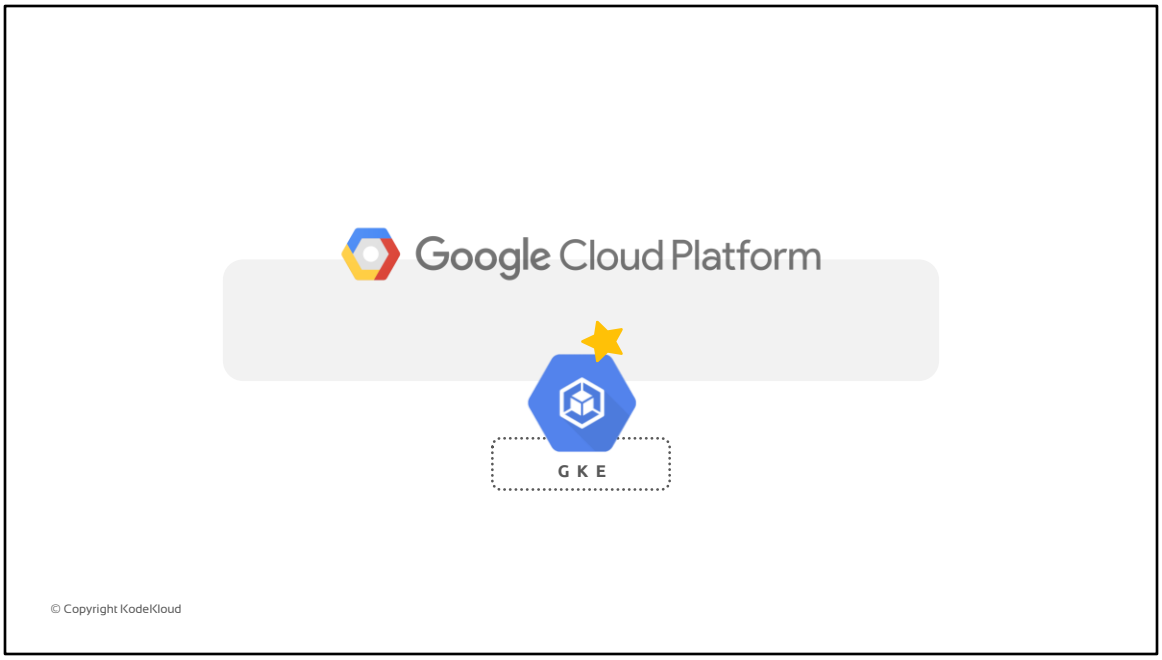
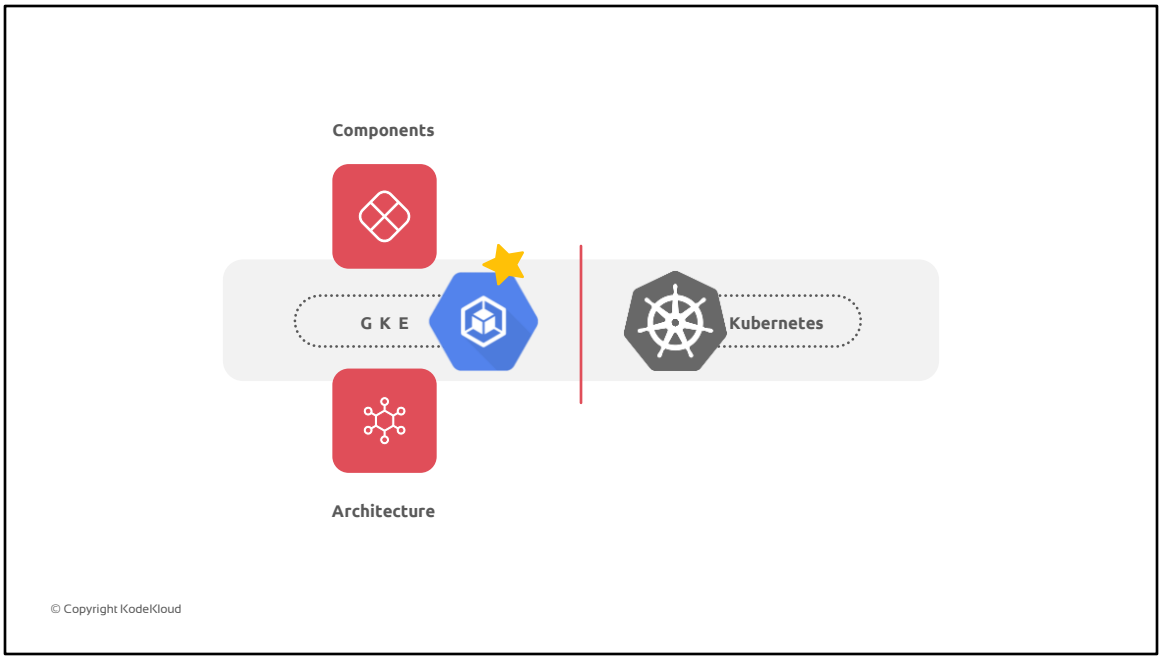
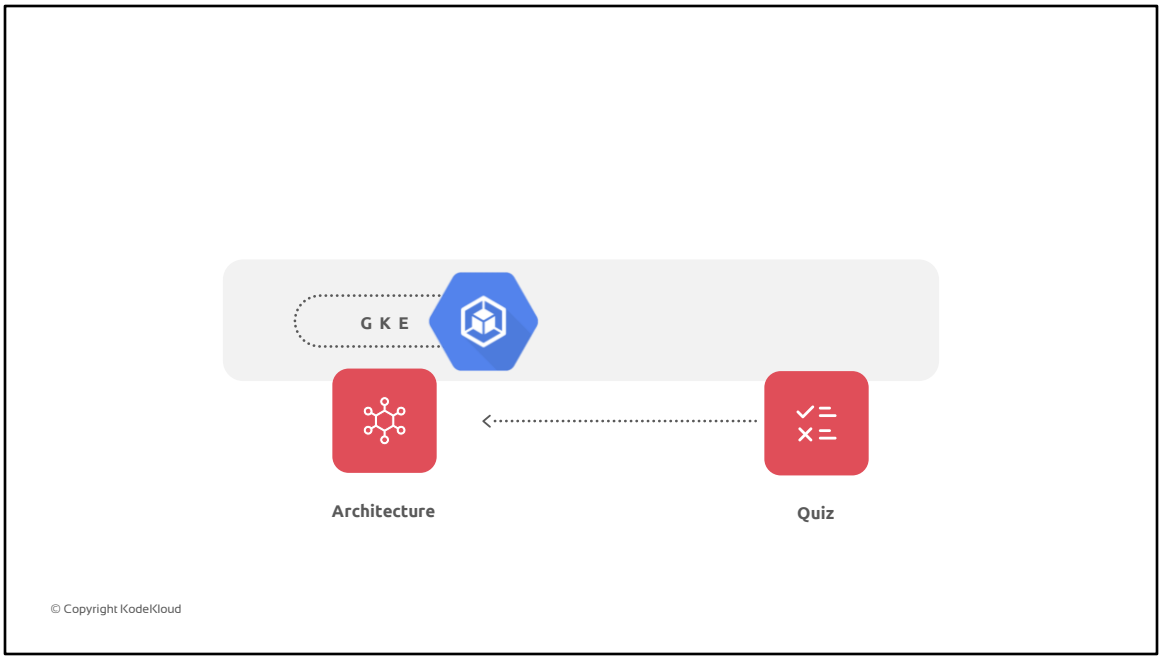
Section Introduction – High-Level Overview



Hello fellow Google cloud enthusiasts! I hope you are all as excited as I am to begin our journey on learning about GKE and diving deep into the specifics of efficient deployment and management of GKE clusters.

