Virtualisierungstechnologie

Theoretische Ausarbeitung

Gruppe 01

Harald Beier

Susanne Peer

Patrick Prugger

Inhalt

Übung 1: Azure Resource Types:

Erstelle eine Liste von mindestens 5 Azure Resource Typen und beschreibe diese im Detail. Was machen diese Ressourcen? Für was können diese Ressourcen eingesetzt? Welche Vor- und Nachteile haben diese Ressourcen?

1) **Azure Virtual Machines (VMs) -** ermöglicht es Usern virtuelle Server in der Cloud zu erstellen, die versch. Betriebssysteme wie Windows oder Linux unterstützen. Bspw. können Unternehmen Webanwendungen hosten, Entwicklungs- und Testumgebungen erstellen oder Datenbanken betreiben. Ein Vorteil von Azure VMs ist die Skalierbarkeit, da Ressourcen dynamisch je nach Bedarf angepasst werden können und diese Maschinen in andere Azure-Dienste wie Azure Monitor oder Azure Backup integrieren, was die Verwaltung erleichtert. VMs können kostenintensiv sein, insbesondere bei kontinuierlichem Betrieb und zudem erfordern sie Verwaltungsaufwand, wie regelmäßige Updates und Patches (azure.microsoft.com, 2025).

2) **Azure Blob Storage -** skalierbarer und kosteneffizienter Object Storage für unstrukturierte Daten, eignet sich für die Speicherung großer Datenmengen wie Backups, Videos oder Big-Data-Dateien. Je nach Anwendungsfall können User zwischen verschiedenen Speicher-Tiers wählen, wie dem Hot-Tier für häufig genutzte Daten oder dem Archive-Tier für selten genutzte Daten. Es lässt sich problemlos in andere Azure-Dienste integrieren und unterstützt den Zugriff über APIs und SDKs. Einschränkungen, wie etwa die geringere Abfragegeschwindigkeit im Archiv-Tier und die fehlende Unterstützung für eine native Verzeichnisstruktur, kann Organisation von Daten erschweren (azure.microsoft.com, 2025).

3) **Azure Monitor** -Überwachungsdienst, Welcher Metriken, Logs und Diagnosedaten von versch. Quellen innerhalb von Azure sammelt, analysiert und visualisiert, dazu gehören nicht nur Ressourcen, sondern auch Anwendungen und Betriebssysteme. Azure Monitoring bietet verschiedene Integrationen wie Dashboards, Log Analytics oder Power BI – um gesammelte Daten übersichtlich darzustellen. Die anfängliche Konfiguration kann komplex sein und die Kosten können durch große Umgebungen oder Datenmengen rasant steigen. Es bietet den Vorteil einer zentralen Überwachung komplexer Systeme, die Skalierbarkeit und die nahtlose Intregration mit anderen Azure Diensten ermöglicht (azure.microsoft.com, 2025)

4) **Defender for Cloud (früher Azure Security Center)** - ein Sicherheitsdienst, der hybride und cloudnative Workloads schützt. Er bietet präventive und reaktive Lösungen wie Schwachstellenbewertungen, Echtzeit-Bedrohungserkennung mit KI und Compliance-Überwachung. Zu den Vorteilen gehören umfassender Schutz, Automatisierung von Sicherheitsprozessen und fortschrittliche

Analysen. Allerdings kann die Integration komplex sein, und erweiterte Funktionen sind in großen Umgebungen oft kostenintensiv (azure.microsoft.com, 2025).

5) **Azure Arc** – Plattform zur zentralen Verwaltung hybrider und Multi-Cloud-Umgebungen, ermöglicht es On-Premises- und nicht Azure-Ressourcen wie Server und Kubernetes Cluster über das Azure Portal zu steuern. Positiv hervorgehoben werden kann die einheitliche Kontrolle, die Nutzung von Azure Diensten in jeder Infrastruktur und die Anwendung von Azure Tools wie Policy und Monitor. Die Integrationen können komplex werden und zugleich kostenintensiv (azure.microsoft.com, 2025).

Quellen

https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/azure-services-resource-providers

Übung 2: Azure Updates

In den letzten Jahren konnte die schnelle Entwicklung von Microsoft Azure beobachtet werden. Sehr viele Services werden neu entwickelt, bestehende Services werden geändert und weitere hingegen werden aus dem Leistungsportfolio von Microsoft Azure entfernt. Im Zuge dieser Übung ist es Ziel, dass du eine Herangehensweise definierst, um immer alle aktuellen und für dich wesentlichen Azure News zu empfangen. Wie könntest du hierbei vorgehen? Wie stellst du sicher, dass auch wesentliche Blog Beiträge und/oder sicherheitsrelevante Informationen empfängst? Erstelle eine Liste von wesentlichen Informationstöpfen (Webseiten, Blogs, …) und beschreibe, wie du diese Informationen (am besten zentral) konsumieren kannst. Teile dieser Liste werden wir anschließend dem gesamten Jahrgang zur Verfügung stellen.

Tipp:

Oftmals werden RSS Feeds angeboten.

Schritt 1: Datenquellen und Integration

Die Lösung beginnt mit der Auswahl und Integration relevanter Datenquellen. Hierzu können Webscraping, RSS-Feeds oder APIs genutzt werden. Neue Quellen können später leicht hinzugefügt werden.

Webseiten:

- https://azure.microsoft.com/en-us/blog/content-type/announcements/
- https://azure.microsoft.com/en-us/updates
- https://learn.microsoft.com/en-us/azure/architecture/changelog
- https://learn.microsoft.com/en-us/azure/?product=popular
- https://azurecitadel.com/
- https://build5nines.com/
- https://azurecharts.com/
- https://learn.microsoft.com/en-us/answers/
- https://feedback.azure.com/d365community
- https://learn.microsoft.com/en-us/samples/browse/?expanded=azure&products=azure-resource-manager

Blog-Sammlungen

- https://feedly.com/i/top/azure-blogs

- https://bloggers.feedspot.com/microsoft_azure_blogs/

Einpersonen Blog:

- https://azurecloudai.blog/
- https://gregorsuttie.com/
- https://www.thomasmaurer.ch/
- https://www.michaelcrump.net/
- https://turbo360.com/blog

Podcasts:

- https://learn.microsoft.com/de-de/shows/azure-friday/
- https://azpodcast.azurewebsites.net/

Communities:

