import Callout from “nextra-theme-docs/callout”;

云原生部署平台

G123 所有游戏均部署在基于Kubernetes构建的云平台上。游戏服务端各容器化应用的镜像推送、部署、DB 配置等内容，以及客户端的 CDN 配置、资源上传等，均可通过Publisher 工具（使用说明书）来实现系统组件的申请与修改、应用程序的部署与发布等。平台同时提供基于Grafana Cloud的日志监控系统。

1. 系统构成

DOs

- 支持 **CDN** 全球节点部署、客户端**静态资源**及公开的客户端用**配置文件**。（提供命令行与拖拽上传）
- 支持各种应用的**分布式**部署。（如游服、登陆等服务的幂等冗余部署、需要保证数据一致性）
- 支持各种**多进程**应用拆分成多容器部署。（如游服应用的主逻辑进程、数据存储进程、网关进程等。任意容器掉线、集群自动拉起。）
- 支持登入**容器内**访问。（Publisher 工具内对应 APP 下的服务页面内可登入 Pod）
- 支持游服类型 APP 的**自动部署**与**自动开服**。
- 支持各种应用的**不停机**滚动更新。（先启动新版本后停止旧版本、推荐登录支付等无状态 APP使用）
- 支持三种外部存储：主从非分片 **Redis** (`5.0`, `6.0`, `7.0`)，集群版 **MySQL** (`5.7`, `8.0`) & 副本集 **MongoDB** (`5.0`). （GMS 用玩家行为数据请分库分表存储、平台不额外提供数据库集群、保留最近一个月的数据即可）
- 支持服务端多 APP **共享配置文件**的管理。（Publisher 工具内服务端的配置文件功能）
- 支持**定时任务**。（有任务容器的部署与启动时间、精准定时任务请通过 API 等方式实现）
- 支持各应用 APP 公网访问地址的 **WSS 协议**通信。（通过负载均衡 ALB 反向代理实现、请在应用层做好协议升级：浏览器 → ALB 某 APP 路径:443 → 某 APP 的“对外”端口:9000）
- 支持 **SSL 证书**托管、无需在应用内实现。

- 支持**自动收集**服务端应用打印到、标准输出(stdout、控制台输出)的日志至Grafana Cloud **实时查询**。
- 支持各系统组件与应用性能的Grafana Cloud **实时监控**。

DON'Ts

- 不提供本地**磁盘**。(容器内/g123/路径下文件、支持持久化存储与下载。适用于低 I/O 场景、勿保存日志文件)
- 不提供 **Windows** 镜像部署。
- 不提供**基础镜像**、请各游戏根据技术栈自行选取。(不推荐使用 3.3 及更早版本Alpine 镜像)
- 不提供默认**时区**的配置、请在各应用镜像 Dockerfile 内定义。(`ENV TZ=Asia/Tokyo`)
- 不提供应用停机前的**内存数据**入库。(务必做好主进程 `PID 1` 监听 OS、发出的 `SIGTERM` 以优雅停机)
- 不提供单应用多端口的**公网**访问。(每个 APP 只有第一个端口对外、包括多容器部署场景、其他端口对外请自行实现转发)
- 不支持负载均衡与服务端超过 **60 秒**闲置的长连接。(请主动实现保活心跳机制、每 60 秒内进行一次报文交互)
- 不支持服务间通过 **IP** 地址连接。(请使用内网域名)
- 不要自己做操作系统级别的 **crontab**. (请使用 Publisher 定时任务功能)
- 不要自己做**中间件**的容器化部署。(如消息队列、Zookeeper 等)
- 不要自己做数据 **pipeline**. (如 Hadoop 等)

