EBook Gratuito

APPRENDIMENTO jenkins

Free unaffiliated eBook created from
Stack Overflow contributors.

#jenkins

Sommario

Di ... 1

Capitolo 1: Inizia con Jenkins ... 2

Osservazioni ... 2

Versioni ... 2

Jenkins ... 2

Jenkins 1.x contro Jenkins 2.x ... 2

Examples ... 5

Installazione ... 5

Aggiornamento di jenkins (installazioni RPM) ... 6

Impostazione del proxy Nginx ... 6

Installazione del plugin da una fonte esterna ... 7

Muovi Jenkins da un PC all'altro ... 7

Configura un progetto in Jenkins ... 7

Introduzione completa di Jenkins in un unico luogo ... 8

Configura un semplice progetto di build con lo script di pipeline di Jenkins 2 ... 14

Capitolo 2: Configura Auto Git Push su una build di successo in Jenkins ... 16

introduzione ... 16

Examples ... 16

Configurazione del lavoro di push automatico ... 16

Capitolo 3: Configurazione di Build Automation per iOS utilizzando Shenzhen ... 25

Examples ... 25

Configurazione di iOS Build Automation con Shenzhen ... 25

Capitolo 4: Configurazione di Jenkins per l'automazione della build di iOS. ... 26

introduzione ... 26

Parametri ... 26

Osservazioni ... 26

Examples ... 26

Esempio di tabella di tempo ... 26

Capitolo 5: Installa Jenkins su Windows con supporto SSH per repository GitHub privati ... 28

Examples ... 28

Le richieste pull GitHub falliscono ... 28
PS Tool PSExec.exe di Microsoft ... 28
Genera una nuova chiave SSH solo per Jenkins usando PSExec o PSExec64 ... 28
Crea le credenziali di Jenkins ... 29
Esegui una richiesta di prova di prova per verificare, e il tuo fatto ... 31
Capitolo 6: Jenkins Groovy Scripting ... 33
Examples ... 33
Crea utente predefinito ... 33
Disabilita installazione guidata ... 33
Come ottenere informazioni sull'istanza di Jenkins ... 34
Come ottenere informazioni su un lavoro di Jenkins ... 34
Capitolo 7: Role Strategy Plugin ... 36
Examples ... 36
Configurazione ... 36
Gestisci ruoli ... 36
Assegna ruoli ... 37
Titoli di coda ... 40

Di

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: jenkins

It is an unofficial and free jenkins ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official jenkins.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capitolo 1: Inizia con Jenkins

Osservazioni

Jenkins è uno strumento di integrazione continua open source scritto in Java. Il progetto è stato biforcato da Hudson dopo una disputa con Oracle .

Jenkins fornisce servizi di integrazione continua per lo sviluppo di software. Si tratta di un sistema basato su server in esecuzione in un contenitore servlet come Apache Tomcat. Supporta gli strumenti SCM inclusi AccuRev, CVS, Subversion, Git, Mercurial, Perforce, Clearcase e RTC e può eseguire progetti basati su Apache Ant e Apache Maven, nonché script di shell arbitrari e comandi batch di Windows. Lo sviluppatore principale di Jenkins è Kohsuke Kawaguchi . Rilasciato sotto licenza MIT, Jenkins è un software gratuito.

Le build possono essere avviate in vari modi, incluso essere attivate da commit in un sistema di controllo versione, programmando tramite un meccanismo simile a cron, costruendo quando altre build sono state completate e richiedendo un URL di build specifico.

Versioni

Jenkins

Versione	Data di rilascio
1.656	2016/04/03
2.0	2016/04/20

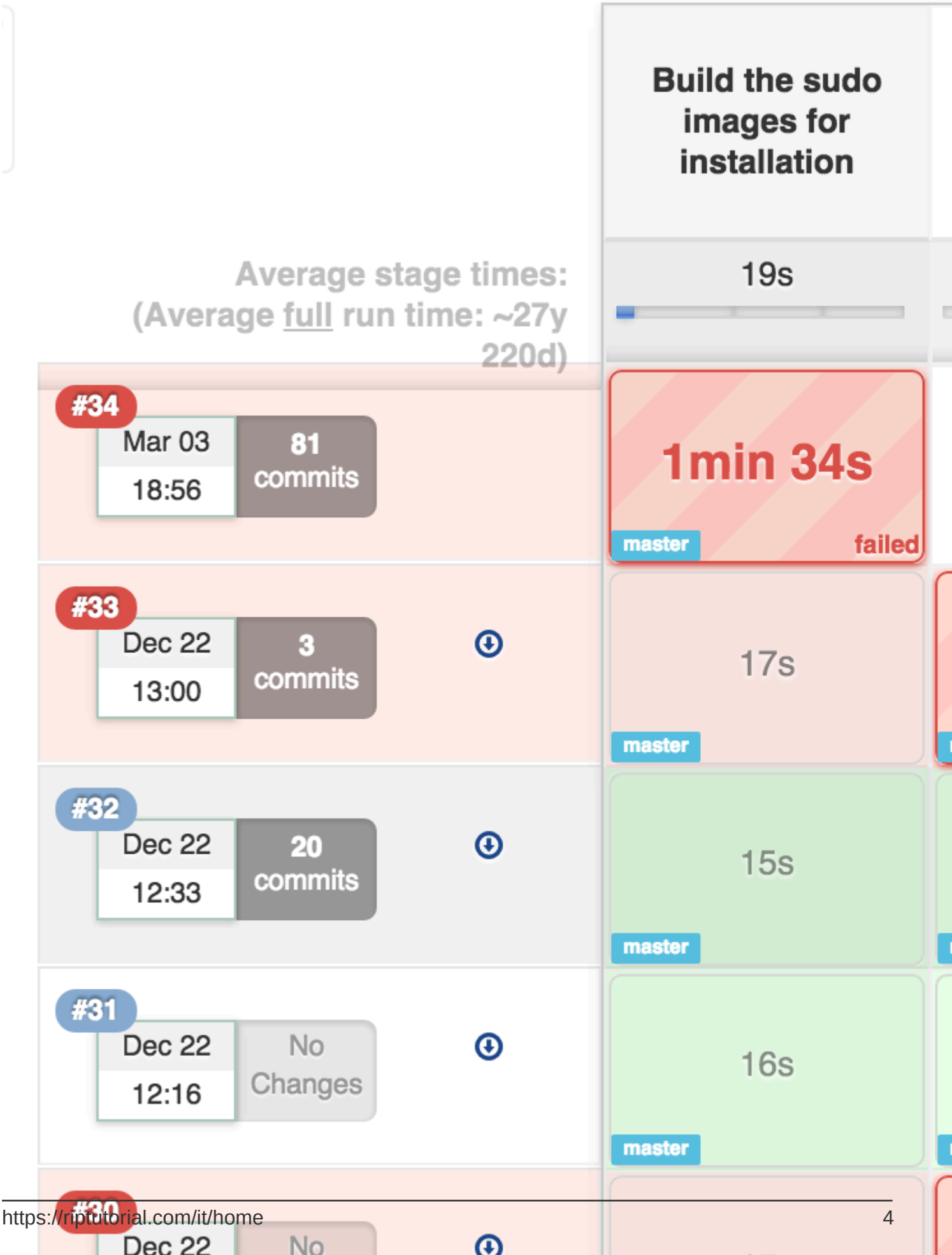
Jenkins 1.x contro Jenkins 2.x

Jenkins è (ed è tuttora) un sistema di integrazione continua (CI) che consente l'automazione del processo di sviluppo del software, come ad esempio la creazione di codice su trigger di commit SCM. Tuttavia, la crescente necessità di fornitura continua (CD) ha richiesto che Jenkins si evolva per un puro sistema CI in un mix di CI e CD. Inoltre, la necessità di industrializzare i lavori di Jenkins è cresciuta e il classico Jenkins 1.x `Freestyle/Maven jobs` iniziato a essere troppo limitato per determinate esigenze.

Sotto il plugin Jenkins 1.xa chiamato `workflow-plugin` apparso per consentire agli sviluppatori di scrivere codice per descrivere i lavori. Jenkins 2 va oltre aggiungendo il supporto integrato per `Pipeline as Code` . Il vantaggio principale è che le pipeline, essendo file di script di Groovy, possono essere più complesse dei lavori freestyle configurati dall'interfaccia utente e possono essere controllate dalla versione. Jenkins 2 aggiunge anche una nuova interfaccia che semplifica

la visualizzazione di diverse "fasi" definite in una pipeline e segue lo stato di avanzamento dell'intera pipeline, come di seguito:

Stage View


Build the sudo images for installation
Average stage times:
(Average full run time: ~27y 220d)
19s
#34
Mar 03
18:56
81 commits
1min 34s
master
failed
#33
Dec 22
13:00
3 commits
17s
master
#32
Dec 22
12:33
20 commits
15s
master
#31
Dec 22
12:16
No Changes
16s
master

Jenkins 2 Overview .

Inoltre, il changelog completo è disponibile sul sito Web di Jenkins.

Examples

Installazione

> Per sistemi basati su apt-get come Ubuntu

Aggiungi il repository Jenkins:

```
wget -q -O - https://jenkins-ci.org/debian/ Jenkins-ci.org.key | sudo apt-key
```

```
sudo sh -c 'echo deb http://pkg.jenkins-ci.org/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
```

Aggiorna i sorgenti e installa Jenkins:

```
sudo apt-get update

sudo apt-get install jenkins
```

Un utente jenkins è ora creato e, per impostazione predefinita, Jenkins verrà eseguito sulla porta 8080.

> Per distribuzioni basate su RPM come Red Hat Enterprise Linux (RHEL), CentOS, Fedora o Scientific Linux

Per scaricare il file di repository per la versione stabile:

```
sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo
```

O se vuoi le ultime uscite settimanali:

```
sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat/jenkins.repo
```

Importa la chiave pubblica:

```
sudo rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key
```

Installa Jenkins usando yum:

```
sudo yum install jenkins
```

Jenkins richiede java per funzionare, per installarlo:

```
sudo yum install java
```

Per avviare / arrestare / riavviare i jenkins utilizzare:

```
sudo service jenkins start/stop/restart
```

Aggiornamento di jenkins (installazioni RPM)

1. Backup della directory home di jenkins
2. Sostituisci jenkins.war nel seguente percorso con il nuovo file WAR. / Usr / lib / Jenkins / jenkins.war`
3. Riavvia Jenkins
4. Controlla i plugin aggiunti e sblocca se necessario
5. Ricarica la configurazione dal disco

nota: per gli aggiornamenti di Jenkins 2 per il server dell'app jetty in bundle, disabilitare la porta AJP (impostare `JENKINS_AJP_PORT="-1"`) in `/etc/sysconfig/jenkins` .

Impostazione del proxy Nginx

Nativamente, Jenkins gira sulla porta 8080. Possiamo stabilire un proxy dalla porta 80 -> 8080, quindi è possibile accedere a Jenkins tramite:

```
http://<url>.com
```

invece del valore predefinito

```
http://<url>.com:8080
```

Inizia installando Nginx.

```
sudo aptitude -y install nginx
```

Rimuovere le impostazioni predefinite per Nginx

```
cd /etc/nginx/sites-available

sudo rm default ../sites-enabled/default
```

Crea il nuovo file di configurazione

```
sudo touch jenkins
```

Copia il seguente codice nel file `jenkins` appena creato.

```
upstream app_server {
  server 127.0.0.1:8080 fail_timeout=0;
}

server {
  listen 80;
  listen [::]:80 default ipv6only=on;
  server_name ;

  location / {
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_redirect off;
```

```
    if (!-f $request_filename) {
        proxy_pass http://app_server;
        break;
    }
  }
}
```

Creare un collegamento simbolico tra siti disponibili e abilitati per i siti:

```
sudo ln -s /etc/nginx/sites-available/jenkins /etc/nginx/sites-enabled/
```

Riavvia il servizio proxy Nginx

```
sudo service nginx restart
```

Jenkins sarà ora accessibile dalla porta 80.

Installazione del plugin da una fonte esterna

```
java -jar [Path to client JAR] -s [Server address] install-plugin [Plugin ID]
```

Il JAR del client deve essere il file JAR CLI, non lo stesso JAR / WAR che esegue Jenkins stesso. Gli ID univoci possono essere trovati su una rispettiva pagina di plugin sulla wiki della Jenkins CLI (https://wiki.jenkins-ci.org/display/JENKINS/Plugins)

Muovi Jenkins da un PC all'altro

Questo ha funzionato per me passare da Ubuntu 12.04 (Jenkins versione 1.628) a Ubuntu 16.04 (Jenkins versione 1.651.2). Per prima cosa ho installato Jenkins dai repository .

1. Fermare entrambi i server Jenkins
2. Copia `JENKINS_HOME` (es. / Var / lib / jenkins) dal vecchio server a quello nuovo. Da una console nel nuovo server:

   ```
   rsync -av username@old-server-IP:/var/lib/jenkins/ /var/lib/jenkins/
   ```

3. Avvia il tuo nuovo server Jenkins

Potresti non averne bisogno, ma dovevo

- `Manage Jenkins` e `Reload Configuration from Disk` .
- Disconnetti e connetti di nuovo tutti gli schiavi.
- Controlla che in `Configure System > Jenkins Location` , l' `Jenkins URL` sia correttamente assegnato al nuovo server Jenkins.

Configura un progetto in Jenkins

Qui controlleremo l'ultima copia del codice del nostro progetto, eseguiremo i test e renderemo l'applicazione live. Per ottenerlo, segui i passaggi seguenti:

1. Apri Jenkins nel browser.
2. Fare clic sul collegamento ***Nuovo lavoro*** .
3. Inserisci il nome del progetto e seleziona il link ***Costruisci un progetto software libero*** .
4. Clicca sul pulsante ***Ok*** .
5. Sotto la ***sezione Gestione codice sorgente*** , seleziona la casella radio accanto allo strumento di gestione del codice sorgente. Nel mio caso ho selezionato ***Git*** .

Fornisci l'url del repository `git://github.com/example/example.git` come `git://github.com/example/example.git`

6. Sotto i ***trigger Build*** , seleziona la casella radio accanto a ***Poll SCM*** .
7. Fornire `*****` nella casella di ***calendario*** . Questa casella è responsabile per attivare la build a intervalli regolari. `*****` specifica che, il processo verrà attivato ogni minuto per le modifiche nel repository git.
8. Sotto la sezione ***Build*** , fai clic sul pulsante ***Aggiungi Step di costruzione*** e poi seleziona l'opzione con cui vuoi costruire il progetto. Ho selezionato ***Execute Shell*** . Nella casella di comando scrivi il comando per creare, eseguire i test e distribuirlo a prod.
9. Scorri verso il basso e ***Salva*** .

Così sopra abbiamo configurato un progetto di base in Jenkins che attiverà la build ad ogni minuto per il cambiamento nel tuo repository git. Nota: per configurare il progetto complesso, potrebbe essere necessario installare alcuni plugin in Jenkins.

