Übung 1: Implementieren einer virtuellen Maschine

Im Zuge dieser Übung soll eine virtuelle Maschine auf Azure erstellt und darauf zugegriffen werden (RDP, SSH, ...). Es muss dokumentiert werden, welche Einstellungen im Zuge der Bereitstellung gewählt worden sind. Hierbei sollten die jeweiligen Einstellungen begründet werden. Des Weiteren sollte danach untersucht werden, inwiefern die virtuelle Maschine überwacht werden kann.

- Compute (virtuelle Maschine)
- Networking (virtuelles Netzwerk)
- Storage (virtuelle Disk)
- Monitoring (Azure Monitor)

Übung 2: Azure Backup

Im Zuge dieser Übung soll eine virtuelle Maschine in Azure bereitgestellt werden, welche anschließend per Azure Backup gesichert wird. Ziel ist es, dass eine Wiederherstellung der gesamten virtuellen Maschine durchgeführt wird. Hierzu bietet sich an, dass nach der Durchführung des Backups eine Datei gelöscht wird, um diese später (nach dem Restore) wieder vorzufinden. Des Weiteren sollte im Protokoll die Frage beantwortet werden, wie hier zwischen Snapshots und Backups im Wiederherstellungsprozess unterschieden wird oder unterscheidet werden kann.

Übung 3: Azure Infrastructure as Code

Da virtuelle Maschinen öfter benötigt werden muss der Bereitstellungsprozess (inkl. Netzwerk, (öffentliche) IP-Adresse, etc.) beschleunigt und vereinfacht werden. Hierbei soll somit eine Herangehensweise mittels Infrastructure as Code verfolgt werden. Ziel dieser Übung ist, dass eine virtuelle Maschine per Infrastructure as Code (ARM, Bicep, Terraform) in Azure bereitgestellt wird.

Des Weiteren kann folgendes Repository bei der Erstellung der Bicep Templates helfen:
https://github.com/Azure/azure-quickstart-templates
(in diesem Repository befinden sich viele verschiedene IaC Templates)

Generell kann selbst entschieden werden, welche IaC Technologie verwendet wird. Im vorliegenden Fall wird jedoch empfohlen, dass Bicep verwendet wird.

Tipp:
Die Visual Studio Code Extension „Bicep“ kann die Entwicklung von Templates stark vereinfachen.

Übung 4: Azure Policies

Im Zuge dieser Übung sollen folgende Azure Policies erstellt werden:

- Azure Region
  Es soll sichergestellt werden, dass ausschließlich die Azure Regionen in Europa für die Bereitstellung von Ressourcen verwendet werden dürfen.

- Azure Storage Account
  Es soll sichergestellt werden, dass ohne Authentifizierung auf keinen Storage Account (bzw. auf Daten in diesem Storage Account) zugegriffen werden kann.

- Öffentliche IP Adressen
  Es soll sichergestellt werden, dass öffentliche IP Adressen (Type: Public IP Address) ausschließlich in einer bestimmten Resource Gruppe zur Verfügung gestellt werden können.

Wie kann sichergestellt werden, dass die genannten Policies den beschriebenen Zweck erfüllen? Teste für diesen Zweck die Policy, in dem Azure Ressourcen entgegen dieser Policys bereitgestellt werden. Wo und wie wird der Vorgang blockiert?

Übung 5: Resource Graph Explorer

Im Zuge einer Bestandsaufnahme möchten wir eine Liste mit allen Azure Ressourcen. Ziel ist es, eine Liste mit allen Azure Ressourcen im Tenant zu erstellen. Des Weiteren soll eine zweite Liste angefertigt werden, welche die Azure Ressourcen nach dem Typ gruppiert und somit die Anzahl der jeweiligen Resource Typen (z.B. Public IP Address, Network Interface Card, ...) erkennbar wird.

Um hier gewisse Ressourcen vorliegen zu haben, erstelle vorab 5 ~ 6 Ressourcen in Azure. Wie können solche Listen angefertigt werden?

Tipp:
Der Azure Resource Graph Explorer könnte hierbei unterstützen.

