OpenLambda Dev Meeting: July 5

Overview

1. Application Progress
2. Load Balancer Research
3. Small Implementation Projects

Application Progress

HR, course management, Google docs

Overview

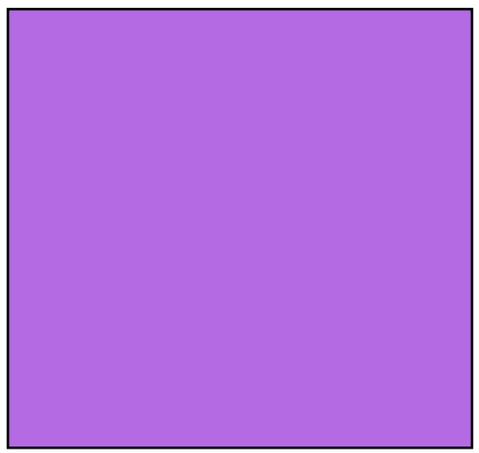
1. Application Progress
2. Load Balancer Research
3. Small Implementation Projects

gRPC Load Balancer

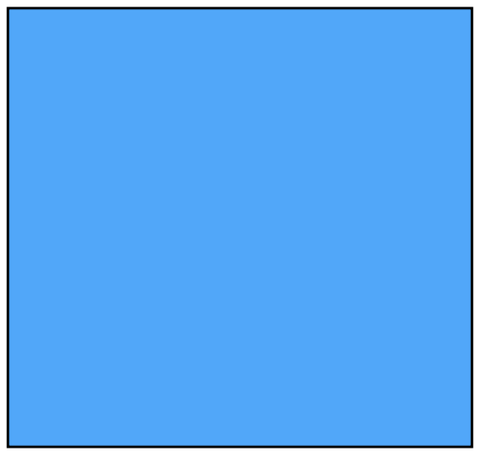
```
syntax = "proto3";

option java_package = "io.grpc.examples";

package helloworld;

// The greeter service definition.
service Greeter {
  // Sends a greeting
  rpc SayHello (HelloRequest) returns (HelloReply) {}
}

// The request message containing the user's name.
message HelloRequest {
  string name = 1;
}

// The response message containing the greetings
message HelloReply {
  string message = 1;
}
```

gRPC Load Balancer

load balancer

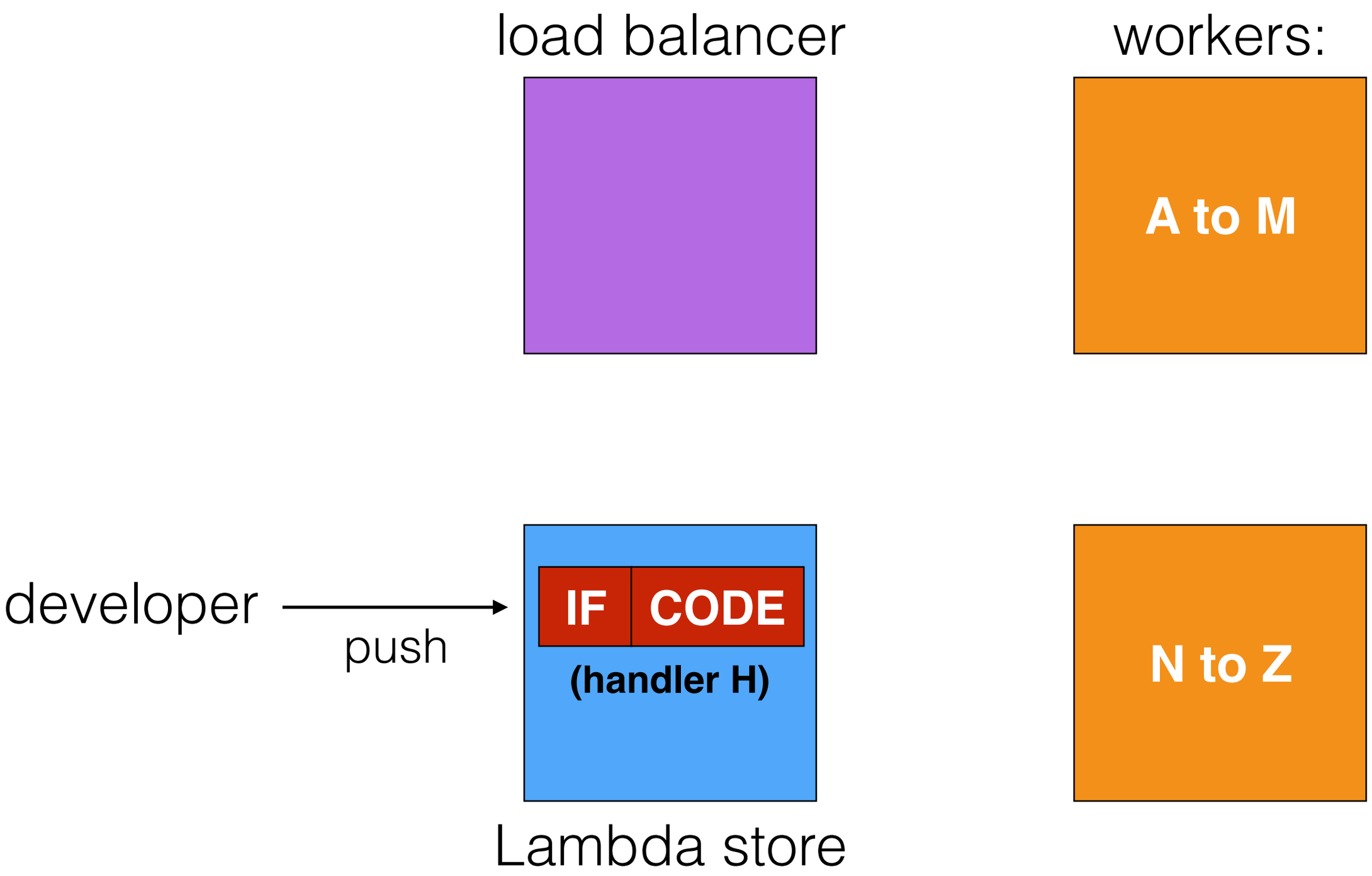


workers:



