TYPES ALL THE WAY DOWN

GENERATING APIS WITH PROTOBUFS AND GRPC

HI!

Christopher Burnett - Lead Core Libraries

Formerly Twitter, VSCO, Posterous

@twoism

CORE LIBRARIES

A little about us...

- HTTP & gRPC Frameworks
- Tooling for Go, Python, and PHP
- Rolling out gRPC within Lyft

AGENDA

- Why choose RPC in the first place?
- Working with legacy systems
- Disrupting workflows, nicely :)
- What we've built

EVERY TEN YEARS...

A furious bout of language and protocol design takes place and a new distributed computing paradigm is announced that is compliant with the latest programming model.

– *A Note On Distributed Computing, Waldo 1994*

LOOK FAMILIAR?

- CORBA
- Thrift
- SOAP
- WDDX
- JSON-RPC
- XML-RPC
- Avro
- HyperMedia
- REST
- MessagePack

A LITTLE HISTORY

Like any good story we begin with a PHP monolith...

- Active decomp efforts
- 100s of Python Microservices
  - Flask HTTP/REST

And to keep things interesting...

- gRPC Core and Compositional Services

DEFINING A CORE SERVICE

- Organizational Primitives
  - User, Rides, Vehicles
- Zero (Service) Dependencies
  - Databases, caches, etc
- Highly Performant

SO, WHY GRPC?

LET’S TALK ABOUT REST

RESTful

RESTish

REST/JSON: S2S COMMUNICATION

```
POST /api/updateUser HTTP/1.0
Content-Type: application/json

{
  "id": 18446744073709551615,
  "username": “chris"
}
```

ALRIGHT, LET’S PAINT THAT SHED...

```
PUT /api/users HTTP/1.0
Content-Type: application/json

{
  "id": 18446744073709551615,
  "username": “chris"
}
```

PUTTING ON ANOTHER COAT...

```
PUT /api/users/18446744073709551615 HTTP/1.0
Content-Type: application/json

{
  "username": “chris"
}
```

FINISHING TOUCHES...

```
PUT /api/v1/users/18446744073709551615 HTTP/1.0
Content-Type: application/json

{
  "username": “chris"
}
```

IDLs are pretty great :)

IDLS ARE PRETTY GREAT

- Single Source of Truth
  - Primitive definitions
- Code Generation
  - APIs, Clients, Servers, Data Models, Docs, Observability
- Extensibility
  - Plugins for everything else

IDL SERVICE DEFINITION

```
package lyft.service.users.v1

service Users {
  rpc Update(UpdateRequest) UpdateResponse;
}

message UpdateRequest {
  uint64 id = 1;
  string name = 2;
}
```

What about existing services?

IDL SERVICE DEFINITION — HTTP

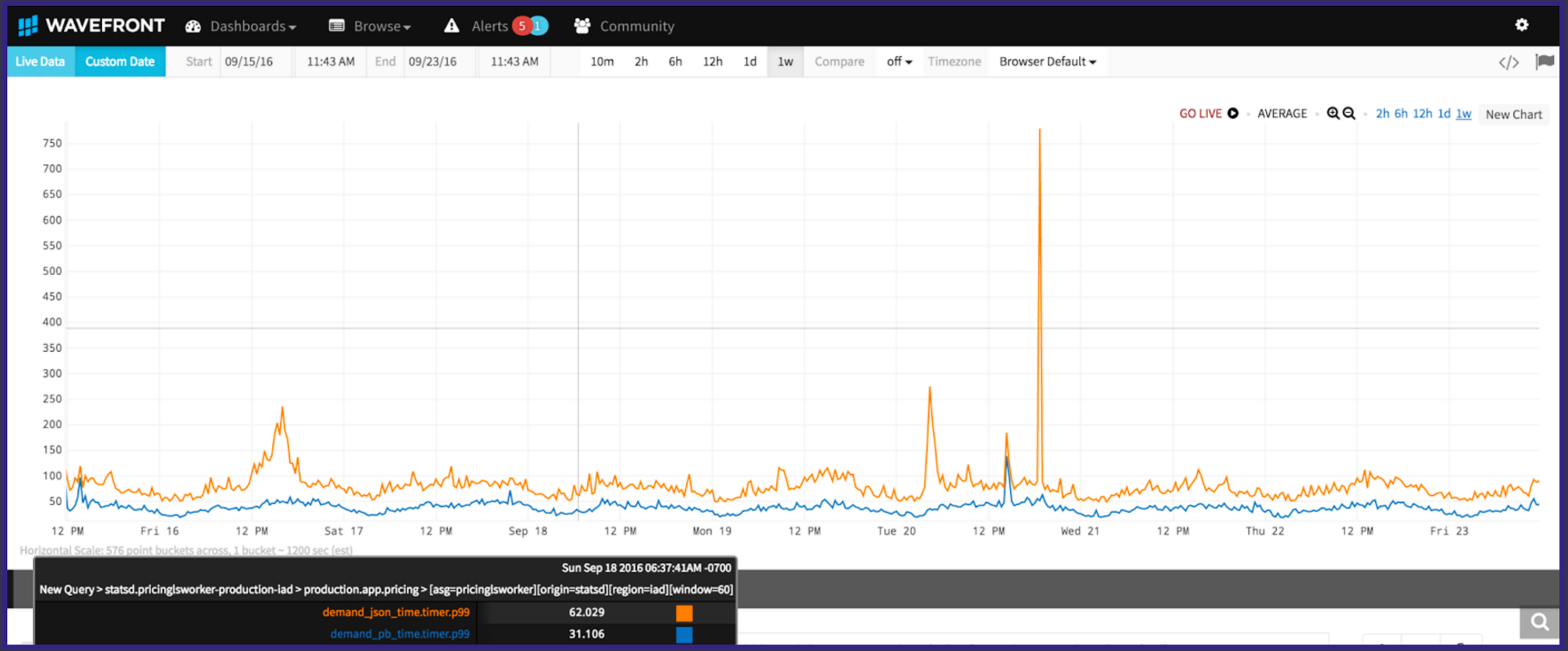
```
package lyft.service.users.v1

service Users {
 option (http.http_server_options).isHttpServer = true;

 rpc Update(UpdateRequest) returns UpdateResponse {
  // Override `path` for legacy URL support
  option (http.http_options).path = "/api/v1/users/:id";
  option (http.http_options).method = "PUT";
 }
}
```

