Federico Guerinoni
Software developer

GitHub Actions

Develer TPS

23/09/2020

develer

INTRODUZIONE

GH Actions permette di automatizzare molti processi manuali da una semplice build, alla gestione delle PR o il rilascio di un applicativo con l'esecuzione di tests.

PERCHÈ

- Quando usarlo?
- Alternative?

CONCETTI

- .github
- .yml
- Workflow
- Action
- Runner

STRUTTURA

- name
- on
- job
- step
- matrix
- env

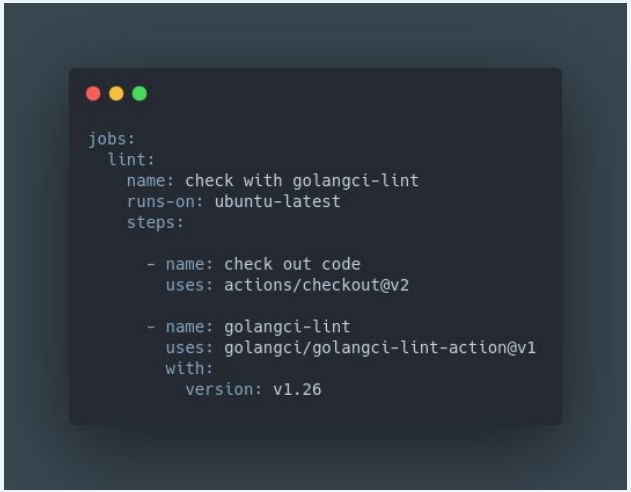
name

Questo tag rappresenta il nome del workflow, spesso e' lo stesso nome del file ed e' quello che da browser viene visualizzato nella sezione actions.

on

Impostazione di trigger del workflow in questione.
Accetta regexp per filtrare i tags ed anche i branches.

job



```yaml
jobs:
  lint:
    name: check with golangci-lint
    runs-on: ubuntu-latest
    steps:

      - name: check out code
        uses: actions/checkout@v2

      - name: golangci-lint
        uses: golangci/golangci-lint-action@v1
        with:
          version: v1.26
```

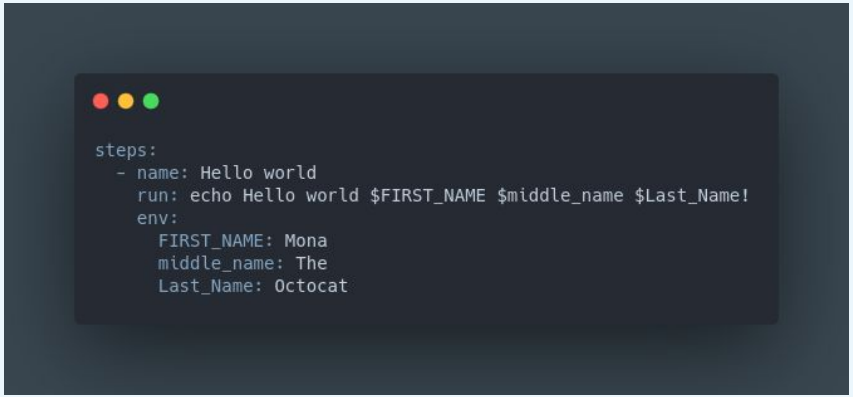
Sotto questo tag si possono configurare piu' jobs, i quali possono avere piu' actions.

matrix

```yaml
jobs:
  build:
    strategy:
      fail-fast: false
      matrix:
        os: [ubuntu-latest, macos-latest, windows-latest]
        include:
          - os: ubuntu-latest
            artifact_name: replacer
            asset_name: replacer-linux-amd64
          - os: windows-latest
            artifact_name: replacer.exe
            asset_name: replacer-windows-amd64
          - os: macos-latest
            artifact_name: replacer
            asset_name: replacer-macos-amd64

    runs-on: ${{ matrix.os }}
    steps:
    - name: Set up Go 1.x
      uses: actions/setup-go@v2
      with:
        go-version: "1.14"

    - name: Check out code into the Go module directory
      uses: actions/checkout@v2

    - name: Build
      run: go build -ldflags="-X 'main.version=${{ github.ref }}'" -o replacer ./cmd/replacer/

    - name: Upload binaries
      uses: svenstaro/upload-release-action@v1-release
      with:
        repo_token: ${{ secrets.GITHUB_TOKEN }}
        file: ${{ matrix.artifact_name }}
        asset_name: ${{ matrix.asset_name }}
        tag: ${{ github.ref }}
```

