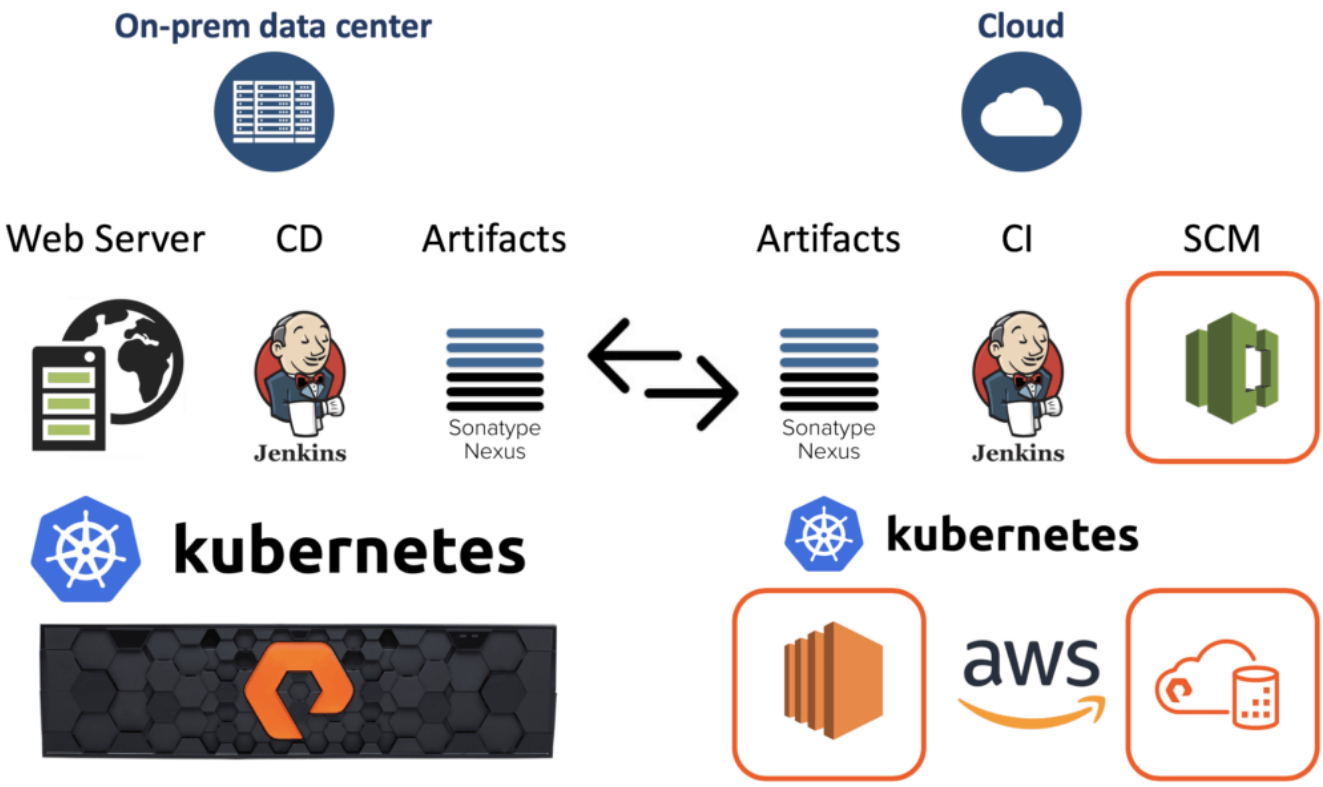
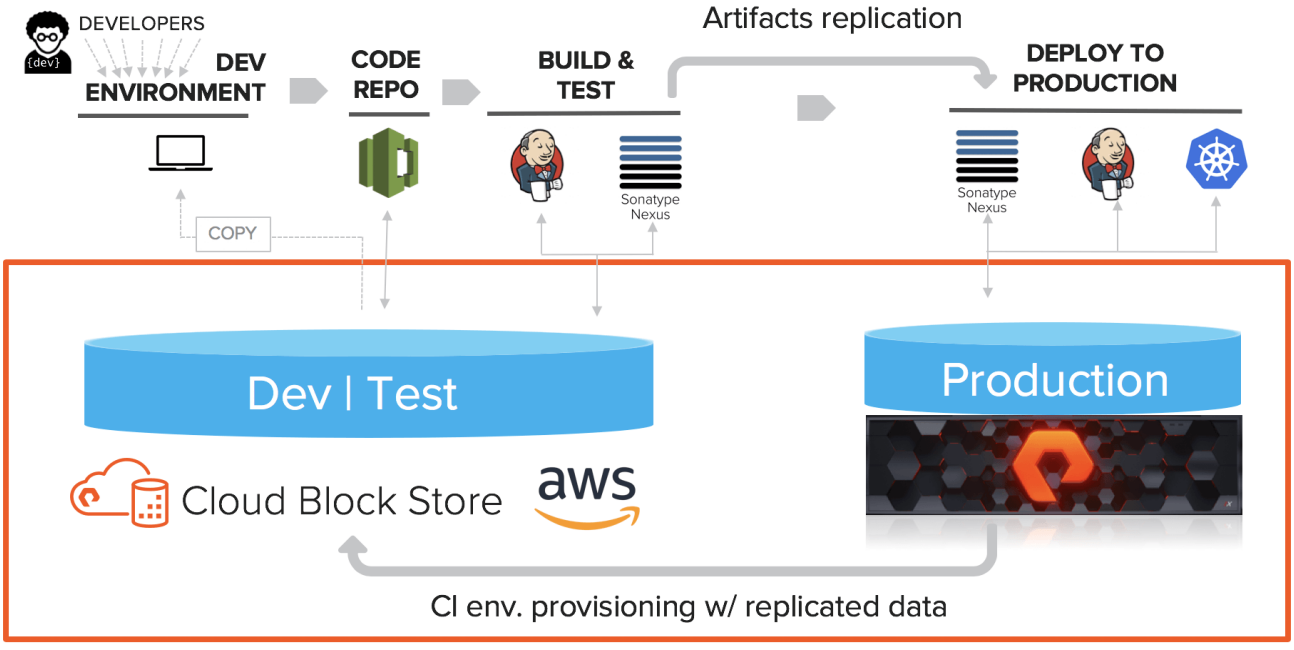
Say Hello to Hybrid CI/CD with Cloud Block Store





