



# Practices of Go Microservices on Post-Kubernetes

郑伟  
石墨文档

# Background in Shimo

## Language

- Go
- Node
- Rust

# Background in Shimo

## Framework

- Gin
- Echo
- gRPC
- ...

# Background in Shimo

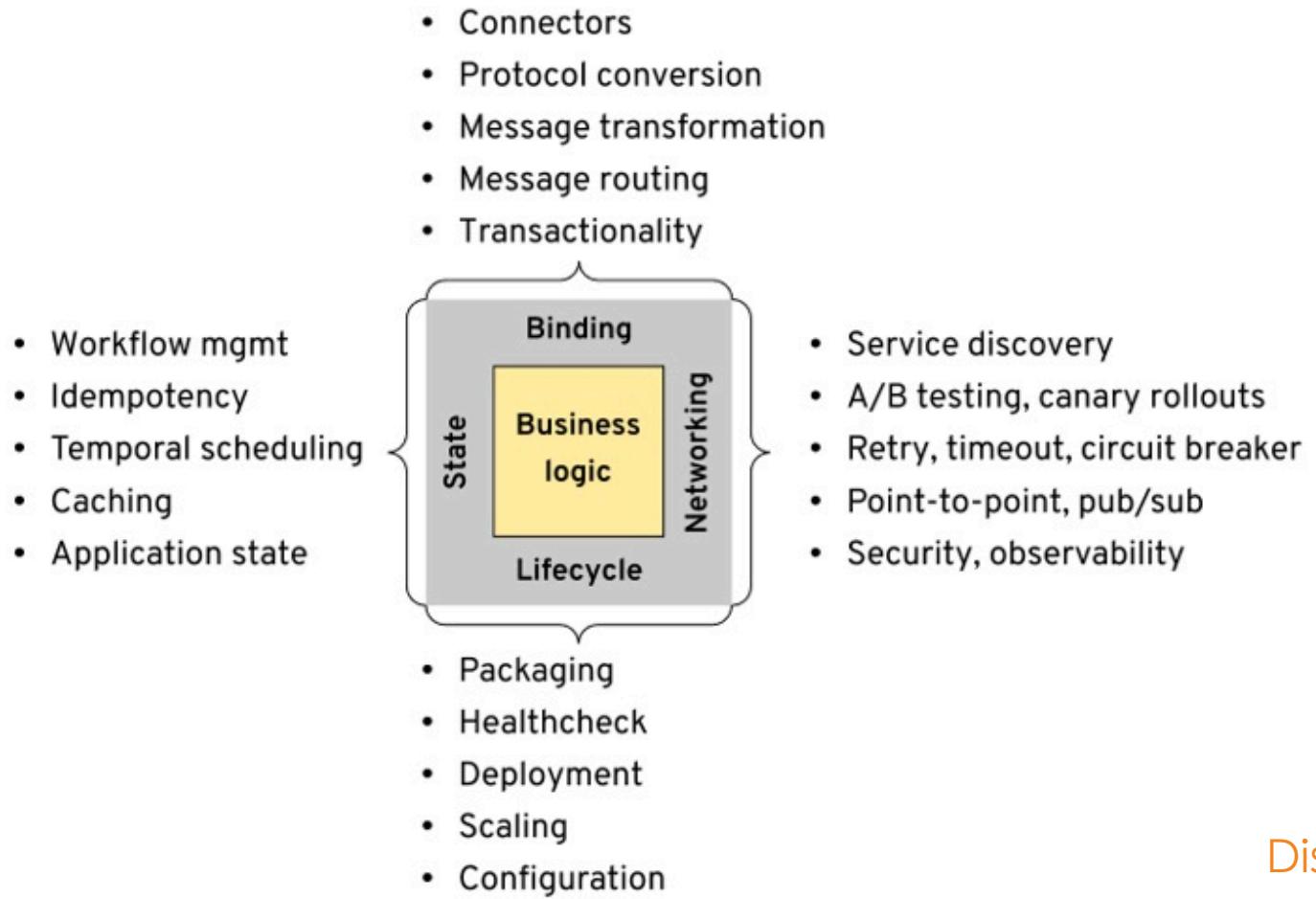
Platform

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

# Background in Shimo

- 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

- gRPC
  - 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

IDL Tool Chain

IDL Format

IDL Lint

IDL breaking check

IDL Docs

IDL Debug

IDL Mock

IDL Skeleton

...

