压测安全

1 项目目的

压测安全是整个压测过程中非常重要关键的一环，我们需要在可能出现问题前采取措施，避免影响扩大。

为此我们需要建设压测安全机制，主要包括以下内容：

1. 压测前检查，所有检查项通过压测任务才可以启动
2. 压测时熔断，中止发送压测流量。
3. 服务自我保护，丢弃压测流量

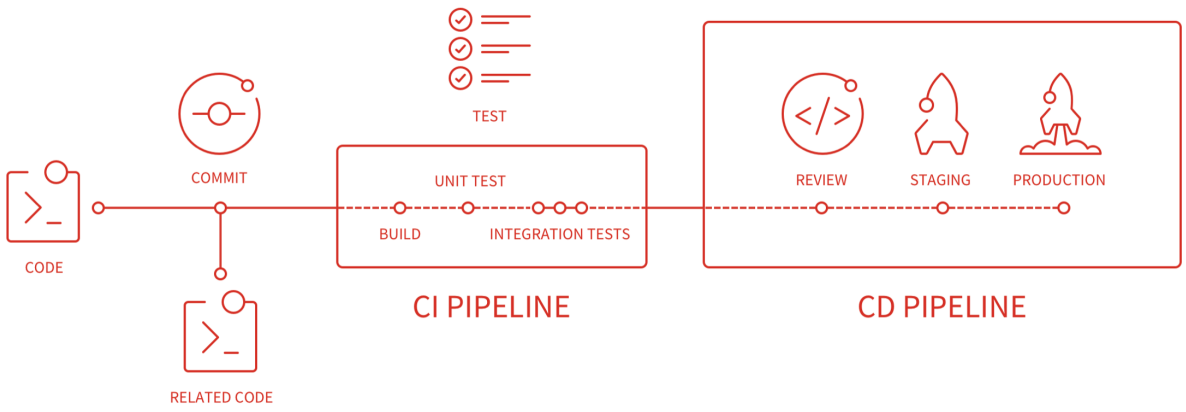
2.1 组件检查

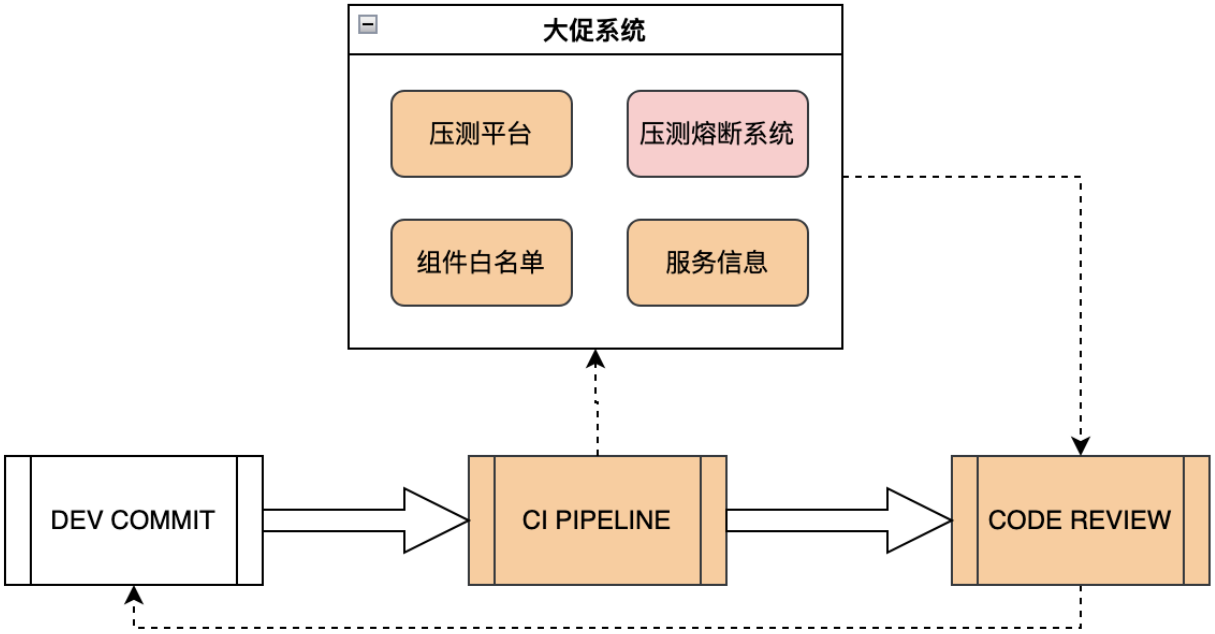
为保障影子 **流量被正确路由**、**链路上报准确**，我们需要使用标准组件。

