**Patterns used in cloud-native applications**

The patterns that follow are listed roughly in the order in which they are introduced in the book:

* ***Request/Response—***A protocol for communication between services where a client makes a request of a remote service and in most cases expects a response. This may be done synchronously or asynchronously and is most often done over HTTP.
* ***Event-driven—***A protocol where entities in a distributed system communicate events, and these events are the means by which the services in a cloud-native application are kept up to date.
* ***CQRS (Command Query Responsibility Segregation)—***A pattern whereby the query processing (read) for a domain entity is separate from the command processing (write).
* ***Multiple service instances—***Deployment of more than one instance of apps/services to support resilience, scalability, and cloud-native operational practices.
* ***Horizontal scaling—***The creation of additional application instances to increase capacity for a service.
* ***Stateless services—***Apps/services that do not store state in memory or on local disk that is needed for subsequent invocations of the service.
* ***Stateful services—***Services such as databases and message queues designed to persist state. Used to provide persistent data for stateless services.
* ***App configuration through environment variables—***Using env variables to inject values into applications on startup.
* ***Configuration service—***A (stateful) service that is used to deliver configuration values to multiple app instances to ensure consistent operations.
* ***Configuration as code—***Managing configurations through files that are versioned and checked into source control.
* ***Zero-downtime upgrades—***A means of upgrading all app/service instances while the app remains fully functional.
* ***Rolling upgrades—***A technique whereby an app is upgraded with zero downtime by upgrading subsets of all instances incrementally, in batches.
* ***Blue/green upgrades—***A technique whereby an app is upgraded by deploying a full set of new app instances and then switching over to those in one fell swoop.
* ***Application health checks—***Implementing an endpoint that can be called to assess the health of an app.
* ***Liveness probes—***The periodic calling of application health endpoints and the re-creation of app instances when the health check fails.
* ***Server-side load balancing—***A means of routing requests across multiple app instances where the client makes the requests to a single entity—the load balancer.
* ***Client-side load balancing—***A means of routing requests across multiple app instances where the client is aware of and controls routing to the multiple instances of a service.
* ***Service discovery—***The means by which a client will find the address or addresses for a service it will invoke.
* ***Retry—***A technique whereby a client repeats a service request when it has received no response.
* ***Safe service—***A service that may be invoked zero or more times, yielding the same outcome.
* ***Idempotent service—***A service that may be invoked once or more than once, yielding the same outcome.
* ***Fallbacks—***Application logic that is executed when a request to a downstream service fails to generate a result.
* ***Circuit-breaker—***A technique used to stop requests to a failing service instance and then allow them through when the service resumes normal operations.
* ***API gateways—***A service proxy used for many things, including access control, auditing, routing and much more.
* ***Sidecars—***An approach to service proxying where the proxy sits right next to the service.
* ***Service mesh—***The network of and control plane for sidecars.
* ***Distributed tracing—***A means by which a thread of execution through a series of related but distributed services can be traced for troubleshooting purposes.
* ***Event sourcing—***A pattern where the source of truth in the software is an event log from which materialized views serve the needs of service instances.