

模块九：Operator AIOps 实战

王炜 / 前腾讯云 CODING 高级架构师

目录

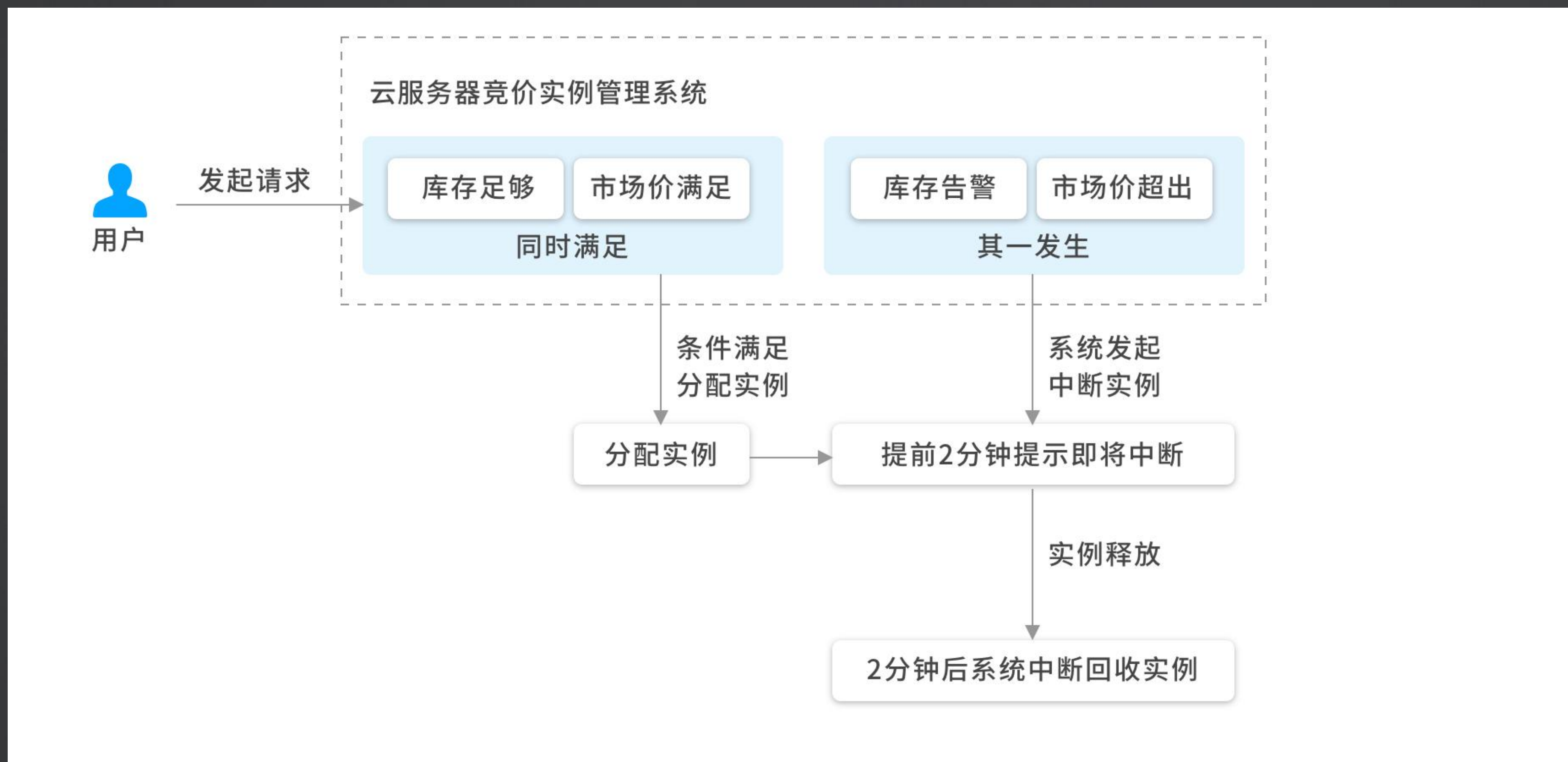
- 1 实战一：开发 Operator 调度 GPU 竞价实例资源池
- 2 实战二：Operator 实现大模型私有部署（Kong API + 多 GPU 负载均衡）
- 3 实战三：开发基于 LLM 的日志流监测 Operator
- 4 实战四：利用 RAGflow 实现基于运维专家知识库故障排查 Operator

