WHITEPAPER

Persistenter Storage für Container leicht gemacht

Applikationsentwicklung beschleunigen und DevOps-Effizienz verbessern

NetApp®

Die Herausforderung: DevOps-Reife erzielen

Fortwährende Innovation. Bessere und schnellere Produkteinführungen. Optimierter Betrieb. Ein stetig wachsender Stamm an zufriedenen Kunden.

All dies sind universelle Ziele für viele Unternehmen. Sie sind zugleich Meilensteine auf dem Weg zu besseren Geschäftsergebnissen, mehr Umsatzwachstum und höheren Gewinnen.

Auf der anderen Seite wird das Streben nach Erfolg meist von den Startschwierigkeiten und dem Frust eines Unternehmens im Wandel begleitet. Anwendungsentwickler brennen darauf, Entwicklungsprozesse zu beschleunigen, um die Anforderungen von Kunden zu erfüllen. Sie sind schnell frustriert von Ticketsystemen und langen Wartezeiten auf Storage- und Computing-Ressourcen. Infrastruktur- und Betriebsteams sind hingegen häufig überfordert von der Flut an Tickets und der Notwendigkeit zur Kontrolle der Infrastrukturauslastung.

Aufgrund dieser Mischung aus Zielen und Anfangsschwierigkeiten suchen Unternehmen nach flexiblen und einfachen Methoden, um die Geschwindigkeit und Effizienz bei der Produktentwicklung zu steigern. Auch wenn flexible und einfache Methoden merkliche Vorteile bieten, lassen sich erheblich größere Vorteile erzielen, wenn man die IT-Infrastruktur und -Prozesse so ausrichtet, dass sie die gewünschte Effizienz und Geschwindigkeit unterstützen.

Genau hier setzt DevOps an. Der DevOps-Ansatz zielt auf die schnelle Transformation der Applikationsentwicklung und des IT-Infrastrukturbetriebs.

Unternehmen mit einem hohen Grad an DevOps-Reife zeichnen sich durch sechs wichtige Aspekte aus:

- **Code-, Artefakt- und Binärmanagement.** Repositorys für die Aufbewahrung und das Management von Softwarekomponenten
- **Konfigurationsmanagement.** Konfiguration und Wartung von Infrastruktur- und Softwaresystemen auf herkömmliche Weise
- **Cloud/PaaS.** Nutzung von Public-, Private- und Hybrid-Cloud-Infrastruktur zur Unterstützung der Softwareentwicklung
- **Container.** Schlanke und zugleich hochskalierbare Applikations-Laufzeitumgebungen
- **Analysen.** Automatisiertes Monitoring und Management der Infrastruktur
- **Continuous Integration/Continuous Deployment (CI/CD).** Vollständig automatisierte Prozesse, die es Entwicklern ermöglichen, Code zu schreiben und automatisch zu implementieren

Einer dieser sechs Aspekte – die Container – spielt für Applikationsentwicklungs- und Infrastruktur-Betriebsteams, die nach mehr Geschwindigkeit und Effizienz streben, eine immer wichtigere Rolle. Dieses Whitepaper befasst sich mit dem Schritt zur DevOps-Reife mit Containern und untersucht eine der größten Herausforderungen: das Management von persistentem Storage für zustandsorientierte Applikationen.

Die große Container-Hürde: Management von persistentem Storage für zustandsorientierte Applikationen

DevOps-Teams, die sich mit praxistauglichen Implementierungen in der Produktion von Container-Applikationen befassen, stoßen auf diverse Herausforderungen. Eine der größten ist das Management von persistentem Daten-Storage für Container.

Der kurze Infokasten auf Seite 4 erklärt Konzepte wie Datenpersistenz mit Containern oder den Unterschied zwischen zustandslosen und zustandsorientierten Applikationen.

Schon zu Beginn des Applikationscontainer-Trends war klar, dass das Management des Daten-Storage eine Herausforderung sein würde. Im Rahmen einer Umfrage der Cloud Native Computing Foundation (CNCF) hat fast die Hälfte (42 %) aller Teilnehmer das Storage- und Ressourcenmanagement als Schlüsselherausforderung für den Einsatz von Containern identifiziert (s. Abbildung 1).[1] Viele davon führten anhaltende Problem mit der Storage-Persistenz an. Andere wünschten sich einfacheren Zugriff auf Netzwerk-Storage.

