Documentazione OSCAT

Indice

Introduzione

Il sistema OSCAT è la piattaforma di Regione Toscana rivolta a Regione Toscana e agli altri enti aderenti al fine di:

1. raccogliere il frutto delle attività di programmazione e documentazione;
2. costruire e rilasciare gli artefatti prodotti attraverso i sorgenti depositati.

La presente guida fornisce informazioni di base per l’utilizzo della piattaforma OSCAT rivolte agli sviluppatori di applicazioni, siano essi dipendenti dell'Ente, collaboratori esterni o fornitori.

Premesse

Seguono i principali principi ispiratori che governano la nuova architettura OSCAT:

1. **Principio 1**: L’artefatto dispiegato in produzione sia generabile attraverso la compilazione del codice depositato sul branch master.
2. **Principio 2**: L’artefatto dispiegato in certificazione e quello in produzione devono essere identici se contengono la stessa versione del codice.
3. **Principio 3**: Il codice degli artefatti dispiegati in produzione sia nella disponibilità dell’ente (e quindi depositati nella piattaforma) e che, in ogni momento (e senza ricorrere a reverse engineering), questo sia in grado di sapere quale effettivamente sia.
4. **Principio 4**: Affinché un artefatto sia dispiegabile in modo automatico è necessario che il codice che lo ha generato rispetti una soglia minima di qualità (detta anche quality gate). Sarà cura dell’ente dare chiara comunicazione di come siano caratterizzate queste soglie che, ad esempio, potrebbero richiedere che il codice non presenti segnalazioni considerate bloccanti/critiche.

Linguaggi e repository supportati

Linguaggi

I linguaggi attualmente supportati dalla piattaforma OSCAT sono:

- JAVA
- JavaScript
- PHP
- Angular

Per necessità diverse da quelle elencate sopra si prega di contattare il supporto indicando le specifiche esigenze al fine di valutare il da farsi.

Repository

Regione Toscana usa Nexus come strumento che funge da software repository cioè da punto di deposito dei pacchetti di codice (detti packages) e in grado di riferire molteplici ulteriori repository.

Nella tabella che segue, i repository considerati trusted da RT che costituiscono dunque gli **unici repository** considerati affidabili e che si possono utilizzare. Rimane inteso che RT che è comunque libera di aggiungere librerie a fronte di motivate richieste da parte degli utenti o toglierne in caso queste costituiscano un rischio per la sicurezza dell'ente.

Linguaggio	**Repository**
Java	Maven Central
Node.js	registry.npmjs.org

Architettura generale

L'architettura OSCAT si compone delle seguenti macro-componenti:

- **GitLab** che ospita sorgenti e documentazione dei progetti e consente di eseguire le pipeline di costruzione e rilascio degli artefatti
- **Modulo Gestione Progetto** che si occupa di:
  - gestire le fasi che riguardano l'approvazione e la creazione di un progetto
  - pubblicare le news del progetto
- **Nexus** che ospita le librerie software e consente l'utilizzo dei seguenti repository:
  - Maven Central
  - registry.npmjs.org (per Node.js)
- **SonarQube** che si occupa dell'analisi statica della qualità del codice.

Tutte le componenti sono in SSO tra loro.

Attenzione! *La piattaforma oscat.rete.toscana.it sarà disponibile solo per il tempo necessario alla migrazione dei progetti.*

Attori della piattaforma

Gli attori principali della piattaforma sono:

- **Gestore della Piattaforma (GP):** si occupa della gestione dell’intera piattaforma.
- **Gestore degli Ambienti Target (GT):** coinvolto nella fase di dispiegamento sugli ambienti target.
- **Referente Ente (RE):** utilizza la piattaforma per la verifica dei prodotti, collabora nella definizione del processo validando alcune informazioni e confermando le richieste di deploy.
- **Incaricato (IN):** coinvolto nel processo di sviluppo e rilascio del software (può essere un dipendete dell'ente o un fornitore esterno).

Acronimi e definizioni

Nel resto del documento si fa uso degli acronimi che seguono.

ACRONIMO	DESCRIZIONE
CI	Continuous Integration
CD	Continuous Deployment
FusionForge	Strumento open source per la gestione collaborativa di progetti software
Git	Software di controllo di versione distribuito
GitFlow	Strategia di gestione dei branch Git
GitLab	Piattaforma web open source che permette la gestione di repository Git, funzioni trouble ticket e molte altre funzionalità
Jenkins	Componente a supporto delle attività di CI/CD
Merge Request	Richiesta, tipicamente da parte di uno sviluppatore, di fare il merge del codice da un branch ad un altro
MGP	Modulo Gestione Progetto è il modulo che si occupa di gestire la fase di approvazione iniziale di un progetto e le news della piattaforma
Nexus	Repository (o repo) per pacchetti software
Quality Gate	Insieme delle condizioni di qualità che è necessario verificare al fine di superare il controllo automatico di qualità (l'invito rimane comunque quello di non presentare segnalazioni alcuna)
SCT	Sistema Cloud Toscana (precedentemente detto TIX)
SonarQube	Precedentemente chiamato Sonar, è una piattaforma open source per effettuare continuous inspection della qualità del codice, analisi statiche del codice, individuare potenziali bug e cattive pratiche in molteplici linguaggi di programmazione
SSO	Single Sign-On

Primi passi

Questa sezione descrive i passi iniziali necessari per il rilascio di quanto realizzato per l'ente. Le sezioni successive riportano dunque come:

- accedere alla piattaforma;
- richiedere l'apertura di un progetto;
- approvare un progetto;
- visualizzare il progetto.

Accesso alla piattaforma

Il punto d'accesso al sistema è il portale Gitlab (https://oscat.regione.toscana.it).

L'accesso a tutti i componenti della piattaforma è erogato tramite l'utilizzo dell'Identità Digitale (SPID o CIE) o della CNS, come offerto da ARPA.

In seguito al primo accesso, l'utente è correttamente loggato su Gitlab, dove gli è stato automaticamente creato un account utilizzando i dati condivisi dalla sua Identità Digitale.

Primo accesso

Il primo accesso alla piattaforma tramite ARPA potrebbe richiede l'inserimento dell'indirizzo email, qualora questo non fosse presente tra i dati condivisi dall'Identità Digitale. Contestualmente, si permette la modifica dell'indirizzo email nel caso in cui l'utente volesse utilizzarne uno differente da quello associato all'Identità Digitale.

L'indirizzo email eventualmente modificato è quello associato al solo account Gitlab, non verrà modificato l'indirizzo associato allo SPID o alla CIE.

La modifica dell'indirizzo email su Gitlab permette agli utenti di ricevere le comunicazioni di OSCAT su un indirizzo differente da quello associato allo SPID poiché, tipicamente, quest'ultimo è un indirizzo personale che non si intende utilizzare in ambito lavorativo.

In questa fase è possibile modificare in autonomia l'indirizzo email, previa validazione tramite OTP. In caso in cui sia necessario modificare l'indirizzo email in una fase successiva sarà necessario richiedere al Supporto OSCAT di effettuare la modifica.

Accessi successivi

Gli accessi successi al primo possono essere effettuati tramite ARPA.

Accesso ai diversi sistemi

L'accesso ai diversi sistemi della piattaforma è federato con Gitlab tramite ARPA e permette il Single Sign On.

Apertura di un progetto

L'apertura di un nuovo progetto è erogata tramite l'utilizzo della piattaforma dedicata Modulo Gestione Progetto. Come anticipato, il login sulla piattaforma avverrà tramite una delle modalità offerte da ARPA.

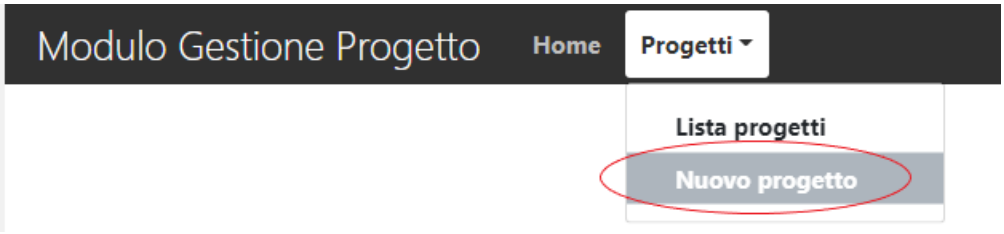
Il processo di apertura di un nuovo progetto passa attraverso due diverse fasi:

1. la Richiesta di apertura da parte dell'incaricato;
2. l'Approvazione del progetto da parte del referente.

Una volta approvati, i progetti sono raggiungibili attraverso un'apposita sezione delModulo Gestione Progetto (che rimanda a GitLab) e direttamente su Gitlab nel gruppo del settore corretto.

Richiesta di apertura

L'incaricato può richiedere l'apertura di un nuovo progetto tramite la sezione dedicata delModulo Gestione Progetto.



Informazioni generali

Ente e referente

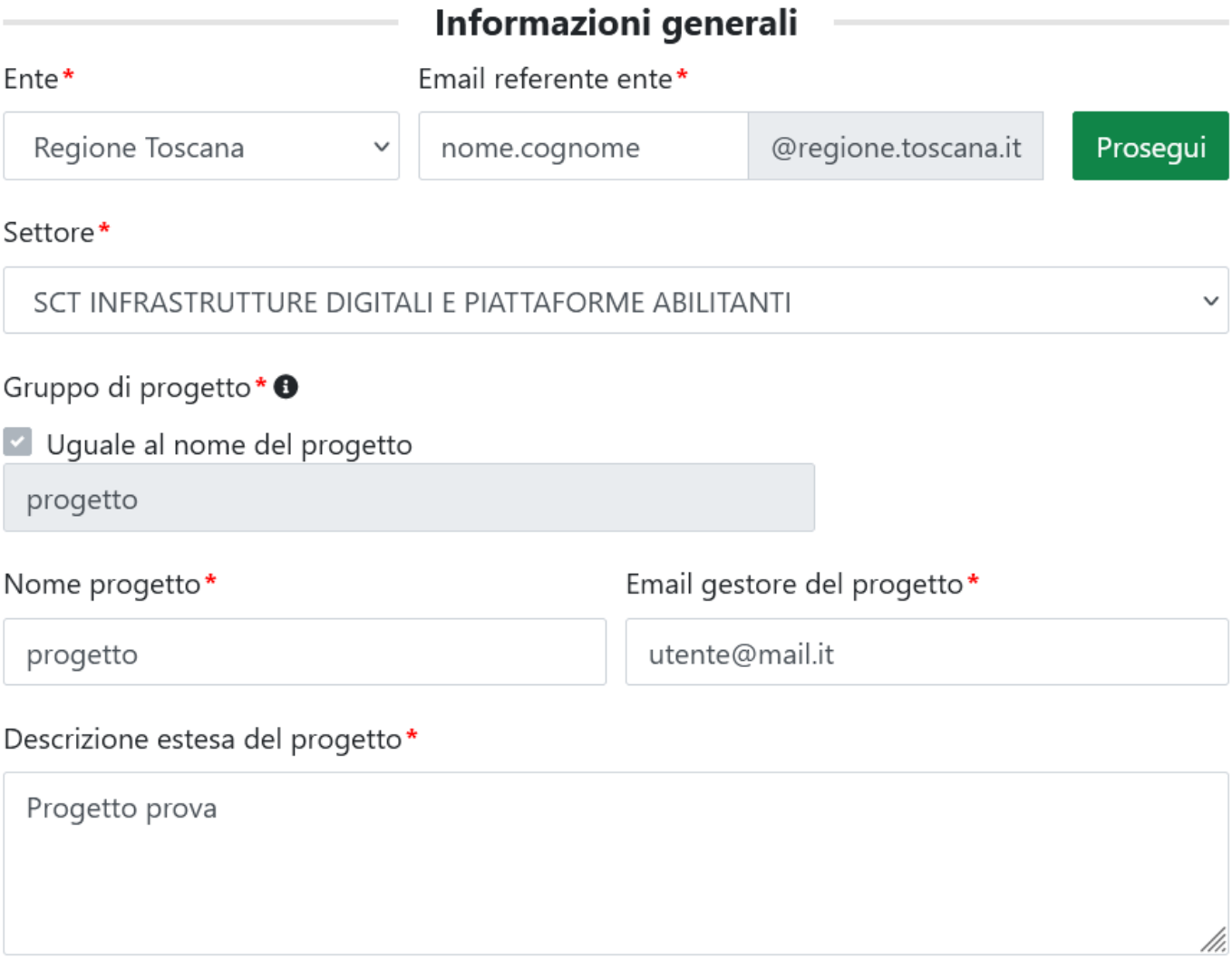
L'incaricato compila la sezione in ogni sua parte, a partire dall'ente. Una volta selezionato l'ente, è necessario inserire il nome del referente per cui viene aperto il progetto. Il campo referente presenta la funzione di auto-completamento per le e-mail dei referenti di Regioni Toscana al momento della compilazione.

Settore

Una volta inserito l'indirizzo e-mail, l'incaricato sceglierà il nome del settore per cui intende sottomettere il nuovo progetto, selezionandolo dalla lista di quelli attualmente associati al referente scelto.

Gruppo, nome e descrizione del progetto

La sezione successiva contiene il gruppo, il nome e la descrizione che saranno poi visualizzati su Gitlab. \ Il gruppo di progetto è un gruppo contenitore di Gitlab, all'interno del quale verrà aggiunto il progetto. Gli applicativi affini, su cui lavora lo stesso gruppo di persone, possono essere aggiunti allo stesso gruppo. \ Il nome del gruppo e il nome del progetto devono essere univoci all'interno del settore. Se l'incaricato non ha necessità di inserire più progetti in un unico gruppo può utilizzare lo stesso nome del progetto come nome del gruppo.\ Per ogni progetto è inoltre richiesta una descrizione esaustiva dello stesso.



Gestore del progetto

Ogni progetto sarà associato ad un utente gestore, che possiederà il compito di gestire l'aggiunta di utenti al progetto e le operazione di amministrazione dello stesso.

Gruppi di lavoro

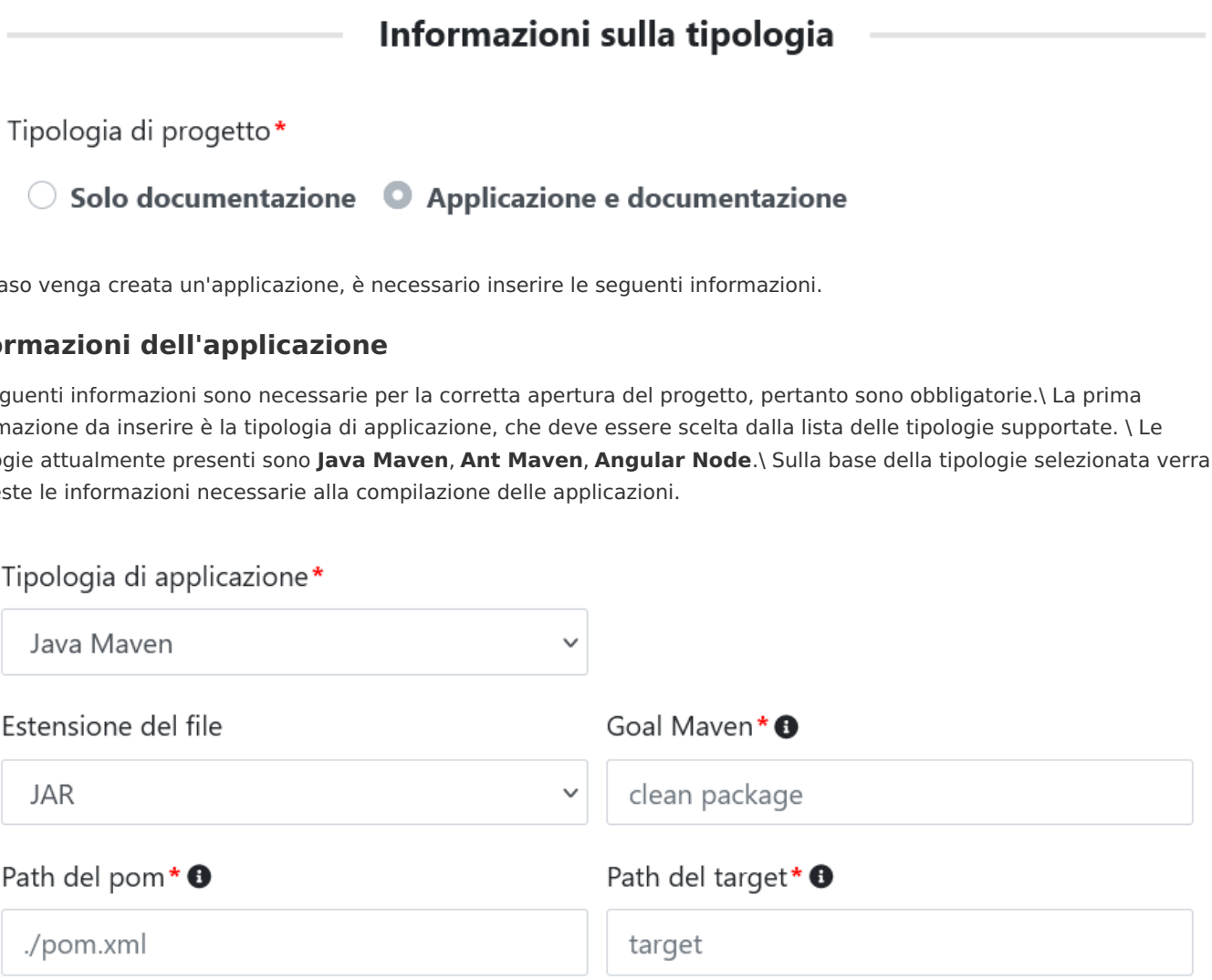
È possibile aggiungere in questa prima fase gli indirizzi e-mail degli utenti che parteciperanno al progetto e gli alias di progetto a cui inviare le comunicazioni riguardanti lo stato del dispiegamento.\ Per ogni indirizzo e-mail verrà effettuata una ricerca su Gitlab e, se l'utente è già presente, egli verrà aggiunto al progetto una volta creato. In caso contrario, verrà comunque aggiunto alle comunicazioni riguardanti il dispiegamento.