1. 实战一：开发 Operator 调度 GPU 竞价实例资源池

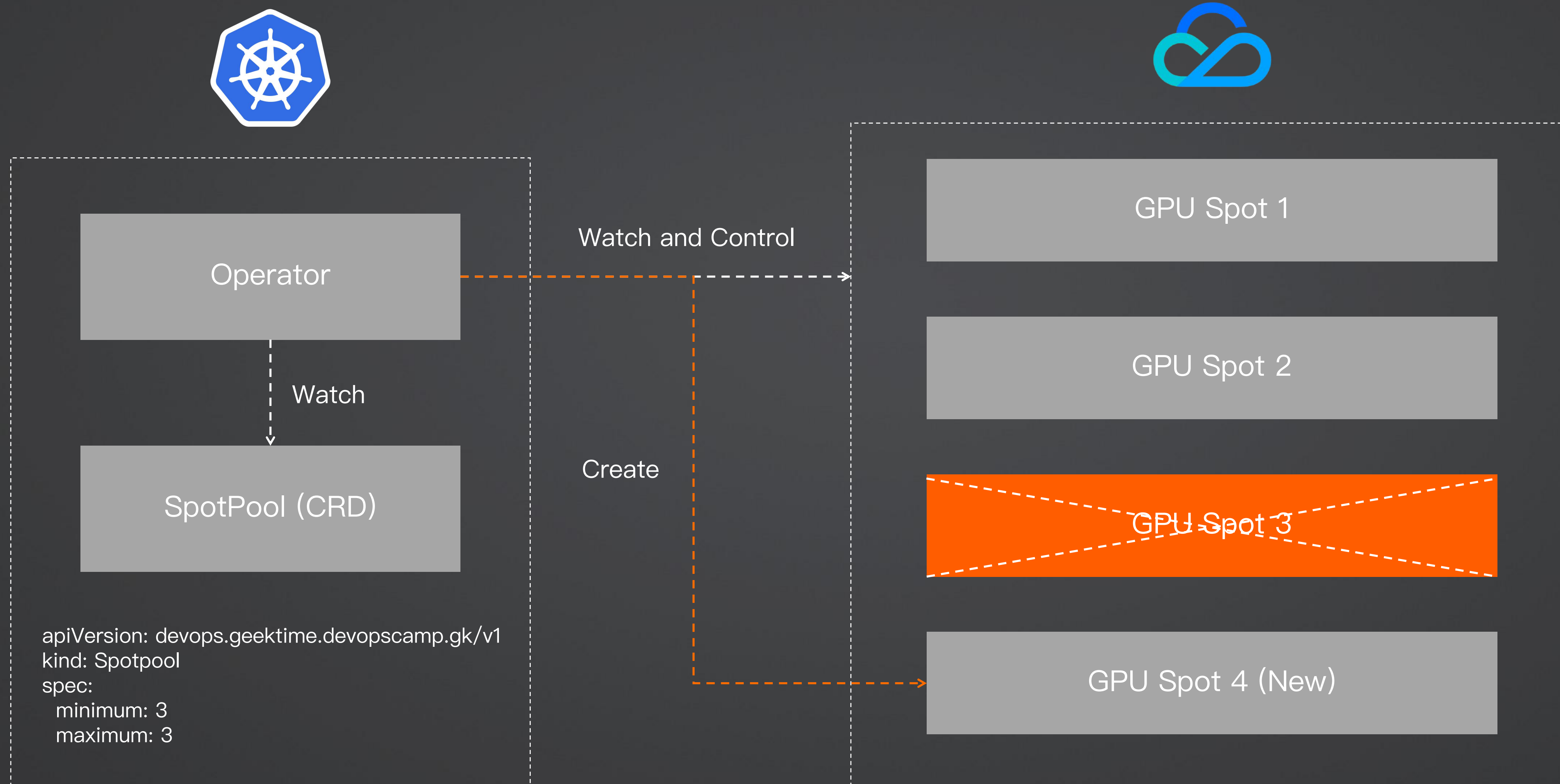
竞价实例价格优势

| GN7.2XLARGE32 | 按量计费 | 竞价实例（参考） |
|---------------------|--------|------------|
| 8C32G, GPU T4-16G | 8.68/H | 1.736/H |
| 系统盘 (50G 云盘) | 0.02 | 0.02 |
| 宽带 (1Mbps 按宽带计费) | 0.06 | 0.06 |
| 收费合计 | 8.76 | 1.816（8 倍） |

GPU 竞价实例



维持 GPU 资源池



本地创建集群

- 使用 Kind 创建本地集群
- <https://kind.sigs.k8s.io/docs/user/quick-start/>
 - `kind create cluster`
- 安装 kubebuilder: <https://book.kubebuilder.io/quick-start#installation>

CRD 设计

- Group: devops.geektime.devopscamp.gk/v1
- Version: v1
- Kind: SpotPool
- 通过 minimum 和 maximum 参数来维持资源池
GPU 实例数量

```
spotpool.yml
1  apiVersion: devops.geektime.devopscamp.gk/v1
2  kind: Spotpool
3  metadata:
4    labels:
5      name: spotpool-sample
6  spec:
7    secretId: AKIDwkKhpiSyHL0Yy8wcSG54z3UxDG53tWv
8    secretKey: 70gLEHJPjw1tX2PYhhliYIonX8VefS6
9    region: ap-singapore
10   availabilityZone: ap-singapore-2
11   instanceType: "GN7.2XLARGE32"
