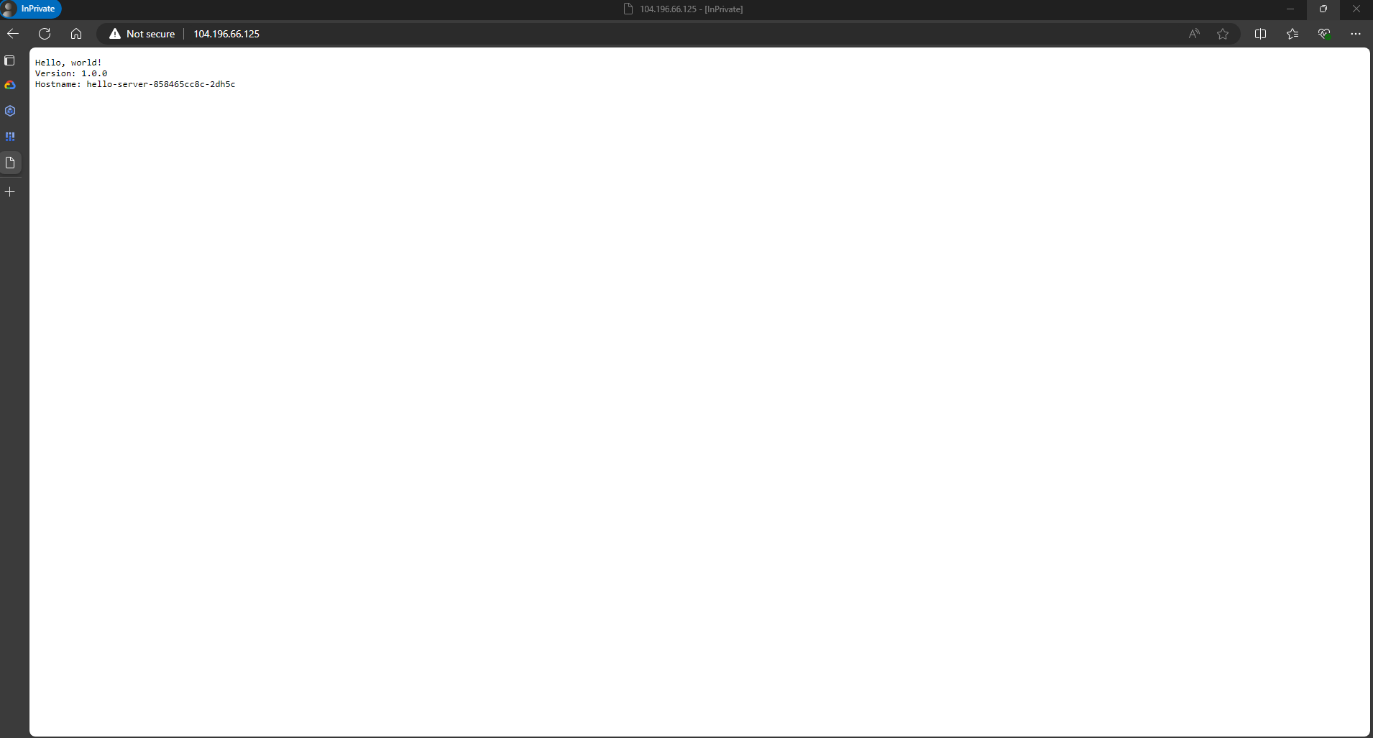