2. 功能模块

以下为发布至G123 平台前需要各游戏开发实现的功能列表

2.1 自动开服

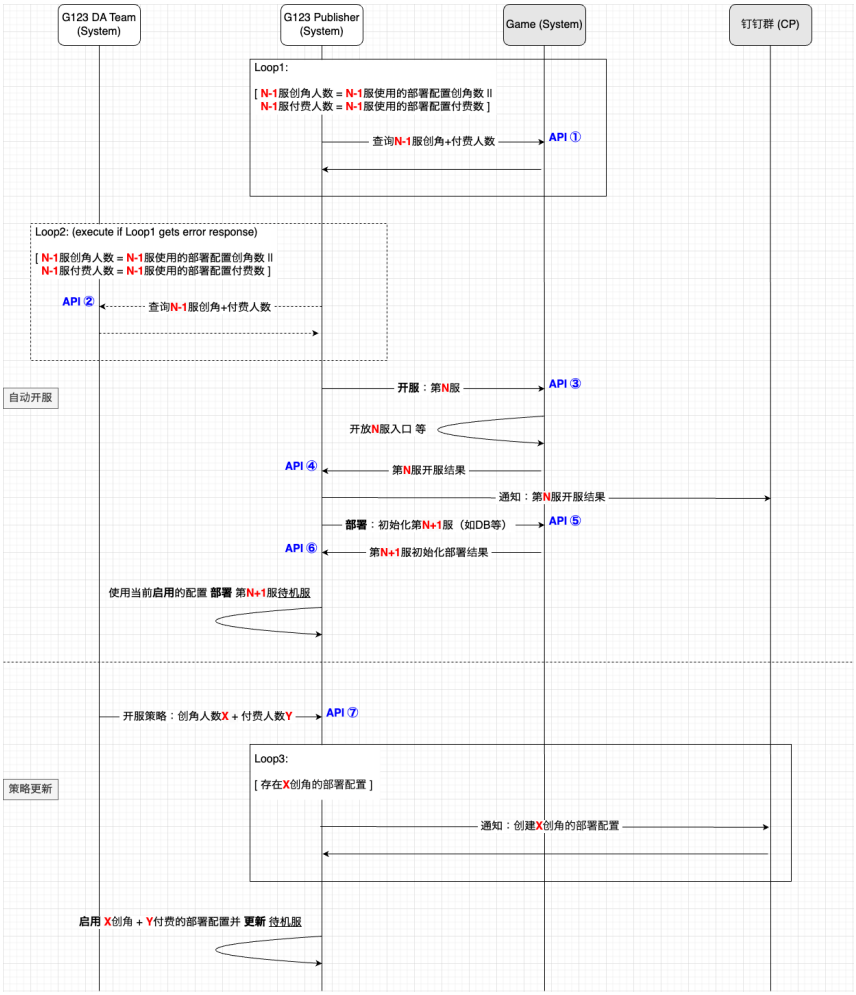
游服类型 APP 需提供该功能。可优先游戏部署、正式上线前必须对应完成

- 游服类型 APP 需支持、在特定开服策略 (如游戏客户端埋点 `g_createrole` 创角 5000 人) 下的自动部署与开服功能。

- 游服 N-1 服达到开服策略定义（如创角 5000 人）时、会触发游服 N 服的“自动开服”操作。如:

  1. 平台查询游服 N-1 服的创角人数、达到开游服 N 服条件
     - API ：实时查询游戏客户端上报给平台的创角埋点定义（默认）
     - API ：实时查询游戏内创角定义、如果必要请提供相关 API 接口并通过自动开服“保护机制”开启（可选）
  2. 平台调用游戏的开服通知 API ：游服 N 服可允许玩家进入、新玩家请导流至此服并更新选服页面
  3. 游戏调用平台的开服回调 API ：能否开服
  4. 平台接收开服回调、并通过小包在钉钉大群通知开服结果
  5. 平台调用游戏的部署通知 API ：准备部署游服 N+1 服、请做好该服的初始化动作（如数据库初始化、配置更新等）
  6. 游戏调用平台的部署回调 API ：能否部署
  7. 平台接收正常的部署回调：环境变量 $SERVER_ID 自动 +1 并用于启动游服 N+1 服

- 开服策略调整时、请更新开服配置（如 CPU/Mem 等）以保证系统稳定运行



2.1.1 部署 当游服 APP 完成 1 服部署并首次开启自动开服功能时、系统会主动发起针对 2 服的部署 API 调用。此时 1 服为**导流服**、2 服为**预备服**（完成部署、等待开服）

2.1.1.1 部署通知 API Webhook

必须是 https 接口、请在 Publisher 内记入 { Webhook }

Authorization

平台调用该 { Webhook } 时会使用、请在 Publisher 内记入 { Token }

Request

> 当自动开服策略 **不包含创角人数**（如按付费人数或者定时开服等）时、请求中的 **serverUsers** 字段将默认传入：**0**

```
{
  "actionId": string          // 操作的 TraceID, publisher 会用该 TraceID 来追踪这次操作的
  "serverUsers": number       // 新服配置的创角人数或0
  "serverId": number          // 当前请求开第几个服
  "newServerNames": []string // 游服类型 APP 下的 Pod 名称、即{游服 APP 名称}{serverId}
  "callbackToken": string     // 调用平台回调 API 时需要回传该 Token
}
```

Response

```
{
  "message": string
}
```

HttpStatusCode	Description
200	OK. 如果 webhook 同步执行，耗时不长，可直接返回 200 无需回调平台
202	Accept. 如果该 webhook 需要异步执行，返回 202. 完成后调用平台的开服回调 API
403	No Permission. 当 Token 不正确时
422	请求不能执行，请返回 message

示例

Webhook

`https://appid-slb.stg.g123-cpp.com/center/new_server`

Authorization

Token: `test_g123_token1234`

Request

POST /center/new_server Host: https://appid-slb.stg.g123-cpp.com
Authorization: test_g123_token1234

```
{
  "actionId": "1000",
  "serverUsers": 5000,              // 新服配置 5000 创角
  "serverId": 2,                    // 新服为 2 服
  "newServerNames": ["game2"]       // 游服类型 APP 名称为 game
  "callbackToken": "^6Q%j0kZ74S"    // 调用平台回调 API 时需要
}
```

2.1.1.2 部署回调 API

如果部署通知 API 返回 200 不需要调用该接口。

- 测试环境:https://game-cloud.stg.g123.jp/cp/api/v1/new_server/callback
- 正式环境:https://game-cloud.g123.jp/cp/api/v1/new_server/callback

Request

```
{
  "appId": string
  "actionId": string      // 回传平台调用开服通知 API 时的 actionId
  "callbackToken": string // 回传平台调用开服通知 API 时的 callbackToken
  "success": bool         // 处理成功
  "message": string       // 处理失败时将原因放在该字段
}
```