Informazioni sulla tipologia

Tipologia di progetto

I progetti creati possono essere di due categorie:

1. sola documentazione; non conterranno quindi codice sorgente.
2. applicazione; conterranno documentazione e codice sorgente, e dove necessario anche la CI/CD.



Nel caso venga creata un'applicazione, è necessario inserire le seguenti informazioni.

Informazioni dell'applicazione

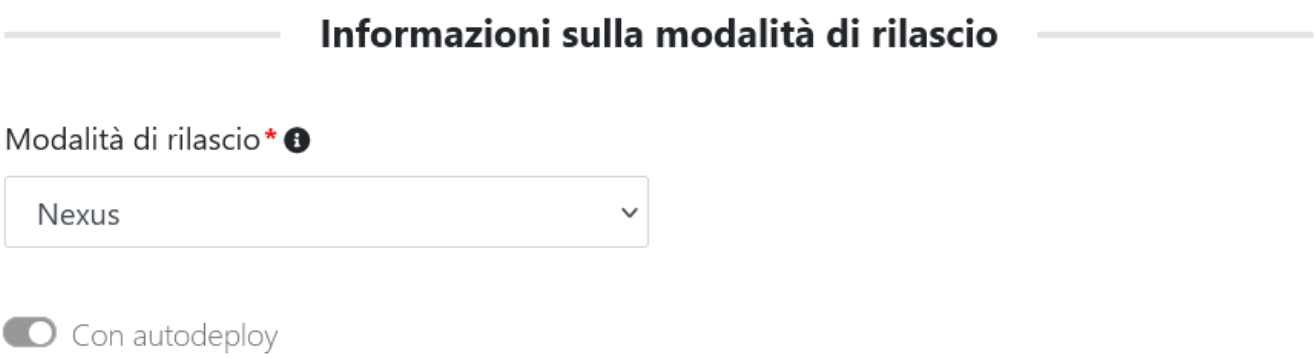
Le seguenti informazioni sono necessarie per la corretta apertura del progetto, pertanto sono obbligatorie.\ La prima informazione da inserire è la tipologia di applicazione, che deve essere scelta dalla lista delle tipologie supportate. \ Le tipologie attualmente presenti sono **Java Maven**, **Ant Maven**, **Angular Node**.\ Sulla base della tipologie selezionata verranno richieste le informazioni necessarie alla compilazione delle applicazioni.

Informazioni sulla modalità di rilascio

Modalità di rilascio

Le possibili modalità di rilascio sono:

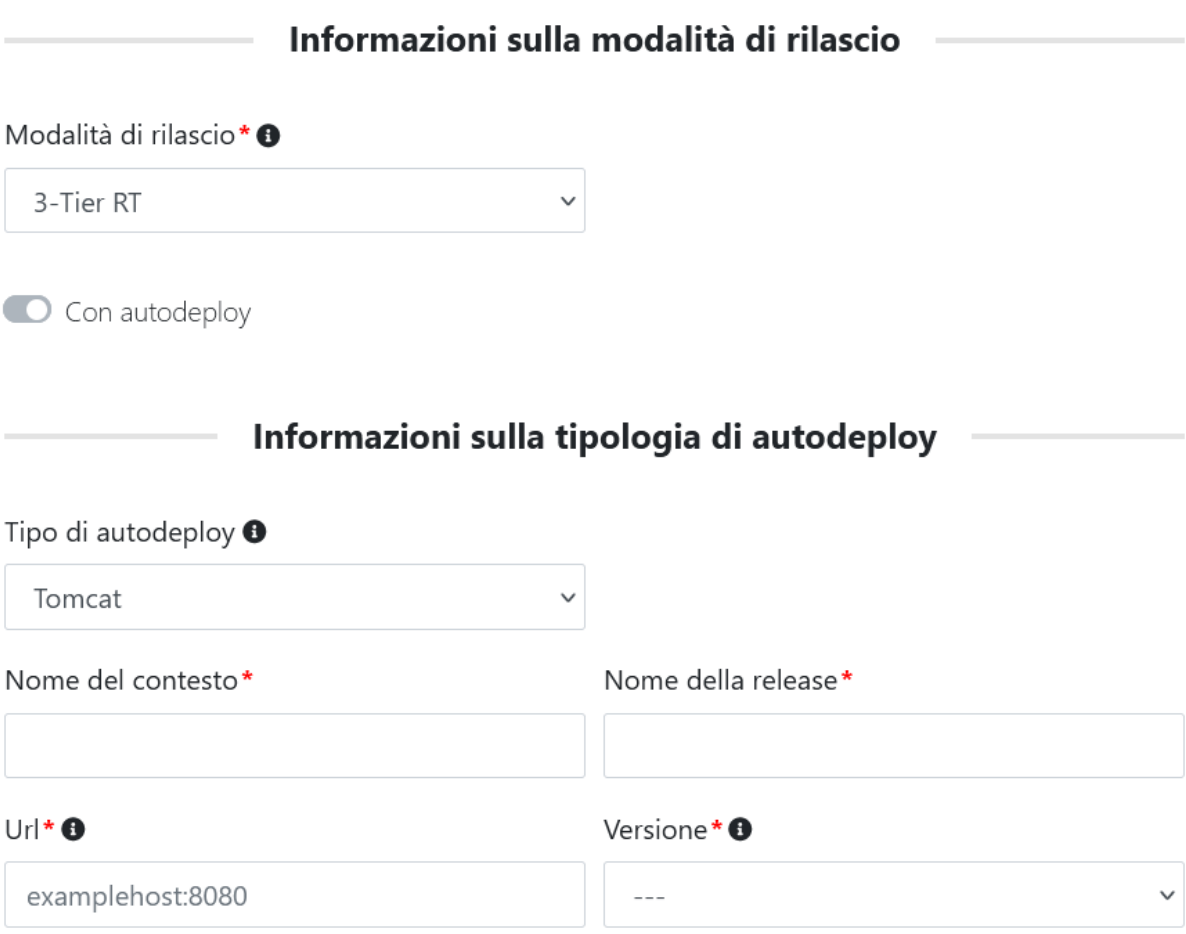
1. Nexus: il codice viene rilasciato come libreria sul Nexus RT e quindi utilizzabile anche da altri progetti.
2. 3-Tier: l'applicazione viene compilata e dispiegata su un application server quali Tomcat o Jboss/wildfly
3. Docker: l'applicazione viene compilata e vengono eseguite build e push dell'immagine sul registry Docker scelto in base al sistema di dispiegamento.
4. Self Managed: la gestione è lasciata all'utente



In base alla modalità scelta verranno richieste alcune informazioni aggiuntive.

Informazioni sulla tipologia di autodeploy

Nella sezione di scelta per la modalità di rilascio è possibile attivare o meno l'opzione autodeploy.\ Per la modalità Nexus e Docker SCT/Kubernetes l'autodeploy è sempre attivo e non sono richieste in questa fase informazioni aggiuntive.\ Invece per 3-Tier e Docker Cochiese/Swarm è possibile scegliere la funzionalità autodeploy già in fase di creazione, ma sono richieste alcune informazioni. E' necessario quindi che l'applicazione sia già stata dispiegata per poter procedere. Altrimenti, sarà sempre possibile per l'utente richiedere l'autodeploy in un secondo momento.



Approvazione dell'apertura

Una volta compilata la form e inviata una mail al referente, quest’ultimo potrà decidere se approvare o rifiutare il progetto accedendo all’applicazione Modulo Gestione Progetto oppure rispondendo alla mail e lasciando quindi all’amministratore dell’applicazione l’onere di accettare o rifiutare.

Attraverso un link contenuto nella mail, il referente potrà accedere all’applicazioneModulo Gestione Progetto e vedere direttamente tutte le informazioni inserite da colui che ha fatto la richiesta di creazione. Attraverso i due bottoni presenti in pagina, potrà decidere se approvare o rifiutare il progetto così come da immagine in calce.

In caso il referente decidesse di rispondere alla e-mail, sarà necessario un intervento dell’amministratore dell’applicazione, che accedendo al pannello amministrativo potrà approvare o rifiutare direttamente i progetti.

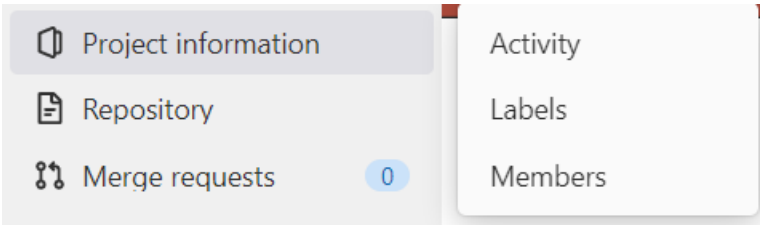
Visualizzazione progetto in Gitlab

Una volta che il progetto è stato approvato e creato, sarà disponibile su Gitlab. In seguito al login su https://oscat.regione.toscana.it, l'utente può visualizzare la lista dei progetti a cui è associato con qualunque ruolo e accedervi.

Modifica membri del progetto

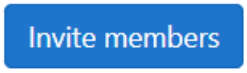
Il *maintainer* (e l'*owner*) del progetto può modificare gli utenti dalla pagina del progetto di Gitlab.

La pagina di progetto possiede un sottomenù dedicato per la gestione degli utenti, raggiungibile dal menù laterale sinistro. La voce è un sottomenù della sezione *Project information*.



In particolare, il *mantainer* può:

1. aggiungere nuovi membri al progetto, tramite l'apposito bottone. In fase di aggiunta è possibile selezionare il ruolo con cui si intendono aggiungere gli utenti al progetto. Per approfondire: https://docs.gitlab.com/ee/user/permissions.html. \

2. Modificare il ruolo posseduto da un utente già aggiunto ad un progetto. \ 

Dal deposito al rilascio di un'applicazione

Il ciclo di vita di un'applicazione si compone di diverse fasi, alcune delle quali saranno eseguite all'interno della piattaforma OSCAT. Nelle sezioni seguenti sono riportate le istruzioni per poter eseguire queste operazioni divise in:

- Realizzazione (dell'applicativo)
- Gestione del flusso di lavoro Git
- Costruzione dell'artefatto (build)
- Modifica della configurazione
- Analisi della qualità del codice
- Rilascio (dell'applicativo)

Realizzazione applicativo

La fase di realizzazione si differenzia in base alla tipologia di progetto creato. Per ogni applicazione si avrà un progetto GitLab in cui sarà possibile usare la sezione *WIKI* per la documentazione e *Repository* per depositare il codice sorgente.

Nel caso di applicazioni che fanno uso della CI/CD, al repository del progetto viene aggiunto un file di configurazione che definisce le operazioni automatiche da eseguire.

Aggiunta della documentazione

Linee guida per il riuso del software

Le *Linee guida per il riuso del software* riportano come "prescrizioni obbligatorie che l'Incaricato è tenuto a rispettare") che:

> La documentazione deve essere scritta in un formato testuale che garantisca il versionamento riga per riga (ad esempio sono ammessi i seguenti formati: HTML, Markdown, reStructuredText, LaTeX). La documentazione in formato ODT, DOCX o PDF non è ammessa [...]. Se nel capitolato è prevista anche la stesura di documentazione sull'utilizzo del software rivolta agli utenti finali («manuale utente» o simile documento), l'obbligo di rilascio si estende anche ad essa. Per tale documentazione sono consentiti anche formati binari, purché aperti, modificabili e multipiattaforma (resta dunque escluso il formato PDF).

Per tale motivo, OSCAT fornisce due approcci alternativi da utilizzare per la produzione di documentazione di nuovi progetti lasciando all'Incaricato, in accordo con l'Ente, la scelta tra:

1. **Progetti standard**, in cui l'Incaricato si limiterà ad utilizzare il Wiki GitLab - fornito con la piattaforma OSCAT - per la scrittura della documentazione e il caricamento dei manuali utente (sempre in formati editabili e aperti).
2. **Progetti multi-documentazione**, in cui l'Incaricato depositerà il codice (versionabile riga per riga) della documentazione in una cartella all'interno dei sorgenti del progetto.

Progetti standard

Con progetti standard intendiamo progetti in cui la documentazione da produrre sia limitata, ad esempio, al manuale di architettura, a quello utente e a un semplice readme per l'installazione. In questi casi, in cui rientra la maggioranza dei progetti attualmente presenti in OSCAT, l'indicazione è di usare le sole funzionalità Wiki che consentono di caricare file e scrivere documentazione in formato Markdown o simili (compatibili con le prescrizioni AgID). Tale approccio, sicuramente veloce e flessibile, potrebbe non soddisfare le aspettative più ambiziose nel caso si abbia a che fare con progetti particolarmente articolati che richiedono quindi una documentazione più strutturata.

Progetti multi-documentazione

Con progetti multi-documentazione intendiamo progetti potenzialmente composti da svariati moduli, librerie e microservizi che, una volta in esercizio, ci si aspetta svolgano un'importante funzione nello scenario dell'Ente. Per un progetto di questa centralità e vastità sembra lecito aspettarsi una documentazione rivolta a molteplici attori (dai vari livelli di supporto applicativo e infrastrutturale fino agli utenti finali) e che sia in grado di favorire uso, manutenzione ed evoluzione del software. La documentazione prevederà presumibilmente uno o più manuali di installazione, amministrazione, estensione, aggiornamento, API e manuale/i utente. Tutti, ad eccezione del manuale utente (per cui, ad ora, è ancora accettato l'utilizzo di un formato binario aperto) necessariamente scritti in un formato testuale che ne garantisca il versionamento riga per riga. Il codice della documentazione (qualunque sia il formato versionabile che l'Incaricato vorrà usare), dovrà essere contenuto in una cartella dei sorgenti (diciamo .../doc) e che il contenuto sia eventualmente convertito e pubblicato in un qualunque formato aperto e raggiungibile dal Wiki di progetto.

Aggiunta del codice sorgente tramite access token

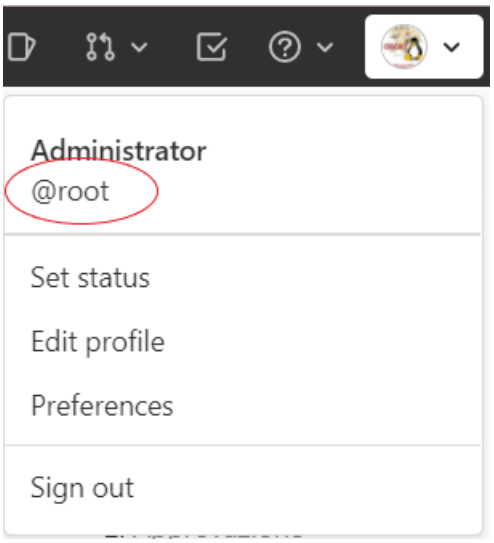
Le credenziali da utilizzare con il client Git sono differenti da quelle utilizzate per l'accesso alla piattaforma.

La coppia username/password da utilizzare deve essere recuperata dal proprio account su Gitlab. Una volta terminato l'accesso, l'utente può seguire le seguenti istruzioni.