Übung 6: Bereitstellung von mehreren virtuellen Maschinen mit gleicher Konfiguration

In dieser Übung kombinieren wir die Ergebnisse von Übung 2 mit der Fähigkeit, dass mehrere virtuelle Maschinen (gleichzeitig) bereitgestellt werden.

Passe das Template von Übung 3 so an, dass zum Bereitstellungszeitpunkt entschieden werden kann, wie viele virtuelle Maschinen bereitgestellt werden sollen (z.B. via Bicep Parameter).

Tipp:
Iterative Loops in Bicep könnten hierbei unterstützen.

Übung 6: Azure Arc

Im Zuge dieser Übung soll Azure Arc auf einem Windows Betriebssystem (On-Premises) installiert werden. Als Voraussetzung zu dieser Übung gilt, dass eine virtuelle Maschine, welche nicht in Azure gehostet wird, vorliegt. Diese virtuelle Maschine soll anschließend per Azure Arc an Azure angebunden werden.

Welche Möglichkeiten bietet die Anbindung per Azure Arc? Teste hierbei zwei Möglichkeiten und dokumentiere diese (z.B. Log Forwarding, Azure Policies, Change Tracking & Inventory, Update Management, Guest Configuration, ...) im Protokoll.

Folgende Beschreibung kann bei der Durchführung der Übung unterstützen:
https://learn.microsoft.com/en-us/azure/azure-arc/servers/learn/quick-enable-hybrid-vm

Übung 7: Configuration as Code

In Übung 2 wurde eine virtuelle Maschine durch Infrastructure as Code erzeugt. Oftmals wird jedoch keine „leere“ virtuelle Maschine benötigt, sondern sie sollte mit vordefinierter Konfiguration und Applikationen ausgestattet werden. Auch hierfür möchten wir den Bereitstellungsprozess mittels „Configuration as Code“ vereinfachen. Ziel dieser Übung ist, dass ein IaC Template so angepasst wird, dass bei der Bereitstellung auch automatisch eine Konfiguration (z.B. Installation von IIS und Anpassung der Startseite) vorgenommen wird. Hierbei soll mit (PowerShell) Desired State Configuration eine Konfiguration deiner Wahl vorgenommen werden (z.B. Installation eines IIS und Anpassung der Startseite).

Des Weiteren sollte die Frage beantwortet werden, inwiefern hierbei Azure Automation State Configuration unterstützen könnte.

Generell kann selbst entschieden werden, welche IaC / CaC Technologie verwendet wird. Im vorliegenden Fall wird jedoch empfohlen, dass Bicep inkl. DSC verwendet wird.

~~Übung 8: Cloud Compliance~~

~~Befasse dich mit der Software cloudquery (natürlich mit Fokus auf Microsoft Azure):~~

- ~~https://www.cloudquery.io/~~
- ~~https://github.com/cloudquery/cloudquery~~
- ~~https://hub.cloudquery.io/plugins/source/cloudquery/azure/v11.4.0/docs?authors=official~~

~~Ziel dieser Übung ist es, dass cloudquery gegenüber deine eigenen Umgebung ausgeführt wird, und die erhobenen Informationen strukturiert in einer Datenbank gespeichert werden. Anschließend soll eine Sicherheitsüberprüfung mit den Built-In Policies ausgeführt werden. Stelle bevor cloudquery ausgeführt wird, damit auch tatsächliche Informationen erhoben werden können, eine virtuelle Maschine und einen App Service in deiner Azure Umgebung bereit.~~

~~Des Weiteren wird hier empfohlen, dass als Destination eine PostgreSQL Datenbank gewählt wird (da ansonsten die Built-In Policies abgeändert werden müssen).~~

~~Vorgehensweise:~~

- ~~PostgreSQL zur Verfügung stellen (z.B. via Docker)~~
- ~~Erstellung des File zur Konfiguration der Source- und Destination Plugins (Azure to PostgreSQL)~~
- ~~cloudquery ausführen (inkl. Erstellung der App Registration, …)~~
- ~~Folgende Built-In Policy gegenüber den erhobenen Informationen ausführen: https://github.com/cloudquery/cloudquery/tree/main/plugins/source/azure/policies~~

~~Konnten Abweichungen mit der Ausführung von cloudquery inkl. der Built-In Policy identifiziert werden? Wie können diese Abweichungen eingesehen werden?~~

Update:

CloudQuery hat sich 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 Services. Dennoch könnt ihr euch mit dem Thema im Zuge der theoretischen Übungen beschäftigen.

