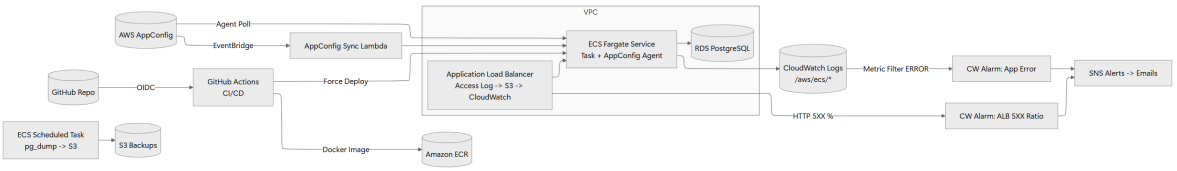
poper devops 交付文档

POPER DevOps 交付文档

1. 项目概览

- **应用**：Laravel 11 / PHP 8.4，容器化并运行在 Amazon ECS (Fargate)。
- **托管区域**：`ap-northeast-1`（东京）。
- **基础设施**：使用 Terraform 管理（目录 `infra/terraform`），包括 VPC、ECS、ALB、RDS、AppConfig、日志/告警等。
- **CI/CD**：GitHub Actions（`.github/workflows/cicd.yml`）负责测试、构建镜像、推送到 ECR，并触发 ECS 滚动发布。
- **配置与变更**：AWS AppConfig 作为运行时配置源，通过 sidecar agent + Lambda 事件保持 ECS 配置实时同步。

2. 架构图



3. 需求对照

模块	条款	状态	说明 / 证据
PHP 部署	#1 ECS(Fargate) 托管 Laravel	✅	`infra/terraform/ecs.tf`, `Dockerfile`
	#2 首页打印环境变量	✅	`src/routes/web.php`, `src/app/Services/AppConfigRepository.php`
	#3 JSON 访问日志含 TraceId 并入 CloudWatch	✅	`src/app/Http/Middleware/RequestLoggingMiddleware.php` + awslogs
	#4 PHP/PHP-FPM 日志入 CloudWatch	✅	使用CloudWatch记录容器日志
	#5 ALB + Access Log → CloudWatch	✅*	已在控制台配置，后续可将 S3 + Lambda (`infra/lambda/alb_logs_to_cw`) 纳入 IaC
	#6 定时扩缩容	✅	`aws_appautoscaling_scheduled_action`

模块	条款	状态	说明 / 证据
	#7 RDS + pg_dump 备份	✅	`infra/terraform/rds.tf`, `infra/terraform/backup.tf`
CI/CD	#1 main push 触发 release	✅	`.github/workflows/cicd.yml` + `softprops/action-gh-release`
	#2 构建镜像推送 ECR	✅	同 workflow
	#3 最新镜像部署 ECS	✅	`aws ecs update-service --force-new-deployment`
	#4 成功/失败通知	✅	SNS 邮件
环境变量 & AppConfig	#1 配置交付至 ECS	✅	AppConfig + agent + repository 读取
	#2 Release 后自动同步	✅	AppConfig 事件 → `appconfig_sync` Lambda → `ecs.update_service`
	#3 Release 通知	✅	Lambda 发布 SNS
日志/监控/告警	#1 全量日志收集	✅	应用/VPC/备份日志 + ALB Access Log*(手动)
	#2 ECS 指标 Dashboard	✅*	控制台手动创建
	#3 ALB HTTP 状态 Dashboard	✅*	控制台手动创建
	#4 日志 ERROR 告警	✅	`infra/terraform/monitoring.tf` log metric filter + alarm
	#5 ALB 500 >10% 告警	✅	`infra/terraform/monitoring.tf` metric math alarm
文档	#1 架构图	✅	第 2 节
	#2 核心操作流程	✅	第 4~7 节
提交包	GitHub 仓库、AWS 链接、告警截图	✅	第8节

