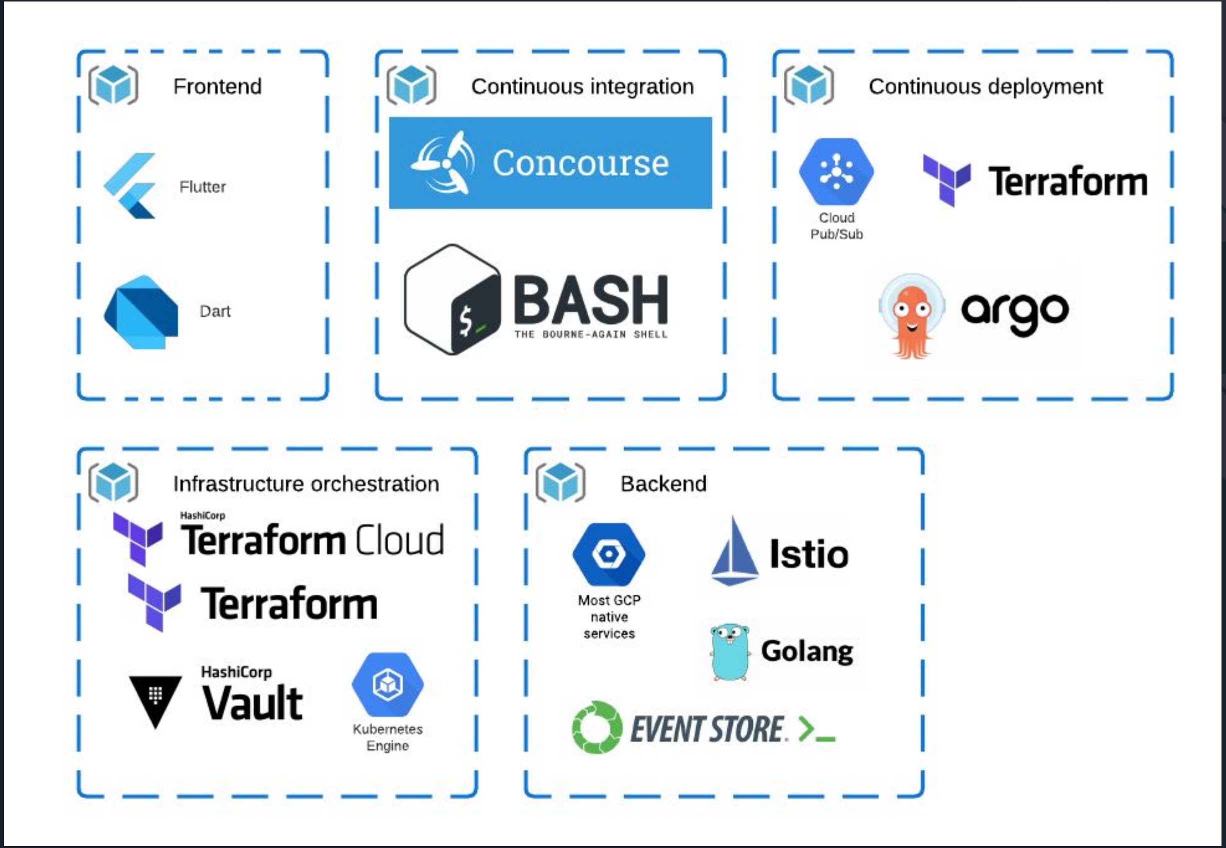
Pvotal Technologies

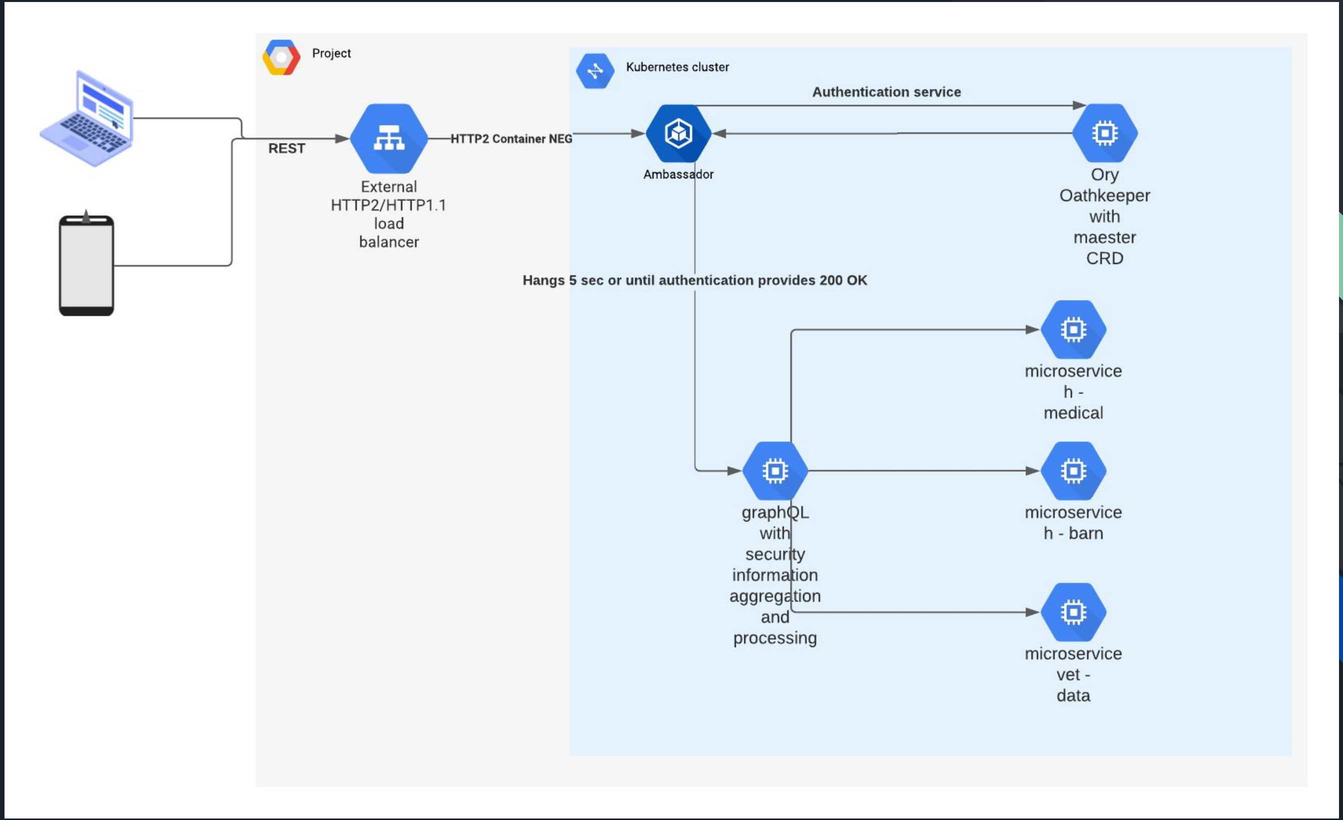
Leveraging Oathkeeper and Keto for complex infrastructure security authorization

Who are we ?

