Projektdokumentation

Automatisierte Bereitstellung von Entwicklungs- und Testsystemen mithilfe von Cloud-Technologien

Auszubildende

[redacted]

Ausbildungsbetrieb
BTC IT Services GmbH
Escherweg 8
26121 Oldenburg

Ausbildungsberuf
Fachinformatikerin Systemintegration

Durchführungszeitraum
28.02.2018 – 28.03.2018

Inhaltsverzeichnis

1. Vorwort 5
2. Einleitung 5
2.1 Vorstellung BTC IT Services GmbH 5
2.2 Projektumfeld 5
2.3 Projektbeschreibung 5
2.4 Projektziel 6
3. Planungsphase 6
3.1 Ist-Analyse 6
3.2 Soll-Konzept 6
3.3 Projektschnittstellen 6
3.4 Zeitplanung 7
3.5 Vergleiche und Auswertungen 7
3.5.1 Vergleich Cloud-Provider 7
3.5.2 Entscheidung Cloud-Provider 7
3.5.3 Entscheidung Skript-Sprache 8
3.6 Vorstellung des ausgewählten Cloud-Providers und der Skript-Sprache beim Kunden 9
4. Durchführungsphase 9
4.1 Grundlegendes 9
4.1.1 PowerShell 9
4.1.2 Azure 9
4.2 Voraussetzungen zum Programmieren 9
4.2.1 Installation PowerShell 5.0 10
4.2.2 Installation Azure PowerShell 10
4.2.3 Kriterien für das Automatisierungs-Skript 10
4.3 Erstellung des Automatisierungs-Skriptes 10
4.3.1 Auswahl Default- oder Individuelle VM 10
4.4 Erstellung einer neuen VM 11
4.4.1 Vorzudefinierende Werte für eine Default-VM 11
4.4.1 Zu übergebende Werte für eine individuelle VM 11
4.4.2 Befehle zur Erstellung einer Windows-Maschine 12
4.5 Erweiterungen der Skripte 14
4.5.1 Überprüfung bereits vorhandener Komponenten 14
4.5.1.1 Abfrage der eingegeben Parameter 15
4.5.1.2 Überschreibungen verhindern 15
4.5.1.3 Admin-User anlegen 16
4.5.2 Ausgewähltes Betriebssystem laden 16
4.5.2.1 Auswahlmöglichkeit eines Windows-Betriebssystems 16

4.5.2.2 Auswahlmöglichkeit eines Linux-Betriebssystems 16
4.5.3 Überprüfung der VM-Größe in gewählter Lokation 17
4.6 Ausführliche Tests 17
4.7 Zusammenfassung 18
5. Abschlussphase 18
5.1 Kostenkalkulation 18
5.1.1 Personalkosten 18
5.1.2 Nutzungsgebühr 18
5.1.3 Gesamtkosten des Projektes 19
5.2 Kosten-Nutzen-Analyse 19
5.3 Soll-Ist-Vergleich 19
5.4 Abweichungen vom Projektantrag 20
5.5 Fazit/Ausblick 20
6. Betriebs- und Kundendokumentation 20
7. Quellenverzeichnis 20

Tabellenverzeichnis*

1 Zeitplanung 7
2 Nutzwertanalyse 8
3 Personalkosten 18
4 Nutzungsgebühr 18
5 Gesamtkosten des Projektes 19
6 Break-Even-Point 19
7 Soll-Ist-Vergleich 20

Bilderverzeichnis*

1 Überprüfung Azure-Installation 10
2 Abfrage auf bestehende Komponenten 15
3 if-Abfrage nach existierender Ressource 15
4 Überprüfung Name der VM 15
5 Admin-User erstellen 16
6 Konfiguration der Windows-Systeme 16
7 Konfiguration des Linux-Systems 16
8 Abfrage SSH-Schlüssel 17
9 Konfiguration SSH-Schlüssel 17
10 Überprüfung VM-Größe in Lokation 17
11 Kontrolle Parameter Azure-Portal 18

*Tabellen und Abbildungen aus der Betriebs- und Kundendokumentation werden nicht mit aufgeführt.

Anlagenverzeichnis

Anlage 1 Vergleich Cloud-Provider 21
Anlage 2 Detaillierte Zeitenplanung 22
Anlage 3 Kosten virtuelle Maschine 22
Anlage 4 Storagekosten 23
Anlage 5 Skript Abfrage Version 1 23
Anlage 6 Skript Default-VM Version 1 24
Anlage 7 Finales Skript Abfrage („VM_Erstellung.ps1“) 25
Anlage 8 Finales Skript für Default-VM („Ablauf_Default_VM.ps1“) 26
Anlage 9 Teil 1 Finales Skript individuelle VM („Ablauf_Indiv_VM.ps1“) 27
Anlage 10 Teil 2 Finales Skript individuelle VM 28
Anlage 11 Screenshots zu ausführlichen Tests 29
Anlage 12 Ausführlicher Soll-Ist-Vergleich 30
Anlage 13 Kundendokumentation 31
Anlage 14 Betriebsdokumentation 37

1. Vorwort

Mein Name ist Jessica Schmieder, ich bin Auszubildende zur Fachinformatikerin für Systemintegration bei der BTC IT Services GmbH. Ich habe im Rahmen meiner betrieblichen Ausbildung dieses Projekt in und für die BTC AG, im Folgenden der Kunde, durchgeführt. Aus Datenschutzgründen habe ich alle Passwörter sowie IP-Adressen unkenntlich gemacht. Ich möchte darauf hinweisen, dass weder das Zustandekommen der Verträge mit dem gewählten Provider noch das Einrichten von entsprechenden Cloud-Zugängen Inhalt meines Abschlussprojektes ist und ich diese Tätigkeiten nicht beschreiben werde.

Alle Angaben zu Servern und virtuellen Maschinen beziehen sich, falls nicht explizit anders erwähnt, immer sowohl auf Windows- als auch auf Linux-Betriebssysteme.

2. Einleitung

2.1 Vorstellung BTC IT Services GmbH

Die BTC IT Services GmbH (im Folgenden BITS genannt) ist eine 100-prozentige Tochtergesellschaft der Business Technology Consulting AG (BTC), welche ihren Hauptsitz in Oldenburg hat. Ihre Gründung fand im August 2009 durch die Ausgliederung aus der BTC statt.

Die BITS ist als Anbieter für Rechenzentren und IT-Dienstleistungen auf mittelständische Kunden und Konzerne ausgerichtet. Für die Betriebsleistung bietet die BITS eine redundante Rechenzentren-Kapazität, welche auf die höchsten Ansprüche an Sicherheit und Verfügbarkeit ausgerichtet ist. Im Firmenportfolio stellt die BITS unter anderem Outsourcing- und Outtasking-Leistungen, Desktop- und Hosting-Services sowie projektorientierte IT-Dienstleistungen für ihre Kunden bereit.

2.2 Projektumfeld

Wie bereits genannt führe ich das Projekt in der BTC durch. Die BTC, gegründet 2000, ist eines der führenden IT-Consulting-Unternehmen in Deutschland und ist international vertreten. Der Hauptsitz befindet sich in Oldenburg. Das Unternehmen beschäftigt mehr als 1600 Mitarbeiter und erzielte im Jahr 2016 einen Umsatz von 170,2 Mio. Euro. Die BTC ist eine Tochtergesellschaft des EWE-Konzerns. Das IT-Dienstleistungsangebot der BTC reicht von der Prozessberatung über die Systemintegration und das Applikations- und Systemmanagement bis zum Betrieb der IT-Systeme. Dabei gibt es hohe Kompetenzen in den Bereichen Energie, Industrie, Telekommunikation und öffentlicher Sektor.[1]

Das Projekt findet im Bereich Application & System Management (kurz ASM) der BTC statt. Zu den Aufgaben des Teams zählen unter anderem Bereitstellung einer Infrastruktur für die Softwareentwicklung im EWE-Konzern, Service-Management für interne und externe Kunden und fachlicher Betrieb der führenden Handelsplattform für Gaskapazitäten in Europa.

2.3 Projektbeschreibung

Der Kunde nutzt aktuell zur Virtualisierung Test-Maschinen auf physischen Servern im Rechenzentrum. Diese Server erreichen in naher Zukunft ihr Supportende und dürfen anschließend nicht mehr verwendet werden. Aus wirtschaftlichen Gründen hat sich der Kunde entschieden keine neue Hardware zu evaluieren, sondern sich der Cloud-Technologie zuzuwenden. Neben der Evaluierung eines Cloud-Providers wünscht sich der Kunde zusätzlich ein beschleunigtes und teilweise automatisiertes Vorgehen zur Erstellung von virtuellen Maschinen (im Folgenden VM genannt) in der letztendlich gewählten

Cloud. Das Team ASM hat mir die Umsetzung der genannten Anforderung zugewiesen.

[1] Vergleiche https://www.btc-ag.com/de/unternehmen.htm (aufgerufen am 28.02.18)

2.4 Projektziel

Nach Umsetzung des Projektes soll der Kunde seine Entwicklungs- und Testsysteme in virtuellen Maschinen mit gleichem Leistungsumfang wie im Rechenzentrum in einer Cloud aufsetzen können. Dies soll über ein von mir erstelltes Skript erfolgen, das teilweise für eine automatische Konfiguration sorgt, sodass eine schnelle Erstellung von virtuellen Maschinen gewährleistet werden kann.

3. Planungsphase

In der Planungsphase sollen der Ist- und Soll-Zustand analysiert sowie eine Nutzwertanalyse durchgeführt werden.

3.1 Ist-Analyse

Zum derzeitigen Stand werden alle virtuellen Maschinen des Kunden manuell erstellt. Das bedeutet es wird ein Auftrag aufgegeben, woraufhin ein Mitarbeiter diese VM entsprechend nach den Ansprüchen konfiguriert und auf einem physischen Server erstellt. Dies nimmt ungefähr zwei Stunden pro virtuelle Maschine in Anspruch. Diese Maschinen werden, wie in 2.3 Projektbeschreibung bereits dargestellt, auf insgesamt 14 dafür verwendeten physischen Servern im Rechenzentrum des Kunden aufgebaut. Damit liegen sowohl die Sicherstellung der Verfügbarkeit als auch jegliche Hardwarekosten aktuell bei dem Kunden.

3.2 Soll-Konzept

Der Kunde möchte eine Cloud-Umgebung haben. Dafür werde ich zunächst eine Nutzwerttabelle von relevanten Cloud-Anbietern erstellen. Nach Absprache mit dem Kunden hat dieser seine Auswahl dabei auf Microsoft Azure und Amazon Web Services (kurz AWS) eingeschränkt, da dies die aktuell vorherrschenden Provider am Markt sind, wie in Anlage 1 zu sehen ist. Die Verantwortung für Hardware sowie deren Wartung soll zukünftig beim Provider der Cloud liegen, um Wartungskosten, Belegung von Höheneinheiten im Rechenzentrum und Hardwarekosten einsparen zu können. Zudem kann so die Hardware bei Bedarf flexibler aufgerüstet und angepasst werden. Auch die Verfügbarkeit soll vom Provider sichergestellt werden und eine Verfügbarkeit von 99,9999 % bieten können.

Nach der Vertragsschließung und Einrichtung soll eine Automatisierung zum Erstellen von VMs erstellt werden. Es soll eine Auswahl von vier Betriebssystemen zur Wahl stehen. Gefordert sind Windows Server 2016, Windows Server 2012R2, Windows 10 sowie Linux 16.04 LTS. Es soll die Möglichkeit geben neben dem Namen auch die gewünschte Größe an CPUs und RAM anzugeben. Als Alternative soll es neben der Angabe dieser Parameter die Auswahl einer Default-VM geben, bei der die Parameter bereits feststehen und nur der Name der VM angegeben wird. Dabei soll es sich um eine VM mit Windows Server 2016 handeln, welcher zwei virtuelle CPUs, 4 GB RAM sowie 20 GiB besitzt.

3.3 Projektschnittstellen

Bei der Durchführung des Projektes ergab sich die Schnittstelle mit der internen IT, welche mir Preise für Microsoft Azure genannt hat. Für AWS habe ich eine Anfrage an die BITS gestellt, da sich diese in der Vorbereitungsphase zur Nutzung von AWS befinden.

3.4 Zeitplanung

Für die Zeitplanung habe ich das Projekt in drei Phasen aufgeteilt. In folgender Tabelle ist eine grobe Zeitplanung zu finden. Eine detaillierte Auflistung ist in Anlage 2 zu finden.

Projektphase	**Zeitplanung in Stunden h**
Planungsphase	7,5
Durchführungsphase	18,5
Abschlussphase	9
Gesamt	**35**

1 Zeitplanung

3.5 Vergleiche und Auswertungen

Um die Angebote zu vergleichen, habe ich wie in Kapitel 3.3 erwähnt die Preise von der internen IT und der BITS angefragt. Alle weiteren Informationen habe ich über die offiziellen Seiten von Microsoft und Amazon herausgefunden.

