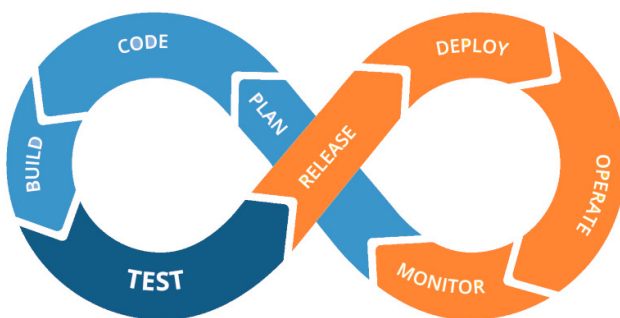




Università degli Studi di Catania
DIPARTIMENTO DI MATEMATICA E INFORMATICA
CORSO DI LAUREA MAGISTRALE IN INFORMATICA - CURRICULUM DATA SCIENCE

Pierpaolo Gumina

Continuous Integration/Continuous Deployment (CI/CD) con Git, Github, Jenkins, Maven, Ansible e Docker



Docente:

Giuseppe Pappalardo

Docente di laboratorio:

Andrea Francesco Fornaia

Anno Accademico - 2021/2022

Indice

1. Cos'è la CI/CD?
 - 1.1. Integrazione continua
 - 1.2. Distribuzione continua
2. Strumenti utilizzati:
 - 2.1. Git
 - 2.2. Github
 - 2.3. Jenkins
 - 2.4. Maven
 - 2.5. Ansible
 - 2.6. Docker
3. Implementazione
 - 3.1. CI/CD pipeline: Git, Jenkins e Maven
 - 3.1.1. Setup del Jenkins Server
 - 3.1.2. Integrare Git con Jenkins
 - 3.1.3. Creare un Elemento che scarica e integra il codice da Github
 - 3.1.4. Integrare Maven con Jenkins
 - 3.1.5. Compilare un progetto Java usando Jenkins
 - 3.2. CI/CD pipeline: Ansible, Dockerhub e Docker
 - 3.2.1. Installazione di Ansible
 - 3.2.2. Integrare Docker con Ansible
 - 3.2.3. Integrare Ansible con Jenkins
 - 3.2.4. Creare un'immagine e un container sul nodo Ansible
 - 3.2.5. Editare un playbook ansible per creare un'immagine ed un container
 - 3.2.6. Creare un container nel Docker server utilizzando un Playbook Ansible
4. Conclusioni e possibili sviluppi

1. Cos'è la CI/CD?

CI/CD è un metodo per la distribuzione frequente delle app ai clienti, che prevede l'introduzione dell'automazione nelle fasi di sviluppo dell'applicazione. Principalmente, si basa sui concetti di integrazione, distribuzione e deployment continui. L'approccio CI/CD supera le difficoltà legate all'integrazione di nuovo codice, una situazione così problematica per i team operativi e di sviluppo da essere conosciuta "inferno dell'integrazione".

Più specificamente, il metodo CI/CD introduce l'automazione costante e il monitoraggio continuo in tutto il ciclo di vita delle applicazioni, dalle fasi di integrazione e test a quelle di distribuzione e deployment. Nel complesso, questi processi interconnessi vengono definiti "flussi CI/CD" e sono supportati dai team operativi e di sviluppo che collaborano secondo modalità agili.

1.1 Integrazione continua

L'integrazione continua è un metodo di sviluppo software DevOps in cui gli sviluppatori aggiungono regolarmente modifiche al codice in un repository centralizzato, quindi la creazione di build e i test vengono eseguiti automaticamente. Per integrazione continua si intende principalmente la fase di creazione di build o di integrazione del processo di rilascio del software e implica sia un componente di automazione (ad es. l'integrazione continua o un servizio di creazione di build) sia un componente culturale (ad es. la decisione di integrare più di frequente nuovo codice). Gli obiettivi principali dell'integrazione continua sono individuare e risolvere i bug con maggiore tempestività, migliorare la qualità del software e ridurre il tempo richiesto per convalidare e pubblicare nuovi aggiornamenti.

Perché è necessaria l'integrazione continua?

In passato, gli sviluppatori di un team potevano lavorare separati gli uni dagli altri per un lungo periodo e cercavano di integrare le loro modifiche al master solo una volta completate. Questo rendeva difficile e dispendiosa in termini di tempo l'unione di modifiche del codice, comportando anche l'accumulo di bug per lunghi

periodi senza che venissero corretti. Questi fattori rendevano molto complicato offrire aggiornamenti rapidi ai propri clienti.

Come funziona l'integrazione continua?

Mediante l'integrazione continua, gli sviluppatori eseguono i commit in modo più frequente in un repository condiviso e impiegano un sistema di controllo della versione, ad esempio Git. Prima di applicare una modifica, possono eseguire unit test in locale, una verifica aggiuntiva importante prima dell'integrazione definitiva. Un servizio di integrazione continua crea automaticamente una build e avvia unit test sulle modifiche più recenti del codice per individuare immediatamente qualsiasi errore.

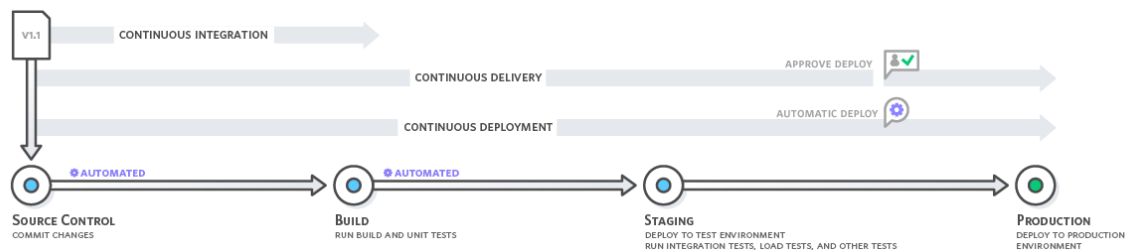


Figura 1.1 - Per integrazione continua si intendono principalmente le fasi di creazione di build e di unit test del processo di rilascio del software. Ogni nuova versione inoltrata attiva un processo automatizzato di creazione di build e di testing.

Vantaggi della distribuzione continua

1. Maggiore produttività per gli sviluppatori. L'integrazione continua consente al team di migliorare la produttività liberando gli sviluppatori dalle attività manuali e incoraggiando le pratiche che riducono il numero di errori e bug nel software distribuito ai clienti.
2. Bug individuati e risolti con maggiore prontezza. Aumentando la frequenza del testing, è più facile individuare tempestivamente e risolvere i bug prima che diventino problemi gravi.
3. Aggiornamenti più rapidi. L'integrazione continua consente di rilasciare aggiornamenti in modo più rapido e con maggiore frequenza.

1.2 Distribuzione continua

La distribuzione continua è un metodo di sviluppo software in cui le modifiche al codice vengono preparate automaticamente per un rilascio in produzione. Fondamento dello sviluppo moderno di applicazioni, la distribuzione continua estende l'integrazione continua distribuendo tutte le modifiche al codice all'ambiente di testing e/o di produzione dopo la fase di creazione di build. Se è implementata correttamente, gli sviluppatori avranno sempre a disposizione un artefatto di build pronto per la distribuzione che ha già superato un processo di testing standardizzato.

La distribuzione continua consente agli sviluppatori di automatizzare il testing oltre gli unit test, in modo da verificare l'applicazione degli aggiornamenti su vari livelli prima di renderli disponibili ai clienti. Queste prove possono includere test dell'interfaccia, test di caricamento, test di integrazione, test di affidabilità delle API e così via. In questo modo è più semplice per gli sviluppatori analizzare gli aggiornamenti più approfonditamente e rilevare preventivamente eventuali problemi. Grazie al cloud, automatizzare la creazione e la replica di più di un ambiente a scopo di testing è un'operazione molto semplice e poco costosa, al contrario di quanto avviene in ambienti locali.

Distribuzione continua e Implementazione continua

Con la distribuzione continua, ogni modifica al codice viene applicata a una build, testata e inoltrata in un ambiente di testing non in produzione o temporaneo. Possono essere previste diverse fasi di test in parallelo prima della distribuzione in produzione. La differenza tra distribuzione continua e implementazione continua è la presenza di un'approvazione manuale per l'aggiornamento della produzione. Con l'implementazione continua, la produzione avviene automaticamente, senza un'approvazione esplicita.

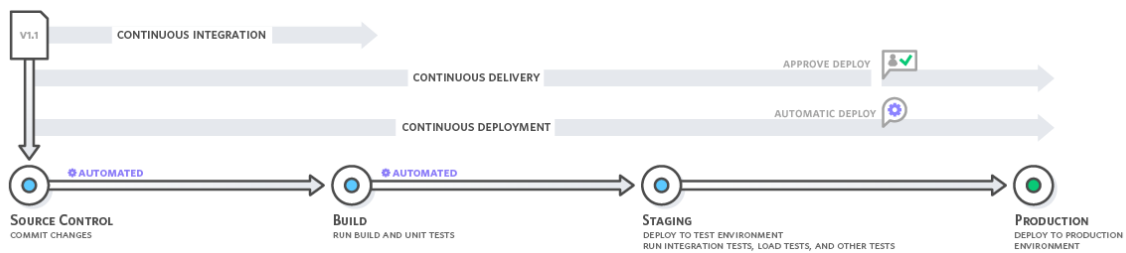


Figura 1.2 - La distribuzione continua automatizza l'intero processo di rilascio del software. Ogni nuova versione inoltrata attiva un flusso di lavoro automatizzato che applica a una build, testa e approva temporaneamente l'aggiornamento. La decisione finale per implementare il nuovo software nell'ambiente di produzione attivo dipende poi dallo sviluppatore.

Vantaggi della distribuzione continua

1. Processo di rilascio del software automatizzato. La distribuzione continua consente al team di applicare a una build, testare e preparare le modifiche al codice per l'inoltro in produzione in modo più efficiente e veloce.
2. Maggiore produttività per gli sviluppatori. Questo metodo consente al team di migliorare la produttività liberando gli sviluppatori dalle attività manuali e incoraggiando le pratiche che riducono il numero di errori e bug nel software rilasciato ai clienti.
3. Bug individuati e risolti con maggiore prontezza. Il team di sviluppo sarà in grado di rilevare e risolvere i bug prima che si trasformino in problemi gravi, grazie a testing più completi e più frequenti. La distribuzione continua consente di eseguire sul codice il genere di test che, senza un'automatizzazione completa, non sarebbe stato possibile effettuare.
4. Aggiornamenti più rapidi. La distribuzione continua consente di rilasciare aggiornamenti in modo più rapido e con maggiore frequenza. Se la distribuzione continua viene implementata correttamente, avrai sempre a disposizione una build temporanea pronta per la distribuzione che ha già passato un processo di testing standardizzato.

2. Strumenti utilizzati

Per la realizzazione del progetto sono state utilizzate i seguenti Tools: Git, Github, Jenkins, Maven, Ansible e Docker.

2.1 Git

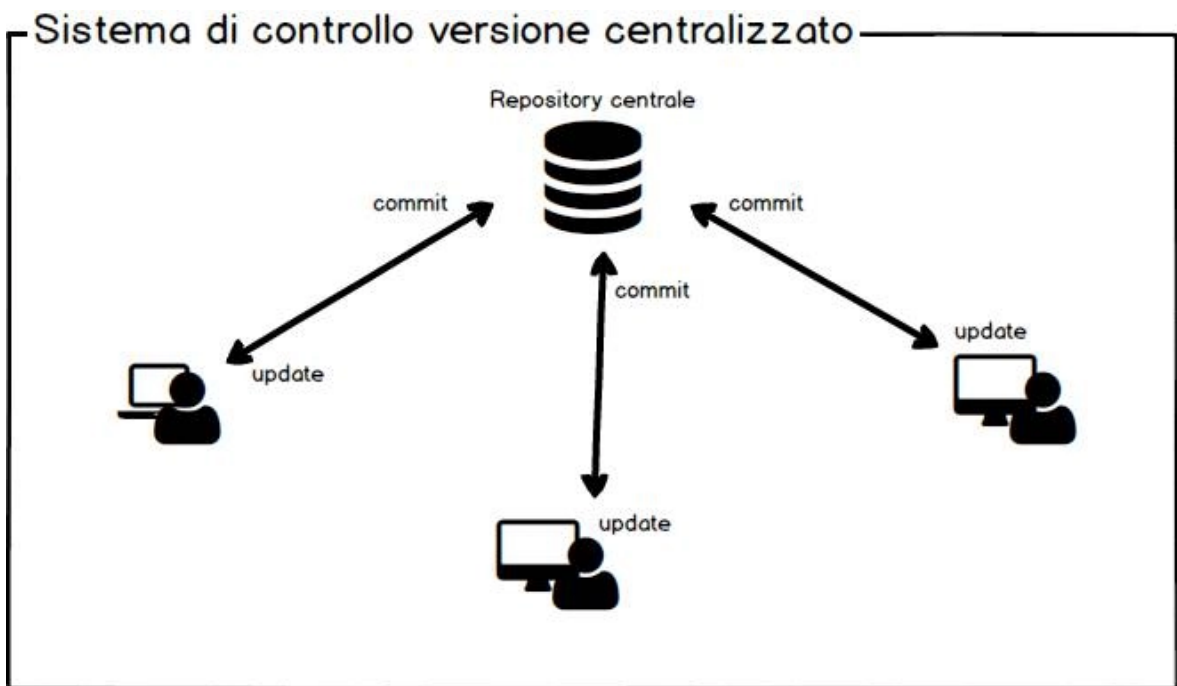


Git è un DVCS (acronimo di Distributed Version Control System, in italiano Sistema di Controllo Versione Distribuito) gratuito e open source che permette di tenere traccia delle modifiche apportate ai file e cartelle di un progetto nel corso del tempo, conservando la storia di tutti i cambiamenti effettuati così che sia sempre possibile accedere alle diverse versioni del progetto stesso. È inoltre possibile ripristinare, in parte o completamente, la struttura della directory base in modo che sia in essa presente una determinata versione dei file e sottocartelle con le modifiche che erano state apportate in un esatto momento del passato.

