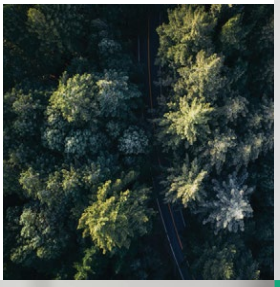
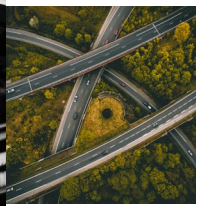




LIVRE BLANC

Simplification du stockage persistant pour les conteneurs

Accélération du développement d'applications tout en augmentant l'efficacité du DevOps

NetApp®

Le challenge : gagner en maturité DevOps

Une innovation constante. Des déploiements de produits plus rapides et de meilleure qualité. Des opérations rationalisées. Un nombre croissant de clients satisfaits.

Ce sont des objectifs universels pour bon nombre d'entreprises. Ce sont également des conditions préalables pour améliorer leurs résultats avec une croissance plus importante du chiffre d'affaires et une meilleure profitabilité.

Ces objectifs de réussite ne sont pas faciles à atteindre et ils sont souvent source de frustrations pour les entreprises en pleine transformation. Les développeurs d'applications, dans leur hâte d'accélérer les efforts de développement pour répondre aux besoins des clients, peuvent se sentir frustrés par les systèmes de gestion des tickets et les longues attentes avant que les ressources de stockage et de calcul soient mises à disposition. Les équipes chargées de l'infrastructure et des opérations (I&O) ont quant à elles du mal à traiter l'afflux des tickets de service et à contrôler l'utilisation de l'infrastructure.

Ces objectifs et difficultés liées à la croissance poussent les entreprises à appliquer des méthodologies agiles et légères pour accélérer le développement de produits et améliorer son efficacité. Les avantages de ces méthodologies sont considérables, mais pourraient être bien supérieurs si les processus et l'infrastructure IT étaient alignés pour répondre aux attentes d'efficacité et de vitesse.

C'est là que le DevOps intervient. Le DevOps est un système conçu pour accélérer la transformation des opérations liées à l'infrastructure IT et au développement d'applications.

Les entreprises qui parviennent à la maturité DevOps se caractérisent souvent par leur degré de maîtrise de six fonctionnalités clés :

- **Gestion du code, des artéfacts et des binaires.** Référentiels pour la conservation et la gestion de composants logiciels.
- **Gestion de la configuration.** Configuration et maintenance de l'infrastructure et des systèmes logiciels selon des méthodes connues.
- **Cloud/PaaS.** Utilisation de l'infrastructure de cloud public, privé et hybride pour prendre en charge le développement logiciel.
- **Conteneurs.** Environnements d'exécution d'applications légers mais extrêmement évolutifs.
- **Analyses.** Surveillance et gestion automatisées de l'infrastructure
- **Intégration continue/Déploiement continu (CI/CD).** Processus automatisés de bout en bout permettant aux développeurs d'écrire et de déployer automatiquement du code.

L'une de ces six fonctionnalités, les conteneurs, joue un rôle de plus en plus important pour les équipes chargées du développement d'applications et des opérations d'infrastructure qui cherchent à gagner en rapidité et en efficacité. Dans ce livre blanc, vous découvrirez comment progresser vers la maturité DevOps avec les conteneurs tout en relevant l'un de ses plus grands défis, à savoir la gestion du stockage persistant pour les applications avec état.

Le grand défi lié aux conteneurs : gérer le stockage persistant pour les applications avec état

Alors que les équipes DevOps commencent à envisager davantage de déploiements en production réels d'applications conteneurisées, de nouvelles difficultés voient le jour. L'une d'entre elles, et non des moindres, est la gestion du stockage persistant de données pour les conteneurs.

Pour en savoir plus sur la persistance des données avec les conteneurs ou la différence entre les applications avec état et sans état, consultez le petit encart page 4.

