

Deploy di Jenkins 2.91 per Mondadori

Attenzione Se ti trovi dietro a un proxy e installi Jenkins su una VM in localhost, nessun servizio Git* sarà in grado di connettersi ad esso.

Se non ti trovi dietro un proxy ti basta sostituire "localhost" con il tuo **IP pubblico** nei vari webhook.

Installazione VM

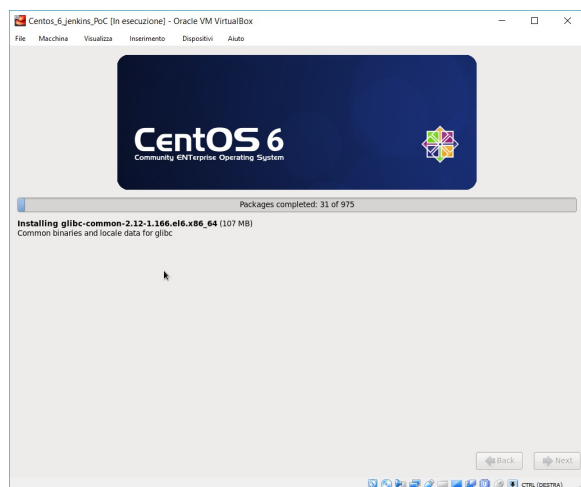
- Download di Centos v6.7 da http://archive.kernel.org/centos-vault/6.7/isos/x86_64/
-

Creazione macchina virtuale su GCP o simile:

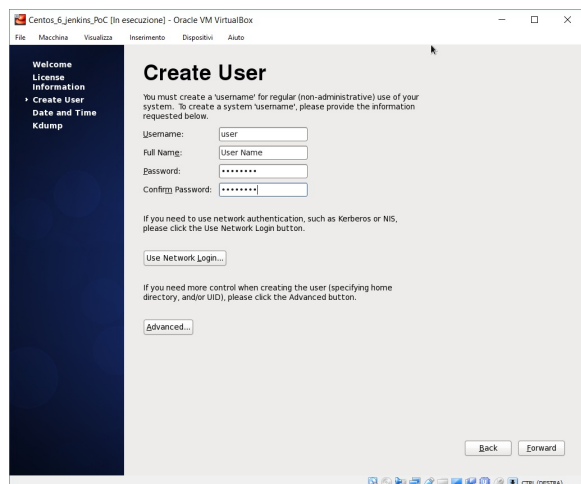
- Centos v6.7
 - 8 GB RAM
 - 20 GB Hdd
-

Installazione di Centos 6.7

- Selezionare lingua inglese
- Selezionare lingua italiana per la tastiera
- Impostare l'orario su "Europe,Rome"
- Root Password : password



- Create a user : **user** con password : **password**



- Reboot
- Imposta il proxy (se sei in sede reply)

```
su
Password: *****

vi /etc/yum.conf

#aggiungi in fondo
proxy=http://proxy.reply.it:8080

vi /etc/profile

#aggiungi in fondo
export http_proxy=http://proxy.reply.it:8080
export https_proxy=http://proxy.reply.it:8080

source /etc/profile
```

- Imposta la connessione automatica al boot

```
vi /etc/sysconfig/network-scripts/ifcfg-eth0

#modifica
ONBOOT=yes
```

- Aggiorna il sistema

```
sudo yum update -y
```

- Se vuoi, installa le Virtual Box guest additions (io lo trovo molto comodo, michele le odia)

```
sudo yum update
sudo yum install gcc
sudo yum install kernel-headers
sudo yum install kernel-devel
```

Installazione di jenkins

Per l'installazione di jenkins seguirò un misto tra questa [guida per Centos 7](#) e questa [guida specifica per Centos 6](#).

Ottenere un utente NON root con privilegi di root

Il nostro utente *user* va più che bene a questo scopo, per garantigli privilegi di root è necessario editare il file `/etc/sudoers` ma, mi raccomando, **NON FARLO MAI CON VIM**.

Usa invece:

```
su
Password: *****

visudo
```

Giusto per capire cosa andremo a fare: **root ALL=(ALL:ALL) ALL**

- *root* indica l'utente
- Il primo *ALL* indica che la regola si applica ad ogni host
- Il secondo *ALL* indica che l'utente root esegue comandi per ogni altro utente (tramite sudo)
- Il terzo *ALL* indica che può eseguire comandi per ogni gruppo
- L'ultimo *ALL* indica che la regola si applica a qualsiasi comando

Decommenta la riga

```
%wheel ALL=(ALL) ALL
```

In modo da garantire tutti i diritti agli utenti del gruppo wheel. Quindi aggiungi *user* al gruppo wheel

```
sudo usermod -aG wheel username  
  
reboot #per rendere valida la modifica
```

Puoi trovare maggiori informazioni su come gestire i diritti sudo a [questa pagine](#).

