

LORENZO SAVINI

Developer

Introduzione a Buildkite

Develer webinar

28/04/2021

develer



OBIETTIVI DEL WEBINAR

- Chiarire il concetto di continuous integration
- Comprendere Buildkite e i suoi concetti principali
- Prendere familiarità con le pipeline e i vari strumenti a disposizione

Programma del webinar

- Overview di Buildkite
 - Architettura
 - Feature principali
- Come si configura il nostro primo progetto
- Pipeline e step
- Agent plugins
- Meta-data e Artifacts
- Pipeline dinamiche
- Conditionals e dependencies
- Pipeline schedulate
- Bonus

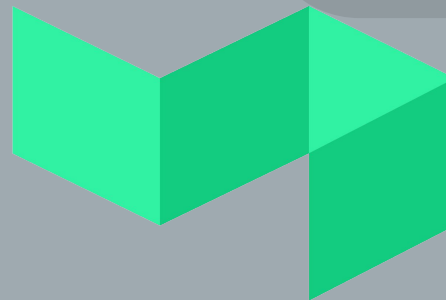
Repository con esempi

<https://github.com/savo92/webinar-buildkite>

Continuous Integration

Metodologia dello sviluppo del software in cui si allinea spesso la mainline del progetto dagli ambienti di lavoro degli sviluppatori (aka: si fa merge spesso delle branch su main/master).

Buildkite: le basi



Buildkite

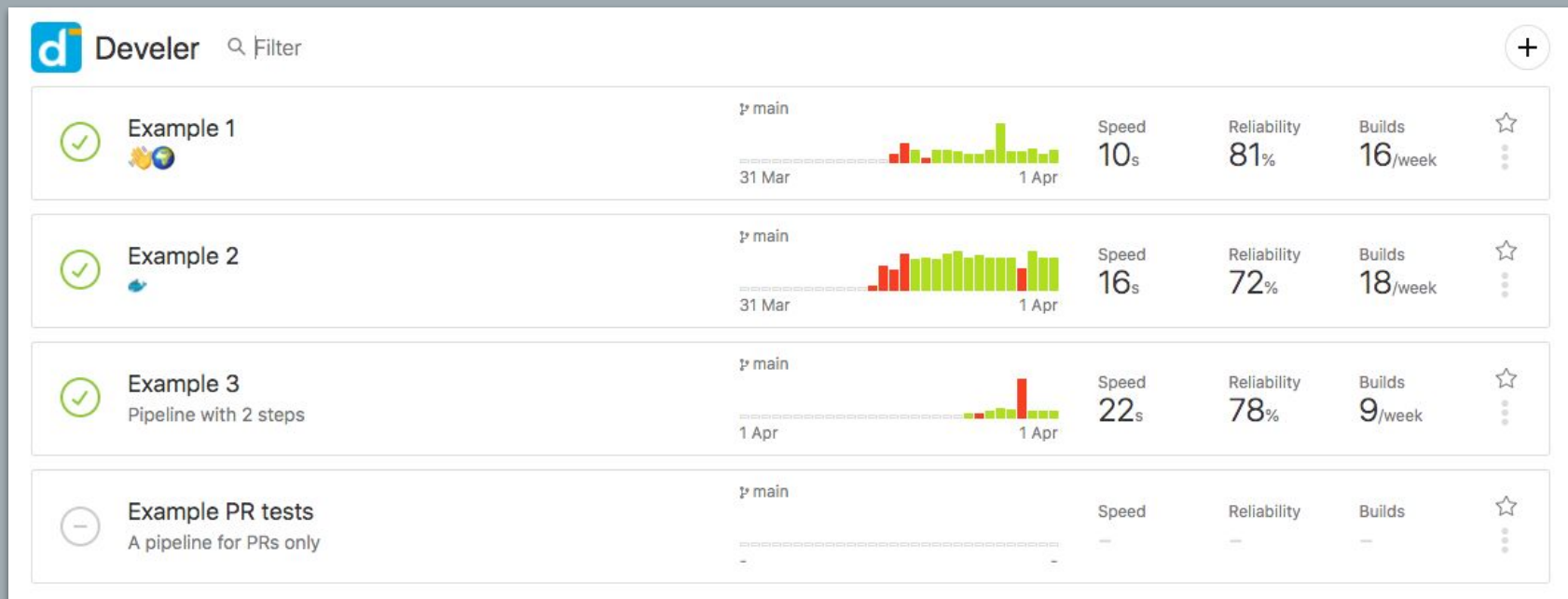
Introduzione a Buildkite – Buildkite: le basi

Buildkite è una piattaforma per continuous integration che permette di creare pipeline per i vostri progetti, che verranno poi eseguite da agent installati sulla vostra infrastruttura.