> 带 * 的条目当前在控制台手动完成，S3的terraform配置文件应用后提示网络错误，多次重试后未成功

4. 基础设施操作流程

1. **初始化**

```
cd infra/terraform
terraform init
```

2. **规划/审查**

```
terraform plan -var-file="terraform.tfvars"
```

3. **部署**

```
terraform apply -var-file="terraform.tfvars"
```

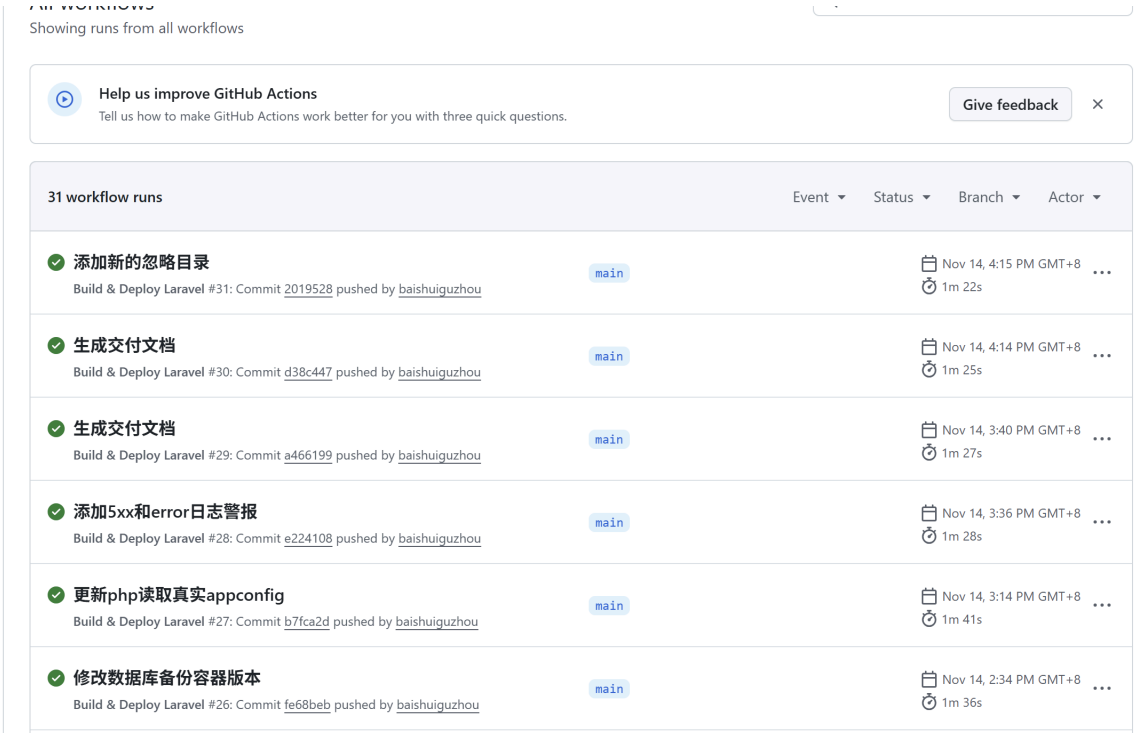
```
Apply complete! Resources: 6 added, 1 changed, 1 destroyed.

Outputs:

alb_dns_name = "poper-devops-dev-alb-1426047183.ap-northeast-1.elb.amazonaws.com"
alb_security_group_id = "sg-0adb7a300cd710e24"
alerts_topic_arn = "arn:aws:sns:ap-northeast-1:827602716863:poper-devops-dev-alerts"
appconfig_application_id = "p0jvhbk"
appconfig_configuration_profile_id = "he8yh14"
appconfig_environment_id = "acgls07"
appconfig_lambda_name = "poper-devops-dev-appconfig-sync"
caller_account_id = "827602716863"
ecr_repository_name = "poper-devops-dev-app"
ecr_repository_url = "827602716863.dkr.ecr.ap-northeast-1.amazonaws.com/poper-devops-dev-app"
ecs_cluster_name = "poper-devops-dev-cluster"
ecs_security_group_id = "sg-09ba989e3c66a5de9"
ecs_service_name = "poper-devops-dev-service"
ecs_task_execution_role_arn = "arn:aws:iam::827602716863:role/poper-devops-dev-ecs-exec"
github_actions_role_arn = "arn:aws:iam::827602716863:role/poper-devops-dev-github-oidc"
nat_gateway_ids = {
  "ap-northeast-1a" = "nat-00fcfcf9ce14a1aa9"
  "ap-northeast-1c" = "nat-07c5a7cbac5d80c30"
}
private_subnet_ids = {
  "ap-northeast-1a" = "subnet-09401c152c6e9d94f"
  "ap-northeast-1c" = "subnet-077e89364937c92b5"
}
```

```
rds_backup_bucket_name = "poper-devops-dev-rds-backups-60b9d0"
rds_endpoint = "poper-devops-dev-postgres.ctu6wosse0lu.ap-northeast-1.rds.amazonaws.com"
rds_security_group_id = "sg-0bb6d1c68fceebed6"
region = "ap-northeast-1"
vpc_flow_log_group_name = "/aws/vpc/poper-devops-dev-flow"
vpc_id = "vpc-020e390a7b2f5e8ce"
```

