Migrazione Ninja dall'on-premise al Cloud con CloudEndure

22 Luglio 2021 - 6 min. read

Application Modernization | Cloud Migration | CloudEndure

"Life is what happens to you while you're busy making other plans"

scrisse John Lennon in "Beautiful Boy".

In questo articolo vedremo che, a volte, anche se sono stati fatti piani accurati e dettagliati gli imprevisti sono sempre dietro l'angolo e spesso ci obbligano a cambiare rapidamente la nostra strategia. Quando si intraprendono percorsi di **migrazione sul Cloud** e **application modernization** uno dei nostri mantra recita che è sempre necessario procedere con attività di refactoring o re-architecting delle applicazioni per poter trarre i massimi benefici dall'utilizzo del cloud. A volte, però, fattori esterni possono rimescolare le carte in tavola e costringerci a cambiare in modo drastico le nostre decisioni, facendoci cambiare del tutto la strada verso il nostro obiettivo.

Alcuni mesi fa un cliente ha chiesto il nostro aiuto per un progetto di migrazione dal provider corrente ad AWS di alcuni siti e-commerce. Come di consueto, abbiamo cominciato stilando un piano seguendo **la regola delle "7R"**.

Dopo aver inventariato tutti i sistemi è emersa la necessità di procedere con azioni sostanziali di **refactoring** e **re-architecting**, mentre nessun sistema doveva essere rinnovato, mantenuto, rilocato o ritirato. Inizialmente l'opzione del rehosting non è stata nemmeno presa in considerazione perché il cliente era desideroso di cambiare il paradigma infrastrutturale attualmente in uso, traendo vantaggio dall'utilizzo del Cloud e dalle sue caratteristiche, specialmente per elasticità, ottimizzazione dei costi ed eccellenza operativa.

La quantità di debito tecnico era considerevole, a causa di sistemi operativi molto vecchi, versioni obsolete di linguaggi di programmazione, librerie e versioni di database end-of-life.

Non ci siamo fatti intimorire: il progetto era molto sfidante ma i prerequisiti erano chiari e l'infrastruttura finale prevedeva applicazioni in **container**, database **RDS** ed alcuni siti statici ospitati sull'accoppiata **Amazon S3 + Amazon CloudFront**.

Lo sforzo maggiore gravava sulle spalle del team di sviluppo del cliente, costantemente supportato dai nostri Cloud Expert: per ogni applicazione andava affrontato un refactoring per renderla **stateless**, utilizzando i **servizi gestiti AWS** (come ad esempio utilizzare S3 per 5 Terabyte di asset statici).

L'inizio del progetto era concordato al termine della stagione natalizia, garantendo agli sviluppatori da un lato la possibilità di rilasciare aggiornamenti e bug fix alla piattaforma, dall'altro concedendo loro tempo a sufficienza per prendere confidenza con termini, servizi e concetti del paradigma.

Tutto sembrava procedere per il meglio, ma la legge di Murphy è sempre in agguato: durante il periodo del Black Friday il vecchio provider ha causato 2 giorni di downtime per problemi gravi, causando al cliente enormi perdite di profitto.

Oltre al danno immediato, l'evento ha fatto da elemento scatenante di numerose altre criticità, cambiando , la priorità del progetto di migrazione da "pianificata" a "appena possibile" con un termine massimo fissato per il mese successivo.