3.5.1 Vergleich Cloud-Provider

Wie in 3.2 Soll-Konzept beschrieben, sollen verschiedene Kriterien bei der Auswahl des Cloud-Providers berücksichtigt werden. Dazu habe ich die möglichen Provider in einer Nutzwertanalyse verglichen. Diese können sich hinsichtlich Abrechnungsmodell sowie den Kosten für entsprechende VMs mit einem Windows- oder Linux-Betriebssystem unterscheiden. Besonders wichtig sind dabei eine hohe Verfügbarkeit, um die eigenen Daten permanent verfügbar zu haben und einen Datenverlust möglichst gering zu halten. Auch das Modell der Kostenabrechnung ist relevant für die Entscheidung. Daneben werden auch die Vertrautheit im Betrieb und die Lizenzkosten berücksichtigt, die jedoch nur einen kleinen Teil einnehmen.

3.5.2 Entscheidung Cloud-Provider

In der folgenden Nutzwerttabelle werden die vom Kunden ausgewählten Cloud-Provider Microsoft Azure und AWS verglichen. Die einzelnen Kriterien werden mit einem Gewichtsfaktor in Prozent angegeben. Je höher der Faktor dabei ist, desto entscheidender ist das entsprechende Kriterium. Die Bewertung liegt bei 1 bis 5, wobei 5 als optimale Bewertung gilt.

		Microsoft Azure		Amazon Web Services	
Kriterien	**Gewichtung**	**Bewertung**	**Gesamt**	**Bewertung**	**Gesamt**
Kostenmodell	20	5	100	5	100
Kosten einer VM (bei 2 CPU, 4 GB RAM) mit Windows	15	3	45	4	60
Kosten einer VM (bei 2 CPU, 4 GB RAM) mit Linux	15	2	30	4	60
Vertrautheit des Providers im Betrieb (Anbieter)	10	5	100	1	10
Lizenzkosten	5	5	25	4	20
Datenverfügbarkeit	20	5	100	5	100
durchschnittliche Storagekosten in $ pro GB	15	4	60	3	45
Gesamt	**100**	24	**410**	22	**395**

2 Nutzwertanalyse

Bei der Bewertung und Entscheidung der Cloud-Provider hat sich herausgestellt, dass Microsoft Azure die bessere Wahl darstellt. Trotz höherer Kosten, wie in Anlage 3 dargestellt, für die virtuellen Maschinen hat dieser Provider überzeugt. Hervorzuheben ist vor allem die starke Vertrautheit im Unternehmen, da Microsoft im Betrieb bereits stark verwendet wird. Da auch die virtuellen Maschinen hauptsächlich für Windows-Maschinen gedacht sind, können hier wiederum Kosten im Lizenzmanagement gespart werden, da Microsoft hier verschiedene Angebote für Kunden, die bereits Windows-Lizenzen besitzen und regelmäßig nutzen, bereithält. Das Kostenmodell für virtuelle Maschinen ist von beiden Providern grundlegend identisch. So muss am Ende nur für die genutzten CPUs und RAM, die aktuell verwendet wurden, gezahlt werden. D. h. sind die Maschinen ausgeschaltet, fallen dafür keine Kosten an. Die Storagekosten werden pro belegten GB berechnet, d. h. diese werden auch bei ausgeschalteten Maschinen berechnet. Da diese Kosten günstiger als bei Amazon Web Services sind, wie in Anlage 4 zu sehen, können auch hier Kosten gespart werden. Eine hohe Datenverfügbarkeit ist bei beiden Providern gesichert. Aus den genannten Argumenten habe ich mich für die Microsoft Azure Cloud entschieden.

3.5.3 Entscheidung Skript-Sprache

Zur Automatisierung der Erstellung soll eine Skriptart verwendet werden, die gut mit Microsoft Azure kompatibel ist. Im Portal von Azure ist eine Konsole für Bash und PowerShell integriert. Aufgrund der Tatsache, dass PowerShell als Standard auf Windows-Rechnern vorinstalliert ist und das Skript somit von jedem Rechner des Betriebes ohne große Aufwände ausführbar ist, habe ich mich für diese Skript-Sprache entschieden. Zusätzlich stellt Azure für PowerShell ein Paket an Azure-Befehlen zur Verfügung um die Arbeit mit Azure zu vereinfachen, welches gleichzeitig die Möglichkeiten der Konfiguration u. ä. stark erhöht.

3.6 Vorstellung des ausgewählten Cloud-Providers und der Skript-Sprache beim Kunden

Bei einem Treffen mit dem Kunden habe ich den ausgewählten Cloud-Provider sowie die daraus folgende Skript-Sprache vorgestellt. Dabei habe ich erläutert, welche Alternative es gibt und aus welchem Grund ich mich für die jeweilige Lösung entschieden habe. Dabei habe ich kurz die Funktionen und Vorteile erläutert, welche dieser Lösungsansatz gegenüber anderen Möglichkeiten bietet. Der Kunde teilte meine Meinung und erteilte uns den verbindlichen Auftrag für die Realisierung des Projektes.

4. Durchführungsphase

4.1 Grundlegendes

Um ein tieferes Verständnis für die folgenden Kapitel und Schritte schaffen zu können, möchte ich hier einen kurzen Überblick über PowerShell und Azure geben. Dabei beschränke ich mich auf die für die Dokumentation relevanten Inhalte und beziehe mich auf die Standardeinstellungen.

4.1.1 PowerShell

In PowerShell werden Parameter immer mit einem Dollarzeichen ($) eingeleitet und werden rot dargestellt. Diese müssen zu Beginn nicht deklariert werden und können je nach Bedarf auch inmitten des Skriptes erstellt werden. Zeilen, die mit einem Rautezeichen beginnen (#), werden beim Ausführen nicht gelesen und entsprechen somit einer Kommentarzeile, welche in grün dargestellt wird. Die Befehle werden zeilenweise gelesen und das Ende der Zeile wird nicht markiert. Wird der Befehl in der nächsten Zeile fortgesetzt, wird das Ende der vorherigen Zeile mit einem ‘ versehen. Bei vielen Befehlen, in blau dargestellt, können noch weitere Parameter mitgegeben werden. Mit einem Spiegelstrich (-) kann jeder Parameter eingeleitet und definiert werden. Diese weiteren Parameter werden in lila dargestellt. Zum besseren Arbeiten verwende ich die PowerShell ISE um Skripte flexibel verändern und ausführen zu können.

4.1.2 Azure

In Azure werden Subscriptions verwendet um Services, Lizenzen etc. abrechnen zu können. Subscriptions stellen somit eine eindeutige Zeichenfolge zu Abrechnungszwecken dar. Man muss also mindestens einer Subscription zugeordnet sein, um in Azure arbeiten zu können. Alle Objekte, die man benötigt, müssen einer Ressourcen-Gruppe zugeordnet werden, die wiederum einer Subscription unterstellt ist. Man kann sich das Ganze somit wie eine Baumstruktur vorstellen die jeweils von einer Subscription ausgeht. Alle Ressourcen werden einer Lokation, z. B. West-Europa oder Ost-USA, zugeordnet, in der sie erstellt werden. Weitere Regelungen und Erläuterungen werden sich ggf. im Laufe der Dokumentation ergeben.

4.2 Voraussetzungen zum Programmieren

Damit virtuelle Maschinen in Azure erstellt werden können, wird ein Zugriff auf das Azure Portal vorausgesetzt. Dieses habe ich über das interne Bestellsystem geordert. Anschließend habe ich mir die uns zur Verfügung stehende Subscription „Enterprise“ zuweisen lassen. Das in Kapitel 3.5.3 erwähnte Paket an Azure-Befehlen muss ebenfalls noch lokal installiert werden, um Skripte auf dem jeweiligen Rechner ausführen zu können. Um diese Befehle ausführen zu können wird mindestens PowerShell 5.0 vorausgesetzt. Anschließend muss überlegt werden, welche Parameter für die Erstellung einer VM benötigt werden.

4.2.1 Installation PowerShell 5.0

Über die offizielle Internetseite von Microsoft[2] habe ich die Dateien für PowerShell 5.0 heruntergeladen und lokal installiert. Es sind keine weiteren Installationen oder Einstellungen nötig, beim nächsten Öffnen von PowerShell wird direkt die neueste Version geladen.

4.2.2 Installation Azure PowerShell

Über die offizielle Internetseite Microsofts über Azure[3] ist in der Rubrik Befehlszeilentools die Windows-Installation für Azure vorhanden. In dieser Installation ist das PowerShellGet-Modul enthalten, welches Voraussetzung für die Installation von den Azure-Befehlen ist, sowie die Befehle an sich. Dieses Paket habe ich heruntergeladen und installiert. Anschließend habe ich die Installation überprüft.

```
PS C:\Windows\System32\WindowsPowerShell\v1.0> Get-Module -Name azureRM -ListAvailable

    Directory: C:\Program Files\WindowsPowerShell\Modules\tools

ModuleType Version    Name                                ExportedCommands
---------- -------    ----                                ----------------
Script     5.0.1      AzureRM
```

1 Überprüfung Azure-Installation

4.2.3 Kriterien für das Automatisierungs-Skript

Nach Recherchen zur Erstellung einer virtuellen Maschine in Azure haben sich für mich folgende Parameter ergeben, die im zu erstellenden Skript definiert bzw. konfiguriert werden müssen:

- Subscription
- Name der VM
- Ressourcengruppe (in Azure ResourceGroup)
- Subnetz (in Azure Subnet)
- Größe der VM
- Öffentliche IP-Adresse (in Azure publicIP)
- Lokation (in Azure location)
- Virtuelle Netzwerkkarte (in Azure nic)
- Netzwerksicherheitsgruppe (in Azure Network security group, kurz nsg)

4.3 Erstellung des Automatisierungs-Skriptes

4.3.1 Auswahl Default- oder Individuelle VM

Zu Beginn des Skriptes soll zunächst eine Auswahl für den User erscheinen, in der er zwischen der Default-VM und einer individuell erstellbaren VM entscheiden soll. Ich habe somit zunächst eine Abfrage erstellt, welche Art von VM gewählt werden soll, mit der Information um welche VM es sich bei einer Default-VM handelt. Wählt der User die Default-VM, wird das entsprechende Skript für eine Default-VM aufgerufen. Andernfalls findet eine erneute Abfrage statt, um welches Betriebssystem es sich bei der individuellen VM handeln soll. Wie vom Kunden gewünscht gibt es die Möglichkeit einen Windows Server 2016, Windows Server 2012 R2, Windows 10 oder Ubuntu Linux 16.04 auszuwählen. Je nach gewähltem Betriebssystem wird eine Variable entsprechend gesetzt, um später das richtige Betriebssystem laden zu können. Anschließend wird das Skript zur Erstellung der individuellen VM ausgeführt. Das Skript ist in der Anlage 5 zu finden.

[2] https://www.microsoft.com/en-us/download/details.aspx?id=50395, aufgerufen am 16.03.2018
[3] https://azure.microsoft.com/de-de/downloads/, aufgerufen am 16.03.2018

4.4 Erstellung einer neuen VM

Zunächst werde ich ein Skript erstellen, das all die in Kapitel 4.2.3 genannten Parameter definiert. Dafür erstelle ich zu jedem Parameter einen entsprechenden Befehl. Dabei gehe ich zunächst davon aus, dass noch keinerlei Parameter in der Cloud existieren und somit alle Parameter noch unbelegt sind. Ich beginne dabei mit dem Standard-Skript, das bedeutet, dass die meisten Werte von mir fest vorgegeben werden und es sich somit immer um die gleiche VM bei der Ausführung handeln wird. Anschließend erstelle ich ein Skript für die Erstellung einer individuellen VM.

4.4.1 Vorzudefinierende Werte für eine Default-VM

Für dieses Skript werden die folgenden Parameter zu Beginn des Skiptes zunächst definiert, um Anpassungen später schneller und einfacher durchführen zu können. Dabei handelt es sich zunächst lediglich um die Bezeichnung dieser Parameter. Ich werde jeweils den PowerShell-Befehl angeben sowie eine kurze Erläuterung.

`$subscription = Read-Host -Prompt "ID der zu verwendenden Subscription"`	Zu verwendende Subscription wird vom User eingegeben
`$vmname = Read-Host -Prompt "Name der VM"`	Der Name der VM wird vom User eingegeben
`$subnet = "Default_Subnet"`	Subnetz wird definiert
`$vmsize = 'Standard_A2'`	Größe der VM wird definiert
`$resourceGroup = 'Default_VM'`	Ressourcengruppe wird definiert
`$location = 'WestEurope'`	Lokation wird definiert
`$vnet = "Default_vnet"`	Virtuelles Netzwerk wird definiert
`$nic = "$vmName"+"_nic"`	Netzwerkkarte wird definiert
`$nsg = "Default_nsg"`	Netzwerksicherheitsgruppe wird definiert

Hinter „Standard_A2“ steht bei Azure eine VM mit 2 CPUs, 4 GB RAM und einem nicht flüchtigen Speicher von 50 GiB. Somit wird diese Anforderung vom Kunden erfüllt bzw. teilweise überschritten.