## API Gateway



## Business logic code

### Framework (Ego)



### Component



MQ

Redis Proxy

DB Proxy

Mongo

## Kubernetes

Micro Service Management

Service Register/Discovery

Authentication

Protocol Conversion

Traffic Management

Opentracing

Metrics

Logging

Configuration



How should we mange our IDLs?

“

## The Increase of Entropy Theorem

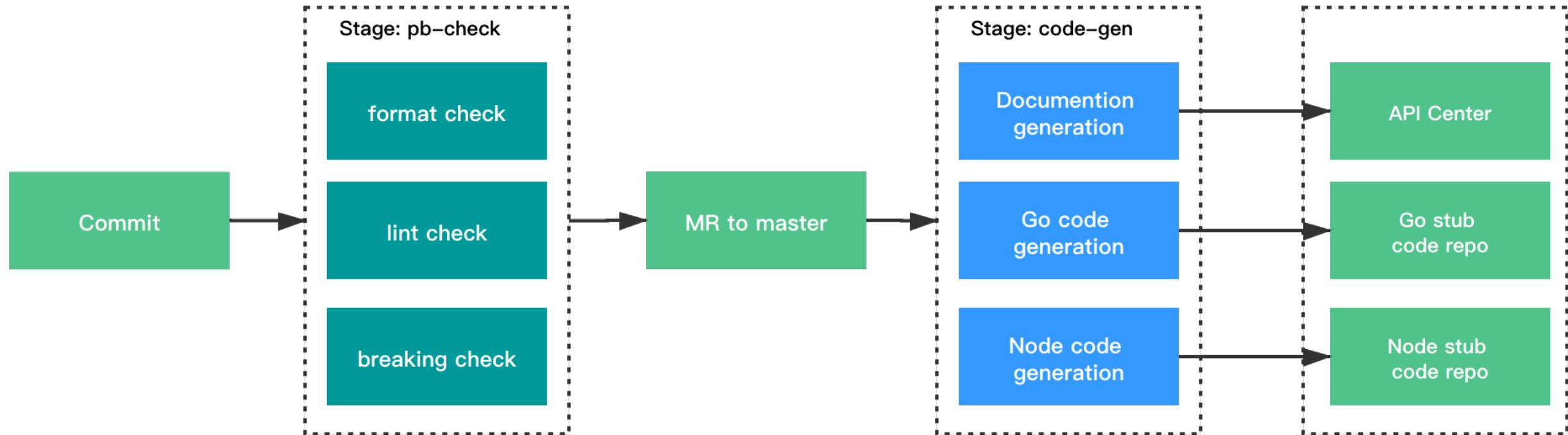


The entropy of an isolated system can increase, but not decrease.