Installazione di Jenkins da pacchetto RPM

Tutti i comandi dovranno essere fatti con l'utente **NON ROOT** tramite sudo.

Installare java 1.8

Per prima cosa installiamo java, questa versione di jenkins richiede java 1.8

Connettiti alla pagina di [oracle](#), accetta la licenza e scarica **jdk-8u151-linux-x64.tar.gz**.

```
cd /opt  
wget http://download.oracle.com/otn-pub/java/jdk/8u151-b12/e758a0de34e24606bca991d704f6dcbf/jdk-8u151-linux-x64.rpm?AuthParam=1511780178_e971f91c277e5926c5a9b60179d896de  
  
sudo rpm -ivh jdk-8u151-linux-x64.rpm?AuthParam=1511780178_e971f91c277e5926c5a9b60179d896de  
  
java -version  
java version "1.8.0_151"  
Java(TM) SE Runtime Environment (build 1.8.0_151-b12)  
Java HotSpot(TM) 64-Bit Server VM (build 25.151-b12, mixed mode)  
  
# per salvare spazio  
sudo rm -rf jdk-8u151-linux-x64.rpm?AuthParam=1511780178_e971f91c277e5926c5a9b60179d896de
```

Download del repo RPM

Scarica il pacchetto RPM di jenkins, puoi spostarti in [/etc/yum.repos.d](#) oppure eseguire direttamente

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

Importa il codice di verifica di validità del pacchetto

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

Se, per una qualche ragione, rpm non riesce a scaricare la chiave restituendo un errore 404 (come è successo a me):

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

Installazione con yum

Una volta fatto possiamo installare jenkins con yum.

```
sudo yum install jenkins
```

Terminata l'installazione, avviamo il servizio di jenkins e impostiamo l'avvio automatico

```
sudo service jenkins start
sudo chkconfig jenkins on
```

File utili

E' stato creato un utente *jenkins* pensato per eseguire tutto quello che è legato al servizio, puoi modificare questo utente nel file di config , ma ricorda di cambiare i privilegi a `/var/log/jenkins`, `/var/lib/jenkins` e `/var/cache/jenkins`.

Troverai i file di log in

Cosa	File
Config	<code>/etc/sysconfig/jenkins</code>
Log	<code>/var/log/jenkins/jenkins.log</code>

Connessione al servizio e firewall

Usa `service jenkins start/stop/status` per gestire il servizio.

All'avvio del servizio, potrai interagire con jenkins tramite il tuo browser alla porta **8080**, il firewall potrebbe darti dei disagi, per disabilitarlo, in Centos 6.7 non possiamo usare `firewalld`, utilizzeremo la TUI di `system-config-firewall`

```
sudo iptables -L #per vedere le attuali regole
```

Per modificare permanentemente il firewall, in modo da aprire la porta **tcp:8080**

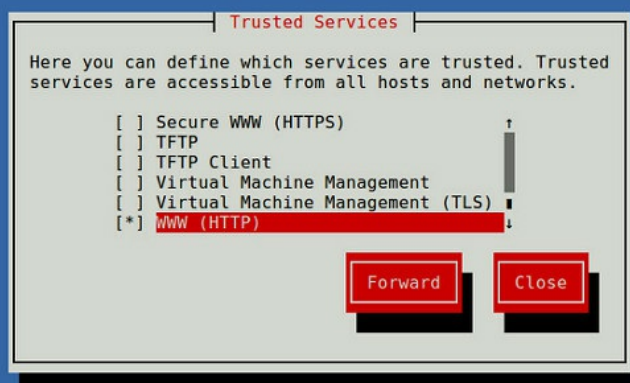
```
sudo system-config-firewall-tui
```

system-config-firewall



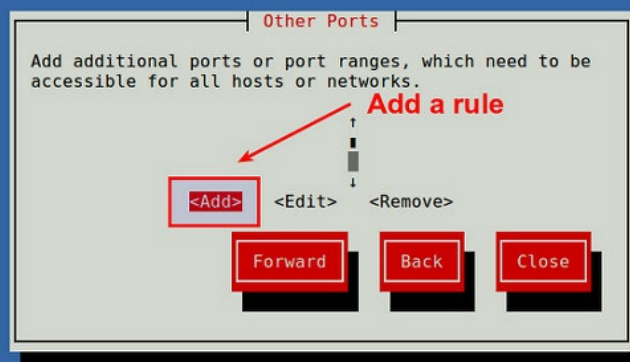
<Tab>/<Alt-Tab> between elements | <Space> selects | <F12> next screen