Dès l'adoption des conteneurs d'applications, il est rapidement apparu que la gestion du stockage de données poserait quelques difficultés. Dans une enquête menée par la « Cloud Native Computing Foundation » (CNCF), presque la moitié des participants (42 %) ont déclaré que la gestion des ressources et du stockage représentait un challenge clé dans l'adoption de conteneurs (Figure 1).[1] Un grand nombre de ces participants ont également signalé des problèmes récurrents avec la persistance du stockage. D'autres souhaitaient un accès plus simple au stockage réseau.

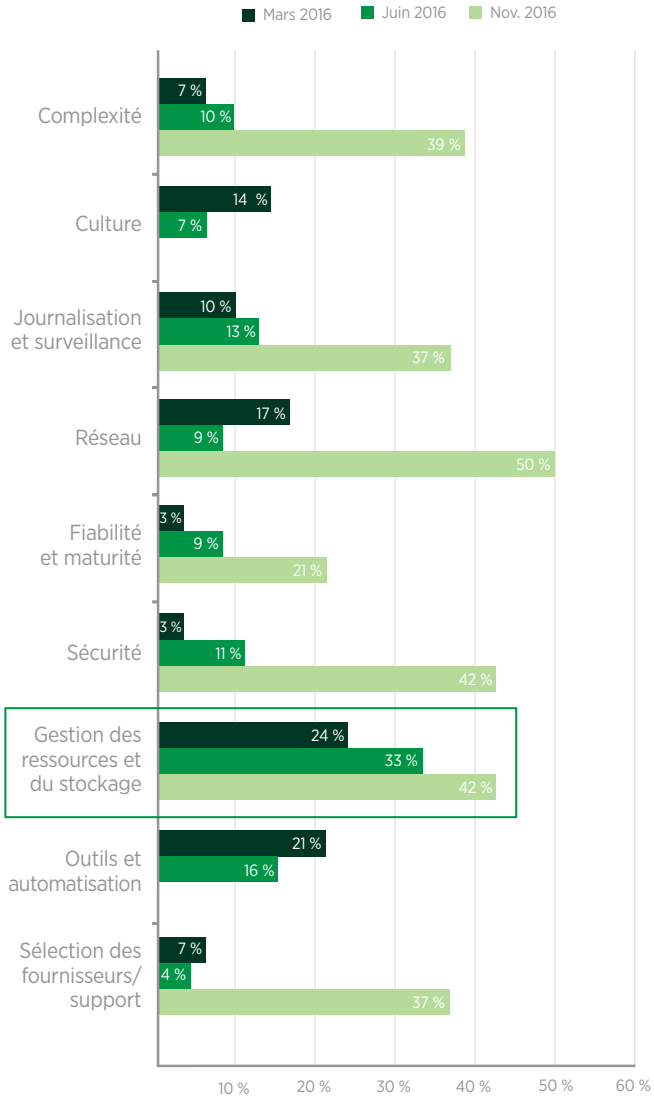


Figure 1 : Challenges liés à l'adoption de conteneurs (plusieurs réponses étaient autorisées).
Source : Cloud Native Computing Foundation.

Les plateformes de conteneurs ont donc commencé à chercher des solutions pour relever les défis liés au stockage persistant. Les premiers efforts en matière de provisionnement du stockage persistant ont été prometteurs, mais les solutions proposées sont demeurées rigides et trop manuelles. L'accélération du pipeline DevOps n'est pas apparue comme une solution envisageable, surtout lorsque les applications devaient être codées, testées et déployées dans des centaines ou des milliers de conteneurs.

Sur le plan de l'évolutivité, le provisionnement manuel du stockage pour des milliers de conteneurs a également atteint ses limites : trop fastidieux, trop sujet aux erreurs et trop difficile à maintenir. Il fallait trouver une meilleure solution.

Avant l'apparition des conteneurs, de nombreuses applications d'entreprise répondaient parfaitement à leurs besoins de stockage persistant de données via une connexion centrale au système de stockage haute performance partagé. Désormais conteneurisées, les applications avec état ne pourraient-elles pas fonctionner tout aussi bien, si elles pouvaient accéder facilement à des fonctions similaires de stockage partagé ? Et comment rendre ce processus « facile » ?

