The great escape: migrating from the on-prem to the AWS Cloud in a quick&pretty way with CloudEndure.

22 July 2021 - 5 min. read

Application Modernization | Cloud Migration | CloudEndure

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

said John Lennon in his song "Beautiful Boy".

In this article, we are going to see that sometimes there are unpredictable situations that can occur and, even if there's an accurate and detailed plan, our strategy needs to change quickly.

When **moving to the Cloud from an on-premise environment,** it is always necessary to refactor or re-platform your application so that you can fully take advantage of the new paradigm. Sometimes external factors can make your decisions change dramatically: you have to change everything to be able to achieve your goal.

A couple of months ago a customer asked us to help him migrate his e-commerce websites from his current hosting provider to AWS: we made a plan for an on-prem to cloud migration following **the "7R" rules**.

After an inventory of all systems and applications it turned out that a lot of **refactoring** and **re-architecting** was needed, no systems had to be repurchased, retained, relocated, or retired, the rehost option wasn't even considered because the customer was eager to change the infrastructure paradigm he was using, taking full advantage of the cloud in terms of elasticity, cost optimization, and operational efficiency.

There was a huge amount of technical debt due to old operating systems, obsolete versions of programming languages, and end-of-life database releases.

We were not scared: the project was challenging but the prerequisites were clear and the final infrastructure will use **containerized applications**, **RDS** databases, and a bit of **Amazon CloudFront + Amazon S3** static sites hosting.

The big effort was on the customer's development team side (supported by us): every application had to be refactored to be **stateless** and to perfectly integrate with **AWS Cloud Services** (like using S3 to host 5 Terabytes of static assets).

We agreed to kick-off the project after the Christmas sales period, giving the developers the opportunity to take confidence with the Cloud and be ready to quickly release bug fixes if needed during the traffic-intensive period.

Everything was going fine until Murphy's law kicked in: during Black Friday the hosting provider had a major outage, resulting in a total downtime of 2 days and causing a big loss of profit.

Once the damage was done there were also other problems and the migration priority was changed to "as soon as possible", with a deadline for the end of next month.

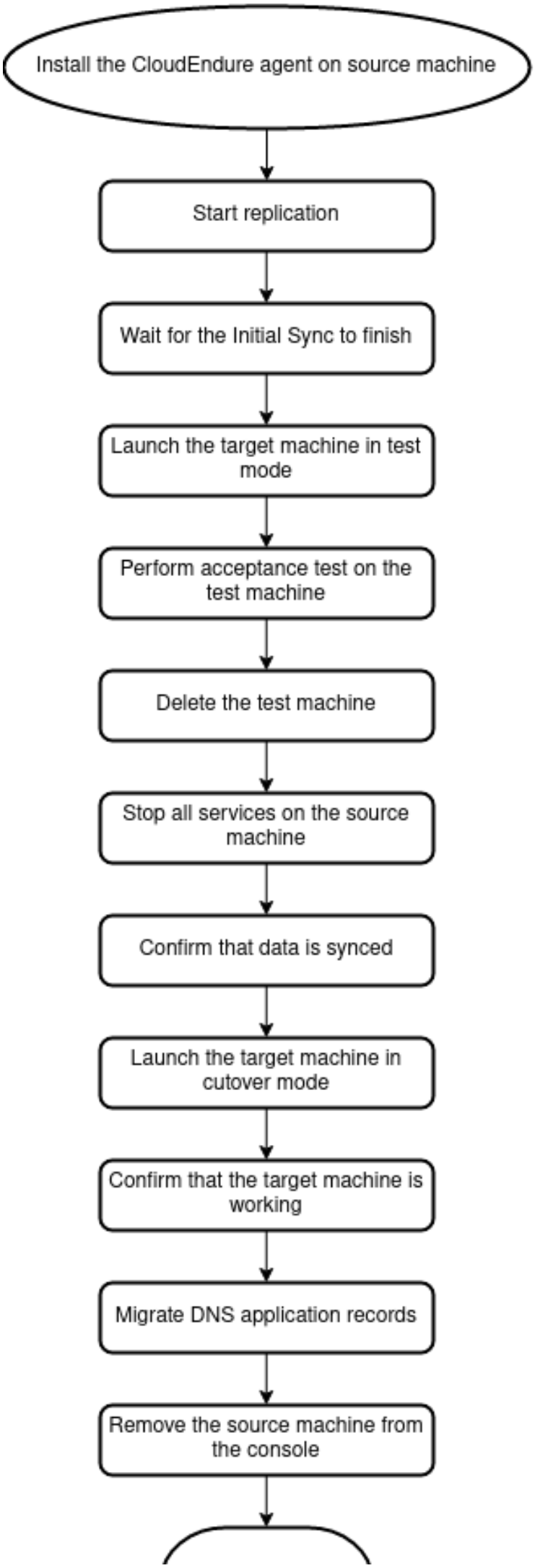
It was obvious that an application refactor wasn't an option: the deadline required something quick and, possibly, not so dirty: the **rehost option** went back in the pool of possible solutions.

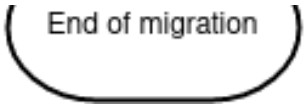
To be able to rehost everything we needed a tool that could help us to migrate on-prem virtual machines to the AWS Cloud, giving us the ability to test them and keep data in sync until we wanted to pull the trigger to switch the production environment.

A quick search led us to **CloudEndure**, an AWS company.

CloudEndure gives you the ability to migrate physical hosts and virtual machines to the AWS Cloud, keeping data synchronized and giving the opportunity to test the resulting instance until the "cutoff", when the migration is considered as complete.