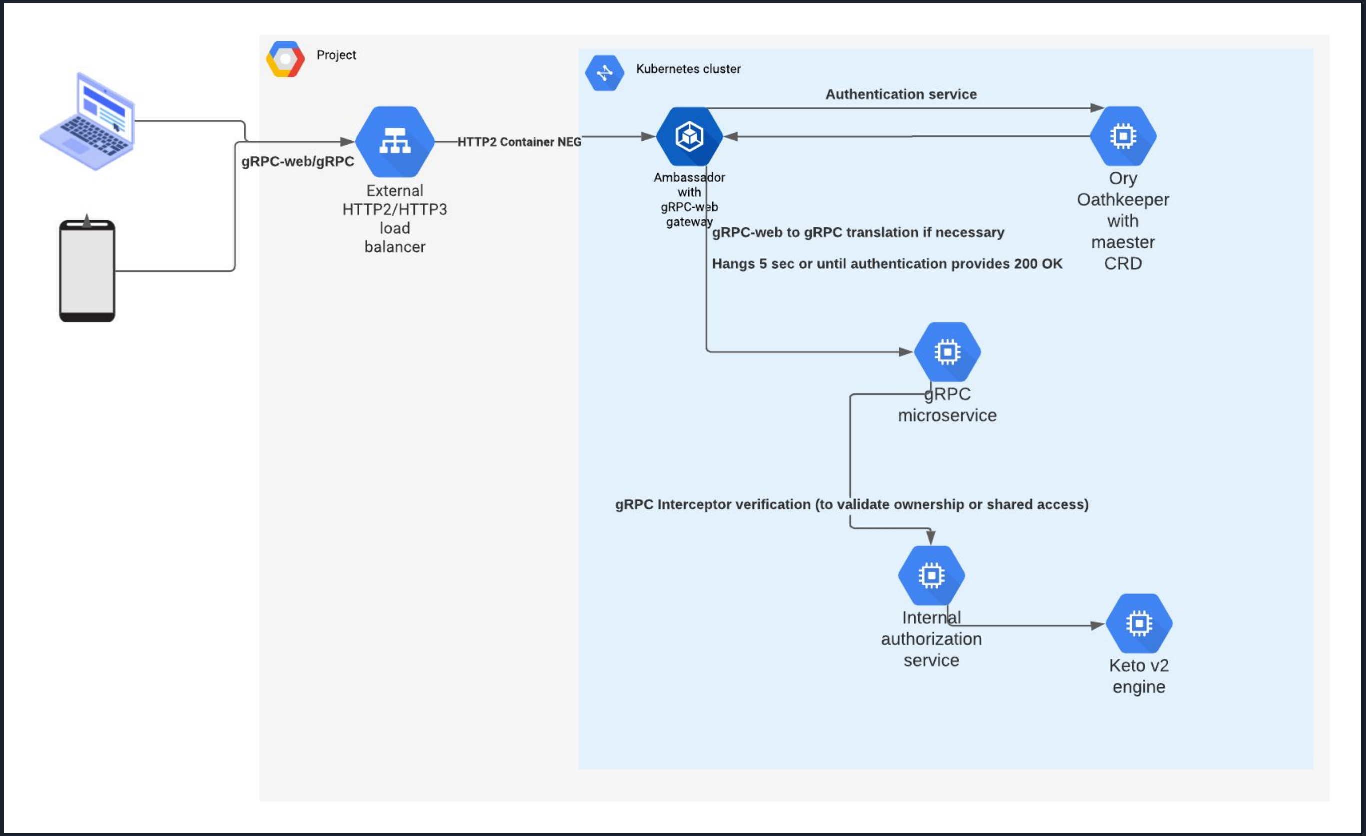
PVOTAL TECHNOLOGIES



Conservative architecture without Keto using REST

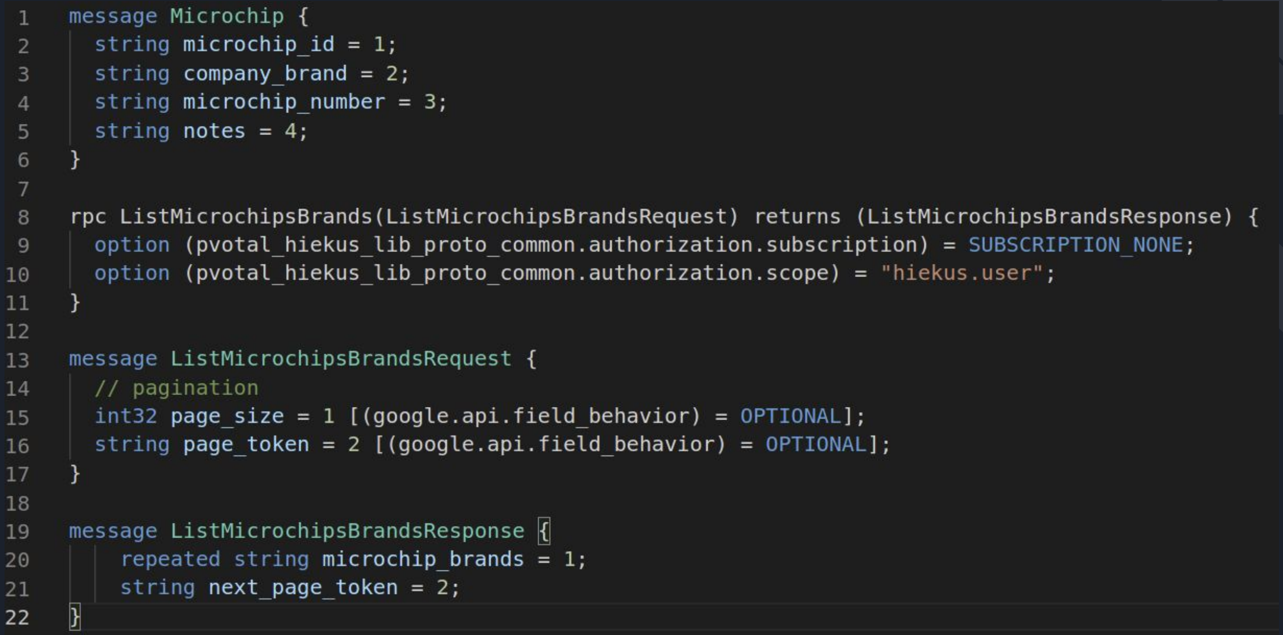

Project
Kubernetes cluster
REST
External HTTP2/HTTP1.1 load balancer
HTTP2 Container NEG
Ambassador
Authentication service
Ory Oathkeeper with maester CRD
Hangs 5 sec or until authentication provides 200 OK
graphQL with security information aggregation and processing
microservice h - medical
microservice h - barn
microservice vet - data

Decentralized authorization with Keto


Project
Kubernetes cluster
gRPC-web/gRPC
External HTTP2/HTTP3 load balancer
HTTP2 Container NEG
Authentication service
Ambassador with gRPC-web gateway
Ory Oathkeeper with maester CRD
gRPC-web to gRPC translation if necessary
Hangs 5 sec or until authentication provides 200 OK
gRPC microservice
gRPC Interceptor verification (to validate ownership or shared access)
Internal authorization service
Keto v2 engine