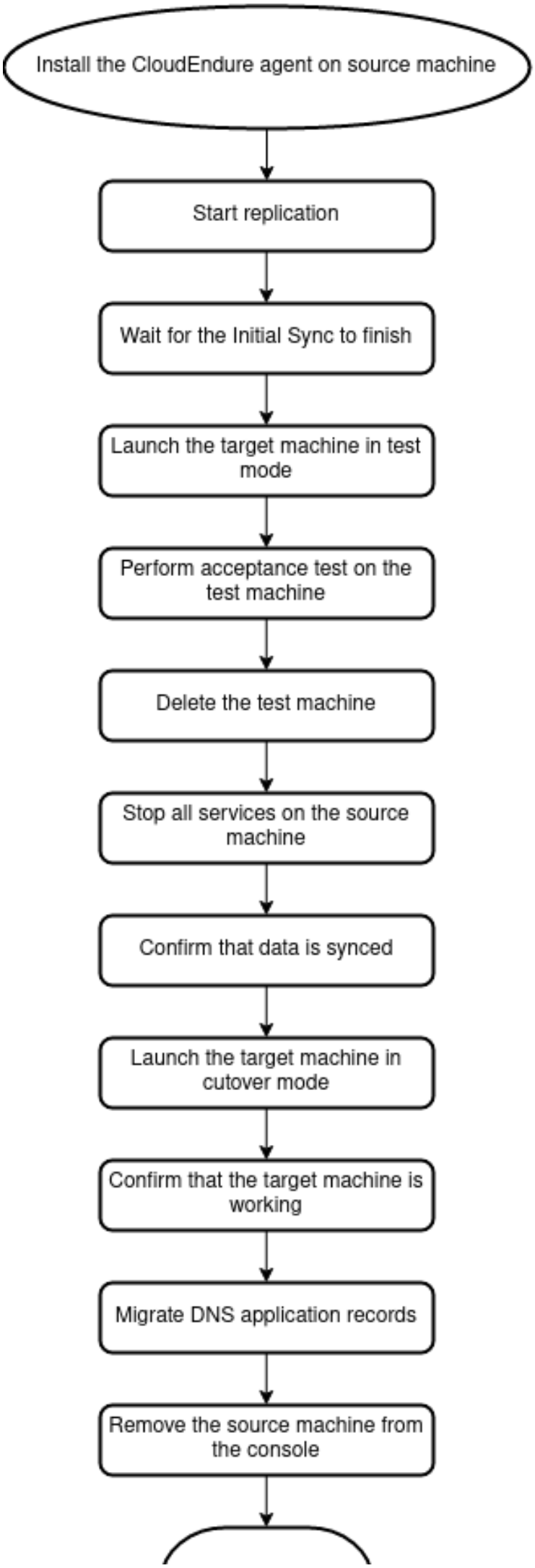
È diventato subito ovvio che, data la scadenza, un refactor applicativo sarebbe stato impossibile: era necessaria una soluzione veloce, ma che garantisse al cliente lo stesso livello di affidibilità. **L'opzione "rehosting"** a questo punto è rientrata nell'insieme delle possibili soluzioni.

Per essere in grado di procedere al rehosting avevamo bisogno di uno strumento che ci permettesse di migrare macchine virtuali on-premise al Cloud di AWS, fornendo anche la possibilità di effettuare test e che mantenesse i dati sincronizzati fino alla messa in produzione del nuovo ambiente.

Dopo una breve ricerca abbiamo deciso di utilizzare **CloudEndure**, un tool di una compagnia acquisita da AWS.

CloudEndure fornisce la possibilità di migrare macchine fisiche e virtuali al Cloud di AWS, mantenendo i dati sincronizzati e permettendo di effettuare test sulle istanze migrate prima del “cutoff” (il momento in cui la migrazione si ritiene conclusa)

Il diagramma di flusso per una migrazione standard effettuata utilizzando CloudEndure è:


Install the CloudEndure agent on source machine
Start replication
Wait for the Initial Sync to finish
Launch the target machine in test mode
Perform acceptance test on the test machine
Delete the test machine
Stop all services on the source machine
Confirm that data is synced
Launch the target machine in cutover mode
Confirm that the target machine is working
Migrate DNS application records
Remove the source machine from the console

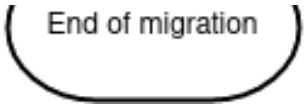


Diagramma di flusso per una migrazione standard con CloudEndure

Dopo aver concluso con successo un breve test utilizzando una configurazione comune (macchine virtuali Ubuntu 20.04) sono emersi i primi problemi: i sistemi operativi del cliente erano troppo vecchi e la loro esecuzione non era supportata sul Cloud AWS.