Esempio di matrice per eseguire le stesse actions con piu' configurazioni.

env



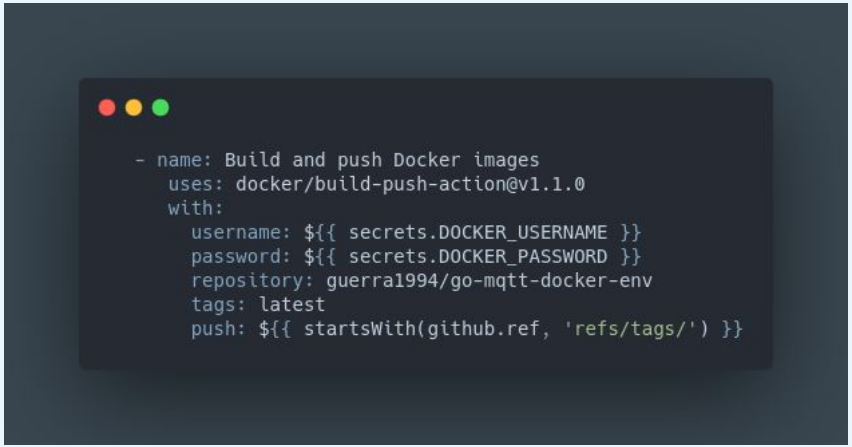
```yaml
steps:
  - name: Hello world
    run: echo Hello world $FIRST_NAME $middle_name $Last_Name!
    env:
      FIRST_NAME: Mona
      middle_name: The
      Last_Name: Octocat
```

In questo esempio possiamo vedere come configurare le variabili di ambiente dove lo scope e' ridotto a questo step.

CONTAINER

- supporto a docker-hub
- ambiente per il workflow
- container registry

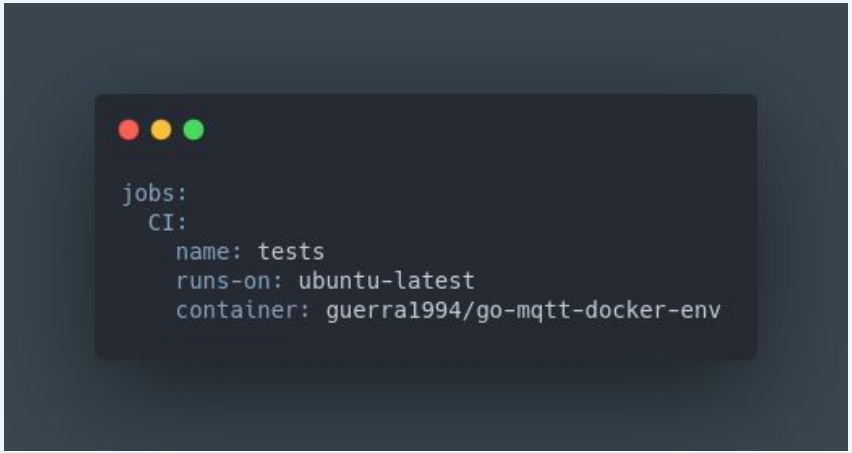
supporto a docker-hub



```yaml
- name: Build and push Docker images
  uses: docker/build-push-action@v1.1.0
  with:
    username: ${{ secrets.DOCKER_USERNAME }}
    password: ${{ secrets.DOCKER_PASSWORD }}
    repository: guerra1994/go-mqtt-docker-env
    tags: latest
    push: ${{ startsWith(github.ref, 'refs/tags/') }}
```

Actions che permette la compilazione e il push su docker-hub se trova un tag.

