Hochschule Wismar
University of Applied Sciences
Technology, Business and Design
Fakultät für Ingenieurwissenschaften

Master-Thesis

Identifizierung und Sicherung von Artefakten in
Infrastructure-as-a-Service Plattformen

Eingereicht am: 26. März 2024

von: Simon Heinrich Bauer

Betreuer: Prof. Dr. Antje Raab-Düsterhöft
Zweitbetreuer: Prof. Dr. Nils Gruschka

Aufgabenstellung

Im Rahmen der Masterthesis ‘Identifizierung und Sicherung von Artefakten in Infrastructure-as-a-Service Plattformen‘ soll ein allgemeiner Leitfaden zur forensischen Untersuchung von IaaS-Plattformen erstellt werden. Der Leitfaden soll die Identifizierung und Sicherung von Artefakten abdecken. Einleitend soll allgemein IaaS-Plattformen und deren Herausforderungen in der IT-Forensik erläutert werden. Weiterführend soll ein Überblick über den aktuellen Stand der Forschung gegeben werden.

Es sind mehrere Ansätze zu vergleichen. Anhand der gewonnenen Erkenntnisse soll ein allgemeiner Leitfaden zur Identifizierung und Sicherung von Artefakten erstellt werden.

Weiterhin soll am Beispiel von mehreren IaaS-Plattformen der Leitfaden evaluiert werden. Dazu sollen zwei Szenarien entwickelt und bei unterschiedlichen IaaS-Plattformen durchgespielt werden.

Kurzfassung

Cloud Computing ist heutzutage fast überall vertreten. Diese Technologie bringt neben Chancen auch Risiken mit sich. Anbieter und Kunden können Opfer von Cybercrime werden. Weiterhin kann Cloud Computing auch als Werkzeug für illegale Aktivitäten genutzt werden. Mithilfe des vorgestellten, allgemeinen Leitfadens können Forensiker:innen die Identifizierung und Sicherung von Artefakte aus IaaS Plattformen realisieren. Der Leitfaden teilt sich in Client- und Cloud-Forensik auf. Die erste Phase deckt Client-Artefakte wie Zugangstoken und Protokolle ab. Die Cloud-Forensik Phase besteht aus Orientierung, Erfassung, Sicherung und Extraktion. Die Evaluierung anhand zwei Szenarien bei jeweils AWS, Azure und Google Cloud zeigt die praktische Anwendung. Mit den gesicherten Artefakten wird ein solides Fundament für weitere Untersuchungen gegeben.

Abstract

Cloud computing is utilized in a variety of different fields. This technology holds prospects as well as risks. Providers and customers can become victims of cybercrime. Cloud computing can furthermore be used as a tool for illegal activities. The general guideline presented, enables forensic experts to identify and acquire artifacts from IaaS platforms. The guideline is divided into client and cloud forensics. The first phase covers client artifacts such as access tokens and logs. The cloud forensics phase consists of orientation, capture, acquisition and extraction. Further, the evaluation, composed of two scenarios respectively at AWS, Azure and Google Cloud, demonstrates practical application. The aquired artifacts provide a solid foundation for further investigations.

Inhaltsverzeichnis

1 Einleitung

In der heutigen digitalen Welt ist die Cloud allgegenwärtig vertreten. Viele Konzerne[6][42][32] nutzen Cloud-Dienste, insbesondere Infrastructure-as-a-Service (IaaS), als eigene Infrastruktur oder zur Bereitstellung ihrer Dienstleistungen. Der Einsatz von Cloud-Dienste bietet aber auch kleinen und mittelständischen Unternehmen attraktive Möglichkeiten[18]: Kurz bis langfristige Einsparungen, Skalierbarkeit, Flexibilität und Schutz vor Datenverlust [25]. Dabei spielen finanzielle, technische, organisatorische, personelle und rechtliche Faktoren eine Rolle bei der Entscheidung für oder gegen einen Einsatz von Cloud Technologie. Eine Nutzung durch Privatpersonen ist dabei nicht ausgeschlossen. Dies kann in der persönlichen Fort- und Weiterbildung oder Nutzung von kostenlosen Kontingenten begründet sein.
Mit den Chancen dieser Technologie kommen aber auch Risiken. Cloud-Anbieter und Cloud-Kunden können Ziele von Cyberkriminalität werden. Dabei können Cloud-Dienste verschiedene Rollen einnehmen[26]. Als aktiven Part, wenn diese für Cybercrime im engeren Sinne[1] genutzt werden. Als passiver Part, wenn diese Opfer von Cybercrime werden. Zusätzlich als Werkzeug, wenn diese für Cybercrime im weiteren Sinne[2] genutzt werden. Cloud-Dienste können dabei bewusst oder unbewusst, beispielsweise durch Übernahme von Virtuellen Maschinen durch Hacker[30], für illegale Aktivitäten benutzt werden.
Aufdeckung und Untersuchung dieser Fälle ist Aufgabe der IT-Forensik. Jedoch sind klassische forensische Methoden nicht eins zu eins auf die Cloud anwendbar[18][1]. Begründet durch den unbekannten Datenstandort, Kontrollverlust und Komplexitätsebenen[27].
Mithilfe des später vorgestellten Leitfadens können Forensiker:innen einen Fuß in die Tür der Cloud bekommen und so ein solides Fundament für eine bestmögliche Untersuchung bilden.

[1] e.g. Hacking, DDoS oder Datenbeschädigung
[2] e.g. Hosting von Darknet-Marktplätze oder Betrug

1.1 Zielsetzung und Abgrenzung

Das Ziel dieser Master-Thesis ist die Erstellung eines Leitfadens zur Identifizierung und Sicherung von Artefakten aus IaaS Plattformen. Die Zielgruppe dieses Leitfadens sind Forensiker:innen sowie Cloud-Ingenieur:inn. Erstere Gruppe soll auf die Besonderheiten und Fallstricke im Umgang mit IaaS Plattformen aufmerksam gemacht werden. Letztere Gruppe wird auf die Anforderungen an forensische Prozesse hingewiesen. Der Leitfaden ersetzt keine eigene Einarbeitung in die Thematik. Eine forensischen Untersuchung sollte erst nach Aufbau eines ausreichenden Kenntnis- und Wissensstand erfolgen[40].
Die Ausgangslage des Leitfadens ist die Untersuchung eines klassischen Asservats wie beispielsweise einen Computer, Laptop, Smartphone oder Server. Falls vor oder während der Untersuchung eine Nutzung von IaaS Plattformen festgestellt wird, so kann mithilfe des Leitfadens weitere Artefakte aus der Plattform gesichert werden. Die Untersuchung einer komplexen Architektur ist prinzipiell möglich, benötigt aber die Zusammenarbeit mehrere Forensiker:innen sowie Cloud-Ingenieur:innen.
Diese Master-Thesis behandelt nicht die Analyse der gesicherten Daten, wie zum Beispiel Festplatten- oder Arbeitsspeicherabbilder von Virtuellen Maschinen. Eine Analyse dieser Daten unterscheidet sich prinzipiell nicht von einer klassischen forensischen Untersuchung.

1.2 Methodik

Zur Erstellung des Leitfadens wird eine Literaturrecherche durchgeführt. Mit dieser werden Cloud-spezifische Herausforderungen und Problemstellungen ermittelt. Bestehende Methoden werden diskutiert und fließen in die Erstellung des Leitfadens mit ein.
Zur Evaluierung des Leitfadens werden zwei Szenarien bei drei Cloud Service Provider (CSP) durchgeführt. Eine Einarbeitung in diese CSP fand mithilfe des jeweiligen Schulungsangebots statt. Folgende Mittel wurden für diese Arbeit genutzt:

- Windows Computer mit Linux Subsystem
- Amazon AWS
- Amazon AWS-Schulungen
- Microsoft Azure
- Microsoft LEARN FÜR AZURE
- Google Cloud Platform
- Google Cloud Skills Boost
- ChatGPT zur Rechtschreibprüfung und Formulierungshilfe
- DrawIO zur Erstellung der Grafiken

2 Grundlagen

In diesem Kapitel werden kurz die Grundlagen zur Digitalen Forensik, Virtuelle Maschinen und Cloud Computing erläutert. Neben theoretischen Grundlagen wird das Kapitel mit praktischen Grundlagen zu Cloud Computing abgeschlossen. Diese praktischen Grundlagen werden vermehrt in der Evaluierung angewandt.

2.1 Digitale Forensik

Digitale Forensik ist die Anwendung wissenschaftlicher Methoden zur Analyse und Aufklärung von Vorfälle in IT-Systeme[49]. Die Digitale Forensik findet Anwendung in der Aufklärung von Cyberangriffe auf IT-Systeme, aber auch in der Beweiserhebung bei (polizeilichen) Ermittlungen.
Digitalen Forensik, von McKemmish [40] noch “Forensic Computing” genannt, wird von ebenjenem in 4 Phasen beschrieben:

1. **Identification** Kenntnis über welche digitale Beweise vorhanden sind und zusätzlich wo und wie diese gespeichert sind
2. **Preservation** Sicherung von digitalen Beweisen unter der Prämisse die gespeicherten Daten nicht zu verändern
3. **Analysis** Analyse und Umwandlung der extrahierten Rohdaten in konkrete Informationen
4. **Presentation** Berichterstattung der festgestellten Informationen und Schlussfolgerungen

Der forensische Prozess wurde von National Institute of Standards and Technology (NIST) [36] in folgende 4 Phasen formalisiert:

1. **Collection** Identifizierung potentieller Datenquellen und Sicherung von Rohdaten
2. **Examination** Untersuchung der Rohdaten zur Determinierung von relevanten Daten

3. **Analysis** Analyse der relevanten Daten um Schlussfolgerungen zum Sachverhalt zu ziehen
4. **Reporting** Berichterstattung der festgestellten Informationen und Schlussfolgerungen

Weiterführend gibt der *Guide to Integrating Forensic Techniques into Incident Response* [36] Empfehlungen zur Durchführung dieses Prozesses. Unter vielen Aspekten werden volatile Daten als Priorität dargestellt. Unter volatilen Daten versteht man flüchtige Daten wie beispielsweise CPU-Registereinträge oder Arbeitsspeicherinhalt.

Der *Leitfaden "IT-Forensik"* [50] unterteilt eine forensische Untersuchung in folgende Phasen:

1. **Strategische Vorbereitung** Pro-aktive Maßnahmen in Hinblick auf eine zukünftige forensische Untersuchungen, beispielsweise Aktivierung von Logdiensten
2. **Operationale Vorbereitung** Maßnahmen vor einer Datensammlung, beispielsweise Identifizierung potentieller Datenquellen
3. **Datensammlung** Sicherung von Rohdaten
4. **Untersuchung** Extraktion von relevanten Artefakte aus den vorher gesicherten Rohdaten
5. **Datenanalyse** Analyse der relevanten Daten um Schlussfolgerungen zum Sachverhalt zu ziehen
6. **Dokumentation** Berichterstattung der festgestellten Informationen und Schlussfolgerungen

Wie bei NIST [36] wird hier auch auf volatile Daten hingewiesen. Hierbei werden die Begriffe **Post-mortem-Analyse** und **Live-Forensik** genutzt um, unter weiteren Aspekten, einen Fokus auf nicht-flüchtige respektive flüchtige Daten zu beschreiben. Post-mortem-Analyse beschreibt die nachträgliche Untersuchung eines Vorfalls. Klassisch werden hier forensische Images von nichtflüchtigen Daten (Datenträgern) erstellt und diese auf den Vorfall hin untersucht. Im Kontrast dazu findet bei der Live-Forensik eine Untersuchung während des Vorfalls statt. Die Priorität liegt bei der Sicherung von flüchtigen Daten aus einem "Live" (laufendem) System. Ein:e Forensiker:in muss bei dieser Untersuchung zwischen zwei Faktoren abwägen: Wie viele/welche Daten werden durch einen Eingriff in das System geändert und wie

viele/welche Daten sind für eine erfolgreiche Untersuchung notwendig?
Zusätzliche werden Anforderungen an eine forensische Untersuchung gestellt[20][50]:

- **Akzeptanz** Genutzten Methoden müssen allgemein anerkannt oder nachweislich Fehlerfrei sein
- **Glaubwürdigkeit** Funktionalität der Methoden muss nachweisbar sein
- **Wiederholbarkeit** Genutzten Methoden und Werkzeuge sollten bei gleicher Ausgangslage dasselbe Ergebnis liefern
- **Integrität** Gesicherte Daten dürfen nicht verändert werden
- **Ursache und Auswirkung** Genutzte Methoden müssen logische Schlussfolgerungen ermöglichen
- **Dokumentation** Jeder Schritt muss ausreichend dokumentiert sein

Der Prozess einer forensischen Untersuchung ist keine einmalig festgelegte Abfolge. Dieser wird stetig weiterentwickelt um neue Sachverhalte, wie beispielsweise Smartphone-, IoT- und Car-Forensik, abdecken zu können. Adaptionen und Weiterentwicklungen für Cloud Forensik werden in Kapitel 3.2 vorgestellt.

2.2 Virtualisierung

2.2.1 Hardwarevirtualisierung

Hardwarevirtualisierung[52] [33] ermöglicht es mehrere Betriebssysteme (Virtuelle Maschine (VM)), auch “Guest” genannt, auf einem physischen Computer, “Host”, auszuführen. Diese Virtualisierungstechnologie abstrahieren ab der Hardwareebene, d. h. das Virtuelle Maschinen auf virtuelle Prozessoren, virtueller Arbeitsspeicher, virtuelle Datenträger und weitere virtuelle Ressourcen zurückgreifen. Ein Hypervisor, eine Software die auf dem Host ausgeführt wird, alloziert physische Ressourcen und stellt diese den Guests als virtuelle Ressourcen bereit.
Virtuelle Maschinen ermöglichen eine geteilte Nutzung der zur Verfügung stehenden Ressourcen (Rechenleistung, Arbeitsspeicher, Datenspeicher). Dabei sind mehrere VMs voneinander logisch getrennt.

2.2.2 Container

Containervirtualisierung abstrahiert im Vergleich zu Virtuellen Maschinen nicht ab der Hardwareebene sondern erst ab dem Betriebssystem. Konkret bedeutet dies, dass mehrere Container auf ein Betriebssytem-Kernel zurückgreifen, jedoch getrennt

voneinander ausgeführt werden. Dies hat zwar die Einschränkung, dass auf einem Host nur Container mit dem gleichen Kernel ausgeführt werden können, jedoch werden Ressourcen besser ausgeschöpft. Im Vergleich zur Hardwarevirutalisierung ist weniger Overhead vorhanden, da nicht mehrere Betriebssysteme ausgeführt werden müssen.

2.3 Cloud Computing

2.3.1 Theoretische Grundlagen

Der Begriff Cloud Computing wurde vom NIST[41] folgendermaßen definiert:

Definition 1 *Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.*

Cloud Computing muss hierbei folgende 5 essentielle Eigenschaften aufweisen:

- **On-demand self-service** Kunden können einseitig und ohne Zutun des Anbieters Ressourcen allozieren
- **Broad Network access** Funktionen sind über das Internet und durch standardisierte Zugänge erreichbar
- **Resource pooling** Ressourcen werden vom Provider gebündelt
- **Rapid elasticity** Ressourcen können, auch automatisch, in kürzester Zeit alloziert und wieder freigegeben werden
- **Measured Service** Ressourcennutzung wird überwacht und protokolliert um für beide Parteien Transparenz herzustellen

Cloud Computing lässt sich in 3 verschiedene Service-Modelle einteilen:

- **Software-as-a-Service (SaaS)** Anwendungen laufen auf der Infrastruktur des Anbieters, werden aber vom Kunden genutzt. Der Kunde hat dabei keinen Zugriff auf die unterliegende Infrastruktur.
- **Platform-as-a-Service (PaaS)** Der Anbieter stellt eine Umgebung für den Kunden bereit, die für Anwendungen genutzt werden kann.

- **Infrastructure-as-a-Service (IaaS)** Der Anbieter stellt die Infrastruktur bereit. Der Kunde entscheidet selbst über über Betriebssystem und Anwendung.

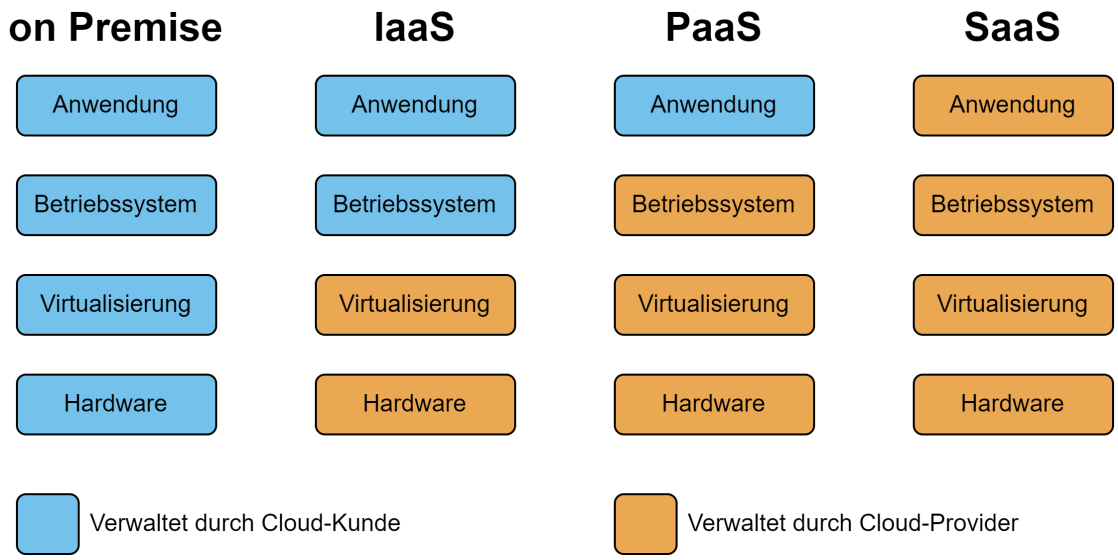


Abbildung 1: Service Modelle (angelehnt an [14])

Die Verwendung eines eigenen Rechenzentrums wird als 'on Premise' bezeichnet. Aus diesen Service-Modellen - on Premise, IaaS, PaaS und SaaS - kann man die einzelnen Verantwortungen für Cloud-Kunde und Cloud-Provider ableiten. Beim ersten Extrem, on Premise, steht alles in der eigenen Verantwortung. Hardwareskalierung und -ausfälle müssen eigenverantwortlich behandelt werden. Im Vergleich dazu werden über die einzelnen Service-Modelle immer mehr Verantwortung und somit Kontrolle an den Cloud-Provider abgegeben.

Weiterhin wird Cloud Computing in vier Einsatzmöglichkeiten unterschieden:

- **Privat Cloud** Die Infrastruktur wird exklusiv für eine Organisation bestehend aus mehreren Konsumenten bereitgestellt.
- **Community Cloud** Die Infrastruktur wird für eine spezielle Zielgruppe bereitgestellt. Ein Teil der Zielgruppe oder Dritte verwalten die Infrastruktur.
- **Public Cloud** Die Infrastruktur wird für die allgemeine Öffentlichkeit bereitgestellt.
- **Hybrid Cloud** Diese Einsatzmöglichkeit ist ein Verflechtung aus zwei oder mehreren Einsatzmöglichkeiten

2.3.2 Praktische Grundlagen

Im folgenden werden praktische Grundlagen zu Infrastructure-as-a-Service erläutert. Eine Interaktion ist allgemein nur durch Application Programming Interface (API) Aufrufe möglich. Diese API-Aufrufe können durch drei unterschiedliche Methoden[9] erfolgen:

- **Webinterface** (auch Console genannt) ist eine benutzerfreundliche Weboberfläche mithilfe dieser API-Aufrufe im Hintergrund durchgeführt werden und das Ergebnis auf der Weboberfläche angezeigt wird. Bei der Erstellung von Ressourcen werden verfügbare Parameter und Abhängigkeiten angezeigt. Das Webinterface verfügt meistens über eine sogenannte Cloud-Shell. Diese ermöglicht die Nutzung von CLI-Tools innerhalb des Webinterface.
- **Command Line Interface (CLI)** ist ein Kommandozeilen-Werkzeug mit dem API-Aufrufe über Texteingabe erfolgen. Aufrufe können mithilfe von Skripten automatisiert werden. Im weiteren Verlauf dieser Arbeit werden fast ausschließlich CLI-Tools aufgrund ihrer Konsistenz genutzt.
- **Software Development Kits (SDKs)** sind Programmbibliotheken, die z.B. in Pyhton-, C++, Java- oder PHP-Programme eingebunden werden können und damit API-Aufrufe ermöglichen. Abläufe von API-Aufrufe können automatisiert abgehandelt und API-Antworten weiter im Programm verarbeitet werden.

API-Aufrufe könne auch 'händisch' erfolgen, indem der API-Endpoint (meistens eine URL) unter Angabe von Zugangstoken und Parameter aufgerufen wird. Dies erfolgt ausnahmsweise einmalig in der Evaluierung (Azure CDN Szenario: Erfassung: Domain und DNS), da die benötigte API-Funktionalität vorhanden ist, jedoch noch nicht im CLI-Tool implementiert wurde.
Im weiteren Verlauf dieser Arbeit werden CLI-Tools der Cloud-Anbieter in Kombination mit weiteren Programme wie `tee`[1] oder `jq`[2] verwendet. Folgend wird allgemein die Nutzung dieser Tools am Beispiel von `gcloud compute addresses list` (Listing 1) erklärt. Hier und im weiteren Verlauf dieser Arbeit wird auf CLI-Aufrufe in Listings mithilfe der Angabe der Zeilennummer verwiesen. Beispielsweise wird der Verweis auf den Aufruf `gcloud compute addresses list` mithilfe der Zeilennummer innerhalb von runden Klammern (2) beschrieben. Dieser Aufrufe listet alle reservierten statischen IP-Adressen eines Projekts in der Google Cloud Platform auf.

[1] https://wiki.ubuntuusers.de/Shell/tee/
[2] https://jqlang.github.io/jq/

```
1  gcloud config set project MyCloudProject
2  gcloud compute addresses list
3  gcloud compute addresses list > gcloud_compute_addresses_list.txt
4  gcloud compute addresses list --format=json >
   ↪ gcloud_compute_addresses_list.json
5  jq '.[].address' gcloud_compute_addresses_list.json
6  gcloud compute addresses list --format=json | tee
   ↪ gcloud_compute_addresses_list.json | jq '.[].address'
```

Listing 1: Beispielnutzung CLI-Tools

Als Voraussetzung muss entweder bei jedem Aufruf der Parameter `--project=MyCloudProject` angegeben werden oder die aktuelle Konfiguration des CLI-Tools wird auf das Projekt *MyCloudProject* angepasst (1). Die Ausgabe des Aufrufs (2) enthält wenig Informationen (siehe Abbildung 2). Diese Ausgabe kann mit dem Zeichen `>` in eine Datei ausgeleitet werden (3). Die Kommandozeile zeigt dabei keine Ausgabe an. Unter Angabe des Parameters `--format=json` wird die Ausgabe in JavaScript Object Notation (JSON) formatiert (siehe Abbildung 3) und in diesem Beispiel direkt in eine Datei ausgeleitet (4). Diese Ausgabe ist mensch- und maschinenlesbar.

File: gcloud_compute_addresses_list.txt

	NAME	ADDRESS/RANGE	TYPE	PURPOSE	NETWORK	REGION	SUBNET	STATUS
1	static-website-external-ip	34.149.119.163	EXTERNAL					IN_USE

Abbildung 2: Ausgabe `gcloud compute addresses list`

Mithilfe des Programms `jq` können JSON-Dateien analysiert bzw. gefiltert werden. In diesem Beispiel wird der Filter `.[].address` und Eingabedatei `glcoud_compute_addresses_list.json` benutzt (5). Das bedeutet, dass aus der Liste aller Einträge (`.[]`) der jeweilige Adresseintrag (`.address`) ausgegeben werden. Das Programm gibt nur `"34.145.119.163"` in der Kommandozeile aus.

File: gcloud_compute_addresses_list.json


```json
[
  {
    "address": "34.145.119.163",
    "addressType": "EXTERNAL",
    "creationTimestamp": "2024-03-08T23:36:59.962-08:00",
    "description": "",
    "id": "4599098829634121588",
    "ipVersion": "IPV4",
    "kind": "compute#address",
    "labelFingerprint": "42WmSpB8rSM=",
    "name": "static-website-external-ip",
    "networkTier": "PREMIUM",
    "status": "IN_USE"
  }
]
```


Abbildung 3: Ausgabe `gcloud compute addresses list --format=json`

Im letzten Beispiel wird die Ausgabe des CLI-Aufruf mit einer Umleitung (Pipe-Operator `|`) in das Programm `tee` weitergeleitet und dessen Ausgabe anschließend mit einer weiteren Umleitung in das Programm `jq` weitergeleitet (6). Das Programm `tee` liest die Eingabe, schreibt diese in die Datei `gcloud_compute_addresses_list.json` und gibt den Inhalt als Ausgabe weiter. Der zusammengesetzte Kommandozeilenaufrufe (6) bewirkt somit dasselbe wie die Aufrufe (4) und (5): Die Ausgabe des Google Cloud CLI-Aufrufs wird in eine Datei geschrieben und anschließend wird die hinterlegte Adresse gefiltert.

Abschließend werden die unterschiedlichen Parameter-Gruppen bei Verwendung eines CLI-Tools erläutert. Im oben genannten Beispiel wird der Parameter `--format=json` verwendet. Bei der Nutzung der CLI-Tools gibt es drei verschiedene Gruppen von Parametern: erforderliche, optionale und globale. Erforderliche Parameter müssen bei einem Aufruf angegeben werden. Wird beispielsweise über `gcloud compute instances create` eine neue VM-Instanz erstellt, so muss der Name der neuen Instanz als Parameter übergeben werden, da das CLI-Tool ansonsten nicht oder fehlerbehaftet ausgeführt wird. Optionale Parameter können bei einem Aufruf angegeben werden. Werden optionale Parameter nicht angegeben, so wird von einem Standardwert ausgegangen. Um beim vorhin erwähnten Beispiel zu bleiben: `--machine-type` ist ein optionaler Parameter bei der Erstellung einer neuen VM-Instanz und spezifiziert den Maschinentyp. Im Maschinentyp ist die Konfiguration

der VM im Sinne von Anzahl Prozessoren, Größe Arbeitsspeicher, Plattform (Intel, AMD, ARM) und Optimierung (Speicheroptimiert, Computing-optimiert) kodiert. Wird der Parameter `--machine-type` nicht explizit angegeben, so wird der Standardwert `n1-standard-1` (1 vCPU, 3,75GB Arbeitsspeicher, Intel Skylake) genutzt. Globale Parameter können bei jedem Aufruf angegeben werden. Diese umfassen unter anderen `--format` um das Ausgabeformat anzugeben.
Im Laufe dieser Arbeit werden vermehrt CLI-Tools verwendet. Bei den Aufrufen werden optionale Parameter nur angegeben, wenn diese für den Aufbau eines Szenarios oder Durchführung des Leitfadens notwendig sind.

3 Leitfaden

Zur Erstellung des Leitfadens wird zuerst über Herausforderungen in der Cloud Forensik diskutiert. Anschließend werden bestehende Methodiken dargestellt. Abschließend wird der Leitfaden gebildet und die einzelnen Phasen im Detail erläutert

3.1 Herausforderungen

Cloud-Forensik ist ein multidimensionales Feld[48]. Neben der technologischen kommt eine rechtliche und organisatorische Dimension zu tragen. Herausforderungen können dabei in ein oder mehrere Dimensionen zugeteilt werden. Entsprechende Lösungen können dabei aus der gleichen oder mehreren anderen Dimension stammen. Die technische Dimension umfasst die Methoden und Werkzeuge die während einer forensischen Untersuchung genutzt werden. Dazu gehört die Identifizierung, Sicherung und Analyse von Artefakte. Die organisatorische Dimension umfasst die involvierten Parteien und deren Interaktionen untereinander bei einer forensischen Untersuchung. Je nach Umfang eines Vorfalls sind viele Parteien eingebunden:

1. Cloud-Kunden, bestehend aus Verantwortlichen, IT-Spezialisten, Rechtsanwälten
2. Cloud Service Provider (CSP), bestehend aus IT-Support, Incident Response IT-Spezialisten, Rechtsanwälten
3. Strafverfolgungsbehörden, bestehend aus Ermittlern, IT-Spezialisten, Staatsanwälten
4. Externe Dienstleister

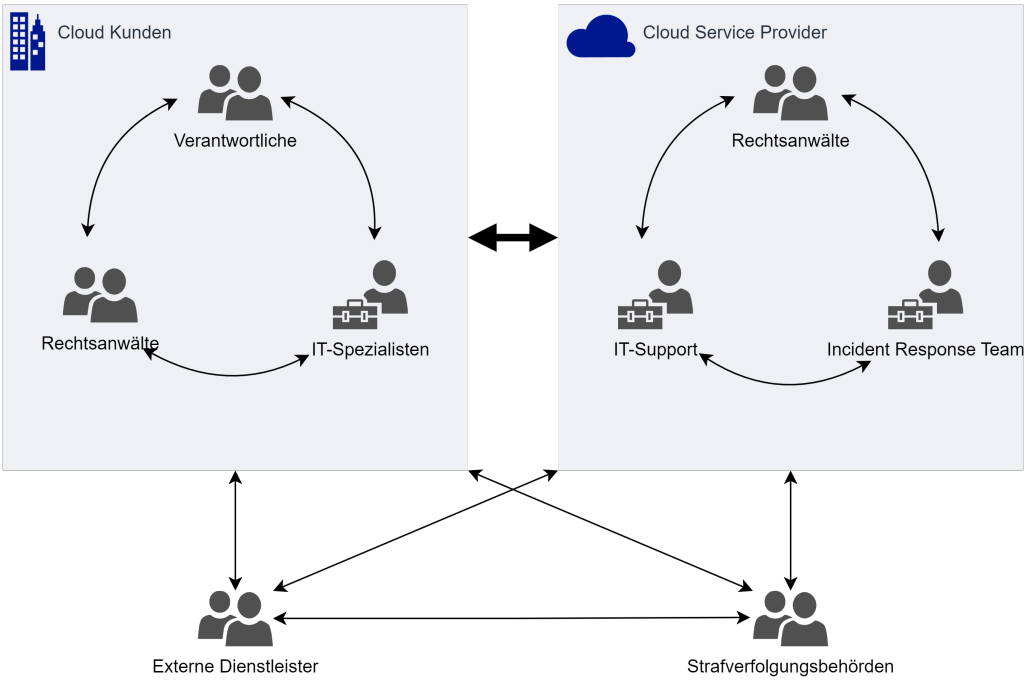


Abbildung 4: Organisatorische Dimension (angelehnt an [48])

Die dritte Dimension umfasst das rechtliche Element. CSP bieten ihr Angebot global an. Die grundlegende Infrastruktur ist auch global vertreten. Daraus kann sich eine komplexe mehrfach-justizielle Situation mit gegebenenfalls unterschiedlichen rechtlichen Umständen entwickeln: CSP mit Sitz in Amerika, Datencenter in Indien[43] und Cloud-Kunde aus Deutschland.