Un bon début : provisionnement basique du stockage persistant pour les conteneurs

Les environnements de conteneurs comme Docker et Kubernetes ont répondu initialement au challenge du stockage persistant avec un mécanisme pratique et semi-automatisé. Ce mécanisme permettait aux utilisateurs de créer et « demander » un volume de stockage persistant qui devait être utilisé par un ou plusieurs processus de conteneurs. Pour cela, il fallait qu'un administrateur de stockage crée d'abord des volumes persistants à partir de plusieurs ressources de stockage réseau sous-jacentes.

Kubernetes : le provisionnement basique en action

Pour Kubernetes, le processus adoptait des mécanismes basés sur un code associés à un volume persistant et à une demande de volume persistant correspondante.

Un volume persistant statique (par exemple, un volume avec 8 Go de capacité de stockage 100 % Flash) devait alors être d'abord créé par un administrateur. Ensuite l'utilisateur de l'application (ou le développeur) pouvait demander ce volume persistant avec quelques lignes de code.

Cette demande et cette liaison de volumes persistants et de demandes de volume persistant étaient intéressantes pour les environnements de conteneurs parce qu'elles permettaient à des applications conteneurisées de commencer à consommer du stockage réseau persistant par le biais du code.

Cependant, même si cette approche de provisionnement de base représentait un bon début pour la consommation du stockage réseau, il restait encore un certain nombre de défis à relever. Le plus important d'entre eux étant le besoin d'automatisation.

Le verdict : quelques progrès, mais encore trop d'attente

L'utilisation de volumes persistants préprovisionnés avec leurs demandes de volume persistant correspondantes représentait une excellente première étape dans la résolution des problèmes de stockage persistant des conteneurs. Cependant, une grande partie du processus de provisionnement du stockage nécessitait encore un effort largement manuel, statique et répétitif de la part des équipes de développement et d'opérations. Cela représentait un frein pour le développement d'applications et les équipes DevOps qui cherchaient à réduire les transferts et à automatiser leurs efforts pour atteindre l'intégration et la livraison continues :

Brève présentation des conteneurs, de la persistance des données et de la différence entre les applications avec état et sans état

Par nature, les conteneurs sont sans état. Leur contenu est éphémère. Cela signifie qu'il est possible de démarrer, stopper, puis redémarrer rapidement les applications ou les processus relatifs à un conteneur pendant une session donnée. Par défaut, cela signifiait que les données créées alors qu'un conteneur était actif seraient détruites dès la désactivation ou la destruction du conteneur.

Malheureusement, lorsque les équipes DevOps ont cherché à développer et déployer davantage d'applications en production sur des plateformes de conteneurs, de nouveaux besoins sont apparus. Il fallait en effet trouver le moyen de conserver ou d'assurer la persistance des données après la disparition d'un conteneur.

Beaucoup ont rapidement compris que presque toutes les applications comptaient au moins un processus ou un microservice qui avait besoin d'un volume persistant pour des données avec état qui devaient persister après la fin de vie d'un conteneur donné.

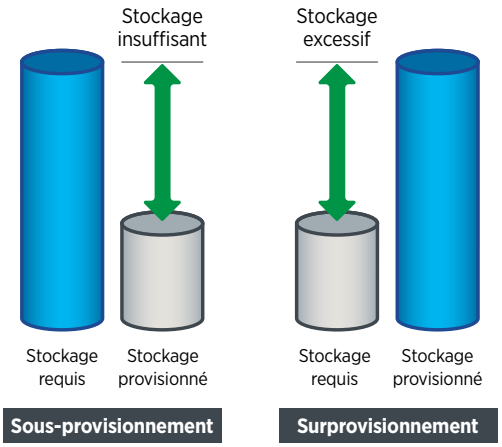
Voici quelques exemples d'applications avec état nécessitant la persistance des données :