- https://www.reddit.com/r/AZURE/

Repos

- https://github.com/Azure-Samples
- https://github.com/Azure/bicep
- https://github.com/Azure

Control:

- https://resources.azure.com/

YouTube:

- https://www.youtube.com/@NTFAQGuy/videos -> Azure Updates gut erklärt.
- https://www.youtube.com/@ITOpsTalk/videos - > IT Pro & Operations content from Microsoft's Modern Hybrid Infrastructure Advocacy team. Video series, screencast demos, exam prep, product team interviews, shows and more.
- https://www.youtube.com/c/MicrosoftAzure
- https://www.microsoftsecurityinsights.com/
- https://www.youtube.com/@TheAzurePodcast/videos
- https://www.youtube.com/@AzureAcademy/videos

Schritt 2: Automatisierte Aggregation

Damit wir automatisch die Daten der verschiedenen Quellen nutzen können müssen wir Webscraping nutzen, RSS-Feeds verwenden sowie Github API oder YouTube API

Webseiten mit RSS-Feeds:

Einige Webseiten (wie die Azure-Updates-Seite) bieten bereits RSS-Feeds. Diese können mit dem Python feedparser direkt abgerufen und verarbeitet werden.

```
# Zuerst importieren wir die Bibliothek feedparser, die uns beim Abrufen und
Verarbeiten von RSS-Feeds hilft
import feedparser  # feedparser wird verwendet, um RSS-Feeds zu lesen und zu
parsen
# URL des RSS-Feeds von Azure-Updates
rss_url = "https://azure.microsoft.com/en-us/updates/rss"
# Der feedparser ruft den RSS-Feed ab und parst ihn
feed = feedparser.parse(rss_url)
# Wir gehen durch jedes Element im Feed (jedes "entry")
for entry in feed.entries:
    # Ausgabe des Titels der aktuellen Nachricht
    print(f"Title: {entry.title}")
        # Ausgabe des Links zur aktuellen Nachricht
    print(f"Link: {entry.link}")
        # Ausgabe des Veröffentlichungsdatums der Nachricht
    print(f"Published: {entry.published}")
        # Ausgabe einer kurzen Beschreibung oder Zusammenfassung der Nachricht
    print(f"Description: {entry.summary}\n")
   # Falls noch nicht installiert: pip install feedparser
# Importieren von feedparser: Dies ist die Bibliothek, die es uns ermöglicht,
RSS-Feeds in Python zu lesen und zu parsen.
# URL des RSS-Feeds: Wir verwenden die Azure-Update-Seite als Beispiel-URL für
den RSS-Feed.
# Feed abrufen und parsen: Mit feedparser.parse(rss_url) wird der Feed
heruntergeladen und geparsed.
# Durchlauf der Feed-Elemente: In einer Schleife durchlaufen wir alle Einträge
des Feeds und geben wichtige Informationen wie Titel, Link,
Veröffentlichungsdatum und eine kurze Beschreibung aus
```

Abbildung 1: Beispiel Phytonscript für RSS-Feeds

Webscraping:

Einige Webseiten wie Azure Citadel und Azure Architecture Changelog bieten keine direkten Feeds. Hier können wir Python-Skripte mit BeautifulSoup nutzen, um die Inhalte zu extrahieren.

```
# Zuerst importieren wir die Bibliotheken requests (für HTTP-Anfragen) und
BeautifulSoup (für das Parsen von HTML)
import requests  # Für HTTP-Anfragen
from bs4 import BeautifulSoup  # Zum Parsen von HTML
# Die URL der Webseite, von der wir die Daten extrahieren möchten
url = "https://azure.microsoft.com/en-us/blog/content-type/announcements/"
# Wir senden eine GET-Anfrage an die URL
response = requests.get(url)
# Wir parsen den HTML-Inhalt der Seite mit BeautifulSoup
soup = BeautifulSoup(response.content, "html.parser")
# Wir suchen nach allen h3-Elementen auf der Seite (diese enthalten in der
Regel die Titel der Ankündigungen)
announcements = soup.find_all('h3')
# Wir gehen durch jedes gefundene h3-Element und extrahieren den Titel und den
Link
for announcement in announcements:
    # Den Text des Titels extrahieren
    title = announcement.get_text()
    # Den Link (href-Attribut) extrahieren, der sich im <a>-Tag befindet
    link = announcement.find('a')['href']
    # Die extrahierten Informationen ausgeben
    print(f"Title: {title}\nLink: {link}\n")
# Falls requests oder beautifulsoup4 noch nicht installiert sind, kannst du
sie mit folgendem PowerShell-Befehl installieren:
# pip install requests beautifulsoup4
# HTTP-Anfrage mit requests: Die Webseite wird mit requests.get(url)
abgerufen.
# HTML-Parsing mit BeautifulSoup: Der HTML-Inhalt der Webseite wird mit
BeautifulSoup geparsed, um die relevanten Daten zu extrahieren.
# Extraktion von h3-Elementen: Der Code sucht nach allen h3-Tags, die in
diesem Beispiel die Ankündigungstitel enthalten.
# Extraktion von Titel und Link: Für jedes h3-Tag wird der Titel mit
.get_text() extrahiert und der Link aus dem href-Attribut des a-Tags
abgerufen.
# Ausgabe: Die Titel und Links der Ankündigungen werden in der Konsole
ausgegeben.
```

Abbildung 2: Beispiel Phytonscript für Webscraping

Podcast-Feeds:

Podcasts können über RSS-Feeds abgerufen werden. Wir können diese Feeds regelmäßig abrufen und neue Episoden als Nachrichten im Newsfeed darstellen.

```
# Wir importieren die benötigte Bibliothek feedparser, um den Podcast-RSS-Feed zu verarbeiten
import feedparser  # feedparser wird verwendet, um RSS-Feeds zu lesen und zu parsen

# URL des Podcast-RSS-Feeds
podcast_url = "https://azpodcast.azurewebsites.net/rss"

# Abrufen und Parsen des RSS-Feeds mit feedparser
feed = feedparser.parse(podcast_url)

# Durch die Episoden im Feed iterieren und relevante Informationen ausgeben
for entry in feed.entries:
    # Ausgabe des Titels der aktuellen Episode
    print(f"Podcast Title: {entry.title}")

    # Ausgabe des Links zur Episode
    print(f"Link: {entry.link}\n")
# Import von feedparser: Wir verwenden feedparser, um RSS-Feeds zu lesen und zu parsen, was auch für Podcast-Feeds gilt.
# URL des Podcast-RSS-Feeds: In diesem Beispiel verwenden wir den RSS-Feed des Azure Podcasts.
# Feed abrufen und parsen: Mit feedparser.parse(podcast_url) wird der RSS-Feed abgerufen und geparsed.
# Durchlaufen der Episoden: Der Code iteriert durch jede Episode im Feed und gibt den Titel sowie den Link zur Episode aus.
```