gRPC security validation

- Authentication service (JWT) per endpoint with Oathkeeper
- Authorization with gRPC interceptor in front of microservice

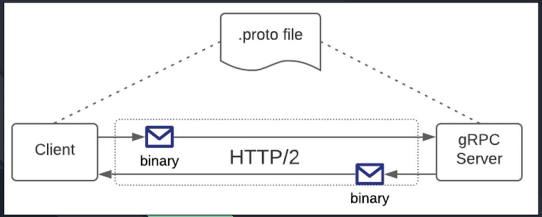


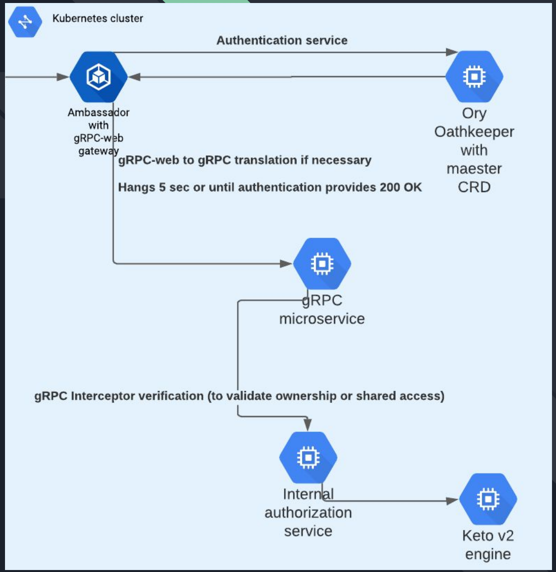
```
message Microchip {
  string microchip_id = 1;
  string company_brand = 2;
  string microchip_number = 3;
  string notes = 4;
}

rpc ListMicrochipsBrands(ListMicrochipsBrandsRequest) returns (ListMicrochipsBrandsResponse) {
  option (pvotal_hiekus_lib_proto_common.authorization.subscription) = SUBSCRIPTION_NONE;
  option (pvotal_hiekus_lib_proto_common.authorization.scope) = "hiekus.user";
}

message ListMicrochipsBrandsRequest {
  // pagination
  int32 page_size = 1 [(google.api.field_behavior) = OPTIONAL];
  string page_token = 2 [(google.api.field_behavior) = OPTIONAL];
}

message ListMicrochipsBrandsResponse {
    repeated string microchip_brands = 1;
    string next_page_token = 2;
}
```

