Practices of Go Microservices on Post-Kubernetes

Wei Zheng

Background in Shimo

石墨文档

Language

- Go
- Node
- Rust

Background in Shimo

石墨文档

Framework

- Gin
- Echo
- gRPC
- …

Background in Shimo

石墨文档

Platform

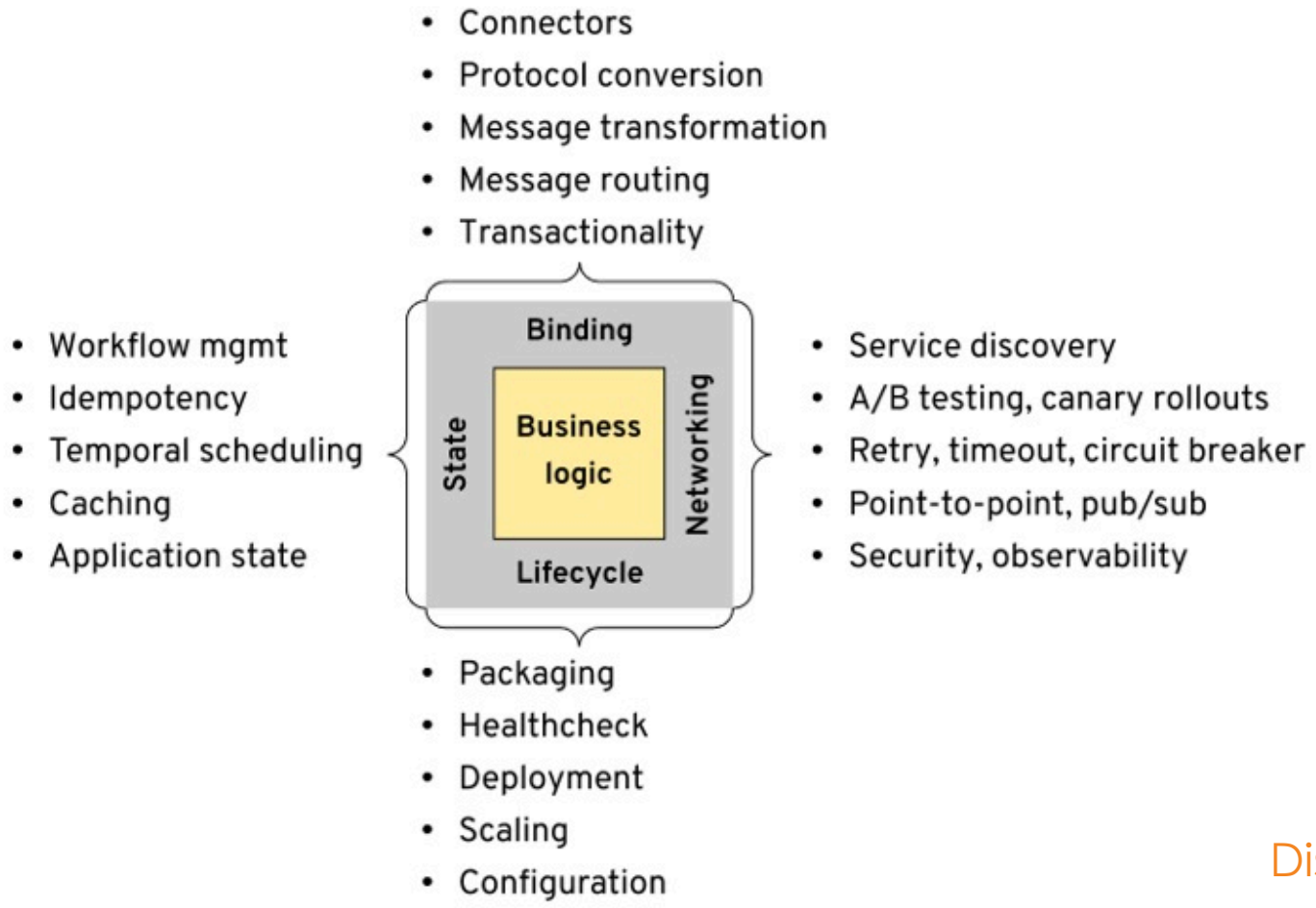
- All in Kubernetes
  - Ingress Controller
  - Stateless Deployment
  - Stateful Sets
  - …
- All in Cloud
  - ECS
  - RDS
  - OSS
  - SLB
  - …

Background in Shimo

石墨文档

- High performance & Stable
- Ecosystem
- Maintainable
- Pluggable
- ...

Background in Shimo



Distributed Application Needs
—Bilgin Ibryam

What platform or framework should we choose?

“

Ockham's Razor

Entities should not be multiplied without necessity.