Lambda store

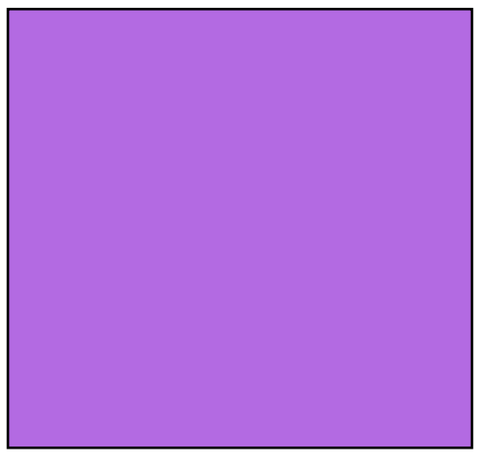
gRPC Load Balancer

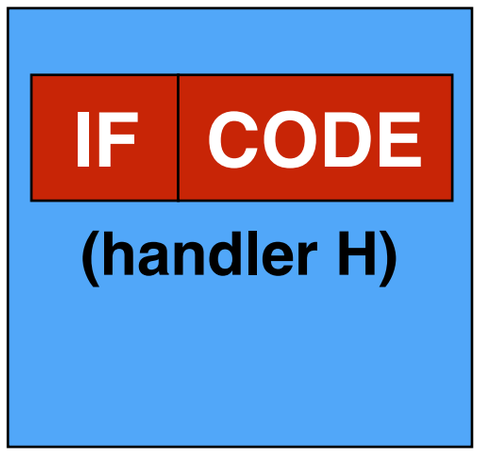


gRPC Load Balancer

load balancer

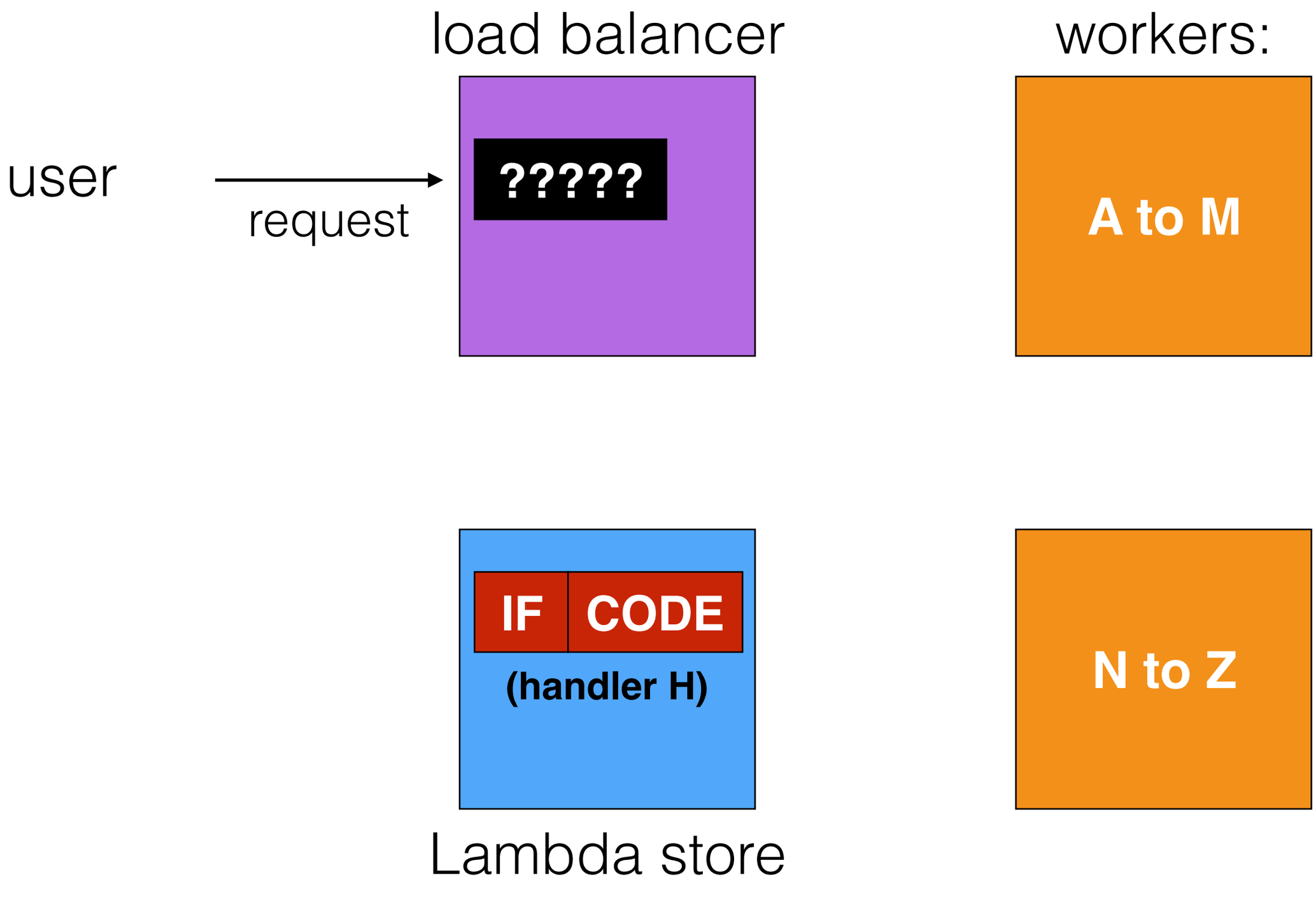
workers:



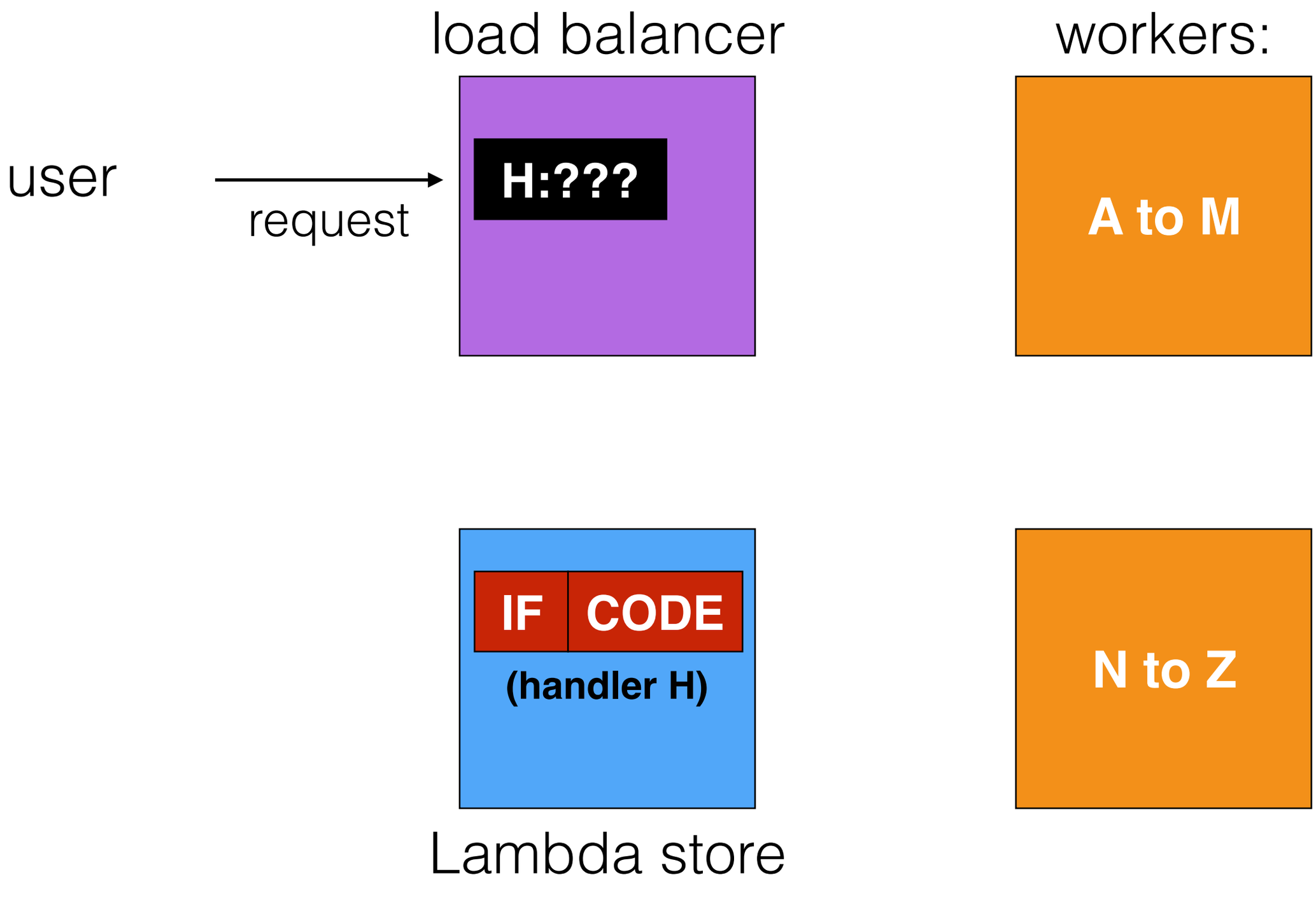


Lambda store

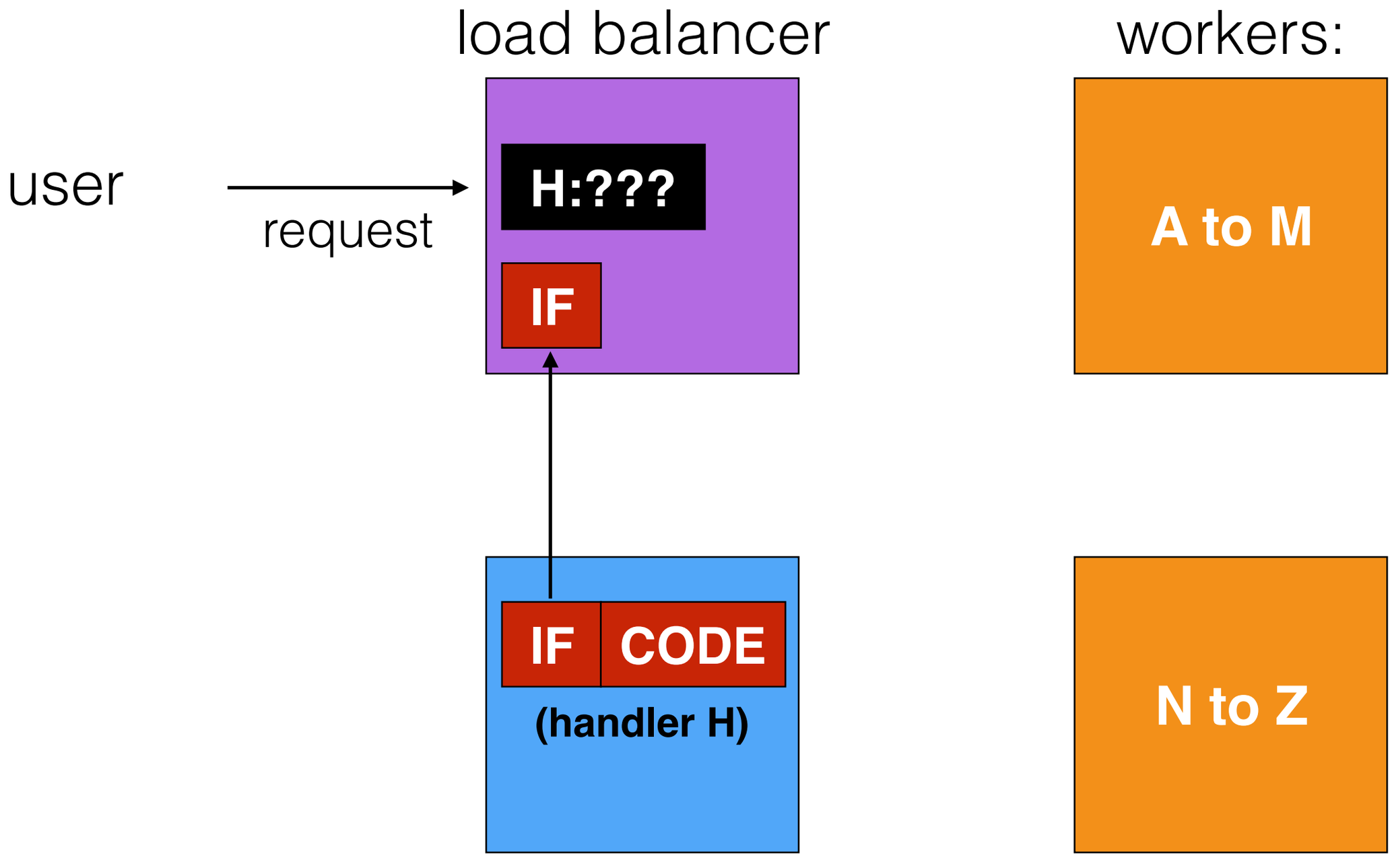
gRPC Load Balancer



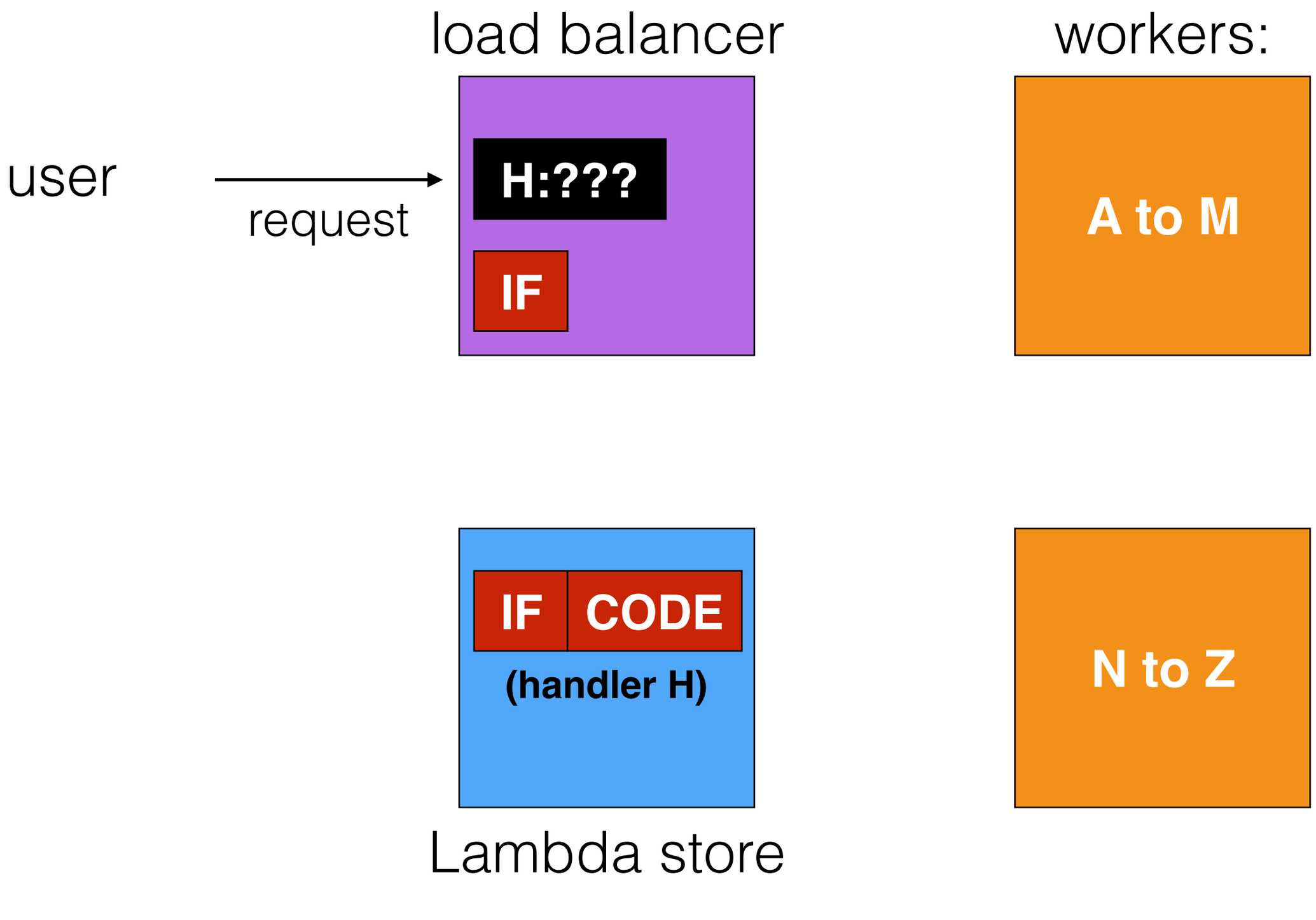
gRPC Load Balancer



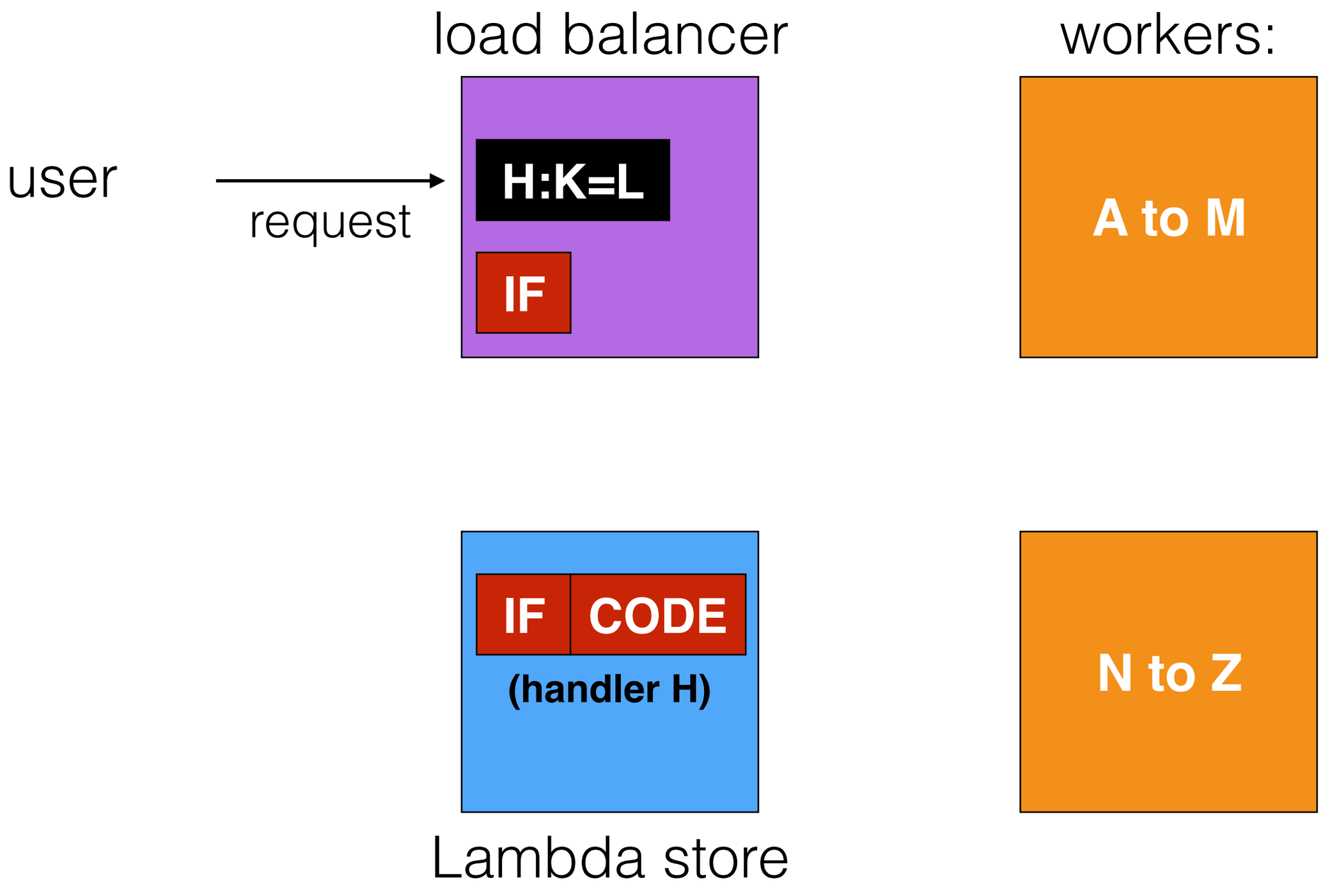