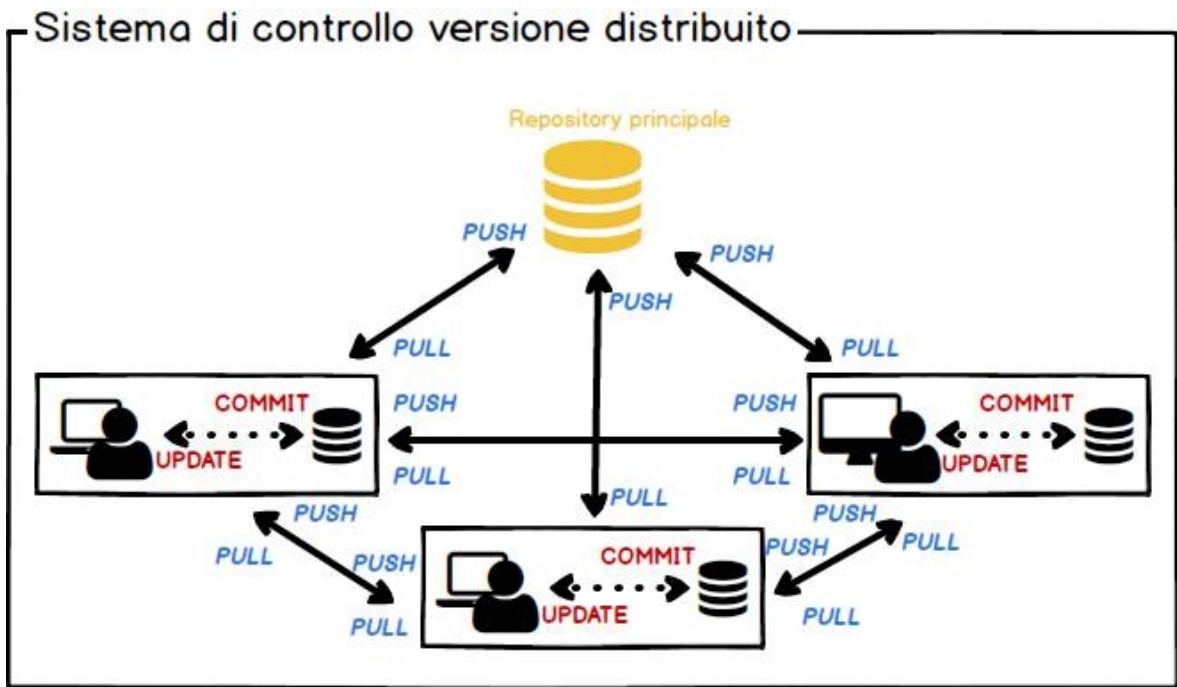
Una volta installato su un computer, (nella prossima lezione vedremo come eseguire l'installazione per i principali sistemi operativi) possiamo iniziare a tener traccia dei file presenti in una cartella lanciando un semplice comando. Git risulta particolarmente utile quando si lavora con dei file di testo semplice. Infatti, mette a disposizione vari strumenti per confrontare le diverse versioni di un file e vedere esattamente quali modifiche sono state effettuate tra una versione e l'altra. Nonostante siano disponibili numerose applicazioni con interfaccia grafica, Git viene solitamente usato tramite interfaccia a riga di comando. Per questo motivo è indicato soprattutto per lo sviluppo di software, ma potrebbe tranquillamente essere impiegato in altri ambiti. Per esempio, può risultare utile se si sta scrivendo un libro utilizzando un linguaggio come Markdown o Latex.

Git è in grado di tener traccia delle modifiche apportate a un gruppo di file e cartelle. Per far ciò salva le informazioni relative a gruppi di file all'interno di una struttura dati che prende il nome di repository. Grazie ai dati salvati nel repository è

possibile 'navigare indietro nel tempo' e visualizzare o ripristinare i dati contenuti in uno o più file o la struttura di intere cartelle.



In sistemi distribuiti come Git, al contrario di sistemi centralizzati, ciascun programmatore mantiene un repository locale e ognuno può lavorare e modificare il proprio repository indipendentemente dagli altri, senza dover necessariamente accedere a un repository centrale. Al contrario, in sistemi centralizzati è di solito presente un repository centrale da cui ognuno aggiorna o scarica l'ultima versione di un certo file. In questi casi è solitamente necessario essere collegati sempre al repository centrale.



Nel caso di sistemi distribuiti, ogni programmatore può lavorare indipendentemente e più velocemente col proprio repository locale. Anche se è possibile scambiare set di modifiche con gli altri collaboratori, in pratica si sceglie di far riferimento a un repository principale. È importante però sottolineare che tale scelta non è imposta dall'architettura di Git. Definito un repository principale, ogni programmatore provvederà a scaricare da quest'ultimo gli aggiornamenti e ad applicarli alla propria copia locale del repository. Invierà poi le proprie modifiche al repository eseguendo l'operazione 'push'.

Sintetizzando alcune delle caratteristiche e delle possibilità offerte da Git, possiamo dire che:

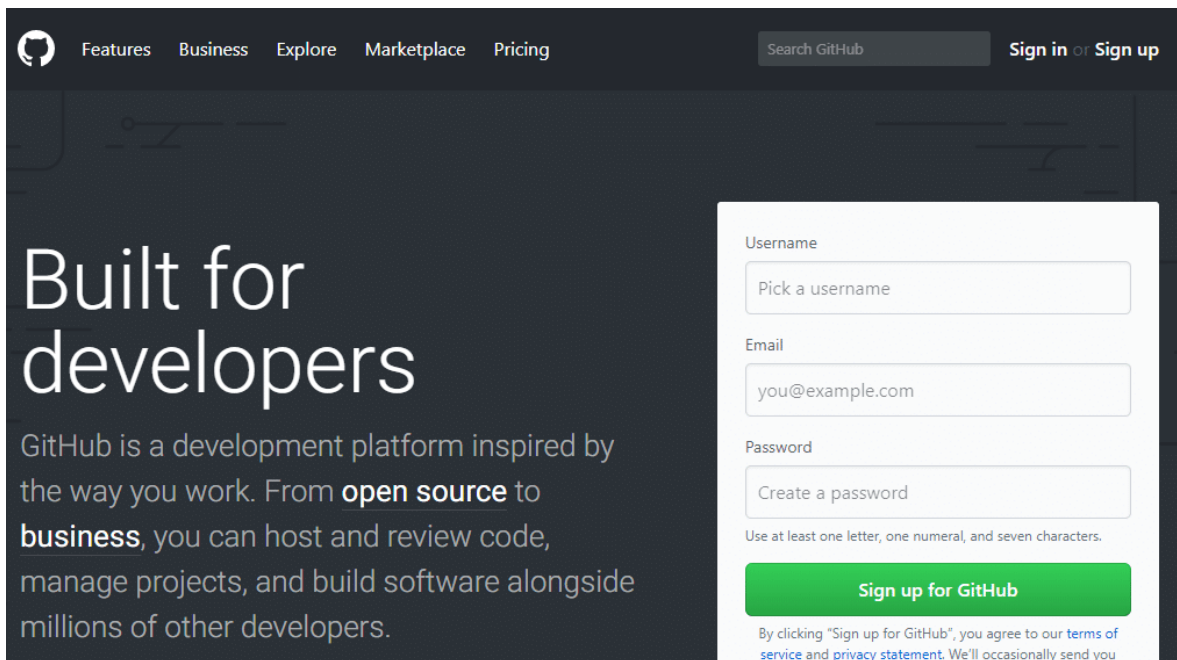
- permette di collaborare con altre persone in maniera efficiente;
- consente di 'andare indietro nel tempo' e visualizzare versioni precedenti di file e cartelle;
- permette di confrontare facilmente due differenti versioni dello stesso file e capire quali modifiche sono state apportate e chi è l'autore di tali cambiamenti;
- si integra alla perfezione con altri sistemi e strumenti per l'esecuzione di unit test e rilascio di nuove versioni di un software;

- consente di aggiungere nuove funzionalità a un progetto isolando le modifiche e le novità introdotte in modo da non causare perdite di dati o malfunzionamenti.

2.2 Github



GitHub è un'azienda a scopo di lucro che offre un servizio di hosting di repository Git basato su cloud. In sostanza, rende molto più facile per individui e team utilizzare Git per il controllo delle versioni e la collaborazione.



The screenshot shows the GitHub website's sign-up interface. The header includes navigation links: Features, Business, Explore, Marketplace, Pricing, a search bar labeled 'Search GitHub', and links for 'Sign in' or 'Sign up'. The main content area has a large heading 'Built for developers' and a descriptive paragraph: 'GitHub is a development platform inspired by the way you work. From **open source** to **business**, you can host and review code, manage projects, and build software alongside millions of other developers.' On the right, there is a sign-up form with fields for 'Username' (placeholder: 'Pick a username'), 'Email' (placeholder: 'you@example.com'), and 'Password' (placeholder: 'Create a password'). Below the password field, a note states: 'Use at least one letter, one numeral, and seven characters.' A prominent green button labeled 'Sign up for GitHub' is at the bottom of the form. A small disclaimer at the very bottom reads: 'By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy statement](#). We'll occasionally send you'.

L'interfaccia di GitHub è abbastanza facile da usare, quindi anche i programmatori alle prime armi possano sfruttare le funzionalità di Git. Senza GitHub, l'utilizzo di Git richiede generalmente una maggiore esperienza tecnica e l'utilizzo della linea di comando.

GitHub è così facile da usare, però, che alcune persone utilizzano GitHub anche per gestire altri tipi di progetti, come scrivere libri.

Inoltre, chiunque può iscriversi e ospitare gratuitamente un repository di codice pubblico, il che rende GitHub particolarmente popolare tra i progetti open-source.

Come azienda, GitHub guadagna vendendo repository ospitate di codice privato e altri piani focalizzati sul business che permettono alle organizzazioni di gestire i membri del team e la sicurezza con semplicità. Da Kinsta utilizziamo ampiamente Github per gestire e sviluppare progetti interni.

È bene sottolineare che Git e GitHub sono due cose diverse. Come già detto in precedenza, Git è un DVCS, ovvero un software che registra e mantiene traccia delle modifiche apportate a una serie di file. GitHub è una piattaforma online creata nel 2008 da Tom Preston-Werner, Chris Wanstrath e PJ Hyett ed è probabilmente la scelta più popolare per l'hosting dei repository Git. Github fornisce alcune funzionalità utili che aiutano gruppi di persone o team a collaborare ad un progetto. Github non è la sola compagnia a fornire un simile servizio, vi sono diversi siti concorrenti che offrono servizi analoghi. Uno dei più apprezzati è sicuramente Bitbucket che, al contrario di GitHub, permette di avere un numero di repository private illimitato sin dall'account base gratuito. GitHub è tuttavia la piattaforma più conosciuta e usata per progetti open source e, ad Aprile 2017, contava ben 20 milioni di utenti e 57 milioni di repository. Tra questi sono presenti anche i repository col codice sorgente di Git e Linux.

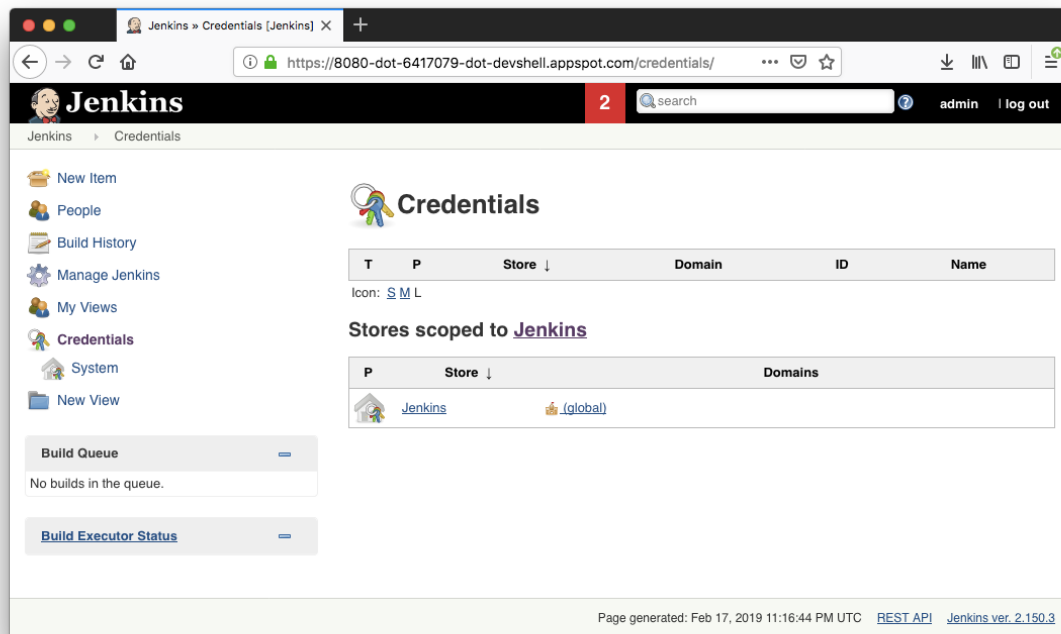
2.3 Jenkins



Jenkins è un'applicazione multiplatforma basata su Java, integrazione continua e distribuzione continua che aumenta la produttività complessiva. Jenkins può essere utilizzato per creare e testare continuamente progetti software, rendendo più facile per gli sviluppatori integrare le modifiche al progetto e rendendo più facile per gli utenti ottenere una nuova build. Consente inoltre di distribuire continuamente il software fornendo modi potenti per definire le pipeline di build e integrandosi con un gran numero di tecnologie di test e distribuzione.

Jenkins è un server di integrazione continua. In parole semplici, l'integrazione continua è la pratica di eseguire automaticamente i test su una macchina non

sviluppatrice ogni volta che qualcuno inserisce nuovo codice nel repository dei sorgenti.



Caratteristiche di Jenkins:

1. Jenkins può essere configurato interamente dalla sua intuitiva GUI web con estesi controlli degli errori al volo e aiuto in linea.
2. Jenkins si integra praticamente con ogni SCM o strumento di compilazione esistente oggi.
3. La maggior parte delle parti di Jenkins può essere estesa e modificata ed è facile creare nuovi plug-in Jenkins. Questa funzione ti consente di personalizzare Jenkins in base alle tue esigenze.
4. Jenkins può distribuire carichi di build / test su più computer con diversi sistemi operativi.

2.4 Maven



Maven, prodotto della Apache Software Foundation, è uno strumento di build automation utilizzato prevalentemente nella gestione di progetti Java. Simile per certi versi a strumenti precedenti, come ad esempio Apache Ant, si differenzia però da quest'ultimo per

quanto concerne la compilazione del codice. Con questo strumento infatti non è più necessaria la compilazione totale del codice, ma viene fatto uso di una struttura di progetto standardizzata su template definita archetype.

Il vantaggio derivante dall'utilizzo di questo tool è da subito evidente: se generalmente per sviluppare un software sono necessarie numerose fasi, con la build automation l'intero processo viene automatizzato, riducendo il carico di lavoro del programmatore e diminuendo le possibilità di errore da parte dello stesso. Alcune delle fasi fondamentali che vengono automatizzate dal tool sono la compilazione in codice binario, il packaging dei binari, l'esecuzione di test per garantire il funzionamento del software, il deployment sui sistemi ed infine la documentazione relativa al progetto portato a termine. Esistono poi alcuni tratti comuni ai prodotti di build automation, a partire dal build file, che contiene tutte le operazioni svolte.

I componenti principali di Apache Maven

Per capire meglio che cos'è Maven e come funziona, dopo aver evidenziato i progetti automatizzati, passiamo a considerare quelli che sono i suoi componenti principali e a spiegarne l'utilità ai fini della realizzazione di un progetto software. La prima caratteristica dello strumento è la presenza del file pom.xml, acronimo di Project Object Model, che ha il compito di definire identità e struttura del progetto. Il file pom.xml è diviso a sua volta in cinque parti, la relazione tra diversi file pom.xml, le build settings, la project information, il build environment e la configurazione dell'ambiente Maven.

Un'altra componente importante di Maven è rappresentata dai goal, ovvero l'insieme delle funzioni che possono essere eseguite sui diversi progetti. Poi bisogna menzionare la cartella repository, tramite la quale l'utente è in grado di gestire il sistema delle librerie. Ultima componente fondamentale del tool è il file settings.xml, usato per la configurazione di repository, proxy e profili. La particolarità di questa componente è che può essere specificata su due livelli, sotto user.home per il singolo utente oppure sotto maven.home per più utenti.