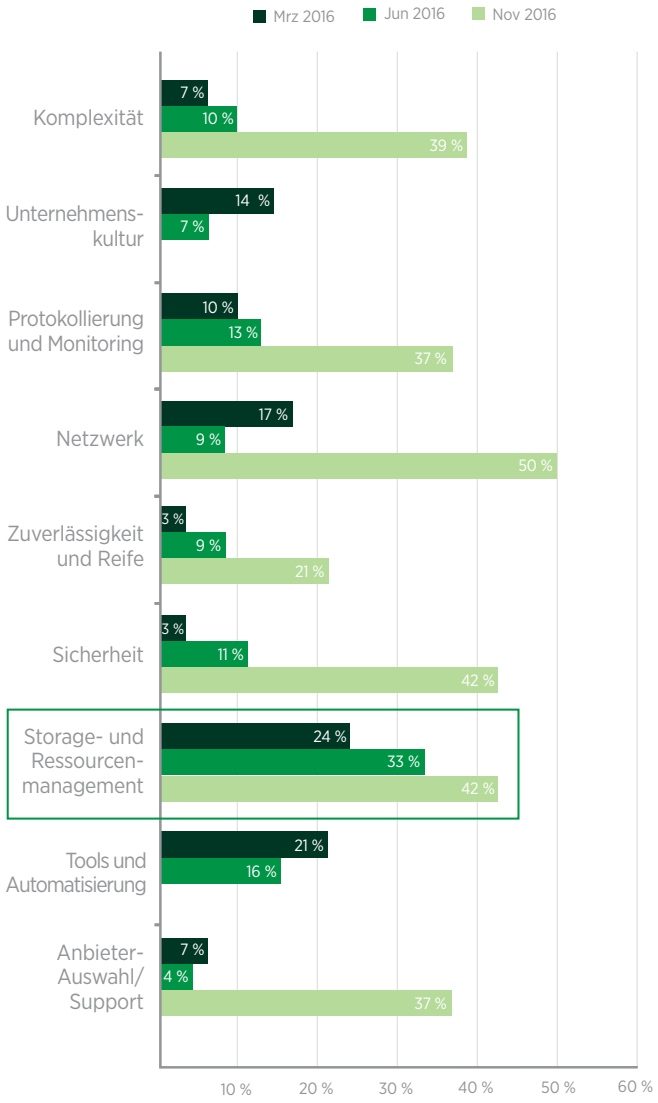

Abbildung 1) Herausforderungen beim Einsatz von Containern (mehrere Antworten zulässig)
Quelle: Cloud Native Computing Foundation

Infolgedessen nahmen Container-Plattformen die Probleme mit persistentem Storage in Angriff. Erste Versuche der Provisionierung von persistentem Storage waren ein guter Anfang, blieben dabei jedoch meist unflexibel und erforderten zu viele manuelle Eingriffe. Zudem schien eine Beschleunigung der DevOps-Pipeline kaum möglich zu sein, schon gar nicht, wenn Applikationen geschrieben, getestet und in Hunderte oder Tausende von Containern implementiert werden mussten.

Hinsichtlich der Skalierbarkeit schien die manuelle Provisionierung von Storage für Tausende von Containern zunehmend untragbar: sie war zu umständlich, zu fehleranfällig und zu wartungsintensiv. Es musste einen besseren Weg geben.

Vor dem Aufkommen von Containern haben viele Enterprise-Applikationen ihren Bedarf an persistentem Daten-Storage mithilfe einer zentralen Anbindung an unternehmensweiten Shared Storage gedeckt. Könnten zustandsorientierte Applikationen in einer Containerwelt nicht ebenso von einem einfachen Zugriff auf vergleichbare Shared-Storage-Funktionen profitieren? Aber die Frage war: Wie leicht ließe sich dies umsetzen?