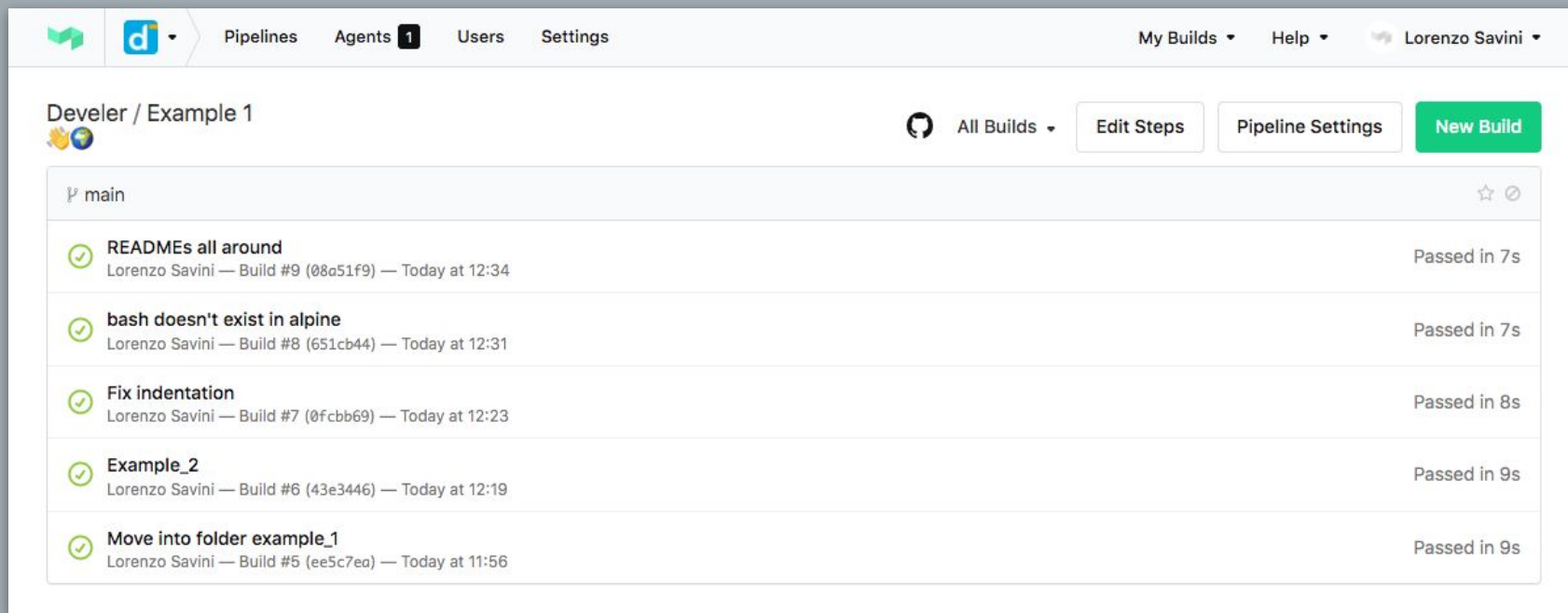
I Concetti base – Buildkite: le basi

- **Pipeline:** insieme di **step** che vengono eseguiti da Buildkite.
- **Build:** una specifica esecuzione di una pipeline.
- **Agent:** build runner che gira sulla vostra infrastruttura (VPS, server, Docker) connesso con Buildkite, ed esegue le vostre build.

Homepage – Buildkite: le basi



UI di una pipeline – Buildkite: le basi



The screenshot displays the Buildkite web interface. At the top, there's a navigation bar with links for Pipelines, Agents (1), Users, and Settings. On the right, there are links for My Builds, Help, and a user profile for Lorenzo Savini. The main content area shows a pipeline titled 'Develer / Example 1' with a branch icon and 'main' branch. To the right of the title are buttons for 'All Builds', 'Edit Steps', 'Pipeline Settings', and a green 'New Build' button. Below the title, a list of five build steps is shown, all with green checkmark icons indicating they passed. Each step includes the step name, the user 'Lorenzo Savini', the build number, a commit hash, and the time taken to pass.

Step Name	User	Build	Commit	Time
READMEs all around	Lorenzo Savini	#9	08a51f9	Passed in 7s
bash doesn't exist in alpine	Lorenzo Savini	#8	651cb44	Passed in 7s
Fix indentation	Lorenzo Savini	#7	0fcb69	Passed in 8s
Example_2	Lorenzo Savini	#6	43e3446	Passed in 9s
Move into folder example_1	Lorenzo Savini	#5	ee5c7ea	Passed in 9s

UI di una build – Buildkite: le basi

Develer / Example 2 / main

All Builds Edit Steps Pipeline Settings **New Build**

Fix indentation Failed in 19s

Build #5 | main | 0fcbb69

buildkite-agent pi... Run a Docker compose ...