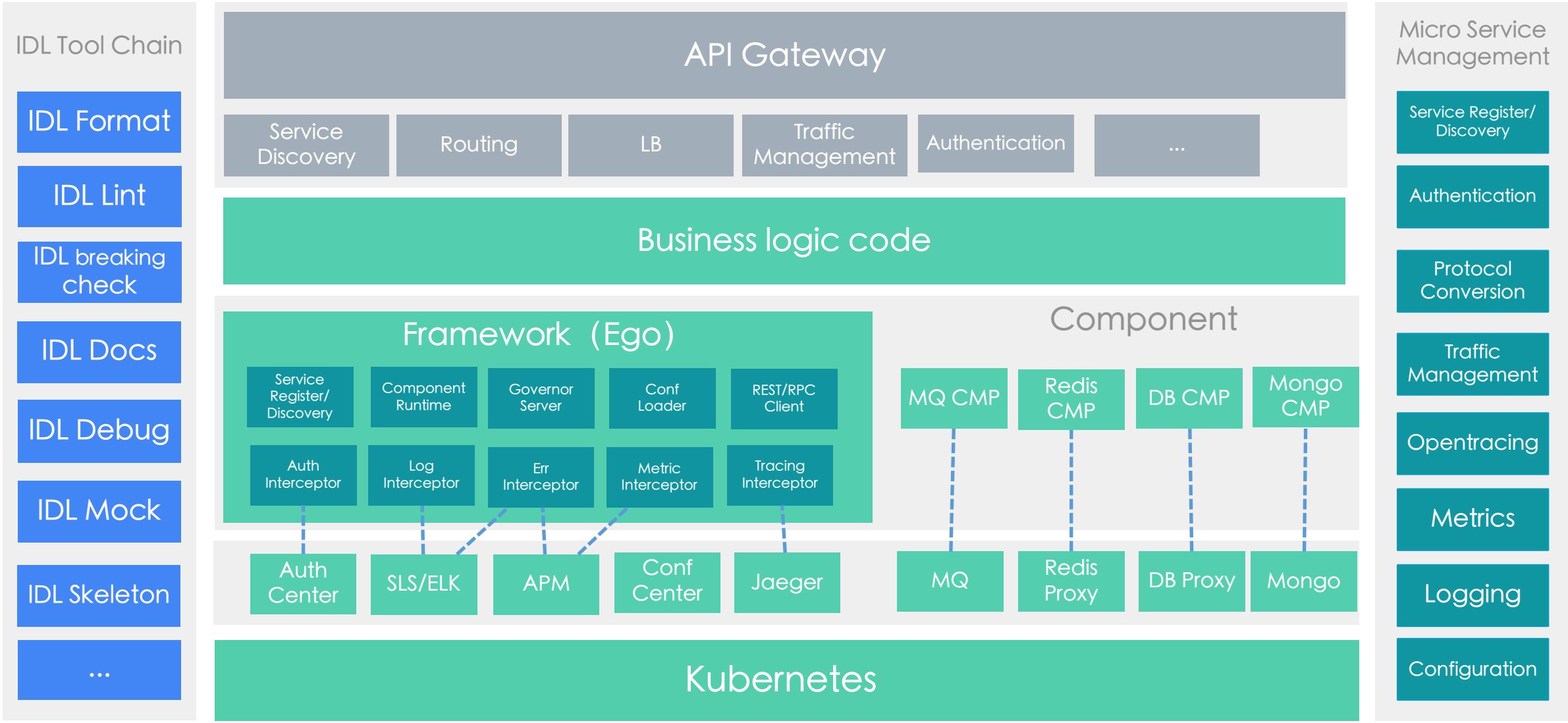
Background in Shimo

石墨文档

- gPRC
  - High performance
  - Multi Language support
  - Pluggable and ecosystem

- Service Mesh（Istio）
  - Cross-Language, out-of-box features, need no application change
  - Additional maintenance cost, not very stable
  - Only implement networking runtime

Architecture



How should we mange our IDLs?

“

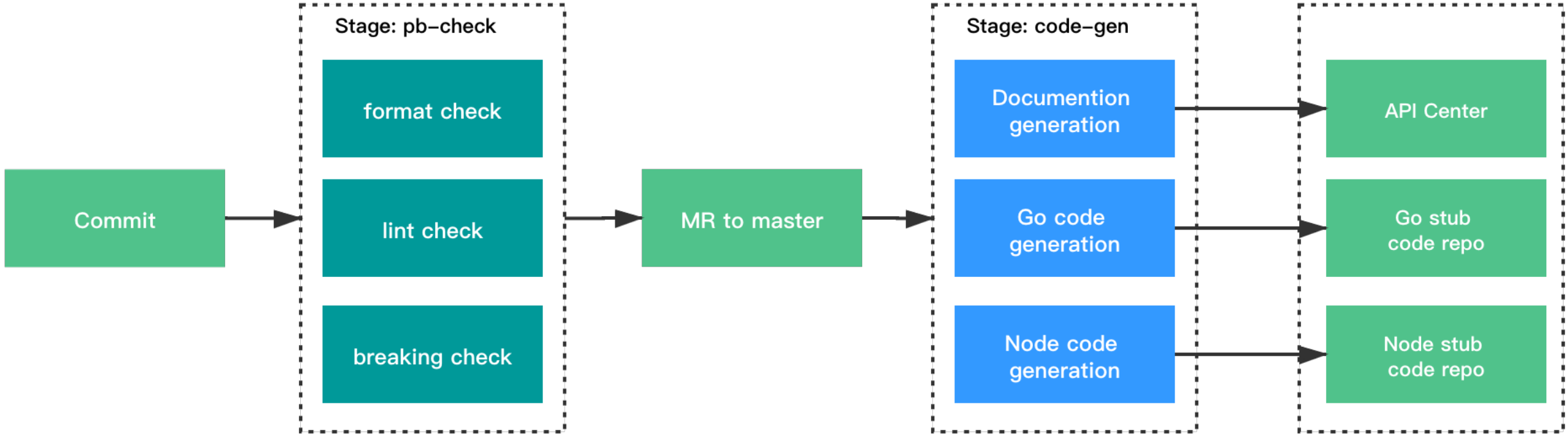
The Increase of Entropy Theorem

The entropy of an isolated system can increase, but not decrease.

IDL Tool Chain - Style Guide

- Protobuf mono repo
- Git/CI workflow
- Custom cli tool
- IDL style guide
    - Google Style
    - Uber V2 Style

IDL Tool Chain - CI Workflow



IDL Tool Chain - Cli

- egoctl: (https://github.com/gotomicro/egoctl)
  - Builtin Uber prototool
    - Lint、format and check breaking change your Protobuf
    - Fix comments lint rule for Chinese
  - Custom IDL
    - Generate front-end code with React and Ant Design
    - Generate back-end code with ego

IDL Tool Chain - Style Guide

Rule	Description
Specing	Use two spaces instead of tabs.
Package Naming	- A package name is a full package name, i.e. uber.trip.v1. - A package sub-name is a part of a package name, ie uber, trip, or v1. - A package version is the last package sub-name that specifies the version, i.e. v1, v1beta.
Package Version	The last package sub-name should be a major version of the package, or the major version followed by the beta version.
Directory Structure	Files should be stored in a directory structure that matches their package sub-names.
Messages	Messages should always be PascalCase.
Message Fields	Message field names should always be lower_snake_case.
Services/RPCs	Services should always be PascalCase. RPCs should always be PascalCase.
Documention	All comments should use // and not /* */.
...	

IDL Tool Chain - gRPC Debugging

- How to debugging service with dependencies?
  - Use VPN to connect develop kubernetes cluster.
  - With Kubernetes API server Resolver, we can resolve dependent services.