Abbiamo a questo punto deciso di procedere con un **replatforming in un ambiente di test** utilizzando una versione di sistema operativo supportata da CloudEndure e, dopo aver validato il corretto funzionamento delle applicazioni, di procedere con la migrazione.

Dopo alcuni tentativi infruttuosi abbiamo scoperto che la nostra unica possibilità era data dall'utilizzo di **Ubuntu 12.04**, la minima versione supportata da CloudEndure.

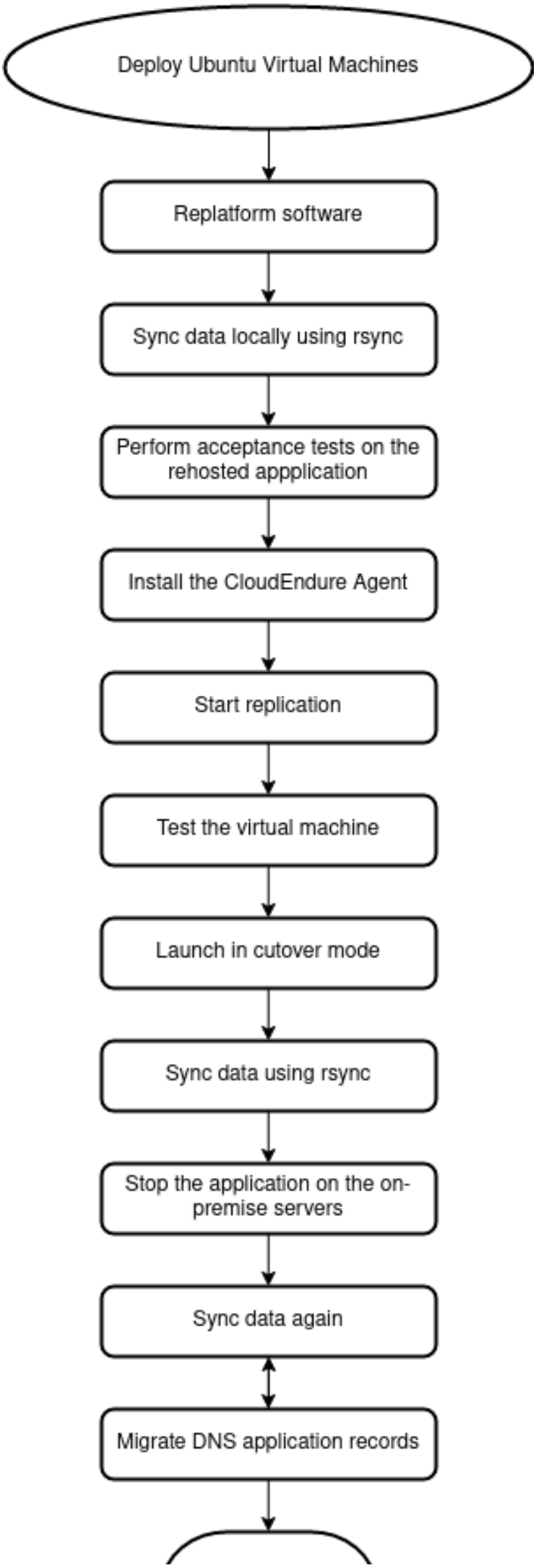
A causa di questa scelta non è stato possibile usufruire di famiglie recenti delle istanze EC2: i driver per la piattaforma Nitro non sono disponibili, per cui abbiamo dovuto ripiegare sulle famiglie m4 per l'ambiente di produzione.

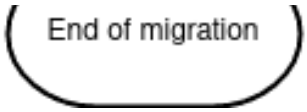
Per l'esecuzione delle istanze sono infatti utilizzati ambienti basati su hypervisor differenti: Xen per le generazioni precedenti, KVM in combinazione con la piattaforma Nitro per le famiglie di istanze più recenti.

