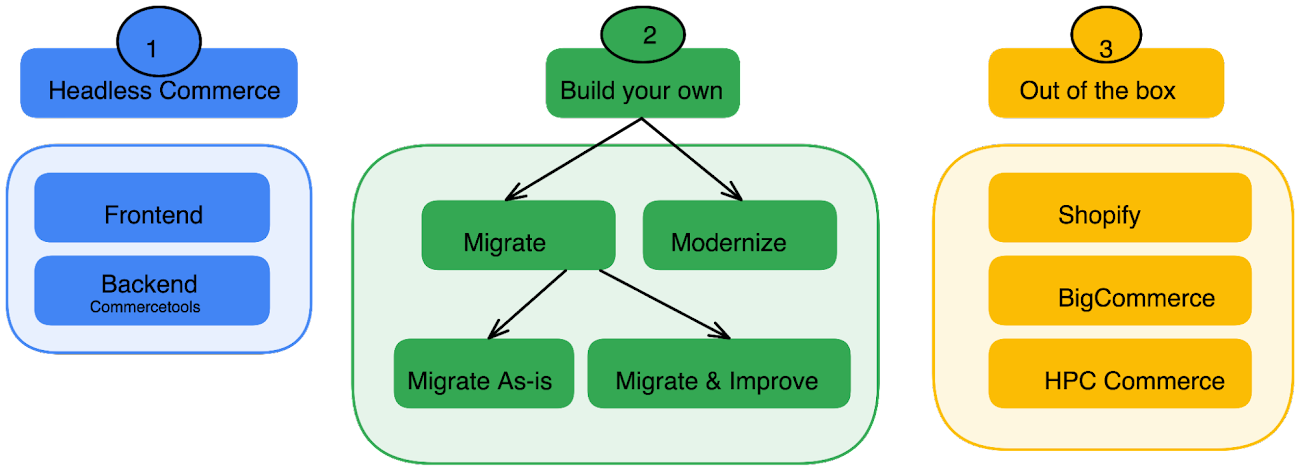
# **Commerce platform on Google Cloud**

Three different ways to modernize your ecommerce platform on Google Cloud. No one way is right or wrong. You can start off on any path that works well for your team and current infrastructure,  and then work your way towards a fully modernized platform.

The three main approaches to modernizing your digital commerce platform are headless commerce, build your own, and out-of-the-box.



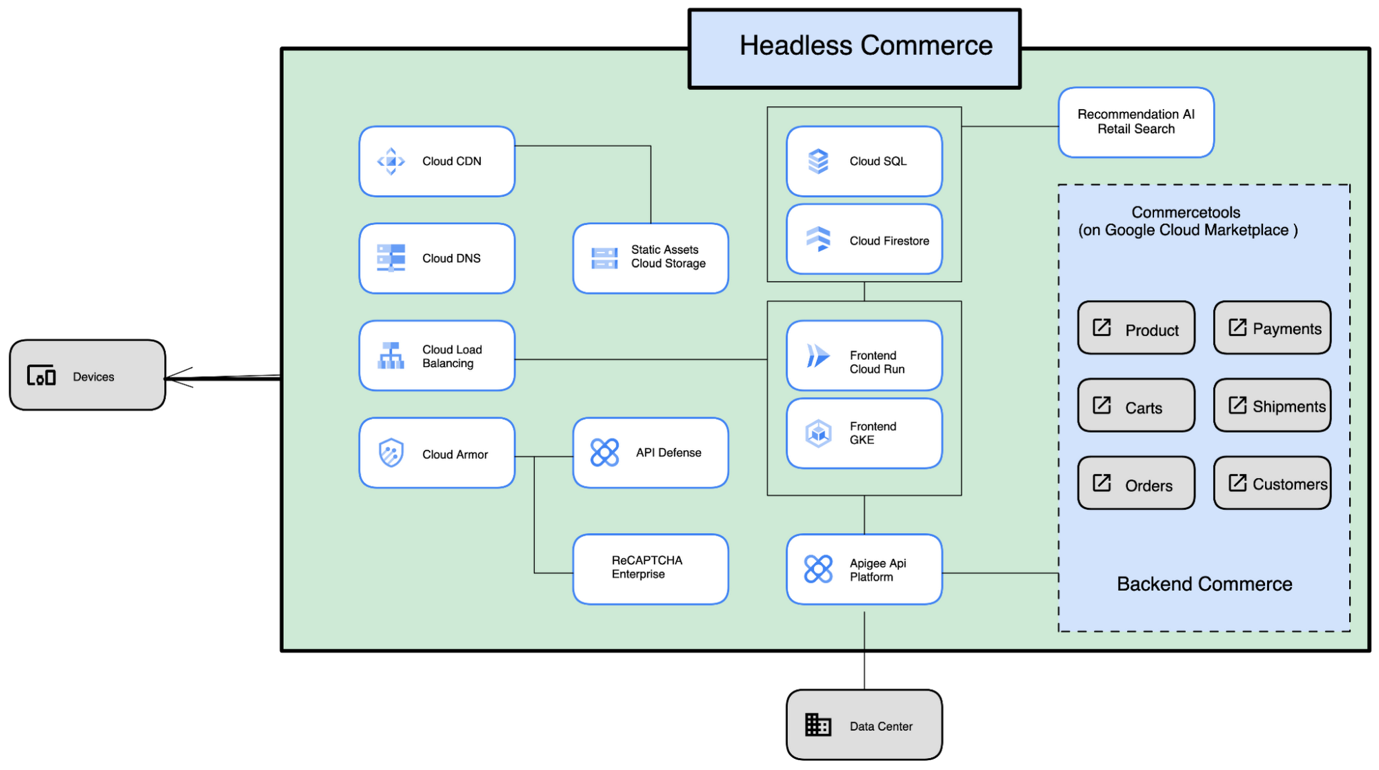
Let’s take a closer look at each of these approaches.