IDL Tool Chain - gRPC Local Debugging

石墨文档

```
→~ cat ~/.kube/config

apiVersion: v1
clusters:
  - cluster:
      certificate-authority-data: CERTIFICATE_TOKEN
      server: https://K8S_SERVICE_HOST:K8S_SERVICE_PORT
    name: dev
```

```
[grpc.otherService]
    addr = "localhost:50051"
```

```
#!/bin/bash

# Usage: ./pf.sh default svc-user
function pf() {
  podName=$(kubectl get pod -n $1 | grep $2 | head -n 1 | awk -F " " '{print $1}')
  echo podName:$podName
  kubectl port-forward $podName -n $1 $3 &
  sleep 2s
}

ps -efw | grep port-forward | grep -v grep | awk '{print $2}' | xargs kill

kubectx dev
port="50051:50051"
pf $1 $2 $port
```

How do we connect service in k8s cluster before :(

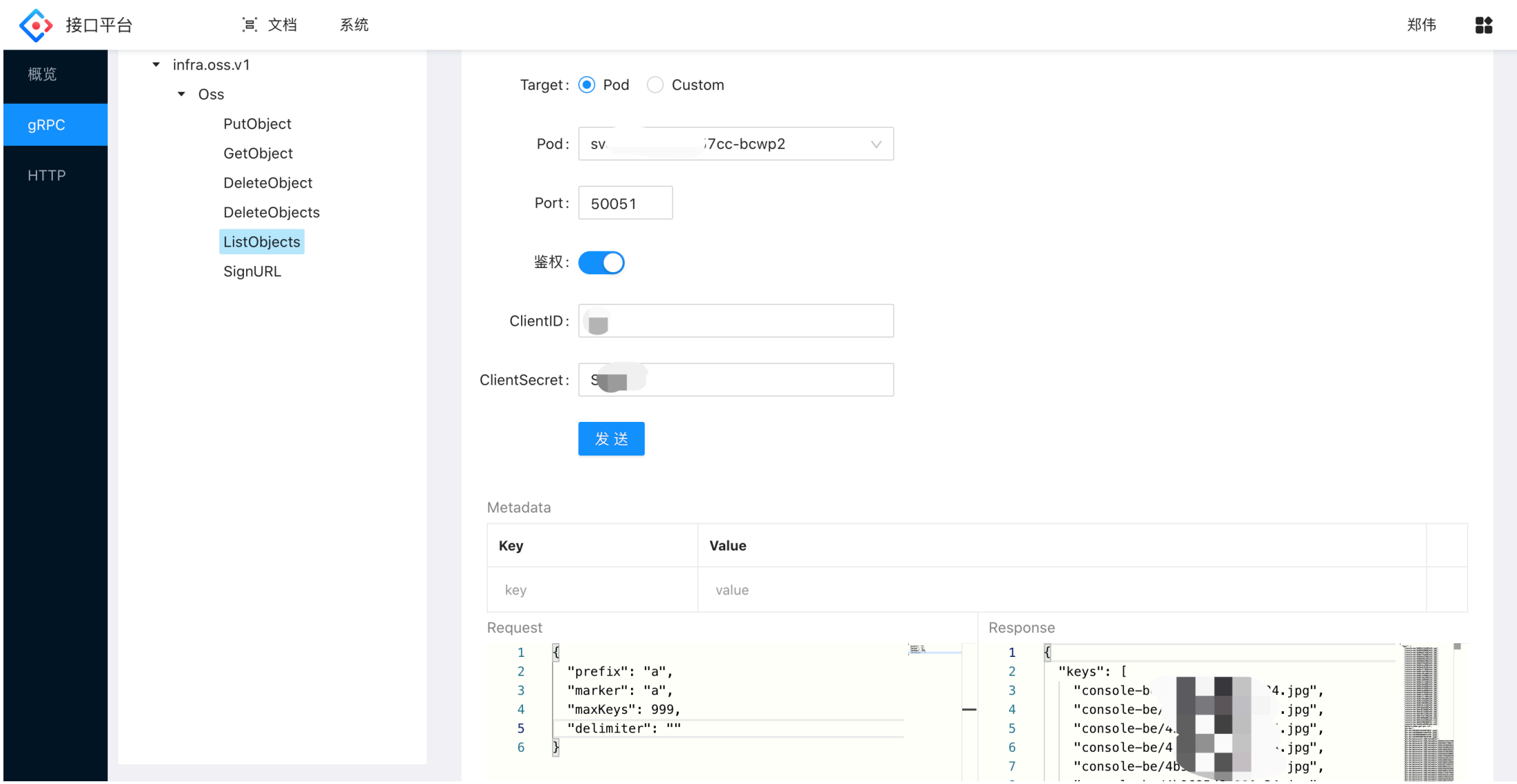
GO CN

IDL Tool Chain - gRPC Local Debugging

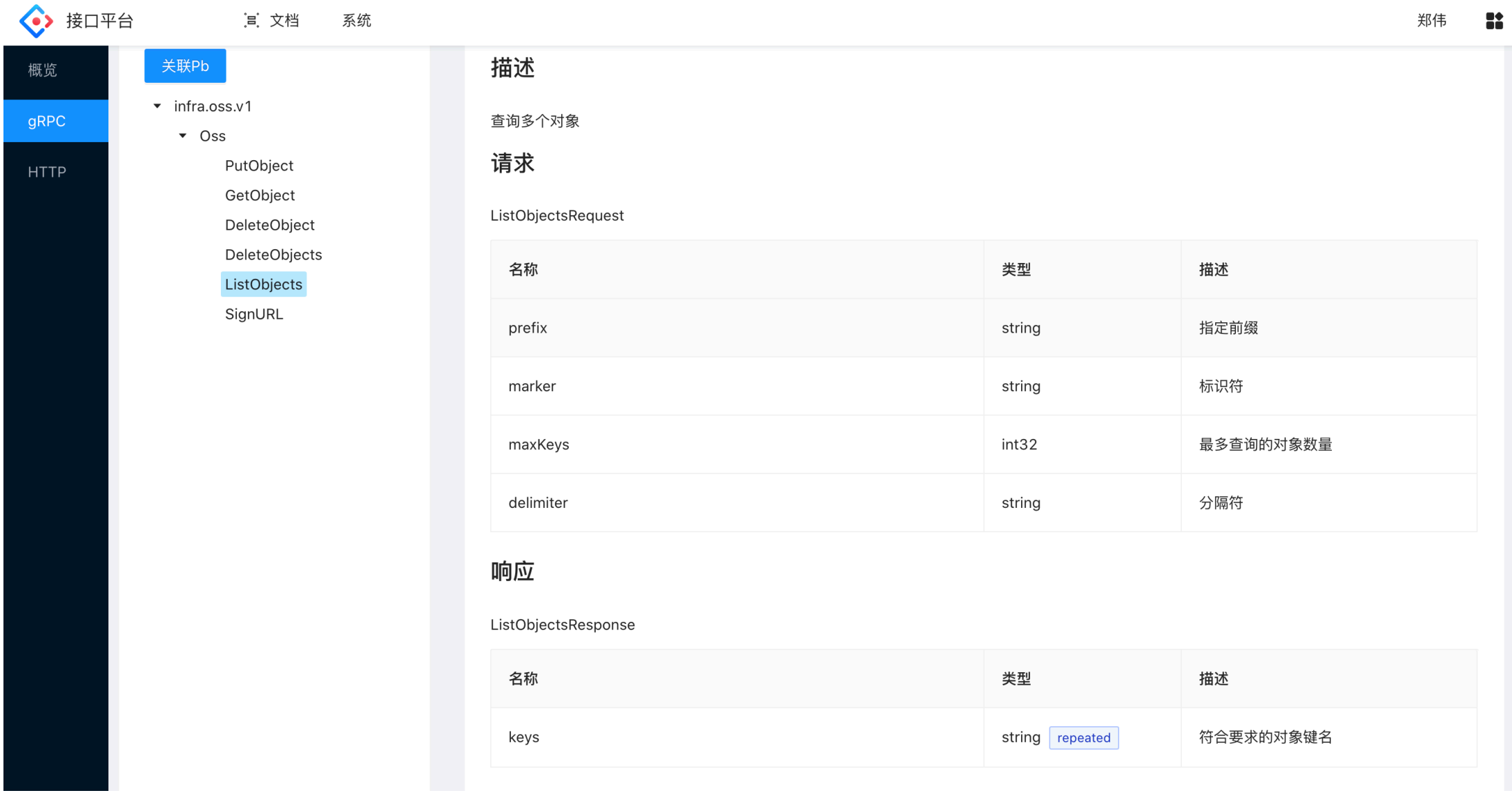

```
[grpc.otherService]
    addr = "k8s:///other-service:50051"
```