Recupero dello username

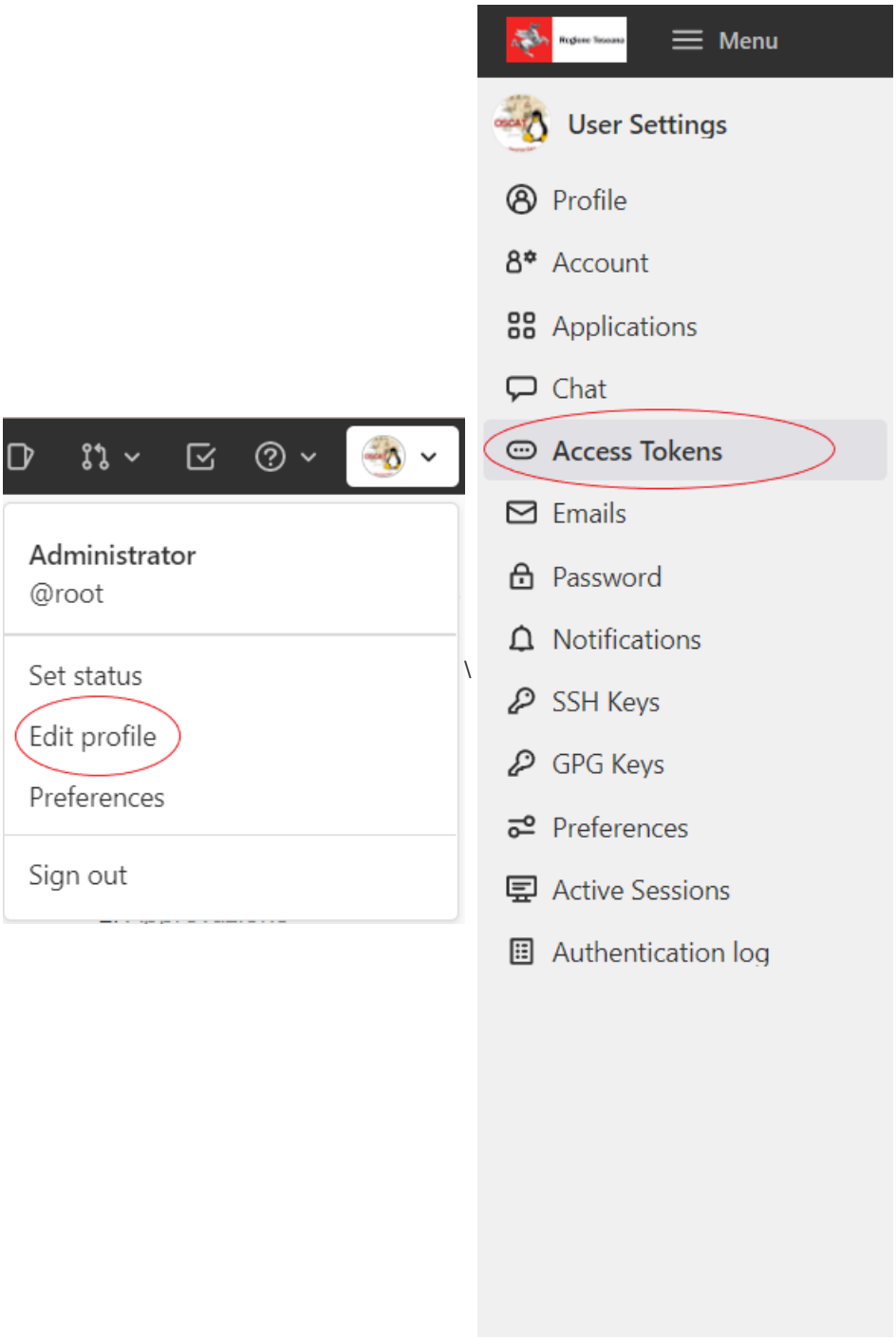
In seguito al login, l'utente ha accesso al servizio Gitlab, che presenta nella navbar in alto un menù con diverse voci. Se è stato inserito il proprio avatar, il menù riporta l'avatar.

Una volta aperto il menù è possibile visualizzare il proprio nome e cognome e, nella riga successiva, lo username.



Generazione dell'Access Token

Una volta recuperato lo username, selezionare la voce Edit profile dallo stesso menù per aprire la sezione di gestione del proprio account su Gitlab e selezionare la voce Access Tokens dal menù laterale sinistro.

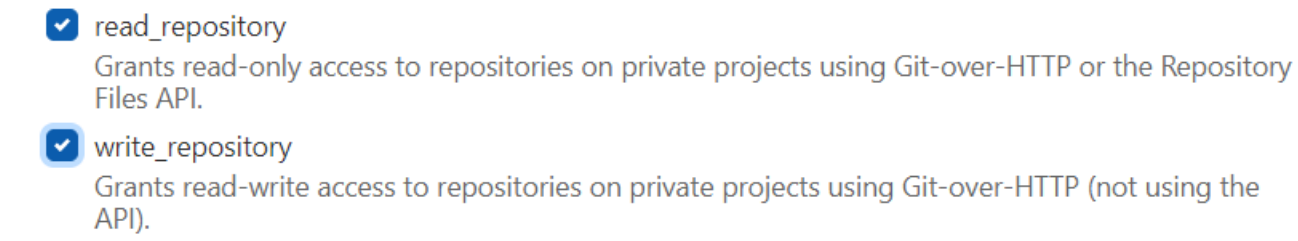


Da questa pagina sarà possibile generare un Access Token che l'utente può utilizzare al posto della password.

La creazione di un Access Token necessita delle seguenti informazioni:

1. un nome che lo identifichi;
2. una data di scadenza, da lasciare nulla se l'utente non necessita di far scadere l'Access Token in una data prestabilita;
3. la lista dei permessi da associare all'Access Token.

Si consiglia l'utilizzo dei soli permessi necessari all'utilizzo del repository.



Attenzione! *Un Access Token può essere visualizzato una volta sola subito dopo la creazione, dopodiché viene salvato cifrato*

e non sarà più ottenibile in chiaro. Si consiglia all'utente di salvarlo subito dopo averlo generato. Sarà necessario generare un nuovo Access token nel caso in cui l'utente lo abbia smarrito o nel caso in cui esso sia scaduto.

Configurazione progetti Java/Maven

Affinché il progetto sia compilato con una specifica versione di Java, a partire dalla versione 1.7, è necessario indicarlo esplicitamente nel pom.xml come da esempio:

```
<plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-compiler-plugin</artifactId>
   <version>...</version>
   <configuration>
     <verbose>true</verbose>
     <fork>true</fork>
     <executable>${JAVA_1_7_HOME}/bin/javac</executable>
     <compilerVersion>${jdk}</compilerVersion>
   </configuration>
</plugin>
```

Valorizzando la variabile ${jdk} con la versione di java scelta, mentre la valorizzazione del tag deve essere scelta fra le seguenti possibilità:

```
<executable>${JAVA_1_7_HOME}/bin/javac</executable>
<executable>${JAVA_1_8_HOME}/bin/javac</executable>
<executable>${JAVA_1_11_HOME}/bin/javac</executable>
```

Nel caso in cui la compilazione Java 11 non vada a buon fine e l'applicazione preveda l'utilizzo di Spring Boot è possibile sostituire il plugin con il sottostante:

```
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
<configuration>
<verbose>true</verbose>
<fork>true</fork>
<executable>${JAVA_1_11_HOME}/bin/javac</executable>
<compilerVersion>11.0.1</compilerVersion>
</configuration>
</plugin>
```

Inoltre è necessario specificare nel pom.xml anche la versione Maven utilizzata tramite il tag <maven.version> nella sezione properties come da esempio:

```
<properties>
...
 <maven.version>${maven}</maven.version>
...
</properties>
```

valorizzando la variabile ${maven} con la versione di maven scelta.

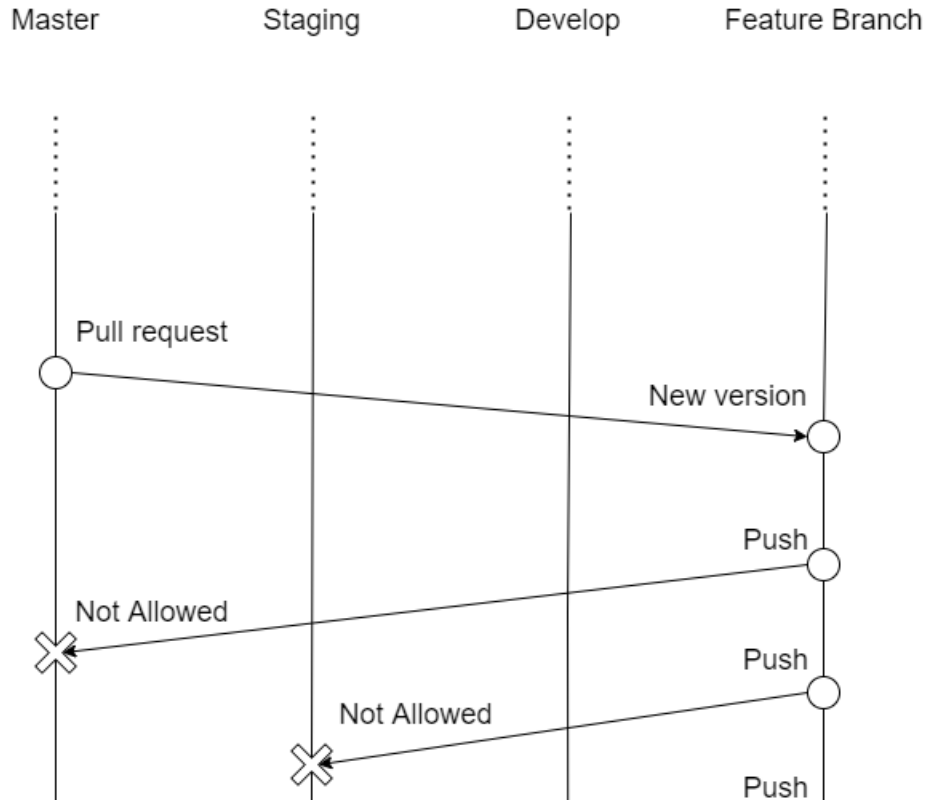
Flusso di lavoro Git

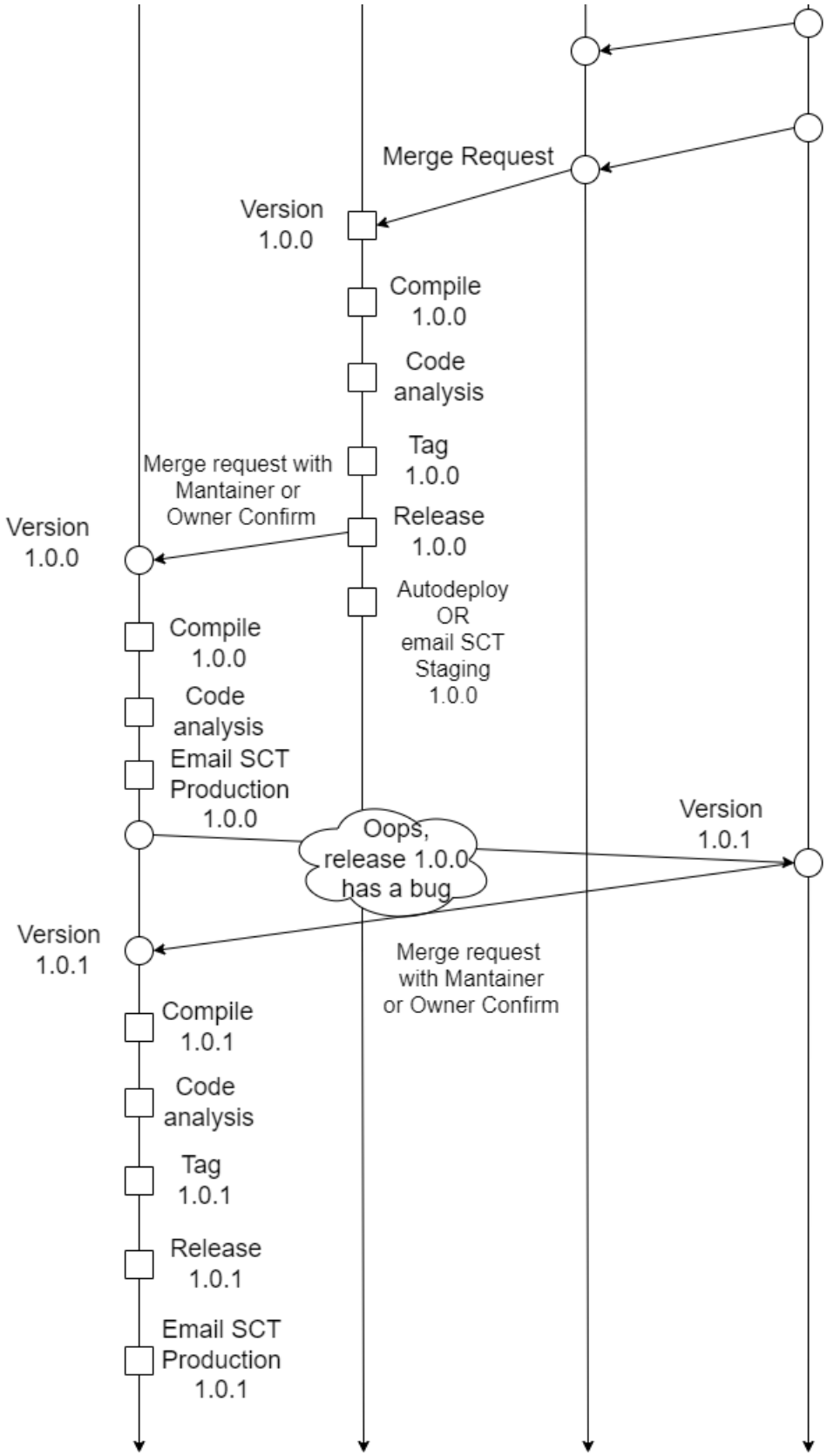
Il ciclo di vita di un'applicazione dotata di CI/CD è ispirato all'approccio noto me Git Flow.

Git Flow

Il modello di branching individuato è il GitFlow, forse uno dei più noti e più utilizzati. La sua schematicità, la presenza di branch separati per le pipeline di certificazione e di produzione e il supporto alle hotfix, sono alcune delle funzionalità che hanno fatto il successo di questo modello di branching. La figura in calce ne mostra uno schema riassuntivo di cui riportiamo di seguito gli elementi essenziali:

- Presenta i branch:
  - master
  - staging
  - develop
  - feature branch
- Ricorre alle seguenti icone:
  -  per indicare i comandi git non consentiti;
  -  per indicare le operazioni Gitlab o i comandi git permessi;
  -  che rappresenta un singolo step della pipeline.
- Descrive i seguenti processi:
  - la creazione di un nuovo feature branch;
  - la merge sul branch develop;
  - la merge request sul branch staging;
  - la merge request sul branch master;
  - la merge request sul branch master per una hotfix.





Branch

Il repository sarà sempre creato con due branch di default:

- staging: è il branch che contiene il codice di certificazione;
- master: è il branch che contiene il codice di produzione.

I due branch saranno branch protetti, per cui non possono essere cancellati. Sarà possibile utilizzare la funzionalità delle Merge Request messa a disposizione da Gitlab per gestire il deposito del codice sorgente su questi due branch. L'utilizzo delle Merge Request non è obbligatorio.

La merge request sul branch staging e la merge request sul branch master potranno essere approvate non solo dagli utenti che possiedono il ruolo di Owner e di Mantainer, quindi ad esempio dal Referente e dal Gestore, ma anche dagli sviluppatori stessi.

Merge request su staging

Una volta che lo sviluppatore intende portare il codice contenuto in un branch di sviluppo sul branch staging, può aprire una merge request dal branch insteressato al branch staging.

La merge request può essere aperta dalla GUI online della piattaforma o tramite le opzioni dedicate della merge di Git, gestite da Gitlab.

Merge request tramite GUI

Gitlab possiede una GUI dedicata alla gestione delle merge request, accedibile dalla voce Merge Requests sulla sinistra. Il bottone riporta un intero che indica il numero di merge request aperte per il progetto.

Nuova merge request

In alto a destra è presente il pulsante per l'apertura di una nuova merge request.

Per prima cosa è necessario scegliere il branch di partenza ed il branch finale su cui verrà effettuato il merge del codice. Nel caso specifico, il branch iniziale sarà il branch che contiene i nuovi sviluppi, il branch finale sarà staging. Fin da questo primo step, la GUI di Gitlab fornisce degli indizi utili sullo stato del codice.

Dal momento che il branch di partenza non sarà né staging né master, non avrà associata nessuna pipeline per la CI/CD. Una volta selezionato questo branch, viene però mostrato il codice sha dell'ultimo commit e il messaggio con cui è stato pushato.

Per il branch staging viene invece mostrato lo stato dell'ultima pipeline eseguita per il codice attualmente presente sul branch.

Premendo il pulsante *Compare branches and continue*, l'utente ha l'accesso alla pagina di apertura della merge request. La descrizione della merge request, prepopolata tramite un template configurato, aiuta il team di sviluppo nella comprensione delle modifiche effettuate.

Merge request aperte

È possibile visualizzare la lista delle merge request ancora aperte dalla pagina apposita e visualizzare il dettaglio di ciascuna.

La pagina di dettaglio fornisce informazioni importanti sullo stato della merge. Oltre al nome e alla descrizione inseriti in fase di creazione, permette di visualizzare il diff dei file e la lista dei commit che saranno portati sul branch di staging. Inoltre, è possibile lasciare dei commenti per ogni merge request aperta, in modo da comunicare le proprio impressioni sul merge.

Ogni merge request può essere modificata, chiusa e rimessa tra le bozze, se si ritiene che il merge non è ancora completo. La chiusura di una merge request avviene tramite il tasto Merge e può essere effettuata da tutti gli utenti del progetto che abbiano almeno il ruolo di Developer.

Merge request tramite linea di comando

È possibile aprire una nuova merge request anche da linea di comando, utilizzando le apposite opzioni git per il comando push.