12   minimum: 2
13   maximum: 2
14   subnetId: subnet-n09geves
15   vpcId: vpc-o3jhu9dn
16   securityGroupIds:
17     - sg-dm9bzpug
18   imageId: img-e734psbk
19   instanceChargeType: SPOTPAID
```


技术细节

- 接入腾讯云 SDK 实现创建竞价实例
 - <https://cloud.tencent.com/document/sdk/Go>
- 借助 Operator RequeueAfter 实现定时检查资源池实例数量是否满足预期

步骤

- `mkdir spotpool && cd spotpool`
- `go mod init github.com/lyzhang1999/spotpool`
- `kubebuilder init --domain=devopscamp.gk`
- `kubebuilder create api --group devops.geektime --version v1 --kind Spotpool`
- 完善 `spotpool/api/v1/spotpool_types.go`
 - 修改 `SpotpoolSpec`
- 生成 CRD: `make manifests`, 查看 `config/crd/bases/devops.geektime.devopscamp.gk_spotpools.yaml` 文件
- 编写 `internal/controller/spotpool_controller.go` Reconcile 业务逻辑
- 将 CRD 安装到集群: `make install`
- 运行 Operator: `make run`
- 编辑 sample 并部署: `kubectl apply -f config/samples/devops.geektime_v1_spotpool.yaml`

2. 实战二：Operator 实现大模型私有部署 (Kong API + 多 GPU 负载均衡)