Introduction

A few years ago (in 2011, precisely), a famous Silicon Valley investor[1] you probably know coined the phrase software is eating the world[2]. 7 years later, this statement has almost become a cliché in the high-tech industry, as anything from data to cloud computing to SaaS is eating the world some way or another. But in all honesty, software has indeed influenced not only our daily lives but also the way businesses work. Information is moving faster – thanks to software. Work gets done more efficiently – thanks to software. Software deployment itself has been largely altered by new practices adopted through the growing popularity of agile software development methodologies, giving rise to the recent DevOps movement. And just as software is pervasive in our daily lives, so is the location where it is made and runs from. Once primarily hosted in private data centers, the cloud has made it, well, cloudy for us to know where the software we use actually runs from.

Some folks – mainly DevOps professionals – must know where software execution happens though. However, making sense of and orchestrating all the moving parts between fully owned data centers, private clouds and public clouds can easily give them a headache.

But it doesn't have to be a byzantine endeavor. Today, Pure Storage® adds its piece to this big hybrid cloud puzzle, with the promise of making data and software location-agnostic.

In this blog post, I would like to focus on a trending DevOps topic, Continuous Integration and Continuous Deployment/Delivery (a.k.a "CI/CD"), and explain how Pure Storage® can help DevOps professionals split their CI/CD stacks between on-premises and cloud environments with surprising ease. Don't expect any code at this point though! The objective of this post is conceptual in nature and we'll go through the use case scenario, the requirements for a new, hybrid and modern platform and the Pure Storage-powered solution using Cloud Block Store for AWS. In part 2, we will dig deeper into the implementation specifics and the code used to build a Hybrid DevOps CI/CD demo. For now, let us explore a real-life scenario where moving to a hybrid CI/CD model becomes a necessity, not just an intellectual exercise.