Ein guter Anfang: Einfache Bereitstellung von persistentem Storage für Container

Containerumgebungen wie Docker und Kubernetes begegneten der Storage-Persistenz-Herausforderung anfänglich mit einem zweckmäßigen halbautomatischen Mechanismus. Mit diesem Mechanismus konnten Anwender einen „Anspruch“ auf ein bestimmtes persistentes Storage-Volume für die Nutzung durch einen oder mehrere Containerprozesse erheben. Hierfür musste ein Storage-Administrator jedoch zuvor persistente Volumes aus den zugrunde liegenden Netzwerk-Storage-Ressourcen erstellen.

Kubernetes: Einfache Provisionierung in Aktion

Kubernetes setzte hierfür auf Code-basierte Mechanismen rund um ein PersistentVolume (PV) und einen dazugehörigen PersistentVolumeClaim (PVC).

Hierbei muss ein Administrator zuerst ein statisches persistentes Volume (z. B. ein Volume mit 8 GB All-Flash-Storage-Kapazität) erstellen. Anschließend können Anwender (oder Entwickler) dieses persistente Volume mit wenigen Codezeilen anfordern oder beanspruchen.

Dieses Anfordern und Binden von PVs an PVCs schien wie geschaffen für Containerumgebungen, da Container-Applikationen auf diese Weise persistenten Netzwerk-Storage über Code nutzen konnten.

Der Ansatz der einfachen Provisionierung war ein guter Start, um die Nutzung von Netzwerk-Storage zu ermöglichen, aber er konnte nicht alle Probleme lösen. Die größte Herausforderung war hierbei die Automatisierung.

Zwischenbilanz: Trotz Fortschritt zu viel Wartezeit

Das Konzept von vorab provisionierten PVs und daran gebundenen PVCs war ein wichtiger erster Schritt, um die Probleme von Containern mit persistentem Storage zu lösen. Die Storage-Provisionierung erforderte aber immer noch zu viele manuelle, statische und wiederkehrende Handgriffe auf Seiten der Entwickler und Betriebsteams. Für Applikationsentwicklungs- und DevOps-Teams, die die Anzahl der Übergaben reduzieren und CI/CD-Prozesse automatisieren wollen, ergaben sich zahlreichen Ineffizienzen:

Ein kurzer Überblick über Container, Datenpersistenz sowie zustandslose und zustandsorientierte Applikationen

Container sind prinzipbedingt zustandslos. Ihre Inhalte kurzlebig. Dadurch können alle Applikationen und Prozesse eines Containers während einer Session schnell gestartet, angehalten und neugestartet werden. Alle Daten, die erzeugt werden, während ein Container aktiv ist, gehen verloren, sobald er beendet oder zerstört wird.

DevOps-Teams wollten immer mehr Applikationen auf Container-Plattformen entwickeln und produktiv nutzen. Hierfür war es jedoch häufig erforderlich, Daten auch nach dem Lebensende eines Containers aufbewahren und persistent machen zu können.

Schnell war klar, dass so gut wie jede Applikation über mindestens einen Prozess oder Microservice verfügte, der ein persistentes Volume (PV) für Statusdaten erforderte, die über das Leben des jeweiligen Containers hinaus gespeichert werden mussten.

Beispiele für zustandsorientierte Applikationen, die Datenpersistenz erfordern:

- **Datenbankumgebungen.** Datenbank-Container erfordern persistenten Storage für ihre Datastores. Aufgrund ihres flüchtigen Designs ist dies mit Containern nicht ohne weiteres möglich. Lokaler Storage war hierfür keine geeignete Lösung. Wird der Container verschoben oder zerstört, verliert er den Zugriff auf die Daten.
- **Umgebungs- oder Session-Daten.** Zustandsorientierte Applikationen sammeln und speichern oft Umgebungsattribute oder Anwender-Session-Daten (Status). Sie dienen als Archivdaten, um das Anwendungserlebnis zu verbessern. So kann die Applikation, wenn der Anwender das nächste Mal auf sie zugreift, relevante Daten anzeigen oder Daten aus einer vorigen Session besser ändern.

