

# ASPI SecDevOps

Pratiche di automazione per standard di sicurezza e qualità nella trasformazione digitale

**Università degli Studi di Padova, 27 Maggio 2022**

# Il Contesto



**Lorenzo Carlà**

**IT Enterprise Architect - Autostrade per l'Italia**

- Laureato nel 2013 in **Ingegneria delle Telecomunicazioni** all'Università di Firenze
- Per i due anni successivi ho svolto lavoro di ricerca sui metodi di trasmissione di contenuti multimediali su reti **LTE-Advanced**
- Approdo in Autostrade nel 2015 come **System Engineer** dove mi sono occupato della gestione e progettazione dei sistemi IT aziendali Unix/Linux, con particolare attenzione rivolta ai sistemi di traffico, viabilità e ai sistemi IT Telepass e Telepass Pay.
- Dal 2020 sono **IT Enterprise Architect** e mi occupo di:
  - Progettazione e definizione delle architetture a supporto delle iniziative di **Digital Transformation** in ambito Cloud AWS
  - Responsabile degli Stream di sviluppo in ambito **SecDevOps** e **Integrazione A2A** (Application-to-Application)
  - Membro del **CCoE** (Cloud Center of Excellence) AWS di Autostrade per l'Italia

# Il Contesto

...erogati tramite piattaforme tecnologiche eterogenee...



## Il Cashback per i tuoi ritardi in autostrada

Ti rimborsiamo fino al **100%** del pedaggio a partire da **10 minuti** di ritardo dovuto a cantieri per lavori sulla rete di Autostrade per l'Italia.

**NOVITÀ** Registrando la tua targa, o il tuo dispositivo di telepedaggio, ottieni il rimborso in automatico ogni volta che ne hai diritto.



Autostrade per l'Italia  
La rete del gruppo Autostrade è  
gestita dalla Società Autostrade per  
Italia e le sue controllate:

- AUTOSTRADA DEL TIRRENO
- TRAIRES MONTE BIANCO (SHE)
- R.A.V.
- Tangenziale di Napoli
- S.A.T.

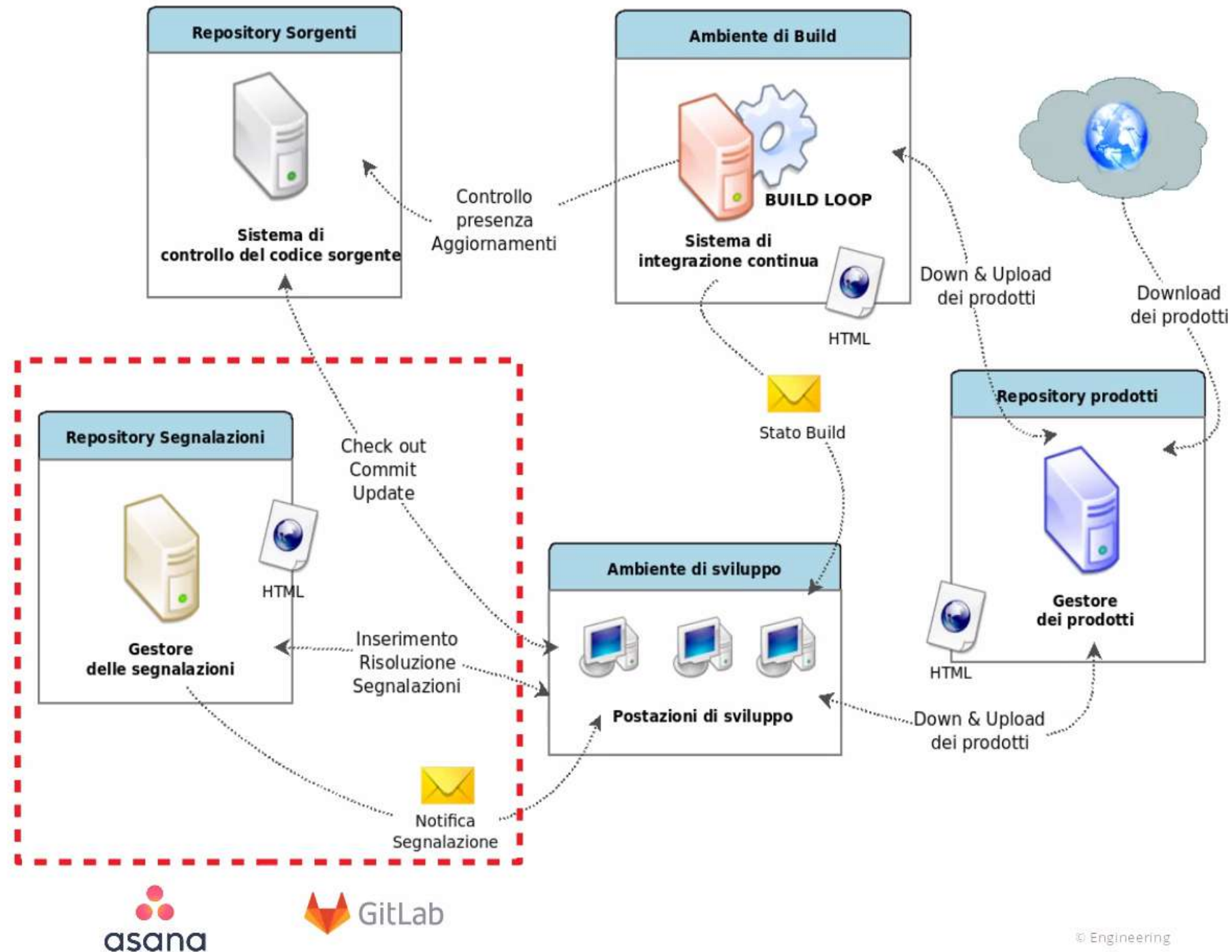


# L'esigenza

- Autostrade per L'Italia (ASPI) ha avviato da tempo un processo di trasformazione digitale
- Dispone di una propria divisione IT
- Sviluppa internamente le proprie soluzioni IT, collaborando anche con altre aziende partner (come Engineering)
- Ogni soluzione IT può essere costituita da più componenti tecnologicamente eterogenee (filieri): app mobile Android/iOS, soluzioni Java, Python, Kotlin, Cobol, Angular, ecc ecc
- Ogni soluzione IT dispone di più ambienti (sviluppo, test, quality, collaudo), a supporto delle attività di test e collaudo che precedono il rilascio definitivo nell'ambiente di produzione
- Ha predisposto un'infrastruttura affinché tutti i gruppi di sviluppo interni ed esterni lavorino secondo i medesimi standard di qualità, grazie all'uso di strumenti comuni, configurati opportunamente
- L'infrastruttura e gli standard di qualità sono definiti da un ufficio apposito, guidato da Lorenzo Carlà, con l'ausilio di personale Engineering
- Il team che si occupa di sviluppare le configurazioni e i servizi a supporto dei team di sviluppo è denominato Team SecDevOps: questo nome pone esplicitamente l'accento sul tema della sicurezza, promuovendo le più moderne pratiche di sviluppo, che vanno sotto il nome di DevOps, in cui la standardizzazione e l'automazione giocano un ruolo centrale (la seconda non sarebbe possibile senza la prima)
- Gli ambiti di standardizzazione e automazione? ITS, VCS, TEST, ARTIFACT REPOSITORY, BUILD AUTOMATION, CI, CD, CM, per citarne alcuni tra i più importanti

# ITS

- Due livelli di tracciamento delle attività:
  - Sprint (pianificazione e gestione attività durante le iterazioni di progetto)
  - Tracciabilità bidirezionale di dettaglio (dalla issue al codice e ritorno)
- Il team SecDevOps adotta entrambi i livelli
  - Asana (pianificazione e gestione attività)
  - Gitlab Issues (analoghe a GitHub Issues) per tracciabilità bidirezionale





ITS

Gitlab Issue

The image displays two overlapping software interfaces. The background interface is Asana, showing a project board for 'DSSEE - SecDevOps - Sviluppo nuova filiera E2E SecDevOps'. The board is divided into columns: 'Task on demand - Backlog', 'Task progettuali - Backlog', and 'Done'. A task titled 'TTA - AWS Serverless - Java - sviluppo nuova pipeline' is highlighted with a red circle. The foreground interface is GitLab, showing the 'Issue Boards' for the 'aspi-jenkins-library' project. The boards are organized into columns: 'Open', 'To Do', 'Doing', 'Done', and 'Closed'. An issue titled 'feature/new-aws-serverless-java-maven-pipeline' is highlighted with a red circle. The GitLab interface also shows a sidebar with navigation options like 'Project information', 'Repository', 'Issues', 'Boards', 'Service Desk', 'Milestones', 'Merge requests', 'CI/CD', 'Security & Compliance', 'Deployments', 'Monitor', 'Infrastructure', 'Packages & Registries', 'Analytics', 'Wiki', 'Snippets', and 'Settings'.