Now we connect service in k8s cluster like this :)

IDL Tool Chain - gRPC Remote Debugging



IDL Tool Chain - gRPC Docs



How to build our own framework?

Framework - Service Register/Discovery

- Based on Kubernetes
  - Server register its replicates to Kubernetes
  - Client discovery server based on Kubernetes API Server resolver

Framework - Service Register/Discovery

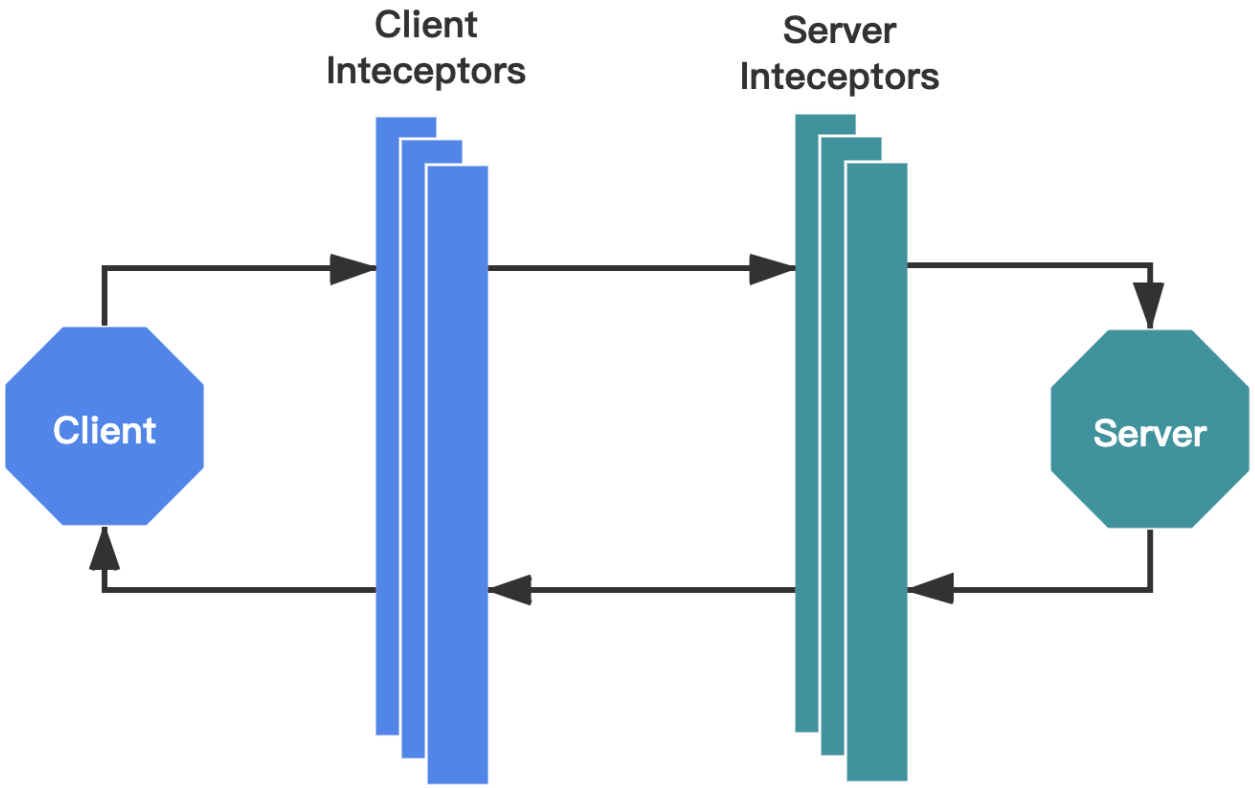
- Kubernetes DNS Resolver VS Kubernetes API Server Resolver
  - DNS resolver is builtin in gRPC framework and its out-of-box for users
  - When connection fail DNS Resolver can resolve name immediately
  - In scale-up scenario, DNS Resolver will not resolve name automatically

Framework - Service Register/Discovery


```
…
informer.Informer().AddEventHandler(cache.ResourceE
ventHandlerFuncs{
        AddFunc:    c.addEndpoints,
        UpdateFunc: c.updateEndpoints,
        DeleteFunc: c.deleteEndpoints,
})
…

[grpc.otherService]
    addr = "k8s:///other-service:50051"
```

K8s API Server Resolver

Framework - Builtin Interceptors



Framework - Builtin Interceptors

Builtin Interceptors

- Authentication Interceptor
- Logging Interceptor
- Tracing Interceptor
- Metrics Interceptor
- Debug Interceptor
- Errors Interceptor
- ...