Una delle basi più importanti nell'utilizzo di Maven è la conoscenza dei plugin disponibili, che mettono a disposizione dell'utente specifici goal. Tra quelli principali vale la pena citare “clean” per la cancellazione dei compilati, “compiler” per i file sorgenti, “deploy” per il deposito nel repository remoto, “install” per il deposito nel repository locale, “site” per la generazione dei documenti relativi alle operazioni svolte nel processo, ed infine “archetype” per la generazione di un modello di struttura di progetto standard su template. Per imparare ad utilizzare questo tool e usufruire appieno delle sue potenzialità è dunque consigliabile rivolgersi ad aziende in grado di fornire formazione e supporto.

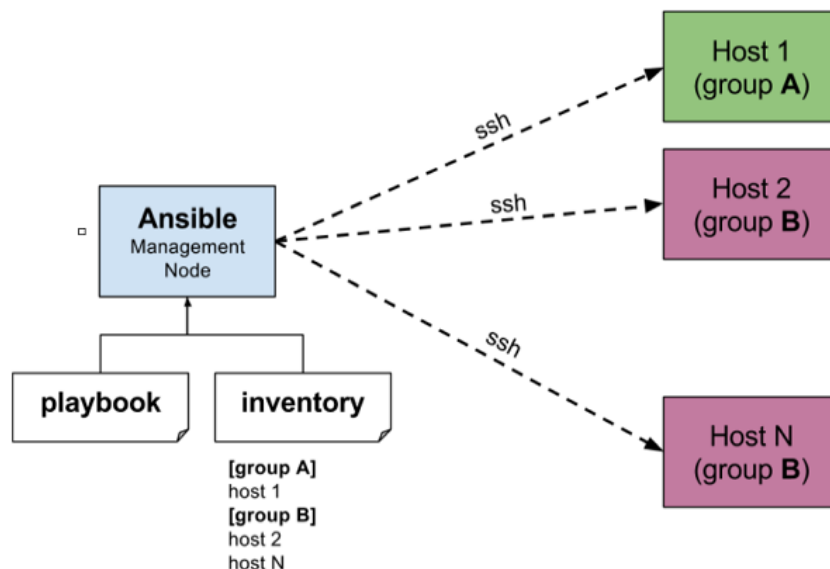
2.5 Ansible



Ansible è uno strumento open source per l'automazione IT che consente di automatizzare il provisioning, la gestione della configurazione, il deployment delle applicazioni, l'orchestrazione e molti altri processi IT manuali. Gli utenti di Ansible (amministratori di sistema, sviluppatori e architetti) possono sfruttare l'automazione Ansible per installare i software, automatizzare le attività quotidiane, eseguire il provisioning dell'infrastruttura, migliorare i livelli di sicurezza e conformità, applicare patch ai sistemi e condividere l'automazione in tutta l'azienda.

Ansible si connette ai sistemi che gli utenti desiderano automatizzare e avvia i programmi incaricati di eseguire le istruzioni che prima si sarebbero svolte manualmente. I programmi sfruttano i moduli Ansible ideati per soddisfare le aspettative specifiche di connettività, interfaccia e comandi dell'endpoint. Ansible esegue i moduli (su standard SSH per impostazione predefinita) e li rimuove una volta terminata la procedura (ove applicabile).

Non sono richiesti server, daemon o database aggiuntivi. In genere l'utente lavora con un programma terminale a sua scelta, un editor di testo e un sistema di controllo delle versioni per tenere traccia delle modifiche ai contenuti.



Provisioning dell'infrastruttura con Ansible

Prima di procedere all'installazione e alla configurazione di un'applicazione, è necessario preparare l'infrastruttura (ad esempio un endpoint server o cloud). Eseguire il provisioning di centinaia o addirittura migliaia di server manualmente è impensabile. Ecco perché le aziende moderne si orientano verso gli Ansible Playbook, che consentono loro di espandere l'ambiente IT rapidamente e in maniera affidabile. Con un Ansible Playbook è possibile creare un'istanza e utilizzare immediatamente quell'istanza o tutti gli altri server aggiuntivi che condividono gli stessi parametri o dettagli dell'infrastruttura. Eseguito il provisioning dell'ambiente, si passa alla configurazione e anche in questa fase del ciclo di vita dell'infrastruttura il contributo di Ansible è di notevole utilità.

Deployment delle applicazioni con Ansible

Ansible consente di eseguire il deployment di applicazioni multilivello in modo affidabile e coerente, da un framework comune. Permette anche di configurare i servizi necessari e di trasferire i componenti applicativi da un unico sistema condiviso.

Il team non deve più scrivere un codice personalizzato per l'automazione dei sistemi, ma soltanto descrizioni delle attività semplici, che anche i meno senior riescono a leggere senza difficoltà. Questo approccio riduce i costi e aumenta la capacità di reagire ai cambiamenti futuri.

2.6 Docker

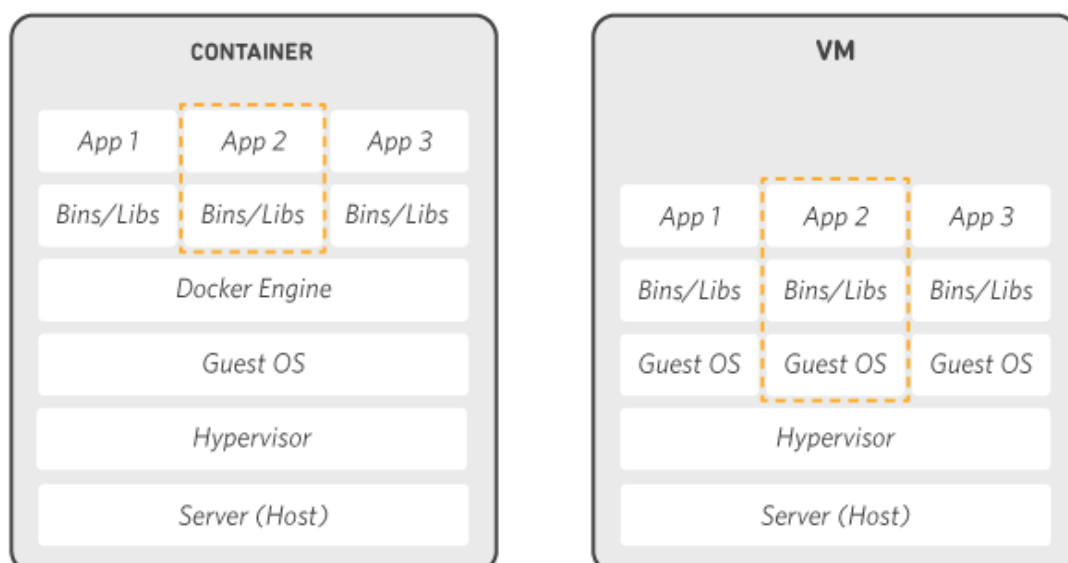


Docker è una piattaforma software che permette di creare, testare e distribuire applicazioni con la massima rapidità. Docker raccoglie il software in unità standardizzate chiamate container che offrono tutto il necessario per la loro corretta esecuzione, incluse librerie, strumenti di sistema, codice e runtime. Con Docker, è possibile distribuire e ricalibrare le risorse per un'applicazione in qualsiasi ambiente, tenendo sempre sotto controllo il codice eseguito.

L'esecuzione di Docker in AWS offre a sviluppatori e amministratori un modo altamente affidabile e poco costoso per creare, spedire ed eseguire applicazioni distribuite su qualsiasi scala.

Confronto tra contenitori Docker e macchine virtuali

- I contenitori includono l'applicazione e tutte le relative dipendenze. Condividono tuttavia il kernel del sistema operativo con altri contenitori, in esecuzione come processi isolati nello spazio utente nel sistema operativo host. Tranne che nei contenitori Hyper-V, in cui ogni contenitore viene eseguito all'interno di una macchina virtuale speciale per contenitore.
- Le macchine virtuali includono l'applicazione, le librerie o i file binari necessari e un sistema operativo guest completo. La virtualizzazione completa richiede più risorse rispetto alla creazione di contenitori.



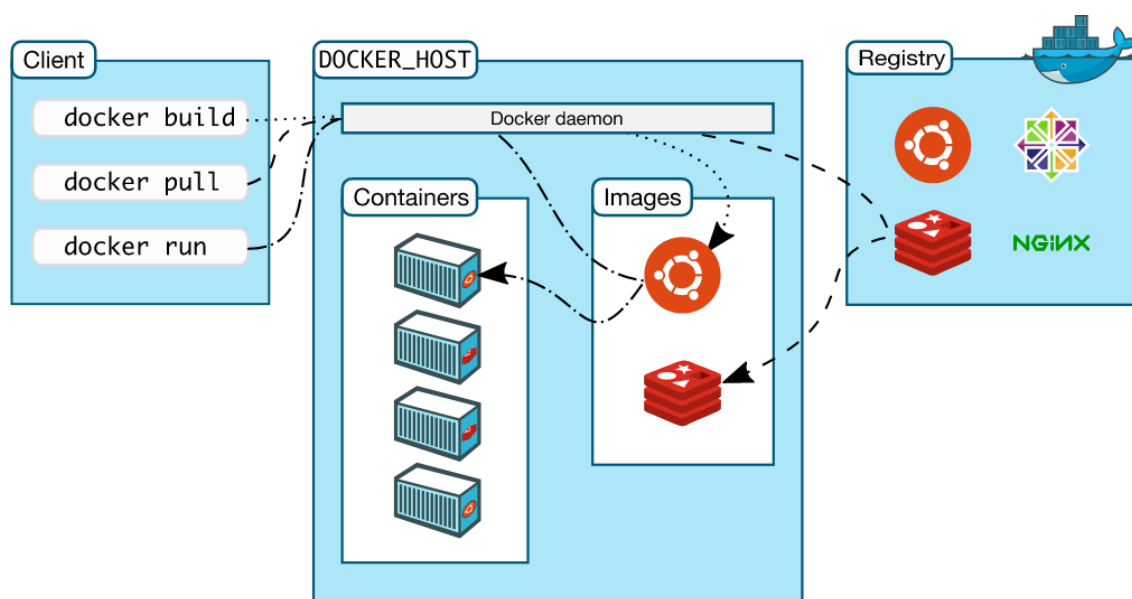
Per le macchine virtuali, sono disponibili tre livelli di base nel server host, dal basso verso l'alto: infrastruttura, sistema operativo host e un hypervisor. Ogni macchina virtuale ha inoltre un proprio sistema operativo e tutte le librerie necessarie. Per Docker, il server host prevede solo l'infrastruttura e il sistema operativo, ma anche il motore del contenitore, che mantiene il contenitore isolato, ma che condivide i servizi di base del sistema operativo.

Poiché i contenitori richiedono un numero molto ridotto di risorse, ad esempio non necessitano di un sistema operativo completo, sono facili da distribuire e si avviano rapidamente. In questo modo è possibile ottenere un aumento della densità, vale a dire che è possibile eseguire più servizi nella stessa unità hardware, riducendo i costi.

Come effetto collaterale dell'esecuzione sullo stesso kernel, si ottiene un isolamento minore rispetto alle macchine virtuali.

L'obiettivo principale di un'immagine è rendere l'ambiente (dipendenze) uguale su distribuzioni diverse. Questo significa che è possibile eseguirne il debug nel computer corrente e quindi distribuirla in un altro computer con lo stesso ambiente garantito.

Un'immagine del contenitore consente di creare un pacchetto di app o servizi e distribuirlo in modo affidabile e riproducibile. Si potrebbe dire che Docker non è solo una tecnologia, ma anche una filosofia e un processo.



Volendo riassumere quindi un container è quindi un ambiente isolato, ed è proprio l'isolamento e la sicurezza che ne derivano che ci permettono di eseguire più container simultaneamente all'interno di un host.

I container sono leggeri in quanto non necessitano della presenza di un hypervisor come nel caso delle macchine virtuali. Si eseguono infatti direttamente all'interno del kernel della macchina host.

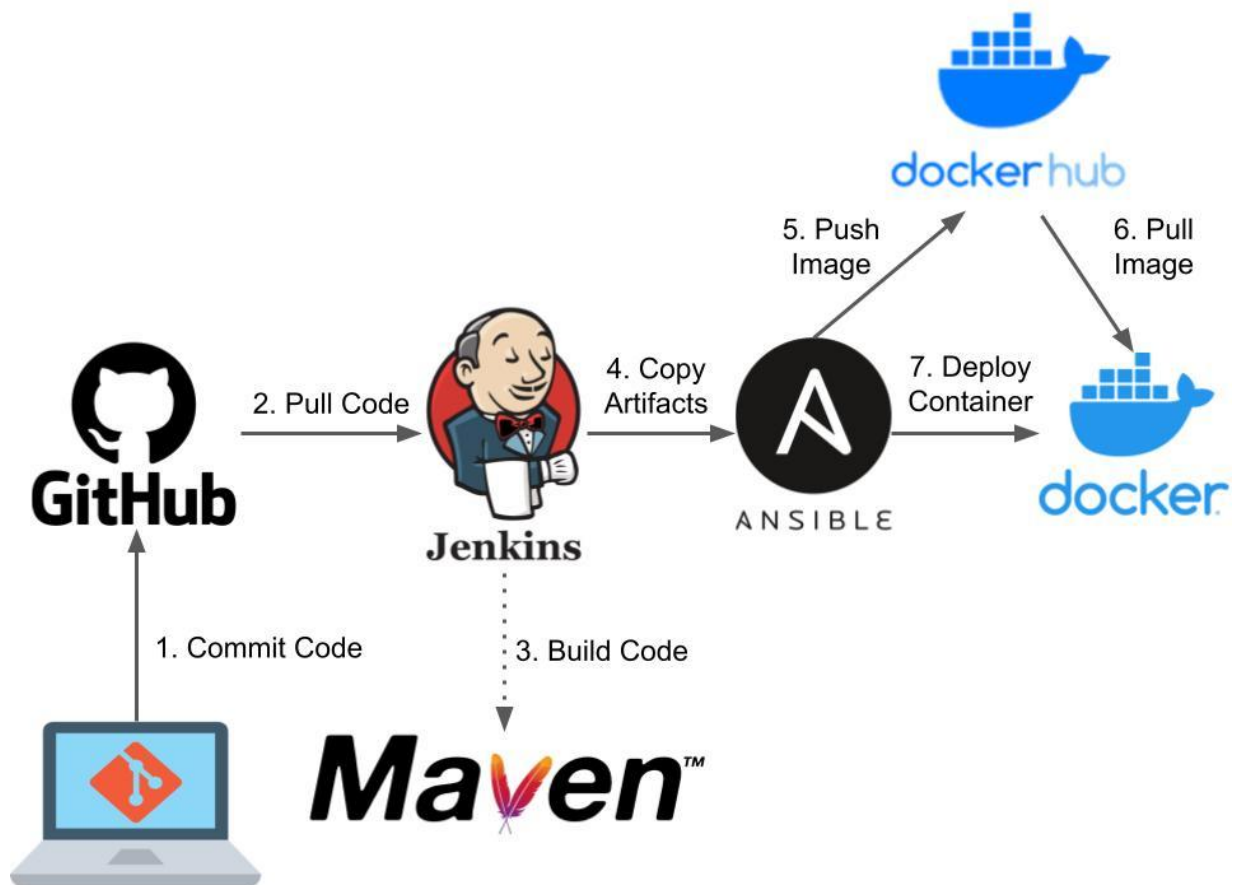
3. Implementazione

L'obiettivo del progetto è descrivere l'intera pipeline di CI/CD per compilare e rilasciare un applicativo java. Durante il progetto andremo ad usare: **Git**, un software per il controllo di versione distribuito; **Github**, un servizio di hosting per progetti software; **Jenkins server**, un tool per la Continuous Integration; **Maven**, uno strumento di build automation; **Ansible**, strumento di automazione delle procedure di configurazione e gestione (provisioning); e **Docker**, per la containerizzazione delle applicazioni.

