sha256:79f6327d01cb57fedfe38cfbf9e525dc46ba9d1ca3df28b8a4cb74d81f46b7ec
Status: Downloaded newer image for python:3.8
```

Imagen 26. Creación imagen contenedor

Etiquetamos la imagen de forma adecuada y la subimos a nuestro Artifact registry.

```
portizmm@cloudshell:~/TradingPredictionsProject/fastapi (proyectofinal-326802)$ docker tag api-forecast:latest us-central1-docker.pkg.dev/proyectofinal-32680
2/container-repo/api-forecast:v1
portizmm@cloudshell:~/TradingPredictionsProject/fastapi (proyectofinal-326802)$ docker push us-central1-docker.pkg.dev/proyectofinal-326802/container-repo/ap
i-forecast:v1
The push refers to repository [us-central1-docker.pkg.dev/proyectofinal-326802/container-repo/api-forecast]
d759f94749c4: Pushed
6b85164decac: Pushed
26bbce0ba582: Pushed
2566c5f55c34: Pushed
171d99a3d6a1: Pushed
a0257e767895: Pushed
913888d474d6: Pushed
eb2d8d4f027c: Pushed
b01a4679eb80: Pushed
b89a111dd8bc: Pushed
b205930be12f: Pushed
f38f5a1e3486: Pushed
```

Imagen 27. Push contenedor a registro

Validamos que nuestro contenedor esté guardado correctamente en nuestro repositorio de Imágenes.

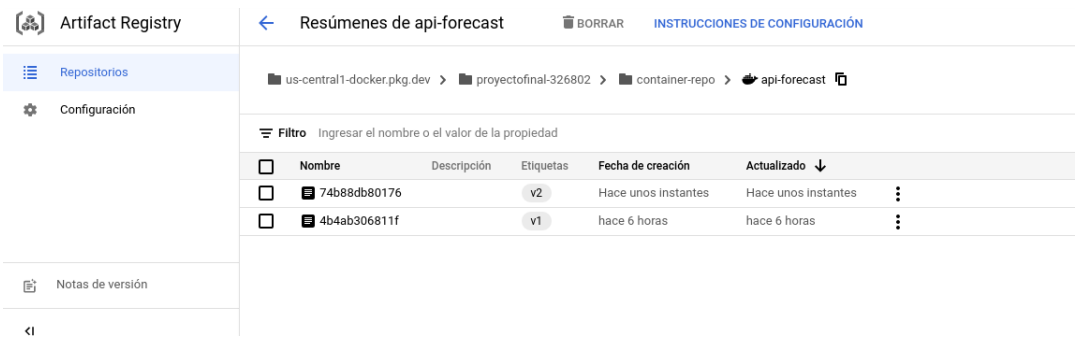


Imagen 28. Validación Artifact registry

Con la imagen arriba, ahora podemos desplegar nuestro contenedor en nuestro cluster. Simplemente configuramos el despliegue desde la interfaz.

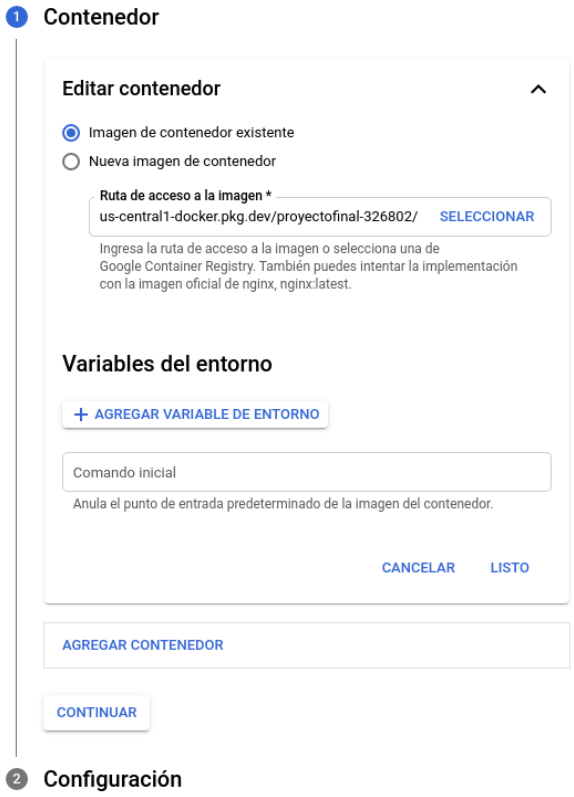


Imagen 29. Creacion despliegue GKE

Seleccionamos la imagen que deseamos desplegar.