Abbildung 3: Beispiel Phytonscript für Podcast-Feeds

GitHub Repositories:

Für Repositories (wie Azure-Samples) können wir GitHub-APIs verwenden, um die neuesten Commits oder Pull Requests abzurufen.

```
# Zuerst importieren wir die Bibliothek requests, um HTTP-Anfragen zu stellen
import requests  # Für HTTP-Anfragen an APIs
# URL der GitHub-API für die Commits eines Repositories (Azure-Samples)
api_url = "https://api.github.com/repos/Azure-Samples/azure-quickstart-
templates/commits"
# Wir senden eine GET-Anfrage an die API-URL, um die Commits abzurufen
response = requests.get(api_url)
# Überprüfen, ob die Anfrage erfolgreich war (Statuscode 200)
if response.status_code == 200:
    # Wir konvertieren die Antwort in ein JSON-Format
    commits = response.json()
     # Durch die Commits iterieren und relevante Informationen ausgeben
    for commit in commits:
        # Die Commit-Nachricht und der Link zum Commit werden extrahiert
        print(f"Commit: {commit['commit']['message']}")
        print(f"Link: {commit['html_url']}\n")
else:
    # Falls die Anfrage nicht erfolgreich war, geben wir eine Fehlermeldung
aus
    print(f"Fehler: Statuscode {response.status_code}")
# Falls requests noch nicht installiert ist, kannst du es mit folgendem
PowerShell-Befehl installieren:
# pip install requests
# GitHub API URL: Wir verwenden die GitHub-API-URL, um Informationen über die
neuesten Commits eines Repositories (in diesem Fall Azure-Samples/azure-
quickstart-templates) abzurufen.
# HTTP-Anfrage: Mit requests.get(api_url) wird eine GET-Anfrage an die GitHub-
API gesendet.
# Verarbeitung der Antwort: Die Antwort wird im JSON-Format zurückgegeben und
mit .json() konvertiert, um die Commit-Daten zu extrahieren.
# Ausgabe der Commit-Nachricht und des Links: Wir extrahieren die Commit-
Nachricht (commit['commit']['message']) und den Link (commit['html_url']) zu
jedem Commit und geben sie aus.
```

Abbildung 4: Beispiel Phytonscript für GitHub Repositories

YouTube-Video-Feeds:

Für YouTube-Videos können wir die YouTube Data API verwenden, um neue Videos zu erhalten.

```
# Zuerst importieren wir die Bibliothek requests, um HTTP-Anfragen zu stellen
import requests  # Für HTTP-Anfragen an die YouTube Data API
# URL der YouTube Data API, um Videos eines bestimmten Kanals abzufragen
# Beachte: Du musst deinen eigenen API-Schlüssel angeben (ersetze
'YOUR_API_KEY' durch deinen tatsächlichen Schlüssel)
youtube_url =
"https://www.googleapis.com/youtube/v3/search?part=snippet&channelId=UC0oVXoB4
on26E3FQpuJDbjA&order=date&type=video&key=YOUR_API_KEY"
# Wir senden eine GET-Anfrage an die YouTube API
response = requests.get(youtube_url)
# Überprüfen, ob die Anfrage erfolgreich war (Statuscode 200)
if response.status_code == 200:
    # Wir konvertieren die Antwort in ein JSON-Format
    videos = response.json()
    # Durch die Videos im Feed iterieren und relevante Informationen ausgeben
    for video in videos['items']:
        # Der Titel des Videos und der Link zum Video werden extrahiert
        print(f"Video Title: {video['snippet']['title']}")
        print(f"Link:
https://www.youtube.com/watch?v={video['id']['videoId']}\n")
else:
    # Falls die Anfrage nicht erfolgreich war, geben wir eine Fehlermeldung
aus
    print(f"Fehler: Statuscode {response.status_code}")
# Falls requests noch nicht installiert ist, kannst du es mit folgendem
PowerShell-Befehl installieren:
# pip install requests
# Erklärung:
# YouTube Data API URL: Diese URL ruft Videos von einem bestimmten YouTube-
Kanal ab. Der Parameter channelId gibt den Kanal an, von dem du Videos abrufen
möchtest (hier für einen Beispielkanal). Der Parameter order=date sorgt dafür,
dass die neuesten Videos zuerst angezeigt werden.
# API-Schlüssel: Du musst einen gültigen API-Schlüssel von der Google Cloud-
Plattform haben, um auf die YouTube Data API zuzugreifen. Ersetze YOUR_API_KEY
durch deinen echten Schlüssel.
# HTTP-Anfrage: Wir senden eine GET-Anfrage an die YouTube Data API.
# Verarbeitung der Antwort: Die Antwort wird im JSON-Format zurückgegeben und
mit .json() konvertiert, um die Video-Daten zu extrahieren.
# Ausgabe der Video-Titel und Links: Der Titel des Videos und der Link zum
Video werden extrahiert und in der Konsole ausgegeben.
```

Abbildung 5: Beispiel Phytonscript für Youtube-Video-Feeds

Power Automate Trigger, Azure Logic Apps und Python Script

Power Automate Trigger:

Power Automate ist eine benutzerfreundliche Möglichkeit, automatisierte Workflows zu erstellen. In deinem Fall wird Power Automate verwendet, um das Sammeln von Informationen aus den verschiedenen Quellen und deren Präsentation auf einer SharePoint-Seite zu automatisieren.

Trigger – RSS-Feed: Du kannst einen Trigger einrichten, der ausgelöst wird, wenn ein neuer Beitrag in einem RSS-Feed veröffentlicht wird. **Beispiel:** Trigger auslösen, wenn ein neuer RSS-Feed von "Azure Updates" erscheint.

Trigger – HTTP Request: Falls du mit APIs arbeitest (z. B. GitHub oder YouTube), kannst du HTTP-Request-Trigger verwenden, um auf Änderungen zu reagieren und die Daten abzurufen.