Framework - Builtin Interceptors


```
2021-04-09 22:04:34     INFO    ego/ego_function.go:204 init default logger                                    {"comp": "core.elog"}
2021-04-09 22:04:34     INFO    ego/ego_function.go:205 init Ego logger                                        {"comp": "core.elog"}
2021-04-09 22:04:34     INFO    ego/ego_function.go:158 init config                                            {"comp": "core.econf", "addr": "config.toml"}
2021-04-09 22:04:34     INFO    file/file.go:77 read watch                                {"comp": "core.econf", "comp": "file datasource", "configFile": "/Users/z
amples/grpc/direct/client/config.toml", "realConfigFile": "/Users/zheng/shimo/ego/examples/grpc/direct/client/config.toml", "dir": "/Users/zheng/shimo/ego/exampl
ient", "fppath": "/Users/zheng/shimo/ego/examples/grpc/direct/client/config.toml"}
2021-04-09 22:04:34     INFO    ego/ego_function.go:199 init max procs                                         {"comp": "app", "value": 8}
2021-04-09 22:04:34     INFO    ego/ego_function.go:183 init trace                                             {"comp": "app"}
2021/04/09 22:04:34 grpc.response grpc.test 127.0.0.1:9002 [0.764ms] /helloworld.Greeter/SayHello | name:"i am client"  => message:"Hello EGO, I'm 0.0.0.0:9002"

2021/04/09 22:04:34 grpc.response grpc.test 127.0.0.1:9002 [0.337ms] /helloworld.Greeter/SayHello | name:"error"  => rpc error: code = Unavailable desc = error
```

```
2021/04/09 22:10:14 [eredis.response] redis [] [2.293ms] [ping] => PONG

2021/04/09 22:10:14 [eredis.response] redis [] [0.229ms] [set name lilei] => OK

2021/04/09 22:10:14 [eredis.response] redis [] [0.161ms] [get name] => lilei
```

Debug Interceptor

Framework - Error Handling

Error Handling

- gRPC Code
- Biz Code
- Biz Message

Framework - Error Handling

Client/Server	gRPC Code	Value	HTTP Code	Description
	OK	0	200	returned on success.
Client	INVALID_ARGUMENT	3	400	The client specified an invalid argument.
	NOT_FOUND	5	404	Some requested entity was not found.
	UNAUTHENTICATED	16	401	The request does not have valid authentication credentials for the operation.
	...			
Server	UNKNOWN	2	500	Unknown error.
	UNIMPLEMENTED	12	501	The operation is not implemented or is not supported/enabled in this service.
	UNAVAILABLE	14	503	The service is currently unavailable.
	...			

Framework - Error Handling

Project/Group	Service	Error Code
100-999	000-999	000-999

Framework - Error Handling


```go
import (
    gstatus "google.golang.org/genproto/googleapis/rpc/status"
    "google.golang.org/grpc/codes"
    "google.golang.org/grpc/status"
)

func NewError(grpcCode codes.Code, bizCode int32, bizMsg string) error {
    st, err := status.New(grpcCode, bizMsg).WithDetails(
        &gstatus.Status{Code: int32(bizCode), Message: bizMsg},
    )
    if err != nil {
        return err
    }

    return st.Err()
}
```

Encoding An Error

Framework - Error Handling

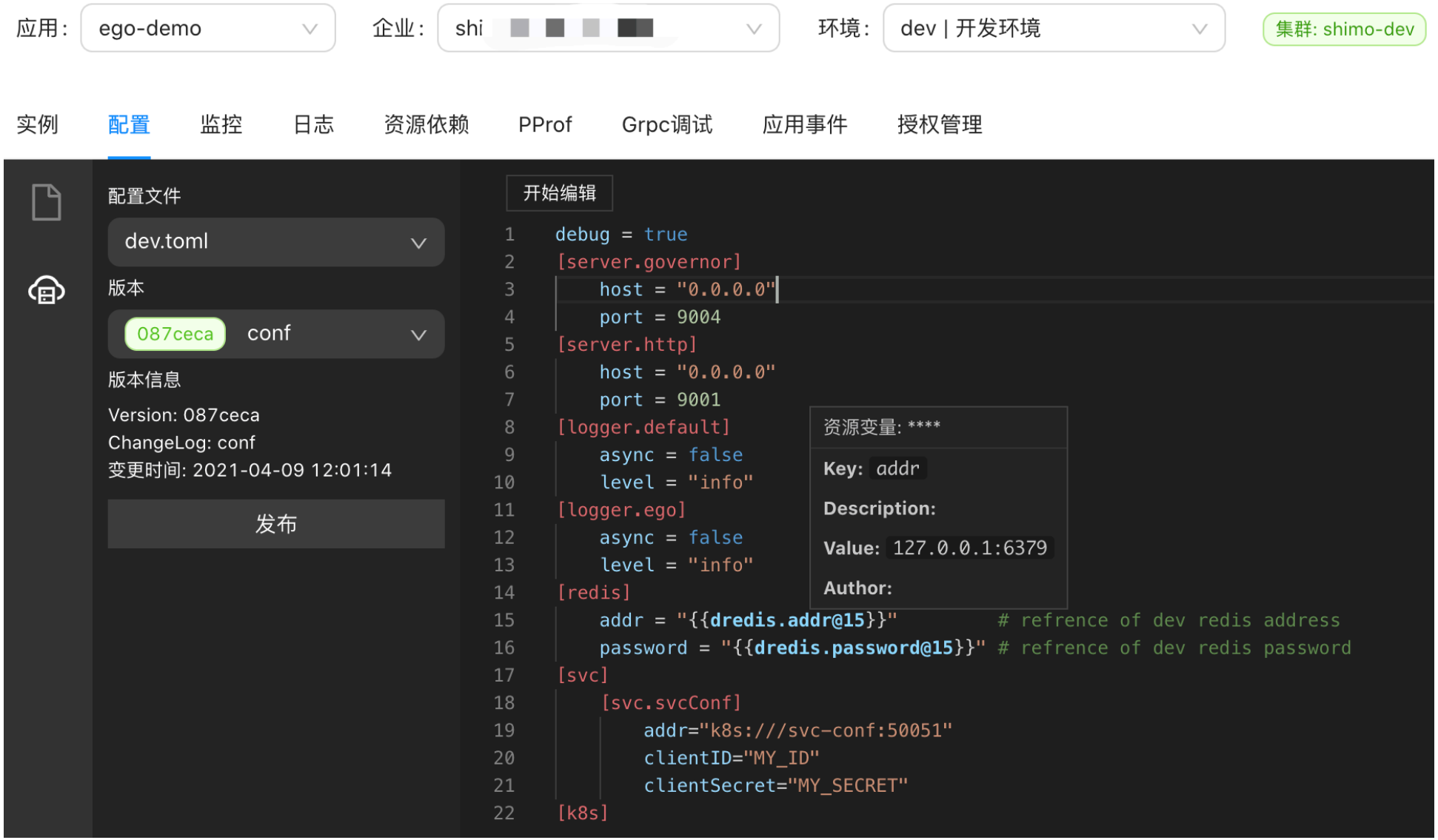

```go
import (
    gstatus "google.golang.org/genproto/googleapis/rpc/status"
    "google.golang.org/grpc/codes"
    "google.golang.org/grpc/status"
)

func ParseError(err error) (grpcCode codes.Code, bizCode int, bizMsg string, e error) {
    st := status.Convert(err)
    for _, detail := range st.Details() {
        switch t := detail.(type) {
        case *gstatus.Status:
            return st.Code(), int(t.Code), t.Message, nil
        }
    }
    e = errors.New("extract error from status fail")
    return
}
```

