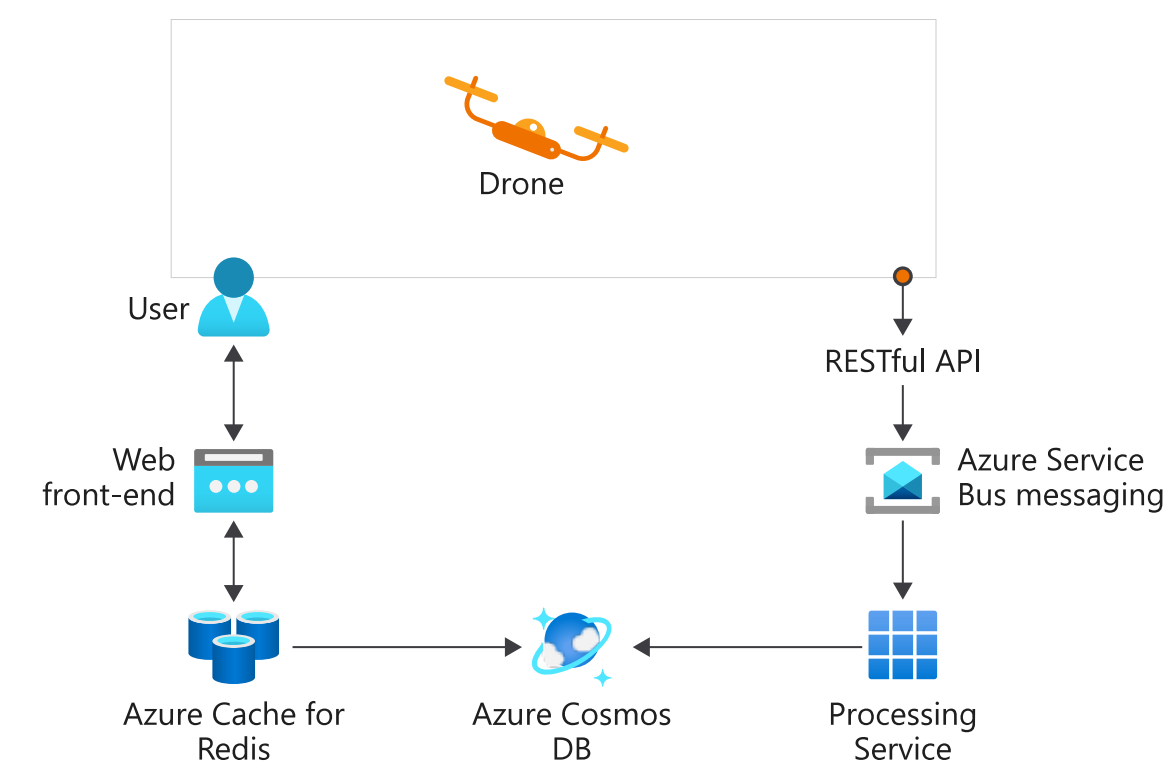


# When to use Kubernetes

4 minutes

The decision to use a container orchestration platform like Kubernetes depends on business and development requirements. Here's a review of the high-level architecture of the drone tracking solution.

The solution is built as microservices that are designed as loosely coupled, collaborative services. To simplify the solution's design and maintenance, you're deploying these services separately from each other. Here's the current configuration of your solution.



- **Web front-end:** Shows maps and information about tracked drones.
- **Cache service:** Stores frequently requested information appearing on the website.
- **RESTful API:** Used by tracked drones to send data about their status, such as a GPS location and battery-charge levels.
- **Queue:** Holds unprocessed data collected by the RESTful API.
- **Data-processing service:** Fetches and processes data from the queue.
- **NoSQL database:** Stores processed tracking data and user information captured from the website and the data-processing service.

Separate teams in your company develop and own these services. Each team uses containers to build and deploy its service. This new strategy allows the development teams to keep up with the requirements of modern software development for automation, testing, and overall stability and quality.

The change in developer thinking has resulted in several process and business benefits for the company. Examples include better use of hosted compute resources, new features that have improved time to market, and improved customer reach.

However, several challenges with container management led your company to investigate container-orchestration solutions. Your teams found that scaling the tracking application to a handful of deployments was relatively easy, but scaling and managing many instances was hard.

There are several other aspects to consider. Examples include dealing with failed containers, storage allocation, network configuration, and managing app secrets.

As you learned earlier, Kubernetes provides support for all of these challenges as an orchestration platform.

You want to use Kubernetes when your company:

- Develops apps as microservices.
- Develops apps as cloud-native applications.
- Deploys microservices by using containers.
- Updates containers at scale.
- Requires centralized container networking and storage management.

# When not to use Kubernetes

Not all applications need to run in Kubernetes. As a result, Kubernetes might not be a good fit for your company.

For example, the effort in containerization and deployment of a monolithic app might be more than the benefits of running the app in Kubernetes. A monolithic architecture can't easily use features such as individual component scaling or updates.

Kubernetes can introduce many business benefits for software development, deployment, management, and streamlining of processes. However, Kubernetes has a steep learning curve. The modular design of Kubernetes introduces potentially new concepts that will affect teams across your company.

Your development teams will have to embrace modern design concepts when developing and designing apps. These concepts include using microservices and containerizing these services. Teams also needs to experiment with container and orchestration environments to make the best use of all the available options.

If your company isn't ready to adopt this change, Kubernetes might not be a good fit for your company.

---

## Next unit: Summary

Continue >