Introduzione completa di Jenkins in un unico luogo

1. Jenkins:

Jenkins è uno strumento di integrazione continua open source scritto in Java. Il progetto è stato biforcato da Hudson dopo una disputa con Oracle.

In breve, Jenkins è il principale server di automazione open source. Costruito con Java, fornisce centinaia di plug-in per supportare la creazione, il test, l'implementazione e l'automazione praticamente per qualsiasi progetto.

Caratteristiche: Jenkins offre le seguenti funzionalità principali e molti altri possono essere aggiunti tramite plugin:

Installazione semplice: basta eseguire java -jar jenkins.war, distribuirlo in un contenitore servlet. Nessuna installazione aggiuntiva, nessun database. Preferisci un programma di installazione o un pacchetto nativo? Abbiamo anche quelli. Facile configurazione: Jenkins può essere completamente configurato dalla sua intuitiva GUI Web con ampi controlli in tempo reale e aiuto in linea. Ecosistema di plugin ricco: Jenkins si integra praticamente con ogni SCM o strumento di creazione esistente. Visualizza plug-in. Estensibilità: la maggior parte delle parti di Jenkins possono essere estese e modificate, ed è facile creare nuovi plugin Jenkins. Questo ti consente di personalizzare Jenkins in base alle tue esigenze. Generazioni distribuite: Jenkins può distribuire carichi di build / test su più computer con diversi sistemi operativi. Costruire software per OS X, Linux e Windows? Nessun problema.

Installazione :

```
$ wget -q -O - https://jenkins-ci.org/debian/jenkins-ci.org.key | sudo apt-key add -

$ sudo sh -c 'echo deb http://pkg.jenkins-ci.org/debian binary/ >
/etc/apt/sources.list.d/jenkins.list'
$ sudo apt-get update
$ sudo apt-get install jenkins
to do more refer link :
```

Rif: https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+on+Ubuntu

Rif: http://www.vogella.com/tutorials/Jenkins/article.html

Rif: https://wiki.jenkins-ci.org/display/JENKINS/Meet+Jenkins

Directory **JENKINS_HOME** Jenkins ha bisogno di spazio su disco per eseguire build e mantenere archivi. Puoi controllare questa posizione dalla schermata di configurazione di Jenkins. Per impostazione predefinita, questo è impostato su ~ /. Jenkins, ma è possibile cambiarlo in uno dei seguenti modi: Impostare la variabile di ambiente "JENKINS_HOME" nella nuova directory home prima di avviare il contenitore servlet. Impostare la proprietà di sistema "JENKINS_HOME" sul contenitore servlet. Impostare la voce dell'ambiente JNDI "JENKINS_HOME" nella nuova directory. Vedere la raccolta di documentazione specifica del contenitore per ulteriori informazioni su come eseguire questa operazione per il contenitore. Puoi cambiare posizione anche dopo aver usato Jenkins per un po '. Per fare ciò, ferma completamente Jenkins, sposta il contenuto dalla vecchia JENKINS_HOME alla nuova casa, imposta la nuova JENKINS_HOME e riavvia Jenkins. JENKINS_HOME ha una struttura di directory abbastanza ovvia che assomiglia al seguente:

JENKINS_HOME

```
+- config.xml     (jenkins root configuration)
 +- *.xml          (other site-wide configuration files)
 +- userContent    (files in this directory will be served under your
http://server/userContent/)
 +- fingerprints   (stores fingerprint records)
 +- plugins        (stores plugins)
 +- workspace (working directory for the version control system)
     +- [JOBNAME] (sub directory for each job)
 +- jobs
     +- [JOBNAME]      (sub directory for each job)
         +- config.xml     (job configuration file)
         +- latest         (symbolic link to the last successful build)
         +- builds
             +- [BUILD_ID]     (for each build)
                 +- build.xml      (build result summary)
                 +- log            (log file)
                 +- changelog.xml  (change log)
```

Lavori di costruzione di Jenkins:

Creare un nuovo lavoro di compilazione in Jenkins è semplice: basta fare clic sulla voce di menu "Nuovo lavoro" sulla dashboard di Jenkins. Jenkins supporta diversi tipi di lavori di compilazione, che vengono presentati quando si sceglie di creare un nuovo lavoro

Progetto di software freestyle

I lavori di creazione di stile libero sono lavori di creazione generici, che offrono il massimo della flessibilità.

Progetto Maven Il " **progetto** maven2 / 3" è un lavoro di costruzione appositamente adattato ai progetti Maven. Jenkins comprende i file Maven Pom e le strutture di progetto e può utilizzare le informazioni raccolte dal file pom per ridurre il lavoro necessario per impostare il progetto.

Flusso di lavoro

Orchestra attività a lungo termine che possono estendersi su più build slave. Adatto per la costruzione di condotte e / o l'organizzazione di attività complesse che non si adattano facilmente al tipo di lavoro in stile libero.

Monitorare un lavoro esterno Il lavoro di compilazione "Monitora un lavoro esterno" consente di tenere d'occhio i processi non interattivi, come i lavori cron.

Lavoro di configurazione multipla Il "progetto di configurazione multipla" (noto anche come "progetto di matrice") consente di eseguire lo stesso lavoro di compilazione in molte configurazioni diverse. Questa potente funzionalità può essere utile per testare un'applicazione in molti ambienti diversi, con database diversi o anche su macchine di compilazione diverse.

1. Costruzione di un progetto software (stile libero)

Jenkins può essere utilizzato per eseguire il tipico lavoro di build del server, come eseguire build continue / ufficiali / notturne, eseguire test o eseguire alcune operazioni batch ripetitive. Questo è chiamato "progetto software di stile libero" in Jenkins. Impostazione del progetto Vai alla pagina principale di Jenkins, seleziona "Nuovo lavoro", quindi scegli "Crea un progetto software in stile libero". Questo tipo di lavoro comprende i seguenti elementi: SCM facoltativo, come CVS o Subversion in cui risiede il codice sorgente. trigger opzionali per controllare quando Jenkins eseguirà build. una sorta di script di build che esegue la build (ant, maven, script della shell, file batch, ecc.) dove il vero lavoro avviene in modo facoltativo per raccogliere informazioni dalla build, come l'archiviazione delle risorse e / o la registrazione di javadoc e test risultati. passaggi facoltativi per notificare ad altre persone / sistemi il risultato della compilazione, come l'invio di e-mail, messaggi istantanei, aggiornamento del tracker dei problemi, ecc.

Build per progetti di controllo non di origine A volte è necessario creare un progetto semplicemente a scopo dimostrativo o l'accesso a un repository SVN / CVS non è disponibile. Scegliendo di configurare il progetto come "Nessuno" in "Gestione codice sorgente" dovrai:

1. Costruisci il Progetto almeno una volta, (fallirà), ma Jenkins creerà la struttura jenkins / workspace / PROJECTNAME /
2. Copia i file di progetto in jenkins / workspace / PROJECTNAME /
3. Costruisci di nuovo e configura in modo appropriato

Jenkins imposta le variabili d'ambiente

Quando viene eseguito un lavoro Jenkins, imposta alcune variabili di ambiente che è possibile utilizzare nello script della shell, nel comando batch, nello script Ant o Maven POM. Vedere l'elenco delle variabili facendo clic su ENVIRONMENT_VARIABLE

Configurazione di build automatiche

Le build in Jenkins possono essere attivate periodicamente (in base a una pianificazione, specificata nella configurazione) o quando sono state rilevate modifiche all'origine nel progetto oppure possono essere attivate automaticamente richiedendo l'URL:

http: // yourhost / Jenkins / lavoro / PROJECTNAME / build

Questo ti permette di collegare le build di Jenkins in una varietà di configurazioni. Per ulteriori informazioni (in particolare fare ciò con la sicurezza abilitata), vedere API di accesso remoto.

Crea per modifiche di origine

Puoi fare in modo che Jenkins esamini il tuo Revision Control System per le modifiche. È possibile specificare la frequenza con cui Jenkins esegue il polling del sistema di controllo di revisione usando la stessa sintassi di crontab su Unix / Linux. Tuttavia, se il periodo di polling è inferiore a quello necessario per eseguire il polling del sistema di controllo di revisione, è possibile che si ottengano più build per ogni modifica. È necessario regolare il periodo di polling in modo che sia più lungo del tempo necessario per eseguire il polling del sistema di controllo di revisione o utilizzare un trigger post-commit. È possibile esaminare il registro di polling per ciascuna build per vedere quanto tempo è necessario per eseguire il polling del sistema.

In alternativa, invece di eseguire il polling su un intervallo fisso, puoi utilizzare un trigger URL (descritto sopra), ma con / polling anziché / build alla fine dell'URL. Ciò fa sì che Jenkins esamini l'SCM per le modifiche anziché costruirlo immediatamente. Ciò impedisce a Jenkins di eseguire una build senza modifiche rilevanti per i commit che riguardano moduli o rami non correlati al lavoro. Quando si utilizza / polling il lavoro deve essere configurato per il polling, ma la pianificazione può essere vuota.

Crea via e-mail (sendmail)

Se si ha l'account root del proprio sistema e si sta utilizzando sendmail, ho trovato il modo più semplice di modificare / etc / alias e aggiungere la seguente voce: jenkins-foo: "| / bin / wget -o / dev / null

http: // YOURHOST / jenkins / job / PROJECTNAME / build "

e quindi eseguire il comando "newaliases" per far sapere a sendmail del cambiamento. Ogni volta che qualcuno invia una e-mail a "jenkins-foo @ yoursystem", questo attiverà una nuova build. Vedi questo per maggiori dettagli sulla configurazione di sendmail. Crea via e-mail (qmail) Con qmail, puoi scrivere /var/qmail/alias/.qmail-jenkins come segue: | / bin / wget -o / dev / null http: // YOURHOST / jenkins / job / PROJECTNAME / costruire "

2. Costruire un progetto Maven

Jenkins offre un tipo di lavoro dedicato a Maven 2/3. Questo tipo di lavoro integra profondamente Jenkins con Maven 2/3 e offre i seguenti vantaggi rispetto al più generico progetto software in stile libero.

Jenkins analizza le POM di Maven per ottenere molte delle informazioni necessarie per svolgere il proprio lavoro. Di conseguenza, la quantità di configurazione viene drasticamente ridotta.

Jenkins ascolta l'esecuzione di Maven e capisce cosa dovrebbe essere fatto da solo. Ad esempio, registrerà automaticamente il rapporto JUnit quando Maven esegue la fase di test. O se esegui l'obiettivo javadoc, Jenkins registrerà automaticamente javadoc.

Jenkins crea automaticamente dipendenze tra progetti che dichiarano dipendenze tra SNAPSHOT. Vedi sotto. Quindi, per lo più, devi solo configurare le informazioni SCM e quali obiettivi vorresti correre, e Jenkins scoprirà tutto il resto.

Questo tipo di progetto può fornire automaticamente le seguenti funzionalità:

Archivia gli artefatti prodotti da una build

Pubblica i risultati dei test

Attivare i lavori per progetti che sono dipendenze downstream

Distribuisci le tue risorse in un repository Maven

Risultati dei test di breakout per modulo

Ricostruisci facoltativamente solo i moduli modificati, accelerando le tue build

Creazione automatica del concatenamento dalle dipendenze dei moduli

Jenkins legge le dipendenze del tuo progetto dal tuo POM, e se sono anche basate su Jenkins, i trigger sono configurati in modo tale che una nuova build in una di queste dipendenze avvii automaticamente una nuova build del tuo progetto. Jenkins comprende tutti i tipi di dipendenza in POM. Vale a dire, padre POM

```
<dependencies> section of your project
<plugins> section of your project
<extensions> section of your project
<reporting> section of your project
```

Questo processo tiene conto delle versioni, quindi puoi avere più versioni / rami del tuo progetto sullo stesso Jenkins e determinerà correttamente le dipendenze. **Nota** che gli intervalli di versione delle dipendenze non sono supportati, vedi [https://issues.jenkins-ci.org/browse/JENKINS-2787][1] per il motivo.

Questa funzione può essere disabilitata su richiesta - vedere l'opzione di configurazione Build ogni volta che viene creata una dipendenza SNAPSHOT

Installazione :

1. vai in Gestisci Jenkins >> configura Sistema

2. nella scheda di prova "Fai clic su installazione di Maven .....

Puoi fare in modo che Jenkins installi automaticamente una versione specifica di Maven o fornisca un percorso per un'installazione Maven locale (puoi configurare tutte le versioni di Maven per i tuoi progetti di build come desideri e utilizzare diverse versioni di Maven per diversi progetti. Se spuntate la casella di controllo Installa automaticamente, Jenkins scaricherà e installerà la versione richiesta di Maven per voi e la installerà nella directory degli strumenti nella home directory di Jenkins.

Come usarlo

Innanzitutto, è necessario configurare un'installazione Maven (questo passaggio può essere saltato se si utilizza DEV @ cloud). Questo può essere fatto andando alla schermata di configurazione del sistema (Gestisci Jenkins-> Configura Sistema). Nella sezione "Installazioni Maven", 1) fai clic sul pulsante Aggiungi, 2) assegna un nome come "Maven 3.0.3" e quindi 3) scegli la versione dal menu a discesa.

Ora, Jenkins installerà automaticamente questa versione ogni volta che è necessario (su qualsiasi nuova macchina di build, ad esempio) scaricandola da Apache e decomprimendola.

Crea un nuovo lavoro di Maven:

1. Facendo clic su "Nuovo lavoro / Nuovo elemento" sulla mano sinistra
2. Dagli un nome
3. Scegli il "Costruisci un progetto Maven 2/3"
4. Salva il tuo lavoro

Ora devi configurare il tuo lavoro

1. Scegli l'SCM che vuoi usare (ad esempio usando git)

2. scegli un target Maven da chiamare

3. aggiungi URL e credenziali del repository.

4. controlla il repository privato dell'utente:

È inoltre possibile definire il percorso di custodia per lo stesso.

5. Costruisci il progetto

Crea il tuo progetto facendo clic su build now e fai clic sulla barra di avanzamento nella parte sinistra "Build Executor Status" per vedere come jenkins installa Maven, controlla il tuo progetto e lo crea usando Maven.

Registrazione:

https://wiki.jenkins-ci.org/display/JENKINS/Logging

Console di script:

Utile per la risoluzione dei problemi, la diagnostica o l'aggiornamento in batch dei lavori Jenkins

fornisce una console di script che consente di accedere a tutti gli interni di Jenkins. Questi script sono scritti in Groovy e in questa pagina ne troverai alcuni esempi.