Daten mussten jedoch nicht nur gespeichert, sondern auch geteilt werden können. Unternehmen, die Container-Applikationen entwickelten oder einführten, merkten schnell, dass alle – von der Entwicklung über den Test bis zum Betrieb – auf dieselben Datensätze auf denselben zentralen Netzwerk-Storage-Ressourcen zugreifen mussten. Auf Netzwerk-Storage gespeicherte persistente Volumes sorgten zudem dafür, dass Container-Applikationen durch Enterprise-Storage-Funktionen besser geschützt waren. Die Applikationen profitierten von besserer Verfügbarkeit, Zuverlässigkeit, Sicherheit und Datensicherung.

- **Manuelle Storage-Anforderungen.** Applikationsentwickler und QA-Mitarbeiter, die am Code oder Testläufen arbeiten, müssen ihre Arbeit womöglich pausieren, um einen Storage-Administrator um die Bereitstellung eines neuen PV zu bitten. Bei der Skalierung großer Container-Applikationen kann dies zu Hunderten oder Tausenden von ständigen Storage-Anforderungen führen.
- **Frustrierende Wartezeiten auf Storage.** Entwickler- oder QA-Teams müssen unter Umständen auf die Bestätigung ihrer Anfrage warten oder darauf, dass ein Administrator einen Zeitpunkt für die manuelle Erstellung eines oder mehrerer PVs festlegt.
- **Ineffiziente Storage-Auslastung.** Storage-Administratoren müssen den Storage der PVs für die Entwicklung und QA im Voraus provisionieren. Hierbei kann leicht zu wenig (Unterprovisionierung) oder zu viel (Überprovisionierung) Storage-Kapazität bereitgestellt werden (s. Abbildung 2). Es können auch zu wenige oder viele IOPS pro Volume reserviert werden:
  - **Risiken der Unterprovisionierung:** Wird im Vorfeld zu wenig Storage für PVs provisioniert, kann die zu einem Engpass für Entwickler werden, die mehr Kapazität benötigen als unmittelbar zur Verfügung steht.
  - **Risiken der Überprovisionierung:** Überprovisionierte Storage-Ressourcen, die durch ein oder mehrere PVs gesperrt sind, bleiben später möglicherweise ungenutzt, wenn sie an einen PVC mit deutlich geringeren Anforderungen an Performance und Kapazität gebunden werden. (Vergleichbar mit den überprovisionierten und zu wenig ausgelasteten Storage-Silos der Vergangenheit)

- **Ineffektive Nutzung von Administrator-Ressourcen.** Storage- und Cluster-Administratoren können für viele Entwicklungsprozesse zum neuen Engpass in einer oder mehreren DevOps-Pipelines werden. Wenn kurzfristig mehrere neue Volume-Anforderungen gestellt werden, muss der Administrator möglicherweise immer noch viele manuelle Schritte durchführen, um die einzelnen statischen Volumes zu erstellen und einzurichten. Dieses Vorgehen war auch nicht sonderlich hilfreich für Administratoren mit der Aufgabe, die Infrastruktur stärker zu automatisieren und ihre Gesamtauslastung zu überwachen.

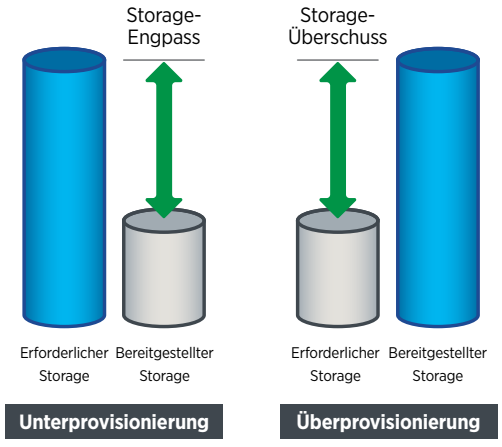

Abbildung 2) Risiken der Unterprovisionierung oder Überprovisionierung

Dies erinnert nicht nur auf den ersten Blick an altherge-brachte Storage-Provisionierung. Zum Abschluss seines Vortrags zum Thema DevOps Through Desired State beim Tech Field Day 2017 bedauerte Andrew Sullivan, Technical Marketing Engineer bei NetApp, die DevOps-Teams, die sich von dieser veralteten Form der Provisionierung befreien wollen. **„Storage sollte keine Ressource sein, bei der man damit leben muss, das Storage-Team ständig um mehr Kapazität anzubetteln“**, sagte er. **„Storage sollte im Jahr 2017 nicht auf dieselbe Weise provisioniert und verbraucht werden wie 1989.“** [2]