NIST[35] stellt eine Übersicht von Herausforderungen in der Cloud Forensik zusammen. Diese sind in folgende Kategorien aufgeteilt: Architektur, Datensicherung, Analyse, Anti-Forensik, Incident Response, Zugriffsverwaltung, Rechtlich, Standards und Training. Mehrere Herausforderungen liegen außerhalb des Umfangs dieser Arbeit. Folgende Herausforderungen werden aber näher betrachtet:

- **FC-08 Log capture (Protokollerfassung)**
- **FC-34 Live forensics**
- **FC-42 Data integrity and evidence preservation (Beweisintegrität)**

Die Erfassung von Protokollen spielt eine wichtige Rolle in der Cloud Forensik, da diese Aufschluss über Aktivitäten und Abläufe geben. Es existieren dabei mehrere unterschiedliche Arten von Protokolle. API-Aufrufe werden in einen eigenen Speicher protokolliert und geben wieder welcher Benutzer welche Aktion ausgeführt hat.

Darüber hinaus können Anwendungs-Informations und -Fehlerprotokolle in einen Cloud-Speicher umgeleitet werden. Abhängig vom CSP werden Protokolle unterschiedlich lange vorgehalten. Protokolle sind essentiell zur Erstellung einer Timeline. Die Live-Forensik spielt bei der Sicherung volatiler Daten eine wichtige Rolle. Im Kontext der Cloud Forensik wird diese Thematik verstärkt, da kein physischer Zugriff auf den Untersuchungsgegenstand gegeben ist. Auch spielt die Reihenfolge der Sicherung volatiler Daten eine Rolle, da aufgrund der Virtualisierung mehr Möglichkeiten bereitstehen. Während einer forensischen Untersuchung muss durchgehend der Umgang mit Beweisen beachtet werden.
Die Integrität der Daten spielt über die gesamte Untersuchung eine entscheidende Rolle, da eine Analyse von fehlerhaften Daten zu einem falschen Ergebnis führt.
Zusätzlich werden folgende Herausforderungen hervorgehoben:

- **Data Acquisition [3] [4] (Datensicherung)**
- **Data Provenance [5] [37] [44] (Datenherkunft)**
- **Semantic Integrity [53] (Datensemantik)**

Die Datensicherung beschreibt die forensische Sicherung von sowohl Metadaten als auch Inhaltsdaten. Metadaten beschreiben hierbei Informationen über eine Cloud-Ressource oder -Dienst wie z.B. Name, Erstellungsdatum oder verwendete Parameter. Inhaltsdaten sind Informationen, die von den Cloud-Ressourcen oder Diensten genutzt oder verarbeitet werden. Explizit sind dies beispielsweise Festplatten- und Arbeitsspeicherinhalt von virtuellen Maschinen, Datenbankeinträge, etc.
Die Datenherkunft spiegelt den Eigentum sowie Historie von Daten wieder. Durch Cloud-Dienste zur Versionierung, Redundanz, sowie Backup und Wiederherstellung kann von einer Datei mehrere Versionen an unterschiedlichen Orten existieren.
Die Herausforderung hinter Datensemantik besteht darin, nicht nur Inhaltsdaten an sich, sondern auch ihre Bedeutung festzustellen. Ein konkretes Beispiel kann eine Datenbank sein. Diese speichert Strukturen und Inhaltsdaten. Die Bedeutung einzelner Spalten wird aber erst durch die korrespondierende Dokumentation oder Analyse der korrespondierenden Anwendung bekannt.

3.2 Bestehende Methoden

Abseits der bestehenden allgemeinen IT-Forensik Methodiken[40][36][50] wurden verschiedene Frameworks für Cloud Forensik entwickelt. In diesem Abschnitt werden drei Frameworks vorgestellt, die in der Erstellung des Leitfadens Einfluss hatten.
Das Framework von Martini und Choo[38] besteht aus 4 Phasen und deckt damit die Phasen der klassischen IT-Forensik ab: Beweisidentifikation, Sicherung, Analyse, Präsentation. Zusätzlich findet bei diesem Framework eine oder mehrere Iterationen von der dritte Phasen (Analyse) zur ersten Phase (Beweisidentifikation) statt, wenn die Benutzung von Cloud Computing oder weiteren Cloud Ressourcen festgestellt wurde.
Martini und Choo[39] zeigen in einer folgenden Arbeit Schritte zur forensischen Sicherung exemplarisch an VMware vCloud. Dieser Prozess, bestehend aus 6 Schritten, reicht von der Feststellung von Administrator-Anmeldeinformationen über die Sicherung von Protokollen und Metadaten bis zur Sicherung der Inhaltsdaten einer VM.
Die Arbeit von Parkash et al.[47] stellt ein Cloud Forensik Framework vor, welches sich über 6 Phasen erstreckt. Die Vorbereitung (1) beschäftigt sich mit rechtlichen und technischen Fragestellungen. Vor der Untersuchung müssen rechtliche Dokumente vorhanden sein, welche den Rahmen der Untersuchung festlegen. Der technische Teil stellt die Vorbereitung und Verfügbarkeit der Werkzeuge sicher. Die Identifikations-Phase (2) beschreibt die Suche und Identifizierung von Beweisquellen. Diesbezüglich muss der Ort und Zugang der Beweise geklärt werden. Die Sicherungs-Phase (3) entscheidet über den Erfolg der nächsten beiden Phasen. Hier werden die vorher identifizierten Beweisquellen gesichert. Die Analyse-Phase (4) beschreibt die Untersuchung der gesicherten Artefakte. Diese können aufgrund ihres Ursprungs eine andere Struktur aufweisen als Artefakte wie beispielsweise aus der Datenträgerforensik. In der anschließend folgenden Rekonstruktions-Phase (5) wird eine Timeline der Ereignisse erstellt. Der Prozess wird mit der Präsentations-Phase (6) abgeschlossen. Untersuchte Artefakte und daraus resultierende Schlussfolgerungen müssen in einer Präsentation, Bericht oder Aussage verständlich formuliert werden.

3.3 Leitfaden

Der Leitfaden wird in Client-Forensik und Cloud-Forensik aufgeteilt (angelehnt an die Arbeit von Pichan et el.[45]). Die Phase der Client-Forensik umfasst die Untersuchung von Client-Geräten die zur Interaktion mit Cloud-Plattformen genutzt wurden. Ziel der Untersuchung ist ein Zugang zur genutzten Cloud-Plattform zu ermitteln (übernommen von [39]), der in der Phase der Cloud-Forensik genutzt werden kann. Zusätzlich können in dieser Phase auch schon Artefakte und Hinweise wie z. B. ausgeführte API Aufrufe oder Architektur-Diagramme festgestellt werden.
Die Phase der Cloud-Forensik umfasst die Identifikation der genutzten Services sowie Sicherung von Meta- und Inhaltsdaten. Am Anfang dieser Phase steht die Orientierung: Welcher Nutzerzugang wird genutzt und auf welche Projekte kann zugegriffen werden. Als nächstes folgt die Erfassung der genutzten Services inkl. Metadaten. Hier steht im Mittelpunkt die Identifizierung von Services und wie diese miteinander interagieren. Darauffolgend die Sicherung von Inhaltsdaten, d. h. im konkreten virtuelle Maschinen, Datenbanken, Objektspeicher und weitere. Diese Phase wird durch die Extraktion abgeschlossen. Je nach Anbieter werden Export- oder Sicherungsfunktionen angeboten.

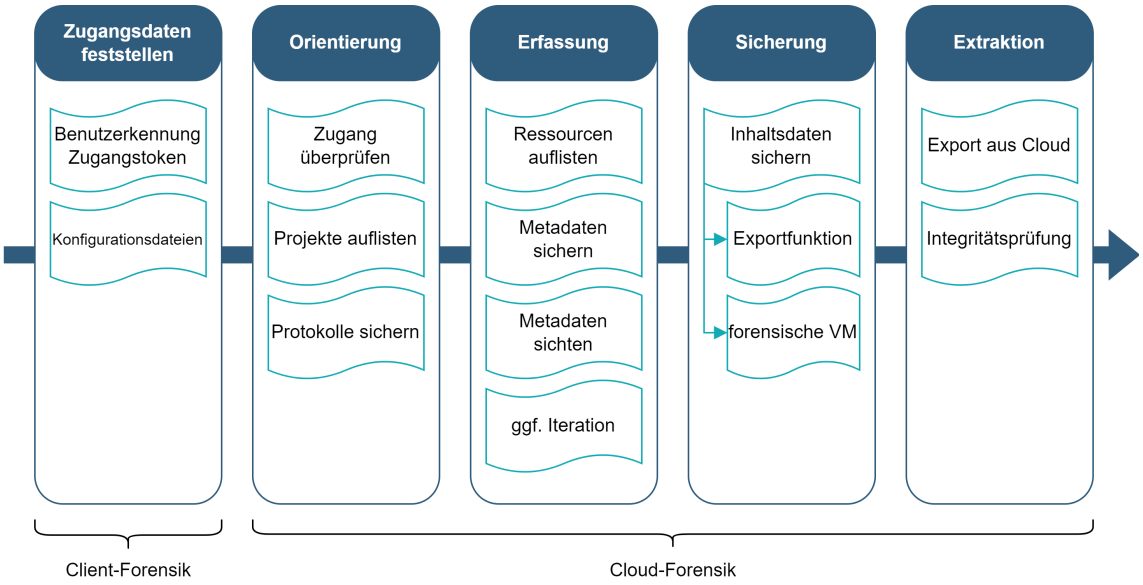


Abbildung 5: Ablauf Leitfaden

3.3.1 Client-Forensik

Die Phase der Client-Forensik hat das primäre Ziel Zugangsdaten und -tokens zu sichern, um die darauffolgende Phase der Cloud-Forensik überhaupt erst zu ermöglichen. Sekundäre Ziele bestehen darin Client-Artefakte wie z. B. API-Aufrufe und Logs für eine spätere Analyse zu sichern.
Zugangsdaten können einerseits als Benutzername-Passwort Paar in einem Browser oder Passwortmanager hinterlegt sein. Zugangstokens sind andererseits als Cookie im Browser oder in Konfigurationsdateien für CLI-Tools hinterlegt. Diese CLI-Tools ermöglichen einen Zugriff auf die API-Schnittstellen der jeweiligen CSP. Um eine Nutzung festzustellen, können folgende Kommandozeilenaufrufe in einem Live-System ausgeführt werden. Eine detaillierte Dokumentation sollte aufgrund der Interaktion mit einem Live-System erfolgen, da Veränderungen am System stattfinden.

```
1    aws --version
2    az version
3    gcloud version
```

Listing 2: AWS/Azure/Google Cloud CLI

Wurden CLI-Tools festgestellt oder befindet man sich in einer Post-mortem-Analyse, so können anschließend die Konfigurationsdateien ausgewertet werden. Abhängig vom lokalem Betriebssystem und CSP sind Zugangstokens an unterschiedlichen Pfaden abgelegt.

Windows	Pfad
AWS	`C:\Users\user\.aws\credentials`
Azure	`C:\Users\user\.azure\msal_token_cache.bin`
Google Cloud	`C:\Users\user\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11\LocalCache\Roaming\gcloud\credentials.db`

Linux	Pfad
AWS	`/home/user/.aws/credentials`
Azure	`/home/user/.azure/msal_token_cache.json`
Google Cloud	`/home/user/.config/gcloud/credentials.db`

MacOS	Pfad
AWS	`/Users/user/.aws/credentials`
Azure	`/Users/user/.azure/msal_token_cache.json`
Google Cloud	`/Users/user/.config/gcloud/credentials.db`

Artefakte zu API-Aufrufe können in der Historie der Kommandozeile zu finden sein. Microsoft's `az` und Google's `gcloud` legen zusätzlich lokale Protokolle an. `az` hält Protokolle an folgendem Pfad vor `/home/user/.azure/commands/`. Bei `gcloud` sind Protokolle an folgendem Pfad zu finden: `/home/user/.config/gcloud/logs/`.
Festgestellte Zugangsdaten oder -tokens werden abschließend genutzt um einen Zugang zur Cloud-Umgebung zu ermöglichen. Zugangsdaten können genutzt werden um CLI-Tools auf einem forensischen Computer einzurichten. Zugangstoken können an die oben genannten Pfade auf einem forensischen Computer kopiert werden, um CLI-Tools direkt verwendet zu können. Abhängig von unterschiedlichen Faktoren können Zugangstokens als ungültig erklärt werden.

3.3.2 Cloud-Forensik

Die Phase der Cloud-Forensik hat das primäre Ziel Artefakte und Metainformationen für eine anschließende Post-mortem-Analyse zu sammeln. Da in der Cloud-Umgebung virtuelle Maschinen, Datenbanken und weitere Ressourcen weiter ausgeführt werden, handelt es sich hierbei um eine Art Live-Forensik. Als Konsequenz müssen hier identische Maßnahmen zur Live-Forensik getroffen werden: Veränderungen auf ein Minimum reduzieren und die durchgeführten Schritte detailliert dokumentieren.

Orientierung Im ersten Abschnitt Orientierung wird der zur Verfügung stehende Zugang auf Berechtigungen und Projekte geprüft. Mithilfe der webbasierten Benutzeroberfläche kann ein Überblick verschafft werden. Zielgerichtet sollten Informationen jedoch über das CLI gesichert werden, da diese als JSON exportiert werden und damit für eine spätere Analyse maschinenlesbar sind. Außerdem können Abfragen über CLI programmatisch durchgeführt werden und erleichtern eine Erfassung von vielen Ressourcen. Ein weitere Vorteil der Nutzung von CLI besteht in der Dokumentation. CLI Sitzungen können mit der eingebauten History oder Tools wie z. B. Asciinema[1] aufgezeichnet werden. Meistens steht innerhalb der Weboberfläche eine Cloud-Shell zur Verfügung. In dieser Shell ist das CLI-Tool des jeweiligen Anbieters schon vorinstalliert und kann direkt genutzt werden. Neben Weboberfläche und CLI kann noch ein Software Development Kit (SDK) zur Interaktion genutzt werden. SDK können in verschiedenen Programmiersprachen verwendet werden um programmatisch API-Aufrufe zu tätigen und Informationen zu verarbeiten.

Zugang überprüfen	CLI-Aufruf
AWS	aws sts get-caller-identity
Azure	az account show
Google Cloud	gcloud config list

Projekte anzeigen	CLI-Aufruf
AWS	(Konzept von Projekten nicht vorhanden)
Azure	az group list
Google Cloud	gcloud projects list

[1] https://asciinema.org/

Der Abschnitt Orientierung wird mit der Sicherung der API-Protokolle abgeschlossen. Die Sicherung wird fast unmittelbar nach der Zugangsüberprüfung durchgeführt, da einerseits letzte Ereignisse gesichtet werden können und andererseits schließt man aus, dass eigene Aufrufe im Protokoll auftreten. Abhängig vom CSP sind unterschiedliche Aufbewahrungszeiträume vordefiniert. Diese können vom Cloud-Kunden meist angepasst werden.

Erfassung Im nächsten Abschnitt Erfassung werden zugreifbare Ressourcen aufgelistet und deren Metadaten gesammelt. Angelehnt an die Arbeit von Martini und Choo[38] finden hier mehrere Iterationen statt. Nachdem eine Ressource festgestellt wurde, werden Metadaten gesichert und gesichtet. Werden bei der Sichtung weitere Ressourcen oder Dienste erkannt, findet eine Iteration statt und die neuen Ressourcen werden erfasst. Es geschehen solange Iterationen bis keine neuen Ressourcen oder Dienste festgestellt werden. Die Erfassung der Metadaten spielt eine essentielle Rolle für die spätere Auswertung und Analyse, da die Metadaten Informationen wie ID, Name, Zeitstempel, Benutzerkennung, andere Ressourcen und allgemeine Parameter beinhalten. Diese Informationen sind in Verbindung mit den API-Protokollen unerlässlich um bei einer späteren Analyse eine Datenherkunft zu bestimmen.
Ein konkretes Beispiel für eine Iteration ist ein virtuelles Netzwerkes (siehe Abbildung 6). Bei der Erfassung des Netzwerks können Subnetze sowie Routing-Tabellen, Sicherheitsgruppen und Internet-Gateways festgestellt werden. In der nächsten Iteration werden diese erfasst, gesichtet und gegebenenfalls eine weitere Iteration angestoßen.
Da schon bei kleinen Projekten viele Cloud-Ressourcen zum Einsatz kommen können, empfiehlt es sich eine grafische Übersicht (siehe Abbildung 6) während oder beim Abschluss der Erfassung zu erstellen. Diese Übersicht dient als Checkliste, damit einerseits keine Ressourcen übersehen werden, andererseits als Ausgangslage für den nächsten Abschnitt der Sicherung. Bei der Erfassung von Ressourcen mit Inhaltsdaten werden diese für eine Sicherung vorgemerkt.

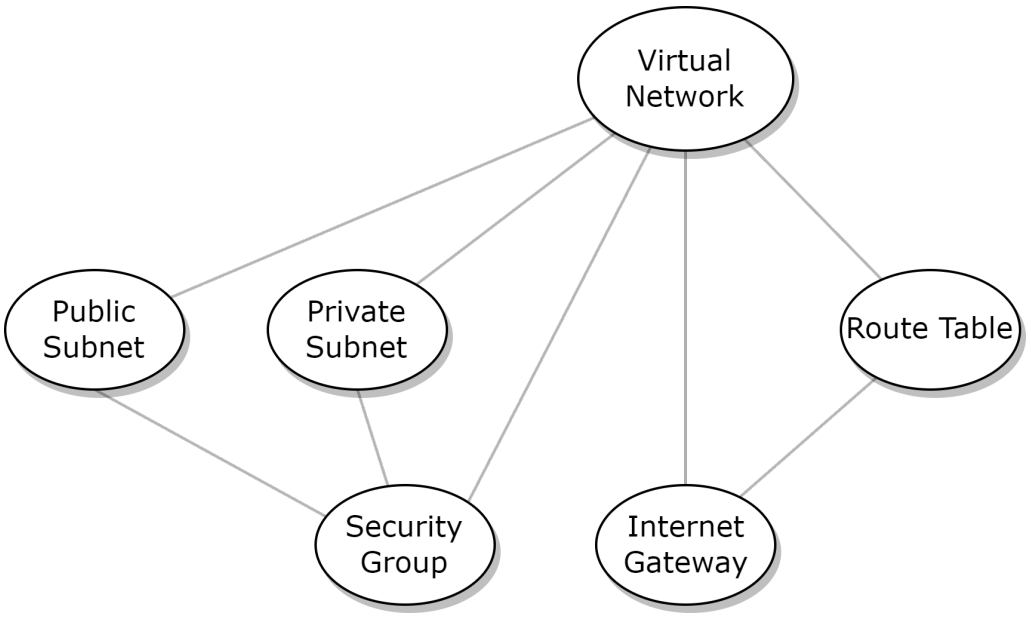


Abbildung 6: Beispiel: grafische Übersicht der Erfassung

Praktisch sollte die Erfassung mithilfe von Tools wie z. B. Cloud Forensics Utils[31] oder Kommandozeilenbefehle erfolgen. Diese bieten die Vorteile der Konsistenz, Reproduzierbarkeit, sowie Maschinen- und Menschen-lesbarem Output. Die Nutzung des Webinterface ist möglich um sich einen Überblick zu verschaffen. Zur besseren Dokumentation sollten jedoch die zuvor erwähnten Mittel genutzt werden. In der später folgenden Evaluierung werden Kommandozeilenbefehle zur Erfassung und Sicherung genutzt.

Sicherung Im Abschnitt Sicherung werden die Inhaltsdaten der vorgemerkten Ressourcen gesichert. Im Konkreten bedeutet das bei virtuelle Maschinen: den Arbeitsspeicher sowie die virtuelle Festplatte, bei Objektspeicher die beinhalteten Objekte (Dateien) und bei Datenbanken die enthaltenen Tabellen und Einträge. Ressourcen wie beispielsweise Serverless-Computing-Service oder virtuelle Netzwerkumgebungen beinhalten keine Inhaltsdaten und wurden daher schon im vorherigen Abschnitt vollständig erfasst.
Bei der Sicherung von VMs ist die Reihenfolge wichtig. Bei der klassischen Live-Forensik ist die Sicherung des Arbeitsspeichers von hoher Priorität[50]. Dies beruht auf der Reihenfolge der Volatilität[20]: von flüchtig (Arbeitsspeicher) zu nicht flüchtig (Datenspeicher). Dem steht bei VMs jedoch die Art und Weise der Sicherung entgegen. Wie folgend genauer erläutert, können Datenspeicher “von außen” ohne

direktes Einwirken auf die VM gesichert werden. Zur Sicherung des Arbeitsspeichers muss jedoch mit der VM interagiert werden. Zusätzlich bringt die Sicherung des Arbeitsspeichers immer das Risiko eines Systemabsturzes mit sich. Deshalb ergibt sich eine andere Reihenfolge[46]: Sicherung des Datenspeichers, Sicherung des Arbeitsspeichers, weitere Live-Forensik Maßnahmen.
Das zentrale Problem bei der Sicherung von Datenspeicher besteht darin, dass meistens keine Möglichkeit für einen direkten Export aus der Cloud-Umgebung existiert. CSP bieten für viele Ressourcen Backupfunktionen an. Diese Backups, bei virtuellen Maschinen Snapshots genannt, verbleiben in der Cloud-Umgebung, können aber über einen Umweg exportiert werden. Dieser Umweg wird durch eine forensische VM realisiert. Diese Methode beruht auf der Arbeit von Dykstra und Sherman[28]. Als erstes wird ein Snapshot des Ziel-Datenträger zur Laufzeit der VM erstellt. Aus diesem Snapshot wird ein Datenträger erstellt. Dieser neue Datenträger ist unabhängig von der VM und wird im Read-Only Modus mit einer forensischen VM verknüpft. Mithilfe dieser forensischen VM kann nun ein forensisches Image erstellt werden. Dykstra und Sherman[28] schlagen noch eine rigorose Alternativ vor. Dabei wird der Datenträger von der virtuellen Maschine getrennt und mit der forensischen VM verknüpft. Diese Methode verändert die ursprüngliche VM und es ist fragwürdig ob diese Alternative überhaupt bei laufenden VMs funktioniert.

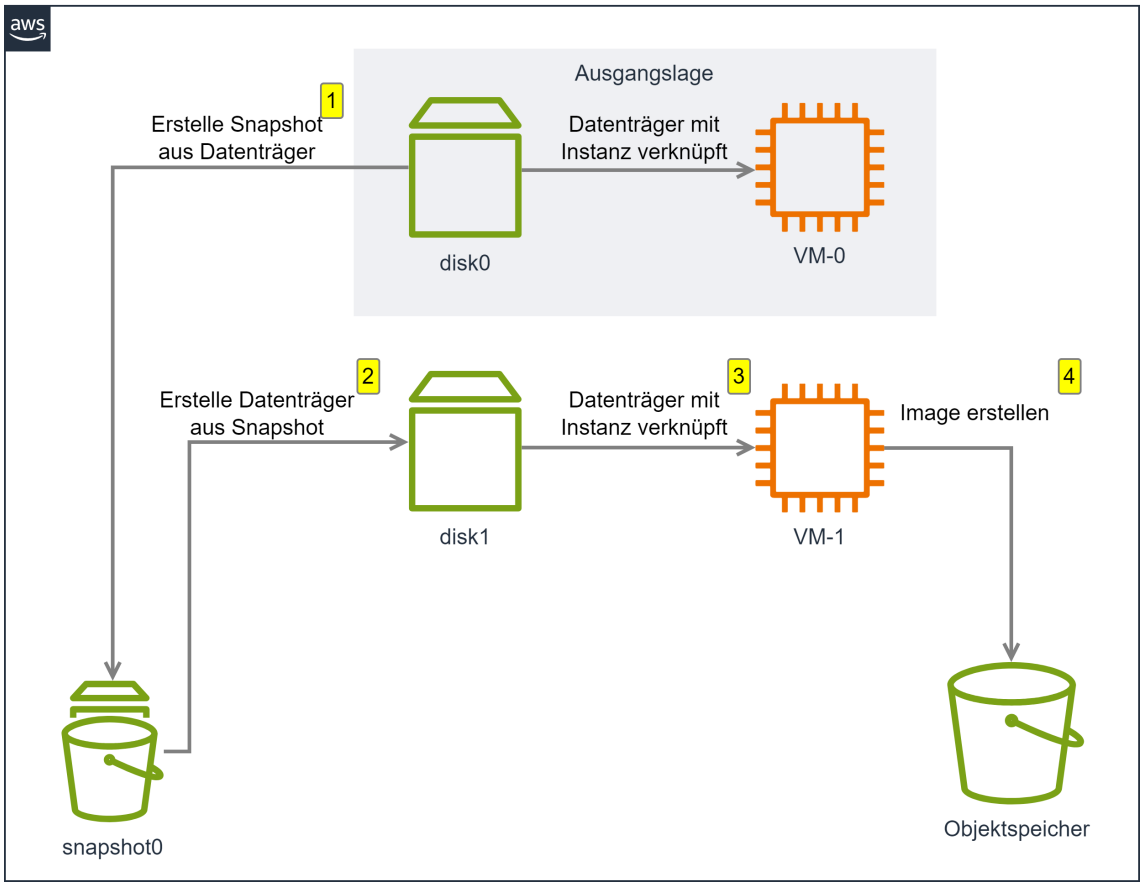


Abbildung 7: Volumesicherung am Beispiel AWS

Diese Methode wird am Beispiel AWS (siehe Abbildung 7) verdeutlicht. In der Ausgangslage ist eine VM-Instanz VM-0 mit verknüpftem Datenträger disk0 vorhanden. Dieser Datenträger kann eine Boot-Disk mit Betriebssystem oder auch ein Speichermedium sein. Im ersten Schritt wird aus dem Datenträger disk0 ein Snapshot snapshot0 erstellt. Dabei ist es unerheblich ob die VM-0 ausgeführt wird oder nicht. Im zweiten Schritt wird aus dem Snapshot snapshot0 ein neuer Datenträger disk1 erstellt. Der Datenträger disk1 wird im dritten Schritt mit der forensischen VM-Instanz VM-1 verknüpft. Mithilfe dieser VM-1 kann nun ein forensisches Image erstellt werden. Die Einbindung des Datenträgers disk1 ermöglicht zusätzlich die Berechnung eines Hashwerts für einen Hashwertabgleich. Während oder nach der Erstellung des Images und Hashwerts werden diese im vierten Schritt in einen Objektspeicher kopiert. Die gesicherten Daten können nun aus dem Objektspeicher exportiert werden.