Configura un semplice progetto di build con lo script di pipeline di Jenkins 2

Qui creeremo una pipeline Groovy in Jenkins 2 per fare i seguenti passi:

- Verifica ogni 5 minuti se il nuovo codice è stato impegnato nel nostro progetto
- Codice di checkout dal repository SCM
- Maven compilare il nostro codice Java
- Esegui i nostri test di integrazione e pubblica i risultati

Ecco i passi che faremo:

1. Assicurati di avere almeno una versione di Jenkins 2.0 (puoi verificarla nell'angolo in basso a destra della pagina) come:

2. Nella home page di Jenkins, fai clic su **Nuovo elemento**

3. Inserisci il nome del progetto e seleziona **Pipeline**

4. Nella sezione **Build Triggers** , seleziona l'opzione **Poll SCM** e aggiungi la seguente programmazione CRON di 5 minuti: `*/5 * * * *`

5. Nella sezione **Pipeline** , selezionare **Pipeline Script** o **Pipeline Script da SCM**

6. Se hai selezionato **Script pipeline da SCM** nel passaggio precedente, ora devi specificare l'URL del **repository** SCM (Git, Mercurial, Subversion) nell'URL del **repository** come `http://github.com/example/example.git` . Devi anche specificare il **percorso** dello **script** del tuo file di script Groovy nel tuo repository example.git, ad esempio `pipelines/example.groovy`

7. Copia il seguente codice Groovy, direttamente nella finestra di script di Groovy se hai precedentemente fatto clic su **Pipeline Script** o nel tuo `example.groovy` se hai scelto **Pipeline Script da SCM**

```
node('remote') {
    // Note : this step is only needed if you're using direct Groovy scripting
    stage 'Checkout Git project'
    git url: 'https://github.com/jglick/simple-maven-project-with-tests.git'
    def appVersion = version()
    if (appVersion) {
        echo "Building version ${appVersion}"
    }

    stage 'Build Maven project'
    def mvnHome = tool 'M3'
    sh "${mvnHome}/bin/mvn -B -Dmaven.test.failure.ignore verify"
    step([$class: 'JUnitResultArchiver', testResults: '**/target/surefire-reports/TEST-
*.xml'])
}
```

```
def version() {
    def matcher = readFile('pom.xml') =~ '<version>(.+)</version>'
    matcher ? matcher[0][1] : null
}
```

Ecco, ora dovresti essere in grado di compilare e testare il tuo primo progetto Jenkins usando la pipeline di Jenkins 2 Groovy.

Leggi Inizia con Jenkins online: https://riptutorial.com/it/jenkins/topic/919/inizia-con-jenkins

Capitolo 2: Configura Auto Git Push su una build di successo in Jenkins

introduzione

Questo documento ti guiderà attraverso i passaggi per configurare un lavoro Jenkins che permetta all'utente di impostare il push automatico su build di successo. L'operazione push può essere controllata dall'utente. L'utente può scegliere se vuole eseguire l'operazione di push automatico su build di successo o meno.

Examples

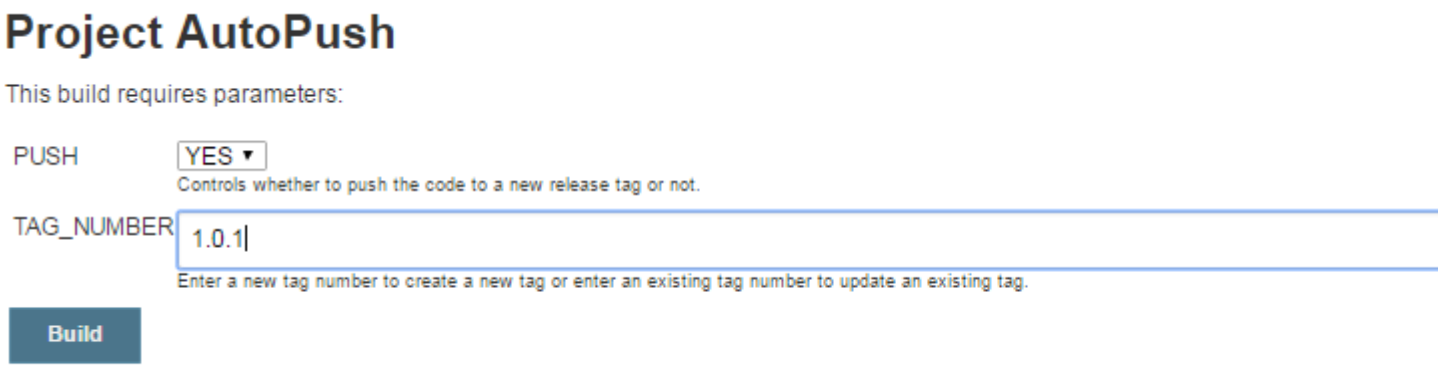
Configurazione del lavoro di push automatico

Crea un lavoro di costruzione (in base alle tue esigenze). Per questo esempio ho creato un lavoro freestyle (AutoPush) per eseguire la build ANT.

Creeremo due variabili, PUSH (Choice Parameter) e TAG_NUMBER (String Parameter).

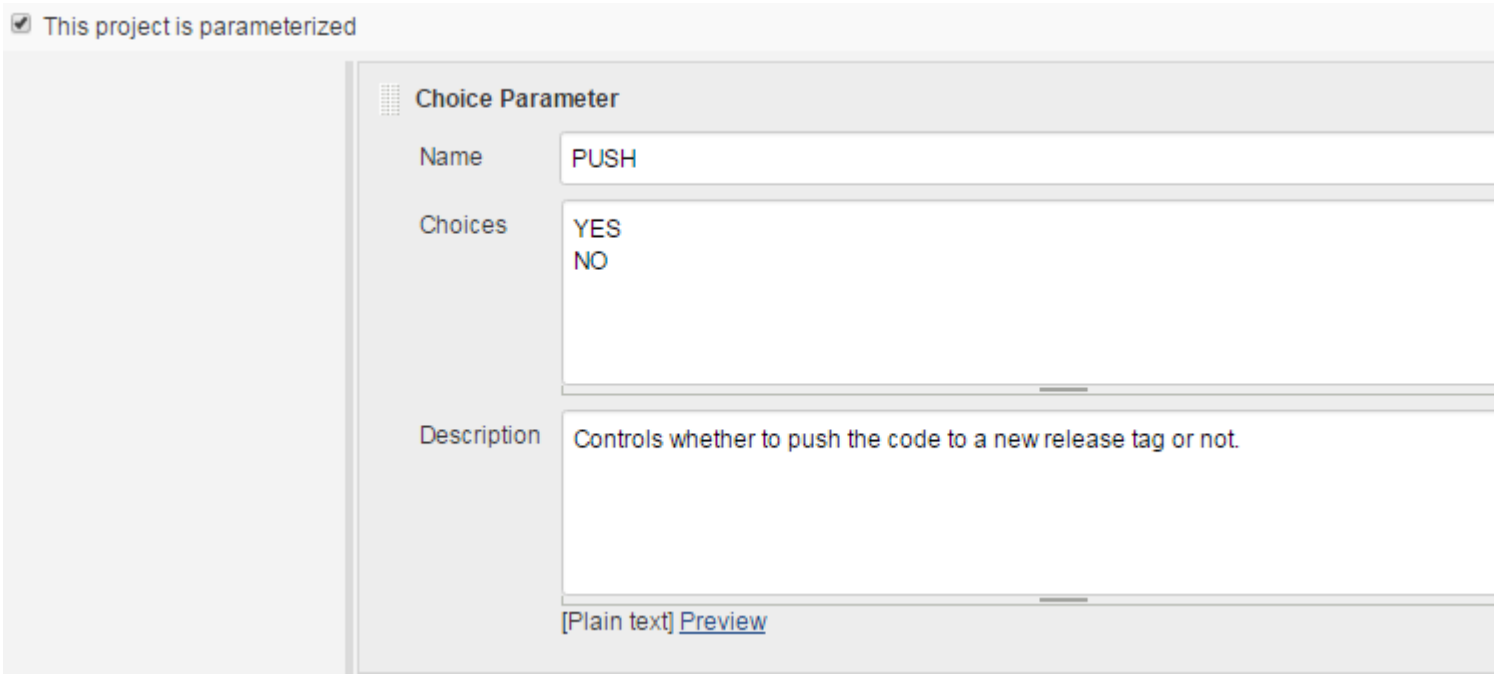
Possiamo scegliere il valore YES o NO per PUSH, questo deciderà se spingere il codice su un tag o meno su build di successo.

Possiamo specificare un nome di tag (ad esempio 1.0.1) per TAG_NUMBER per creare un nuovo tag (ad esempio 1.0.1) nel repository remoto con lo stesso nome o specificare un nome di tag esistente per aggiornare un tag esistente.

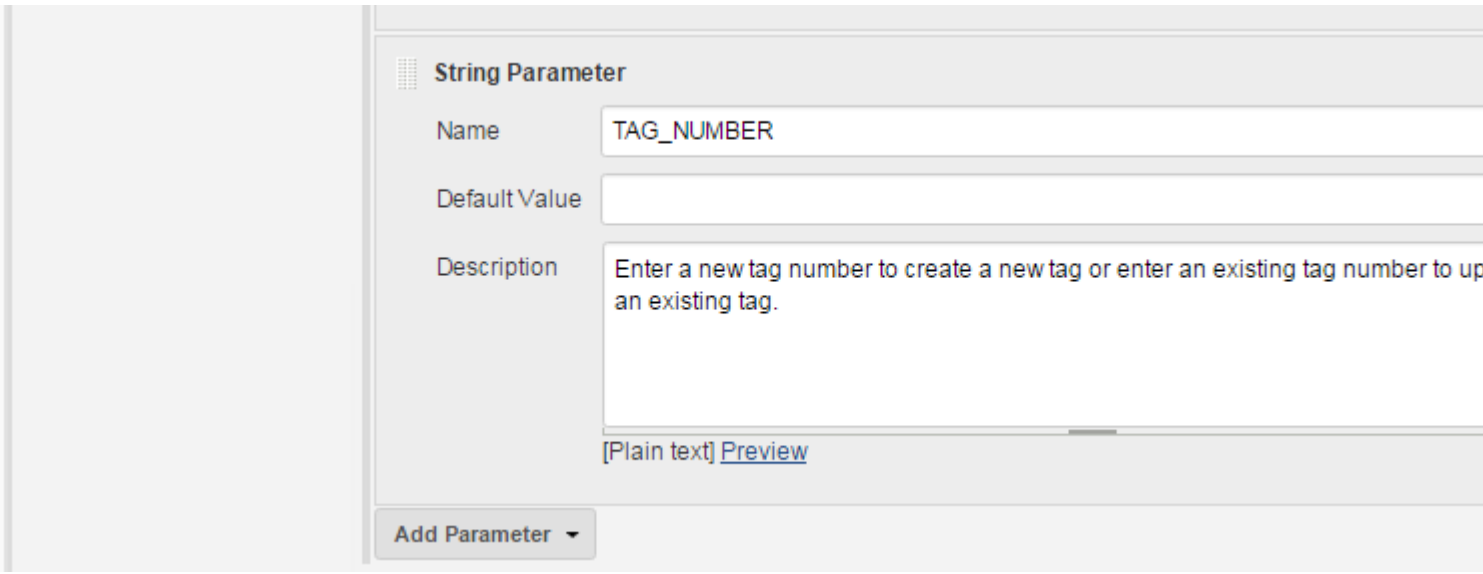


Passiamo ora alla configurazione del lavoro.

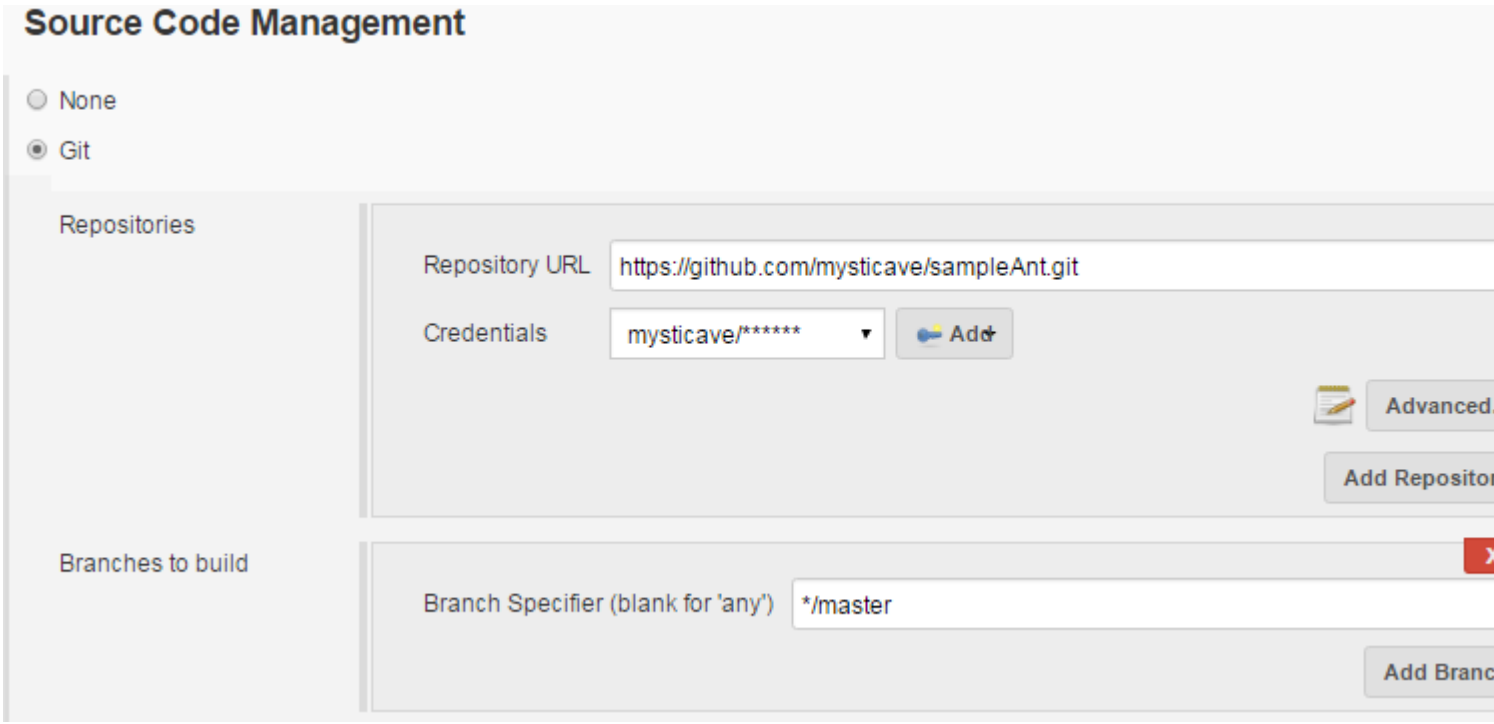
1. Seleziona la casella di controllo "Questo progetto è parametrizzato" e crea un parametro di scelta chiamato "PUSH" e fornisci SÌ e NO come opzioni. Questo parametro deciderà se si desidera inviare il codice a un Tag / Release specifico o meno.



