Adam Trujillo
(Adam.Trujillo@parasoft.com)
ist seit über vier Jahren als Technischer Autor für Parasoft tätig, wo er regelmäßig über Themen wie Sevice-Virtualisierung, Testautomatisierung, DevOPs und funktionelles Testen schreibt.

Deploy and Destroy – Die Bereitstellung von vollständigen Testumgebungen in weniger als 10 Minuten

Testumgebungen auf Basis von Service-Virtualisierung und Containern ermöglichen es, jederzeit testen und einfacher Testdaten erheben zu können. Ferner lassen sich Testsysteme und -services zwischen Abteilungen friktionsfrei teilen. Auf diese Weise können Testabläufe und Testumgebungen individuell, unabhängig, flexibel und vor allem problemangemessen definiert und dennoch effizient geteilt werden. Mithilfe von Service-Virtualisierung, Containern, Microservices und Docker können Sie sich selbst jede erdenkliche Testumgebung erschaffen – zu jeder Zeit und vollständig. In diesem Artikel erfahren Sie: Wie Sie die vollständige Kontrolle über das Verhalten, die Daten und die Leistung all der abhängigen Systeme erreichen, die anderen Gruppen oder Organisationen „gehören". Wie Sie Ihre Testumgebung eigenständig entwickeln und Ihren Testprinzipien treu bleiben können. Wie Sie jedes Mal mithilfe einer automatisierten „Deploy and Destroy"-Strategie eine neue Testumgebung verwenden können, ohne den Aufwand „aufräumen" zu müssen. Wie Sie Entwickler durch eine vereinfachte gemeinsame Nutzung der Testumgebung von Ihrer Methodik überzeugen können.

Heutzutage stehen Entwickler- und Testteams unter enormem Druck, immer mehr innovative Software abzuliefern und das in immer kürzerer Zeit. Da die meisten Organisationen auf Software als primäre Schnittstelle zwischen sich und den Kunden angewiesen sind, können keine Kompromisse in der Qualität zugunsten der Durchlaufzeit eingegangen werden.

Die Frage ist, wie Entwickler und Tester in die Lage versetzt werden, Qualität bei zunehmender Geschwindigkeit abliefern zu können. Leider gibt es auf diese Frage keine einfache Antwort, allerdings kann ungehinderter Zugang zu realistischen Testumgebungen als eine der wesentlichen Voraussetzungen angesehen werden (wie z. B. die Anwendung unter Test und ihre abhängigen Komponenten). Die Folgen von Änderungen im Code lassen sich sonst nicht akkurat validieren und die Nutzererfahrung könnte möglicherweise erheblich beeinträchtigt werden.

Der Zugang zu vollständigen Testumgebungen verhilft nicht nur zu einer schnelleren Durchlaufgeschwindigkeit, er versetzt Entwickler und Tester auch in die Lage, das Risiko eines Freistellungskandidaten während des CI/CD-Prozesses zu erkennen und Kandidaten mit einem hohen Risiko bereits früh in der Lieferkette zu identifizieren.

Leider ist der Zugang zu vollständigen Testumgebungen in den heutigen komplexen Systemen eher die Ausnahme als die Regel. Eine der Möglichkeiten, um eine 100 % realistische Testumgebung zu schaffen, ist das Testen in der Produktionsumgebung, was jedoch aus einer Reihe von Gründen in der Regel nur ungern gesehen wird.

Während eines Webinars, das Parasoft vor einigen Monaten abhielt, gaben nur 12 % von 316 Teilnehmern auf Anfrage an, Zugang zu der Produktionsumgebung zu Testzwecken zu haben. Auch wenn es sich hierbei um keine wissenschaftlich durchgeführte Statistik handelt, kann man dennoch davon ausgehen, dass diese Umfrage im Wesentlichen die Realität widerspiegelt.

Obwohl es für Testabteilungen früher typisch war, eine komplette physische Testumgebung einzurichten, verbietet sich die-

se Möglichkeit heutzutage aus Kostengründen und aufgrund der Komplexität moderner Anwendungen.