Le macchine virtuali, su cui risiede l'intera infrastruttura di CI/CD, sono quelle offerte dal servizio cloud di Amazon: EC2 (Elastic Compute Cloud). Amazon EC2 è un servizio web in grado di fornire capacità di calcolo scalabile, più precisamente permette l'esecuzione di server virtuali in grado di ospitare servizi e applicazioni, ambienti di elaborazione virtuale meglio conosciuti come Istanze.

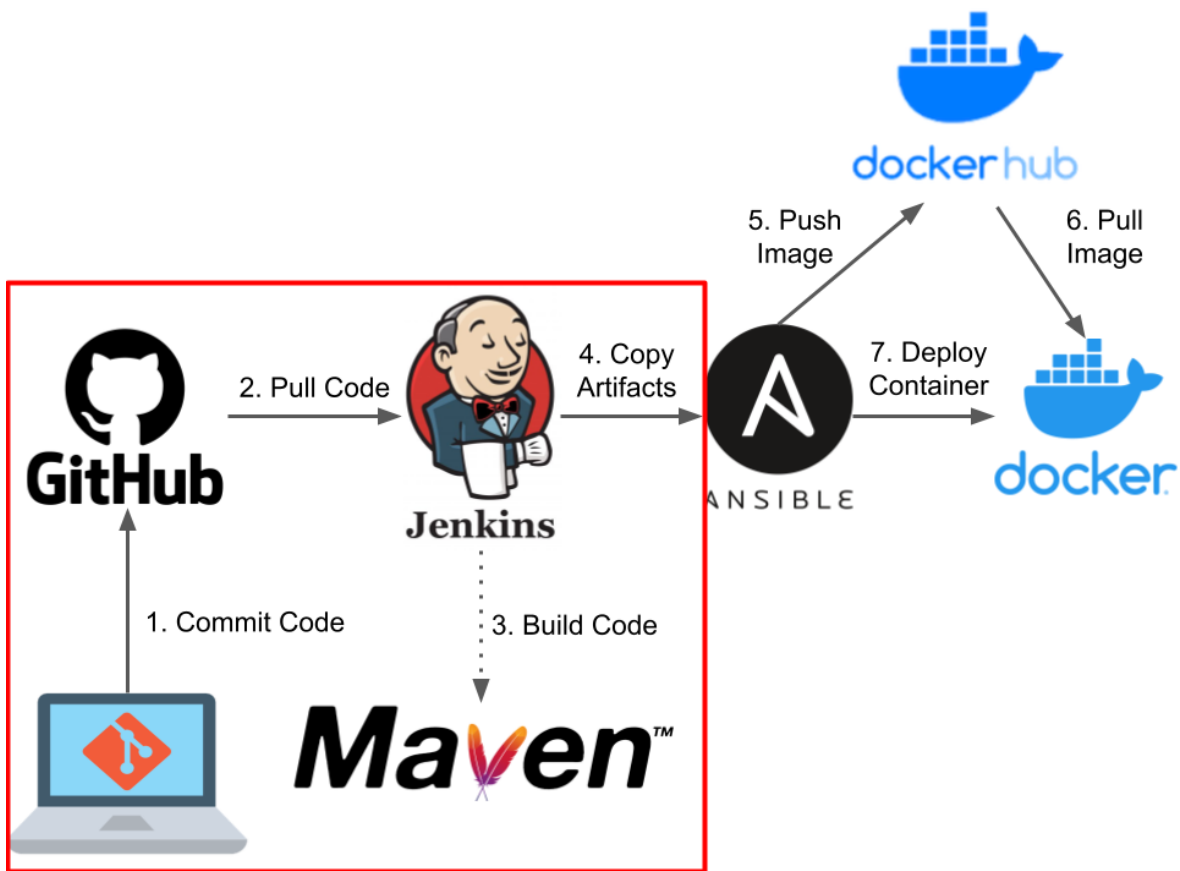
Una delle peculiarità più interessanti di questo servizio, da qui la definizione “Elastic”, consiste nella possibilità di gestione delle variazioni in termini di requisiti, popolarità e necessità proprie dell'impresa e della piattaforma: in sostanza, Amazon EC2 consente sia di calibrare gli interventi all'interno dello spazio preposto, che di ottenere e configurare capacità di elaborazione sicura e ridimensionabile.

Diamo uno sguardo come queste tecnologie cooperano:



1. Lo sviluppatore fa una commit del proprio codice nel branch master del Repository.
2. Poiché il codice deve essere compilato, il server Jenkins si occupa di prelevare il codice da Github.
3. Successivamente Jenkins compila il codice tramite Maven producendo un artefatto.
4. L'artefatto prodotto viene quindi copiato nel server Ansible.
5. Ansible provvede a creare un'immagine docker a partire dall'artefatto e la caricherà su docker hub.
6. Sempre il server Ansible avvierà il container docker sul server docker, il quale preleverà l'immagine dal docker hub.

3.1 CI/CD pipeline: Git, Jenkins e Maven



Nel processo di CI/CD abbiamo bisogno di compilare il nostro codice e distribuirlo in un ambiente "target". In questo lavoro abbiamo un progetto java memorizzato in un account Github, il quale dovrà essere compilato mediante Maven restituendo un artefatto. Il tutto deve essere pilotato dal server Jenkins, il quale verificherà se ci sono stati cambiamenti nel codice avviando di conseguenza il processo di CI/CD.

3.1.1 Setup del Jenkins Server

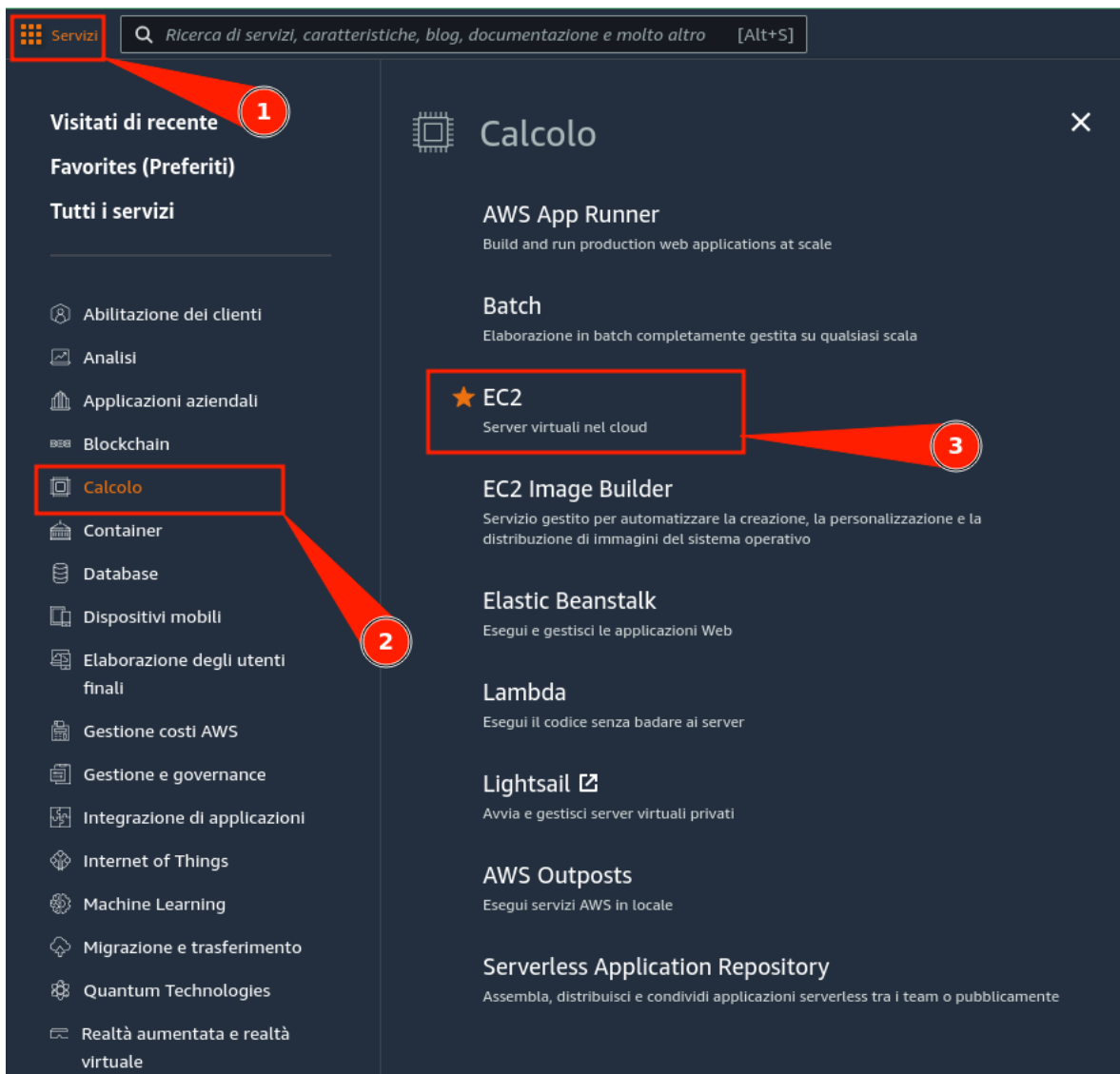
Per installare il Server Jenkins dobbiamo svolgere i seguenti passaggi:

1. Avvio di un'istanza EC2;
2. Installazione di Java;
3. Installazione Jenkins;
4. Avvio del servizio Jenkins;
5. Accedere tramite interfaccia web alla porta 8080.

Avvio di un'istanza EC2:

1. Una volta effettuato l'accesso al proprio account AWS dirigersi nel menu

Servizi →Calcolo →EC2:



2. Dal Pannello di controllo EC2 premere su Avvia istanza:

New EC2 Experience
Tell us what you think

Pannello di controllo EC2

- Visualizzazione di EC2
 - Global
 - Eventi
 - Tag
 - Limiti
- Istanze**
 - Istanze **New**
 - Tipi di istanza
 - Modelli di avvio
 - Richieste Spot
 - Savings Plans
 - Istanze riservate **New**
 - Host dedicati
 - Prenotazioni di capacità
- Immagini**
 - AMI **New**
 - Catalogo AMI

Risorse

Stai utilizzando le seguenti risorse Amazon EC2 nella regione Europa (Londra):

Istanze (in esecuzione)	0	Copie di chiavi
Gruppi di sicurezza	1	Host dedicati
Istanze	0	Sistemi di bilanciamento
Volumi	0	

Avvia istanza

Per iniziare, avvia un'istanza Amazon EC2, che è un server virtuale nel cloud.

Avvia istanza ▼ **Effettua la migrazione di un server** ↗

Nota: le tue istanze saranno avviate nella regione Europa (Londra)

3. Scegliere un'Amazon Machine Image (AMI), premere quindi su **Seleziona**:

1. Seleziona AMI 2. Scegli il tipo di istanza. 3. Configura l'istanza 4. Aggiungi storage 5. Aggiungi tag 6. Configura il gruppo di sicurezza 7. Esamina

Fase 1: Scegliere un'Amazon Machine Image (AMI) [Annulla ed esci](#)

L'AMI è un modello che contiene la configurazione software (sistema operativo, server di applicazioni e applicazioni) richiesta per avviare la tua istanza. Puoi selezionare un'AMI fornita da AWS, la nostra community di utenti o AWS Marketplace, oppure puoi scegliere una delle tue AMI.

Cerca un'AMI inserendo un termine di ricerca, ad esempio "Windows"

Ricerca per parametro Systems Manager

Quick Start

- Le mie AMI
- AWS Marketplace
- AMI della community
- ☐ Solo piano gratuito ⓘ

Amazon Linux
Idoneo al piano

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type - ami-078a289ddf4b09ae0 (64-bit (x86)) / ami-08ebecb25179560f7 (64-bit (Arm))

Amazon Linux 2 comes with five years support. It provides Linux kernel 5.10 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is now under maintenance only mode and has been removed from this wizard.

Tipo dispositivo root: ebs Tipo di virtualizzazione: hvm ENA abilitato: SI

Seleziona

- ☒ 64-bit (x86)
- ☐ 64-bit (Arm)

Amazon Linux
Idoneo al piano

Amazon Linux 2 AMI (HVM) - Kernel 4.14, SSD Volume Type - ami-030770b178fa9d374 (64-bit (x86)) / ami-0f2913dd376f23830 (64-bit (Arm))

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for

Seleziona

- ☒ 64-bit (x86)
- ☐ 64-bit (Arm)

4. Scegliere il tipo di istanza. In questo caso va bene una idonea al piano Free Tier. Premere quindi su **Successivo: Configura i dettagli dell'istanza**

Fase 2: Scegli il tipo di istanza.

Amazon EC2 offre un'ampia gamma di tipi di istanze ottimizzati per soddisfare diversi casi d'uso. Le istanze sono server virtuali che possono eseguire le applicazioni. Offrono diverse combinazioni di CPU, memoria, storage e capacità di rete per garantire la flessibilità di poter scegliere il mix di risorse più adatto alle tue applicazioni. [Ulteriori informazioni](#) sui tipi di istanze e su come possono soddisfare le tue necessità di calcolo.

Filtra per: Tutte le famiglie di istanze Generazione attuale Mostra/nascondi colonne

Attualmente selezionato: t2.micro (- ECU, 1 vCPU, 2.5 GHz, -, 1 GiB memoria, Solo EBS)

	Famiglia	Tipo	vCPU	Memoria (GiB)	Storage istanze (GB)	Disponibile ottimizzato EBS	Prestazioni di rete	Supporto per il protocollo IPv6
<input type="checkbox"/>	t2	t2.nano	1	0.5	Solo EBS	-	Da basso a moderato	Sì
<input checked="" type="checkbox"/>	t2	t2.micro <small>Idoneo al piano gratuito</small>	1	1	Solo EBS	-	Da basso a moderato	Sì
<input type="checkbox"/>	t2	t2.small	1	2	Solo EBS	-	Da basso a moderato	Sì
<input type="checkbox"/>	t2	t2.medium	2	4	Solo EBS	-	Da basso a moderato	Sì
<input type="checkbox"/>	t2	t2.large	2	8	Solo EBS	-	Da basso a moderato	Sì
<input type="checkbox"/>	t2	t2.xlarge	4	16	Solo EBS	-	Moderato	Sì
<input type="checkbox"/>	t2	t2.2xlarge	8	32	Solo EBS	-	Moderato	Sì
<input type="checkbox"/>	t3	t3.nano	2	0.5	Solo EBS	Sì	Fino a 5 Gigabit	Sì
<input type="checkbox"/>	t3	t3.micro	2	1	Solo EBS	Sì	Fino a 5 Gigabit	Sì

[Annulla](#)
[Precedente](#)
[Analizza e avvia](#)
[Successivo: Configura i dettagli dell'istanza](#)

- Configura i dettagli dell'istanza. Lasciare i parametri invariati e premere su **Successivo: Aggiungi storage**

Fase 3: Configura i dettagli dell'istanza

Configura l'istanza in base ai tuoi requisiti. Puoi avviare molteplici istanze dalla stessa AMI, richiedere istanze Spot per approfittare di prezzi inferiori, assegnare all'istanza un ruolo di gestione degli accessi e molto altro ancora.

