

An aerial photograph of a historic Italian city, likely Florence, showing a dense urban landscape with terracotta roofs, a winding river, and a large square with a prominent building. The image is used as a background for the conference title.

DELPHIDAY

italian conference

Continuous Deployment

con le GitHub Actions



ALBERTO **DAL DOSSO**

LBA TXT GROUP - DEVELOPER

@ alberto.daldosso@gmail.com

twitter.com/daldosso

github.com/daldosso

in linkedin.com/in/daldosso

DELPHIDAY

italian conference

9-10 Giugno 2026
Piacenza





AGENDA

- 1. Introduzione a Git & GitHub (Storia, radici e curiosità)**
- 2. Strategie di Branching (Feature Branches, Hotfixes e il GitHub Flow)**
- 3. Il Linguaggio dell'Automazione (Sintassi YAML ed ecosistema GitHub Actions)**
- 4. I Runner nel Mondo Delphi (Self-Hosted Runner)**
- 5. La Pipeline di CD (Compilazione IDE, Push e Live Deploy)**



Git & GitHub

1



2005: La Crisi che ha creato Git

- Il caos di Linux Kernel
- La rottura: BitKeeper revoca l'uso gratuito alla community Linux.
- La sfida: Linus Torvalds non trova nessun tool alternativo abbastanza veloce.
- Il miracolo: Isolatosi per un weekend, Linus scrive Git in soli 10 giorni.





GitHub: La Nascita (2008)

- I Fondatori:

Tom Preston-Werner,
Chris Wanstrath,
PJ Hyett e Scott Chacon.

- L'Idea iniziale:

Semplificare radicalmente la
condivisione del codice Git, allora
complessa.





L'Acquisizione da Record

- Giugno 2018: Microsoft annuncia l'acquisizione di GitHub.
- La Cifra: Acquistata per l'incredibile somma di 7.5 miliardi di dollari in azioni.
- Il cambio di rotta: Da nemica dell'Open Source sotto Ballmer, Microsoft diventa il player principale con Satya Nadella.



Branching

2



Strategie di Branching

- Feature Branches
- Hotfixes
- Trunk-Based
- GitHub Flow



Dettagli del Branching

Strategia	Descrizione
Feature branches	Branch per ogni funzionalità, merge su main / develop
Hotfixes	Branch urgenti per correzioni critiche in produzione
Trunk-based	Tutto su main, branch molto brevi + feature flags
GitHub Flow	Un solo branch main, feature brevi e PR frequenti





YAML

YAML Ain't Markup Language

3



YAML: Sintassi

- Indentazione con 2 spazi
- Case-sensitive
- I due punti: significano "chiave: valore"
- Dopo i : ci deve essere uno spazio
- Il trattino - indica un elemento di una lista
- Niente punto e virgola ; alla fine della riga
- # commento

```
name: Nome Workflow          ← chiave: valore

on:                           ← chiave complessa
  push:                       ← sotto-chiave
    branches:                 ← altra chiave
      - main                  ← lista (-)
      - develop

jobs:                          ← chiave
  build:                      ← nome del job (chiave)
    runs-on: ubuntu-latest    ← chiave: valore
    steps:                    ← chiave (lista di step)
      - name: Checkout        ← elemento della lista
        uses: actions/checkout@v4 ← chiave: valore
      - name: Build
        run: |                ← | = testo multilinea
          echo "Prima riga"
          dotnet build
```




Runner

4



Runners

☰

 daldosso / DelphiDayDemo

🔍 Type to search

<> Code

🕒 Issues

🔗 Pull requests

👤 Agents

⌚ Actions

📁 Projects

📖 Wiki

🛡️ Security and quality

📊 Insights

⚙️ Settings

⚙️ General

Access

👤 Collaborators

🗨️ Moderation options

Code and automation

📄 Rules

⌚ Actions

General

Runners

OIDC

🔗 Models

🔗 Webhooks

👤 Copilot

📁 Environments

📄 Codespaces

📄 Pages

Security and quality

Runners / Add new self-hosted runner · daldosso/DelphiDayDemo


Add new self-hosted runner · daldosso/DelphiDayDemo


⚠️


Using self-hosted runners in public repositories is not recommended. Forks of your public repository can potentially run dangerous code on your self-hosted runner by creating a pull request. [Learn more about security hardening for self-hosted runners.](#)