- **Environnements de bases de données.** Un conteneur de base de données doit pouvoir conserver son stockage pour son datastore. Compte tenu de la nature éphémère des conteneurs, cela n'est pas possible sans aide. Le stockage local ne représentait pas non plus une bonne option. Si le conteneur était déplacé ou détruit, l'accès à ces données était perdu.
- **Données d'environnement ou de session.** Les applications avec état collectent et enregistrent souvent les attributs de l'environnement applicatif ou les données de sessions client (avec état). Ce fonctionnement est similaire à celui des données historiques pour améliorer l'expérience client. Dans ce cas, la prochaine fois que le client interagira avec l'application, des données pertinentes lui seront présentées ou la manipulation des données créées lors d'une session précédente sera améliorée.

Le besoin de partager les données avec état s'est également fait sentir et est apparu tout aussi important que celui de les préserver. Les entreprises qui conçoivent des applications conteneurisées ou migrent vers ces applications ont vite compris la nécessité pour tout le monde, du développement aux tests et aux opérations, d'accéder aux mêmes données à partir des mêmes ressources de stockage réseau centralisées. Les volumes persistants conservés sur le stockage réseau permettaient également de s'assurer que les applications conteneurisées étaient mieux protégées par les fonctions de stockage haute performance, et que ces applications étaient plus disponibles, fiables et sécurisées, avec une meilleure protection des données.

- **Demandes de stockage manuelles.** Les développeurs d'applications ou les équipes QA travaillant sur le code ou testant les sprints devaient interrompre leur activité en cours pour demander à l'administrateur du stockage de créer un volume persistant. S'agissant d'applications de conteneurs plus importantes, cela pouvait représenter des centaines, voire des milliers de demandes de stockage régulières.
- **Frustration liée à l'attente avant la mise à disposition du stockage.** Une fois les demandes émises, les équipes de développement/QA devaient attendre qu'elles soient approuvées ou que l'administrateur trouve un moment pour créer manuellement un ou plusieurs volumes persistants.
- **Utilisation inefficace du stockage.** Avec le stockage préprovisionné pour les volumes persistants destinés au développement et au QA, l'administrateur du stockage courrait le risque de provisionner une capacité de stockage insuffisante (sous-provisionnement) ou excessive (surprovisionnement) (Figure 2). Il pouvait aussi réserver trop ou trop peu d'IOPS par volume :
  - **Risque de sous-provisionnement.** Une capacité insuffisante de stockage provisionné à l'avance pour les volumes persistants pouvait entraîner un goulot d'étranglement qui limite les développeurs dont les besoins excèdent les ressources déjà disponibles.
  - **Risque de surprovisionnement.** Les ressources de stockage surprovisionnées, désormais bloquées dans un ou plusieurs volumes persistants, risquaient d'être gaspillées si elles étaient liées à une demande de volume persistant nécessitant beaucoup moins de capacité ou de performances. (Souvenez-vous de ces anciens silos de stockage surprovisionnés et sous-utilisés.)

- **Utilisation Inefficace des ressources de l'administrateur.** En cas d'efforts de développement d'applications multiples, ceux qui étaient liés à l'administration du stockage ou du cluster pouvaient former un nouveau goulot d'étranglement dans un ou plusieurs pipelines DevOps. Si plusieurs nouvelles demandes de volume arrivaient à la dernière minute, l'administrateur devait encore effectuer plusieurs étapes manuelles dans la configuration et la création de chaque volume persistant statique. Cette pratique n'aidait pas non plus les administrateurs chargés de l'automatisation de l'infrastructure et de la surveillance de sa consommation globale.



(Figure 2) Le risque de sous-provisionnement ou surprovisionnement.

