Cloud Design Patterns

03/01/2018 • 6 minutes to read • Contributors all

In this article

Challenges in cloud development

Catalog of patterns

These design patterns are useful for building reliable, scalable, secure applications in the cloud.

Each pattern describes the problem that the pattern addresses, considerations for applying the pattern, and an example based on Microsoft Azure. Most of the patterns include code samples or snippets that show how to implement the pattern on Azure. However, most of the patterns are relevant to any distributed system, whether hosted on Azure or on other cloud platforms.

Challenges in cloud development

Availability

Availability is the proportion of time that the system is functional and working, usually measured as a percentage of uptime. It can be affected by system errors, infrastructure problems, malicious attacks, and system load. Cloud applications typically provide users with a service level agreement (SLA), so applications must be designed to maximize availability.

Data Management

Data management is the key element of cloud applications, and influences most of the quality attributes. Data is typically hosted in different locations and across multiple servers for reasons such as performance, scalability or availability, and this can present a range of challenges. For example, data consistency must be maintained, and data will typically need to be synchronized across different locations.

Design and Implementation

Good design encompasses factors such as consistency and coherence in component design and deployment, maintainability to simplify administration and development, and reusability to allow components and subsystems to be used in other applications and in other scenarios. Decisions made during the design and implementation phase have a huge impact on the quality and the total cost of ownership of cloud hosted applications and services.

Messaging

The distributed nature of cloud applications requires a messaging infrastructure that connects the components and services, ideally in a loosely coupled manner in order to maximize scalability. Asynchronous messaging is widely used, and provides many benefits, but also brings challenges such as the ordering of messages, poison message management, idempotency, and more

Management and Monitoring

Cloud applications run in a remote datacenter where you do not have full control of the infrastructure or, in some cases, the operating system. This can make management and monitoring more difficult than an on-premises deployment. Applications must expose runtime information that administrators and operators can use to manage and monitor the system, as well as supporting changing business requirements and customization without requiring the application to be stopped or redeployed.

Performance and Scalability

Performance is an indication of the responsiveness of a system to execute any action within a given time interval, while scalability is ability of a system either to handle increases in load without impact on performance or for the available resources to be readily increased. Cloud applications typically encounter variable workloads and peaks in activity. Predicting these, especially in a multitenant scenario, is almost impossible. Instead, applications should be able to scale out within limits to meet peaks in demand, and scale in when demand decreases. Scalability concerns not just compute instances, but other elements such as data storage, messaging infrastructure, and more.

Resiliency

Resiliency is the ability of a system to gracefully handle and recover from failures. The nature of cloud hosting, where applications are often multitenant, use shared platform services, compete for resources and bandwidth, communicate over the Internet, and run on commodity hardware means there is an increased likelihood that both transient and more permanent faults will arise. Detecting failures, and recovering quickly and efficiently, is necessary to maintain resiliency.

Security

Security is the capability of a system to prevent malicious or accidental actions outside of the designed usage, and to prevent disclosure or loss of information. Cloud applications are exposed on the Internet outside trusted on-premises boundaries, are often open to the public, and may serve untrusted users. Applications must be designed and deployed in a way that protects them from malicious attacks, restricts access to only approved users, and protects sensitive data.

Catalog of patterns

Pattern	Summary
Ambassador	Create helper services that send network requests on behalf of a consumer service or application.
Anti-Corruption Layer	Implement a façade or adapter layer between a modern application and a legacy system.
Backends for Frontends	Create separate backend services to be consumed by specific frontend applications or interfaces.
Bulkhead	Isolate elements of an application into pools so that if one fails, the others will continue to function.
Cache-Aside	Load data on demand into a cache from a data store
Circuit Breaker	Handle faults that might take a variable amount of time to fix when connecting to a remote service or resource.
Claim Check	Split a large message into a claim check and a payload to avoid overwhelming a message bus.
Compensating Transaction	Undo the work performed by a series of steps, which together define an eventually consistent operation.

Pattern	Summary
Competing Consumers	Enable multiple concurrent consumers to process messages received on the same messaging channel.
Compute Resource Consolidation	Consolidate multiple tasks or operations into a single computational unit
CQRS	Segregate operations that read data from operations that update data by using separate interfaces.
Event Sourcing	Use an append-only store to record the full series of events that describe actions taken on data in a domain.
External Configuration Store	Move configuration information out of the application deployment package to a centralized location.
Federated Identity	Delegate authentication to an external identity provider.
Gatekeeper	Protect applications and services by using a dedicated host instance that acts as a broker between clients and the application or service, validates and sanitizes requests, and passes requests and data between them.
Gateway Aggregation	Use a gateway to aggregate multiple individual requests into a single request.
Gateway Offloading	Offload shared or specialized service functionality to a gateway proxy.
Gateway Routing	Route requests to multiple services using a single endpoint.
Health Endpoint Monitoring	Implement functional checks in an application that external tools can access through exposed endpoints at regular intervals.
Index Table	Create indexes over the fields in data stores that are frequently referenced by queries.
Leader Election	Coordinate the actions performed by a collection of collaborating task instances in a distributed application by electing one instance as the leader that assumes responsibility for managing the other instances.
Materialized View	Generate prepopulated views over the data in one or more data stores when the data isn't ideally formatted for required query operations.
Pipes and Filters	Break down a task that performs complex processing into a series of separate elements that can be reused.
Priority Queue	Prioritize requests sent to services so that requests with a higher priority are received and processed more quickly than those with a lower priority.
Publisher/Subscriber	Enable an application to announce events to multiple interested consumers asynchronously, without coupling the senders to the receivers.
Queue-Based Load Leveling	Use a queue that acts as a buffer between a task and a service that it invokes in order to smooth intermittent heavy loads.
Retry	Enable an application to handle anticipated, temporary failures when it tries to connect to a service or network resource by transparently retrying an operation that's previously failed.
Scheduler Agent Supervisor	Coordinate a set of actions across a distributed set of services and other remote resources.

Pattern	Summary
Sharding	Divide a data store into a set of horizontal partitions or shards.
Sidecar	Deploy components of an application into a separate process or container to provide isolation and encapsulation.
Static Content Hosting	Deploy static content to a cloud-based storage service that can deliver them directly to the client.
Strangler	Incrementally migrate a legacy system by gradually replacing specific pieces of functionality with new applications and services.
Throttling	Control the consumption of resources used by an instance of an application, an individual tenant, or an entire service.
Valet Key	Use a token or key that provides clients with restricted direct access to a specific resource or service.