Asana Card

Tracciabilità bidirezionale tra  
issue e codice sorgente

**GitLab** Menu

**aspi-jenkins-library**

- Project information
- Repository
- Issues** 7
  - List
  - Boards
  - Service Desk
  - Milestones
- Merge requests 0
- CI/CD
- Security & Compliance
- Deployments
- Monitor
- Infrastructure
- Packages & Registries
- Analytics

SYSUNIX > jenkins > aspi-jenkins-library > Issues > #67

**Open** Created 1 week ago by **Gagliardi, Daniele** Maintainer

## feature/new-aws-serverless-jav

Occorre sviluppare una nuova pipeline per il deploy di lambda AW

Drag y

Linked issues 0 +

0 0

**Gagliardi, Daniele** @87006896 mentioned in commit [78266fc2](#) 4 days ago

**Gagliardi, Daniele** @87006896 mentioned in commit [3e516323](#) 4 days ago

**Gagliardi, Daniele** @87006896 mentioned in commit [82402459](#) 4 days ago

SYSUNIX > jenkins > aspi-jenkins-library > Commits > **78266fc2**

Commit **78266fc2** authored 4 days ago by **Gagliardi, Daniele** [Browse files](#) [Options](#)

## #67 aggiunto supporto esplicito a branch main

parent [3e516323](#) [feature/67-new-aws-serverless-java-maven-pipeline](#)

No related merge requests found

**Changes** 1

Showing **1 changed file** with **8 additions** and **8 deletions** [Hide whitespace changes](#) [Inline](#) [Side-by-side](#)

[src/it/autotrade/jenkinssharedlibrary/builds/MavenJavaProjectBuild.groovy](#) [View file @78266fc2](#)

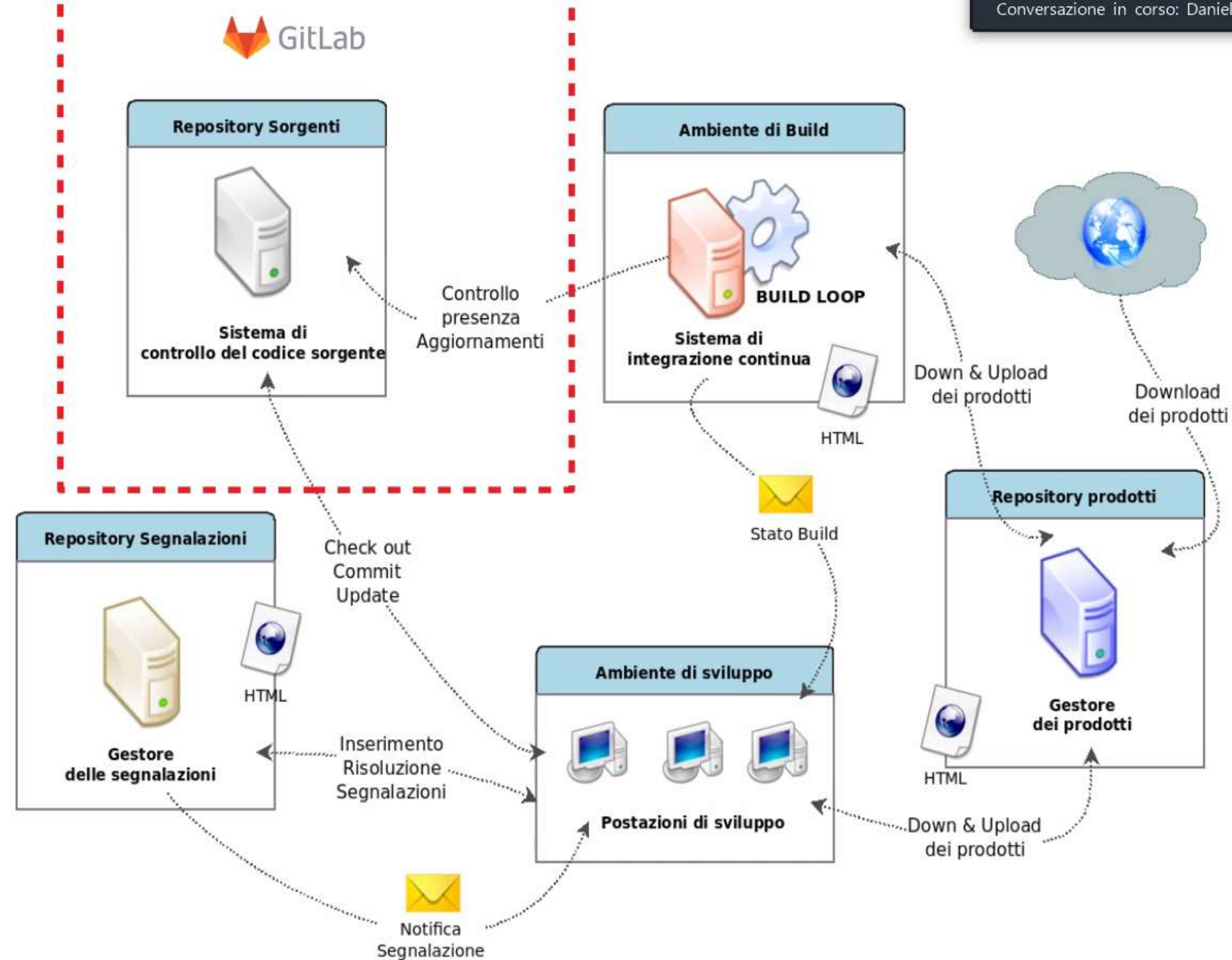
```
... 39 39 @@ -39,8 +39,8 @@ class MavenJavaProjectBuild implements IBuild, Serializable, IUrbancodeDeploy {
... 48 48     if (pomVersion.isEmpty()) {
... 41 41         throw new BuildFailedException("POM Property project.version cannot be empty")
... 42 -     if ("${this.workspaceContext.env.GIT_BRANCH}" == "master" && pomVersion.contains("-SNAPSHOT")) {
... 43 -         throw new BuildFailedException("POM Version cannot contain -SNAPSHOT on Master branch")
... 42 +     if (( "${this.workspaceContext.env.GIT_BRANCH}" == "master" ||
... 43 +         "${this.workspaceContext.env.GIT_BRANCH}" == "main" ) && pomVersion.contains("-SNAPSHOT")) {
... 44 44         throw new BuildFailedException("POM Version cannot contain -SNAPSHOT on Master/main branch")
... 44 44     }
```

#67 aggiunto supporto esplicito  
a branch main



# VCS

- Ospita i sorgenti di tutto il parco software ASPI
- Ospita i sorgenti delle pipeline SecDevOps, che supportano i processi di sviluppo e rilascio del parco software ASPI
- Ospita i sorgenti degli agenti utilizzati per i processi CI/CD





# VCS

- Informazioni generali di progetto
- Accesso sorgenti
- Supporto multi-branch
- Supporto merge request (equivalenti alle pull request in GitHub)
- Accesso ITS integrato

GitLab interface for the repository **aspi-jenkins-library** (Project ID: 1344).

Statistics: 1,793 Commits, 81 Branches, 12 Tags, 3.9 MB Files, 3.9 MB Storage. Groovy 45.3%.