# 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	<ul style="list-style-type: none"><li>- A package name is a full package name, i.e. uber.trip.v1.</li><li>- A package sub-name is a part of a package name, ie uber, trip, or v1.</li><li>- A package version is the last package sub-name that specifies the version, i.e. v1, v1beta.</li></ul>
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.
Documentation	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

```
→~ 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 :)

# IDL Tool Chain - gRPC Local Debugging



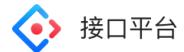
```
[grpc.otherService]  
addr = "k8s:///other-service:50051"
```

Now we connect service in k8s cluster like this :)

# IDL Tool Chain - gRPC Remote Debugging

The screenshot shows a cloud-based interface for managing Object Storage Service (OSS). The left sidebar includes links for Overview, gRPC, and HTTP. The main area displays the 'infra.oss.v1' API documentation under the 'Oss' section. A 'ListObjects' endpoint is currently selected. The configuration panel allows setting the target to 'Pod' or 'Custom', selecting a pod ('sv'), specifying a port ('50051'), enabling authentication ('鉴权'), and providing client credentials ('ClientID' and 'ClientSecret'). Below this is a 'Send' button and a 'Metadata' table with columns for 'Key' and 'Value'. The 'Request' pane shows a JSON object with fields like 'prefix', 'marker', 'maxKeys', and 'delimiter'. The 'Response' pane shows a JSON object with a 'keys' array containing file names such as '4.jpg', '.jpg', '4.jpg', '.jpg', and '4b.jpg'. The interface also features a dark mode theme.

# IDL Tool Chain - gRPC Docs



接口平台

文档

系统

郑伟



概览

gRPC

HTTP

关联Pb

- ▼ infra.oss.v1
  - ▼ Oss
    - PutObject
    - GetObject
    - DeleteObject
    - DeleteObjects
    - ListObjects
    - SignURL

## 描述

查询多个对象

## 请求

ListObjectsRequest

名称	类型	描述
prefix	string	指定前缀
marker	string	标识符
maxKeys	int32	最多查询的对象数量
delimiter	string	分隔符

## 响应

ListObjectsResponse

名称	类型	描述
keys	string <small>repeated</small>	符合要求的对象键名



# 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

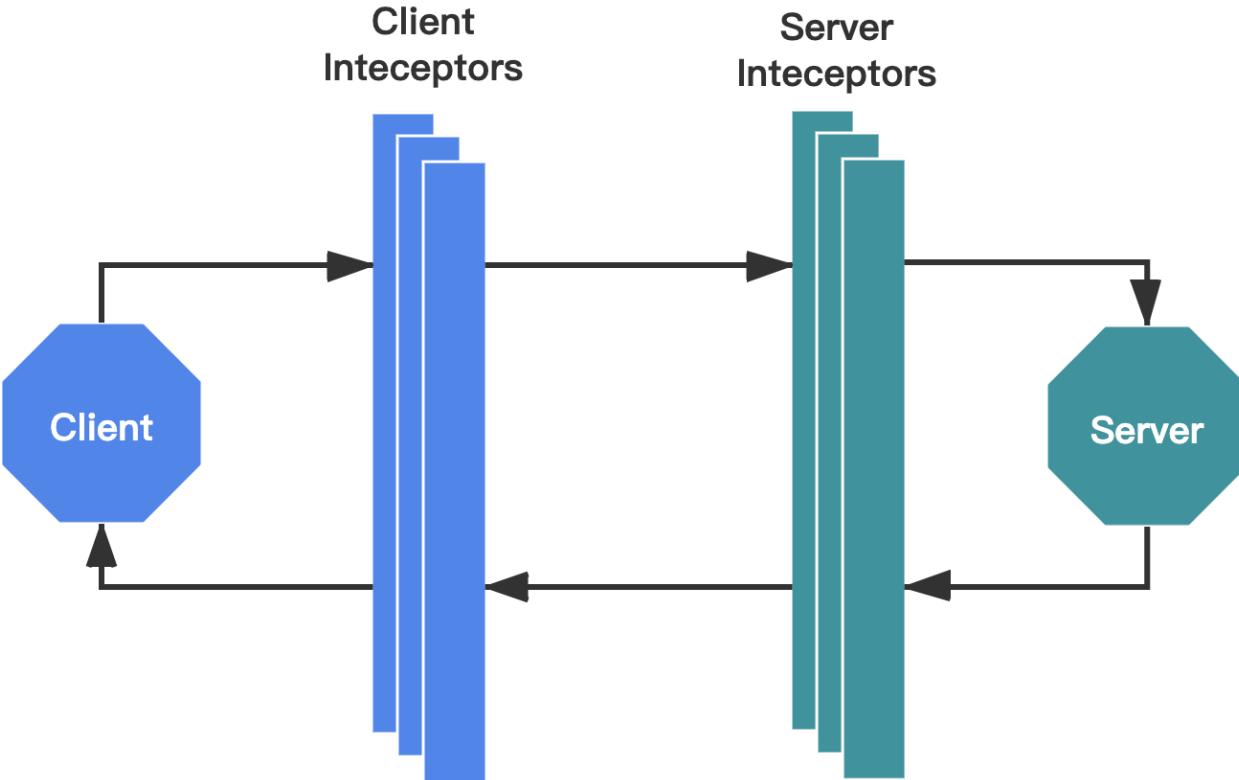
- Kubernetes DNS Resolver VS Kubernetes API Server Resolver
  - DNS resolver is builtin in gRPC framework and its out-of-box for users
  - When connection fail DNS Resolver can resolve name immediately
  - In scale-up scenario, DNS Resolver will not resolve name automatically

# 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"}      ↪ Falling edge detector sometimes doesn't work
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"} ↪ other grad student at your
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/examp
ient", "fppath": "/Users/zheng/shimo/ego/examples/grpc/direct/client/config.toml"} ↪ deprecated GREP_OPTIONS. That's only true in J
2021-04-09 22:04:34    INFO  ego/ego_function.go:199 init max procs        {"comp": "app", "value": 8}   ↪ Why did the Supreme Court vacate the ruling
2021-04-09 22:04:34    INFO  ego/ego_function.go:183 init trace           {"comp": "app"}      ↪ Trump could not block Twitter user? ↪ Human on an interstellar hibernation ship wa
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
Client	OK	0	200	returned on success.
	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