Numero di istanze ⓘ	<input type="text" value="1"/>	Avvio nel gruppo Auto Scaling ⓘ
Opzione di acquisto ⓘ	<input type="checkbox"/> Richiesta di istanze Spot	
Rete ⓘ	<input type="text" value="vpc-829930eb (default)"/>	Crea nuovo VPC
Sottorete ⓘ	<input type="text" value="Nessuna preferenza (sottorete predefinita in qualsiasi sottorete)"/>	Crea una nuova sottorete
Assegna automaticamente IP pubblico ⓘ	<input type="text" value="Utilizza le impostazioni della sottorete (Abilita)"/>	
Tipo di nome host ⓘ	<input type="text" value="Utilizza le impostazioni della sottorete (Nome IP)"/>	
DNS Hostname ⓘ	<input checked="" type="checkbox"/> Enable IP name IPv4 (A record) DNS requests <input checked="" type="checkbox"/> Abilita le richieste DNS IPv4 (registro A) basate su risorse <input type="checkbox"/> Abilita le richieste DNS IPv6 (registro AAAA) basate su risorse	
Gruppo di collocamento ⓘ	<input type="checkbox"/> Aggiungi l'istanza a un gruppo di collocamento	
Prenotazione di capacità ⓘ	<input type="text" value="Apri"/>	
Directory aggiunta dominio ⓘ	<input type="text" value="Nessuna directory"/>	Crea una nuova directory
Ruolo IAM ⓘ	<input type="text" value="Nessuno"/>	Crea un nuovo ruolo IAM

Annulla
Precedente
Analizza e avvia
Successivo: Aggiungi storage

6. Anche nella sezione **Aggiungi storage** lasciamo i parametri invariati e procediamo premendo su **Successivo: Aggiungi tag**

1. Seleziona AMI 2. Scegli il tipo di istanza. 3. Configura l'istanza 4. Aggiungi storage 5. Aggiungi tag 6. Configura il gruppo di sicurezza 7. Esamina

Fase 4: Aggiungi storage

La tua istanza verrà avviata con le seguenti impostazioni relative al dispositivo di storage. Puoi collegare volumi EBS e volumi instance store aggiuntivi alla tua istanza o modificare le impostazioni del tuo volume root. Puoi anche collegare volumi EBS dopo l'avvio di un'istanza ma non volumi instance store. [Ulteriori informazioni](#) sulle opzioni di storage in Amazon EC2.

Tipo di volume ⁱ	Dispositivo ⁱ	Snapshot ⁱ	Dimensione (GiB) ⁱ	Tipo di volume ⁱ	IOPS ⁱ	Throughput (MB/s) ⁱ	Elimina al termine ⁱ	Crittografia ⁱ
Root	/dev/xvda	snap-0b8d2827c3cf43bef	8	SSD per utilizzo generico (gp2)	100 / 3000	N/D	<input checked="" type="checkbox"/>	Non crittogra ⁱ

Aggiungi nuovo volume

I clienti aventi diritto al piano gratuito possono ricevere fino a 30 GB di storage General Purpose (SSD) o Magnetic EBS. [Ulteriori informazioni](#) sull'idoneità e le limitazioni d'uso del piano di utilizzo gratuito.

Shared file systems ⁱ

You currently don't have any file systems on this instance. Select "Add file system" button below to add a file system.

Add file system

Annulla Precedente Analizza e avvia **Successivo: Aggiungi tag**

7. Aggiungiamo un tag e premere su **Successivo: Configura il gruppo di sicurezza**

1. Seleziona AMI 2. Scegli il tipo di istanza. 3. Configura l'istanza 4. Aggiungi storage 5. Aggiungi tag 6. Configura il gruppo di sicurezza 7. Esamina

Fase 5: Aggiungi tag

Ciascun tag è costituito da una coppia chiave-valore che distingue tra lettere maiuscole e minuscole. Ad esempio, potresti definire un tag con la chiave = Nome e il valore = Server Web.

Una copia di un tag può essere applicata a volumi, istanze o entrambi.

I tag saranno applicati a tutte le istanze e i volumi. [Ulteriori informazioni](#) sull'applicazione di tag alle risorse Amazon EC2.

Chiave (massimo 128 caratteri)	Valore (massimo 256 caratteri)	Istanze ⁱ	Volumi ⁱ	Interfacce di rete ⁱ
Name	Jenkins_Server	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Aggiungi un altro tag (Fino a massimo 50 tag)

Aggiungi un altro tag (Fino a massimo 50 tag)

Annulla Precedente Analizza e avvia **Successivo: Configura il gruppo di sicurezza**

8. Nella fase di **Configura il gruppo di sicurezza** andiamo a creare un nuovo

gruppo di sicurezza, e dopo aver dato un nome e una descrizione, inseriamo una nuova regola: la porta visibile all'esterno del server jenkins (8080). Premere quindi su **Analizza e avvia**:

1. Seleziona AMI 2. Scegli il tipo di istanza. 3. Configura l'istanza 4. Aggiungi storage 5. Aggiungi tag 6. Configura il gruppo di sicurezza 7. Esamina

Fase 6: Configura il gruppo di sicurezza

Il gruppo di sicurezza è un insieme di regole del firewall che controllano il traffico della tua istanza. In questa pagina, puoi aggiungere le regole per consentire a un traffico specifico di raggiungere la tua istanza. Ad esempio, se vuoi impostare un server Web e consentire al traffico Internet di raggiungere la tua istanza, devi aggiungere regole che consentano un accesso senza restrizioni alle porte HTTP e HTTPS. Puoi creare un nuovo gruppo di sicurezza o sceglierne uno esistente tra quelli elencati di seguito. [Ulteriori informazioni](#) sui gruppi di sicurezza Amazon EC2.

Assegna un gruppo di sicurezza: ☒ Crea un nuovo gruppo di sicurezza 1
☐ Seleziona un gruppo di sicurezza esistente 2

Nome del gruppo di sicurezza:
Descrizione:

Tipo	Protocollo	Intervallo porte	Origine	Descrizione
SSH	TCP	22	Personaliz: 0.0.0.0/0	ad esempio SSH for Admin Desl
Regola TCP p	TCP	8080	Personaliz: 0.0.0.0/0, ::/0	ad esempio SSH for Admin Desl

Aggiungi regola 3 4

Avviso

Le regole con origine 0.0.0.0/0 consentono a tutti gli indirizzi IP di accedere alla tua istanza. Consigliamo di configurare le regole dei gruppi di sicurezza per consentire l'accesso solo da indirizzi IP noti.

[Annulla](#) [Precedente](#) [Analizza e avvia](#) 5

9. Prima di avviare l'istanza è possibile visionare i passaggi precedenti modificando i parametri qualora servisse. Premere su **Lancio**:

1. Seleziona AMI 2. Scegli il tipo di istanza. 3. Configura l'istanza 4. Aggiungi storage 5. Aggiungi tag 6. Configura il gruppo di sicurezza 7. Esamina

Fase 7: Rivedi l'avvio dell'istanza

Analizza i dettagli di lancio della tua istanza. Puoi tornare indietro per modificare i cambiamenti di ogni sezione. Fai clic su **Avvia** per assegnare una coppia di chiavi alla tua istanza e completare il processo relativo.

Avviso

Migliora la sicurezza dell'istanza. Il gruppo di sicurezza Jenkins_Security_Group è aperto a tutti. Qualsiasi indirizzo IP può accedere all'istanza. È consigliabile aggiornare le regole dei gruppi di sicurezza per consentire l'accesso solo da indirizzi IP noti. Nel gruppo di sicurezza è anche possibile aprire porte aggiuntive per facilitare l'accesso all'applicazione o al servizio in esecuzione, ad es., HTTP (80) per i server Web. [Modifica i gruppi di sicurezza](#)

- ▶ Dettagli AMI [Modifica AMI](#)
- ▶ Tipo di istanza [Modifica tipo di istanza](#)
- ▶ Gruppi di sicurezza [Modifica i gruppi di sicurezza](#)
- ▶ Dettagli istanza [Modifica dettagli istanza](#)
- ▶ Storage [Modifica storage](#)
- ▶ Tag [Modifica tag](#)

[Annulla](#) [Precedente](#) [Lancio](#) 1

10. Creare un coppia di chiavi per accedere tramite SSH o, qualora si disponga

già di una, selezionarla e premere la spunta "di essere in possesso della chiave selezionata". Premere quindi su **Avvia le istanze**:

Seleziona una coppia di chiavi esistente oppure crea una nuova coppia di chiavi

Una coppia di chiavi è costituita da una **chiave pubblica** che AWS archivia, e da un **file di una chiave privata** che tu archivi. Insieme ti consentono di connetterti all'istanza in modo sicuro. Per le AMI di Windows, il file della chiave privata è necessario per ottenere la password di accesso alla tua istanza. Per le AMI di Linux, il file della chiave privata consente un SSH in sicurezza alla tua istanza. Amazon EC2 supporta i tipi di coppie di chiavi ED25519 e RSA.

Nota: la coppia di chiavi selezionata sarà aggiunta al set di chiavi autorizzate per questa istanza. Ulteriori informazioni in [Eliminare coppie di chiavi esistenti da un'AMI pubblica](#).

Scegli una coppia di chiavi esistente

Selezionare una coppia di chiavi

nessuna coppia di chiavi trovata

1 **2** **3**

nessuna coppia di chiavi trovata
Non hai coppie di chiavi. Crea una nuova coppia di chiavi selezionando l'opzione **Crea una nuova coppia di chiavi** qui sopra per proseguire.

Annulla **Avvia le istanze**

11. Infine per collegarsi all'istanza appena creata: aprire il terminale (Linux o MacOS), dirigersi nella stessa directory della chiave appena scaricata e digitare i seguenti comandi:

```
$ chmod 400 <nome_chiave>
$ ssh -i "<nome_chiave>"
ec2-user@ec2-15-161-103-40.eu-south-1.compute.amazonaws.com
```

nb: l'indirizzo IP sopra descritto è a titolo di esempio.

Installazione di Java e Jenkins:

Per installare Java e Jenkins nell'istanza creata precedentemente digitare i seguenti comandi:

```
$ sudo su -
$ wget -O /etc/yum.repos.d/jenkins.repo
https://pkg.jenkins.io/redhat-stable/jenkins.repo
```

```
$ rpm --import
https://pkg.jenkins.io/redhat-stable/jenkins.io.key
$ amazon-linux install epel
$ amazon-linux install java-openjdk11
$ yum install jenkins
```

Abbiamo installato con successo Jenkins adesso avviamo il servizio demone con il comando:

```
$ service jenkins start
```

Verifichiamo il corretto funzionamento del servizio Jenkins da browser digitando nella barra di ricerca: <indirizzo ipv4 dell'istanza>:8080. Ci verrà richiesto di inserire la password di default, quest'ultima reperibile mediante il comando:

```
$ cat /var/lib/jenkins/secrets/initialAdminPassword
```

3.1.2 Integrare Git con Jenkins

Solitamente il codice sorgente memorizzato in Github viene caricato mediante operazioni di "Pull" tramite Git. Similmente per copiare (Pull) il codice da Github a Jenkins è necessario installare Git sul server Jenkins. Fatto ciò sarà possibile integrare Github con Jenkins. I passi da seguire sono quindi:

1. Installare Git nell'istanza Jenkins;
2. Installare il plugin Github da Jenkins;
3. Configurare Git in Jenkins.

Installare Git nell'istanza Jenkins:

Per prima cosa cambiamo l'host name dell'istanza:

```
$ vi /etc/hostname
```

Per validare bisogna riavviare, digitare quindi:

```
$ init 6
```

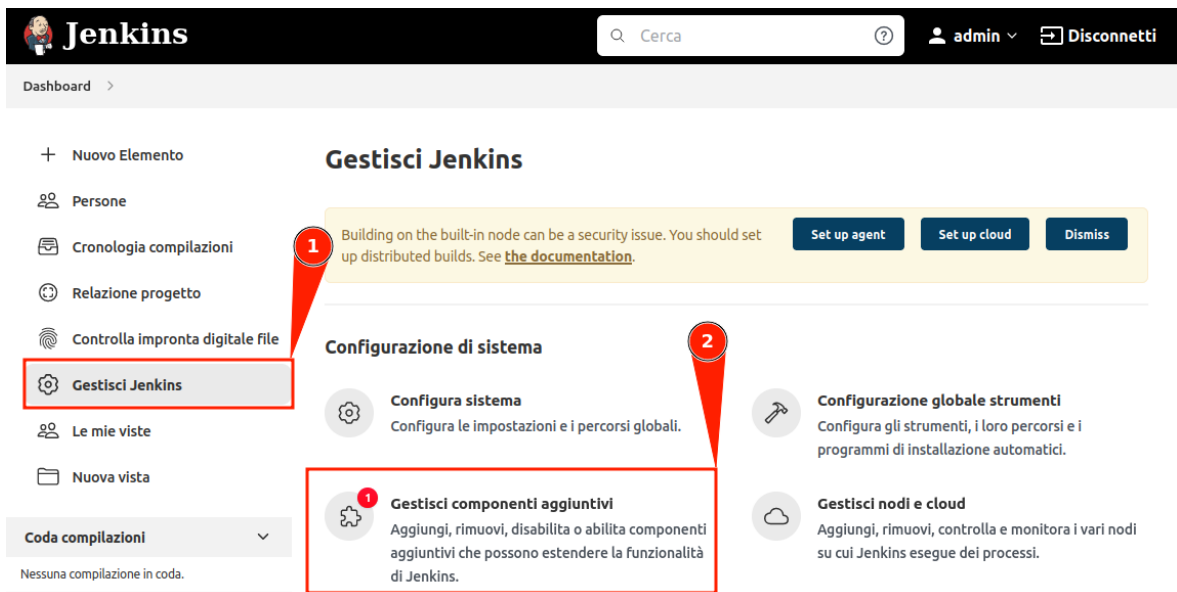
Riaccedere all'istanza e procediamo ad installare Git:

```
$ sudo su -
$ yum install git
$ git --version //per verificare la corretta installazione
```