ambiente per il workflow



Per il job “CI” verranno eseguite le actions all’interno di un container lanciato dall’immagine indicata.

container registry

```yaml
obs:
  GHCR:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2

    - name: Build
      run: docker build --build-arg "host_uid=$(id -u)" --build-arg "host_gid=$(id -g)" -t buildroot-
env .

    - name: Login to GitHub Container Registry
      uses: docker/login-action@master
      with:
        registry: ghcr.io
        username: ${{ github.repository_owner }}
        password: ${{ secrets.CR_PAT }}

    - name: Publish
      run: |
        docker tag buildroot-env ghcr.io/guerinoni/buildroot-env:latest
        docker push ghcr.io/guerinoni/buildroot-env
```

Con questa action e' possibile compilare e pushare il package in github container registry sfruttando il secret configurato.

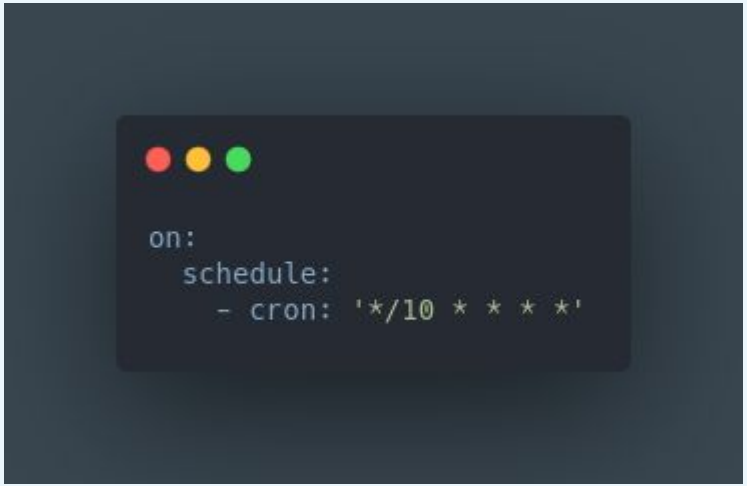
Configurazione minima

-> push	-> build.yml
-> pull_request	-> tests.yml
-> tag	-> release.yml

Bonus Actions

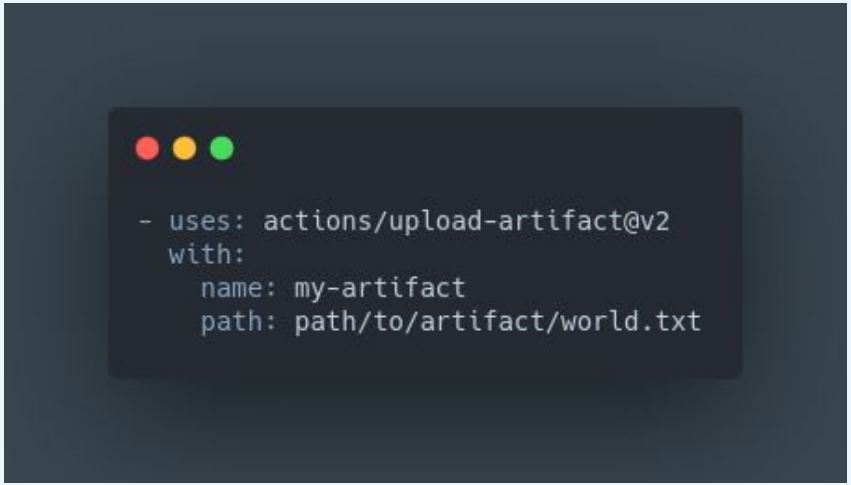
- trigger programmato
- caricamento artefatti
- label automatica
- merge automatico
- notifica slack
- coverage

trigger programmato



Sintassi di crontab supportata per programmare l'esecuzione di un workflow.

caricamento artefatti



Action che carica il file indicato nelle release integrate nel repository di GitHub.

label automatica

```yaml
name: labeler

on: [pull_request]

jobs:
  label:
    runs-on: ubuntu-18.04
    steps:
      - uses: actions/checkout@v2
      - name: Labeler
        uses: docker://decathlon/pull-request-labeler-action:2.0.0
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
          CONFIG_PATH: ${{ secrets.GITHUB_WORKSPACE }}/.github/label-pr.yml
```