Response

```
{
  "type": string
  "message": string
  "reason": string
}
```

HttpStatusCode	Type	Description
200		OK
403	NO_PERMISSION	没有权限
422	NO_WAITING_ACTION	该开服流程没有使用回调（请查看部署通知 API 在实现时返回的 200）

2.1.2 开服

默认游服 **APP** 在开启自动开服功能前、已手动完成 1 服的部署与开服。以下为针对从游服 2 服起的 API 调用

2.1.2.1 开服通知 API Webhook

必须是 https 接口、请在 Publisher 内记入 { Webhook }

Authorization

平台调用该 { Webhook } 时会使用、请在 Publisher 内记入 { Token }

Request

> 当自动开服策略 **不包含创角人数**（如按付费人数或者定时开服等）时、请求中的 **serverUsers** 字段将默认传入：**0**

```
POST { Webhook }
Authorization: { Token }

{
  "actionId": string          // 操作的 TraceID, publisher 会用该 TraceID 来追踪这次操作的
  "serverUsers": number       // 新服配置的创角人数或0
  "serverId": number          // 当前请求开第几个服
  "newServerNames": []string // 游服类型 APP 下的 Pod 名称、即{游服 APP 名称}{serverId}
  "callbackToken": string     // 调用平台回调 API 时需要回传该 Token
}
```

Response

```
{
  "message": string
}
```

HttpStatusCode	Description
200	OK. 如果 webhook 同步执行，耗时不长，可直接返回 200 无需回调平台
202	Accept. 如果该 webhook 需要异步执行，返回 202. 完成后调用平台的开服回调 API
403	No Permission. 当 Token 不正确时
422	请求不能执行，请返回 message

示例

Webhook

`https://appid-slb.stg.g123-cpp.com/center/open_server`

Authorization

Token: `test_g123_token1234`

Request

POST `/center/open_server` Host: `https://appid-slb.stg.g123-cpp.com` Authorization: `test_g123_token1234`

```
{
  "actionId": "1000",
  "serverUsers": 5000,              // 新服配置 5000 创角
  "serverId": 2,                    // 新服为 2 服
  "newServerNames": ["game2"]       // 游服类型 APP 名称为 game
  "callbackToken": "^6Q%j0kZ74S"    // 调用平台回调 API 时需要
}
```

2.1.2.2 开服回调 API

如果开服通知 API 返回 200 则无需调用该接口。

- 测试环境:https://game-cloud.stg.g123.jp/cp/api/v1/open_server/callback
- 正式环境:https://game-cloud.g123.jp/cp/api/v1/open_server/callback

Request

```
{
  "appId": string
  "actionId": string       // 回传平台调用开服通知 API 时的 actionId
  "callbackToken": string // 回传平台调用开服通知 API 时的 callbackToken
  "success": bool          // 处理成功
  "message": string        // 处理失败时将原因放在该字段
}
```

Response

```
{
  "type": string
  "message": string
  "reason": string
}
```

HttpStatusCode	Type	Description
200		OK
403	NO_PERMISSION	没有权限
422	NO_WAITING_ACTION	该开服流程没有使用回调（请查看开服通知 API在实现时返回的 200）

2.2 游服合服

随着游戏上线时间拉长、既存游服的 DAU 会逐渐降低、影响游戏内的玩家生态。建议通过监控游服数据、做**不定期合服**来提高游戏玩家的活跃度等。

请参考以下合服流程、提前做好游服相关数据库表 Schema 的设计、方便合并。

如：游服 *1*、*2*、*3*、*4* 合服的大致流程如下 - **数据库**：游服 1、2、3 的库表合并至 4 服的库表（策划提供方案、各 *CP* 服务端自行完成）- **配置文件**：1、2、3、4 服的相关配置做合并或更新（*CP*

服务端）- **客户端**：将 1、2、3 服导流至 4 服服务端（*CP* 客户端）- **游服**：删除游服 1、2、3 的 Pod（*CP* 服务端）

3. 附录

3.1 系统架构图

G123 平台主推基于 Kubernetes 的云原生游戏系统、需要各 CP 根据下面的示意图、调整优化游戏架构，并进行特定功能的开发。

- 公网层（红）CDN 通过全球边缘节点下载客户端资源至玩家浏览器
- 公网层（红）ALB 反向代理游戏服务端所有应用 APP 与玩家浏览器通信
- 计算层（绿）ALB 通过将工具添加的应用 APP 名称做为路径导流至应用（公网访问地址）
- 计算层（绿）APP 之间的通信走域名（内网访问地址）做内网的服务发现
- 存储层（蓝）DB 理论上只能内网访问、公网连接需在工具内配置 IP 白名单