4.4.1 Zu übergebende Werte für eine individuelle VM

Wie die Bezeichnung schon aussagt, werden bei dieser Form der Erstellung die Werte individuell vom User gesetzt. Daher werden hier lediglich die Bezeichnung der Netzwerkkarte sowie die Lokation vordefiniert. Die Netzwerkkarte stellt lediglich eine formelle Eingabe für jede einzelne VM dar und hat keine weitere Relevanz. Die Lokation soll ebenfalls nicht verändert werden, da der Kunde vorgibt alle Ressourcen in West-Europa laufen zu lassen. Somit werden hier zunächst folgende Abfragen und Angaben gemacht:

$subscription = Read-Host -Prompt "ID der zu verwendenden Subscription"	Zu verwendende Subscription wird vom User eingegeben
$vmname = Read-Host -Prompt "Name der VM"	Der Name der VM wird vom User eingegeben
$subnet = Read-Host -Prompt "Name des Subnetzes"	Subnetz wird vom User eingegeben
$vmsize = Read-Host -Prompt "Grösse der VM"	Größe der VM wird vom User eingegeben
$resourceGroup = Read-Host -Prompt "ResourceGroup"	Ressourcengruppe wird vom User eingegeben
$location = “westeurope"	Lokation wird definiert
$vnet = Read-Host -Prompt "Name des Netzwerkes"	Virtuelles Netzwerk wird vom User eingegeben
$nic = "$vmName"+"_nic"	Netzwerkkarte wird definiert
$nsg = Read-Host -Prompt "Name der NSG"	Netzwerksicherheitsgruppe wird vom User eingegeben

4.4.2 Befehle zur Erstellung einer Windows-Maschine

Nach der Definition sollen nun die einzelnen Komponenten erstellt werden, die zunächst für eine Windows-Maschine benötigt werden. Dafür werden folgende Befehle verwendet:

1. Zunächst muss in die angegebene Subscription gewechselt werden. Dafür wird folgender Befehl verwendet.

   ```
   Select-AzureRmSubscription -SubscriptionName $subscription
   ```

2. Die benötigte Ressourcen-Gruppe wird erstellt, indem man den Namen der Gruppe sowie die gewählte Lokation übergibt.

   ```
   New-AzureRmResourceGroup -Name $resourceGroup -Location $location
   ```

3. Das Subnetz wird erstellt und konfiguriert, indem man dem Subnetz einen Namen vergibt sowie einen Adressbereich mit Subnetzmaske definiert.

   ```
   $subnetConfig = New-AzureRmVirtualNetworkSubnetConfig -Name $Subnet ‘
   -AddressPrefix 192.168.x.x/24
   ```

4. Die öffentliche Adresse wird erstellt mit den Angaben der Ressourcen-Gruppe sowie Lokation. Zusätzlich wird die Methode auf statisch gesetzt und der TimeOut wird auf vier Minuten gesetzt. Diese wird dann in der Variable $pip gespeichert.

```
$pip = New-AzureRmPublicIpAddress -ResourceGroupName $resourceGroup ‘
-Location $location -AllocationMethod Static -IdleTimeoutInMinutes 4 ‘
-Name "mypublicdns$(Get-Random)"
```

5. Das virtuelle Netzwerk wird erstellt, indem die Ressourcen-Gruppe, die Lokation, der zukünftige Name des Netzwerkes sowie das konfigurierte Subnetz angegeben werden. Außerdem wird der Adresspräfix angegeben. Diese wird dann in der Variable $Network gespeichert.

```
$Network = New-AzureRmVirtualNetwork -ResourceGroupName $resourceGroup ‘
-Location $location -Name $vnet -AddressPrefix 192.168.x.x/16
-Subnet $subnetConfig
```

6. Es müssen verschiedene Regeln erstellt werden, um Portfreigaben zu ermöglichen, damit eine Kommunikation mit der VM möglich ist. Dafür habe ich eine Regel $nsgRuleRDP für den Port 3389 (RDP) und $nsgRuleWeb für Port 80 (http) erstellt und diese anschließend in der Variablen $SecurityGroup zusammengefasst.

```
$nsgRuleRDP = New-AzureRmNetworkSecurityRuleConfig ‘
-Name NetworkSecurityGroupRuleRDP  -Protocol Tcp `
-Direction Inbound -Priority 1000 -SourceAddressPrefix * -SourcePortRange * ‘
-DestinationAddressPrefix * -DestinationPortRange 3389 -Access Allow

$nsgRuleWeb = New-AzureRmNetworkSecurityRuleConfig ‘
-Name NetworkSecurityGroupRuleWWW  -Protocol Tcp `
-Direction Inbound -Priority 1001 -SourceAddressPrefix * -SourcePortRange * ‘
-DestinationAddressPrefix * -DestinationPortRange 80 -Access Allow

$SecurityGroup = New-AzureRmNetworkSecurityGroup ‘
-ResourceGroupName $resourceGroup -Location $location `
-Name $nsg -SecurityRules $nsgRuleRDP,$nsgRuleWeb
```

7. Die virtuelle Netzwerkkarte wird erstellt, indem ich den zukünftigen Namen, die Ressourcen-Gruppe, die Lokation, das erste Subnets im Netzwerk, die ID der öffentlichen Adresse sowie die IP der Security-Gruppe angebe. Diese wird dann in der Variable $networkcard gespeichert.

```
$networkcard = New-AzureRmNetworkInterface -Name $nic ‘
-ResourceGroupName $resourceGroup -Location $location ‘
-SubnetId $Network.Subnets[0].Id -PublicIpAddressId $pip.Id ‘
-NetworkSecurityGroupId $SecurityGroup.Id
```

8. Um alle Parameter zusammenzufassen werden diese in einer einzigen Variable gespeichert. Dafür wird der Name, die Größe der Maschine, das Betriebssystem, der Name des Rechners, die Daten des ersten Nutzers und des Herausgebers angegeben.

```
$vmConfig = New-AzureRmVMConfig -VMName $vmName -VMSize $vmsize | ‘
Set-AzureRmVMOperatingSystem -Windows -ComputerName $vmName ‘
 Set-AzureRmVMSourceImage -PublisherName MicrosoftWindowsServer ‘
-Offer WindowsServer Skus 2016-Datacenter -Version latest | ‘
Add-AzureRmVMNetworkInterface -Id $networkcard.Id
```

Nachdem alle benötigten Parameter erstellt wurden, kann nun der Befehl zur Erstellung der virtuellen Maschine erfolgen. Da bereits alle notwendigen Angaben in der Variablen $vmConfig gespeichert wurden, sind hier nur noch wenige Parameter zu übergeben.

9. Die virtuelle Maschine wird erstellt, indem die Ressourcen-Gruppe, die Lokation und alle weiteren Parameter, die in $vmConfig gespeichert sind, übergeben werden.

```
New-AzureRmVM -ResourceGroupName $resourceGroup -Location $location ‘
-VM $vmConfig
```

Fügt man nun alle Parameter und Befehle aus 4.4.1 bzw. 4.4.2 und diesem Kapitel zusammen, ergibt sich das vollständige Skript für die Erstellung einer virtuellen Maschine mit den vom Kunden angegebenen Anforderungen. Zusätzlich muss noch der Befehl „Login-AzureRmAccount“ eingefügt werden, bevor die Befehle ausgeführt werden können, damit eine Verbindung zu Azure hergestellt werden kann.

Das gesamte Skript zum aktuellen Zeitpunkt ist unter Anlage 6 zu finden. Die Abbildung entspricht dabei der Default-VM. Für die individuelle VM muss lediglich die Parameterbezeichnung angepasst werden.

4.5 Erweiterungen der Skripte

Die erstellten Skripte ermöglichen nun das Auswählen der Default-VM oder individuellen VM. Bei letzterer gibt es zusätzlich die Auswahl zwischen mehreren Betriebssystem. Anschließend wird diese VM erstellt. Alle Befehle erstellen dabei jeweils eine neue Komponente. Es muss nun auch beachtet werden, besonders bei der Default-VM, dass einige Parameter wie Ressourcen-Gruppe, Security-Gruppe etc. bereits existieren können. Ebenfalls ist bei den individuellen VMs zu beachten, dass verschiedene Betriebssysteme zur Auswahl stehen und jeweils die korrekten Parameter verwendet werden. Zusätzlich wird dort auch eine Linux-Maschine angeboten. Dafür müssen ein SSH-Schlüssel sowie eine Regel zur Freigabe des SSH-Ports erstellt werden. Auch ein Admin-User muss beim Anlegen der VM erstellt werden. Außerdem sind nicht alle Größen an VMs in allen Lokationen verfügbar, sodass auch dies überprüft werden muss. Diese Punkte sollen in den folgenden Kapiteln umgesetzt werden.

4.5.1 Überprüfung bereits vorhandener Komponenten

Die folgenden Schritte werden in beiden Skripten durchgeführt.

4.5.1.1 Abfrage der eingegeben Parameter

Um das Erstellen der Komponenten zu umgehen, wenn diese bereits bestehen, wird zunächst überprüft, ob diese Komponente bereits existiert. Das Ergebnis wird jeweils in einer neuen Variablen gespeichert. Gibt der Befehl keinen Wert zurück, soll es keine Fehlermeldung geben um Verwirrung beim User zu vermeiden, da diese in dem Moment korrekt ist, und das Skript soll ohne Meldung weiter ausgeführt werden. Diese Abfrage sieht wie folgt aus:

```
#Abfragen der eingegebenen Parameter ob diese schon existieren + in Parameter speichern
$Group = Get-AzureRmResourceGroup -Name $resourceGroup
$VM = Get-AzureRmVM -ResourceGroupName $resourceGroup -Name $vmName
$Network = Get-AzureRmVirtualNetwork -Name $vnet -ResourceGroupName $resourceGroup
$SecurityGroup = Get-AzureRmNetworkSecurityGroup -Name $nsg -ResourceGroupName $resourceGroup
$networkcard = Get-AzureRmNetworkInterface -Name $nic -ResourceGroupName $resourceGroup
$Subnetz = Get-AzureRmVirtualNetworkSubnetConfig -Name $subnetname -VirtualNetwork $Network
```

2 Abfrage auf bestehende Komponenten

Nahezu alle Komponenten sind, wie im Befehl zu erkennen, von der Ressourcen-Gruppe abhängig, da es jede Komponente innerhalb einer Ressourcen-Gruppe nur einmal geben darf. In einer anderen Gruppe wäre der Name ggf. möglich. Eine Ausnahme bildet dabei nur das Subnetz, das vom Netzwerk abhängig ist, welches wiederum von der Ressourcen-Gruppe abhängig ist.

4.5.1.2 Überschreibungen verhindern

Um nun zu verhindern, dass bereits bestehende Komponenten überschrieben werden, muss um jeden Befehl aus Kapitel 4.4.2 eine if-Anweisung gestellt werden. Mit Ausnahme des Namen ($VM) wird bei jeder if-Abfrage geschaut, ob die Variable, die die Bezeichnung überprüft, leer ist, sodass es den eingegebenen Parameter noch nicht gibt und er erstellt wird. Andernfalls wird diese Anweisung übersprungen. Die Abfrage sieht, am Beispiel der Ressourcen-Gruppe, wie folgt aus:

```
if($Group -eq $null)
{
    New-AzureRmResourceGroup -Name $resourceGroup -Location $location
}
```

3 if-Abfrage nach existierender Ressource

Die Variable $VM, die den Namen der VM überprüft, muss im Gegensatz zu den anderen Komponenten leer sein, da es jeden Namen pro Ressourcen-Gruppe nur einmal geben darf. Existiert dieser Name schon, muss somit ein neuer Name eingegeben werden, bis dieser eindeutig ist. Auch der Name der Netzwerkkarte muss anschließend an den neuen Namen angepasst werden. Um dies umzusetzen habe ich an dieser Stelle eine while-Schleife verwendet.

```
while ($VM -ne $null)
{
    echo "schon vorhanden, bitte neu wählen"
    $vmName = Read-Host -Prompt "Neuer Name"
    $nic = "$vmName"+"_nic"
    $VM = Get-AzureRmVM -Name $vmName -ResourceGroupName $resourceGroup

}
```

4 Überprüfung Name der VM

4.5.1.3 Admin-User anlegen

Um sich nach der Erstellung erstmalig auf der VM anmelden zu können müssen bei der Erstellung Anmeldedaten für den Admin-User definiert werden. Der Code wird in beiden Skripten hinzugefügt und sieht wie folgt aus:

```
$securePassword = ConvertTo-SecureString '[redacted]' -AsPlainText -Force
$cred = New-Object System.Management.Automation.PSCredential ("azureuser", $securePassword)
```

5 Admin-User erstellen

4.5.2 Ausgewähltes Betriebssystem laden

Dieses Kapitel bezieht sich lediglich auf das Skript zur Erstellung einer individuellen VM, da nur dort verschiedene Betriebssysteme zur Verfügung stehen müssen. Wie bereits erwähnt, stehen für die individuellen VMs die Betriebssysteme Windows Server 2016, Windows Server 2012 R2, Windows 10 und Ubuntu Linux 16.04 zur Verfügung.

