# Pipeline nell'era post-AWS CodeCommit

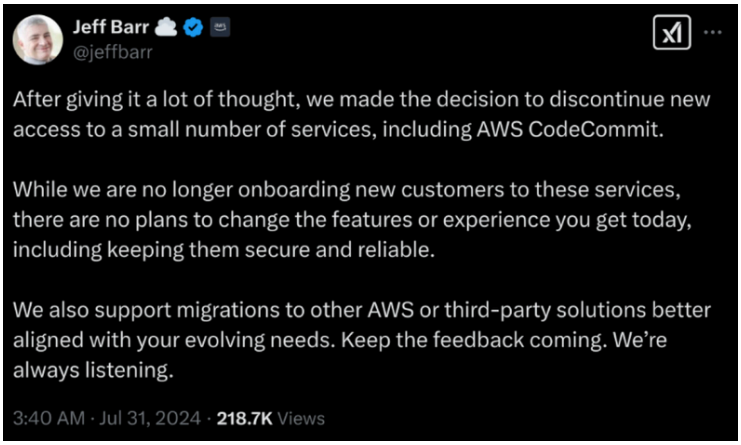
*26 Febbraio 2025 - 8 min. read*

AWS CodeCommit | CI/CD | GitHub | GitLab | Landing Zone

La filosofia DevOps e l'evoluzione dei processi di sviluppo del software hanno portato ad automatizzare le operazioni di build e deploy delle applicazioni. E così, grazie alle pipeline applicative ed infrastrutturali, finalmente non ci sono più sviluppatori che portano chiavette USB nell'ufficio dei sistemisti per fare il deploy della nuova release del software!

La prima fase per la creazione di una applicazione Cloud-native su AWS comprende sempre la creazione di un repository Git e, il passaggio più naturale a partire dal 9 Luglio 2015, è sempre stato fare click nella sezione CodeCommit della console AWS e cliccare "Create Repository".

A poco più di 9 anni dopo (il 31 Luglio 2024) siamo venuti a conoscenza che questa abitudine dovrà cambiare a seguito dell'annuncio di Jeff Barr su X.



La notizia ha colto tutti di sorpresa: in sostanza, tutti gli account AWS aperti dopo il 31 Luglio 2024 non potranno più accedere al servizio, mentre gli account parte di una Organization in cui erano già presenti repository potranno continuare as accedervi normalmente.

Il servizio non è però in fase di dismissione, come era stato ipotizzato nei giorni successivi all'annuncio; sono garantiti gli aggiornamenti di sicurezza, la risoluzione dei bug e la manutenzione, ma non verranno sviluppate nuove feature.

Nonostante fosse un servizio che non offriva opzioni avanzate per la collaborazione collettiva, la sua forza stava nell'integrazione con i vari strumenti di automazione e rilascio offerti da AWS (come ad esempio CodePipeline, CodeDeploy ed Amplify).

Sulle motivazioni che hanno portato AWS a prendere questa decisione sono già state fatte molte discussioni. Il nostro obiettivo è invece quello di vagliare le differenti alternative disponibili, mettendo in luce pro e contro in base anche ai differenti scenari.

Per poter decidere quale soluzione possa sostituirlo al meglio, capiamo prima quali erano i punti di forza e le mancanze di AWS CodeCommit.

# Perchè AWS CodeCommit era comodo

AWS CodeCommit era sicuramente uno dei servizi più facili e intuitivi tra quelli offerti da AWS e la sua integrazione nativa con AWS CodePipeline rendeva lo sviluppo di tutto il comparto CI/CD molto più immediato e leggero.

Un setup intuitivo e veloce, permetteva agli sviluppatori di concentrarsi principalmente sul codice applicativo, ma senza rinunciare alla funzionalità di rilascio continuo e testing. Basta pensare che tutti i linguaggi di IaC supportano direttamente la configurazione di pipeline con uno step di source direttamente integrato con il repository.

Un altro aspetto che semplificava molto l'operatività era l'integrazione con IAM, che forniva la possibilità di gestire i permessi Git utilizzando gli utenti già censiti: era possibile definire le git Operations (Pull, push, merge...) consentite agli utenti direttamente all'interno delle policy IAM, risultando veloce e di facile gestione.