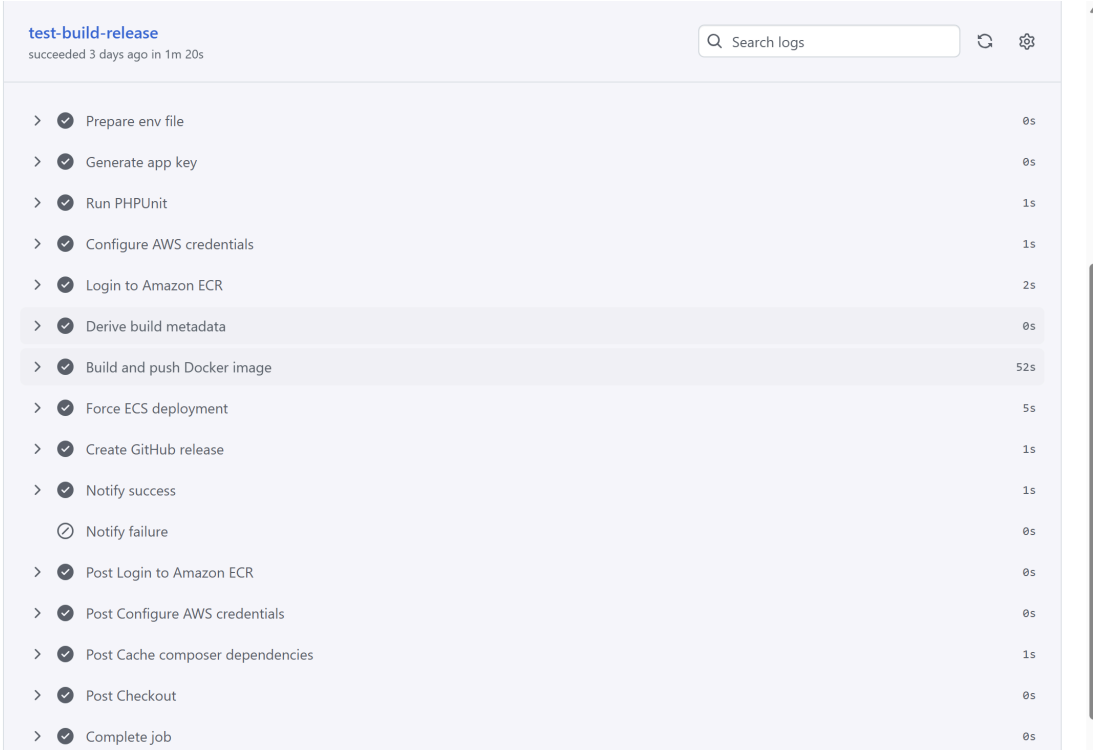
4. **输出** `terraform output` 可获取常用信息（ALB DNS、ECR 仓库、ECS 服务名、AppConfig IDs 等）。
5. **注意事项**
   - `terraform.tfvars` 中的 `app_key`、数据库凭据、S3 bucket 名称需按账户实际值更新。