NetApp – und weitere Unterstützer von Containerumgebun-gen – erkannten, dass es für DevOps-Teams eine bessere Möglichkeit geben musste, um Storage nach Bedarf zu verbrauchen und dynamisch zum gewünschten Zeitpunkt am gewünschten Ort zu provisionieren.

Storage-Klassen, Storage-Pools und Trident: Persistenter Storage auf Abruf

Andrew Sullivan hat im letzten Teil seines Vortrags darauf hingewiesen, dass es einen besseren Weg zur Provisionierung von Storage-Ressourcen geben muss. Einen Weg, der die ursprüngliche Vision von Containern wiederspiegelt – Funk-tionen, die nahezu ohne manuelle Eingriffe je nach Bedarf dynamisch erzeugt und implementiert werden.

Für Kubernetes bedeutete dies die Einführung der dynamischen Storage-Provisionierung mithilfe eines neuen Konzepts, der StorageClass. Es bedeutete auch hochgradige Automatisierung im Bereich der Storage-Provisionierung, wie bei Trident für Kubernetes:

- **Persistenter Storage nach Bedarf.** Trident ist ein Open-Source-Projekt von NetApp für dynamische Stora-ge-Provisionierung. Es stellt Container-Applikationen auf Anforderung in Echtzeit persistente Volumes auf NetApp Storage bereit – ohne Wartezeit.
- **Leistungsstarke Storage-Funktionen.** Containerum-gebungen mit Trident unterstützen die leistungsstarken Storage-Funktionen der zugrunde liegenden NetApp Datenmanagementplattformen (wie NetApp HCI, ONTAP, NetApp SolidFire Element OS, SANtricity, Cloud Volumes).
- **Nicht nur für Kubernetes.** Trident für Docker und OpenShift bieten vergleichbare Funktionen für persistente Volumes.

Kann ein Storage-Klassen-Katalog so einfach sein?

Mit Trident können Entwickler in Kubernetes-Umgebungen persistente Volumes dynamisch provisionieren, indem sie eine Storage-Klasse aus einem virtuellen Pool von zugrunde liegendem Storage anfordern.

Über Storage-Klassen werden persistente Volumes automa-tisch provisioniert, wenn sie per Code angefordert werden. Anwender erstellen einfach einen Persistent Volume Claim und geben die gewünschte Trident Storage-Klasse wie Gold, Silver oder Bronze mit an.

