Pillars of software quality

08/30/2018 • 9 minutes to read • Contributors

In this article

A successful cloud application will focus on these five pillars of software quality: Scalability, availability, resiliency, management, and security.

Pillar	Description
Scalability	The ability of a system to handle increased load.
Availability	The proportion of time that a system is functional and working.
Resiliency	The ability of a system to recover from failures and continue to function.
Management	Operations processes that keep a system running in production.
Security	Protecting applications and data from threats.

Scalability

Scalability is the ability of a system to handle increased load. There are two main ways that an application can scale. Vertical scaling (scaling *up*) means increasing the capacity of a resource, for example by using a larger VM size. Horizontal scaling (scaling *out*) is adding new instances of a resource, such as VMs or database replicas.

Horizontal scaling has significant advantages over vertical scaling:

- True cloud scale. Applications can be designed to run on hundreds or even thousands of nodes, reaching scales that are not possible on a single node.
- Horizontal scale is elastic. You can add more instances if load increases, or remove them during quieter periods.
- Scaling out can be triggered automatically, either on a schedule or in response to changes in load.
- Scaling out may be cheaper than scaling up. Running several small VMs can cost less than a single large VM.
- Horizontal scaling can also improve resiliency, by adding redundancy. If an instance goes down, the application keeps running.

An advantage of vertical scaling is that you can do it without making any changes to the application. But at some point you'll hit a limit, where you can't scale any up any more. At that point, any further scaling must be horizontal.

Horizontal scale must be designed into the system. For example, you can scale out VMs by placing them behind a load balancer. But each VM in the pool must be able to handle any client request, so the application must be stateless or store state externally (say, in a distributed cache). Managed PaaS services often have horizontal scaling and autoscaling built in. The ease of scaling these services is a major advantage of using PaaS services.

Just adding more instances doesn't mean an application will scale, however. It might simply push the bottleneck somewhere else. For example, if you scale a web front-end to handle more client requests, that might trigger lock

contentions in the database. You would then need to consider additional measures, such as optimistic concurrency or data partitioning, to enable more throughput to the database.

Always conduct performance and load testing to find these potential bottlenecks. The stateful parts of a system, such as databases, are the most common cause of bottlenecks, and require careful design to scale horizontally. Resolving one bottleneck may reveal other bottlenecks elsewhere.

Use the Scalability checklist to review your design from a scalability standpoint.

Scalability guidance

- Design patterns for scalability and performance
- Best practices: Autoscaling, Background jobs, Caching, CDN, Data partitioning

Availability

Availability is the proportion of time that the system is functional and working. It is usually measured as a percentage of uptime. Application errors, infrastructure problems, and system load can all reduce availability.

A cloud application should have a service level objective (SLO) that clearly defines the expected availability, and how the availability is measured. When defining availability, look at the critical path. The web front-end might be able to service client requests, but if every transaction fails because it can't connect to the database, the application is not available to users.

Availability is often described in terms of "9s" — for example, "four 9s" means 99.99% uptime. The following table shows the potential cumulative downtime at different availability levels.

% Uptime	Downtime per week	Downtime per month	Downtime per year
99%	1.68 hours	7.2 hours	3.65 days
99.9%	10 minutes	43.2 minutes	8.76 hours
99.95%	5 minutes	21.6 minutes	4.38 hours
99.99%	1 minute	4.32 minutes	52.56 minutes
99.999%	6 seconds	26 seconds	5.26 minutes

Notice that 99% uptime could translate to an almost 2-hour service outage per week. For many applications, especially consumer-facing applications, that is not an acceptable SLO. On the other hand, five 9s (99.999%) means no more than five minutes of downtime in a *year*. It's challenging enough just detecting an outage that quickly, let alone resolving the issue. To get very high availability (99.99% or higher), you can't rely on manual intervention to recover from failures. The application must be self-diagnosing and self-healing, which is where resiliency becomes crucial.

In Azure, the Service Level Agreement (SLA) describes Microsoft's commitments for uptime and connectivity. If the SLA for a particular service is 99.95%, it means you should expect the service to be available 99.95% of the time.

Applications often depend on multiple services. In general, the probability of either service having downtime is independent. For example, suppose your application depends on two services, each with a 99.9% SLA. The composite SLA for both services is 99.9% × 99.9% ≈ 99.8%, or slightly less than each service by itself.

Availability guidance

- Design patterns for availability
- Best practices: Autoscaling, Background jobs

Resiliency

Resiliency is the ability of the system to recover from failures and continue to function. The goal of resiliency is to return the application to a fully functioning state after a failure occurs. Resiliency is closely related to availability.

In traditional application development, there has been a focus on reducing mean time between failures (MTBF). Effort was spent trying to prevent the system from failing. In cloud computing, a different mindset is required, due to several factors:

- Distributed systems are complex, and a failure at one point can potentially cascade throughout the system.
- Costs for cloud environments are kept low through the use of commodity hardware, so occasional hardware failures must be expected.
- Applications often depend on external services, which may become temporarily unavailable or throttle high-volume users.
- Today's users expect an application to be available 24/7 without ever going offline.