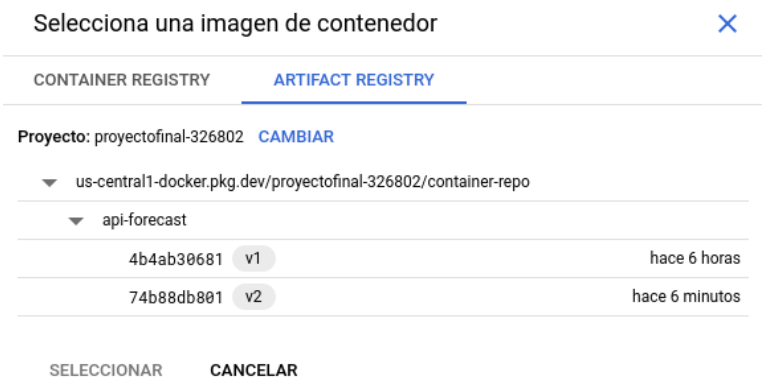


Imagen 30. Selección de imagen de contenedor

Establecemos las variables de entorno necesarias. En nuestro caso el puerto del servidor. Y lo desplegamos.

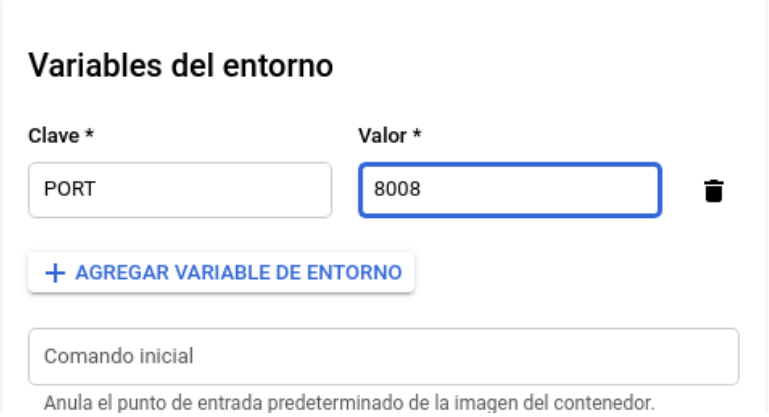


Imagen 31. Variables Entorno contenedor

Cuando nuestro contenedor está listo lo veremos en nuestra lista de cargas de trabajo, desde aquí podemos monitorear de forma fácil.

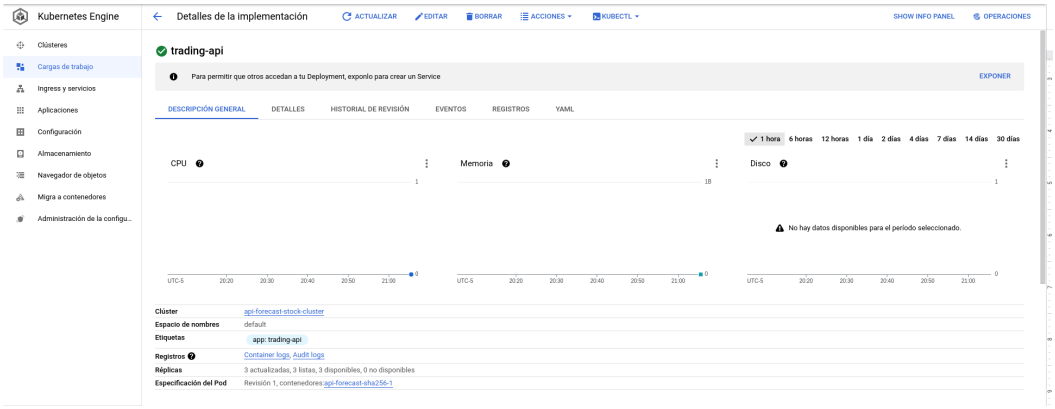


Imagen 32.Despliegue exitoso contenedor

Podemos observar todas las réplicas de nuestro contenedor, en este caso está establecido en 3.