Adding a self-hosted runner requires that you download, configure, and execute the GitHub Actions Runner. If you do not already have an existing volume licensing agreement for your GitHub purchases, by downloading and configuring the GitHub Actions Runner, you agree to the [GitHub Customer Agreement](#).

Runner image

☐  macOS

☐  Linux

☒  Windows

Architecture

x64

Download

We recommend configuring the runner under "actions-runner". This will help avoid issues related to service identity folder permissions and long path restrictions on Windows.

Create a folder under the drive root



Installazione

```
Administrator: C:\Program Files\PowerShell\powershell.exe
PowerShell 7.6.1
PS C:\Windows\System32> cd C:\dev\Personal\DelphiDay\actions-runner
PS C:\dev\Personal\DelphiDay\actions-runner> Invoke-WebRequest -Uri https://github.com/actions/runner/releases/download/v2.334.0/actions-runner-win-x64-2.334.0.zip -OutFile actions-runner-win-x64-2.334.0.zip
Web request status [Downloaded: 47,4 MB of 95,2 MB]
PS C:\dev\Personal\DelphiDay\actions-runner> Invoke-WebRequest -Uri https://github.com/actions/runner/releases/download/v2.334.0/actions-runner-win-x64-2.334.0.zip -OutFile actions-runner-win-x64-2.334.0.zip
PS C:\dev\Personal\DelphiDay\actions-runner> Add-Type -AssemblyName System.IO.Compression.FileSystem ; [System.IO.Compression.ZipFile]::ExtractToDirectory("$PWD/actions-runner-win-x64-2.334.0.zip", "$PWD")
PS C:\dev\Personal\DelphiDay\actions-runner> ./config.cmd --url https://github.com/dalosso/DelphiDayDemo --token AACCQOCKF7VXXJX7J7GTQWLKCBYRQ

-----
GitHub Actions
Self-hosted runner registration
-----

# Authentication

V Connected to GitHub

# Runner Registration

Enter the name of the runner group to add this runner to: [press Enter for Default]
Enter the name of runner: [press Enter for LBA-LPT039] daldoDev

This runner will have the following labels: 'self-hosted', 'Windows', 'X64'
Enter any additional labels (ex. label-1,label-2): [press Enter to skip]

V Runner successfully added

# Runner settings

Enter name of work folder: [press Enter for _work]

V Settings Saved.

Would you like to run the runner as service? (Y/N) [press Enter for N]
PS C:\dev\Personal\DelphiDay\actions-runner> ./run.cmd
1 file(s) copied.

V Connected to GitHub

Current runner version: '2.334.0'
2026-05-22 14:26:44Z: Listening for Jobs
Exiting...
Terminate batch job (Y/N)? Y
PS C:\dev\Personal\DelphiDay\actions-runner>
```