4.5.2.1 Auswahlmöglichkeit eines Windows-Betriebssystems

In Kapitel 4.3.1 Auswahl Default- oder individuelle VM wurde die Variable $skus je nach Wahl des Betriebssystems definiert. Diese Variable wird nun in einer if-Abfrage verwendet, um die entsprechende Konfiguration auszuwählen. Für jedes Betriebssystem sind dafür eigene Parameter nötig, die ich über Recherchen im Internet herausgesucht habe. Dadurch ergeben sich für jedes Betriebssystem eigene zu übergebende Parameter. Dafür habe ich ifelse-Verknüpfungen erstellt, die jeweils die Variable $skus abgleichen und dann die entsprechende $vmconfig verwenden. Für die Windows-Systeme ergibt sich somit folgender Code:

```
if($skus -eq '2016-Datacenter')
{
    $vmConfig = New-AzureRmVMConfig -VMName $vmName -VMSize $vmsize | `
    Set-AzureRmVMOperatingSystem -Windows -ComputerName $vmName -Credential $cred | `
    Set-AzureRmVMSourceImage -PublisherName MicrosoftWindowsServer -Offer WindowsServer `
    -Skus 2016-Datacenter -Version latest | Add-AzureRmVMNetworkInterface -Id $networkcard.Id
}
elseif($skus -eq '2012_R2')
{
    $vmConfig = New-AzureRmVMConfig -VMName $vmName -VMSize $vmsize | `
    Set-AzureRmVMOperatingSystem -Windows -ComputerName $vmName -Credential $cred | `
    Set-AzureRmVMSourceImage -PublisherName MicrosoftWindowsServer -Offer WindowsServer `
    -Skus 2012-R2-Datacenter -Version latest | Add-AzureRmVMNetworkInterface -Id $networkcard.Id
}
elseif($skus -eq 'Windows-10-N-x64')
{
    $vmConfig = New-AzureRmVMConfig -VMName $vmName -VMSize $vmsize | `
    Set-AzureRmVMOperatingSystem -Windows -ComputerName $vmName -Credential $cred | `
    Set-AzureRmVMSourceImage -PublisherName MicrosoftVisualStudio -Offer Windows-10 `
    -Skus RS2-Pro -Version latest | Add-AzureRmVMNetworkInterface -Id $networkcard.Id
}
```

6 Konfiguration der Windows-Systeme

4.5.2.2 Auswahlmöglichkeit eines Linux-Betriebssystems

Die Konfiguration des Linux-Betriebssystems entspricht grundsätzlich dem Vorgehen wie in Kapitel 4.4.2 mit entsprechenden Parametern für Ubuntu Linux 16.04 LTS und wird ebenfalls nur im Skript für die individuelle VM verwendet. Dieser Befehl sieht entsprechend wie folgt aus:

```
elseif($skus -eq '16.04-LTS')
{
    $vmConfig = New-AzureRmVMConfig -VMName $vmName -VMSize $vmsize | `
    Set-AzureRmVMOperatingSystem -Linux -ComputerName $vmName -Credential $cred -DisablePasswordAuthentication | `
    Set-AzureRmVMSourceImage -PublisherName Canonical -Offer UbuntuServer -Skus 16.04-LTS -Version latest | `
    Add-AzureRmVMNetworkInterface -Id $networkcard.Id
```

7 Konfiguration des Linux-Systems

Zusätzlich wird für die Linux-Maschine ein SSH-Zugang benötigt. Der User ist für die

Konfiguration eines SSH-Schlüssels zuständig und muss diesen vor Ausführung des Skriptes speichern. Der User wird bei Auswahl des Linux-Betriebssystems darauf hingewiesen und erhält einen Link der Microsoft-Seite als Unterstützung, falls er noch keinen Schlüssel besitzt. Anschließend wird das Skript beendet bis er bei erneutem Aufruf angibt einen Schlüssel zu besitzen. Diese Nachfrage wird in das Skript zur Abfrage aus Kapitel 4.3.1 hinzugefügt und ergibt sich wie folgt:

```
$titel = 'Kontrolle des SSH-Schlüssels'
$message = 'Ist ein SSH-Schlüssel vorhanden?
Eine Installation kann über https://docs.microsoft.com/de-de/azure/virtual-machines/linux/mac-create-ssh-keys erfolgen.'
$ssh = New-Object System.Management.Automation.Host.ChoiceDescription "&SSH", "SSH-Schlüssel ist vorhanden."
$keinssh = New-Object System.Management.Automation.Host.ChoiceDescription "&Kein SSH", "Es ist noch kein SSH-Schlüssel vorhanden."
$options = [System.Management.Automation.Host.ChoiceDescription[]]($ssh,$keinssh)
$schluessel=$host.ui.PromptForChoice($title, $message, $options, 0)

if($schluessel -eq 0)
{
   $skus = '16.04-LTS'
}
else
{
   echo "Bitte SSH-Schlüssel installieren und Skript anschließend erneut ausführen."
   Exit
```

8 Abfrage SSH-Schlüssel

In dem Skript wird in der Konfiguration der Linux-Maschine nun der SSH-Schlüssel entsprechend abgefragt und an die Variable $vmconfig angehängt. Dadurch ergibt sich der ergänzende Code wie folgt:

```
#Suche SSH Keys
$sshPublicKey = Get-Content "$env:USERPROFILE\.ssh\id_rsa.pub"
Add-AzureRmVMSshPublicKey -VM $vmconfig -KeyData $sshPublicKey -Path "/home/azureuser/.ssh/authorized_keys"
```

9 Konfiguration SSH-Schlüssel

4.5.3 Überprüfung der VM-Größe in gewählter Lokation

Nicht alle Größen der VMs sind in allen angebotenen Lokationen von Azure möglich. So ist z. B. die Größe „Standard DS2“ nicht in West-Europa möglich. Um Fehler zu vermeiden muss hier eine Überprüfung und ggf. eine Anpassung erfolgen. Dafür verwende ich eine while-Schleife, bei der jeweils überprüft wird, ob die angegebene Größe in der gewünschten Lokation möglich ist. Ist diese Kombination nicht möglich, wird der User zu einer Eingabe einer anderen Größe aufgefordert. Eine Anpassung der Lokation ist hierbei wie bereits in Kapitel 4.4.1 erwähnt vom Kunden nicht gewünscht. Diese Anpassung muss nur im Skript für die individuelle VM eingetragen werden, da diese Parameter bei der Default-VM bereits fest vordefiniert sind. Der Code wird wie folgt im Skript eingefügt:

```
while ((Get-AzureRmVMSize -Location $location | Select Name | Where Name -eq $vmsize) -eq $null)
{
    echo "Diese Grösse der VM ist in dieser Lokation leider nicht verfügbar. Bitte wählen Sie eine passende Grösse"
    $vmsize = Read-Host -Prompt "Size"
    $Test = Get-AzureRmVMSize -Location $location | Select Name | Where Name -eq $vmsize
}
```

10 Überprüfung VM-Größe in Lokation

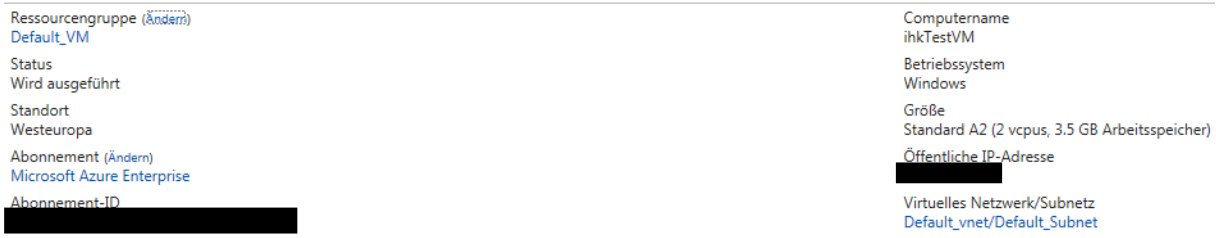
Nach diesen Anpassungen sind die Skripte fertig und können in den Anlagen 7-10 vollständig betrachtet werden.

4.6 Ausführliche Tests

Nach der Erstellung, Konfiguration und Ergänzung aller Skripte habe ich diese ausgiebig getestet. Ich habe das Abfrage-Skript geprüft, indem ich die verschiedenen Auswahlmöglichkeiten probiert habe. Dabei verliefen alle Abfragen wie geplant und es wurden immer die richtigen Skripte aufgerufen. In der Anlage 11 habe ich einige Screenshots

hinterlegt.

Nach jeder Erstellung habe ich die VMs im Azure-Portal auf ihre Korrektheit geprüft. Dabei sind keine Fehler aufgetreten.



11 Kontrolle Parameter Azure-Portal

Ich habe mehrere Default-VMs erstellt, um das Verhalten bei bestehenden Parametern zu überprüfen. Das Ergebnis war, wie erwartet, das Hinzufügen der neuen VM zu bestehenden Komponenten. Ich habe individuelle VMs erstellt und dabei sowohl Windows- als auch Linux-Betriebssysteme verwendet. Auch hier ergaben sich die Ergebnisse wie erwartet. Das Anmelden an den erstellten VMs mit den gesetzten Admin-Daten über RDP bzw. SSH hat immer funktioniert. Auch der Zugang zum Internet über die VMs funktioniert ohne Probleme.

4.7 Zusammenfassung

Es liegen nun drei Skripte vor, die allen Anforderungen des Kunden gerecht werden. Diese habe ich endgültig benannt mit „VM_Erstellung.ps1“, in der sich die Abfrage befindet. Des Weiteren gibt es anschließend die Dateien „Ablauf_Default_VM.ps1“ sowie „Ablauf_Indiv_VM.ps1“, die jeweils vom ersten Skript aufgerufen werden. Der gesamte Code findet sich in der Anlage 8 für die Default-VM sowie Anlage 9 und 10 für die individuelle VM. Die Abfrage kann, wie bereits erwähnt, in Anlage 7 eingesehen werden.

5. Abschlussphase

Zum Abschluss habe ich das Projekt rückwirkend bewertet, indem ich eine Kostenkalkulation erstellt sowie meine Zeitenplanung auf Korrektheit überprüft habe. Anschließend gebe ich in einem Fazit einen Ausblick auf die Weiterentwicklung dieses Projektes.

5.1 Kostenkalkulation

5.1.1 Personalkosten

Person/Abteilung	Gewichtung	Einzelkosten €/h	Gesamtkosten in €
Jessica Schmieder	35 h	75,00	2.625,00

3 Personalkosten

5.1.2 Nutzungsgebühr

Für die Qualitätssicherung habe ich durch Testen verschiedener Szenarien diverse VMs erstellt und getestet. Dadurch ergaben sich für dieses Projekt folgende Kosten, die über das Azure-Portal ermittelt werden konnten. Für Lizenzen fallen hierbei keine Kosten an.

Beschreibung	Kosten in €
Subscription Microsoft Azure Enterprise	245,17

4 Nutzungsgebühr

5.1.3 Gesamtkosten des Projektes

Kostenart	Betrag in €
Jessica Schmieder	2.625,00
Subscription Microsoft Azure Enterprise	245,17
Gesamtsumme	2.870,17

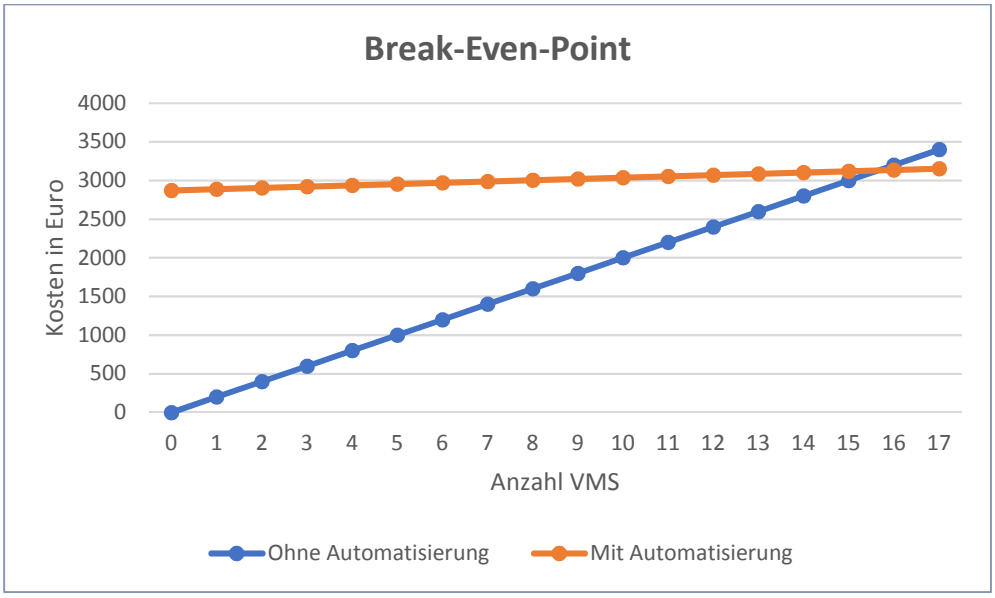
5 Gesamtkosten des Projektes

Für dieses Projekt gibt es keine fortlaufenden Kosten, da es sich hier um eine einmalige Erstellung handelt.

5.2 Kosten-Nutzen-Analyse

Bisher wurde eine VM von einem Mitarbeiter manuell konfiguriert. Diese Einrichtung dauerte je nach Bedarf ca. 2 Stunden. Ein Mitarbeiter kostet dabei pro Stunde 100,00 €, das bedeutet das Errichten kostet 200,00 € pro VM. Durch das Automatisierungs-Skript reduziert sich diese Zeit auf ca. 10 Minuten. Umgerechnet bedeutet dies einen Kostensatz von 16,67 € pro VM. Dabei muss der Einmal-Aufwand für das Skript mit 2.870,17 € berücksichtigt werden.

Im folgenden Diagramm ist die Entwicklung der Kosten auf der y-Achse pro erstellter VM auf der x-Achse wiedergegeben.