Die Sicherung des Arbeitsspeichers einer VM wird als nächstes diskutiert. Allgemein werden die physische Ressourcen durch den Hypervisor verwaltet und den VMs zugeteilt. Theoretisch müsste der Hypervisor in der Lage sein den Arbeitsspeicher einer

VM zur Laufzeit zu sichern. Dies ist bei Hypervisor wie VMware ESXi/vSphere, Microsoft Hyper-V, Proxmox VE, KVM und Citrix Hypervisor möglich[2]. Bei CSP wie AWS, Azure und Google Cloud Platform wird diese Funktionalität jedoch nicht angeboten. Der Arbeitsspeicher kann, wie bei der klassischen Live-Forensik, von der VM selbst gesichert werden. Dazu muss eine Interaktive Sitzung mit der VM aufgebaut werden, beispielsweise mittels Secure Shell (SSH) oder Remote Desktop Protocol (RDP). Anschließend kann mithilfe des passenden Werkzeugs (Lime, linpmem, AVML, etc.) eine Arbeitsspeichersicherung angestoßen werden. Ist eine Sicherung erfolgt, so bietet es sich an direkt einen Hashwert zu errechnen. Abschließend kann die Sicherung und Hashwert in einen Objektspeicher kopiert werden. Dieser Prozess, bestehend aus Verbindungsaufbau, Sicherung und Export, wird auch in automatisierten Workflows von AWS für EC2[7] widergespiegelt. In beiden Varianten, per Hand oder automatisiert, werden durch die Interaktion mit dem System Veränderungen verursacht. Diese Veränderungen sollten ausreichend dokumentiert werden. Die Durchführung der Arbeitsspeichersicherung sollte per Hand erfolgen, damit einerseits der oder die Forensiker:in im Stande ist die durchgeführten Aktionen vor Gericht zu vertreten und andererseits der vorher erwähnten Workflow als nicht gerichtsfest von AWS beschrieben wird[8].
Die zuvor vorgestellte forensische VM kann auch zur Sicherung von Datenbanken genutzt werden, falls keine Exportfunktion angeboten wird. Dazu müssen die zuvor erfassten Metadaten gesichtet und ausgewertet werden. Die Instanziierung der forensischen VM muss entsprechend so erfolgen, dass ein Zugriff von jener auf die Datenbanken möglich ist. Mithilfe von API-Aufrufe zur Datenbankadministrierung kann ein Zugang zur Datenbank generiert oder für ein bestehender Zugang das Passwort neu gesetzt werden. Datenbanken können anschließend mit dem passenden Werkzeug (mysqldump, pg_dump, etc.) gesichert werden.
Bei der Sicherung von Objektspeichern müssen drei Thematiken beachtet werden: Änderungszeitstempel, gelöschte Dateien, Datenintegrität. Änderungszeitstempel müssen erhalten bleiben, um eine korrekte Timeline erstellen zu können. Es kann jedoch vorkommen, dass diese bei einer Sicherung nicht übernommen werden. Dieses Problem kann mithilfe einer Inhaltsübersicht gelöst werden. Diese sollte schon mit dem vorherigen Abschnitt Erfassung vorhanden sein. Gelöschte Dateien werden in der klassischen digitalen Forensik mittels Carving wiederhergestellt. Dies ist in einer Cloud Umgebung nicht möglich, da nicht eindeutig bestimmt werden kann wo sich eine Datei befand, befindet oder befinden wird[18]. Gelöschte Dateien können aber mithilfe von Backups oder Versionierung wiederhergestellt werden. Bei Objektspeichern muss überprüft werden ob diese Funktionalitäten vorhanden sind. Die Daten-

integrität kann mithilfe von Hashwertabgleich geprüft werden. Einige CSP berechnen für jede Datei einen Message-Digest Algorithm 5 (MD5)-Hash und speichern diesen in der Inhaltsübersicht ab.
Das Angebot an Diensten und Ressourcen ist je nach CSP unterschiedlich und ändert sich ständig. Allgemein muss bei einem Dienst oder Ressource geprüft werden ob Inhaltsdaten vorhanden sind. Anschließend wird geprüft, ob eine Exportfunktion genutzt werden kann. Alternativ kann eine Sicherung unter Umständen mithilfe einer forensische VM erfolgen.

Extraktion Der letzte Abschnitt Extraktion beschäftigt sich mit dem Export aus der Cloud-Umgebung. Gesicherte Inhaltsdaten sollten in einem, ausschließlich für forensische Zwecke erstellten, Objektspeicher zwischengespeichert sein und können nun aus der Cloud exportiert werden. Um die Integrität der Daten zu gewährleisten sollten die extrahierten Daten mithilfe einem Hashwertabgleich geprüft werden.

4 Evaluierung

In diesem Kapitel wird der vorgestellte Leitfaden anhand von zwei Szenarien bei drei CSP evaluiert. Anfangs wird die Auswahl der CSP erläutert und anschließend die Gestaltung der Szenarien diskutiert. Darauf folgt die Durchführung der Szenarien, aufgeteilt in jeweils Aufbau des Szenarios und Durchführung des Leitfadens. Dieses Kapitel wird mit einer Zusammenfassung abgeschlossen.

4.1 Auswahl der Cloud Service Provider

Zur Evaluierung wurden die drei IaaS-Anbieter mit dem größtem Marktanteil[51] gewählt. Diese sind in der Reihenfolge des Marktateils: Amazon Web Services (AWS), Microsoft Azure sowie Google Cloud Platform. Dadurch werden ca. 66% des Marktes abgedeckt. AWS kann, durch die Einführung von Elastic Compute Cloud (EC2) in 2006, als Wegbereiter für das IaaS Model gezählt werden[19].
Die Angebote der drei genannten Anbieter überschneiden sich in mehreren Gebieten. Beispielsweise wird von allen Anbietern Virtuelle Maschinen, Datenbanken, Objektspeicher, Containerdienste und ereignisgesteuerte Computing-Dienste angeboten. Die Voraussetzungen, Nutzung und Abhängigkeiten der einzelnen Dienste unterscheiden sich teilweise von Anbieter zu Anbieter. Dadurch unterscheiden sich die später gewählten Szenarien im Aufbau und geben neue Ausgangssituationen zur Durchführung des Leitfadens.

4.2 Gestaltung der Szenarien

Die Erstellung der Szenarien gestaltet sich schwierig, da diese mehrere Kriterien erfüllen sollen. Die Szenarien sollen einfach aufgebaut sein um eine Reproduzierbarkeit zu gewährleisten. Jedoch sollen sie auch mehrere Dienste abdecken, um eine ausreichende Fülle an Artefakten zu generieren. Weiterhin sollen nur Dienste genutzt werden, die auch bei allen drei CSP zur Verfügung stehen.

Als erstes Szenario wurde ein Wordpress-Anwendung mit mehreren Eigenschaften gewählt. Die Wordpress-Anwendung soll:

1. auf einer virtuellen Maschine ausgeführt werden,
2. aus dem Internet erreichbar sein,
3. als Datenbank ein Cloud-Service nutzen.
4. zusätzlich soll die Datenbank aus dem Internet nicht erreichbar sein.

Bei diesem Szenario wird auf Technologien zurückgegriffen, die auch in einem On-Premise Rechenzentrum zum Einsatz kommen: virtuelle Maschinen und Datenbanken. Diese werden in einen Cloud-Kontext gebracht um aufzuzeigen welche Einschränkungen oder auch Möglichkeiten im Vergleich zu On-Premise Einsätze entstehen. Der Datenträger, sowie Arbeitsspeicher der virtuellen Maschine und die Datenbank stellen dabei Inhaltsdaten dar. Die Konfiguration der Ressourcen zur Zugriffsbeschränkung sind weitere Artefakte. Bei diesem Szenario wird explizit auf Redundanz verzichtet, da zum einen doppelte VMs nur quantitativ und nicht qualitativ mehr Artefakte erzeugen und weiterhin Dienste für Redundanz wie Load-Balancer keine Inhaltsdaten innehalten. Es wurde die Anwendung Wordpress genutzt, da diese wenig Abhängigkeiten besitzt und leicht zu installieren ist. Zusätzlich wird durch die Anwendung bei der Einrichtung Inhaltsdaten in die Datenbank geschrieben. Diese werden bei der forensischen Untersuchung, zusätzlich zur virtuellen Maschine, gesichert.
Als zweites Szenario wurde eine statische Website mit mehreren Eigenschaften gewählt. Die Website soll:

1. in einem Objektspeicher abgelegt sein,
2. mithilfe eines Content Delivery Network (CDN) Inhalte cachen,
3. über eine eigene Domain erreichbar sein.

Bei diesem Szenario werden mehrere Cloud-Services verwendet, die im ersten Szenario nicht verwendet wurden. Dadurch werden andere Artefakte aufgezeigt. Die statische Website, abgelegt im Objektspeicher, stellen Inhaltsdaten dar. Informationen über die Domain sowie CDN sind weitere Artefakte.
Dieses Szenario ist angelehnt an die Cloud Resume Challenge[29], die daraus besteht einen eigenen Lebenslauf mithilfe von IaaS-Platformen zu hosten und zu verwalten.

4.3 Durchführung

Um die Durchführung reproduzierbar zu gestalten werden hauptsächlich Kommandozeilenbefehle genutzt. Diese können über die jeweiligen Cloud-Shells oder lokal genutzt werden. Für eine lokale Nutzung ist die Installation und Initialisierung (Anmeldung am Cloud-Provider) der Tools Voraussetzung:

1. **AWS CLI[10]** `aws`
2. **Azure CLI[17]** `az`
3. **Google Cloud CLI[23]** `gcloud`

Eine Durchführung der Client-Forensik wird übersprungen, da diese bei allen Szenarien identisch ist und primär nur das Ziel hat, einen Zugang zur Plattform festzustellen. Die Durchführung startet mit der Cloud-Forensik und geht von einem Administrator-Zugang aus.

4.3.1 AWS: Wordpress

Beim CSP AWS wird eine virtuelle Maschine (*EC2*) und eine von AWS verwaltete relationale Datenbank (*Relational Database Service (RDS)*) verwendet. Die virtuelle Maschine (Abbildung 8: 1) befindet sich in einem Subnetz in der Region us-east-1 in einem virtuellen Netzwerk (*Virtual Private Cloud (VPC)*). Eine Routing Tabelle des Public Subnet ermöglicht, dass die VM einerseits Ressourcen des ganzen Virtuellen Netzwerks erreicht, andererseits aber auch, dass Netzwerkverkehr aus und zu dem Internet über ein Internet Gateway (2) geroutet wird. Die VM ist durch eine öffentliche IP-Addresse aus dem Internet erreichbar, mit einer lokalen IP-Addresse aus dem lokalen VPC. Eine Routing-Tabelle im Private Subnet sorgt für ein Routing aus den Private Subnet zum Public Subnet. Die virtuelle Maschine ist mit einer *Security Group* verknüpft. Diese erlaubt jeden ausgehenden Netzwerkverkehr, beschränkt jedoch den eingehenden Netzwerkverkehr auf Port 22 (Standard SSH Port) und 80 (Standard HTTP Port) von jeder Quelle. Die Datenbank (3) befindet sich im Private Subnet und ist nicht aus dem Internet erreichbar. Eine zweite Security Group, die mit der Datenbank verknüpft ist, sorgt dafür, dass eine Netzwerkverbindung nur zu Port 3306 (Standard MySQL Port) und aus dem IP-Bereich des virtuellen Netzwerks aufgebaut werden kann.

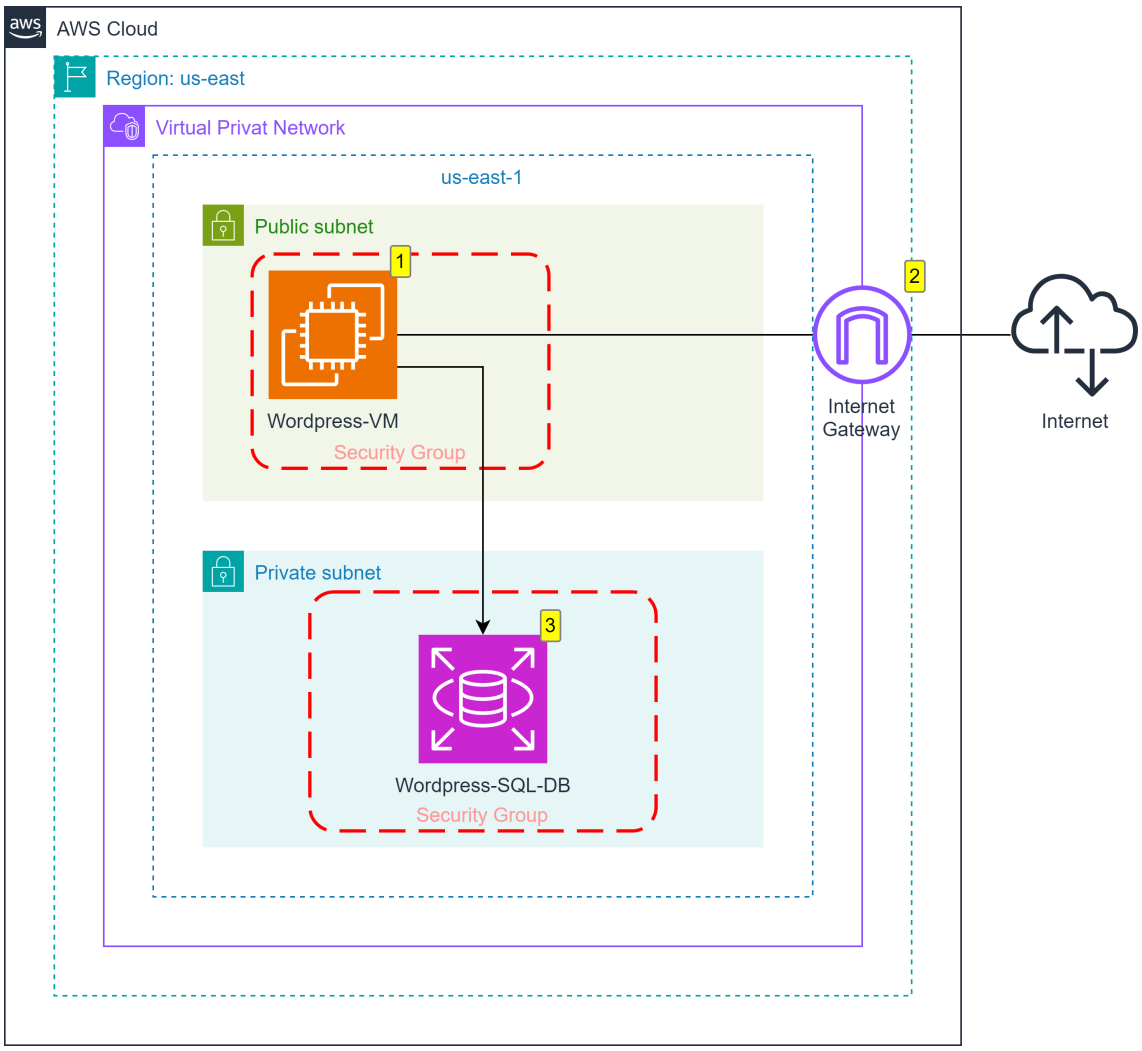


Abbildung 8: AWS Wordpress Szenario

Zur Vereinfachung wurde in der Abbildung auf ein zweites Private Subnet verzichtet. Dies wird zu Erstellung einer sogenannten *DB subnet group* benötigt. Eine DB subnet group muss sich über zwei Subnetze und zwei Availability Zones erstrecken. Dies kann genutzt werden um mithilfe von mehrere Datenbank-Instanzen eine Ausfallsicherheit zu garantieren. In diesem Szenario wird jedoch nur eine Datenbank-Instanz verwendet.

Aufbau

Zu Beginn (Listing 3) wird ein virtuelles Netzwerk Wordpress-vpc (1) mit 3 Subnetze (Wordpress-public-subnet(2), Wordpress-private-subnet1 (3), Wordpress-private-subnet2(4)) erstellt.

```
1 aws ec2 create-vpc --cidr-block 10.10.0.0/16 --tag-specification
  ↪ ResourceType=vpc,Tags='[{Key=Name,Value=Wordpress-vpc}]' --query
  ↪ Vpc.VpcId --output text
2 aws ec2 create-subnet --vpc-id vpc-024f68210dd4ee200 --cidr-block
  ↪ 10.10.0.0/24 --availability-zone us-east-1a --tag-specification
  ↪ ResourceType=subnet,Tags='[{Key=Name,Value=Wordpress-public-subnet}]'
  ↪ --query Subnet.SubnetId --output text
3 aws ec2 create-subnet --vpc-id vpc-024f68210dd4ee200 --cidr-block
  ↪ 10.10.1.0/24 --availability-zone us-east-1a --tag-specification
  ↪ ResourceType=subnet,Tags='[{Key=Name,Value=Wordpress-private-subnet1}]'
  ↪ --query Subnet.SubnetId --output text
4 aws ec2 create-subnet --vpc-id vpc-024f68210dd4ee200 --cidr-block
  ↪ 10.10.2.0/24 --availability-zone us-east-1b --tag-specification
  ↪ ResourceType=subnet,Tags='[{Key=Name,Value=Wordpress-private-subnet2}]'
  ↪ --query Subnet.SubnetId --output text
```

Listing 3: AWS Wordpress Szenario: VPC & Subnet Einrichtung

Als nächstes (Listing 4) wird ein Internet Gateway erstellt (1) und mit den virtuellen Netzwerk verknüpft (2). Eine Routing Tabelle (3) mit Route zum Internet (4) wird mit dem Public Subnet verknüpft (5). Die erste Security Group mit Namen AllowSSHandHTTPTraffic (7) erlaubt eingehender SSH (8) und HTTP (9) Verbindungen aus allen Quellen. Die zweite Security Group mit Namen AllowDBAccess (11) erlaubt eingehende MySQL (12) Verbindungen aus dem gesamten virtuellen Netzwerk.

```
1 aws ec2 create-internet-gateway --query InternetGateway.InternetGatewayId
  ↪ --output text
2 aws ec2 attach-internet-gateway --vpc-id vpc-024f68210dd4ee200
  ↪ --internet-gateway-id igw-0515ea491812bcac0
3 aws ec2 create-route-table --vpc-id vpc-024f68210dd4ee200 --query
  ↪ RouteTable.RouteTableId --output text
4 aws ec2 create-route --route-table-id rtb-0789f760cd506e4e0
  ↪ --destination-cidr-block 0.0.0.0/0 --gateway-id igw-0515ea491812bcac0
5 aws ec2 associate-route-table --route-table-id rtb-0789f760cd506e4e0
  ↪ --subnet-id subnet-0ab32bde87c720b1b
6
7 aws ec2 create-security-group --group-name AllowSSHandHTTPTraffic
  ↪ --description "AllowSSHandHTTPTraffic" --vpc-id vpc-024f68210dd4ee200
8 aws ec2 authorize-security-group-ingress --region us-east-1 --group-id
  ↪ sg-031ddd7fff971ec0a --protocol tcp --port 22 --cidr 0.0.0.0/0
9 aws ec2 authorize-security-group-ingress --region us-east-1 --group-id
  ↪ sg-031ddd7fff971ec0a --protocol tcp --port 80 --cidr 0.0.0.0/0
10
11 aws ec2 create-security-group --group-name AllowDBAccess --description
  ↪ "AllowDBAccess" --vpc-id vpc-024f68210dd4ee200
12 aws ec2 authorize-security-group-ingress --region us-east-1 --group-id
  ↪ sg-079e21a1df30ee4c3 --protocol tcp --port 3306 --cidr 10.10.0.0/24
```

Listing 4: AWS Wordpress Szenario: Internet Gateway & Security Groups

Darauffolgend (Listing 5) wird die virtuelle Maschine Wordpress-vm (1) im Public Subnet und mit der passenden Security Group gestartet. Wie am Anfang dieses Abschnitts begründet muss noch eine DB subnet group erstellt werden (2). Die nun erstellte MySQL Datenbank-Instanz (3) wird in der selben Availabilty Zone wie die virtuelle Maschine instanziiert. Zur späteren Nutzung wird noch die Adresse der Datenbank-Instanz festgestellt (4).

```
1 aws ec2 run-instances --image-id ami-0a3c3a20c09d6f377  --instance-type
  ↪ t2.micro --subnet-id subnet-0ab32bde87c720b1b
  ↪ --associate-public-ip-address  --security-group-ids sg-031ddd7fff971ec0a
  ↪ --region us-east-1 --tag-specifications
  ↪ 'ResourceType=instance,Tags=[{Key=Name,Value=Wordpress-vm}]'
2 aws rds create-db-subnet-group --db-subnet-group-name
  ↪ wordpress-privat-subnet --db-subnet-group-description "privat subnet for
  ↪ DB" --subnet-ids
  ↪ '["subnet-0ad37b05df160b277","subnet-014ec80f8a86adddd"]'
3 aws rds create-db-instance --db-instance-identifier wordpress-sql-db
  ↪ --db-instance-class db.t2.micro --engine mysql --db-subnet-group-name
  ↪ wordpress-privat-subnet --no-publicly-accessible --availability-zone
  ↪ us-east-1a --vpc-security-group-ids '["sg-079e21a1df30ee4c3"]'
  ↪ --master-username root --master-user-password securepassword
  ↪ --allocated-storage 20
4 aws rds describe-db-instances  --db-instance-identifier wordpress-sql-db
  ↪ --query "DBInstances[*].Endpoint.Address"
```

Listing 5: AWS Wordpress Szenario: EC2 & RDS Instanz erstellen

Vor der Wordpress-Installation muss die Datenbank noch eingerichtet werden (Listing 6). Dazu wird eine SSH-Verbindung mit der virtuellen Maschine (1) hergestellt, den mysql-client installiert (2) und eine kaskadierende Verbindung zur Datenbank hergestellt(3). In dieser wird ein Benutzer (5) und Datenbank (6) erstellt. Danach werden noch Berechtigungen gesetzt (7,8)

```
1     aws ec2-instance-connect ssh --instance-id i-0354b4055dd186553
2     sudo dnf install mariadb105
3     mysql -u root -h
      ↪ wordpress-sql-db.cxwq8coomx2p.us-east-1.rds.amazonaws.com -P 3306 -D
      ↪ mysql -p
4
5     CREATE USER 'wordpress-user'@'%' IDENTIFIED BY 'securepassword';
6     CREATE DATABASE `wordpress-db`;
7     GRANT ALL PRIVILEGES ON `wordpress-db`.* TO "wordpress-user"@"%";
8     FLUSH PRIVILEGES;
9     exit
```

Listing 6: AWS Wordpress Szenario: Datenbankeinrichtung

Abschließend (Listing 7) wird noch Wordpress installiert und initialisiert. Dazu wird auf der VM der Apache HTTP Webserver und PHP installiert (1). Weiter wird

die aktuellste Wordpress Version heruntergeladen (2), entpackt (3) und konfiguriert (4,5). In der Konfiguration müssen noch die genannten Felder (7-10) angepasst werden. Schließlich wird noch Wordpress in das Webserver-Verzeichnis kopiert (12) und der Webserver gestartet (14). Die Wordpress Anwendung kann nun über die externe IP-Adresse der Virtuellen Maschine aufgerufen, eingerichtet und verwaltet werden. Der Aufbau ist hiermit abgeschlossen.

```
1     sudo dnf install -y httpd php php-mysqli
2     wget https://wordpress.org/latest.tar.gz
3     tar -xzf latest.tar.gz
4     cp wordpress/wp-config-sample.php wordpress/wp-config.php
5     nano wordpress/wp-config.php
6
7         define( 'DB_NAME', 'wordpress-db' );
8         define( 'DB_USER', 'wordpress-user' );
9         define( 'DB_PASSWORD', 'securepassword' );
10        define( 'DB_HOST',
          ↪  wordpress-sql-db.cxwq8coomx2p.us-east-1.rds.amazonaws.com' );
11
12    cp -r wordpress/* /var/www/html/
13    sudo systemctl start httpd
```

Listing 7: AWS Wordpress Szenario: Installation

Durchführung Leitfaden

Orientierung Im Abschnitt Orientierung wird der eigene Zugang auf Berechtigungen geprüft (Listing 8) und Logs der API-Calls gesammelt (Listing 9). Mithilfe von `aws sts get-caller-identity` werden Informationen zum aktuellen Benutzer gesammelt (1) und die enthaltene Amazon Resource Name (ARN) notiert. Im nächsten Schritt werden alle Informationen über Benutzer, Gruppe, Rollen und Policies gesammelt (2) und Berechtigungen des eigenen Nutzers angezeigt.

```
1 aws sts get-caller-identity | tee aws-own-identity.json | jq '.Arn'
  ↪ aws-own-identity.json
2 aws iam get-account-authorization-details | tee aws-account-details.json |
  ↪ jq '.[] |.[] | select (.Arn=="arn:aws:iam::12345:user/CLI")'
```

Listing 8: AWS Wordpress Szenario: Orientierung: Eigener Zugang prüfen

Die API-Logs (Listing 9) werden beim AWS-Dienst *Cloudtrail* verwaltet und in

Objektspeicher (*Buckets*) gespeichert. Um diese zu exportieren werden zuerst alle Protokollgruppen (*Trails*) gelistet (1) und anschließend der Bucketname für die `management-events` gesucht. Die `management-events` enthalten Protokolle über API-Aufrufe. Als nächstes können die API-Logs exportiert werden (3). Abschließend sollten die API-Logs in ein Archiv gepackt werden (4) um Veränderungen zu vermeiden.

```
1 aws cloudtrail list-trails | tee aws_cloudtrail_list-trails.json | jq
  ↪ '.Trails | .[] | .Name'
2 aws cloudtrail get-trail --name management-events | tee
  ↪ aws_cloudtrail_get-trail.json | jq '.Trail.S3BucketName'
3 aws s3 sync s3://aws-cloudtrail-logs-734635139315-6e37996e log-export
4 tar czf aws-cloudtrail-logs-734635139315-6e37996e.tar.gz logs-export
```

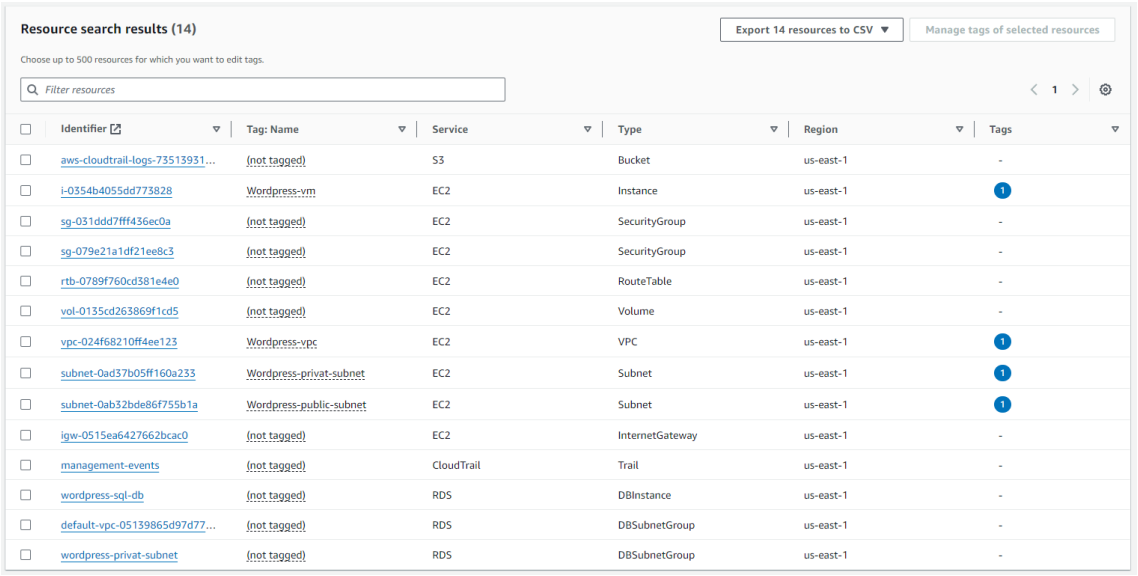
Listing 9: AWS Wordpress Szenario: Orientierung: API-Logs exportieren

Erfassung Der erste Schritt in der Erfassung besteht darin die genutzten Ressourcen zu suchen und aufzulisten. Dafür gibt es bei AWS mehrere Ansätze. Mithilfe des gezeigten CLI-Kommandos (Listing 10) werden alle Ressourcen mit Tags aufgeführt. Ressourcen ohne Tags werden hier nicht aufgeführt. Ein weiterer Ansatz besteht darin, den Tag Editor[1] auf dem AWS Webinterface zu nutzen. Dieser durchsucht alle unterstütze Ressourcen mit und ohne Tags und listet diese auf. Als letzte Alternative können in einer Art Brute-Force-Methode alle verfügbaren CLI-Kommandos mit dem Schlüsselwort `list` oder `describe` ausgeführt werden. An dieser Stelle wurde der Tag Editor genutzt (Abbildung 9). Anhand diesem wurde die Nutzung von folgenden Diensten festgestellt: Simple Storage Service (S3)-Bucket, EC2, CloudTrail und RDS.

```
1 aws resourcegroupstaggingapi get-resources | tee aws-resources.json | jq
  ↪ '.[] | .[] |.ResourceARN'
```

Listing 10: AWS Wordpress Szenario: Erfassung: Ressourcen auflisten

[1] https://console.aws.amazon.com/resource-groups/tag-editor/find-resources



Resource search results (14)

Export 14 resources to CSV | Manage tags of selected resources

Choose up to 500 resources for which you want to edit tags.

