


Designing a microservices architecture

02/26/2019 • 2 minutes to read • Contributors 

In this article

[Prerequisites](#)

[Reference implementation](#)

[Scenario](#)

[Next steps](#)

Microservices have become a popular architectural style for building cloud applications that are resilient, highly scalable, independently deployable, and able to evolve quickly. To be more than just a buzzword, however, microservices require a different approach to designing and building applications.

In this set of articles, we explore how to build and run a microservices architecture on Azure. Topics include:

- [Interservice communication](#)
- [API design](#)
- [API gateways](#)
- [Data considerations](#)
- [Design patterns](#)

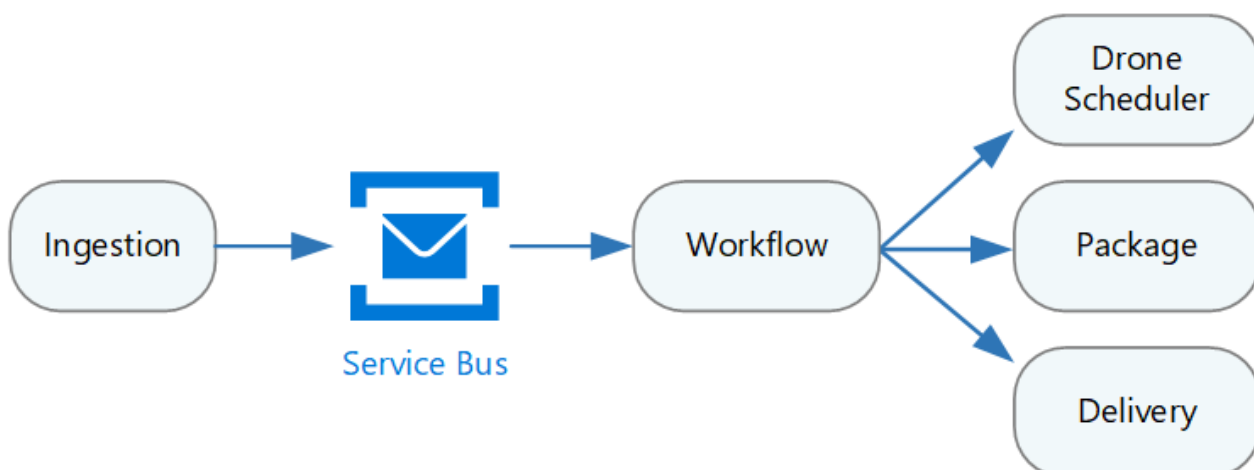
Prerequisites

Before reading these articles, you might start with the following:

- [Introduction to microservices architectures](#). Understand the benefits and challenges of microservices, and when to use this style of architecture.
- [Using domain analysis to model microservices](#). Learn a domain-driven approach to modeling microservices.

Reference implementation

To illustrate best practices for a microservices architecture, we created a reference implementation that we call the Drone Delivery application. This implementation runs on Kubernetes using Azure Kubernetes Service (AKS). You can find the reference implementation on [GitHub](#).



Scenario

Fabrikam, Inc. is starting a drone delivery service. The company manages a fleet of drone aircraft. Businesses register with the service, and users can request a drone to pick up goods for delivery. When a customer schedules a pickup, a backend system assigns a drone and notifies the user with an estimated delivery time. While the delivery is in progress, the customer can track the location of the drone, with a continuously updated ETA.

This scenario involves a fairly complicated domain. Some of the business concerns include scheduling drones, tracking packages, managing user accounts, and storing and analyzing historical data. Moreover, Fabrikam wants to get to market quickly and then iterate quickly, adding new functionality and capabilities. The application needs to operate at cloud scale, with a high service level objective (SLO). Fabrikam also expects that different parts of the system will have very different requirements for data storage and querying. All of these considerations lead Fabrikam to choose a microservices architecture for the Drone Delivery application.

Note

For help in choosing between a microservices architecture and other architectural styles, see the [Azure Application Architecture Guide](#).

Our reference implementation uses Kubernetes with [Azure Kubernetes Service](#) (AKS). However, many of the high-level architectural decisions and challenges will apply to any container orchestrator, including [Azure Service Fabric](#).

Next steps

Choose a compute option