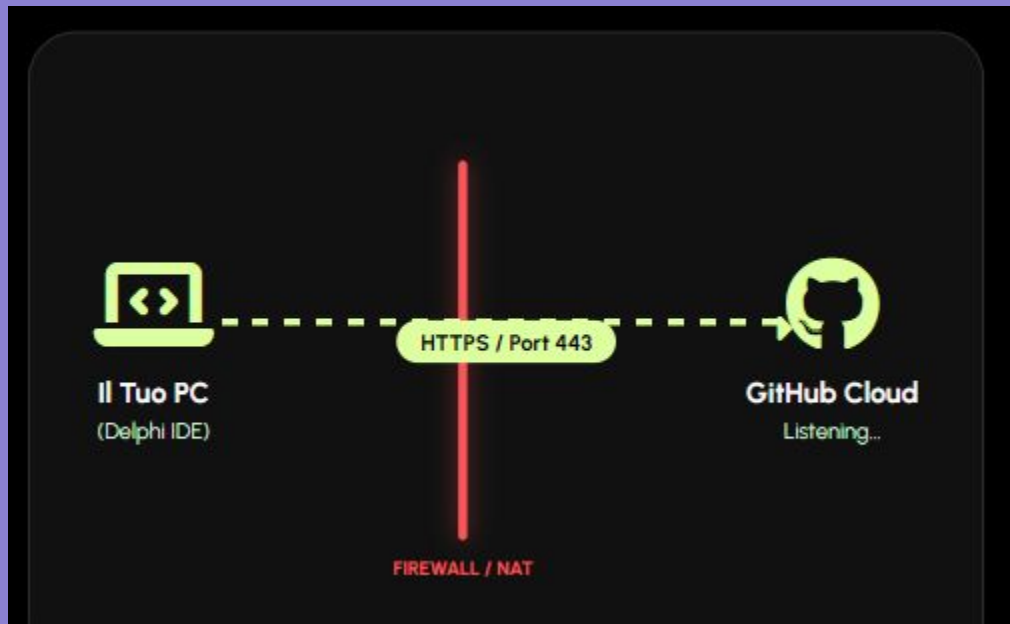


Runner in esecuzione

- Zero Configurazione Rete
- Solo Traffico Outbound: Il PC contatta GitHub, non il contrario.
- Nessuna Porta Aperta: Il firewall aziendale blocca tutto ciò che arriva dall'esterno.
- Porta 443 (HTTPS): Usa lo stesso protocollo sicuro della navigazione web standard.
- Meccanismo di "Long Polling": Il runner resta in attesa di istruzioni su un canale sicuro già stabilito dall'interno.



Runner in esecuzione





VISUAL DASHBOARD

ea-Apps-CustomerMaster

Search Type to search

Requests 3

Agents

Actions

Projects

Wiki

Security and quality 18

Insights

Settings

ter-dev #11

Manually triggered 4 months ago

Status

Total duration

Artifacts

ralessirccl

756c837

main

Success

2m 51s

1

cicd.yml

on: workflow_dispatch

✓ build

1m 23s

✓ Deploy to DEV Server

20s

⌚ Deploy to TEST Server

0s

⌚ Deploy to STAGE Server

0s



Pipeline CD

5



Struttura del Workflow

Anatomia del file YAML

```
name: Hello World Workflow
on:
  workflow_dispatch
jobs:
  hello:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Hello World
        run: |
          echo "Hello World!"
```

name Nome del workflow visibile nell'interfaccia GitHub

on Evento trigger:
workflow_dispatch = manuale

jobs Insieme di job da eseguire in parallelo

steps Lista di passi del job: actions o comandi shell



Compilazione Delphi

```
- name: Compile Delphi Project
  run: |
    call "%BDSBIN%\rsvars.bat"
    msbuild DelphiDayDemo.dproj /p:Config=Release /t:Build
  shell: cmd
```



Verifica

```
- name: Verify Compiled Binary
  run: |
    if (Test-Path "Win32\Release\DelphiDayDemo.exe") {
      Write-Host "Verification Success: DelphiDayDemo.exe found."
    } else {
      Write-Error "Verification Failed: Executable not found in Win32\Release."
      exit 1
    }
  shell: powershell
```



Backup

```
- name: Backup Remote Binary
  run: |
    $url = "ftp://myserver.com/app.exe"
    $ftp = [System.Net.FtpWebRequest]::Create($url)
    $ftp.Credentials = New-Object Net.NetworkCredential(`
      "${{ secrets.USER }}", "${{ secrets.PASS }}")
    $ftp.Method = [Net.WebRequestMethods+Ftp]::Rename
    $ftp.RenameTo = "app_$(Get-Date -f yyyyMMdd).bak"
    $ftp.GetResponse().Close()
  shell: powershell
```