Una merge request effettuata tramite linea di comando richiede di eseguire il seguente comando dal branch di partenza in seguito ad una merge:

```
git push -o merge_request.create -o merge_request.title="Merge su branch staging" -o merge_request.target=staging -o merge_request.description="Descrizione della merge request"
```

Si consiglia l'utilizzo della GUI vista la sua praticità e chiarezza.

Merge request su master

Una volta che lo sviluppatore intende portare il codice contenuto nel branch staging dul branch master, può aprire una merge request dal branch staging al branch master.

La merge request può essere aperta dalla GUI online della piattaforma o tramite le opzioni dedicate della merge di Git, gestite da Gitlab.

Merge request tramite GUI

Gitlab possiede una GUI dedicata alla gestione delle merge request, accedibile dalla voce Merge Requests sulla sinistra. Il bottone riporta un intero che indica il numero di merge request aperte per il progetto.

Nuova merge request

In alto a destra è presente il pulsante per l'apertura di una nuova merge request.

Per prima cosa è necessario scegliere il branch di partenza ed il branch finale su cui verrà effettuato il merge del codice. Nel caso specifico, il branch iniziale sarà il branch staging, il branch finale sarà master. Fin da questo primo step, la GUI di Gitlab fornisce degli indizi utili sullo stato del codice.

Per entrambi i branch viene mostrato lo stato dell'ultima pipeline eseguita per il codice attualmente presente sul branch.

Premendo il pulsante Compare branches and continue, l'utente ha l'accesso alla pagina di apertura della merge request. La descrizione della merge request, prepopolata tramite un template configurato, aiuta il team di sviluppo nella comprensione delle modifiche effettuate.

Merge request aperte

È possibile visualizzare la lista delle merge request ancora aperte dalla pagina apposita e visualizzare il dettaglio di ciascuna.

La pagina di dettaglio fornisce informazioni importanti sullo stato della merge. Oltre al nome e alla descrizione inseriti in fase di creazione, permette di visualizzare il diff dei file e la lista dei commit che saranno portati sul branch di master. Inoltre, è possibile lasciare dei commenti per ogni merge request aperta, in modo da comunicare le proprio impressioni sul merge.

Ogni merge request può essere modificata, chiusa e rimessa tra le bozze, se si ritiene che il merge non è ancora completo. La chiusura di una merge request avviene tramite il tasto Merge e può essere effettuata da tutti gli utenti del progetto che abbiano almeno il ruolo di Developer.

Merge request tramite linea di comando

È possibile aprire una nuova merge request anche da linea di comando, utilizzando leapposite opzioni git per il comando push.

Una merge request effettuata tramite linea di comando richiede di eseguire il seguente comando dal branch staging in seguito ad una merge:

```
git push -o merge_request.create -o merge_request.title="Merge su branch master" -o merge_request.target=master -o merge_request.description="Descrizione della merge request"
```

Si consiglia l'utilizzo della GUI vista la sua praticità e chiarezza.

Hotfix

Durante il ciclo di vita di un'applicazione, lo sviluppatore potrebbe avere la necessità di rilasciare la correzione di un bug nell’ambiente di produzione senza prima rilasciarlo in quello di certificazione. Sebbene l'iter non sia quello presentato finora, esistono delle situazioni in cui ciò può essere necessario.

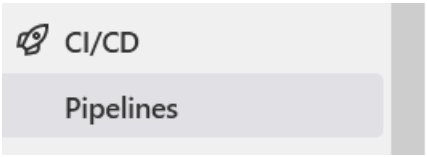
Per gestire questo caso, è previsto che lo sviluppatore crei un branch a partire dal master in cui farà push delle modifiche. Una volta terminati i commit riguardanti la fix, questi possono essere portati direttamente sul master tramite una merge request. La merge request sul master (a partire dal branch contenente la fix) deve essere confermata dal Referente o da un collaboratore da lui indicato. Per confermare la merge request sarà necessario premere l’apposito bottone Merge presente nella pagina riassuntiva della merge request.

Per approfondire la gestione delle Merge request sul branch master, è possibile consultare la pagina Merge request su Master, tenendo presente che l'unica differenza è il branch di partenza.

Build degli applicativi

Le operazioni di compilazione, analisi del codice, creazione di release e rilascio vengono effettuate tramite pipeline Gitlab, cioè una serie di istruzioni contenute nel file *.gitlab-ci.yml* presente nella radice del repository.

La pipeline assegnata ad ogni progetto dipende dalle caratteristiche dello stesso e in particolare dal linguaggio di programmazione (Java, Angular, ecc), dal package manager (Maven, Npm) e dal tipo di rilascio (Nexus, 3-tier, Docker) e viene avviata ad ogni push sul repository. Inoltre può essere anche avviata manualmente.



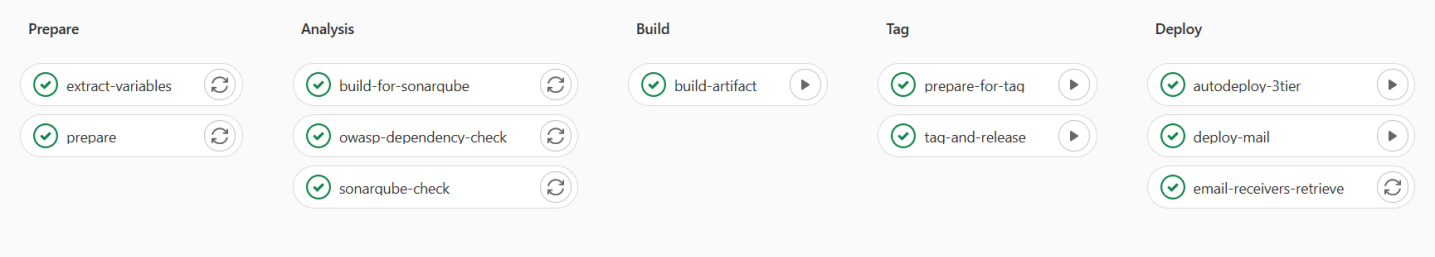
Nei casi in cui la CI/CD, per qualche valida ragione, non sia configurata fin da subito viene comunque assegnata una pipeline d'ufficio per garantire l'analisi statica del codice.

Stage delle pipeline

Stage

Ogni pipeline è composta da varie fasi, dette stage, dove ognuna può prevedere uan sequenza di operazioni (job).

Esempio per progetto 3-tier

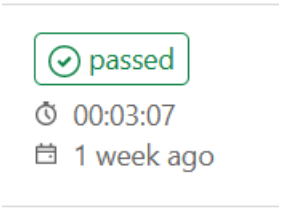


Le pipeline OSCAT sono costituite da 5 stage:

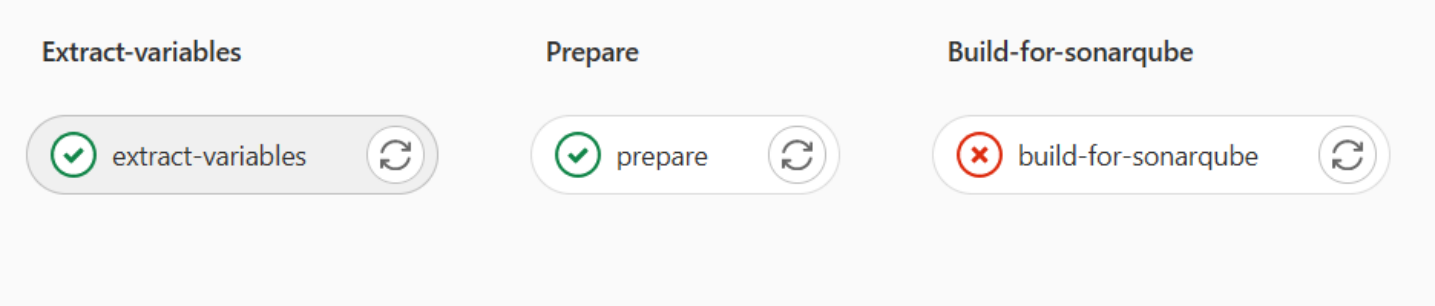
1. *Prepare:* questo stage raggruppa le operazioni necessarie per recuperare le configurazioni e le variabili necessarie per eseguire la pipeline. Per approfondire: Modifica configurazione build applicativi.
2. *Analysis:* questo stage raggruppa le analisi eseguite sul codice, sui binary e/o sui test. Per approfondire: Analisi SonarQube.
3. *Build:* questo stage compila e costruisce il pacchetto da dispiegare.
4. *Tag:* questo stage si occupa della storicizzazione del codice e degli artefatti. Per approfondire:Storicizzazione.
5. *Deploy:* questo stage si occupa del dispiegamento dell'artefatto. Per approfondire:Rilascio di un'applicazione.

Stato

Tramite il pulsante verde "passed" (o rosso nel caso di "failed") è possibile visualizzare lo stato della pipeline.



La spunta verde a sinistra del nome di ogni job indica che l'operazione è stata eseguita con successo, al contrario la x rossa indica un errore.



Lista delle pipeline

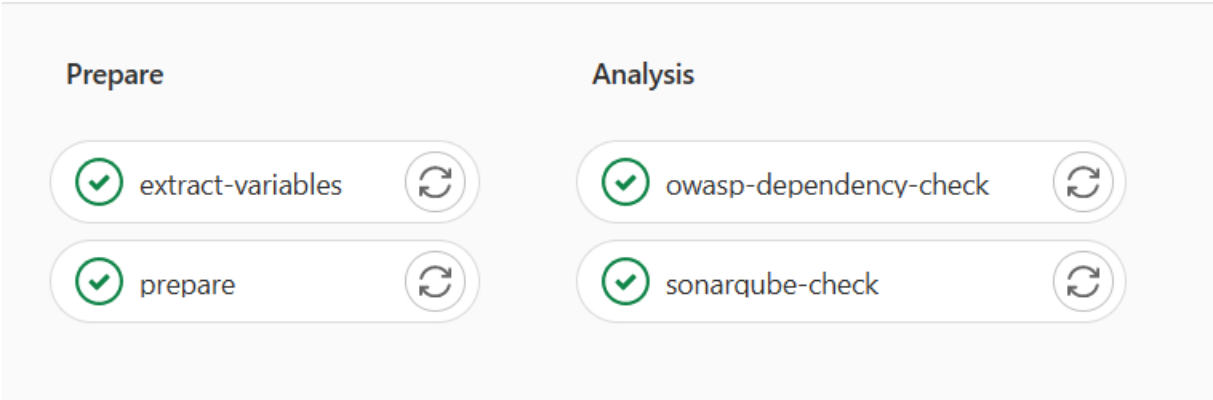
Nella sezione *CI/CD > Pipelines* di GitLab sono disponibili tutte le pipeline eseguite per quel determinato progetto.

Se lo stato riportato è "passed" in verde significa che tutte le operazioni previste sono state eseguite con successo e la pipeline è stata completata. Se invece lo stato è "failed" in rosso significa che c'è stato un errore e la pipeline non è terminata correttamente.

Log delle pipeline

Il log di ogni singolo job è accessibile sia dalla pagina principale della sezione pipeline

sia dalla pagina di dettaglio.

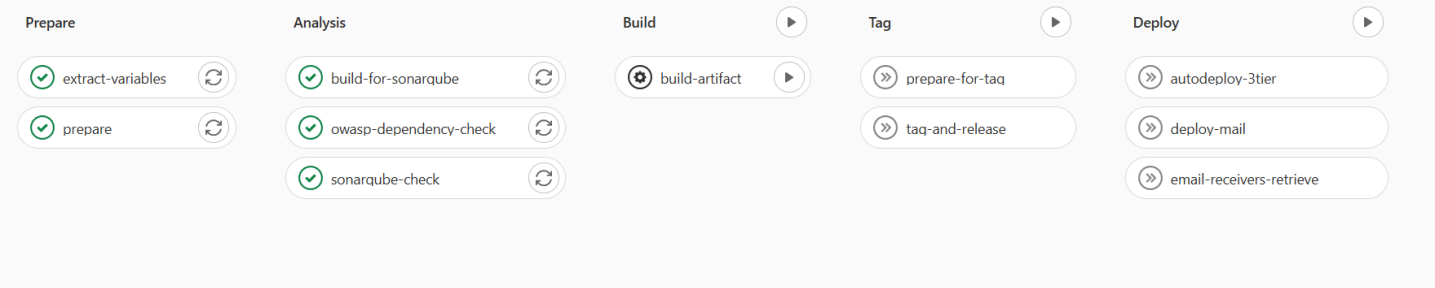


E' possibile visualizzaer il log facendo click sul nome nel job eseguito, sia in caso di successo che di errore.

```
191 VAR_PATH_TARGET=dist
192 VAR_FILE_EXTENSION=tgz
193 VAR_NODE_VERSION=10.16.3
195 Uploading artifacts for successful job                    00:01
196 Using docker image sha256:c825f5b6f2432ce1b294c549766ebbaa418d998ec625d8ca0a4f2e075394892a for gitlab/gitlab-runner-helper:ubuntu-x86_64-v14.10.0-pwsh with digest gitlab/gitlab-runner-helper@sha256:4385f22af4435cb093e101cad4583e9d368f876ff6715d19b64c54cc958df91f ...
197 Uploading artifacts...
198 values.props: found 1 matching files and directories
199 Uploading artifacts as "dotenv" to coordinator... 201 Created  id=8261 responseStatus=201 Created token=8ymzvEfx
201 Cleaning up project directory and file based variables    00:00
202 Using docker image sha256:c825f5b6f2432ce1b294c549766ebbaa418d998ec625d8ca0a4f2e075394892a for gitlab/gitlab-runner-helper:ubuntu-x86_64-v14.10.0-pwsh with digest gitlab/gitlab-runner-helper@sha256:4385f22af4435cb093e101cad4583e9d368f876ff6715d19b64c54cc958df91f ...
204 Job succeeded
```

Stage opzionali

I vari job solitamente vengono eseguiti in modo automatico uno dopo l'altro. In alcuni casi, al fine di garantire la massima flessibilità agli utilizzatori, si è scelto di rendere opzionale l'esecuzione di alcuni job. E' possibile avviari questi job manualmente. I job manuali sono riconoscibili da una rotella sulla sinistra del job e sulla destra un pulsante con freccia nera da premere (e.g. build-artifact).



Modifica della configurazione della build degli applicativi

Le configurazioni richieste in fase di apertura di un progetto potrebbero richiedere una modifica. Per richiederne l'aggiornamento è necessario inviare una mail al supporto OSCAT che si farà carico di aggiornarle. Alla pagina Richiesta di esempio è presente un template che è possibile utilizzare per la richiesta.

Alla pagina Lista delle configurazioni è possibile trovare una lista non esaustiva delle configurazione previste per le applicazioni.

Richiesta di esempio

To: Supporto OSCAT \<oscat.support@tai.it>\ Subject: Richiesta di modifica configurazioni *Nome applicativo*\ Body:

Buongiorno,\ con la presente si richiede l'aggiornamento delle configurazioni per l'applicativo*Nome applicativo* con ID su Gitlab _ID_.

Chiave	Vecchio valore	Nuovo valore
Configurazione1	Vecchio valore	Nuovo valore
ConfigurazioneN	Vecchio valore	Nuovo valore

Cordiali saluti,

Firma

Lista delle configurazioni

Si riporta una lista delle configurazioni che è necessario configurare per le diverse applicazioni. La lista potrebbe non essere completa e/o in corso di aggiornamento.

Compilazione

Java Maven

Per i progetti Java che utilizzano Maven come package manager è possibile specificare:

Chiave	Descrizione
goal_maven	Il gola maven da invocare durante la fase di compilazione
path_pom	Il path del pom da utilizzare in compilazione e per estrarre le informazioni del progetto
path_target	Il path del target in cui sono salvati i binari
file_extension	Il tipo di artefatto prodotto: jar, ear o war

Java Ant

Per i progetti Java che utilizzano Ant come package manager è possibile specificare:

Chiave	Descrizione
path_buildxml	Il path del file build.xml da utilizzare in compilazione e per estrarre le informazioni del progetto
path_target	Il path del target in cui sono salvati i binari
file_extension	Il tipo di artefatto prodotto: jar, ear o war

Angular NPM

Per i progetti Angular che utilizzano NPM come package manager è possibile specificare:

Chiave	Descrizione
path_package_json	Il path del file package.json da utilizzare per estrarre le informazioni del progetto
path_target	Il path del target in cui sono salvati i binari
file_extension	Il tipo di artefatto da produrre, ad esempio tar.gz

Rilascio

Per la fase di rilascio è possibile specificare le seguenti configurazioni.

Chiave	Descrizione
Modalità di rilascio	Può essere una tra: Nexus, 3-tier, Docker, Self Managed
Autodeploy attivo	Specifica se si intende attivare l'autodeploy o meno
path_dockerfile	Il path al Dockerfile, avrà valenza se e solo se la Modalità di Rilascio è Docker
docker_registry	Il Docker registry su cui salvare l'immagine, avrà valenza se e solo se la Modalità di Rilascio è Docker

Autodeploy

Nel caso in cui l'autodeploy sia attivo, è possibile specificare diverse configurazioni sulla base dell'autodeploy che si intende avere.