Navigation sidebar (left):

- Project information
- Repository
- Issues (7)
- Merge requests (0)
- CI/CD
- Security & Compliance
- Deployments
- Monitor
- Infrastructure
- Packages & Registries
- Analytics
- Wiki
- Snippets
- Settings

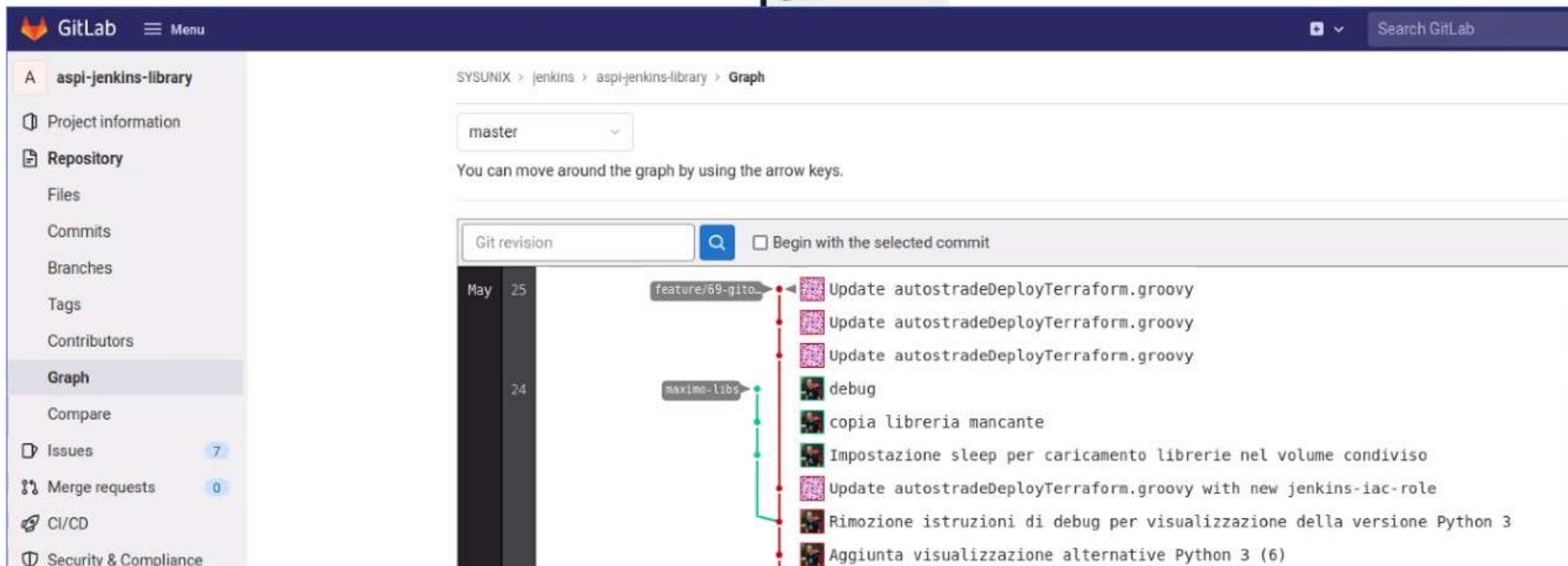
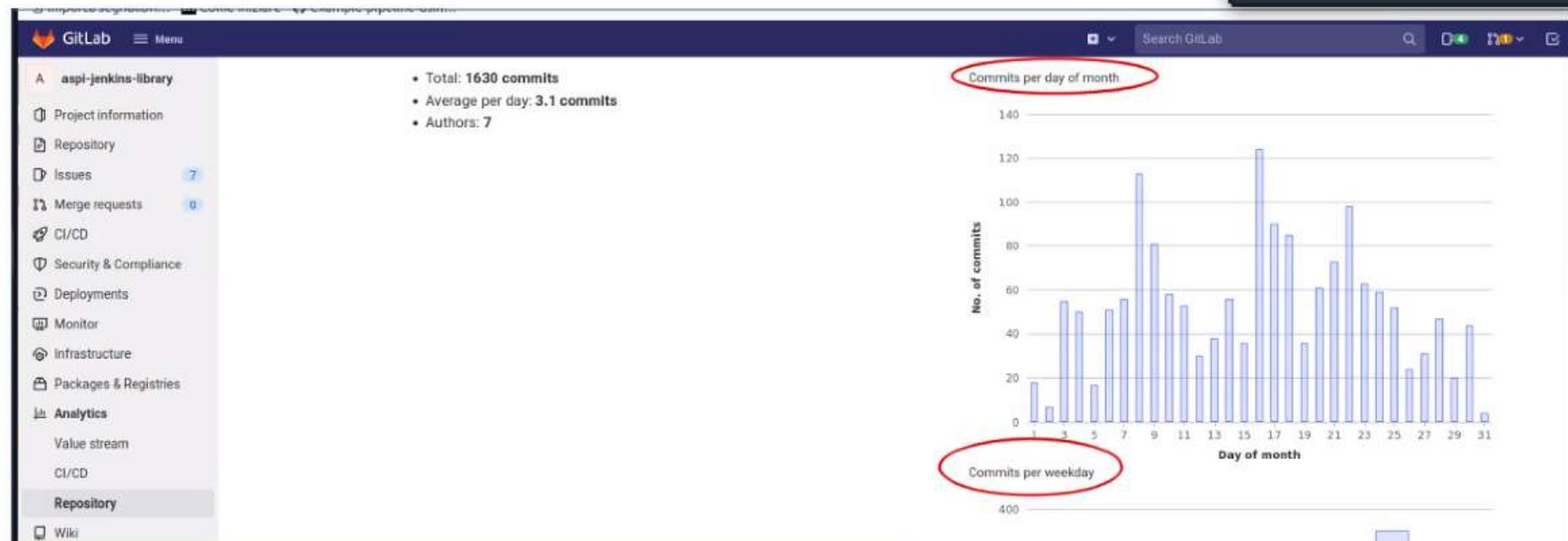
Recent commit: **Update autostradeDeployTerraform.groovy** by Carla, Lorenzo (authored 2 hours ago). Commit hash: 4bb5f5e7.

File list table:

Name	Last commit	Last update
bin	add terraform credential	4 weeks ago
docs	Update docs/PMD_TEST.md, docs/APPSCA...	9 months ago
gradle/wrapper	first commit	1 year ago
src/it/autostrade/jenkinssharedli...	Merge remote-tracking branch 'origin/master'	4 days ago
test	Fix #59 Invio notifiche security ASPI circa sc...	1 week ago
vars	Update autostradeDeployTerraform.groovy	2 hours ago
.gitignore	first version for AWS Infra ECS	7 months ago
CONTRIBUTING.md	add test for Openshift and README file	1 year ago
Jenkinsfile	fix	1 year ago

# VCS

- Workflow  
(visualizzazione del workflow pattern adottato: GitFlow GitHub Flow, ecc)
- Statistiche attività