Filter resources

Identifier	Tag: Name	Service	Type	Region	Tags
aws-cloudtrail-logs-73513931...	(not tagged)	S3	Bucket	us-east-1	-
i-0354b4055dd773828	Wordpress-vm	EC2	Instance	us-east-1	1
sg-031ddd7fff436ec0a	(not tagged)	EC2	SecurityGroup	us-east-1	-
sg-079e21a1df21ee8c3	(not tagged)	EC2	SecurityGroup	us-east-1	-
rtb-0789f760cd381e4e0	(not tagged)	EC2	RouteTable	us-east-1	-
vol-0135cd263869f1cd5	(not tagged)	EC2	Volume	us-east-1	-
vpc-024f68210ff4ee123	Wordpress-vpc	EC2	VPC	us-east-1	1
subnet-0ad37b05ff160a233	Wordpress-privat-subnet	EC2	Subnet	us-east-1	1
subnet-0ab32bde86f755b1a	Wordpress-public-subnet	EC2	Subnet	us-east-1	1
igw-0515ea6427662bcac0	(not tagged)	EC2	InternetGateway	us-east-1	-
management-events	(not tagged)	CloudTrail	Trail	us-east-1	-
wordpress-sql-db	(not tagged)	RDS	DBInstance	us-east-1	-
default-vpc-05139865d97d77...	(not tagged)	RDS	DBSubnetGroup	us-east-1	-
wordpress-privat-subnet	(not tagged)	RDS	DBSubnetGroup	us-east-1	-

Abbildung 9: AWS Wordpress Szenario: Ressourcenübersicht

Das virtuelle Netzwerk und seine Bestandteile werden zuerst erfasst (Listing 11). Das virtuelle Netzwerk (1) und die Subnetze werden erfasst (2). Anhand der verbliebenen IP-Adressen (siehe Abbildung 10) kann berechnet werden, wie viele Adressen vergeben sind. Die Subnetzmaske `\\24` gibt an, dass insgesamt 256 Adressen im jeweiligen Subnetz zur Verfügung stehen, davon sind jedoch 5 Adressen reserviert[12]:

- 10.10.xxx.0 Netzwerkadresse
- 10.10.xxx.1 Reserviert durch AWS für VPC Router
- 10.10.xxx.2 Reserviert durch AWS für DNS Server
- 10.10.xxx.3 Reserviert durch AWS für zukünftige Nutzung
- 10.10.xxx.255 Broadcastadresse

Daraus ergibt sich, dass im `Wordpress-public-subnet` und `Wordpress-privat-subnet` jeweils eine Adresse vergeben ist und im `Wordpress-privat-subnet2` keine. Weiterhin werden die Routing-Tabellen (3), Internetgateways (4) und Security-Groups (5) erfasst.

```
1 aws ec2 describe-vpcs > aws_ec2_describe-vpcs.json
2 aws ec2 describe-subnets | tee aws_ec2_describe-subnets.json | jq
  ↪  '.Subnets[] | (.Tags[].Value, .CidrBlock, .AvailableIpAddressCount)'
3 aws ec2 describe-route-tables > aws_ec2_describe-route-tables.json
4 aws ec2 describe-internet-gateways > aws_ec2_describe-internet-gateways.json
5 aws ec2 describe-security-groups > aws_ec2_describe-security-groups.json
```

Listing 11: AWS Wordpress Szenario: Erfassung: virtuelles Netzwerk

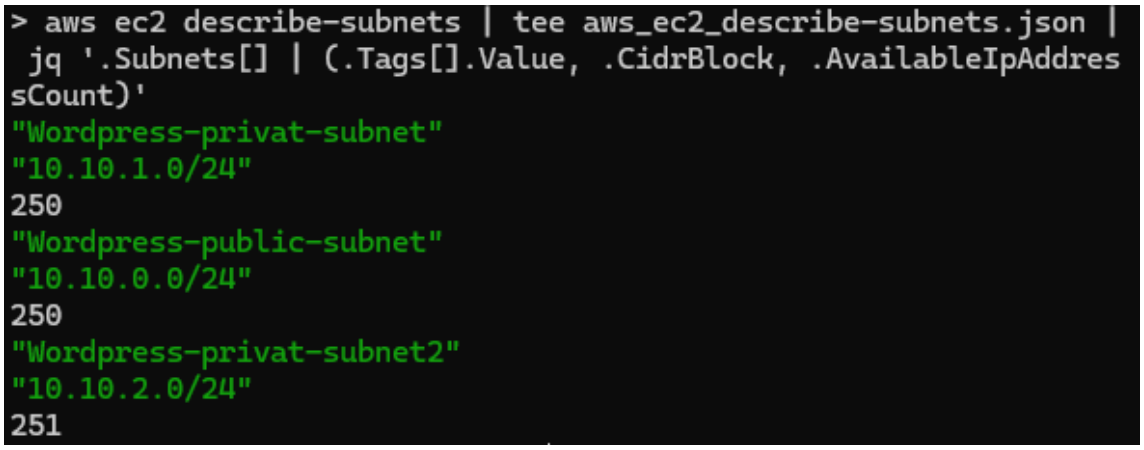


Abbildung 10: AWS Wordpress Szenario: Subnetz Übersicht

Im nächsten Schritt wird die Virtuelle Maschine (*EC2-Instanz*) erfasst (Listing 12). Die Metadaten der virtuellen Maschine werden gesichert und der Name der Maschine ausgegeben (1). In einem weiteren Schritt werden die eingebundenen virtuellen Festplatten festgestellt (2). In Vorbereitung auf die Sicherung der virtuellen Maschine wird von der eingebundenen virtuellen Festplatte ein Snapshot mit Namen `ForensicSnapshot` erstellt (3). Den Fortschritt des Snapshots wird geprüft (4) und nach Abschluss wird die SnapshotId `snap-0024bb064ea67be70` notiert. Dieser Snapshot wird später bei der Sicherung mithilfe einer forensischen VM gesichert. Dieser Umweg ist notwendig, da ein Snapshot zwar zur Wiederherstellung genutzt, jedoch nicht direkt aus AWS exportiert werden kann.

```
1 aws ec2 describe-instances | tee aws_ec2_describe-instances.json | jq
  ↪ '.[].[].Instances[].Tags[].Value'
2 jq '.Reservations[].Instances[].BlockDeviceMappings[].Ebs.VolumeId'
  ↪ aws_ec2_describe-instances.json
3
4 aws ec2 create-snapshot --volume-id vol-068d5b97246a42caa --description
  ↪ "ForensicSnapshot" --tag-specifications
  ↪ 'ResourceType=snapshot,Tags=[{Key=Name,Value=ForensicSnapshot}]'
5 aws ec2 describe-snapshots --filters Name=tag:Name,Values=ForensicSnapshot
```

Listing 12: AWS Wordpress Szenario: Erfassung: virtuelle Maschine

Die Erfassung wird mit der Datenbank-Instanz abgeschlossen (Listing 13). Diese Übersicht enthält Metadaten zur Datenbank-Instanz, jedoch keine Informationen über Datenbanken, Tabellen oder Einträge.

```
1 aws rds describe-db-instances > aws_rds_describe-db-instances.json
```

Listing 13: AWS Wordpress Szenario: Erfassung: Datenbank

Sicherung Im ersten Teil des Abschnitts Sicherung wird die virtuelle Maschine gesichert. Zuerst wird mithilfe einer forensischen VM die virtuelle Festplatte gesichert und anschließend der Arbeitsspeicher über Live-Forensik Maßnahmen. Während der Erfassung wurde von der virtuellen Festplatte ein Snapshot erstellt. Dieser Snapshot wird bei Erstellung (Listing 14) einer forensischen VM (1) als weitere Festplatte eingebunden. Als Ziel für das forensische Image wird ein Objektspeicher erstellt (3). Damit die forensische Maschine auf den Objektspeicher zugreifen kann, muss zuerst eine Berechtigung erfolgen. Dies geschieht, indem eine neue Rolle `ForensicS3Access` mit einer von AWS vordefinierten Policy erstellt wird (5). Aus dieser wird ein InstanceProfile erstellt (6) und der forensischen VM zugewiesen (7). Alternativ kann, wie im Szenario von Google Cloud: Wordpress, der Objektspeicher mittels pre-signed URLs beschreibbar gemacht werden. Aufgrund der vorher gesichteten Sicherheitsgruppe empfiehlt sich eine Erstellung der forensischen VM in dieser, sowie im identischen Subnetz. Alternativ muss für die forensische VM ein eigenes Netz aufgebaut werden.

```
aws ec2 run-instances --image-id ami-0a3c3a20c09d6f377  --instance-type
  ↪ t2.micro --subnet-id subnet-0ab32bde87c720b1b
  ↪ --associate-public-ip-address  --security-group-ids sg-031ddd7fff971ec0a
  ↪ --region us-east-1 --tag-specifications
  ↪ 'ResourceType=instance,Tags=[{Key=Name,Value=ForensicVM}]'
  ↪ --block-device-mappings
  ↪ "[{\"DeviceName\":\"/dev/sdf\",\"Ebs\":{\"SnapshotId\":\"snap-
  ↪ 0024bb064ea67be70\"}}]"

aws s3api create-bucket --bucket wordpress-forensic-bucket --region
  ↪ us-east-1

aws iam attach-role-policy --role-name ForensicS3Access --policy-arn
  ↪ "arn:aws:iam::aws:policy/AmazonS3FullAccess"
aws iam create-instance-profile --instance-profile-name
  ↪ s3AccessInstanceProfile
aws ec2 associate-iam-instance-profile --instance-id i-05130058693c956a8
  ↪ --iam-instance-profile Name=s3AccessInstanceProfile
```

Listing 14: AWS Wordpress Szenario: Sicherung: Vorbereitung

Als nächstes kann die virtuelle Festplatte gesichert werden (Listing 15). Dazu wird die InstanceId der forensischen VM angezeigt (1) um sich im nächsten Schritt über SSH auf diese zu verbinden (2). Dort wird ein Hashwert der Festplatte erstellt (4), diesen Hashwert in den Objektspeicher kopiert (5) und anschließend die Festplatte direkt in den Objektspeicher kopiert. Abschließend kann die SSH-Verbindung getrennt werden.

```
aws ec2 describe-instances --query
  ↪ 'Reservations[*].Instances[*].{Instance:InstanceId}' --filters
  ↪ Name=tag:Name,Values=ForensicVM --output json
aws ec2-instance-connect ssh --instance-id i-05130058693c956a8

sudo sha256sum /dev/sdf > sdf.bin.sha256
aws s3 cp sdf.bin.sha256
  ↪ s3://wordpress-forensic-bucket/images/sdf.bin.sha256
sudo dd if=/dev/sdf bs=5M status=progress | aws s3 cp -
  ↪ s3://wordpress-forensic-bucket/images/sdf.bin
```

Listing 15: AWS Wordpress Szenario: Sicherung: virtuelle Festplatte

Im Anschluss wird der Arbeitsspeicher der virtuellen Maschine gesichert (Listing

16). Der virtuellen Maschine wird dasselbe InstanceProfile zugeordnet wie der forensischen VM um Zugriff auf den Objektspeicher zu gewähren (1). Darauffolgend wird eine SSH-Verbindung aufgebaut (2). Als Tool wurde AVML[2] (Acquire Volatile Memory for Linux) gewählt. Dies wird heruntergeladen (4), ausführbar gemacht (5) und ausgeführt (6). Vom erstellten Speicherabbild wird ein Hashwert erstellt (7) und beide Dateien in den Objektspeicher kopiert (8, 9).

```
1 aws ec2 associate-iam-instance-profile --instance-id i-0354b4055dd186553
  ↪ --iam-instance-profile Name=s3AccessInstanceProfile
2 aws ec2-instance-connect ssh --instance-id i-0354b4055dd186553
3
4 wget https://github.com/microsoft/avml/releases/download/v0.13.0/avml
5 chmod +x avml
6 sudo ./avml --compress ./wordpressvm-ram.bin
7 sudo sha256sum wordpressvm-ram.bin > wordpressvm-ram.bin.sha256
8 aws s3 cp wordpressvm-ram.bin
  ↪ s3://wordpress-forensic-bucket/ram/wordpressvm-ram.bin
9 aws s3 cp wordpressvm-ram.bin.sha256
  ↪ s3://wordpress-forensic-bucket/ram/wordpressvm-ram.bin.sha256
```

Listing 16: AWS Wordpress Szenario: Sicherung: Arbeitsspeicher

Darauf folgt die Sicherung der Datenbank (Listing 17). Für die Sicherung ist entweder das sogenannte Master-Password notwendig oder es kann auf ein neues Passwort zurückgesetzt werden (1). Das Master-password wird bei der Erstellung der Datenbank vergeben. Anschließend wird die forensische VM genutzt um einen Datenbankexport zu erstellen. Es wird eine SSH-Verbindung aufgebaut und die passende Client-Software installiert (4). Daraufhin wird ein Datenbankexport erstellt (5). Die Adresse der Datenbank kann aus der erfassten Übersicht entnommen werden (2). Abschließend wird vom Export ein Hashwert berechnet und beide Dateien in den Objektspeicher kopiert.

[2]https://github.com/microsoft/avml

```
1 aws rds modify-db-instance --db-instance-identifier wordpress-sql-db
  ↪ --master-user-password newpassword
2 jq '.[].[].Endpoint.Address' aws_rds_describe-db-instances.json
3
4 sudo dnf install mariadb105
5 mysqldump --single-transaction -h
  ↪ wordpress-sql-db.cxwq8coomx2p.us-east-1.rds.amazonaws.com -u root -p
  ↪ --all-databases > wordpress-db.sql
6 sha256sum wordpress-db.sql > wordpress-db.sql.sha256
7 aws s3 cp wordpress-db.sql
  ↪ s3://wordpress-forensic-bucket/database/wordpress-db.sql
8 aws s3 cp wordpress-db.sql.sha512
  ↪ s3://wordpress-forensic-bucket/database/wordpress-db.sql.sha256
```

Listing 17: AWS Wordpress Szenario: Sicherung: Datenbank

Extraktion Die Extraktion schließt diese Phase ab (Listing 18). Die gesicherten Daten werden aus dem Objektspeicher kopiert (1). Abschließend wird die Übertragung mit einem Hashwertabgleich geprüft (3,4,5).

```
1 aws s3 sync s3://wordpress-forensic-bucket ./s3/
2
3 sha256sum -c sdf.bin.sha256
4 sha256sum -c wordpressvm-ram.bin.sha256
5 sha256sum -c wordpress-db.sql.sha256
```

Listing 18: AWS Wordpress Szenario: Extraktion

4.3.2 Azure: Wordpress

Beim CSP Microsoft Azure wird eine virtuelle Maschine (*VM*) und eine von Microsoft verwaltete relationale Datenbank (*Azure Database for MySQL*) verwendet. Die virtuelle Maschine (Abbildung 11: 1) befindet sich im Subnetz PublicSubnet des virtuellen Netzwerks Wordpress-VPC. Eine Netzwerkschnittstelle (*Network Interface Card (NIC)*) (2) ermöglicht Netzwerkverbindungen innerhalb des virtuellen Netzwerkes sowie nach und von außerhalb. Mit der Netzwerkschnittstelle ist eine Netzwerksicherheitsgruppe (*Network Security Group (NSG)*) (3) verknüpft. Aufgrund dieser NSG werden jegliche ausgehende Verbindungen zugelassen, eingehende Verbindungen jedoch auf Port 22 und 80 beschränkt. An der Netzwerkschnittstelle ist zusätzlich eine öffentliche IP-Adresse (*PublicIP*) verknüpft. Über diese kann die virtuelle Maschine aus dem Internet aus angesprochen werden. Das zweite Subnetz PrivateSubnet wird an den Datenbankserver (*Azure Database for MySQL - Flexible Server*) (4) delegiert. Das Subnetz kann durch die Delegierung nur noch durch Datenbankserver genutzt werden [16] und Netzwerkverbindungen werden nur innerhalb des virtuellen Netzwerks zugelassen.

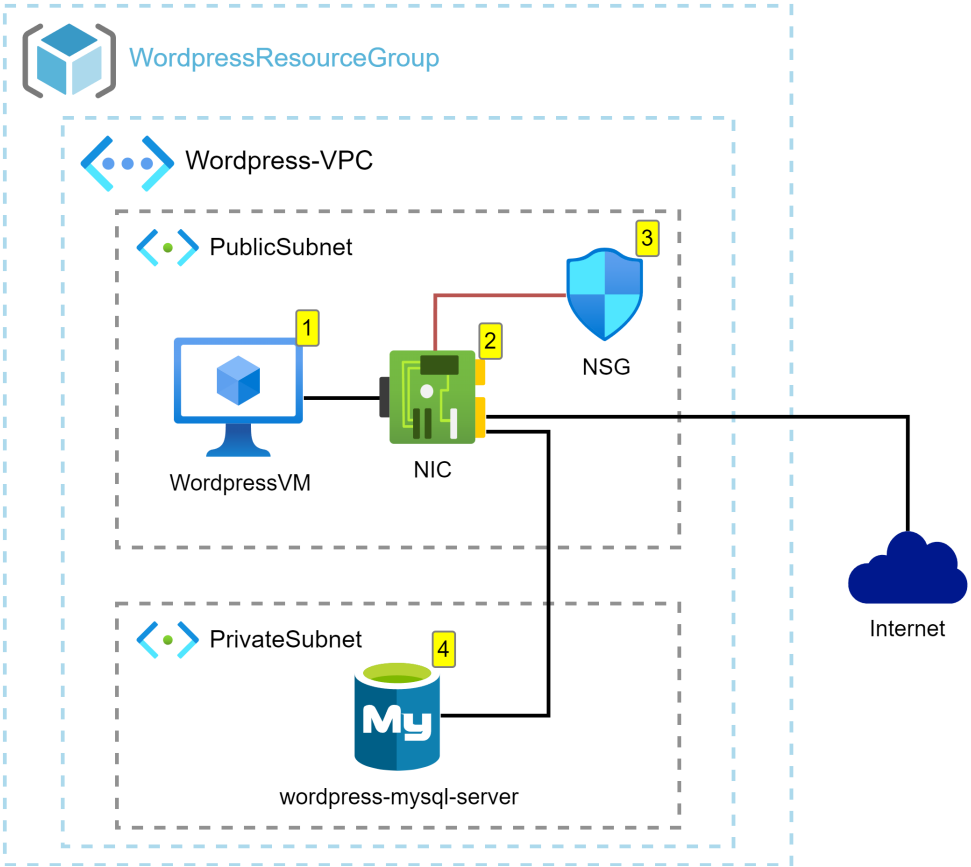


Abbildung 11: Azure Wordpress Szenario

Aufbau

Zu Beginn (Listing 19) wird eine Ressourcengruppe (*ResourceGroup*) erstellt (1). In dieser Ressourcengruppe sind alle weiteren Ressourcen gebündelt. Als nächstes wird ein virtuelles Netzwerk inkl. PublicSubnet erstellt (2). Zusätzlich wird noch das PrivateSubnet erstellt (3). Der Datenbankserver benötigt eine Privat-DNS Zone. Diese wird hier schon eingerichtet (4).

```
1 az group create --name WordpressResourceGroup --location eastus
2 az network vnet create --resource-group WordpressResourceGroup --name
  ↪  Wordpress-VPC --address-prefix 10.10.0.0/16 --subnet-name PublicSubnet
  ↪  --subnet-prefixes 10.10.0.0/24
3 az network vnet subnet create --resource-group WordpressResourceGroup
  ↪  --vnet-name Wordpress-VPC --name PrivateSubnet --address-prefixes
  ↪  10.10.1.0/24
4 az network private-dns zone create --resource-group WordpressResourceGroup
  ↪  --name dns.private.mysql.database.azure.com
```

Listing 19: Azure Wordpress Szenario: Ressourcengruppe & VPN Einrichtung

In diesem Abschnitt (Listing 20) wird die virtuelle Maschine erstellt (1). Bei der Erstellung werden auch ssh-keys (kryptographische Schlüssel) generiert und auf dem aktuellen Client-System abgelegt. Diese werden später für eine SSH-Verbindung verwendet. Automatisiert wird zeitgleich noch das Netzwerkinterface und die Netzwerksicherheitsgruppe erstellt. Für letzteres wird der Name abgefragt (2) um eine neue Regel zu erstellen (3). Diese Regel erlaube eingehende Netzwerkverbindungen zu Port 80. Die öffentliche IP-Adresse der virtuellen Maschine wird abgefragt (4) und eine SSH-Verbindung wird getestet (5).

```
1 az vm create -n WordpressVM -g WordpressResourceGroup --vnet-name
  ↪ Wordpress-VPC --subnet PublicSubnet --image Ubuntu2204 --size
  ↪ Standard_B1s --admin-username azureuser --generate-ssh-keys
2 az network nsg list | jq '.[].name'
3 az network nsg rule create -g WordpressResourceGroup --nsg-name
  ↪ WordpressVMNSG --name AllowHTTPTraffic --priority 4096
  ↪ --source-address-prefixes '*' --destination-port-ranges 80 --access
  ↪ Allow --protocol Tcp --direction Inbound --description "Allow from all
  ↪ on Port 80 "
4 az vm show --show-details -g WordpressResourceGroup --name WordpressVM
  ↪ --query publicIps
5 ssh -i ~/.ssh/id_rsa azureuser@13.14.15.16
```

Listing 20: Azure Wordpress Szenario: VM & NSG Einrichtung

Darauffolgend (Listing 21) wird der Datenbankserver erstellt (1). Durch die Angabe des Parameter `--subnet PrivateSubnet` wird dieses Subnetz an den Datenbankserver delegiert. Der Datenbankserver greift zusätzlich auf die DNS-Zone zurück. Durch eine Abfrage (2) wird der Domainname des Datenbankservers bekannt. Dieser kann innerhalb des virtuellen Netzwerks genutzt werden. Um eine unverschlüsselte Verbindung zuzulassen, muss noch der Parameter `require_secure_transport` auf `OFF` gesetzt werden (3). Alternativ[15] kann ein Zertifikat heruntergeladen und in die virtuelle Maschine eingepflegt werden. Dabei sind jedoch noch zusätzliche Anpassungen der Wordpress-Anwendung notwendig.

```
1 az mysql flexible-server create --resource-group WordpressResourceGroup
  ↪ --name wordpress-mysql-server --admin-user wordpressdbadmin
  ↪ --admin-password securepassword --sku-name Standard_B1s --storage-size
  ↪ 20 --vnet Wordpress-VPC --subnet PrivateSubnet --private-dns-zone
  ↪ dns.private.mysql.database.azure.com
2 az mysql flexible-server show --resource-group WordpressResourceGroup --name
  ↪ wordpress-mysql-server --query fullyQualifiedDomainName
3 az mysql flexible-server parameter set --resource-group
  ↪ WordpressResourceGroup --server-name wordpress-mysql-server --name
  ↪ require_secure_transport --value OFF
```

Listing 21: Azure Wordpress Szenario: Datenbankserver erstellen

Im Anschluss (Listing 22) wird über die virtuelle Maschine die Datenbank für Wordpress vorbereitet. Dazu wird eine SSH-Verbindung zur virtuellen Maschine (1) hergestellt, den mysql-client installiert (2) und eine kaskadierende Verbindung zur Da-

tenbank hergestellt(3). In dieser wird ein Benutzer (5) und Datenbank (6) erstellt. Danach werden noch Berechtigungen gesetzt (7,8)

```
1 ssh -i ~/.ssh/id_rsa azureuser@13.14.15.16
2 sudo apt install mysql-client-core-8.0
3 mysql flexibleserverdb --host
  ↪ wordpress-mysql-server.mysql.database.azure.com --user wordpressdbadmin
  ↪ --password=securepassword
4
5 CREATE USER 'wordpress-user'@'%' IDENTIFIED BY 'securepassword';
6 CREATE DATABASE `wordpress-db`;
7 GRANT ALL PRIVILEGES ON `wordpress-db`.* TO "wordpress-user"@"%";
8 FLUSH PRIVILEGES;
9 exit
```

Listing 22: Azure Wordpress Szenario: Datenbankeinrichtung

Abschließend (Listing 23) wird noch Wordpress installiert und initialisiert. Dazu wird auf der VM der Apache HTTP Webserver und PHP installiert (1). Weiter wird die aktuellste Wordpress Version heruntergeladen (2), entpackt (3) und konfiguriert (4,5). In der Konfiguration müssen noch die genannten Felder (7-10) angepasst werden. Schließlich wird noch Wordpress in das Webserver-Verzeichnis kopiert (12) und die Standard Apache2 Landing-Page gelöscht (14). Die Wordpress Anwendung kann nun über die zuvor festgestellte öffentliche IP-Adresse aufgerufen, eingerichtet und verwaltet werden. Der Aufbau ist hiermit abgeschlossen.

```
1 sudo apt install apache2 php libapache2-mod-php php-mysql
2 wget https://wordpress.org/latest.tar.gz
3 tar -xzf latest.tar.gz
4 cp wordpress/wp-config-sample.php wordpress/wp-config.php
5 nano wordpress/wp-config.php
6
7 define( 'DB_NAME', 'wordpress-db' );
8 define( 'DB_USER', 'wordpress-user' );
9 define( 'DB_PASSWORD', 'securepassword' );
10 define( 'DB_HOST', 'wordpress-mysql-server.mysql.database.azure.com' );
11
12 cp -r wordpress/* /var/www/html/
13 rm /var/www/html/index.html
```

Listing 23: Azure Wordpress Szenario: Installation

Durchführung Leitfaden

Orientierung Im Abschnitt Orientierung wird der eigene Zugang auf Berechtigungen geprüft (Listing 24). Beginnend mit dem aktuell eingeloggten Nutzer (1) wird dessen ID notiert. Weiter wird einer Übersicht des eingeloggten Accounts (2) und ein Übersicht über alle Nutzer gesichert (3). Mit den nächsten zwei Aufrufe wird die Role zur vorher notierten ID festgestellt (4) und die Berechtigungen dieser Rolle überprüft.

```
1 az ad signed-in-user show | tee az_ad_signed-in-user_show.json | jq '.id'
2 az account list > az_account_list.json
3 az ad user list > az_ad_user_list.json
4 az role assignment list --assignee 793f0893-3d0b-4bfb-9936-f2cf45d181a4 >
  ↪  az_role_assignment_list.json
5 az role definition list --name Owner > az_role_definition_list.json
```

Listing 24: Azure Wordpress Szenario: Orientierung: Eigener Zugang prüfen

Als nächstes werden die API-Logs gesichert (Listing 25). Der Aufbewahrungszeitraum beträgt standardmäßig 90 Tage. Eine Abfrage liefert standardmäßig 50 Einträge, die getätigte Abfrage (1) liefert bis zu 10000 Einträge. Die exportierten API-Logs (*Activity-Logs*) enthalten ca. 2000 Einträge (2). Wäre die Anzahl an Einträgen ausgeschöpft, so müsste eine erneute Abfrage mit entweder mehr Einträge oder angepasster Start- und Endzeit getätigt werden. Um einen Überblick zu verschaffen werden die letzten 10 Log-Einträge gesichtet (3).

```
1 az monitor activity-log list --offset 90d --max-events 10000 >
  ↪  az_monitor_activity-log_list.json
2 jq length az_monitor_activity-log_list.json
3 jq '.[] | .properties.message' az_monitor_activity-log_list.json | head
```

Listing 25: Azure Wordpress Szenario: Orientierung: Logs exportieren

Erfassung Zu Beginn der Erfassung (Listing 26) werden die Ressourcen-Gruppen aufgelistet (1). Anschließend werden die entsprechenden Ressourcen aufgelistet (Abbildung 12) und gesichtet (2).

```
az group list
az resource list --resource-group WordpressResourceGroup | tee
  ↪ az_resource_list_--resource-group.json | jq '.[] | .type'
```

Listing 26: Azure Wordpress Szenario: Erfassung: Ressourcen auflisten

```
> jq '.[] | .type' az_resource_list_--resource-group.json
"Microsoft.Compute/disks"
"Microsoft.Compute/virtualMachines"
"Microsoft.DBforMySQL/flexibleServers"
"Microsoft.Network/networkInterfaces"
"Microsoft.Network/networkSecurityGroups"
"Microsoft.Network/privateDnsZones"
"Microsoft.Network/privateDnsZones/virtualNetworkLinks"
"Microsoft.Network/publicIPAddresses"
"Microsoft.Network/virtualNetworks"
```

Abbildung 12: Azure Wordpress Szenario: Ressourcenübersicht

Wie der Ressourcenübersicht zu entnehmen ist sind mehrere Netzwerkressourcen vorhanden. Diese werden als nächstes erfasst (Listing 27). Die virtuelle Netzwerke werden erfasst und Name, Subnetze und IP-Ranges gesichtet (1) (Abbildung 13). Weiter werden Netzwerkinterfaces (2), externe IP-Adressen (3) und Netzwerksicherheitsgruppen (4) erfasst. Abgeschlossen wird die Netzwerkerfassung der Erfassung der DNS-Zone (5) und deren Einträge (6).

```
az network vnet list -g WordpressResourceGroup | tee
  ↪ az_network_vnet_list.json | jq '.[] | .name, (.subnets[] |
  ↪ .name,.addressPrefix)'
az network nic list -g WordpressResourceGroup > az_network_nic_list.json
az network public-ip list -g WordpressResourceGroup >
  ↪ az_network_public-ip_list.json
az network nsg list -g WordpressResourceGroup > az_network_nsg_list.json
az network private-dns zone list -g WordpressResourceGroup | tee
  ↪ az_network_private-dns_zone_list.json | jq '.[].name'
az network private-dns record-set list -g WordpressResourceGroup -z
  ↪ dns.private.mysql.database.azure.com >
  ↪ az_network_privat-dns_record-set_list.json
```

Listing 27: Azure Wordpress Szenario: Erfassung: virtuelles Netzwerk

```
> jq '.[] | .name, (.subnets[] | .name,.addressPrefix)' az_network_vnet_list.json
"Wordpress-VPC"
"PublicSubnet"
"10.10.0.0/24"
"PrivateSubnet"
"10.10.1.0/24"
```

Abbildung 13: Azure Wordpress Szenario: Netzwerkübersicht

Die Erfassung wird mit der virtuellen Maschine fortgesetzt (Listing 28). Alle virtuelle Maschinen werden erfasst (1). Darauffolgend werden die IP-Adressen, sowohl externe als auch interne, erfasst (2). Zur festgestellten VM wird die *Instance-View* erfasst (3). Diese enthält zusätzliche Informationen wie virtuelle Festplatten und ob die VM ausgeführt wird. Zur Vorbereitung der Sicherung wird der eindeutige Bezeichner (URI) der virtuellen Festplatte notiert (5).

```
1 az vm list --show-details > az_vm_list.json
2 az vm list-ip-addresses > az_vm_list-ip-addresses.json
3 az vm get-instance-view -g WordpressResourceGroup --name WordpressVM >
  ↪  az_vm_get-instance-view.json
4
5 jq '.storageProfile.osDisk.managedDisk.id' az_vm_get-instance-view.json
```

Listing 28: Azure Wordpress Szenario: Erfassung: virtuelle Maschine

Die Erfassung wird mit der Datenbank beendet (Listing 29). Alle Datenbankserver werden erfasst und deren Name, Domain-Name und Login-Name notiert (1). Die enthaltenen Datenbanken des festgestellten Datenbankserver werden erfasst und aufgelistet (2).

```
1 az mysql flexible-server list -g WordpressResourceGroup | tee
  ↪  az_mysql_flexible-server_list.json | jq '.[] | .name,
  ↪  .fullyQualifiedDomainName , .administratorLogin'
2 az mysql flexible-server db list -g WordpressResourceGroup --server-name
  ↪  wordpress-mysql-server | tee az_mysql_flexible-server_db_list.json | jq
  ↪  '.[].name'
```