TYPES ON THE WIRE

- Simplified API I/O
  - Structs In, Structs Out
- Safety
  - Big wins for dynamic languages
- Transfer Cost
  - Improved latencies


WAVEFRONT
Dashboards
Browse
Alerts 5 1
Community
Live Data
Custom Date
Start 09/15/16 11:43 AM
End 09/23/16 11:43 AM
10m 2h 6h 12h 1d 1w
Compare off
Timezone Browser Default
GO LIVE
AVERAGE
2h 6h 12h 1d 1w
New Chart
750 700 650 600 550 500 450 400 350 300 250 200 150 100 50
12 PM Fri 16 12 PM Sat 17 12 PM Sep 18 12 PM Mon 19 12 PM Tue 20 12 PM Wed 21 12 PM Thu 22 12 PM Fri 23
Horizontal Scale: 576 point buckets across, 1 bucket ~ 1200 sec (est)
Sun Sep 18 2016 06:37:41AM -0700
New Query > statsd.pricingIsworker-production-iad > production.app.pricing > [asg=pricingIsworker][origin=statsd][region=iad][window=60]
demand_json_time.timer.p99 62.029
demand_pb_time.timer.p99 31.106

Types on the wire eliminate an entire class of errors

TypeError: unsupported operand type(s) for -: 'NoneType' and 'float'

HTTP/2.0

HTTP/2.0

- Full Duplex Streaming
- Binary Transport
- Push
- Header Compression

WHAT'S NOT SO GREAT?

Introducing a new protocol or language can be highly traumatic for teams.

“How do I cURL this?”

WHAT CAN MAKE THIS BETTER?

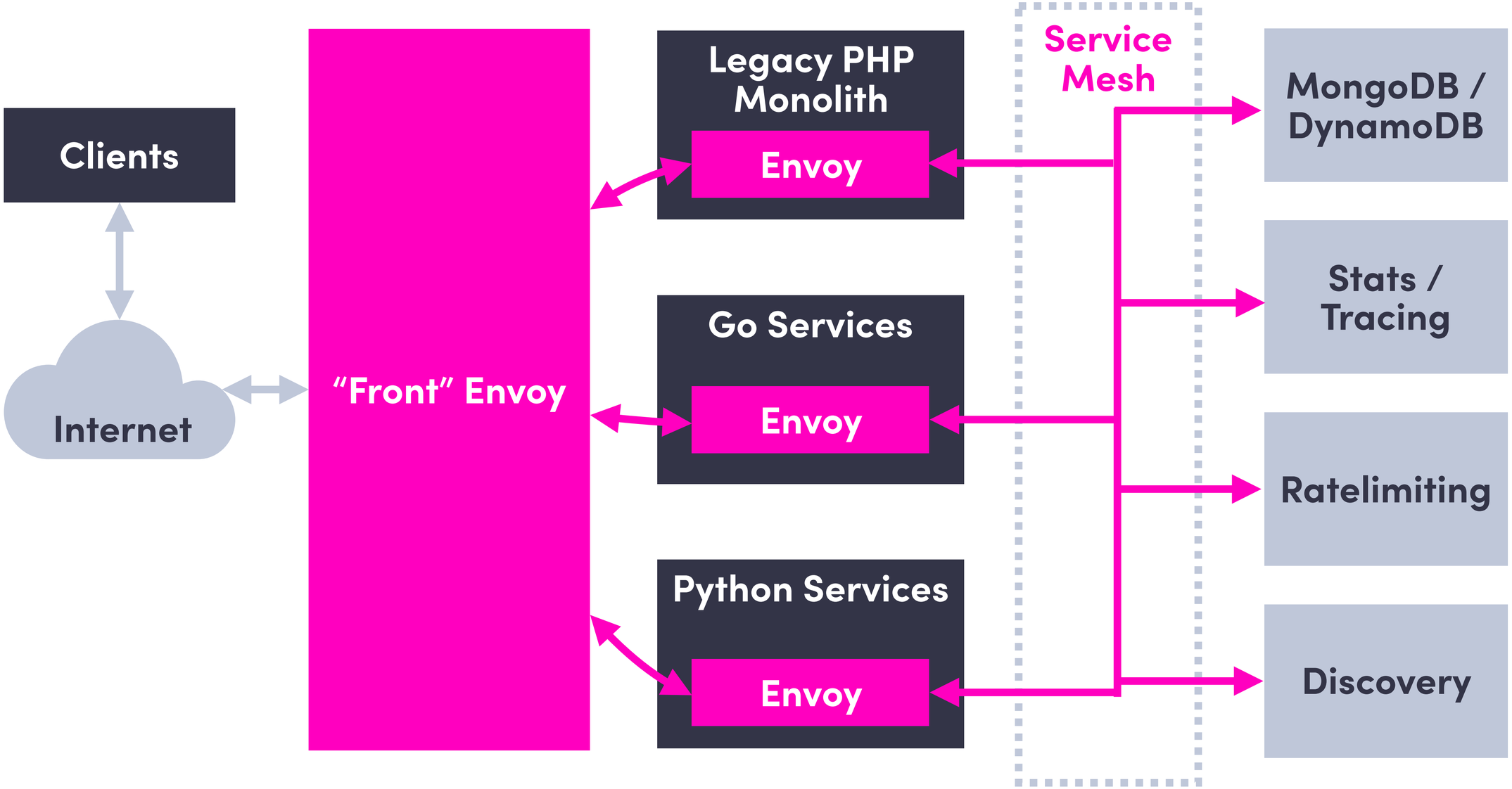
- Incremental Adoption
  - Allow teams to opt-in to the new shiny things
- Familiarity
  - Tooling that feels welcoming
  - Standardized framework patterns
- Roll Forward
  - Wire format first, then the protocol and frameworks