gRPC Load Balancer



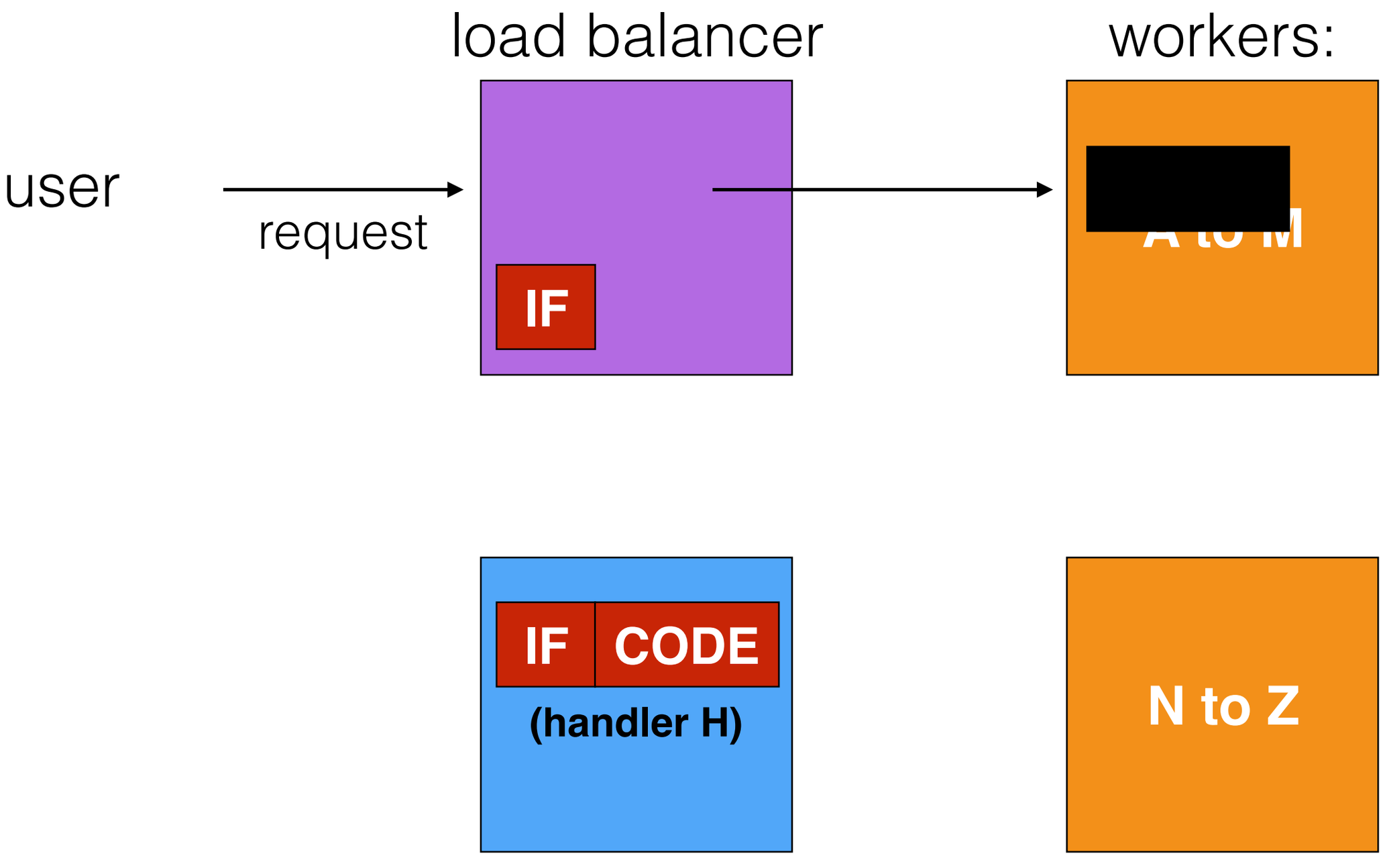
gRPC Load Balancer



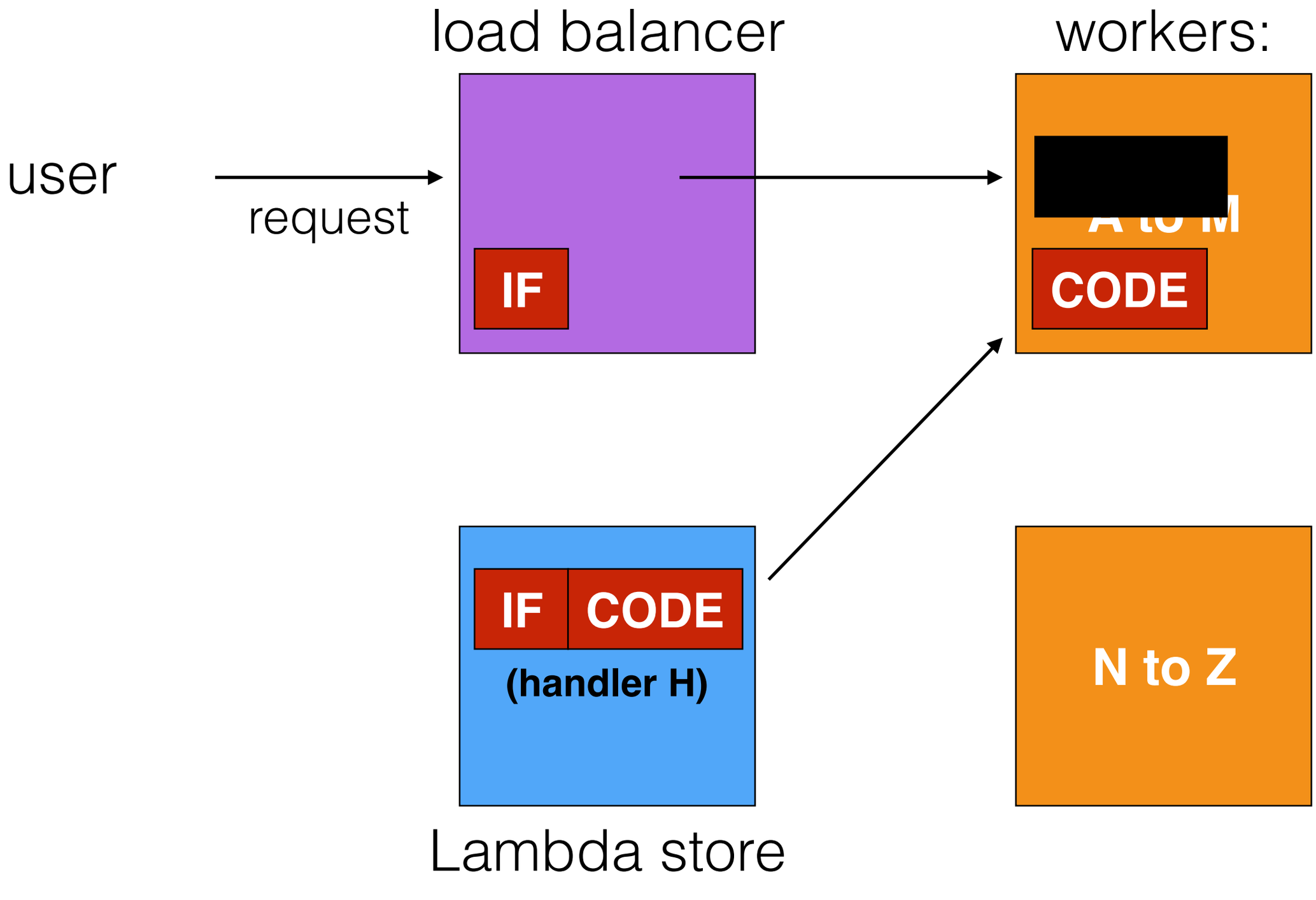
gRPC Load Balancer



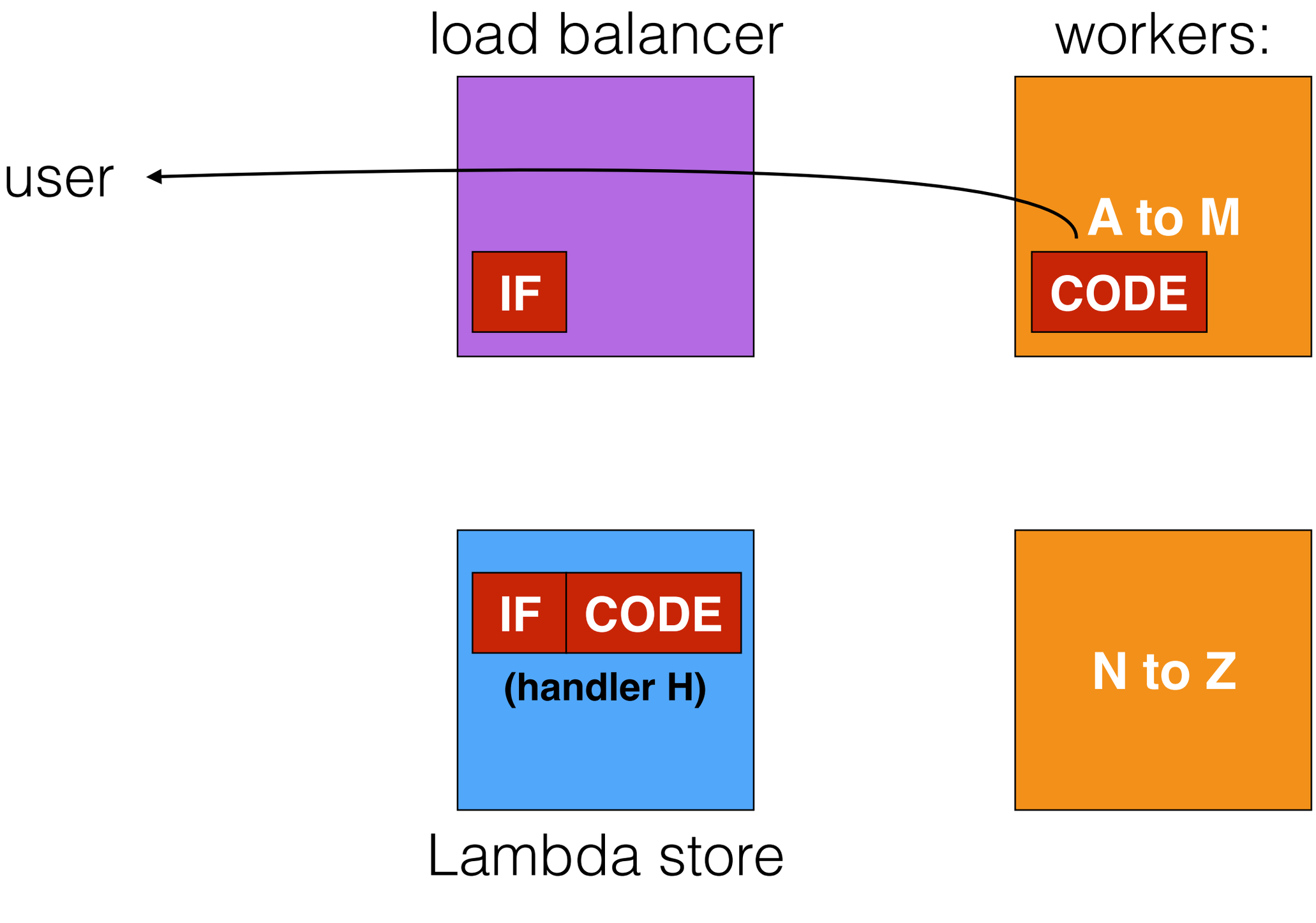
gRPC Load Balancer