Listing 29: Azure Wordpress Szenario: Erfassung: Datenbank

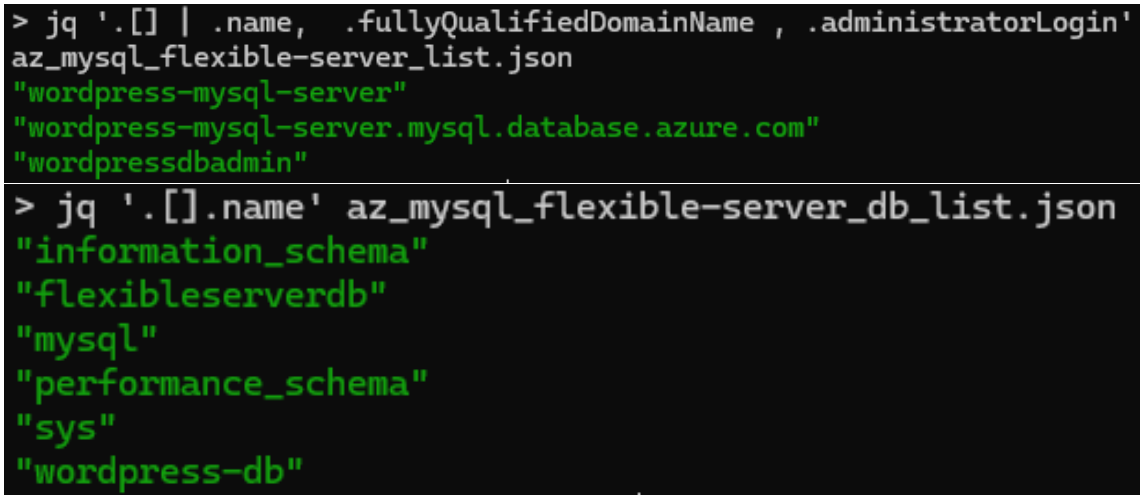


Abbildung 14: Azure Wordpress Szenario: Datenbankübersicht

Sicherung Diese Phase wird mit der Sicherung der virtuellen Festplatte begonnen (Listing 30). Mit der zuvor notierten URI der virtuellen Festplatte wird die Erstellung eines Snapshots angestoßen (1). Der Snapshot kann ohne Umweg durch eine Freigabe (2) heruntergeladen werden.

```
1 az snapshot create -g WordpressResourceGroup -n ForensicSnapshot --source
  ↪ <source-uri>
2 az snapshot grant-access --duration-in-seconds 3600 -g
  ↪ WordpressResourceGroup -n ForensicSnapshot >
  ↪ az_snapshot_grant-access.json
```

Listing 30: Azure Wordpress Szenario: Sicherung: virtuelle Festplatte

Als nächstes wird die Sicherung des Arbeitsspeicher der VM vorbereitet (Listing 31). Als Ziel des Arbeitsspeichers wird ein neuer Objektspeicher-Account erstellt (1) und dessen Account-Key ausgelesen (2). Mithilfe des Account-Keys wird ein Container innerhalb des Objektspeicher-Accounts erstellt. In diesen Container wird im nächsten Schritt der Arbeitsspeicher kopiert. Um von der virtuellen Maschine aus in den Container schreiben zu können, wird ein sogenanntes Shared Access Signature (SAS)-Token erstellt (5). Dieses Token ist vergleichbar mit einer pre-signed URL und ermöglicht einen Zugriff mit vordefinierten Berechtigungen. Hier wurden die Berechtigungen `--permissions aw` gewählt für: **a**dd und **w**rite. Die vollständige URL lässt sich anhand Objektspeicher-Account, Containername und SAS Token zusammensetzen: `https://<accountname>.blob.core.windows.net/<containername>/<dateiname>?<SASToken>` . Die letzte Vorbereitung besteht darin, einen foren-

sischen SSH-Schlüssel auf der virtuellen Maschine zu hinterlegen, falls der ursprüngliche SSH-Schlüssel nicht zur Verfügung steht. Dazu wird ein neuer SSH-Schlüssel erzeugt (7) und mit dem Nutzername `forensic` auf der VM hinterlegt (8).

```
1 az storage account create --name forensicstorage -g WordpressResourceGroup
2 az storage account keys list -g WordpressResourceGroup -n forensicstorage |
  ↪  jq '.[0].value'
3 az storage container create --name forensiccontainer --account-name
  ↪  forensicstorage --account-key <account-key>
4
5 az storage container generate-sas --name forensiccontainer --account-name
  ↪  forensicstorage --account-key <account-key> --full-uri --permissions aw
  ↪  --expiry 2024-04-01 > az_storage_container_generate-sas.json
6
7 ssh-keygen -f ./Forensic.key -P ""
8 az vm user update --resource-group WordpressResourceGroup --name WordpressVM
  ↪  --username forensic --ssh-key-value ./Forensic.key.pub
```

Listing 31: Azure Wordpress Szenario: Sicherung: Arbeitsspeicher Vorbereitung

Folgend wird der Arbeitsspeicher der virtuellen Maschine gesichert (Listing 32). Bevor eine SSH-Verbindung aufgebaut werden kann, muss die Netzwerksicherheitsgruppe der virtuellen Maschine geprüft werden, ob eine SSH-Verbindung möglich ist. Anschließend kann eine SSH-Verbindung unter Angabe des zuvor erstellten SSH-Schlüssels aufgebaut werden (1). Zur Sicherung wird AVML (Acquire Volatile Memory for Linux) genutzt. Dies wird heruntergeladen (3), ausführbar gemacht (4) und ausgeführt (5). Vom erstellten Speicherabbild wird ein Hashwert erstellt (6) und beide Dateien in den Objektspeicher kopiert (8, 9).

```
az ssh vm --resource-group WordpressResourceGroup --vm-name WordpressVM
  ↪ --local-user forensic --certificate-file ./Forensic.key.pub
  ↪ --private-key-file ./Forensic.key

wget https://github.com/microsoft/avml/releases/download/v0.13.0/avml
chmod +x avml
sudo ./avml --compress ./wordpressvm-ram.bin
sha256sum wordpressvm-ram.bin > wordpressvm-ram.bin.sha256

curl -H "x-ms-blob-type: BlockBlob" --upload-file wordpressvm-ram.bin --url
  ↪ "https://forensicstorage.blob.core.windows.net/forensiccontainer/
  ↪ ram/wordpressvm-ram.bin?<SASToken>"
curl -H "x-ms-blob-type: BlockBlob" --upload-file wordpressvm-ram.bin.sha256
  ↪ --url "https://forensicstorage.blob.core.windows.net/forensiccontainer/
  ↪ ram/wordpressvm-ram.bin.sha256?<SASToken>"
```

Listing 32: Azure Wordpress Szenario: Sicherung: Arbeitsspeicher

Diese Phase wird mit der Sicherung der Datenbank abgeschlossen (Listing 33). Mithilfe von `az mysql flexible-server backup create` kann eine Backup-Erstellung angestoßen werden. Dieses Backup kann jedoch nur zur Wiederherstellung genutzt werden. Eine Export-Funktionalität ermöglicht den Export des Datenbankservers (1). Dazu wird erneut das zuvor generierte SAS-Token genutzt.

```
az mysql flexible-server export create -g WordpressResourceGroup -n
  ↪ wordpress-mysql-server -b forensicbackup -u
  ↪ "https://forensicstorage.blob.core.windows.net/forensiccontainer/db-
  ↪ export/?<SASToken>"
```

Listing 33: Azure Wordpress Szenario: Sicherung: Datenbank

Extraktion Die virtuelle Festplatte kann über die zuvor generierte Freigabe in Form einer URL heruntergeladen werden. Die Sicherung des Arbeitsspeichers und Export des Datenbank-Servers werden aus dem Objektspeicher heruntergeladen (Listing 34) (1). Die Übertragung des Arbeitsspeichers wird mit einem Hashwertabgleich überprüft (4). Die Übertragung des Exports kann mithilfe des Container-Inhaltsverzeichnis überprüft werden (2). Diese beinhaltet unter anderem auch den MD5-Hashwert jeder Datei. Der Hashwert wird jedoch als Base64-codierte Repräsentation des binären MD5-Hashwert[34] abgespeichert. Programme wie `md5sum` oder `sha256sum` geben jedoch die hexadezimale Repräsentation wieder. Vor einem Hashwert-Abgleich müssen die Repräsentation angepasst werden. Eine Anleitung

zur Umwandlung kann dem Blogeintrag von Ben Reader (Powers-Hell[3]) entnommen werden.

```
1 az storage blob download-batch -s forensiccontainer --account-name
  ↪  forensicstorage --account-key <accountkey> -d "./extraction"
2 az storage blob list --container-name forensiccontainer --account-name
  ↪  forensicstorage --account-key <accountkey> >
  ↪  az_storage_blob_list-forensiccontainer.json
3
4 sha256sum -c wordpressvm-ram.bin.sha256
```

Listing 34: Azure Wordpress Szenario: Extraktion

[3]https://powers-hell.com/2021/12/31/calculate-validate-md5-hashes-on-azure-blob-storage-files-with-powershell/

4.3.3 Google Cloud: Wordpress

Beim CSP Google Cloud wird ebenfalls eine virtuelle Maschine (*Compute Engine*) und eine von Google verwaltete relationale Datenbank (*Cloud SQL*) verwendet. Die virtuelle Maschine (Abbildung 15: 1) befindet sich in einem Subnetz in der Region us-east-1 in einem virtuellen Netzwerk (*Virtual Privat Cloud*). Anhand von *Cloud Firewall Rules* können Benutzer von außerhalb auf die VM zugreifen. Die Datenbank (2) befindet sich außerhalb des VPC. Durch einen sogenannten *Google Privat Path* (3) wird der Datenbank eine IP-Adresse aus dem VPC zugeteilt. Dadurch kann nur von innerhalb des VPC auf die Datenbank zugegriffen werden. Von Google wird empfohlen [22] die Datenbank-Instanz in derselben Region wie die VM zu platzieren. Der Unterschied zu AWS und Azure besteht darin, dass die Datenbank sich nicht in einem abgegrenzten Subnetz befindet, sondern durch ein Google Privat Path im Netzwerk der VM erreichbar ist.

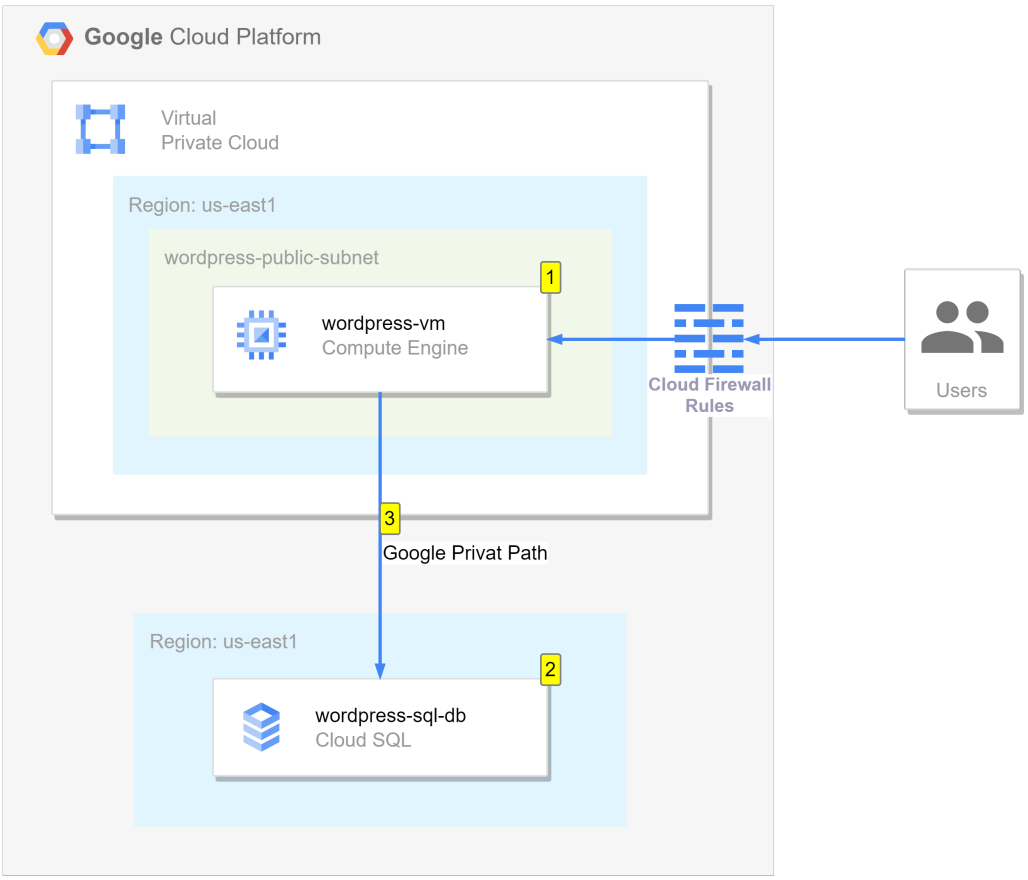


Abbildung 15: Google Cloud Wordpress Szenario

Aufbau

Als Voraussetzung (Listing 35) wird ein neues Projekt wordpress-project erstellt(1), das Rechnungskonto notiert (2) und dieses mit dem Projekt verbunden (3). Zusätzlich müssen noch die zu nutzenden Services aktiviert werden (4,5,6).

```
1 gcloud projects create wordpress-project --name="Wordpress Project"
2 gcloud billing accounts list --filter=open=true
3 gcloud billing projects link wordpress-project
  ↪ --billing-account=012345-6789AB-CDEFGH
4 gcloud services enable compute.googleapis.com --project=wordpress-project
5 gcloud services enable servicenetworking.googleapis.com
  ↪ --project=wordpress-project
6 gcloud services enable sqladmin.googleapis.com --project=wordpress-project
```

Listing 35: Google Cloud Wordpress Szenario: Vorraussetzungen

Als nächstes (Listing 36) wird ein virtuelles Netzwerk wordpress-virtual-network (1) und dazu ein Subnetz (2) mit IP-Bereich 10.0.0.0/24 erstellt. Mithilfe einer Firewallregel (3) kann auf die Ressourcen des Netzwerks wordpress-virtual-network vom Internet aus auf die Ports 22 und 80 zugegriffen werden. Abschließend muss noch ein IP-Bereich für Google Dienste reserviert (4) und diese mit dem virtuellen Netzwerk verbunden (5) werden.

```
1 gcloud compute networks create wordpress-virtual-network
  ↪ --subnet-mode=custom --mtu=1460 --bgp-routing-mode=regional
  ↪ --project=wordpress-project
2 gcloud compute networks subnets create wordpress-public-subnet
  ↪ --range=10.0.0.0/24 --stack-type=IPV4_ONLY
  ↪ --network=wordpress-virtual-network --enable-private-ip-google-access
  ↪ --region=us-east1 --project=wordpress-project
3 gcloud compute firewall-rules create wordpress-virtual-network-allow-custom
  ↪ --network=projects/wordpress-project/global/networks/wordpress-virtual-
  ↪ network --description=ErlaubeEingehendSSHundHTTP --direction=INGRESS
  ↪ --priority=65534 --source-ranges=0.0.0.0/0 --action=ALLOW
  ↪ --rules=tcp:22,80 --project=wordpress-project
4 gcloud compute addresses create wordpress-google-managed-services --global
  ↪ --purpose=VPC_PEERING --prefix-length=16 --description="peering range
  ↪ for Google services" --network=wordpress-virtual-network
  ↪ --project=wordpress-project
5 gcloud services vpc-peerings connect
  ↪ --service=servicenetworking.googleapis.com
  ↪ --ranges=wordpress-google-managed-services
  ↪ --network=wordpress-virtual-network --project=wordpress-project
```

Listing 36: Google Cloud Wordpress Szenario: Netzwerkerstellung

Weiter (Listing 37) wird eine Google Cloud SQL Instanz wordpress-sql-db erstellt (1) und deren IP im virtuellen Netzwerk notiert (2). Darauffolgend wird eine Virtuelle Maschine wordpress-vm instanziiert (3). Die VM ist mit der angezeigten (4) externen IP-Adresse aus dem Internet erreichbar.

```
1 gcloud beta sql instances create wordpress-sql-db
  ↪ --database-version=MYSQL_8_0_31 --tier=db-f1-micro --edition=ENTERPRISE
  ↪ --region=us-east1 --network=wordpress-virtual-network --no-assign-ip
  ↪ --enable-google-private-path --project=wordpress-project
2 gcloud sql instances list --project=wordpress-project
  ↪ --filter=name:wordpress-sql-db --format="value(PRIVATE_ADDRESS)"
3 gcloud compute instances create wordpress-vm --project=wordpress-project
  ↪ --zone=us-east1-b --machine-type=e2-micro --network-interface=network-
  ↪ tier=STANDARD,stack-type=IPV4_ONLY,subnet=wordpress-public-subnet
4 gcloud compute instances describe wordpress-vm --project=wordpress-project
  ↪ --format='get(networkInterfaces[0].accessConfigs[0].natIP)'
```

Listing 37: Google Cloud Wordpress Szenario: SQL & VM Instanz erstellen

Vor der Wordpress-Installation muss die Datenbank noch eingerichtet werden (Lis-

ting 38). Dazu wird eine SSH-Verbindung mit der Virtuellen Maschine (1) hergestellt, den mysql-client installiert (2) und eine kaskadierende Verbindung zur Datenbank hergestellt(3). In dieser wird ein Benutzer (5) und Datenbank (6) erstellt. Danach werden noch Berechtigungen gesetzt (7,8)

```
1  gcloud compute ssh cloud@wordpress-vm --zone=us-east1-b
   ↪  --project=wordpress-project
2  sudo apt install default-mysql-client
3  mysql -u root -h 10.9.0.3 -P 3306 -D mysql
4
5  CREATE USER 'wordpress-user'@'%' IDENTIFIED BY 'securepassword';
6  CREATE DATABASE `wordpress-db`;
7  GRANT ALL PRIVILEGES ON `wordpress-db`.* TO "wordpress-user"@"%";
8  FLUSH PRIVILEGES;
9  exit
```

Listing 38: Google Cloud Wordpress Szenario: Datenbankeinrichtung

Abschließend wird noch Wordpress installiert und initialisiert(Listing 39). Dazu wird auf der VM der Webserver Apache2 (1) und PHP installiert (2). Weiter wird die aktuellste Wordpress Version heruntergeladen (3), entpackt (4) und konfiguriert (5,6). In der Konfiguration müssen noch die genannten Felder (8-11) angepasst werden. Schließlich wird noch Wordpress in das Apache2-Verzeichnis kopiert (13) und die Standard Apache2 Landing-Page gelöscht (14). Die Wordpress Anwendung kann nun über die externe IP-Adresse aufgerufen, eingerichtet und verwaltet werden. Der Aufbau ist hiermit abgeschlossen.

```
1   sudo apt install apache2
2   sudo apt install php libapache2-mod-php php-mysql
3   wget https://wordpress.org/latest.tar.gz
4   tar -xzf latest.tar.gz
5   cp wordpress/wp-config-sample.php wordpress/wp-config.php
6   nano wordpress/wp-config.php
7
8       define( 'DB_NAME', 'wordpress-db' );
9       define( 'DB_USER', 'wordpress-user' );
10      define( 'DB_PASSWORD', 'securepassword' );
11      define( 'DB_HOST', '10.9.0.3' );
12
13  cp -r wordpress/* /var/www/html/
14  rm /var/www/html/index.html
```

Listing 39: Google Cloud Wordpress Szenario: Installation

Durchführung Leitfaden

Im Abschnitt Orientierung wird der eigene Zugang auf Berechtigungen geprüft (Listing 40). Beginnend mit dem aktuell eingeloggten Nutzer (1) werden anschließend alle Projekte aufgelistet (2). Zum Projekt `wordpress-project` werden die berechtigten Accounts gesichtet (3). Die verfügbaren Logs werden angezeigt (5) und die Aufbewahrungszeit wird protokolliert (6). Standardmäßig[24] werden Logs in folgende zwei Objektspeicher abgelegt: `_Required` für unter anderem Administratoraktivitäten und Systemereignisse; `_Default` für unter anderem Datenzugriffe. Ersteres hat dabei eine Aufbewahrungszeit von 400 Tagen und kann nicht konfiguriert oder deaktiviert werden. Letzteres eine Aufbewahrungszeit von 30 Tagen und kann konfiguriert, aber nicht deaktiviert werden. Die `_Default` Logs werden exportiert (7) und darauf die `_Required` Logs (8). Die Logs können grob auf Create oder Insert Ereignisse analysiert werden (9).

```
1 gcloud config list > gcloud_config_list.json
2 gcloud projects list --format=json | tee gcloud_projects_list.json | jq
  ↪ '.[].projectId'
3 gcloud projects get-ancestors-iam-policy wordpress-project --format=json >
  ↪ gcloud_projects_get-ancestors-iam-policy.json
4
5 gcloud logging logs list --project=wordpress-project >
  ↪ gcloud_logging_logs_list.json
6 gcloud logging buckets list --project=wordpress-project >
  ↪ gcloud_logging_buckets_list.json
7 gcloud beta logging read "" --project=wordpress-project --format=json
  ↪ --bucket=_Default --location=global --view=_AllLogs --freshness=30d >
  ↪ gcloud-logs_Default.json
8 gcloud beta logging read "" --project=wordpress-project --format=json
  ↪ --bucket=_Required --location=global --view=_AllLogs --freshness=400d >
  ↪ gcloud-logs_Required.json
9
10 jq '.[] | .protoPayload | .methodName | select(. !=null)'
  ↪ gcloud-logs_Required.json | grep -E "create|insert"
```

Listing 40: Google Cloud Wordpress Szenario: Orientierung

Erfassung Die Erfassung beginnt mit der Auflistung aller Ressourcen (Listing 41). Alle Ressourcen werden erfasst und deren Typ ausgegeben (1). Hierbei sind viele Standard-Ressourcen wie virtuelles Netzwerk mit Subnetze vorhanden. Diese können gefiltert werden um eine übersichtliche Ansicht zu erhalten (2) (Abbildung 16). Unter den Ressourcen sind viele `serviceusage.googleapis.com/Service` Einträge vorhanden. Von diesen Einträgen wir der `displayName` angezeigt (3). Viele dieser Einträge verweisen auf Dienste, die standardmäßig bei jeder Projekterstellung aktiviert werden[21]. Folgende Dienste sind standardmäßig nicht aktiv und wurden aktiviert: `compute.googleapis.com`, `servicenetworking.googleapis.com` und `sqladmin.googleapis.com`.

```
1 gcloud asset search-all-resources --scope=projects/wordpress-project
  ↪ --format=json | tee gcloud_asset_search-all-resources.json | jq '.[] |
  ↪ .assetType'
2 jq '.[] | select(.displayName | contains("default") | not) | .assetType'
  ↪ gcloud_asset_search-all-resources.json
3 jq '.[] | select(.assetType == "serviceusage.googleapis.com/Service") |
  ↪ .displayName' gcloud_asset_search-all-resources.json
```

Listing 41: Google Cloud Wordpress Szenario: Erfassung: Ressourcen

```
> jq '.[] | select(.displayName | contains("default") | not) | .assetType'
gcloud_asset_search-all-resources.json
"serviceusage.googleapis.com/Service"
"serviceusage.googleapis.com/Service"
"serviceusage.googleapis.com/Service"
"serviceusage.googleapis.com/Service"
"serviceusage.googleapis.com/Service"
"iam.googleapis.com/ServiceAccountKey"
"compute.googleapis.com/Firewall"
"compute.googleapis.com/Instance"
"compute.googleapis.com/Disk"
"compute.googleapis.com/Route"
"compute.googleapis.com/Address"
"compute.googleapis.com/Subnetwork"
"compute.googleapis.com/Network"
"compute.googleapis.com/Project"
"cloudresourcemanager.googleapis.com/Project"
"serviceusage.googleapis.com/Service"
"serviceusage.googleapis.com/Service"
"serviceusage.googleapis.com/Service"
"serviceusage.googleapis.com/Service"
"serviceusage.googleapis.com/Service"
"serviceusage.googleapis.com/Service"
"serviceusage.googleapis.com/Service"
"serviceusage.googleapis.com/Service"
"serviceusage.googleapis.com/Service"
"serviceusage.googleapis.com/Service"
"serviceusage.googleapis.com/Service"
"serviceusage.googleapis.com/Service"
"serviceusage.googleapis.com/Service"
"serviceusage.googleapis.com/Service"
"serviceusage.googleapis.com/Service"
"serviceusage.googleapis.com/Service"
"serviceusage.googleapis.com/Service"
"serviceusage.googleapis.com/Service"
"serviceusage.googleapis.com/Service"
"cloudbilling.googleapis.com/ProjectBillingInfo"
```

```
> jq '.[] | select(.assetType == "serviceusage.googleapis.com/Service") | .displayName'
 gcloud_asset_search-all-resources.json
"serviceusage.googleapis.com"
"sqladmin.googleapis.com"
"storage-api.googleapis.com"
"dataplex.googleapis.com"
"servicemanagement.googleapis.com"
"bigqueryreservation.googleapis.com"
"bigquerydatapolicy.googleapis.com"
"cloudtrace.googleapis.com"
"bigquerystorage.googleapis.com"
"bigquerymigration.googleapis.com"
"compute.googleapis.com"
"logging.googleapis.com"
"datastore.googleapis.com"
"cloudapis.googleapis.com"
"bigquery.googleapis.com"
"bigqueryconnection.googleapis.com"
"monitoring.googleapis.com"
"storage.googleapis.com"
"dataform.googleapis.com"
"storage-component.googleapis.com"
"oslogin.googleapis.com"
"servicenetworking.googleapis.com"
"analyticshub.googleapis.com"
"sql-component.googleapis.com"
```

Abbildung 16: Google Cloud Wordpress Szenario: Ressourcenübersicht

Als nächstes wird das virtuelle Netzwerk erfasst (Listing 42). In der Liste der virtuellen Netzwerke (1) ist auch das Standard Netzwerk vorhanden. Weiter werden die Subnetze des virutellen Netzwerks `wordpress-virtual-network` erfasst (2). Zum genannten Netzwerk werden auch die Firewall-Regeln erfasst (3). Die Übersicht der Routen (4) umfasst auch die Routen des Standard Netzwerks. Diese können gefiltert werden (5). Abschließend werden statische IP-Adressen abgerufen (6).

```
1  gcloud compute networks list --project=wordpress-project --format=json >
   ↪ gcloud_compute_networks_list.json
2  gcloud compute networks subnets list --network=wordpress-virtual-network
   ↪ --project=wordpress-project --format=json >
   ↪ gcloud_compute_networks_subnets_list.json
3  gcloud compute networks get-effective-firewalls
   ↪ wordpress-virtual-network --project=wordpress-project --format=json
   ↪ > gcloud_compute_networks_get-effective-firewallss.json
4  gcloud compute routes list --project=wordpress-project --format=json >
   ↪ gcloud_compute_routes_list.json
5  jq '.[] | select( .network | contains("default") | not) | .description,
   ↪ .destRange' gcloud_compute_routes_list.json
6  gcloud compute addresses list --project=wordpress-project --format=json
   ↪ > gcloud_compute_addresses_list.json
```

Listing 42: Google Cloud Wordpress Szenario: Erfassung: virtuelles Netzwerk

Diese Phase wird mit der Erfassung der virtuellen Maschine und Datenbank abgeschlossen (Listing 43). In der Anfangs erfassten Übersicht aller Ressourcen sind Informationen über die VM vorhanden (1). Detaillierte Informationen zur VM werden erfasst (2). Eine Übersicht der Datenbank-Instanzen wird erstellt (3). Unter `.settings.ipConfiguration` wird erkenntlich, dass die Datenbank über einen Google Privat Path im oben erfassten virtuellen Netzwerk erreichbar ist. Weiter werden die einzelnen Datenbanken (siehe Abbildung 17) der Datenbank-Instanz erfasst (5).

```
1   jq '.[] | select(.assetType == "compute.googleapis.com/Instance") | .'
    ↪ gcloud_asset_search-all-resources.json
2   gcloud compute instances describe wordpress-vm
    ↪ --project=wordpress-project --format=json >
    ↪ gcloud_compute_instances_describe.json
3
4   gcloud sql instances list --project=wordpress-project --format=json >
    ↪ gcloud_sql_instances_list.json
5   jq '.[] | .settings.ipConfiguration' gcloud_sql_instances_list.json
6   gcloud sql databases list --instance=wordpress-sql-db
    ↪ --project=wordpress-project --format=json | tee
    ↪ gcloud_sql_databases_list.json | jq '.[].name'
```

Listing 43: Google Cloud Wordpress Szenario: Erfassung: VM und Datenbank

```
> jq '.[].name' gcloud_sql_databases_list.json
"mysql"
"information_schema"
"performance_schema"
"sys"
"wordpress-db"
```

Abbildung 17: Google Cloud Wordpress Szenario: Datenbankübersicht

Sicherung Diese Phase beginnt mit der Vorbereitung zur Sicherung der Festplatte der virtuellen Maschine (Listing 44). Dazu wird ein Snapshot der Festplatte erstellt (1) und Informationen über jenen Snapshot gesammelt (2). Als Zwischenspeicher wird ein Objektspeicher angelegt (4). Eine forensische VM wird erstellt (6). Folgende Parameter müssen beachtet werden:

- `--create-disk=auto-delete=yes,boot=yes,device-name=instance-1,image=projects/debian-cloud/global/images/debian-12-bookworm-v20240213,mode=rw,size=10` Erstellung einer Bootdisk auf Debian-basis
- `--create-disk=device-name=disk-1,mode=ro,auto-delete=no,name=snapshot-disk,size=10,source-snapshot=projects/wordpress-project/global/snapshots/wordpress-vm-snapshot1` Einbindung des zuvor erstellten Snapshots im Read-Only Modus
- `--scopes=https://www.googleapis.com/auth/devstorage.read_write` Berechtigung der VM um in den Objektspeicher zu schreiben

```
1 gcloud compute snapshots create wordpress-vm-snapshot1
  ↪ --source-disk-zone=us-east1-b --source-disk=wordpress-vm
  ↪ --project=wordpress-project --format=json >
  ↪ gcloud_compute_snapshots_create.json
2 gcloud compute snapshots describe wordpress-vm-snapshot1
  ↪ --project=wordpress-project --format=json >
  ↪ gcloud_compute_snapshots_describe.json
3
4 gcloud storage buckets create gs://wordpress-project-forensic-bucket
  ↪ --location=us --project=wordpress-project
5
6 gcloud compute instances create forensic-vm --project=wordpress-project
  ↪ --zone=us-east1-b --machine-type=e2-micro --network-interface=network-
  ↪ tier=STANDARD,stack-type=IPV4_ONLY,subnet=wordpress-public-subnet
  ↪ --create-disk=auto-delete=yes,boot=yes,device-name=instance-
  ↪ 1,image=projects/debian-cloud/global/images/debian-12-bookworm-
  ↪ v20240213,mode=rw,size=10
  ↪ --create-disk=device-name=disk-1,mode=ro,auto-delete=no,name=snapshot-
  ↪ disk,size=10,source-snapshot=projects/wordpress-
  ↪ project/global/snapshots/wordpress-vm-snapshot1
  ↪ --project=wordpress-project
  ↪ --scopes=https://www.googleapis.com/auth/devstorage.read_write
  ↪ --format=json > gcloud_compute_instances_create.json
```

Listing 44: Google Cloud Wordpress Szenario: Sicherung Festplatte, Vorbereitung

Anschließend wird mithilfe der forensischen VM der Snapshot gesichert (Listing 45). Dazu wird eine SSH-Verbindung aufgebaut (1) und die benötigten Werkzeuge installiert (3). Mithilfe `sudo fdisk -l` werden die angeschlossenen Festplatten geprüft. Darauf wird der Snapshot mit `dc3dd` gesichert und direkt in den Objektspeicher kopiert (5). `dc3dd` errechnet direkt beim Kopieren eine Hashwert, welcher später zur Integritätsprüfung verwendet wird. Dieser Hashwert wird ebenfalls in den Objektspeicher kopiert (7).

```
gcloud compute ssh cloud@forensic-vm --zone=us-east1-b
  ↪ --project=wordpress-project

sudo apt install fdisk dc3dd
sudo fdisk -l

sudo dc3dd if=/dev/sdb hash=sha256 hlog=wordpressvm.bin.sha256 | gsutil cp -
  ↪ gs://wordpress-project-forensic-bucket/forensic-images/wordpressvm.bin

gsutil cp wordpressvm.bin.sha256 gs://wordpress-project-forensic-
  ↪ bucket/forensic-images/wordpressvm.bin.sha256
```

Listing 45: Google Cloud Wordpress Szenario: Sicherung Festplatte

Als nächstes wird die Sicherung des Arbeitsspeichers vorbereitet (Listing 46). Die Ziel-VM hat gegebenenfalls kein schreibenden Zugriff auf den Objektspeicher. Um dies vorzubeugen werden pre-signed URLs zur Übertragung erstellt. Dazu wird ein Dienstkontoschlüssel `glcoud.key` unter Angabe des Dienstkontos (1) erstellt (2). Mithilfe diesen Dienstkontoschlüssel lassen sich pre-signed URLs für jeweils den Arbeitsspeicher (4) und Hashwert (5) erstellen. Diese URLs sind für 3 Stunden gültig.

```
gcloud iam service-accounts list --project=wordpress-project --format=json |
  ↪ tee gcloud_iam_service-accounts_list.json | jq '.[].email'
gcloud iam service-accounts keys create gcloud.key
  ↪ --iam-account=123456789-compute@developer.gserviceaccount.com
  ↪ --format=json

gcloud storage sign-url
  ↪ gs://wordpress-project-forensic-bucket/ram/wordpressvm.ram
  ↪ --http-verb=PUT --duration=3h
  ↪ --headers=content-type=application/octet-stream
  ↪ --private-key-file=gcloud.key
gcloud storage sign-url
  ↪ gs://wordpress-project-forensic-bucket/ram/wordpressvm.ram.sha256
  ↪ --http-verb=PUT --duration=3h
  ↪ --headers=content-type=application/octet-stream
  ↪ --private-key-file=gcloud.key
```

Listing 46: Google Cloud Wordpress Szenario: Sicherung Arbeitsspeicher, Vorbereitung

Im Anschluss wird der Arbeitsspeicher der virtuellen Maschine gesichert (Listing

47). Dazu wird eine SSH-Verbindung aufgebaut (1). Als Tool wurde AVML (Acquire Volatile Memory for Linux) gewählt. Dies wird heruntergeladen (3), ausführbar gemacht (4) und ausgeführt (5). Vom erstellten Speicherabbild wird ein Hashwert erstellt (6) und beide Dateien in den Objektspeicher kopiert (8, 9).

```
1 gcloud compute ssh cloud@wordpress-vm --zone=us-east1-b
  ↪  --project=wordpress-project
2
3 wget https://github.com/microsoft/avml/releases/download/v0.13.0/avml
4 chmod +x avml
5 sudo ./avml --compress /tmp/wordpressvm-ram.bin
6 sha256sum /tmp/wordpressvm-ram.bin > /tmp/wordpressvm-ram.bin.sha256
7
8 curl -X PUT -H 'Content-Type: application/octet-stream' --upload-file
  ↪  /tmp/wordpressvm-ram.bin <pre-sigend url>
9 curl -X PUT -H 'Content-Type: application/octet-stream' --upload-file
  ↪  /tmp/wordpressvm-ram.bin.sha256  <pre-sigend url>
```

Listing 47: Google Cloud Wordpress Szenario: Sicherung Arbeitsspeicher

Darauf folgt die Sicherung der Datenbank (Listing 48). Als Vorbereitung muss der ServiceAccount des Datenbankdienstes festgestellt werden (1) um anschließend diesem Service-Account schreibender Zugriff auf den forensischen Objektspeicher zu gewähren (2). Abschließend kann pro Datenbank ein SQL-Dump durchgeführt werden (4-8). Die Datenbanken wurden im vorherigen Abschnitt erfasst (siehe Abbildung 17).

```
jq '.[].serviceAccountEmailAddress' gcloud_sql_instances_list.json
gsutil iam ch serviceAccount:p123456789-etfvr7@gcp-sa-cloud-
  ↪ sql.iam.gserviceaccount.com:objectAdmin
  ↪ gs://wordpress-project-forensic-bucket

gcloud sql export sql wordpress-sql-db
  ↪ gs://wordpress-project-forensic-bucket/database/mysql.sql
  ↪ --database=mysql --project=wordpress-project
gcloud sql export sql wordpress-sql-db
  ↪ gs://wordpress-project-forensic-bucket/database/information_schema.sql
  ↪ --database=information_schema --project=wordpress-project
gcloud sql export sql wordpress-sql-db
  ↪ gs://wordpress-project-forensic-bucket/database/performance_schema.sql
  ↪ --database=performance_schema --project=wordpress-project
gcloud sql export sql wordpress-sql-db
  ↪ gs://wordpress-project-forensic-bucket/database/sys.sql --database=sys
  ↪ --project=wordpress-project
gcloud sql export sql wordpress-sql-db
  ↪ gs://wordpress-project-forensic-bucket/database/wordpress-db.sql
  ↪ --database=wordpress-db  --project=wordpress-project
```

Listing 48: Google Cloud Wordpress Szenario: Sicherung Datenbank

Extraktion Die Extraktion schließt die Phase der Cloud-Forensik ab (Listing 49). Die gesicherten Daten werden aus dem Objektspeicher lokal kopiert (1). Abschließend wird die Übertragung mit einem Hashwertabgleich geprüft (4,5). Die Übertragung des Objektspeichers kann mithilfe des Objektspeicher-Inhaltsverzeichnis (2) überprüft werden. Diese beinhaltet unter anderem auch den MD5-Hashwert jeder Datei.

```
gcloud storage rsync gs://wordpress-project-forensic-bucket
  ↪ wordpress-project-forensic-bucket --recursive
gcloud storage ls --recursive --all-versions --json
  ↪ gs://wordpress-project-forensic-bucket --project=wordpress-project >
  ↪ gcloud_storage_ls_--recursive_--all-versions.json

sha256sum -c wordpressvm.bin.sha256
sha256sum -c wordpressvm-ram.bin.sha256
```

Listing 49: Google Cloud Wordpress Szenario: Extraktion

4.3.4 AWS: CDN

Zur Realisierung dieses Szenarios werden auf folgende drei AWS-Dienste zurückgegriffen: *Route53* zur Domainregistrierung und DNS-Verwaltung, *Cloudfront* zum Caching und *S3* als Datenspeicher. Zuerst wird ein Objektspeicher (*S3-Bucket*) erstellt und die statische Website, bestehend aus index.html und cat.jpg, hochgeladen. Auf diese Daten kann standardmäßig nicht zugegriffen werden. Durch ein Anpassen der Bucket-Policy kann ausschließlich der Cloudfront-Dienst auf diese Daten zugreifen. Nach Erstellen des Cloudfront-Dienst ist die Website über eine Cloudfront-URL erreichbar. Nachdem über Route53 eine Domain gebucht wurde, wird ein Alias-DNS-Record erstellt. Dieser verweist von der gebuchten Domain auf die Cloudfront-URL. Abschließend wird mithilfe einer manuellen Verifizierung ein SSL-Zertifikat für die gebuchte Domain erstellt.

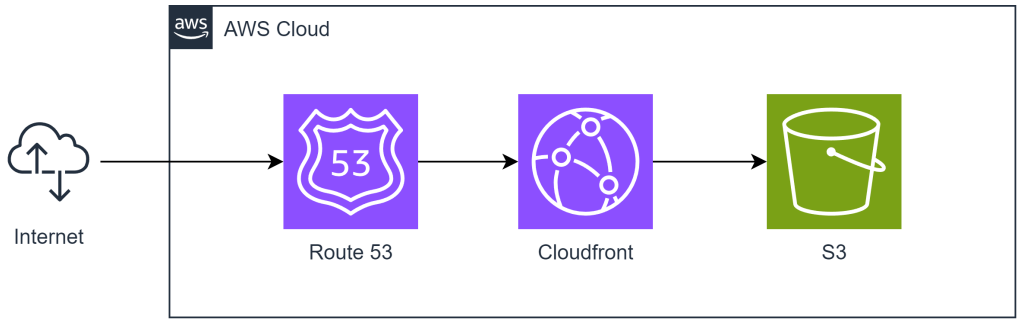


Abbildung 18: AWS CDN Szenario

Aufbau

Zu Beginn (Listing 50) wird ein S3-Bucket erstellt (1) und die statische Website (5-15) hochgeladen (2,3). Die Daten sind standardmäßig vor Zugriffen von anderen Diensten oder von außen geschützt.

```
1     aws s3api create-bucket --bucket static-website-bucket --region
      ↪  us-east-1
2     aws s3 cp index.html s3://static-website-bucket/
3     aws s3 cp cat.jpg s3://static-website-bucket/
4
5     <!DOCTYPE html>
6     <html>
7     <body>
8
9     <h1>Willkommen!</h1>
10
11    <p>Lorem ipsum dolor sit amet</p>
12
13    <img src=./cat.jpg>
14    </body>
15    </html>
```

Listing 50: AWS CDN Szenario: S3-Bucket erstellen und befüllen

Als nächstes wird die Cloudfront Distribution erstellt (Listing 51). Dazu wird als erstes ein Origin Access Control (OAC) erstellt (1). Dies wird in Verbindung mit einer Bucket-Policy genutzt um sicherzustellen, dass nur die Cloudfront Distribution Zugriff auf den S3-Bucket hat. Die ID der gerade erstellen OAC wird notiert (2). Daraufhin wird die Cloudfront Distribution erstellt (3). Wenn diese von AWS bereitgestellt wurde, kann ID und DomainName notiert werden (4).

```
1 aws cloudfront create-origin-access-control --origin-access-control-config
  ↪ Name=static-website-bucket.s3.amazonaws.com,Description=StaticPageOAC,
  ↪ SigningProtocol=sigv4,SigningBehavior=always,
  ↪ OriginAccessControlOriginType=s3
2 aws cloudfront list-origin-access-controls | jq
  ↪ '.OriginAccessControlList.Items | .[] | .Id'
3 aws cloudfront create-distribution --origin-domain-name
  ↪ static-website-bucket.s3.amazonaws.com --default-root-object index.html
4 aws cloudfront list-distributions | jq '.DistributionList.Items | .[] | .Id,
  ↪ .Status, .DomainName'
```

Listing 51: AWS CDN Szenario: Cloudfront Distribution erstellen

Direkt folgend wird die Konfiguration der Cloudfront Distribution angepasst(Listing 52). Dazu wird die aktuelle Konfiguration heruntergeladen (1) und die vorhin notierte ID des OAC ergänzt (3). Bevor die Konfiguration gepatcht (6) werde kann,

muss noch der aktuelle Entity Tag (ETag) notiert werden (5). ETag bezeichnet hier den Hashwert der Konfiguration.

```
1 aws cloudfront get-distribution-config --id E2S6APVTUPMICQ | jq
  ↪ '.DistributionConfig' > aws-dist.config
2
3 "OriginAccessControlId": "EMKEMNBRV5P8E"
4
5 aws cloudfront get-distribution-config --id E2S6APVTUPMICQ | jq '.ETag'
6 aws cloudfront update-distribution --id E2S6APVTUPMICQ --distribution-config
  ↪ file://aws-dist.config --if-match ESKLQUP0CI2QB
```

Listing 52: AWS CDN Szenario: Cloudfront Distribution konfigurieren

Damit die Cloudfront Distribution noch Zugriff auf den S3-Bucket erhält, muss dessen Policy[11] gepatcht werden (Listing 53). Dafür wird eine Datei policy.json erstellt (1-20) und gepatcht (22).

- `"Effect": "Allow"` bezeichnet ein Erlauben,
- `"Service": "cloudfront.amazonaws.com"` welchem Service etwas erlaubt wird,
- `"Action": "s3:GetObject"` welche Aktionen erlaubt sind,
- `"Resource": "arn:aws:s3:::static-website-bucket/*"` auf welche Ressourcen und
- `"AWS:SourceArn": "arn:aws:cloudfront::734635139315:distribution/E2S6APVTUPMICQ"` nur von der vorher erstellten Cloudfront Distribution.

```
{
    "Version": "2008-10-17",
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
        {
            "Sid": "AllowCloudFrontServicePrincipal",
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudfront.amazonaws.com"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::static-website-bucket/*",
            "Condition": {
                "StringEquals": {
                  "AWS:SourceArn":
                    ↪ "arn:aws:cloudfront::734635139315:distribution/E2S6APVTUPMICQ"
                }
            }
        }
    ]
}

aws s3api put-bucket-policy --bucket static-website-bucket --policy
↪ file://policy.json
```

Listing 53: AWS CDN Szenario: S3-Bucket Policy patchen

Als letzter Bestandteil wird mithilfe von *Route53* eine Domain gebucht (Listing 54). Dazu wird eine Preisliste der Top-Level-Domains eingeholt (1) und die Domain auf Verfügbarkeit überprüft (2). Diese kann anschließend, unter Angabe der Kontaktdaten in der register-domain.json Datei, registriert werden (3). Nach mehrmaligem Prüfen, ob die Domain registriert wurde (4), wird ein Alias DNS-Record gesetzt (5). In der record.json Datei (Listing 55) ist die gebuchte Domain sowie Cloudfront-URL hinterlegt.

```
aws route53domains list-prices | jq '.Prices | .[] |
↪ (.RegistrationPrice.Price|tostring) + " - " + .Name' | cut -f2 -d'"' |
↪ sort -n
aws route53domains check-domain-availability --domain-name myawsproject.com
aws route53domains register-domain --region us-east-1 --cli-input-json
↪ file://register-domain.json
aws route53domains list-operations
aws route53 change-resource-record-sets --hosted-zone-id
↪ Z09207711FA5QCJZCJXDW --change-batch file://record.json
```

Listing 54: AWS CDN Szenario: Domain buchen und DNS-Record setzen

```
{
    "Comment": "Creating Alias resource record sets in Route 53",
    "Changes": [
        {
            "Action": "CREATE",
            "ResourceRecordSet": {
                "Name": "myawsproject.com",
                "Type": "A",
                "AliasTarget": {
                  "HostedZoneId": "Z2FDTNDATAQYW2",
                  "DNSName": "d33ozd2nyaik8f.cloudfront.net",
                  "EvaluateTargetHealth": false
                }
            }
        }
    ]
}
```

Listing 55: AWS CDN Szenario: Alias DNS-Record

Abschließend wird für gebuchte Domain ein SSL-Zertifikat generiert (Listing 56). Das Zertifikat wird beantragt (1). Als Validierungsmethode wird DNS angegeben, d. h. ein DNS-Eintrag muss manuell gesetzt werden. Der zu setzende Wert wird ausgelesen (2) und ein neuer DNS-Record gesetzt (3). Nachdem die Validierung erfolgreich war (4) wird der DNS-Record wieder gelöscht (5). Abschließend wird noch der `"Aliases"` sowie `"ViewerCertificate"` Abschnitt in der Cloudfront Distribution Konfiguration angepasst und geupdatet (6). Dies ist notwendig, damit der Cloudfront-Dienst Anfragen über die Domain myawsproject.com mit dem angegebenen Zertifikat beantwortet. Die statische Webseite im Objektspeicher ist nun durch die gebuchte Domain erreichbar.

```
1 aws acm request-certificate --domain-name myawsproject.com
  ↪ --validation-method DNS
2 aws acm describe-certificate --certificate-arn arn:aws:acm:us-east-
  ↪ 1:734635139315:certificate/d84c6122-1d20-4d19-b220-dabf2e5f769b | jq
  ↪ '.Certificate.DomainValidationOptions'
3 aws route53 change-resource-record-sets --hosted-zone-id
  ↪ Z09207711FA5QCJZCJXDW --change-batch file://record_val.json
4 aws acm describe-certificate --certificate-arn arn:aws:acm:us-east-
  ↪ 1:734635139315:certificate/d84c6122-1d20-4d19-b220-dabf2e5f769b | jq
  ↪ '.Certificate.DomainValidationOptions | .[].ValidationStatus'
5 aws route53 change-resource-record-sets --hosted-zone-id
  ↪ Z09207711FA5QCJZCJXDW --change-batch file://record_val_del.json
6 aws cloudfront update-distribution --id E2S6APVTUPMICQ --distribution-config
  ↪ file://aws-dist.config --if-match ESKLQUP0CI2QB
```

Listing 56: AWS CDN Szenario: SSL-Zertifikatsgenerierung

Durchführung Leitfaden

Orientierung Im Abschnitt Orientierung wird der eigene Zugang auf Berechtigungen geprüft (Listing 57) und Logs der API-Aufrufe gesammelt (Listing 58). Mithilfe von `aws sts get-caller-identity` werden Informationen zum aktuellen Benutzer gesammelt (1) und die enthaltene ARN notiert. Im nächsten Schritt werden zuerst alle Informationen über Benutzer, Gruppe, Rollen und Policies gesammelt (2) und Berechtigungen des eigenen Nutzers angezeigt.

```
1 aws sts get-caller-identity | tee aws-own-identity.json | jq '.Arn'
  ↪ aws-own-identity.json
2 aws iam get-account-authorization-details | tee aws-account-details.json |
  ↪ jq '.[] |.[] | select (.Arn=="arn:aws:iam::12345:user/CLI")'
```

Listing 57: AWS CDN Szenario: Orientierung: Eigener Zugang prüfen

Die API-Logs (Listing 58) werden beim AWS-Dienst *Cloudtrail* verwaltet und in Objektspeicher (*Buckets*) gespeichert. Um diese zu exportieren werden zuerst alle alle Protokollgruppen (*Trails*) gelistet (1) und anschließend der Bucketname für die `management-events` gesucht. Die `management-events` enthalten Protokolle über API-Aufrufe. Als nächstes können die API-Logs exportiert werden (3). Abschließend sollten die API-Logs in ein Archiv gepackt werden (4) um Veränderungen zu vermeiden.

```
1 aws cloudtrail list-trails | tee aws_cloudtrail_list-trails.json | jq
  ↪ '.Trails | .[] | .Name'
2 aws cloudtrail get-trail --name management-events | tee
  ↪ aws_cloudtrail_get-trail.json | jq '.Trail.S3BucketName'
3 aws s3 sync s3://aws-cloudtrail-logs-734635139315-6e37996e log-export
4 tar czf aws-cloudtrail-logs-734635139315-6e37996e.tar.gz logs-export
```

Listing 58: AWS CDN Szenario: Orientierung: API-Logs exportieren

Erfassung Der erste Schritt in der Erfassung besteht darin die genutzten Ressourcen zu suchen und aufzulisten. Wie bereits bei dem AWS Wordpress Szenario erläutert gibt es hierfür mehrere Ansätze. An dieser Stelle wurde der Tag Editor genutzt (Abbildung 19). Anhand diesem wurde die Nutzung von folgenden Diensten festgestellt: CertificateManager, CloudFront, CloudTrail, Route53 und S3-Bucket.

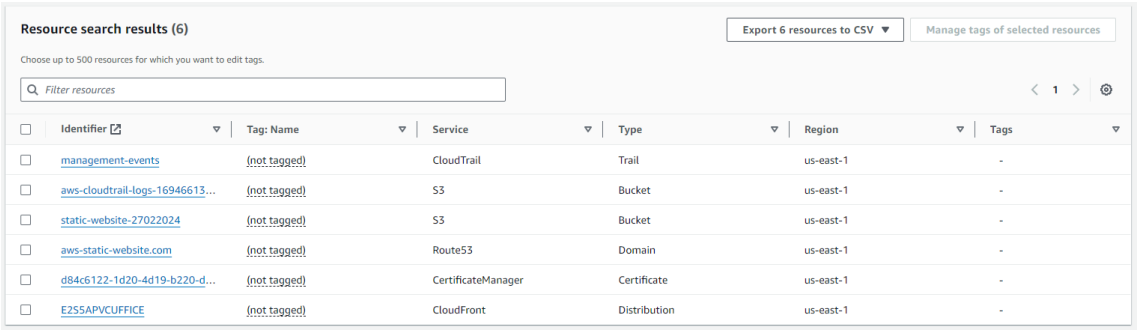


Abbildung 19: AWS CDN Szenario: Ressourcenübersicht

Nach der Feststellung der genutzten Dienste werden diese einzeln erfasst, angefangen mit dem CertificateManager (Listing 59). Alle Zertifikate werden aufgelistet und deren ARN angezeigt (1). Mithilfe der ARN wird das Zertifikat exportiert. Bei Sichtung der Liste sowie Prüfung des Zertifikats durch das Programm `openssl` wird eine Domain myawsproject.com festgestellt. Wird diese Domain nicht bei Erfassung der restlichen Dienste festgestellt, so wird eine nächsten Iteration ausgeführt.

```
1 aws acm list-certificates | tee aws_acm_list-certificates.json | jq
  ↪ '.[].[].CertificateArn'
2 aws acm get-certificate --certificate-arn arn:aws:acm:us-east-
  ↪ 1:734635139315:certificate/d84c6122-1d20-4d19-b220-dabf2e5f769b >
  ↪ aws_cert_us-east1.json
```

Listing 59: AWS CDN Szenario: Erfassung: CertificateManager

Als nächstes werden Ressourcen aus CloudFront erfasst (Listing 60). Dazu werden alle CloudFront Distributions (1) und anschließend alle OAC (2) gelistet. Bei der Sichtung wird ein Zertifikat, die Domain myawsproject.com sowie der S3-Bucket static-website-bucket.s3.us-east-1.amazonaws.com festgestellt.

```
1 aws cloudfront list-distributions > aws_cloudfront_list-distributions.json
2 aws cloudfront list-origin-access-controls >
  ↪ aws_cloudfront_list-origin-access-controls.json
```

Listing 60: AWS CDN Szenario: Erfassung: Cloudfront

Bei der Erfassung von Route53 (Listing 61) werden die sog. Hosted-Zones erfasst (1). In Hosted-Zones werden DNS-Einträge verwaltet. Diese DNS-Einträge werden unter Angabe der Hosted-Zone-ID ebenfalls erfasst (2). Anschließend werden die Domains

erfasst (3,4,5). Hier sind Informationen wie z.B. Buchungsdatum, Ablaufdatum und Kontaktdaten hinterlegt.

```
1 aws route53 list-hosted-zones | tee aws_route53_list-hosted-zones.json | jq
  ↪ '.[].[].Id'
2 aws route53 list-resource-record-sets --hosted-zone-id Z09207711FA5QCJZCJXDW
  ↪ > aws_route53_list-resource-record-sets.json
3 aws route53domains list-domains > aws_route53domains_list-domains.json
4 aws route53domains list-operations > aws_route53domains_list-operations.json
5 aws route53domains get-domain-detail --domain-name myawsproject.com >
  ↪ aws_route53domains_get-domain-detail.json
```

Listing 61: AWS CDN Szenario: Erfassung: Route53

Diese Phase wird durch die Erfassung des S3-Buckets abgeschlossen (Listing 62). Alle Buckets werden aufgelistet (1). Hier wird auch der CloudTrail Bucket angezeigt. Als nächstes wird ein Inhaltsverzeichnis des Buckets erstellt (2). Eine Versionierung kann geprüft (3) und eingesehen (4) werden. Zusätzlich werden die AccessControlList (5), Verschlüsselungsstatus (6) und Policy (7) erfasst.

```
1 aws s3api list-buckets | tee aws_s3api_list-buckets.json | jq
  ↪ '.Buckets.[].Name'
2 aws s3api list-objects --bucket static-website-bucket >
  ↪ aws_s3api_list-objects.json
3 aws s3api get-bucket-versioning --bucket static-website-bucket >
  ↪ aws_s3api_get-bucket-versioning.json
4 aws s3api list-object-versions --bucket static-website-bucket >
  ↪ aws_s3api_list-object-versions.json
5 aws s3api get-bucket-acl --bucket static-website-bucket >
  ↪ aws_s3api_get-bucket-acl.json
6 aws s3api get-bucket-encryption --bucket static-website-bucket >
  ↪ aws_s3api_get-bucket-encryption.json
7 aws s3api get-bucket-policy --bucket static-website-bucket >
  ↪ aws_s3api_get-bucket-policy.json
```

Listing 62: AWS CDN Szenario: Erfassung: S3-Bucket

Abschließend wird die grafische Übersicht geprüft. Es sind keine weiteren Iterationen notwendig, da bei der Sichtung der einzelnen Ressourcen keine neuen Ressourcen festgestellt wurden.

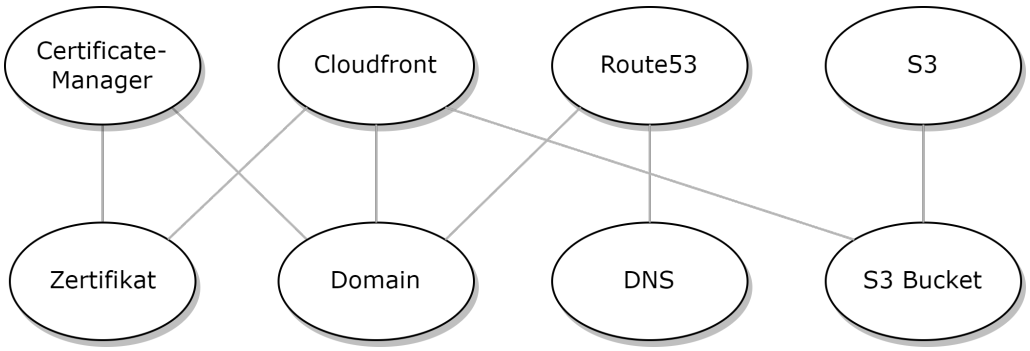


Abbildung 20: AWS CDN Szenario: grafische Übersicht der Erfassung

Sicherung und Extraktion Die einzigen Inhaltsdaten sind im S3-Bucket gespeichert. Dieser wird folgendermaßen gesichert (Listing 63). Die Daten werden aus dem S3-Bucket auf den lokalen Computer gesichert (1), in ein Archiv gepackt (2) und zur Integritätssicherung gehasht (3). Bei der Sicherung bleiben die Änderungszeitstempel erhalten. Diese sind zusätzlich in der Erfassung des S3-Buckets hinterlegt.

```
1 aws s3 sync s3://static-website-bucket ./s3/
2 tar czf s3_static-website-bucket.tar.gz s3/
3 sha256sum s3_static-website-bucket.tar.gz >
  ↪ s3_static-website-bucket.tar.gz.sha256
```

Listing 63: AWS CDN Szenario: Sicherung

4.3.5 Azure: CDN

Dieses Szenario beim CSP Microsoft Azure besteht aus 3 Komponenten: Domain, CDN und Objektspeicher. Zur Domainregistrierung wird *App Service domain* genutzt. Bei Buchung einer Domain wird automatisch eine *DNS-Zone* (1) erstellt. Diese verwaltet die korrespondierenden DNS-Records. Das CDN besteht aus 2 Elemente: einen Endpoint und ein Origin. Der Endpoint (2) ist durch eine `.azureedge.net` URL erreichbar und greif auf die Ressourcen aus dem Origin zurück. Der Origin verweist auf den Objektspeicher. Im Objektspeicher (3) wird die statische Website, bestehend aus index.html und cat.jpg, abgelegt. Durch Konfiguration des Objektspeichers ist die statische Website zugreifbar.
Aus hier nicht weiter erläuterten Gründen konnte keine Domain über Azure gebucht werden. Die korrespondierenden CLI-Befehle werden trotzdem aufgeführt.

