HTTP and gRPC

gRPC from HTTP

- gRPC is a powerful way to communicate with your service.
- However, a traditional HTTP connection may provide a wider reach.
- It provides an alternate way to access your API that compatible across a variety of clients.
- The Akka Serverless proxy allows you to access your services via HTTP and JSON as well as gRPC.

Specifying HTTP Endpoints

```
message MyCommandMessage {
  string message_id = 1;
}

service MyService {
  rpc MyCommand(MyCommandMessage) returns (...) {
    option (google.api.http) = {
      get: "/v1/messages/{message_id}"
    };
  }
}
```

- Use option (google.api.http) to specify HTTP mappings.
- This allows a custom URL to be used for your APIs
- It also binds URL parameters to fields in your protocol buffers
- In this case our protobuf *MyCommandMessage* will have the message_id field bound to the value in the URL.

Anatomy of an HTTP rule

```
service MyService {
  rpc MyCommand(MyCommandMessage) returns (...) {
    option (google.api.http) = {
      get: "/v1/messages/{message_id}",
      body: "*",
      additional_bindings: {
        get: "/v1/messages/{message_id}/myfield",
        response_body: "sub_field"
      }
    };
  }
}
```

- An HTTP rule is comprised of the following:
  - Pattern: get, put, post, delete, patch
  - body: The field you want to map the body to. Use "*" for all fields.
  - additional_bindings: (Optional) Create multiple HTTP bindings for a single endpoint.
  - response_body: (Optional) Extracts a single field into the response.

HTTP Rule: pattern

```
post: "/v1/orders/{category}/{product_id}"
```

- The *pattern* in the HTTP binding will be one of the following:
  - get: for *reading* resources
  - put: for idempotently *replacing* or *creating* resources in entirety
  - post: for *creating* resources
  - patch: for updating fields in existing resources
  - delete: for deleting resources
- The value is a url which can extract patterns from it bound to certain fields.

HTTP Rule: body

```
message MyCommandMessage {
  MyObject field_name = 1;
}

body: "field_name"
```

- The body will bind the JSON provided to a single field in the Protobuf *MyCommandMessage*.
- In this example, the JSON will be used to create an instance of *MyObject*.

```
body: "*"
```

- "*" will bind every JSON field to the corresponding field in the protobuf *Command*.

HTTP Rule: response_body

```
response_body: "user_id"
```

- response_body extracts just one field from your Protobuf response, and converts it to JSON.
- This is optional, and can help with hiding specific information from users of your API.

HTTP Rule: additional_bindings

```
additional_bindings: {
  post: "/orders",
  body: "*"
}
```

- Allows you to reuse the same Protobuf rpc API from multiple HTTP endpoints.
- You can also provide different filters and reuse the HTTP rules in a nested fashion.
- Useful for enabling multiple HTTP methods (i.e. GET and POST) on the same resource.