gRPC Load Balancer



gRPC Load Balancer



Load Balancer Work

Support gRPC in OpenLambda

Basic balancer

Name inspection

Content inspection

Code store

IF cache (at LB)

Lots of policy: DB, session, code, load

Code: https://github.com/open-lambda/load-balancer

Overview

1. Application Progress
2. Load Balancer Research
3. Small Implementation Projects

Contributing Code

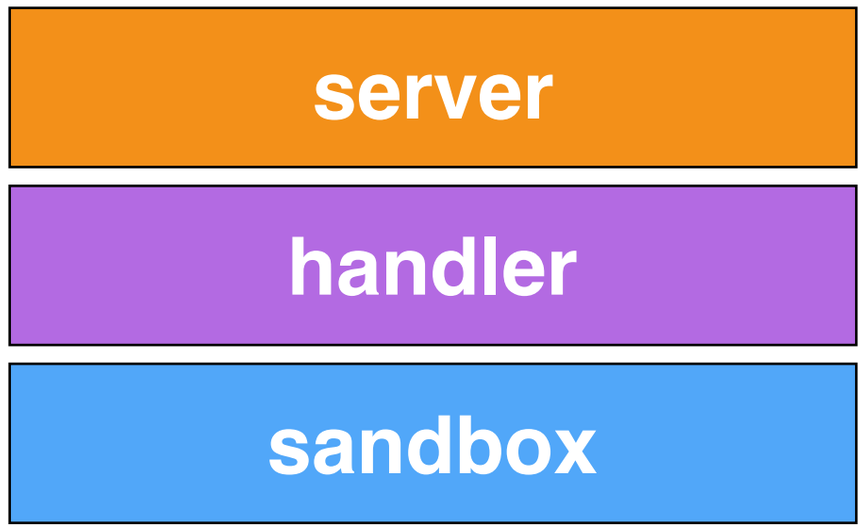
For untested changes:

```
git branch test
git checkout test
write and commit whatever
git push
```

make test (do **before push to master**)

TOKEN=XXXXX testing/digitalocean/test.py (**before release**)

