【CI/CD 实践】 Github Actions 配置 CI/CD 自动发布 docker 镜像

作者：Allenxuxu
原文链接：https://ld246.com/article/1585835652680
来源网站：链滴

Github Actions 是 Github 内置的 CI/CD 工具，现在已经对所有的开源项目免费开放了。

本文主要记录使用 Github Actions 实践 CI/CD 的一些配置。

功能目标

- 代码静态检查
- 代码单元测试
- release/tag 时自动 build 镜像并推送到 docker hub

项目 Dockerfile 和 Makefile

项目主要目录

```
.
├── LICENSE
├── Makefile
├── README.md
├── config-srv
│   ├── Makefile
│   └── main.go
├── deployments
│   ├── docker
│   │   ├── config-srv
│   │   │   └── Dockerfile
├── go.mod
├── go.sum
```

- config-srv 目录：服务代码
- deployments 目录：所有服务的 Dockerfile
- Makefile 顶层 Makefile：build Docker 镜像

我们先看下顶层的 Makefile

```
.PHONY: config-srv
config-srv:
    docker build -f deployments/docker/config-srv/Dockerfile . -t config-srv
```

我们可以在后面配置的 Github Actions 配置文件中执行 make config-srv，这样就会执行 docker build -f deployments/docker/config-srv/Dockerfile . -t config-srv ，构建一个 docker 镜像。

接下来，看一下 deployments/docker/config-srv/Dockerfile ，配置 Dockerfile 多阶段构建 。

```
FROM golang:1.13-alpine as builder
WORKDIR /root
COPY ./  ./
RUN export GO111MODULE=on && CGO_ENABLED=0 GOOS=linux go build -o build/config-srv config-srv/main.go


FROM alpine:latest
```

```
RUN apk --no-cache add ca-certificates
WORKDIR /root
COPY --from=builder /root/build/config-srv ./

ENTRYPOINT ["/root/config-srv"]
```

Github Actions 具体配置

commit 提交自动 CI

```
name: CI
on:
  push:
    branches:
      - master
    paths-ignore:
    - 'README.md'
  pull_request:
    branches:
      - master
    paths-ignore:
    - 'README.md'
jobs:
  lint: # 使用golangci-lint 进行静态检查
    name: Lint
    runs-on: ubuntu-latest
    steps:
    - name: Set up Go 1.13
      uses: actions/setup-go@v1
      with:
        go-version: 1.13
      id: go
    - name: Code
      uses: actions/checkout@v1
    - name: Intsall Golangci-lint
      run: curl -sfL https://raw.githubusercontent.com/golangci/golangci-lint/master/install.sh|
h -s -- -b . latest
    - name: Lint
      run: ./golangci-lint run --skip-dirs=".git|.github|dashboard|doc" --timeout=5m

  test:
    name: Unit Testing # go test
    runs-on: ${{ matrix.os }}
    strategy:
      matrix:
        os: [ubuntu-latest]      # 选择系统类型
    steps:
    - name: Set up Go 1.13
      uses: actions/setup-go@v1
      with:
        go-version: 1.13
      id: go
    - name: Code
```