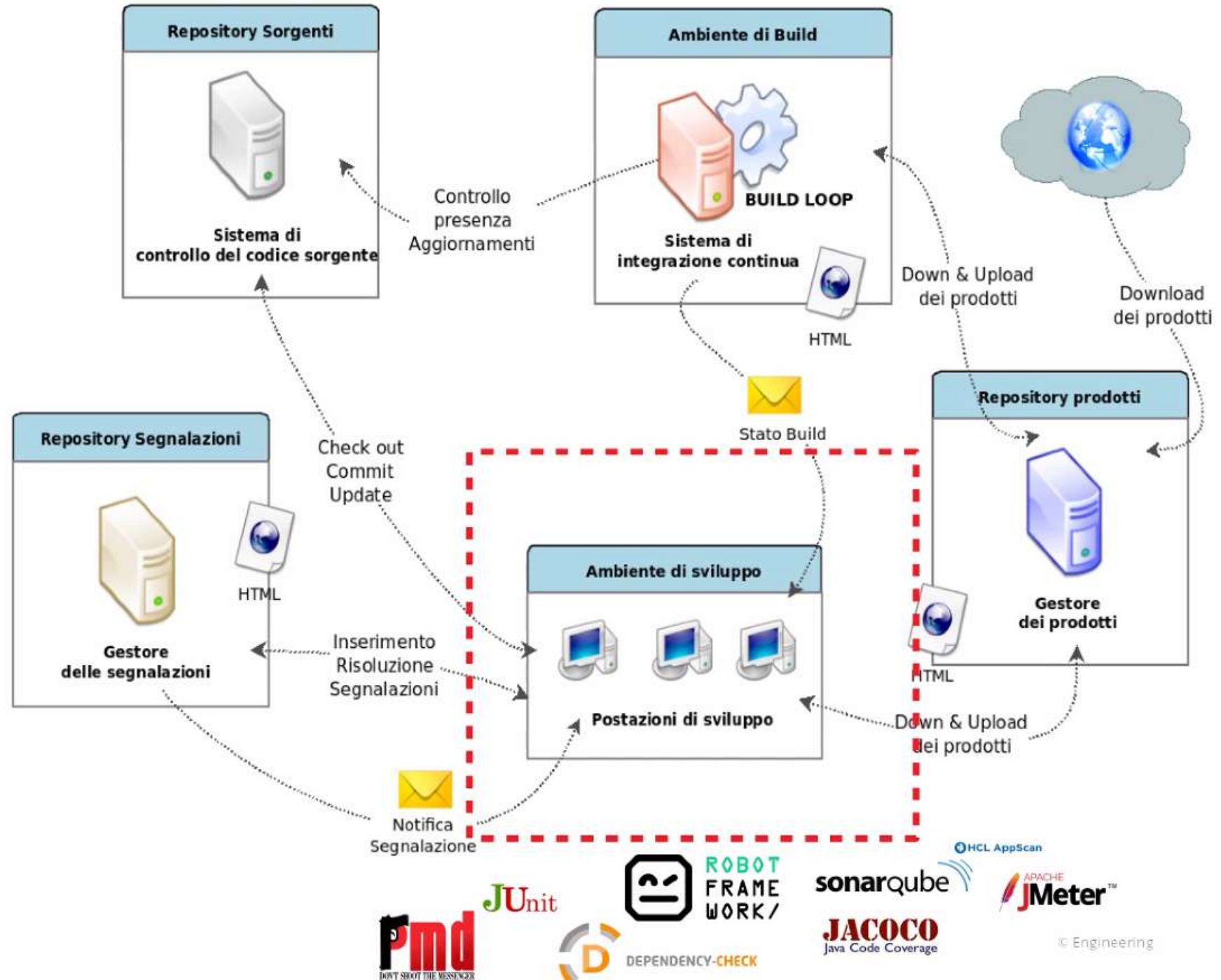


Commits per day hour (UTC)



# TEST

- Tutti i team progettano ed eseguono test, secondo i livelli e i tipi stabiliti dalla propria strategia di test
- Sono supportati da un CoE (Center of Excellence) trasversale esperto in software testing
- Livelli coperti: unit, integration, system, collaudo
- Tipi di test: funzionali, sicurezza, performance, qualità del codice, non regressione




# TEST

- Durante l'esecuzione di unit e integration test, è calcolato automaticamente anche il code coverage
- Per progetti Java lo strumento utilizzato è Jacoco (strumento open source)
- Progetti Salesforce hanno uno strumento integrato che calcola la copertura


https://autostrade.it/job/Free2X/job/HRP/job/hrp-common-avro-jar/job/master/29/


Come iniziare example-pipeline-usin...

HRP > HRP / hrp-common-avro-jar > master > #29


 The following steps that have been detected may have insecure interpolation of sensitive variables ([click here for an explanation](#)):

- sh: [GIT\_PASSWORD, GIT\_USER]

 **Test Result** (nessun errore)

 **Jacoco - Overall Coverage Summary**

INSTRUCTION	69%	<div><div></div></div>
BRANCH	63%	<div><div></div></div>
COMPLEXITY	73%	<div><div></div></div>
LINE	67%	<div><div></div></div>
METHOD	89%	<div><div></div></div>
CLASS	86%	<div><div></div></div>

 **Jacoco - Overall Coverage Summary**

INSTRUCTION	69%	<div><div></div></div>
BRANCH	63%	<div><div></div></div>
COMPLEXITY	73%	<div><div></div></div>
LINE	67%	<div><div></div></div>
METHOD	89%	<div><div></div></div>
CLASS	86%	<div><div></div></div>

17ed06382f90e  
de.it/hrp/hrp-common-avro-jar.git

SALESFORCE > DCR / sforg > Merge Requests (112) > MR-1230 > #2

Test Success [8]

Name	Method
JTUserProvisioningHandlerTest	createUserTest
DeactiveUsersTest	deactiveUsersExecuteTest
coe.NoteTriggerHandlerTest	onAfterInsertTest
DeactiveUsersTest	runUsersDeactivationK0Test
DeactiveUsersTest	runUsersDeactivationTest
JTUserProvisioningHandlerTest	setPSUserInfoTest
DeactiveUsersTest	testSendEmail
JTUserProvisioningHandlerTest	updateUserTest

**Apex Code Coverage**

Name	% Covered	Uncovered Lines
DeactiveUsers	83%	66,70,71,72,73,74,75,104
JTUserProvisioningHandler	83%	140,151,154,241,242,277,278,289,322,333,334,348,376,384,385,386,387,413,422,423,427,428,430,431,436,437,438,439,469,493,502,523,526,555,572,573,575
NoteTrigger	100%	

Total Test Time: 10050.0  
[Pipeline] updateGitLabCoverageStatus



# TEST

- I system test funzionali automatici sono eseguiti dalle postazioni degli sviluppatori o in pipeline dedicate
- Robot Framework è lo strumento per la progettazione ed esecuzione dei test
- Al termine dell'esecuzione è generato un report di dettaglio sull'esito dei test
- In caso di errori il report riporta anche una schermata per semplificare l'indagine sul bug rilevato

https://autostrade.it/job/SALESFORCE/job/Test Automation/job/Test funzionali automatici TGD - Test Su

Importa segnalibri...

Come iniziare

example-pipeline-usin...

# TEST

- Il codice sorgente è sottoposto a SAST (Static Application Security Testing) tramite HCL AppScan
- Le eventuali dipendenze/librerie sono a loro volta sottoposte ad analisi di vulnerabilità, tramite OWASP Dependency Checker
- I test di sicurezza dinamica (system security test: vulnerability assessment, penetration testing, ecc) sono effettuati da una struttura dedicata di esperti di sicurezza informatica

https://autostrade.it/job/SALESFORCE/job/sforg/job/master/54/consoleFull

libri... Come iniziare example-pipeline-usin...

SALESFORCE ▸ DCR / sforg ▸ master ▸ #54

```
Scanning tru_Approved2SentToSAPSchedulertest.cls (278 of 286)
Scanning BatchDailyAsr_Detection.cls (279 of 286)
Scanning fsl_SuppliersRegistryCallout.cls (280 of 286)
Scanning fsl_MockInterface.cls (281 of 286)
Scanning fsl_FinalBalanceInvocable.cls (282 of 286)
Scanning BatchUpdateAsr_DetectionTEST.cls (283 of 286)
Scanning cfa_TkpApiGatewayControllerErrorMock.cls (284 of 286)
Scanning AgentWorkTrigger_TEST.cls (285 of 286)
Scanning aas_TestDataFactory.cls (286 of 286)
Applying machine learning...
Scanned application DCR : File(s) scanned: 286 Lines scanned: 33554 Total findings: 548
Scan completed: File(s) scanned: 286 Lines scanned: 33554 Total findings: 548
Elapsed Time - 0 Hour(s) 2 Minute(s) 2 Second(s)
```

```
-----
Total Call Sites: 548
Total Definitive Security Findings with High Severity: 0
Total Definitive Security Findings with Medium Severity: 548
Total Definitive Security Findings with Low Severity: 0
Total Suspect Security Findings with High Severity: 0
Total Suspect Security Findings with Medium Severity: 0
Total Suspect Security Findings with Low Severity: 0
Total Scan Coverage Findings with High Severity: 0
Total Scan Coverage Findings with Medium Severity: 0
Total Scan Coverage Findings with Low Severity: 0
```

autostrade.it: /project/extension/dependencycheck/report\_page?id=it:autostrade:trf%3ATIS&qualifier=TRK

Importa segnalibri... Come iniziare example-pipeline-usin...

sonarcube Projects Issues Rules Quality Profiles Quality Gates

TIS master

Last analysis had 2 warnings May 5, 2022, 6:12 PM Version: 4.4.14-SNAPSHOT

Overview Issues Security Hotspots Measures Code Activity More

**DEPENDENCY-CHECK**

How to read the report | Suppressing false positives | Getting Help: [github issues](#)