6 Break-Even-Point

In dem Diagramm ist zu erkennen das der blaue Graph (ohne Automatisierung) zwar bei 0,00 € beginnt, jedoch schneller ansteigt als der orange Graph (mit Automatisierung), welcher bei 2.870,17 € beginnt. Somit ergibt sich beim Erstellen der 16. VM ein erstmaliger Kostennutzen von 63,11 €, der sich anschließend immer weiter ausbaut. Da VMs immer häufiger verwendet werden, sind die 16 VMs leicht zu erreichen und stellen ein realistisches Ziel dar.

5.3 Soll-Ist-Vergleich

Das Projekt konnte ich innerhalb der 35 Stunden umsetzen. Dabei haben sich allerdings in einigen Punkten die geplanten Zeiten geändert.

Projektphase	Geplante Zeit in h	Tatsächliche Zeit in h	Differenz in h
Planungsphase	7,5	6	-1,5
Durchführungsphase	18,5	19	+0,5
Abschlussphase	9	10	+1
Gesamt	**35**	**35**	

7 Soll-Ist-Vergleich

Wie in der Tabelle zu erkennen ist, habe ich trotz Änderungen der Zeiten die vorgegebenen 35 Stunden einhalten können. Die Planungsphase benötigte nicht so viel Zeit wie geplant, da das Soll-Konzept schneller als geplant definiert werden konnte. Aufgrund der verschiedenen Anpassungen die an den Skripten durchgeführt werden mussten, ergab sich bei der Erstellung der Skripte eine leichte Verzögerung von 0,5 Stunden. Ebenfalls dauerte die Dokumentation aufgrund der Betriebs- und Kundendokumentation länger als vorher angenommen, sodass das Projekt letztendlich 35 Stunden benötigte. Eine ausführliche Auflistung ist in der Anlage 12 zu finden.

5.4 Abweichungen vom Projektantrag

Anders als im Projektantrag beschrieben, habe ich die Kosten-Nutzen-Analyse in der Abschlussphase durchgeführt, um so variable Kosten besser einbeziehen zu können. Stattdessen habe ich mich in der Planungsphase für eine Nutzwertanalyse zur Findung des Cloud-Providers entschieden. Zusätzlich habe ich die Zeitenplanung etwas detaillierter als im Projektantrag gegliedert, um eine bessere Übersicht zu schaffen. Ebenso ergaben sich Änderungen in der Zeitplanung, welche ich in der Tabelle 3 Soll-Ist-Vergleich gegenübergestellt habe, dabei wurden die 35 Stunden jedoch nicht überschritten.

5.5 Fazit/Ausblick

Wie im Soll-Ist-Vergleich zu sehen ist, wurde die vorgegebene Zeit von 35 Stunden trotz Änderung in der Zeitenplanung nicht überschritten. Das Skript wird auf einem Sharepoint intern zur Verfügung gestellt und kann von allen Mitarbeitern, die einen Azure-Zugang besitzen, nun verwendet werden. Da der Umgang mit Azure noch am Anfang steht, wird es viele Möglichkeiten der Weiterentwicklung und Verbesserung geben. Azure bietet viele Möglichkeiten den Nutzern schnell und effizient VMs zur Verfügung zu stellen. So ist dieses umgesetzte Projekt ein erster Meilenstein, der weiter ausgearbeitet werden wird. In welche Richtung sich die Cloud intern entwickeln wird, steht dabei noch aus.

6. Betriebs- und Kundendokumentation

Für die Betriebs- und Kundendokumentation verweise ich auf die Anlage 13 und 14.

7. Quellenverzeichnis

www.microsoft.com

www.azure.microsoft.com
www.aws.amazon.com
www.btc-ag.com

Anlagenverzeichnis

Anlage 1 Vergleich Cloud-Provider

Quelle: www.computerwoche.de, aufgerufen am 26.03.2018

Anlage 2 Detaillierte Zeitenplanung

Projektphase mit Teilaufgaben	Projektphase gesamt in h	Dauer in h
1. Planungsphase	7,5	
Ist-Analyse		1,5
Soll-Konzept		3
Nutzwertanalyse		3
2. Durchführungsphase	18,5	
Voraussetzungen überprüfen		0,5
Recherchen zur Programmierung		3
Erstmalige Erstellung der Skripte für das Aufsetzen von virtuellen Maschinen		6
Ergänzungen der Skripte		4
Fehlersuche/Fehlerbehebung		3
Testen auf Funktionalität		2
3. Abschlussphase	9	
Hauptdokumentation		6
Betriebsdokumentation		2
Kundendokumentation		2
Gesamtdauer	**35**	

Anlage 3 Kosten virtuelle Maschine

Bei den virtuellen Maschinen handelt es sich jeweils um eine Maschine mit 2 CPUs und 4 GiB RAM. Die ausgewählten Preise basieren auf möglichst gleichen Maschinen. Alle Maschinen bei Microsoft Azure erhalten 20 GiB inklusive.

	Microsoft Azure		Amazon Web Services	
Betriebssystem	**Kosten in $/h**	**Kosten in €/h**	**Kosten in $/h**	**Kosten in €/h**
Windows	0,650	0,531*	0,0716	0,058*
Linux	0,287	0,234*	0,0536	0,044*

*Umrechnung mit einem Wechselkurs von 1:0,81673 (Stand: 20.03.2018)

Anlage 4 Storagekosten

	Microsoft Azure		Amazon Web Services	
Betriebssystem	**Kosten in $/GB**	**Kosten in €/GB**	**Kosten in $/GB**	**Kosten in €/GB**
Erste 50 TB/Monat	0,0184	0,015*	0,0245	0,020*
Nächste 450 TB/Monat	0,0177	0,014*	0,0235	0,019*
Über 500 TB/Monat	0,0170	0,014*	0,0225	0,019*

*Umrechnung mit einem Wechselkurs von 1:0,81673 (Stand: 20.03.2018)

Anlage 5 Skript Abfrage Version 1

```
1
2   $defaultskript="[redacted]Default_VM.ps1"
3   $skript="[redacted]\Individuelle_Windows.ps1"
4
5   #-----------------------------------------------------------------------------------------------------
6   #Abfrage ob Default-Einstellungen verwendet werden sollen oder individuelle Eingabe
7   $titel = 'default-Einstellungen?(J/N)'
8   $message = 'Möchten Sie die Default-Einstellung verwenden? Die Default-Einstellung entspricht einer
9   Windows Server 2016 Maschine mit 2 CPUs und 4GB RAM sowie einer 50GB HDD-Festplatte.'
10  $yes = New-Object System.Management.Automation.Host.ChoiceDescription "&Ja", "Ja"
11  $no = New-Object System.Management.Automation.Host.ChoiceDescription "&Nein", "Nein"
12  $options = [System.Management.Automation.Host.ChoiceDescription[]]($yes, $no)
13  $default=$host.ui.PromptForChoice($title, $message, $options, 0)
14
15  if ($default -eq 0)
16      {
17          #Default; Skript mit vordefinierten Parametern
18          &$defaultskript
19      }
20      else
21      {
22          $titel = 'Wahl des Betriebssystems'
23          $message = 'Welches Betriebssystem soll verwendet werden?'
24          $WS2k16 = New-Object System.Management.Automation.Host.ChoiceDescription "&Windows Server 2016", "Windows Server 2016"
25          $WS2k12R2 = New-Object System.Management.Automation.Host.ChoiceDescription "Windows Server 2012 R&2", "Windows Server 2012 R2"
26          $Win10 = New-Object System.Management.Automation.Host.ChoiceDescription "Windows &10", "Windows 10"
27          $UL1604LS = New-Object System.Management.Automation.Host.ChoiceDescription "&Ubuntu Linux 16.04 LTS", "Ubuntu Linux 16.04 LTS"
28          $options = [System.Management.Automation.Host.ChoiceDescription[]]($WS2k16, $WS2k12R2,$Win10,$UL1604LS)
29          $BS=$host.ui.PromptForChoice($title, $message, $options, 0)
30
31          switch ($BS)
32          {
33              0
34              {
35                  #Windows Server 2016
36                  $skus = '2016-Datacenter'
37              }
38              1
39              {
40                  #Windows Server 2012R2
41                  $skus = '2012_R2'
42              }
43              2
44              {
45                  #Windows 10
46                  $skus = 'Windows-10-N-x64'
47
48              }
49              3
50              {
51                  #$OS='Linux'
52                  $skus = '16.04-LTS'
53              }
54          }
55         &$skript
56      }
57
```

Anlage 6 Skript Default-VM Version 1

```
#Parameter

$subscription = Read-Host -Prompt "ID der zu verwendenden Subscription"
$vmname = Read-Host -Prompt "Name der VM"
$resourceGroup = 'Default_VM'
$location = 'westeurope'
$nic = "$vmName"+"_nic"
$vnet = "Default_vnet"
$nsg = "Default_nsg"
$vmsize = 'Standard_A2'
$subnet = "Default_Subnet"

#-------------------------------------------------------------------------------------------------------------------------------
#Anmeldung

#Einloggen bei Azure
Login-AzureRmAccount

#In entsprechende Subscription wechseln
Select-AzureRmSubscription -SubscriptionName $subscription

#-------------------------------------------------------------------------------------------------------------------------------
#Erstellung der Paramter
New-AzureRmResourceGroup -Name $resourceGroup -Location $location

$subnetConfig = New-AzureRmVirtualNetworkSubnetConfig -Name $subnet -AddressPrefix 192.168.[redacted]/24

$Network = New-AzureRmVirtualNetwork -ResourceGroupName $resourceGroup -Location $location `
           -Name $vnet -AddressPrefix 192.168.[redacted]/16 -Subnet $subnetConfig

$pip = New-AzureRmPublicIpAddress -ResourceGroupName  $resourceGroup -Location $location `
       -AllocationMethod Static -IdleTimeoutInMinutes 4 -Name "mypublicdns$(Get-Random)"

$nsgRuleRDP = New-AzureRmNetworkSecurityRuleConfig -Name NetworkSecurityGroupRuleRDP  -Protocol Tcp `
              -Direction Inbound -Priority 1000 -SourceAddressPrefix * -SourcePortRange * -DestinationAddressPrefix *
              -DestinationPortRange 3389 -Access Allow


$nsgRuleWeb = New-AzureRmNetworkSecurityRuleConfig -Name NetworkSecurityGroupRuleWWW  -Protocol Tcp `
              -Direction Inbound -Priority 1001 -SourceAddressPrefix * -SourcePortRange * -DestinationAddressPrefix *
              -DestinationPortRange 80 -Access Allow


$SecurityGroup = New-AzureRmNetworkSecurityGroup -ResourceGroupName $resourceGroup -Location $location `
                 -Name $nsg -SecurityRules $nsgRuleRDP,$nsgRuleWeb

$networkcard = New-AzureRmNetworkInterface -Name $nic -ResourceGroupName $resourceGroup -Location $location `
               -SubnetId $Network.Subnets[0].Id -PublicIpAddressId $pip.Id -NetworkSecurityGroupId $SecurityGroup.Id

$vmConfig = New-AzureRmVMConfig -VMName $vmName -VMSize $vmsize | `
            Set-AzureRmVMOperatingSystem -Windows -ComputerName $vmName -Credential $cred | `
            Set-AzureRmVMSourceImage -PublisherName MicrosoftWindowsServer -Offer WindowsServer `
            -Skus 2016-Datacenter -Version latest | Add-AzureRmVMNetworkInterface -Id $networkcard.Id

New-AzureRmVM -ResourceGroupName $resourceGroup -Location $location -VM $vmConfig
```

Anlage 7 Finales Skript Abfrage („VM_Erstellung.ps1“)

```
$defaultskript="[redacted]Ablauf_Default_VM.ps1.ps1"
$skript="[redacted]Ablauf_Indiv_VM.ps1.ps1"

#-----------------------------------------------------------------------------------------------------
#Abfrage ob Default-Einstellungen verwendet werden sollen oder individuelle Eingabe
$titel = 'default-Einstellungen?(J/N)'
$message = 'Möchten Sie die Default-Einstellung verwenden? Die Default-Einstellung entspricht einer
Windows Server 2016 Maschine mit 2 CPUs und 4GB RAM sowie einer 50GB HDD-Festplatte.'
$yes = New-Object System.Management.Automation.Host.ChoiceDescription "&Ja", "Ja"
$no = New-Object System.Management.Automation.Host.ChoiceDescription "&Nein", "Nein"
$options = [System.Management.Automation.Host.ChoiceDescription[]]($yes, $no)
$default=$host.ui.PromptForChoice($title, $message, $options, 0)

if ($default -eq 0)
    {
        #Default; Skript mit vordefinierten Parametern
        &$defaultskript
    }
    else
    {
        $titel = 'Wahl des Betriebssystems'
        $message = 'Welches Betriebssystem soll verwendet werden?'
        $WS2k16 = New-Object System.Management.Automation.Host.ChoiceDescription "&Windows Server 2016", "Windows Server 2016"
        $WS2k12R2 = New-Object System.Management.Automation.Host.ChoiceDescription "Windows Server 2012 R&2", "Windows Server 2012 R2"
        $Win10 = New-Object System.Management.Automation.Host.ChoiceDescription "Windows &10", "Windows 10"
        $UL1604LS = New-Object System.Management.Automation.Host.ChoiceDescription "&Ubuntu Linux 16.04 LTS", "Ubuntu Linux 16.04 LTS"
        $options = [System.Management.Automation.Host.ChoiceDescription[]]($WS2k16, $WS2k12R2,$Win10,$UL1604LS)
        $BS=$host.ui.PromptForChoice($title, $message, $options, 0)

        switch ($BS)
        {
            0
            {
                #Windows Server 2016
                $skus = '2016-Datacenter'
            }
            1
            {
                #Windows Server 2012R2
                $skus = '2012_R2'
            }
            2
            {
                #Windows 10
                $skus = 'Windows-10-N-x64'
            }
            3
            {
                $titel = 'Kontrolle des SSH-Schlüssels'
                $message = 'Ist ein SSH-Schlüssel vorhanden?
                Eine Installation kann über https://docs.microsoft.com/de-de/azure/virtual-machines/linux/mac-create-ssh-keys erfolgen.'
                $ssh = New-Object System.Management.Automation.Host.ChoiceDescription "&SSH", "SSH-Schlüssel ist vorhanden."
                $keinssh = New-Object System.Management.Automation.Host.ChoiceDescription "&Kein SSH", "Es ist noch kein SSH-Schlüssel vorhanden."
                $options = [System.Management.Automation.Host.ChoiceDescription[]]($ssh,$keinssh)
                $schluessel=$host.ui.PromptForChoice($title, $message, $options, 0)

                if($schluessel -eq 0)
                {
                   $skus = '16.04-LTS'
                }
                else
                {
                   echo "Bitte SSH-Schlüssel installieren und Skript anschließend erneut ausführen."
                   Exit
                }
            }
        }
       &$skript
    }
```