system-config-firewall



<Tab>/<Alt-Tab> between elements | <Space> selects | <F12> next screen

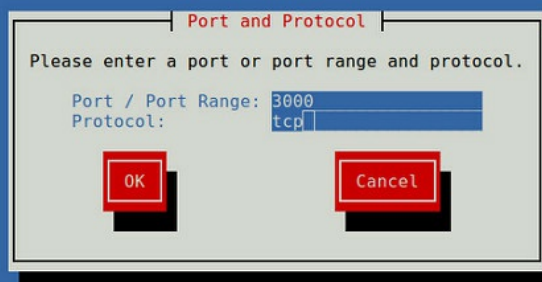
system-config-firewall



<Tab>/<Alt-Tab> between elements | <Space> selects | <F12> next screen

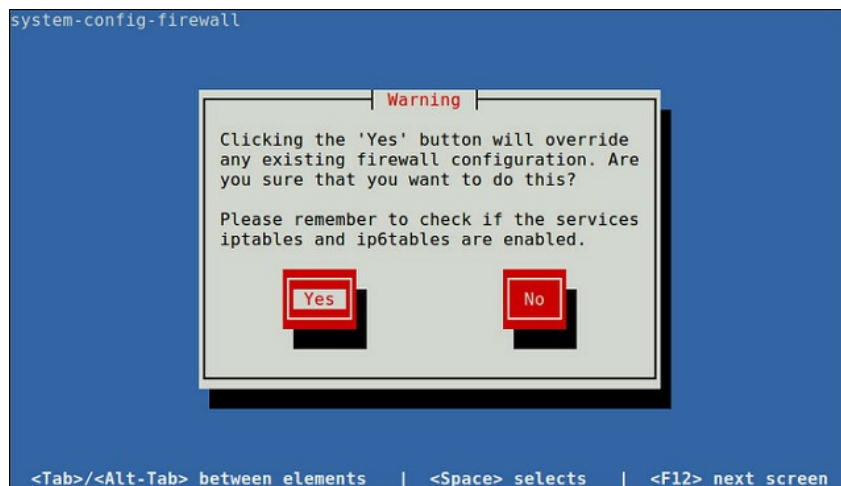
Ovviamente inserisci la porta 8080

system-config-firewall



<Tab>/<Alt-Tab> between elements | <Space> selects | <F12> next screen

OK -> Back -> Close -> Ok

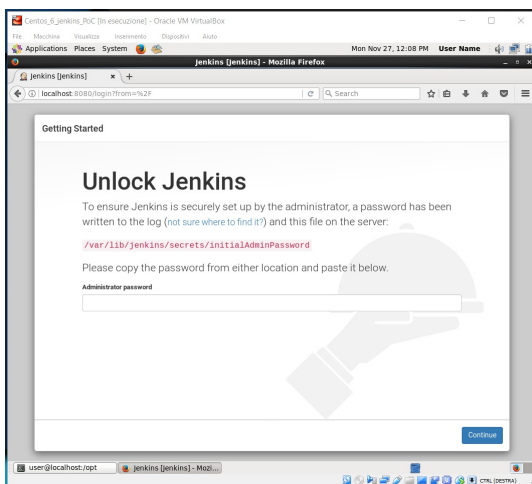


Avvio e utilizzo di jenkins

```
sudo service jenkins start
Starting Jenkins
```

[OK]

E collegati alla pagina <http://35.196.238.203:8080>, dovrebbe aprirsi una pagina del genere.



Scopriamo la password autogenerata

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
ab21de47d4cf4d70856f57d953fb84bf
```

Installazione plugin

Quindi installiamo i plugin che ci servono (io devo fare una prova, ho installato quelli di default).



Attenzione se sei dietro a un proxy questa procedura fallirà malamente, questo perchè jenkins ha bisogno di una configurazione proxy ma nessuno ci ha dato la possibilità di configurarlo.

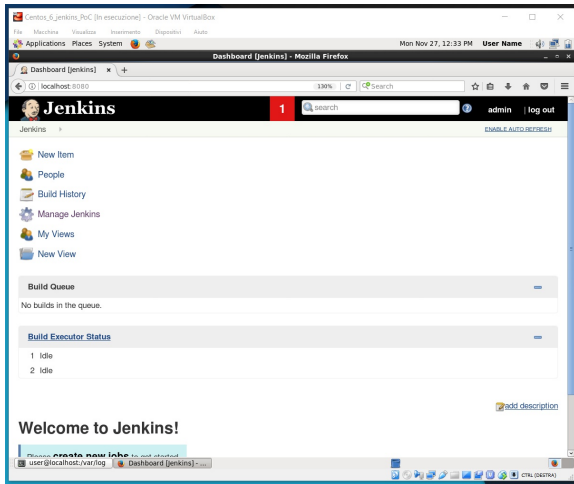
Non preoccuparti per ora, fai continua e ci pensiamo dopo.

Creazione utente admin

Per comodità ho creato un utente admin default:

- Username = admin
- Password = admin

Se tutto va bene dovrebbe aprirsi l'homepage di jenkins.



Configurare Jenkins

Proxy

Manage Jenkins -> Manage Plugins -> "Advanced" tab. Qui puoi impostare il proxy reply.

Enable Global Security

Manage Jenkins -> Configure Global Security

Se la spunta non è presente, spuntare *Enable Security*

//TODO Spuntare Matrix-based security e configura un utente admin (non c'è il checkbox) (<https://wiki.jenkins.io/display/JENKINS/Matrix-based+security>)

Installare Plugin

Manage Jenkins -> Manage Plugins -> "Available" tab

Seleziona i plugin che desideri, per la mia Demo ho selezionato solo **GIT plugin** e **GitHub Plugin** dal momento che il progetto da gestire si trova su github. In ogni caso jenkins ha installato anche tutti i plugin che erano falliti nell'installazione iniziale.

Esegui l'installazione con restart e, al termine, torna alla Top Page.

Configurare GIT

Se Git non è installato nell'host questo non funzionerà nemmeno su Jenkins (grazie al cazzo).