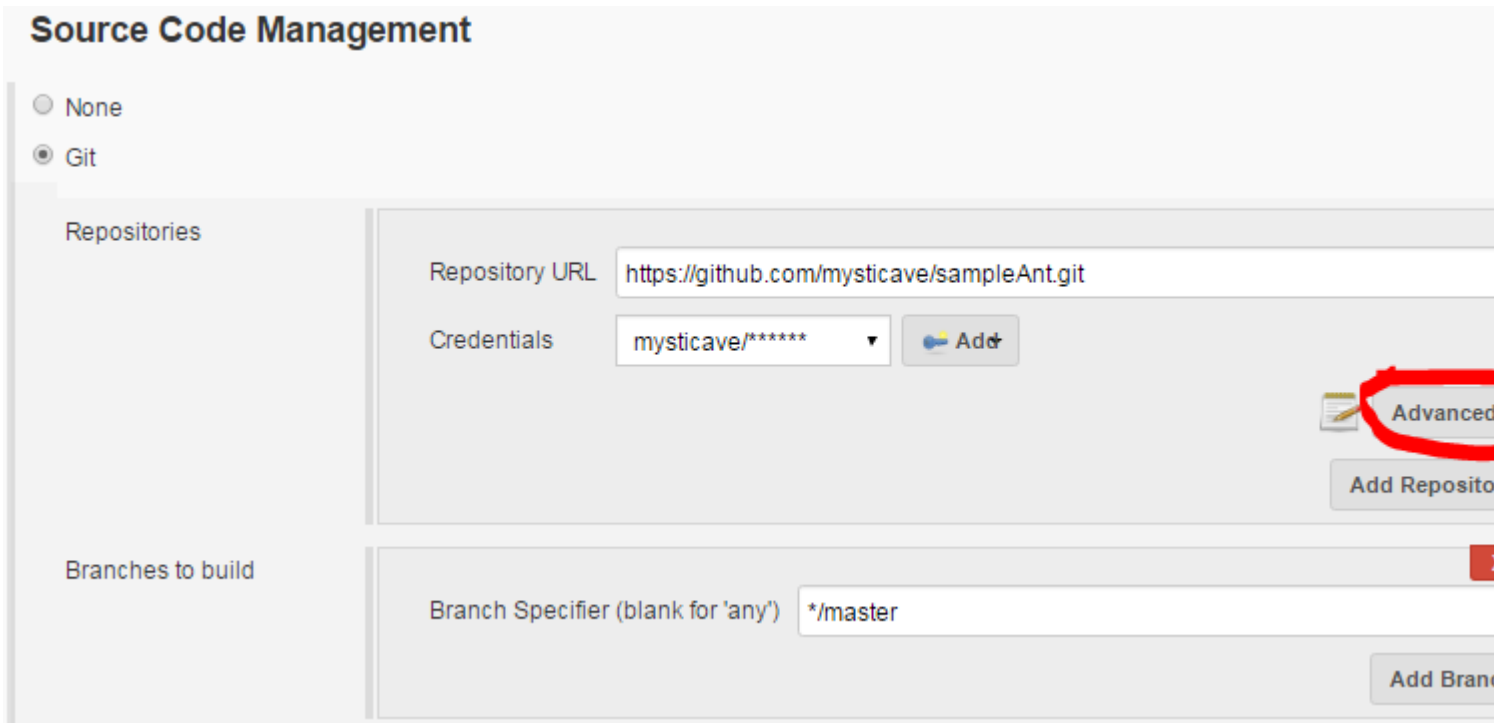
2. Quindi crea un parametro stringa chiamato "TAG_NUMBER", utilizzando questo parametro possiamo specificare un nuovo numero di tag per creare un nuovo tag o specificare un numero di tag esistente per aggiornare un tag esistente.

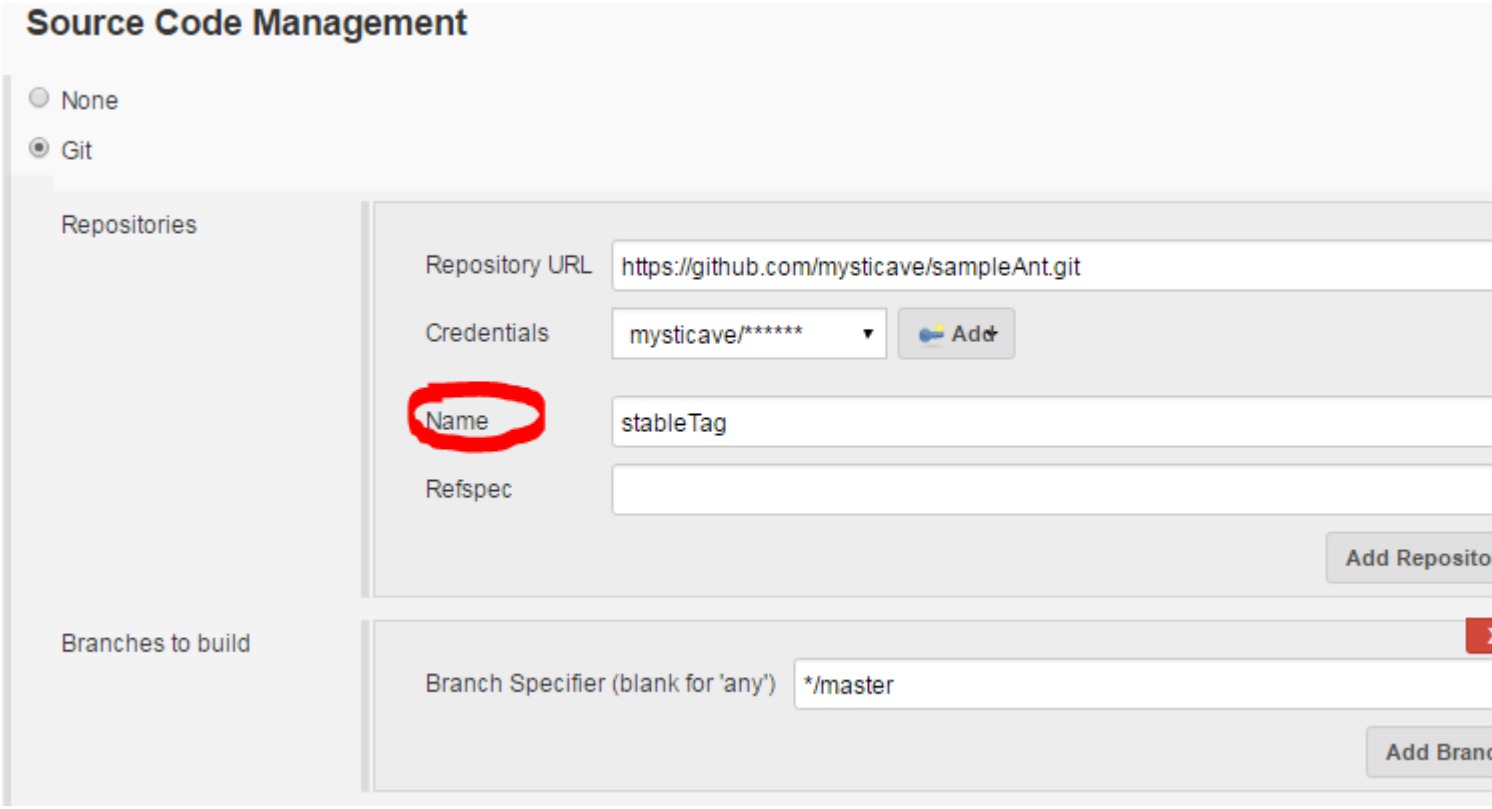


3. Nella sezione Gestione codice sorgente scegli Git e fornisci l'URL del repository. Questo repository contiene il codice sorgente che si sta per costruire e dopo una build di successo verrà creato un tag di rilascio sullo stesso repository.



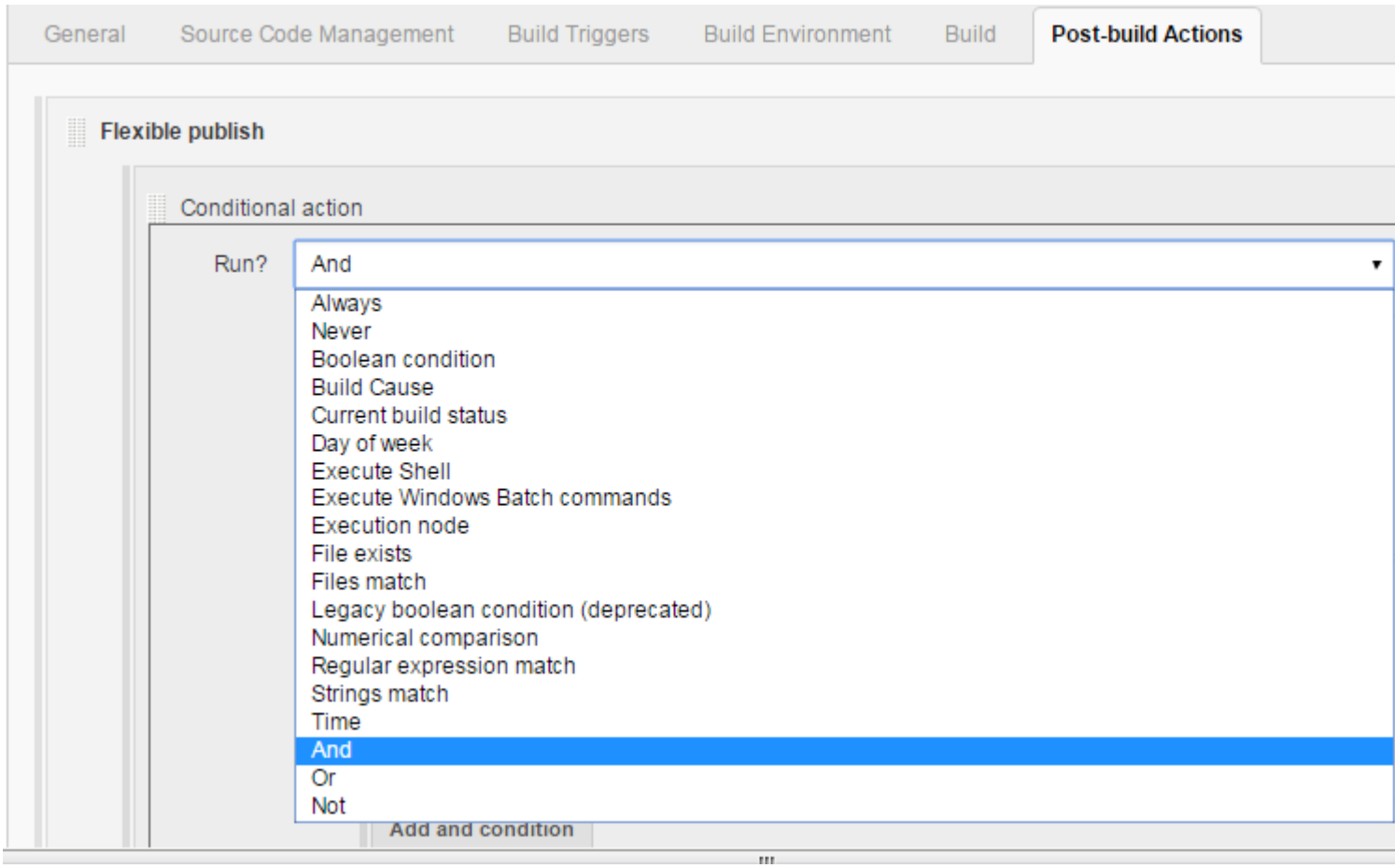
4. Dopo aver aggiunto i dettagli del repository, fare clic su Avanzate e fornire un nome al repository che verrà successivamente indicato nel plug-in Git Publisher per identificare il repository.