3.2 日志与监控

我们基于 Grafana LGTM 技术栈,构建了一套可观察方案。为了“轻”且“快”,我们采用了 Grafana Cloud 的全托管式。

3.2.1 日志查询

3.2.1.1 基础篇

1. 访问 Grafana Cloud 并通过 Auth0 登录

- 用户名和密码和登录 Publisher 工具是一样的。
- 如果登录后，发现找不到 `Explore`，请联系 CTW SRE。首次登录后，才能赋权。

2. 左边栏选择 `Explore`
3. 数据源 `Data Source` 选择对应游戏的 `${game}-logs`

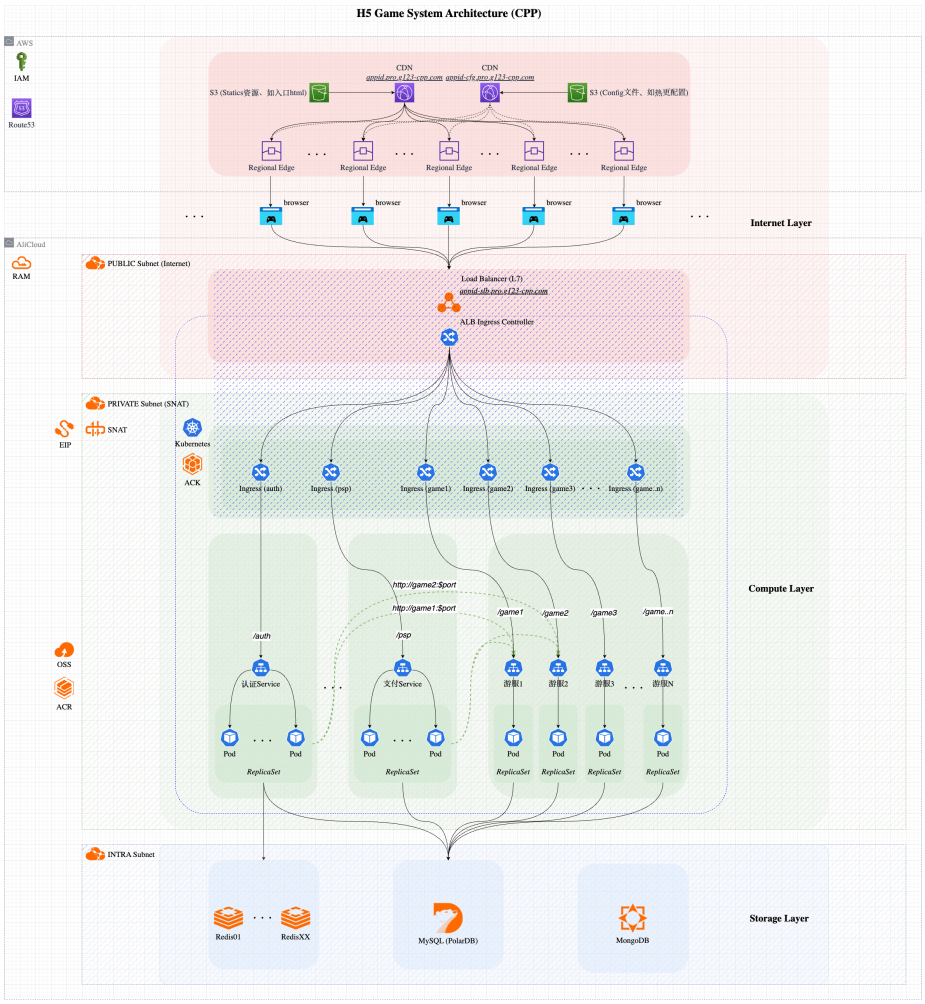

H5 Game System Architecture (CPP)
AWS
IAM
Route53
CDN
CDN
S3 (Statics资源、如入口html)
S3 (Config文件、如热更配置)
Regional Edge
Regional Edge
Regional Edge
Regional Edge
Regional Edge
browser
browser
browser
browser
browser
Internet Layer
AliCloud
RAM
PUBLIC Subnet (Internet)
Load Balancer (L7)
ALB Ingress Controller
PRIVATE Subnet (SNAT)
EIP
SNAT
Kubernetes
ACK
Ingress (auth)
Ingress (psp)
Ingress (game1)
Ingress (game2)
Ingress (game3)
Ingress (game..n)
Compute Layer
http://game2:$port
http://game1:$port
/game1
/game2
/game3
/game..n
/auth
/psp
OSS
ACR
认证Service
支付Service
游服1
游服2
游服3
游服N
Pod
ReplicaSet
iNTRA Subnet
Redis01
RedisXX
MySQL (PolarDB)
MongoDB
Storage Layer