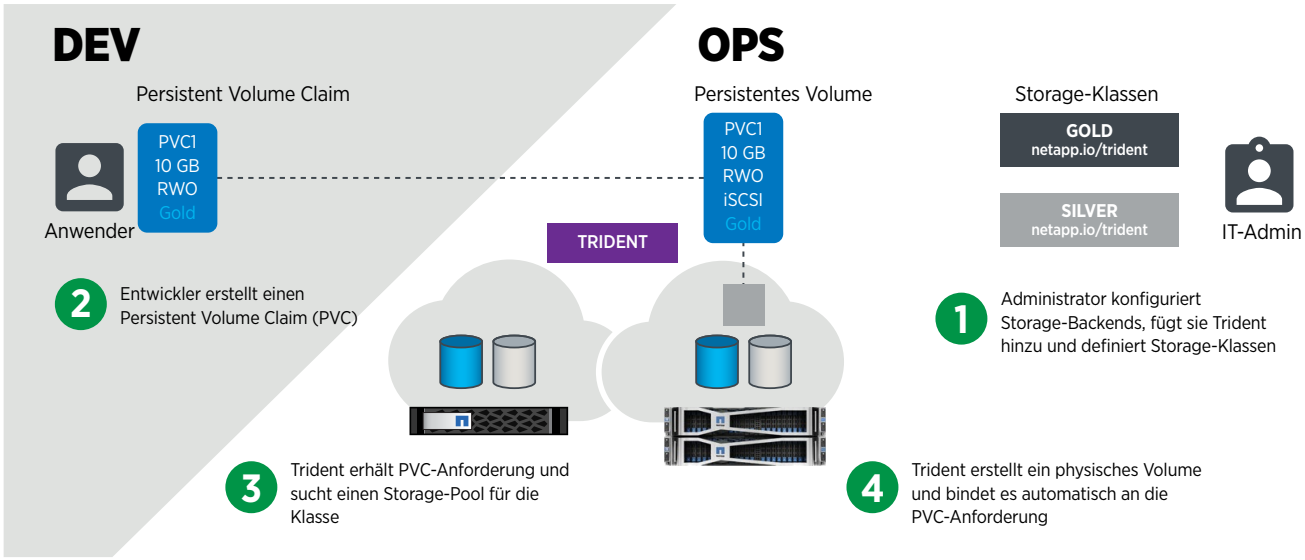

Abbildung 3) Dynamische Provisionierung persistenter Volumes mit Trident

Hinweis: Die Konfiguration der zugrunde liegenden Storage-Klassen-Attribute und -Namenskonvention ist eine Back-End-Funktion, die in der Regel von einem Administrator übernommen wird, wenn er den ersten Katalog persistenter Storage-Klassen definiert. Die Storage-Klassen eines Unternehmens könnten zum Beispiel Dev, Staging und Produktion heißen und die eines anderen Schnell und Langsam. Weitere Informationen zu dieser Art von Konfigurationsfunktionen finden sich in der Trident-Dokumentation.[3]

Im zugrunde liegenden NetApp Storage-Pool wird dann ein persistentes Volume der angeforderten Storage-Klasse erstellt und an den Persistent Volume Clain (PVC) gebunden. Die Anwender müssen sich nicht mit dem zugrunde liegenden Storage auseinandersetzen. Trident kümmert sich um diese Details.

Trident in Aktion sehen

Der Nutzen von Trident erschließt sich am besten, wenn Sie die dynamische Provisionierung mit eigenen Augen sehen. Sehen Sie sich hierfür eine der folgenden Online-Demos an:

- **Kurze Demo** (3:20 Minuten): Using Trident for Dynamic Storage Provisioning with OpenShift [4]
- **Längere Demo** (23:53 Minuten): Managing Persistent Data in Kubernetes [5]

Welche Vorteile bietet diese Form der Provisionierung unterschiedlichen Mitarbeitern von IT- und DevOps-Teams?

- **Für Entwicklungs- und QA-Teams:** Kein Warten mehr auf Service-Tickets oder die Bestätigung von Storage-Anforderungen. Keine Übergaben. Storage mit zugesicherten SLAs kann jederzeit genau dort verbraucht werden, wo er benötigt wird. Die Provisionierung erfolgt dynamisch über vertraute Programmierschnittstellen. Entwickler erhalten mehr Freiheiten durch ein dynamisches und flexibles automatisches Provisionierungssystem, das dennoch der Kontrolle des Betriebsteams unterliegt.
- **Für IT- und Storage-Administratoren:** Kein Stress mehr durch eilige Storage-Provisionierungs-Anfragen oder endlose Service-Tickets. Weniger Storage-Administration. Stärker automatisierte Infrastrukturskalierung. Besser planbare Kontrolle über den Storage-Verbrauch und einfacheres Ressourcen-Monitoring.
- **Für IT-Führungskräfte:** Schnellere Produktbereitstellung, bessere Prozesse und signifikante Ressourceneinsparungen.

Die Zukunft von DevOps

Der Full-IT-Service-Provider DARZ bietet DevOps-Flexibilität mit Docker & Container-as-a-Service. Dieses Angebot basiert auf NetApp All-Flash-Storage und Trident für Docker. Kunden können Applikationscontainer schnell hoch- oder herunterfahren – auch ohne ein ausgewachsenes Betriebssystem. So können die Computing-Anforderungen um bis zu 75 % reduziert werden.