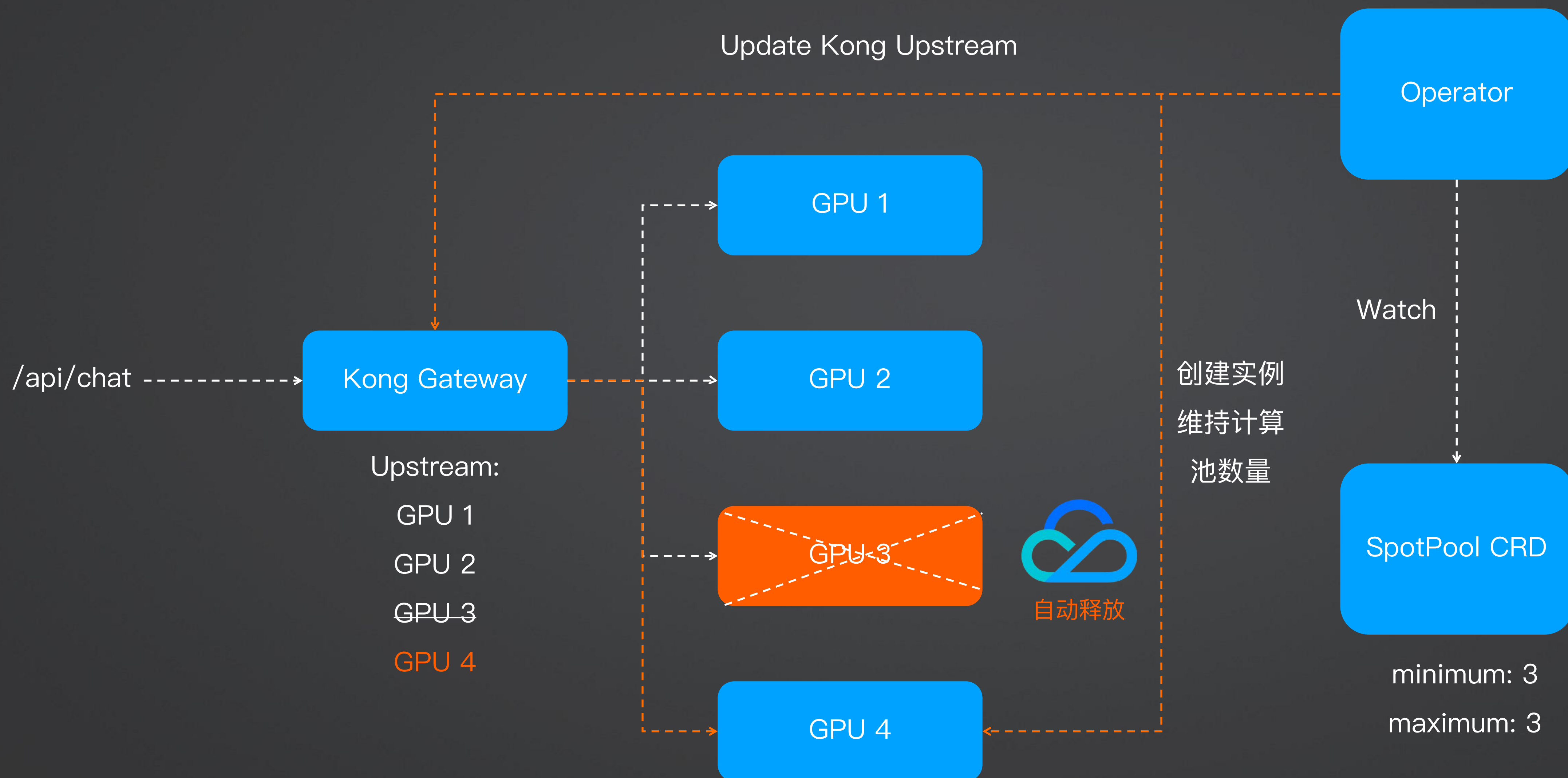
Ollama

- 借助 Ollama 私有部署大模型
- /api/chat 接口兼容 OpenAI 数据结构
- 单实例推理较慢，借助网关实现多 Ollama 实例负载均衡
- 网关选型：Kong、Nginx 等

```
ollama.sh
1  curl http://localhost:11434/api/chat -d '{
2      "model": "llama3.1",
3      "messages": [
4          {
5              "role": "user",
6              "content": "why is the sky blue?"
7          }
8      ]
9  }'
```

```
> ollama run qwen2.5:0.5b
pulling manifest
pulling c5396e06af29... 100% 397 MB
pulling 66b9ea09bd5b... 100% 68 B
pulling eb4402837c78... 100% 1.5 KB
pulling 832dd9e00a68... 100% 11 KB
pulling 005f95c74751... 100% 490 B
verifying sha256 digest
writing manifest
success
>>> 你是谁?
我是阿里云自主研发的超大规模语言模型，我叫通义千问。作为一个AI助手，我的目标是提供客观、中立和有益的信息，帮助用户更好地理解 and 解决问题。如果您有任何问题或需求，请随时告诉我，我会尽力为您提供帮助。同时，如果您有特定的话题需要讨论，也可以随时提问。
```

Ollama 多实例负载均衡部署



新 GPU 实例初始化问题

- 新实例初始化
 - 下载 Ollama 及其依赖
 - 下载模型
 - 启动 Ollama 服务: ollama serve