Übung 9: Implementierung von AKS

Implementieren einer Azure Kubernetes Umgebung.

Folgende Funktionalitäten sollen dabei berücksichtigt werden:

- Automatische Skalierung der Workload-Nodes
- Verwendung von aktuellen Netzwerk-Features:
  - CNI
  - Calico
- Integration in die Azure Plattform nutzen:
  - Monitoring
  - Storage (CSI)
- Ingress-Controller:
  - Welche Möglichkeiten stehen zur Verfügung?
  - Vor- und Nachteile inkl. Begründung zur Auswahl

Nützliche & unterstützende Ressourcen:

- https://learn.microsoft.com/en-us/azure/aks/learn/quick-kubernetes-deploy-portal?tabs=azure-cli
- https://learn.microsoft.com/en-us/azure/aks/scale-cluster?tabs=azure-cli
- https://learn.microsoft.com/en-us/azure/aks/csi-storage-drivers
- https://learn.microsoft.com/en-us/azure/aks/managed-azure-ad
- https://kubernetes.github.io/ingress-nginx/

Übung 10: Webseite auf Web App bereitstellen und mit CI/CD ausstatten

Im Zuge der Cloud Migration einer Webapplikation möchten wir die Chance nutzen und rudimentäre CI/CD Fähigkeiten implementieren. Erstelle hierzu eine Web App in Azure und beschäftige dich mit der Frage, wie die Web App automatisch mit dem Code aus einem Code Repository ausgestattet werden kann. Ziel ist es, dass bei Änderungen in dem Code Repository auch die Web App mit dem neuen Code ausgestattet wird.

Es ist nicht Ziel, dass im Zuge dieser Übung eine komplexe Applikation verwendet wird. Eine PHP Datei mit 3 ~ 4 Zeilen sollte für den Showcase ausreichen. Dokumentiere (wie gewohnt) die Vorgehensweise sowie allfällige Herausforderungen.

Tipp:

Hierzu könnte das Deployment Center im Web App sowie ein GitHub Repository unterstützen.

Übung 11: Hub- and Spoke Architektur (zählt für zwei Übungen)

Beschäftige dich mit dem Thema „Hub & Spoke Netzwerk Topologie“.

Unterstützende Ressourcen:
(hier kann nachgelesen werden was man unter Hub & Spoke versteht):
https://learn.microsoft.com/en-us/azure/architecture/reference-architectures/hybrid-networking/hubspoke?tabs=cli

Stelle anschließend eine Hub and Spoke Architektur in Azure bereit. Hierzu kann das Azure Portal oder Infrastructure as Code (z.B. Biceps) verwendet werden. Folgende Struktur soll als Unterstützung bei der Erarbeitung dieser Architektur helfen:

In der Azure Subscription werden folgende Resource Gruppen erstellt:

- Resource Group für den Hub (RG-Hub)
- Resource Group für den Spoke (RG-Spoke1)

Folgende Netzwerke werden bereitgestellt:

- Es wird ein virtuelles Netzwerk im Hub bereitgestellt (vnet-hub in der Resource Group rg-hub)
- Es wird ein virtuelles Netzwerk im Spoke bereitgestellt (vnet-spoke im der Resource Group rg-spoke)

Um zwischen Hub und den Spoke kommunizieren zu können muss ein VNET Peering zwischen den zwei bestehenden Netzwerken (vnet-hub, vnet-spoke) eingerichtet werden.