OpenLambda Worker

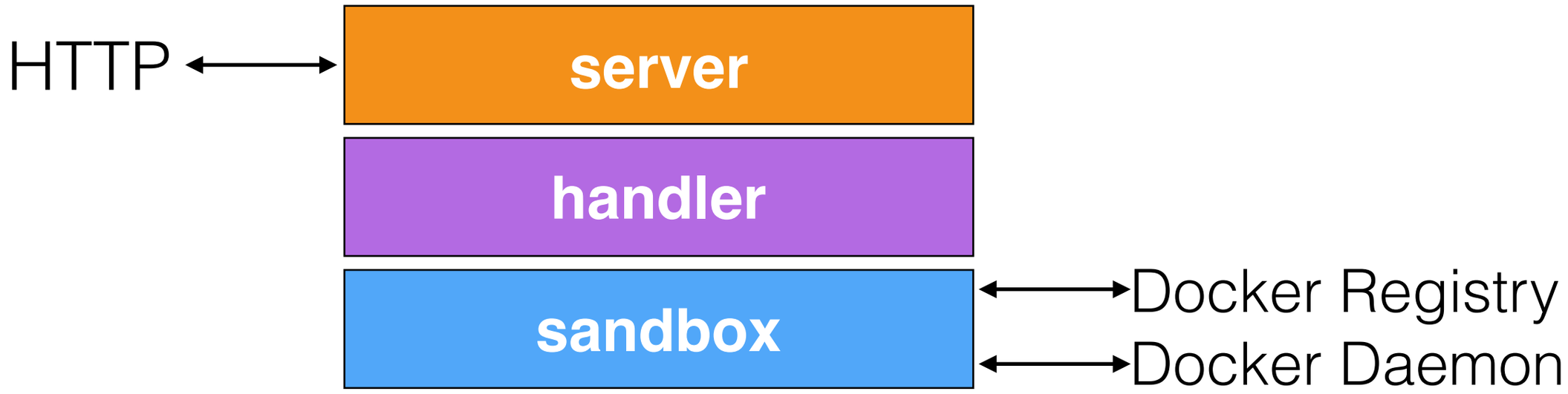


Web Server

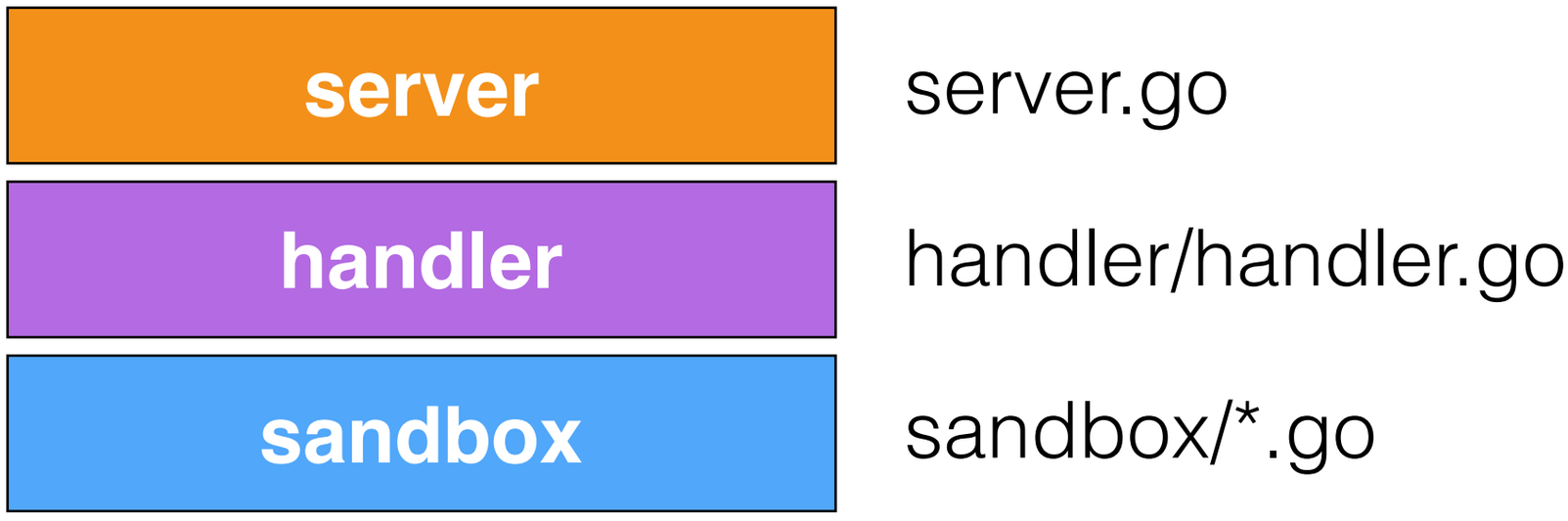
Policy

Simple execution

OpenLambda Worker



OpenLambda Worker



server.go

handler/handler.go

sandbox/*.go

Sandbox Interface

```
type Sandbox interface {
        // Starts a given sandbox
        Start() error

        // Stops a given sandbox
        Stop() error

        // Pauses a given sandbox
        Pause() error

        // Unpauses a given sandbox
        Unpause() error

        // Frees all resources associated with a given lambda
        // Will stop if needed
        Remove() error

        // Return recent log output for sandbox
        Logs() (string, error)

        // Get current state
        State() (state.HandlerState, error)

        // What port can we use to forward requests?
        Port() (string, error)
}
```

```
type SandboxManager interface {
        Create(name string) (Sandbox, error)
        Pull(name string) error
}
```

Handler Interface

```
func (h *HandlerSet) Get(name string) *Handler

func (h *Handler) RunStart() (port string, err error)

func (h *Handler) RunFinish()
```

Server Interface

```
http.HandleFunc("/runLambda/", server.RunLambda)
log.Fatal(http.ListenAndServe(":8080", nil))
```

Worker Work

Use named pipes for Lambdas

Limit container resources

Re-pull after some fixed time

Start concurrent requests in different sandboxes

Trace each level and visualize

Hide internal network interface from handlers

Build sandbox implementation that uses cgroups directly

Build “sandbox” implementation that is just processes