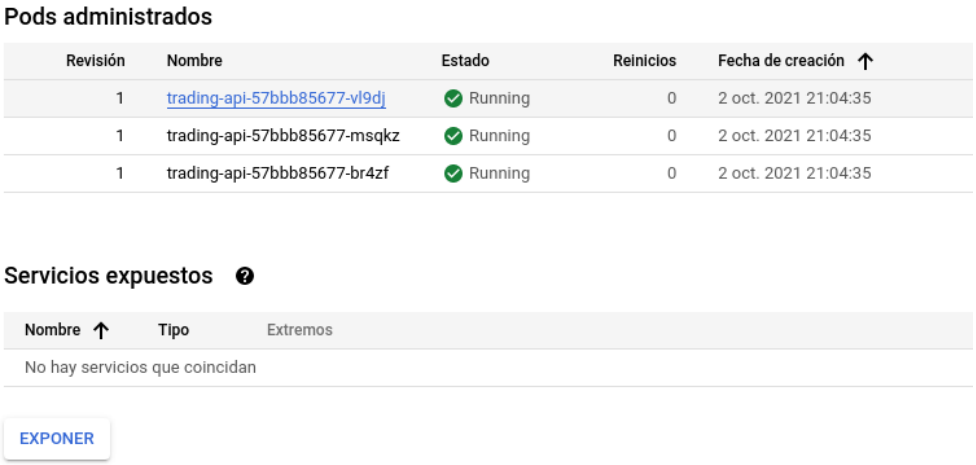


Imagen 33. Pods Running GKE

Los contenedores están aislados nativamente. por lo que debemos exponerlos como un servicio, haciendo el port mapping de los puertos del contenedor a los del balanceador de cargas del cluster. En este caso lo queremos exponer al público, entonces debe ser un Load Balancer Service, donde podremos acceder a este desde la IP externa del Cluster.

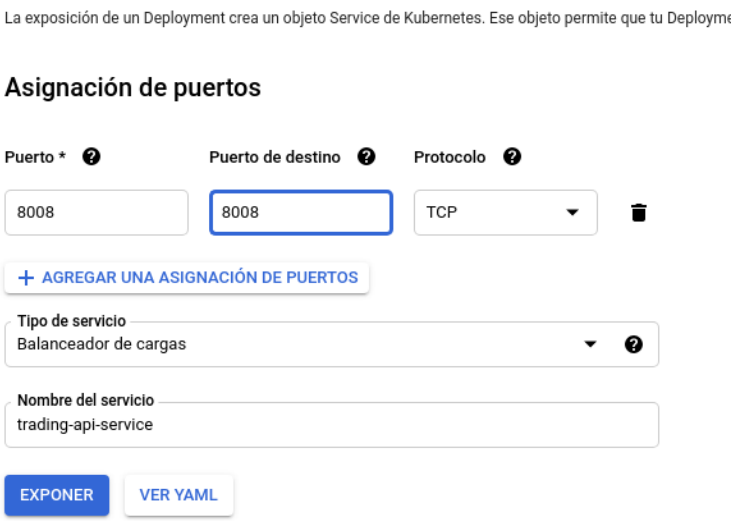


Imagen 34. Service Expose GKE

Una vez nuestro servicio esté listo, podemos hacer pruebas tanto utilizando Curl.

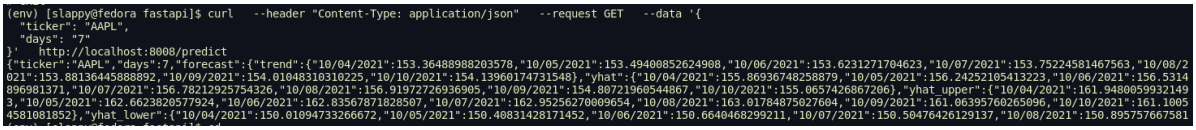


Imagen 35.Curl al API

Como desde nuestro navegador.

