Workshop GCP DESARROLLO

Prerrequisitos (Instalar)

1.1. Python 3.10
1.2. Git local
1.3. Docker
1.4. Visual Code
1.5. Extension Cloud Code
1.6. SDK Cloud (https://cloud.google.com/sdk/docs/install-sdk#windows)

Primero comenzaremos por conocer kubernetes, desplegar una aplicación, luego pasaremos por spanner para conectar la aplicación django desplegada en kubernetes y por último gestionar los registros generados con las herramientas anteriores en cloud loggin.

1. Kubernete

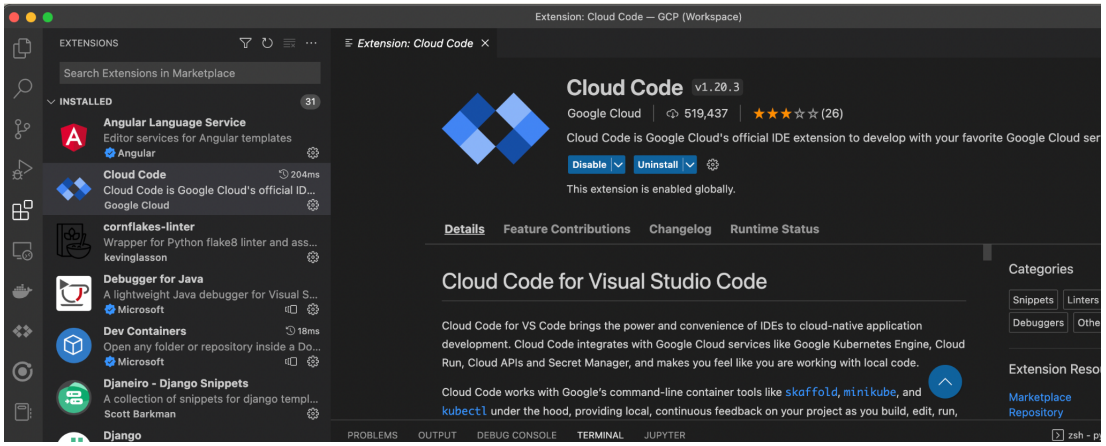
Descripción
Google Kubernetes Engine (GKE) proporciona un entorno administrado para implementar, administrar y escalar las aplicaciones en contenedores mediante la infraestructura de Google. El entorno de GKE consta de varias máquinas (en particular, instancias de Compute Engine) que se agrupan para formar un clúster.

Características claves

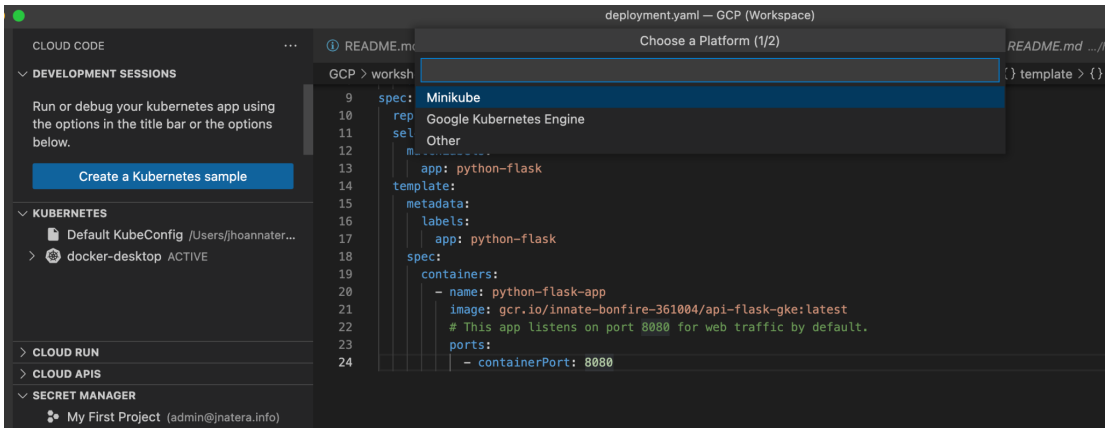
1. Ajuste de escala automático de Pods y clústeres
2. Aplicaciones y plantillas predefinidas de Kubernetes
3. Seguridad y redes nativas de contenedores
4. Migrar cargas de trabajo tradicionales a contenedores de GKE con facilidad
5. Administración de identidades y accesos
6. Supervisión y registros integrados.
7. Opciones de clústeres.
8. Ajuste de escala automático.
9. Aislamiento del contenedor.
10. Compatibilidad con imágenes de Docker.
11. Portabilidad de las cargas de trabajo locales y en la nube.
12. Balanceo de cargas global.
13. Canales de versiones.
14. Seguridad de la cadena de suministro del software
15. Facturación por segundo

