Procedure più veloci grazie alla distribuzione continua

Mark Mansour

Procedure più veloci grazie alla distribuzione continua

Miglioramenti continui e automazioni software

Oltre 10 anni fa, iniziammo un progetto presso Amazon per comprendere la velocità con cui i nostri team riuscivano a trasformare le idee in sistemi di produzioni di alta qualità. Misurammo il throughput del software con l'obiettivo di migliorare la qualità dell'esecuzione. Scoprimmo che erano necessari, in media, 16 giorni dall'estrazione del codice alla produzione. In Amazon, i team partivano da un'idea e solitamente serviva un giorno e mezzo per scriverne il codice relativo e realizzarla. Era necessario meno di un'ora per creare e distribuire il nuovo codice. Il resto del tempo, quasi 14 giorni, era trascorso in attesa: dovevamo aspettare che i membri del team iniziassero la compilazione, per poi eseguire le distribuzioni ed eseguire test. Alla fine del progetto, suggerimmo l'automazione dei processi post-estrazione per aumentare la velocità di esecuzione. L'obiettivo era quello di eliminare i ritardi pur mantenendo o addirittura migliorando la qualità.

Il nucleo di questo suggerimento era la creazione di un programma di continuo miglioramento finalizzato ad aumentare la velocità dell'esecuzione. La decisione di migliorare la velocità di esecuzione fu basata sul principio di leadership di insistere sugli standard più alti. Questo principio consiste nel perseguire standard estremamente alti, aumentando continuamente il livello delle prestazioni, ottenendo così prodotti, servizi e processi di alta qualità. Il nostro Principio di leadership riflette in tutto e per tutto le modalità di lavoro di Amazon, quelle di dirigenza dei leader, e descrive l'importanza del ruolo del cliente, tenuto al centro nel momento delle decisioni.

Amazon aveva già creato strumenti di sviluppo software per aumentare la produttività degli ingegneri informatici. Creammo un sistema di compilazione ospitato e centralizzato, tutto nostro, Brazil, che eseguiva una serie di comandi su un server; l'obiettivo era quello di generare un artefatto che potesse essere distribuito. Tuttavia, Brazil non ascoltava i cambiamenti del codice sorgente. Era necessario che una persona iniziasse la compilazione. Avevamo inoltre il nostro sistema di distribuzione, Apollo, che richiedeva il caricamento di un artefatto di compilazione per poter iniziare la distribuzione. Motivati dall'interesse per la distribuzione continua, che continuava a crescere nel nostro settore, creammo un sistema nostro, Pipelines, per automatizzare il processo di consegna del software tra Brazil e Apollo.

Pipelines: il nostro strumento per la distribuzione continua

Iniziammo un programma pilota per automatizzare il processo di consegna del software per alcuni team. Una volta concluso il progetto, il team principale del progetto pilota avevo ottenuto una riduzione del 90% sulle tempistiche totali necessarie dall'estrazione alla produzione.

Il progetto convalidò l'idea secondo cui tramite una pipeline era possibile definire tutti i passaggi necessari per rilasciare software ai clienti. Il primo passaggio di una pipeline è la compilazione di un artefatto. In seguito, la pipeline esegue l'artefatto della compilazione attraverso una serie di passaggi finché l'artefatto non viene rilasciato a tutti i clienti. Utilizziamo le pipeline per ridurre il rischio che una nuova modifica di un codice possa avere un impatto negativo sui nostri clienti. Ogni passaggio della pipeline dovrebbe confermare la certezza che l'artefatto della compilazione non presenti alcun difetto. Se la produzione presenta un difetto, la nostra priorità è ripristinarne lo stato d'integrità quanto più velocemente possibile.

Quando lanciammo Pipelines, poteva solo modellare un singolo processo di rilascio per applicazione. Questa limitazione portò alla coerenza, alla standardizzazione e alla semplificazione dei processi di rilascio di un team. Il risultato fu una riduzione del numero dei difetti. Prima di iniziare a utilizzare le pipeline, era frequente che i team seguissero processi di rilascio diversi per le correzioni di bug e per i rilasci di funzionalità principali. Quando gli altri team si resero conto del successo dei team che utilizzavano la consegna automatica, iniziarono a migrare i loro processi di rilascio manuali in pipeline per migliorarne la coerenza. I team che prima seguivano diversi processi di rilascio adesso ne avevano uno standardizzato e utilizzato da tutti. Inoltre, ogni volta che i processi di rilascio venivano trasferiti in uno strumento, i membri del team avevano la possibilità di rivisitare il proprio approccio e spesso scoprivano modi per semplificare il processo.

