


# Build for the needs of the business

08/30/2018 • 2 minutes to read • Contributors 

## In this article

[Every design decision must be justified by a business requirement](#)

[Recommendations](#)

## Every design decision must be justified by a business requirement

This design principle may seem obvious, but it's crucial to keep in mind when designing a solution. Do you anticipate millions of users, or a few thousand? Is a one-hour application outage acceptable? Do you expect large bursts in traffic or a predictable workload? Ultimately, every design decision must be justified by a business requirement.

## Recommendations

**Define business objectives**, including the recovery time objective (RTO), recovery point objective (RPO), and maximum tolerable outage (MTO). These numbers should inform decisions about the architecture. For example, to achieve a low RTO, you might implement automated failover to a secondary region. But if your solution can tolerate a higher RTO, that degree of redundancy might be unnecessary.

**Document service level agreements (SLA) and service level objectives (SLO)**, including availability and performance metrics. You might build a solution that delivers 99.95% availability. Is that enough? The answer is a business decision.

**Model the application around the business domain.** Start by analyzing the business requirements. Use these requirements to model the application. Consider using a domain-driven design (DDD) approach to create [domain models](#) that reflect the business processes and use cases.

**Capture both functional and nonfunctional requirements.** Functional requirements let you judge whether the application does the right thing. Nonfunctional requirements let you judge whether the application does those things *well*. In particular, make sure that you understand your requirements for scalability, availability, and latency. These requirements will influence design decisions and choice of technology.

**Decompose by workload.** The term "workload" in this context means a discrete capability or computing task, which can be logically separated from other tasks. Different workloads may have different requirements for availability, scalability, data consistency, and disaster recovery.

**Plan for growth.** A solution might meet your current needs, in terms of number of users, volume of transactions, data storage, and so forth. However, a robust application can handle growth without major architectural changes. See [Design to scale out](#) and [Partition around limits](#). Also consider that your business model and business requirements will likely change over time. If an application's service model and data models are too rigid, it becomes hard to evolve the application for new use cases and scenarios. See [Design for evolution](#).

**Manage costs.** In a traditional on-premises application, you pay upfront for hardware as a capital expenditure. In a cloud application, you pay for the resources that you consume. Make sure that you understand the pricing model for the services that you consume. The total cost will include network bandwidth usage, storage, IP addresses, service consumption, and other factors. For more information, see [Azure pricing](#). Also consider your operations costs. In the cloud, you don't have to manage the hardware or other infrastructure, but you still need to manage your applications, including DevOps, incident response, disaster recovery, and so forth.