1.1. Instalar de Cloud Code y Crear cluster en GCP

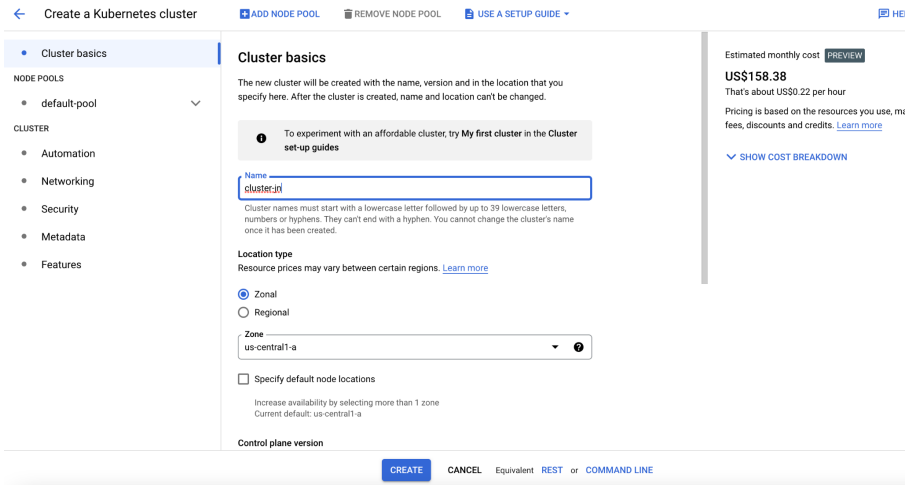
- Vamos a loguearnos en google desde la consola con gcloud auth login
- Instalamos la extensión Cloud Code, para administrar Kubernetes desde nuestro IDE Visual Code, para ello vamos al área de extensiones para proceder a instalar.