Lorenzo Savini Created today at 12:27 Triggered from Web Rebuilt from #4 **Rebuild**

buildkite-agent pipeline upload ./example_2/.buildkite/pipeline-2.yml Ran in 4s Waited 1s Kraken-1

Run a Docker compose service bash ./script/hello-world-from-docker Ran in 14s Waited 1s Kraken-1

Log Artifacts Timeline Environment **Retry**

+ Expand groups - Collapse groups Delete Download Raw Jump to end

```
1  ▶ Preparing plugins 0s
3  ▶ Running plugin docker-compose pre-checkout hook 1s
5  ▶ Preparing working directory 2s
19 ▶ Running plugin docker-compose command hook 1s
21 ▶ Building Docker Compose Service: example_2 5s
48 ▶ Starting dependencies 1s
52 ▶ Running /bin/sh -e -c 'bash ./script/hello-world-from-docker' in service example_2 1s
53 $ docker-compose -f ./example_2/docker-compose.yml -p buildkite80759b22e0554dfc9231da5ea8fa08e8 run --name
    buildkite80759b22e0554dfc9231da5ea8fa08e8_example_2_build_5 -e ENV_VAR_1 --rm example_2 /bin/sh -e -c 'bash ./script/hello-world-from-docker'
54 Creating buildkite80759b22e0554dfc9231da5ea8fa08e8_example_2_run ... done
55 /bin/sh: bash: not found
56 ERROR: 127 0s
58 ▶ Failed to run command, exited with 127 0s
59 Error: The command exited with status 127
62 user command error: The plugin docker-compose command hook exited with status 127
63 ▶ Running plugin docker-compose pre-exit hook 1s
65 ▶ Cleaning up after docker-compose 2s
```

Exited with status 127 Back to Job

I Integrazione con repository Git – Buildkite: le basi



- GitHub
- GitHub Enterprise Server



- GitLab
- GitLab Community / Enterprise Edition

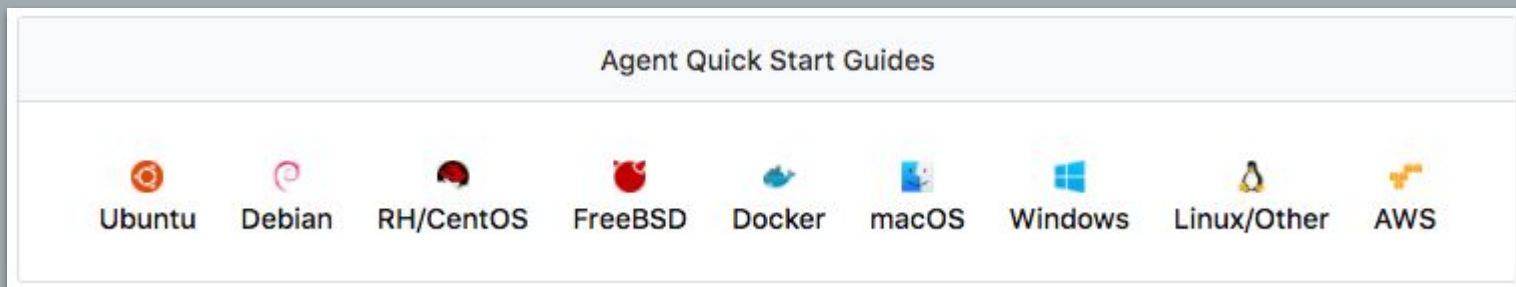


- Bitbucket
- Bitbucket Server



Qualsiasi Git server
(niente integrazione assistita però)

Agent – Buildkite: le basi



I Agent su MacOS – Buildkite: le basi

```
$ brew install buildkite/buildkite/buildkite-agent  
$ sed -i '' "s/xxx/<YOUR AGENT TOKEN>/g" "$(brew  
--prefix)"/etc/buildkite-agent/buildkite-agent.cfg
```

```
~/develer/webinar/buildkite/example % main ± buildkite-agent start
```

Pipeline e step

- | Creazione di una pipeline
- | Tipi di step

Dichiarare una pipeline – Pipeline e step

Una pipeline si dichiara in YAML:

- direttamente dalla UI di Buildkite;
- con un file YAML nella vostra repository.

A screenshot of the Buildkite web interface. The header shows 'Develer / Webinar BuildKite Example 1' with a globe icon on the left and a 'Save Steps' button on the right. Below the header is a code editor with a light blue background. It contains a YAML configuration for a pipeline step. The code is as follows:

```
1 steps:
2   - command: "buildkite-agent pipeline upload ./example_1/.buildkite/pipeline-1.yml"
3     timeout_in_minutes: 30
4
```


Dichiarare una pipeline in un file YAML – Pipeline e step

Se volete dichiarare una pipeline tramite un file YAML nella vostra repository, in alcuni casi dovrete comunque usare la UI per instruire Buildkite a caricare la configurazione.