Trigger – SharePoint (Element erstellt oder geändert): Wenn neue Inhalte auf SharePoint veröffentlicht werden, könnte dies einen weiteren Trigger auslösen, um eine Benachrichtigung zu senden oder den nächsten Schritt zu starten.

Aktionen: Nachdem der Trigger ausgelöst wurde, kannst du automatisch eine E-Mail oder Microsoft Teams-Benachrichtigung senden. Du kannst auch eine neue Zeile in einer SharePoint-Liste oder auf einer Seite erstellen, um die neuesten Nachrichten darzustellen.

Azure Logic Apps Workflow:

Trigger:

RSS-Feed: Azure Logic Apps bietet eine native Unterstützung für RSS-Feeds, sodass du regelmäßig nach neuen Updates suchen und diese automatisch abrufen kannst.

API-Request (GitHub, YouTube): Du kannst einen Trigger konfigurieren, der regelmäßig API-Anfragen stellt, um neue Daten zu erhalten.

Aktionen:

Verarbeiten der Daten: Verwende Bedingungen, um bestimmte Arten von Updates (z. B. Sicherheits-Patches) zu filtern.

Benachrichtigungen senden: Wenn ein kritisches Update erkannt wird, sendet Logic Apps automatisch eine E-Mail oder Microsoft Teams-Nachricht.

Daten in SharePoint veröffentlichen: Du kannst die abgerufenen Daten in einer SharePoint-Liste oder auf einer SharePoint-Seite anzeigen.

Beispiel-Workflow:

Trigger: RSS-Feed für Azure Updates.

Aktion: Neue Nachrichten extrahieren und filtern.

Aktion: Daten in SharePoint speichern.

Aktion: E-Mail-Benachrichtigung senden, wenn eine sicherheitsrelevante Änderung erkannt wird.

Zusammengefasster Workflow:

Neue Quellen integrieren: RSS-Feeds und Webscraping werden genutzt, um neue Quellen zu integrieren. Jede Quelle wird mit einem entsprechenden Python-Skript oder API-Request abgerufen.

Automatisierung: Power Automate und Azure Logic Apps werden genutzt, um Trigger und Benachrichtigungen zu erstellen. Die gesammelten Daten werden auf einer zentralen SharePoint-Seite angezeigt.

Echtzeit-Benachrichtigungen: Besonders kritische Updates (z. B. Sicherheits-Patches) werden sofort über E-Mail oder Teams benachrichtigt.

Erweiterbarkeit: Neue Quellen können leicht hinzugefügt werden, und der gesamte Prozess bleibt flexibel und benutzerfreundlich.

Python-Implementierung zur Integration von Azure-News

Das folgende Python-Skript umfasst:

1. **RSS-Feed-Sammlung**: Liest die angegebenen Azure-RSS-Feeds aus.
2. **Formatierung**: Wandelt die gesammelten News in ein HTML-Format um, das für eine SharePoint-Newsfeed-Seite geeignet ist.
3. **SharePoint-Integration**: Veröffentlicht die News über die SharePoint-API (wird hier als Beispiel integriert, aber die API-Konfiguration muss durchgeführt werden).

```
import feedparser
import requests
from bs4 import BeautifulSoup
import os

# Define the RSS feed URLs
rss_feeds = [
    "https://azure.microsoft.com/en-us/updates/feed/",
    "https://learn.microsoft.com/en-us/azure/architecture/changelog/feed/rss",
    "https://build5nines.com/feed/"
]

# Function to fetch and parse RSS feeds
def fetch_rss_feeds(feed_urls):
    news_items = []
    for url in feed_urls:
        feed = feedparser.parse(url)
        for entry in feed.entries:
            news_items.append({
                "title": entry.title,
                "link": entry.link,
                "summary": BeautifulSoup(entry.summary, "html.parser").text if
'summary' in entry else "",
                "published": entry.published if 'published' in entry else ""
            })
    return news_items

# Fetch news items
news = fetch_rss_feeds(rss_feeds)

# Function to format news for SharePoint integration
def format_for_sharepoint(news_items):
    formatted_items = ""
    for item in news_items:
        formatted_items += f"<h3>{item['title']}</h3>\n"
        formatted_items += f"<p><a href='{item['link']}'
target='_blank'>{item['link']}</a></p>\n"
        formatted_items += f"<p>{item['summary']}</p>\n"
        formatted_items += f"<p><em>Published:
{item['published']}</em></p><hr>\n"
    return formatted_items

# Generate the HTML content for SharePoint
html_content = format_for_sharepoint(news)

# Example: POST to SharePoint Newsfeed (requires SharePoint API configuration)
def post_to_sharepoint(site_url, list_name, html_content):
```

Übung 3: Defender for Cloud

Ein wesentlicher Bestandteil beim Betrieb von Cloud Ressourcen ist die Sicherheit. Hierbei sollte jedoch nicht nur die Sicherheit bei der Konfiguration der Cloud Ressourcen (Eigenschaften von Azure Resource Typen) relevant. Vielmehr muss auch die Sicherheit innerhalb der Cloud Ressourcen (z.B. virtuelle Maschinen, Container, K8s, ...) sichergestellt werden. Microsoft stellt im Bereich Sicherheit hierzu die Defender Suite zur Verfügung. Im Zuge dieser Übung sollte recherchiert werden, welche Features „Defender for Cloud“ zur Verfügung stellt und welche Ressourcen damit geschützt werden können. Beantworte dabei auch, ob es unterschiedliche Versionen (P1 / P2 / ...) in den jeweiligen Angeboten gibt und wo hierbei der Unterschied liegt.

Beschreibung Defender for Cloud

Defender for Cloud wird immer mit Defender XDR geliefert und Microsoft gibt and das folgende Fähigkeiten vorliegen:

- Eine DevSecOps-Lösung (Development Security Operations), die das Sicherheitsmanagement auf Code-Ebene in Multi-Cloud- und Multi-Pipeline-Umgebungen vereinheitlicht
- Eine Cloud Security Posture Management (CSPM)-Lösung, die Maßnahmen zur Verhinderung von Sicherheitsverletzungen aufzeigt
- Eine Cloud Workload Protection Platform (CWPP) mit spezifischen Schutzfunktionen für Server, Container, Speicher, Datenbanken und andere Workloads

Defender for Cloud bietet die Möglichkeit **Azure security baseline for Microsoft Defender for Cloud** zu integrieren sowie verschiedene Partner Lösungen zu integrieren für zum Beispiel Software composition analysis (SCA), API security testing integrations, Weiterleitung der Insights an verschiedene Applikationen wie Elastic, Kondukto und integriert sich auch mit großen Lösungen wie Check Point und F5 Networks.