- 使用自定义虚拟机镜像代替启动脚本初始化

Ollama 自启动问题

- Service 方式部署，实现自启动
- 创建 /etc/systemd/system/ollama.service
- 启动服务
 - sudo systemctl daemon-reload
 - sudo systemctl enable ollama
- 将配置好的虚拟机制作自定义镜像，拉起新的 GPU 实例时指定该镜像 ID 即可

```
ollama.sh
1 [Unit]
2 Description=Ollama Service
3 After=network-online.target
4
5 [Service]
6 ExecStart=/usr/bin/ollama serve
7 User=ollama
8 Group=ollama
9 Restart=always
10 RestartSec=3
11 Environment="PATH=$PATH"
12
13 [Install]
14 WantedBy=default.target
```

步骤：制作 Ollama 镜像

- IaC 创建虚拟机
- 执行 Shell 脚本安装 Ollama
- 创建 Service，并启动
- 预加载模型
- 关机，制作自定义镜像，得到镜像 ID

步骤：部署 Kong 网关

- IaC 创建虚拟机
- 安装 Docker
- 修改 Kong admin_listen 和 admin_gui_listen 配置，监听在 0.0.0.0 端口
 - kong/docker-compose/docker-compose.yaml
 - 暴露 API，便于 Operator 控制
- 以 Docker Compose 方式启动 Kong Gateway

构建 Operator 步骤

- 基于 demo_1 继续完善
- CRD 增加 kongGatewayIP，便于 Operator 请求 Kong API 控制动态负载均衡列表
- 修改 Reconcile，实现 GPU 实例和 Kong 负载均衡同步