1. Headless commerce

As a retailer you might want to own and control the frontend for providing an engaging and differentiated shopping experience, and use ready-to-use backend commerce building blocks for capabilities such as product catalog and cart functionality, pricing, promotions, shipping, and account management.

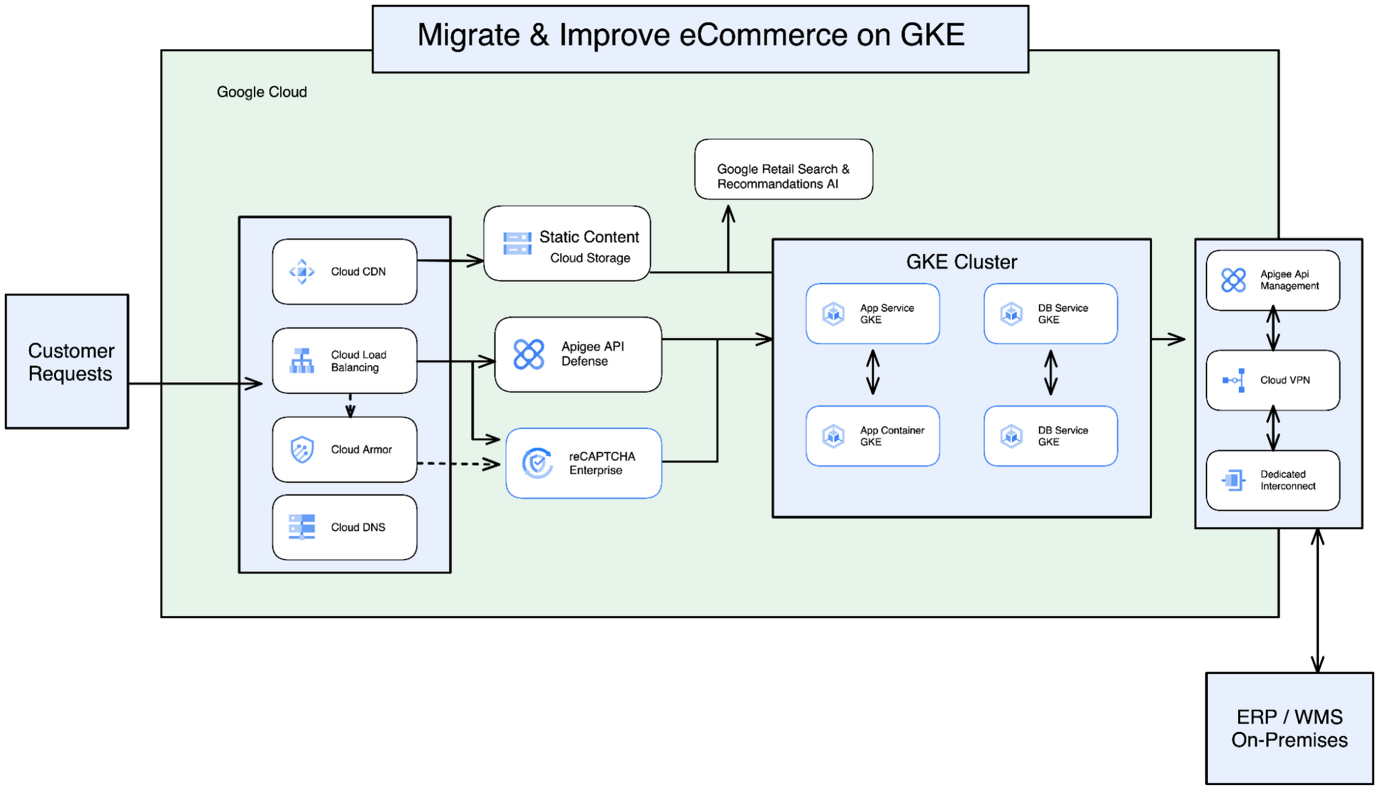
When you decouple the backend from the frontend in a headless commerce approach it becomes easier for your teams to:

* Update the client-facing site (for example, to handle seasonal campaigns) without needing backend skills and without risking regressions in the backend processing.
* Harness shopper interaction data (i.e. first-party data) and then use that data to gain insights into shopping behavior and activate data-driven, personalized offerings to increase sales conversions.
* Make nimble updates to the frontend while maintaining compliance with security and other standards.
* Maintain consistent performance for the end-to-end client experience as you apply updates to the frontend.

Here is a architecture of a headless commerce implementation on Google Cloud using [Commercetools](https://commercetools.com/?location=usa" \t "_blank), a Google ISV partner, as a backend and [Cloud Run](https://cloud.google.com/run) or [Google Kubernetes Engine](https://cloud.google.com/kubernetes-engine) (GKE) as the frontend..[](https://storage.googleapis.com/gweb-cloudblog-publish/images/headlesscommerce.max-2000x2000.png)

2. Build your own commerce platform on Google Cloud

With the build-your-own approach, you assemble your own solution using building blocks from Google Cloud, including [AI / ML tools](https://cloud.google.com/products/ai) and [data management tools](https://cloud.google.com/products/databases). There are three main ways to implement this approach:

**a. Migrate your existing solution as-is**by hosting your existing solution in Google Cloud instead of running it in your own data center. This minimizes the changes required to get to the cloud, but you don’t get the benefits of a microservices architecture.  
**b. Migrate (and improve) your existing solution to a microservices architecture** that uses containers and managed services in Google Cloud. This approach requires more changes than migrating your existing solution as-is. But once you’ve migrated, you’ll realize the benefits of a cloud-native, containerized architecture, including easier maintenance, improved flexibility, and scalability. Here is a sample architecture for migrating and improving your commerce platform using GKE.[](https://storage.googleapis.com/gweb-cloudblog-publish/images/migrate_and_improve.max-2000x2000.png)

**c. Modernize the architecture completely.**Let’s say you have migrated to Google Cloud using GKE or Compute Engine. Or maybe you are building an entire modernized commerce platform from scratch. How would you then progress to a modernized fully containerized microservices architecture

Think about implementing the platform with smaller services separated into four layers:

* **Presentation layer** - Your single-page application (SPA) builder and content.
* **Services Layer** - Services such as session, search, account, inventory, orders, and so on.
* **Storage layer** - Your storage choices for the services. These can vary based on the type of service, for example:[CloudSQL](https://cloud.google.com/sql-server) or [Cloud Spanner](https://cloud.google.com/spanner) for inventory and products, and [Firestore](https://cloud.google.com/firestore) for user sessions.
* **Cache layer** - Your transient cache, which can use [Memorystore](https://cloud.google.com/memorystore/) to access recently queried data without querying the storage layer again.

In addition to being  the most flexible option, this architecture has many benefits, which include enabling your organization to:

* Independently develop and scale microservices to accommodate traffic spikes and lower TCO.
* Add/deploy new services at any time without affecting any other services.
* Have different teams working on different services in different languages.
* Establish a closed loop between the data platform and the commerce platform to better understand customer behavior and offer personalized experiences.