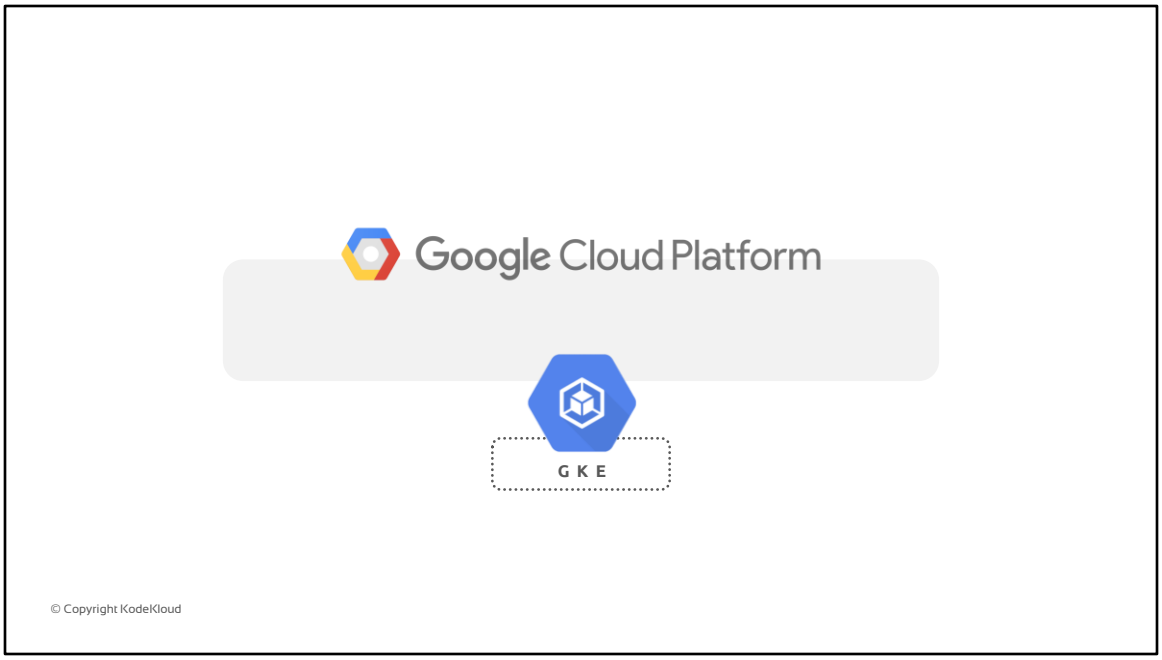


In this section, we'll discuss how Google Kubernetes engine is different from open-source Kubernetes platform and how you can benefit from using GKE in your environment. Then we'll take a look at the high-level architecture of Google Kubernetes engine and different components that are involved in forming a GKE cluster.

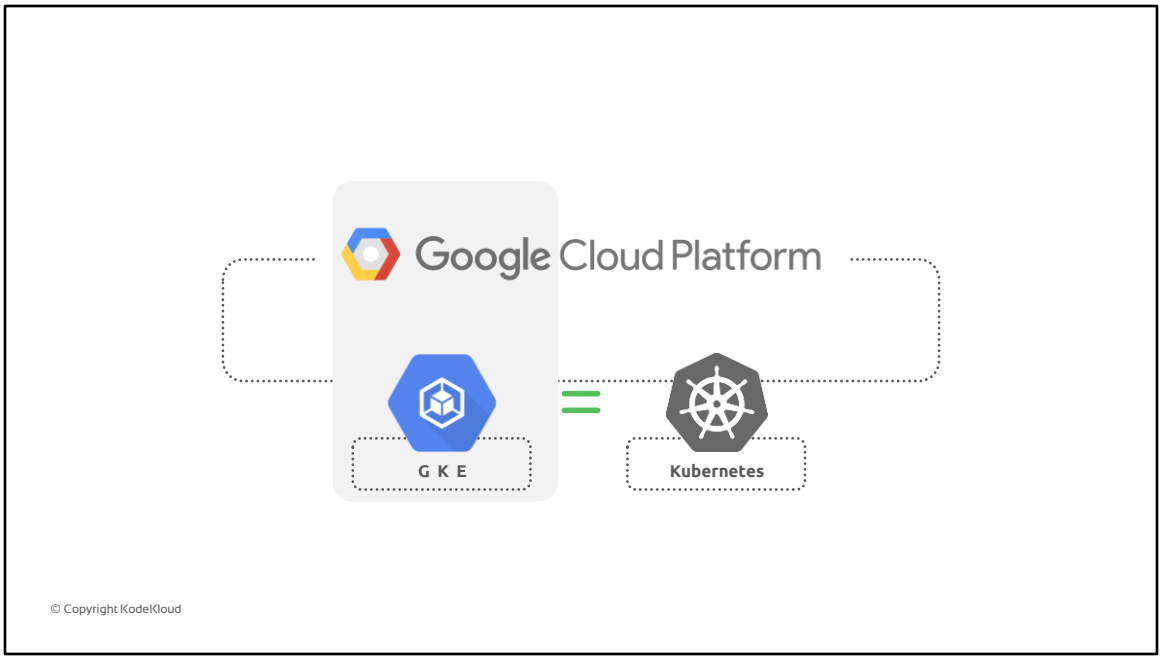


At the end of this section, we'll also have a quiz to test our knowledge on the concepts of GKE architecture.

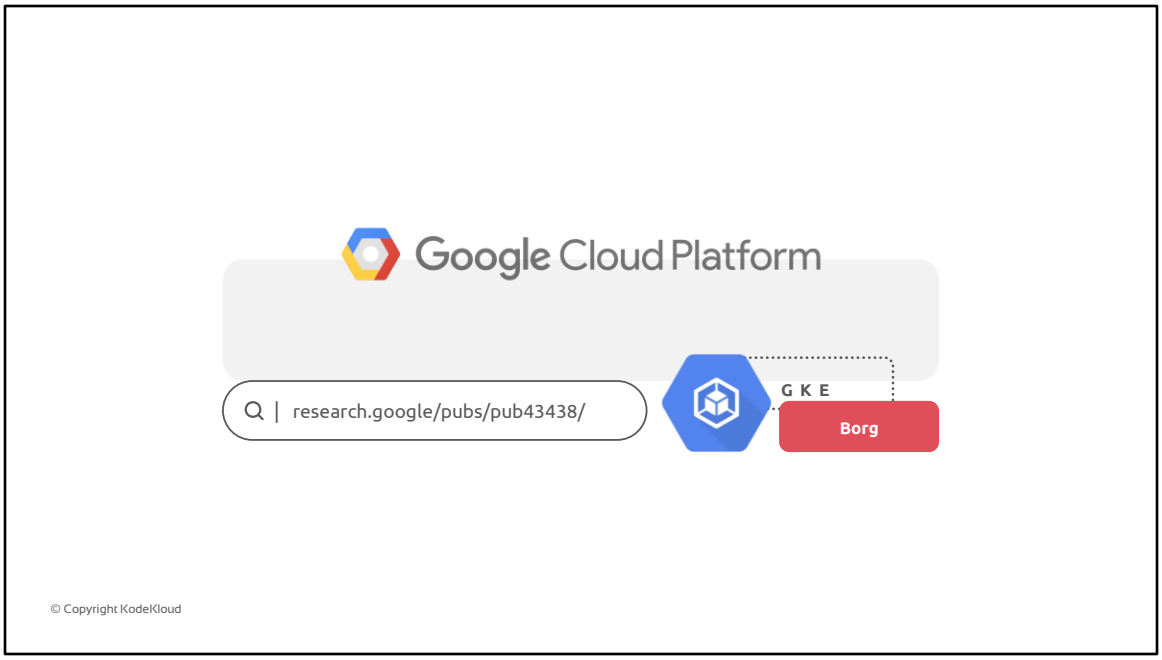
Google Kubernetes Engine: A Google Managed Kubernetes Service



Google Kubernetes Engine, also called GKE, is a Google-managed implementation ....

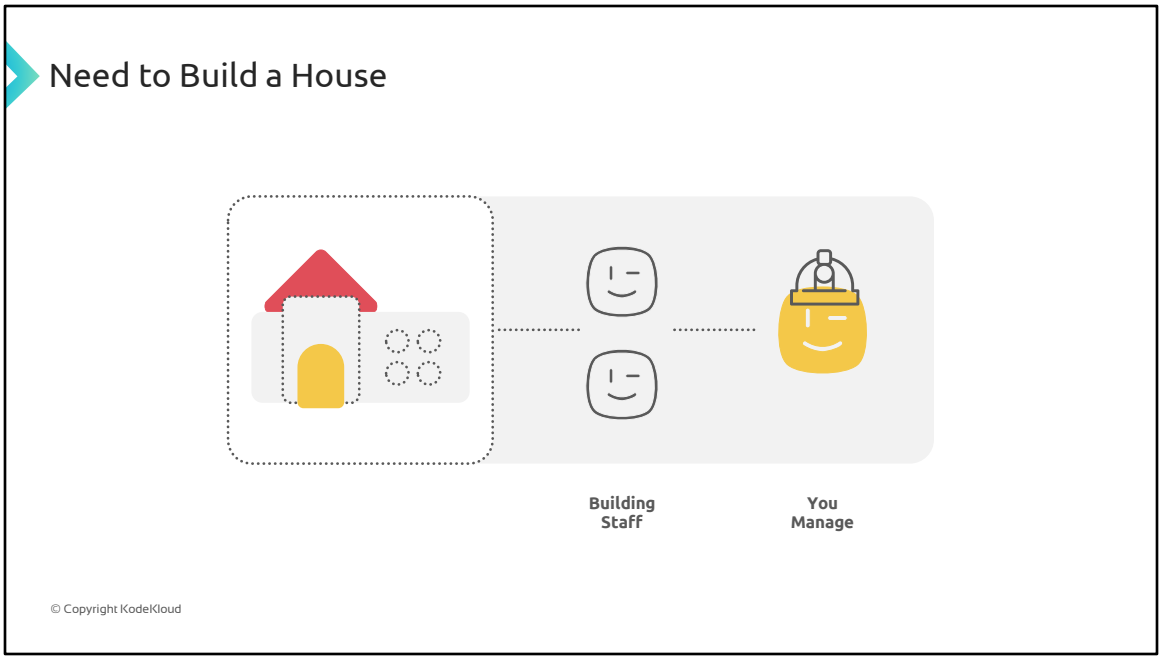


.....of the open-source orchestration platform Kubernetes. Just like Kubernetes, GKE can also be used to deploy and operate containerized applications at scale but using Google's infrastructure.