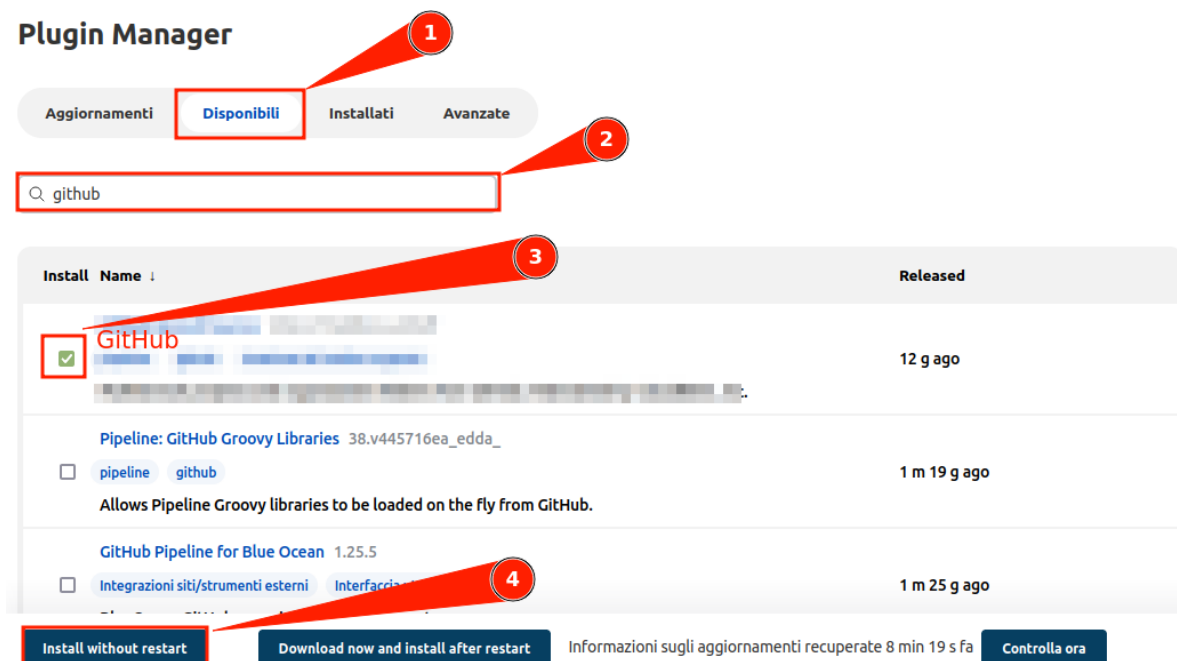
Installare il plugin Github da Jenkins:

Per installare il Plugin Github da Jenkins recarsi premere su **Gestisci Jenkins** e

successivamente su **Gestisci componenti aggiuntivi**:

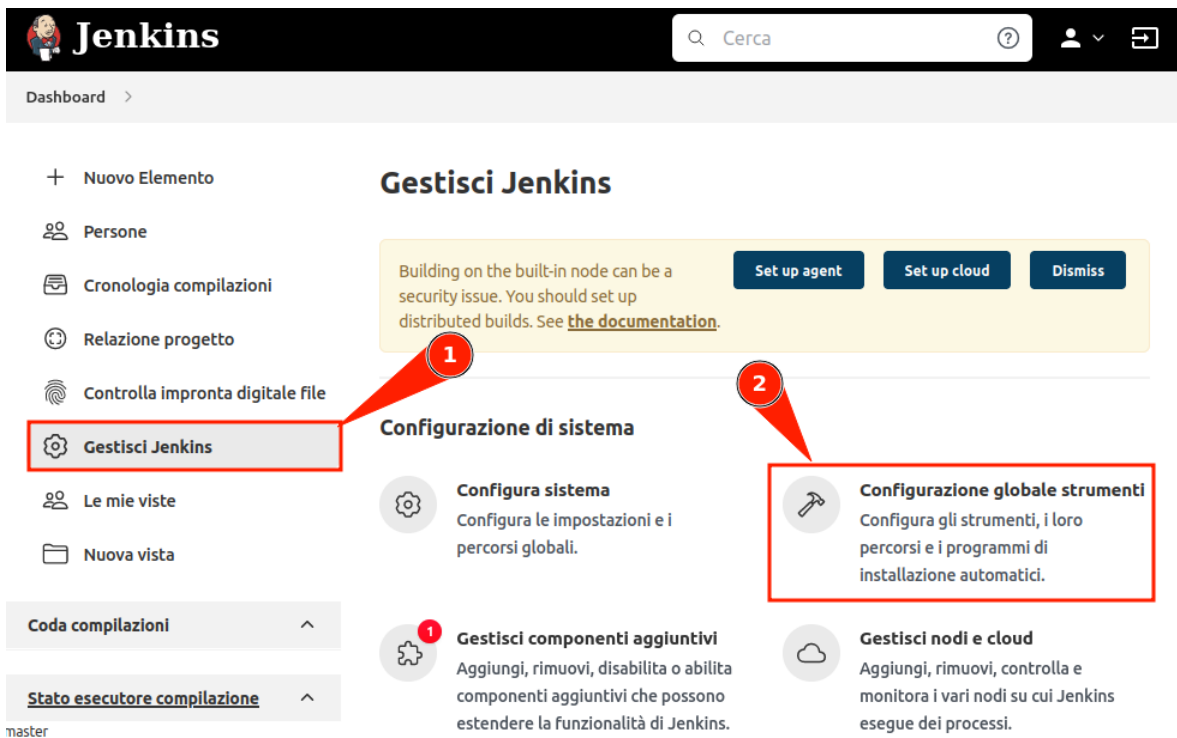


Premere su **Disponibili**, digitare nel campo di ricerca "github", spuntare la casella del plugin chiamato "Github" e infine premere su **Install without restart**:



Configurare Git in Jenkins:

Per configurare Git con Jenkins recarsi su **Gestisci Jenkins** → **Configurazione globale strumenti**:



Compilare i campi in modo che siano come l'immagine sottostante. Premere quindi su **Applica** e poi **Salva**:

Git

Installazioni di git

Git

Nome

Git

Percorso del file eseguibile git ?

git

☐

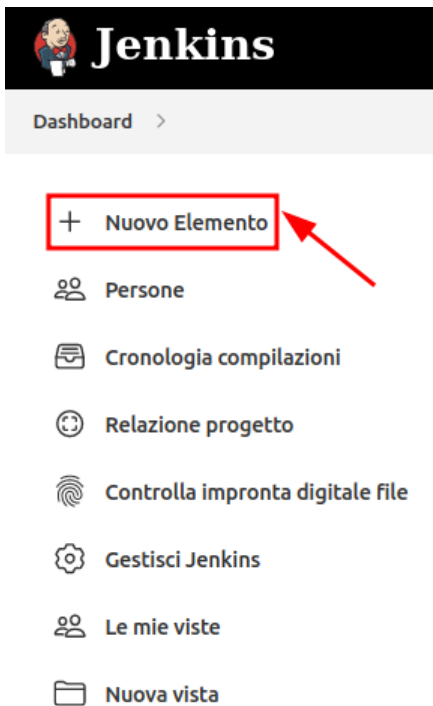
Installa automaticamente ?

Aggiungi git ▾

Salva

Applica

3.1.3 Creare un Elemento che scarica e integra il codice da Github



Successivamente inserire un **nome**, selezionare **Progetto libero** e premere su **OK**:

The screenshot shows the 'Immettere un nome elemento' form. It has a text input field (1) with a red box around it and a red callout bubble with the number 1. Below the input field is the text '» Campo obbligatorio'. To the right of the input field is a red callout bubble with the number 2. Below the input field are two options: 'Progetto libero' (2) and 'Maven project'. The 'Progetto libero' option is highlighted with a red box and has a red callout bubble with the number 2. Below these options is a section titled 'Se si desidera creare un elemento da un altro esistente, è possibile utilizzare quest'opzione:'. This section contains a 'Copia da' option with a red callout bubble with the number 3. At the bottom of the form is an 'OK' button (3) with a red box around it and a red callout bubble with the number 3.

Su **Sistema di gestione del codice sorgente** mettere la spunta su **Git** e inserire l'URL del repository. Lasciare le credenziali su "**Nessuno**" nel caso in cui il repository fosse pubblico.

Sistema di gestione del codice sorgente

☐ Nessuno

☒ **Git** ?

Depositi ?

URL di Deposito ? ✕

https://github.com/pierpaologumina/hello-world.git

Credenziali ?

- Nessuno -

+ Aggiungi

Avanzate...

Add Repository

Inserire il **Ramo**, e premere quindi su **Applica** e poi **Salva**:

Rami a costruire ?

Ramo (lasciare in bianco per alcune) ? ✕

*/main

Add Branch

3.1.4 Integrare Maven con Jenkins

Per integrare Maven sul Server Jenkins bisogna:

1. Installare Maven sul server Jenkins;

2. Installare le variabili di ambiente. Questo perché Maven sfrutta Java;
3. Installare il plugin Maven su Jenkins;
4. Configurare Maven e Java su Jenkins.

Installare Maven sul server Jenkins:

Per installare Maven sul server Jenkins digitare:

```
$ cd /opt
$ wget
https://dlcdn.apache.org/maven/maven-3/3.8.6/binaries/apache-maven-3.8.6-bin.tar.gz
```

Estraiamo il contenuto con:

```
$ tar -xvzf apache-maven-<versione>-bin.tar.gz
```

Per semplicità rinominiamo la cartella:

```
$ mv apache-maven-<versione> maven
$ cd maven/bin
$ ./mvn -v
```

Installare le variabili di ambiente

```
$ cd ~
$ vi .bash_profile
```

Aggiungiamo il path di Java e Maven:

M2_HOME=/opt/maven

M2=/opt/maven/bin

JAVA_HOME=/usr/lib/jvm/java-.. //per verificarlo digitare: **find / -name java-11***

Infine modificare la riga:

PATH=\$PATH:\$HOME/bin

in

PATH=\$PATH:\$HOME/bin:\$JAVA_HOME:\$M2_HOME:\$M2

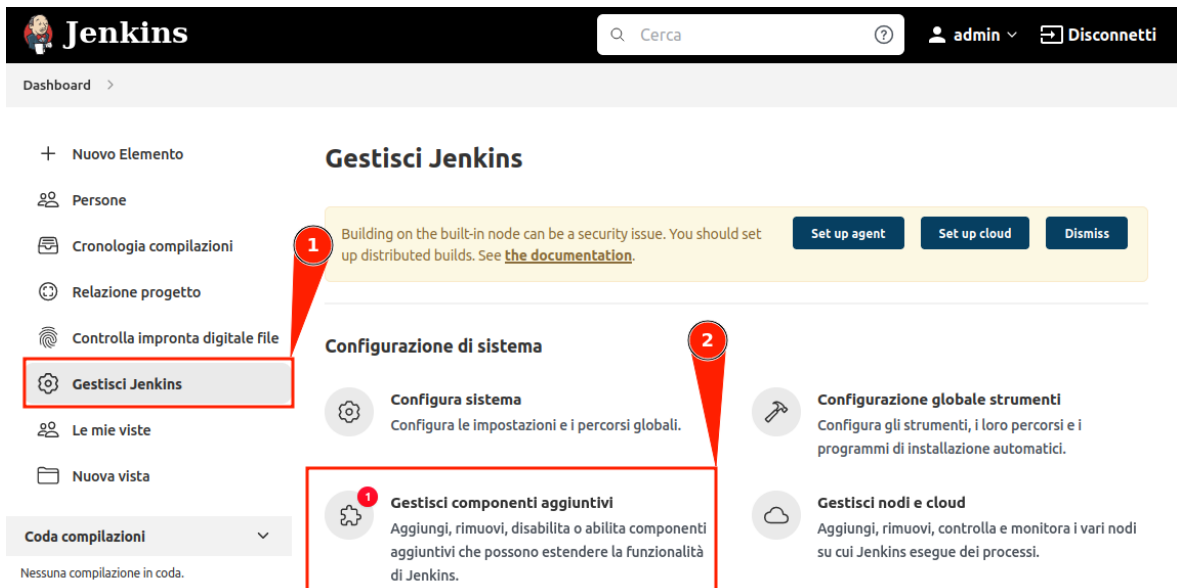
digitare per convalidare:

```
$ source .bash_profile
```

Dobbiamo adesso dire a Jenkins che java e maven sono disponibili nei path appena descritti.

Installare il plugin Maven su Jenkins:

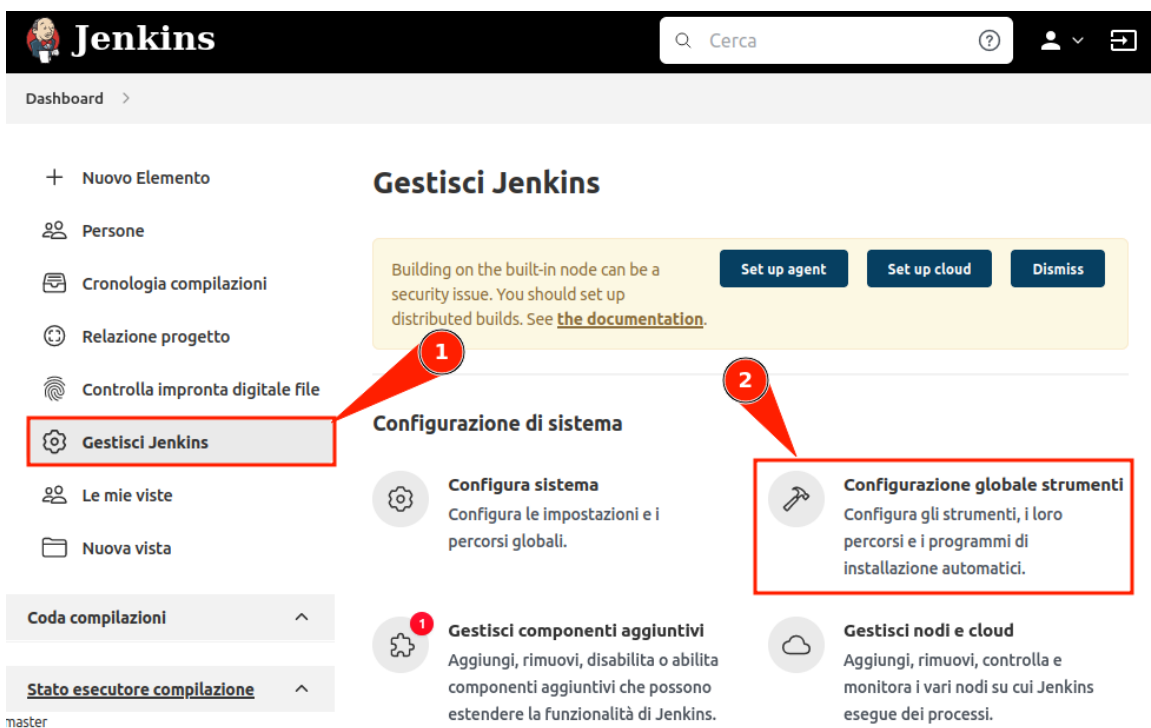
Per installare il Plugin Maven da Jenkins recarsi premere su **Gestisci Jenkins** e successivamente su **Gestisci componenti aggiuntivi**:



Premere su **Disponibili**, digitare nel campo di ricerca "maven", spuntare la casella del plugin chiamato "Maven Integration" e infine premere su **Install without restart**.

Configurare Maven e Java su Jenkins:

Per configurare Maven con Jenkins recarsi su **Gestisci Jenkins** → **Configurazione globale strumenti**:



Modificare i seguenti campi:

JDK

Installazioni di JDK

Elenco di installazioni di JDK su questo sistema

Aggiungi JDK

JDK

Nome

java-11

JAVA_HOME

/usr/lib/jvm/java-11-openjdk-11.0.13.0.8-1.amzn2.0.3.x86_64

 /usr/lib/jvm/java-11-openjdk-11.0.13.0.8-1.amzn2.0.3.x86_64 non sembra essere una directory JDK

☐ Installa automaticamente ?