5. 应用发布流程（CI/CD）

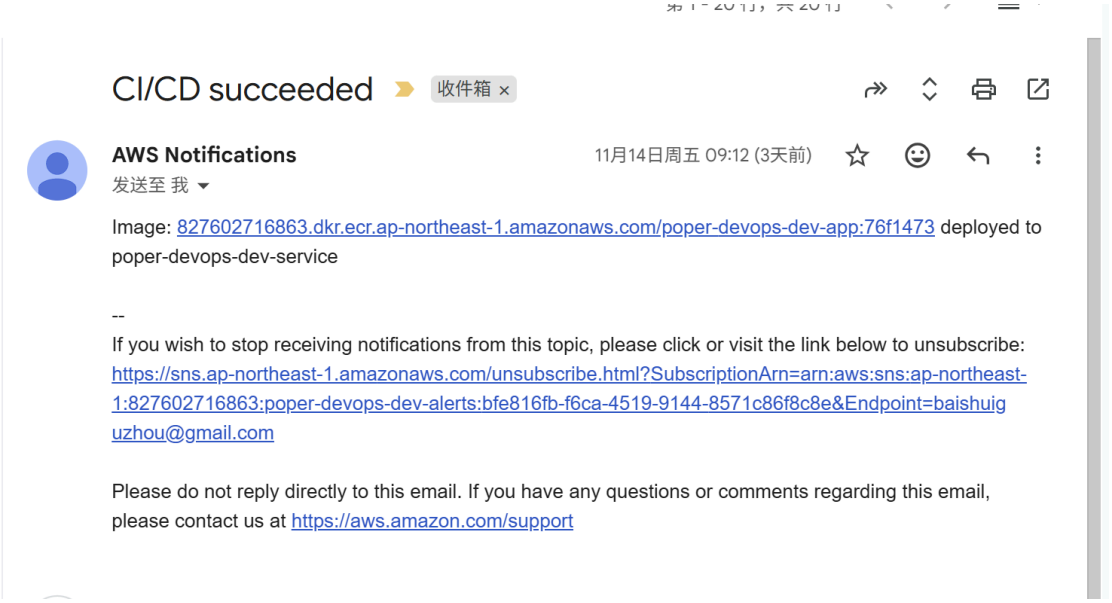
1. **开发者提交**到 `main`：触发 GitHub Actions。

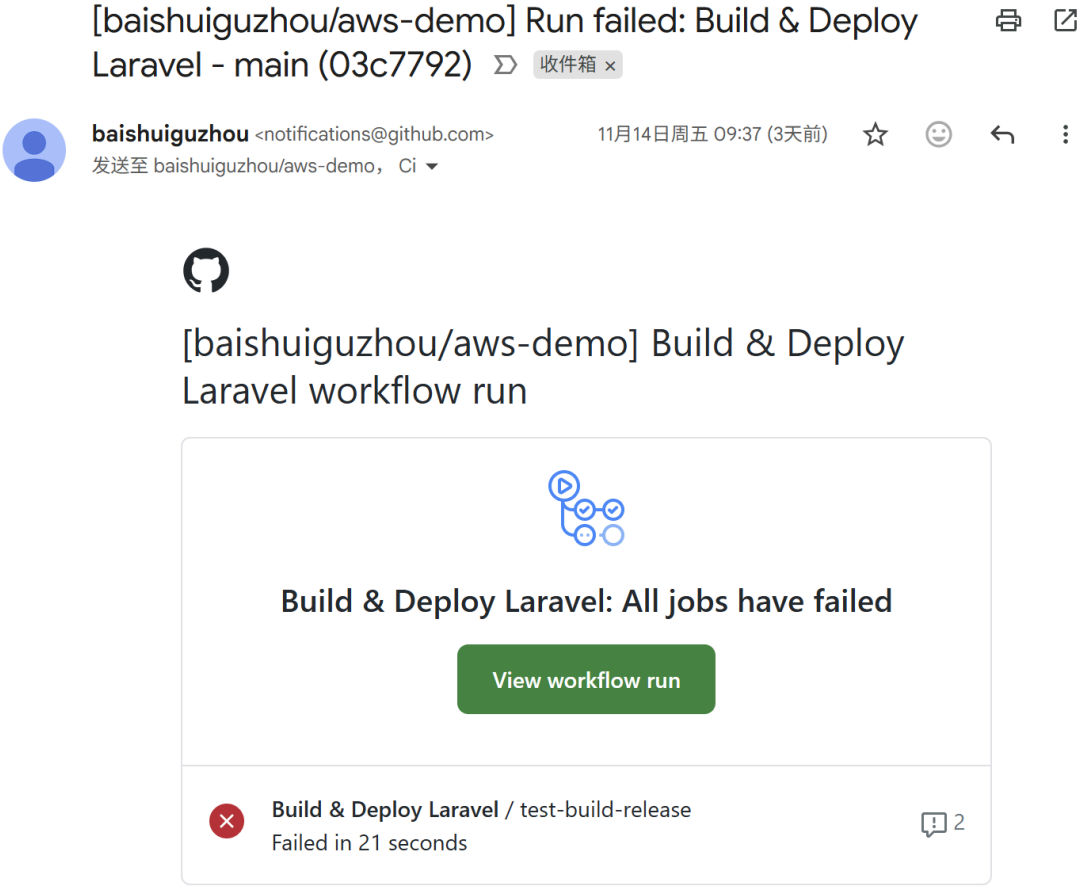


2. **Workflow 阶段：**
   - 安装 PHP 依赖、生成 `.env`、运行 `php artisan test`。
   - 通过 OIDC AssumeRole 登录 AWS、登录 ECR。
   - 构建 Docker 镜像，打上 `latest` + `SHORT_SHA` 双 tag，并推送。