```
sudo yum install git -y

# Per sapere dove è stato installato
sudo which git
/usr/bin/git
```

Inoltre dovremo configurarlo in Jenkins, andiamo quindi in Manage Jenkins -> Configure System -> sezione Git Plugin e inseriamo nome utente e email dell'account usato nel repository.

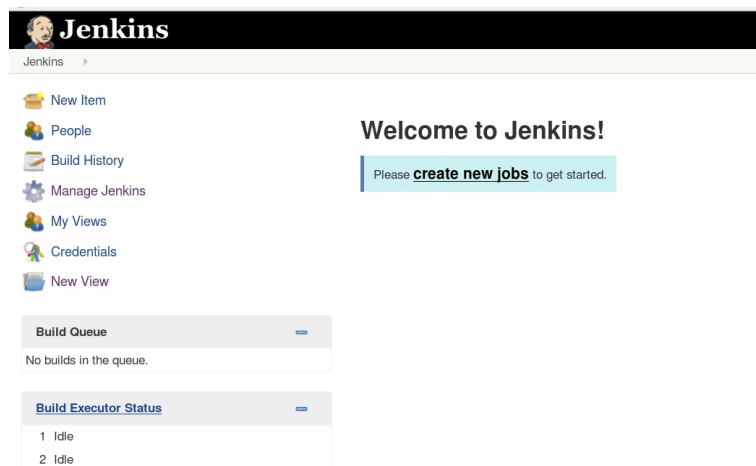
Git plugin

Global Config user.name Value	<input type="text" value="dpasquali"/>	
Global Config user.email Value	<input type="text" value="d.pasquali@reply.it"/>	
Create new accounts based on author/committer's email	<input type="checkbox"/>	

Non possiamo però ancora accedere al repository, la configurazione sarà completata all'inserimento del link (prossima sezione).

Demo

Nella homepage creiamo un *New Job* dal pulsante centrale se non ne abbiamo o da *New Item* a sinistra.



Metti il nome che preferisci e seleziona *Freestyle Project*

Enter an item name

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Imposta la descrizione, spunta *GitHub project* e inserisci il link del repository (intendo l'**URL nella barra di navigazione**)

The screenshot shows the 'General' tab of the Jenkins configuration interface. The 'Pipeline name' field is set to 'Demo'. The 'Description' field contains 'Jenkins demo with Github'. Below the description, there are checkboxes for 'Discard old builds', 'Do not allow concurrent builds', and 'GitHub project' (which is checked). The 'Project url' field is set to 'https://github.com/dpasqualiReply/JenkinsDemo'. An 'Advanced...' button is located at the bottom right.

Qui invece imposta il **link per clonare il repository** e le credenziali dell'account GitHub

The screenshot shows the 'Source Code Management' section. The 'Git' radio button is selected. Under 'Repositories', the 'Repository URL' is 'https://github.com/dpasqualiReply/JenkinsDemo.git' and the 'Credentials' dropdown is set to 'dpasqualiReply/*****'. There are 'Advanced...' and 'Add Repository' buttons. Under 'Branches to build', the 'Branch Specifier (blank for \'any\')' is set to '*/master', with an 'Add Branch' button. The 'Repository browser' is set to '(Auto)'.

Imposta poi il build del job Jenkins in modo che venga eseguito ogni volta che viene fatto un commit git.

Inoltre puoi impostare una serie di comandi che vengono fatti durante la build, strumento potentissimo, **da imparare ad usare**.

The screenshot shows two sections of the Jenkins configuration. The 'Build Triggers' section has checkboxes for 'Trigger builds remotely (e.g., from scripts)', 'Build after other projects are built', 'Build periodically', 'GitHub hook trigger for GITScm polling' (checked), and 'Poll SCM'. The 'Build' section shows an 'Execute shell' step with the command 'echo "Build Done"'.

Configurazione Webhook GitHub

Un webhook è un collegamento che permette di scatenare un determinato evento tra due differenti servizi web. In questo caso, vogliamo fare in modo che, nel momento in cui viene fatto il commit di su GitHub, venga fatta una POST <http://35.196.238.203:8080/hithub-webhook>. Questa chiamata scatenerà build, test, deploy e compagnia bella.

La configurazioni deve essere fatta lato GitHub, al momento è implementata come un servizio, non come un webhook, andiamo quindi nella pagina **Integration & Services**

Imposta il link del server su cui è presente jenkins.

Jenkins GitLab Demo

Seguirò questa [guida](#).

GitLab ha davvero un sacco di problemi con Jenkins, da quel che dice il web funziona solo con la versione enterprise di GitLab, oppure con il tool di CI proprietario di GitLab (ovviamente a pagamento, bastardi).

Chiave SSH per il deploy

Per prima cosa è necessario configurare una chiave RSA per il deploy, questa procedura è comoda perchè permette al server jenkins di accedere ai repository di nostro interesse, senza la necessità di creare utenti "dummy" con diritti custom.

Generare la chiave