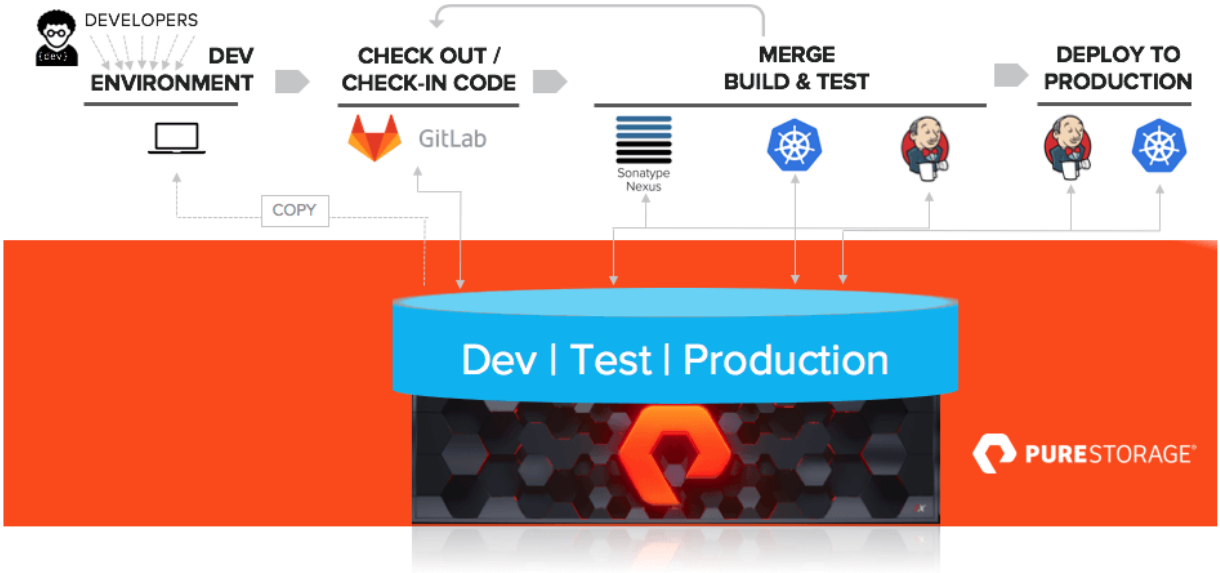
The situation

Jane works in the digital marketing department of GeekTrain.com, an online training company for developers. She heads the development team responsible for maintaining the corporate website and up until now, everything is done and runs on premises, from design to development to deployment. Their current CI/CD platform (represented below) runs on top of Kubernetes and leverages state-of-the-art CI/CD tools:

- GitLab to host their website code and creative assets
- Sonatype Nexus Repository OSS as the Docker registry (for their build containers) and the artifacts repository (to store the website code packages)
- Jenkins to run the CI (Continuous Integration) and CD (Continuous Deployment) pipelines

This 100% on-premises CI/CD platform backend is also powered by a Pure Storage FlashArray™ which provides the reliability, performance, capacity and storage reduction GeekTrain expects both to run its development builds, but also to serve the many terabytes of video content it makes available to its online visitors. The general CI/CD workflow Jane's team follows is fairly standard and straightforward. Here is a simplified version of it:

- Developers download the website code onto their local development boxes from the Git repositories (hosted in GitLab)
- They check in their code changes to GitLab
- Every check-in kicks off a build pipeline in Jenkins which, if successful, stores the build artifact into Nexus Repository Manager (a QA environment is also automatically refreshed with the latest build using a non-represented CD workflow)
- When the build running in QA has been manually verified by all, Jane kicks off a CD pipeline in Jenkins which pulls the latest artifact from Nexus and deploys in to the web servers (hosted in Kubernetes containers in the diagram below)



That has worked very well for the past year and the GeekTrain team plans to continue using this platform to host its public website in the foreseeable future. However, as the website becomes more sophisticated and business demands require more advanced customizations, relying solely on Jane's team has proved impractical, both from a skillset and cost perspective.