Il team Pipelines aveva obiettivi annuali per aumentare l'utilizzo tramite la tecnica dell'"adozione tramite seduzione". In altre parole, dovevano rendere la qualità del prodotto talmente alta, che tutti avrebbe voluto utilizzarlo. Contammo il numero dei team che utilizzavano una pipeline per distribuire il software alla produzione e classificammo le pipeline per livello di automazione. Alcuni team avevano l'obiettivo di utilizzare una pipeline per il rilascio software ed erano intenzionati a spostarsi completamente verso i rilasci automatizzati. Tuttavia, notammo che in alcune organizzazioni, il modo in cui misuravamo la qualità poteva portare i team ad automatizzare il processo di rilascio senza eseguire alcun test.

La risposta alla domanda "quanti test sono sufficienti?" è relativa e non assoluta. È necessario che un team comprenda il contesto in cui sta operando. In questa situazione utilizzammo un altro principio di leadership, la Responsabilità. Questo principio consiste nel pensare a lungo termine, non sacrificando il valore a lungo termine per ottenere risultati a breve termine. I team Software di Amazon hanno alti livelli di testing in cui investono grande sforzi, perché assumersi la responsabilità di un prodotto significa anche assumersi la responsabilità delle conseguenze dovute a possibili difetti del prodotto. Nel caso in cui sorga un problema che ha un impatto sui clienti, sono i membri del team software monotematico a gestire il problema e a risolverlo in tempo reale. La tensione che si crea tra l'aumento della velocità di esecuzione e la capacità di rispondere ai problemi di produzioni fa sì che i team siano motivati ad eseguire test in modo adeguato. Tuttavia, se indaghiamo troppo nell'ambito del testing, potremmo non aver successo, dato che altri si sono mossi più velocemente in questa direzione. Cerchiamo sempre di migliorare i nostri processi di rilascio di software senza diventare un elemento di blocco per il settore.

Un altro problema da affrontare fu il fatto che i team non stavano trasmettendo l'uno all'altro le procedure consigliate per il rilascio di software I team monotematici vengono incoraggiati a lavorare in modo autonomo e dunque gli ingegneri risolvevano i loro problemi di distribuzione in modo indipendente. Una volta trovata la soluzione che rispondesse alle loro esigenze di rilascio software, promuovevano la tecnica scoperta con altri ingegneri per email, durante riunioni o tramite altri canali di comunicazioni. Due sono i problemi correlati a questo stile di comunicazione. Primo, questi canali di comunicazione non sono affidabili, infatti non tutti ricevevano le informazioni sull'esistenza di nuove tecniche. Secondo, i leader che incoraggiano i propri team ad adottare le nuove procedure consigliate non hanno modo di verificare se i loro team hanno effettivamente eseguito il lavoro necessario per poter adottare tali procedure.

Ci rendemmo conto che dovevamo aiutare gli ingegneri ad accedere alle procedure consigliate scoperte, e fornire ai leader la possibilità di identificare le pipeline da rivedere.

La soluzione fu quella di meccanizzare il processo aggiungendo un sistema di verifica sulle procedure consigliate all'interno degli strumenti utilizzati per compilare e rilasciare il software. Comprendiamo che una procedura ottimale per un'organizzazione potrebbe non esserlo per un'altra azienda, quindi abbiamo fatto in modo che questo sistema di verifica sia configurato organizzazione per organizzazione. Il sistema di verifica sulle procedure consigliate a livello di organizzazione ha fornito ai leader la capacità di personalizzare i propri processi di rilascio per soddisfare le esigenze del proprio settore. I leader che volevano far applicare o incoraggiare l'utilizzo di una nuova procedura consigliata furono in grado di fornire un'avvertenza tramite gli strumenti utilizzati quotidianamente dagli ingegneri. Inserendo messaggi all'interno degli strumenti, veniva garantito quasi completamente che i membri del team venissero a conoscenza delle procedure consigliate e che le utilizzassero. Scoprimmo che concedendo ai team il tempo di imparare e dibattere sulle procedure consigliate, l'organizzazione aveva la possibilità di reiterare e migliorare il sistema di verifica delle procedure consigliate. Ciò portò infine un miglioramento della qualità delle procedure consigliate e maggiore supporto e predisposizione de parte degli ingegneri.