Dies belegt auch die folgende Statistik:

- Die Durchschnittszeit zur Bereitstellung einer Testumgebung beträgt 18 Tage [vr1].
- Die Zeit, die Testumgebung zu konfigurieren, beträgt weitere 12 bis 14 Tage [vr1].
- Die Durchschnittskosten für ein Pre-Production Lab betragen ca. 12 Millionen Dollar [sky].
- 78 % aller Teams sahen sich mit Einschränkungen durch dritte Parteien, zeitlichen Begrenzungen oder Gebühren bei dem Versuch konfrontiert, die für die Tests notwendigen abhängigen Systeme zu testen [vr2].
- Teams benötigen im Schnitt Zugang zu 52 abhängigen Anwendungen oder Systemen, um vollständige Tests durchführen zu können, die meisten haben lediglich Zugang zu 23 oder weniger.

In Anbetracht dieser Einschränkungen ist es nicht verwunderlich, dass die große Mehrheit der Entwicklerteams und Qualitätsprüfer als Folge des limitierten Zugangs zu Testumgebungen Verzögerungen erleben.

Allerdings sind die Technologien heute so weit fortgeschritten, dass es möglich ist, realistische, simulierte Testumgebungen bei Bedarf zur Verfügung zu stellen. Aufgrund der Verfügbarkeit von Technologien wie z. B. Cloud (für flexible Skalierbarkeit), Container (für schnelles Ausbringen) und Service-/Anwendungs-Virtualisierung (für das Simulieren und den Zugriff auf abhängige Systeme) besteht eigentlich kein Grund mehr, warum eine realistische Testumgebung nicht verfügbar sein sollte.

An diesem Punkt kann eine „Deploy and Destroy"-Strategie wichtig sein.

Was versteht man unter „Deploy and Destroy"?

Im Wesentlichen handelt es sich hierbei um die Fähigkeit, eine vollständige Testumgebung in weniger als 10 Minuten bereitzustellen (*to deploy*). Die meisten abhängigen Komponenten (wie z. B. APIs, Services von Dritten, Datenbanken, Anwendungen oder andere Endpunkte), die zu einem bestimmten Zeitpunkt verfügbar sein müssen, werden in einem zentralen Repository zusammengefasst und automatisch bereitgestellt.

Empfehlenswert und bewährt ist dabei eine Kombination von realen und simulierten Komponenten, die mithilfe von Service-Virtualisierung bereitgestellt wird. Eine „Deploy and Destroy"-Umgebung ist typischerweise leicht und Cloud-kompatibel und daher einfach und nach Bedarf skalierbar. Darüber hinaus kann sie auch leicht wieder entsorgt werden. Sie kann direkt von einem Template generiert, benutzt und anschließend wieder zerstört werden (*to destroy*). Es benötigt keinen Zeitaufwand, die Testumgebung und Testdaten wieder in ihren ursprünglichen Zustand zu versetzen. Die gleiche Umgebung kann unmittelbar bei Bedarf wieder erzeugt werden, wenn z. B. Fehler reproduziert oder verifiziert werden müssen.

Die Wertschöpfung von „Deploy and Destroy"

Deploy and Destroy bietet vielerlei Vorteile für Entwickler- und Testteams.

Frühes und kontinuierliches Testen

Der unmittelbare Zugriff und die simultane Verfügbarkeit von realistischen und flexiblen Testumgebungen versetzt Teams in die Lage, eine ganze Anzahl von End-to End-Tests auszuführen. Sobald eine User Story vollständig ist, kann mit dem Testen begonnen werden und eine Vielzeit von User Stories kann gleichzeitig während des Continuous-Testing-Prozesses validiert werden, jede in der spezifischen Testumgebung, die für die jeweiligen Tests benötigt wird.

Mehr Flexibilität

Im Vergleich zu einer realen, physischen Testumgebung erlaubt die Simulation, Testumgebungen flexibler bereitzustellen. In realen Testumgebungen ist es in der Regel schwierig, das Verhalten oder die Leistung von abhängigen Anwendungen zu beeinflussen, beispielsweise von Mainframes oder ERPs. Es erfordert üblicherweise eine zeitaufwendige Konfiguration der Hardware, die Verteilung von Speicherkapazitäten, das Einrichten von VMs, usw., um nur einige Beispiele zu nennen.

Durch den Einsatz von Simulationstechnologien wie z. B. Service-Virtualisierung kann dieser Aufwand eingespart werden. Mit Leichtigkeit lassen sich Bedingungen herstellen, die in einer Produktions- oder physischen Testumgebung unmöglich zu konfigurieren wären. Der Testhorizont lässt sich dadurch erheblich erweitern und ermöglicht auch extreme Fälle zu testen, wie z. B. Concurrency, Ausfälle, Resiliency, Failover oder Load Balancing-Probleme.