```
su
Password: ****

cd /var/lib/jenkins/.ssh
sudo -u jenkins ssh-keygen

Generating public/private rsa key pair.
Enter file in which to save the key (/var/lib/jenkins/.ssh/id_rsa): id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_rsa.
Your public key has been saved in id_rsa.pub.
The key fingerprint is:
```

```
SHA256:pELwRD6I+aWpM3XZ2R2mbZxh018dLNL4BBsm9/LPbzY jenkins@devops-worker
The keys randomart image is:
+---[RSA 2048]-----+
|  ..o   .=.  |
| o *      o + . |
|o . *    .. + + o |
| . + . o . =.. o |
| o . . S .*o. o |
| o . o o Bo= .. |
| . o o o o *. . |
| + .      . oEo |
| o          =+ |
+-----[SHA256]-----+
```

Una volta generata la chiave, copiala e inseriscila nel repository GitLab

```
cat id_rsa.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQAC9WxJsoJXl1XpMQk2bAaRSfKzYjBq5NeN0khzvCP2cnyKm0NK409I1WpDpTec5q0SaOsJ6e
0IGftKHxVRXLnH39jOWZVGEbZ096pzZIyGZAMdq2xM+Azm6CpcS6V1/XGyer8gtsvKDj5kD5q8EUfALKV6y642V7dh6UIcacyQuL6
tF6xumaCLW+20ASUmkucqmdQ9aSi08c8HsxiiIAQmZs6XCS8ecCRo08LyhLCLSGuwwaMFtWpomINTOApH/6k9nQDWWM3K1/1+uUoo
mAyuj7G1zj7gHAp4oxX05/rIZ4l0YnvYw9ods8/NWmy6349/5nXnITuD0afrQLJnaj0Dv jenkins@devops-worker
```

Inserire la chiave su GitLab

Apri il repository GitLab a cui vuoi applicare la Chiave di Deploy

The screenshot shows the GitLab interface for a repository named 'jenkins-CI-test' by user 'Dario Pasquali'. The repository is currently empty. The page includes navigation tabs for Project, Issues, Merge Requests, Pipelines, Wiki, Snippets, Members, and Settings. Below the repository name, there are buttons for Star, HTTP, and a link to the repository URL. A message states: 'The repository for this project is empty. If you already have files you can push them using command line instructions below. Otherwise you can start with adding a README, a LICENSE, or a .gitignore to this project. You will need to be owner or have the master permission level for the initial push, as the master branch is automatically protected.'

Settings -> Repository -> Deploy Keys

The screenshot shows the 'Settings' page for the repository, with the 'Repository' tab selected. Under the 'Repository' tab, the 'Deploy Keys' section is visible. It includes a heading 'Protected Branches' with a description 'Keep stable branches secure and force developers to use merge requests.' and an 'Expand' button. Below that is the 'Protected Tags' section with a description 'Limit access to creating and updating tags.' and an 'Expand' button. At the bottom is the 'Deploy Keys' section with a description 'Deploy keys allow read-only or read-write (if enabled) access to your repository. Deploy keys can be used for CI, staging or production servers. You can create a deploy key or add an existing one.' and an 'Expand' button.

Dai un nome alla chiave, copiala e spunta *Write access allowed*

Deploy Keys

Deploy keys allow read-only or read-write (if enabled) access to your repository. Deploy keys can be used for CI, staging or production servers. You can create a deploy key or add an existing one.

Create a new deploy key for this project

Title

jenkins-ci

Key

ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDIBtLSjBeSFxOGg9ieJhsvEBCn4DldXp/5D+qrlQP74m/hk29O9pDaE6nRP5bTgst6B3kqg6kboB