Im Hub Netzwerk werden anschließend folgende Ressourcen zur Verfügung gestellt:

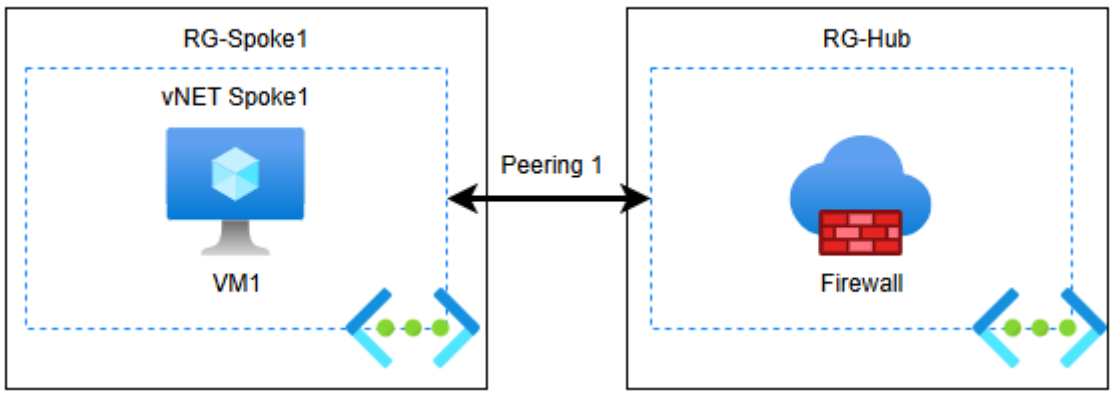
- Azure Firewall (inkl. Support-Ressourcen (Public IP, ...))
- Kleine SKU wählen (aufgrund der Kosten)

Im Spoke Netzwerk werden anschließend folgende Ressourcen zur Verfügung gestellt:

- Routing Table (Next Hop sollte hierbei die Azure Firewall im Hub sein)
- Virtuelle Maschine (inkl. Support-Ressourcen (NIC, OS Disk, ...))

Das Netzwerk im Spoke sollte jeden Netzwerkverkehr mit einem Ziel, das nicht am lokalen Netzwerk (vnetspoke) anliegt, über den Hub (als zentrale Routingstelle) routen.

Beispielhafte Abbildung der Infrastruktur:

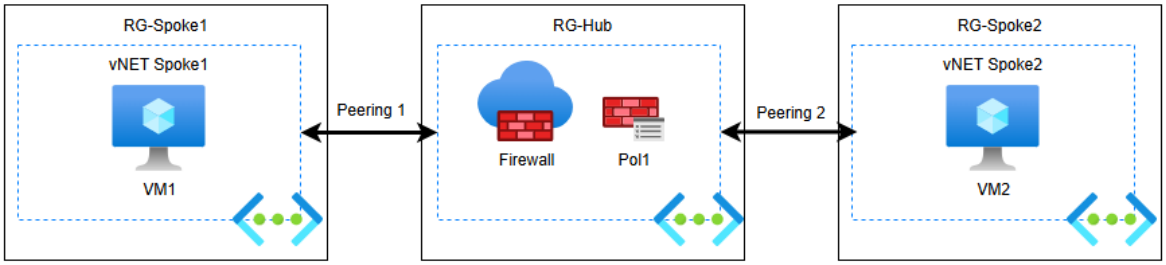


Übung 12: Implementieren eines zweiten Spokes mit VM & Monitoring

Erweitere Übung 1.1 mit einem weiterem Spoke und zusätzlicher VM.
Dazu sind **zusätzlich** die folgenden Resourcen (inkl. Support-Ressourcen) zu erstellen:

- Ressource Group
- VM
- VNet
- Peering

Die Architektur erweitert sich wie in der LV besprochen:



Teste den Datentransfer (z.B. Ping) zwischen den VMs. Erstelle eine Firewall Policy und blockiere den Datentransfer zwischen den beiden VMs. Der Zugriff von den VMs in das Internet soll aber weiterhin möglich sein. Gehe sicher, dass die VMs nicht direkt eine öffentliche IP zugewiesen haben und stelle sicher, dass der Datentransfer über die Firewall geleitet wird (Network Watcher).

Überlege wo idealerweise der “Bastion” Host für administrativen Remote-Zugriff platziert werden sollte. Vom Bastion-Host aus, sollen beide VMs verwaltbar sein. Notiere deine Überlegungen.