Sponsor

Project: TIS

it:autostrade:trf:TIS-4.4.14-SNAPSHOT

Scan Information (show all)

- dependency-check version: 8.3.2
- Report Generated On: Fri, 6 May 2022 16:12:11 GMT
- Dependencies Scanned: 536 (644 unique)
- Vulnerable Dependencies: 152
- Vulnerabilities Found: 635
- Vulnerabilities Suppressed: 0

Summary

Display: [Showing Vulnerable Dependencies \(click to show all\)](#)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
<a href="#">org-1.4.jar</a>	<a href="#">CVE-2021-44228</a>	<a href="#">org.apache.commons:commons-lang3:3.12.0</a>	HIGH	4	Highest	14
<a href="#">junit-4.12.jar</a>	<a href="#">CVE-2021-44228</a>	<a href="#">org.apache.commons:commons-lang3:3.12.0</a>	CRITICAL	5	Highest	26

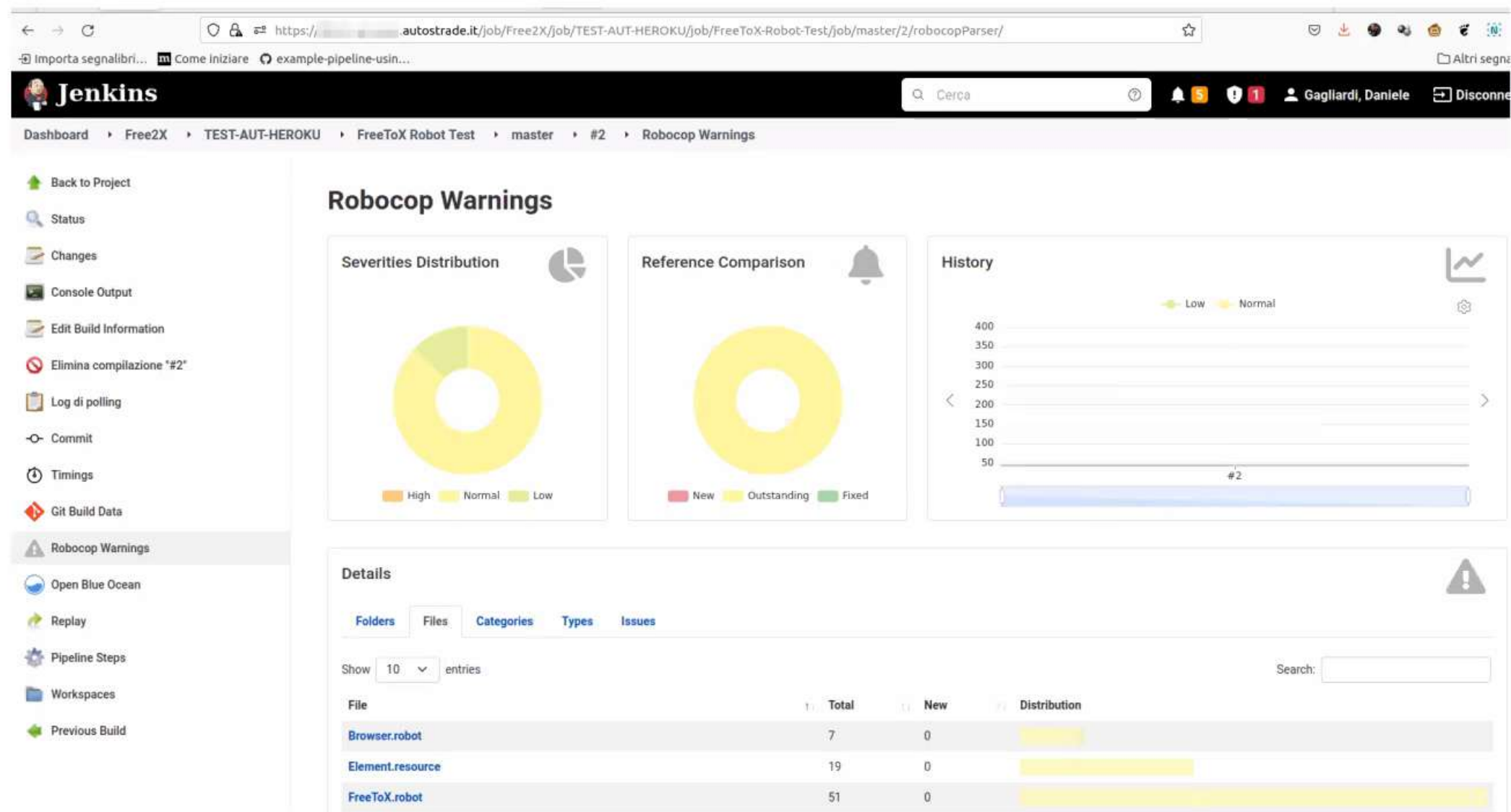


- Il codice sorgente è sottoposto a test di qualità
- SonarQube e PMD sono gli strumenti principali utilizzati
- La loro esecuzione è automatizzata: ad ogni modifica dei sorgenti nel VCS, vengono avviate le attività di verifica

The screenshot displays the SonarQube web interface for the 'autostrade.it' project. The left sidebar contains filters for Quality Gate (Passed: 40, Failed: 7), Reliability (Bugs: 8, 1, 22, 2, 14), Security (Vulnerabilities: 34, 0, 0, 3, 10), Security Review (Security Hotspots: ≥ 80%, 70% - 80%, 50% - 70%, 30% - 50%, < 30%), and Maintainability (Code Smells: 43, 4, 0, 0, 0). The main area shows a list of 47 projects. The 'hrp-common-avro-jar' project is highlighted, showing a 'Passed' status and a 'Coverage' of 66.0% (circled in red). Other projects shown include 'exa' (Passed), 'file-integration' (Failed), 'FTX\_be' (Passed), and 'hrp-common-dao-jar' (Passed).

Project	Status	Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
exa	Passed	2	0	-	35	18.7%	0.0%	422
file-integration	Failed	22	0	-	266	0.0%	0.0%	3.9k
FTX_be	Passed	8	0	-	324	0.0%	10.3%	2.7k
hrp-common-avro-jar	Passed	1	0	-	66	66.0%	0.0%	591
hrp-common-dao-jar	Passed	-	-	-	-	-	-	-

- Il codice sorgente è sottoposto a test di qualità
- SonarQube e PMD sono gli strumenti principali utilizzati
- La loro esecuzione è automatizzata: ad ogni modifica dei sorgenti nel VCS, vengono avviate le attività di verifica
- **Anche gli script di test funzionali automatici sono soggetti ad analisi di qualità, tramite un tool chiamato Robocop**





- Laddove necessario, vengono eseguiti test di performance (system test non funzionali) tramite lo strumento open source Apache JMeter
- L'esecuzione può essere effettuata automaticamente tramite il server Jenkins

The screenshot shows the Jenkins web interface for the 'JMeter Performance Test' job in the 'develop' branch. The breadcrumb trail is: Dashboard > Free2X > TEST-AUT-HEROKU > JMeter Performance Test > develop.