Autodeploy JBoss

Chiave	Descrizione
jboss_version	La versione di JBoss da utilizzare
jboss_servers	Il nome del server su cui dispiegare l'applicazione
artifact_path	Il path all'artefatto da dispiegare, generalmente la cartella target

Autodeploy Tomcat

Chiave	Descrizione
tomcat_version	La versione di Tomcat da utilizzare. Può essere 6 oppure 7 o più
tomcat_url	Il nome del server su cui dispiegare l'applicazione
tomcat_release_name	Il nome del file da dispiegare
tomcat_context_name	Il nome del contesto su cui dispiegare l'applicazione

Autodeploy Docker

Chiave	Descrizione
docker_prefix	Il prefisso usato in fase di avvio del container
docker_component	Il nome del componente usato in fase di avvio del container

Analisi SonarQube

Ad ogni compilazione, vengono eseguiti gli eventuali test e un'analisi dei sorgenti in modo da rendere disponibile un'analisi della qualità del codice prodotto. Le analisi saranno eseguite sul branch staging e sul branch master.

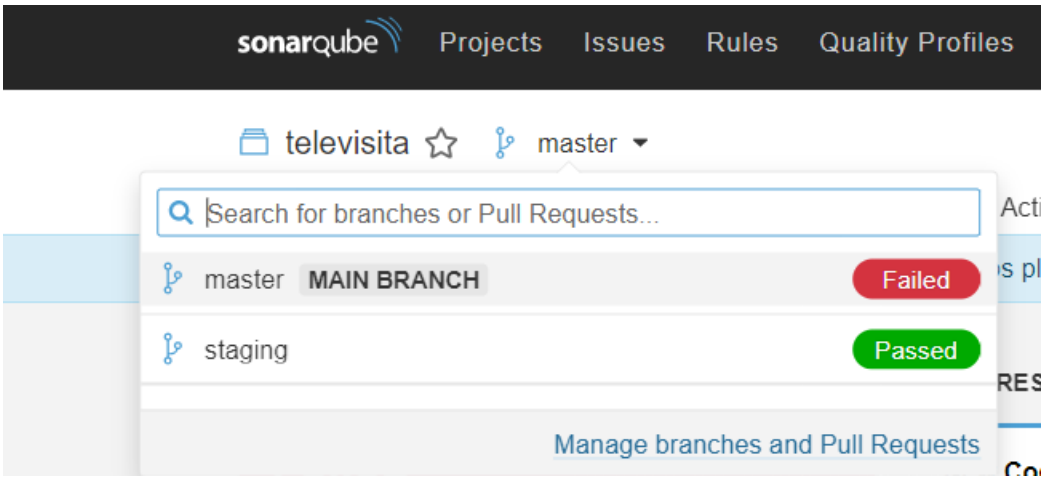
I progetti che non necessitano della CI/CD, avranno comunque una pipeline che permette la compilazione e l'analisi statica del codice sorgente.

Link a SonarQube

La dashboard di SonarQube è disponibile al seguente link.

Ogni utente possiede la stessa visibilità di progetti posseduta su Gitlab.

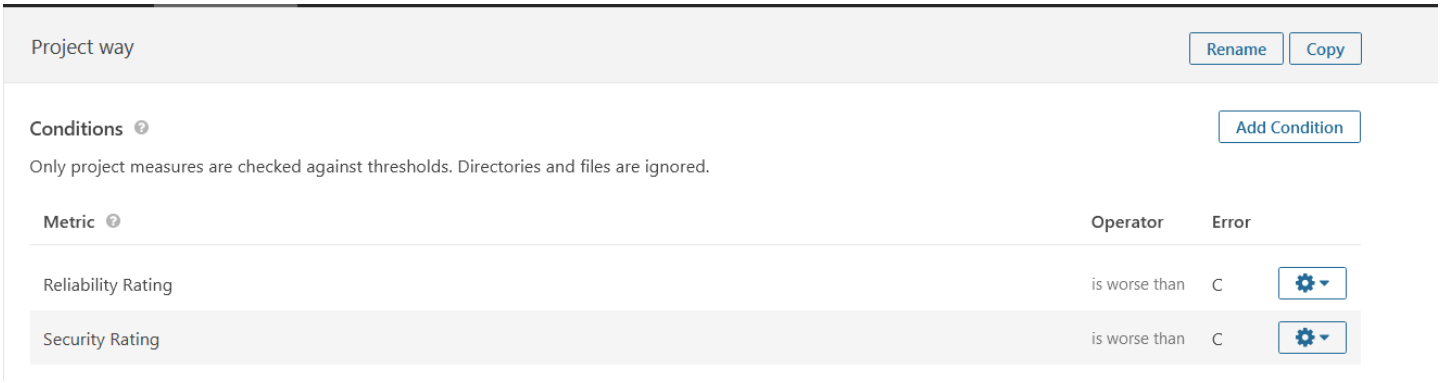
Una volta terminato il job dell'analisi di SonarQube, il log riporterà il link alla dashboard del progetto. Per distinguere tra le analisi di staging e quella di master selezionare il branch corretto nella tendina dedicata sulla navbar.



Quality Gate

Al fine di limitare i rischi per l'ente e migliorare la qualità del codice nelle sue disponibilità, tutti gli Incaricati sono invitati a fare il possibile per eliminare ogni rischio dalle proprie applicazioni.

Sebbene l'invito sia quello già fatto nel preambolo di questa sezione, in calce si riporta il Quality Gate attualmente definito. L'Ente naturalmente può modificare questo livello di qualità in ogni momento oltre che impedire lo scaricamento di librerie che abbiano vulnerabilità note.



Reliabity rating

Questo parametro è calcolato sulla tipologia e sul numero dei bug presenti. Dalla documentazione ufficiale si evince che:

Reliability Rating (`reliability_rating`)
A = 0 Bugs
B = at least 1 Minor Bug
C = at least 1 Major Bug
D = at least 1 Critical Bug
E = at least 1 Blocker Bug

Un reliability rating di almeno C garantisce l'assenza di bug Critical e Blocker.

Security rating

Questo parametro è calcolato sulla tipologia e sul numero di vulnerabilità presenti. Delladocumentazione ufficiale si evince che:

Security Rating (`security_rating`)
A = 0 Vulnerabilities
B = at least 1 Minor Vulnerability
C = at least 1 Major Vulnerability
D = at least 1 Critical Vulnerability
E = at least 1 Blocker Vulnerability

Un security rating di almeno C garantisce l'assenza di vulnerabilità Critical e Blocker.

Rilascio di un'applicazione

OSCAT offre diverse tipologie di dispiegamento sulla base dell'ambiente considerato.

Ambiente di Certificazione

In questo ambiente è attualmente possibile effettuare un dispiegamento con una delle seguenti modalità:

- **deploy tramite invio email:** nel caso in cui non siano soddisfatte le condizioni per l'autodeploy, è possibile richiedere il dispiegamento della propria applicazione tramite email al Supporto OSCAT.
- **autodeploy:** il meccanismo di dispiegamento automatico consente di dispiegare senza attendere alcuna autorizzazione usando la pipeline associata al branch di staging. Affinché sia possibile ricorrere a tale dispiegamento è necessario di aver prima eseguito un rilascio con le modalità tradizionali (ciò perché è prima necessario sciogliere alcune questioni preliminari, al momento, non automatizzate).

Ambiente di Produzione

Il rilascio in produzione avviene sempre tramite l'invio di una mail al Gestore dell'ambiente target. La fase di rilascio è comunque condizionata dalla ricezione dell'autorizzazione esplicita del Referente del progetto.

Caricamento librerie su Nexus

Per poter fruire dell'autodeploy nel caricamento di librerie su Nexus è necessario che il Quality Gate del progetto risulti superato.

Autodeploy

La fase di autodeploy è offerta attraverso la pipeline associata al progetto.

Sulla base delle configurazioni comunicate in apertura del progetto e/o al Supporto OSCAT, ogni progetto dotato di autodeploy sarà rilasciato automaticamente al termine dell'esecuzione della pipeline.

Tipologie di autodeploy

Autodeploy Nexus

I progetti da depositare sul Nexus di Regione Toscana necessitano di autodeploy, poiché è la modalità raccomandata per questo tipo di dispiegamento. La funzionalità di autodeploy è garantita a tutti gli applicativi che abbiano superato il Quality Gate di SonarQube fissato da Regione Toscana. In caso contrario è possibile richiedere un deploy manuale tramite mail al Supporto OSCAT che aspetterà l'autorizzazione del referente.

Autodeploy 3-tier e autodeploy Docker

Questi tipi di rilasci possono essere utilizzati per tutti i rilasci successivi al primo. Infatti i puntamenti alle macchine su cui dispiegare l'applicativo sono comunicati da SCT in seguito al primo rilascio. Una volta che SCT ha condiviso con l'Incaricato le informazioni necessarie per la configurazione, quest'ultimo può richiedere al Supporto OSCAT di inserirle nell'apposita sezione su Gitlab per garantire che i rilasci successivi siano automatici.

Deploy tramite email

Le applicazioni per le quali non è possibili richiedere l'attivazione dell'autodeploy possono utilizzare la modalità standard di rilascio, che richiede l'invio di un'email al Supporto OSCAT.

La richiesta di rilascio in Produzione necessita inoltre dell'email di approvazione da parte del Referente di Regione Toscana.

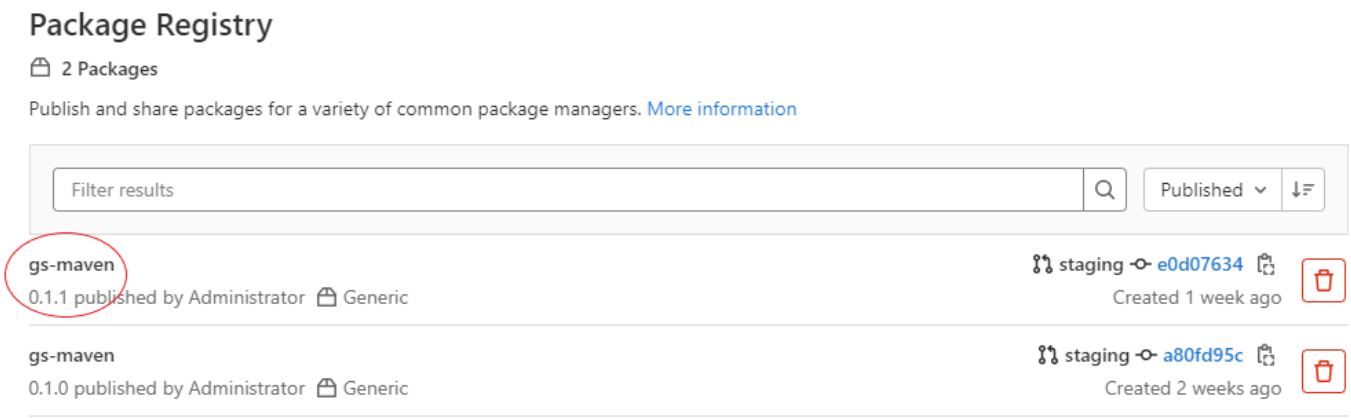
L'email di richiesta partirà in seguito all'esecuzione della pipeline corrispondente all'ambiente su cui si intende effettuare il dispiegamento. La richiesta di rilascio per l'ambiente di Certificazione avverrà in seguito al termine della pipeline di staging. La richiesta di rilascio per l'ambiente di Produzione avverrà in seguito al termine della pipeline di master.

Scaricamento artefatto

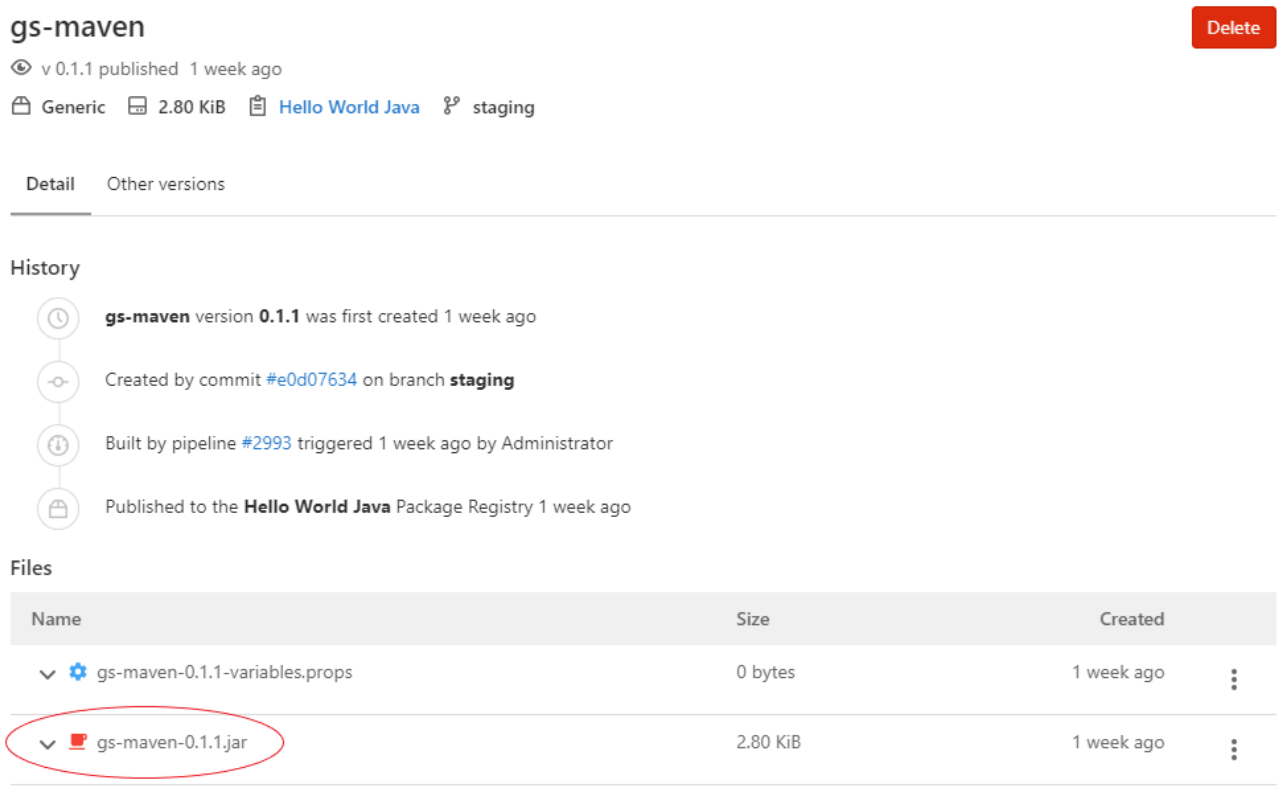
L'artefatto prodotto viene salvato e versionato all'interno delPackage Registry di Gitlab e diventa disponibile per il download in diversi punti del sistema.

Package Registry

Una volta selezionata la versione per cui si intende scaricare il pacchetto, è sufficiente recarsi nella sezione*Package Registry* e individuare la versione nella lista dei pacchetti caricati. Ad esempio, per scaricare il pacchetto prodotto per la versione 0.1.1 per l'applicazione gs-maven:



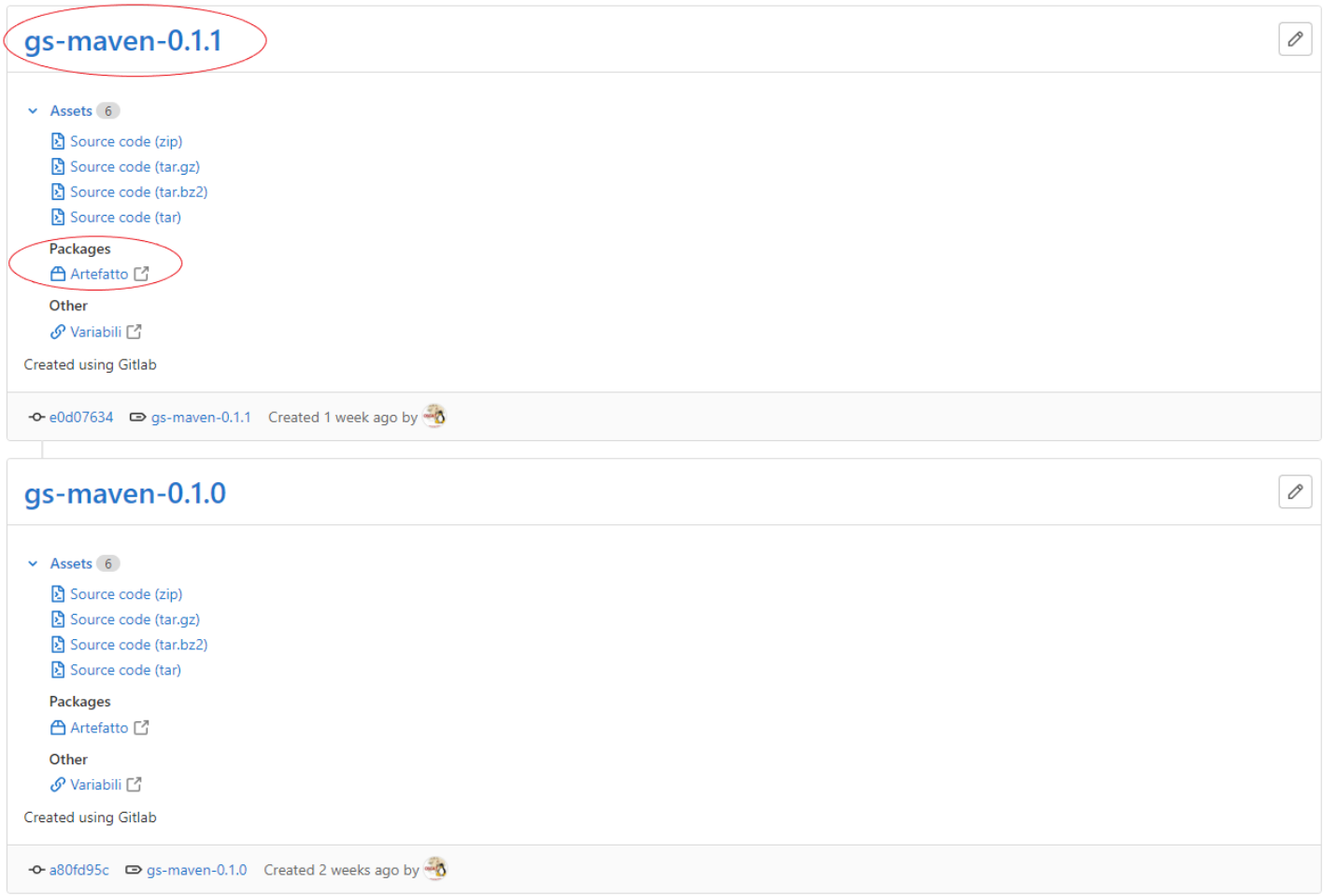
All'interno della pagina di dettaglio, è presente un link che scatena lo scaricamento diretto del pacchetto.



