In a **GKE (Google Kubernetes Engine) cluster**, the **API Server** (often referred to as **kube-apiserver**) is a critical component of the Kubernetes control plane. It serves as the front-end for the Kubernetes control plane and handles all the RESTful API requests.

**Exact Work of the API Server in GKE:**

1. **Central Control Point**:
   * The **API Server** is the primary entry point for all interactions with the Kubernetes cluster. It exposes the Kubernetes API and is responsible for receiving, validating, and processing API requests (such as creating, updating, deleting Kubernetes resources) from both users and internal components.
   * All Kubernetes operations (like managing Pods, Deployments, Services, etc.) are submitted via the API server, which is responsible for forwarding these requests to the appropriate backend services.
2. **Request Validation**:
   * When you send a request to the Kubernetes cluster (for example, kubectl apply -f pod.yaml), the **API Server** validates the request to ensure that the request format is correct and complies with the required structure.
   * The API server also checks whether the user or service has sufficient permissions (using **RBAC** or **ABAC** policies) to perform the requested action.
3. **Resource Management**:
   * The API server interacts with the **etcd** storage backend, which stores all the configuration and state of the Kubernetes resources.
   * It reads and writes cluster state data from **etcd** and ensures the consistency of the cluster state across all nodes. When you make changes (e.g., creating a new pod), the API server records this in **etcd**, and the appropriate controllers (like the Deployment controller) act on it to achieve the desired state.
4. **Authentication and Authorization**:
   * The **API Server** authenticates incoming requests, typically using tokens or certificates, to verify the identity of users or services interacting with the cluster.
   * It also handles **authorization**, determining whether the authenticated user has the necessary permissions (via **RBAC** or **ABAC** policies) to perform the requested operation.
5. **Admission Control**:
   * The **API Server** can be configured with various **Admission Controllers**, which are plugins that intercept incoming requests before they are persisted to etcd.
   * Admission controllers can be used to enforce policies such as limiting resource usage, applying security settings, and ensuring compliance with best practices.
6. **API Aggregation Layer**:
   * The **API Server** supports an **API aggregation layer**, which allows it to integrate with third-party APIs. This is important for extending the Kubernetes API with custom resources or external services.
   * The aggregation layer can combine multiple APIs into a unified API, providing seamless access to both native Kubernetes resources and additional services or tools.
7. **Cluster Communication**:
   * The **API Server** acts as the **central communication hub** in the Kubernetes cluster. It is responsible for routing requests to the appropriate backend components. For example:
     + When a new pod is created, the API server triggers the **Scheduler** to determine where the pod should run, and the **Controller Manager** to ensure that the pod is running correctly.
     + The **API Server** also communicates with the **Kubelet** on worker nodes to ensure the desired state of the cluster is maintained.
8. **Expose API Endpoints**:
   * It exposes several endpoints for interacting with cluster resources:
     + /api/v1 for core Kubernetes resources like Pods, Services, etc.
     + /apis/apps/v1 for API groups like Deployments, StatefulSets, etc.
     + /apis/extensions/v1beta1 for other resources like Ingress and ReplicaSets.
   * These endpoints are consumed by tools like kubectl, the Kubernetes Dashboard, or other third-party applications that interact with Kubernetes clusters.

**How the API Server fits in the Kubernetes Architecture:**

* **Control Plane**: The **API Server** is a central part of the **Kubernetes control plane**, along with the **Controller Manager**, **Scheduler**, and **etcd**.
* **Communication Hub**: All Kubernetes components interact with the API Server. For example, when a **Kubelet** on a worker node needs to check the desired state of the cluster, it queries the API Server for updates.

**In Summary:**

The **API Server** in a GKE cluster is the interface between users, the Kubernetes control plane, and the underlying components of the Kubernetes cluster. It is responsible for:

* Accepting, validating, and processing requests to interact with Kubernetes resources.
* Providing authentication, authorization, and admission control.
* Coordinating communication between the various components of the Kubernetes cluster (like the Controller Manager, Scheduler, and Kubelet).
* Storing cluster state in **etcd** and ensuring the desired state of the cluster is met.

Without the **API Server**, Kubernetes wouldn't be able to operate as a cohesive system since it serves as the main entry point for cluster management and operations.