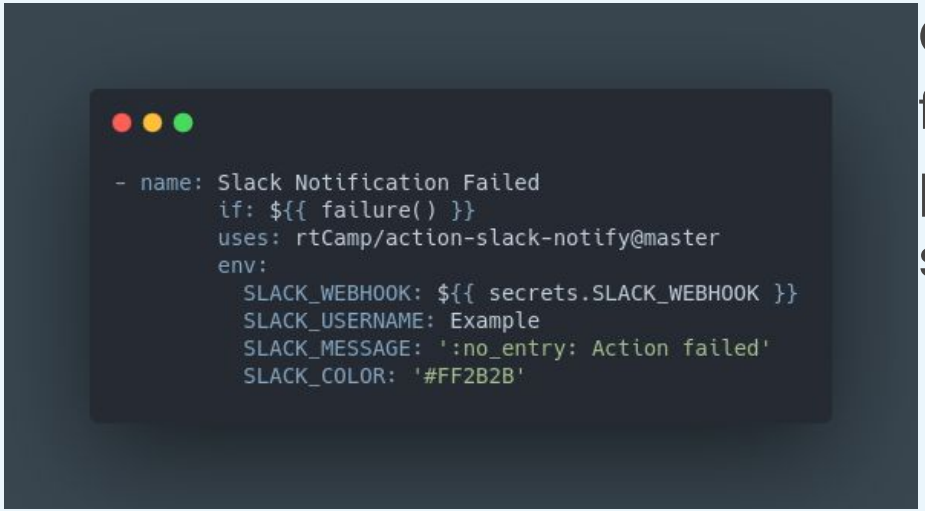
```yaml
- regExp: ".*\\.go+$"
  labels: ["source"]
- regExp: ".*\\.md+$"
  labels: ["documentation"]
- regExp: ".*\\.md+$"
  labels: ["continuos integration"]
```

merge automatico

```yaml
name: automerge
on:
  pull_request:
    types:
      - labeled
      - unlabeled
      - synchronize
      - opened
      - edited
      - ready_for_review
      - reopened
      - unlocked
  pull_request_review:
    types:
      - submitted
  check_suite:
    types:
      - completed
  status: {}

jobs:
  automerge:
    runs-on: ubuntu-latest
    steps:
      - name: automerge
        uses: "pascalgn/automerge-action@f81beb99aef41bb55ad072857d43073fba833a98"
        env:
          GITHUB_TOKEN: "${{ secrets.GITHUB_TOKEN }}"
          MERGE_LABELS: "automerge"
          MERGE_METHOD: "rebase"
```

Configurando una label specifica il bot di GH accodera' le PR da mergiare e verranno mergiate in automatico con la modalita' specificata.

notifica slack



```yaml
- name: Slack Notification Failed
        if: ${{ failure() }}
        uses: rtCamp/action-slack-notify@master
        env:
          SLACK_WEBHOOK: ${{ secrets.SLACK_WEBHOOK }}
          SLACK_USERNAME: Example
          SLACK_MESSAGE: ':no_entry: Action failed'
          SLACK_COLOR: '#FF2B2B'
```

Configurabile anche solo al failure, come nell'immagine, per non intasare i canali slack.

coverage

Actions famosa che manda il file coverage.txt al servizio di terze parti che ritornera' il coverage in percentuale.

Limitazioni

- Posso testare queste actions in locale?
- Il mio job ha una durata di >6h
- Se ho >20 jobs paralleli?
- Se faccio >1000 richieste API?
- Non viene garantita l'esecuzione immediata dei workflow.

SUGGERIMENTI

- Workflow con unica funzione
- Limitare le matrici
- Limitare variabili di ambiente
- Test in un repository privato

Federico Guerinoni

guerra@develer.com

Vuoi rimanere aggiornato sugli eventi Develer?
Seguici nei nostri canali social:

develer

www.develer.com