

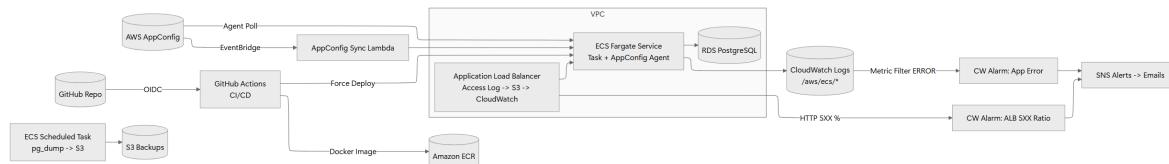
# poper devops 交付文档

## 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	✓	<code>infra/terraform/rds.tf</code> , <code>infra/terraform/backup.tf</code>
CI/CD	#1 main push 触发 release	✓	<code>.github/workflows/cicd.yml</code> + <code>softprops/action-gh-release</code>
	#2 构建镜像推送 ECR	✓	同 workflow
	#3 最新镜像部署 ECS	✓	<code>aws ecs update-service --force-new-deployment</code>
	#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 → <code>appconfig_sync</code> Lambda → <code>ecs.update_service</code>
	#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	✓	<code>infra/terraform/monitoring.tf</code> log metric filter + alarm
	#5 ALB 500 >10% 告警	✓	<code>infra/terraform/monitoring.tf</code> 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。

The screenshot shows the GitHub Actions 'All Workflows' page. At the top, there's a banner asking for feedback on GitHub Actions. Below it, a table lists 31 workflow runs. Each row contains a green checkmark icon, the workflow name, the commit details, the branch (main), the execution time (e.g., Nov 14, 4:15 PM GMT+8, 1m 22s), and a three-dot menu icon.