Decoding An Error

Framework - Configuration

- Toml、JSON、Yaml Supported
- Dynamic change configuration on the fly
- Human friendly UI for configuration edit
- Integration with Kubernetes ConfigMap
- …

Framework - Configuration



Framework - Configuration

Integrated with ConfigMap

```
→  ~ kc describe cm dev-ego-demo
Name:         dev-ego-demo
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
__metadata.dev.toml:
----
{"version":"087ceca3d0caa93d0872548f4e86ef42","change_log":"conf","uid":485}
dev.toml:
----
debug = true
[server.governor]
    host = "0.0.0.0"
    port = 9004
[server.http]
    host = "0.0.0.0"
    port = 9001
[logger.default]
    async = false
    level = "info"
[logger.ego]
    async = false
    level = "info"
[redis]
    addr = "127.0.0.1:6379"    # refrence of dev redis address
    password = "YOUR_PASSWORD" # refrence of dev redis password
[svc]
    [svc.svcConf]
        addr="k8s:///svc-conf:50051"
        clientID="MY_ID"
        clientSecret="MY_SECRET"
[k8s]
Events:  <none>
```

Framework

Stability

- Compatible with semantic version
- Ensure unit test coverage
- Foresight in design
- Defensive and pluggable design for APIs

Framework

Iteration

- Manage iteration lifecycle with issues
- Do iteration with you framework users
- Open source for all codes and welcome any necessary PRs, reviews or suggestions

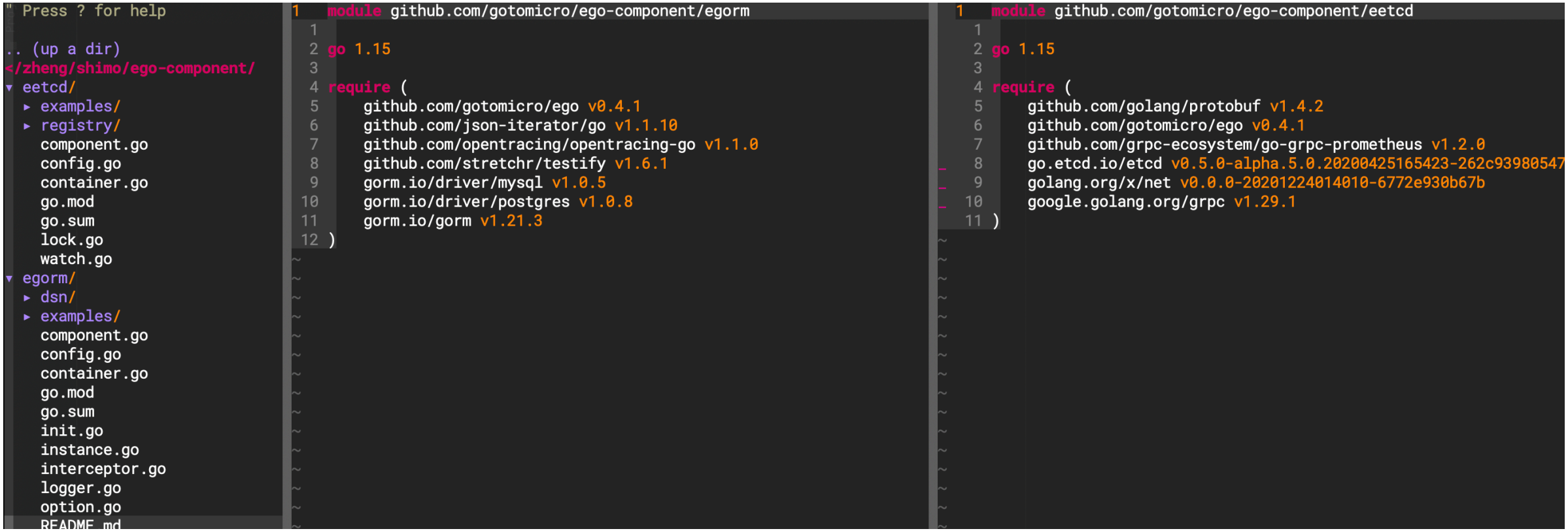
How do we design our components and libraries?

Components

Independent Module

- All components in mono repo
- All components has its own module

Components


" Press ? for help
.. (up a dir)
</zheng/shimo/ego-component/
▾ eetcd/
▸ examples/
▸ registry/
component.go
config.go
container.go
go.mod
go.sum
lock.go
watch.go
▾ egorm/
▸ dsn/
▸ examples/
component.go
config.go
container.go
go.mod
go.sum
init.go
instance.go
interceptor.go
logger.go
option.go
README.md
module github.com/gotomicro/ego-component/egorm
go 1.15
require (
github.com/gotomicro/ego v0.4.1
github.com/json-iterator/go v1.1.10
github.com/opentracing/opentracing-go v1.1.0
github.com/stretchr/testify v1.6.1
gorm.io/driver/mysql v1.0.5
gorm.io/driver/postgres v1.0.8
gorm.io/gorm v1.21.3
)
module github.com/gotomicro/ego-component/eetcd
go 1.15
require (
github.com/golang/protobuf v1.4.2
github.com/gotomicro/ego v0.4.1
github.com/grpc-ecosystem/go-grpc-prometheus v1.2.0
go.etcd.io/etcd v0.5.0-alpha.5.0.20200425165423-262c93980547
golang.org/x/net v0.0.0-20201224014010-6772e930b67b
google.golang.org/grpc v1.29.1
)

