Anatomy of Akka Serverless

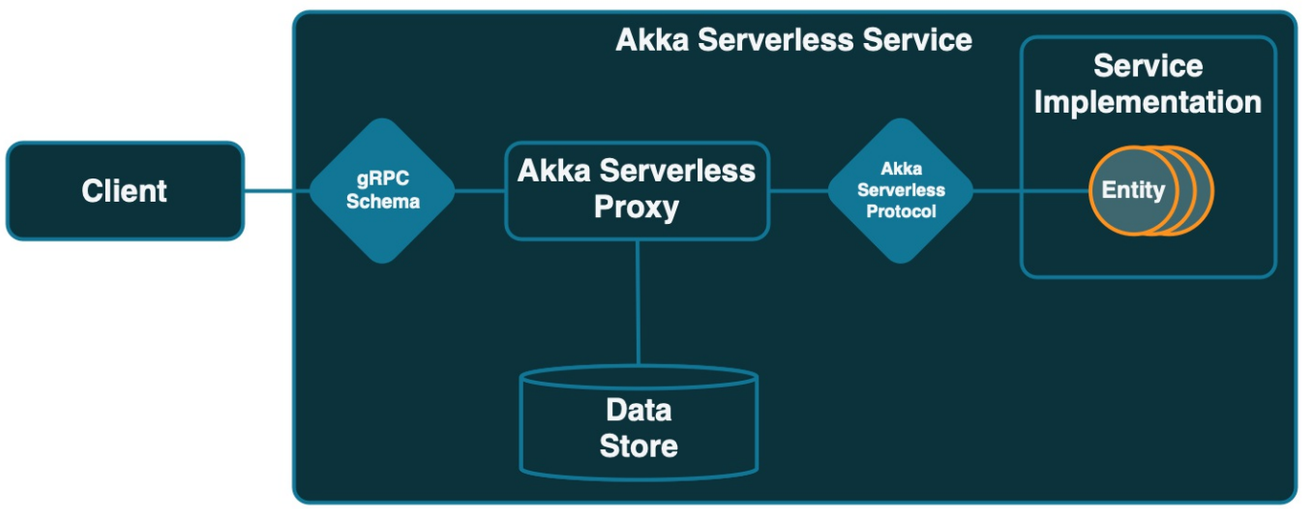
Under the Hood

- Akka Serverless is doing a lot that is invisible to the developer.
- Despite that, it is helpful to understand what is happening *under the hood*.
- Knowing the components of Akka Serverless, and how they interact, will help with maintenance.

Akka Serverless Protocol

- The Akka Serverless *protocol* is built using *gRPC*.
- A series of *language support software development kits (SDKs)* have been created using the *protocol*.
- These SDKs are at various stages of development and support.
- The JavaScript and Java SDKs are built, maintained and supported by Lightbend.
- SDKs for several other languages are community supported:
  - Python
  - Go
  - Kotlin
  - Dart
  - C#

Runtime Components



- A *Service* consists of Developer Components, and Platform Components.
- Developers are responsible for:
  - A *Service Implementation* that hosts *Entities*
  - *Clients* that communicate with the system using a *gRPC Schema*
- The Akka Serverless Platform will provide:
  - A *Proxy* or *Sidecar*
  - A *Data Store*
- We will discuss each of these in more detail.