```
● ● ●  
  
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



```
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

应用: ego-demo

企业: shi

环境: dev | 开发环境

集群: shimo-dev

实例

配置

监控

日志

资源依赖

PProf

Grpc调试

应用事件

授权管理

配置文件

dev.toml

版本

087ceca conf

版本信息

Version: 087ceca

ChangeLog: conf

变更时间: 2021-04-09 12:01:14

发布

开始编辑

```
1 debug = true
2 [server.governor]
3 host = "0.0.0.0"
4 port = 9004
5 [server.http]
6 host = "0.0.0.0"
7 port = 9001
8 [logger.default]
9 async = false
10 level = "info"
11 [logger.ego]
12 async = false
13 level = "info"
14 [redis]
15 addr = "{{redis.addr@15}}" # refrence of dev redis address
16 password = "{{redis.password@15}}" # refrence of dev redis password
17 [svc]
18 [svc.svcConf]
19     addr="k8s://svc-conf:50051"
20     clientID="MY_ID"
21     clientSecret="MY_SECRET"
22 [k8s]
```

资源变量: \*\*\*\*

Key: addr

Description:

Value: 127.0.0.1:6379

Author:

# 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"    # reference of dev redis address
  password = "YOUR_PASSWORD" # reference 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

1 module github.com/gotomicro/ego-component/egorm
2 go 1.15
3
4 require (
5   github.com/gotomicro/ego v0.4.1
6   github.com/json-iterator/go v1.1.10
7   github.com/opentracing/opentracing-go v1.1.0
8   github.com/stretchr/testify v1.6.1
9   gorm.io/driver/mysql v1.0.5
10  gorm.io/driver/postgres v1.0.8
11  gorm.io/gorm v1.21.3
12 )

1 module github.com/gotomicro/ego-component/eetcd
2 go 1.15
3
4 require (
5   github.com/golang/protobuf v1.4.2
6   github.com/gotomicro/ego v0.4.1
7   github.com/grpc-ecosystem/go-grpc-prometheus v1.2.0
8   go.etcd.io/etcd v0.5.0-alpha.5.0.20200425165423-262c93980547
9   golang.org/x/net v0.0.0-20201224014010-6772e930b67b
10  google.golang.org/grpc v1.29.1
11 )
```