Realistischer als die tatsächliche Testumgebung

In vielen Fällen ist eine Deploy and Destroy-Umgebung effektiver als reale, physisch eingerichtete Umgebungen, in denen man in der Regel mit Hardwarerestriktionen zu tun hat, zumal die Leistung dieser Testumgebungen beschränkt ist. Service-Virtualisierung gibt Testern größere Kontrolle darüber, wie die abhängigen Systeme und Anwendungen reagieren, es lassen sich Rückmeldezeiten wie in einer Produktionsumgebung erzielen und dadurch kann die Wiedergabetreue der Testumgebungen erhöht werden.

Mithilfe von Service-Virtualisierung können Fehler und Ausfälle in realen Testumgebungen kompensiert werden, als Beispiel sei der Fall genannt, in dem abhängige Systeme gerade dann nicht verfügbar sind, wenn Tests ausgeführt werden müssen. Eine Deploy and Destroy-Umgebung kompensiert die Nichtverfügbarkeit, in dem sie den Verkehr auf simulierte Endpunkte umleitet und dadurch die Durchführung oder Fortführung der Tests ermöglicht.

Datensicherheit und Privatsphäre

Durch die simulierte Umgebung wird Zugriff auf Datensätze gewährt, die bereits gesäubert bzw. anonymisiert sind, daher spielen Sicherheitsbedenken keine Rolle. Selbst wenn sich jemand unbefugt Zugang verschafft oder ein Asset angreift, gibt es keine Grund zur Besorgnis, da es nicht real ist und unmittelbar nach dem Test wieder zerstört wird.

Konsistenz und Genauigkeit

Deploy and Destroy stellt sicher, dass alle Beteiligten übereinstimmend ihre Aktivitäten durch Zugriff auf die neuesten Testobjekte/Daten und virtuellen Artefakte koordinieren können. Jedes Mal, wenn eine neue Testumgebung bereitgestellt wird, wird sie über den Zugriff auf den Masterdatensatz, die virtuellen Assets und Testobjekte in dem Repository von Grund auf neu kreiert. Dadurch lässt sich verhindern, dass Tests in „verunreinigten" Umgebungen ausgeführt werden, veraltete Daten oder nicht autorisierte

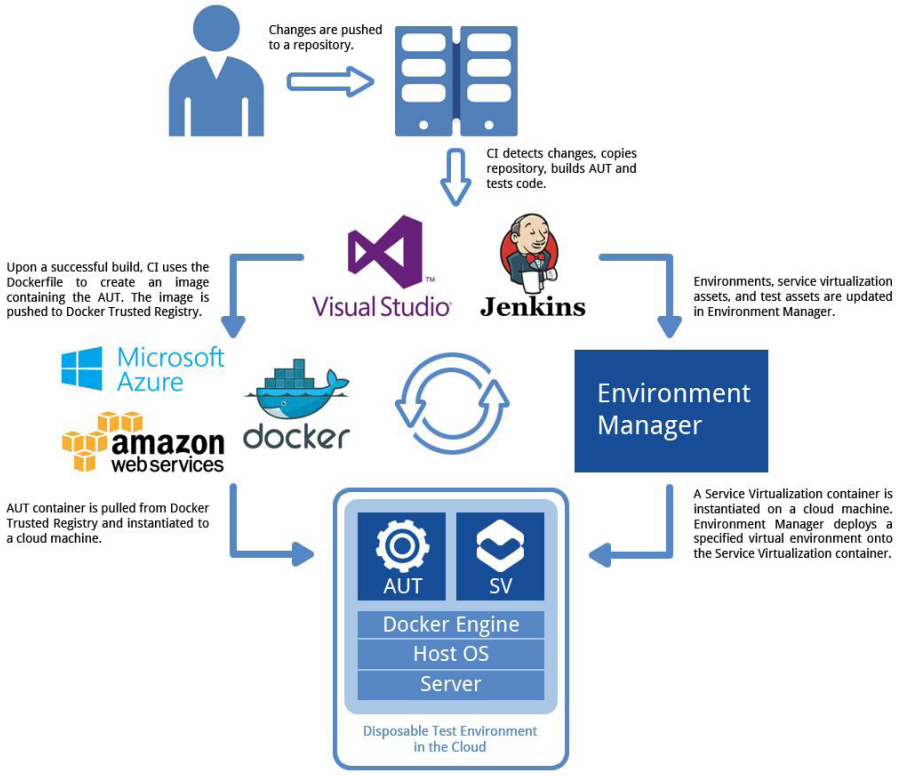

Abb. 1: Beispiel einer Deploy and Destroy-Testumgebung

Versionen von virtuellen Assets benutzt werden.