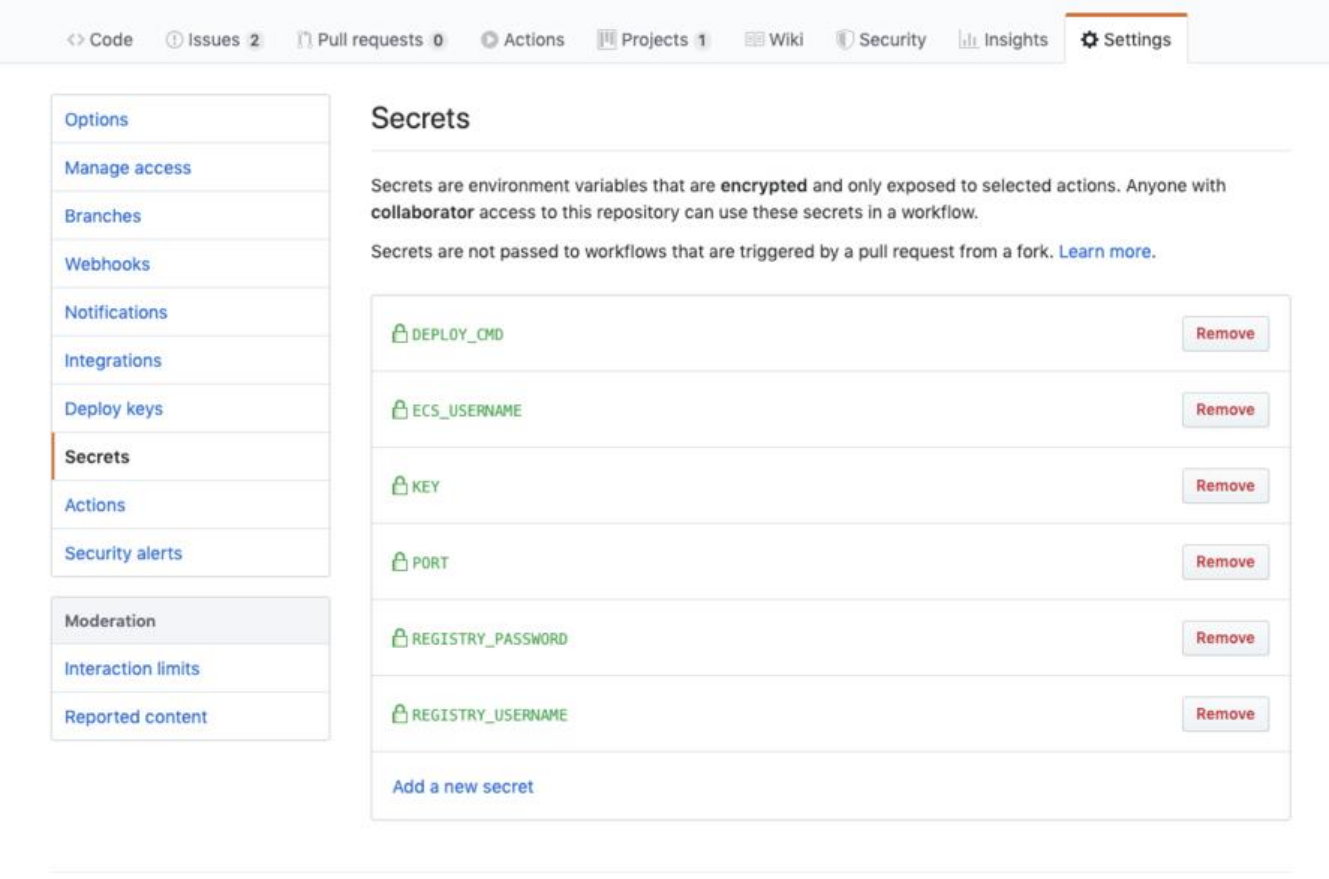
```
      uses: actions/checkout@v1
    - name: Go Get dependencies
      run: go get -v -t -d ./...
    - name: Go Test
      run: go test -v ./...
```