Basta uno step con questo comando:

```
buildkite-agent pipeline upload ./example_1/.buildkite/pipeline-1.yml
```

Prima pipeline – Pipeline e step

Create your first pipeline

You're connected to GitHub! Fill in a repository URL and get ready to run a build 🚀

Name — Required

The name for the pipeline

GitHub Repository — Required

 **savo92/webinar-buildkite**
Repository per il webinar Develer #1 2021 ✕

Checkout using: ☒ SSH ☐ HTTPS

The repository your agents will use to checkout your code. Need to [choose another repository or URL?](#)

☒ **Auto-create webhooks**

Webhooks are used to trigger builds in Buildkite. When selected, we'll attempt to create them automatically.

Prima di cominciare... – Pipeline e step

Adesso alcuni provider, come GitHub, creano repositories con “main” come default branch.

Buildkite di default usa ancora “master”, quindi dovreste configurare a dovere la vostra pipeline se necessario:

Default Branch

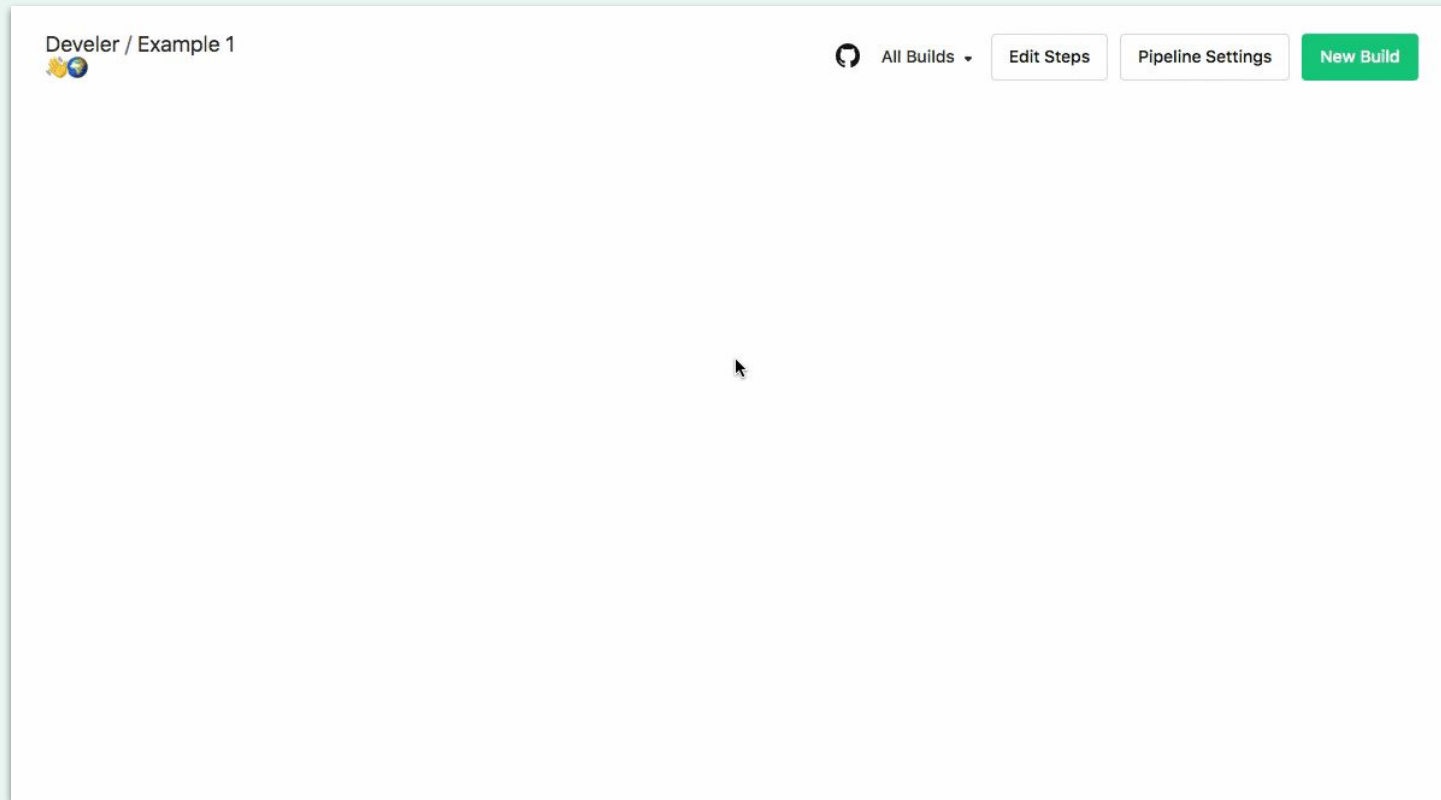
The default branch is used to prefill the branch when new builds are created or triggered, and to filter the builds and metrics shown on the Pipelines page. Leave blank if your pipeline has no default branch.