Backup Remote Com Environment

```
jobs:
  backup-and-deploy:
    environment: ${GITHUB_EVENT.inputs.target_env}
    runs-on: windows-latest
    steps:
      - name: Backup Remote Binary
        run: |
          $url = "${vars.FTP_URL}"
          $ftp = [System.Net.FtpWebRequest]::Create($url)
          $ftp.Credentials = New-Object Net.NetworkCredential(
            "${secrets.USER}", "${secrets.PASS}")
          $ftp.Method = [Net.WebRequestMethods+Ftp]::Rename
          $ftp.RenameTo = "app_$(Get-Date -f yyyyMMdd).bak"
          $ftp.GetResponse().Close()
        shell: powershell
```



Environments

```
on:
  workflow_dispatch: # Allows manual execution from
    inputs:
      target_env:
        description: 'Select the target environment'
        type: choice
        required: true
        default: 'staging'
        options:
          - dev
          - staging
          - production
```

The screenshot shows a GitHub Actions workflow configuration interface. At the top right, there is a button labeled "Run workflow". Below it, a section titled "Use workflow from" contains a dropdown menu showing "Branch: main". Further down, a section titled "Select the target environment *" features a dropdown menu with the following options: "dev", "staging", and "production". The "dev" option is currently selected and highlighted in blue.



Deploy

```
- name: Deploy to Live Server
  run: |
    $dest = "C:\LiveServer\bin\"
    if (-not (Test-Path $dest)) {
      New-Item -ItemType Directory -Force -Path $dest
    }
    Copy-Item -Path "Win32\Release\DelphiDayDemo.exe" `
      -Destination $dest -Force
    Write-Host "Deployment completed successfully to $dest"
  shell: powershell
```



Deploy FTP

```
- name: Deploy via FTP
  uses: SamKirkland/FTP-Deploy-Action@v4.3.5
  with:
    server: ${ secrets.FTP_SERVER }
    username: ${ secrets.FTP_USERNAME }
    password: ${ secrets.FTP_PASSWORD }
    local-dir: ./Win32/Release/
    server-dir: /var/www/app/bin/
```



Deploy Strutturato

```
- name: Prepare Staging Directory
  run: |
    New-Item -ItemType Directory -Force -Path "dist\bin"
    New-Item -ItemType Directory -Force -Path "dist\config"
    Copy-Item "Win32\Release\*.exe" -Destination "dist\bin"
    Copy-Item "assets\config.json" -Destination "dist\config"
  shell: powershell

- name: Deploy via FTP (Mirroring)
  uses: SamKirkland/FTP-Deploy-Action@v4.3.5
  with:
    server: ${ secrets.FTP_SERVER }
    username: ${ secrets.FTP_USERNAME }
    password: ${ secrets.FTP_PASSWORD }
    local-dir: ./dist/
    server-dir: /var/www/app/
```



Rollback

```
- name: Deploy Application
  run: |
    # If this step fails, GitHub Actions stops normal execution
    Copy-Item "Win32\Release\app.exe" -Destination "C:\LiveServer\bin\" -Force

- name: Automatic Rollback on Failure
  if: failure() # Triggers ONLY if any previous step failed
  run: |
    Write-Host "Deploy failed! Restoring last stable backup..."
    if (Test-Path "C:\LiveServer\backup\app_old.bak") {
      Copy-Item "C:\LiveServer\backup\app_old.bak" -Destination "C:\LiveServer\bin\app.exe"
    }
  shell: powershell
```