release/tag 时自动 CI/CD

```
name: Release
on:    # 限定在 master 分支 release  操作时触发
  release:
    types: [published]
    branches:
      - master
jobs:
  lint:  # 静态检查
    name: Lint
    runs-on: ubuntu-latest
    steps:
      - name: Set up Go 1.13
        uses: actions/setup-go@v1
        with:
          go-version: 1.13
        id: go
      - name: Code
        uses: actions/checkout@v1
      - name: Intsall Golangci-lint
        run: curl -sfL https://raw.githubusercontent.com/golangci/golangci-lint/master/install.sh|
sh -s -- -b . latest
      - name: Lint
        run: ./golangci-lint run --skip-dirs=".git|.github|dashboard|doc" --timeout=5m

  test:
    name: Unit Testing   # 单元测试
    runs-on: ${{ matrix.os }}
    strategy:
      matrix:
        os: [ubuntu-latest]
    steps:
      - name: Set up Go 1.13
        uses: actions/setup-go@v1
        with:
          go-version: 1.13
        id: go
      - name: Code
        uses: actions/checkout@v1
      - name: Go Get dependencies
        run: go get -v -t -d ./...
      - name: Go Test
        run: go test -v ./...

  docker:  # build docker 镜像并且推送
    needs: [lint, test]
    name: docker build and push
```

```
    runs-on: ubuntu-latest
    steps:
      - name: Code
        uses: actions/checkout@v1
      - name: Set env
        run: echo ::set-env name=RELEASE_VERSION::$(echo ${GITHUB_REF:10}) # 取tag名称，面作为 docker 的 tag
      - name: tag
        run: echo ${{ env.RELEASE_VERSION }}
      - name: config-srv  # 执行代码仓库中的 Makefile ， 在 Makefile 中执行 docker build 操作
        run: |
          make config-srv
          docker login docker.io -u ${{ secrets.REGISTRY_USERNAME }} -p ${{ secrets.REGISTRY_PSSWORD }}
          docker tag config-srv:latest ${{ secrets.REGISTRY_USERNAME }}/config-srv:latest
          docker tag config-srv:latest ${{ secrets.REGISTRY_USERNAME }}/config-srv:${{ env.RELESE_VERSION }}

          docker push ${{ secrets.REGISTRY_USERNAME }}/config-srv:latest
          docker push ${{ secrets.REGISTRY_USERNAME }}/config-srv:${{ env.RELEASE_VERSION }}
```

上述的 ${{ secrets.REGISTRY_USERNAME }} 等环境变量，皆在 Github - settings - secrets 中设。登陆 docker hub 所需要的账号密码或者其他敏感信息都在此设置，避免明文写在 yaml 中。