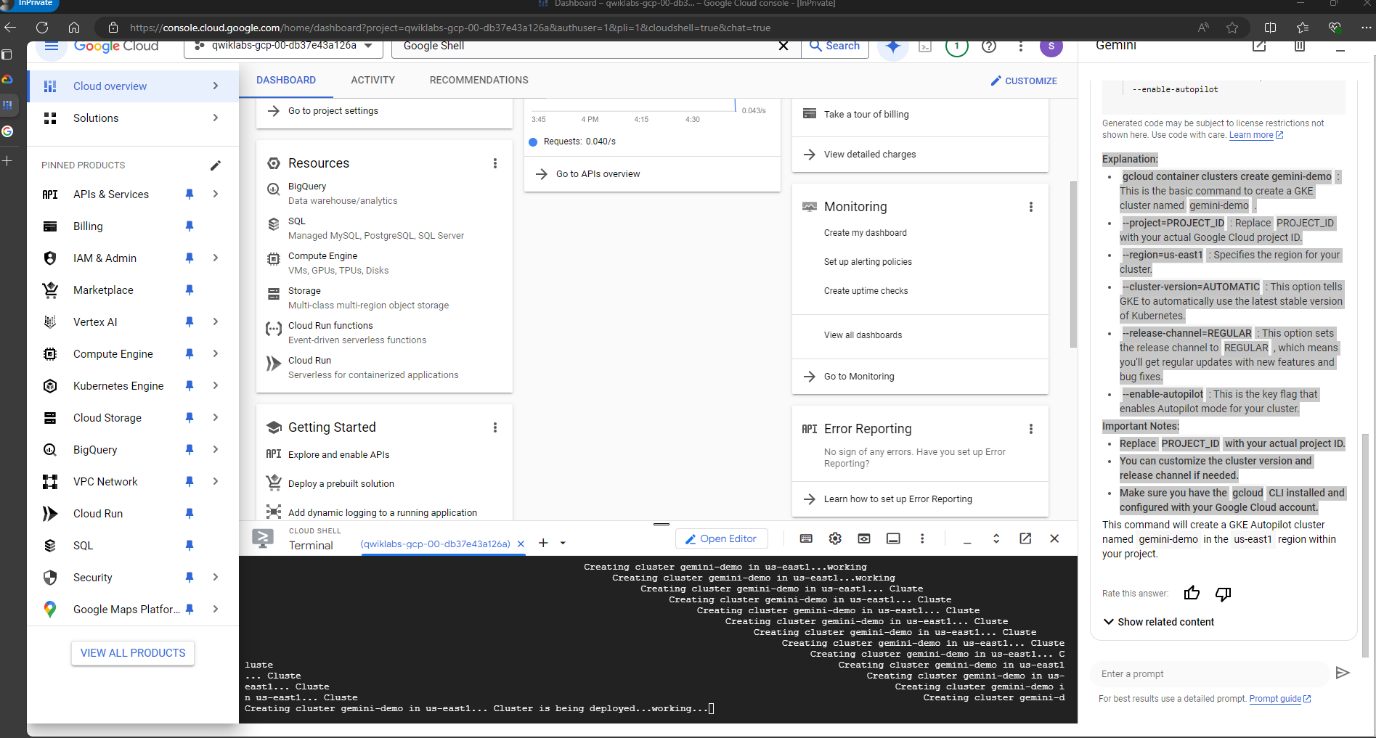
****