图 1: workflow

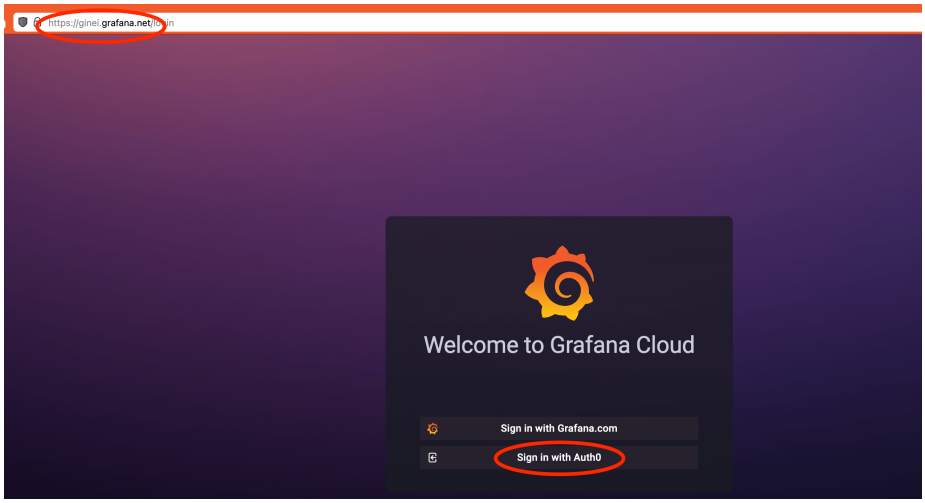


图 2: workflow

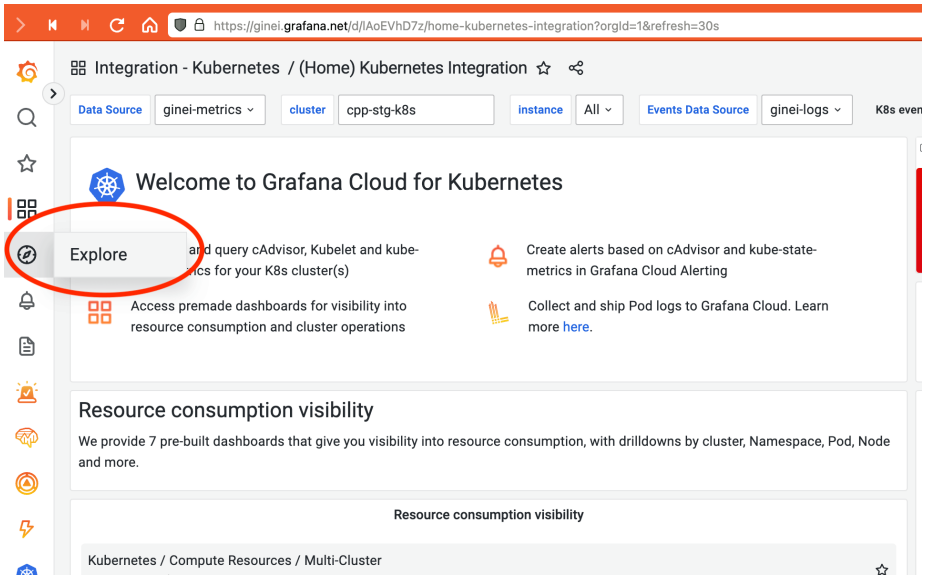


图 3: workflow

- 选择 `${game}-logs`
- 选择集群：测试环境的 cluster 是 `cpp-stg-k8s`，生产环境的 cluster 是 `cpp-pro-k8s`
- 选择自己游戏的 `namespace`. 可以追加 `app` 纬度来过滤出特定 APP
- 可以过滤任意字段

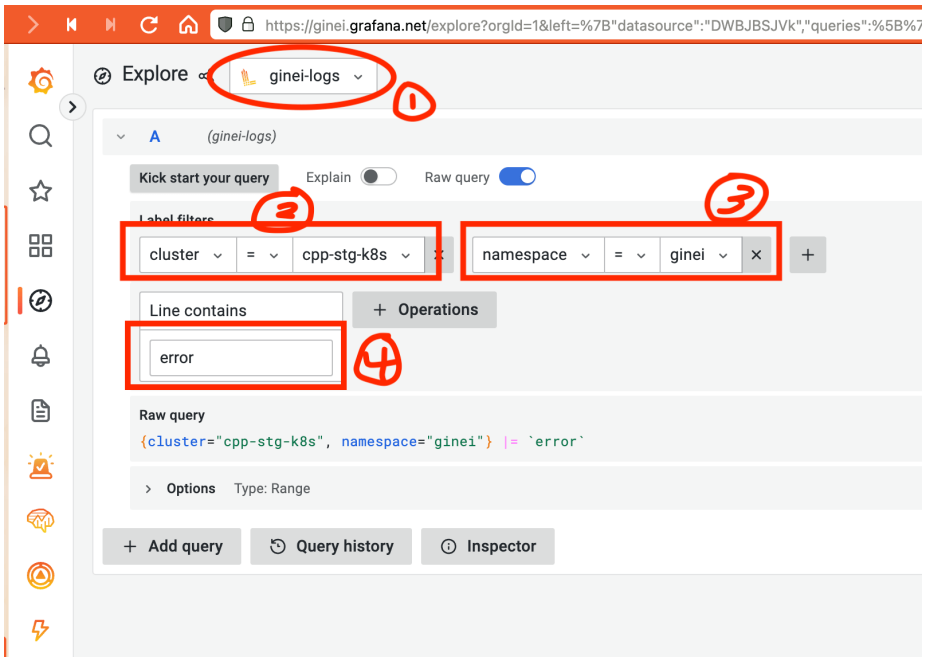


图 4: workflow

4. 加速查询

- 建议通过选择时间来缩小查询日志的范围。范围越小，查询速度越快。
- 注意时区。默认是浏览器时区。

5. 其他最佳实践

- 可以在自由编辑和辅助编辑模式之间切换
- 可以直接分享你的查询语句、这在调试时非常实用
- 可以查看你的查询历史记录、支持收藏
- 实时 Live 模式、类似 `tail -f`、日志会实时刷屏、可监控发版情况