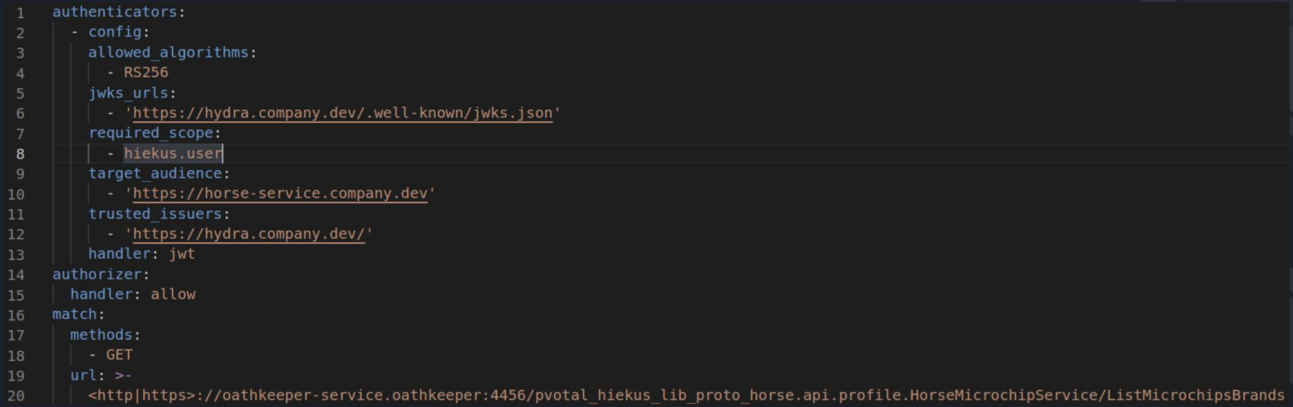




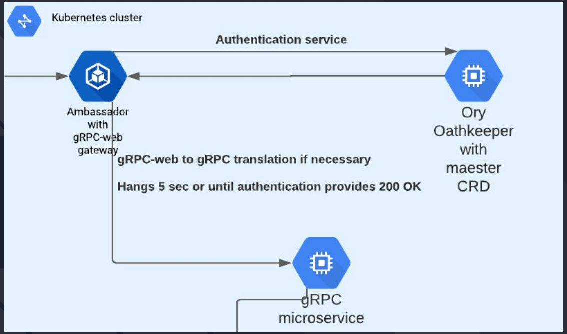
Kubernetes cluster
Authentication service
Ambassador with gRPC-web gateway
Ory Oathkeeper with maester CRD
gRPC-web to gRPC translation if necessary
Hangs 5 sec or until authentication provides 200 OK
gRPC microservice
gRPC Interceptor verification (to validate ownership or shared access)
Internal authorization service
Keto v2 engine

Oathkeeper validation

- Authentication service (JWT) per endpoint with Oathkeeper



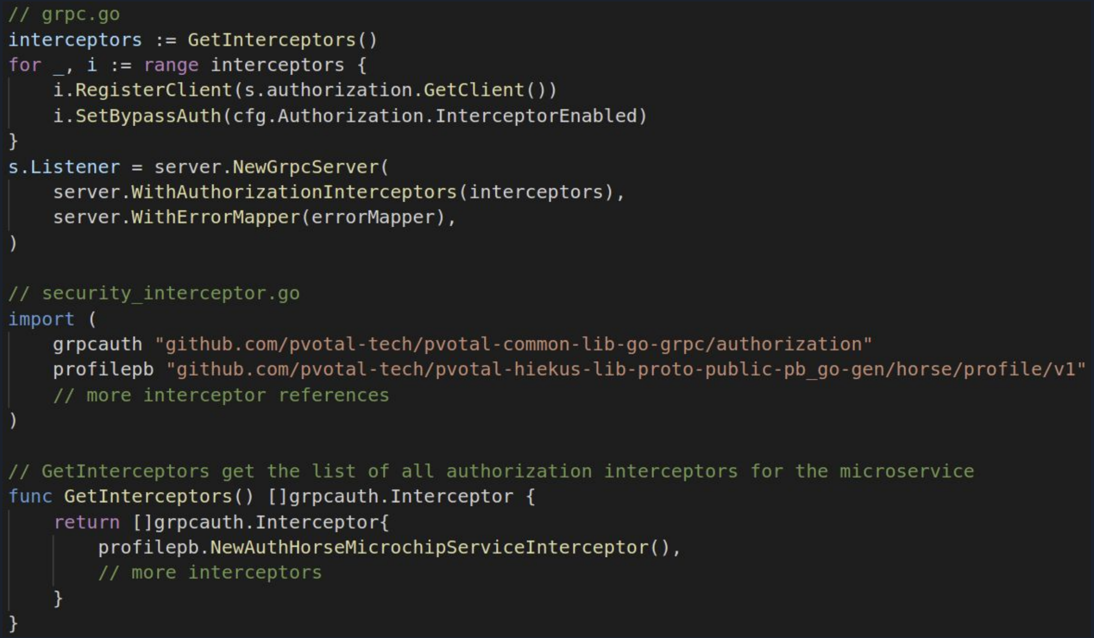
```yaml
authenticators:
  - config:
    allowed_algorithms:
      - RS256
    jwks_urls:
      - 'https://hydra.company.dev/.well-known/jwks.json'
    required_scope:
      - hiekus.user
    target_audience:
      - 'https://horse-service.company.dev'
    trusted_issuers:
      - 'https://hydra.company.dev/'
    handler: jwt
authorizer:
  handler: allow
match:
  methods:
    - GET
  url: >-
    <http|https>://oathkeeper-service.oathkeeper:4456/pvotal_hiekus_lib_proto_horse.api.profile.HorseMicrochipService/ListMicrochipsBrands
```