**Branch develop**  
Nome completo progetto: Free2X/TEST-AUT-HEROKU/JMeter Performance Test/develop

**Stage View**

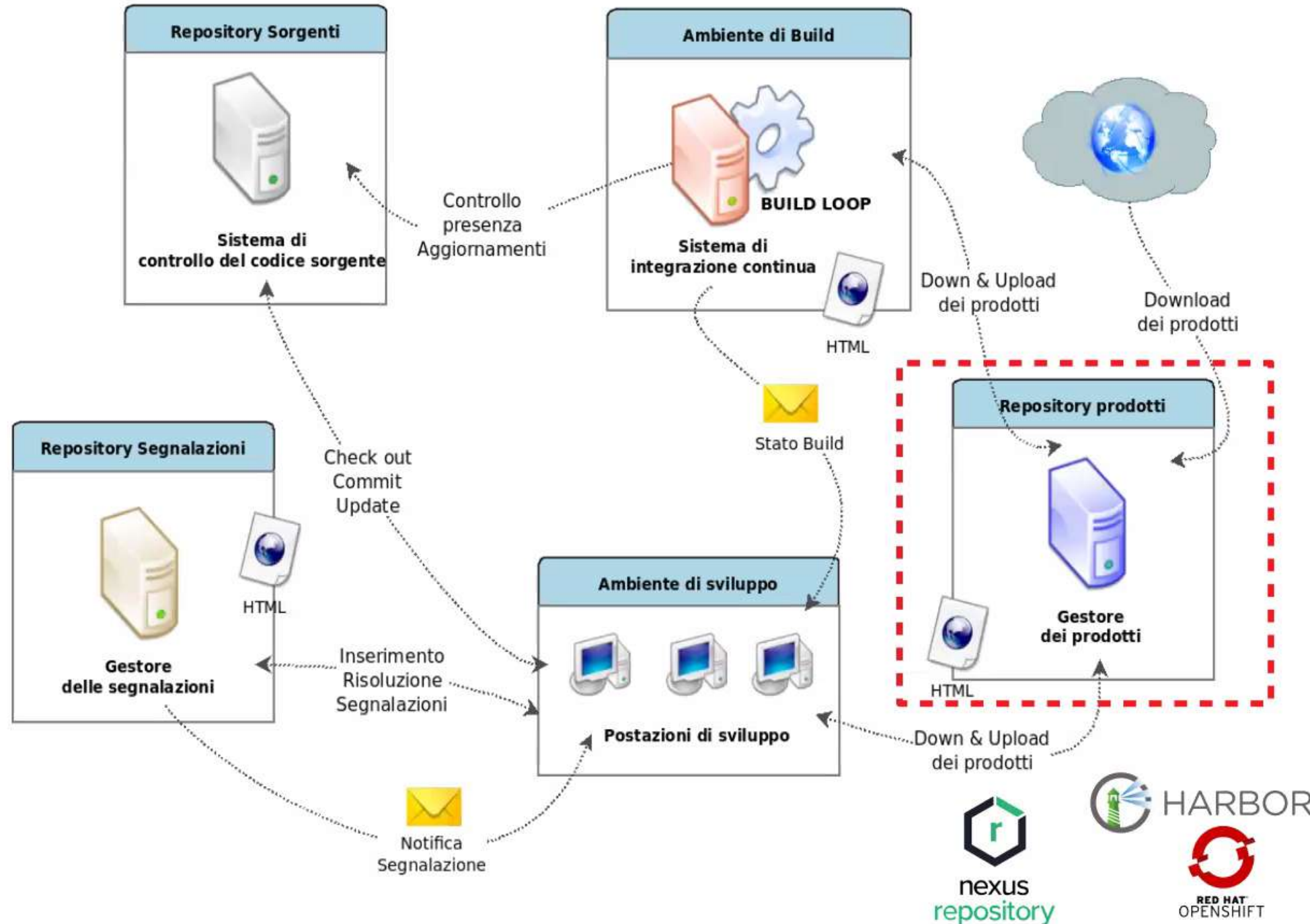
	Initialization and Clone	Build	Esecuzione Test	Generazione report
Average stage times: (Average full run time: ~3min 23s)	1s	707ms	1min 41s	463ms
#2 Mar 29 17:57 538 commits	1s	750ms	13s	108ms
#1 Dec 15 10:19 No Changes	1s	664ms	3min 8s	819ms

**Collegamenti permanenti**

- Ultima compilazione (#2), 1 m 27 g fa
- Ultima compilazione stabile (#1), 5 m 12 g fa
- Ultima compilazione riuscita (#1), 5 m 12 g fa
- Ultima compilazione non riuscita (#2), 1 m 27 g fa
- Ultima compilazione non riuscita (#2), 1 m 27 g fa
- Ultima compilazione completata (#2), 1 m 27 g fa

# ARTIFACT REPOSITORY

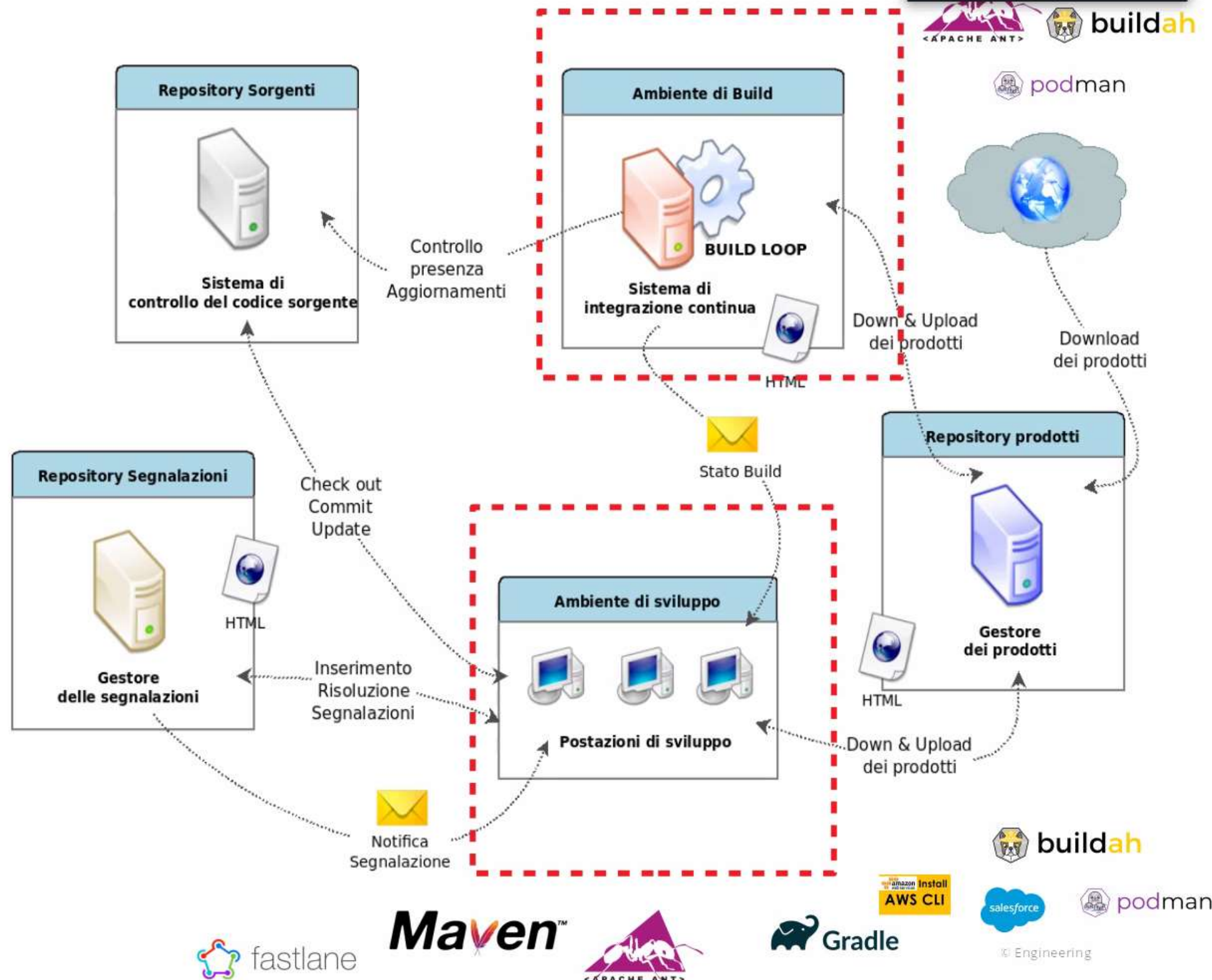
- Ogni filiera tecnologica (Java, Python, ecc) produce artefatti software (binari) che vengono ospitati in artifact repository
- L'artifact repository di riferimento è Nexus OSS
- Anche i container hanno un repository dedicato, detto container registry, basato su un'altra soluzione open source, chiamata Harbor
- Esiste poi un repository dedicato agli agent Jenkins resi disponibili come container, basato sulla soluzione Red Hat Openshift





# BUILD AUTOMATION

- Ogni filiera tecnologica ha i propri strumenti di build
- Maven, Ant, Gradle, Fastlane (per app mobile Android e iOS) sono i più diffusi
- Immagini di container Docker richiedono strumenti dedicati, come Buildah e Podman
- Sistemi proprietari, come Salesforce, hanno a loro volta i propri sistemi di build, solitamente strumenti a riga di comando
- Anche la produzione di pacchetti software destinati ad ambienti cloud è ottenuta tramite strumenti specifici, come la AWS CLI, strumento a riga di comando per l'ecosistema AWS
- Questi strumenti sono usati sia localmente dagli sviluppatori presso le proprie postazioni, sia centralmente dallo strumento di continuous integration