C+Lbhnmos55b0+Fco8zFBxuOl8cNbtoH1814tTJ5k6j7LUnMqIZ2RovyvHsKWmSd0n8THbRNCNqSGo5ld53GIFNA1EPaAwSFukGKny7p/v
1gdKrTAVo7UjokkYxgh7G8hxlUKUOdBnOzSsrPw02E2wWkXYwjsNidJOLBCikk91n0dffYH50j36PqfOUoEnUvqZRBMoAHyPR4lEdhkZG1/T
GgN8jaDwQ92QThen8Pk61fGiHeR7XslUTZMmV9bjh4EU8nE6DbR/t dario_pasquali93@devops-worker

Paste a machine public key here. Read more about how to generate it [here](#)

☒ Write access allowed

Allow this key to push to repository as well? (Default only allows pull access.)

Add key

Inserire la chiave in jenkins

La stssa chiave deve essere inserita in Jenkins, per farlo vai in Credentials -> System -> freccina nel dominio -> Add Credentials

Jenkins

Cerca

admin | dis

Jenkins > Credentials > System >

Nuovo Elemento

Utenti

Cronologia compilazioni

Relazioni fra progetti

Controlla impronta file

Gestisci Jenkins

Le mie viste

Credentials

System

Add domain

New View

System

Domain	Description
Global credentials (unrestricted)	Credentials that should be available irrespective of domain specification to requirements matching.

Icona: S M

Add credentials

Scope

Global (Jenkins, nodes, items, all child items, etc)

Username

jenkins

Private Key

Enter directly

From a file on Jenkins master

File

/var/lib/jenkins/.ssh/id_rsa

From the Jenkins master ~/.ssh

Passphrase

ID

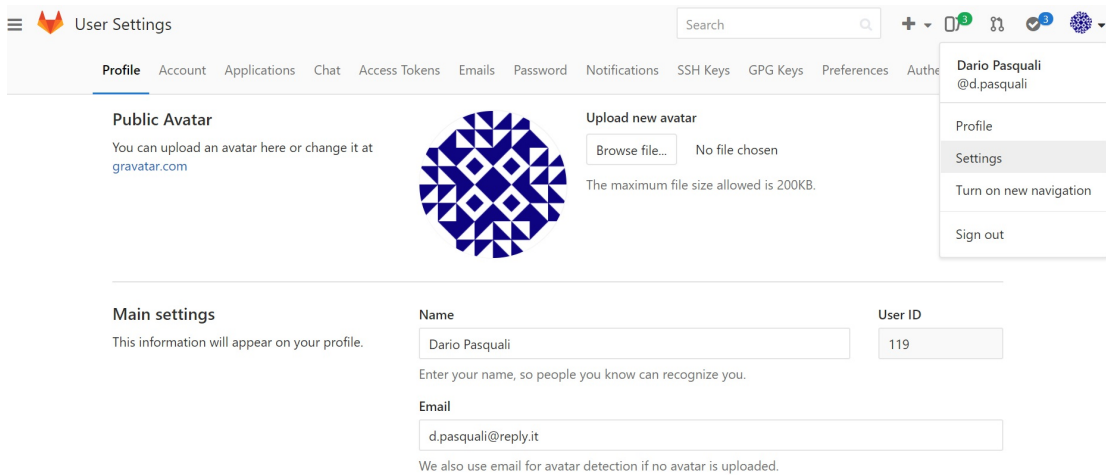
cf832894-f3dc-4707-8f10-fd90dddcfe9e

Description

Save

Creare un Access Token GitLab

Accedere alle configurazioni del profilo in GitLab -> Access Tokens



The image shows the GitLab User Settings page. At the top, there's a navigation bar with 'User Settings' and a search bar. Below it, a sub-navigation bar includes 'Profile', 'Account', 'Applications', 'Chat', 'Access Tokens', 'Emails', 'Password', 'Notifications', 'SSH Keys', 'GPG Keys', 'Preferences', and 'Auth'. The 'Profile' tab is active. The main content area is divided into two sections. The top section is 'Public Avatar', showing a blue geometric avatar and an 'Upload new avatar' button. The bottom section is 'Main settings', which includes fields for 'Name' (Dario Pasquali), 'User ID' (119), and 'Email' (d.pasquali@reply.it). A dropdown menu is open on the right, showing options like 'Profile', 'Settings', 'Turn on new navigation', and 'Sign out'.

Dai un nome al tuo token e seleziona una expiration date

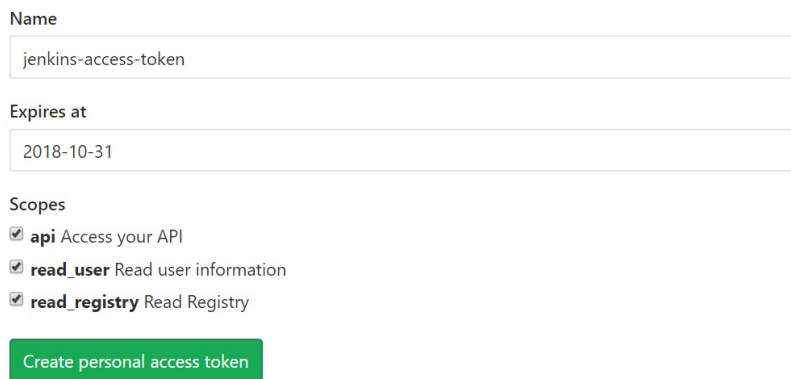
Personal Access Tokens

You can generate a personal access token for each application you use that needs access to the GitLab API.

You can also use personal access tokens to authenticate against Git over HTTP. They are the only accepted password when you have Two-Factor Authentication (2FA) enabled.

Add a personal access Token

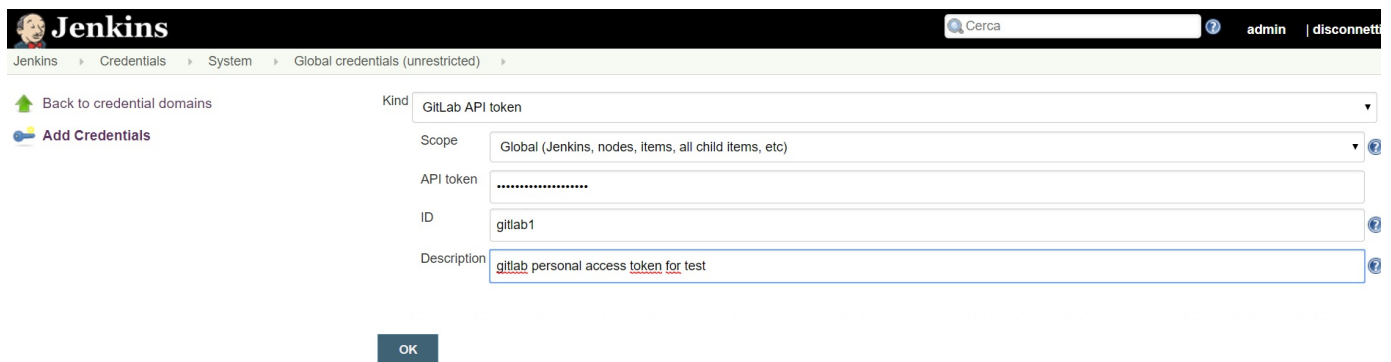
Pick a name for the application, and we'll give you a unique personal access Token.



The form for adding a personal access token has three main sections. The first section is 'Name', with a text input field containing 'jenkins-access-token'. The second section is 'Expires at', with a date input field showing '2018-10-31'. The third section is 'Scopes', which includes three checked checkboxes: 'api Access your API', 'read_user Read user information', and 'read_registry Read Registry'. At the bottom of the form is a green button labeled 'Create personal access token'.