Save Default Branch

Esempio 1: Prima pipeline - Pipeline e step

Esempio #1

Esempio 1: Prima pipeline - Pipeline e step



Un esempio di sintassi – Pipeline e step

```
steps:  
  - name: 'Hello world!'  
    command: 'bash ./script/hello-world'  
  
  - name: 'Another Hello world'  
    command: 'bash ./script/hello-world'
```

Per dichiarare una pipeline è sufficiente creare un file YAML contenente una lista di steps, ciascuno con un comando ed altri metadati (tipo il nome dello step, per visualizzarlo nella UI di Buildkite).

Nota bene: in questo esempio i 2 step vengono eseguiti in **parallelo** su differenti agent, se disponibili.

Tipi di step

Buildkite permette di dichiarare 5 diversi tipi di step:

- Command step
- Wait step
- Block step
- Input step
- Trigger step

Command step – Tipi di step – Pipeline e step

```
steps:  
- command: "dosomething.sh"  
  
- commands:  
  - "yarn install && yarn test"  
  - "dosomethingelse.sh"
```

Un command step permette di eseguire uno o più comandi.

Nel secondo step, che prevede una lista di comandi, un eventuale fallimento di Yarn impedirebbe l'esecuzione del secondo comando dosomethingelse.sh.

Esempio 2 – Pipeline e step

Esempio #2

Esempio 2 – Pipeline e step

The screenshot shows the Buildkite web interface for a pipeline named 'Example 2 - 2 steps'. At the top, there are tabs for 'All Builds', 'Edit Steps', 'Pipeline Settings', and a prominent 'New Build' button. Below the tabs, a message states 'There are no builds for this pipeline yet.' and provides instructions on how to trigger a build via the REST API or a GitHub webhook. A code block contains a cURL command for triggering a build. A modal window titled 'New Build' is open in the foreground, featuring a 'Message' input field, a 'Commit' dropdown menu (set to 'HEAD'), a 'Branch' dropdown menu (set to 'main'), and an 'Options' section. At the bottom of the modal is a large green 'Create Build' button.

Develer / Example 2 - 2 steps
Pipeline with 2 steps

All Builds Edit Steps Pipeline Settings **New Build**

There are no builds for this pipeline yet.

You can trigger a build using the New Build button above, or via the [REST API](#), for example:

```
curl -H "Authorization: Bearer $TOKEN" "https://api.buildkite.com/v2/organizations/develer-1/pipelines/example-2-2-steps/builds" \
  -X "POST" \
  -F "commit=HEAD" \
  -F "branch=master" \
  -F "message=First build :rocket:"
```

You can also setup a [GitHub webhook](#) to automatically trigger builds.

New Build

Message

Description of the build. If left blank, the commit message will be used once the build starts.

Commit — Required

Branch — Required

Options ▾

Create Build

Wait step – Tipi di step – Pipeline e step

```
steps:  
  - command: "dosomething.sh"  
  - wait  
  - command: "dosomethingelse.sh"
```

Un wait step, di default, attende che tutti i comandi precedenti siano terminati con successo.

Permette quindi di impedire la parallelizzazione che BK applica di norma.

Buildkite permette anche di configurare il wait step in modo da continuare anche in caso di fallimento di uno step precedente.

Esempio 3: Pipeline con wait step – Pipeline e step

Esempio #3

Esempio 3: Pipeline con wait step - Pipeline e step

Developer / Example 3 - wait step

Pipeline w/ wait steps

All Builds

Edit Steps

Pipeline Settings

New Build

New Build

Message

Description of the build. If left blank, the commit message will be used once the build starts.

Commit — Required

HEAD

Branch — Required

main

Options

Create Build

Block step – Tipi di step – Pipeline e step

```
steps:  
  - command: "dosomething.sh"  
  - block: "Go ahead!"  
  - commands: "release.sh"
```

Un block step mette in pausa l'esecuzione di una build e aspetta di essere sbloccato da un'operazione manuale dalla UI o via API.

Questo step crea delle dipendenze implicite su gli step precedenti e successivi.

Può essere configurato anche per richiedere parametri.