How can we make protocol and infra changes flexible and transparent?

envoy

by lyft

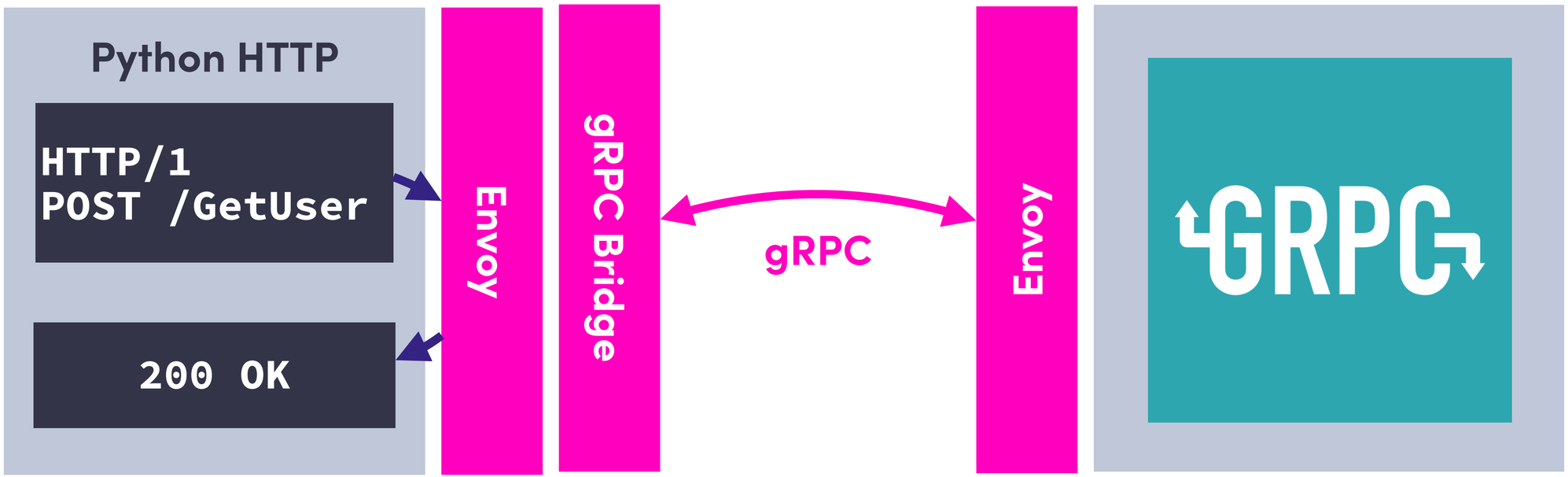
ENVOY TOPOLOGY



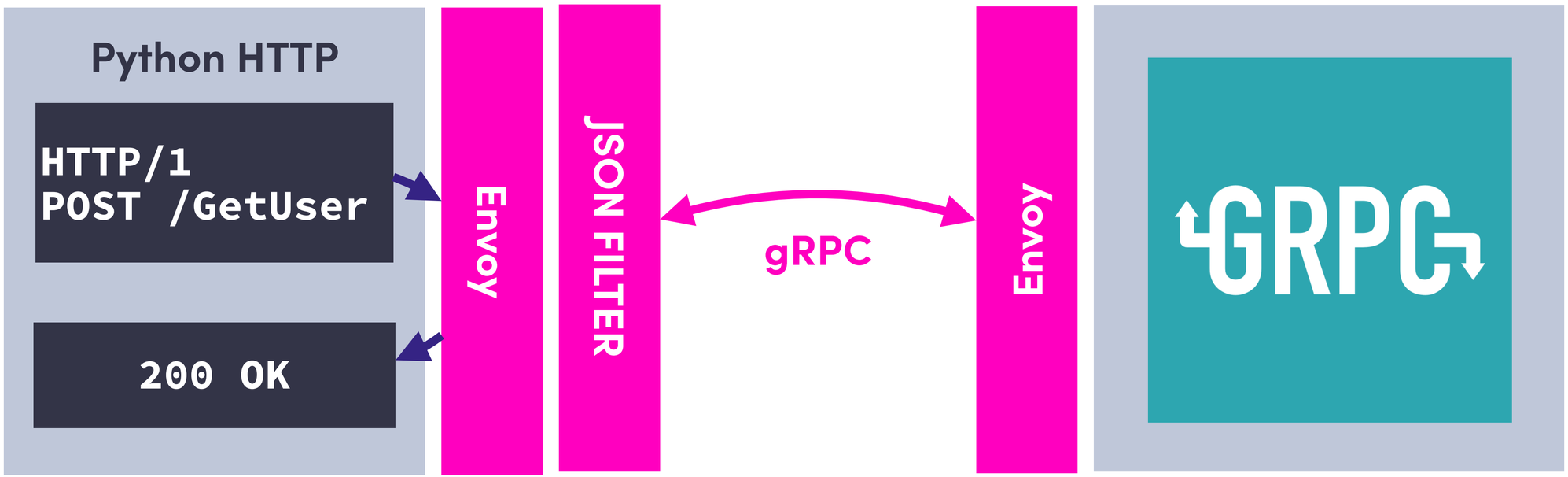
SIDECAR?