**GOOGLE CLOUD SERIES**

**PART 1**

**I am new to Google Cloud and I want to use the Cloud Code extension. Give me some examples of Google services that I can use to build and deploy a sample app.**

Here are some examples of Google Cloud services you can use with Cloud Code to build and deploy a sample app, along with ideas for simple projects:

**1. Google App Engine (Flexible Environment):**

* **What it is:** A fully managed platform for deploying and scaling web applications. You don't manage VMs; focus on code. Flexible environment gives you more control over runtime and OS.
* **Sample Project:** A simple web app with a form that collects user data and stores it in a database (see Cloud SQL below).
* **Languages:** Wide range (Java, Python, Node.js, Go, PHP, .NET).

**2. Google Cloud Run:**

* **What it is:** Serverless platform to run containers. Scales down to zero when not in use, only paying when your code runs.
* **Sample Project:** An API that processes images using a pre-trained machine learning model (see Vertex AI below for model options).
* **Languages:** Any language that can run in a container (Docker image).

**3. Google Kubernetes Engine (GKE):**

* **What it is:** Managed Kubernetes for deploying, managing, and scaling containerized applications. More complex, but very powerful for larger apps.
* **Sample Project:** A microservices-based application where each service runs in a separate container and communicates with others.
* **Languages:** Any language that can run in a container.

**Additional Services to Integrate:**

* **Cloud SQL:** Managed database services (MySQL, PostgreSQL, SQL Server). Use this to store data for your app.
* **Cloud Storage:** Object storage for files. Use this to store user uploads, static assets, etc.
* **Vertex AI:** Machine learning platform. Use pre-trained models or build your own.

**Getting Started with Cloud Code:**

1. **Install Cloud Code:** Get the extension for your IDE (VS Code, IntelliJ, etc.).
2. **Choose a Tutorial:** Cloud Code has excellent tutorials for App Engine, Cloud Run, and GKE. Start with one that matches your chosen service and language.
3. **Deploy and Explore:** Cloud Code makes it easy to deploy your sample app to Google Cloud and see it running.

Remember, these are just starting points. Google Cloud offers a vast ecosystem of services. As you gain experience, you can explore and integrate more services to build more sophisticated applications.

What is the difference between Cloud Run and Cloud Functions?

Here's a breakdown of the key differences between Cloud Run and Cloud Functions to help you choose the right service:

**Cloud Functions:**

* **Focus:** Event-driven computing. Functions are triggered by specific events like file uploads, database changes, or HTTP requests.
* **Granularity:** Designed for smaller, self-contained functions that perform a single task.
* **Scaling:** Scales automatically based on the number of events. Can scale down to zero when inactive.
* **Management:** Minimal infrastructure management. You focus solely on your code.
* **Cold Starts:** Can experience noticeable cold starts if a function hasn't been used recently.
* **Execution Time Limits:** Stricter limits on how long a function can run (usually a few minutes).

**Cloud Run:**

* **Focus:** Running containerized web applications or APIs. More versatile for general-purpose workloads.
* **Granularity:** Can handle larger, more complex applications composed of multiple containers.
* **Scaling:** Also scales automatically based on traffic. Can scale down to zero, but cold starts might be more noticeable than Functions.
* **Management:** More control over the container environment and dependencies.
* **Execution Time Limits:** More generous time limits, making it suitable for longer-running processes.

**In a Nutshell:**

* **Choose Cloud Functions for:** Short-lived, event-driven tasks where you want minimal operational overhead.
* **Choose Cloud Run for:** Web applications, APIs, or longer-running processes that benefit from containerization and more control over the environment.

Let me know if you have any other questions.

**Task 5. Develop a Python app**

How do I create a new Cloud Run app in Cloud Code using the command palette? What languages are supported?

To create a new Cloud Run application in Cloud Code using the command palette, follow these steps:

1. **Open the command palette:** Press Ctrl+Shift+P (Windows/Linux) or Cmd+Shift+P (macOS).
2. **Start typing "Cloud Code: Create New Application":** Select this option from the dropdown menu.
3. **Choose a location for your project:** Select an existing folder or create a new one.
4. **Select "Cloud Run" as the deployment target.**
5. **Choose your preferred language:** Cloud Code supports the following languages for Cloud Run:
   * **Go**
   * **Java**
   * **Node.js**
   * **Python**
   * **PHP**
   * **.NET**
6. **Select a sample application:** Cloud Code provides several pre-built templates to help you get started quickly.
7. **Provide a name for your application:** This will be used as the name of your Cloud Run service.
8. **Choose a region for your Cloud Run service.**
9. **Wait for the project to be created:** Cloud Code will generate the necessary files and configure your project for deployment to Cloud Run.

Once the project is created, you can use Cloud Code's built-in tools to build, deploy, and debug your application on Cloud Run.

