

Configurazione del workflow file

In questa sezione del file di workflow definisco il nome del workflow e le condizioni che ne determinano l'esecuzione su GitHub Actions.

Nel mio caso, il workflow viene avviato automaticamente quando viene effettuato un push sulla repository oppure quando viene aperta una nuova pull request.

```
1   name: Test e Deploy della Repository finale
2
3   on:
4     push:
5     pull_request:
```

In questa sezione definisco il job e il suo nome(**test-repository**), specificando il tipo di runner su cui verranno eseguiti i test.

Nel mio caso, ho scelto di utilizzare **ubuntu-latest** come ambiente di esecuzione.

```
7  jobs:
8    test-repository:
9      name: test della repository
10     runs-on: ubuntu-latest
```

In questa sezione clono la **repository** e definisco i vari **step** che verranno eseguiti dal runner.

Per ogni step assegno un nome descrittivo e specifico il comando bash da eseguire.

In questo caso, caso ad installare alcuni strumenti come **shellcheck** e **perl**, che utilizzerò successivamente per eseguire test sugli script presenti nella repository.

```
steps:
- name: clonazione repository
  uses: actions/checkout@v4

- name: Installare shellcheck,tree e perl
  run: sudo apt install tree shellcheck perl

- name: check degli script bash
  run: |
    find "$GITHUB_WORKSPACE/" -iname "*.sh" | while read -r file; do
      echo "Checking $file"
      shellcheck -e SC2181 -e SC2148 -e SC2086 -e SC2164 -e SC2046 -e SC2210 -e SC2002 "$file"
    done
```

In questa sezione del **workflow**, eseguo test sugli script Perl e Bash utilizzando i pacchetti e le dipendenze installati in precedenza.

```
- name: check degli script bash
run: |
    find "$GITHUB_WORKSPACE/" -iname "*.sh" | while read -r file; do
        echo "Checking $file"
        shellcheck -e SC2181 -e SC2148 -e SC2086 -e SC2164 -e SC2046 -e SC2210 -e SC2002 "$file"
    done

- name: Install System Dependencies
run: |
    sudo apt update
    sudo apt install -y cpanminus build-essential libssl-dev libexpat1-dev \
        libdbi-perl libdbd-mysql-perl libwww-perl

- name: Install Perl Dependencies
run: |
    sudo cpanm --notest Daemon::Control MongoDB Cache::Memcached \
        Dancer2 Config::Tiny DBIx::Class DateTime

- name: test script perl
run: |
    find "$GITHUB_WORKSPACE/" -iname "*.pl" | while read -r file; do
        echo "Checking $file"
        perl -c "$file"
    done
```

In questa sezione eseguo lo script utilizzato per la configurazione automatica del server, con l'obiettivo di verificare che tutti gli script e le configurazioni presenti nella repository siano corretti e funzionanti.

Alla fine del processo, controllo che i servizi installati siano operativi e funzionino come previsto.

```
50
51 - name: Creazione cartella /base
52   run: sudo mkdir /base && sudo rsync -a --copy-links $GITHUB_WORKSPACE/servers/base/ /base
53
54 - name: Verifica dei file dentro /base
55   run: tree /base
56
57 - name: test /base
58   run: cd /base && sudo sh ./0_step.sh
59
60 - name: testing daemons
61   run: sudo systemctl start mariadb.service mongod.service munin-node.service memcached.service nginx.service && sudo s
62
63 - name: test configurazione di nagios
64   run: sudo systemctl restart nagios.service && sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
```

In questa workflow definisco un nuovo job chiamato **Deploy**. Utilizzando la parola chiave `needs`, specifico che questo job verrà eseguito automaticamente solo se il job **test-repository** è stato completato con successo.

```
1  name: CD della Repository
2
3  on:
4    pull_request:
5      branches:
6        - main
7    types: [closed]
8  jobs:
9    CD-repository:
10     name: Deploy della repository
11     runs-on: ubuntu-latest
12
13     steps:
14       - uses: actions/checkout@v4
15
16       - name: List workspace contents
17         run: ls -R $GITHUB_WORKSPACE
18
19       - name: rsync deployments
20         uses: burnett01/rsync-deployments@7.0.2
21         with:
22           switches: -avzr --delete --copy-links
23           path: servers/base
24           remote_path: /root/
25           remote_host: *
26           remote_port: 22
27           remote_user: root
28           remote_key: ${ secrets.SSH_PRIVATE_KEY }
```

In questa sezione del job Deploy, utilizzo l'action [burnett01/rsync-deployments](#) per trasferire i file dalla repository al server remoto tramite SSH.

Il primo step sincronizza la directory servers/base con la directory /root/ del server, utilizzando l'opzione --delete per rimuovere eventuali file obsoleti.

Il secondo step, invece, trasferisce il contenuto della cartella src/ nella directory /root/base/ del server.

L'autenticazione avviene tramite chiave SSH privata, gestita in modo sicuro attraverso i [GitHub Secrets](#).

I GitHub Secrets sono variabili sicure che possiamo definire nel repository per memorizzare informazioni sensibili, come chiavi private, password o token di accesso. In questo caso, la chiave privata SSH è memorizzata come `SSH_PRIVATE_KEY` nel pannello delle impostazioni di GitHub, e viene utilizzata nel workflow tramite `${{ secrets.SSH_PRIVATE_KEY }}`. In questo modo, le informazioni sensibili non sono visibili nel codice o nei log di esecuzione, garantendo maggiore sicurezza.

```
1  name: CD della Repository
2
3  on:
4    pull_request:
5      branches:
6        - main
7      types: [closed]
8  jobs:
9    CD-repository:
10     name: Deploy della repository
11     runs-on: ubuntu-latest
12
13     steps:
14       - uses: actions/checkout@v4
15
16       - name: List workspace contents
17         run: ls -R $GITHUB_WORKSPACE
18
19       - name: rsync deployments
20         uses: burnett01/rsync-deployments@7.0.2
21         with:
22           switches: -avzr --delete --copy-links
23           path: servers/base
24           remote_path: /root/
25           remote_host: *
26           remote_port: 22
27           remote_user: root
28           remote_key: ${{ secrets.SSH_PRIVATE_KEY }}
```