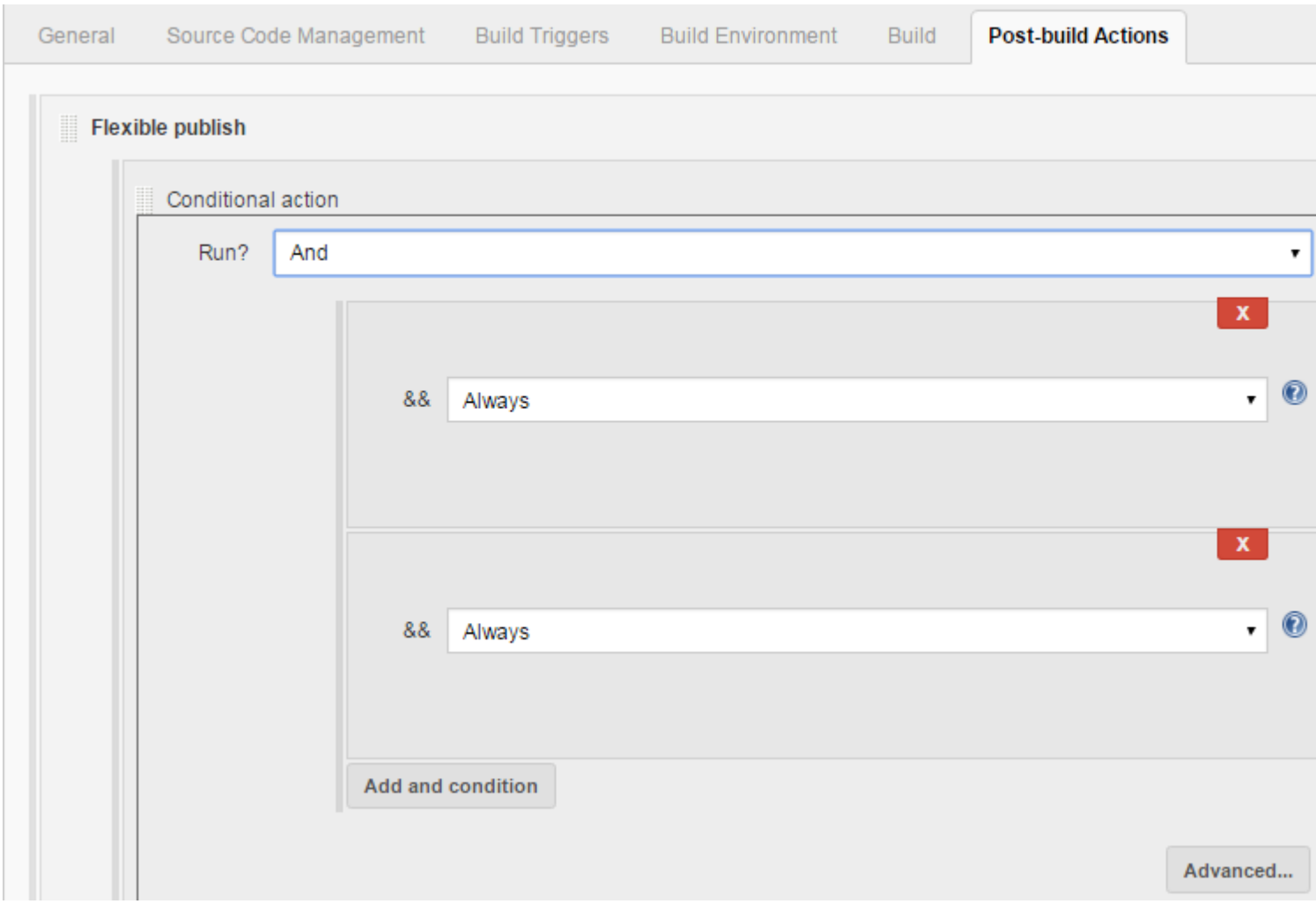


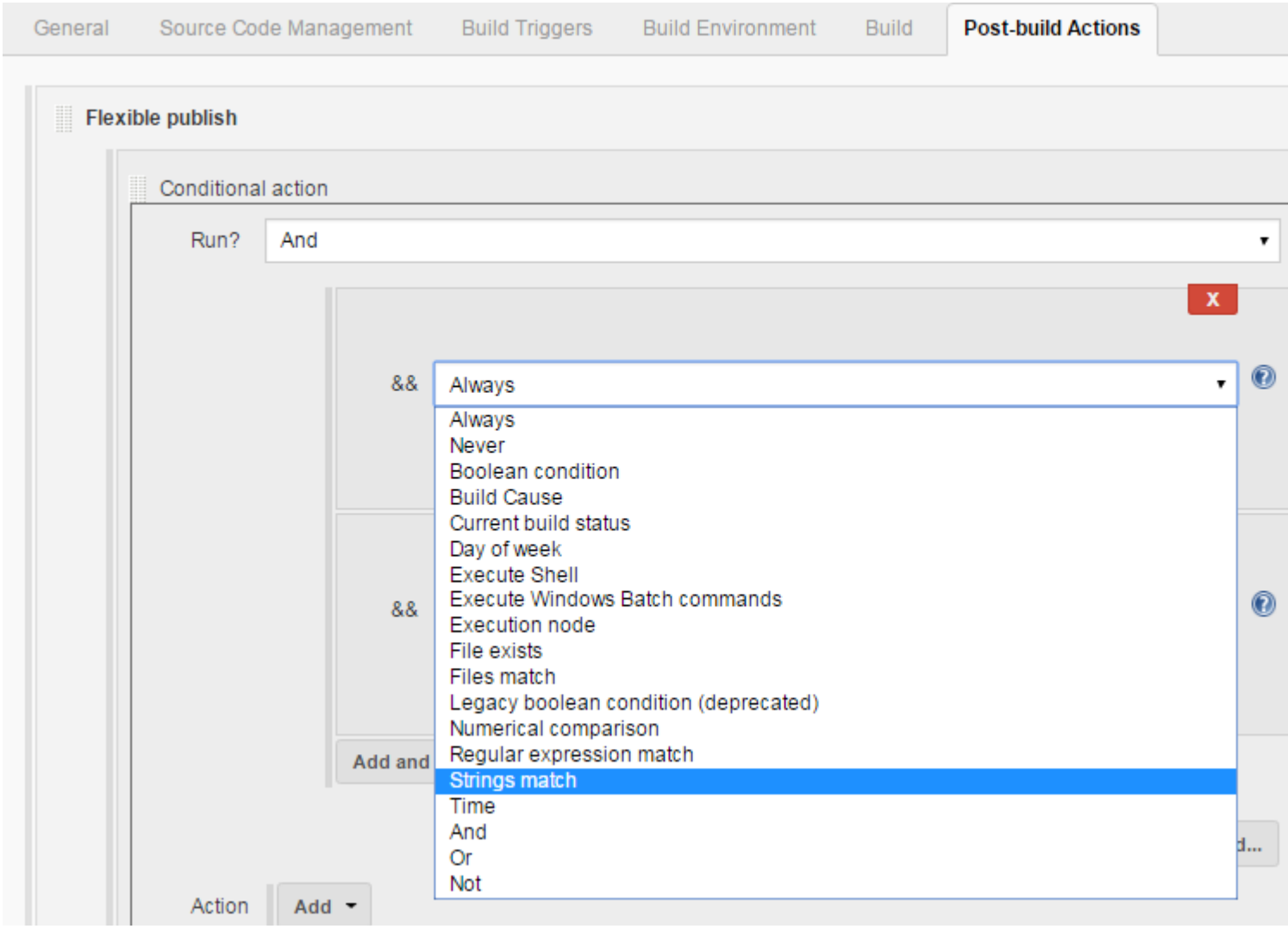


5. Quindi aggiungi il passaggio di costruzione. In questo esempio sto costruendo un progetto ANT.

6. Ora nella sezione "Azioni post-compilazione" seleziona il plugin "Flexi Publish". Seleziona il valore "E" dal menu a discesa per l'azione Condizionale (Esegui?). Quindi seleziona "String Match" dal menu a discesa per la condizione Run (&&).

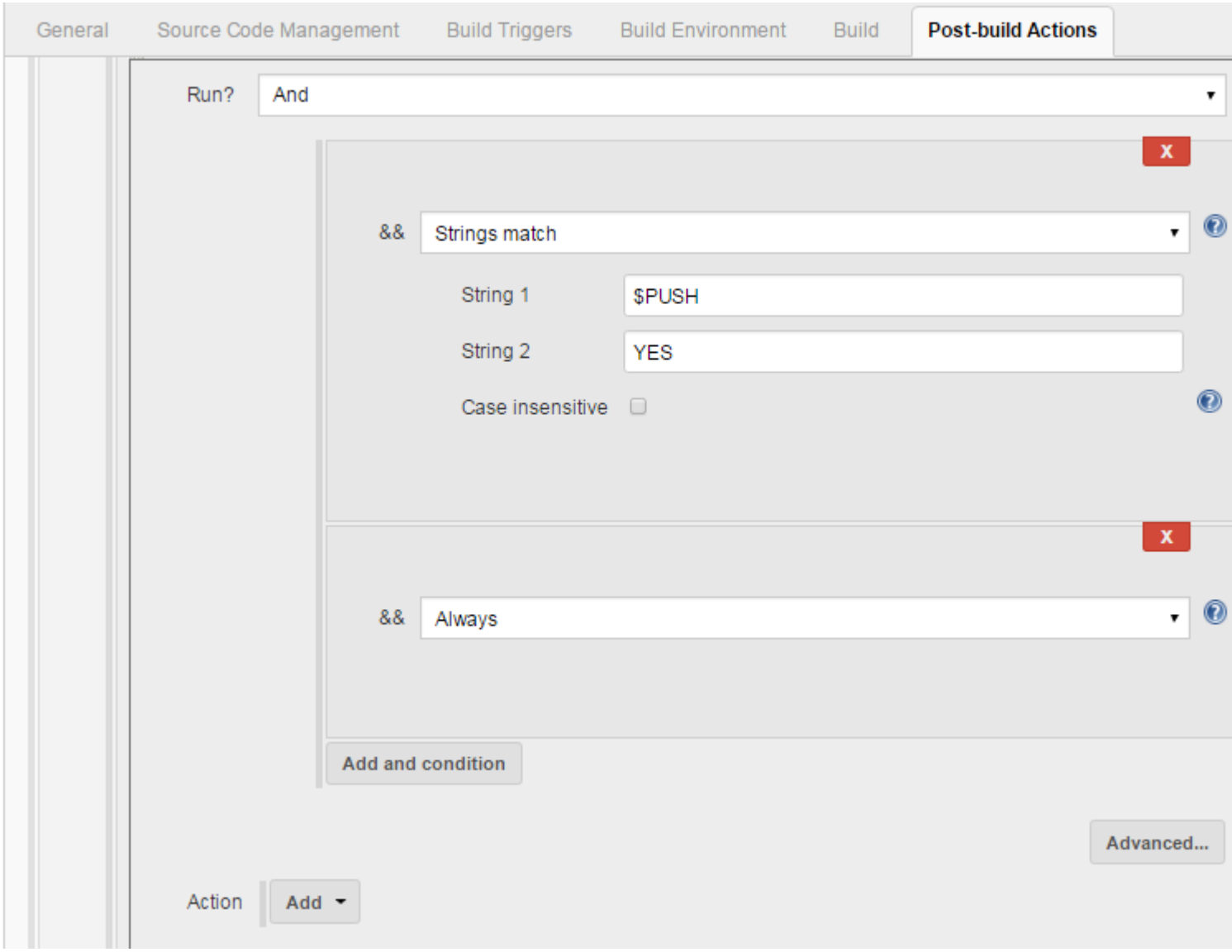

General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Flexible publish
Conditional action
Run?
And
Always
Never
Boolean condition
Build Cause
Current build status
Day of week
Execute Shell
Execute Windows Batch commands
Execution node
File exists
Files match
Legacy boolean condition (deprecated)
Numerical comparison
Regular expression match
Strings match
Time
And
Or
Not
Add and condition


General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Flexible publish
Conditional action
Run?
And
x
&&
Always
x
&&
Always
Add and condition
Advanced...

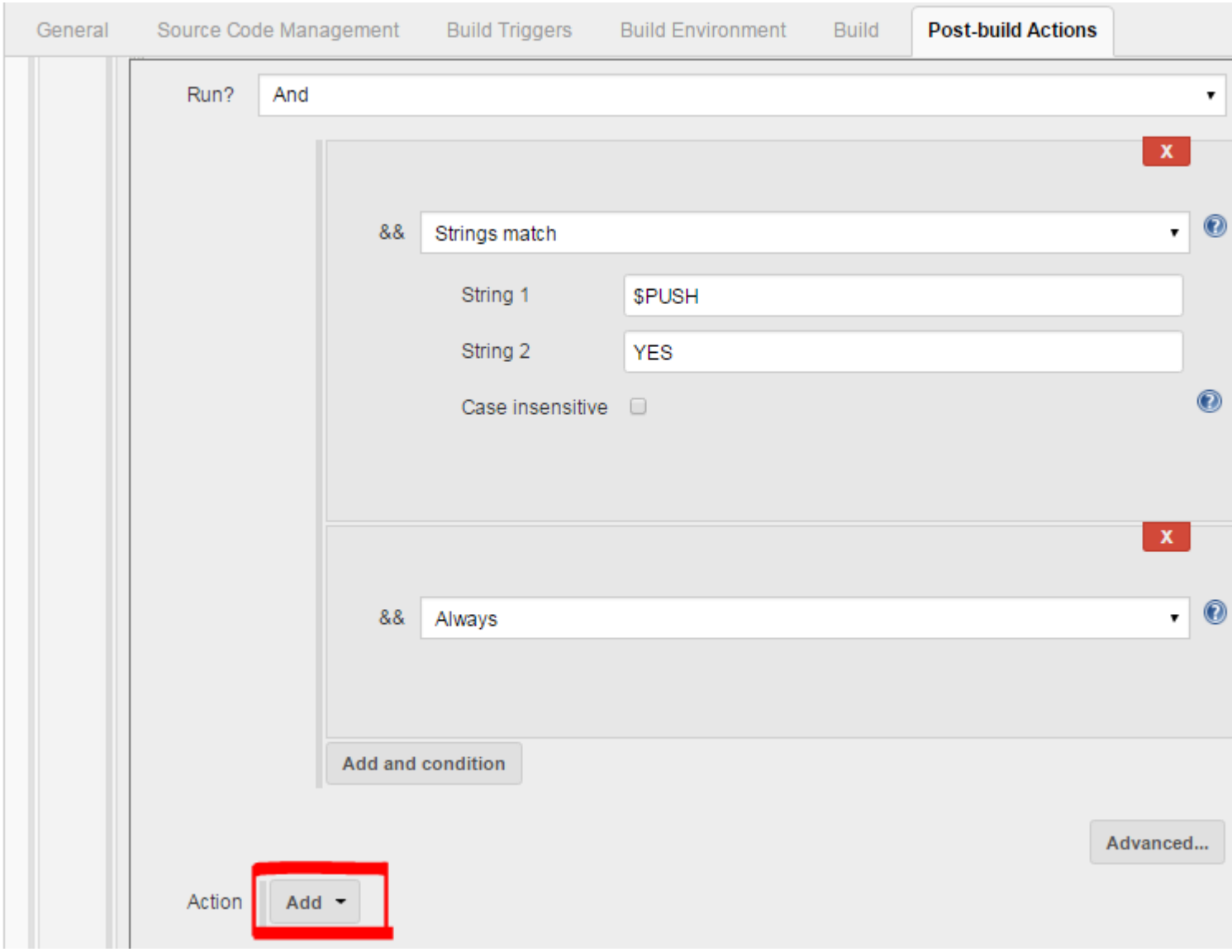


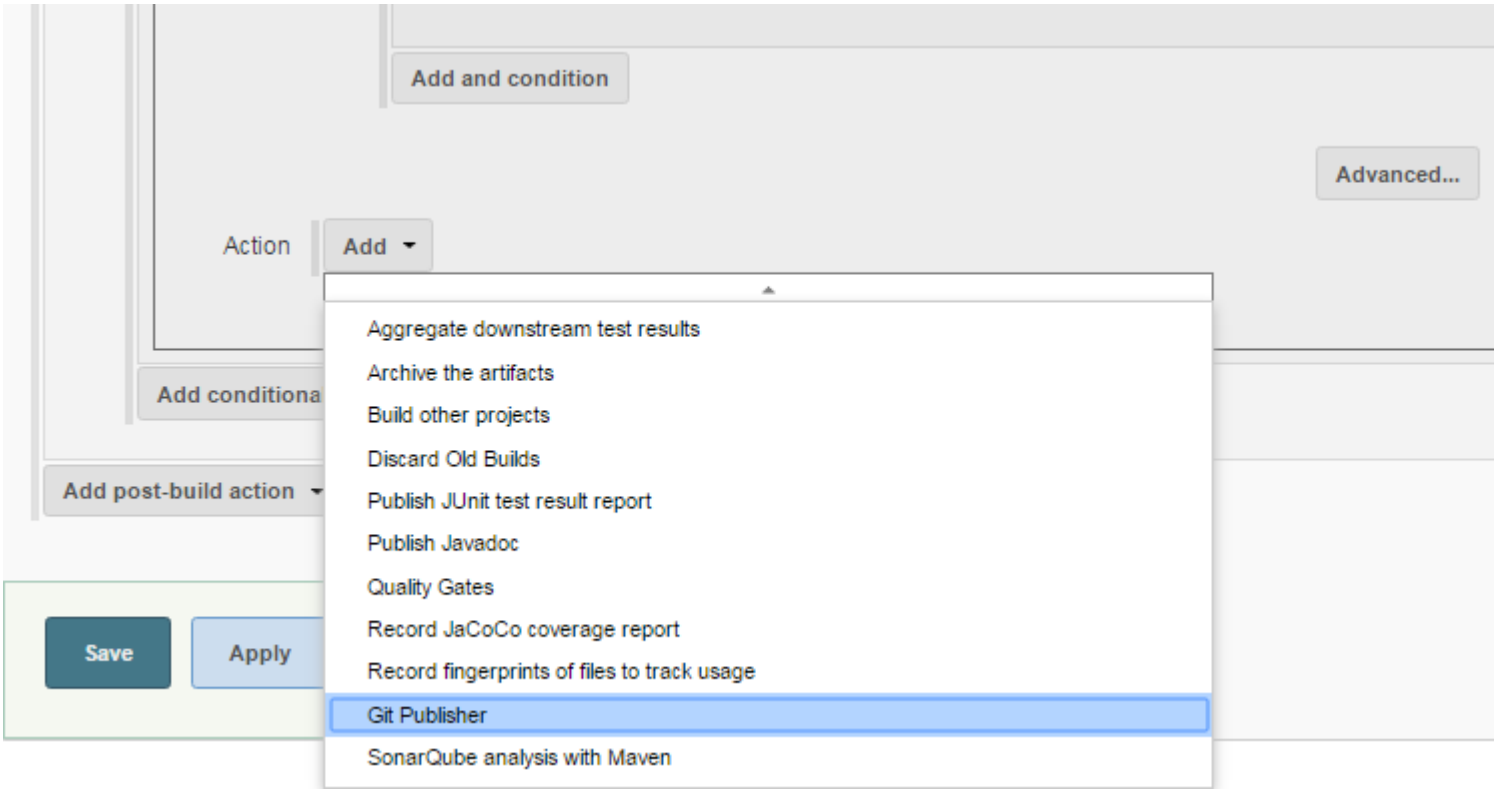
7. Dopo aver selezionato la corrispondenza della stringa, specificare $ PUSH come valore String 1 e YES come valore String 2. Quindi, quando eseguirai la build se scegli il valore di PUSH come SÌ, confronterà la stringa 1 (= $ PUSH) e la stringa 2 (= SÌ) e attiverà l'operazione Git push e se scegli NO non lo farà attiva l'operazione push Git.

```
Choose the value of PUSH -> YES OR NO -> Chosen value “YES”
then, $PUSH = YES
AS String 1 = $PUSH => String 1 = YES
Again, String 2 = YES, hence String 2 == String 1 (String match)
Then, trigger the Git push action.
```