效果

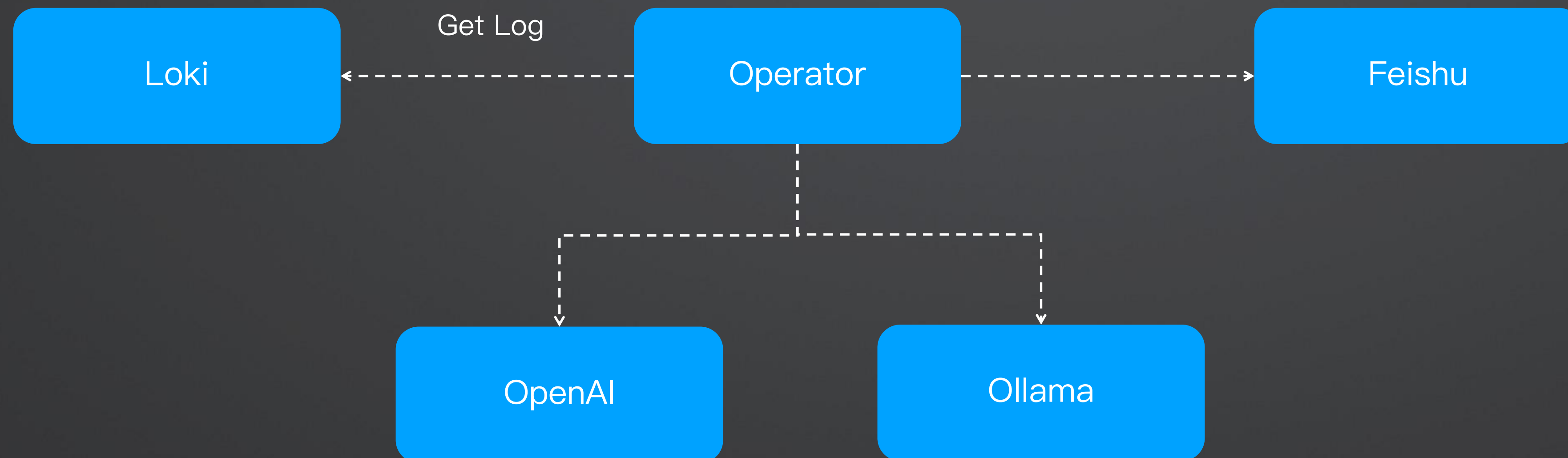
```
ollama.sh
1  curl http://${kong_ip}/ollama-chat -d '{
2    "model": "qwen2:0.5b",
3    "messages": [
4      {
5        "role": "user",
6        "content": "你是谁? "
7      }
8    ]
9  }'
```

- 当请求 Kong 网关时，流量被转发到多台 GPU 实例进行推理
- 实现大规模 LLM 私有部署和负载均衡

3. 实战三：开发基于 LLM 的日志流监测 Operator

LogPilot: LLM 日志流检测 Operator

- 从 Loki 获取错误日志
- 将日志发送至大模型并获取建议
- 将严重的问题发送至飞书通知



CRD 设计


- Group: log.aiops.com
- Version: v1
- Kind: LogPilot
- 从指定的 Loki URL 获取日志
- 将日志发送至指定的模型服务商（商业模型或自托管模型）

```
logpilot.yml
1 apiVersion: log.aiops.com/v1
2 kind: LogPilot
3 metadata:
4   labels:
5     name: logpilot-sample
6 spec:
7   lokiURL: "http://43.154.108.115:31000"
8   lokiPromQL: '{app="payment"} != `ERROR`'
9   llmEndpoint: "https://vip.apiyi.com/v1"
10  llmToken: "sk-6zmAX27TiKbpRefB14Ad5387A2504492B66c9e5b02628655"
11  llmModel: "gpt-4o"
12  feishuWebhook: "https://open.feishu.cn/open-apis/bot/v2/hook/d5e267dc-a92f-43d3-bc45-106b5e718c49"
13
```

步骤

- `mkdir logpilot && cd logpilot`
- `go mod init github.com/lyzhang1999/llm-log-operator`
- `kubebuilder init --domain=aiops.com`
- `kubebuilder create api --group log --version v1 --kind LogPilot`
- 修改: `api/v1/logpilot_types.go` `LogPilotSpec`
- 修改: `internal/controller/logpilot_controller.go` 增加相关业务逻辑
- 拉起 Prometheus Stack, 获得 Loki IP
- 修改 `config/samples/log_v1_logpilot.yaml` 并部署

效果

 自定义机器人 机器人 | 通过webhook将自定义服务的消息推送至飞书 9月18日 20:15

以下是日志信息的分析：

错误等级分析

1. **Memory OOM**

- 时间: 2024-09-18 12:14:18.772 及 2024-09-18 12:13:58.758
- 描述: 容器 'payment-service' 超过了内存限制
- 错误等级: 严重

2. **服务 500 错误**

- 时间: 2024-09-18 12:14:16.771 及 2024-09-18 12:13:56.757
- 描述: 下游服务 'order-processing' 返回了状态码 500
- 错误等级: 严重 [feishu]
- 建议: 立即检查并修复 'order-processing' 服务，以确保支付服务的正常运行。