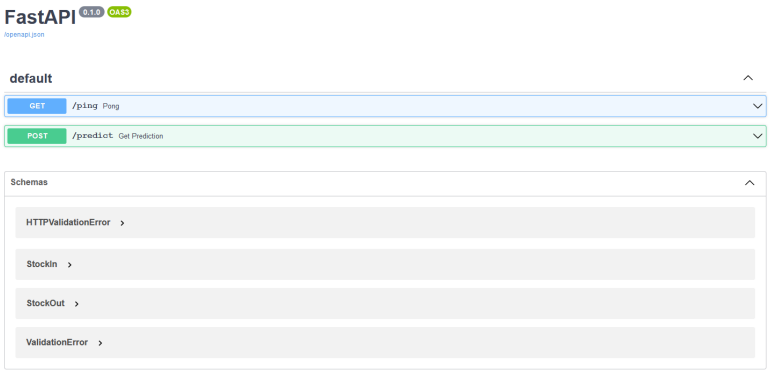


Imagen 37. OpenAPI docs del API desplegado

Dashboard

Para el despliegue de nuestro tablero, decidimos utilizar una Instancia de compute engine ya que esta puede soportar mejor la pesada carga de trabajo del tablero. Lo primero que necesitamos es crear una instancia con los requerimientos necesarios.

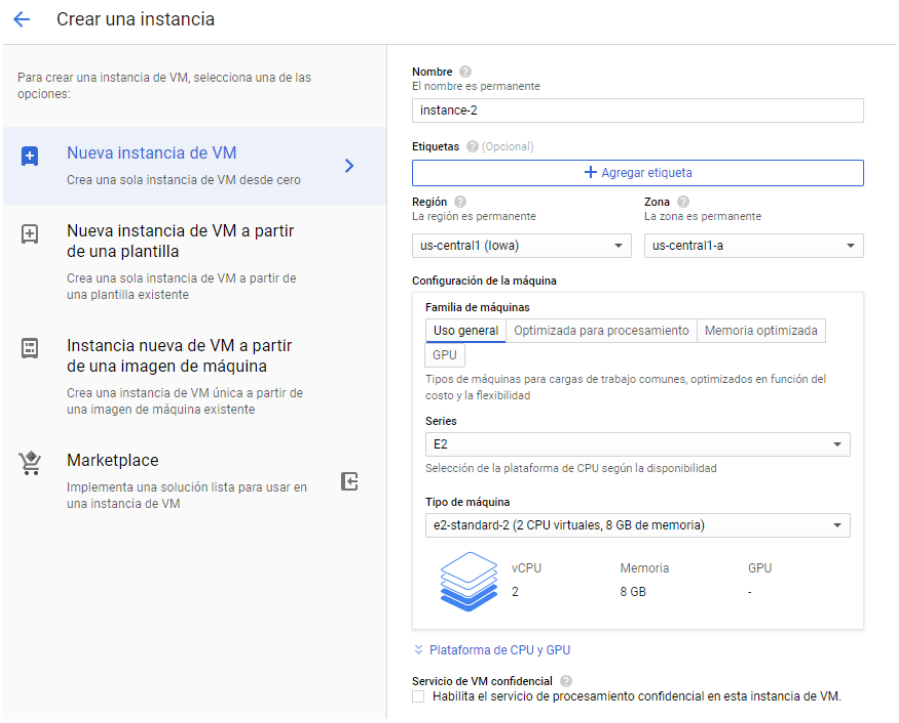


Imagen 38. Creación Instancia Compute Engine

Instalar todas las librerías y paquetes necesarios, la secuencia de comandos es la siguiente:

1. sudo dnf install git
2. sudo dnf group install "Development Tools"
3. sudo dnf install python3-devel -y
4. sudo python3 -m pip install --upgrade setuptools
5. sudo python3 -m pip install cython numpy pandas pystan==2.19.1.1
6. sudo python3 -m pip install convertdate==2.1.2 lunarcalendar==0.0.9 holidays==0.10.3
7. sudo python3 -m pip install wheel
8. sudo python3 -m pip install prophet
9. sudo python3 -m pip install plotly yfinance datetime
10. sudo dnf install cmake
11. sudo python3 -m pip install --upgrade pip
12. sudo python3 -m pip install pyarrow
13. sudo python3 -m pip install streamlit

Antes de levantar el servidor necesitamos establecer las reglas de firewall necesarias para permitir tráfico http desde el exterior. Streamlit utiliza el puerto 8501, por lo que agregamos el puerto al firewall de la máquina.