envoy
by lyft

ENVOY: GRPC BRIDGE FILTER



ENVOY: JSON PROXY FILTER

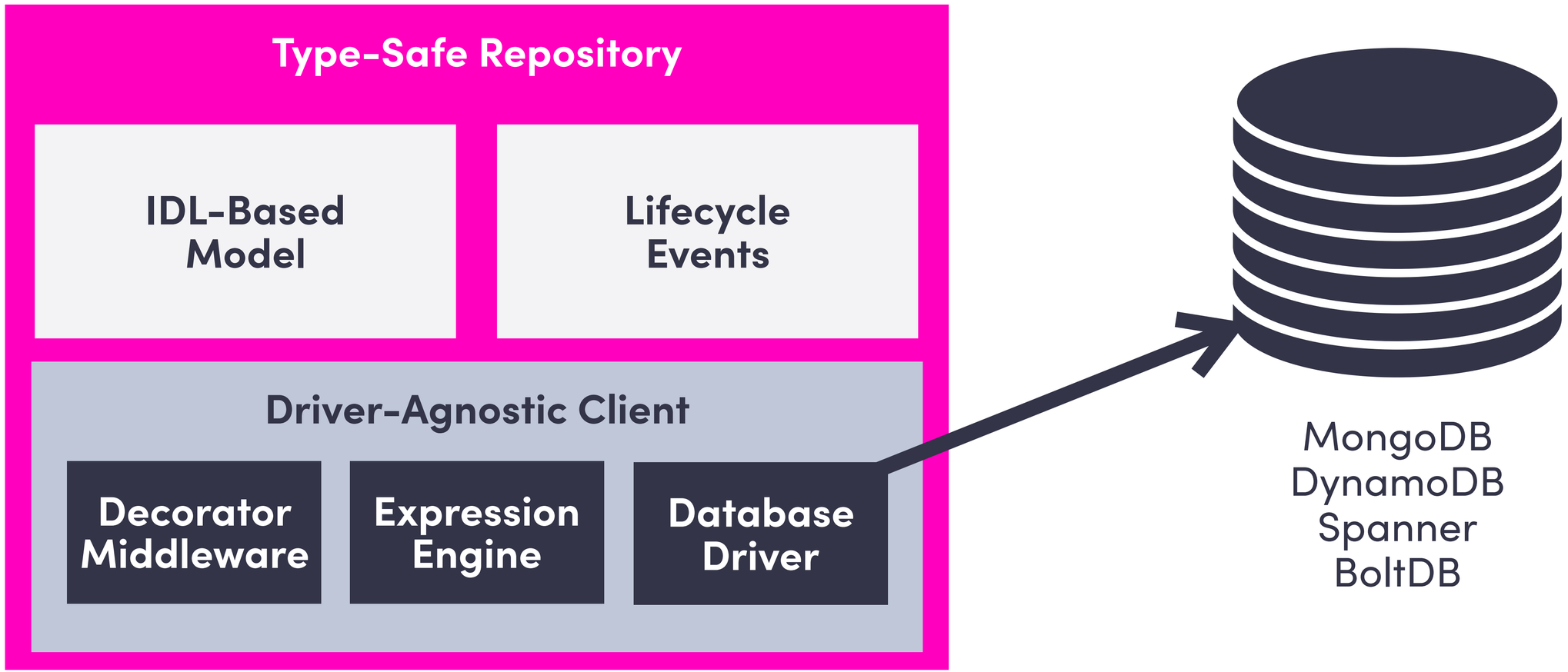


With Envoy, services and clients can evolve independently.

*Next Chris

How can we leverage IDLs beyond the API?

ODIE: IDLs MEET THE DATASTORE



ODIE: MODELS AS PROTOCOL BUFFERS

```
message User {
  option (odie.mongo).enabled = true;

  string id    = 1 [(odie.mongo).primary   = true,
                    (odie.type).object_id = true];

  string name = 2 [(odie.mongo).name          = "username"];
  int64  date = 3 [(odie.type).datetime       = true];
  uint32 vers = 4 [(odie.locking).revision = true];
}
```

ODIE: MODELS AS PROTOCOL BUFFERS

```
type UserModel struct {
  Id    bson.ObjectId `bson:"_id"`
  Name  string        `bson:"username"`
  Date  time.Time
  Vers  uint32
}

func (pb *User) ToModel() *UserModel
func (m *UserModel) ToProto() *User
```