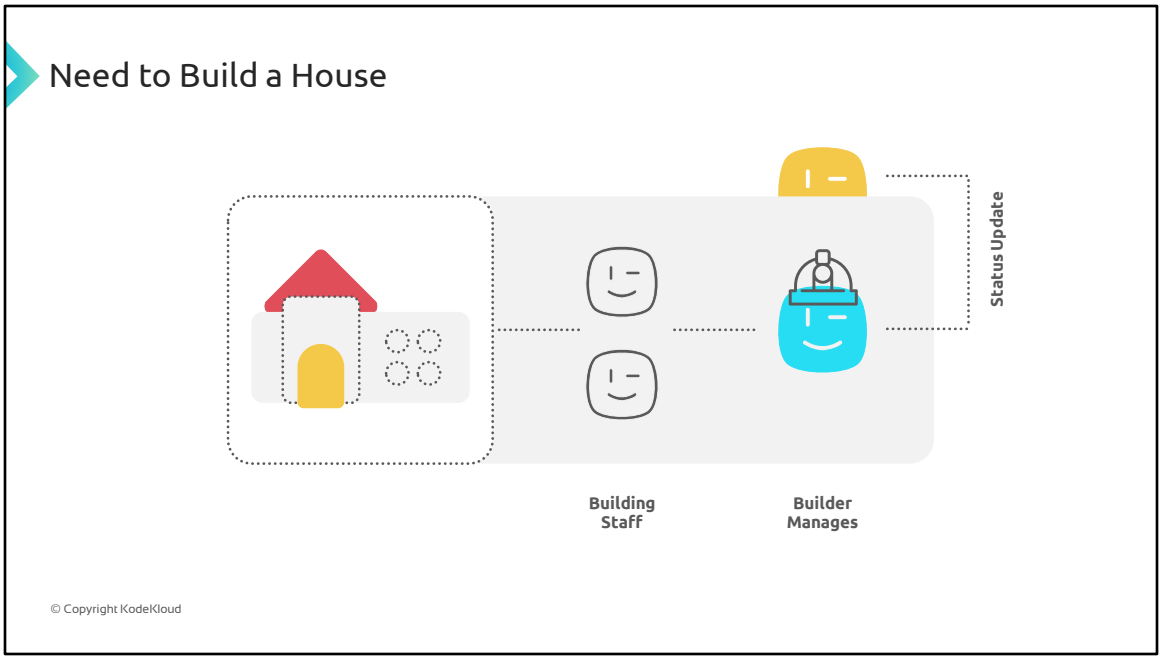
In fact, Kubernetes was also developed by Google based on their in-house cluster management system Borg. If you want to learn more about Borg, you can check this link to the publication by Google.

More on Borg: https://research.google/pubs/pub43438/



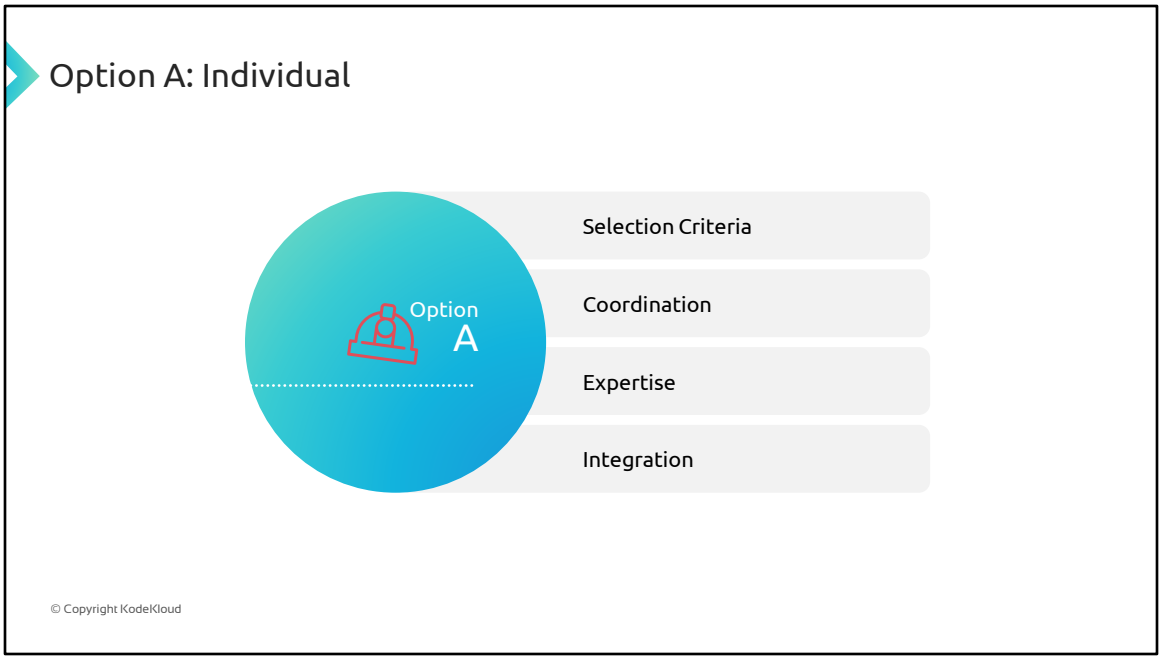
To help you understand where GKE fits and how it's different from the self-managed Kubernetes platform, let's imagine you need to build a house. You'll have two options:

- Selecting individual tradies and managing them yourself or
- Hiring a builder who manages the selection and management of required tradies for you.



To help you understand where GKE fits and how it's different from the self-managed Kubernetes platform, let's imagine you need to build a house. You'll have two options:

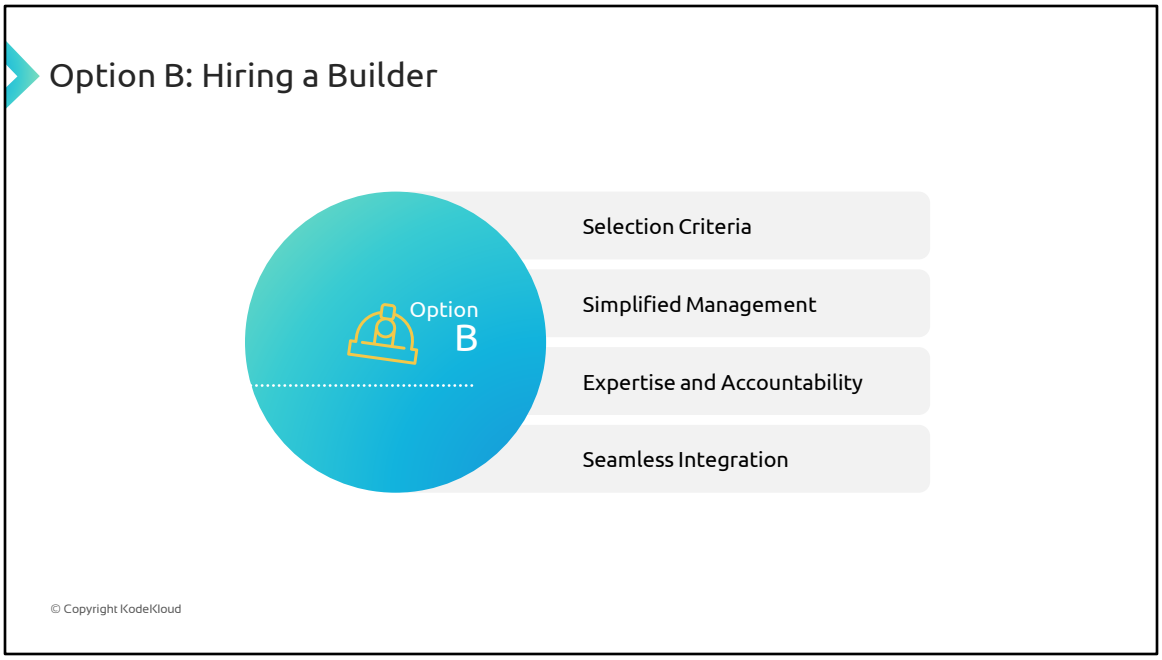
- Selecting individual tradies and managing them yourself or
- Hiring a builder who manages the selection and management of required tradies for you.