Kubernetes cluster
Authentication service
Ambassador with gRPC-web gateway
Ory Oathkeeper with maester CRD
gRPC-web to gRPC translation if necessary
Hangs 5 sec or until authentication provides 200 OK
gRPC microservice

gRPC interceptor validation

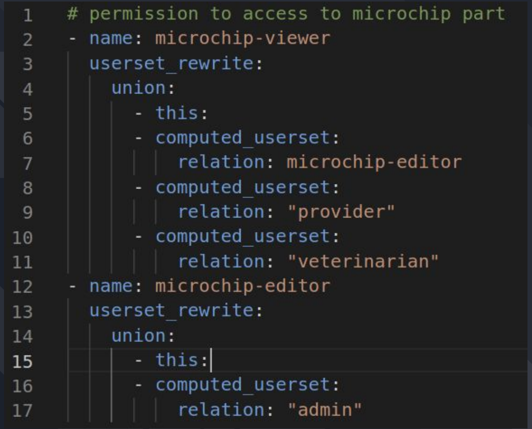
- Authorization with Zanzibar rules extension



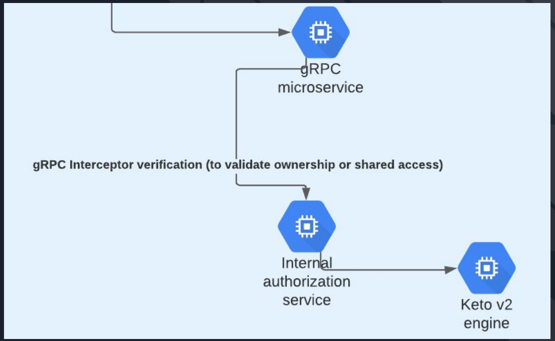
```go
// grpc.go
interceptors := GetInterceptors()
for _, i := range interceptors {
    i.RegisterClient(s.authorization.GetClient())
    i.SetBypassAuth(cfg.Authorization.InterceptorEnabled)
}
s.Listener = server.NewGrpcServer(
    server.WithAuthorizationInterceptors(interceptors),
    server.WithErrorMapper(errorMapper),
)

// security_interceptor.go
import (
    grpcauth "github.com/pvotal-tech/pvotal-common-lib-go-grpc/authorization"
    profilepb "github.com/pvotal-tech/pvotal-hiekus-lib-proto-public-pb_go-gen/horse/profile/v1"
    // more interceptor references
)

// GetInterceptors get the list of all authorization interceptors for the microservice
func GetInterceptors() []grpcauth.Interceptor {
    return []grpcauth.Interceptor{
        profilepb.NewAuthHorseMicrochipServiceInterceptor(),
        // more interceptors
    }
}
```



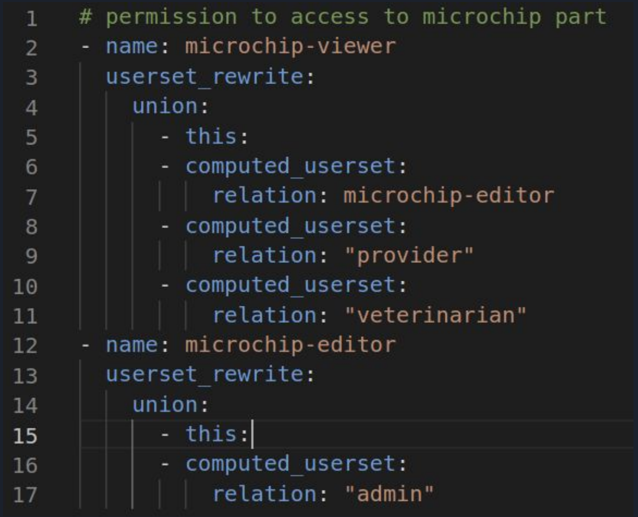
```yaml
# permission to access to microchip part
- name: microchip-viewer
  userset_rewrite:
    union:
      - this:
      - computed_userset:
          relation: microchip-editor
      - computed_userset:
          relation: "provider"
      - computed_userset:
          relation: "veterinarian"
- name: microchip-editor
  userset_rewrite:
    union:
      - this:
      - computed_userset:
          relation: "admin"
```


gRPC microservice
gRPC Interceptor verification (to validate ownership or shared access)
Internal authorization service
Keto v2 engine