Esempio 4: Pipeline con block step – Pipeline e step

Esempio #4

Esempio 4: Pipeline con block step – Pipeline e step

The screenshot displays the Develer web interface. At the top, a navigation bar includes links for Pipelines, Agents, Users, and Settings, along with a user profile for 'Lorenzo Savini'. The main header shows the current pipeline: 'Develer / Example 4 - block step' with a sub-label 'Pipeline w/ a block step'. Action buttons for 'All Builds', 'Edit Steps', 'Pipeline Settings', and a prominent 'New Build' button are visible. A modal window titled 'New Build' is open in the center, featuring a 'Message' text area, a 'Commit' dropdown (set to 'HEAD'), and a 'Branch' dropdown (set to 'main'). An 'Options' dropdown is also present. A large green 'Create Build' button is at the bottom of the modal.

Develer / Example 4 - block step
Pipeline w/ a block step

All Builds Edit Steps Pipeline Settings New Build

New Build

Message

Description of the build. If left blank, the commit message will be used once the build starts.

Commit — Required

HEAD

Branch — Required

main

Options ▾

Create Build

Input step – Tipi di step – Pipeline e step

```
steps:  
  - input: "Request Release"
```

Un input step richiede informazioni all'utente.

A differenza del block step, questo tipo **NON** crea dipendenze implicite su gli step precedenti e successivi. Questa configurazione deve essere esplicita (vedremo tra poco come si dichiarano step dependencies esplicite).

Trigger step – Tipi di step – Pipeline e step

```
steps:  
  - trigger: "another-pipeline-slug"
```

Il trigger step permette di creare una nuova build su un'altra pipeline.

Di default, l'esecuzione della nuova build metterà in attesa la build madre fino al completamento. La build madre verrà interrotta se la nuova build fallisce.

È però possibile anche avviare build affinché vengano eseguite in maniera asincrona.

Agent plugin

I Cosa sono i plugin – Agent plugin

I plugin sono repository Git che permettono di estendere il comportamento di un command step utilizzando degli hook.

È possibile creare nuovi plugin oppure utilizzare quelli messi a disposizione da Buildkite o da contributor terzi, che si possono cercare qui <https://buildkite.com/plugins>.

I plugin non hanno bisogno di essere *installati* ma alcuni necessitano di un supporto degli agent (ad esempio: per i plugin legati a Docker, serve Docker attivo sull'agent).

I **Plugin più utilizzati** - Agent plugin

Tra alcuni dei plugin più utilizzati:

- Docker: per eseguire i tuoi step all'interno di un container;
- Docker compose: utilizzare Docker compose per eseguire i tuoi step sempre all'interno di un container;
- Cache: per attivare una cache persistente fra gli step;

Ma esistono anche plugin per AWS (ECS, S3 , Lambda), K8S, Go, Terraform, keyring.

Utilizzare il plugin Docker-compose – Agent plugin

```
steps:
  - name: 'Run a Docker compose service'
    command: 'dosomething.sh'
    plugins:
      docker-compose#v3.3.0:
        run: example_2
        config:
          - ./docker-compose.yml
        env:
          - ENV_VAR_1
```

Vediamo un esempio di un command step eseguito all'interno di un Docker container creato tramite Docker compose.

In questo caso, BK eseguirà `dosomething.sh` all'interno del service `example_2`, creato a partire dal `docker-compose.yml` indicato.

I **Esempio 5: Command step con Docker-compose** – Agent plugin

Esempio #5

Esempio 5: Command step con Docker-compose – Agent plugin

Develer / Example 5 – Docker

All Builds Edit Steps Pipeline Settings New Build

There are no builds for this pipeline yet.

You can trigger a build using the New Build button above, or via the REST API, for example:

```
curl -H "Authorization: Bearer $TOKEN" \
  -X "POST" \
  -F "commit=HEAD" \
  -F "branch=main" \
  -F "message=first build :rocket:"
```

You can also setup a GitHub webhook to auto

New Build

Message

Description of the build. If left blank, the commit message will be used once the build starts.

Commit — Required

Branch — Required

Options ^

Environment Variables

Place each environment variable on a new line, in the format `KEY=value`

☐ **Clean checkout**
Force the agent to remove any existing build directory and perform a fresh checkout

Create Build

Meta-data e artifacts

Passare informazioni e artefatti tra step

Metadati – Meta-data e artifacts

Buildkite permette di salvare dati, della dimensione massima di 1 KB, per essere utilizzati fra step, che potrebbero anche girare su agent differenti.

Per esempio, si può salvare la tag di una build da rilasciare.