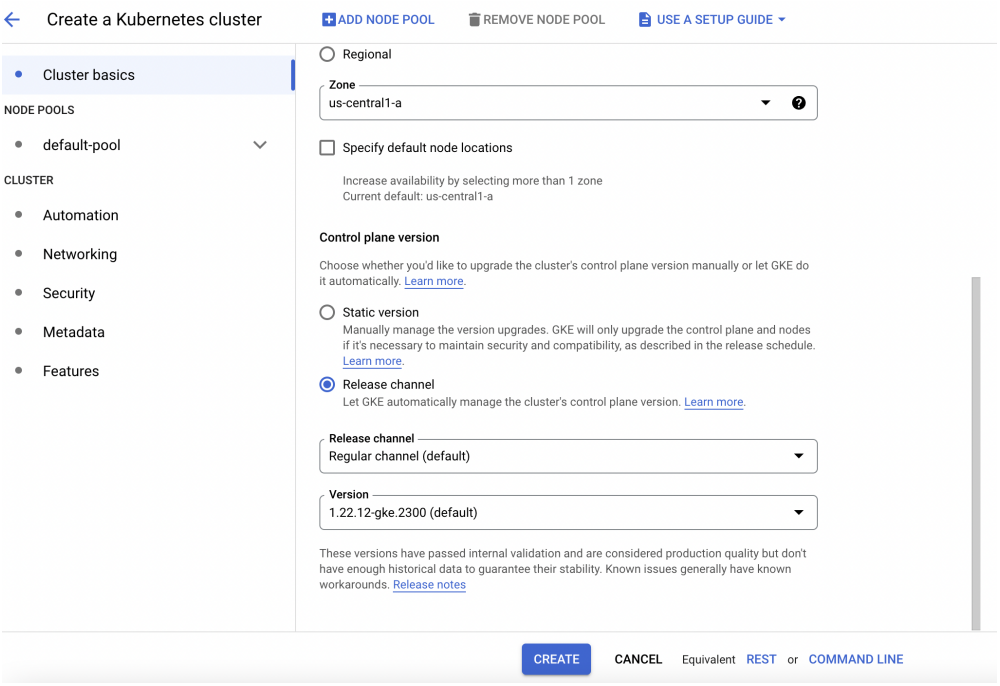


- Luego vamos a abrir esta extensión y presionamos botón de CREATE NEW CLUSTER y nos saldrá la siguiente imagen y luego seleccionamos Google Kubernetes Engine y abrirá la página para crear cluster en la nube.

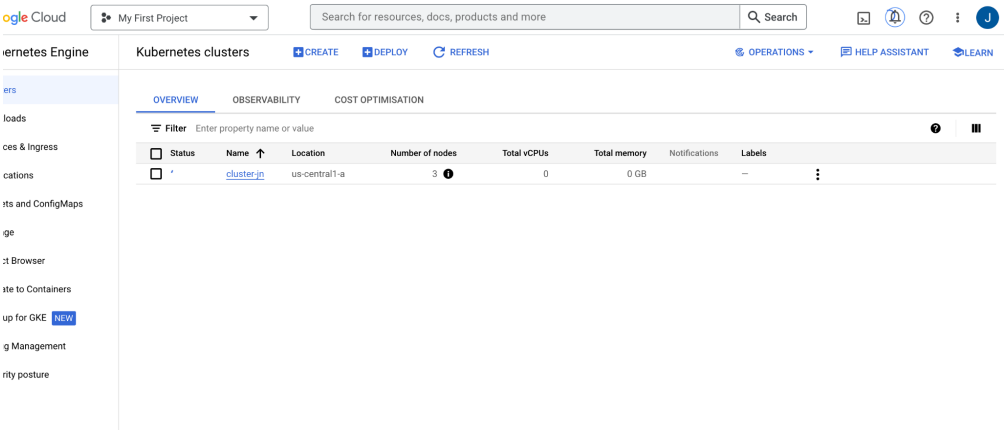


- A continuación ingresamos el nombre del cluster y mantenemos el resto de información por defecto.

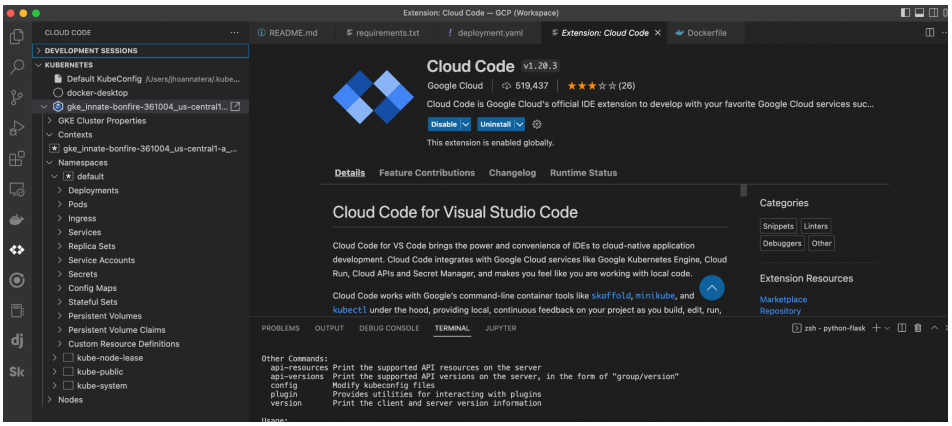




- Luego esperamos unos minutos a que se cree el nuevo cluster creado.



- Luego de terminar el proceso de creación del cluster vía web, le damos refrescar en IDE para que aparezca el nuevo cluster creado, al desplegar podemos visualizar todas las opciones necesarias para administrar kubernetes y otros servicios relacionados.



