**What is Microservices Architecture?**

Microservices architecture (often shortened to microservices) refers to an architectural style for developing applications. Microservices allow a large application to be separated into smaller independent parts, with each part having its own realm of responsibility. To serve a single user request, a microservices-based application can call on many internal microservices to compose its response.

[Containers](https://cloud.google.com/containers) are a well-suited microservices architecture example, since they let you focus on developing the services without worrying about the dependencies. Modern cloud-native applications are usually built as microservices using containers.

A microservices architecture is a type of application architecture where the application is developed as a collection of services. It provides the framework to develop, deploy, and maintain microservices architecture diagrams and services independently.

Within a microservices architecture, each microservice is a single service built to accommodate an application feature and handle discrete tasks. Each microservice communicates with other services through simple interfaces to solve business problems.

## **What is microservices architecture used for?**

Typically, microservices are used to speed up application development. Microservices architectures built using Java are common, especially Spring Boot ones. It’s also common to compare microservices versus service-oriented architecture. Both have the same objective, which is to break up monolithic applications into smaller components, but they have different approaches. Here are some microservices architecture examples:

### **Website migration**

A complex website that’s hosted on a monolithic platform can be migrated to a cloud-based and container-based microservices platform.

### **Media content**

Using microservices architecture, images and video assets can be stored in a scalable object storage system and served directly to web or mobile.

### **Transactions and invoices**

Payment processing and ordering can be separated as independent units of services so payments continue to be accepted if invoicing is not working.

### **Data processing**

A microservices platform can extend cloud support for existing modular data processing services.



Let’s imagine that you responsible for a business critical business application that has a [monolithic architecture](https://microservices.io/patterns/monolithic.html) and you are struggling to meet the needs of the business. Should you consider migrating to a [microservice architecture](https://microservices.io/patterns/microservices.html)? The short answer is that it depends.

It’s important [to make the most of your monolithic architecture](https://microservices.io/post/refactoring/2020/07/28/six-principles-for-refactoring-to-microservices.html), e.g. adopt DevOps, and reorganize into loosely coupled, small teams.

In many cases, once you have embraced the success triangle, your monolithic architecture is sufficiently loosely coupled, testable and deployable to enable rapid software delivery.

But sometimes an application can outgrow its monolithic architecture and become an obstacle to rapid, frequent and reliable software delivery. This typically happens when the application becomes large and complex and is developed by many teams. For example, its deployment pipeline become a bottleneck. When this occurs, you should consider migrating to microservices.