ODIE: TYPE-SAFE REPOSITORIES

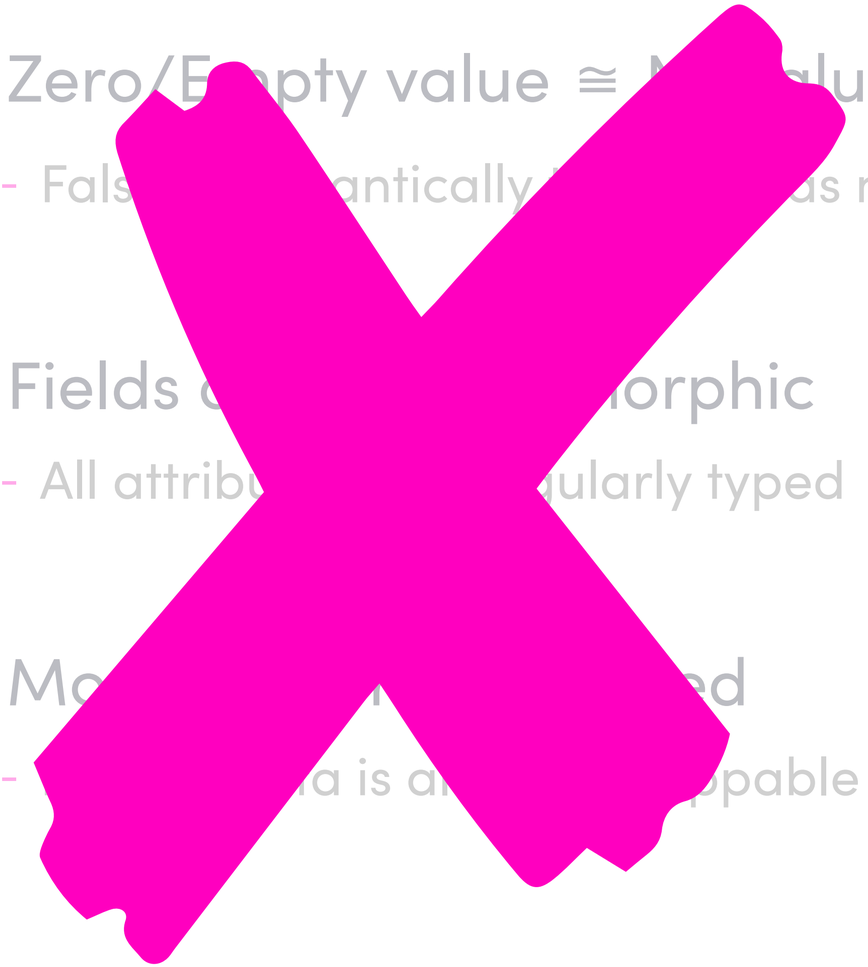
```
type UserRepo interface {
 Events() *Events

 Get(ctx context.Context, id bson.ObjectId) *GetBuilder
 Put(ctx context.Context, m *UserModel) *PutBuilder
 Delete(ctx context.Context) *DeleteBuilder
 Update(ctx context.Context) *UpdateBuilder
 Query(ctx context.Context) *QueryBuilder
}
```

ODIE: THE PLATONIC IDEAL

- Zero/Empty value ≅ Nil value
  - False is semantically the same as null
- Fields are not polymorphic
  - All attributes are singularly typed
- Models are rigidly typed
  - Legacy data is already mappable