Non è stato nemmeno possibile migrare il database sulla piattaforma RDS a causa della versione dell'engine troppo datata. Un cambio di engine in corsa avrebbe richiesto anche il cambio di alcune librerie utilizzate dal framework con risultati incerti.

Una volta effettuato il deploy delle macchine virtuali nell'ambiente di test per il replatforming, è stato possibile effettuare la migrazione utilizzando CloudEndure, mantenendo i dati sincronizzati utilizzando **rsync**.

Il workflow di migrazione è diventato:


Deploy Ubuntu Virtual Machines
Replatform software
Sync data locally using rsync
Perform acceptance tests on the rehosted appplication
Install the CloudEndure Agent
Start replication
Test the virtual machine
Launch in cutover mode
Sync data using rsync
Stop the application on the on-premise servers
Sync data again
Migrate DNS application records



New migration workflow

Non entriamo nei dettagli tecnici sull'utilizzo di rsync o del processo di dump/restore del database in quanto si tratta di comandi standard. Se avete dubbi o curiosità al riguardo segnalatecelo nei commenti e saremo felici di offrire un maggior dettaglio!

Dopo aver messo in atto il nuovo piano tutto è andato per il meglio con un unico disservizio pianificato di 3 ore dovuto al rallentamento del trasferimento dei dati causato dallo storage del provider sorgente. Dopo una attenta valutazione, le virtual machine sono state spente, fornendo al cliente un ambiente di produzione più stabile.

Takeaways

Come spesso accade, anche questo progetto ha dovuto adattarsi in corso d'opera a situazioni non previste e intraprendere una direzione del tutto in attesa e piuttosto limitata rispetto ai vantaggi che il piano originario avrebbe garantito contando sui servizi gestiti di AWS. Nonostante ciò, la buona notizia è che ora il progetto originale può iniziare e, finalmente, il processo di modernizzazione dell'applicazione può essere avviato.

Pur non essendo quasi mai una scelta ottimale, a volte una migrazione di tipo "Lift and Shift" è richiesta o quasi "obbligata" a causa di fattori esterni come tempo, costi di refactor o fattibilità tecnica. Occorre sempre tenere presente che il debito tecnico che molto spesso si accumula non aggiornando i sistemi, utilizzando librerie vecchie e versioni di linguaggi di programmazione non più mantenuti, ad esempio, nasconde delle insidie. Prima tra tutte un costo nascosto consistente che potrebbe manifestarsi e compromettere non solo la buona riuscita di progetti importanti di migrazione, ma addirittura precludere le infinite possibilità che il Cloud offre anche in termini di spinta innovativa.

Per maggiori informazioni su come procedere al refactoring e re-architecting delle applicazioni vi consigliamo la metodologia descritta nel sito della Twelve-Factor-app: anche se non state ancora prevedendo di migrare al cloud, può essere comunque utile per prevenire molti problemi futuri!

Vi siete mai trovati ad affrontare sfide simili o improvvise che vi hanno costretti ad una cambio di passo repentino? Raccontateci le vostre imprese DevOps!

A tra 14 giorni su **#Proud2beCloud** con un nuovo articolo!

Damiano Giorgi

Ex sistemista on-prem, pigro e incline all'automazione di task noiosi. Alla ricerca costante di novità tecnologiche e quindi passato al cloud per trovare nuovi stimoli.L'unico hardware a cui mi dedico ora è quello del mio basso; se non mi trovate in ufficio o in sala prove provate al pub o in qualche aeroporto!

Simone Merlini

CEO e co-fondatore di beSharp, Cloud Ninja ed early adopter di qualsiasi tipo di soluzione *aaS. Mi divido tra la tastiera del PC e quella a tasti bianchi e neri; sono specializzato nel deploy di cene pantagrueliche e nel test di bottiglie d'annata.