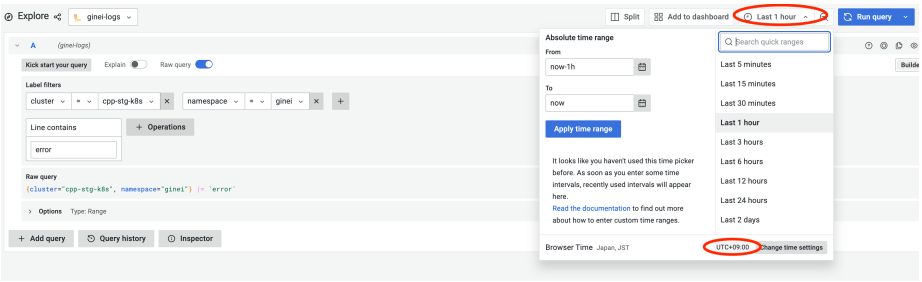


图 5: workflow

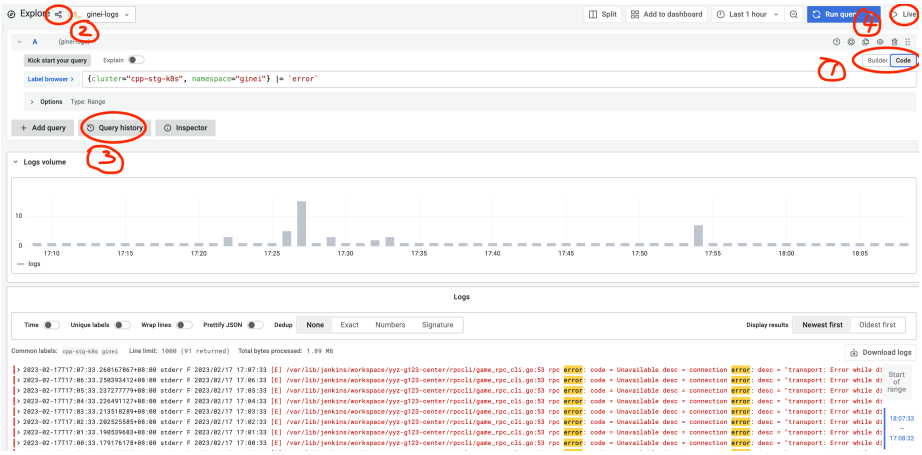


图 6: workflow

3.2.1.2 进阶篇

1. For more advanced usage, please read and learn the LogQL.
    - 日志搜索语言是LogQL，可以参考这个链接去学习完整的用法。
2. 日志可以是结构化的，比如 JSON，也可以是平文。LogQL 内置了几种分析器，最常用的是

- json 用来分析 JSON 格式日志
- pattern 用来分析空格分开的日志。例：时间组件文件行号日志内容
- logfmt 用来分析 logfmt 格式 (key=value) 的日志。例：`time=xxxx component=xxxx file=xxxx`
- regexp 用来做正则表达式捕捉。
    - 注意：这里的正则表达是 Golang RE2 syntax，不是常用的 Perl 文法。
    - 注意：正则表达捕捉很慢。

示例 1

(1) label 是可以做正则匹配的。(2) 这里用了 `| pattern ` <_> <stream> <_>` ` 来分析日志。

- `|` 是连接符，用来把多个处理串联起来。
- `pattern` 指明要用的分析器。
- ` ` ` 用来把分析语言包裹起来,也可以用双引号",但要注意分析语言里面的双引号需要 escape。
- `<_>` 每个匹配用 `<>` 包裹起来，下划线 `_` 表明不需要 index 这个捕捉。
- `<_> <stream> <_>` 这里的意思是，第一个用空格分离的字段—时间—不要。第二个字短捕捉并命名为 stream，后面所有的字段都不要。
    - 注意：这里捕捉成功后，会自动增加一个叫做 stream 的索引。后面可以直接用这个索引。
    - 这是 LogQL 最强大的地方，运行时动态索引。

示例 2

复杂化上面的示例 1，我们做点无用的复杂来展示一个完整的分析功能。LogQL 如下：

```
sum by (dest) (
  count_over_time(
    {cluster="cpp-stg-k8s", namespace="ginei", pod_container =~ "gameserver[0-9]{1,4}"}
```