Release

Per ogni Release aggiunta è presente un link che scatena lo scaricamento diretto del pacchetto.

Una volta individuata la versione per cui si intende scaricare il pacchetto è sufficiente premere il link per scaricare l'artefatto. Ad esempio, per scaricare il pacchetto prodotto per la versione 0.1.1 per l'applicazione gs-maven:

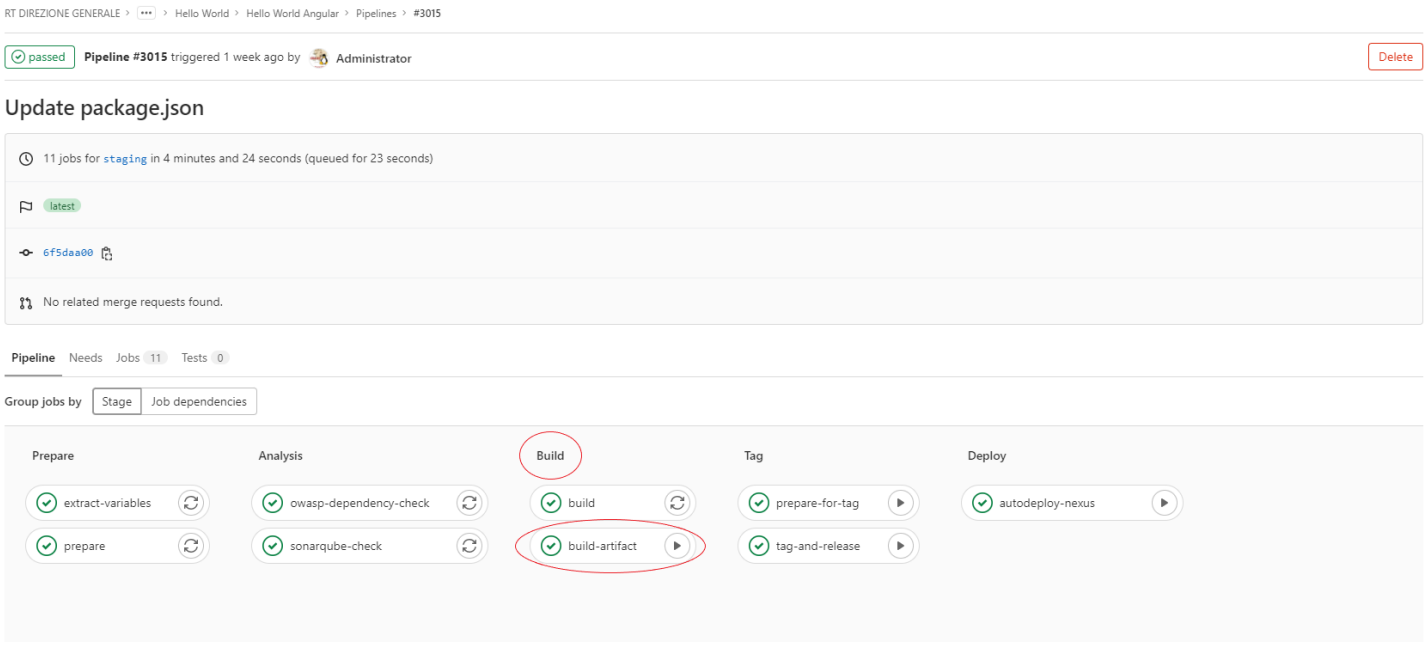


Artefatto della pipeline

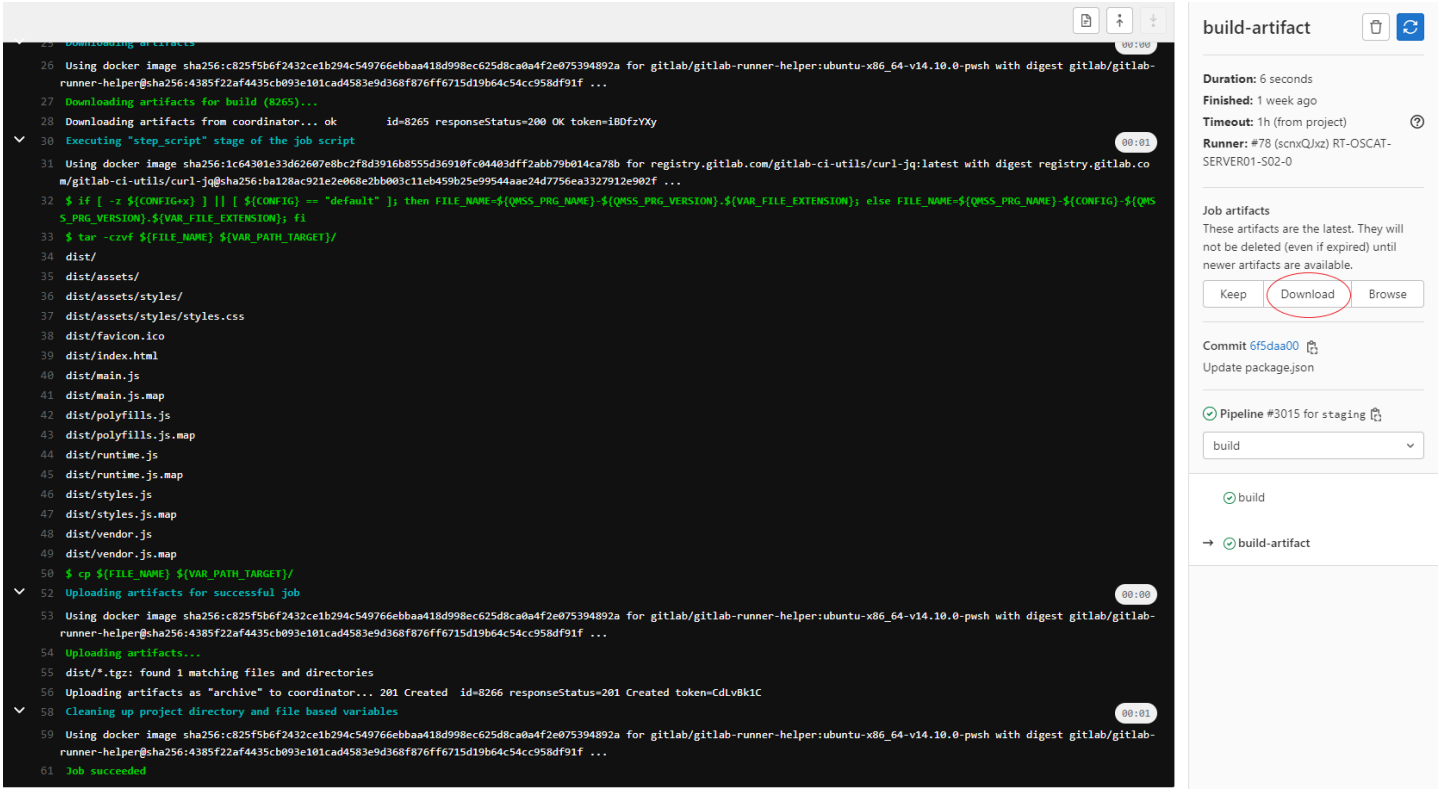
Al termine di ogni pipeline, gli artefatti prodotti sono disponibili per lo scaricamento. È possibile scaricare il pacchetto dal job o dalla pipeline.

Download da job

Il pacchetto può essere scaricato accedendo alla pagina di dettaglio del job che lo ha prodotto e premendo sul Pulsante download. Il job sarà sempre parte dello stage **Build** e si chiamerà *build-artifact*, a prescindere dal linguaggio e dal package manager utilizzato.

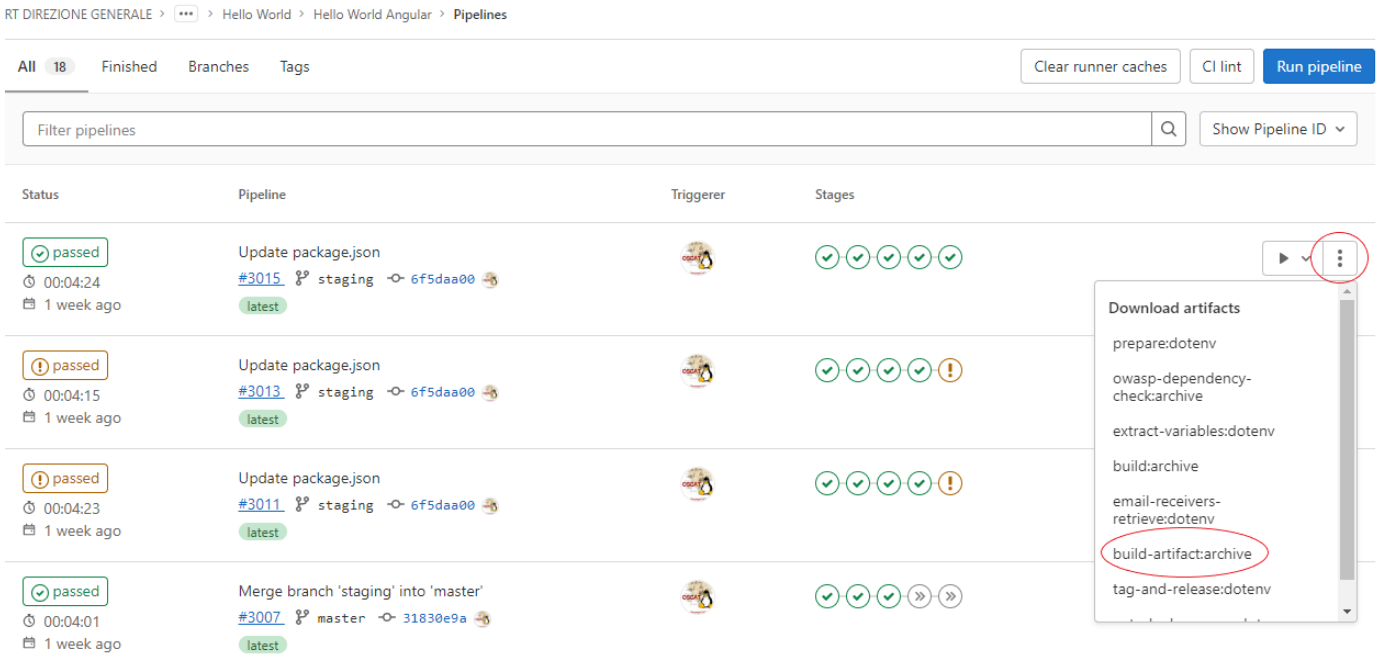


Il pulsante di Download si trova nello specchietto sulla destra.



Download dalla pipeline

Accedendo alla sezione artefatti della singola pipeline, è possibile scaricare l'artefatto senza aprire il dettaglio del job. Una volta aperta la pagina che riporta la lista delle pipeline, individuare la pipeline corretta e aprire il menù Download Artifacts. L'artefatto da scaricare si chiamerà *build-artifact:archive*, a prescindere dal linguaggio e dal package manager utilizzato.



Storicizzazione

Al fine di dar seguito ai principi riportati nelle pagine iniziali di questa documentazione, il codice sorgente e i binari prodotti di ogni progetto sono storicizzati e gestiti come descritto di seguito.

Branch

I branch di staging e di master sono associati ad una pipeline che, prima della fase di rilascio, carica il prodotto della compilazione nel *Package Registry* di Gitlab, crea un nuovo tag e infine crea una nuova release. Tramite queste entità è possibile risalire al codice sorgente di quanto dispiegato in Certificazione e in Produzione.

Package Registry

Gli artefatti prodotti dalle pipeline e il file che racchiude le configurazioni utilizzate sono caricati all'interno del *Package Registry* di Gitlab e associati alla versione del codice compilato. Per approfondire vedi: Package Registry.

Tag

Prima di ogni rilascio, il codice viene taggato e il nome del tag è ricavato a partire dalla versione del codice. Nel caso in cui il tag esista già e si stia quindi tentando di dispiegare la stessa versione già presente, la pipeline terminerà con errore.

Release

Per ogni tag aggiunto, Gitlab crea un'ulteriore entità chiamata *Release*. Una *Release* è un aggregatore dei seguenti componenti:

- codice sorgente riferito dal tag;
- informazioni relative al tag (sha dell'ultimo commit, nome, messaggio del commit...);
- una lista di url ritenuti rilevanti.

La pagina riassuntiva fornita dalla Release garantisce una panoramica esaustiva del codice sorgente e dei binari associati ad una specifica versione.

Tag

Il tag del codice costituisce una parte importante del ciclo di vita dello sviluppo. Esso permette di cristallizzare una lista di commit e segnare un punto del codice a cui è sempre possibile risalire nonostante il branch sia stato modificato.

Nella soluzione OSCAT, i tag sono prodotti per identificare la versione del codice sorgente che si intende dispiegare (siano essi in Certificazione che in Produzione).

Allo scopo di rispettare il Principio 2, i tag saranno gestiti come segue:

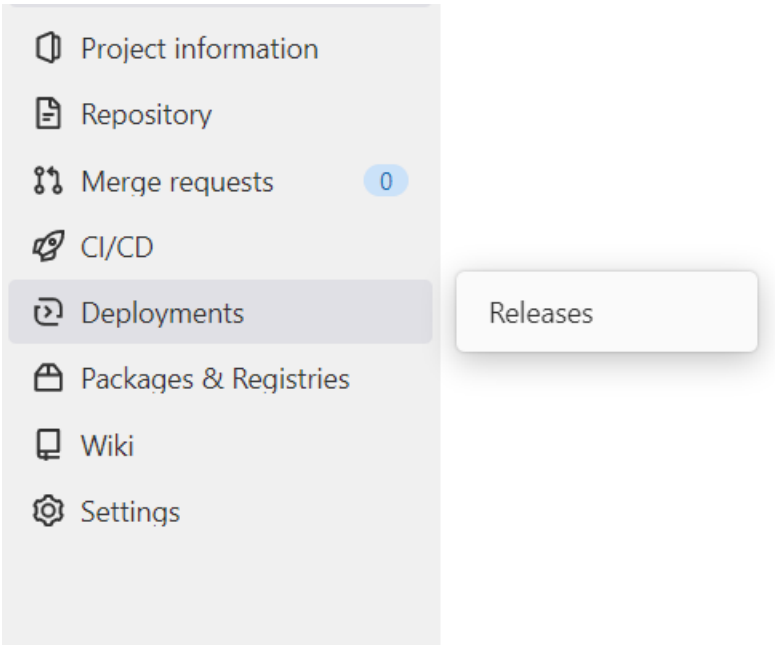
- il codice presente sul branch staging sarà taggato prima di ogni rilascio in Certificazione.
- Il codice sul branch master sarà taggato prima di un rilascio in Produzione se e solo se non sia già presente un tag della stessa versione.

Infatti, sul branch master il tag potrebbe essere già presente se si sta parlando della stessa versione presente su staging, per cui non è necessario produrre un nuovo tag. Questo garantisce che il codice dispiegato in Certificazione sia lo stesso di quello dispiegato in Produzione a parità di versione.