Si cela vous rappelle le provisionnement de stockage à l'ancienne, sachez que vous n'êtes pas le seul à faire ce rapprochement. En 2017, Andrew Sullivan, Ingénieur marketing et technique, NetApp, terminait sa présentation lors du Tech Field Day, DevOps Through Desired State, en exprimant sa compassion avec les équipes DevOps qui voulaient se libérer de ce type de provisionnement. **« Le stockage n'est pas une ressource que je devrais accepter comme elle est, et quémander à l'équipe de stockage pour obtenir plus de capacité. »** a-t-il déclaré. **« Je ne veux pas provisionner du stockage. En 2017, je ne veux pas consommer du stockage de la même façon que je le faisais en 1989. »** [2]

Heureusement, NetApp et d'autres entreprises prenant en charge les environnements de conteneurs savaient qu'il devait y avoir une meilleure façon pour les équipes DevOps de consommer du stockage à la demande et de le provisionner de façon dynamique en tout lieu et à tout moment selon les besoins.

Classes de stockage, pools de stockage et Trident : le stockage persistant à la demande

Ce qu'évoquait Andrew Sullivan dans la dernière partie de sa présentation était la nécessité de mieux provisionner les ressources de stockage. Cela correspondait à la promesse initiale des conteneurs : des fonctions créées et déployées de manière dynamique, à la demande, sans intervention manuelle ou presque.

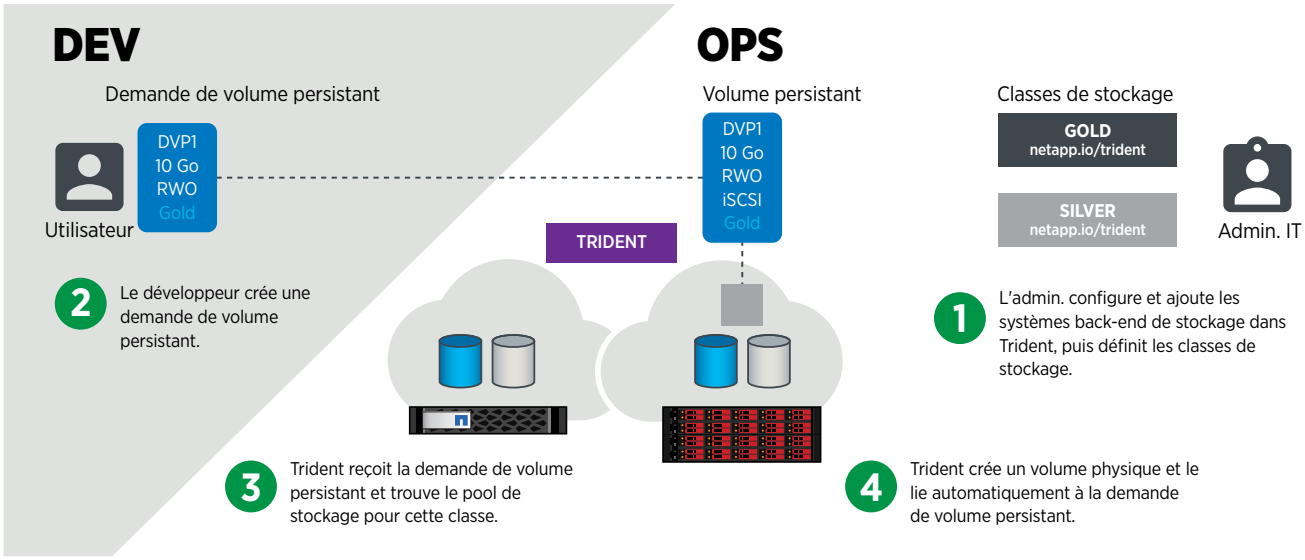
Pour Kubernetes, cela signifiait l'introduction d'un provisionnement de stockage dynamique par le biais d'un autre concept : la classe de stockage. Cela signifiait également beaucoup d'automatisation chez les fournisseurs de stockage, tels que Trident pour Kubernetes :

- **Stockage persistant à la demande.** Le projet open source Trident, développé par NetApp, est un fournisseur de stockage dynamique qui permet aux applications conteneurisées d'accéder à la demande aux volumes persistants du stockage NetApp® dont elles ont besoin, quand elles en ont besoin. Fini les temps d'attente.
- **Libération de la puissance du stockage sous-jacent**. Les environnements de conteneurs utilisant Trident sont capables d'exploiter des fonctions de stockage performantes provenant de plateformes de gestion des données NetApp performantes (telles que NetApp HCI, ONTAP®, NetApp SolidFire® Element® OS, SANtricity®).
- **Pas seulement pour Kubernetes.** Une fonctionnalité similaire pour les volumes persistants est également disponible via Trident pour Docker et OpenShift.