All of these factors mean that cloud applications must be designed to expect occasional failures and recover from them. Azure has many resiliency features already built into the platform. For example:

- Azure Storage, SQL Database, and Cosmos DB all provide built-in data replication, both within a region and across regions.
- Azure managed disks are automatically placed in different storage scale units to limit the effects of hardware failures.
- VMs in an availability set are spread across several fault domains. A fault domain is a group of VMs that share a common power source and network switch. Spreading VMs across fault domains limits the impact of physical hardware failures, network outages, or power interruptions.

That said, you still need to build resiliency into your application. Resiliency strategies can be applied at all levels of the architecture. Some mitigations are more tactical in nature — for example, retrying a remote call after a transient network failure. Other mitigations are more strategic, such as failing over the entire application to a secondary region. Tactical mitigations can make a big difference. While it's rare for an entire region to experience a disruption, transient problems such as network congestion are more common — so target these first. Having the right monitoring and diagnostics is also important, both to detect failures when they happen, and to find the root causes.

When designing an application to be resilient, you must understand your availability requirements. How much downtime is acceptable? This is partly a function of cost. How much will potential downtime cost your business? How much should you invest in making the application highly available?

Resiliency guidance

- Designing reliable Azure applications
- Design patterns for resiliency
- Best practices: Transient fault handling, Retry guidance for specific services

Management and DevOps

This pillar covers the operations processes that keep an application running in production.

Deployments must be reliable and predictable. They should be automated to reduce the chance of human error. They should be a fast and routine process, so they don't slow down the release of new features or bug fixes. Equally important, you must be able to quickly roll back or roll forward if an update has problems.

Monitoring and diagnostics are crucial. Cloud applications run in a remote datacenter where you do not have full control of the infrastructure or, in some cases, the operating system. In a large application, it's not practical to log into VMs to troubleshoot an issue or sift through log files. With PaaS services, there may not even be a dedicated VM to log

into. Monitoring and diagnostics give insight into the system, so that you know when and where failures occur. All systems must be observable. Use a common and consistent logging schema that lets you correlate events across systems.

The monitoring and diagnostics process has several distinct phases:

- Instrumentation. Generating the raw data, from application logs, web server logs, diagnostics built into the Azure platform, and other sources.
- Collection and storage. Consolidating the data into one place.
- Analysis and diagnosis. To troubleshoot issues and see the overall health.
- Visualization and alerts. Using telemetry data to spot trends or alert the operations team.

Use the DevOps checklist to review your design from a management and DevOps standpoint.

Management and DevOps guidance

- Design patterns for management and monitoring
- Best practices: Monitoring and diagnostics

Security

You must think about security throughout the entire lifecycle of an application, from design and implementation to deployment and operations. The Azure platform provides protections against a variety of threats, such as network intrusion and DDoS attacks. But you still need to build security into your application and into your DevOps processes.

Here are some broad security areas to consider.

Identity management

Consider using Azure Active Directory (Azure AD) to authenticate and authorize users. Azure AD is a fully managed identity and access management service. You can use it to create domains that exist purely on Azure, or integrate with your on-premises Active Directory identities. Azure AD also integrates with Office365, Dynamics CRM Online, and many third-party SaaS applications. For consumer-facing applications, Azure Active Directory B2C lets users authenticate with their existing social accounts (such as Facebook, Google, or LinkedIn), or create a new user account that is managed by Azure AD.

If you want to integrate an on-premises Active Directory environment with an Azure network, several approaches are possible, depending on your requirements. For more information, see our Identity Management reference architectures.

Protecting your infrastructure

Control access to the Azure resources that you deploy. Every Azure subscription has a trust relationship with an Azure AD tenant. Use role-based access control (RBAC) to grant users within your organization the correct permissions to Azure resources. Grant access by assigning RBAC role to users or groups at a certain scope. The scope can be a subscription, a resource group, or a single resource. Audit all changes to infrastructure.

Application security

In general, the security best practices for application development still apply in the cloud. These include things like using SSL everywhere, protecting against CSRF and XSS attacks, preventing SQL injection attacks, and so on.

Cloud applications often use managed services that have access keys. Never check these into source control. Consider storing application secrets in Azure Key Vault.

Data sovereignty and encryption

Make sure that your data remains in the correct geopolitical zone when using Azure's highly available. Azure's geo-replicated storage uses the concept of a paired region in the same geopolitical region.

Use Key Vault to safeguard cryptographic keys and secrets. By using Key Vault, you can encrypt keys and secrets by using keys that are protected by hardware security modules (HSMs). Many Azure storage and DB services support data encryption at rest, including Azure Storage, Azure SQL Database, Azure SQL Data Warehouse, and Cosmos DB.

Security resources

- Azure Security Center provides integrated security monitoring and policy management across your Azure subscriptions.
- Azure Security Documentation
- Microsoft Trust Center