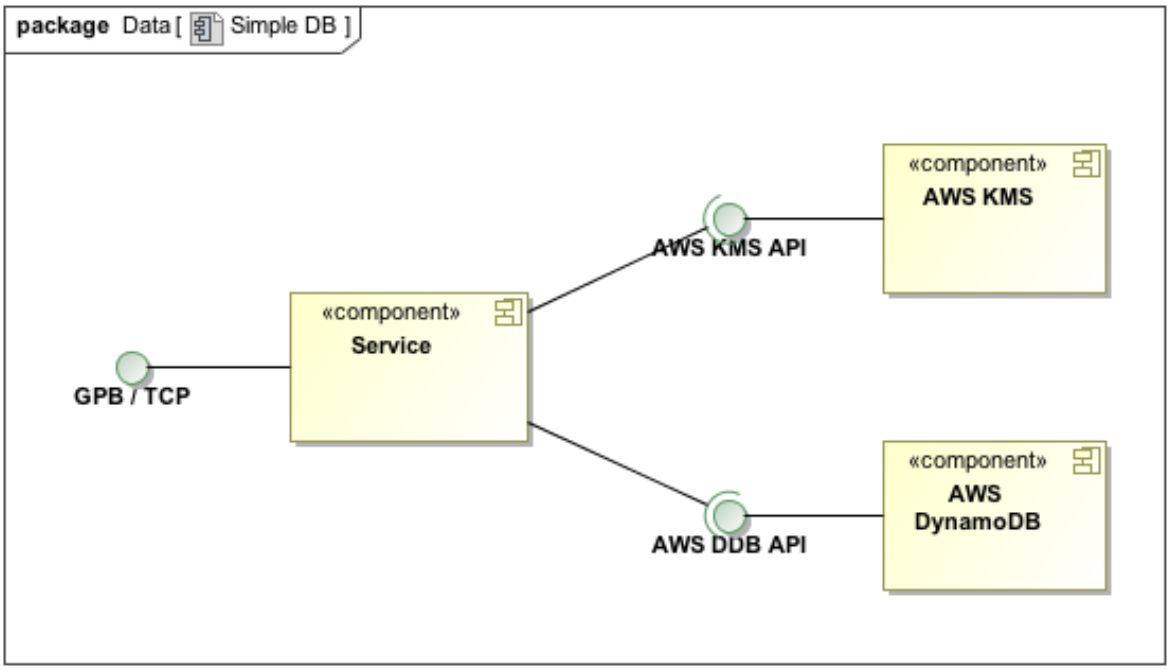
Test task for Miniclip backend developer

Given a following Google protocol buffer definition:

```
package kv.proto;

message data {
    required string key = 1;
    required string value = 2;
};

message set_request {
    required data req = 1;
};

message set_response {
    enum error_t {
        ok = 1;
        internal = 2;
    };
    required error_t error = 1;
};

message get_request {
    required string key = 1;
};

message get_response {
    enum error_t {
        ok = 1;
        not_found = 2;
        internal = 3;
    };
    required error_t error = 1;
    optional data req = 2;
};

message req_envelope {
    enum msg_type {
        set_request_t = 1;
        set_response_t = 2;
        get_request_t = 3;
        get_response_t = 4;
    };
    required msg_type type = 1;
    optional set_request set_req = 2;
    optional set_response set_resp = 3;
    optional get_request get_req = 4;
    optional get_response get_resp = 5;
};
```



Develop a **Service** and **CloudFormation template** using C++ which implements a simplified interface for storing/retrieving data in AWS DynamoDB.

Following must apply:

1) The data at rest in DynamoDB is encrypted using AWS KMS service.
2) Data bigger than 4K can be stored.
3) Service exposes an API over TCP using serialized Google protocol buffer messages as payload.
4) Use a build tool (CMake, Scons)
5) Use AWS C++ SDK (https://github.com/aws/aws-sdk-cpp)
6) A simple test-client is included with server which invokes requests over **GPB / TCP** interface to verify the **Service** functionality.
7) The server design should account for large scale concurrent clients access. (ie: use libuv, multithreaded)

Discussion

1) What are the limits of operation of the service you developed (ex: concurrent users, memory usage)?
2) Your service implementation is scalable, we mean, it's able to handle large number of concurrent users in multiple servers? How can/did you achieve scalability, software and infrastructure wise?
3) What was more challenging in the development of the service?
4) If you had infinite time how do you improve the service?