```
[portizmm@instance-1 ~]$ sudo firewall-cmd --permanent --add-port 8501/tcp

success
[portizmm@instance-1 ~]$ sudo firewall-cmd --reload
success
```

Imagen 39.Configuración firewall CentOS

Y de igual manera creamos la regla en nuestro proyecto de GCP con Firewall Rules.

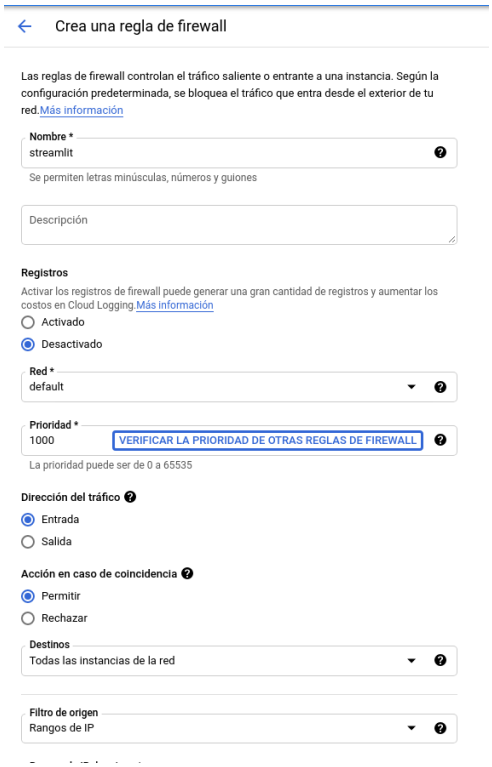


Imagen 40. Configuración Firewall GCP

Ahora ya podemos clonar nuestro repositorio de github y desplegar el servidor para visualizar nuestro dashboard.

```
[portizmm@instance-1 webapp]$ streamlit run app.py

  You can now view your Streamlit app in your browser.

  Network URL: http://10.128.0.10:8501
  External URL: http://35.226.244.108:8501
```

Imagen 41. Despliegue Streamlit

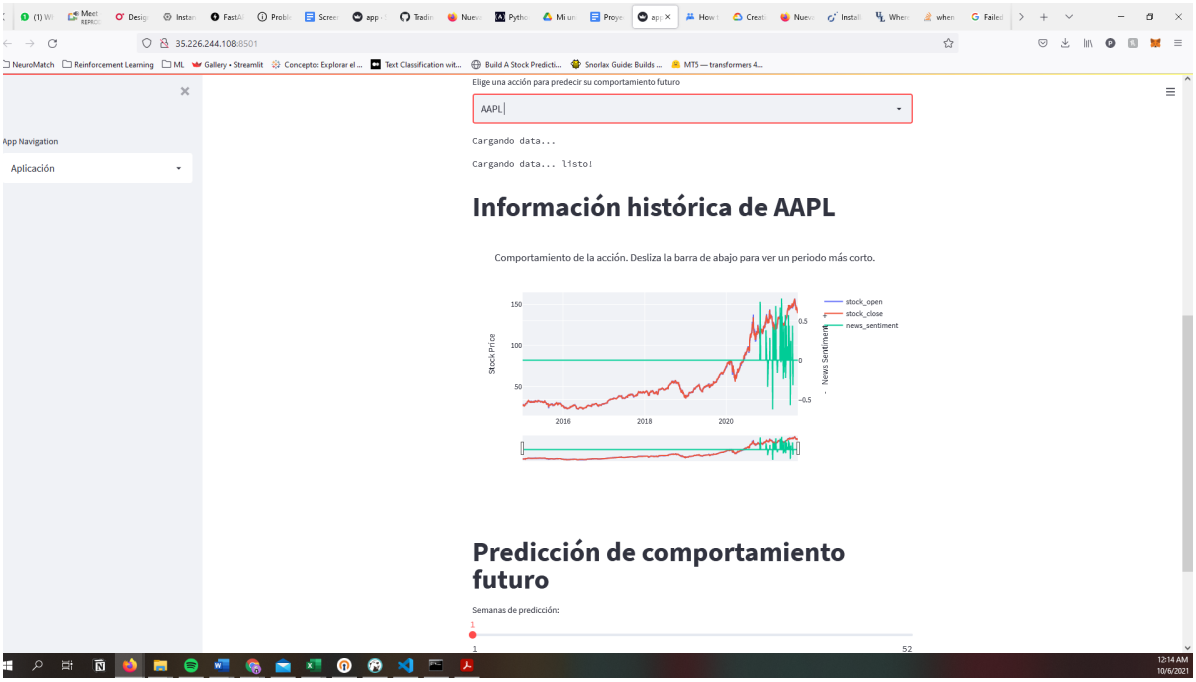


Imagen 42. Tablero en Streamlit

Conclusiones

1. El modelo generado es útil para predecir el valor de las acciones en un horizonte de tiempo de aproximadamente corto (menor a 100 días) para horizontes más largo.
2. Las noticias tienen un impacto en el valor de las acciones en los días subsecuentes a partir de su publicación, pero no inmediatamente, por lo cuál tiene un valor agregado incluir el análisis de sentimientos de las acciones de manera que las noticias sean cuantificables.
3. La creación de cualquier servicio debe tener siempre una aplicación que sea verdaderamente útil para el negocio o los usuarios que se quieran. En ese sentido, decidir el caso de uso es crítico previo a desarrollar cualquier cosa.
4. La creación de una aplicación que facilite el cruce de distintas fuentes de información para los fondos de inversión es realmente útil ya que ayuda a eficientar procesos al concentrar varias cosas en el mismo lugar.