Quellen:

https://github.com/MicrosoftDocs/SecurityBenchmarks/blob/master/Azure%20Offer%20Security%20Baselines/3.0/microsoft-defender-for-cloud-azure-security-benchmark-v3-latest-security-baseline.xlsx

https://learn.microsoft.com/en-us/security/benchmark/azure/baselines/microsoft-defender-for-cloud-security-baseline?toc=/azure/defender-for-cloud/TOC.json

https://learn.microsoft.com/en-us/azure/defender-for-cloud/

https://azure.microsoft.com/en-us/pricing/details/defender-for-cloud/

https://learn.microsoft.com/en-us/azure/defender-for-cloud/defender-for-cloud-features

https://learn.microsoft.com/en-us/azure/defender-for-cloud/partner-integrations

Übung 4: Cloud Security Posture Management

Im Zuge der Übung 8 der praktischen Übungen wurde erkannt, dass sich CloudQuery leider in den letzten Monaten dazu entschieden, dass das Source Plugin für Microsoft Azure als „Premium Plugin“ eingestuft wird. Aus dem tollen Open Source Projekt wurde somit (leider) ein kostenpflichtiger Service. Im Zuge dieser Übung sollten die folgenden Fragen beantwortet werden:

a) Was versteht man unter „Cloud Security Posture Management“?

Cloud Security Posture Management (CSPM) ist die Praxis, Sicherheitsrisiken in Cloud Infrastrukturen zu erkennen, zu überwachen und zu beheben. CSPM Tools prüfen Cloud Umgebungen auf Fehlkonfigurationen, Compliance Verstößen und Sicherheitslücken. Sie automatisieren die Analyse von Ressourcen, geben Handlungsempfehlungen zur Behebung und helfen bei der Einhaltung von Standards wie GDPR oder ISO 27001 (microsoft.com, 2025) (Gillis, 2024).

b) Kann die Fähigkeit von CloudQuery auch mit anderen Tools realisiert werden?

Ja sie können auch mit anderen Tools realisiert werden, Azure-native Dienste wie Azure Defender for Cloud oder Azure Policy (learn.microsoft.com, 2025) bieten ähnliche Funktionen zur Überwachung und Durchsetzung von Sicherheitsrichtlinien. Alternativ gibt es Open-Source-Tools wie Steampipe oder kommerzielle Lösungen wie Palo Alto Prisma Cloud (paloaltonetworks.com, 2025), die Multi-Cloud- und Hybrid-Umgebungen unterstützen.

Zudem bietet Microsoft für das Inventory und die Überprüfung folgendes Skript zu Verfügung: https://github.com/microsoft/ARI dieses erfolgt per Powershell Skript und Abfrage der REST API. Es wird dann ein DrawIO XML exportiert sowie ein Excel bzw. CSV mit unterschiedlichen Reitern.

Sowie: https://github.com/JulianHayward/Azure-MG-Sub-Governance-Reporting

c) Inwiefern könnte der Azure Resource Graph Explorer hierbei unterstützen?

Der Azure Resource Graph Explorer unterstützt CSPM, indem er eine zentrale Quelle für Ressourcendaten bereitstellt. Mit KQL-Abfragen können Sicherheitsprobleme gezielt identifiziert werden, beispielsweise unverschlüsselte Speicher oder exponierte Netzwerke. Diese Daten lassen sich maschinenlesbar exportieren und in andere Tools integrieren, um Sicherheitsanalysen weiterzuführen (learn.microsoft.com, 2025).

Tipp:

Beachtet die besprochene Eigenschaft, dass alle Ressourcen in einem maschinenlesbaren Format exportiert werden können.

Übung 5: Berechnung des Service Level

Ein Unternehmen hat folgende Architektur, wo im Zuge der Übung anhand der SLAs berechnet werden soll, wie hoch die Verfügbarkeit des Webservices für einen User ist:

Der User greift auf ein VPN-Gateway (99,95%) zu. Anschließend wird er über ein Application Gateway (99,95%) und eine Azure Firewall (99,95%) auf die Applikation in einem App Service (99,95%) weitergeleitet. Für die Funktionsfähigkeit des App Service ist eine Azure SQL Database (99,99%) notwendig. Um die Ausfallzeiten zu verringern, wurden bereits zwei App Services (inkl. Datenbanken) in unterschiedlichen Regionen bereitgestellt.

User -> VPN-Gateway -> Application Gateway -> Azure Firewall -> App Service -> SQL Database -> App Service -> SQL Database

Achtung:

Hierbei handelt es sich um ein stark vereinfachtes Beispiel. Wir gehen davon aus, dass wir hierbei VPN-Gateway, Application Gateway, Azure Firewall und der Verbund der App Services (inkl. Datenbanken) in einem seriellen abhängig sind. Die Berechnung der Verfügbarkeit des Webservice sollte berücksichtigen, dass es sich hierbei um einen Parallelbetrieb der App Services (inkl. der jeweiligen SQL-Datenbank) handelt.

Wenn Fragen offen sind, können zu dokumentierende Annahmen getroffen werden.

Tipp:

https://www.microsoft.com/licensing/docs/view/Service-Level-Agreements-SLA-for-Online-Services

Annahmen:

Alle Komponenten (VPN-Gateway, Application Gateway, Azure Firewall) und der App Service Verbund sind seriell voneinander abhängig. Der App Service-Verbund (App Service + SQL Database) wird in zwei Regionen parallel betrieben. Ausfallzeiten durch andere Faktoren (z.B. Netzwerk Latenzen) wurden nicht berücksichtigt.

Serielle Abhängigkeiten:

Komponenten: VPN-Gateway, Application Gateway, Azure Firewall und der App Service (bestehend aus App-Service + SQL Database in zwei Regionen)

Parallelbetrieb der Services:

Zwei parallel betriebene App Services die jeweils mit einer eigenen SQL-Datenbank, die Verfügbarkeit des Services wird wie folgt berechnet.

Die **Verfügbarkeiten** der einzelnen Komponenten sind der Angabe entnommen.