Triage

```
name: Issue triage

on:
  issues:
    types: [opened, edited]

jobs:
  triage:
    runs-on: ubuntu-latest
    steps:
      - name: Label issue
        run: |
          if (contains(github.event.issue.body, 'bug')) {
            echo "::add-labels: bug"
          } else if (contains(github.event.issue.body, 'feature')) {
            echo "::add-labels: feature"
          } else {
            echo 'Labeling issue as needs-triage'
            echo "::add-labels: needs-triage"
          }
        }
```



Quale SHELL usare??

pwsh (PowerShell 7)

Lo standard moderno. Cross-platform, gestisce errori in modo nativo e sicuro.

cmd (Command Prompt)

Fondamentale per rsvars.bat. Richiama l'ambiente Delphi senza frizioni.

powershell (WinPS 5.1)

La shell classica. Utile per script che dipendono da moduli Windows legacy.

bash (Git Bash)

Inclusa nei runner Windows. Ideale per chi preferisce la sintassi Unix.



Gestire gli Ambienti

1. Configurazione

GitHub Environments permette di isolare segreti e regole di protezione. Ogni ambiente (Dev, Staging, Live) ha le sue variabili specifiche.

```
jobs: deploy: environment: ${  
github.event.inputs.target_env }}
```

2. Scelta dell'Utente

Usa workflow_dispatch con un choice per forzare l'utente a selezionare il target di deploy prima di far partire la pipeline.

```
on: workflow_dispatch: inputs: target_env: type:  
choice options: [dev, staging, live]
```



GitHub Actions Marketplace

18.000+ azioni pronte all'uso

Le più utilizzate:

- ◆ `actions/checkout@v4` → Scarica il codice
- ◆ `actions/setup-dotnet@v4` → Setup .NET
- ◆ `actions/setup-node@v4` → Setup Node.js
- ◆ `actions/upload-artifact@v4` → Salva file
- ◆ `actions/download-artifact@v4` → Scarica file
- ◆ `dawidd6/action-send-mail@v3` → Invio email
- ◆ `azure/login@v2` → Login Azure
- ◆ `github/codeql-action/analyze@v3` → Sicurezza



Creare una Custom Action

1 I Metadati

Crea un file chiamato obbligatoriamente `action.yml` per definire input, output e il motore di esecuzione.

2 La Logica

Scrivi il codice applicativo nativo (es. uno script PowerShell o un file JavaScript/Node.js).

3 Il Rilascio

Pubblica il repository inserendo un **Tag di versione** (es. `v1.0.0`) per renderla immediatamente riutilizzabile.

ACTION.YML

```
name: 'Delphi Compiler Action'
description: 'Compila progetti Delphi via MSBuild'
inputs:
  project-path:
    description: 'Path del file .dproj'
    required: true
runs:
  using: 'composite'
  steps:
    - shell: cmd
      run: |
        call "%BDSBIN%\rsvars.bat"
        msbuild ${{ inputs.project-path }} /t:Build
```



Pubblicare sul Marketplace



1. Prerequisiti

Il tuo repository deve rispettare alcuni vincoli fondamentali prima del rilascio:

- Deve essere **Pubblico**
- Il file `action.yml` deve essere nella root
- Un file `README.md` descrittivo



2. Il Banner Web

Appena GitHub rileva il file di metadati, ti mostrerà un box speciale in cima al repository:

- Clicca su **"Draft a release"**
- Spunta la casella **"Publish this Action to the GitHub Marketplace"**



3. Metadati & Tag

Definisci l'estetica e rilascia il codice sulla piattaforma:

- Scegli una categoria e un'icona
- Crea un Tag Git (es. `v1.0.0`)
- Clicca su **Publish Release**

An aerial photograph of a city, likely Florence, Italy, showing a dense urban landscape with a river (Arno) winding through it. A large, historic square (Piazza della Signoria) is visible in the lower center, featuring a prominent building with arches. The image is overlaid with a semi-transparent dark layer to accommodate text.

DELPHIDAY


italian conference

THANK YOU