With option A, you take on the responsibility of managing and coordinating different trades such as carpenters, electricians, plumbers, and more. Each tradie focuses on their specific task, and you must ensure their work aligns and integrates smoothly to complete the project successfully which is building the house.

There are some challenges that you'll need to address while working with Individual Tradies such as:

- Selection criteria of tradies, the material for your house, and standards for different jobs. Ensuring that you are selecting the resources that are best fit for that job can be quite overwhelming.
- Coordination of the schedules, activities, and communication among different tradies can be very complex and time-consuming.
- Ensuring that each tradie has the necessary expertise and skillset in their field to deliver quality work.
- Integration of the different components of house delivered by each tradie to ensure that the work aligns seamlessly to create a cohesive and functional final product.



In contrast, option B, that is, working with a builder simplifies the construction process for you. The builder takes on the responsibility of managing and coordinating the various trades. They oversee the project, ensure timely execution, and deliver a finished product.

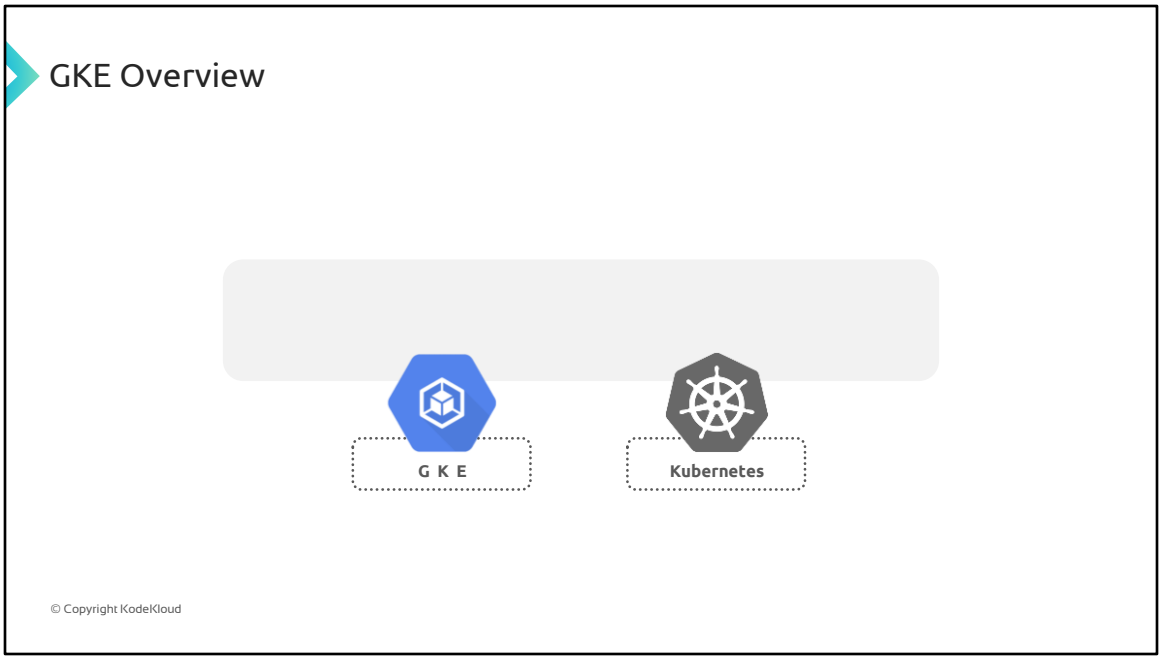
Even in this scenario, you'll need to do some research in selecting the builder, but once you choose a reliable and trustworthy builder, they'll handle rest of the underlying jobs for you.

Selection criteria: With their experience in the field, they can help you to select the material and standards from their recommendations or they can choose the material for you.

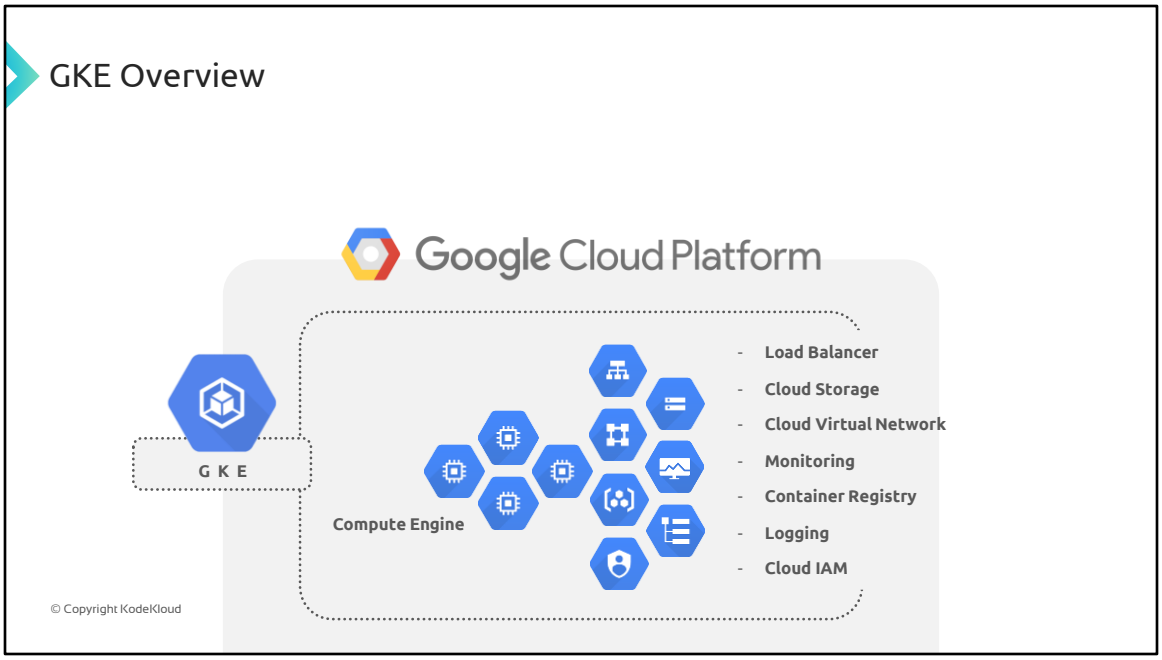
Simplified management: Having a single point of contact (the builder in this case) streamlines project management and communication.

Expertise and accountability: The builder ensures that each tradie is skilled, accountable, and delivers high-quality work.

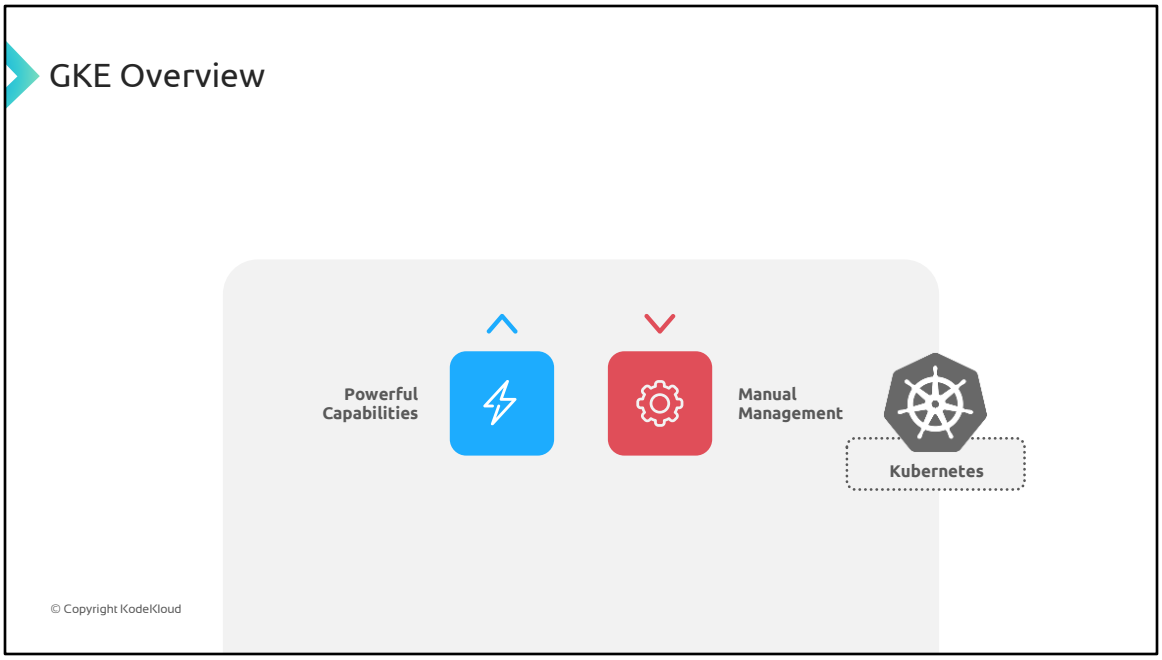
Seamless integration: The builder orchestrates the different tradies, ensuring their work integrates smoothly to achieve a cohesive and functional end result.