Era anche possibile realizzare flussi di notifica e approvazione direttamente integrati con il Cloud AWS (ad esempio Amazon SNS, Amazon EventBridge), mentre cambiando Git provider, potrebbe rendersi necessario realizzare integrazioni non più direttamente configurabili su AWS.

Inoltre, il costo del servizio era davvero irrisorio e permetteva a chiunque di approcciarsi senza doversi districare fra le feature offerte da una selva di piani di utilizzo differenti.

Un ultimo aspetto, ma molto importante, era la sua integrazione nella Landing Zone AWS, che permette di avere una governance unica di tutti i servizi.

In questo modo, quindi, anche l'accesso ai repository ed i permessi potevano essere definiti utilizzando IAM, facilitando così i controlli e le review ed accentrando di fatto il controllo in un unico punto centralizzato che può essere federato con l'IdP aziendale.

Tutte queste caratteristiche hanno aiutato non solo le realtà più strutturate, ma anche le startup e imprese più piccole. Il non dover dipendere da altri team di gestione per strumenti "corporate", l'integrazione e il basso costo hanno infatti permesso la nascita e il test di molti progetti, facilitando la sperimentazione e sposando perfettamente il principio del "fail-fast" tipico del Cloud, entrambi punti fondamentali della filosofia di AWS che mira ad abilitare sempre più "builder" a mettere a terra soluzioni innovative velocemente.

## Debolezze

Nonostante tutti gli aspetti positivi descritti, era innegabile che AWS CodeCommit avesse alcun limiti abbastanza evidenti.

Il primo tra tutti era il ridotto numero di integrazioni con elementi esterni ad AWS. Al giorno d'oggi è essenziale per uno strumento di code repository fornire svariate integrazioni oltre a gestire branch e risolvere conflitti. Ormai praticamente tutti i vendor più famosi offrono soluzioni direttamente integrate con prodotti di terze parti per poter offrire all'utente un esperienza a 360° sulle loro piattaforme.

Il secondo aspetto critico era la scarsa implementazione di feature per lo sviluppo e le review collaborative. Anche se la possibilità di effettuare "pull & merge request" era presente, la loro implementazione era molto basilare. Per non parlare della user experience; ad oggi, gli strumenti di sviluppo collaborativo offrono interfacce utente

molto più pulite, pur offrendo numerosissime feature anche molto complesse e avanzate per review e comparazione di codice.

Riflettendo sull'annuncio si può pensare quindi che la strategia di AWS non fosse quella di competere con i big player offrendo un servizio sostitutivo e completo, ma di offrire un'opzione integrata in un'ottica di governance centralizzata degli account e delle risorse.

# Codebases alternative: GitHub, Gitlab, Bitbucket

Come abbiamo già detto, AWS CodeCommit non è più disponibile per i nuovi arrivati su AWS, ma il servizio non è in dismissione: non è necessario pensare a strategie di emergenza, ma bisogna comunque iniziare a valutare tra le diverse opzioni disponibili.

Prima di tutto dobbiamo capire quale, tra le tante scelte, è quella che più si adatta alle nostre esigenze e risorse.

AWS suggerisce direttamente alcuni vendor, tra cui troviamo i più famosi **GitHub**, **GitLab** e **BitBucket**. Il driver principale dietro a questo suggerimento è proprio - di nuovo - l'**integrazione tra queste piattaforme e i servizi AWS.**

L'ago della bilancia può essere spostato da una semplice domanda:

"qual'è l'utilizzo effettivo del mio repository?"

Cercheremo di abbinare le risposte più comuni a questa domanda con i vari prodotti disponibili sul mercato.

Tra i nomi citati prima, è naturale che l'occhio ricada sui due colossi presenti nella lista: GitHub e GitLab. Dato il loro elevato utilizzo, sono veramente pochi i servizi (AWS e non) con i quali queste piattaforme non offrono un'integrazione. Tutte e tre le soluzioni possono essere configurate direttamente come step di source per AWS CodePipeline attraverso la creazione di una CodeStar connection. In questo caso, sia per GitHub che per GitLab, possiamo utilizzare la versione SaaS o quella self-hosted.

Se intendiamo approcciare una di queste soluzioni, viene anche naturale porsi la domanda "ho veramente bisogno di AWS CodePipeline?".

Se prima utilizzavamo AWS CodeCommit perché comodo e presente nella stessa conosole di AWS CodePipeline, ora che dovremo utilizzare un altro servizio esterno potrebbe non valere più la pena di usare le pipeline di AWS.