Un catalogue de classes de stockage peut-il être aussi simple ?

Avec Trident, les développeurs dans les environnements Kubernetes peuvent désormais provisionner de façon dynamique des volumes persistants uniquement en demandant une classe de stockage provenant d'un pool virtuel de stockage sous-jacent.

Voyons maintenant comment cela fonctionne. Avec les classes de stockage, le provisionnement de volumes persistants se fait automatiquement à la demande, en utilisant du code. Il suffit à l'utilisateur de faire une demande de volume persistant en précisant une classe de stockage spécifique, comme Gold, Silver ou Bronze.



(Figure 3) Provisionnement dynamique de volumes persistants avec Trident.

Remarque : la configuration des attributs de classes de stockage et des conventions de nommage sous-jacents est une fonction back-end effectuée généralement par un administrateur lorsqu'il crée un catalogue initial de classes de stockage. Les classes de stockage d'une entreprise peuvent tout simplement être nommées Dev, Staging, et Production, ou encore Fast et Slow. Pour obtenir plus de détails sur ces types de fonctions de configuration, consultez la documentation Trident.[3]

Un volume persistant est alors créé à partir de cette classe de stockage dans le pool de stockage NetApp sous-jacent. Il est ensuite lié à la demande de volume persistant de l'utilisateur. Ce dernier n'a donc plus besoin d'être informé sur le stockage sous-jacent. Trident s'occupe de ces détails. C'est tout !

Voir Trident en action

La meilleure façon de comprendre la valeur de Trident est de voir son provisionnement dynamique en action. Prenez quelques instants pour regarder l'une des démonstrations suivantes :

- **Démonstration rapide** (3:20 min) : L'utilisation de Trident pour le provisionnement de stockage dynamique avec OpenShift [4]
- **Démonstration plus longue** (23:53 min) : La gestion des données persistantes dans Kubernetes [5]

Que représente ce type de fonctionnalité de provisionnement pour les différents membres des équipes IT et DevOps ?

- **Pour les équipes de développement ou QA.** Fini l'attente des tickets de service et des approbations de demandes de stockage. Fini aussi les transferts. La consommation du stockage est conforme au niveau de service promis. Le stockage est provisionné de façon dynamique à l'aide d'interfaces de codes connues et mis à disposition au moment et à l'endroit où vous en avez besoin. Les développeurs disposent désormais d'un système de provisionnement automatique dynamique et flexible qui leur offre plus de liberté tout en maintenant le contrôle des opérations.
- **Pour les administrateurs IT ou de stockage.** Fini les demandes de provisionnement de stockage de dernière minute et les innombrables tickets de service. L'administration du stockage est réduite. L'évolutivité de l'infrastructure est plus automatisée. Le contrôle de la consommation du stockage est plus prévisible et la surveillance des ressources plus facile.
- **Pour les responsables IT.** Accélération de la fourniture des produits, amélioration des processus et économies significatives des ressources.

L'avenir du DevOps

DARZ, fournisseur de services IT complet, facilite l'agilité DevOps grâce à son offre Docker & Container-as-a-Service. Cette offre est basée sur le stockage 100 % Flash NetApp et Trident pour Docker. Les clients peuvent rapidement lancer ou arrêter des conteneurs d'applications sans avoir besoin d'un système d'exploitation complet, et avec des besoins en ressources de calcul divisés par quatre.