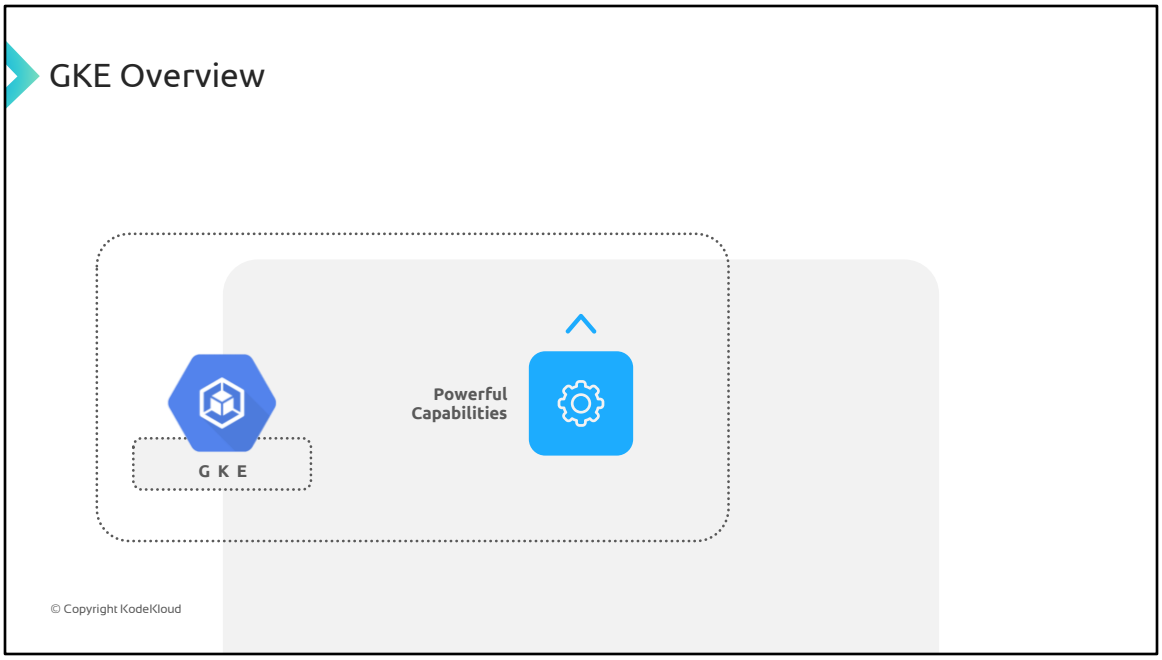
Similarly, when it comes to container orchestration, you can either work directly with Kubernetes or leverage Google Kubernetes Engine (GKE) as a managed service. Let's explore the analogy discussed to understand GKE's role and how it differs from Kubernetes.



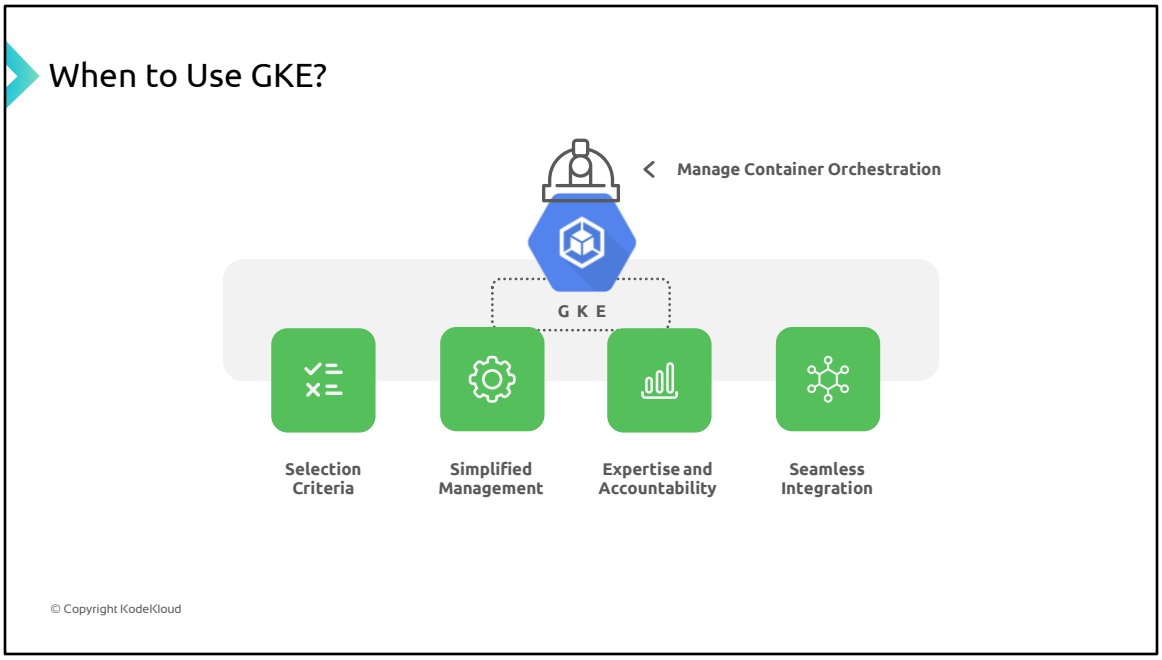
In the context of container orchestration, Google Kubernetes Engine (GKE) serves as the builder. It is a managed service provided by Google Cloud Platform (GCP) that handles the complexities of managing Kubernetes infrastructure.



While Kubernetes provides powerful capabilities for container orchestration, it requires a lot of manual management and coordination from the user's end. Users must set up and manage the control plane, handle scaling of clusters, and ensure high availability themselves.



Just as working with a builder simplifies the construction process, Google Kubernetes Engine (GKE) simplifies all the management and coordination required for Kubernetes infrastructure. GKE serves as a builder for Kubernetes, offering simplified management, expertise, and seamless integration. By leveraging GKE, users can focus on deploying and managing their containerized applications efficiently, while GKE takes care of the underlying infrastructure complexities.



Now, let's switch gears and see when it makes most sense to use GKE and how working with GKE to manage our container orchestration solves those challenges that we discussed:

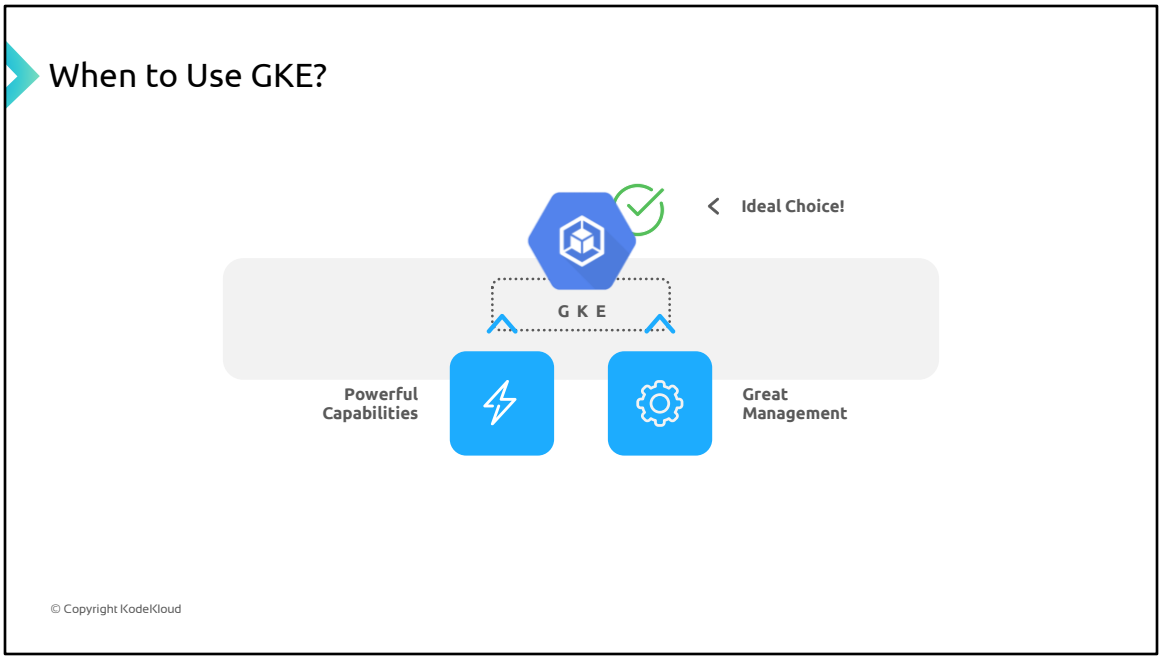
1. Selection criteria: With Google's experience in the field using cluster management systems in-house, Borg, for years, GKE provides services and standards that are best suited for the applications. For example, improved security posture from hardened node operating system.