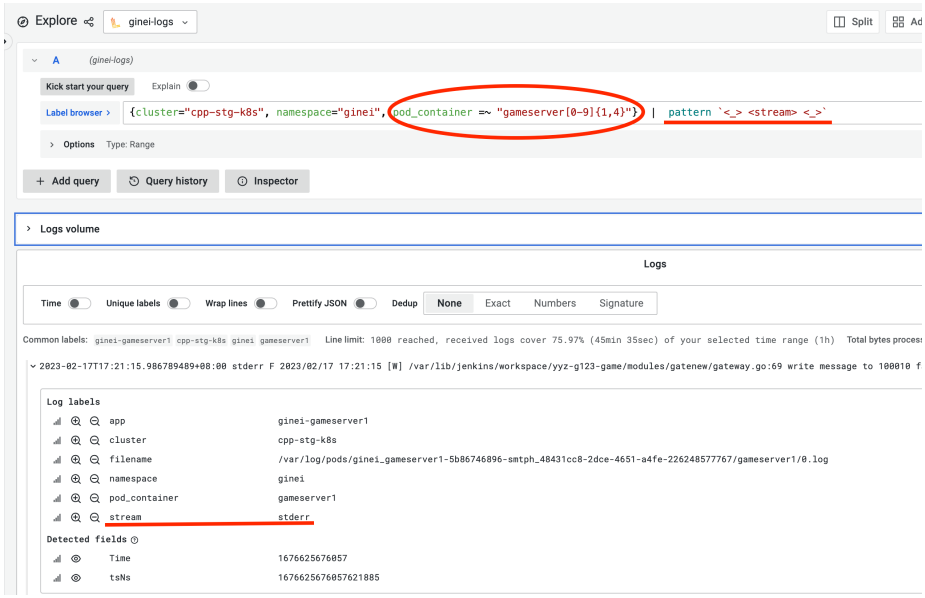


图 7: workflow

```
    | pattern `<_> <stream> <_> write message to <dest> fail`
    | __error__ =""
    | stream = "stderr"
  [1m])
)
```

- `pattern `<_> <stream> <_> write message to <dest> fail`` 会匹配这样的日志:
  - `2023-02-17T17:55:26.204880125+08:00 stderr F 2023/02/17 17:55:26 [W] /var/lib/jenkins/workspace/yyz-g123-game/modules/gatenew/gateway.g write message to 100010 fail, not exist` 并取出 stderr 和 100010 部分，分别索引为 stream 和 dest。
- `__error__ =""` 用来过滤掉分析器出错的日志。**error** 是一个内部变量，用来保存 LogQL 的执行错误。
- `stream = "stderr"` 用我们前面分析器里建立的 index 来过滤日志。
- `count_over_time( xxxx [$interval])` 用来计数指定间隔 $internal 内有多少条日志命中，这里是 1 分钟。

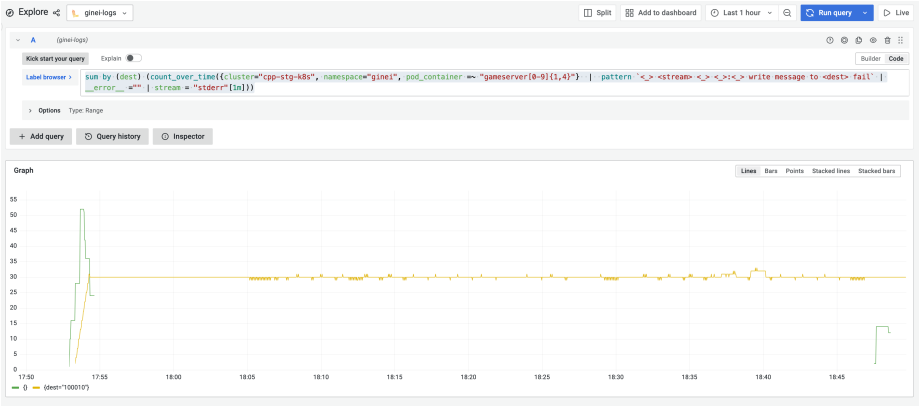


图 8: workflow

- `sum by (dest) (xxxx)` 用 dest 来做聚合。因此，上面的 LogQL 用来统计，出现 write message to xxx 这样的错误日志每分钟，每个 dest 有多少条。

3.2.2 监控查询

3.2.2.1 系统监控

1. 访问 Grafana Cloud 并通过 Auth0 登录

- 用户名和密码和登录 Publisher 工具是一样的。
- 如果登录后，发现找不到 `Dashboard`，请联系 CTW SRE。首次登录后，才能赋权。

2. 左边栏选择 **`Dashboard`** 并找到自己的 appid

- 如果大盘有数据或者系统组件监控项缺失，请联系 CTW SRE。

3.2.2.2 游服监控 为完整监控各游服的状态或者协助自动开服，请实现以下游服列表 zonelist API 接口供平台调用查询游服状态。

1. API 需要通过公网访问，所以需要认证。认证可以是 API token，也可以是 bare auth。实现方式可由游戏开发方决定。API endpoint 中的 key，是用 API token 实现的例子。
2. healthcheck 的深度由游戏开发方决定。但至少要反应 登录服务是否健康。因为经常会遇到游戏服务器进程和端口健康，但玩家无法登录的情况。

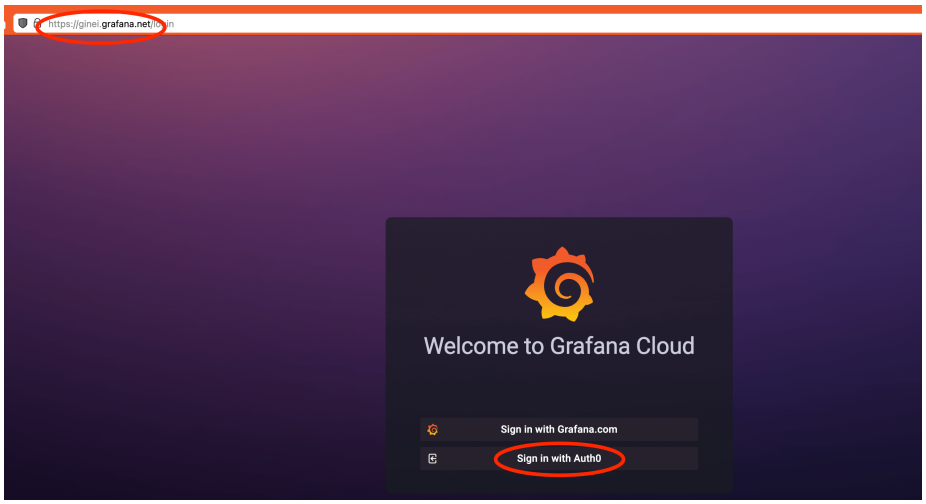


图 9: workflow

3. API 的字段含义在 `{Properties}` 小结有描述。
4. healthcheck 的实现必须是异步，非阻塞的。否则会对游戏造成影响。一个外部监控点，访问 /healthcheck 的频率为 60 秒 / 次。但可能会从多点访问，所以 API 的 rqs 会 >= 1。