Components

Configuration

- Standard configuration schema
- Integrated with Kubernetes ConfigMap

Components

Lifecycle

- Control component lifecycle

Components

Debugging

- Dump Request/Response during debugging

Components

Tracing

- Add context to all APIs for logging tracing span during all operations

Components

Logging

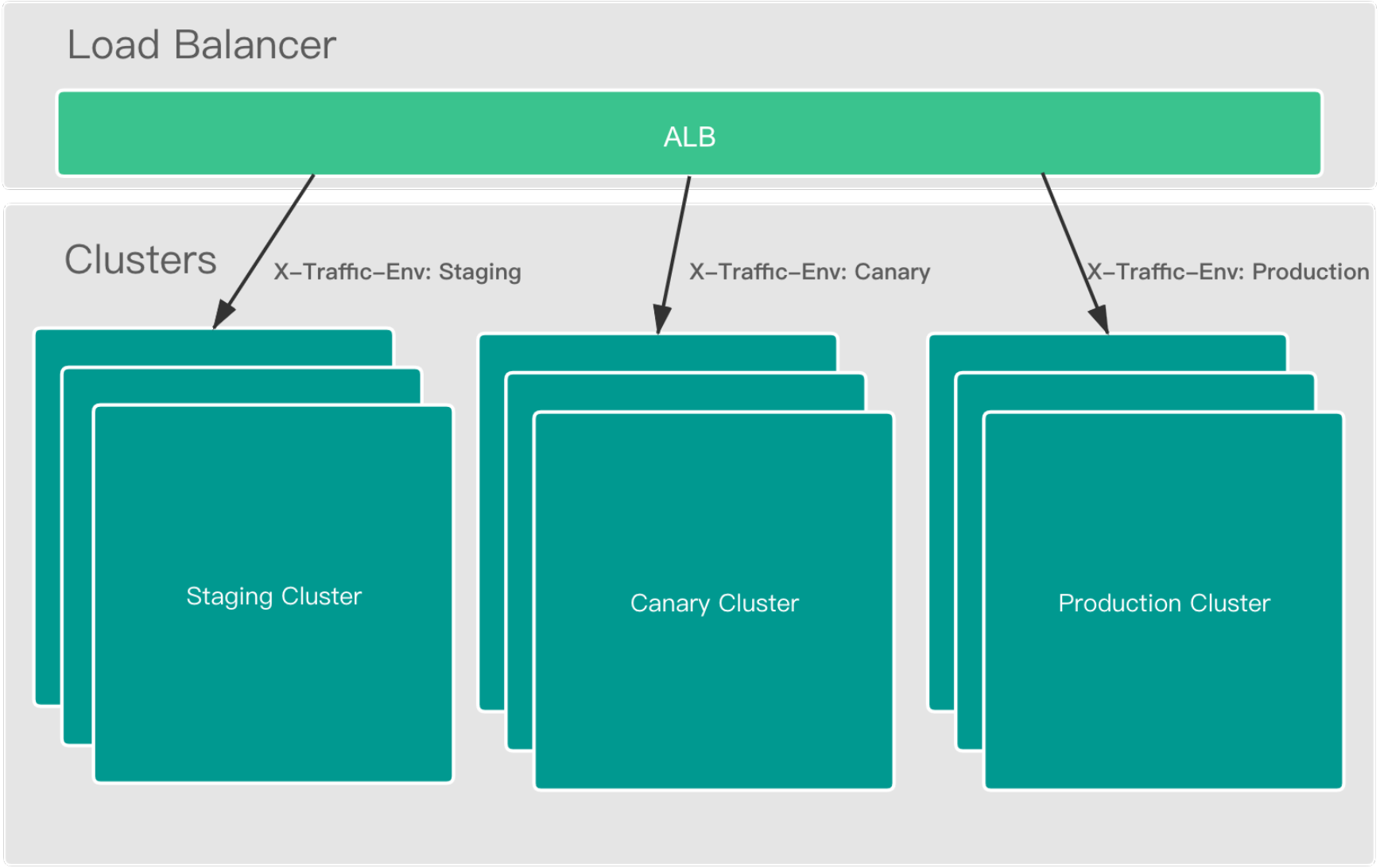
- Append standard logging when error occurs

How do we manage our microservices?

Traffic Management - Rate limiting/Circuit Breaker

- Ingress Controller
- Client/Server side interceptor

Traffic Management - Canary



Scaling

- Based on Kubernetes
  - User manage service resource configuration(CPU/MEM) by themself
  - User Scaling-up/Scaling-down replicate manually or via Kubernetes Autoscaler

Observability - Logging

- Based on Uber zap
  - Auto logging with TraceID、ServiceName、Framework Version…
  - Structured logging for friendly searching
  - Extract business logger from framework logger

Observability - Metrics

- Based on Kubernetes/Prometheus/Grafana
  - Builtin framework and Go runtime collectors，Export APIs for user to add their own metrics
  - Declare Prometheus annotations for service discovery and metrics scraping