Anlage 8 Finales Skript für Default-VM („Ablauf_Default_VM.ps1“)

```
#-------------------------------------------------------------------------------------------------------------------------------
#Parameter

$subscription = Read-Host -Prompt "ID der zu verwendenden Subscription"
$vmname = Read-Host -Prompt "Name der VM"
$resourceGroup = 'Default_VM'
$location = 'WestEurope'
$subnetname = "Default_Subnet"
$nic = "$vmName"+"_nic"
$vnet = "Default_vnet"
$nsg = "Default_nsg"
$vmsize = 'Standard_A2'

#-------------------------------------------------------------------------------------------------------------------------------
#Anmeldung

#Einloggen bei Azure
Login-AzureRmAccount

#In entsprechende Subscription wechseln
Select-AzureRmSubscription -SubscriptionName $subscription
#-------------------------------------------------------------------------------------------------------------------------------
#Überprüfungen


#Abfragen der eingegebenen Parameter ob diese schon existieren + in Parameter speichern
$Group = Get-AzureRmResourceGroup -Name $resourceGroup -ErrorAction SilentlyContinue
$VM = Get-AzureRmVM -ResourceGroupName $resourceGroup -Name $vmName -ErrorAction SilentlyContinue
$Network = Get-AzureRmVirtualNetwork -Name $vnet -ResourceGroupName $resourceGroup -ErrorAction SilentlyContinue
$SecurityGroup = Get-AzureRmNetworkSecurityGroup -Name $nsg -ResourceGroupName $resourceGroup -ErrorAction SilentlyContinue
$networkcard = Get-AzureRmNetworkInterface -Name $nic -ResourceGroupName $resourceGroup -ErrorAction SilentlyContinue
$Subnetz = Get-AzureRmVirtualNetworkSubnetConfig -Name $subnetname -VirtualNetwork $Network -ErrorAction SilentlyContinue

#Überprüfen ob Name der VM in Resource-Gruppe schon vorhanden ist
$VM = Get-AzureRmVM -Name $vmName -ResourceGroupName $resourceGroup

while ($VM -ne $null)
{
    echo "schon vorhanden, bitte neu wählen"
    $vmName = Read-Host -Prompt "Neuer Name"
    $nic = "$vmName"+"_nic"
    $VM = Get-AzureRmVM -Name $vmName -ResourceGroupName $resourceGroup

}


#User und Passwort vordefinieren
$securePassword = ConvertTo-SecureString [redacted] -AsPlainText -Force
$cred = New-Object System.Management.Automation.PSCredential ("azureuser", $securePassword)

#-------------------------------------------------------------------------------------------------------------------------------
#Erstellung


#Erstelle neue Ressoucen-Gruppe falls noch nicht vorhanden
if($Group -eq $null)
{
    New-AzureRmResourceGroup -Name $resourceGroup -Location $location
}

if($Subnetz -eq $null)

{
# Create a subnet configuration
$subnetConfig = New-AzureRmVirtualNetworkSubnetConfig -Name $subnetname -AddressPrefix 192.168.[redacted]/24
}

#Erstelle neues Netzwerk falls noch nicht vorhanden
if($Network -eq $null)
{
    $Network = New-AzureRmVirtualNetwork -ResourceGroupName $resourceGroup -Location $location `
            -Name $vnet -AddressPrefix 192.168.[redacted]/16 -Subnet $subnetConfig
}

# Create a public IP address and specify a DNS name
$pip = New-AzureRmPublicIpAddress -ResourceGroupName  $resourceGroup -Location $location `
    -AllocationMethod Static -IdleTimeoutInMinutes 4 -Name "mypublicdns$(Get-Random)"

if($SecurityGroup -eq $null)
{
        #Erstelle eine Gruppenregel für port 3389 (inbound)
        $nsgRuleRDP = New-AzureRmNetworkSecurityRuleConfig -Name NetworkSecurityGroupRuleRDP  -Protocol Tcp `
                -Direction Inbound -Priority 1000 -SourceAddressPrefix * -SourcePortRange * -DestinationAddressPrefix * `
                -DestinationPortRange 3389 -Access Allow

        #Erstelle eine Gruppenregel für port 80 (inbound)
        $nsgRuleWeb = New-AzureRmNetworkSecurityRuleConfig -Name NetworkSecurityGroupRuleWWW  -Protocol Tcp `
                -Direction Inbound -Priority 1001 -SourceAddressPrefix * -SourcePortRange * -DestinationAddressPrefix * `
                -DestinationPortRange 80 -Access Allow

        #Erstelle Security-Gruppe
        $SecurityGroup = New-AzureRmNetworkSecurityGroup -ResourceGroupName $resourceGroup -Location $location `
                    -Name $nsg -SecurityRules $nsgRuleRDP,$nsgRuleWeb
}

# Erstelle virtuelle Netzwerkkarte und verbinde diese  mit public IP address und der Security-Group
$networkcard = New-AzureRmNetworkInterface -Name $nic -ResourceGroupName $resourceGroup -Location $location `
            -SubnetId $Network.Subnets[0].Id -PublicIpAddressId $pip.Id -NetworkSecurityGroupId $SecurityGroup.Id

$vmConfig = New-AzureRmVMConfig -VMName $vmName -VMSize $vmsize | `
    Set-AzureRmVMOperatingSystem -Windows -ComputerName $vmName -Credential $cred | `
    Set-AzureRmVMSourceImage -PublisherName MicrosoftWindowsServer -Offer WindowsServer `
    -Skus 2016-Datacenter -Version latest | Add-AzureRmVMNetworkInterface -Id $networkcard.Id

New-AzureRmVM -ResourceGroupName $resourceGroup -Location $location -VM $vmConfig
```

Anlage 9 Teil 1 Finales Skript individuelle VM („Ablauf_Indiv_VM.ps1“)

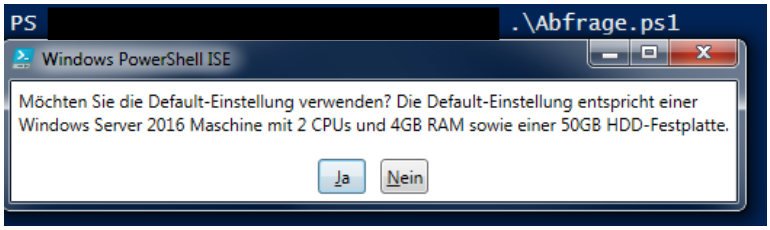
```
#-------------------------------------------------------------------------------------------------------------------------------
#Parameter


#Parametereingabe
$subscription = Read-Host -Prompt "ID der zu verwendenden Subscription"
$vmName = Read-Host -Prompt "Name der VM"
$resourceGroup = Read-Host -Prompt "ResourceGroup"
$vnet = Read-Host -Prompt "Name des Netzwerkes"
$subnet = Read-Host -Prompt "Name des Subnetzes"
$vmsize = Read-Host -Prompt "Grösse der VM"
$nsg = Read-Host -Prompt "Name der NSG"

$location = "westeurope"
$nic = "$vmName"+"_nic"

#-------------------------------------------------------------------------------------------------------------------------------
#Anmeldung

#Einloggen bei Azure
Login-AzureRmAccount

Select-AzureRmSubscription -SubscriptionName $subscription
#-------------------------------------------------------------------------------------------------------------------------------
#Überprüfungen
Get-AzureRmResourceGroup

#Abfragen der eingegebenen Parameter ob diese schon existieren + in Parameter speichern
$Group = Get-AzureRmResourceGroup -Name $resourceGroup -ErrorAction SilentlyContinue
$Network = Get-AzureRmVirtualNetwork -Name $vnet -ResourceGroupName $resourceGroup -ErrorAction SilentlyContinue
$SecurityGroup = Get-AzureRmNetworkSecurityGroup -Name $nsg -ResourceGroupName $resourceGroup -ErrorAction SilentlyContinue
$networkcard = Get-AzureRmNetworkInterface -Name $nic -ResourceGroupName $resourceGroup -ErrorAction SilentlyContinue
$Subnetz = Get-AzureRmVirtualNetworkSubnetConfig -Name $subnetname -VirtualNetwork $Network -ErrorAction SilentlyContinue

#Überprüfen ob VM-Grösse in der lokation auch möglich ist
while ((Get-AzureRmVMSize -Location $location | Select Name | Where Name -eq $vmsize) -eq $null)
{
    echo "Diese Grösse der VM ist in dieser Lokation leider nicht verfügbar. Bitte wählen Sie eine passende Grösse"
    $vmsize = Read-Host -Prompt "Size"
    $Test = Get-AzureRmVMSize -Location $location | Select Name | Where Name -eq $vmsize
}

#Überprüfen ob Name der VM in Resource-Gruppe schon vorhanden ist
while ((Get-AzureRmVM -Name $vmName -ResourceGroupName $resourceGroup) -ne $null)
{
    echo "schon vorhanden, bitte neu wählen"
    $vmName = Read-Host -Prompt "Neuer Name"
    $VM = Get-AzureRmVM -Name $vmName -ResourceGroupName $resourceGroup
}


#User und Passwort vordefinieren
$securePassword = ConvertTo-SecureString [redacted] -AsPlainText -Force
$cred = New-Object System.Management.Automation.PSCredential ("azureuser", $securePassword)

#-------------------------------------------------------------------------------------------------------------------------------
#Erstellung

#Erstelle neue Ressoucen-Gruppe falls noch nicht vorhanden
if($Group -eq $null)
{
    New-AzureRmResourceGroup -Name $resourceGroup -Location $location
}

if($Subnetz -eq $null)

{
# Create a subnet configuration
$subnetConfig = New-AzureRmVirtualNetworkSubnetConfig -Name $subnetname -AddressPrefix 192.168.[redacted]/24
}

#Erstelle neues Netzwerk falls noch nicht vorhanden
if($Network -eq $null)
{
    $Network = New-AzureRmVirtualNetwork -ResourceGroupName $resourceGroup -Location $location `
            -Name $vnet -AddressPrefix 192.168.[redacted]/16 -Subnet $subnetConfig
}

#Erstelle neue public IP-Addresse und bestimme DNS-Name
$pip = New-AzureRmPublicIpAddress -ResourceGroupName $resourceGroup -Location $location `
    -AllocationMethod Static -IdleTimeoutInMinutes 4 -Name "mypublicdns$(Get-Random)"