Aggiungi JDK

Maven

Installazioni di Maven

Elenco di installazioni di Maven su questo sistema

Aggiungi Maven

Maven

Nome

maven-3.8.6

MAVEN_HOME

/opt/maven

☐ Installa automaticamente ?

Aggiungi Maven

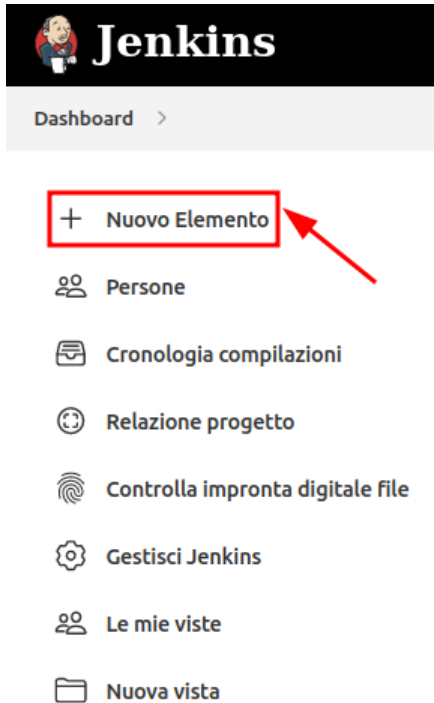
Salva

Applica

3.1.5 Compilare un progetto Java usando Jenkins

Nel paragrafo precedente abbiamo integrato Maven in Jenkins, adesso procediamo a creare il processo di compilazione.


Per farlo dirigersi nel menu **Nuovo Elemento**:



Immettere un **Nome**, selezionare **Maven Project** e premere su **OK**:


Immettere un nome elemento

» Questo campo non può essere vuoto, immettere un nome valido

 **Progetto libero**
Questa è la funzionalità principale di Jenkins. Jenkins eseguirà la compilazione del progetto, combinando qualunque sistema di gestione del codice sorgente con qualunque sistema di compilazione, e questo progetto può essere anche utilizzato per compiti diversi dalla compilazione di software.

 **Maven project**
Effettua una build di un progetto maven. Jenkins sfrutta i file POM e riduce drasticamente la configurazione.

Se si desidera creare un elemento da un altro esistente, è possibile utilizzare quest'opzione:

 Copia da

Integrare la parte di **Git** come visto precedentemente. Nell parte di **Build** specificare il file **pom.xml** e il **ciclo di vita della build**.

In questo progetto sono stati inseriti:

- clean: rimuove tutti i file generati dalla build precedente;
- install: installa il pacchetto nel repository locale, per utilizzarlo come dipendenza in altri progetti locali.

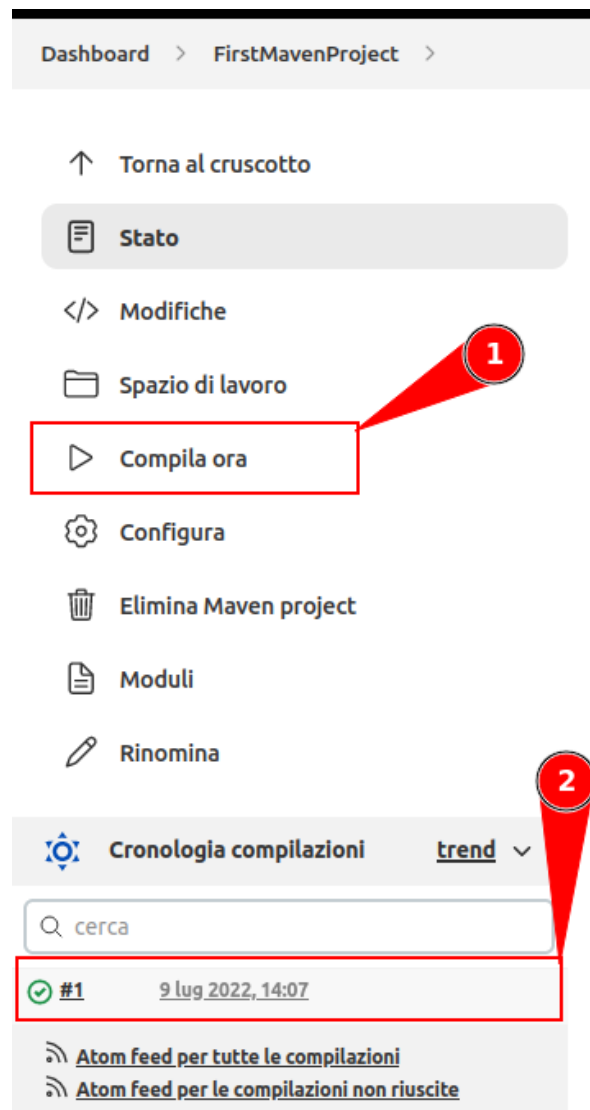
Per ulteriori dettagli sui cicli di vita di una build consultare [12].

Build

Root POM ?

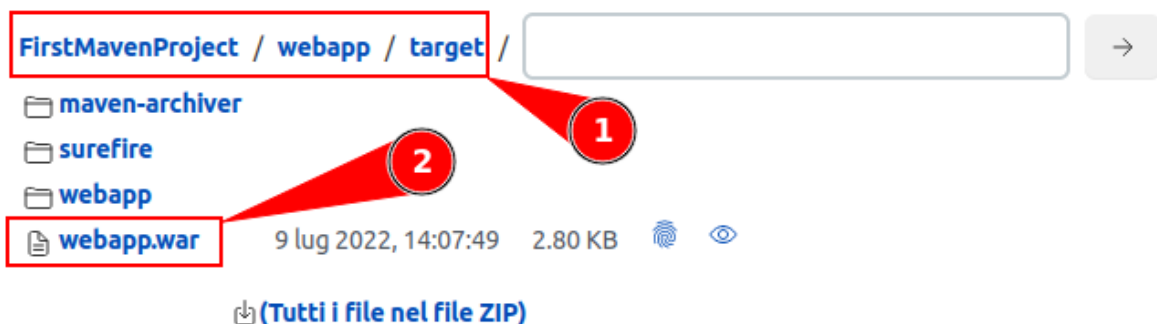
Goals and options ?

Per convalidare premere su **Applica** e **Salva**. Per avviare questo il task appena creato premere su **Compila Ora**; l'esecuzione potrà essere visionata in basso:

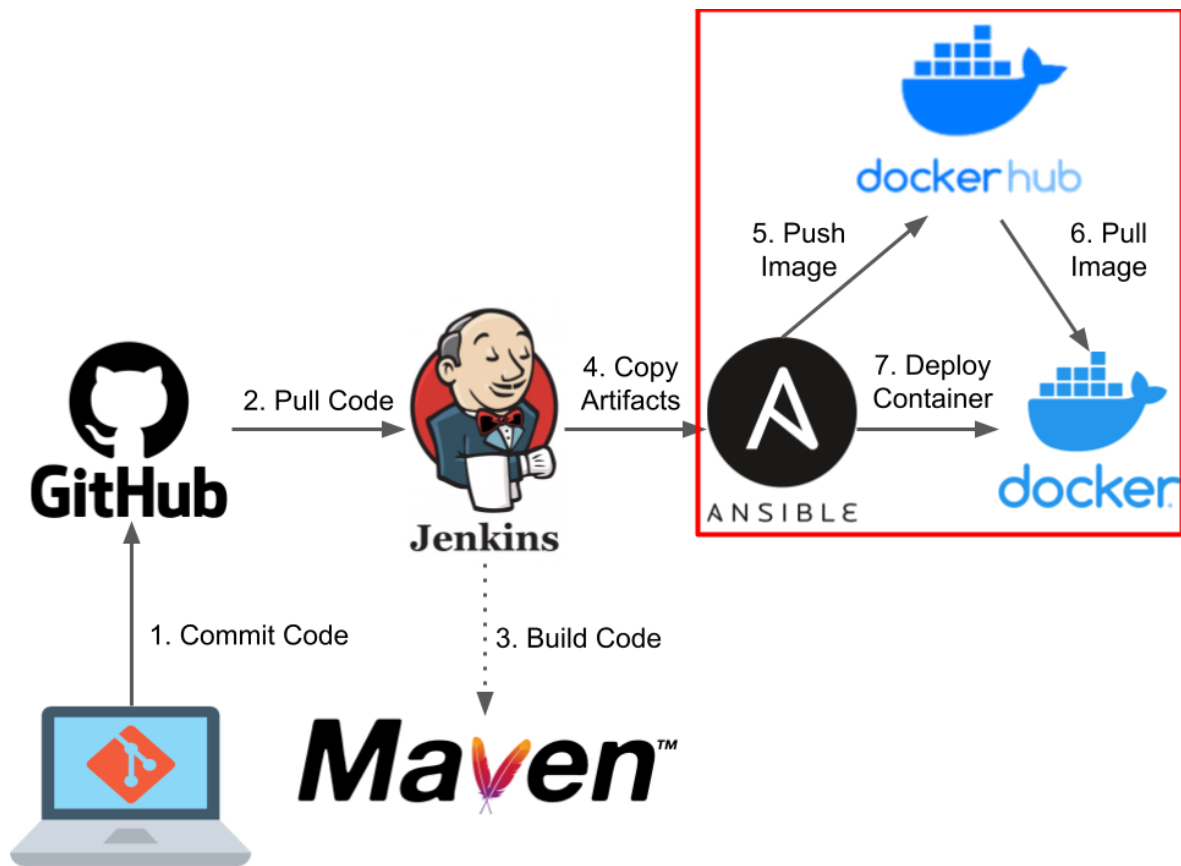


Una volta eseguito il task, Maven produrrà un artefatto, quest'ultimo sarà memorizzato in:

Spazio di lavoro di FirstMavenProject su master



3.2 CI/CD pipeline: Ansible, Dockerhub e Docker



Procedendo nella pipeline CI/CD andremo a creare due istanze: una adibita ad ospitare il server Ansible e un'altra per la containerizzazione dell'applicazione Java. Il server Ansible ha lo scopo di prendere l'artefatto prodotto da Maven, creare un'immagine Docker, pubblicarla nel docker hub e infine eseguire il dockerfile necessario per l'esecuzione del container.

3.2.1 Installazione di Ansible

Per la creazione di un server Ansible bisogna:

1. Creare un'istanza EC2 per il nodo Ansible;
2. Cambiare l'hostname;
3. Creare un nuovo utente chiamato `ansadmin`;
4. Aggiungere l'utente `ansadmin` nel file `sudoers`, per dargli i privilegi di amministratore;
5. Abilitare il login basato su password;
6. Generare chiavi ssh per l'utente `ansadmin`, per usare un'autenticazione basata sulle chiavi ssh;
7. Installare Ansible.

Creare un'istanza EC2 per il nodo Ansible:

I passaggi sono gli stessi visti nel paragrafo precedente con l'unica accortezza di non creare un nuovo Gruppo di Sicurezza bensì selezionare quello creato precedentemente. Accedere quindi all'istanza tramite ssh.

Cambiare l'hostname:

Rinominare l'hostname in ansible-server, mediante il comando:

```
$ vi /etc/hostname  
$ init 6
```

Creare un nuovo utente chiamato ansadmin:

Per creare un nuovo utente ed assegnare una password eseguire:

```
$ sudo su -  
$ useradd ansadmin  
$ passwd ansadmin
```

Aggiungere l'utente ansadmin nel file sudoers:

Il file Sudoers è proprio come qualsiasi altro file sul sistema operativo Linux. Ma svolge un ruolo fondamentale nella gestione di ciò che un “Utente” o “Utenti in un gruppo” può fare sul sistema. Inseriamo l'utente ansadmin:

```
$ visudo
```

In fondo inserire la riga:

```
ansadmin    ALL=(ALL)        NOPASSWD: ALL
```

Il significato della riga sopra è che l'utente ansadmin può eseguire qualsiasi comando senza alcuna password.

Abilitare il login basato su password:

Eseguire i seguenti comandi:

```
$ vi /etc/ssh/sshd_config
```

Modificare le seguenti righe:

```
# To disable tunneled clear text passwords, change to no here!  
PasswordAuthentication yes  
#PermitEmptyPasswords no  
#PasswordAuthentication no
```

Abilitiamo le modifiche con il comando:

```
$ service sshd reload
```

Generare chiavi ssh per l'utente ansadmin:

Fare login nel server Ansible con l'utente appena creato, per esempio:

```
$ ssh -i "Jenkins_Project.pem"
ansadmin@ec2-15-160-159-238.eu-south-1.compute.amazonaws.co
m
```

Procedere con la generazione delle chiavi tramite il comando:

```
$ ssh-keygen
```

Premere "Invio" durante le 3 richieste.

L'obiettivo è quello di copiare la chiave pubblica appena creata nel nodo target così da poterci accedere tramite ssh.

Installare Ansible:

Per installare linux eseguire i seguenti comandi. Nota: l'installazione di ansible prevede l'installazione automatica di python e delle sue librerie.

```
$ sudo su -a
$ amazon-linux-extras install ansible2
```

Infine, per verificare la corretta installazione, eseguire:

```
$ python --version
$ ansible --version
```

3.2.2 Integrare Docker con Ansible

Per fare in modo che Ansible possa gestire il server **Docker** è necessario:

1. Creare un'istanza EC2 per il nodo Docker;
2. Installare docker;
3. Cambiare l'hostname;
4. Creare un utente ansadmin nel nodo Ansible;
5. Aggiungere l'utente ansadmin nel file sudoers;
6. Abilitare il login mediante password.

Nel nodo **Ansible** bisogna invece:

1. Aggiungere al file hosts il nodo docker;
2. Copiare le chiavi ssh del nodo ansible nel nodo target;
3. Testare la connessione tra i due nodi.

Creare un'istanza EC2 per il nodo Docker:

I passaggi sono gli stessi visti nel paragrafo precedente con l'unica accortezza di non creare un nuovo Gruppo di Sicurezza bensì selezionare quello creato precedentemente. Accedere quindi all'istanza tramite ssh.

Installare docker:

```
$ sudo su -  
$ yum install docker -y
```

Terminata l'installazione avviamo il servizio con il comando:

```
$ service docker start
```

Cambiare l'hostname:

Rinominare l'hostname in dockerhost, mediante il comando:

```
$ vi /etc/hostname  
$ init 6
```

Creare un utente ansadmin nel nodo Ansible:

Per creare l'utente *ansadmin* nel nodo Ansible digitare i seguenti comandi:

```
$ useradd ansadmin  
$ passwd ansadmin
```

Aggiungere l'utente ansadmin nel file sudoers:

```
$ visudo
```

In fondo inserire la riga:

```
ansadmin    ALL=(ALL)        NOPASSWD: ALL
```

Il significato della riga sopra è che l'utente *ansadmin* può eseguire qualsiasi comando senza alcuna password.

Abilitare il login mediante password:

Eseguire i seguenti comandi:

```
$ vi /etc/ssh/sshd_config
```

Modificare le seguenti righe:

```
# To disable tunneled clear text passwords, change to no here!  
PasswordAuthentication yes  
#PermitEmptyPasswords no  
#PasswordAuthentication no
```

Abilitiamo le modifiche con il comando:

```
$ service sshd reload
```

Adesso accediamo al nodo Ansible.