**PART 2**

**Task 2. Deploy GKE clusters**

**How can I run Kubernetes on Google Cloud without having to own management of nodes and the control plane?**

You can run Kubernetes on Google Cloud without managing nodes and the control plane by using Google Kubernetes Engine (GKE) in Autopilot mode .

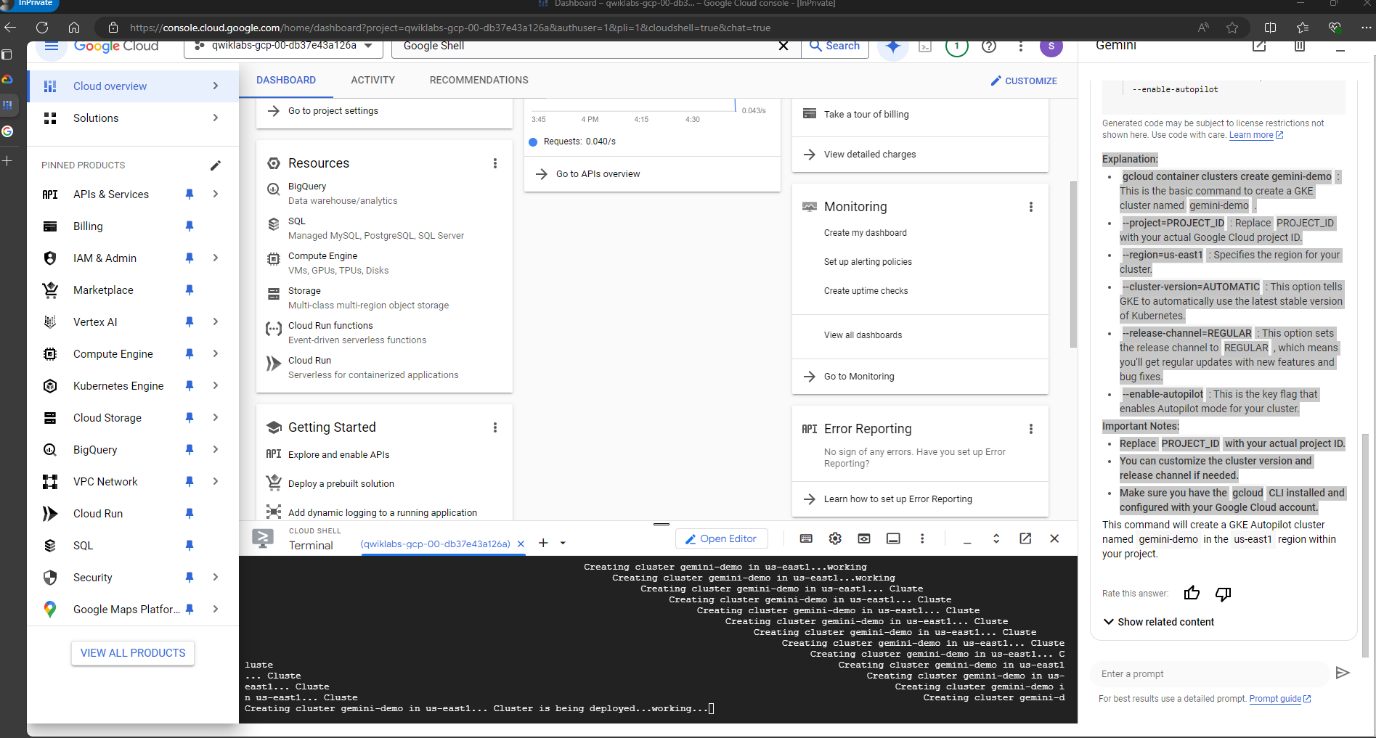
Here's how Autopilot works:

* GKE manages the infrastructure: Google takes care of provisioning, configuring, and managing the underlying nodes and infrastructure for your Kubernetes cluster. This includes scaling, upgrades, and maintenance.
* You focus on your applications: You can concentrate on deploying and managing your applications and services without worrying about the complexities of Kubernetes infrastructure.
* Simplified operations: Autopilot simplifies cluster management, making it easier to get started with Kubernetes and reducing the operational overhead.