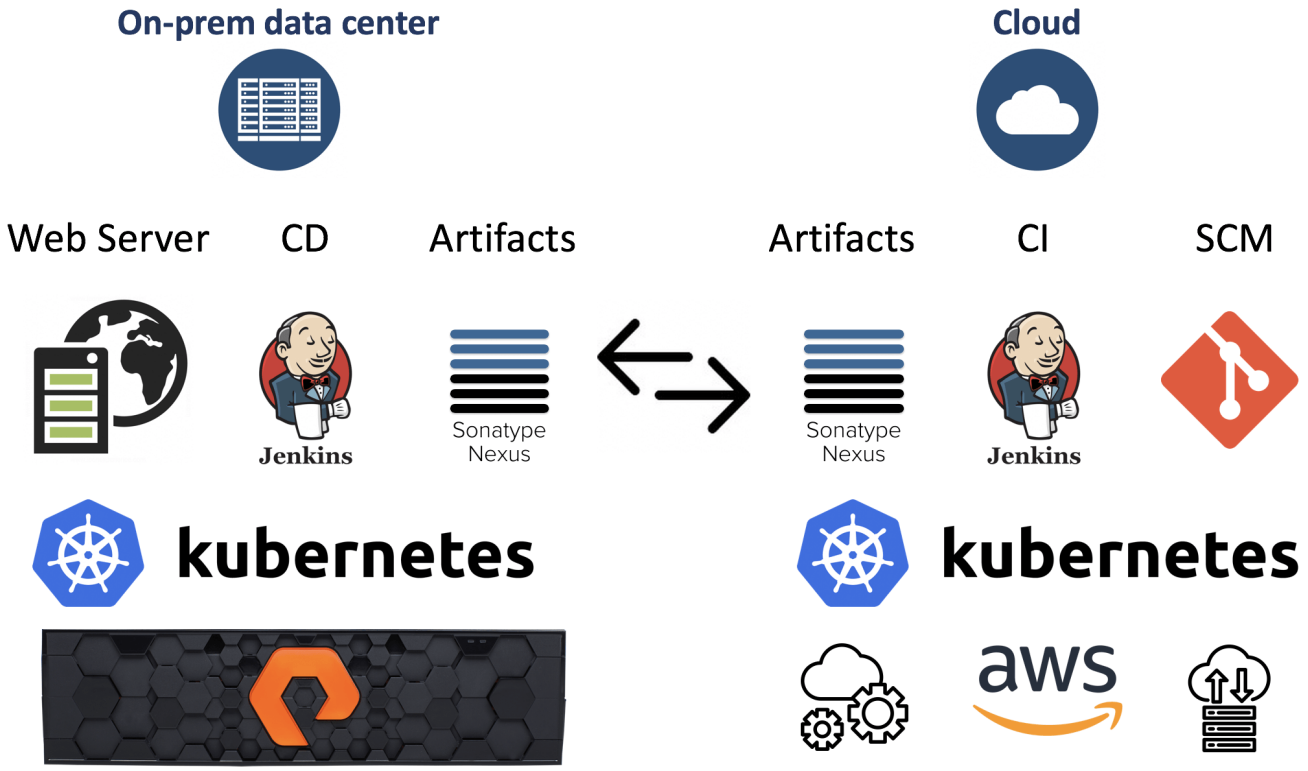
The necessary evolution to a hybrid model

To alleviate the situation, Jane has recommended that they externalize the most complex development tasks to a specialized outside firm, whose services they would only need a few times a year. As cost-effective as it is, implementing such a change creates another daunting challenge: how would the external developers and Jane's internal team work together efficiently on the same code base and reliably test each code change? Jane has explored a few options:

- To request that the external developers send their code changes daily through a file-sharing platform so that Jane's colleagues can check them in to the internal Git repository. This suggestion is met with stiff resistance from Jane's team, which rightfully considers this process to be highly inefficient and error-prone.
- To grant external developers VPN access to GeekTrain's internal GitLab environment so they can contribute their changes directly into the central repository. Unfortunately, Jane has received a firm no-go answer from the IT security team, which has a strict policy to only grant VPN access to full-time employees.

Facing these dead-end on-premises-only options, Jane has decided to explore cloud options, particularly Amazon Web Services, which a few IT teams at GeekTrain have started to use to offload some of their non-critical workloads. Jane also had a recent discussion with Bob, GeekTrain's CIO, who expressed concerns that Jane's CI/CD platform puts undue strain on its compute platform around the end of each financial quarter, when Jane's team frantically checks in code and runs a flurry of daily builds in order to meet their release commitments.

With these considerations in mind, Jane decides it's time to embrace a new, more flexible development and deployment model for GeekTrain's website, one that could accommodate both her and Bob's needs. The new platform architecture would conceptually look like the following:



In Jane's mind, the CI/CD platform would be split into two distinct CI (build/test) and CD (deploy) blocks:

- The cloud-hosted CI block would provide a Git-compatible Source Control Management (SCM) system (to make it easy to migrate the GitLab repository) that can be accessed by both GeekTrain employees and the external developers. It would also run the CI pipeline using the same on-prem process (modulo a few environment-specific parameters) and publish the build artifacts into Sonatype Nexus.
- The on-premises hosted CI block would continue to host the web servers, Jenkins as the CD tool and Nexus as the artifact repository manager. The Nexus artifacts created in the cloud would sync back to the on-premises datacenter and be deployed to GeekTrain's web servers using a specific CD pipeline running in Jenkins.