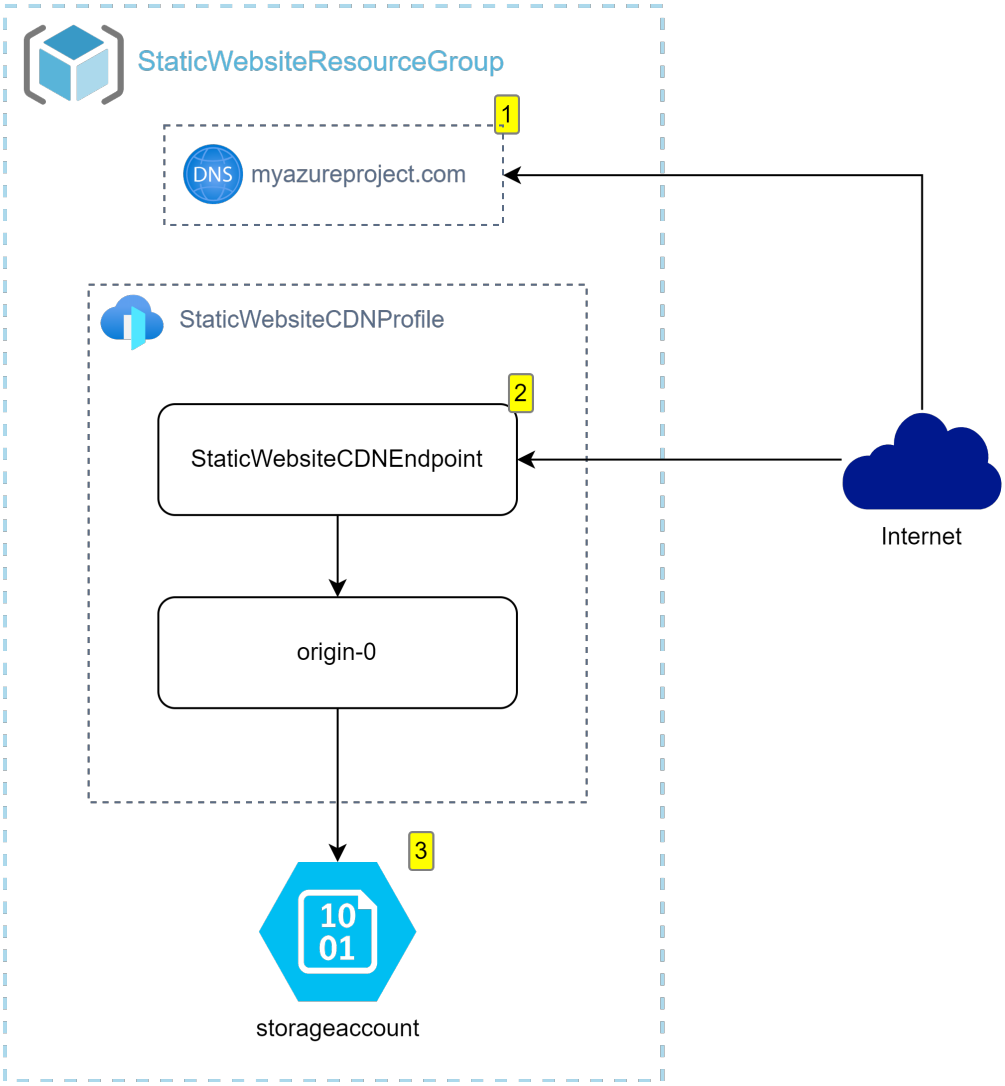


Abbildung 21: Azure CDN Szenario

Aufbau

Zu Beginn (Listing 64) wird eine Ressourcengruppe (*ResourceGroup*) erstellt (1). Alle weiteren Ressourcen werden in dieser Ressourcengruppe gebündelt.

```
1 az group create --name StaticWebsiteResourceGroup --location eastus
```

Listing 64: Azure CDN Szenario: Ressourcengruppe

Als nächstes wird der Objektspeicher angelegt (Listing 65). Dazu wird ein Storage Account erstellt (1) und dessen Account-Key ausgelesen (2). Für diesen Storage Account wird das *Static Website* Feature aktiviert. Dadurch wird automatisch auch ein Container `$web` erstellt. In diesen Container werden die index.html und cat.jpg (3,4) hochgeladen. Abschließend wird die Static Website URL notiert (5).

```
1 az storage account create --name storageaccount -g
  ↪  StaticWebsiteResourceGroup --allow-blob-public-access true
2 az storage account keys list -g StaticWebsiteResourceGroup -n storageaccount
  ↪  | jq '.[0].value'
3 az storage blob service-properties update --account-name storageaccount
  ↪  --static-website --index-document index.html
4 az storage blob upload -c '$web' --account-name storageaccount -f
  ↪  ./index.html -n index.html --account-key <account-key>
5 az storage blob upload -c '$web' --account-name storageaccount -f ./cat.jpg
  ↪  -n cat.jpg --account-key <account-key>
6 az storage account show -n storageaccount -g StaticWebsiteResourceGroup
  ↪  --query "primaryEndpoints.web" --output tsv
```

Listing 65: Azure CDN Szenario: Objektspeicher

Anschließend wird die Domain gebucht (Listing 66). Dazu wird die Verfügbarkeit und Preis geprüft (1) und darauffolgend gebucht (2). Folgend wird noch das SSL-Zertifikat erstellt (3).

```
1 az appservice domain show-terms --hostname myazureproject.com
2 az appservice domain create -g StaticWebsiteResourceGroup --hostname
  ↪  myazureproject.com --contact-info=@'./contact_info.json' --accept-terms
3 az webapp config ssl create --resource-group StaticWebsiteResourceGroup
  ↪  --name MyAzureProject --hostname myazureproject.com
```

Listing 66: Azure CDN Szenario: Domainbuchung

Abschließend wird das CDN erstellt und konfiguriert (Listing 67). Ein CDN Profil wird erstellt (1) und darin ein Endpoint (2). Allgemein können im CDN Profilen mehrere Endpoints gesammelt werden. Bei der Erstellung des Endpoints (2) wurde durch Angabe des Objektspeichers ein korrespondierender Origin erstellt. Ein CName DNS-Record der gebuchten Domain wird erstellt (4) und verweist auf den CDN Endpoint (5). Zum Schluss wird ein zusätzlicher Endpoint mit verweis auf die Domain erstellt (7) und HTTPS aktiviert (8)

```
1 az cdn profile create --resource-group StaticWebsiteResourceGroup --name
  ↪  StaticWebsiteCDNProfile --sku Standard_Microsoft
2 az cdn endpoint create --resource-group StaticWebsiteResourceGroup --name
  ↪  StaticWebsiteCDNEndpoint --profile-name StaticWebsiteCDNProfile --origin
  ↪  storageaccount.z13.web.core.windows.net --origin-host-header
  ↪  storageaccount.z13.web.core.windows.net
3
4 az network dns record-set cname create -g StaticWebsiteResourceGroup -z
  ↪  myazureproject.com -n www
5 az network dns record-set cname set-record -g StaticWebsiteResourceGroup -z
  ↪  myazureproject.com -n www -c StaticWebsiteCDNEndpoint.azureedge.net
6
7 az cdn custom-domain create --name StaticWebsiteCustomDomainEndpoint
  ↪  --resource-group StaticWebsiteResourceGroup --endpoint-name
  ↪  StaticWebsiteCDNEndpoint --profile-name StaticWebsiteCDNProfile
  ↪  --hostname myazureproject.com
8 az cdn custom-domain enable-https --resource-group
  ↪  StaticWebsiteResourceGroup --endpoint-name StaticWebsiteCDNEndpoint
  ↪  --profile-name StaticWebsiteCDNProfile --name
  ↪  StaticWebsiteCustomDomainEndpoint
```

Listing 67: Azure CDN Szenario: CDN Konfiguration

Durchführung Leitfaden

Orientierung Im Abschnitt Orientierung wird der eigene Zugang auf Berechtigungen geprüft (Listing 68). Beginnend mit dem aktuell eingeloggten Nutzer (1) wird dessen ID notiert. Weiter wird einer Übersicht des eingeloggten Accounts (2) und ein Übersicht über alle Nutzer gesichert (3). Mit den nächsten zwei Aufrufen wird die Rolle zur vorher notierten ID festgestellt (4) und die Berechtigungen dieser Rolle überprüft.

```
az ad signed-in-user show | tee az_ad_signed-in-user_show.json | jq '.id'
az account list > az_account_list.json
az ad user list > az_ad_user_list.json
az role assignment list --assignee 793f0893-3d0b-4bfb-9936-f2cf45d181a4 >
  ↪  az_role_assignment_list.json
az role definition list --name Owner > az_role_definition_list.json
```

Listing 68: Azure CDN Szenario: Orientierung: Eigener Zugang prüfen

Als nächstes werden die API-Logs gesichert (Listing 69). Der Aufbewahrungszeitraum beträgt standardmäßig 90 Tage. Eine Abfrage liefert standardmäßig 50 Einträge, die getätigte Abfrage (1) liefert bis zu 10000 Einträge. Die exportierten Activity-Logs enthalten ca. 2000 Einträge (2). Wäre die Anzahl an Einträgen ausgeschöpft, so müsste eine erneute Abfrage mit entweder mehr Einträge oder angepasster Start- und Endzeit getätigt werden. Um einen Überblick zu verschaffen werden die letzten 10 Log-Einträge gesichtet.

```
az monitor activity-log list --offset 90d --max-events 10000 >
  ↪  az_monitor_activity-log_list.json
jq length az_monitor_activity-log_list.json
jq '.[] | .properties.message' az_monitor_activity-log_list.json | head
```

Listing 69: Azure CDN Szenario: Orientierung: Logs exportieren

Erfassung Zu Beginn der Erfassung (Listing 70) werden die Ressourcen-Gruppen aufgelistet (1). Anschließend werden die entsprechenden Ressourcen aufgelistet (Abbildung 22) und gesichtet (2).

```
az group list
az resource list --resource-group StaticWebsiteResourceGroup | tee
  ↪  az_resource_list_--resource-group.json | jq '.[] | .type'
```

Listing 70: Azure CDN Szenario: Erfassung: Ressourcen auflisten

```
> jq '.[] | .type ' az_resource_list_--resource-group.json
"Microsoft.Cdn/profiles"
"Microsoft.Cdn/profiles/endpoints"
"Microsoft.Network/dnszones"
"Microsoft.Storage/storageAccounts"
```

Abbildung 22: Azure CDN Szenario: Ressourcenübersicht

Im nächsten Schritt werden Informationen über die Domain und DNS erfasst (Listing 71). Eine Abfrage der Domain ist im Azure CLI-Tool zum aktuellen Zeitpunkt nicht implementiert. Jedoch können mithilfe des CLI-Tool direkte Anfragen an die Azure API gesendet werden (1). Weiter werden alle DNS-Zonen exportiert und deren Namen aufgelistet (2). Anschließend werden die DNS-Einträge der DNS-Zonen gesichert (3).

```
1  az rest --method get --url
   ↪ 'https://management.azure.com/subscriptions/82b3c588-eb34-4a5f-85dd-
   ↪ aafc4c344cba/resourceGroups/StaticWebsiteResourceGroup/providers/
   ↪ Microsoft.DomainRegistration/domains?api-version=2023-01-01' >
   ↪ az_rest_list_domains.json
2  az network dns zone list -g StaticWebsiteResourceGroup | tee
   ↪ az_network_dns_zone_list.json | jq '.[].name'
3  az network dns record-set list -g StaticWebsiteResourceGroup -z
   ↪ myazureproject.com > az_network_dns_record-set_list.json
```

Listing 71: Azure CDN Szenario: Erfassung: Domain und DNS

Im Anschluss wird das CDN Profil und korrespondierender Origin erfasst (Listing 72). Dies beinhaltet 3 Abfragen: alle CDN-Profile (1), alle Endpoints zu einem CDN-Profil (2) und alle Origins zu einem Endpoint (3).

```
1 az cdn profile list -g StaticWebsiteResourceGroup | tee
  ↪ az_cdn_profile_list.json | jq '.[].name'
2 az cdn endpoint list -g StaticWebsiteResourceGroup --profile-name
  ↪ StaticWebsiteCDNProfile | tee az_cdn_endpoint_list.json | jq '.[].name'
3 az cdn origin list -g StaticWebsiteResourceGroup --profile-name
  ↪ StaticWebsiteCDNProfile2 --endpoint-name StaticWebsiteCDNEndpoint >
  ↪ az_cdn_origin_list.json
```

Listing 72: Azure CDN Szenario: Erfassung: CDN Profil

Die Erfassung wird mit dem Objektspeicher abgeschlossen (Listing 73). Zuerst werden alle Storage Accounts aufgelistet und deren Namen angezeigt (1). Um weiter Abfragen zu den Storage Accounts durchführen zu können wird der Account Key benötigt (2). Zum festgestellten Storage Account wird die Logging Konfiguration abgerufen (3). Hier kann Einsicht genommen werden, welche Aktionen (Delete, Read, Write) protokolliert werden und wie lange der Aufbewahrungszeitraum ist. Mit der nächsten Abfrage werden alle Container gelistet (4). Zum festgestellten Container wird eine Inhaltsliste abgefragt (5). Eine grafische Übersicht ist hier nicht notwendig,

da alle Ressourcen aufgelistet werden konnten und nur der Objektspeicher Inhaltsdaten speichert, welche im nächsten Abschnitt gesichert werden.

```
1 az storage account list -g StaticWebsiteResourceGroup | tee
  ↪  az_storage_account_list.json | jq '.[].name'
2 az storage account keys list -g StaticWebsiteResourceGroup -n storageaccount
  ↪  | jq '.[].value'
3 az storage logging show --account-name storageaccount --account-key
  ↪  <account-key>  > az_storage_logging_show.json
4 az storage container list --account-name storageaccount --account-key
  ↪  <account-key> > az_storage_container_list.json
5 az storage blob list --account-name storageaccount --account-key
  ↪  <account-key> --container-name '$web' > az_storage_blob_list.json
```

Listing 73: Azure CDN Szenario: Erfassung: Objektspeicher

Sicherung und Extraktion Zur Sicherung des Objektspeichers (Listing 74) wird der zuvor festgestellte Account Key benötigt. Die Daten aus dem Objektspeicher werden auf den lokalen Computer gesichert (1), in ein Archiv gepackt (2) und zur Integritätssicherung gehasht (3). Bei der Sicherung aus Azure werden die Änderungszeitstempel nicht übernommen. Die Änderungszeitstempel können in der zuvor erfassten Inhaltsliste eingesehen werden. Die Übertragung des Objektspeichers kann mithilfe des Container-Inhaltsverzeichnis überprüft werden. Diese beinhaltet unter anderem auch den MD5-Hashwert jeder Datei.

```
1 az storage blob download-batch -d az_storageaccount_web/ -s '$web'
  ↪  --account-name storageaccount --account-key <account-key>
2 tar czf az_storageaccount_web.tar.gz az_storageaccount_web/
3 sha256sum az_storageaccount_web.tar.gz > az_storageaccount_web.tar.gz.sha256
```

Listing 74: Azure CDN Szenario: Sicherung: Objektspeicher

4.3.6 Google Cloud: CDN

Dieses Szenario wird bei der Google Cloud Platform mithilfe von 3 Komponenten realisiert (Abbildung 23). Die Domain wird über die Google Cloud gebucht und über eine DNS-Zone (1) werden die dazugehörigen DNS-Einträge verwaltet. Ein DNS-Eintrag verweist dabei auf die statische IP des Load-Balancers. Dieser Load-Balancer (2) enthält die CDN-Funktionalität (Caching) und greift auf das SSL-Zertifikat der Website zurück. Mithilfe eines Proxy und URL-Map werden eingehende Anfragen an das Backend gesendet. Dieser wiederum greift auf einen Objektspeicher (3) zurück. Der Proxy nimmt die Anfragen entgegen und mithilfe der URL-Map werden Anfragen auf Backends verteilt. Hier ist nur ein Backend vorhanden, also werden alle Anfragen an dieses weitergeleitet. Der Backend Service übernimmt gleichzeitig auch das Caching der Inhaltsdaten.

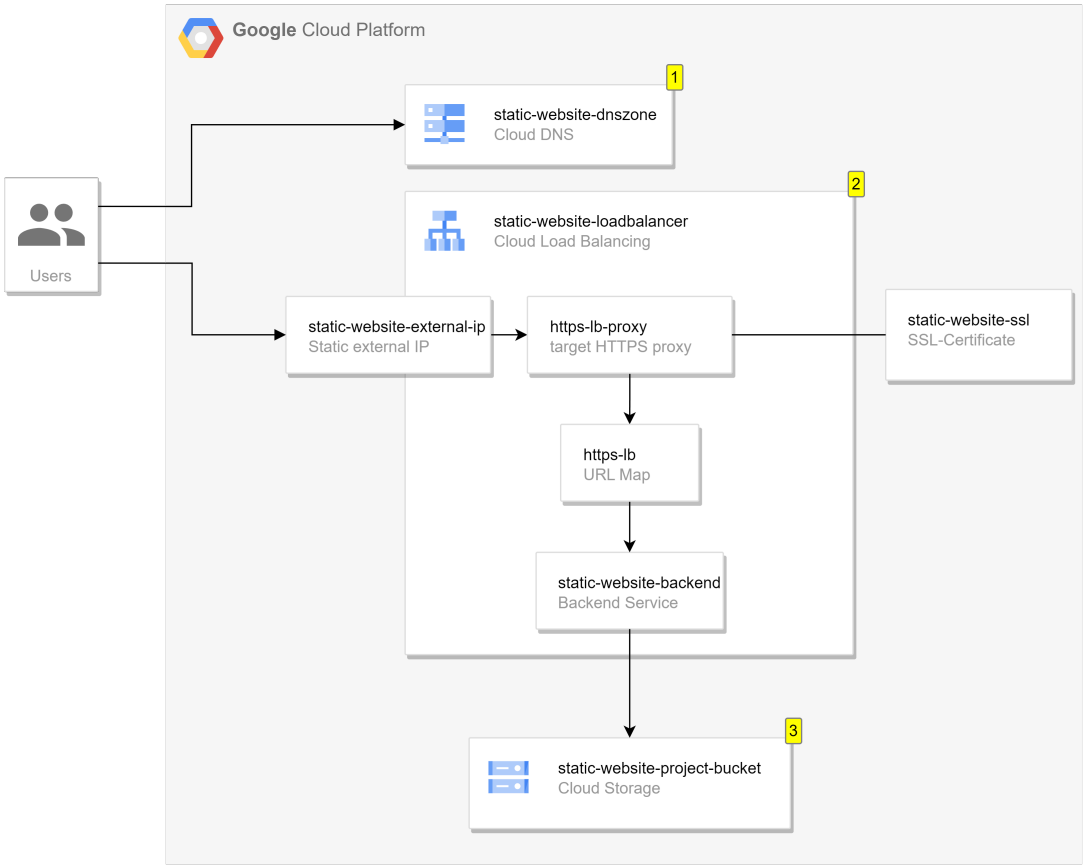


Abbildung 23: Google Cloud CDN Szenario

Aufbau

Als Voraussetzung (Listing 75) wird ein neues Projekt static-website-project erstellt(1), das Rechnungskonto notiert (2) und dieses mit dem Projekt verbunden (3). Zusätzlich müssen noch die zu nutzenden Services aktiviert werden (4,5,6).

```
1 gcloud projects create static-website-project --name="Static Website CDN"
2 gcloud billing accounts list --filter=open=true
3 gcloud billing projects link static-website-project
  ↪ --billing-account=012345-6789AB-CDEFGH
4 gcloud services enable compute.googleapis.com
  ↪ --project=static-website-project
5 gcloud services enable domains.googleapis.com
  ↪ --project=static-website-project
6 gcloud services enable dns.googleapis.com --project=static-website-project
```

Listing 75: Google Cloud CDN Szenario: Voraussetzungen

Zu Beginn wird der Objektspeicher (Listing 76) erstellt (1) und die Inhaltsdaten hochgeladen (2,3). Die Website entspricht derselben wie bei den vorherigen Szenarien (siehe Listing 50). Der Objektspeicher wird noch öffentlich zugänglich gemacht (5) und die index.html Datei als Standarddatei bei einer Anfrage festgelegt (6).

```
1 gcloud storage buckets create gs://static-website-project-bucket-240308
  ↪ --location=us --project=static-website-project
2 gcloud storage cp index.html
  ↪ gs://static-website-project-bucket-240308/index.html
3 gcloud storage cp cat.jpg gs://static-website-project-bucket-240308/cat.jpg
4
5 gcloud storage buckets add-iam-policy-binding
  ↪ gs://static-website-project-bucket-240308 --member=allUsers
  ↪ --role=roles/storage.objectViewer --project=static-website-project
6 gcloud storage buckets update gs://static-website-project-bucket-240308
  ↪ --web-main-page-suffix=index.html --project=static-website-project
```

Listing 76: Google Cloud CDN Szenario: Objektspeicher

Als nächstes wird die Domain gebucht und der korrespondierende DNS-Record eingerichtet (Listing 77). Es wird geprüft, ob der bevorzugte Domainname verfügbar ist (1). Dann wird für diese Domain eine DNS-Zone eingerichtet (2). Diese DNS-Zone verwaltet die DNS-Einträge. Als nächstes wird die Domain unter Angabe der

Kontaktdaten in der `contacts.yaml` Datei gebucht (3). Eine externe IP wird reserviert (5) und diese notiert (6). Unter dieser IP-Adresse ist später der Load-Balancer erreichbar. Die IP-Adresse wird als DNS-Eintrag in der DNS-Zone hinterlegt (7). Abschließend wird noch eine SSL-Policy erstellt (9) und ein SSL-Zertifikat für die Domain erstellt (10). Zur Verifizierung des SSL-Zertifikates muss die Domain zeitnah auf einen Load-Balancer verweisen.

```
1  gcloud domains registrations search-domains mygoogleproject.com
   ↪  --project=static-website-project
2  gcloud dns managed-zones create static-website-dnszone
   ↪  --description="DNS-Zone for mygoogleproject.com"
   ↪  --dns-name=mygoogleproject.com --visibility=public
   ↪  --project=static-website-project
3  gcloud domains registrations register mygoogleproject.com
   ↪  --contact-data-from-file=contacts.yaml
   ↪  --contact-privacy=private-contact-data --yearly-price="7.00 USD"
   ↪  --cloud-dns-zone=static-website-dnszone --quiet
   ↪  --project=static-website-project
4
5  gcloud compute addresses create static-website-external-ip
   ↪  --network-tier=PREMIUM --ip-version=IPV4 --global
   ↪  --project=static-website-project
6  gcloud compute addresses describe static-website-external-ip
   ↪  --format="get(address)" --global --project=static-website-project
7  gcloud dns record-sets create mygoogleproject.com. --rrdatas=34.129.119.163
   ↪  --type=A --ttl=60 --zone=static-website-dnszone
   ↪  --project=static-website-project
8
9  gcloud compute ssl-policies create static-website-ssl-policy
   ↪  --profile=COMPATIBLE --min-tls-version=1.0
   ↪  --project=static-website-project
10 gcloud compute ssl-certificates create mygoogleproject-com-ssl
   ↪  --description="Certificate for mygoogleproject.com"
   ↪  --domains=mygoogleproject.com --global --project=static-website-project
```

Listing 77: Google Cloud CDN Szenario: Domain & DNS

Zum Abschluss des Aufbaus wird der Load-Balancer inklusive CDN eingerichtet (Listing 78). Das Backend wird mit Verweis auf den Objektspeicher erstellt (1). Der Parameter `--enable-cdn` aktiviert dabei das Caching. Als nächstes wird eine URL-Map (2) und Proxy (3) erstellt. Der Proxy greift dabei auf das SSL-Zertifikat der Domain zurück. Abschließend wird der Load-Balancer erstellt (4). Dieser nutzt die

vorher reservierte externe IP-Adresse.

```
1 gcloud compute backend-buckets create static-website-backend
  ↪ --gcs-bucket-name=static-website-project-bucket-240308 --enable-cdn
  ↪ --project=static-website-project
2 gcloud compute url-maps create https-lb
  ↪ --default-backend-bucket=static-website-backend
  ↪ --project=static-website-project
3 gcloud compute target-https-proxies create https-lb-proxy --url-map=https-lb
  ↪ --ssl-certificates=mygoogleproject-com-ssl --global-ssl-certificates
  ↪ --global --ssl-policy=static-website-ssl-policy
  ↪ --project=static-website-project
4 gcloud compute forwarding-rules create static-website-loadbalancer
  ↪ --load-balancing-scheme=EXTERNAL --network-tier=PREMIUM --global
  ↪ --address=static-website-external-ip --target-https-proxy=https-lb-proxy
  ↪ --ports=443 --project=static-website-project
```

Listing 78: Google Cloud CDN Szenario: CDN konfigurieren

Durchführung Leitfaden

Orientierung Im Abschnitt Orientierung wird der eigene Zugang auf Berechtigungen geprüft (Listing 79). Beginnend mit dem aktuell eingeloggten Nutzer (1) werden anschließend alle Projekte aufgelistet (2). Zum Projekt `static-website-project` werden die berechtigten Accounts gesichtet (3). Die verfügbaren Logs werden angezeigt (5) und die Aufbewahrungszeit wird protokolliert (6). Standardmäßig[24] werden Logs in folgende zwei Objektspeicher abgelegt: `_Required` für unter anderem Administratoraktivitäten und Systemereignisse; `_Default` für unter anderem Datenzugriffe. Ersteres hat dabei eine Aufbewahrungszeit von 400 Tagen und kann nicht konfiguriert oder deaktiviert werden. Letzteres eine Aufbewahrungszeit von 30 Tagen und kann konfiguriert, aber nicht deaktiviert werden. Die `_Default` Logs werden exportiert (7) und darauf die `_Required` Logs (8). Die Logs können grob auf Create oder Insert Ereignisse analysiert werden (9).

```
gcloud config list > gcloud_config_list.json
gcloud projects list --format=json | tee gcloud_projects_list.json | jq
  ↪ '.[].projectId'
gcloud projects get-ancestors-iam-policy static-website-project
  ↪ --format=json > gcloud_projects_get-ancestors-iam-policy.json

gcloud logging logs list --project=static-website-project >
  ↪ gcloud_logging_logs_list.json
gcloud logging buckets list --project=static-website-project >
  ↪ gcloud_logging_buckets_list.json
gcloud beta logging read "" --project=static-website-project --format=json
  ↪ --bucket=_Default --location=global --view=_AllLogs --freshness=30d >
  ↪ gcloud-logs_Default.json
gcloud beta logging read "" --project=static-website-project --format=json
  ↪ --bucket=_Required --location=global --view=_AllLogs --freshness=400d >
  ↪ gcloud-logs_Required.json

jq '.[] | .protoPayload | .methodName | select(. !=null)'
  ↪ gcloud-logs_Required.json | grep -E "create|insert"
```

Listing 79: Google Cloud CDN Szenario: Orientierung

Erfassung Die Erfassung beginnt mit der Auflistung aller Ressourcen (Listing 80). Alle Ressourcen werden erfasst und deren Typ ausgegeben (1). Hierbei sind viele Standard-Ressourcen wie virtuelles Netzwerk mit Subnetze vorhanden. Diese können gefiltert werden um eine übersichtliche Ansicht (Abbildung 24) zu erhalten (2).

```
gcloud asset search-all-resources --scope=projects/static-website-project
  ↪ --format=json | tee gcloud_asset_search-all-resources.json | jq '.[] |
  ↪ .assetType'
jq '.[] | select(.displayName | contains("default") | not) |
  ↪ select(.assetType | contains("serviceusage") | not) .assetType'
  ↪ gcloud_asset_search-all-resources.json
```

Listing 80: Google Cloud CDN Szenario: Erfassung: Ressourcen

```
> jq '.[] | select(.displayName | contains("default") | not) | select(.assetType |
  contains("serviceusage") | not) .assetType' gcloud-resources.json
"compute.googleapis.com/ForwardingRule"
"compute.googleapis.com/TargetHttpsProxy"
"compute.googleapis.com/UrlMap"
"compute.googleapis.com/BackendBucket"
"compute.googleapis.com/SslCertificate"
"compute.googleapis.com/Address"
"compute.googleapis.com/SslPolicy"
"domains.googleapis.com/Registration"
"dns.googleapis.com/ManagedZone"
"compute.googleapis.com/Project"
"storage.googleapis.com/Bucket"
"cloudbilling.googleapis.com/ProjectBillingInfo"
"cloudresourcemanager.googleapis.com/Project"
"logging.googleapis.com/LogSink"
"logging.googleapis.com/LogSink"
"logging.googleapis.com/LogBucket"
"logging.googleapis.com/LogBucket"
```

Abbildung 24: Google Cloud CDN Szenario: Ressourcenübersicht

Als nächstes wird die Domain und DNS erfasst (Listing 81). Angefangen mit der Domain (1) wird anschließend die DNS-Zone erfasst (2). Zur DNS-Zone werden die DNS-Einträge (3) und Historie (4) erfasst.

```
1 gcloud domains registrations list --format=json
  ↪ --project=static-website-project >
  ↪ gcloud_domains_registrations_list.json
2 gcloud dns managed-zones list --format=json --project=static-website-project
  ↪ | tee gcloud_dns_managed-zones_list.json | jq '.[].name'
3 gcloud dns record-sets list --zone=static-website-dnszone --format=json
  ↪ --project=static-website-project > gcloud_dns_record-sets_list.json
4 gcloud dns record-sets changes list --zone=static-website-dnszone
  ↪ --format=json --project=static-website-project >
  ↪ gcloud_dns_record-sets_changes_list.json
```