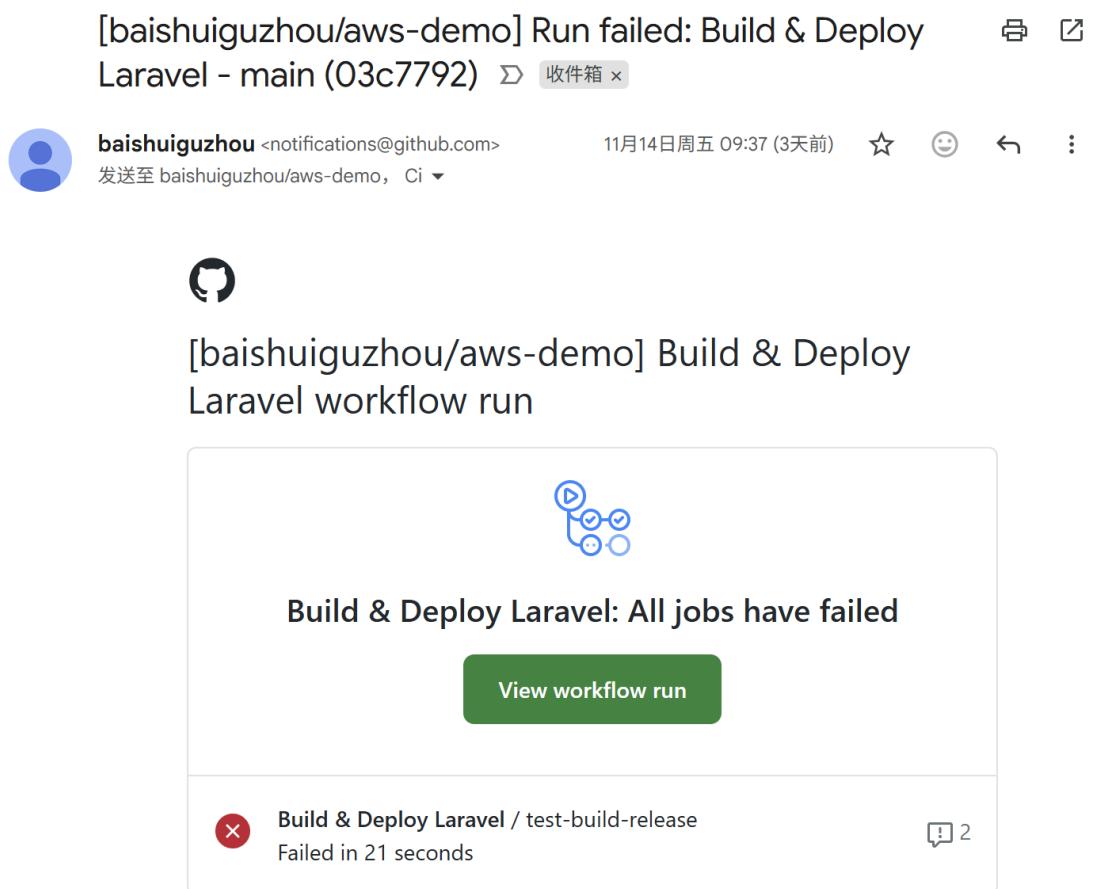
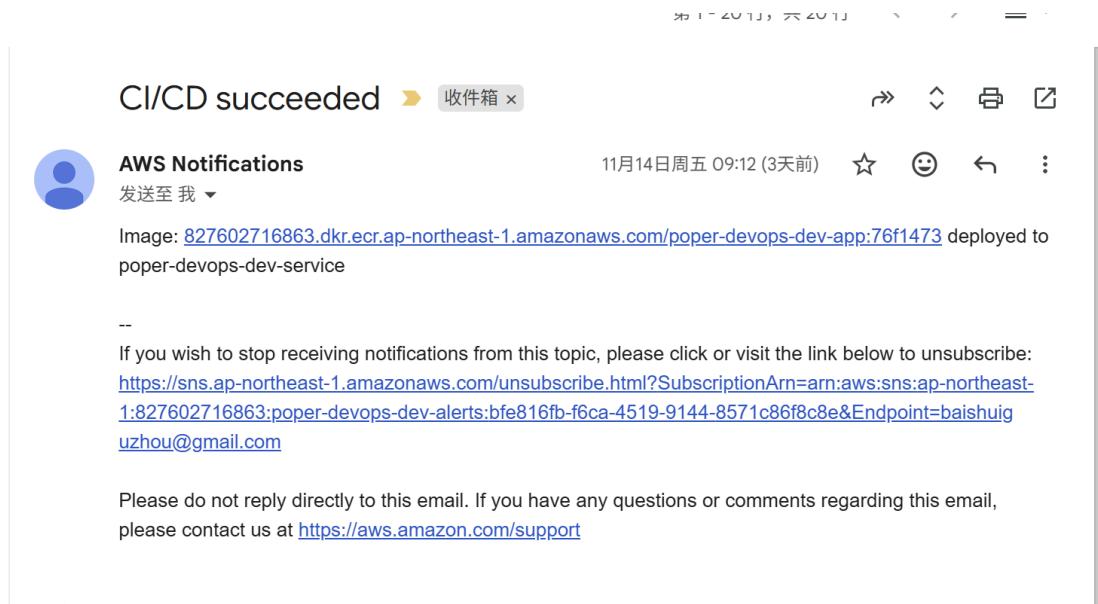
31 workflow runs		Event	Status	Branch	Actor
✓ 添加新的忽略目录	Build & Deploy Laravel #31: Commit <a href="#">2019528</a> pushed by <a href="#">baishuiguzhou</a>	main	Nov 14, 4:15 PM GMT+8	1m 22s	...
✓ 生成交付文档	Build & Deploy Laravel #30: Commit <a href="#">d38c447</a> pushed by <a href="#">baishuiguzhou</a>	main	Nov 14, 4:14 PM GMT+8	1m 25s	...
✓ 生成交付文档	Build & Deploy Laravel #29: Commit <a href="#">a466199</a> pushed by <a href="#">baishuiguzhou</a>	main	Nov 14, 3:40 PM GMT+8	1m 27s	...
✓ 添加5xx和error日志警报	Build & Deploy Laravel #28: Commit <a href="#">e224108</a> pushed by <a href="#">baishuiguzhou</a>	main	Nov 14, 3:36 PM GMT+8	1m 28s	...
✓ 更新php读取真实appconfig	Build & Deploy Laravel #27: Commit <a href="#">b7fca2d</a> pushed by <a href="#">baishuiguzhou</a>	main	Nov 14, 3:14 PM GMT+8	1m 41s	...
✓ 修改数据库备份容器版本	Build & Deploy Laravel #26: Commit <a href="#">fe68beb</a> pushed by <a href="#">baishuiguzhou</a>	main	Nov 14, 2:34 PM GMT+8	1m 36s	...

## 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。

The screenshot shows a GitHub Action run named 'test-build-release'. It succeeded 3 days ago in 1m 20s. The log is divided into sections: 'Prepare env file' (0s), 'Generate app key' (0s), 'Run PHPUnit' (1s), 'Configure AWS credentials' (1s), 'Login to Amazon ECR' (2s), 'Derive build metadata' (0s), 'Build and push Docker image' (52s), 'Force ECS deployment' (5s), 'Create GitHub release' (1s), 'Notify success' (1s), 'Notify failure' (0s), 'Post Login to Amazon ECR' (0s), 'Post Configure AWS credentials' (0s), 'Post Cache composer dependencies' (1s), 'Post Checkout' (0s), and 'Complete job' (0s).