To create a GKE cluster in Autopilot mode, you can use the Google Cloud Console, the gcloud command-line tool, or the GKE API.

Autopilot is a great option if you want to:

* Get started with Kubernetes quickly and easily.
* Focus on your applications and services.
* Reduce the operational overhead of managing Kubernetes infrastructure.

Let me know if you have any other questions about GKE or Autopilot.

**Task 3: Deploy a GKE Autopilot cluster**

**How do I create a GKE Autopilot cluster?**

You can create a GKE Autopilot cluster using the Google Cloud Console, the Google Cloud CLI, or Terraform.

Here's a breakdown of how to do it using the Google Cloud Console:

1. Navigate to Kubernetes Engine: In the Google Cloud Console navigation menu, click Kubernetes Engine .
2. Click "Create": Click the *add\_box*Create button.
3. Select Autopilot: Click Configure next to the GKE Autopilot option.
4. Configure your cluster:
   * Name: Enter a name for your cluster.
   * Region: Choose a region for your cluster.
   * Node Pools: You can configure node pools if needed.
   * Other settings: Configure other settings like network, authentication, and logging.
5. Click "Create": Click Create to start the cluster creation process.

This will create a GKE Autopilot cluster, which is a managed Kubernetes environment that simplifies cluster management and configuration.

Remember that Autopilot clusters are designed for ease of use and are ideal for applications that don't require fine-grained control over the underlying infrastructure.

**What is the command to create a GKE Autopilot cluster in my current project and in the us-east1 region named gemini-demo using the gcloud CLI?**

**The code,**

gcloud container clusters create gemini-demo \

--project=PROJECT\_ID \

--region=us-east1 \

--cluster-version=AUTOMATIC \

--release-channel=REGULAR \

--enable-autopilot

**Explanation:**

* **gcloud container clusters create gemini-demo : This is the basic command to create a GKE cluster named gemini-demo .**
* **--project=PROJECT\_ID : Replace PROJECT\_ID with your actual Google Cloud project ID.**
* **--region=us-east1 : Specifies the region for your cluster.**
* **--cluster-version=AUTOMATIC : This option tells GKE to automatically use the latest stable version of Kubernetes.**
* **--release-channel=REGULAR : This option sets the release channel to REGULAR , which means you'll get regular updates with new features and bug fixes.**
* **--enable-autopilot : This is the key flag that enables Autopilot mode for your cluster.**

**Important Notes:**

* **Replace PROJECT\_ID with your actual project ID.**
* **You can customize the cluster version and release channel if needed.**
* **Make sure you have the gcloud CLI installed and configured with your Google Cloud account.**

**Task 4 : Deploy a sample web application**

**What is the kubectl command to create a deployment called hello-server for the image us-docker.pkg.dev/google-samples/containers/gke/hello-app:1.0?**

**Solution:**

**The kubectl command to create a deployment called hello-server for the image us-docker.pkg.dev/google-samples/containers/gke/hello-app:1.0 is:**

kubectl create deployment hello-server --image=us-docker.pkg.dev/google-samples/containers/gke/hello-app:1.0

**This command will create a Deployment object in your Kubernetes cluster. The Deployment will use the specified image to create Pods that run your application.**

**What is the kubectl command to expose this deployment on port 80 and target port 8080 with a load balancer?**

kubectl expose deployment hello-server --type=LoadBalancer --port=80 --target-port=8080

**This command will create a Service object in your Kubernetes cluster. The Service will expose the Deployment on port 80, and it will forward traffic to the Pods running the Deployment on port 8080. The --type=LoadBalancer flag tells Kubernetes to create a LoadBalancer service, which will expose the service to the outside world through a public IP address.**