Endpoint

URL 路径格式可以自主设定

```
GET https://${appid}-slb.pro.g123-cpp.com/gs/zonelist
```

Authentication

Use HTTP header.
Authorization: `Token` configured in publisher

Response

Success

```
HTTP 200 OK
```

```
{
```

```
  "code": 0,
  "data": {
    "zonelist": [
      {
        "ID": 1,
        "capacity": 3974,
        "createrole": 3010,
        "online": 27,
        "paiduser": 57,
        "opentime": "2022-07-16T19:20+09:00",
        "updatetime": "2022-07-16T19:20+09:00",
        "path": "/gs0001",
        "svrname": "s1",
        "status": 0,
        "url": "https://${appid}-slb.pro.g123-cpp.com/login/zonelogin/1"
      },
      ...
    ]
  }
}
```


Error

```
HTTP 400 Bad Request
```


```
{
  "code": 1,
  "message": "Error description"
}
```


Properties

code ***(int)*** status of the zonelist API itself

- 0: The API is healthy.
- 1: There is a failure.

data (object) | Parameter | Description | | ——— - | ——————————————————————————————–| | **zonelist** (array) | An array of game zones. | | **ID** (int) | UID，游服的唯一标识。不可变更。| | **capacity** (int) | 游服的创角上限。| | **createrole** (int) | 游服的实时创角数。createrole => capacity 意味着满服。务必实现。| | **online** (int) | 实时在线玩家数。如何判断玩家是否在线，由 CP 开发方决定。| | **paiduser** (int) | 游服的实时付费数。务必实现。| | **opentime** (string) | 开服时间。时间格式是 YYYY-MM-DDTHH:MM+09:00。| | **updatetime** (string) | 游服最后一次更新时间。比如合服，开服，维护等。时间格式是 YYYY-MM-DDTHH:MM+09:00。| | **path** (string) | 该游服的路由路径。配置在路由规则里的路径。比如 /game1。| | **svrname** (string) | 游服表示名。比如 game1。| | **status** (int) | 该游服状态码：- 0：正常 - 101：异常 - 102：维护中 - 201：可开服 - 202：导流中（开服完毕）- 301：已合服 - 302：合服中 - 501：特殊游服，比如先锋服，灰度发布服，蓝绿发布服等。| | **url** (string) | 登录该游服的 URL。|

message (string) error message in response

3.2.2.3 运行时监控

会提供一些常见编程语言的运行时数据收集方案。

Java Instrumentation

我们可以通过 `OpenTelemetry` 把 `Java` 程序的运行时（Runtime）数据收集并发送到 Grafana Cloud，从而实现实时分析和监控。

配置方法

1. 在 `publisher.g123.jp` 里配置程序的环境变量。

在工具内特定 APP 的部署配置页面，添加以下环境变量：

```
OTEL_EXPORTER_OTLP_PROTOCOL=http/protobuf
OTEL_RESOURCE_ATTRIBUTES=service.version=${ENV},service.namespace=${GAME},service.name=
OTEL_EXPORTER_OTLP_ENDPOINT=http://flow-grafana-agent.grafana:4317
OTEL_SDK_DISABLED=fase
```

- 替换变量 `${ENV}` 为你要配置的环境，只能是 staging 或者 production.
- 替换变量 `${GAME}` 为你要配置的游戏名，比如 yowapeda, transformers, etc.
- 替换变量 `${SVRNAME}` 为你要配置的服务名，比如 gameserver, login_server or center

- 例子。在生产环境 production 里，为游戏 transfomrers 的游戏服 gameserver 配置：

```
OTEL_EXPORTER_OTLP_PROTOCOL=http/protobuf
OTEL_RESOURCE_ATTRIBUTES=service.version=production,service.namespace=transformers,serv
OTEL_EXPORTER_OTLP_ENDPOINT=http://flow-grafana-agent.grafana:4317
OTEL_SDK_DISABLED=fase
```

2. 在 Java 程序启动行加载 opentelemetry-java-instrumentation

- 下载最新的 opentelemetry-java-instrumentation.jar 文件。下载链接
- 添加 opentelemetry-java-instrumentation.jar 到程序启动中。

```
java -javaagent:path/to/opentelemetry-javaagent.jar \
    -jar myapp.jar
```

比如某游戏，配置 opentelemetry-java-instrumentation 后的实际启动命令行如下：

```
kubectl exec -it pod/gameserver1-75cbb79f94-r7nfz -- ps ww 1
  PID TTY      STAT   TIME COMMAND
    1 ?        Ssl    3:19 java -javaagent:opentelemetry-javaagent.jar -XX:MaxRA
```