Developer Components

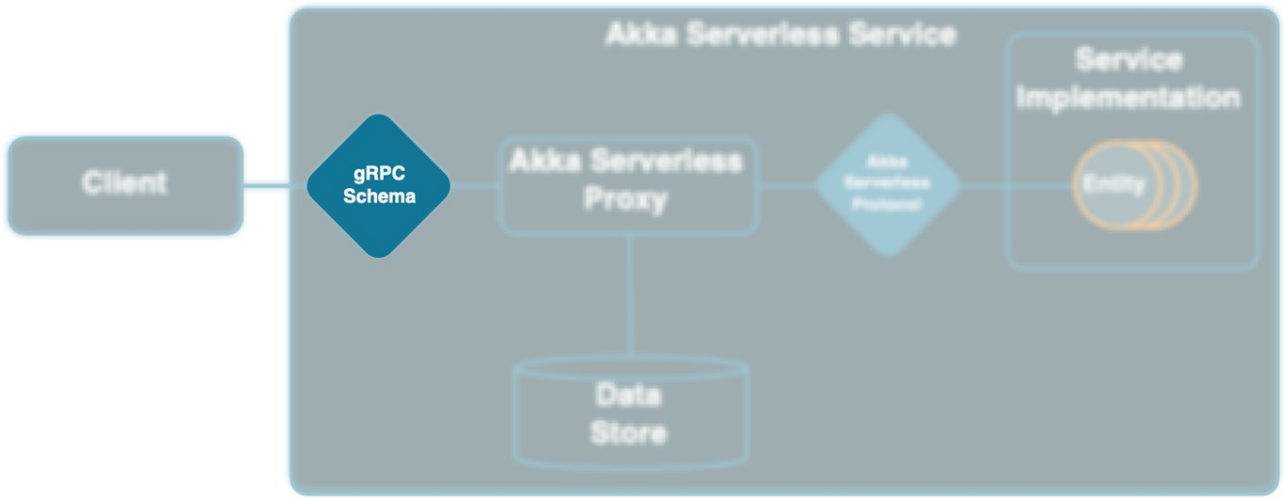
Service Implementation

- The *Service Implementation* contains code written by the developer.
- It consists of a collection of *Entities*.
- It can be written in any language with support for the Akka Serverless Protocol.
- The *Service* will be packaged into a Docker container.
- The *Service Implemention* and its *Entities* are the main focus of the developer.

Entity

- Each instance of the *Service* will host multiple *Entities*.
- *Entities* are associated with a unique identifier (eg. Order Id).
- *Entities* process messages one at a time to guarantee consistency.
- They encapsulate domain logic and state.
- Scalability is provided by running multiple *Entities* in parallel.
  - Potentially across multiple machines.
- Developers will be building *Entities*.

gRPC Schema



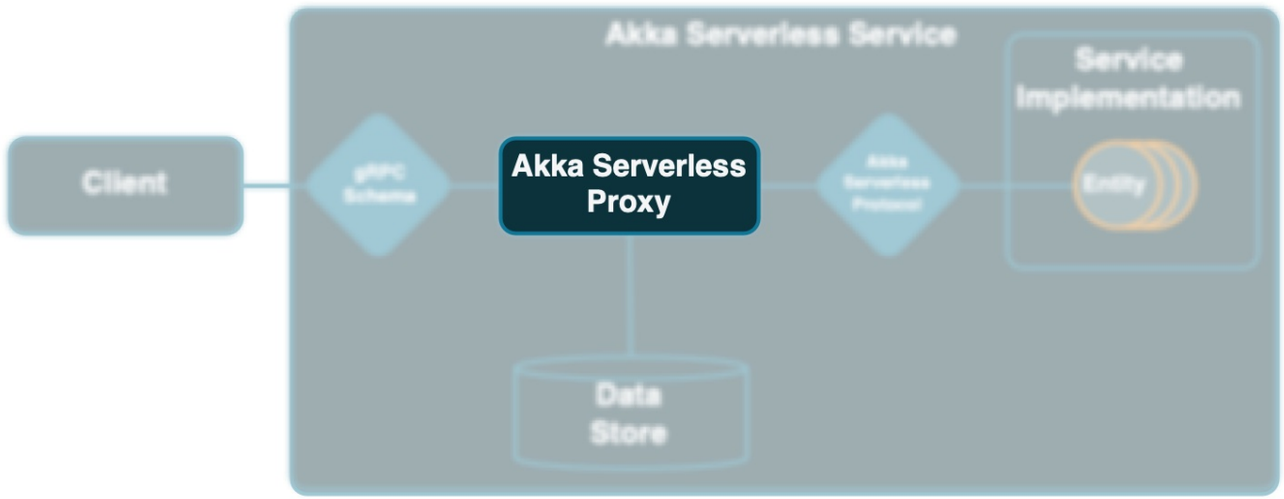
- The interface with our *Service* is defined by our *gRPC Schema*.
- It uses *protocol buffers* to define an API.
- This *gRPC Schema* can be shared with the *Client* in order to ensure compatibility.
- The *gRPC Schema* will be defined by the developer.

Client

- In addition to the application, there will be a *Client* communicating with that application.
- Rather than implementing the *Client* interface, a developer can generate one using the *gRPC Schema*.