```

Anlage 10 Teil 2 Finales Skript individuelle VM

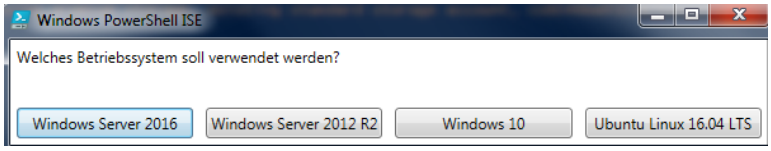
```
#Erstelle eine Netzwerk-Security-Gruppe falls noch nicht vorhanden
if($SecurityGroup -eq $null)
{
    if($skus -eq '16.04-LTS')
    {
        #Erstelle eine Gruppenregel für port 3389 (inbound)
        $nsgRuleRDP = New-AzureRmNetworkSecurityRuleConfig -Name myNetworkSecurityGroupRuleRDP  -Protocol Tcp `
                -Direction Inbound -Priority 1000 -SourceAddressPrefix * -SourcePortRange * -DestinationAddressPrefix * `
                -DestinationPortRange 3389 -Access Allow

        #Erstelle eine Gruppenregel für port 80 (inbound)
        $nsgRuleWeb = New-AzureRmNetworkSecurityRuleConfig -Name myNetworkSecurityGroupRuleWWW  -Protocol Tcp `
                -Direction Inbound -Priority 1001 -SourceAddressPrefix * -SourcePortRange * -DestinationAddressPrefix * `
                -DestinationPortRange 80 -Access Allow

        #Erstelle eine Gruppenregel für port 22 (inbound)
        $nsgRuleSSH = New-AzureRmNetworkSecurityRuleConfig -Name myNetworkSecurityGroupRuleSSH  -Protocol Tcp `
                    -Direction Inbound -Priority 999 -SourceAddressPrefix * -SourcePortRange * -DestinationAddressPrefix * `
                    -DestinationPortRange 22 -Access Allow

        #Erstelle Security-Gruppe
        $SecurityGroup = New-AzureRmNetworkSecurityGroup -ResourceGroupName $resourceGroup -Location $location `
                -Name myNetworkSecurityGroup -SecurityRules $nsgRuleRDP,$nsgRuleWeb,$nsgRuleSSH
    }
    else
    {
        #Erstelle eine Gruppenregel für port 3389 (inbound)
        $nsgRuleRDP = New-AzureRmNetworkSecurityRuleConfig -Name myNetworkSecurityGroupRuleRDP  -Protocol Tcp `
                -Direction Inbound -Priority 1000 -SourceAddressPrefix * -SourcePortRange * -DestinationAddressPrefix * `
                -DestinationPortRange 3389 -Access Allow

        #Erstelle eine Gruppenregel für port 80 (inbound)
        $nsgRuleWeb = New-AzureRmNetworkSecurityRuleConfig -Name myNetworkSecurityGroupRuleWWW  -Protocol Tcp `
                -Direction Inbound -Priority 1001 -SourceAddressPrefix * -SourcePortRange * -DestinationAddressPrefix * `
                -DestinationPortRange 80 -Access Allow

        #Erstelle Security-Gruppe
        $SecurityGroup = New-AzureRmNetworkSecurityGroup -ResourceGroupName $resourceGroup -Location $location `
                -Name myNetworkSecurityGroup -SecurityRules $nsgRuleRDP,$nsgRuleWeb
    }
}

# Erstelle virtuelle Netzwerkkarte und verbinde diese  mit public IP address und der Security-Group
$networkcard = New-AzureRmNetworkInterface -Name $nic -ResourceGroupName $resourceGroup -Location $location `
            -SubnetId $Network.Subnets[0].Id -PublicIpAddressId $pip.Id -NetworkSecurityGroupId $SecurityGroup.Id


#Erstelle Konfiguration der einzelnen OS
if($skus -eq '2016-Datacenter')
{
    $vmConfig = New-AzureRmVMConfig -VMName $vmName -VMSize $vmsize | `
    Set-AzureRmVMOperatingSystem -Windows -ComputerName $vmName -Credential $cred | `
    Set-AzureRmVMSourceImage -PublisherName MicrosoftWindowsServer -Offer WindowsServer `
    -Skus 2016-Datacenter -Version latest | Add-AzureRmVMNetworkInterface -Id $networkcard.Id
}
elseif($skus -eq '2012_R2')
{
    $vmConfig = New-AzureRmVMConfig -VMName $vmName -VMSize $vmsize | `
    Set-AzureRmVMOperatingSystem -Windows -ComputerName $vmName -Credential $cred | `
    Set-AzureRmVMSourceImage -PublisherName MicrosoftWindowsServer -Offer WindowsServer `
    -Skus 2012-R2-Datacenter -Version latest | Add-AzureRmVMNetworkInterface -Id $networkcard.Id
}
elseif($skus -eq 'Windows-10-N-x64')
{
    $vmConfig = New-AzureRmVMConfig -VMName $vmName -VMSize $vmsize | `
    Set-AzureRmVMOperatingSystem -Windows -ComputerName $vmName -Credential $cred | `
    Set-AzureRmVMSourceImage -PublisherName MicrosoftVisualStudio -Offer Windows-10 `
    -Skus RS2-Pro -Version latest | Add-AzureRmVMNetworkInterface -Id $networkcard.Id
}
elseif($skus -eq '16.04-LTS')
{
    $vmConfig = New-AzureRmVMConfig -VMName $vmName -VMSize $vmsize | `
    Set-AzureRmVMOperatingSystem -Linux -ComputerName $vmName -Credential $cred -DisablePasswordAuthentication | `
    Set-AzureRmVMSourceImage -PublisherName Canonical -Offer UbuntuServer -Skus 16.04-LTS -Version latest | `
    Add-AzureRmVMNetworkInterface -Id $networkcard.Id

    #Suche SSH Keys
    $sshPublicKey = Get-Content "$env:USERPROFILE\.ssh\id_rsa.pub"
    Add-AzureRmVMSshPublicKey -VM $vmconfig -KeyData $sshPublicKey -Path "/home/azureuser/.ssh/authorized_keys"
}

#Erstelle VM
New-AzureRmVM -ResourceGroupName $resourceGroup -Location $location -VM $vmConfig
```

Anlage 11 Screenshots zu ausführlichen Tests

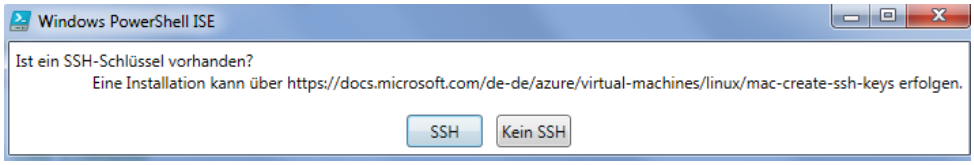
Abfrage nach Default-VM



Auswahl des Betriebssystems bei individueller VM



Abfrage nach SSH-Schlüssel



Anlage 12 Ausführlicher Soll-Ist-Vergleich

Projektphase mit Teilaufgaben	Geplant in h		Umgesetzt in h		Differenz
	Projektphase gesamt	Dauer	Projektphase	Dauer	
1. Planungsphase	7,5		6		**-1,5**
Ist-Analyse		1,5		1,5	
Soll-Konzept		3		1,5	-1,5
Nutzwertanalyse		3		3	
2.Durchführungsphase	18,5		19		**+0,5**
Voraussetzungen überprüfen		0,5		0,5	
Recherchen zur Programmierung		3		3	
Erstmalige Erstellung		6		6	
Ergänzungen der Skripte		4		4,5	0,5
Fehlersuche/Fehlerbehebung		3		3	
Testen auf Funktionalität		2		2	
3.Abschlussphase	9		10		**+1**
Hauptdokumentation		6		6	
Betriebsdokumentation		2		2,5	0,5
Kundendokumentation		2		2,5	0,5
Gesamtdauer	**35**		**35**		**+/- 0**

Kundendokumentation

Automatisierte Bereitstellung von Entwicklungs- und Testsystemen mithilfe von Cloud-Technologien

Bearbeitet von Jessica Schmieder

1. Voraussetzungen

Zur Ausführung der Skripte wird mindestens PowerShell 5.0 benötigt. Ebenfalls benötigen Sie die Azure-Erweiterung für PowerShell. Beide Installationen können Sie über die offizielle Microsoft-Seite herunterladen und installieren. Es sind keine weiteren Konfigurationen zu beachten.
Bei Interesse eine Linux-Maschine zu erstellen ist außerdem ein SSH-Schlüssel unter dem User-Profil im Ordner „.ssh“ mit der Bezeichnung „id_rsa.pub“ nötig.

2. Admin-User

Beim Anlegen jeder VM wird ein Admin-User mit folgenden Anmeldedaten erstellt.

Benutzername: Azureuser
Passwort: xxxxxxxxxxxxxxx

Hinweis: Das Passwort sollte nach erstmaligem Gebrauch geändert werden.

3. Bezeichnung VM-Größen

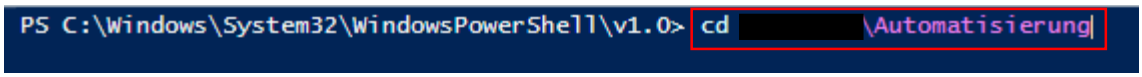
Azure verwendet bestimmte Bezeichnungen zur Benennung der einzelnen VM-Größen. Nur diese Bezeichnungen dürfen bei der Angabe der VM-Größe während der Ausführung der Skripte verwendet werden. Auf der offiziellen Microsoft-Seite (https://docs.microsoft.com/de-de/azure/virtual-machines/windows/sizes) können Sie sich die passende Größenbezeichnung heraussuchen.

4. Ablageort des Skriptes

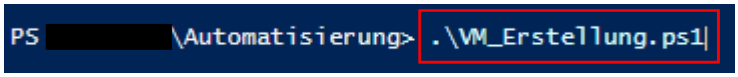
Sie finden das benötigte Skript unter [redacted]\Automatisierung.

5. Das Skript ausführen

Öffnen Sie zunächst die PowerShell-Konsole oder ISE. Um das Skript ausführen zu können müssen Sie sich in dem entsprechenden Ordner befinden. Wechseln Sie daher an dieser Stelle in das entsprechende Verzeichnis. Verwenden Sie dazu folgenden Befehl:



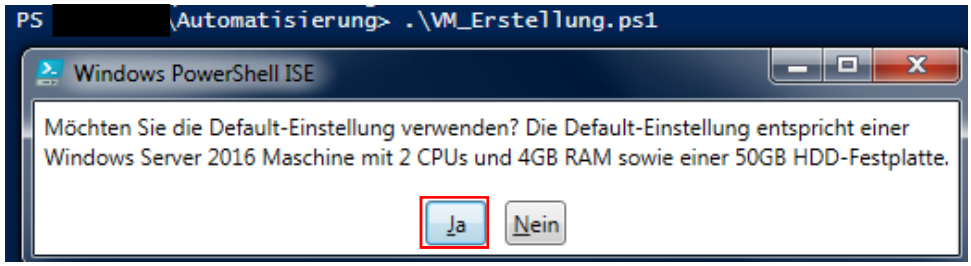
Um anschließend das Skript aufzurufen verwenden Sie folgenden Befehl:



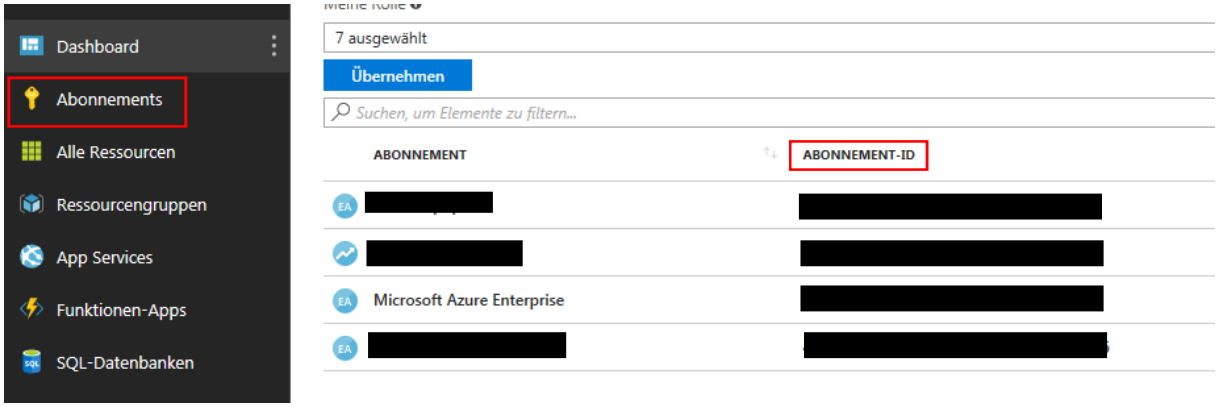
6. Wie erstelle ich eine virtuelle Maschine?

a. Default-VM

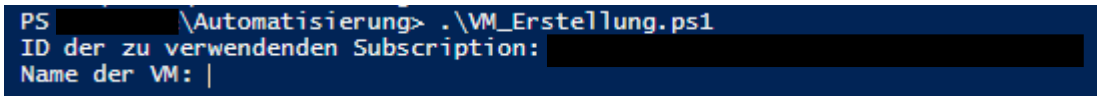
Führen Sie das Skript entsprechend Punkt 4 aus. Es erscheint nun eine Abfrage nach der Art der zu erstellenden VM. Wählen Sie hier die Default-VM mit „Ja“ aus. Wie im Fenster beschrieben handelt es sich dabei um eine Windows Server 2016 Maschine mit 2 CPUs sowie 4 GB RAM und 50GB HDD-Festplatte.



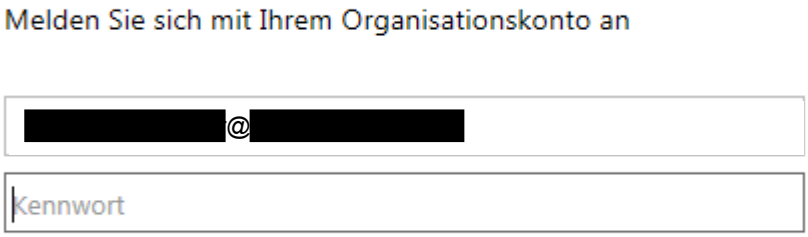
Anschließend werden Sie nach der ID der zu verwendenden Subscription gefragt. Sie können die Subscription über das Azure-Portal unter „Abonnements“ heraussuchen.



Geben Sie nun die entsprechende Abonnement-ID ein und bestätigen Sie diese mit Enter.
Anschließend werden Sie aufgefordert einen Namen für die VM einzugeben. Tun Sie dies und bestätigen Sie wieder mit Enter.