Avec un environnement conteneurisé agile et flexible, les clients peuvent raccourcir les cycles de test, accélérer le développement et déployer de nouveaux produits plus rapidement. En simplifiant les interactions entre le conteneur et le système de stockage à l'aide du panneau de commandes du volume Docker, Trident facilite la gestion des données persistantes dans un environnement Docker.[6]

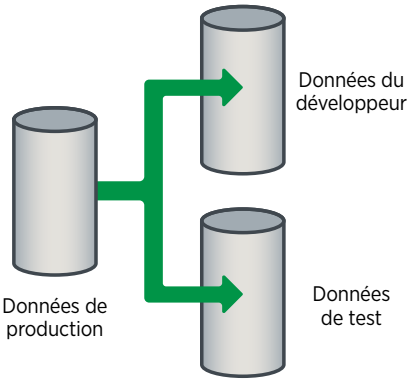
Transformer votre façon de travailler

Au-delà de leur capacité à provisionner de façon dynamique les volumes persistants, les outils comme Trident permettent également aux équipes DevOps de faire beaucoup plus de choses.

Elles bénéficient notamment de l'accès à la demande, via du code, à des fonctionnalités d'efficacité du stockage NetApp, comme les copies Snapshot™ et le clonage. Ces fonctionnalités peuvent changer la donne pour les équipes de développement ou QA qui doivent faire plus en moins de temps et avec un minimum de ressources.

L'utilisation des copies Snapshot et du clonage via NetApp et Trident présente de nombreux avantages :

- **Création rapide de clones en temps réel de datasets de production complets.** Cela signifie qu'il est possible de créer en quelques secondes de nouveaux espaces de travail pour les développeurs ou les tests, de façon dynamique et avec seulement quelques lignes de code. Voir Figure 4.
- **Restauration facile et rapide de données.** Avec les copies Snapshot, les développeurs peuvent rapidement restaurer les données dans une version précédente. Cette manipulation est pratique pour tester le code ; les développeurs peuvent ainsi rapidement effectuer des itérations sans craindre de devoir recréer les datasets de test.
- **Gains de capacité de stockage avec les copies ou clones Snapshot.** Les entreprises qui font de multiples clones de leurs données de production pour le développement ou les tests réalisent souvent 40 à 90 %[7] de gains de capacité de stockage.



Clonage pour prendre en charge rapidement les workflows de développement et test

(Figure 4) La performance du clonage NetApp pour accélérer les workflows DevOps.

Pour en savoir plus sur le fonctionnement du clonage NetApp et de Snapshot avec Trident :

- Clonage de volumes en libre-service avec Kubernetes[8]
- Copies Snapshot et restauration de volumes en libre-service avec Trident[9]
- Dans ce podcast Tech ONTAP, Jeff Steiner, expert Oracle de NetApp, explique comment cloner une grande base de données Oracle en seulement 22 secondes grâce à Docker avec Trident[10]

Une aide pour vous et votre entreprise

Résoudre des problèmes pour répondre aux besoins des développeurs et des ingénieurs n'a rien de nouveau pour NetApp, c'est au contraire toute son expérience. En tant que société, nous avons rapidement compris que l'infrastructure de stockage pouvait faciliter et accélérer le travail des développeurs et des ingénieurs, et servir les objectifs plus larges de leur entreprise. Chez NetApp, notre mission est de concevoir et proposer à nos clients des solutions pour faciliter la gestion des données et la consommation du stockage. Pour nous, les écosystèmes ouverts comme les conteneurs représentent un domaine émergent qui ouvre la voie à l'innovation. Et cette innovation est soutenue par les membres de sa communauté. NetApp fait partie de cette communauté et s'engage activement dans le développement de son innovation. Pour atteindre cet objectif, NetApp recherche de multiples façons de faciliter la consommation de stockage pour les membres de la communauté, où et quand ils en ont besoin. Aujourd'hui, NetApp est fier de permettre une consommation de stockage fluide dans un grand nombre d'écosystèmes ouverts. Nous continuons à développer l'un des ensembles d'API et d'intégrations les plus complets du secteur pour des environnements tels que Docker, Kubernetes, OpenShift, OpenStack, Ansible, Chef et Puppet, entre autres.