2. Simplified Management: GKE manages the underlying Kubernetes infrastructure, including the

control plane and cluster orchestration. Users can focus on deploying and managing their applications without worrying about infrastructure management tasks.

3. Expertise and Reliability: GKE benefits from Google's expertise and infrastructure, ensuring high availability, scalability, and reliability of the Kubernetes clusters.

4. Seamless Integration: GKE handles the integration of Kubernetes components such as pods, services, and deployments, allowing for efficient container orchestration. It also offers the capability to integrate our Kubernetes cluster with other GCP services such as artifact registry, cloud monitoring and logging, load balancing, etc.



Google Kubernetes Engine (GKE) is an ideal choice when you want to leverage the power of Kubernetes for container orchestration without the burden of managing the underlying infrastructure.

GKE provides some key benefits and capabilities out of the box as compared to open-source Kubernetes platform. For example:

1. Managed Kubernetes clusters: GKE provides fully managed Kubernetes clusters, which means that Google takes care of the underlying infrastructure, including the nodes, storage, and networking. This frees up organizations to focus on developing and deploying their applications.
2. Autoscaling: GKE supports autoscaling of Kubernetes clusters, which means that the clusters can automatically scale up or down based on the

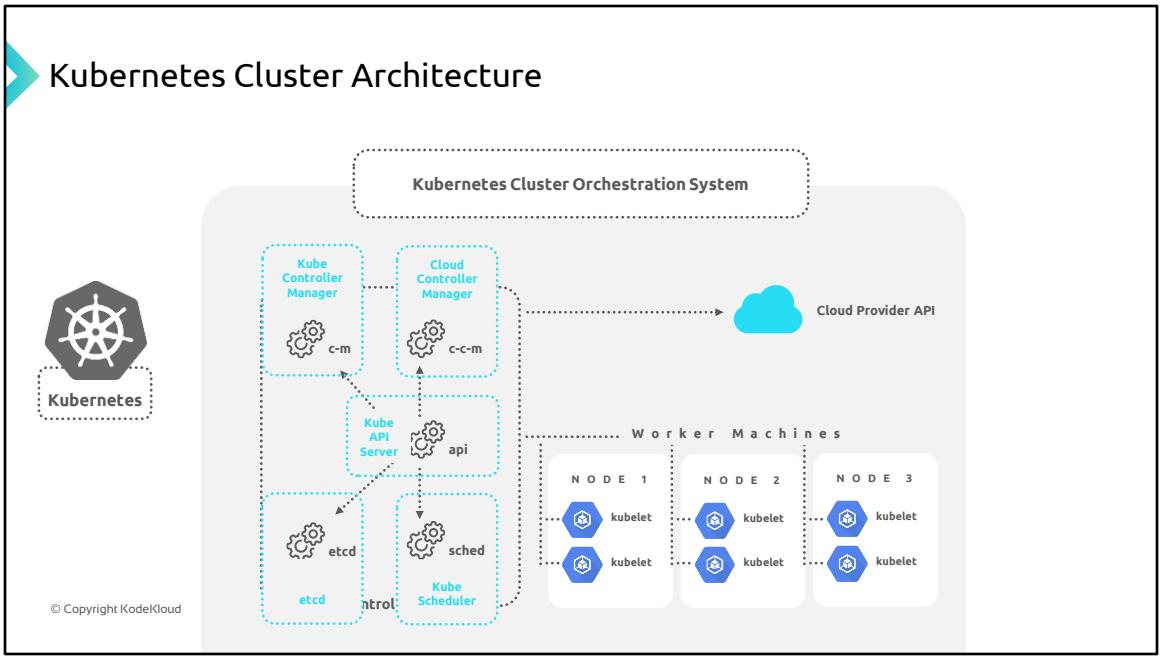
demand for applications. This also helps in cost optimization.

3. Load balancing: GKE also supports load balancing of Kubernetes pods, which means that the pods can be automatically distributed across multiple nodes. This helps to ensure that applications are always available and that they can handle spikes in traffic.

4. Logging and monitoring: GKE provides built-in loggig and monitoring capabilities, which can help troubleshoot problems and to understand how the applications are performing.

5. Integration with other Google Cloud Platform services: GKE can be integrated with other Google Cloud Platform services, such as Cloud Storage, Cloud SQL, and Cloud Monitoring.

Google Kubernetes Engine: Architecture



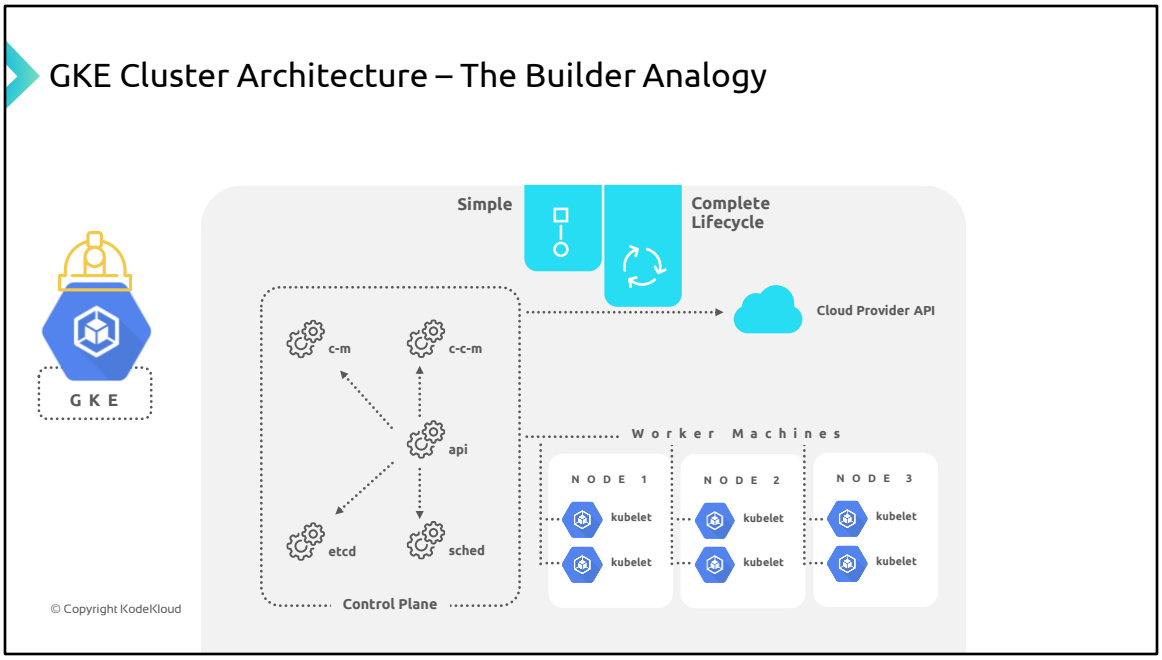
Next, let's look at the concepts specific to Google Kubernetes Engine. This diagram is a representation of a Kubernetes cluster. A Kubernetes cluster consists of a control plane and one or more nodes that are known as worker machines to run containerized applications. The control plane and nodes make up the Kubernetes cluster orchestration system.

The control plane manages the worker nodes and the pods in the cluster. There are a lot of components that constitute a control plane:

- Cloud Controller Manager: It links the cluster into a cloud provider's API and separates out the components that interact with that cloud platform from components that only interact with the cluster.
- Next is Kube Controller Manager: It's a Control plane component that runs controller processes such as:
  - Node controller
  - Job controller
  - EndpointSlice controller
  - ServiceAccount controller
- Kube API Server is a component that exposes the Kubernetes API and is the frontend for the Kubernetes control plane

- Etcd is a consistent and highly available key-value storage for storing all cluster data
- The last component of a control plane is Kube Scheduler. It watches for newly created pods with no assigned node and selects a node for them to run on.