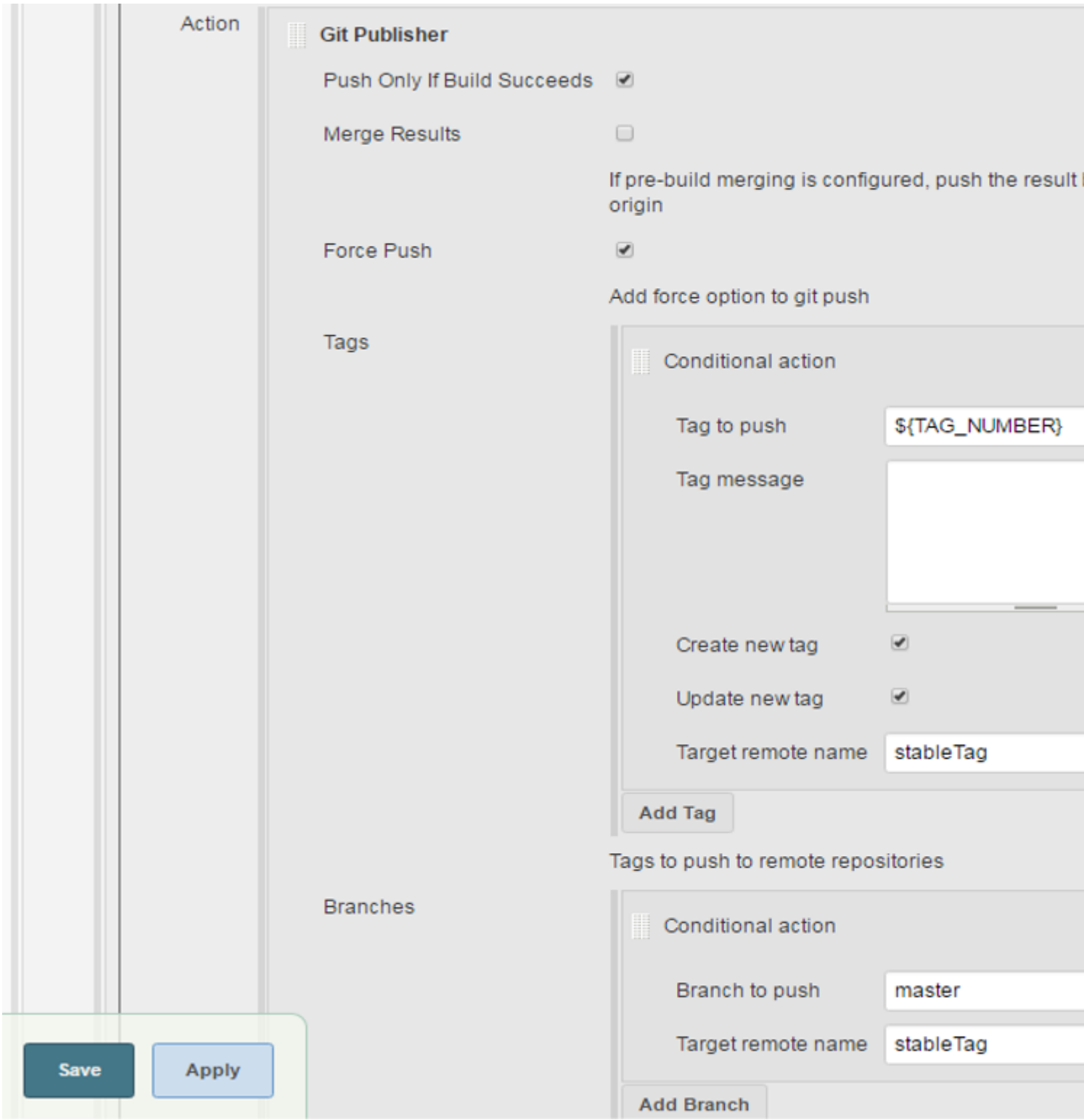


8. Ora fai clic su Aggiungi l'opzione a discesa per aggiungere l'azione di Git publisher che verrà attivata in base alla condizione di corrispondenza della stringa.


General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Run?
And
X
&&
Strings match
String 1
$PUSH
String 2
YES
Case insensitive
X
&&
Always
Add and condition
Advanced...
Action
Add


Add and condition
Advanced...
Action
Add
Aggregate downstream test results
Archive the artifacts
Build other projects
Discard Old Builds
Publish JUnit test result report
Publish Javadoc
Quality Gates
Record JaCoCo coverage report
Record fingerprints of files to track usage
Git Publisher
SonarQube analysis with Maven
Add conditiona
Add post-build action
Save
Apply

9. Dopo aver selezionato Git Publisher, eseguire la configurazione come segue:



Dopo la configurazione, salva il lavoro e hai finito.

Leggi Configura Auto Git Push su una build di successo in Jenkins online: https://riptutorial.com/it/jenkins/topic/8972/configura-auto-git-push-su-una-build-di-successo-in-jenkins

Capitolo 3: Configurazione di Build Automation per iOS utilizzando Shenzhen

Examples

Configurazione di iOS Build Automation con Shenzhen

Parte I: installazione del computer Mac per utilizzare shenzhen

Vai al terminale

Installa Shenzhen

> sudo gem install shenzhen
>
> sudo gem installa nomad-cli

Scarica l'utilità della riga di comando XCode

> xcode-select --install

Popup si presenta con il testo sottostante

> Il comando xcode-select richiede gli strumenti di sviluppo della riga di comando.
> Vorresti installare gli strumenti adesso? "

Fare clic su Installa

Crea la directory del progetto

gitclone il tuo progetto

> git clone https: //akshat@bitbucket.org/company/projectrepo.git

Costruisci il progetto usando il comando di sotto

> build ipa --verbose

PS: se vedi errori, seleziona il profilo di provisioning attivo e esegui il commit sui file di progetto. ed esegui `ipa build --verbose` nuovo.

Leggi Configurazione di Build Automation per iOS utilizzando Shenzhen online: https://riptutorial.com/it/jenkins/topic/8002/configurazione-di-build-automation-per-ios-utilizzando-shenzhen

Capitolo 4: Configurazione di Jenkins per l'automazione della build di iOS.

introduzione

Ora è possibile definire il processo di integrazione continua e consegna continua (**CI / CD**) come codice con Jenkins 2.0 per i propri progetti in iOS 10. Attività come la creazione, test, copertura del codice, controllo dello stile, report e notifiche possono essere descritte in una sola file.

Per leggere l'articolo completo vai su Pipeline in Jenkins 2.0 come Codice per iOS 10 e XCode 8

Parametri

Parametro	Dettagli
nodo ('Nodo iOS')	Nodo Jenkins con Mac OS. Se Jenkins è installato in Mac OS usa il `node {....}`

Osservazioni

L'articolo è scritto in entrambe le lingue: inglese e spagnolo.

Examples

Esempio di tabella di tempo

Il codice sorgente può essere clonato o scaricato da GitHub per testarlo.

```
node('iOS Node') {

    stage('Checkout/Build/Test') {

        // Checkout files.
        checkout([
            $class: 'GitSCM',
            branches: [[name: 'master']],
            doGenerateSubmoduleConfigurations: false,
            extensions: [], submoduleCfg: [],
            userRemoteConfigs: [[
                name: 'github',
                url: 'https://github.com/mmorejon/time-table.git'
            ]]
        ])

        // Build and Test
        sh 'xcodebuild -scheme "TimeTable" -configuration "Debug" build test -destination
```

```
"platform=iOS Simulator,name=iPhone 6,OS=10.1" -enableCodeCoverage YES |
/usr/local/bin/xcpretty -r junit'

        // Publish test restults.
        step([$class: 'JUnitResultArchiver', allowEmptyResults: true, testResults:
'build/reports/junit.xml'])
    }

    stage('Analytics') {

        parallel Coverage: {
            // Generate Code Coverage report
            sh '/usr/local/bin/slather coverage --jenkins --html --scheme TimeTable
TimeTable.xcodeproj/'

            // Publish coverage results
            publishHTML([allowMissing: false, alwaysLinkToLastBuild: false, keepAll: false,
reportDir: 'html', reportFiles: 'index.html', reportName: 'Coverage Report'])


        }, Checkstyle: {

            // Generate Checkstyle report
            sh '/usr/local/bin/swiftlint lint --reporter checkstyle > checkstyle.xml || true'

            // Publish checkstyle result
            step([$class: 'CheckStylePublisher', canComputeNew: false, defaultEncoding: '',
healthy: '', pattern: 'checkstyle.xml', unHealthy: ''])
        }, failFast: true|false
    }

    stage ('Notify') {
        // Send slack notification
        slackSend channel: '#my-team', message: 'Time Table - Successfully', teamDomain: 'my-
team', token: 'my-token'
    }
}
```

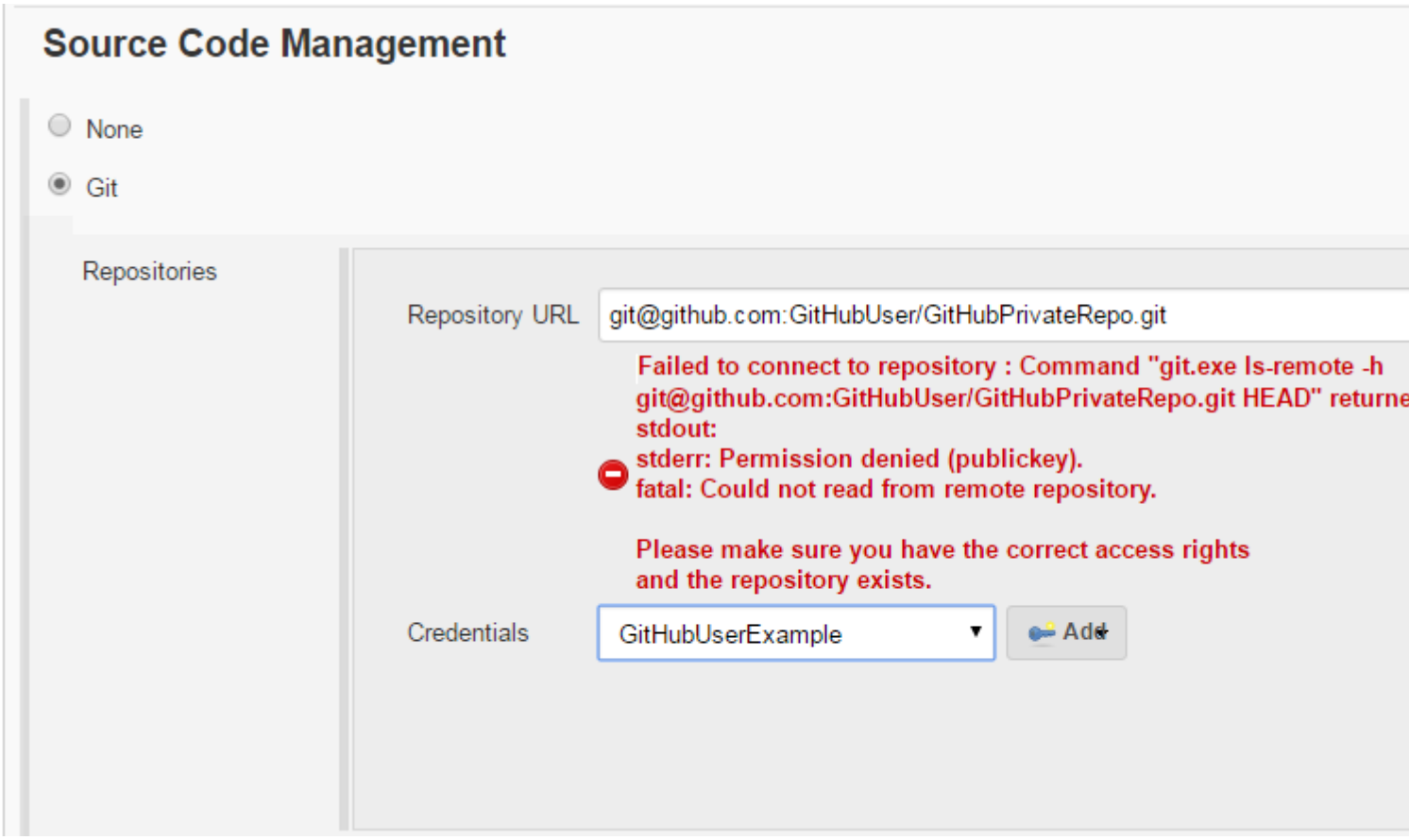
Leggi Configurazione di Jenkins per l'automazione della build di iOS. online: https://riptutorial.com/it/jenkins/topic/8868/configurazione-di-jenkins-per-l-automazione-della-build-di-ios-

Capitolo 5: Installa Jenkins su Windows con supporto SSH per repository GitHub privati

Examples

Le richieste pull GitHub falliscono

L'installazione immediata di Jenkins con i plugin Git e SSH non funzionerà quando si tenta di estrarre un repository privato da GitHub.



PS Tool PSExec.exe di Microsoft

Il primo passo per risolvere questo problema è stato scaricare PSTools ed estrarre gli strumenti in una posizione comoda sul server di build (ad es. C: \ Programmi \ PSTools è li ho estratto).