3. **数据库连接失败**

- 时间: 2024-09-18 12:14:14.769 及 2024-09-18 12:13:54.756
- 描述: 无法连接到数据库 '[db.payment.local](#)'
- 错误等级: 致命 [feishu]
- 建议: 马上检查数据库连接配置和数据库服务状态，以恢复支付服务。

4. **欺诈检测失败**

- 时间: 2024-09-18 12:14:08.765
- 描述: 付款被标记为潜在欺诈
- 错误等级: 重要，但非致命
- 建议: 检查欺诈检测算法和相关数据，确保准确性和系统安全性。

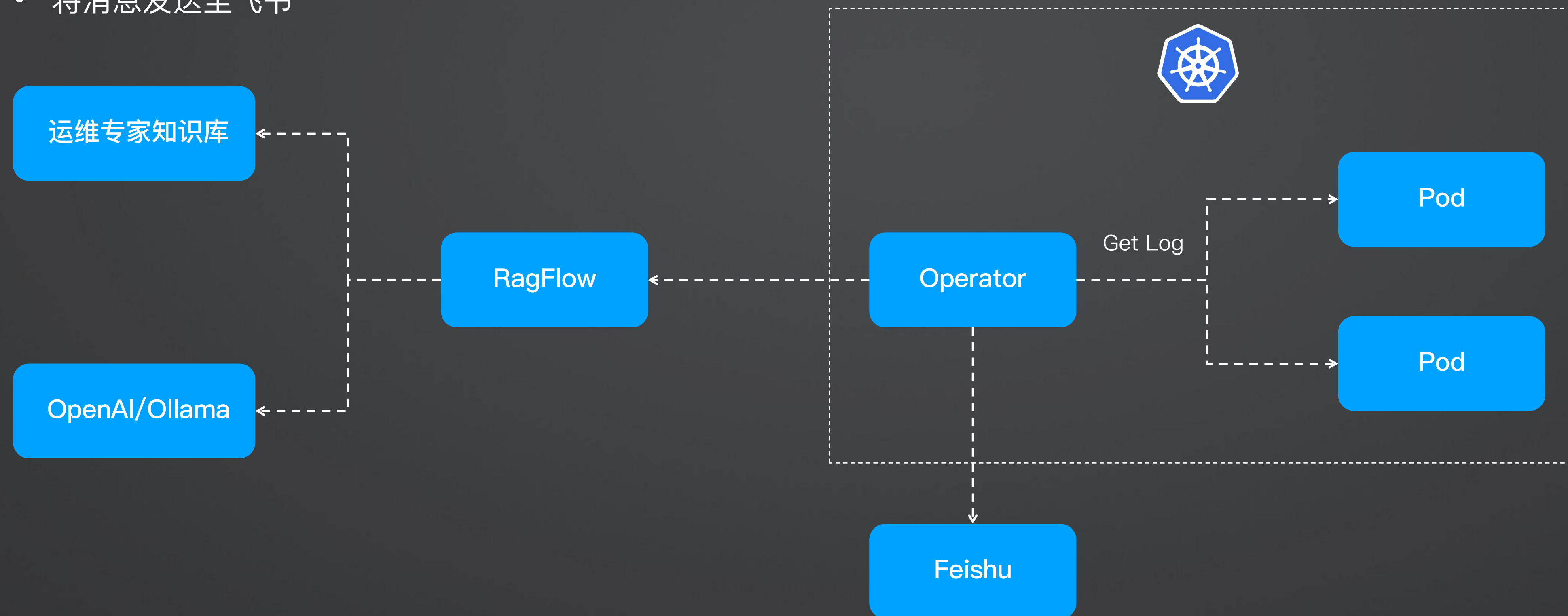
总结与建议

- **内存超限**: 尝试优化 payment-service 服务的内存使用，或增加其内存配额。
- **服务 500 错误**: 立即检查并修复 order-processing 服务。 [feishu]
- **数据库连接失败**: 马上检查数据库连接配置和数据库服务状态。 [feishu]
- **欺诈检测失败**: 检查欺诈检测算法和数据准确性。