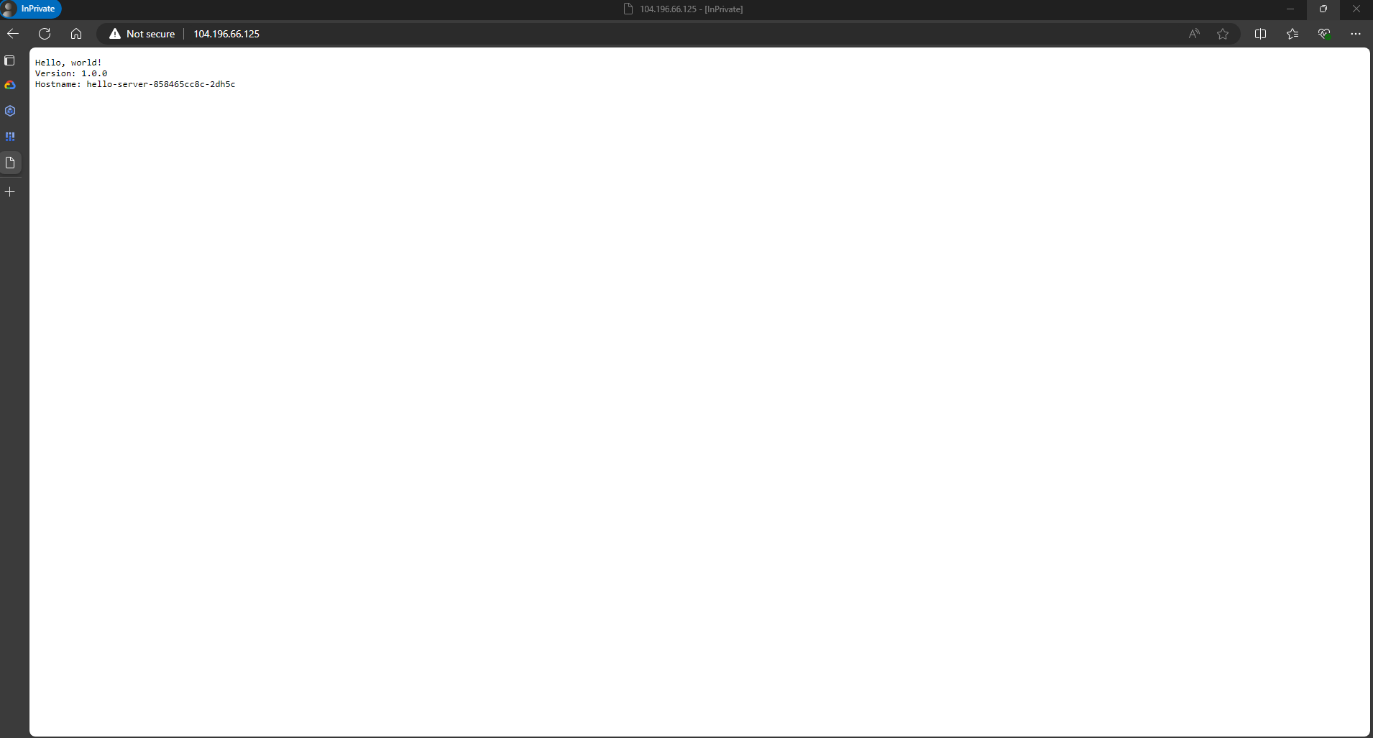
**What is the kubectl command to see the external IP address associated with this load balancer?**

kubectl get service hello-server -o jsonpath='{.status.loadBalancer.ingress[\*].ip}'

This command will output the external IP address of the LoadBalancer service.

**Here's a breakdown:**

* **kubectl get service hello-server : This part retrieves the details of the service named "hello-server".**
* **-o jsonpath='{.status.loadBalancer.ingress[\*].ip}' : This specifies the output format as JSONPath and extracts the IP address from the status.loadBalancer.ingress field.**

**This command will print the external IP address of the LoadBalancer service, which you can then use to access your application from outside the cluster.**