   - 调用 `aws ecs update-service --force-new-deployment`，强制任务拉取最新镜像。



- 创建 GitHub Release，成功/失败分别 SNS 邮件通知。





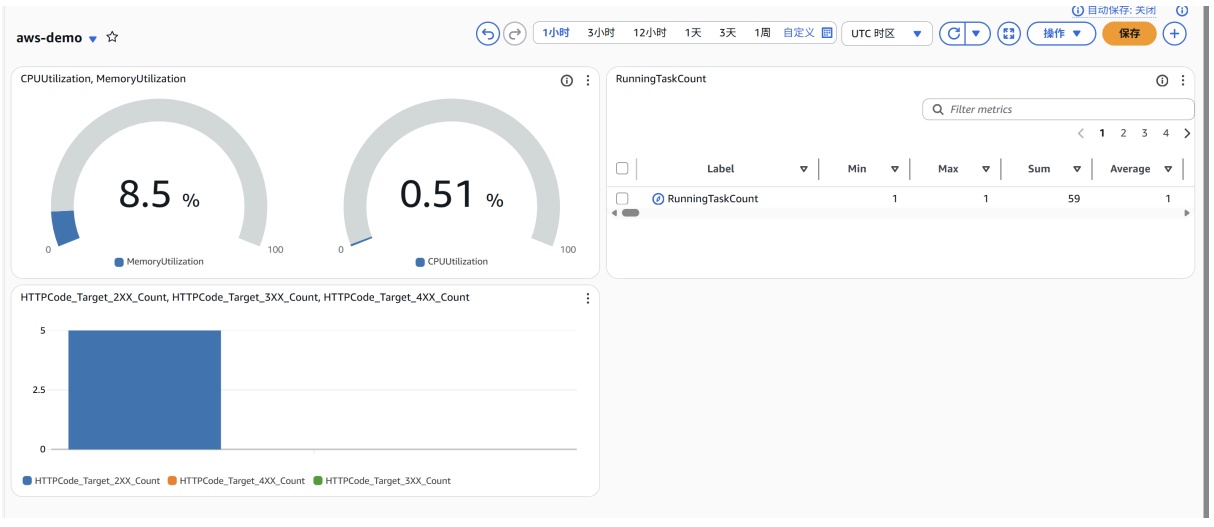
3. **回滚**：可通过 GitHub Release 选择旧版本 tag 并重新部署，或在 Terraform 中指定 `app_image` 固定镜像 URI 后 `ecs.update_service`。