Identificavamo in modo sistematico le procedure consigliate da applicare. Un gruppo di ingegneri senior fece una liste delle ragioni più comuni correlate al non funzionamento di un rilascio. Identificarono i passaggi che avrebbero fatto funzionare il rilascio. Poi utilizzammo quella lista per creare un set di verifiche per le procedure consigliate. Attraverso questo processo ci rendemmo conto che se da un lato volevamo che i nuovi aggiornamenti software venissero trasmessi ai nostri clienti in modo istantaneo, senza sforzo e senza comprometterne la disponibilità, la nostra priorità era innanzitutto la disponibilità, poi la velocità e poi la semplificazione per i nostri ingegneri.

Riducendo il rischio della presenza di difetti nel prodotto finale

È normale che ogni ingegnere introduca prima o poi un difetto in uno dei nostri sistemi. I nostri sistemi di rilascio, incluso le pipeline e i sistemi di distribuzione, devono essere progettati per identificare tali difetti più rapidamente possibile ed evitare che abbiano un impatto sui nostri clienti. Dobbiamo assicurarci che i nostri processi di rilascio siano configurati correttamente e che il nostro artefatto di compilazione funzioni come dovuto.

Protezione della distribuzione: la forma più basica del testing di distribuzione assicura che l'artefatto appena distribuito possa essere acceso e che sia in grado di rispondere. Parte del flusso di lavoro post-distribuzione consiste nell'esecuzione di rapide verifiche per assicurare che l'artefatto appena distribuito sia acceso e stia gestendo il traffico. Per esempio, utilizziamo hook del ciclo di vita eventi nel file AWS CodeDeploy AppSpec per azionare gli script di esempio all'arresto, all'accensione o alla convalida della distribuzione. Controlliamo inoltre di avere una capacità sufficiente per poter gestire il traffico del cliente. Abbiamo creato tecniche come host d'integrità minimi in CodeDeploy per verificare di avere sempre una capacità sufficiente per fornire il servizio ai nostri clienti. Infine, se il motore di distribuzione riesce a rilevare un errore,

dovrebbe eseguire il rollback di tale modifica per minimizzare il tempo durante il quale i clienti vedono il difetto.

Testing precedenti alla produzione: una delle procedure consigliate di Amazon è quella di automatizzare il testing di unità, di integrazione e quello precedente alla produzione, e aggiungere tali test nella pipeline. Insistiamo sull'esecuzione di testing di carico e di sicurezza e siamo in favore dell'aggiunta di questi test nelle pipeline. Quando parliamo di test di unità, intendiamo tutti i test che si vogliono eseguire su una macchina di compilazione, incluso controlli di stile, copertura dei codici, complessità dei codici e molto altro. I test d'integrazione includono tutti i testing off-box come il fault injection, il testing del browser automatizzato e simili. Sono molti gli articoli che parlano di test di unità e di integrazione in modo eccellente, quindi non entrerò nei dettagli.

L'obiettivo del testing di unità e di integrazione è quello di verificare che il comportamento dell'artefatto di compilazione sia funzionale e corretto. Più verifiche eseguiamo, meno sono i rischi che un difetto venga esposto ai nostri clienti. Per ridurre le tempistiche necessarie per far sì che un prodotto giunga nelle mani dei nostri clienti, cerchiamo di rilevare un difetto il prima possibile nel processo di rilascio. In generale, ciò significa che se i tuoi test sono più brevi e più rapidi, riceverai un feedback più rapidamente su qualsiasi problema correlato alle tue modifiche.

Presso Amazon, utilizziamo inoltre una tecnica chiamata "*testing precedente alla produzione*". L'ambiente di pre-produzione è l'ultimo luogo in cui il testing avviene prima che le nostre modifiche vengano distribuite in produzione. Un test dell'ambiente di pre-produzione utilizza la configurazione di produzione del sistema quindi agisce esattamente come un sistema di produzione. Questo approccio ha due vantaggi. Primo, gli ambienti di pre-produzione testano la configurazione della produzione per assicurare che il servizio si possa connettere correttamente a tutte le risorse di produzione, inclusi gli archivi dati della produzione. Secondo, assicura che il sistema interagisca correttamente con le API dei servizi di produzione da cui dipende. Gli ambienti di pre-produzione sono esclusivamente utilizzati dal team che è responsabile di tale servizio, e non ricevono mai traffico dai clienti. L'esecuzione di testi di pre-produzione aumenta la sicurezza che lo stesso codice e la stessa configurazione funzionino in produzione.