Genera una nuova chiave SSH solo per Jenkins usando PSExec o PSExec64

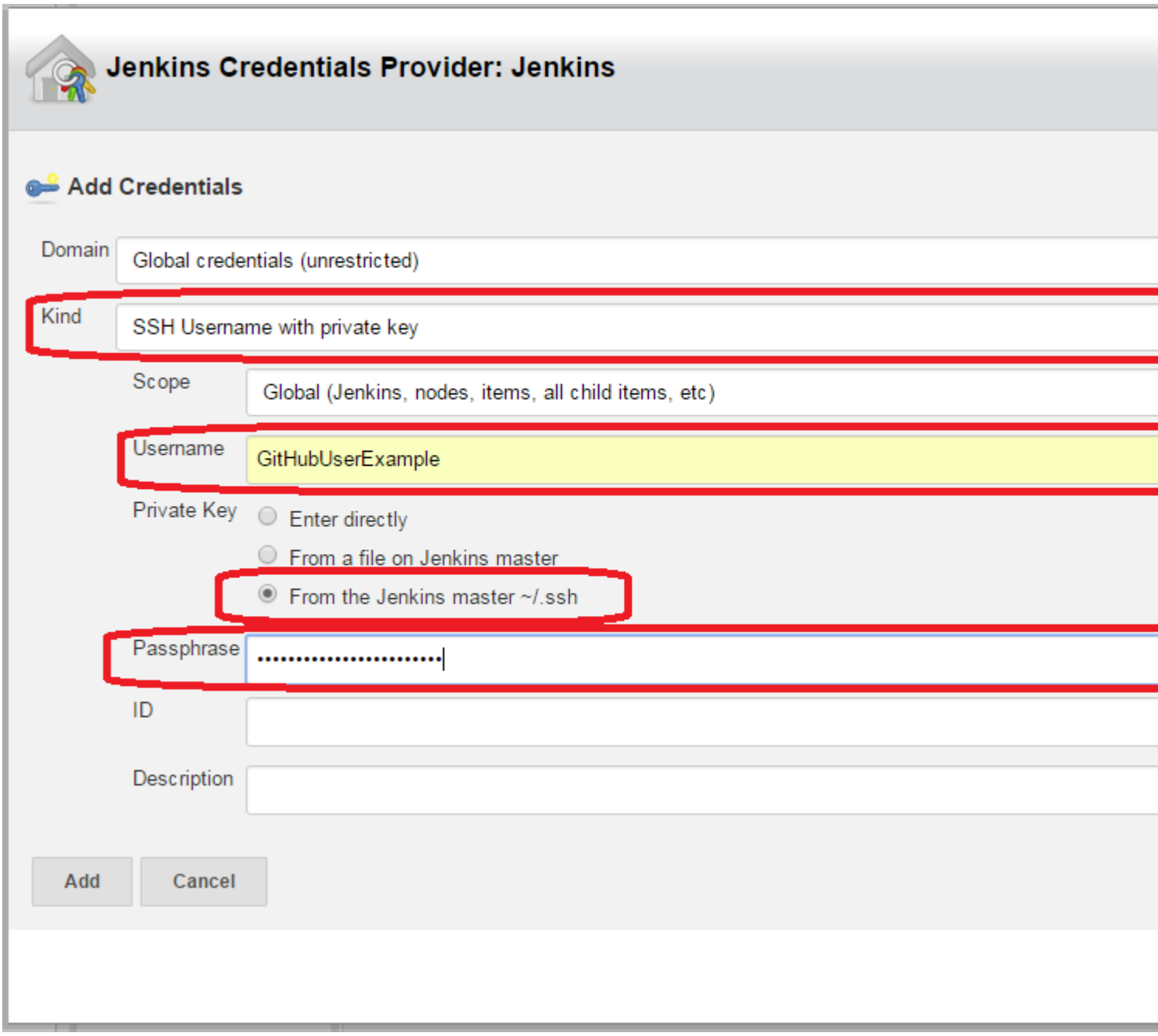
1. Per prima cosa aprire il prompt dei comandi e "Esegui come amministratore".
2. Una volta aperto il prompt dei comandi, accedere alla directory di PSTools.
3. Dal prompt dei comandi è necessario eseguire git-bash utilizzando PSExec o PSExec64 come servizio locale, che Jenkins è in esecuzione sul build server come impostazione predefinita.
4. Utilizzeremo l'opzione -i per eseguire PSExec come interattivo e l'opzione -s per eseguire

git-bash come servizio locale
5. Segui le istruzioni per la creazione di una chiave ssh su GitHub - Generazione di una nuova chiave SSH e aggiunta a ssh-agent
6. Se si utilizza un sistema Windows a 64 bit, copiare la cartella .ssh in C: \ Windows \ SysWOW64 \ config \ systemprofile.ssh (questo non era necessario sul mio sistema Windows a 64 bit, ma lì dove alcune istruzioni indicavano i file .ssh dovrebbe essere memorizzato lì, qualcosa da tenere a mente se hai ancora problemi).
7. Aggiungi la chiave pubblica SSH alle tue chiavi github.

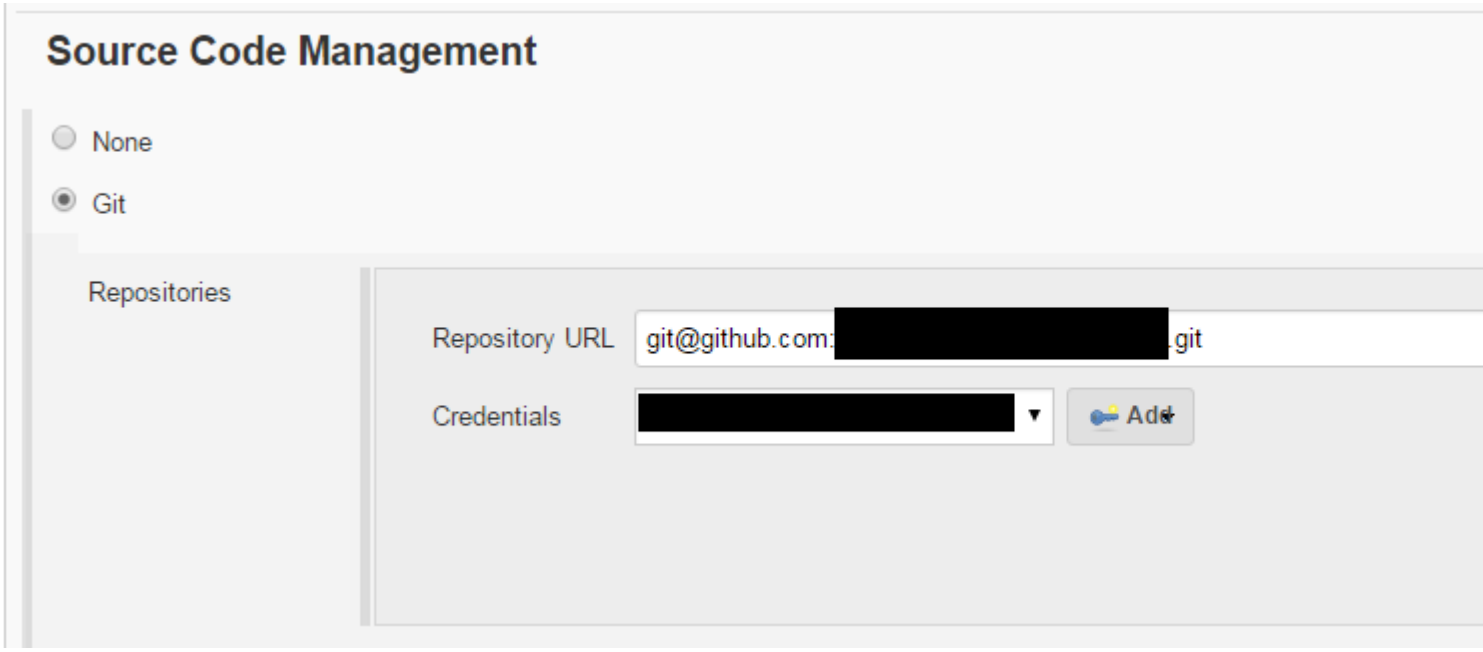
```
Your Commandline should look similar to this:

C:\Programs\PSTools> PSExec.exe -i -s C:\Programs\Git\git-bash
```

Crea le credenziali di Jenkins

La parte difficile è finita! Ora basta creare le credenziali da utilizzare in Jenkins. Usa il tuo nome utente e la passphrase usati per creare la chiave SSH.

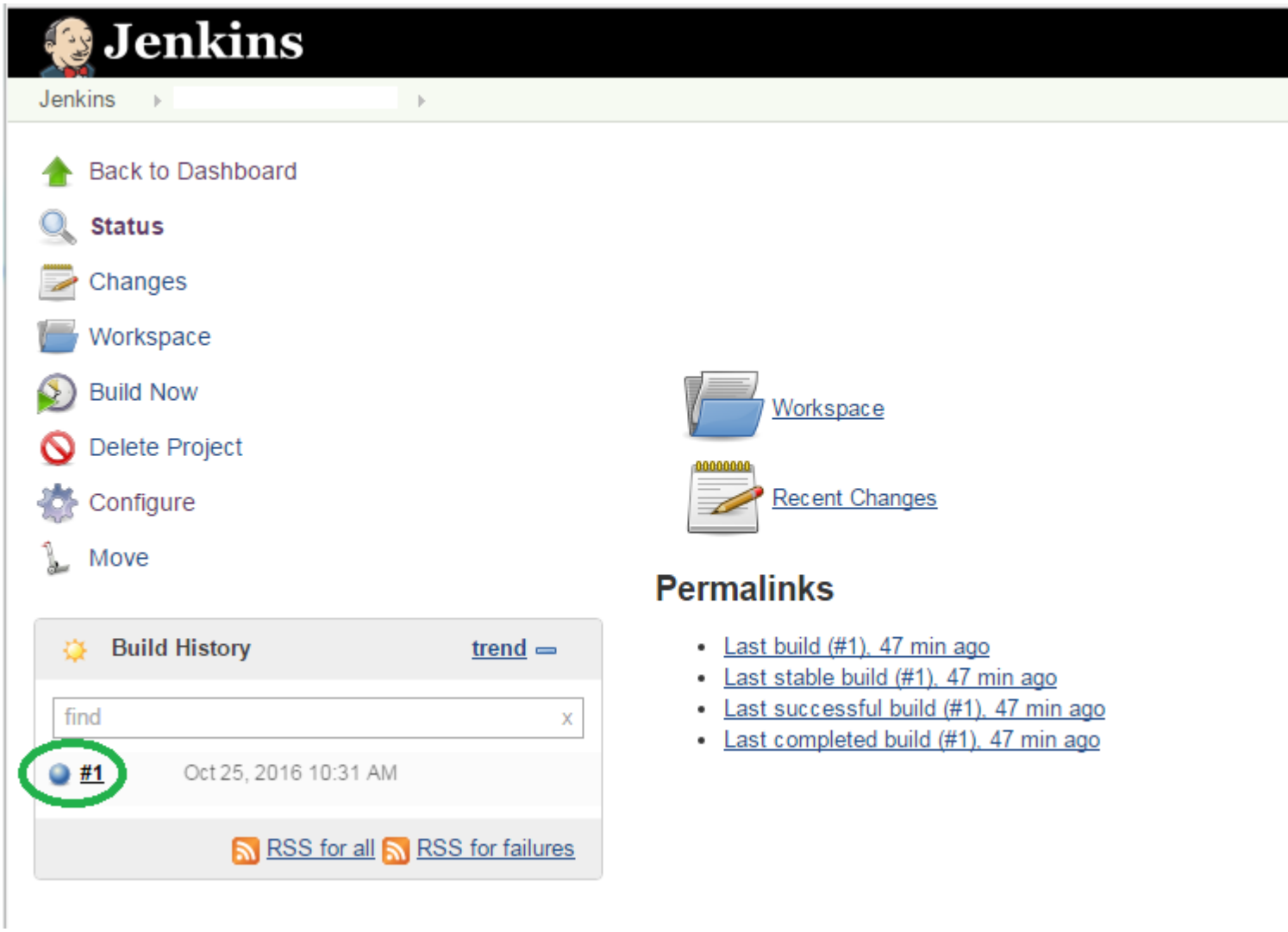


Ecco come dovrebbe apparire ora (con il tuo repository Github privato e nome utente:



Esegui una richiesta di prova di prova per verificare, e il tuo fatto.

Salva ed esegui una richiesta di estrazione di prova e non dovresti più avere problemi con l'uso di SSH sul tuo computer Windows.



Leggi Installa Jenkins su Windows con supporto SSH per repository GitHub privati online: https://riptutorial.com/it/jenkins/topic/7626/installa-jenkins-su-windows-con-supporto-ssh-per-repository-github-privati

Capitolo 6: Jenkins Groovy Scripting

Examples

Crea utente predefinito

1. Crea file groovy per percorso `$JENKINS_HOME/init.groovy.d/basic-security.groovy`

In Ubuntu 16 la home directory di Jenkins si trova in `/var/lib/jenkins`

2. Inserisci nel file il prossimo codice

```
#!groovy

import jenkins.model.*
import hudson.security.*

def instance = Jenkins.getInstance()

def hudsonRealm = new HudsonPrivateSecurityRealm(false)

hudsonRealm.createAccount("admin_name","admin_password")
instance.setSecurityRealm(hudsonRealm)
instance.save()
```

3. Riavvia il servizio Jenkins
4. Dopo l'avvio di Jenkins è necessario rimuovere `$JENKINS_HOME/init.groovy.d/basic-security.groovy` file

Disabilita installazione guidata

1. Aprire il file di configurazione di default Jenkins e aggiungere in `JAVA_ARGS` prossimo chiave `-Djenkins.install.runSetupWizard=false`

In Ubuntu 16 il file predefinito si trova in `/etc/default/jenkins`

2. Crea file groovy per percorso `$JENKINS_HOME/init.groovy.d/basic-security.groovy`

In Ubuntu 16 la home directory di Jenkins si trova in `/var/lib/jenkins`

3. Inserisci nel file il prossimo codice

```
#!groovy

import jenkins.model.*
import hudson.util.*;
import jenkins.install.*;

def instance = Jenkins.getInstance()

instance.setInstallState(InstallState.INITIAL_SETUP_COMPLETED)
```

4. Riavvia il servizio Jenkins
5. Dopo l'avvio di Jenkins è necessario rimuovere `$JENKINS_HOME/init.groovy.d/basic-security.groovy` file

Dopo questo, Jenkins non ti chiede di confermare che sei amministratore e non vedrai la pagina di installazione dei plugin.

Come ottenere informazioni sull'istanza di Jenkins

Apri la tua console di script di istanze jenkins http: // yourJenkins: port / script following è un esempio di come ottenere informazioni su questa istanza. copia il codice nella console e fai clic su "Esegui".

```
/* This scripts shows how to get basic information about Jenkins instance */
def jenkins = Jenkins.getInstance()
println "Jenkins version: ${jenkins.getVersion()}"
println "Available JDKs: ${jenkins.getInstance().getJDKs()}"
println "Connected Nodes:"
jenkins.getNodes().each{
  println it.displayName
}
println "Configured labels: ${jenkins.getLabels()}"
```

In questo esempio verranno visualizzate informazioni sulla versione, JDK, agenti (slave) ed etichette di Jenkins.