5. Para diversos fondos de inversión es atractivo voltear a ver a los servicios de la nube para albergar no solo aplicaciones como esta sino distintos servicios internos y externos. Esto les permitirá escalar y mantener vigente la capacidad y poder de su infraestructura y servicios de cómputo.
6. La generación de servicios en la nube debe, desde un inicio, contemplar no solo las funcionalidades generadas con el código sino también los requerimientos de infraestructura para evitar retrabajos y la necesidad de cambiar avances por falta de planeación y mal diseño
7. El mundo de los servicios en la nube deberá ser cada vez más y mejor explotado por los proveedores de estos servicios a través de la atracción de empresas pequeñas y medianas que verán en los servicios las ventajas de no crecer la infraestructura física y los esquemas atractivos de pago por uso.
8. Existen muchas formas de resolver el mismo problema con las distintas soluciones que nos proveen los cloud providers. Por lo que, cada quien puede adaptar su arquitectura a lo que sea más cómodo.
9. Si bien las tecnologías de estado del arte como Kubernetes son muy útiles, para desarrollos ágiles y escalables, su curva de aprendizaje es mucho mayor y requerirá siempre costos de capacitación y tiempo adicional de implementación.

Referencias

- Google. (n.d.). *Documentation | google cloud*. Google. Recuperado el 16 de septiembre de 2021, desde https://cloud.google.com/docs.
- *Forecasting at scale.* Prophet. (n.d.). Recuperado el 6 de septiembre de 2021, desde https://facebook.github.io/prophet/.
- *Welcome to streamlit*. - Streamlit 1.0.0 documentation. (n.d.). Recuperado el 4 de septiembre de 2021, desde https://docs.streamlit.io/en/stable/
- Reese, G., Cloud Application Architectures: Building Applications and Infrastructure in the Cloud, O'Reilly, 2009.
- Geewax, J., Google Cloud Platform in Action, Manning, 2018.