- A continuación descargamos el repositorio para los códigos de prueba.
  git clone https://github.com/jnatera/workshop-izzi-dev.git

- Luego nos ubicamos en la carpeta “Kubernetes/python-flask”
- Vamos a comprobar que la aplicación de prueba funciona corriendo en docker local

```
docker build -t kube-app .


docker run -e PORT 8080  -p 8080:8080 kube-app
```

  - Compilamos la imagen en la nube

```
gcloud builds submit --tag
gcr.io/$GOOGLE_CLOUD_PROJECT/python-flask-nombre
```

  - Mantener seleccionado el cluster de gcp en vez del que se encuentra por defecto en la extensión cloud code “Desktop”

1.2. Creación de un Deployment

- Primero adaptamos el nombre del proyecto en el archivo yaml para que se registre la imagen gke.
- Nos ubicamos en la carpeta python-flask
- Compilamos la imagen con el siguiente comando

```
gcloud builds submit --tag
gcr.io/$GOOGLE_CLOUD_PROJECT/python-flask-nombre-gke .
```

  - Cambiamos el nombre del proyecto en el archivo deployment.yaml para poderlo aplicar.

```
image: gcr.io/cambiar-aqui/python-flask-nombre-gke:latest
```

  - Ejecutamos el siguiente comando para aplicar los cambios en un deployment.

```
kubectl apply -f gke/dev/1-deployment.yaml
```

  - Con el siguiente comando podemos verificar si el deployment se encuentra ready. El parámetro -w permite esperar que exista cambios en el cluster para refrescar la consola.

```
kubectl get deployments -w
```

1.3. Creación de Services

- Luego que se encuentre el deployment generado, procedemos a crear el services para que pueda exponerse el servicio hacia el exterior con una ip pública en este caso.
- Ejecutamos el siguiente comando para aplicar los cambios.

```
kubectl apply -f gke/dev/2-service.yaml
```

- Ahora confirmamos que se encuentra desplegado el servicio y con una ip pública generada.

```
kubectl get services
```

1.4. Creación de Ingress

- Ejecutamos el comando para aplicar cambios de ingress

```
kubectl apply -f gke/dev/3-ingress.yaml
```

1.5. Uso de Skaffold

Skaffold es una herramienta de línea de comandos que permite el desarrollo continuo para aplicaciones nativas de Kubernetes. Puedes usar Skaffold a fin de configurar un lugar de trabajo de desarrollo local para usarlo con las canalizaciones de entrega continua de Google Cloud Deploy.

Estas son algunas de las ventajas que proporciona Skaffold cuando se usa con Google Cloud Deploy:

- **Integración sencilla**
  Comienza con un bucle de desarrollo local. Puedes compartir el archivo skaffold.yaml con tu equipo, lo que puede ayudar a que los miembros nuevos del equipo sean coherentes.
- **Control coherente** sobre la renderización en diversos destinos de implementación
  Puedes usar perfiles de Skaffold, con implementación y renderización diferentes para distintos destinos.
- **Elección de herramientas de renderización** sin comprometer cómo se definen las canalizaciones de entrega
  El uso de Skaffold permite que Google Cloud Deploy separe la definición de la canalización de entrega de los detalles de procesamiento. Esta separación te permite experimentar con tus manifiestos sin interrumpir la canalización de entrega.
- **Proceso de renderización coherente**
  La fuente de renderización y los contenedores se usan para generar manifiestos renderizados.
- **Verificaciones de estado de la implementación**
  Skaffold las realiza y Google Cloud Deploy las usa.

- Nos vamos a la carpeta hello-work
- Cambiamos el nombre del proyecto en los archivos yaml que corresponda.
- Ejecutamos el siguiente comando para que realice todo el trabajo configurado incluyendo compilar la imagen y de esta forma nos evitamos el conjunto de pasos

```
skaffold run -f skaffold.yaml
```

-

2. Cloud SQL Spanner

2.1. Qué es, ventajas y características claves
2.2. Crear una instancia
2.3. Crear una base de datos
2.4. Uso de Spanner emulator
2.5. CRUD para editar, eliminar, crear y consultar desde la aplicación Django.

3. Cloud Loggin

3.1. Qué es, ventajas y características claves
3.2. Explorar registros
3.3. Registros de auditorías
3.4. Uso de Error reporting
3.5. Enrutador de registros.