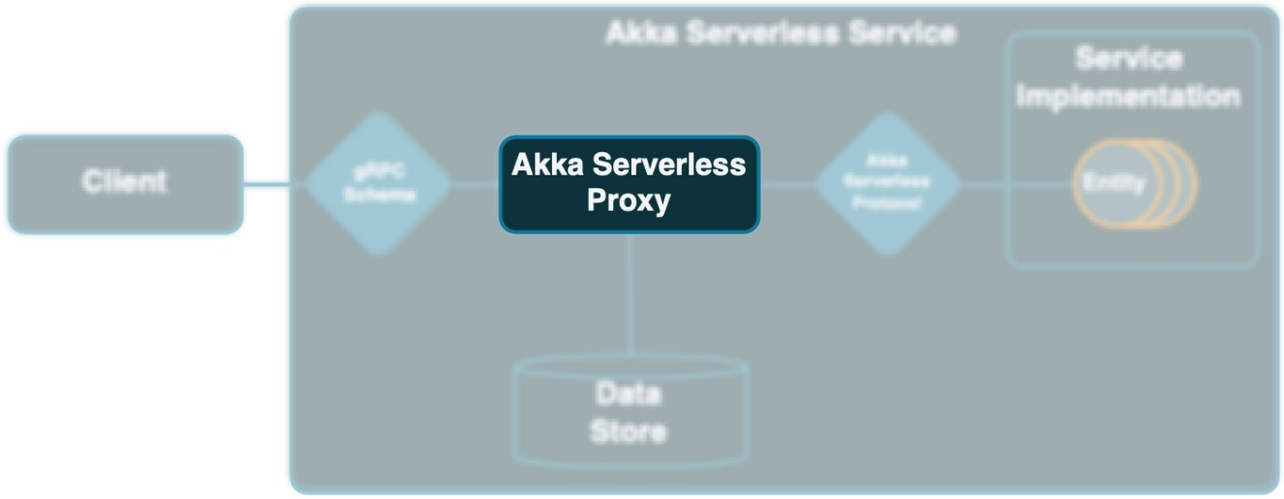
Platform Components

Proxy



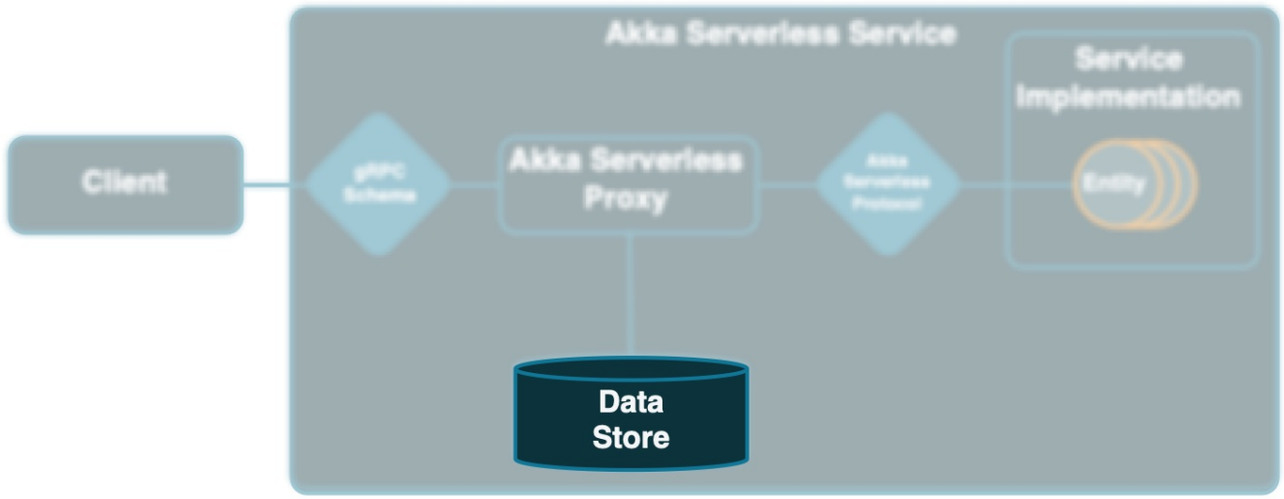
- The *Proxy* is the engine that drives Akka Serverless.
- It is a prepackaged, deployable unit.
- It is built using Akka.
- It is automatically deployed as a *Sidecar* along with your *Service Implementation*.

Proxy Responsibilities



- The *Proxy* has multiple responsibilities.
  - It maintains data in the *Data Store*.
  - It exposes the *gRPC* endpoints to the *Client*.
  - It manages communication with the *Entities* living in the *Service Implementation* using the *Akka Serverless Protocol*.

Data Store



- *Entities* can be implemented using a variety of techniques, including *Event Sourcing*.
- *Event Sourced* entities require durable storage for their *Events*.
- These *Events* are stored in a database, referred to as the *Data Store*.