Authorizer additions from Zanzibar

What is userset rewrite rule?
The userset rewrite rule is the boolean function that is applied when a check is performed. Given a check tuple, the userset rewrite produces all the usersets that need to be checked and how the result has to be combined. Moreover, the checks are done recursively, meaning that additional checks may have several userset rewrite rules as well. As Zanzibar defines it, the userset rewrite comes in several kinds:

- _this (Implemented and used)
- computed_userset (Implemented and used)
- tuple_to_userset
- union (Implemented and used)
- intersection
- exclusion



```
# permission to access to microchip part
- name: microchip-viewer
  userset_rewrite:
    union:
      - this:
      - computed_userset:
          relation: microchip-editor
      - computed_userset:
          relation: "provider"
      - computed_userset:
          relation: "veterinarian"
- name: microchip-editor
  userset_rewrite:
    union:
      - this:
      - computed_userset:
          relation: "admin"
```

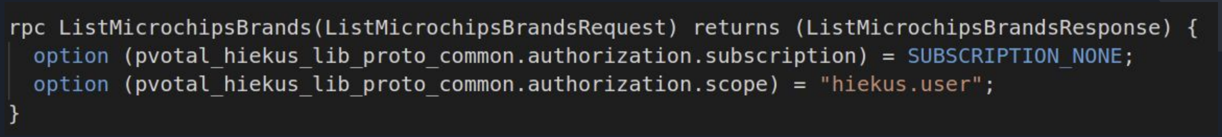
What are we trying to achieve ?

Objective : Share and allow people of the equine domain first to interact online to make most use of their data, empowering more client operations in particular offline in the future.

Diverse roles : Barn manager, Barn employee, Horse trainer, Farrier, Groomer, Veterinarian

Privileges on rpc operations for example :