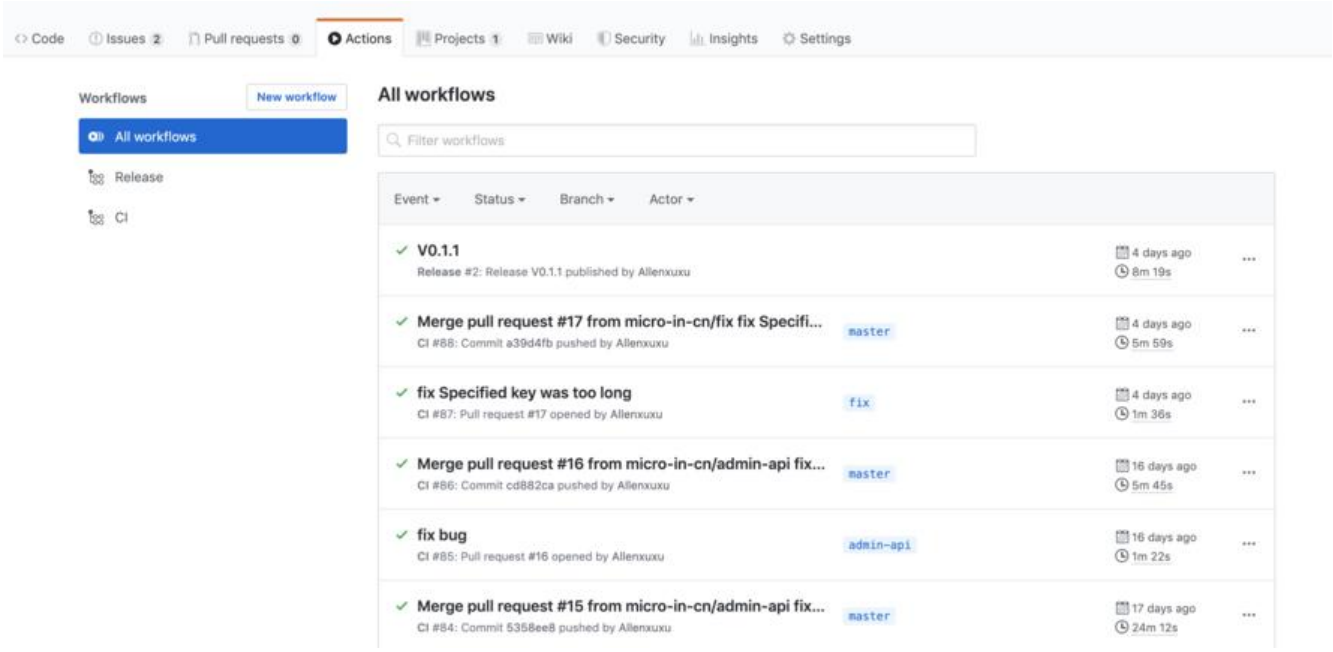


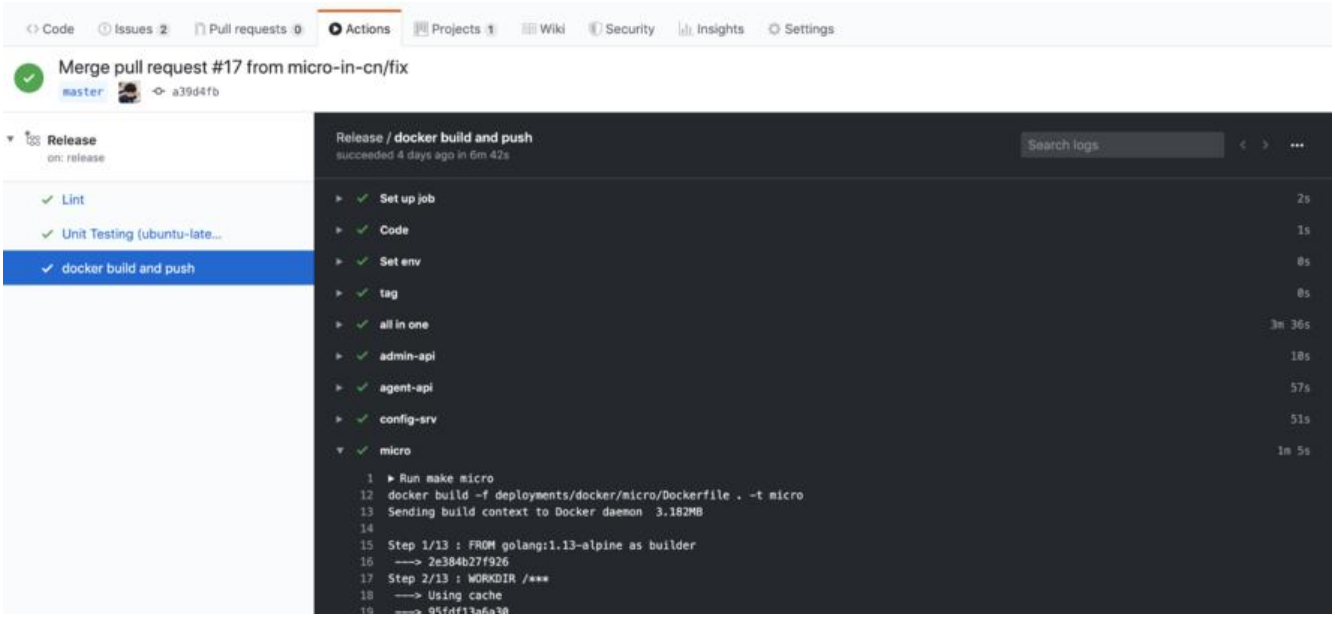
在 Github 的项目仓库顶层 .github/workflows 文件夹放置两个 yml 文件即可。每次 commit 提交会触发 ci.yml 中的配置逻辑，当 release 发布时就会触发 release.yml 中的配置逻辑。

.github

```
└── workflows
    ├── ci.yml
    └── release.yml

1 directory, 2 files
```

触发的 pipeline 可以在项目主页 Tab 栏点击 Actions 查看。





本文 Makefile ， Dockefile ， CI 配置都节取自 XConf 项目，可以去仓库查看完整配置。

相关实践项目

- XConf : 基于 go-micro 构建的分布式配置中心。
- Gev: 一个轻量、快速的基于 Reactor 模式的非阻塞 Golang TCP 网络库，支持自定义协议，轻松速搭建高性能服务器。