Convalida a livello di produzione: quando rilasciamo un codice ai nostri clienti, non lo facciamo in una sola volta. Il rilascio di un difetto a tutti i clienti in una sola volta, e l'impatto consequente, sarebbe di portata troppo grande. Preferiamo distribuire a celle, un'istanza di servizio completamente indipendente. Quando distribuiamo una modifica al nostro primo gruppo di clienti nella prima cella, operiamo con estrema cautela. Facciamo in modo che solo un piccolo gruppo di clienti vedano la nuova modifica e raccogliamo feedback sul funzionamento del nuovo codice. Monitoriamo la quantità di errori che i nostri servizi emettono dopo una distribuzione canary. Se il livello di errore aumenta, eseguiamo automaticamente il rollback della modifica. Per esempio, potremmo aspettare di ricevere 3000 punti dati positivi, con nessun punto dato negativo, prima di continuare una distribuzione.

Può sorgere una complicazione se i test automatizzati si perdono un caso d'uso. Facciamo il massimo per individuare tutti gli errori con i nostri test strutturati e ripetibili, sia che siano automatizzati o manuali. Tuttavia, anche quando facciamo il nostro meglio, è possibile che un difetto ci sfugga. Per testare i nostri test, lasciamo la nuova modifica in produzione per un

periodo di tempo stabilito per vedere se un membro di un altro team trova errori. Abbiamo trascorso molto tempo a discutere su è bene che le modifiche rimangano in produzione per un periodo oppure quanto tempo è bene attendere dopo la distribuzione canary prima di eseguire la distribuzione al resto del gruppo di distribuzione. Molti dei nostri team hanno deciso di attende un periodo di tempo stabilito oltre alla raccolta di punti dati positivi prima di andare avanti con la routine di distribuzione. Il tempo di attesa di una pipeline è strettamente dipendente dal team. Alcuni team attendono ore, altri attendono minuti. Più è alto l'impatto e più è lungo il periodo di tempo necessario per correggere un problema, più è lento il processo di rilascio.

Una volta acquisita fiducia nella prima cella, esponiamo la modifica del nuovo codice a sempre più clienti fino al completo rilascio. Proprio come per il canale di distribuzione, anche in questo caso attendiamo per acquisire fiducia nella distribuzione della prima nuova cella prima di andare avanti con la prossima cella. Una volta acquisita più fiducia nell'artefatto di compilazione, riduciamo il tempo di verifica della modifica del codice. Il risultato è un modello in cui l'obiettivo è quello di giungere dall'estrazione al primo cliente di produzione quanto più rapidamente possibile. Tuttavia, una volta raggiunta la produzione, rilasciamo lentamente il nuovo codice ai clienti, e facciamo di tutto per acquisire maggiore fiducia, mentre incrementiamo la velocità del resto delle nostre distribuzioni.

Per assicurarci che i nostri sistemi di produzione continuano a soddisfare le richieste dei clienti, generiamo traffico sintetico sui nostri sistemi. Se il nostro servizio non funziona correttamente, vogliamo ricevere un feedback tempestivo, perciò eseguiamo test sintetici almeno ogni minuto. Progettiamo test sintetici per assicurarci che tutti i processi in esecuzione siano integri e che tutte le dipendenze siano sottoposte a test, che spesso includono anche controlli a tutte le API pubbliche.

Avere il controllo nel momento del rilascio del software: per avere il controllo sul livello di sicurezza dei rilasci, abbiamo progettato meccanismi che consentono di avere il controllo sulla velocità con la quale le modifiche attraversano la pipeline. Utilizziamo parametri, finestre di tempo e controlli di sicurezza per avere il controllo al momento del rilascio del software.

Quando viene inviato un avviso in seguito a una modifica dei parametri, le pipeline possono essere configurate per evitare la distribuzione. Utilizziamo le metriche in modo accurato e disponiamo di avvisi sull’integrità dei sistemi, sull’integrità delle celle, su regioni e zone di disponibilità e molto altro ancora. Configuriamo le pipeline in modo che si blocchi la distribuzione del codice nel momento in cui viene attivato un avviso da un parametro importante. Tuttavia, capita che un team abbia bisogno di distribuire una correzione indirizzata all’avviso di sistema. In questo caso, i team eseguono l’override degli avvisi evitando che le modifiche si spostino su una pipeline.