Observability - Metrics

Kubernetes Prometheus Annotations

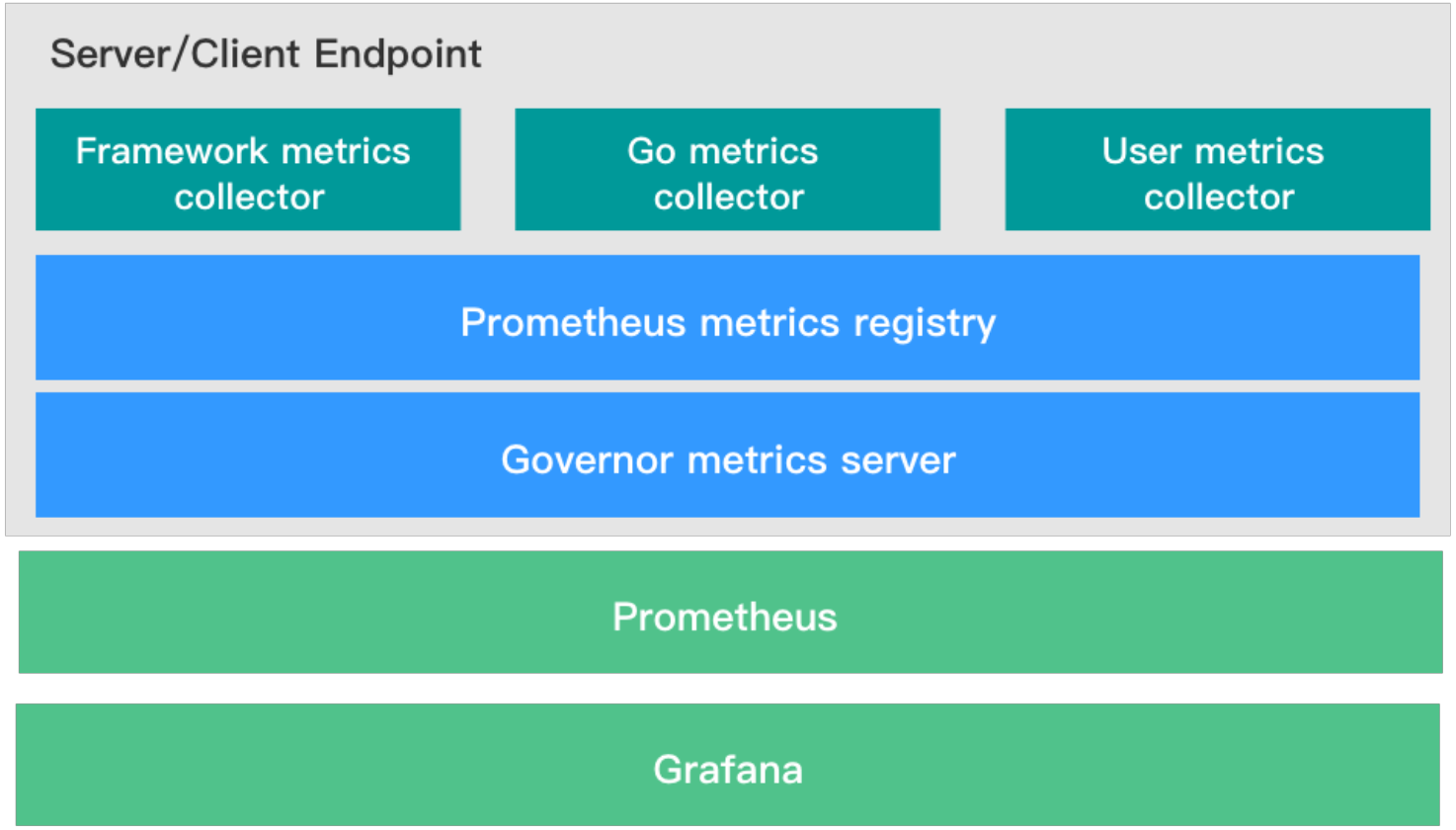
```
Pod Template:
  Labels:
    app=my-app
    enable=true
    version=pro
  Annotations:
    prometheus.io/port: 9003
    prometheus.io/scrape: true
```

Observability - Metrics

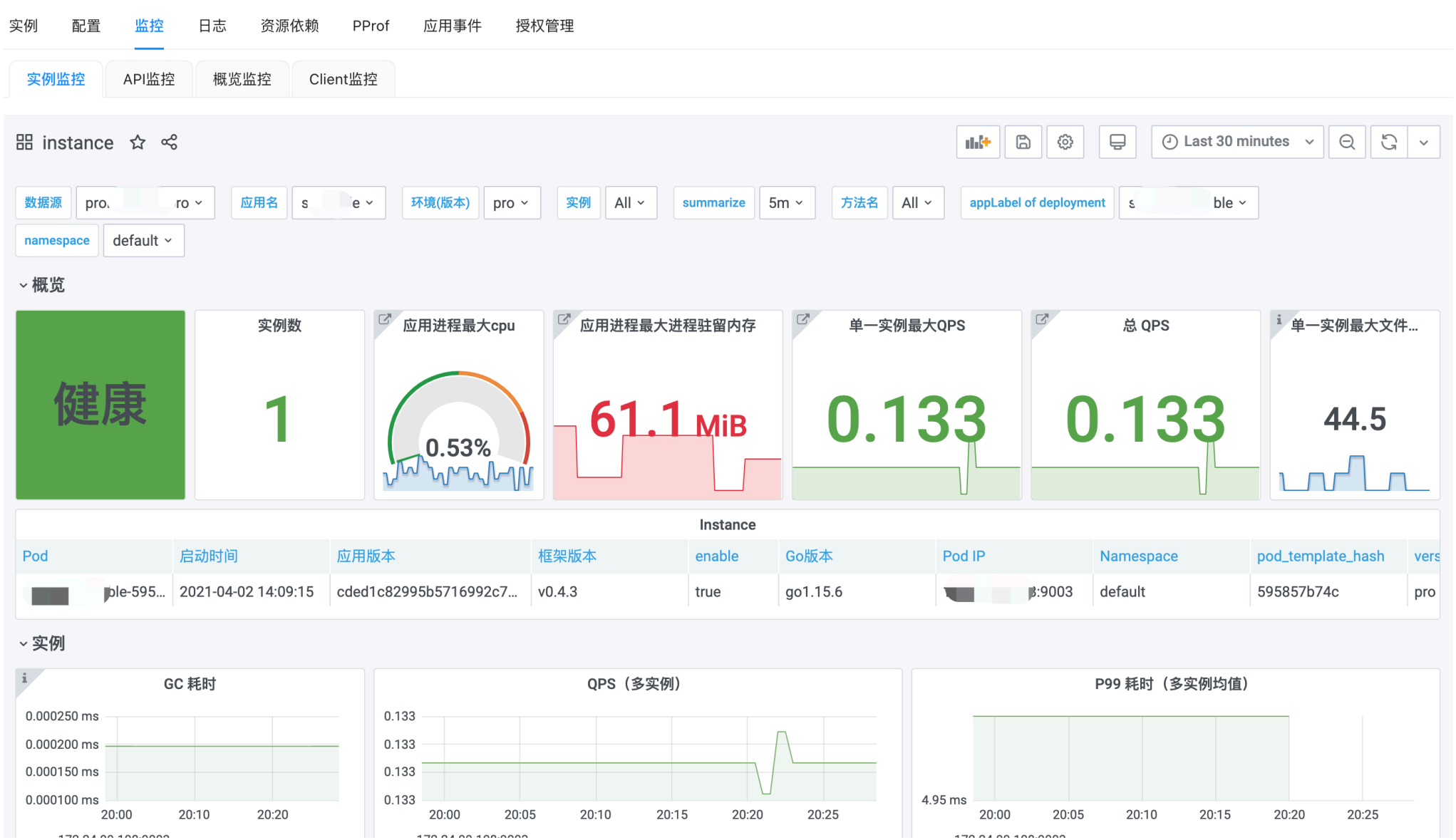
Naming

- Metric name should be snake_case
- User metric name should start with custom_
- Metric name should end with _total、_seconds、_bytes、_ratio、_info

Observability - Metrics



Observability - Metrics



Observability - Opentracing

- Adding context style APIs for all component:
  - HTTP Request
  - DB Query
  - Redis Query
  - Mongo Query
  - Kafka Command
- Following tracing span tags naming guide: db.type、db.instance、db.statement…
- Integrating trace-id when logging

Q & A

石墨文档

文档实时协同 · 知识沉淀管理 · 数据安全可控