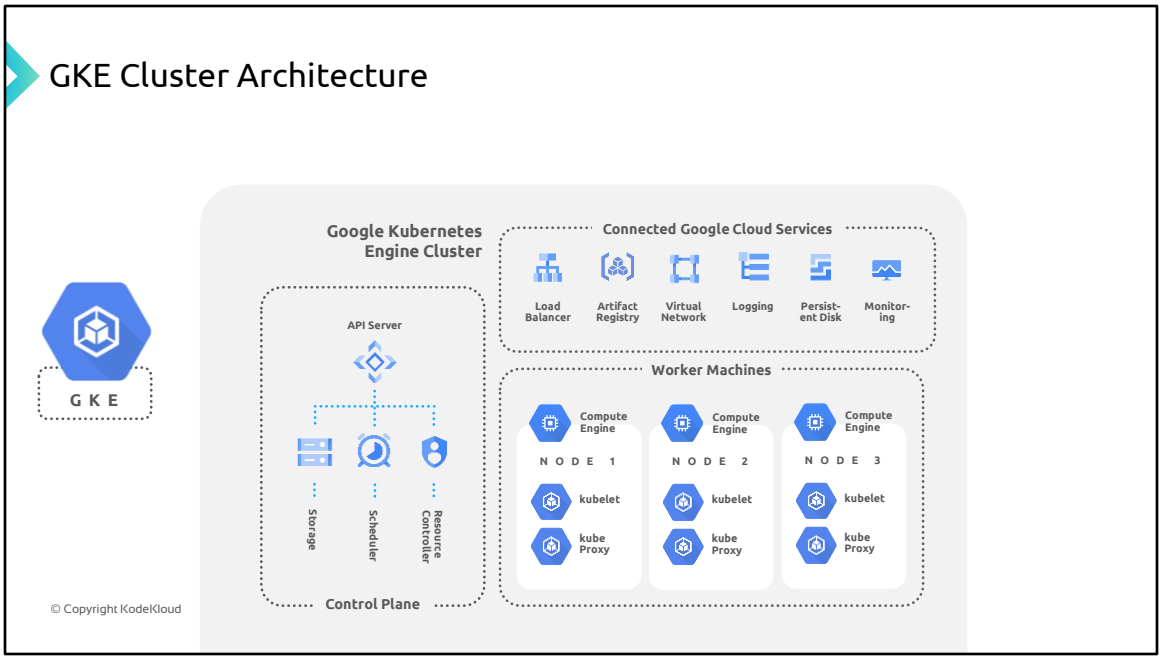
During setup, a lot of work is required to get a complete and working Kubernetes cluster. In any Kubernetes environment, nodes are needed to be created externally by cluster administrators and are not handled by Kubernetes itself



Now, if we refer back to the analogy of hiring a builder to build a house for us by selecting and managing different tradies, GKE is the builder in this case, and it'll create and manage the required resources for us.

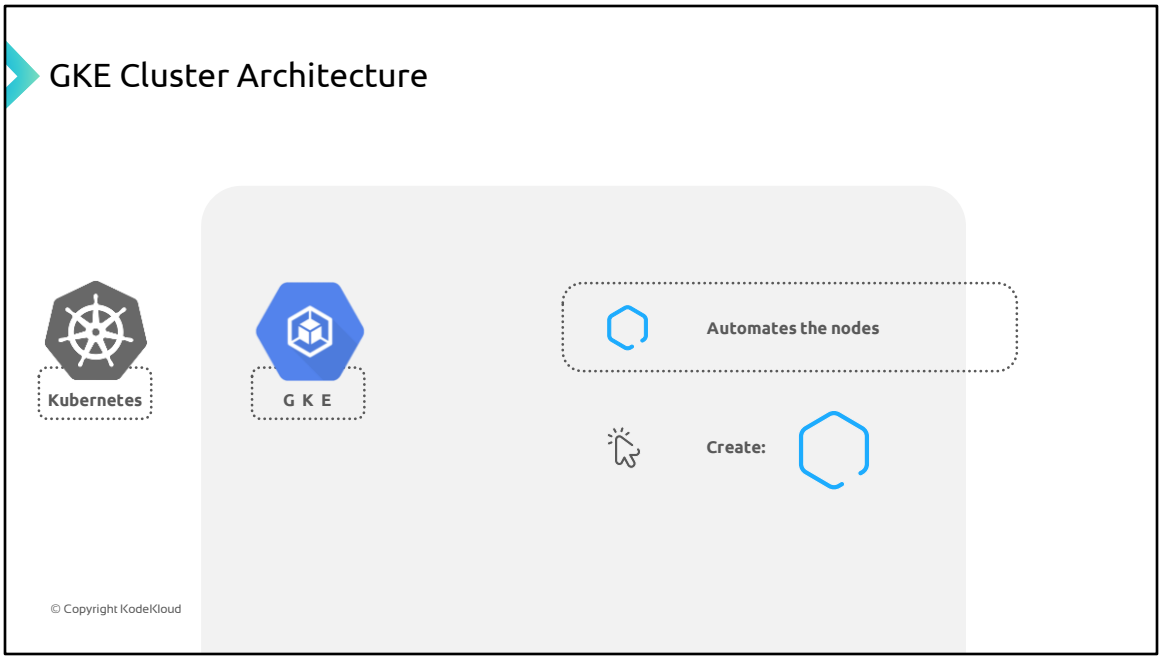
Managing the control plane in a GKE cluster is a lot **simpler** from a user's perspective. GKE manages the **complete lifecycle** of the control plane and all its system components, from cluster creation to deletion, including upgrades to the Kubernetes version running on the **control plane**. Although an

IP address is still exposed to which all the Kubernetes API requests are sent, GKE takes the responsibility for provisioning and managing all of the control plane infrastructure behind it. It also abstracts away from having a separate control plane.

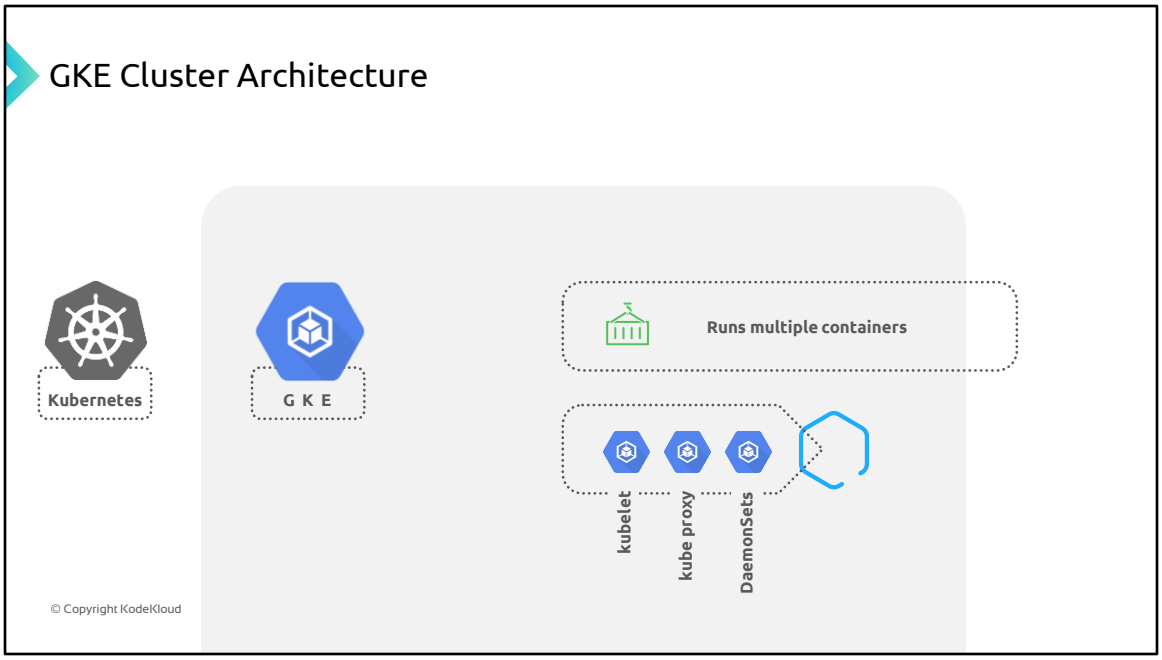


In the Google Kubernetes Engine or GKE world, this is how a Google managed Kubernetes cluster would look like. The services on the left hand side are part of a control plane. These services include:

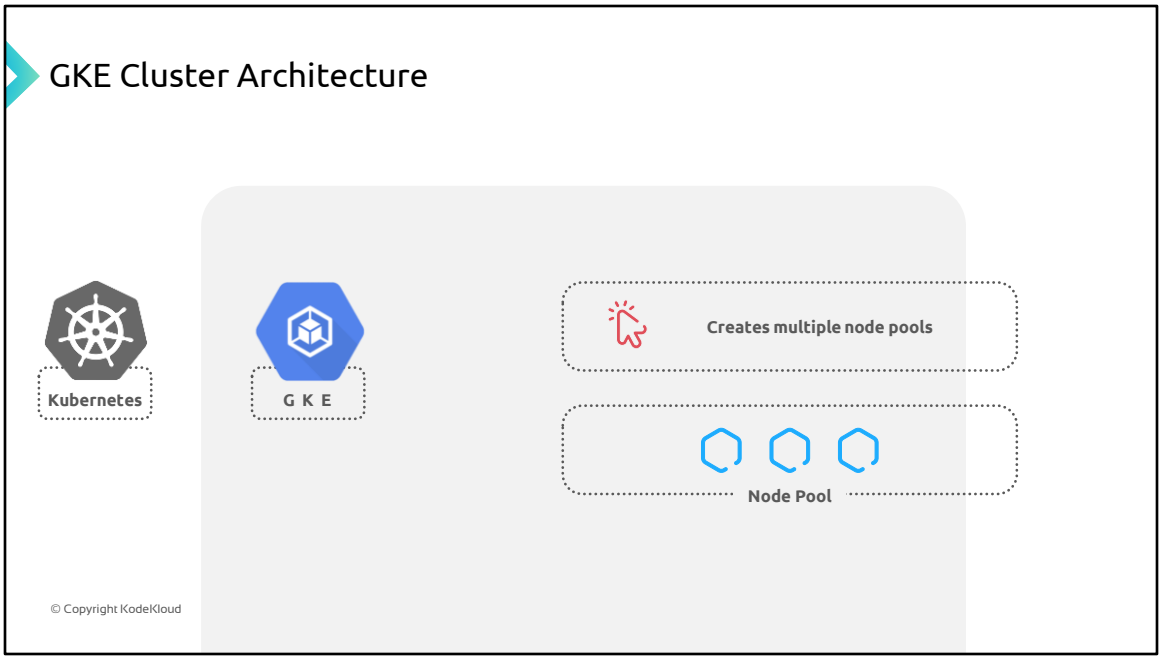
- Resource Controllers
- API Server
- Scheduler
- Storage



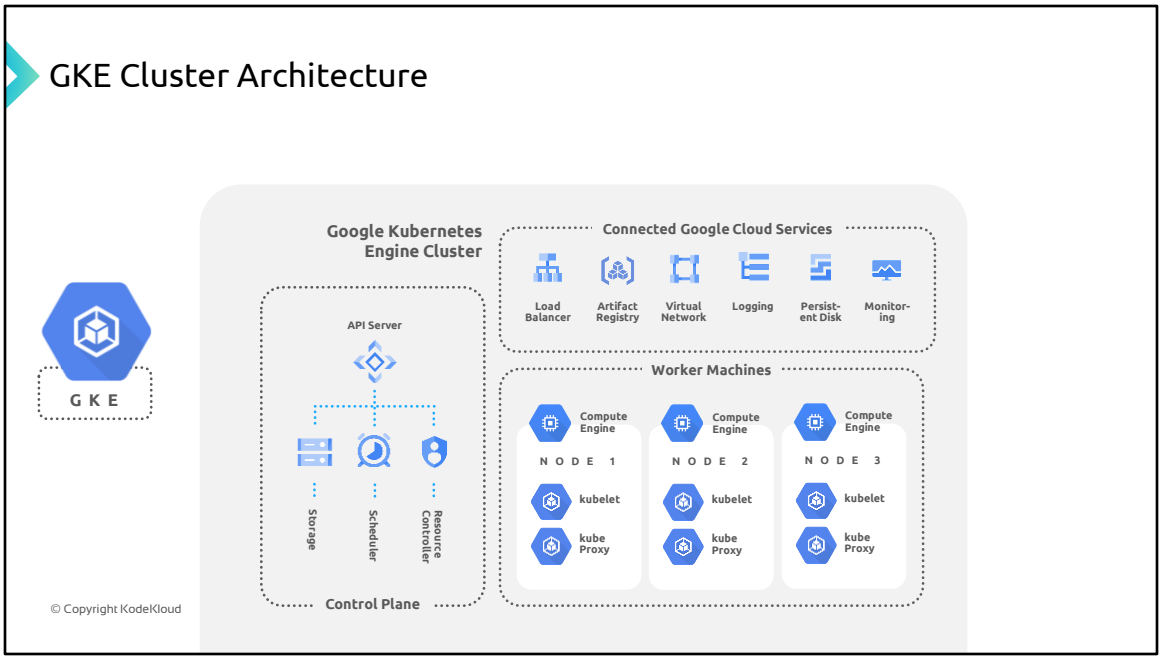
Unlike Kubernetes, **GKE also automates the node** or worker machine creation process by launching Compute Engine virtual machine instances and registers them as nodes.



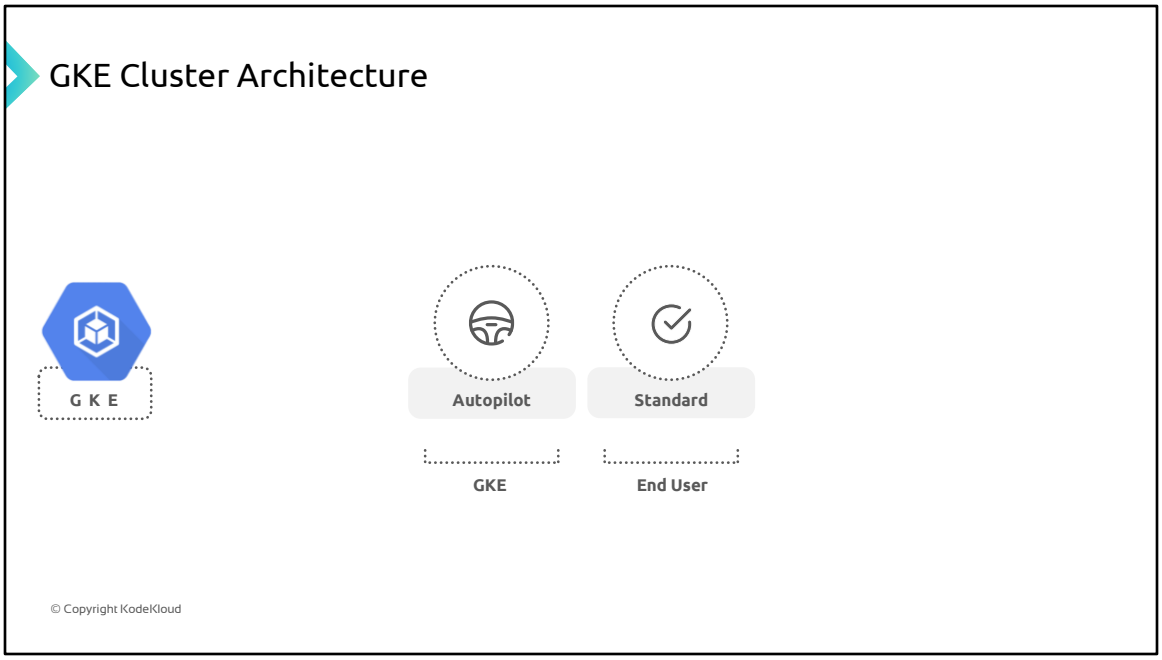
Other than Kubernetes node agent Kubelet and Kube proxy, **GKE also runs a number of system containers** as per-node agents called DaemonSets. These DaemonSets provide functionality such as log collection and intra-cluster network connectivity.



GKE also offers a distinct capability to **select multiple node machine** types by creating multiple node pools. A node pool is like a subset of nodes within a cluster that share a common configuration, such as memory or CPU generation. Node pools also provide an easy way to ensure that workloads run on the right hardware within a GKE cluster.



There are a few **Google cloud services, some as shown on the top right corner, that connect with a GKE cluster** for its deployment and management. For example, when a GKE cluster is created or updated, GKE pulls container images for the Kubernetes system software running on the control plane and nodes from the Google Cloud Artifact Registry.



Just an important note here is that, depending on the mode of operation of the cluster, how worker machines or nodes, are managed also varies. There are two modes of operation of GKE: Autopilot and Standard.

In an Autopilot mode, along with the control plane and all system components, nodes are also managed by the GKE cluster.
However, in a GKE Standard mode, GKE manages the control plane and system components, whereas

the nodes are managed by the end user.

We'll discuss about the modes of operations in more detail in next section.