# 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

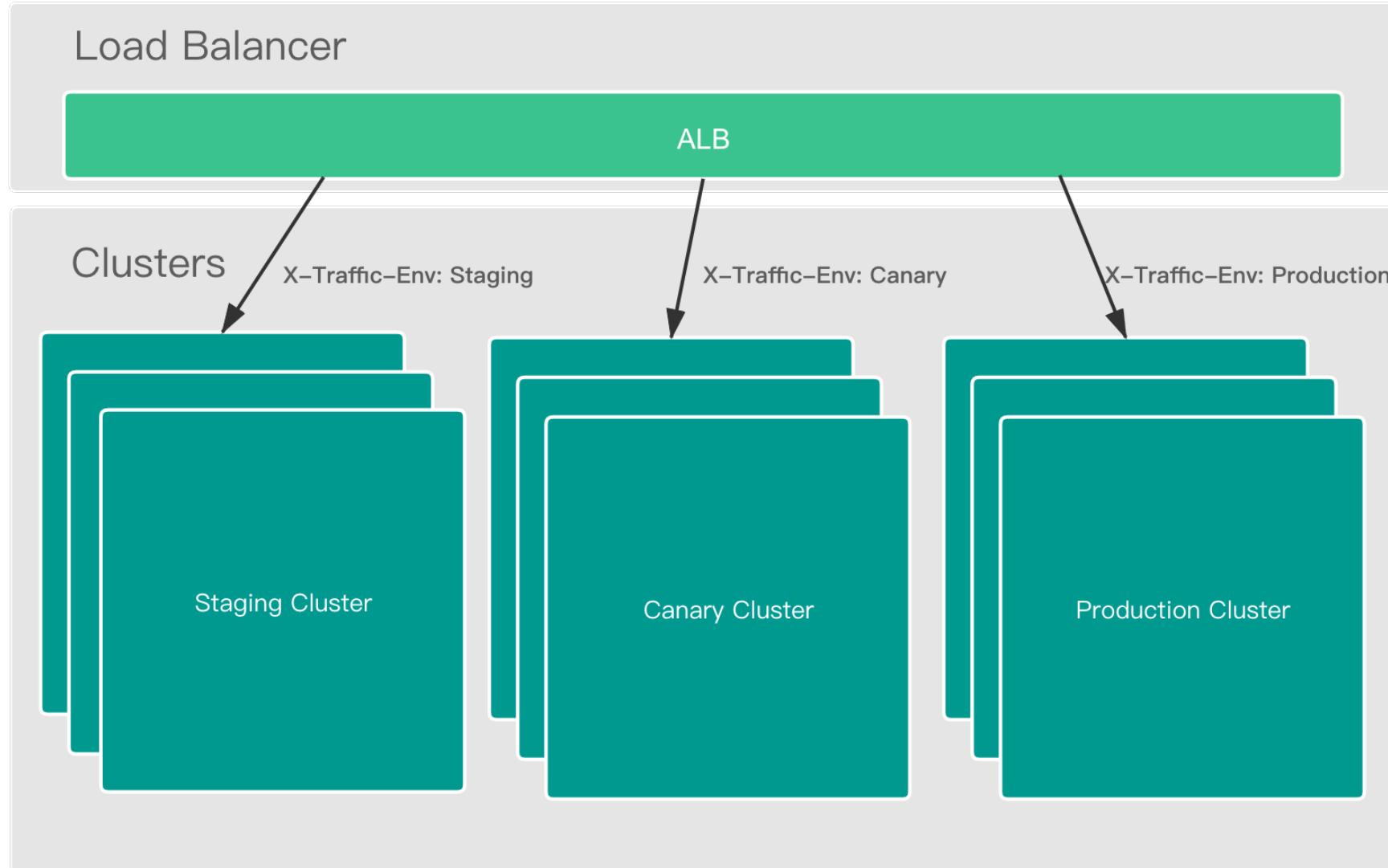
- Append standard logging when error occurs