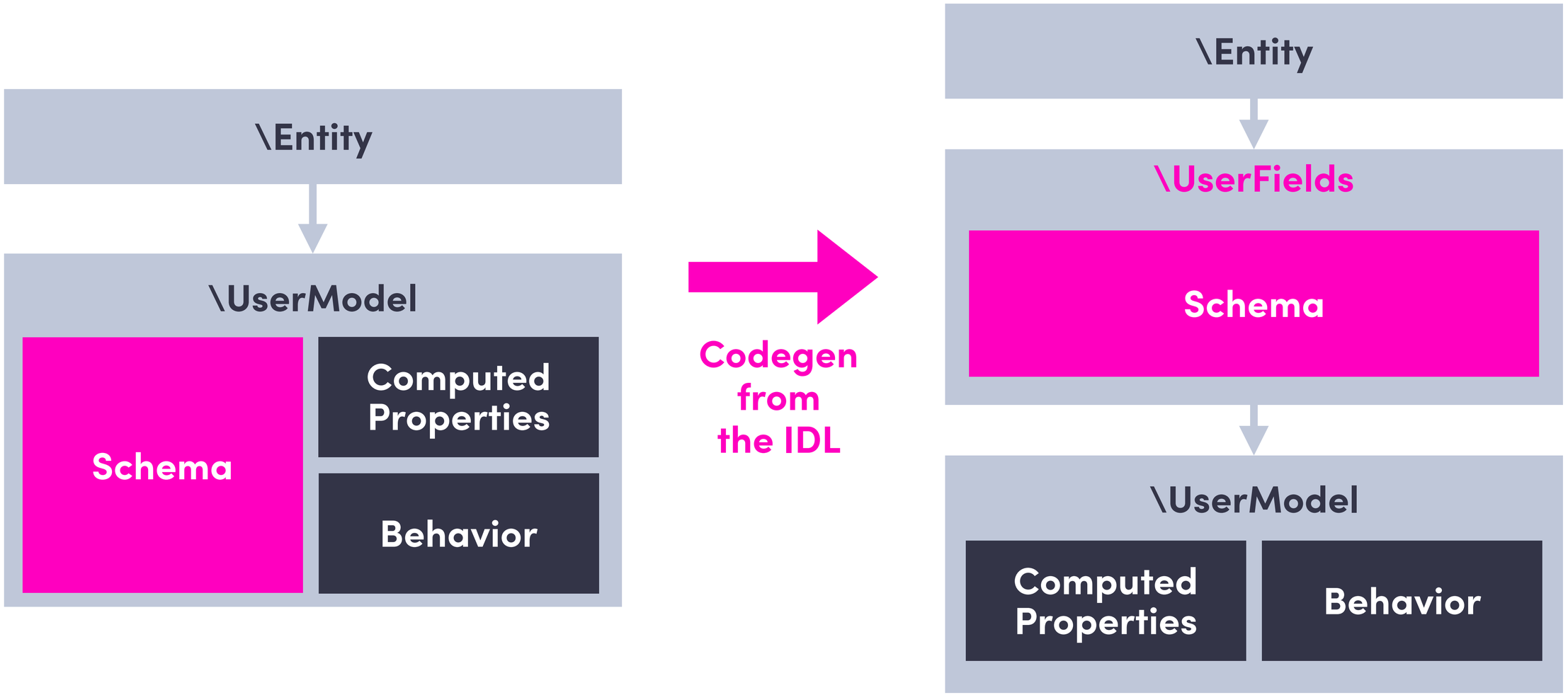
ODIE: THE REALITY



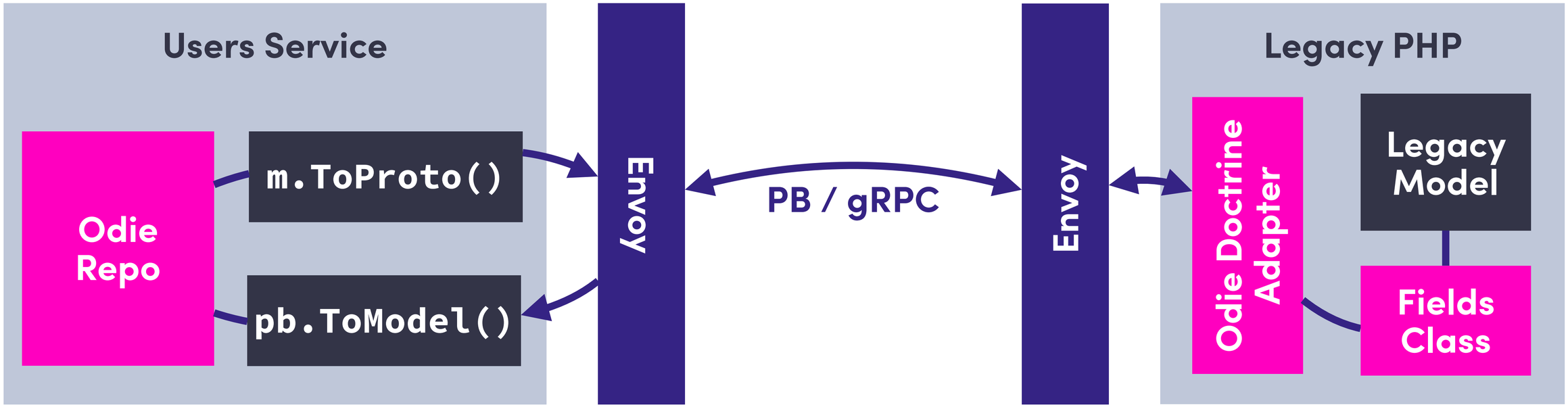
- `true` and `false` ***...and*** `nil`
  - Allow proto2 syntax models
- Covariant polymorphic repeated discriminator maps
  - Enough said.
- Need to decomp from existing PHP Doctrine models
  - Move source of truth without breaking legacy code.

“How do you change the wheels of a moving train?”

MOVE THE SOURCE OF TRUTH



MOVE THE SOURCE OF TRUTH



Reduce trauma with transitional tooling

PYTHON ENVOY-GRPC CLIENTS

```
// Envoy-gRPC Client
class UsersClient(…):
  def get(self, request)
  def multiget(self, request)
  def update(self, request)
```

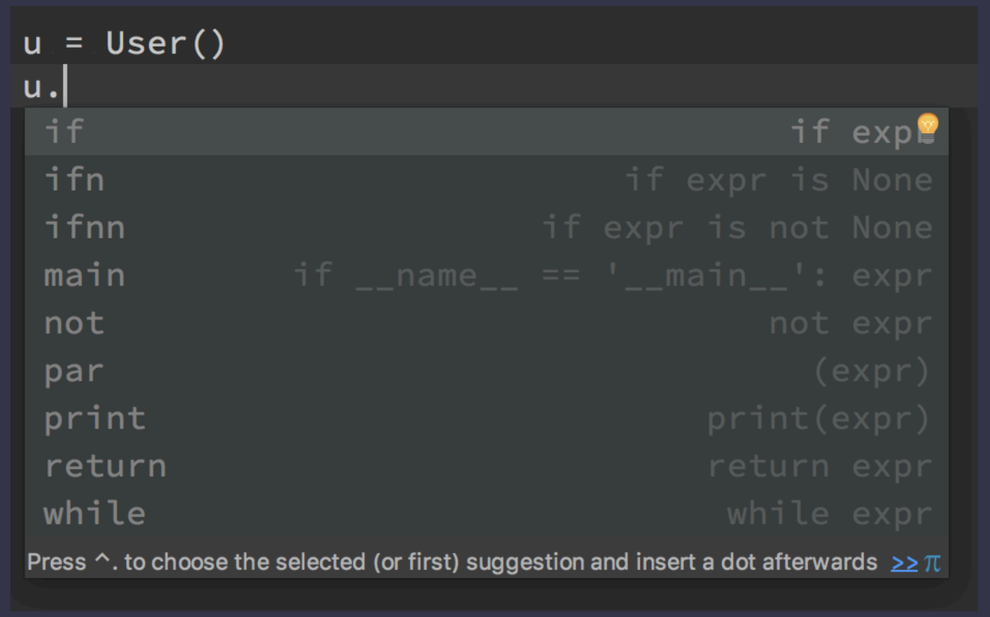
```
 // Official gRPC client
class UsersServiceClient(…):
  def get(self, request)
  def multiget(self, request)
  def update(self, request)
```