随着代码迭代、架构升级，不断可能会有预期外的依赖组件引入。

所以我们需要持续扫描各个服务的组件使用情况，分析组件及其版本是否在白名单内。如发现预期外的组件出现，及早作出提示并熔断压测平台，直到组件及其版本符合预期。

我们选择在GITLAB CI PIPELINE阶段检查各个服务组件情况，并上报到大促系统。大促系统做汇总展示，并熔断压测平台，避免因组件导致压测故障。





- 在gitlab配置ci pipline, merge时执行命令检查依赖信息

```
go list -m -f '{{if not .Indirect}}{{.Path}} {{.Version}}{{end}}' all
```

可以得到

```
git.garena.com/shopee/bg-logistics/logistics/celestial
git.garena.com/shopee/bg-logistics/go/chassis v0.3.2-a3.0.20220902103959-02c2af1e346e
git.garena.com/shopee/bg-logistics/go/gocommon v0.4.0-b1
git.garena.com/shopee/bg-logistics/go/scorm v0.0.22
git.garena.com/shopee/bg-logistics/service/saturn-rpc-job v1.1.0
github.com/aws/aws-sdk-go v1.38.61
github.com/bsm/redislock v0.7.0
github.com/confluentinc/confluent-kafka-go v1.4.2
github.com/dgrijalva/jwt-go v3.2.0+incompatible
github.com/emicklei/go-restful v2.12.0+incompatible
github.com/go-chassis/go-archaius v1.6.0-beta1
github.com/go-redis/redis v6.15.9+incompatible
github.com/go-redis/redis/v8 v8.1.0
github.com/golang/protobuf v1.5.2
github.com/google/pprof v0.0.0-20210720184732-4bb14d4b1be1
github.com/google/wire v0.5.0
github.com/hashicorp/golang-lru v0.5.4
github.com/jinzhu/copier v0.0.0-20190924061706-b57f9002281a
github.com/json-iterator/go v1.1.11
github.com/patrickmn/go-cache v2.1.0+incompatible
github.com/pkg/errors v0.9.1
github.com/satori/go.uuid v1.2.0
github.com/segmentio/kafka-go v0.3.4
github.com/spf13/cast v1.3.1
github.com/stretchr/testify v1.7.0
github.com/tealeg/xlsx v1.0.5
github.com/tidwall/gjson v1.8.1
github.com/vcaesar/murmur v0.20.1
github.com/vmihailenco/msgpack v4.0.4+incompatible
golang.org/x/net v0.0.0-20210726213435-c6fcb2dbf985
golang.org/x/oauth2 v0.0.0-20211104180415-d3ed0bb246c8
gonum.org/v1/plot v0.9.0
google.golang.org/api v0.68.0
google.golang.org/grpc v1.45.0
gopkg.in/go-playground/validator.v9 v9.29.1
gopkg.in/gomail.v2 v2.0.0-20160411212932-81ebce5c23df
gopkg.in/yaml.v2 v2.4.0
gorgonia.org/gorgonia v0.9.17
gorgonia.org/tensor v0.9.17
gorm.io/driver/mysql v1.1.1
gorm.io/gorm v1.21.12
```