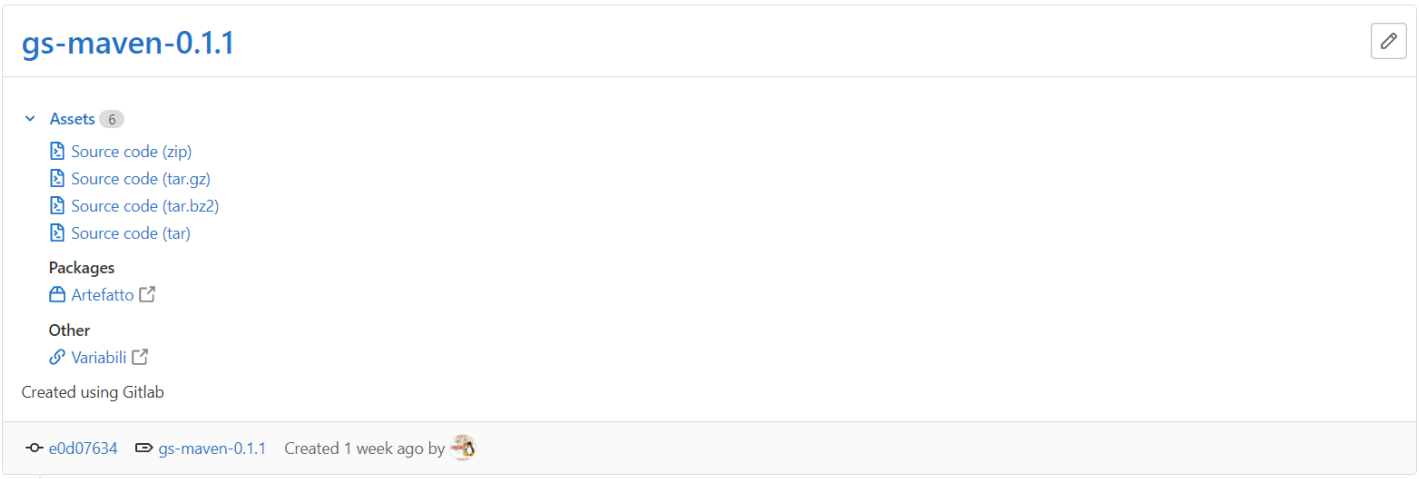
In caso contrario, siamo di fronte ad una hotfix, per cui il codice verrà taggato direttamente su master.

Release

La pagina *Releases*, raggiungibile tramite il menù laterale sulla sinistra, racchiude la lista delle release presenti per il progetto.



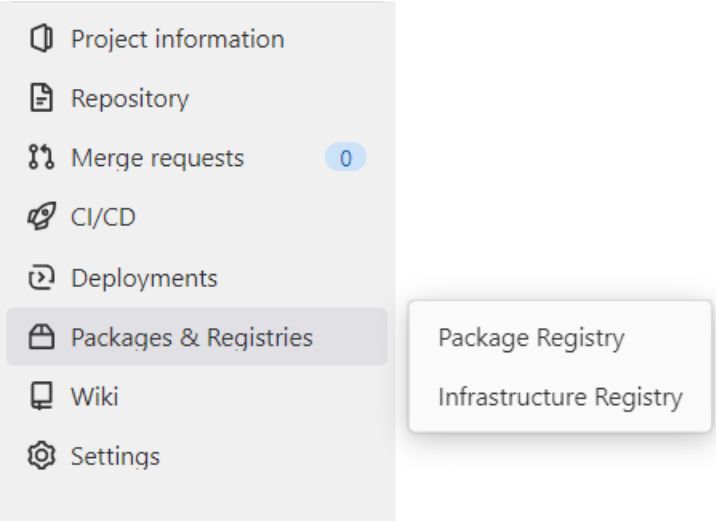
Ogni release è caratterizzata da un nome composto dalla concatenazione del nome dell'applicazione, la versione dell'applicazione e una stringa che rappresenta la configurazione usata (nel caso ne siano presenti più di una).



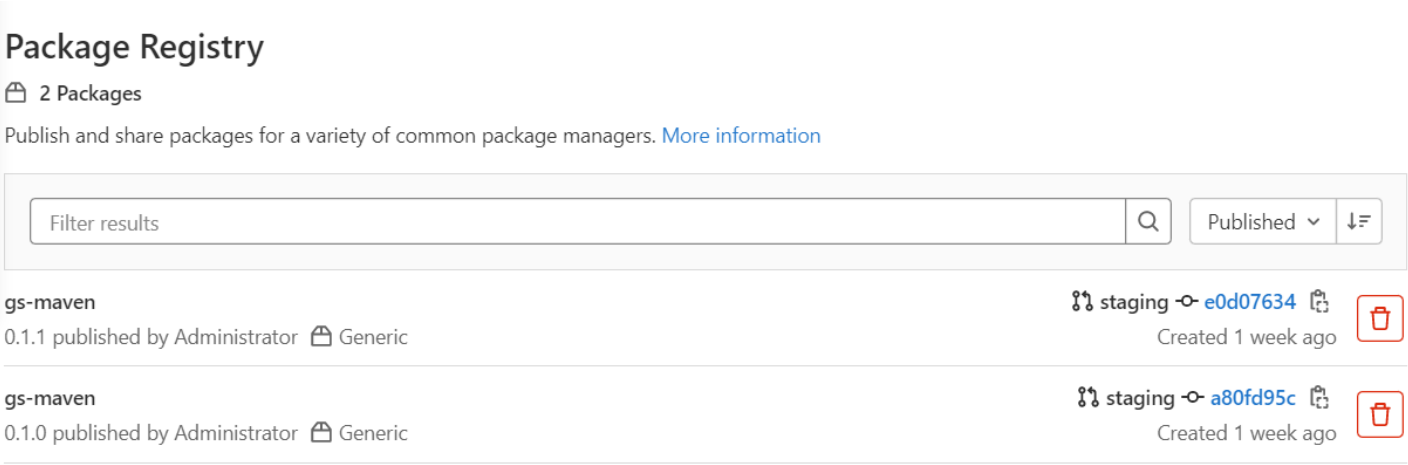
Navigando attraverso gli url presenti nello specchietto, è possibile scaricare i diversi componenti della*Release* in locale per visionarli.

Package Registry

La pagina *Package Registry*, raggiungibile tramite il menù laterale sulla sinistra, racchiude la lista dei file e degli artefatti caricati nella suddetta sezione dello specifico progetto.

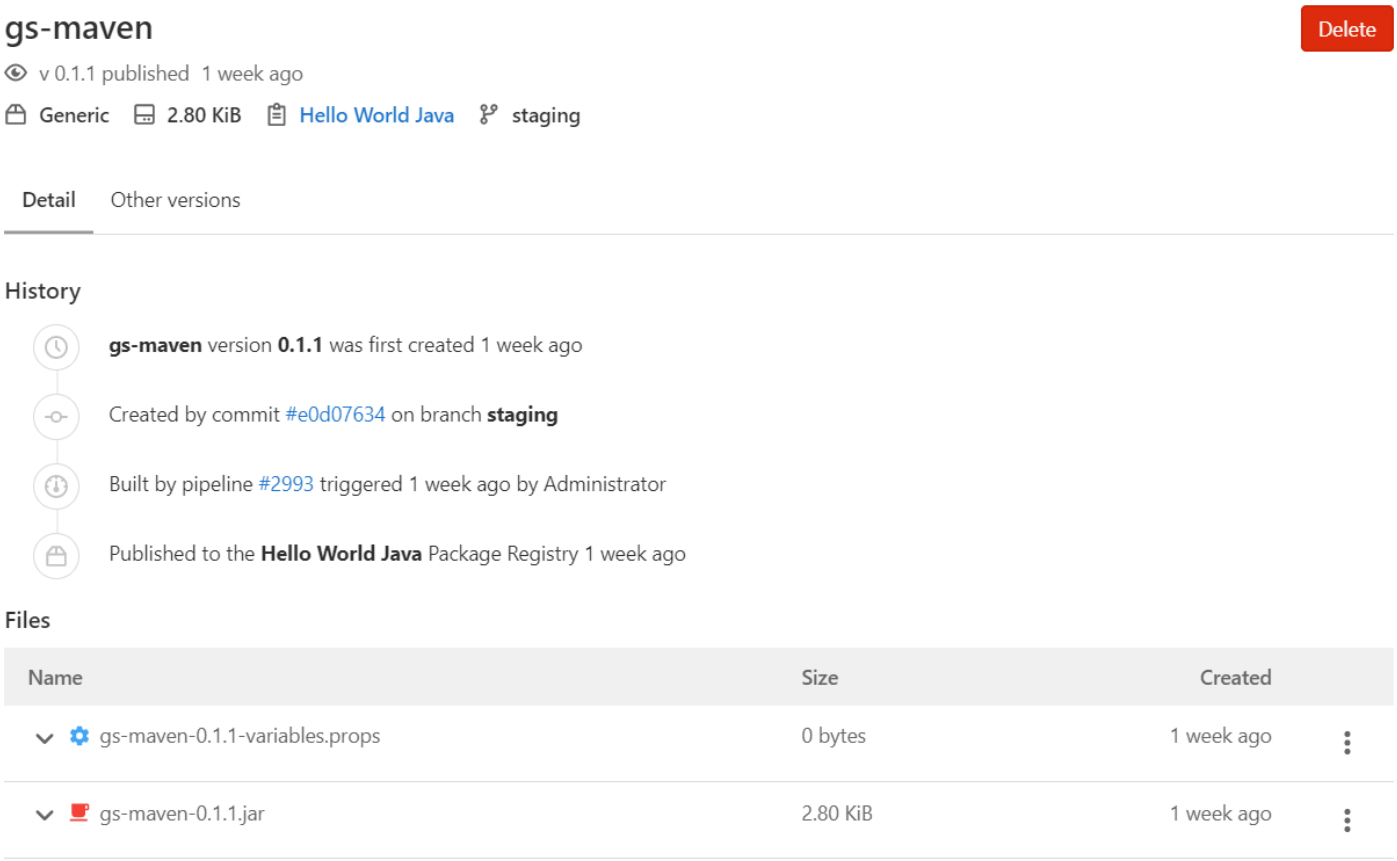


La pagina racchiude la lista delle release del singolo progetto.



Ogni release è associata ad una versione del progetto.

Inoltre, dalla pagina di dettaglio di ogni release si può risalire alle seguenti informazioni:



- Lo SHA del commit che ha scatenato la pipeline che ha caricato il pacchetto.
- L'identificativo della pipeline che ha caricato il pacchetto.
- Il branch per cui ha girato la pipeline.

Files

Per ogni versione sono caricati tutti i file necessari a riprodurre lo stato dell'ambiente considerato (Certificazione o Produzione).

L'**artefatto dispiegato** è sempre disponibile al download da parte dell'Incaricato, da parte di Regione Toscana e da parte di DXC. Quest'ultimo può scaricarlo da questa sezione al fine di effettuare il rilascio.

Il **file della configurazione** utilizzata per far girare la pipeline è caricato ogni volta che viene caricato l'artefatto, in modo da garantire la possibilità di poter ricostruire l'artefatto a partire dal codice sorgente.

Casi d'uso

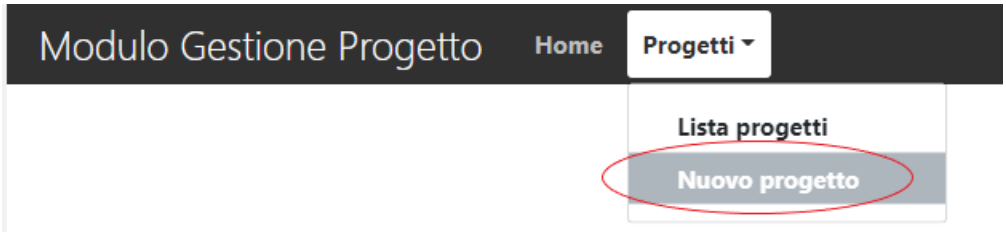
Si riportano gli esempi di casistiche più frequenti.

Creazione di un progetto senza codice sorgente

Una volta effettuato il primo login sulla piattaforma, è necessario recarsi presso Modulo Gestione Progetto per richiedere l'apertura del progetto.

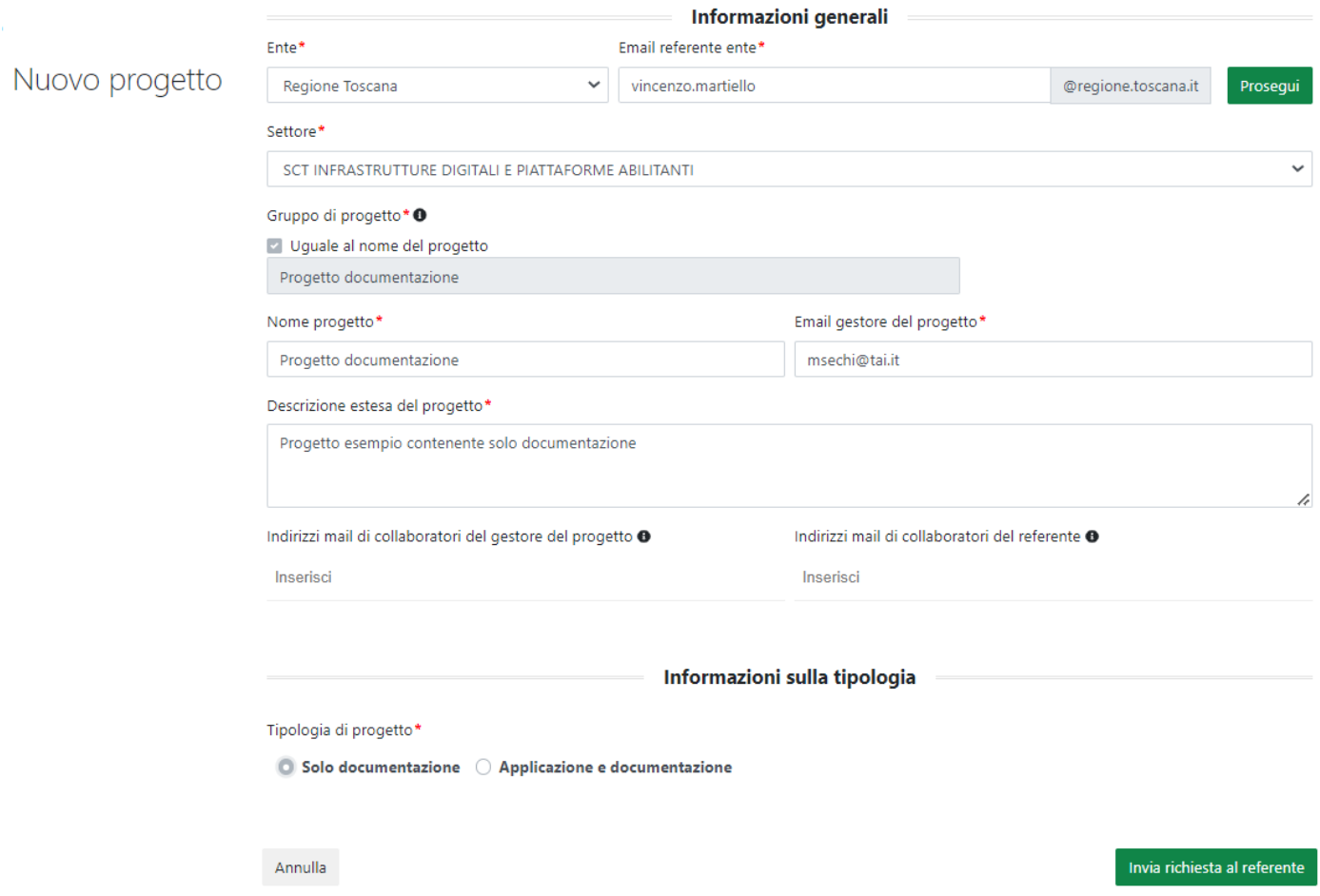
Richiesta di apertura

L'Incaricato si autentica sul Modulo Gestione Progetto e si reca alla pagina di apertura di un nuovo progetto.



Le operazioni da eseguire sono le seguenti:

1. Scelta dell'ente.
2. Scelta del Referente di Progetto.
3. Scelta del settore tra quelli associati al Referente.
4. Inserimento del nome del progetto e di una sua descrizione.
5. Eventuale modifica del gestore del progetto.
6. Selezione della tipologia del progetto. In questo esempio la tipologia è**Sola documentazione**.



Per approfondire consultare: Richiesta di apertura

Approvazione dell'apertura

Il Referente del progetto può accettare il progetto dalla sezione dedicata diModulo Gestione Progetto oppure rispondendo alla mail che ha ricevuto per l'approvazione dell'apertura.

Per approfondire consultare: Approvazione dell'apertura

Visualizzazione su Gitlab

Una volta approvato, il progetto sarà creato su Gitlab sotto un gruppo corrispondente al nome del settore scelto. Il Referente di progetto sarà l'Owner del progetto. L'utente inserito come Gestore del progetto sarà il Mantainer del progetto. Gli utenti indicati in fase di apertura del progetto che già possiedono un account su Gitlab saranno aggiunti come Developer al progetto.