Ideally, that mixed platform could be spun up almost instantly as needed (typically around end of quarters where she plans to regularly hire the external developers). Also, to avoid any versioning issue, Jane also wishes to make the creation of the cloud CI platform as swift as possible and identical to the on-premises platform at the time it's spun up. In order to achieve this CI platform duplication, it is essential that the underlying data (build workflow and history, build container images, build artifacts, etc...) be seamlessly available both on-premises and in the cloud at any point in time.

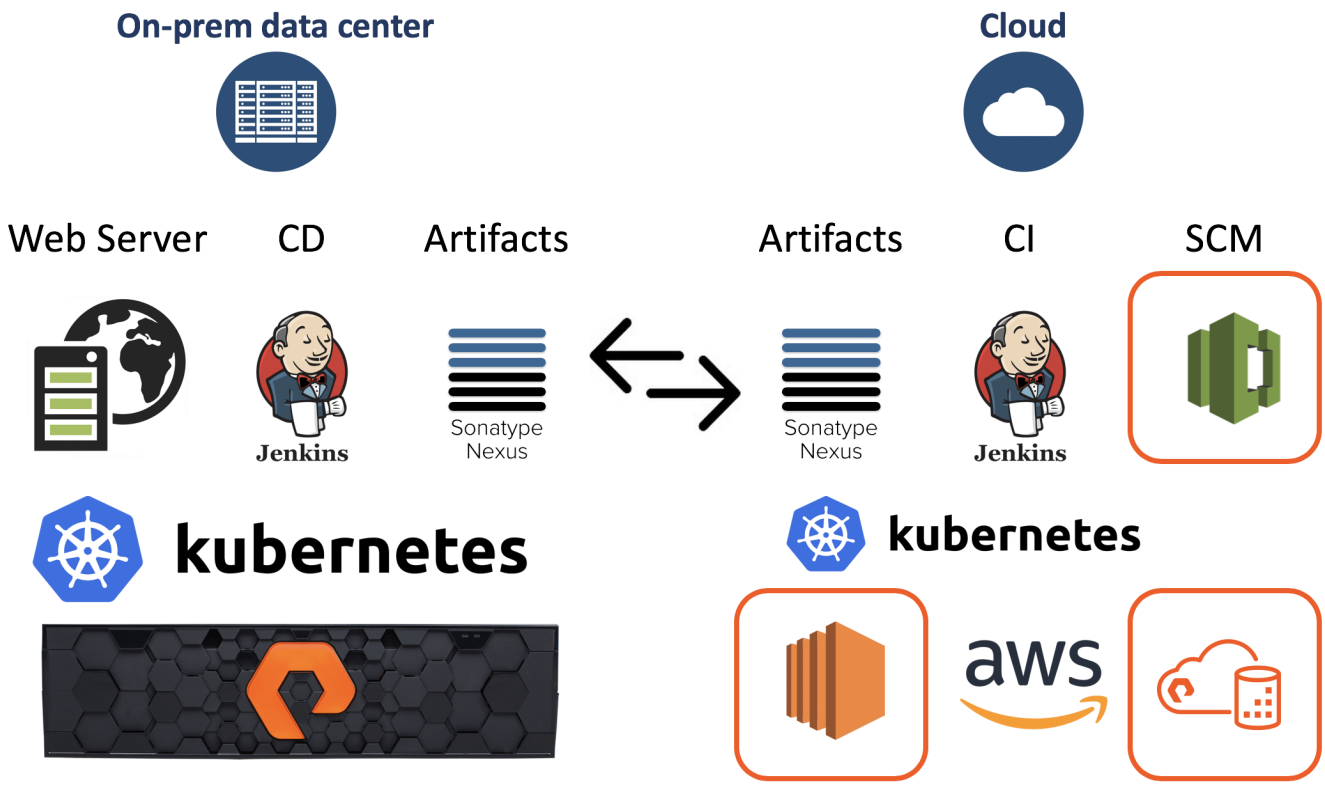
The solution

Fortunately, Jane's dreams can now be fulfilled using Pure Storage Cloud Block Store for AWS

Cloud Block Store allows Jane to seamlessly deploy an identical copy of her on-premises CI stack by combining the power of Kubernetes deployments with the data snapshot and replication capability of the Purity Operating Environment that runs on FlashArray™ and Cloud Block Store.