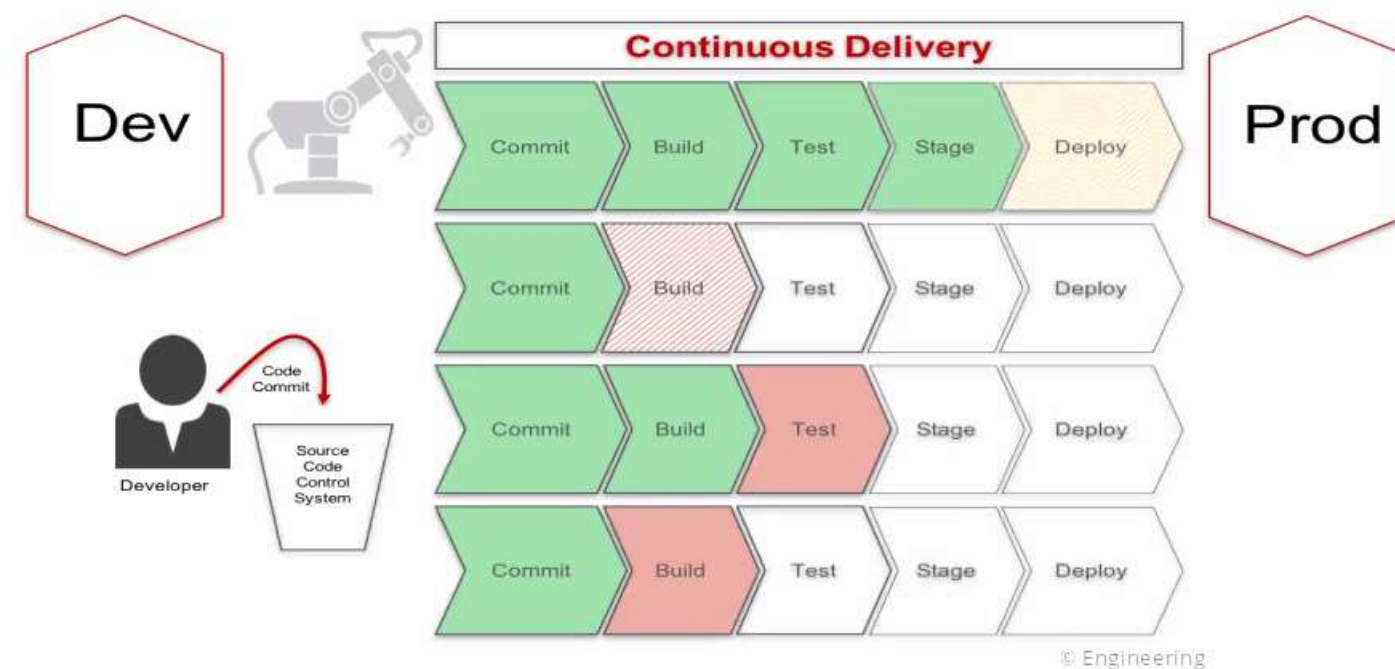


## CI/CD

- I processi di sviluppo per ciascuna filiera seguono standard ufficiali e best practice definite per la filiera stessa: utilizzo di un VCS e di una branch strategy definita, progettazione di unit e integration test che garantiscano copertura adeguata, best practice di codifica, promosse dall'uso di strumenti di analisi statica, progettazione di system test funzionali e non funzionali che garantiscano il rispetto dei requisiti di prodotto, standard di versionamento e di struttura del pacchetto software per la conservazione in artifact repository e per il rilascio nei diversi ambienti, secure coding e best practice di sicurezza supportati da strumenti specifici, e così via
- L'uso di Jenkins, un build-automation server open source, garantisce ulteriormente il rispetto degli standard e delle best practice, imponendo un feedback loop entro cui operano tutti i team applicativi e altre figure coinvolte: esperti di sicurezza (che ricevono notifiche sull'esecuzione continua di test di sicurezza da parte del build-automatino server), responsabili di progetto che ricevono notifiche circa l'esito di unit e system test, di eventuali rilasci in produzione e di eventuali rilasci bloccati a causa, del mancato superamento di quality gate



- Ciascun processo è descritto da file, scritti in linguaggio Groovy, che descrivono i vari step, controllano esiti e quality gate, e generano report
- Ciascun processo in esecuzione nel build-automation server è detto **pipeline**
- Ciascuna pipeline è disponibile in una **shared library di Jenkins**, in modo da assicurare che tutti i team usino le stesse pipeline (i team non si sviluppano le proprie pipeline)





# CI/CD

- Nell'esempio sottostante la pipeline di build, dopo aver rilasciato nell'ambiente di quality, invoca una seconda pipeline per l'esecuzione dei system test funzionali, con framework Karate
- I test falliscono, la seconda pipeline restituisce il risultato alla prima, che provvede a fare un rollback dell'ambiente di quality, ripristinando l'ultima versione funzionante; poi invia una mail di notifica ai responsabili di progetto



Build, unit test, rilascio in ambiente quality, invocazione pipeline di system test



Esecuzione dei system test



- 

[←](#) [→](#) [↻](#) [🔍](#) [👤](#) [⚙️](#) [https://autostrade.it/blue/organizations/jenkins/Free2X%2FHHR%2Fuser-service/detail/release/71/pipeline](#) [★](#) [📧](#) [📄](#) [🔗](#) [🔖](#) [🔧](#) [🔍](#) [☰](#)

[📖 Importa segnalibri...](#) [📌 Come iniziare](#) [🔗 example-pipeline-usin...](#) [📖 Altri segnalibri](#)

[✕](#) [📁 / HHR / HHR / Heroku user-service < 71 >](#) [Pipeline](#) [Changes](#) [Tests](#) [Artifacts](#) [🔄](#) [✎](#) [⚙️](#) [📦](#) [Logout](#) [✕](#)

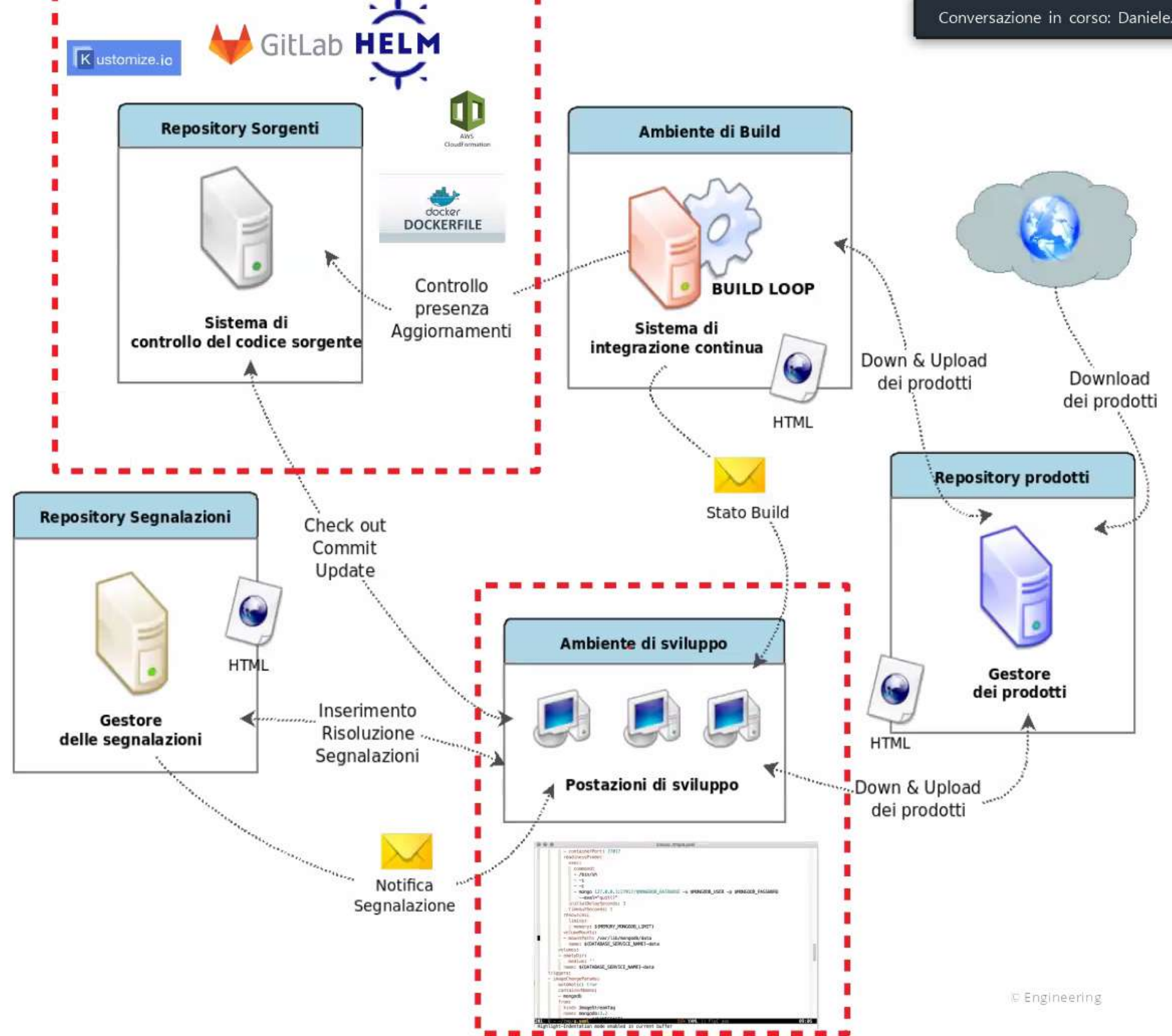
[Branch: release/71](#) [👤 Jim 4/66](#) [Changes by Lorenzo Rita](#)