4. 实战四：利用 RAGflow 实现基于运维专家 知识库的智能故障排查 Operator

架构设计

- 直接获取 Pod 日志
- 将日志发送至 RagFlow 基于内部运维专家知识库获取解决方案
- 将消息发送至飞书



CRD 设计

- Group: log.aiops.com
- Version: v1
- Kind: RagLogPilot
- 获取命名空间下所有 Pod 的日志
- 将日志发送至 RagFlow API 获取解决方案

```
logpilot.yml
1  apiVersion: log.aiops.com/v1
2  kind: RagLogPilot
3  metadata:
4    labels:
5    name: raglogpilot-sample
6  spec:
7    workloadNamespace: default
8    ragFlowEndpoint: "http://101.32.208.76/v1/api"
9    ragFlowToken: "ragflow-U1MTk10Tc1N2E1NTEtZWY5NjZkMDI0Mm"
10
```

内部运维知识库示例

内部运维知识库和解决方案手册

1. Database connection failed: 数据库连接失败, 请检查数据库是否正常运行, 数据库连接配置是否正确 (账号密码)
2. Service 500 Error: 下游服务 500 错误, 请找对应的服务负责人:
 1. order-processing 服务: 小王
 2. payment-processing 服务: 小李
 3. user-processing 服务: 小张
3. Memory OOM: 内存溢出, 请检查应用内存使用情况, 是否存在内存泄漏, 如有必要, 请联系小王提高容器内存限制
4. Fraud detection failed Payment flagged as potentially fraudulent: 欺诈检测失败, 请联系小李检查欺诈检测服务是否正常

RagFlow 核心 API

- 创建对话
 - GET /api/new_conversation?user_id=xxx
 - data.id = conversation_id
- 基于运维知识库获取回复
 - POST /api/completion
 - conversation_id、messages、stream=false
 - 回复: data.answer

步骤

- `mkdir raglogpilot && cd raglogpilot`
- `go mod init github.com/lyzhang1999/rag-log-operator`
- `kubebuilder init --domain=aiops.com`
- `kubebuilder create api --group log --version v1 --kind RagLogPilot`
- 修改: `api/v1/raglogpilot_types.go` `RagLogPilotSpec`
- 修改: `internal/controller/raglogpilot_controller.go` 增加相关业务逻辑
- 拉起 RagFlow 并创建内部运维知识库
- 修改 `config/samples/log_v1_raglogpilot.yaml` 并部署

效果

根据提供的日志和运维知识库，以下是针对每个错误的解答：

1. **Database connection failed: Unable to connect to database at 'db.payment.local'**
 - 解决方案：数据库连接失败，请检查数据库是否正常运行，数据库连接配置是否正确（账号密码）。
2. **Service 500 Error: Downstream service 'order-processing' returned status code 500**
 - 解决方案：下游服务 `order-processing` 返回了状态码 500 错误，请联系对应的服务负责人小王。
3. **Memory OOM: Container 'payment-service' exceeded memory limit**
 - 解决方案：内存溢出，请检查应用内存使用情况，是否存在内存泄漏，如有必要，请联系小王提高容器内存限制。
4. **Fraud detection failed: Payment flagged as potentially fraudulent**
 - 知识库中未找到您要的答案！

总结：

- 对于数据库连接失败的问题，请检查数据库运行状态和连接配置。
- 对于 `order-processing` 服务返回的 500 错误，请联系小王。
- 对于内存溢出问题，请检查应用内存使用情况，并可能需要联系小王提高容器内存限制。
- 关于欺诈检测失败的问题，知识库中没有相关信息。

课后作业

- 尝试修改实战三，并接入 Ollama 实现自托管大模型（Qwen2）推理（选做）
- 修改实战四，配置 RagFlow 接入 Ollama 实现自托管大模型推理

THANKS