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

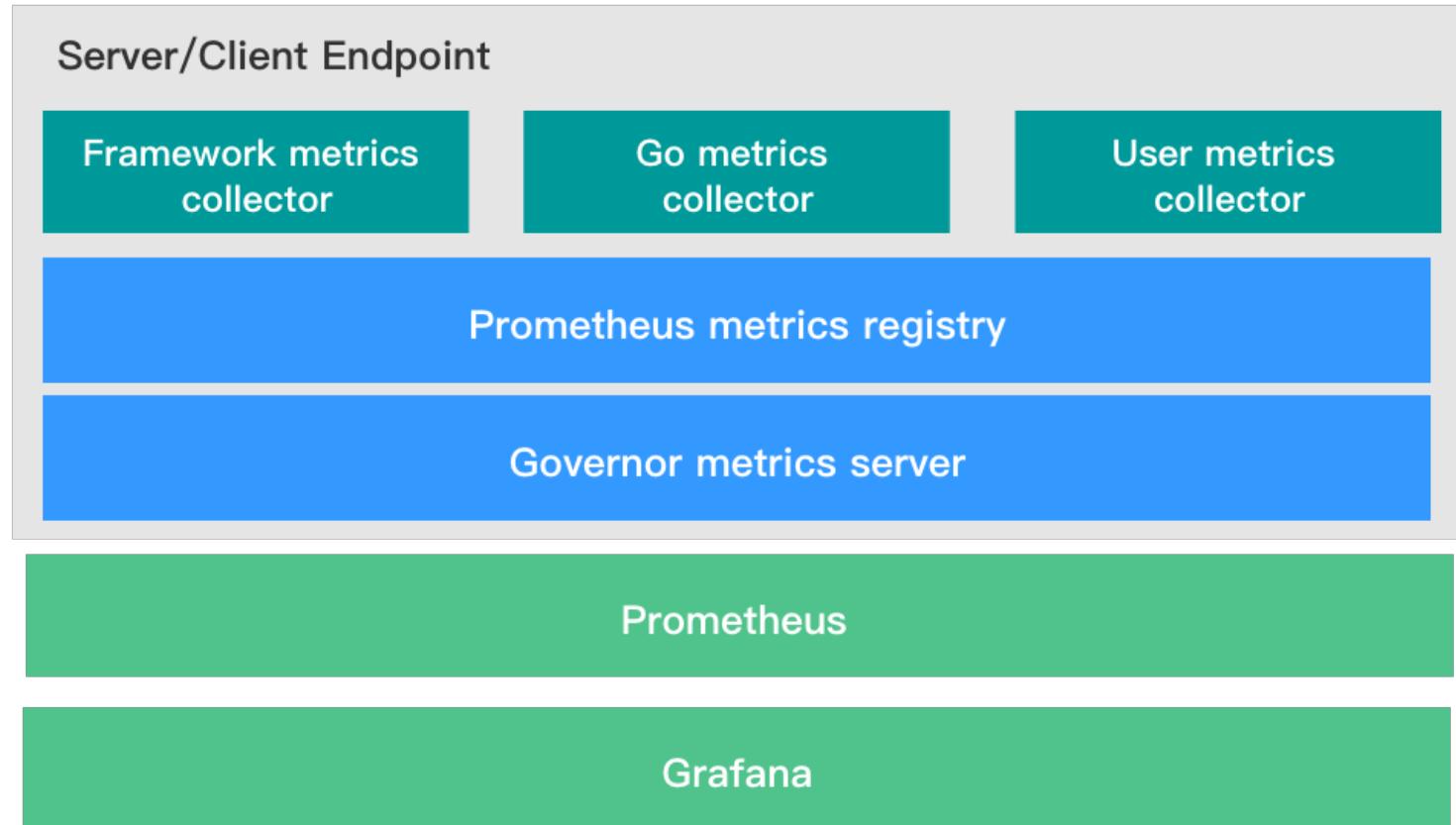
prometheus.io/scrape: true

# 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

实例 配置 监控 日志 资源依赖 PProf 应用事件 授权管理

实例监控 API监控 概览监控 Client监控

instance ☆ ☀

仪表盘 文档 设置 帮助 最近 30 分钟

数据源: process | 环境(版本): production | 实例: All | summarize: 5m | 方法名: All | appLabel of deployment: service | namespace: default

概览



Instance											
Pod	启动时间	应用版本	框架版本	enable	Go版本	Pod IP	Namespace	pod_template_hash	vers	status	lastTransitionTime
service-595... (blue)	2021-04-02 14:09:15	cded1c82995b5716992c7...	v0.4.3	true	go1.15.6	172.24.22.120:9003	default	595857b74c	prod	Running	2021-04-02 14:09:15

实例



# 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

文档实时协同 · 知识沉淀管理 · 数据安全可控

—— 石墨文档