Tra lo step di set e quello di get ci deve essere una dipendenza o un wait step.

```
buildkite-agent meta-data set "release-name" "v1"  
  
buildkite-agent meta-data get "release-name"
```

Metadati – Meta-data e artifacts

```
steps:
  - block: "Request Release"
    fields:
      - text: "Code Name"
        key: "release-name"

  - name: 'Print the code name'
    command: 'buildkite-agent meta-data get "release-name"'
```

Esempio 6 – Meta-data e artifacts

Esempio #6

Esempio 6 – Meta-data e artifacts

Develer / Example 6 - Meta-data
A pipeline using BK meta-data

All Builds ▾ Edit Steps Pipeline Settings **New Build**

New Build

Message

Description of the build. If left blank, the commit message will be used once the build starts.

Commit — Required

Branch — Required

Options ▾

Create Build

Artefatti – Meta-data e artifacts

Per tutte le altre risorse con dimensioni superiori ad 1 KB, è possibile utilizzare gli artifacts di Buildkite, per salvare e scaricare file in diversi step di una build.

Di default, Buildkite salva questi file in un loro bucket S3 ma è anche possibile utilizzare un proprio bucket S3, Google Cloud Storage o Artifactory.

Tra lo step di upload e quello di download ci deve essere una dipendenza o un wait step.

Upload di Artefatti tramite configurazione - Meta-data e artifacts

```
steps:  
  - command: "buildsomething.sh"  
    artifact_paths:  
      - "dist/**/*"
```

È sufficiente configurare l'opzione `artifact_paths` in uno step per istruire Buildkite a caricare i file indicati sullo storage.

Upload/Download tramite CLI - Meta-data e artifacts

```
buildkite-agent artifact upload dist/index.html  
  
buildkite-agent artifact download dist/* local-dist/
```

Potete anche scaricare artefatti caricati da una build lanciata da un trigger step, basta aggiungere
`--build $BUILDKITE_TRIGGERED_FROM_BUILD_ID`

Esempio 7 – Meta-data e artifacts

Esempio #7

Esempio 7 – Meta-data e artifacts

The screenshot shows the Develer web interface. At the top, there's a navigation bar with links for Pipelines, Agents (2), Users, and Settings. On the right, it says 'My Builds', 'Help', and the user name 'Lorenzo Savini'. Below the navigation bar, the main header shows 'Develer / Example 7 - Artifacts' and 'Pipeline uploading and downloading artifacts'. To the right of the header are buttons for 'All Builds', 'Edit Steps', 'Pipeline Settings', and a green 'New Build' button. A modal window titled 'New Build' is open in the center. It contains a 'Message' text input field with a description: 'Description of the build. If left blank, the commit message will be used once the build starts.' Below this is a 'Commit' field (marked as required) with 'HEAD' entered. Then a 'Branch' field (also marked as required) with 'main' entered. There's an 'Options' section with a dropdown arrow. At the bottom of the modal is a large green 'Create Build' button.

Develer / Example 7 - Artifacts
Pipeline uploading and downloading artifacts

All Builds Edit Steps Pipeline Settings New Build

New Build

Message

Description of the build. If left blank, the commit message will be used once the build starts.

Commit — Required

Branch — Required

Options ▾

Create Build

Pipeline dinamiche

Buildkite permette di generare
dinamicamente nuovi step in
una pipeline

Pipeline dinamiche

Buildkite permette di generare pipeline dinamiche, semplicemente passando ulteriori step a buildkite-agent:

```
./pipeline.sh | buildkite-agent pipeline upload
```

Script per pipeline dinamiche

```
# Opzione 1
echo "  - name: 'Step #${i}'"
echo "    command: 'echo \"This is step #${i}\"'"

# Opzione 2, usando heredoc
cat <<YAML
- name: 'Step #${i}'
  command: 'echo "This is step #${i}"'
YAML
```

Esempio 8 – Pipeline dinamiche

Esempio #8

Esempio 8 – Pipeline dinamiche

Develer / Example 8 - Dynamic pipeline

Pipeline that dynamically creates new steps

All Builds - Edit Steps Pipeline Settings New Build

There are no builds for this pipeline yet.

You can trigger a build using the New Build button above, or via the REST API, for example:

```
curl -H "Authorization: Bearer: $TOKEN" "https://api.buildkite.com/v2/organizations/develer-1/pipelines/example-8-dynamic-pipeline/builds" \
-X "POST" \
-F "commit=HEAD" \
-F "branch=master" \
-F "message=First build :rocket:"
```

You can also setup a GitHub webhook to auto

New Build

Message

Description of the build. If left blank, the commit message will be used once the build starts.

Commit — Required

HEAD

Branch — Required

main

Options ▾

Create Build

Conditionals and dependencies

Vediamo ora un po' di opzioni
extra per gli step

If - Conditionals e dependencies

```
steps:  
  - name: 'Hello world!'  
    command: './hello-world'  
    if: build.message !~ /skip hello world/
```

Buildkite permette di configurare i nostri step per essere eseguiti soltanto in certe condizioni.