Sia GitHub che Gitlab, ma anche BitBucket, offrono il loro servizio di pipeline integrate. Rispetto ad AWS questi servizi sono costruiti per sfruttare le tecnologie più disparate e i prodotti di terze parti più comuni. Adottare questo tipo di Pipeline potrebbe agevolare tutti quegli step intermedi che spesso dovevano essere orchestrati con complessi automatismi di AWS CodeBuild.

Al contempo, l'effort di sviluppo degli strumenti dedicati alla developer experience su AWS si è spostato ed offre integrazioni migliori, come nel caso di GitLab: è possibile ora configurare i runner utilizzando AWS CodeBuild come piattaforma di compute, permettendo di impostare un  controllo dei permessi più granulare rispetto al passato per interagire con le risorse cloud. A chi non è mai capitato di vedere runner su EC2 che avevano effettivamente i permessi di AdministratorAccess?

Anche le GitHub actions ora sono integrate, ed è appena stata annunciata l'integrazione con BuildKite.

### Il "fatto in casa"

Esiste anche poi l'alternativa "homemade": anche un mero repository Git accessibile via ssh ha la possibiltà di invocare hook e compiere azioni, ma gestire in autonomia il proprio repository, spendendo tempo per garantire l'alta affidabilità, la sicurezza ed effettuare le manutenzioni non porta benefici rispetto all'utilizzo di servizi gestiti che garantiscono già questi aspetti e che permettono invece di concentrarsi sulle  vere esigenze di business. Questa ipotesi non è comunque da escludere per requisiti e casi molto particolari.

## Conclusione

Dopo l'annuncio di Jeff Bar di alcuni mesi fa, in molti avranno pensato di dover correre ai ripari scegliendo in fretta e furia una nuova codebase.

In realtà non si tratta di una dismissione del servizio, ma è comunque giusto iniziare a valutare le diverse opzioni che il mercato ha da offrire come alternativa a un servizio che - almeno per chi ha un ecosistema su AWS - era diventato quasi uno standard.

Con i suoi PRO e CONTRO, AWS CodeCommit ci lascia un'importante eredità di insegnamenti riguardo alla gestione delle pipeline, utilissima durante il processo decisionale per la scelta di una valida alternativa:

Avere una console centralizzata per la gestione di accessi e configurazioni, ad esempio, è fondamentale per mantenere la governance.

L'integrazione con gli IDP di terze parti è un altro aspetto importante; molti Git-Repository si integrano ormai nativamente e permettono di configurare un meccanismo di SSO per fornire agli utenti un'esperienza trasparente e agevolata nell'utilizzo di un nuovo strumento.

Un altro punto importante riguarda la UX: quando si effettua un cambio di questo genere, il primo impatto sull'usabilità di un prodotto ne decreta spesso un feedback positivo o negativo da parte degli utenti che dovranno approcciarsi a questo cambio.

Una volta scelta una nuova piattaforma, anche se tutto funziona sulla carta, è consigliabile iniziare lo sviluppo dei nuovi progetti per prendere confidenza con gli strumenti offerti e pianificare quindi una migrazione graduale dei repository già esistenti in ottica di unificazione della gestione, eventualmente affiancati da un partner.

Avete già riflettuto su quale strumento accoglierà i vostri repository in futuro? Fatecelo sapere nei commenti!

---

## About Proud2beCloud

Proud2beCloud è il blog di beSharp, APN Premier Consulting Partner italiano esperto nella progettazione, implementazione e gestione di infrastrutture Cloud complesse e servizi AWS avanzati. Prima di essere scrittori, siamo Solutions Architect che, dal 2007, lavorano quotidianamente con i servizi AWS. Siamo innovatori alla costante ricerca della soluzione più all'avanguardia per noi e per i nostri clienti. Su Proud2beCloud condividiamo regolarmente i nostri migliori spunti con chi come noi, per lavoro o per passione, lavora con il Cloud di AWS. Partecipa alla discussione!

---

**Riccardo Fragnelli**

DevOps @beSharp.Ho un passato on-prem prima di redimermi con il Cloud. Molto pignolo e abbastanza pigro. Mi piace passare il tempo fra videogiochi e GDR. Con AWS ho scoperto una branca dell'informatica tutta nuova che mi affascina sempre di più.

---