Viene quindi generato e mostrato il tuo Personal Access Token super segretissimo, mi raccomando **SALVALO** perchè non te lo faranno vedere mai più.

Metti quindi il token in jenkins creando una nuova Credentials come prima.



The image shows the Jenkins 'Add Credentials' page. The breadcrumb trail at the top is 'Jenkins > Credentials > System > Global credentials (unrestricted)'. The page has a sidebar with 'Back to credential domains' and 'Add Credentials'. The main form is for adding a 'GitLab API token'. It includes fields for 'Kind' (GitLab API token), 'Scope' (Global (Jenkins, nodes, items, all child items, etc)), 'API token' (masked with dots), 'ID' (gitlab1), and 'Description' (gitlab personal access token for test). An 'OK' button is at the bottom.

Configurare la connessione tra Jenkins e GitLab

Torniamo in Manage Jenkins -> Configure System.

Sotto **Git Lab** sarà tutto rosso-incazzato, inserisci le informazioni richieste e sarà contento.

Fai **attenzione**, il *Gitlab host URL non è quello del tuo progetto, ma la radice, cioè il server su cui gira Gitlab.

Premendo *Test Connection* potrai ottenere un dolcissimo e gratificante *Success*.

GitLab

Enable authentication for "/project" end-point

GitLab connections

☒

Connection name

Mondadori GitLab

A name for the connection

http://jarvis.datareply.eu

The complete URL to the Gitlab server (i.e. http://gitlab.org)

GitLab API token (gitlab personal access token for !

Add

API Token for accessing Gitlab

autodetect

API Level for accessing Gitlab

Ignore SSL Certificate Errors

Connection timeout (in seconds)

10

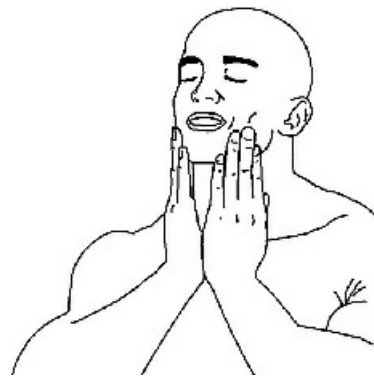
Read timeout (in seconds)

10

Test Connection

Elimina

Creazione del Job Jenkins



Esattamente come per Github, notare che la *GitLab Connection* viene inserita da sola

General

Gestione codice sorgenteTrigger compilazioneAmbiente di compilazioneCompilaAzioni post compilazione

Nome Progetto

GitLab-demo

Descrizione

gitlab demo for Mondadori

[Testo semplice] [Anteprima](#)

☐ Elimina compilazioni precedenti

☐ GitHub project

GitLab connection

Mondadori GitLab

☐ Questo progetto è parametrizzato

☐ Throttle builds

☐ Disabilita questo progetto

☐ Esegui compilazioni parallele se necessario (beta)

Avanzate...

Nella sezione *Gestione Codice Sorgente* inserisci

Config	Value
URL	url del repository Git Lab
Credentials	Le credenzioni RSA configurate con l'utente jenkins
Name	origin
Refspec	+refs/heads/:refs/remotes/origin/ +refs/merge-requests//head:refs/remotes/origin/merge-requests/
Branch	origin/\${gitlabSourceBranch}

Gestione codice sorgente

☐ Nessuno
☒ Git

Depositi

URL di Deposito:

Credenziali: [Add](#)

Nome:

Refspec:

[Aggiungi Deposito](#)

Rami a costruire

Ramo (lasciare in bianco per alcune):

[Aggiungi Ramo](#)

Browser repository:

Comportamenti supplementari: [Aggiungi](#)

Impostiamo in modo che la compilazione avvenga ogni volta che avviene un qualche evento sul repository. Ovviamente configurare sulla base delle proprie esigenze.

Attenzione Segnati l'URL del servizio, nel mio caso <http://35.196.238.203:8080/project/GitLab-demo>.