Mit einer schlanken, flexiblen Containerumgebung können Kunden Testzyklen verkürzen, die Entwicklung beschleunigen und neue Produkte schneller einsetzen. Trident erleichtert das Management persistenter Daten in einer Docker-Umgebung und vereinfacht die Storage-Interaktion mit einer Reihe von Docker-Volume-Befehlen.[6]

Moderne Arbeitsweise

Tools wie Trident bieten DevOps-Teams nicht nur die Möglichkeit, persistente Volumes dynamisch zu provisionieren, sondern noch viel mehr.

Zum Beispiel Code-basierten On-Demand-Zugriff auf NetApp Storage-Effizienz-Funktionen wie Snapshot Kopien und Klonen. Für Entwicklungs- und QA-Teams können diese Funktionen entscheidend sein, um in kürzerer Zeit und mit möglichst wenigen Ressourcen mehr zu erreichen.

Snapshot Kopien und Klonen über NetApp und Trident ermöglichen:

- **Schnelles Erstellen von Echtzeitklonen vollständiger Produktionsdatensätze.** Hierdurch können in Sekundenschnelle mit wenigen Codezeilen neue Entwicklungs- und Testumgebungen dynamisch erzeugt werden (s. Abbildung 4).
- **Schnelle und einfache Daten-Recovery.** Mit Snapshot Kopien können Entwickler schnell vorherige Versionen von Datensätzen wiederherstellen. Dies ist praktisch, um Code zu testen – Entwickler können schnell iterieren, ohne Testdatensätze jedes Mal neu erzeugen zu müssen.
- **Storage-Kapazitätseinsparungen für Snapshot Kopien oder Klone.** Unternehmen, die zu Entwicklungs- und Testzwecken mehrere Klone ihrer Produktionsdaten anlegen, sparen hierbei in der Regel zwischen 40 % bis 90 %[7] Storage-Kapazität ein.

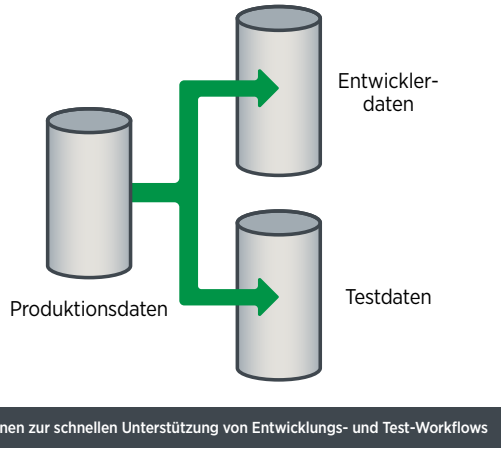

Abbildung 4) Die NetApp Klontechnologie beschleunigt DevOps-Workflows

Hier finden Sie weitere Informationen, wie die NetApp Klontechnologie und Snapshot mit Trident funktionieren:

- Self-service volume cloning with Kubernetes[8]
- Snapshot copies and self-service volume recovery with Trident[9]
- Dieser Tech ONTAP Podcast erläutert, wie der NetApp Oracle-Experte Jeff Steiner eine große Oracle-Datenbank mit Docker und Trident in nur 22 Sekunden geklont hat[10].

Gut für Sie, gut für Ihr Unternehmen

NetApp hat reichlich Erfahrung darin, Probleme gemäß den Anforderungen von Entwicklern und Technikern zu lösen. Als Unternehmen haben wir schon früh erkannt, dass Storage-Infrastruktur die wichtige Arbeit von Entwicklern und Technikern unterstützt, sodass auch die weiter gefassten Unternehmensziele schneller erreicht werden. NetApp arbeitet konsequent an besseren Wegen für das Datenmanagement und die Storage-Nutzung in Unternehmen. Offene Ecosysteme wie Container bieten uns neue Möglichkeiten für weitere Innovationen. Die Community ist der Antrieb für diese Innovation. NetApp ist Teil der Community und stärkt aktiv ihre Innovationskraft. Hierfür arbeitet NetApp an vielen Möglichkeiten, die es der Community ermöglichen, Storage einfacher zu nutzen, ganz gleich, wo und wann sie ihn benötigt. Wir freuen uns, dass NetApp heute die nahtlose Storage-Nutzung über eine Vielzahl offener Ecosysteme unterstützt. Wir bieten eines der branchenweit umfangreichsten Angebote an APIs und Integrationen für Umgebungen wie Docker, Kubernetes, OpenShift, OpenStack, Ansible, Chef, Puppet und viele mehr und entwickeln es immer weiter.