Listen to your customers

“Protobufs are really hard to use”

OFFICIAL PYTHON MESSAGES

```
User = _reflection.GeneratedProtocolMessageType('User',
(_message.Message,), dict(
  DESCRIPTOR = _USER,
  __module__ = 'pb.lyft.user_pb2'
 ))
_sym_db.RegisterMessage(User)
```

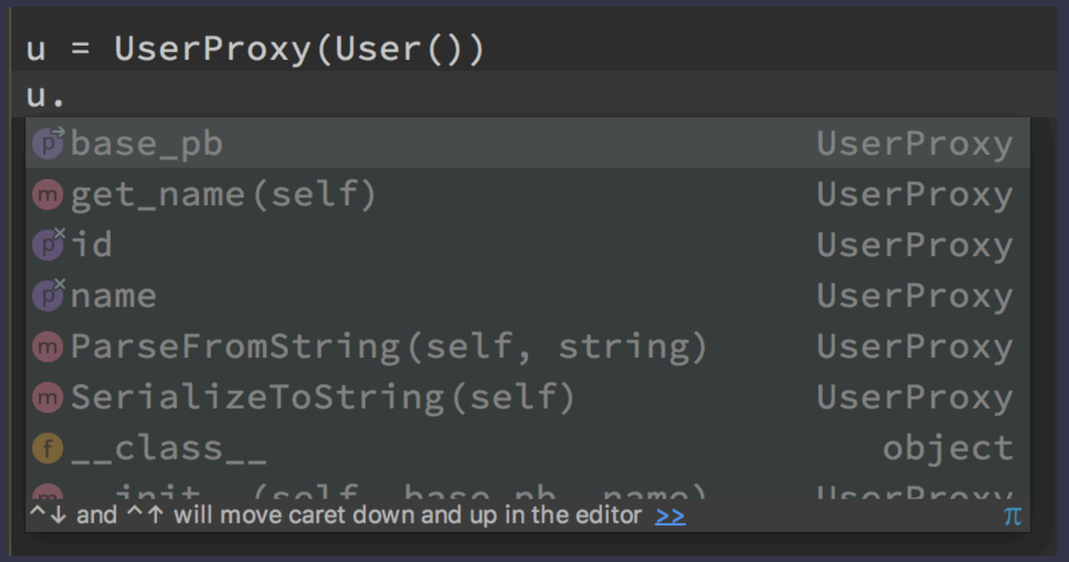


PYTHON MESSAGE PROXIES

```
class UserProxy(object):
    def __init__(self, base_pb):
        self._base = base_pb

    @property
    def id(self):
        return self._base.id

    @id.setter
    def id(self, value):
        self._base.id = value
```



FURTHER CODE GENERATION

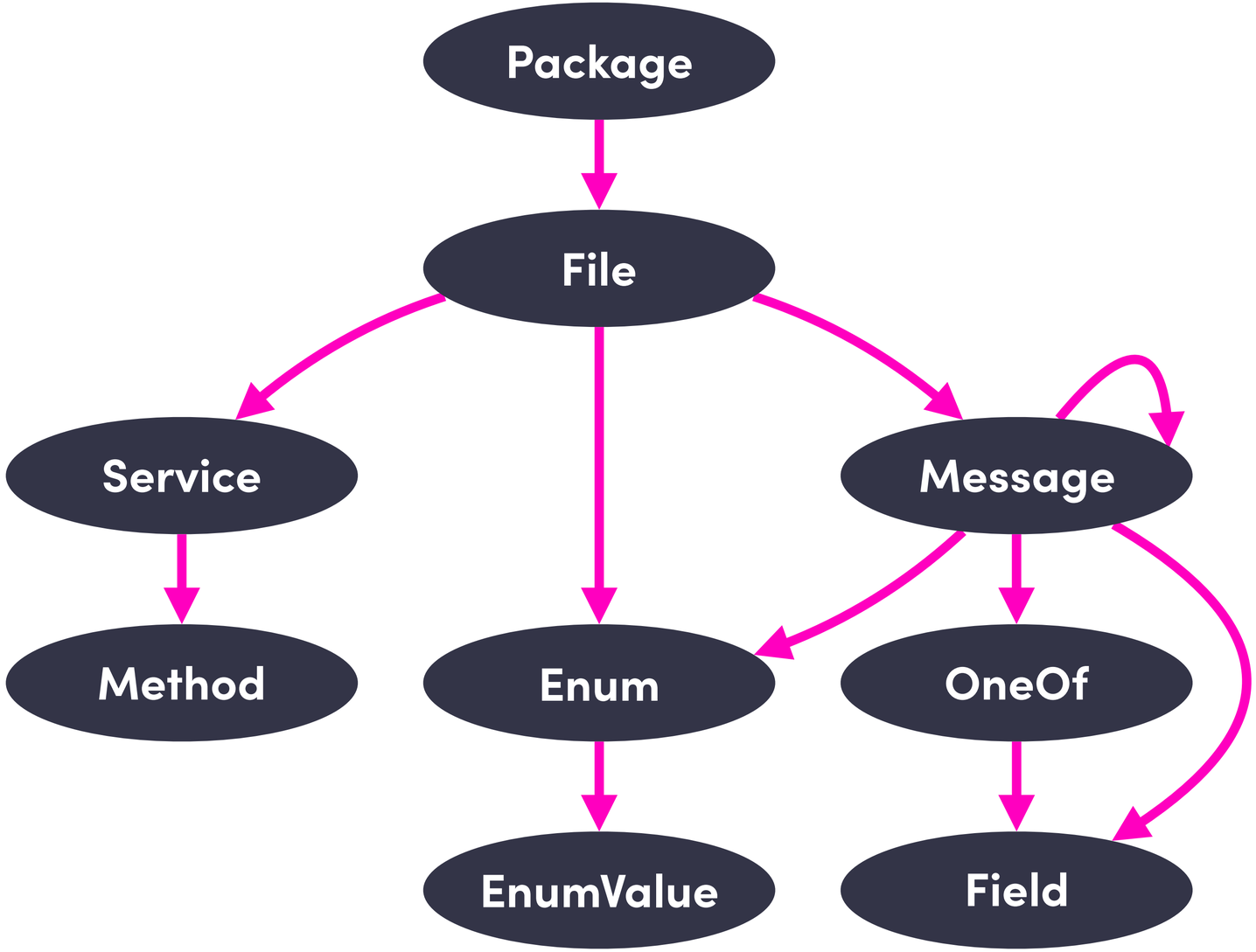
- Go/PHP Envoy-aware clients
- PB over HTTP clients/server
- Observability interceptors
- Ergonomics helpers
- Response caching
- CLI