Verfügbarkeit einer Instanz (seriell)

$$\text{Verfügbarkeit App Service} + \text{SQLDatabase (einzelne Instanz)} = \\ = 0{,}9995 * 0{,}9999 = 0{,}9994$$

Da zwei Instanzen in versch. Regionen parallel betrieben werden

$$\text{Verfügbarkeit AppService(Verbund)} = 1 - ((1 - 0{,}9994) * (1 - 0{,}9994) \\ = 1 - (0{,}0006 * 0{,}0006) = 1 - 0{,}00000036 = 0{,}99999964$$

Verfügbarkeit des App Service (Verbund): **99,999964%**

Gesamte Verfügbarkeit (serieller Betrieb):

Für gesamten Webservice --> alle Komponenten seriell abhängig.

$$Gesamtverfügbarkeit \\ = VPN\ Gateway * Application\ Gateway * Azure\ Firewall \\ * Verfügbarkeit\ des\ App\ Service\ (Verbund) \\ = 0{,}9995 \times 0{,}9995 \times 0{,}9995 \times 0{,}99999964 = 0{,}998500389879$$

Gesamtverfügbarkeit: **99,85003899 %**

(Die Gesamtverfügbarkeit ergibt sich aus der schrittweisen Multiplikation der einzelnen Verfügbarkeiten).

Diese Berechnung erfolgt unter der Annahme das hier nicht eine Architektur wie im Azure Architektur Zentrum für High Availabilty gewählt wurde siehe dazu Abbildung 1, wobei vor dem Traffic Manager noch das VPN Gateway vorgeschalten würde.

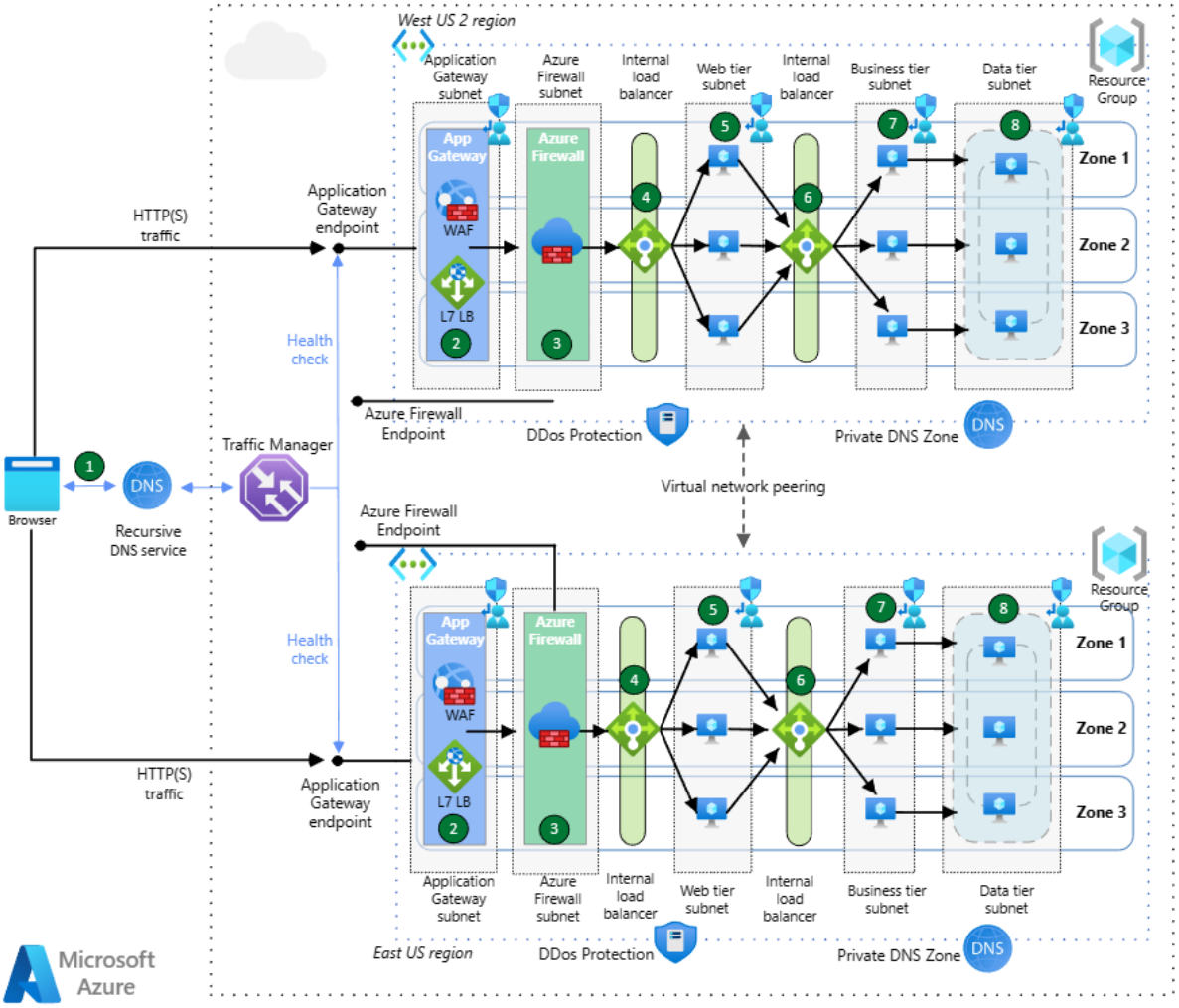

Abbildung 6 Link: https://learn.microsoft.com/en-us/azure/architecture/high-availability/reference-architecture-traffic-manager-application-gateway

Sondern das eine Architektur angelehnt an Abbildung 2 wo eine Landing Zone mit WAF und Webapplication Gateway genutzt wird.