D-DMZ/classes



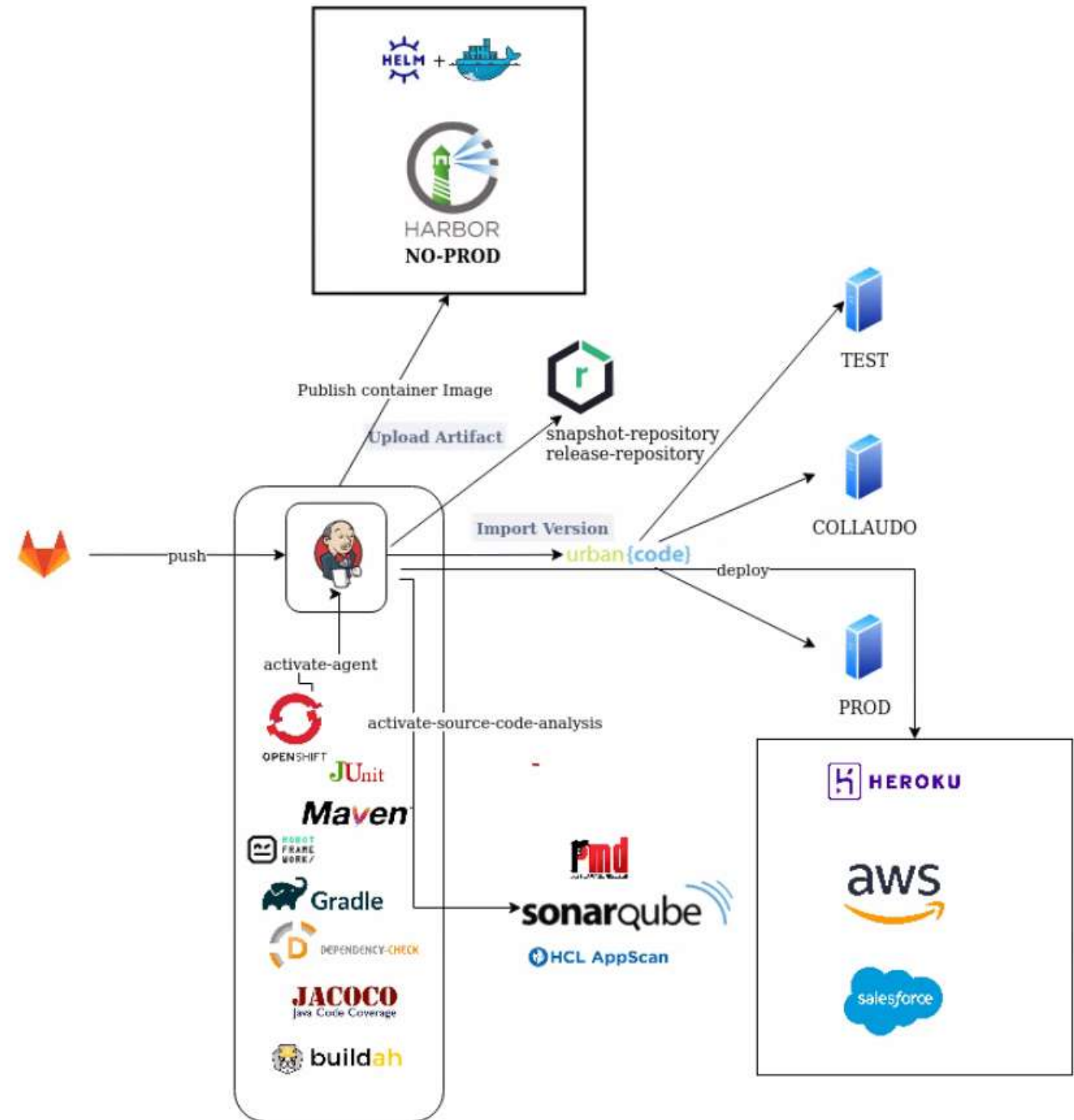
## CM

- Non solo i sorgenti, ma anche le configurazioni applicative
- Dockerfile che descrivono il pacchetto
- File YAML che definiscono la configurazione per il rilascio min un certo ambiente (ogni ambiente - sviluppo, test, collaudo, ecc ha una sua configurazione di risorse: storage, database, ma risorse computazionali dedicate come memoria, ecc)
- Helm e Kustomize per la gestione della configurazione applicativa
- Per ambienti AWS si hanno template Cloudformation
- Tecnicamente sono tutti file di testo con formato e contenuto specifici



# Riepilogo

- Dal push al rilascio
- Jenkins istanzia l'agent necessario per la specifica filiera tecnologica
- Sono eseguiti i passi previsti dalla pipeline e applicati i quality gate
- Se tutto va a buon fine, avviene il rilascio sull'ambiente target
- Jenkins scegli l'ambiente target in base al branch in cui è avvenuto il push:
  - master/main → produzione
  - Develop → test
  - Quality → collaudo
  - ...
- In caso di merge request, step di verifica e validazione per fornire criteri di approvazione o rifiuto della merge (push) request





# Eat your own food!

- Come detto, le pipeline sono codice scritto in linguaggio Groovy
- Pertanto anche per il processo di sviluppo delle pipeline si usano:
  - ITS: per tracciare modifiche a pipeline esistenti, bug o sviluppo di nuove pipeline
  - VCS: per tenere sotto controllo tutte le modifiche e gestire il processo di sviluppo tramite il workflow pattern Gitflow. Nuove versioni di pipeline sono sviluppate in branch dedicati, e vengono riportate nel branch master/main solo al superamento dei system test
  - TEST: vengono scritti ed eseguiti unit e integration test, facendo uso di mock, per simulare l'esecuzione della pipeline all'interno di Jenkins. Vengono poi eseguiti system test con progetti software di prova
  - CM: Jenkins fa uso di agent istanziati al volo per le diverse filiere. Ad esempio per un progetto Java Maven Jenkins istanzia al volo un agent specializzato in build Maven. Questi agent sono container Docker, e il loro Dockerfile è conservato in un progetto in Gitlab
  - BUILD AUTOMATION: quando viene modificata la definizione di un agent all'interno del Dockerfile, al push della modifica, parte una pipeline che effettua la build del container secondo la nuova specifica
  - ARTIFACT REPOSITORY: terminata la build del container con la nuova versione dell'agent, la nuova immagine viene salvata nel registry