- 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:

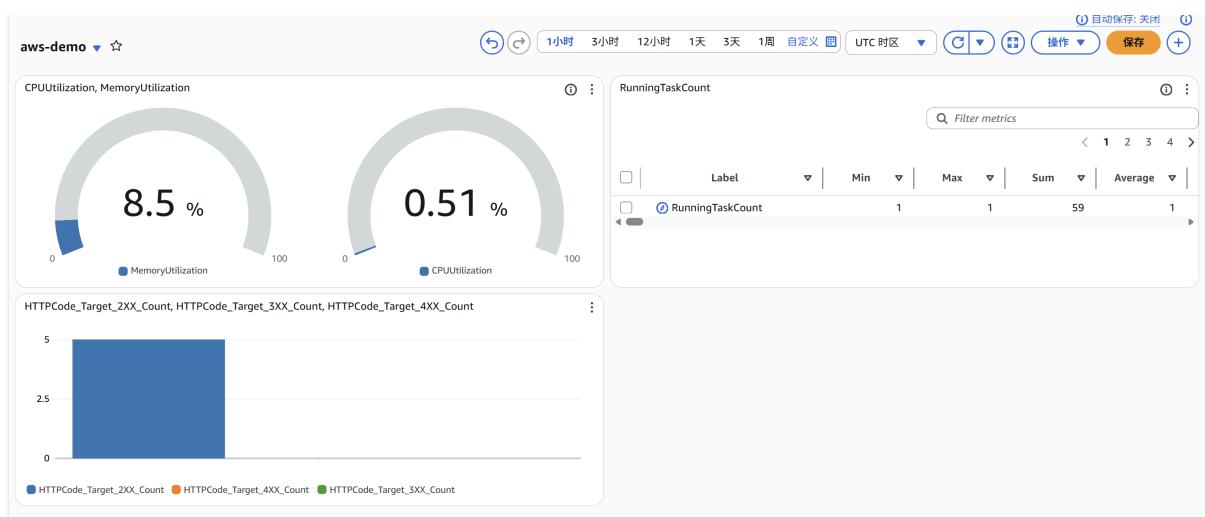
- AppConfig Hosted Config → AppConfig Agent sidecar → Laravel `AppConfigRepository` 读取 `homepage_env` / `feature_message`。
- AppConfig 新版本发布 → 触发 EventBridge 事件 → `appconfig_sync` Lambda → `ecs.update_service` → ECS 滚动替换任务。

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	<code>/aws/ecs/&lt;project&gt;-app</code> 、 <code>/aws/ecs/&lt;project&gt;-backup</code> 、 <code>/aws/vpc/&lt;project&gt;-flow</code>	应用、备份任务、VPC 流日志
Log Metric Filter	<code> \${local.name_prefix}-app-error-filter</code>	匹配日志中的 <code>ERROR</code> 字符串
Alarm - App Error	<code> \${local.name_prefix}-app-error</code>	60s 内出现 <code>ERROR</code> 即触发, 通知 SNS
Alarm - ALB 5XX Ratio	<code> \${local.name_prefix}-alb-5xx-ratio</code>	<code>HTTPCode_ELB_5XX_Count / RequestCount &gt; 10%</code> 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 ( <code>infra/lambda/alb_logs_to_cw</code> ) 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	<a href="#">baishuiguzhou/aws-demo</a>
Terraform State / S3	<a href="#">S3 存储桶   S3   ap-northeast-1</a>
ALB (Console)	<a href="#">负载均衡器   EC2   ap-northeast-1</a>
ECS Service	<a href="#">集群   Elastic Container Service   ap-northeast-1</a>
ECR Repository	<a href="#">Elastic Container Registry - Private repositories</a>
AppConfig Application	<a href="#">服务任务   Elastic Container Service   ap-northeast-1</a>
RDS 实例	<a href="#">数据库   Aurora and RDS   ap-northeast-1</a>
CloudWatch Dashboards	<a href="#">控制面板   aws-demo   CloudWatch   ap-northeast-1</a>
ALB Access Log / Lambda 配置截图	

```
alb_logs_to_cw
lambda_function.py

1 import boto3
2 import gzip
3 import os
4 import time
5 from botocore.exceptions import ClientError
6
7 logs = boto3.client('logs')
8 s3 = boto3.client('s3')
9 LOG_GROUP = os.environ['LOG_GROUP']
10
11
12 def chunk_lines(lines, size=200):
13     for i in range(0, len(lines), size):
14         yield lines[i:i + size]
15
16
17 def ensure_stream(name):
18     try:
19         logs.create_log_stream(logGroupName=LOG_GROUP, logStreamName=name)
20     except ClientError as exc:
21         if exc.response['Error']['Code'] == 'ResourceAlreadyExistsException':
22             resp = logs.describe_log_streams(
23                 logGroupName=LOG_GROUP,
24                 logStreamNamePrefix=name,
25                 limit=1,
26             )
27             streams = resp.get('logStreams', [])
28             if streams:
29                 return streams[0].get('uploadSequenceToken')
30
31 else:
```