Ein technischer Blick auf Deploy and Destroy

Der erste Schritt einer Deploy and Destroy-Umgebung ist die Schaffung einer Bibliothek von Service-Virtualisierungs Assets. Üblicherweise wird dies durch die Aufnahme des Zusammenspiels von Anwendungen erstellt, kann aber auch durch andere Methoden erzeugt werden, wie z. B. durch die Nutzung von Swagger (eine komplette Frameworkimplementierung für Webservices und APIs) oder anderer Definitionen, wie z. B. Traffic-Dateien, usw.

Falls gewünscht, können diese Assets durch weitere Daten, Leistungsprofile oder Response-Logik erweitert und verbessert werden.

Sobald ein Fundament von virtuellen Assets erstellt wurde, können diese in einer Umgebung zusammengefasst werden. Ausgehend von einem Kernsystemplan definiert eine Umgebung, welche Zustände die Abhängigkeiten der Anwendung unter Test einnehmen dürfen. *Beispiel*: eine Anwendung oder ein Service Dritter kann durch zehn verschiedene virtuelle Versionen dargestellt werden, jede mit jeweils verschiedenen Leistungs- oder Datenprofilen, zusätzlich zur realen Version.

Beispiel: eine virtuelle API, die Netzwerküberlastung simuliert + Fehlerrückmeldung + reale Datenbank + virtuelle Datenbank, die positive Rückmeldung gibt.

Um den Bereitstellungsprozess zu verfeinern und zu beschleunigen, sollten die bereitgestellten Umgebungen mit einer Containertechnologie wie. z. B. Docker verknüpft werden, was Testumgebungen in Sekundenschnelle zur Verfügung stellt.

Die Abbildung stellt eine der Möglichkeiten dar, wie Container genutzt werden können, um eine Deploy and Destroy-Strategie umzusetzen. Das 15-minütige Video [you] veranschaulicht, wie eine „Deploy and Destroy Strategy“ Schritt für Schritt umgesetzt werden kann.

Fazit

Zusammenfassend lässt sich sagen, dass die Implementierung einer Deploy and Destroy-Strategie schnelle und messbare geschäftsrelevante Auswirkungen auf Entwicklungs- und Testorganisationen hat. Kosten und Wartezeiten in Verbindung mit dem Zugriff auf abhängige Systeme oder Datenbanken können reduziert bzw. eliminiert werden.

Das Risiko für Release-Kandidaten wird minimiert, agile und iterative Entwicklungsmethoden werden effizient unterstützt und Entwicklungs- bzw. Testzyklen erheblich verkürzt. Zusätzlich kann die Effizienz von Testprozessen durch die Nutzung von Open Source Frameworks wie Swagger zum Testen von APIs oder Docker, welches die Bereitstellung von Anwendungen innerhalb von Software-Containern automatisiert, gesteigert werden.

In einer Zeit, die von Entwicklern verlangt, Anwendungen immer schneller und fehlerfrei abzuliefern, wird Service-Virtualisierung eine zunehmend wichtige Komponente, um Engpässe während der Testphase zu verringern, was wiederum „Time to Market“ beschleunigt und erhebliche Wettbewerbsvorteile nach sich zieht. Deploy and Destroy kann somit für Unternehmen und Anwendungsanbieter eine Strategie sein, sich von Mitbewerbern zu unterscheiden und dazu beitragen, den eigenen finanziellen Erfolg abzusichern. ■

Referenzen

[vr1] voke Research, Market Snapshot Report: Virtual and Cloud-Based Labs
https://www.skytap.com/research-center/white-papers-analysis/voke-report-cloud-based-labs/
[sky] Skytap, Transform Your SDLC with Parallel Development and Testing in Cloud Environment
https://www.skytap.com/research-center/white-papers-analysis/transform-your-sdlc/
[vr2]voke Research, Market Snapshot Report: Service Virtualization
http://www.vokeinc.com/market-snapshot-service-virtualization.html
[you] https://youtu.be/xe9sMxR2Dxk