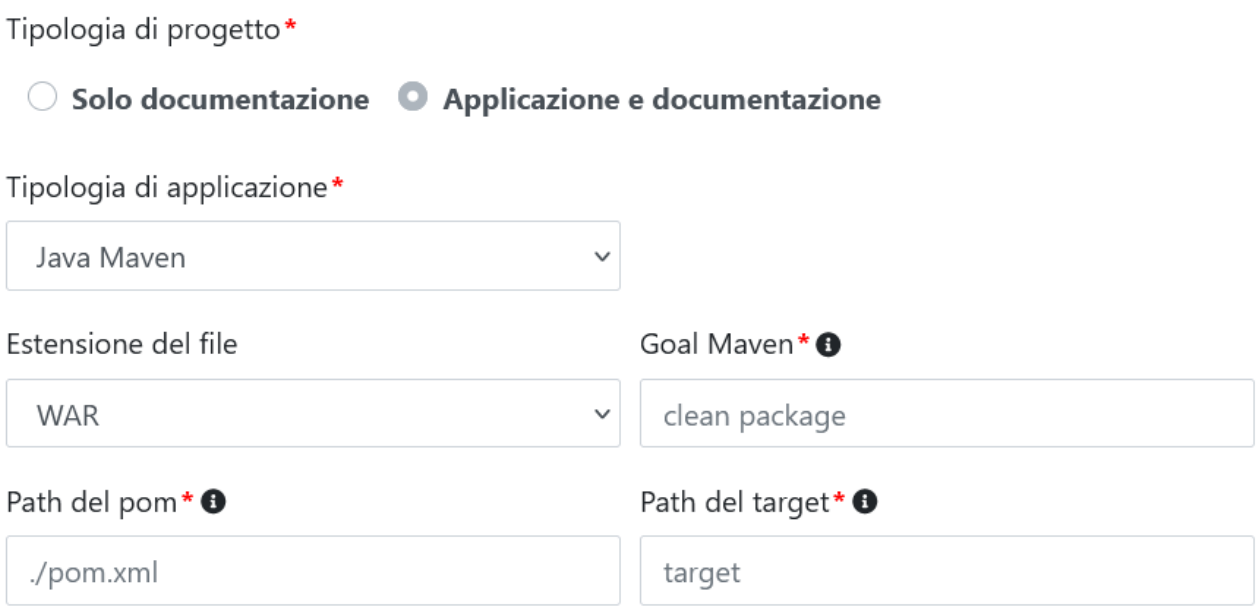
Autodeploy 3tier

La funzionalità di autodeploy permette all'utente di effettuare in autonomia il rilascio in ambiente certificazione e deve essere richiesta dall'utente dopo che è stato effettuato il primo rilascio.\ Ciò può avvenire già in fase di creazione progetto se si tratta di un'applicazione esistente che viene importata sulla piattaforma. Nel caso in cui l'applicazione non sia stata ancora rilasciata, non è possibile abilitare l'autodeploy in fase di creazione del progetto e occorre richiederne l'attivazione al Supporto Oscat in seguito al primo deploy.

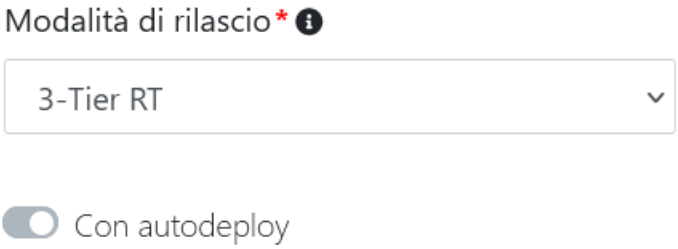
Richiesta di apertura

Le operazioni da eseguire sono le seguenti:

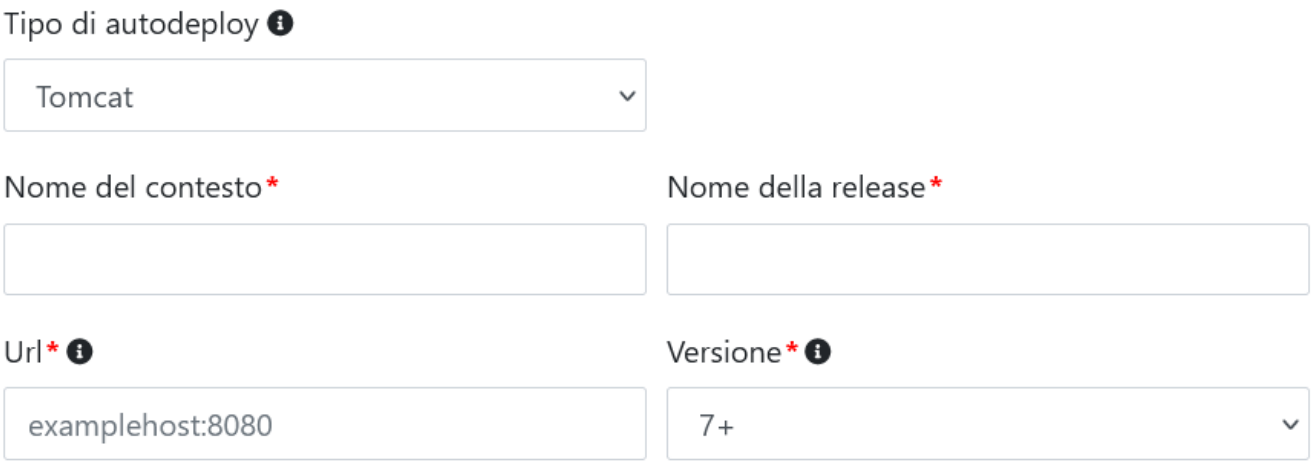
1. Scelta dell'ente.
2. Scelta del Referente di Progetto
3. Inserimento del nome del progetto e di una sua descrizione
4. Eventuale modifica del gestore del progetto
5. Selezione della tipologia del progetto, scegliendo **Applicazione e documentazione**
6. Inserire le informazioni riguardanti la compilazione.



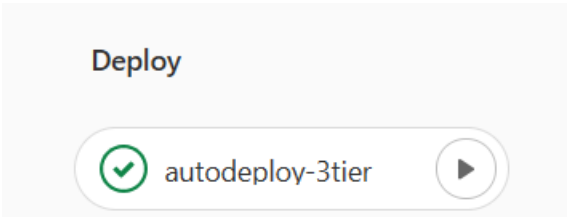
7. Scegliere la modalità di rilascio 3-Tier e nel caso l'applicazione sia già stata dispiegata abilitare l'autodeploy.



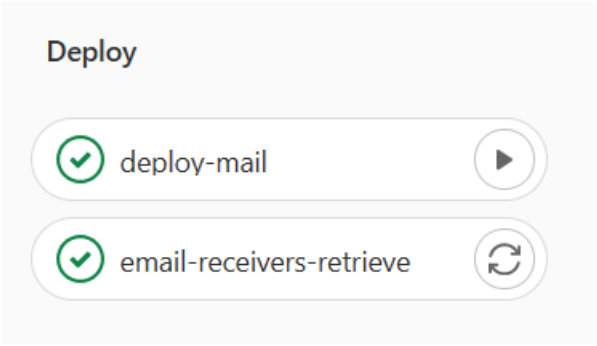
8. Scegliere l'application server da utilizzare e inserire le informazioni riguardanti l'applicazione e l'aimbiente di dispiegamento.



Una volta che le informazioni sono state inserite e la funzionalità è stata attivata, la pipeline del progetto avrà come ultimo passo il job di autodeploy



Fino a che la funzionalità non viene attivata lo stage di Deploy contiene solo i job per la richiesta di rilascio manuale.



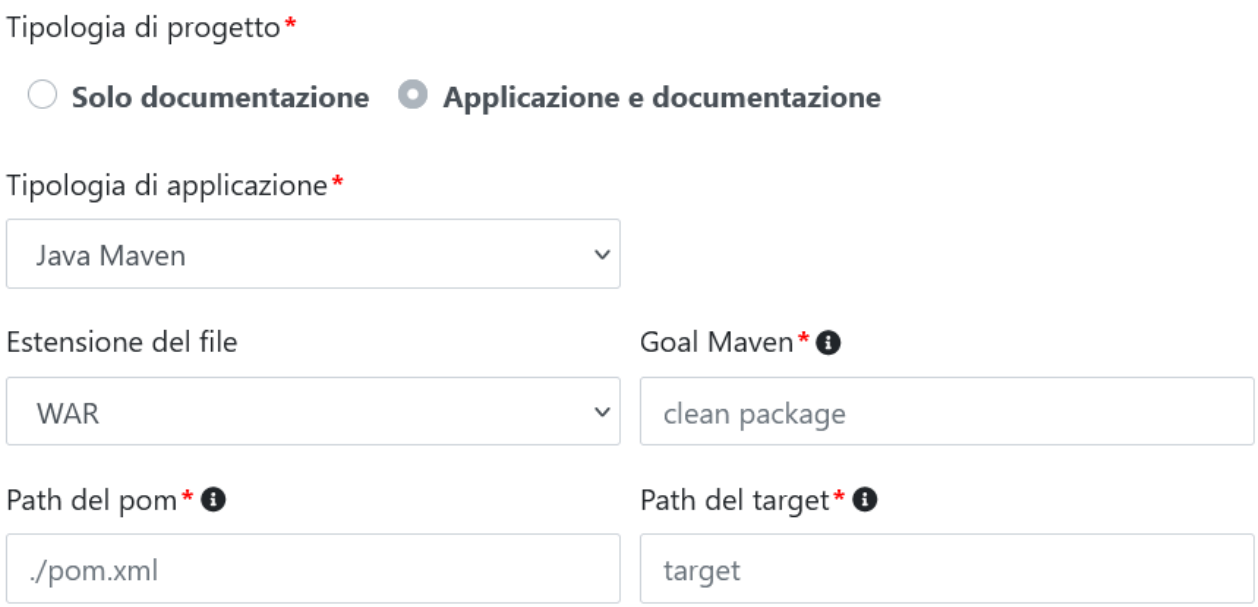
Autodeploy Docker Cochise/Swarm

La funzionalità di autodeploy permette all'utente di effettuare in autonomia il rilascio in ambiente certificazione e deve essere richiesta dall'utente dopo che è stato effettuato il primo rilascio.\ Ciò può avvenire già in fase di creazione progetto se si tratta di un'applicazione esistente che viene importata sulla piattaforma. Nel caso in cui l'applicazione non sia stata ancora rilasciata, non è possibile abilitare l'autodeploy in fase di creazione del progetto e occorre richiederne l'attivazione al Supporto Oscat in seguito al primo deploy.

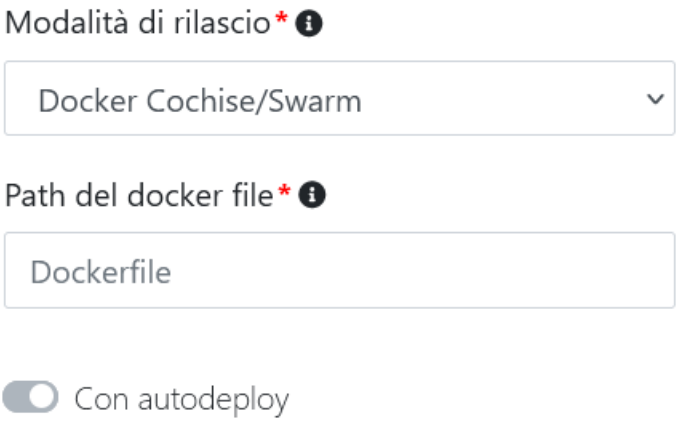
Richiesta di apertura

Le operazioni da eseguire sono le seguenti:

1. Scelta dell'ente.
2. Scelta del Referente di Progetto
3. Inserimento del nome del progetto e di una sua descrizione
4. Eventuale modifica del gestore del progetto
5. Selezione della tipologia del progetto, scegliendo **Applicazione e documentazione**
6. Inserire le informazioni riguardanti la compilazione.



7. Scegliere la modalità di rilascio Docker/Swarm e nel caso l'applicazione sia già stata dispiegata abilitare l'autodeploy.



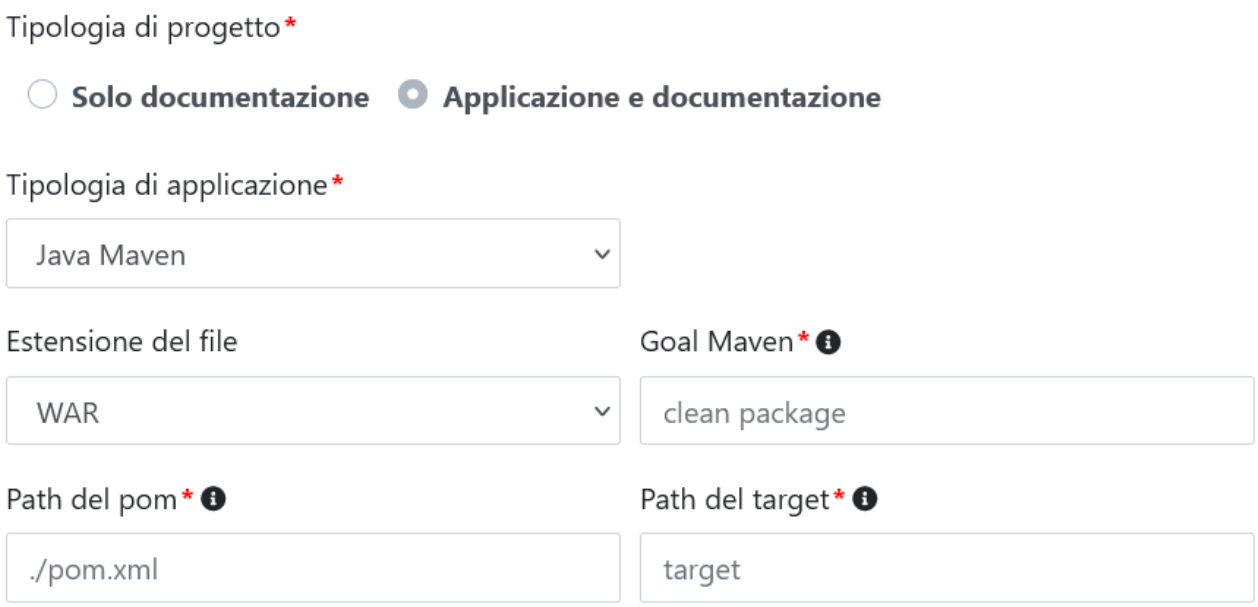
Autodeploy Nexus

La funzionalità di autodeploy permette all'utente di effettuare in autonomia il rilascio della libreria sul Nexus RT nel caso in cui l'analisi del codice superi il livello minimo di qualità richiesto.

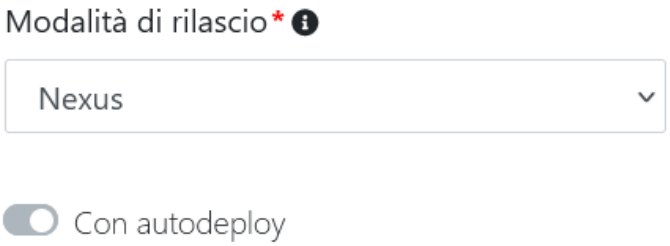
Richiesta di apertura

Le operazioni da eseguire sono le seguenti:

1. Scelta dell'ente.
2. Scelta del Referente di Progetto
3. Inserimento del nome del progetto e di una sua descrizione
4. Eventuale modifica del gestore del progetto
5. Selezione della tipologia del progetto, sciegliendo **Applicazione e documentazione**
6. Inserire le informazioni riguardanti la compilazione.



7. Scegliere la modalità di rilascio Nexus



In questo caso la funzionalità di autodeploy viene abilitata automaticamente e non è richiesta nessuna informazione aggiuntiva all'utente.

FAQ

E' possibile creare un progetto con sola documentazione?

Si, il Modulo Gestione Progetto richiede di selezionare la tipologia di progetto durante la fase di creazione (per maggiori info vedi Creazione di un progetto senza codice sorgente).

Il progetto di cui mi sto occupando ha sia documentazione privata che pubblica. Come devo procedere?

L'indicazione è di creare due progetti:

- uno pubblico (per cui è necessario chiedere al supporto OSCAT) in cui saranno messe le info destinate a tutti;
- uno privato in cui depositare quanto deve rimanere accedibile alle sole utente abilitate.

Come posso aggiungere/rimuovere un utente dal progetto?

Il *mantainer* (o l'*owner*) del progetto può modificare i membri di un progetto con Modifica membri del progetto.

E' possibile dispiegare in certificazione branch diversi da quello di stage?

Così come il master è il branch deputato ad essere dispiegato in produzione, quello di stage è l'unico candidato a raggiungere la certificazione. La necessità di dispiegare diverse versioni richiederebbe la presenza di ulteriori ambienti (sviluppo, pre-produzione o altro) inoltre, la versione in uso del SonarQube ha il limite di poter analizzare un solo branch - che è il master (al momento, esteso attraverso una customizzazione anche al branch di stage).

Come posso cancellare una release?

La cancellazione di una release è un'operazione che non può essere svolta in autonomia poiché gli artefatti dispiegati devono essere necessariamente conservati. Poiché non esiste un meccanismo per distinguere gli artefatti dispiegati dagli altri, l'operazione di cancellazione richiedere di contattare il supporto.