执行curl命令将以上信息post到大促系统接口

- 大促系统维护组件白名单
- 大促系统-压测平台展示各服务组件信息，以及和组件白名单的差异

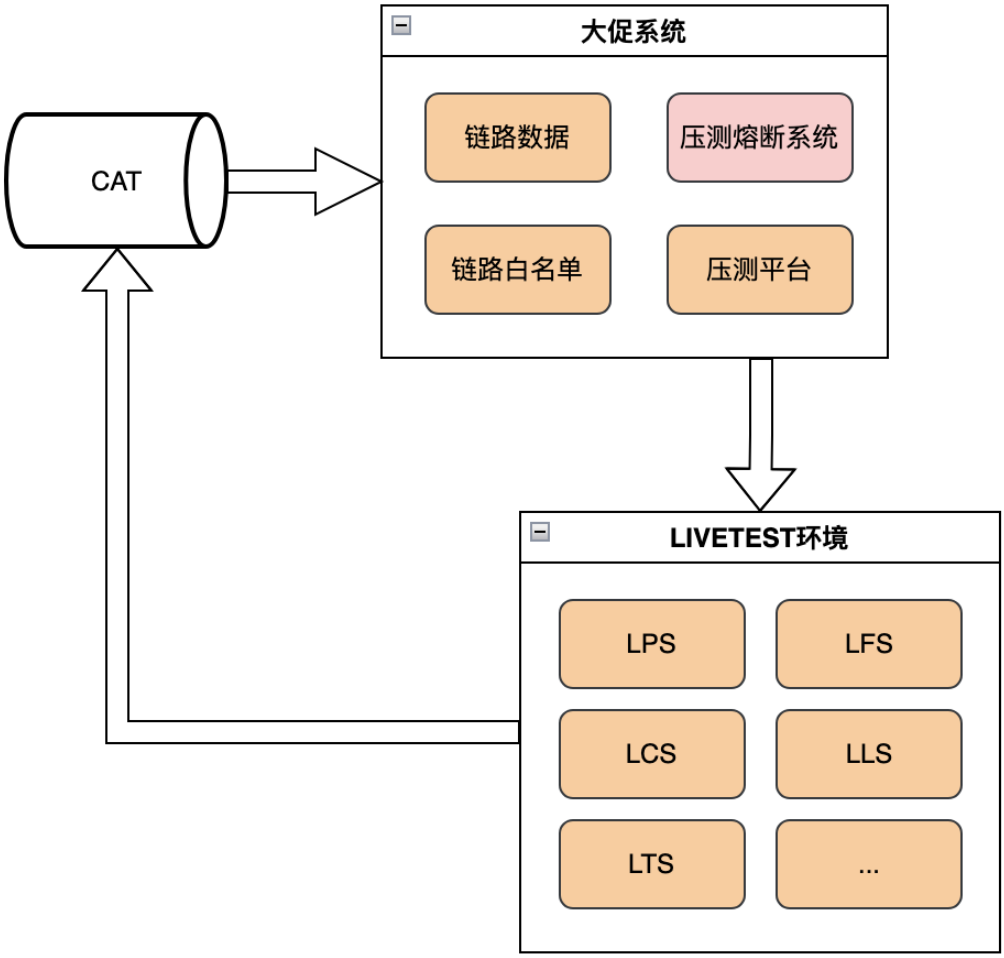
3.2 链路检查

压测链路以及影子流量可控是确保压测安全的关键。

随着代码迭代、架构升级，不断可能会有预期外的服务或中间件引入。

所以我们需要持续检查压测链路的情况，分析压测链路中的服务和中间件是否在白名单内。如发现预期外的链路出现，及早作出提示并熔断压测平台，直到压测链路符合预期。

因为live链路是抽样采集，live服务请求量比较大，不容易采集到我们发送的影子请求，我们选择在livetest环境采集和检查链路。



2.3 流量覆盖率检查

参考：动态覆盖率方案探讨（new）Golang覆盖率服务接入

为了避免部分代码逻辑没有被压测流量覆盖，导致性能问题被忽略，引发live故障，我们需要检测压测流量覆盖率。

覆盖率检查已经在dms内实现，各项目组接入后大促系统会采集检查结果，在检测结果达标前阻塞正式压测。

2.4 影子库检查

选取一个非live环境，确保服务只配置影子库，发送流量测试。

如果没有正确传递影子标识，未写入影子库会报错。

3 压测时熔断

参考：全链路压测平台--技术文档-压测熔断

4 服务自我保护，丢弃压测流量

压测live造成服务过载时，一方面压测平台需要熔断，另一方面服务需要自我保护。

服务需要配置限流，在达到上限时丢弃压测流量，以保证服务正常响应用户请求。

chassis提供相关功能，@todo 沟通使用方式