Trigger compilazione

☐ Attiva una compilazione da remoto (ad esempio da uno script)
☐ Compila dopo aver compilato gli altri progetti
☐ Compila periodicamente
☒ Build when a change is pushed to GitLab. GitLab CI Service URL: <http://35.196.238.203:8080/project/GitLab-demo>

Enabled GitLab triggers

Push Events	<input checked="" type="checkbox"/>
Opened Merge Request Events	<input checked="" type="checkbox"/>
Accepted Merge Request Events	<input checked="" type="checkbox"/>
Closed Merge Request Events	<input checked="" type="checkbox"/>
Rebuild open Merge Requests	<input type="text" value="Never"/>
Comments	<input checked="" type="checkbox"/>
Comment (regex) for triggering a build	<input type="text" value="Jenkins please retry a build"/>

[Avanzate...](#)

☐ GitHub hook trigger for GITScm polling
☐ Esegui polling sul sistema di controllo del codice sorgente

Aggiungi i passi di compilazione utili: **QUI VERRANNO INSERITI I TEST DIREI**

Compila

Esegui shell

Command:

[Si veda l'elenco delle variabili d'ambiente disponibili](#)


[Avanzate...](#)


Aggiungi passo di compilazione

- Esegui comando batch Windows
- Esegui shell
- Invoca target Maven principali
- Invoke Ant
- Invoke Gradle script
- Run with timeout
- Set build status to "pending" on GitHub commit

Aggiungi un'azione di post compilazione, possiamo per esempio inviare una email a destinatari custom, oppure, ome in questo caso, mostrare sulla GUI di GitLab lo stato del commit.

Azioni post compilazione

 Publish build status to GitLab commit (GitLab 8.1+ required)  

Build name 

Mark unstable builds as success ☐

Aggiungi azione post compilazione ▼

Creazione webhook lato GitLab

Lato GitLab è necessario configurare il Webhook in modo che venga fatta una POST in risposta a determinati eventi. Ovviamente configura il webhook in modo consistente con la configurazione degli eventi in Jenkins.

Project	Issues	Merge Requests	Pipelines	Wiki	Snippets	Members	Settings
General	Integrations	Repository	Pipelines				

Integrations

Webhooks can be used for binding events when something is happening within the project.

URL

Secret Token

Use this token to validate received payloads. It will be sent with the request in the X-Gitlab-Token HTTP header.

Trigger

- ☒ **Push events**
This URL will be triggered by a push to the repository
- ☐ **Tag push events**
This URL will be triggered when a new tag is pushed to the repository
- ☒ **Comments**
This URL will be triggered when someone adds a comment
- ☐ **Issues events**
This URL will be triggered when an issue is created/updated/merged
- ☐ **Confidential issues events**
This URL will be triggered when a confidential issue is created/updated/merged
- ☒ **Merge Request events**
This URL will be triggered when a merge request is created/updated/merged
- ☐ **Job events**
This URL will be triggered when the job status changes
- ☐ **Pipeline events**
This URL will be triggered when the pipeline status changes
- ☐ **Wiki Page events**
This URL will be triggered when a wiki page is created/updated

SSL verification

- ☐ **Enable SSL verification**

[Add webhook](#)

Test

Ok, dovrebbe essere tutto a posto, ora prega e fai un bel commit.