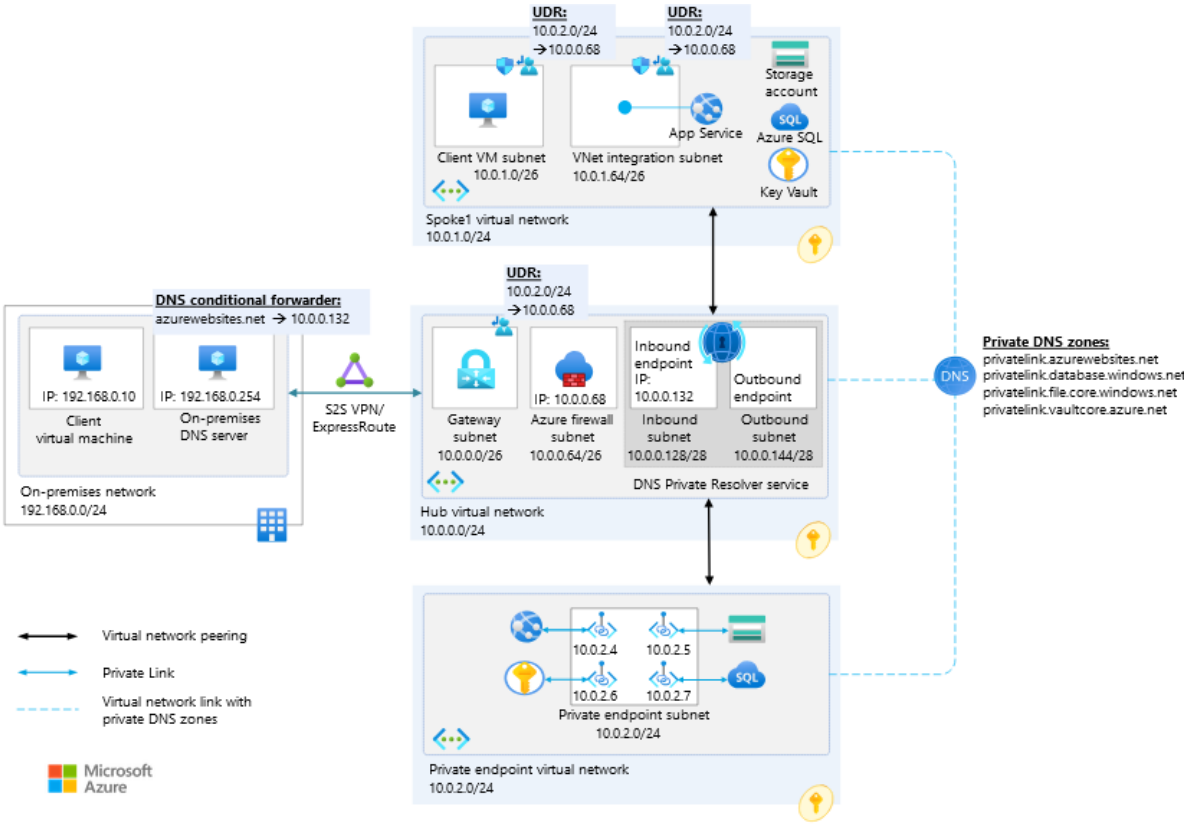

Abbildung 7:Link: https://learn.microsoft.com/en-us/azure/architecture/web-apps/guides/networking/access-multitenant-web-app-from-on-premises

Übung 6: Azure Policies mit DSC

Im Zuge der Lehrveranstaltung und der Übung 4 haben wir uns mit Azure Policies als Werkzeug im Bereich Governance beschäftigt. Des Weiteren haben wir in der Lehrveranstaltung gesehen, für welchen Einsatzzweck DSC (Desired State Configuration) eingesetzt werden kann.

Im Zuge dieser Übung sollte beantwortet werden, inwiefern sich Azure Policies und PowerShell DSC ergänzen können.

Azure Policies legen Governance-Richtlinien für Ressourcen fest, während PowerShell DSC den definierten Zustand von Betriebssystemen sicherstellt. Die Integration über Azure Policy Guest Configuration ermöglicht die Anwendung von DSC-Methoden, um Richtlinien auf Betriebssystemebene (z.B. Software-Compliance, Sicherheitsrichtlinien) durchzusetzen. Gemeinsam stellen sie sicher, dass Ressourcen konform und korrekt konfiguriert bleiben (Greene, 2019).

Auf welchen Ziel-Plattformen (On-Premises, Azure VMs, ...) kann welche Technologie zum Einsatz gebracht werden?

Azure Policies - für Azure-Ressourcen, um Governance und Compliance sicherzustellen (learn.microsoft.com, 2024). Guest Configuration - für Azure VMs und On-Prem.-Ressourcen (via Azure Arc), um Betriebssystemrichtlinien zu überwachen (learn.microsoft.com, 2024) . PowerShell DSC - für Windows- und Linux-Server (Azure, On-Prem., andere Clouds) zur granularen Konfiguration und Zustandsüberwachung (docs.azure.cn, 2024).

*Tabelle 1 - **Azure Policies mit DSC***

Technologie	Zielplattform	Einsatzmöglichkeiten
Azure Policies	Azure Ressourcen (z.B. VMs, Storage)	Governance & Compliance für Azure Ressouren Durchsetzung von Regeln für Ressourcenerstellung, Konfiguration und Tags
Azure Policy Guest Configuration	Azure VMs On-Prem. VMs (via Azure Arc)	Betriebssystemrichtlinien wie Software Compliance, Sicherheitsrichtlinien und Konfigurationsmanagement
PowerShell DSC	Windows und Linux Server (Azure VMs, On-Prem., andere Clouds)	Granulare Konfiguration von Servern Sicherstellung eines definierten Zustands (z.B. installierte Software, Registry Einträge, Dienste)

		Nur mehr On-Prem. verfügbar

Gibt es hier Einschränkungen?

Azure Policies wirken primär auf Azure Ressourcen (für On-Prem. Wird Azure Arc benötigt). Powershell DSC erfordert Infrastruktur (z.B. Pull Server) und bietet begrenzt Linux Unterstützung. Guest Configuration ist ausschließlich mit Azure VMs oder Azure Arc verfügbar und benötigt zusätzliches Fachwissen für benutzerdefinierte Richtlinien (docs.azure.cn, 2023).

Tipp:

“Azure Policy Guest Configuration”

Übung 7: Implementieren eines „Azure Resource Approval Prozess"

Stell dir vor, dass du in einem Unternehmen für die Cloud Security verantwortlich bist. Damit du entscheiden kannst, welche Cloud Security Resourcen (und insbesondere in welcher Konfiguration) im Unternehmen eingesetzt werden dürfen, implementierst du einen Prozess zur Informationsbeschaffung zu den jeweiligen Cloud Services (Azure Resource Typen).