That's an awful lot of codegen...

PROTOC-GEN-STAR (PG*)

Code generation framework

- AST of Lyft primitives
- Simplifies code generation
- Highly testable

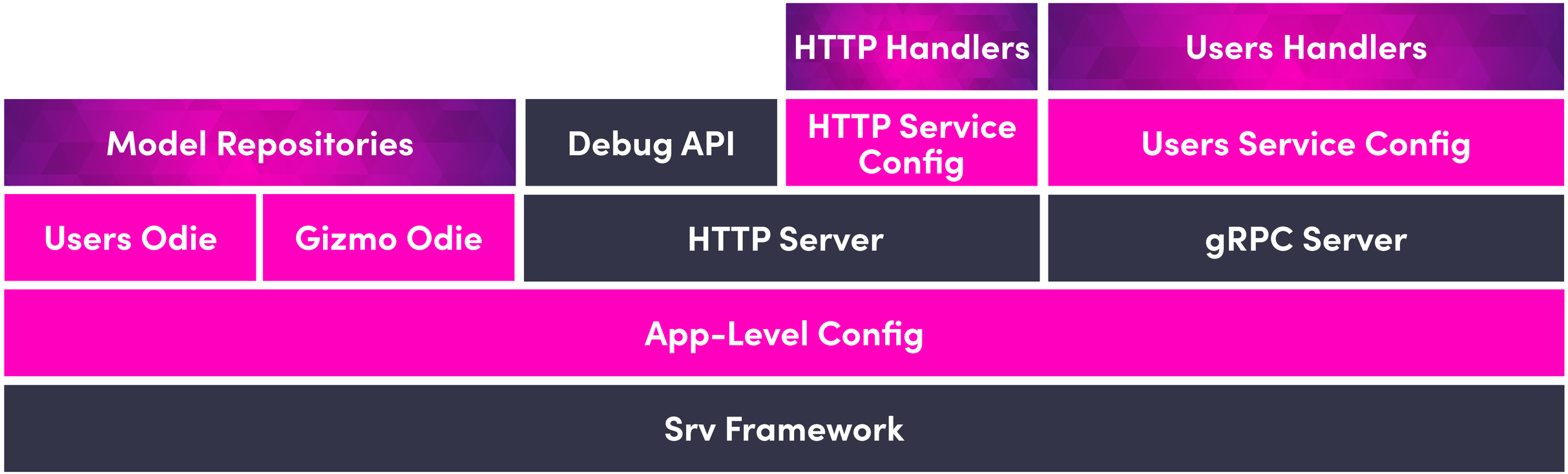


PG*: WALK THE AST

```
type Visitor interface {
 VisitPackage(Package) (v Visitor, err error)
 VisitFile(File) (v Visitor, err error)
 VisitMessage(Message) (v Visitor, err error)
 VisitEnum(Enum) (v Visitor, err error)
 VisitEnumValue(EnumValue) (v Visitor, err error)
 VisitField(Field) (v Visitor, err error)
 VisitOneOf(OneOf) (v Visitor, err error)
 VisitService(Service) (v Visitor, err error)
 VisitMethod(Method) (v Visitor, err error)
}
```

How far can we take this?

SERVICE GENERATION



FUTURE TOOLS

Linting & Static Analysis

- Enforce best practices
- Protect production code
- Core Libraries ≠ IDL Police

Mocks & Test Fixtures

- Scenarios of valid state
- Reduce reliance on integration tests
- Developer confidence

gRPC on Mobile

- Reduced payload size
- Leverage streaming APIs
- Global consistency

The incremental march continues...

Having an ideal is good

Awareness of reality is essential

THANKS!

Christopher Burnett - Lead Core Libraries

@twoism

Chris Roche - Core Libraries

@rodaine