Le pipeline possono specificare un intervallo di tempo in cui è consentito che le modifiche progrediscano nella pipeline. I team possono individuare un intervallo di tempo per limitare il periodo in cui le modifiche raggiungono i clienti. I team di AWS preferiscono rilasciare il software quando sono presenti molte persone, affinché possano rispondere rapidamente e mitigare un problema provocato da una distribuzione. Per rendere tutto ciò effettivo, i team generalmente impostano i propri intervalli di tempo in modo da poter effettuare la

distribuzione solamente durante le ore lavorative. Gli altri team Amazon preferiscono rilasciare il software quando il traffico di clienti è basso. Questi intervalli di tempo possono essere superati se necessario.

Esiste anche la possibilità di bloccare una pipeline sulla base dei contenuti dell'artefatto della compilazione. Ad esempio, possiamo bloccare un artefatto della compilazione che contiene un pacchetto non idoneo conosciuto o un riferimento GIT specifico. Abbiamo utilizzato questa caratteristica quando abbiamo scoperto che una modifica a un pacchetto conteneva una regressione delle prestazioni. Se rimuovessimo solo il pacchetto dal repository di pacchetti, le pipeline che contengono il pacchetto difettoso, continuerebbero a distribuire la modifica non idonea ai clienti.

L'approccio usato per la velocità dell'esecuzione

Abbiamo constatato che i team sono molto favorevoli all'automazione. Siamo tutti molto motivati a realizzare e rilasciare funzionalità per i clienti che migliorano la qualità della vita e la consegna continua lo rende fattibile. Abbiamo notato che l'automazione restituisce tempo agli ingegneri, eliminando lavoro manuale laborioso, frustrante e tendente all'errore. Abbiamo dimostrato che la distribuzione continua ha un impatto positivo sulla qualità. Abbiamo visto che l'automazione consente ai team di effettuare rilasci di frequente, una modifica alla volta, semplificando l'identificazione delle regressioni.

Quando i sistemi sono nuovi, la superficie da testare è generalmente comprensibile dalla maggior parte dei membri del team, il che rende gestibile un testing manuale. Tuttavia, il valore dell'automazione aumenta con la maggiore complessità dei sistemi e con il cambiamento dei membri dei team. Ci piace automatizzare i sistemi in modo da poterci concentrare sull'aggiunta di valore per il cliente piuttosto che nella gestione del processo di rilascio ai clienti.

Per molti anni Amazon ha eseguito programmi di miglioramento continuo focalizzati sulla velocità con cui vengono rilasciati software ai clienti e sulla sicurezza di tali rilasci. Non abbiamo iniziato con tutti i controlli di rischio e i test descritti in questo articolo. Negli anni abbiamo inventato modi per identificare e mitigare il rischio.

I programmi di miglioramento continuo sono eseguiti da leader aziendali a diversi livelli dell'organizzazione. Ciò consente a ogni leader aziendale di regolare il proprio processo di rilascio del software per calibrare i rischi e l'impatto sul proprio business. Alcuni dei programmi di miglioramento continuo sono eseguiti su ampie sezioni di Amazon e a volte i leader di organizzazioni più piccole eseguono dei programmi propri. Sappiamo che ci sono sempre eccezioni alla regola. I nostri sistemi sono dotati di meccanismi di rifiuto esplicito, in modo da non rallentare i team che hanno bisogno di un'esenzione temporanea o permanente. Fondamentalmente, i nostri team sono responsabilità del comportamento dei propri software e hanno il compito di investire nel processo di rilascio del software in modo appropriato.

Iniziammo misurando i nostri problemi, affrontandoli, e ripetendo. Per rendere questa attività sostenibile, è stato necessario eseguirla in maniera incrementale, celebrandone i successi volta per volta. Quando Amazon iniziò a utilizzare le pipeline, molti team non erano sicuri che la distribuzione continua avrebbe funzionato. Per iniziare, incoraggiammo i team a codificare il

loro processo di distribuzione attuale, inclusi i passaggi manuali, il tutto in una pipeline. Per molti team, la pipeline era vista come un'interfaccia visiva del loro processo di rilascio, e i team non si sentivano incentivati alla realizzazione di artefatti di compilazione tramite un processo di rilascio. Con l'aumento della fiducia, l'automazione venne gradualmente adottata in alcuni passaggi della pipeline finché arrivarono al punto in cui non ebbero più la necessita di azionare manualmente nessun passaggio della pipeline.

Avanzamento rapido fino a oggi. Amazon ha oggi raggiunto il punto in cui i team mirano alla piena automazione nel momento di scrittura del codice. È grazie all'automazione che siamo riusciti a far crescere le nostre attività sempre di più.