Come ottenere informazioni su un lavoro di Jenkins

Apri la tua console di script di istanze di jenkins http: // yourJenkins: port / script following è un esempio di come ottenere informazioni su un lavoro specifico. copia il codice nella console, modifica il nome del lavoro nel lavoro richiesto e fai clic su "Esegui".

```
/*This script shows how to get basic information about a job and its builds*/
def jenkins = Jenkins.getInstance()
def jobName = "myJob"
def job = jenkins.getItem(jobName)

println "Job type: ${job.getClass()}"
println "Is building: ${job.isBuilding()}"
println "Is in queue: ${job.isInQueue()}"
println "Last successfull build: ${job.getLastSuccessfulBuild()}"
println "Last failed build: ${job.getLastFailedBuild()}"
println "Last build: ${job.getLastBuild()}"
println "All builds: ${job.getBuilds().collect{ it.getNumber()}}"
```

prima otteniamo l'oggetto istanza di Jenkins, quindi usando questa istanza otteniamo l'oggetto lavoro (elemento). dall'oggetto di lavoro possiamo ottenere informazioni diverse come: è attualmente in fase di costruzione, è in coda, l'ultima build, l'ultima build per stato e molto altro.

Leggi Jenkins Groovy Scripting online: https://riptutorial.com/it/jenkins/topic/7562/jenkins-groovy-

scripting

Capitolo 7: Role Strategy Plugin

Examples

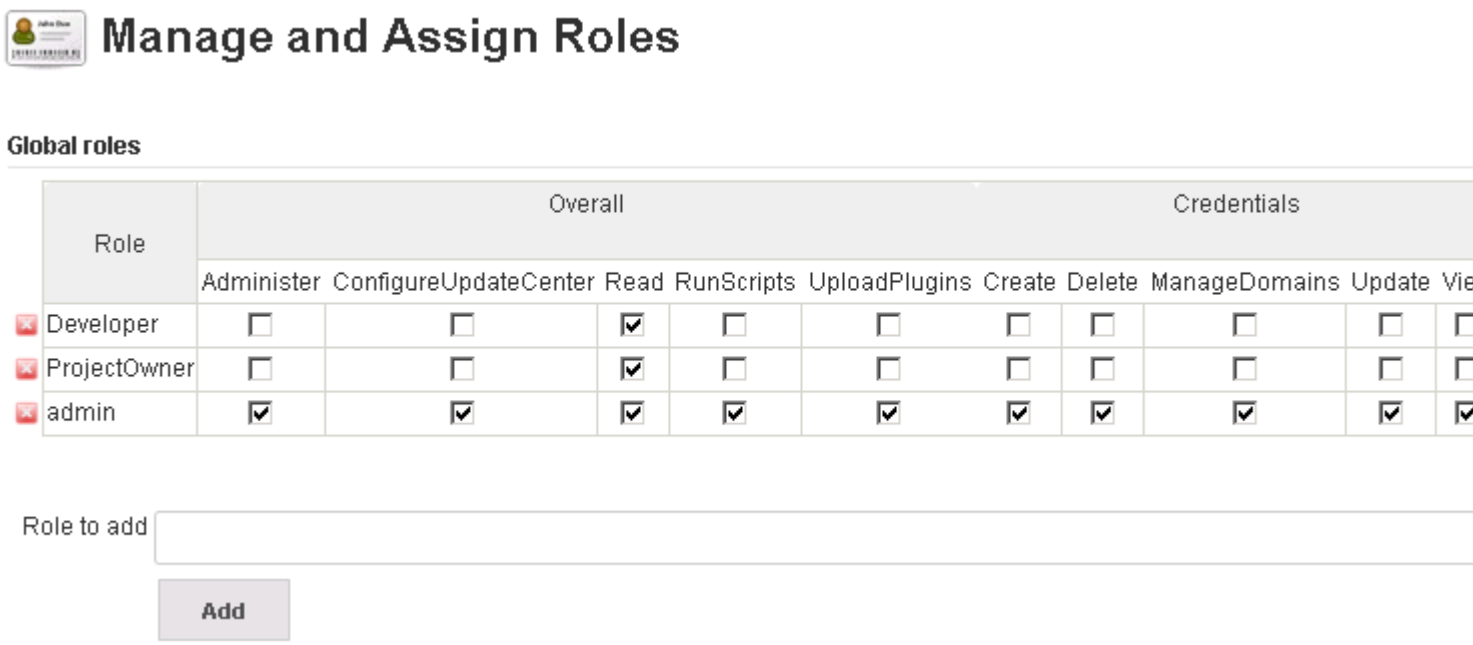
Configurazione

Gestisci ruoli

Ruoli globali : crea ruoli con il set selezionato di funzionalità di Jenkins, ad esempio, di solito, per un progetto di sviluppo, è possibile creare 2 ruoli.

1. Sviluppatore: il ruolo globale può essere impostato solo su **Generale** : Leggi
2. ProjectOwner: il ruolo globale può essere impostato su **Overall** : Read

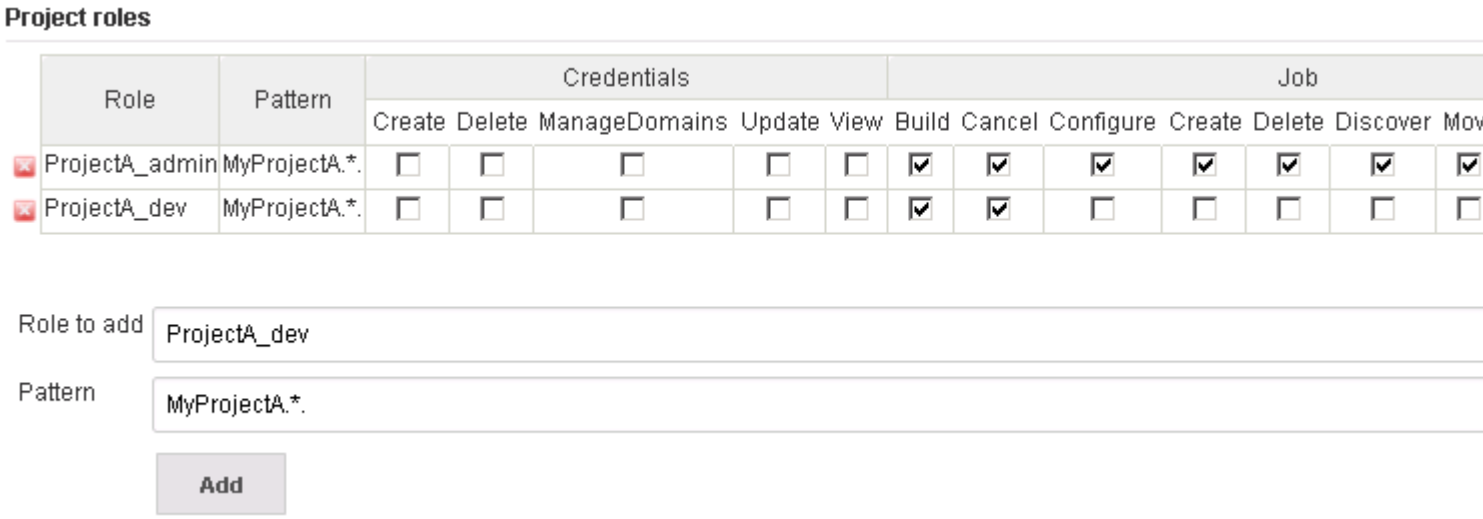
Ciò limita lo sviluppatore e il proprietario del progetto a leggere l'accesso a tutte le funzionalità di Jenkins.



Ruoli del progetto : consente di creare ruoli limitando l'accesso degli utenti a funzioni di lavoro e credenziali jenkins utilizzando espressioni regolari.

Ad esempio per un progetto di sviluppo "MyProjectA"; i proprietari del progetto devono disporre di autorizzazioni complete per i lavori e gli sviluppatori devono creare l'accesso ai lavori di Jenkins. Quindi creiamo i seguenti ruoli:

- **ProjectA_admin** - controlla tutte le opzioni in Job viz. *Crea, Annulla, Configura, Crea, Elimina, Scopri, Sposta, Leggi, Area di lavoro*
- **ProjectA_dev** - controlla le opzioni Build, Cancel, Read, Workspace in Job



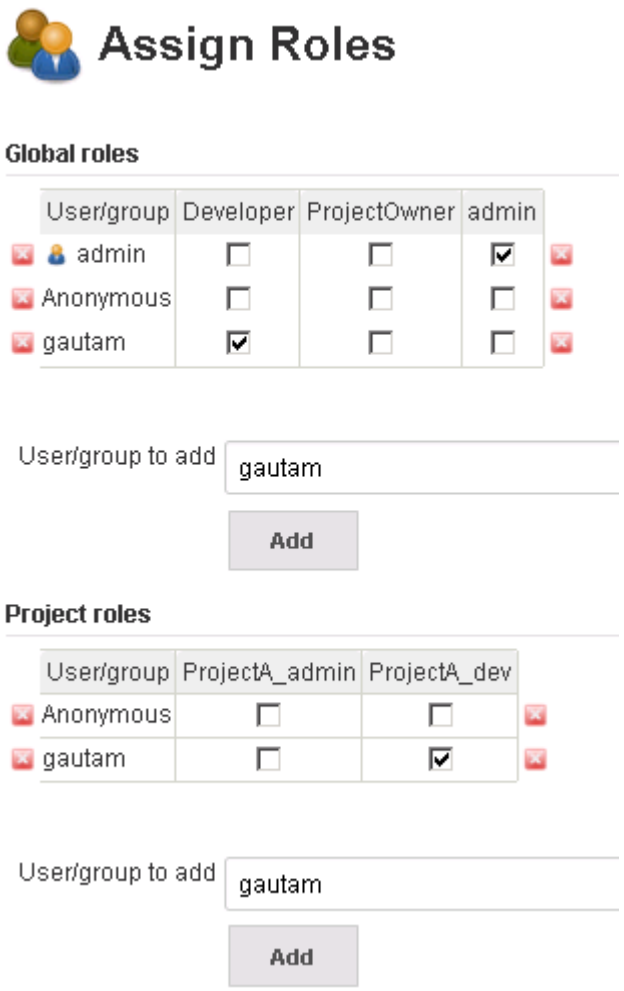
Per limitare i progetti sopra ai rispettivi proprietari e sviluppatori di progetti, tutti i lavori devono seguire uno schema predefinito.

Supponiamo che 'MyProjectA' abbia bisogno di 3 jenkins build job: *MyProjectA_Dev_Build, MyProjectA_QA_Build, MyProjectA_Nightly_Sonar_Analysis*

Per limitare il proprietario del progetto e gli sviluppatori del progetto 'MyProjectA' ai lavori di compilazione precedenti, fornire ' *Pattern* ' come **MyProjectA.** *.

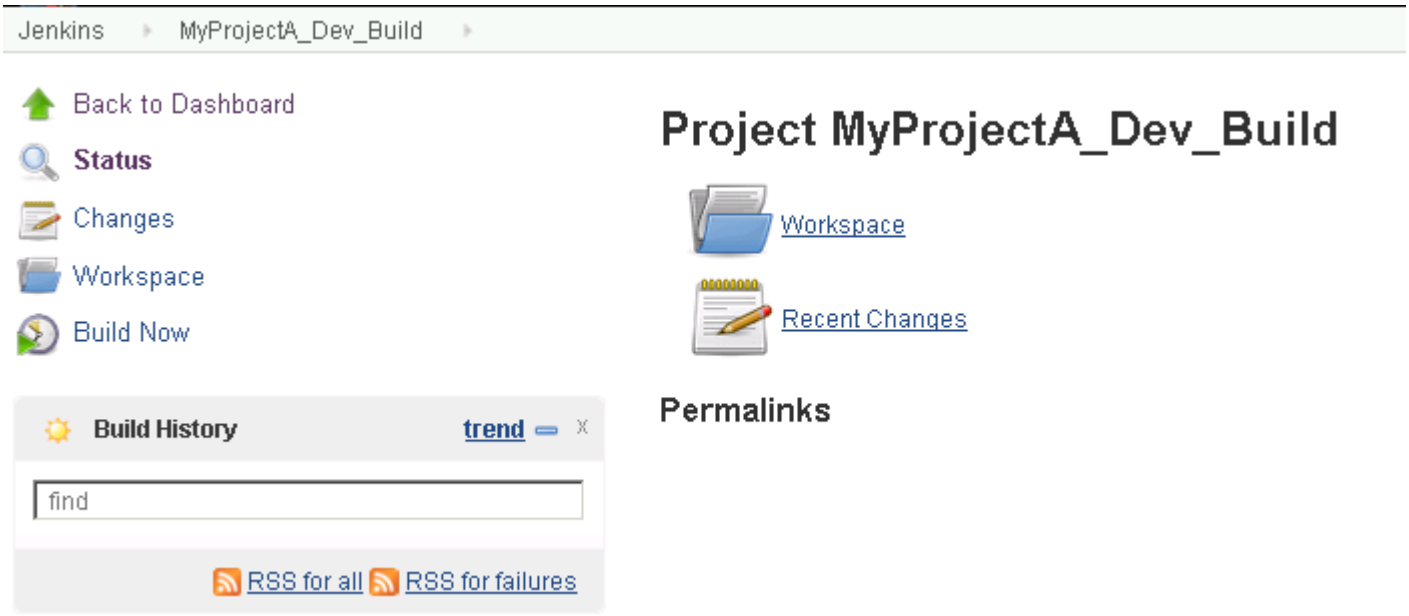
Assegna ruoli

Aiuta ad assegnare utenti o gruppi di progetti ai rispettivi ruoli globali o di progetto. Ad esempio, per assegnare uno sviluppatore 'Gautam' al ruolo globale dello sviluppatore, fornire il nome utente 'Gautam', fare clic su *Aggiungi* e selezionare la casella di controllo accanto a 'Gautam' e sotto il ruolo globale dello sviluppatore.



Allo stesso modo aggiungere l'utente in ruoli di progetto e selezionare i rispettivi ruoli di progetto per assegnare i ruoli di progetto richiesti.

Se noti che sotto screenshot puoi vedere l'utente 'gautam' ha accesso solo ai progetti che iniziano con MyProjectA. Inoltre, l'accesso dell'utente è limitato alla compilazione e manca la configurazione.



Leggi Role Strategy Plugin online: https://riptutorial.com/it/jenkins/topic/5741/role-strategy-plugin

Titoli di coda

S. No	Capitoli	Contributors
1	Inizia con Jenkins	acalb, Community, Gautam Jose, Girish Kumar, Katu, Pablo, Pom12, Priyanshu Shekhar, Rogério Peixoto, S.K. Venkat, Seb, Tyler
2	Configura Auto Git Push su una build di successo in Jenkins	ANIL
3	Configurazione di Build Automation per iOS utilizzando Shenzhen	Ichthyocentaurs
4	Configurazione di Jenkins per l'automazione della build di iOS.	Manuel Morejón
5	Installa Jenkins su Windows con supporto SSH per repository GitHub privati	Riana
6	Jenkins Groovy Scripting	RejeeshChandran, serieznyi, Tidhar Klein Orbach
7	Role Strategy Plugin	Gautam Jose