Es öffnet sich nun ein Fenster. Geben Sie dort Ihre Microsoft-Anmeldedaten an.



Anschließend wird das Skript ausgeführt. Dies kann bis zu 10 Minuten dauern. Bei Beenden des Skriptes sollte Folgendes ausgegeben werden:

```
RequestId           :
IsSuccessStatusCode : True
StatusCode          : OK
ReasonPhrase        : OK
```

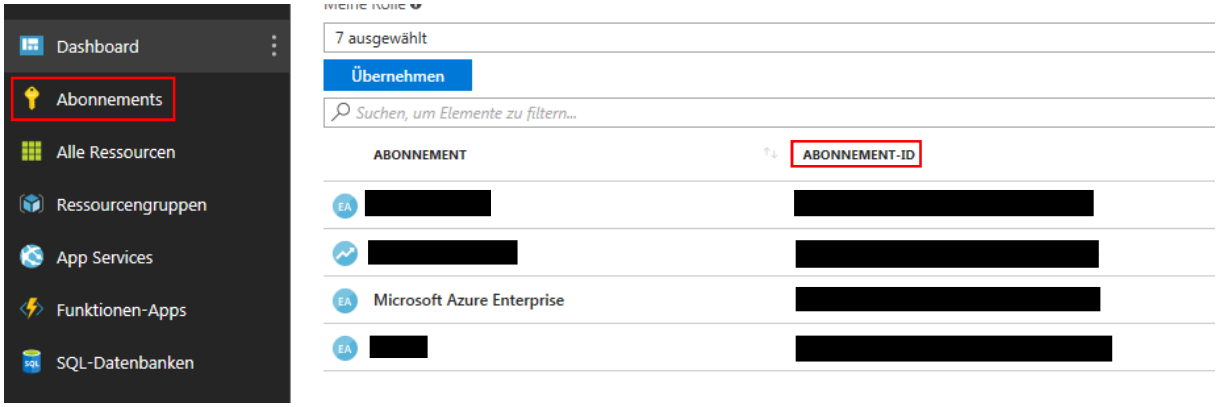
Nach Abschluss steht Ihnen die VM im Azure-Portal unter „Virtuelle Computer“ zur Verfügung.

b. Individuelle VM

Führen Sie das Skript entsprechend Punkt 4 aus. Es erscheint nun eine Abfrage nach der Art der zu erstellenden VM. Wählen Sie hier „Nein“ aus, um eine individuelle VM zu erstellen (Vgl. 5. a). Dadurch haben Sie nun die Freiheit zwischen vier verschiedenen Betriebssystemen zu wählen.

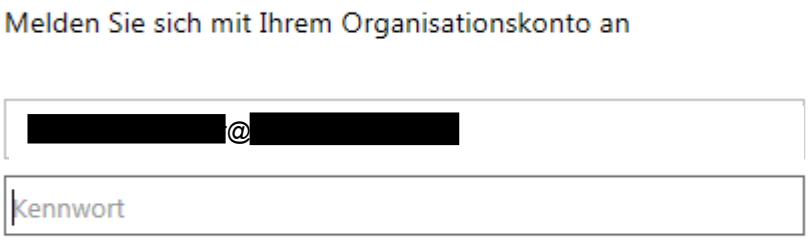
Bitte beachten Sie, dass bei Auswahl des Ubuntu-Systems ein SSH-Schlüssel vorhanden sein muss!

Geben Sie anschließend die geforderten Bezeichnungen entsprechend Ihren Wünschen ein. Die Subscription können Sie im Azure-Portal unter Abonnements heraussuchen.




```
PS [redacted]\Automatisierung> .\VM_Erstellung.ps1
ID der zu verwendenden Subscription: [redacted]
Name der VM: TestVM
ResourceGroup: TestResourceGroup
Name des Netzwerkes: TestNetzwerk
Name des Subnetzes: TestSubnetz
Grösse der VM: Standard_v2
Name der NSG: TestNSG
```

Es öffnet sich nun ein Fenster. Geben Sie dort Ihre Microsoft-Anmeldedaten an.



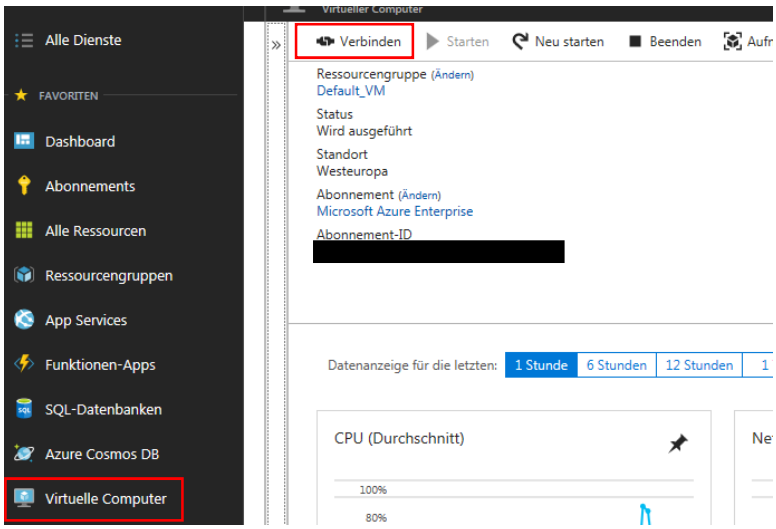
Anschließend wird das Skript ausgeführt. Dies kann bis zu 10 Minuten dauern. Bei Beenden des Skriptes sollte Folgendes ausgegeben werden:

```
RequestId           :
IsSuccessStatusCode : True
StatusCode          : OK
ReasonPhrase        : OK
```

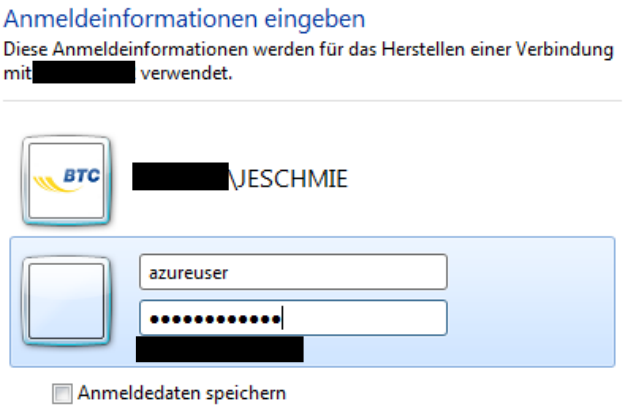
Nach Abschluss steht Ihnen die VM im Azure-Portal unter „Virtuelle Computer“ zur Verfügung.

7. Mit einer VM verbinden

Um sich auf die Maschine zu schalten gehen Sie über „Verbinden“.



Für eine Verbindung wählen Sie in der sich öffnenden Remotedesktopverbindung „Anderes Konto verwenden“ und geben dort die Daten des Admin-Users entsprechend Kapitel 2 ein.



Anschließend können Sie mit Ihrer erstellten VM arbeiten.

8. Ansprechpartner

Name	Kontaktdaten
Jessica Schmieder	E-Mail: [redacted] Telefon:xxxx

Betriebsdokumentation

Automatisierte Bereitstellung von Entwicklungs- und Testsystemen mithilfe von Cloud-Technologien

Bearbeitet von Jessica Schmieder

1. Voraussetzungen

- Mindestens PowerShell 5.0
- Azure-Modul für PowerShell
- Ggf. SSH-Schlüssel

2. Admin-User

Beim Anlegen jeder VM wird ein Admin-User erstellt.

Benutzername: Azureuser
Passwort: xxxxxxxxxxxxxxx

3. Bezeichnung VM-Größen

Siehe https://docs.microsoft.com/de-de/azure/virtual-machines/windows/sizes

4. Ablageort des Skriptes

[redacted]\Automatisierung

5. Erläuterung der Skripte

a. VM_Erstellung.ps1

Beinhaltet:

- Pfade der Skripte Ablauf_Default_VM.ps1 und Ablauf_Indiv_VM.ps1 sind angegeben
- Abfrage nach Default-VM
- Bei Zustimmung wird Ablauf_Default_VM.ps1 aufgerufen
- Bei Ablehnung wird nach Betriebssystem gefragt, Ergebnis wird in $skus gespeichert, danach wird Ablauf_Indiv_VM.ps1 aufgerufen
- Wenn Linux gewählt wird, erscheint eine Abfrage ob SSH-Schlüssel vorhanden ist, bei Verneinung Abbruch des Skriptes

b. Ablauf_Default_VM.ps1

Beinhaltet:

- Abfrage offener Variablen (Subscription, Name der VM)
- Definition der Variablen (Ressourcen-Gruppe, Lokation, virtuelles Netzwerk, virtuelle Netzwerkkarte, Größe der VM, Subnetz, Netzwerksicherheitsgruppe)
- Anmelden im Azure-Portal
- Wechseln in die gewählte Subscription
- Abrufen der eingegebenen Parameter; Überprüfen, ob Komponenten bereits vorhanden sind
- Kontrolle, ob Name der VM bereits vorhanden, dann muss neuer Name angegeben werden
- Admin-User wird angelegt
- Alle Komponenten, die noch nicht existieren, werden erstellt
- Alle Parameter werden in der Variablen $vmconfig gespeichert
- Die VM wird erstellt mit den in $vmconfig gespeicherten Parametern

c. Ablauf_Indiv_VM.ps1

Beinhaltet:

- Abfrage offener Variablen (Subscription, Name der VM, Ressourcen-Gruppe, Lokation, virtuelles Netzwerk, virtuelle Netzwerkkarte, Größe der VM, Subnetz, Netzwerksicherheitsgruppe)
- Anmelden im Azure-Portal
- Wechseln in die gewählte Subscription
- Abrufen der eingegebenen Parameter; Überprüfen, ob Komponenten bereits vorhanden sind
- Überprüfen, ob angegebene Größe in gewählter Lokation vorhanden ist, dann muss neue Größe angegeben werden
- Kontrolle ob Name der VM bereits vorhanden, dann muss neuer Name angegeben werden
- Admin-User wird angelegt
- Alle Komponenten, die noch nicht existieren, werden erstellt
- Je nach gewähltem Betriebssystem in „VM_Erstellung.ps1“ werden entsprechende Parameter in $vmconfig gespeichert
- Die VM wird erstellt mit den in $vmconfig gespeicherten Parametern

6. Ändern von Parametern

a. Ändern der Default-VM-Parameter

Für die Default-VM sind die Parameter bereits vordefiniert. Besteht Bedarf diese zu ändern, kann dies einfach im Skript „Ablauf_Default_VM.ps1“ in den ersten Zeilen angepasst werden.

```
#Parameter

$subscription = Read-Host -Prompt "ID der zu verwendenden Subscription"
$vmname = Read-Host -Prompt "Name der VM"
$resourceGroup = 'Default_VM'
$location = 'WestEurope'
$subnetname = "Default_Subnet"
$nic = "$vmName"+"_nic"
$vnet = "Default_vnet"
$nsg = "Default_nsg"
$vmsize = 'Standard_A2'
```

b. Ändern der angebotenen Betriebssysteme

Die zur Auswahl stehenden Betriebssysteme können im Skript „VM_Erstellung.ps1“ angepasst werden. Dazu muss eine neue Zeile eingefügt werden und in $BS eingetragen werden.

```
$titel = 'Wahl des Betriebssystems'
$message = 'Welches Betriebssystem soll verwendet werden?'
$WS2k16 = New-Object System.Management.Automation.Host.ChoiceDescription "&Windows Server 2016", "Windows Server 2016"
$WS2k12R2 = New-Object System.Management.Automation.Host.ChoiceDescription "Windows Server 2012 R&2", "Windows Server 2012 R2"
$Win10 = New-Object System.Management.Automation.Host.ChoiceDescription "Windows &10", "Windows 10"
$UL1604LS = New-Object System.Management.Automation.Host.ChoiceDescription "&Ubuntu Linux 16.04 LTS", "Ubuntu Linux 16.04 LTS"
$options = [System.Management.Automation.Host.ChoiceDescription[]]($WS2k16, $WS2k12R2,$Win10,$UL1604LS)
$BS=$host.ui.PromptForChoice($title, $message, $options, 0)
```

Anschließend muss im darauf folgenden switch-Befehl die $skus-Variable zusätzlich definiert werden. Die benötigten Informationen sind unter https://docs.microsoft.com/de-de/azure/virtual-machines/windows/cli-ps-findimage zu finden. Diese Informationen werden weitergehend im Skript „Ablauf_indiv_VM.ps1“ benötigt, da dort die Übergabe der Parameter erfolgt. Dafür muss die ifelse-Anweisung mit den entsprechenden Parametern wie folgt aussehen:

```
elseif($skus -eq '2012_R2')
{
    $vmConfig = New-AzureRmVMConfig -VMName $vmName -VMSize $vmsize | `
    Set-AzureRmVMOperatingSystem -Windows -ComputerName $vmName -Credential $cred | `
    Set-AzureRmVMSourceImage -PublisherName MicrosoftWindowsServer -Offer WindowsServer `
    -Skus 2012-R2-Datacenter -Version latest | Add-AzureRmVMNetworkInterface -Id $networkcard.Id
}
```

7. Ansprechpartner

Name	Kontaktdaten
Jessica Schmieder	E-Mail: [redacted] Telefon: xxxx