Trident est un exemple parmi d'autres des efforts déployés dans ce domaine. Nous vous encourageons à essayer Trident avec les environnements de conteneurs de votre entreprise pour découvrir les gains d'efficacité et les économies que vous pourriez réaliser dans votre pipeline DevOps.

Pour aller plus loin

Pour en savoir plus sur Trident et toutes les intégrations DevOps de NetApp, nous vous invitons à consulter les ressources suivantes :

En savoir plus sur NetApp et Trident

Solutions NetApp pour les conteneurs	
Présentation des fonctionnalités Trident : Présentation de Trident	
Présentation du stockage persistant Kubernetes avec Trident	
Clonage : Présentation du clonage de volumes avec Trident pour Kubernetes	
Documentation Trident	
Téléchargement de GitHub pour Trident	

En savoir plus sur NetApp et le DevOps

Solutions NetApp pour le DevOps	
thePub (netapp.io)	
Channel NetApp Slack (netapp.io/slack)	
@NetAppPub	

Notes :

[1] « Meeting Challenges in Using and Deploying Containers », par Sarah Conway, 27 avril 2017, Cloud Native Computing Foundation, https://www.cncf.io/blog/2017/04/27/meeting-challenges-using-deploying-containers/. Reproduit avec ajout d'un cercle rouge, sous la licence Creative Commons CC-BY 4.0.
[2] « DevOps Through Desired State », présenté par Andrew Sullivan, NetApp, Jour 14, 11 mai 2017, Tech Field Day, https://www.youtube.com/watch?v=btLZl7M6gnY&list=PLinuRwpnsHacYmunO7zyES6SyrsrFfu5O&index=4.
[3] La documentation Trident la plus récente est disponible sur le site https://netapp-trident.readthedocs.io/.
[4] « Using Trident for Dynamic Storage Provisioning with OpenShift », Démonstration en ligne, 3:20 min, par The Pub @ NetApp, 17 fév. 2017, https://www.youtube.com/watch?v=97VZWYssL2E.
[5] « Managing Persistent Data in Kubernetes », Démonstration en ligne, 23:53 min, par The Pub @ NetApp, 15 mai 2017, https://www.youtube.com/watch?v=XIuN91vG2wM.
[6] « DARZ Docker & Container-as-a-Service Drives Digital Transformation Through DevOps », témoignage client, NetApp, 2017, https://www.netapp.com/fr/media/cs-darz-devops.pdf.
[7] « How NetApp IT Shortened Development Cycles Using FlexClone », blog de la communauté NetApp, par Gopal Parthasarathy, 8 oct. 2015, https://community.netapp.com/t5/Technology/How-NetApp-IT-Shortened-Development-Cycles-Using-FlexClone/ba-p/110581.
[8] « Trident 18.01 beta 1: Introducing volume cloning to Kubernetes! » par Garrett Mueller, NetApp, 14 déc. 2017, https://netapp.io/2017/12/14/trident-18-01-beta-1-introducing-volume-cloning-kubernetes/. (Voir également : « Trident 18.01 is Here » par Andrew Sullivan, NetApp, 25 janvier 2018, https://netapp.io/2018/01/25/trident-18-01/).
[9] « Self-Service Data Recovery using Trident and NFS » par Andrew Sullivan, 3 avril 2018, NetApp,https://netapp.io/2018/04/03/self-service-data-recovery-using-trident-nfs/.
[10] « Episode 99 - Databases as a Service: Containers », podcast Tech ONTAP, 2017, https://soundcloud.com/techontap_podcast/episode-99-databases-as-a-service-containers.

Reportez-vous à la matrice d'interopérabilité (IMT, Interoperability Matrix Tool) sur le site de support NetApp pour vous assurer que les versions de produits et de fonctionnalités mentionnées dans le présent document sont prises en charge par votre environnement. La matrice d'interopérabilité de NetApp (IMT) définit les composants et les versions de produits qu'il est possible d'utiliser pour créer des configurations prises en charge par NetApp. Les résultats dépendent des installations de chaque client et de leur conformité aux spécifications publiées.

WP-7270-071-frFR