Additionally, AWS provides a Git-compatible Source Code Management service, AWS CodeCommit, which Jane happily picks as it allows her to easily grant external developers access to her team's Git repository and build a roadmap to integrate with other AWS DevOps services, such as AWS CodeBuild and AWS CodePipeline.

Overall, Jane's target architecture would look like the following:

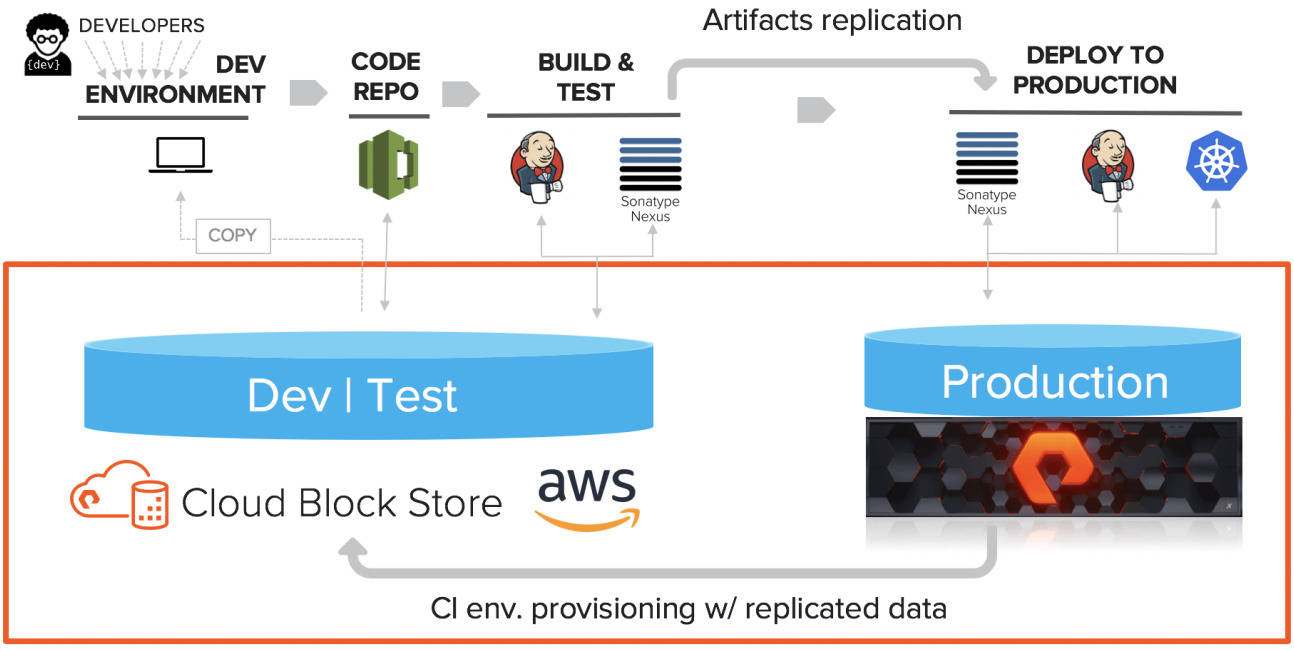


Using the automation features of Kubernetes, the replication capability of FlashArray™ to Cloud Block Store and the volume import facility of Pure Service Orchestrator, each initial CI cloud environment can be

provisioned in an EC2-hosted Kubernetes cluster as an identical copy of the on-premises environment. The EC2-hosted Kubernetes cluster is connected to a Cloud Block Store instance that can easily be provisioned using AWS Cloud Formation templates. Migrating a Git repository to AWS Commit is also very straightforward and can easily be automated.

Once the on-premises Git repository running on GitLab has been migrated to AWS CodeCommit, developers start committing their code to the new publicly available Git repository. This in turn triggers builds in the cloud-hosted Jenkins server and the publication of build artifacts in the cloud-hosted Nexus server.