The workflow for a typical CloudEndure migration is:


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



A typical CloudEndure migration workflow

After a quick successful test using a common configuration to validate the tool (Ubuntu 20.04 virtual machines), we started with the customer environment. The first problems started to arise: some operating systems were way too old to be able to run on AWS Cloud

We decided to do **a replatform in a test environment** using a Linux os release supported by CloudEndure and migrate them after validating that everything was working properly.

After some trial and error to accommodate requirements for old libraries and supported operating systems our only possible choice turned out to be **Ubuntu 12.04**, which is also the minimum version supported by CloudEndure.

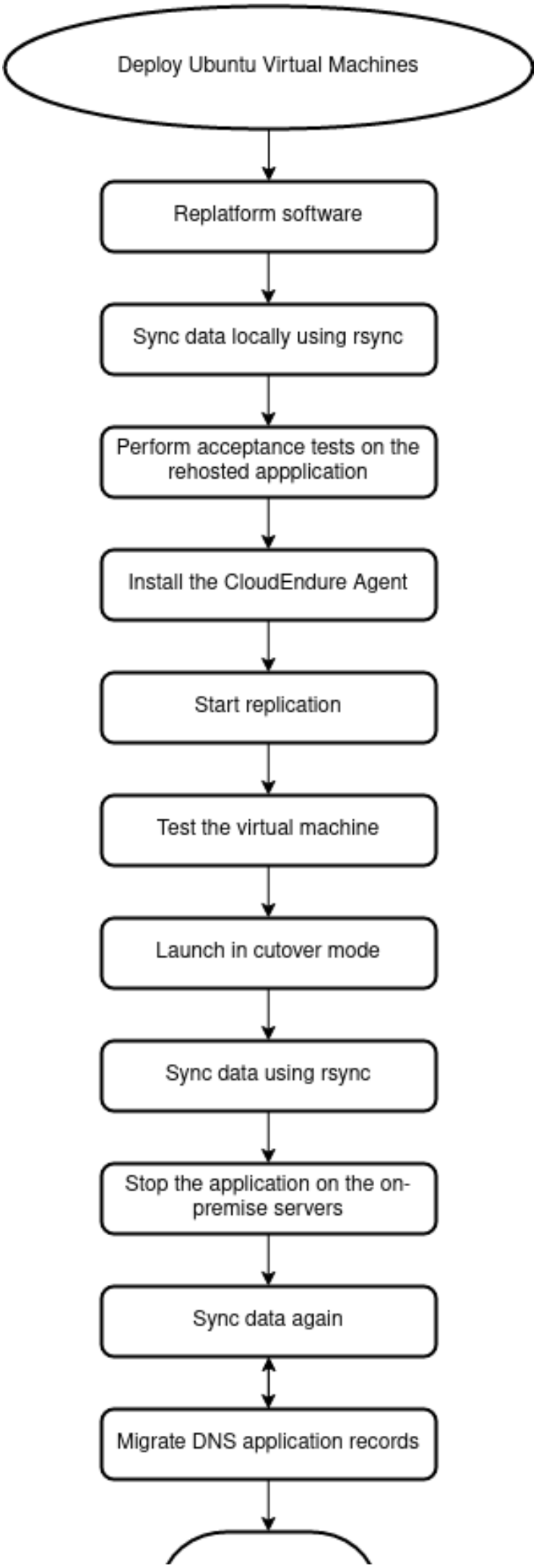
Because of this, we were unable to take advantage of the last generation of ec2 instances, as there were missing drivers for the Nitro platform. We had to use the m4 family for production use.

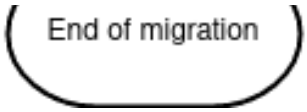
Different instance families run on different virtualization platforms: Xen-based on older generation and KVM based on the newest. The newest generations also can take advantage of the Nitro Platform.

We were also unable to migrate the database to RDS because of the engine version; upgrading it would have required a change of some libraries used by the application framework with unpredictable results.

Once we deployed the virtual machines to re-platform and adapt the application we were able to migrate them using CloudEndure, while data was kept in sync using **rsync**.

Our workflow turned out to be:


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

We will not enter into technical details of how rsync was used and the database dump/restore process because they are pretty standard commands.

After applying this plan everything worked fine with a scheduled downtime of 3 hours (due to the hosting provider storage that sometimes slowed down data transfer to ec2 instances); after a careful review, we were able to shut down the old virtual machines, giving the customer a more stable production environment.

Takeaways

As often happens, this project took an unexpected direction. Nevertheless, we were able to quickly respond to the new needs. Even if we did not take any advantage of all the managed services and features available on the AWS Cloud, the good news is that now the original migration project can start and, finally, we can continue modernizing the application.

Sometimes a "Lift and Shift" migration is required (or mandatory) because of external factors like time constraints, refactoring costs, or technical feasibility: always keep in mind that the cost of technical debt (leaving systems out of date, use old libraries and programming languages) is hidden but it will give you a big headache at some point in the future.

For more information on how to refactor and re-architect your application take a look at the Twelve-Factor app site: even if you are not planning to migrate to the Cloud it can help you to prevent a lot of trouble in the future!

Have you ever faced something similar in your DevOps life? Tell us about your extraordinary feat!

And see you again in 14 days here on **#Proud2beCloud**!

Damiano Giorgi

Ex on-prem systems engineer, lazy and prone to automating boring tasks. In constant search of technological innovations and new exciting things to experience. And that's why I love Cloud Computing! At this moment, the only "hardware" I regularly dedicate myself to is that my bass; if you can't find me in the office or in the band room try at the pub or at some airport, then!

Simone Merlini

CEO and co-founder of beSharp, Cloud Ninja and early adopter of any type of * aaS solution. I divide myself between the PC keyboard and the one with black and white keys; I specialize in deploying gargantuan dinners and testing vintage bottles.