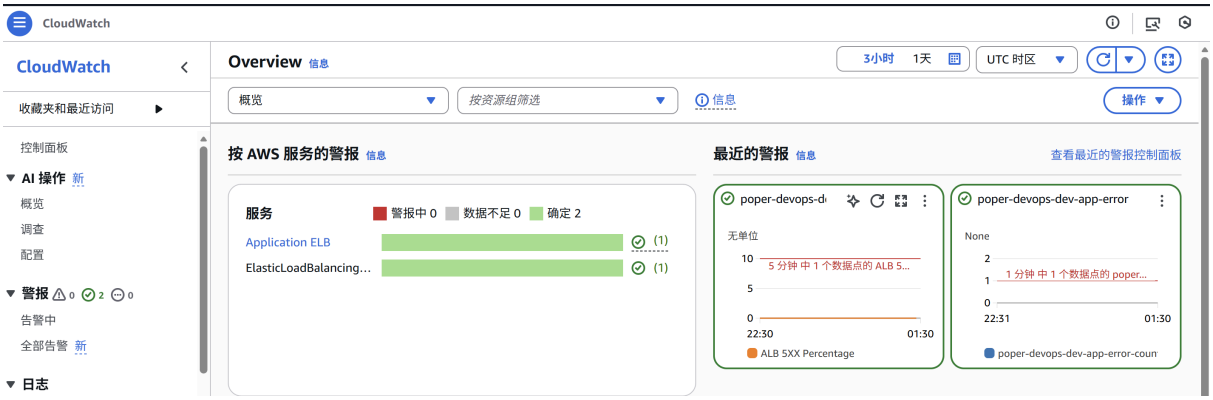
6. AppConfig & 运行时配置

1. **配置位置**：`infra/appconfig/config.json`（初始值）或 AWS AppConfig 控制台。
2. **生效路径**：
   - AppConfig Hosted Config → AppConfig Agent sidecar → Laravel `AppConfigRepository` 读取 `homepage_env` / `feature_message`。
   - AppConfig 新版本发布 → 触发 EventBridge 事件 → `appconfig_sync` Lambda → `ecs.update_service` → ECS 滚动替换任务。
3. **操作步骤**：
   - 在 AppConfig 中新增配置版本，选择现有 Deployment Strategy。
   - 发布时填写推出目标（environment），观察 Lambda 日志/SNS 通知确认同步成功。
   - 如需强制刷新，可在 ECS 服务上手动 “Force New Deployment”。

7. 监控 & 告警

类型	位置	说明
CloudWatch Logs	`/aws/ecs/<project>-app`、`/aws/ecs/<project>-backup`、`/aws/vpc/<project>-flow`	应用、备份任务、VPC 流日志
Log Metric Filter	`${local.name_prefix}-app-error-filter`	匹配日志中的 `ERROR` 字符串
Alarm - App Error	`${local.name_prefix}-app-error`	60s 内出现 ERROR 即触发，通知 SNS
Alarm - ALB 5XX Ratio	`${local.name_prefix}-alb-5xx-ratio`	`HTTPCode_ELB_5XX_Count / RequestCount > 10%` 时告警
Dashboard	*控制台手动创建*	包含 ECS CPU/Mem、ALB HTTP 状态等图表
ALB Access Log	*控制台手动创建*	ALB → S3 → Lambda（`infra/lambda/alb_logs_to_cw`）将日志写入 CloudWatch