Finally, using replication back from Cloud Block Store to FlashArray™, the Nexus artifacts generated in the cloud are seamlessly re-hydrated in the on-premises environment and can be deployed to the on-premises web servers using the same Jenkins CD pipeline previously used in the full on-premises model. A picture is worth a thousand words, so here is the new, hybrid CI/CD process Jane's team happily adopted:



The solution above has now become a reality thanks to the following key capabilities:

- The ability of FlashArray™ and Cloud Block Store to bi-directionally replicate snapshots between each other
- The FlashArray REST API which automates the process of restoring a snapshot to a named volume
- The ability of Pure Service Orchestrator to easily import an existing volume (created from that replicated snapshot) and mount it as a ready-to-use volume in a Docker container.

The way it works in this CI/CD context is straightforward:

- The Jenkins and Nexus data volumes are continuously replicated as snapshots from FlashArray™ to Cloud Block Store in AWS.
- Whenever Jane needs to provision a new replica of the CI environment, her provisioning script restores the latest snapshots into volumes in Cloud Block Store.
- Finally, the Persistent Volume Claims YAML definition files use a special "import" metadata

annotation to use the restored volumes as the source for the PSO-powered Kubernetes containers. This effectively creates an exact cloud copy of the on-premises CI stack (modulo a few deltas that may have happened since the last snapshot occurred).
- Conversely, the Nexus cloud data is replicated back to the on-premises FlashArray™ where it can be re-hydrated into the on-prem Nexus container using the same process.

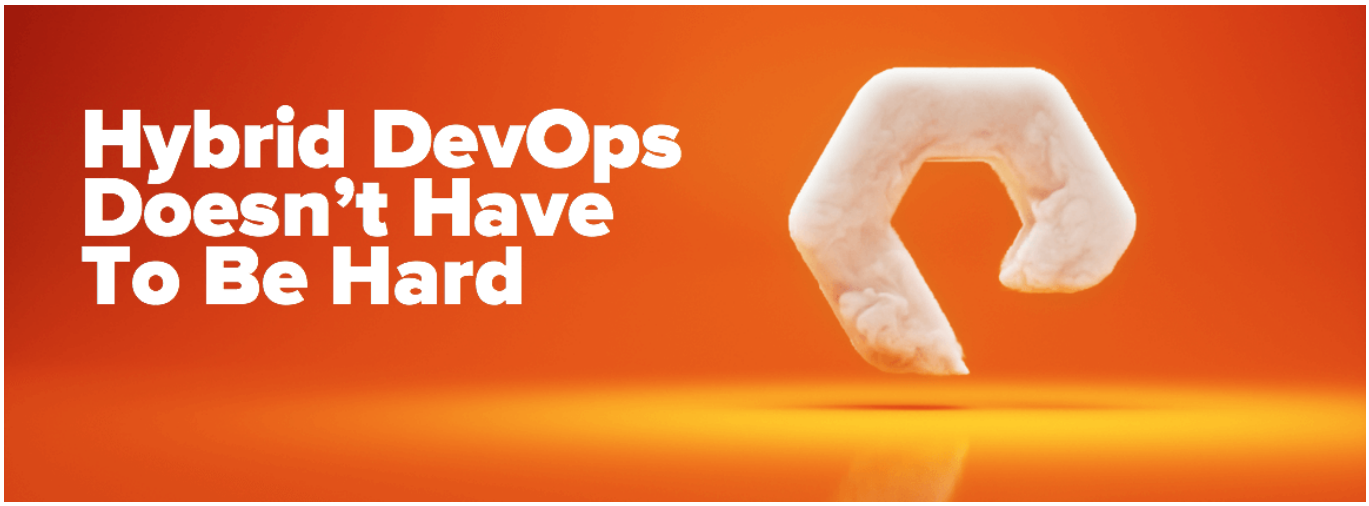
Conclusion

I hope this post made you want to try Cloud Block Store for AWS. At the time of writing, Cloud Block Store is available in limited public beta. You can sign up on our Cloud Block Store page if you're interested in being a beta participant.

Meanwhile, I encourage you to read the additional blog posts highlighting other Cloud Block Store scenarios you may find useful.

Last but not least, stay tuned for the upcoming Part 2 of this blog post where I will deep dive into the demo implementation of the above scenario.

Cherry on the cake, you will also get access to the demo scripts that will allow you to rebuild and test this hybrid CI/CD platform in your own FlashArray™ and Cloud Block Store environments!



[1] Marc Andreesen, of course.

[2] See A16Z blog