Una delle opzioni che possiamo utilizzare è `if`.

Branches – Conditionals e dependencies

```
steps:  
  - name: 'Hello world!'  
    command: './hello-world'  
    branches: 'main'
```

Un'altra opzione è branches, per attivare uno step soltanto per alcune branch.

Esempio 9: if e branches – Conditionals e dependencies

Esempio #9

Esempio 9: if e branches – Conditionals e dependencies

Develer / Example 9 - Conditionals
Pipeline with if and branches

All Builds ▾ Edit Steps Pipeline Settings **New Build**

New Build

Message

Description of the build. If left blank, the commit message will be used once the build starts.

Commit — Required

Branch — Required

Options ▾

Create Build

Depends_on - Conditionals e dependencies

```
steps:  
  - key: 'hello-world-1'  
    command: './hello-world'  
  
  - key: 'hello-world-2'  
    command: './hello-world'  
    depends_on: 'hello-world-1'
```

Depends_on invece permette di dichiarare dipendenze esplicite tra step.

È un'alternativa più flessibile all'uso di wait, ad esempio.

Di default, lo step hello-world-2 non partirà se hello-world-1 dovesse fallire. Ma è possibile settare `allow_dependency_failure: true` per cambiare questo comportamento.

Esempio 10: `depends_on` - Conditionals e dependencies

Esempio #10

Esempio 10: depends_on - Conditionals e dependencies

Develer / Example 10 - Dependencies

 All Builds ▾

 Edit Steps

 Pipeline Settings

New Build

New Build

Message

Description of the build. If left blank, the commit message will be used once the build starts.

Commit — Required

HEAD

Branch — Required

main

Options ▾

Create Build

Concurrency - Conditionals e dependencies

```
steps:
  - key: 'hello-world-1'
    command: './hello-world'
    concurrency: 1
    concurrency_group: 'hello-world/run'
```

È anche possibile creare gruppi di step che dovranno rispettare regole di concorrenza (anche tra pipeline).

In questo caso abbiamo 2 parametri:

- concurrency_group
- concurrency

Esempio 11: concurrency - Conditionals e dependencies

Esempio #11

Esempio 11: concurrency - Conditionals e dependencies

Develer / Example 11 - 2 - Concurrency

All Builds + Edit Steps Pipeline Settings New Build

There are no builds for this pipeline yet.

You can trigger a build using the New Build button above, or via the REST API, for example:

```
curl -H "Authorization: Bearer $TOKEN" https://api.buildkite.com/v2/organizations/develer-1/pipelines/example-11-2-concurrency/builds" \
  -X "POST" \
  -F "commit=HEAD" \
  -F "branch=master" \
  -F "message=first build (rocket)"
```

You can also setup a GitHub webhook to automatically trigger builds.

New Build

Message

Description of the build. If left blank, the commit message will be used once the build starts.

Commit — Required

HEAD

Branch — Required

main

Options ▾

Create Build

Develer / Example 11 - Concurrency

All Builds + Edit Steps Pipeline Settings New Build

Pipeline with some concurrency rules.

New Build

Message

Description of the build. If left blank, the commit message will be used once the build starts.

Commit — Required

HEAD

Branch — Required

main

Options ▾

Create Build

Pipeline schedule

Buildkite permette di creare uno o più schedules per una pipeline, così da poterla eseguire a intervalli specifici

Pipeline schedule

Pipeline Settings

General

Builds

GitHub

Schedules 1

Build Badges

Personal Email Settings

An amazing schedule 

0 */4 * * *

EditDelete

Message

An amazing message for a scheduled build

Commit

HEAD

Branch

main

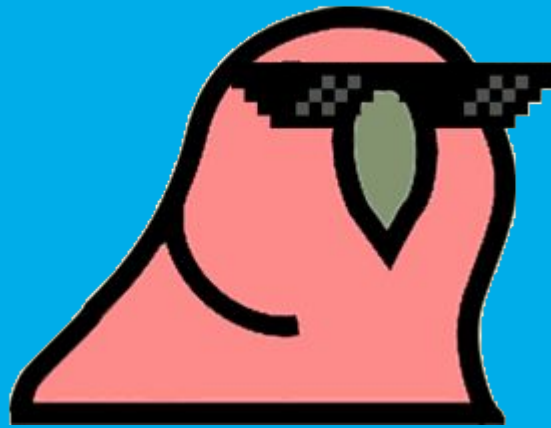
Schedule Created By

Lorenzo Savini

Recent Builds

Next build scheduled for April 01, 2021 20:00

Bonus





New Build

Message

Description of the build. If left blank, the commit message will be used once the build starts.

Commit — Required**Branch** — Required**Options** ▾**Create Build**

Lorenzo Savini

savo@develer.com

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



develer

www.develer.com