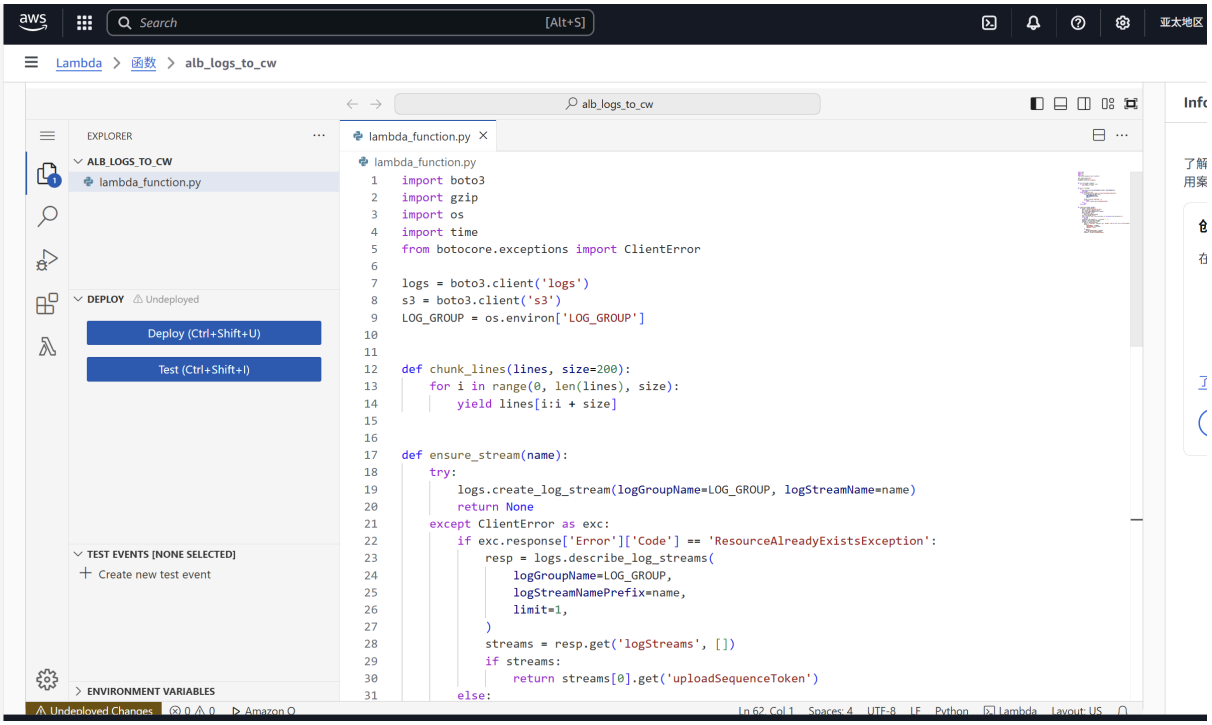




> 告警邮箱：te.qi+wangkui@poper.co, hu.zhangjie+wangkui@poper.co 等，详见 terraform.tfvars。

8. 链接与截图（交付时补充）

项目	链接 / 截图
GitHub 仓库	baishuiguzhou/aws-demo
Terraform State / S3	S3 存储桶 | S3 | ap-northeast-1
ALB (Console)	负载均衡器 | EC2 | ap-northeast-1
ECS Service	集群 | Elastic Container Service | ap-northeast-1
ECR Repository	Elastic Container Registry - Private repositories
AppConfig Application	服务任务 | Elastic Container Service | ap-northeast-1
RDS 实例	数据库 | Aurora and RDS | ap-northeast-1
CloudWatch Dashboards	控制面板 | aws-demo | CloudWatch | ap-northeast-1
ALB Access Log / Lambda 配置截图	




```python
import boto3
import gzip
import os
import time
from botocore.exceptions import ClientError

logs = boto3.client('logs')
s3 = boto3.client('s3')
LOG_GROUP = os.environ['LOG_GROUP']


def chunk_lines(lines, size=200):
    for i in range(0, len(lines), size):
        yield lines[i:i + size]


def ensure_stream(name):
    try:
        logs.create_log_stream(logGroupName=LOG_GROUP, logStreamName=name)
        return None
    except ClientError as exc:
        if exc.response['Error']['Code'] == 'ResourceAlreadyExistsException':
            resp = logs.describe_log_streams(
                logGroupName=LOG_GROUP,
                logStreamNamePrefix=name,
                limit=1,
            )
            streams = resp.get('logStreams', [])
            if streams:
                return streams[0].get('uploadSequenceToken')
        else:
```