Aggiungere al file hosts il nodo docker:

Bisogna modificare l'inventory host del nodo Ansible inserendo l'indirizzo del server Docker:

```
$ vi /etc/ansible/hosts
```

Cancellare tutte le righe di default e inserire l'indirizzo IP privato del nodo Docker.

Copiare le chiavi ssh del nodo ansible nel nodo target:

```
$ sudo su - ansadmin  
$ cd ~  
$ cd .ssh
```

All'interno della cartella vi sono due file:

- `id_rsa`: contiene la chiave privata. NON deve essere condivisa con nessuno;
- `id_rsa.pub`: contiene la chiave pubblica.

Quest'ultima dovrà essere copiata nel nodo *target*, tramite il comando:

```
$ ssh-copy-id <indirizzo_ip_privato_nodoTarget>
```

Testare la connessione tra i due nodi:

```
$ ansible all -m ping
```

3.2.3 Integrare Ansible con Jenkins

In questo paragrafo vedremo come integrare Ansible con Jenkins. In particolare Jenkins copierà l'artefatto `.war`, i `playbook-ansible` e il `docker file` nel server Ansible per poter fare il `deploy` del container nel server Docker.

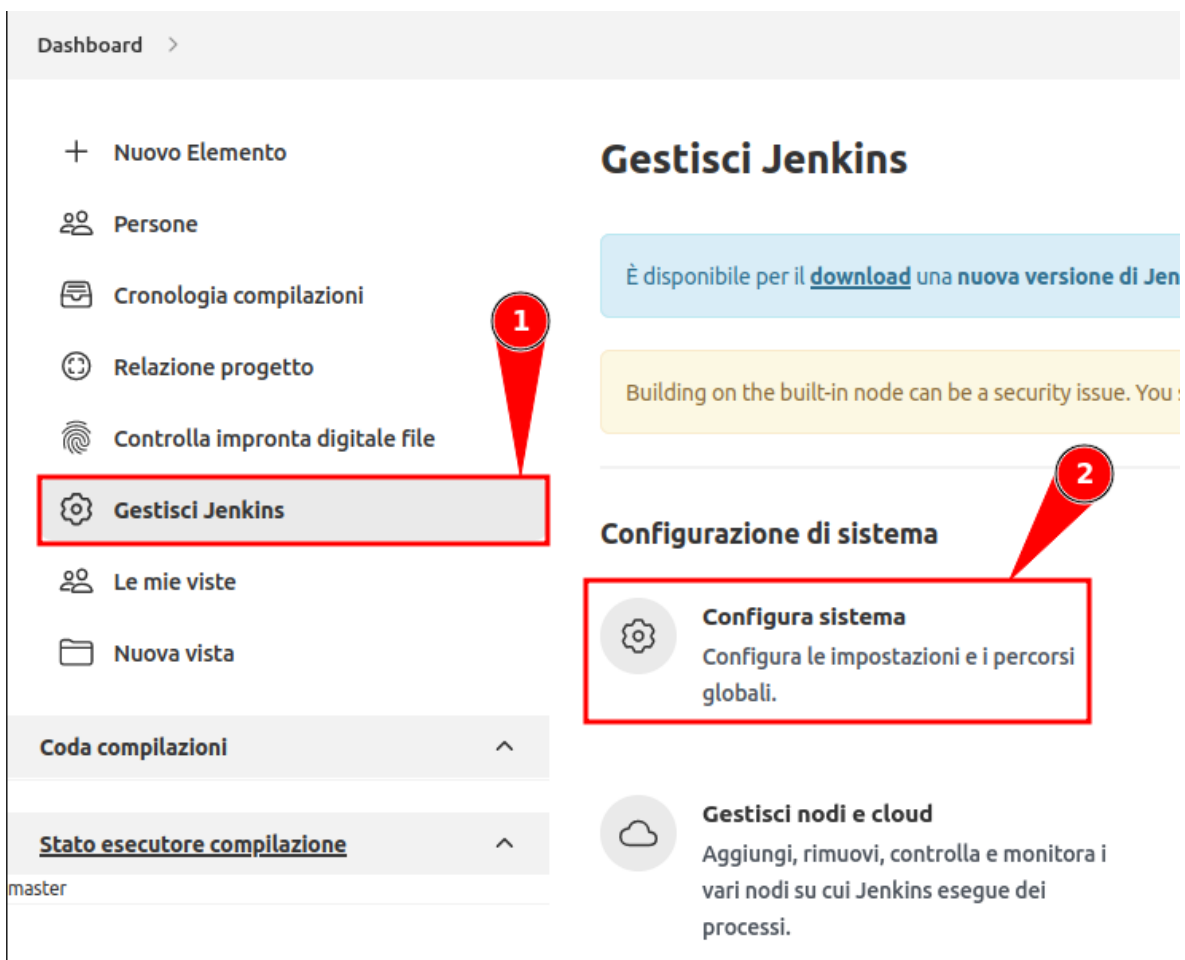
Per integrare Ansible in Jenkins abbiamo bisogno di installare il plugin "Publish Over SSH" seguendo gli stessi passaggi descritti precedentemente per Git.

Una volta completata l'installazione del plugin si procede ad inserire le credenziali nelle impostazioni di Jenkins.

Prima però creiamo una cartella sul server ansible dove memorizzare i vari file provenienti dal server jenkins:

```
$ cd /opt
$ sudo mkdir docker
$ sudo chown ansadmin:ansadmin docker
```

Adesso procediamo a configurare Jenkins:



Compilare e seguenti campi:

The image shows a configuration window titled "SSH Server" with a close button in the top right corner. The form contains several fields and a checkbox, each highlighted with a red callout bubble containing a number:

- 1**: Points to the "Name" field, which contains "ansible-server". Below the field is a red error message: "Required. Cannot contain < & ' " \".
- 2**: Points to the "Hostname" field, which contains "123.456.678.901". Below the field is a red error message: "Richiesto".
- 3**: Points to the "Username" field, which contains "ansadmin". Below the field is a red error message: "Richiesto".
- 4**: Points to the "Remote Directory" field, which is empty.
- 5**: Points to the "Passphrase / Password" field, which contains ".....".
- 6**: Points to the "Applica" button.
- 7**: Points to the "Salva" button.

There is also a checkbox labeled "Use password authentication, or use a different key" which is checked. A red callout bubble with the number "4" points to this checkbox.

Procediamo modificando il task jenkins precedente creato inserendo i file da mandare al server Ansible:

Post Steps

- ☐ Run only if build succeeds
- ☐ Run only if build succeeds or is unstable
- ☒ Run regardless of build result

Should the post-build steps run only for successful builds, etc.

 **Send files or execute commands over SSH** 



SSH Publishers

SSH Server

Name 

ansible-server 

Avanzate...

Transfers

Transfer Set

Source files 

webapp/target/*.war

Remove prefix ?

webapp/target

Remote directory ?

//opt//docker

Exec command ?

```
ansible-playbook /opt/docker/regapp.yml;
sleep 10;
ansible-playbook /opt/docker/deploy_regapp.yml
```

All of the transfer fields (except for Exec timeout) support substitution of **Jenkins environment variables**

Avanzate...

Add Transfer Set

Add Server

Avanzate...

Inseriamo un'altra azione, questa volta per inviare i due playbook e il dockerfile. Premere quindi **Applica** e **Salva**.

Azioni di post-compilazione

Send build artifacts over SSH ?

SSH Publishers

SSH Server Name ?

ansible-server

Avanzate...

Transfers

Transfer Set

Source files ?

Dockerfile,deploy_regapp.yml,regapp.yml

Remove prefix ?

Remote directory ?

//opt//docker

Salva

Applica

3.2.4 Creare un'immagine e un container sul nodo Ansible

Ottenuto l'artefatto dobbiamo creare un'immagine docker a partire da esso. L'immagine docker viene creata nel nodo Ansible e successivamente pubblicata nel docker hub. Il server Docker farà il pull dell'immagine memorizzata nel docker hub creando quindi un container.

Accediamo sul nodo ansible con l'utente ansadmin e procediamo ad installare docker:

```
$ sudo yum install docker -y
```

Aggiungiamo ansadmin al docker group, altrimenti non potremmo eseguire i comandi docker con l'utente ansadmin:

```
$ sudo usermod -aG docker ansadmin
```

Avviamo il servizio demone docker:

```
$ sudo service docker start
```

Per creare l'immagine docker utilizziamo il dockerfile inviato da jenkins e memorizzato su github. Il dockerfile si compone delle seguenti righe:

<i>/opt/docker/Dockerfile</i>
FROM tomcat:latest RUN cp -R /usr/local/tomcat/webapps.dist/* /usr/local/tomcat/webapps COPY ./*.war /usr/local/tomcat/webapps

In caso di errore sull'avvio del docker file o del playbook ansible digitare, sia sul server docker che ansible, il comando:

```
$ sudo chmod 777 /var/run/docker.sock
```

3.2.5 Editare un playbook ansible per creare un'immagine ed un container

Modificare il file host di ansible inserendo due ip:

```
$ sudo vi /etc/ansible/hosts
```

```
[dockerhost]
<indirizzo_ip_privato_dockerhost>

[ansible]
<indirizzo_ip_privato_ansibleserver>
```

Per esempio:

```
[dockerhost]
172.31.25.50

[ansible]
172.31.28.22
```

Per creare un'immagine docker e farne il push in docker hub si utilizza il playbook denominato *regapp.yml*. Quest'ultimo si compone delle seguenti righe:

```

---
- hosts: ansible

tasks:
  - name: create docker image
    command: docker build -t regapp:latest .
    args:
      chdir: /opt/docker

  - name: create tag to push image onto dockerhub
    command: docker tag regapp:latest
pierpaolodev/regapp:latest

  - name: push docker image
    command: docker push pierpaolodev/regapp:latest

```

Prima di eseguire il playbook qui sopra è necessario immettere le credenziali di docker. Usare il comando:

```
$ docker login
```

3.2.6 Creare un container nel Docker server utilizzando un Playbook Ansible

Nella fase precedente abbiamo creato, nel server Ansible, un'immagine di container a partire da un artefatto e caricata nel docker hub.

Lo step successivo consiste nel deploy dell'applicazione in un container. Il server Ansible manda al Docker host le istruzioni necessarie per scaricare l'immagine docker dal docker hub e avviare il container.

Per far ciò Ansible utilizza un altro playbook, denominato *deploy_regapp.yml*:

```

---
- hosts: dockerhost

tasks:
  - name: stop existing container
    command: docker stop regapp-server
    ignore_errors: yes

```

```
- name: remove the container
  command: docker rm regapp-server
  ignore_errors: yes

- name: remove image
  command: docker rmi pierpaolodev/regapp:latest
  ignore_errors: yes

- name: create container
  command: docker run -d --name regapp-server -p
8082:8080 pierpaolodev/regapp:latest
```

Infine, Per automatizzare tutta la pipeline CI/CD modificare il task Jenkins inserendo il **Trigger di compilazione**:

Trigger di compilazione

☒ Build whenever a SNAPSHOT dependency is built ? 1

☐ Schedule build when some upstream has no successful builds ?

☐ Scatena compilazioni da remoto (ad esempio da uno script) ?

☐ Esegui la compilazione dopo aver compilato gli altri progetti ?

☐ Esegui compilazione periodicamente ?

☐ GitHub hook trigger for GITScm polling ?

☒ Esegui polling del sistema di gestione del codice sorgente ? 2

Pianificazione ? 3

⚠ Si intende veramente "ogni minuto" quando si dice "***"? Forse si intendeva "H*****" per eseguire il polling una volta all'ora**

Ultima esecuzione teorica pianificata il Saturday, July 16, 2022 at 9:55:05 AM Coordinated Universal Time;
prossima esecuzione pianificata il Saturday, July 16, 2022 at 9:55:05 AM Coordinated Universal Time.

☐ Ignora hook post commit ?

4. Conclusioni e possibili sviluppi

Obiettivo di questo progetto era creare una completa pipeline CI/CD ovvero una metodologia per la distribuzione continua di prodotti e/o servizi mediante l'utilizzo di processi automatizzati e basata sui concetti di integrazione, distribuzione e deployment.



Approcciarsi alla metodologia CI/CD consente agli sviluppatori di superare i normali problemi di integrazione del codice che avvengono tipicamente negli scenari in cui lo sviluppo procede per vie parallele, e consente di distribuirne il risultato automaticamente.

Grazie ad essa è possibile concentrare le risorse disponibili principalmente sullo sviluppo del codice accelerando l'introduzione di nuove funzionalità e rendendo i cicli di rilascio del software più celeri e soprattutto automatizzati.

In definitiva possiamo confermare quanto la metodologia di CI/CD sia uno degli strumenti di maggiore successo ed utilità nella produzione e gestione di prodotti e servizi digitali.

Avere processi automatizzati significa essere più consapevoli di cosa si produce e di cosa si distribuisce e permette di sviluppare più agevolmente nuovi modelli di business. Modelli che godranno di maggiori risorse in quanto lo sviluppatore, non dovendo più concentrarsi anche sulle procedure di rilascio, avrà a disposizione più tempo per concentrarsi sullo sviluppo.

Da un lato avremo quindi un sensibile miglioramento della qualità del processo di rilascio, che diventerà più solido, coerente e sicuro, e dall'altro un'ottimizzazione dei tempi (e quindi delle risorse) necessarie a produrre e gestire i nostri software.

Per quanto riguarda il progetto, una possibile implementazione futura potrebbe essere quella di inserire un server Kubernetes per l'orchestrazione dei container. Il problema attualmente è il seguente: non appena vi sono dei cambiamenti nel codice la procedura sviluppata termina il container in esecuzione creandone uno nuovo. In questo lasso di tempo gli utenti non possono accedere all'applicazione. Questo problema potrebbe essere ovviato inserendo un server Kubernetes che non appena registra la terminazione di un container ne riesegue uno nuovo.

Bibliografia

1. Cos'è l'integrazione continua? -
<https://aws.amazon.com/it/devops/continuous-integration/>
2. Cos'è la distribuzione continua? -
<https://aws.amazon.com/it/devops/continuous-delivery/>
3. Introduzione a Git (ed al controllo di versione distribuito) -
https://www.mrw.it/programmazione/introduzione-git_12465.html
4. Cosa è GitHub? Introduzione a GitHub per Principianti -
<https://kinsta.com/it/knowledgebase/cosa-e-github/>
5. Tutto quello che devi sapere sull'integrazione continua con Jenkins -
<https://it.ichlese.at/all-you-need-know-about-continuous-integration-with-jenkins>
6. Cos'è Maven e come si usa - <https://www.nextre.it/cose-maven-si-usa/>
7. Cos'è Ansible? -
<https://www.redhat.com/it/technologies/management/ansible/what-is-ansible>
8. Cos'è Docker? - <https://aws.amazon.com/it/docker/>
9. Installazione di Jenkins - <https://pkg.jenkins.io/redhat-stable/>
10. Installazione Apache Maven - <https://maven.apache.org/install.html>
11. Apache Maven binary -
<https://dlcdn.apache.org/maven/maven-3/3.8.6/binaries/apache-maven-3.8.6-bin.tar.gz>
12. Maven: cicli di vita di una build -
https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html#Lifecycle_Reference
13. Progetto - <https://github.com/pierpaologumina/hello-world>
14. Progetto Maven - <https://github.com/jleetutorial/maven-project>