Trident ist nur ein Beispiel unserer Bemühungen. Nutzen Sie Trident für die Containerumgebungen Ihres Unternehmens. Wir freuen uns darauf zu hören, welche unglaublichen Effizienzen und Einsparungen Sie damit für Ihre DevOps-Pipeline erzielen können.

Weitere Informationen

Wir empfehlen einen Blick auf die folgenden Quellen, um mehr über Trident und anderen NetApp DevOps-Integrationen zu erfahren:

Infos zu NetApp und Trident

NetApp Lösungen für Container	
Funktionsumfang von Trident: Introducing Trident	
Introduction to Kubernetes persistent storage with Trident	
Cloning: Introducing Volume Cloning with Trident to Kubernetes	
Trident-Dokumentation	
Trident GitHub-Download	

Infos zu NetApp mit DevOps

NetApp Lösungen für DevOps	NetApp
thePub (netapp.io)	thePub netapp.io
NetApp Slack-Channel (netapp.io/slack)	
@NetAppPub	

Fußnoten

[1] „Meeting Challenges in Using and Deploying Containers“ von Sarah Conway, 27. April 2017, Cloud Native Computing Foundation, https://www.cncf.io/blog/2017/04/27/meeting-challenges-using-deploying-containers/. Reproduktion mit zusätzlichem roten Kreis unter Creative Commons CC-BY-Lizenz 4.0.
[2] „DevOps Through Desired State“, präsentiert von Andrew Sullivan, NetApp, Tag 14, 11. Mai 2017, Tech Field Day, https://www.youtube.com/watch?v=btLZl7M6gnY&list=PLinuRwpnsHacYmunO7zyES6SyrsrFfu5O&index=4.
[3] Die neueste Trident-Dokumentation finden Sie unter https://netapp-trident.readthedocs.io/.
[4] „Using Trident for Dynamic Storage Provisioning with OpenShift“-Online-Demo, 3:20 Min., von The Pub @ NetApp, 17. Feb. 2017, https://www.youtube.com/watch?v=97VZWYssL2E.
[5] „Managing Persistent Data in Kubernetes“-Online-Demo, 23:53 Min., von 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“, Kundenreferenz, NetApp, 2017, https://www.netapp.com/us/media/cs-darz-devops.pdf.
[7] „How NetApp IT Shortened Development Cycles Using FlexClone“, NetApp Community-Blog-Beitrag von Gopal Parthasarathy, 8. Okt. 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!“ von Garrett Mueller, NetApp, 14. Dez. 2017, https://netapp.io/2017/12/14/trident-18-01-beta-1-introducing-volume-cloning-kubernetes/. (s. auch: „Trident 18.01 is Here“ von Andrew Sullivan, NetApp, 25. Januar 2018, https://netapp.io/2018/01/25/trident-18-01/).
[9] „Self-Service Data Recovery using Trident and NFS“ von Andrew Sullivan, 3. April 2018, NetApp,https://netapp.io/2018/04/03/self-service-data-recovery-using-trident-nfs/.
[10] „Episode 99 - Databases as a Service: Containers“ Tech ONTAP Podcast, 2017, https://soundcloud.com/techontap_podcast/episode-99-databases-as-a-service-containers.

Überprüfen Sie mithilfe des Interoperability Matrix-Tools (IMT) auf der NetApp Support-Website, ob die in diesem Dokument angegebenen Produktversionen und Funktionen in Ihrer IT-Umgebung unterstützt werden. NetApp IMT definiert die Produktkomponenten und -versionen, die für von NetApp unterstützte Konfigurationen verwendet werden können. Die jeweiligen Ergebnisse sind von der kundenspezifischen Installation bzw. den technischen Daten abhängig.

WP-7270-071-deDE