Listing 81: Google Cloud CDN Szenario: Erfassung: Domain & DNS

Anschließend wird der Load-Balancer inkl. Bestandteile erfasst (Listing 82): Der Load-Balancer an sich (1), der Proxy (2), die URL-Map (3) sowie das Backend (4). Zusätzlich wird das SSL-Zertifikat mit Erstellungs- und Ablaufzeitstempel erfasst (5). Die erfasste externe IP-Adresse (6) enthält einen Verweis auf den Load-Balancer.

```
gcloud compute forwarding-rules list --format=json
  ↪ --project=static-website-project  >
  ↪ gcloud_compute_forwarding-rules_list.json
gcloud compute target-https-proxies list --format=json
  ↪ --project=static-website-project  >
  ↪ gcloud_compute_target-https-proxies_list.json
gcloud compute url-maps list --format=json --project=static-website-project
  ↪ > gcloud_compute_url-maps_list.json
gcloud compute backend-buckets list --format=json
  ↪ --project=static-website-project  >
  ↪ gcloud_compute_backend-buckets_list.json
gcloud compute ssl-certificates list --format=json
  ↪ --project=static-website-project  >
  ↪ gcloud_compute_ssl-certificates_list.json
gcloud compute addresses list --format=json --project=static-website-project
  ↪ > gcloud_compute_addresses_list.json
```

Listing 82: Google Cloud CDN Szenario: Erfassung: CDN

Die Erfassung wird mit dem Objektspeicher abgeschlossen (Listing 83). Alle Objektspeicher werden erfasst und deren Namen ausgegeben (1). Die Berechtigungen des Buckets werden erfasst (2) und anschließend wird eine Inhaltsliste erstellt (3). Der Parameter `--all-versions` erwirkt, dass alle Versionen der Dateien aufgelistet werden, falls eine Versionierung aktiviert ist. Eine grafische Übersicht ist hier nicht notwendig, da alle Ressourcen aufgelistet werden konnten und nur der Objektspeicher Inhaltsdaten speichert, welche im nächsten Abschnitt gesichert werden.

```
gcloud storage buckets list --format=json --project=static-website-project |
  ↪ tee gcloud_storage_buckets_list.json | jq '.[].name'
gcloud storage buckets get-iam-policy
  ↪ gs://static-website-project-bucket-240308 --format=json
  ↪ --project=static-website-project >
  ↪ gcloud_storage_buckets_get-iam-policy.json
gcloud storage ls --recursive --all-versions --json
  ↪ gs://static-website-project-bucket-240308
  ↪ --project=static-website-project >
  ↪ gcloud_storage_ls_--recursive_--all-versions.json
```

Listing 83: Google Cloud CDN Szenario: Erfassung: Objektspeicher

Sicherung und Extraktion Die einzigen Inhaltsdaten sind im Objektspeicher gespeichert. Dieser wird folgendermaßen gesichert (Listing 84). Die Daten aus dem

Objektspeicher auf den lokalen Computer gesichert (1), in ein Archiv gepackt (2) und zur Integritätssicherung gehasht (3). Bei der Sicherung bleiben die Änderungszeitstempel erhalten. Diese sind zusätzlich in der Erfassung hinterlegt. Die Übertragung des Objektspeichers kann mithilfe des Objektspeicher-Inhaltsverzeichnis überprüft werden. Diese beinhaltet unter anderem auch den MD5-Hashwert jeder Datei.

```
1 gcloud storage rsync gs://static-website-project-bucket-240308
  ↪ static-website-project-bucket-240308 --recursive
2 tar czf static-website-project-bucket-240308.tar.gz
  ↪ static-website-project-bucket-240308/
3 sha256sum static-website-project-bucket-240308.tar.gz >
  ↪ static-website-project-bucket-240308.tar.gz.sha256
```

Listing 84: Google Cloud CDN Szenario: Sicherung und Extraktion

4.4 Zusammenfassung

Die größte Hürde bei der Durchführung war die Erfassung der genutzten Ressourcen beim Anbieter AWS. Dieser bietet standardmäßig keine direkte Möglichkeit alle Ressourcen aufzulisten. Dies muss über Umwege, wie z. B. den Tag-Editor oder die Abrechnungsübersicht, erfolgen. Weiter war beim Anbieter Azure keine forensische virtuelle Maschine notwendig, da der erstellte Snapshot für einen Download freigegeben werden kann.
Differenzen haben sich auch bei den jeweiligen Objektspeichern gezeigt. Azure und Google Cloud berechnen und speichern für jede Datei einen MD5-Hash ab. Dieser kann zur Integritätsprüfung genutzt werden. Der Änderungszeitstempel bleibt bei der Synchronisierung aus AWS und Google Cloud erhalten. Bei einer Übertragung aus Azure werden die Änderungszeitstempel auf den Downloadzeitpunkt gesetzt. Der Änderungszeitstempel ist in der Objektspeicherübersicht bei den genutzten CSP vorhanden.
Bei allen drei Anbietern gibt es unterschiedliche Möglichkeiten von einer virtuellen Maschine auf einen Objektspeicher zuzugreifen. Als konsistente Methode hat sich die Generierung von pre-signed URLs herauskristallisiert. Hierbei müssen an einer VM keine Berechtigungen geändert werden. Zusätzlich muss sich der Objektspeicher nicht im untersuchten Benutzerkonto und nicht einmal beim selben Cloud-Anbieter befinden. Hier kann in Betracht gezogen werden, unabhängig von einer einzelnen Untersuchung, einen allgemeinen Objektspeicher für forensische Zwecke vorzuhalten.
Bei keinem Anbieter ist die Sicherung des Arbeitsspeichers ohne Eingreifen in die virtuelle Maschine möglich. Beispielsweise kann mithilfe des AWS Systems Manager[13] eine automatisierte Sicherung ausgelöst werden. Diese nutzt die identischen Tools und Methoden[7] zur Arbeitsspeichersicherung wie bei der Durchführung vorgestellt. Die Sicherung erfolgt im Unterschied dazu automatisiert. AWS beschreibt diesen Workflow als nicht gerichtsfest[8].
Alle Dienste konnten mit einem einheitlichem Ansatz erfasst werden. Die CLI-Tools der Anbieter stellen Funktionen wie `describe`, `get`, `list` und `show` bereit. Mit diesen konnten die Metadaten der Ressourcen detailliert erfasst werden. Im Gegensatz dazu muss bei der Sicherung von Inhaltsdaten der oder die Forensiker:in aus einer Auswahl an Methoden die Passende wählen. Exemplarisch kann eine Datenbank-Instanz entweder per Hand mit einem Datenbank-Client gesichert werden (AWS), einen Datenbank-Export anstoßen (Azure) oder die Datenbanken einzeln exportieren (Google Cloud).

5 Fazit und Ausblick

5.1 Fazit

Das Ziel dieser Master-Thesis war die Erstellung eines Leitfadens zur Identifizierung und Sicherung von Artefakten aus Infrastructure-as-a-Service Plattformen. Dazu wurde in Kapitel 1 der Einsatzzweck und Abgrenzung definiert.
In Kapitel 2 wurden die benötigten Grundlagen erläutert. Die theoretischen und praktischen Grundlagen zu Cloud Computing verdeutlichen, dass bei der Nutzung von Cloud Computing sowohl Verantwortung als auch Kontrolle an den Cloud-Provider abgegeben wird. IaaS Plattformen überlassen, im Vergleich zu anderen Cloud Computing Modellen, dem Cloud-Kunden am meisten Kontrolle und bieten Vorteile[25] wie Skalierbarkeit, Flexibilität, Kosteneinsparung und Datensicherung.
Die Erstellung des Leitfadens fand in Kapitel 3 statt. Dazu wurden bestehende Frameworks erläutert und mehrere Herausforderungen in der Cloud-Forensik hervorgehoben. Die Herausforderungen der Protokollerfassung, Live-Forensik und Beweisintegrität werden vom Leitfaden behandelt. Eine Protokollerfassung findet mitunter zu Beginn der Cloud-Forensik Phase statt. Die erfassten Protokolle ermöglichen einen Sichtung der letzten Aktivitäten und sind das Grundgerüst zur Erstellung einer Timeline. Aspekte der Live-Forensik in einer Cloud-Umgebung wurden erläutert und Differenzen zur klassischen Live-Forensik aufgezeigt. Diese begründen sich durch die Virtualisierung und Zugriffsmöglichkeiten. Die Beweisintegrität kann mithilfe von Hashwertabgleichen sowohl beim Export aus der Cloud als auch langfristig sichergestellt werden. Allgemein gilt immer noch der Grundsatz “Nicht mit Original-Daten arbeiten”[20].
Verschiedene Lösungen zur Herausforderung der Datensicherung wurden ausführlich im Leitfaden diskutiert. Abhängig vom CSP werden Exportfunktionen für Inhaltsdaten angeboten. Existieren diese nicht, so muss ein Umweg über eine forensische VM gegangen werden. Dieser Umweg bringt den Vorteil mit sich, direkt einen Hashwert für eine Integritätsprüfung zu berechnen. Offen bleiben die Herausforderungen der Datenherkunft und Datensemantik. Um die Datenherkunft zu bestimmen, müssen mehrere Artefakt untersucht werden. Abhängig von der Protokollierung, Versionie-

rung und deren Aufbewahrungszeit kann eine Datenherkunft aufgedeckt werden. Für die Bestimmung der Datensemantik in einer folgenden Analyse wird durch die erfassten und gesicherten Daten eine solides Fundament gegeben. Interaktionen zwischen Cloud-Ressourcen können anhand der erfassten Metadaten nicht zweifelsfrei festgestellt werden. Dazu müssen die gesicherten Inhaltsdaten in einer Analyse untersucht werden.
Der Leitfaden wurde im Kapitel 4 anhand zwei Szenarien bei drei CSP evaluiert. Aufgrund der Anzahl an unterschiedlicher Cloud Ressourcen, Abhängigkeiten und Interaktionen untereinander resultieren schon bei überschaubaren Szenarien viele Artefakte. Bei der Evaluierung konnte man beobachten, dass die Erfassung bei den drei CSP identisch abläuft. Mithilfe von Aufrufen wie `describe`, `get`, `list` und `show` können Ressourcen erfasst werden. Bei der Sicherung der Ressourcen wurden Differenzen zwischen den gewählten CSP sichtbar. Je nach CSP und zu sichernde Ressource werden Exportfunktionen angeboten. Sind keine Exportfunktionen vorhanden, so kann über einen Umweg (forensische VM) eine Sicherung erfolgen.

5.2 Ausblick

Mit der Identifizierung und Sicherung von Artefakte aus IaaS Plattformen wird ein Fundament für eine anstehende Analyse gelegt. Dabei fließt in die Analyse die klassischen Artefakte wie Arbeits- und Datenspeicher ein. Zusätzlich fließen Cloud-spezifische Artefakte wie virtuelle Netzwerke, Sicherheitsgruppen und API-Protokolle ein. Diese beiden Artefaktgruppen müssen zusammen betrachtet werden um Datenherkunft und Datensemantik bestimmen zu können. In die Erstellung der Timeline fließen mehrere Quellen ein. Die Protokollierung der API-Aufrufe bildet ein Grundverständnis, welche Ressourcen und Dienste wann erstellt, geändert und ggf. gelöscht wurden. Abhängig vom Ziel einer forensischen Untersuchung ist es notwendig die Architektur einer Cloud-Umgebung festzustellen. Dazu sind die erfassten Metadaten ein wichtiger Bestandteil. Diese enthalten Informationen über die jeweilige Ressource sowie deren Abhängigkeiten. Interaktionen zwischen Ressourcen können allein dadurch nicht immer festgestellt werden. Hierzu sind Artefakte aus einer Arbeitsspeicher- und Datenspeicheruntersuchung notwendig.
Der Leitfaden ermächtigt, in Verbindung mit der praktischen Anwendung in der Evaluierung, eine:n Forensiker:in die forensische Untersuchung auf eine IaaS-Plattform auszuweiten. Mit den gesicherten Artefakten wird ein solides Fundament für weitere Untersuchungen gegeben.

Literaturverzeichnis

[1] Sandesh Achar. „Cloud Computing Forensics“. In: *International Journal of Computer Engineering and Technology* 13.3 (2022).

[2] Igor Akhmetov. *How to get a memory dump of a virtual machine from its hypervisor*. 2023. URL: https://forum.kaspersky.com/topic/how-to-get-a-memory-dump-of-a-virtual-machine-from-its-hypervisor-36407/ (besucht am 24. 03. 2024).

[3] M Edington Alex und R Kishore. „Forensics framework for cloud computing“. In: *Computers & Electrical Engineering* 60 (2017), S. 193–205.

[4] Syed Ahmed Ali, Shahzad Memon und Farhan Sahito. „Challenges and solutions in cloud forensics“. In: *Proceedings of the 2018 2nd International Conference on Cloud and Big Data Computing*. 2018, S. 6–10.

[5] Saad Alqahtany u. a. „Cloud forensics: a review of challenges, solutions and open problems“. In: *2015 international conference on cloud computing (ICCC)*. IEEE. 2015, S. 1–9.

[6] Amazon. *EC2 customers*. 2024. URL: https://aws.amazon.com/de/ec2/customers/ (besucht am 24. 03. 2024).

[7] AWS. *Automated Forensics Orchestrator for Amazon EC2*. 2023. URL: https://github.com/aws-solutions/automated-forensic-orchestrator-for-amazon-ec2 (besucht am 24. 03. 2024).

[8] AWS. *Automated Forensics Orchestrator for Amazon EC2*. 2023. URL: https://docs.aws.amazon.com/solutions/latest/automated-forensics-orchestrator-for-amazon-ec2/welcome.html (besucht am 24. 03. 2024).

[9] AWS. *Accessing AWS services*. 2024. URL: https://docs.aws.amazon.com/whitepapers/latest/aws-overview/accessing-aws-services.html (besucht am 24. 03. 2024).

[10] AWS. *AWS Command Line Interface User Guide for Version 2*. 2024. URL: https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html (besucht am 24. 03. 2024).

[11] AWS. *IAM JSON policy elements reference*. 2024. URL: https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements.html (besucht am 24. 03. 2024).

[12] AWS. *Subnet CIDR blocks*. 2024. URL: https://docs.aws.amazon.com/vpc/latest/userguide/subnet-sizing.html (besucht am 24. 03. 2024).

[13] AWS. *Was ist AWS Systems Manager?* 2024. URL: https://docs.aws.amazon.com/de_de/systems-manager/latest/userguide/what-is-systems-manager.html (besucht am 19. 03. 2024).

[14] Azure. *Strategie zur Cloudüberwachung*. 2024. URL: https://learn.microsoft.com/de-de/azure/cloud-adoption-framework/strategy/monitoring-strategy (besucht am 24. 03. 2024).

[15] Microsoft Azure. *Connect to Azure Database for MySQL - Flexible Server with encrypted connections*. 2024. URL: https://learn.microsoft.com/en-us/azure/mysql/flexible-server/how-to-connect-tls-ssl (besucht am 24. 03. 2024).

[16] Microsoft Azure. *Create and manage virtual networks for Azure Database for MySQL - Flexible Server*. 2024. URL: https://learn.microsoft.com/en-us/azure/mysql/flexible-server/how-to-manage-virtual-network-cli (besucht am 24. 03. 2024).

[17] Microsoft Azure. *Dokumentation zur Azure-Befehlszeilenschnittstelle*. 2024. URL: https://learn.microsoft.com/de-de/cli/azure/ (besucht am 24. 03. 2024).

[18] Dominik Birk, Dennis Heinson und Christoph Wegener. „Virtuelle Spurensuche: Digitale Forensik in Cloud-Umgebungen“. In: *Datenschutz und Datensicherheit-DuD* 35.5 (2011), S. 329–332.

[19] Giulia Borgoni. *The Evolution of Cloud Computing: From the Early Ideas of John McCarthy to Modern Platforms*. 2023. URL: https://www.elemento.cloud/post/the-evolution-of-cloud-computing-from-the-early-ideas-of-john-mccarthy-to-modern-platforms (besucht am 19. 03. 2024).

[20] Dominique Brezinski und Tom Killalea. *RFC3227: Guidelines for Evidence Collection and Archiving*. 2002.

[21] Google Cloud. *Aktivierte Dienste*. 2024. URL: https://cloud.google.com/service-usage/docs/enabled-service?hl=de (besucht am 24. 03. 2024).

[22] Google Cloud. *Cloud SQL - Instanzen erstellen*. 2024. URL: https://cloud.google.com/sql/docs/mysql/create-instance (besucht am 24. 03. 2024).

[23] Google Cloud. *gcloud CLI – Übersicht*. 2024. URL: https://cloud.google.com/sdk/gcloud (besucht am 24. 03. 2024).

[24] Google Cloud. *Routing und Speicher – Übersicht*. 2024. URL: https://cloud.google.com/logging/docs/routing/overview (besucht am 24. 03. 2024).

[25] Google Cloud. *Vor- und Nachteile von Cloud-Computing*. 2024. URL: https://cloud.google.com/learn/advantages-of-cloud-computing (besucht am 24. 03. 2024).

[26] Suchana Datta, Koushik Majumder und Debashis De. „Review on cloud forensics: an open discussion on challenges and capabilities“. In: *International Journal of Computer Application* 145.1 (2016), S. 1–8.

[27] Josiah Dykstra und Damien Riehl. „Forensic collection of electronic evidence from infrastructure-as-a-service cloud computing“. In: *Rich. JL & Tech.* 19 (2012), S. 1.

[28] Josiah Dykstra und Alan T Sherman. „Acquiring forensic evidence from infrastructure-as-a-service cloud computing: Exploring and evaluating tools, trust, and techniques“. In: *Digital Investigation* 9 (2012), S90–S98.

[29] Random Forenst. *Cloud Resume Challenge*. 2024. URL: `https://cloudresumechallenge.dev/docs/the-challenge` (besucht am 24. 03. 2024).

[30] Dan Goodin. *Zeus bot found using Amazon's EC2 as C&C server*. 2009. URL: `https://www.theregister.com/2009/12/09/amazon_ec2_bot_control_channel/` (besucht am 24. 03. 2024).

[31] Google. *Cloud Forensics Utils*. 2024. URL: `https://github.com/google/cloud-forensics-utils` (besucht am 24. 03. 2024).

[32] Google. *Cloud-Kunden*. 2024. URL: `https://cloud.google.com/customers` (besucht am 24. 03. 2024).

[33] Red Hat. *Was sind virtuelle Maschinen (VM) und wie funktionieren sie?* 2024. URL: `https://www.redhat.com/de/topics/virtualization/what-is-a-virtual-machine` (besucht am 24. 03. 2024).

[34] Ben Reader (Powers Hell). *Calculate & Validate MD5 hashes on Azure blob storage files with PowerShell*. 2021. URL: `https://powers-hell.com/2021/12/31/calculate-validate-md5-hashes-on-azure-blob-storage-files-with-powershell/` (besucht am 24. 03. 2024).

[35] Martin Herman u. a. *Nist cloud computing forensic science challenges*. US Department of Commerce, National Institute of Standards und Technology, 2020. URL: `https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8006.pdf`.

[36] Karen Kent, Suzanne Chevalier und Tim Grance. „Guide to integrating forensic techniques into incident“. In: *Tech. Rep. 800-86* (2006).

[37] Rongxing Lu u. a. „Secure provenance: the essential of bread and butter of data forensics in cloud computing“. In: *Proceedings of the 5th ACM symposium on information, computer and communications security*. 2010, S. 282–292.

[38] Ben Martini und Kim-Kwang Raymond Choo. „An integrated conceptual digital forensic framework for cloud computing“. In: *Digital investigation* 9.2 (2012), S. 71–80.

[39] Ben Martini und Kim-Kwang Raymond Choo. „Remote programmatic vCloud forensics: a six-step collection process and a proof of concept“. In: *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE. 2014, S. 935–942.

[40] Rodney McKemmish. *What is forensic computing?* Australian Institute of Criminology Canberra, 1999.

[41] Peter Mell, Tim Grance u. a. „The NIST definition of cloud computing“. In: (2011).

[42] Microsoft. *Customer Stories*. 2024. URL: https://customers.microsoft.com/ (besucht am 24. 03. 2024).

[43] Microsoft. *Datacenters*. 2024. URL: https://datacenters.microsoft.com/globe/ (besucht am 24. 03. 2024).

[44] Stephen O'shaughnessy und Anthony Keane. „Impact of cloud computing on digital forensic investigations". In: *Advances in Digital Forensics IX: 9th IFIP WG 11.9 International Conference on Digital Forensics, Orlando, FL, USA, January 28-30, 2013, Revised Selected Papers 9*. Springer. 2013, S. 291–303.

[45] Ameer Pichan, Mihai Lazarescu und Sie Teng Soh. „Cloud forensics: Technical challenges, solutions and comparative analysis". In: *Digital investigation* 13 (2015), S. 38–57.

[46] Jonathon Poling. „Instance memory acquisition techniques for effective incident response". AWS re:Inforce https://www.youtube.com/watch?v=qdYRNuRy-E4&t=484s. 2022. URL: https://d1.awsstatic.com/events/aws-reinforce-2022/TDR401_Instance-memory-acquisition-techniques-for-effective-incident-response.pdf.

[47] Vijay Prakash u. a. „Cloud-Based Framework for Performing Digital Forensic Investigations". In: *International Journal of Wireless Information Networks* (2022), S. 1–23.

[48] Keyun Ruan u. a. „Cloud forensics". In: *Advances in Digital Forensics VII: 7th IFIP WG 11.9 International Conference on Digital Forensics, Orlando, FL, USA, January 31–February 2, 2011, Revised Selected Papers 7*. Springer. 2011, S. 35–46.

[49] Bundesamt für Sicherheit in der Informationstechnik. *IT-Grundschutz-Kompendium*. 2023.

[50] Deutschland Bundesamt für Sicherheit in der Informationstechnik. *Leitfaden ÏT-Forensik": Version 1.0.1 (März 2011)*. Bundesamt für Sicherheit in der Informationstechnik - BSI, 2010. URL: https://books.google.de/books?id=WOe8uQEACAAJ.

[51] Statista. *Worldwide market share of leading cloud infrastructure service providers in Q4 2023*. 2024. URL: https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/ (besucht am 24. 03. 2024).

[52] Broadcom - vmware. *Virtuelle Maschine*. 2024. URL: https://www.vmware.com/de/topics/glossary/content/virtual-machine.html (besucht am 24. 03. 2024).

[53] S Wolthusen. *Overcast: Forensic discovery in cloud environments," in proceedings of the Fifth International Conference on IT Security Incident Management and IT Forensics (IMF)*. 2009.

Abbildungsverzeichnis

Quellcodeverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface. 13, 18, 21, 22, 24, 51, 77, 97, 106, *Glossar:* Application Programming Interface
ARN	Amazon Resource Name. 39, 78, 106, *Glossar:* Amazon Resource Name
CDN	Content Delivery Network. 32, 81, 87, 106, *Glossar:* Content Delivery Network
CLI	Command Line Interface. 13–15, 22, 24, 95, 103, 106, *Glossar:* Command Line Interface
CSP	Cloud Service Provider. 7, 17–19, 25, 27, 29–31, 34, 47, 58, 81, 96, 97, 106, *Glossar:* Cloud Service Provider
EC2	Elastic Compute Cloud. 31, 34, 40, 106, *Glossar:* Elastic Compute Cloud
ETag	Entity Tag. 73, 106, *Glossar:* Entity Tag
IaaS	Infrastructure-as-a-Service. 6, 7, 12, 31, 96, 97, 106, *Glossar:* Infrastructure-as-a-Service
JSON	JavaScript Object Notation. 14, 24, 106, *Glossar:* JavaScript Object Notation
MD5	Message-Digest Algorithm 5. 30, 56, 86, 95, 106, *Glossar:* Message-Digest Algorithm 5
NIC	Network Interface Card. 47, 106, *Glossar:* Network Interface Card
NSG	Network Security Group. 47, 106, *Glossar:* Network Security Group
RDP	Remote Desktop Protocol. 29, 106, *Glossar:* Remote Desktop Protocol
RDS	Relational Database Service. 34, 40, 106, *Glossar:* Relational Database Service
S3	Simple Storage Service. 40, 71, 106, *Glossar:* Simple Storage Service
SAS	Shared Access Signature. 54, 56, 106, *Glossar:* Shared Access Signature
SDK	Software Development Kit. 24, 106, *Glossar:* Software Development Kit
SSH	Secure Shell. 29, 48, 49, 61, 106, *Glossar:* Secure Shell
VM	Virtuelle Maschine. 10, 15, 16, 20, 26–30, 34, 58, 95, 96, 106, *Glossar:* Virtuelle Maschine
VPC	Virtual Private Cloud. 34, 58, 106, *Glossar:* Virtual Private Cloud

Glossar

Amazon Resource Name	eindeutige Bezeichnung einer Ressource beim Provider AWS. 39, 106
Application Programming Interface	Schnittstelle zum Austausch von Informationen. Meistens wird diese Schnittstelle von Programmen aufgerufen. 13, 106
Cloud Service Provider	Anbieter einer IaaS Plattform. 7, 106
Cloud-Shell	ist die Möglichkeit über das Webinterface von IaaS Provider die jeweiligen CLI-Werkzeuge zu benutzen. 13, 24, 33
Command Line Interface	textbasierte Computerschnittstelle zum Ausführen von Befehlen. 13, 106
Content Delivery Network	Netzwerk aus global verteilten Servern zur beschleunigten Auslieferung von Inhalten. 32, 106
Elastic Compute Cloud	Angebot von AWS zur Erstellung, Verwaltung und Ausführung von VMs. 31, 106
Entity Tag	Hashwert eines Objekts. 73, 106
Hashwert	numerischer Wert, meistens in hexadezimaler Schreibweise, der durch Berechnung eines Inputs mittels Hash-Algorithmus entsteht. Hash-Algorithmen sind beispielsweise MD5, SHA-256, SHA-512 . 28, 29, 44, 45, 55, 56, 67, 86, 94, 96
Hashwertabgleich	Integritätsprüfung die mittels Vergleich eines übermittelten Hashwerts mit dem berechnet Hashwerts erfolgt. 28, 30, 46, 56, 70, 96
Hypervisor	Software zur Erstellung, Verwaltung und Ausführen von virtuellen Maschinen. Ressourcen werden vom Hypervisor den VMs zugeteilt. 10, 28
Infrastructure-as-a-Service	Der Anbieter stellt die Infrastruktur bereit. Der Kunde entscheidet selbst über über Betriebssystem und Anwendung.. 6, 106

JavaScript Object Notation	menschen- und maschinenlesbares Datenaustauschformat. 14, 106
Live-Forensik	Untersuchung während eines Vorfalls mit Fokus auf flüchtige Daten. 9, 19, 24, 26, 29, 96
Load-Balancer	Soft- oder Hardware zur Verteilung von Rechen- oder Netzwerklast auf mehrere Server. 32, 87, 89
Message-Digest Algorithm 5	Hashalgorithmus. 30, 106
Network Interface Card	Speziell für Azure: Netzwerkanbindung einer VM an ein virtuelles Netzwerk oder das Internet. 47, 106
Network Security Group	Speziell für Azure: enthält Sicherheitsregeln zur Regelung von ein- und ausgehenden Netzwerkverbindungen. Vergleichbar mit einer Firewall. 47, 106
Post-mortem-Analyse	Untersuchung von nichtflüchtigen Daten nach einem Vorfall. 9, 24
pre-signed URL	Im Kontext von Objektspeicher ermöglicht eine pre-signed URL einen zeitlich beschränkten Lese- und/oder Schreibzugriff auf ein vordefiniertes Objekt. 43, 54, 68, 95
Relational Database Service	Angebot von AWS zur Erstellung, Verwaltung und Betreiben einer relationaler Datenbank. 34, 106
Remote Desktop Protocol	Protokoll für den Fernzugriff auf entfernte Computer. 29, 106
Secure Shell	verschlüsselte Netzwerkverbindung zu einem entfernten Computer, meist wird darüber eine Kommandozeile beim entfernten Computer benutzt. 29, 106
Shared Access Signature	Speziell für Azure: Zugriffstoken für Speicherkonten (Objektspeicher). 54, 106
Simple Storage Service	Angebot von AWS zur Verwaltung von Objektspeichern. 40, 106
Snapshot	Zustandsaufnahme eines Datenträgers zum Zeitpunkt der Aufnahme. 27, 42, 66, 95
Software Development Kit	Programmbibliotheken zur Einbindung von zusätzlicher Funktionalität. 24, 106

Timeline	zeitliche Auflistung wichtiger Ereignisse eines Vorfalls. 19, 20, 29, 96, 97
Virtual Private Cloud	Virtuelles Netzwerk, meistens bestehend aus Subnetze, Routing-Tabellen und Firewalls. 34, 106
Virtuelle Maschine	mithilfe eines Hypervisor ausgeführtes virtuelles Betriebssystem, das auf zugeteilte Ressourcen zurückgreift. 10, 106

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die anderen Quellen im Wortlaut oder dem Sinn nach entnommen wurden, sind durch Angaben der Herkunft kenntlich gemacht. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet.

Ich erkläre ferner, dass ich die vorliegende Arbeit in keinem anderen Prüfungsverfahren als Prüfungsarbeit eingereicht habe oder einreichen werde.

Die eingereichte schriftliche Arbeit entspricht der elektronischen Fassung. Ich stimme zu, dass eine elektronische Kopie gefertigt und gespeichert werden darf, um eine Überprüfung mittels Anti-Plagiatssoftware zu ermöglichen.

Mainz, 26.03.2024

Simon Heinrich Bauer

Thesen

Master-Thesis
Identifizierung und Sicherung von Artefakten in Infrastructure-as-a-Service Plattformen

Eingereicht am:	26. März 2024
von:	Simon Heinrich Bauer
Betreuer:	Prof. Dr. Antje Raab-Düsterhöft
Zweitbetreuer:	Prof. Dr. Nils Gruschka

- Cybercrime wird in Zukunft eine größere Rolle im Zusammenhang mit Cloud Computing spielen
- Der Leitfaden ersetzt keine eigene Einarbeitung in die Thematik Cloud Computing
- Mithilfe des Leitfadens kann ein solides Fundament für weitere Untersuchungen gebildet werden
- Eine forensische VM ist vielseitig einsetzbar zur Sicherung von Inhaltsdaten