Als Teil dieses Prozesses musst du identifizieren, mit welcher Konfiguration Storage Accounts im Unternehmen eingesetzt werden dürfen. Hierbei gilt es als **ersten Schritt** zu identifizieren, welche sicherheitsrelevanten Eigenschaften im Zuge der Bereitstellung und Konfiguration eines Storage Accounts bestehen?

Im **zweiten Schritt** möchtest du evaluieren, welche Logs von Storage Accounts geschrieben werden.

Im konkreten sollten dann folgende Fragen beantwortet werden:

- **Frage 1:** Welche sicherheitsrelevanten Eigenschaften können bei der Bereitstellung und Konfiguration eines Storage Accounts ausgewählt werden? Welche Einstellung würdest du für diese Einstellung empfehlen? Formuliere hierzu mindestens drei Empfehlungen.

Um die Sicherheit von Azure Storage Accounts zu gewährleisten, sind mehrere Schlüsseleinstellungen wichtig, es sollte der Netzwerkzugriff strikt reguliert werden. Durch die Nutzung privater Endpunkte wird der Zugriff alleinig über vertrauenswürdige Azure Virtual Networks ermöglicht, während die integrierte Firewall auf autorisierte IP-Bereiche oder Subnetze beschränkt werden sollte. Dies minimiert das Risiko externer Angriffe erheblich. Für die Verschlüsselung ist per default die serverseitige Verschlüsselung (SSE) eine gute Basis, für höhere Anforderungen empfehlen sich jedoch kundenseitig verwaltete Schlüssel (CMK), die über Azure Key Vault kontrolliert werden und Compliance-Anforderungen wie DSGVO oder HIPAA erfüllen.

Bei der Authentifizierung und Autorisierung ist Azure Active Directory (Azure AD) der bevorzugte Ansatz, um Zugriffe sicher zu steuern. Wenn temporäre Zugriffstoken (SAS) genutzt werden müssen, sollten diese auf maximal 24 Stunden Laufzeit begrenzt und mit minimalen Berechtigungen ausgestattet werden. Zudem sind veraltete Storage-Kontoschlüssel zu deaktivieren.

Für kritische Daten empfiehlt sich RA-GZRS (Read-Access Geo-Zone-Redundant Storage) so werden Daten nicht nur innerhalb einer Region, sondern auch regionsübergreifend repliziert, was die Ausfallsicherheit maximiert.

Schließlich sollte der öffentliche Zugriff auf Blob-Container standardmäßig auf Privat gesetzt werden, um Datenlecks zu vermeiden. Ausnahmen, wie etwa für statische Websites, bedürfen einer expliziten Genehmigung und kontinuierlicher Überwachung.

Quellen:

https://learn.microsoft.com/en-us/azure/storage/blobs/authorize-access-azure-active-directory

https://learn.microsoft.com/en-us/azure/storage/common/storage-service-encryption

https://learn.microsoft.com/en-us/azure/storage/blobs/authorize-access-azure-active-directory

https://learn.microsoft.com/en-us/azure/storage/common/storage-sas-overview#best-practices-when-using-sas

https://learn.microsoft.com/en-us/azure/storage/common/storage-redundancy

https://learn.microsoft.com/en-us/azure/storage/blobs/anonymous-read-access-configure?tabs=portal

- **Frage 2:** Welche Logs / Logkategorien werden von Storage Accounts geschrieben? Welche würdest du hierbei als sicherheitsrelevant Einschätzen und in einem zentralen Speicherort vorhalten?

Azure Storage Accounts protokollieren vier zentrale Kategorien, diese sind StorageRead (Lesezugriffe), StorageWrite (Schreibvorgänge), StorageDelete (kritische Löschungen) und StoragePolicy (Richtlinienänderungen). Diese Logs sind relevant, um Datenexfiltration, unbefugte Änderungen oder böswillige Löschungen zu erkennen.

Es empfiehlt sich somit die Logs zentral in Azure Log Analytics zu speichern und mit Azure Sentinel zu analysieren, mindestens für die Dauer von einem Jahr eine Retention für Compliance zu schaffen und Azure Monitoring für Warnungen bei Anomalien (z.B. Massendatenlöschungsvorgänge) zu nutzen. Tools wie Azure Policy, Defender for Storage und Infrastructure as Code sind hierbei nützlich.

Quellen:

https://learn.microsoft.com/en-us/azure/storage/blobs/monitor-blob-storage?tabs=azure-portal

Literaturverzeichnis

azure.microsoft.com. (2025). Von https://azure.microsoft.com/en-us/products/storage/blobs abgerufen

azure.microsoft.com. (2025). Von https://learn.microsoft.com/en-us/azure/azure-monitor/ abgerufen

azure.microsoft.com. (2025). Von https://learn.microsoft.com/de-de/azure/defender-for-cloud/defender-for-cloud-introduction abgerufen

azure.microsoft.com. (2025). Von https://azure.microsoft.com/en-us/products/virtual-machines abgerufen

azure.microsoft.com. (2025). Von https://azure.microsoft.com/de-de/products/azure-arc abgerufen

docs.azure.cn. (2023). Von https://docs.azure.cn/en-us/governance/policy/how-to/guest-configuration-create-definition abgerufen

docs.azure.cn. (2024). Von https://docs.azure.cn/en-us/virtual-machines/extensions/dsc-windows abgerufen

Gillis, A. S. (April 2024). *techtarget.com*. Von https://www.techtarget.com/searchsecurity/definition/Cloud-Security-Posture-Management-CSPM abgerufen

Greene, M. (7th. June 2019). *devblogs.microsoft.com*. Von https://devblogs.microsoft.com/powershell/azure-policy-guest-configuration-client/ abgerufen

learn.microsoft.com. (2024). Von https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/manage/azure-server-management/guest-configuration-policy abgerufen

learn.microsoft.com. (2024). Von https://learn.microsoft.com/en-us/azure/governance/machine-configuration/overview abgerufen

learn.microsoft.com. (2025). Von https://learn.microsoft.com/en-us/azure/defender-for-cloud/concept-cloud-security-posture-management abgerufen

learn.microsoft.com. (2025). Von https://learn.microsoft.com/en-us/azure/governance/resource-graph/overview abgerufen

microsoft.com. (2025). Von https://www.microsoft.com/en-us/security/business/security-101/what-is-cspm abgerufen

paloaltonetworks.com. (2025). Von https://www.paloaltonetworks.com/prisma/cloud/cloud-security-posture-management abgerufen