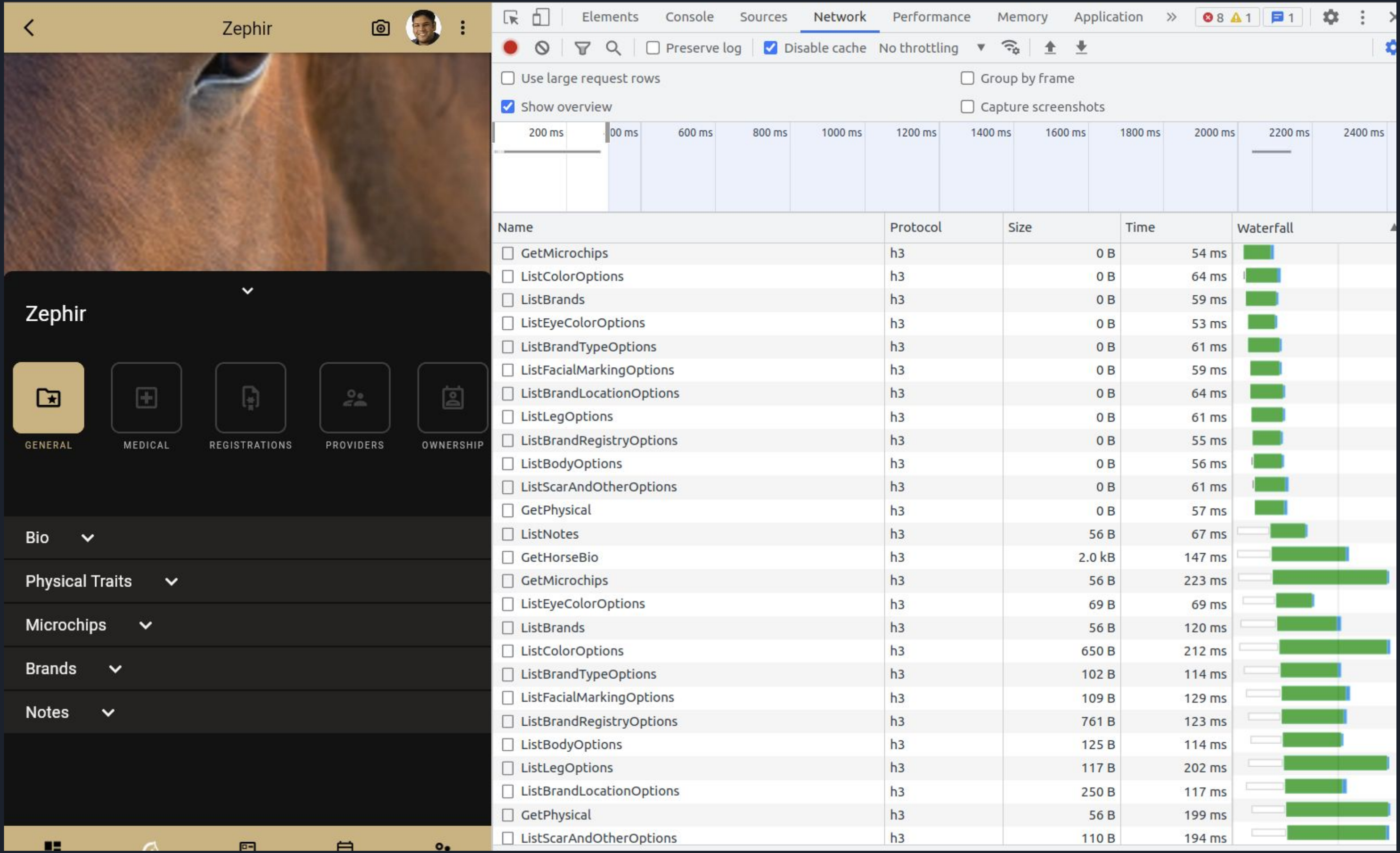
- general-viewer : the user that can view general information about the horse
- medical-viewer : the user can view the medical information about the horse
- owner-viewer: the owner who can only read information about the horse
- calendar-viewer : has medical permissions to view the horse medical calendar events



```
rpc ListMicrochipsBrands(ListMicrochipsBrandsRequest) returns (ListMicrochipsBrandsResponse) {
  option (pvotal_hiekus_lib_proto_common.authorization.subscription) = SUBSCRIPTION_NONE;
  option (pvotal_hiekus_lib_proto_common.authorization.scope) = "hiekus.user";
}
```

We want to minimize the number of rules and support privilege modification using the Zanzibar rule system. Leverage as much templating from our protobufs custom options for all security deployments configurations and zanzibar management.
We are currently addressing the front-end to match the shared permissions visibility (edit buttons, comments/notes edit button visibility, etc..)

Browser network analysis



Name	Protocol	Size	Time	Waterfall
GetMicrochips	h3	0 B	54 ms	
ListColorOptions	h3	0 B	64 ms	
ListBrands	h3	0 B	59 ms	
ListEyeColorOptions	h3	0 B	53 ms	
ListBrandTypeOptions	h3	0 B	61 ms	
ListFacialMarkingOptions	h3	0 B	59 ms	
ListBrandLocationOptions	h3	0 B	64 ms	
ListLegOptions	h3	0 B	61 ms	
ListBrandRegistryOptions	h3	0 B	55 ms	
ListBodyOptions	h3	0 B	56 ms	
ListScarAndOtherOptions	h3	0 B	61 ms	
GetPhysical	h3	0 B	57 ms	
ListNotes	h3	56 B	67 ms	
GetHorseBio	h3	2.0 kB	147 ms	
GetMicrochips	h3	56 B	223 ms	
ListEyeColorOptions	h3	69 B	69 ms	
ListBrands	h3	56 B	120 ms	
ListColorOptions	h3	650 B	212 ms	
ListBrandTypeOptions	h3	102 B	114 ms	
ListFacialMarkingOptions	h3	109 B	129 ms	
ListBrandRegistryOptions	h3	761 B	123 ms	
ListBodyOptions	h3	125 B	114 ms	
ListLegOptions	h3	117 B	202 ms	
ListBrandLocationOptions	h3	250 B	117 ms	
GetPhysical	h3	56 B	199 ms	
ListScarAndOtherOptions	h3	110 B	194 ms	

Key points

- Complex authorization rules require graph capabilities for validation on check
- Mimicking Zanzibar rules who have proven to be resilient to most use cases at Google

Questions ?