Dokumentation M120

By Nando Sterki + Jori Villiger

Inhaltsverzeichnis

[IaaS, PaaS, … 2](#_Toc129675338)

[Create service principal: 3](#_Toc129675339)

[Connect to server (Task 1) 3](#_Toc129675340)

[Create App (Task 2) 3](#_Toc129675341)

[Get set up on server (Task 3) 4](#_Toc129675342)

[Set everything up (Task 4) 4](#_Toc129675343)

[Deploy App (jar File) (Task 5) 4](#_Toc129675344)

[Delete App (when youre done) 4](#_Toc129675345)

[View logs: 4](#_Toc129675346)

[Logs in einer Pipeline: 4](#_Toc129675347)

[.gitlab-ci.yml 4](#_Toc129675348)

[Tipps & Tricks: 5](#_Toc129675349)

[Vorlage für Namenskonzept beim Erstellen von bsw. Apps 8](#_Toc129675350)

[RBAC 9](#_Toc129675351)

[Basuc features 9](#_Toc129675352)

[Lis role assignments 10](#_Toc129675353)

[List role assignments for a resource group 10](#_Toc129675354)

[Grant access 10](#_Toc129675355)

[Remove access 10](#_Toc129675356)

[View activity logs 11](#_Toc129675357)

[Netzwerk und Firewall 11](#_Toc129675358)

# IaaS, PaaS, …

SaaS (Software as a Service) bezieht sich auf die Bereitstellung von Softwareanwendungen über das Internet, die von einem Drittanbieter gehostet und gewartet werden. Der Benutzer greift über einen Webbrowser oder eine App auf die Anwendung zu und zahlt in der Regel eine monatliche oder jährliche Gebühr.

FaaS (Function as a Service) bezieht sich auf die Bereitstellung von Funktionen oder Code-Segmenten als Service über das Internet. Die Funktionen werden auf Anfrage ausgeführt und der Benutzer zahlt nur für die tatsächlich genutzten Ressourcen.

IaaS (Infrastructure as a Service) bezieht sich auf die Bereitstellung von IT-Infrastrukturkomponenten wie Server, Speicher, Netzwerke und virtuelle Maschinen über das Internet. Der Benutzer kann die Infrastrukturkomponenten nach Bedarf konfigurieren und verwalten, ohne sich um den physischen Betrieb und die Wartung kümmern zu müssen.

PaaS (Platform as a Service) bezieht sich auf die Bereitstellung einer Entwicklungs- und Bereitstellungsplattform über das Internet. Die Plattform bietet eine Umgebung für die Entwicklung, Tests und Bereitstellung von Anwendungen. Der Benutzer kann seine Anwendung auf der Plattform ausführen, ohne sich um die zugrunde liegende Infrastruktur kümmern zu müssen.

# Create service principal:

az ad sp create-for-rbac --scopes /subscriptions/<RESOURCE\_GROUP\_SUBSCRIPTION\_ID> --role "Owner"

Creating 'Owner' role assignment under scope '/subscriptions/965aa8a5-9a39-4f68-8908-06e3c4d82d85'

The output includes credentials that you must protect. Be sure that you do not include these credentials in your code or check the credentials into your source control. For more information, see https://aka.ms/azadsp-cli

#Username, password, someStringThatIsNeeded

#Username

eac396b0-c31e-4c92-af4c-5e2c19d132ba

#Tenant

02b7e332-b310-470b-8eef-729bfcf99667

#Password

Cr.8Q~NCijtNw5EpSwa\_SPy7IQ0~uNxESXOXjdl6

### Connect to server (Task 1)

az mysql flexible-server connect   
-n <server-name>  
-u <db-admin-user-name>  
--interactive

Create App (simpler)

az spring create \

--resource-group <resource-group-name> \

--name <app-service-name>

### Create App (Task 2)

az spring app create \

--resource-group <resource-group-name> \

--service <app-service-name> \

--name <app-name> \

--runtime-version Java\_17 \

--assign-endpoint true

Es wird eine Url ausgegeben, auf welcher später die Applikation verfügbar sein wird. Diese sollte folgender URL ähneln:

[https://studle2-studletheapp.azuremicroservices.io](https://studle2-studletheapp.azuremicroservices.io¨)

sobald die App läuft, kann unter der angepassten Url die App aufgerufen werden:

<generierte-url>/swagger-ui/index.html

### Get set up on server (Task 3)

az spring connection create mysql-flexible \

-g <resource-group-name> \

--tg <resource-group-name> \

--connection <connection\_name> \

--service <app-service-name> \

--app <app-name> \

--server <server-name> \

--database <database-name> \

--client-type springBoot \

--secret name=<db-admin-user-name> secret=<db-admin-password>

### Set everything up (Task 4)

az mysql flexible-server parameter set \

--resource-group <resource-group-name> \

--server-name <mysql-server-name> \

--name require\_secure\_transport \

--value OFF

### Deploy App (jar File) (Task 5)

az spring app deploy \

--resource-group <resource-group-name> \

--service <app-service-name> \

--name <app-name> \

--artifact-path <path-to-jar-file>.jar

### Delete App (when youre done)

az spring app delete --name <app-name> \  
 -g <resource-group-name> \  
 -s <app-service-name>

az spring delete --name <app-service-name> -g <resource-group-name>

### View logs:

az spring app logs -g <resource-group-name> -s <app-service-name> -n <app-name>

# Logs in einer Pipeline:

CI/CD > Jobs > Status:passed

# .gitlab-ci.yml

/\*

default:

image: alpine:3

\*/

variables:

RG\_NAME:

value: "uek-210"

LOCATION:

value: "switzerlandnorth"

CONTAINER\_REGISTRY:

value: "DEINE\_CONTAINER\_REGISTRY"

IMAGE\_NAME:

value: studle

job1:

stage: build

script:

- echo "Executed on all branches"

job2:

stage: build

script:

- echo "Executed only on the production branch"

rules:

- if: $CI\_MERGE\_REQUEST\_SOURCE\_BRANCH\_NAME == /main/

when: on\_success

- if: $CI\_MERGE\_REQUEST\_SOURCE\_BRANCH\_NAME == /test1/

when: never

job3:

stage: build

script:

- echo "Executed on the branches main and staging"

rules:

- if: $CI\_MERGE\_REQUEST\_SOURCE\_BRANCH\_NAME == /main/

when: on\_success

- if: $CI\_MERGE\_REQUEST\_SOURCE\_BRANCH\_NAME == /staging/

when: on\_success

- if: $CI\_MERGE\_REQUEST\_SOURCE\_BRANCH\_NAME == /test1/

when: never

deploy-app-name.bicep  
param name **string** = 'azure-studle'

param location **string** = **resourceGroup**().location

param image **string**

param registry **string**

param username **string**

@**secure**()

param password **string**

resource containerGroups\_azure\_studle\_name\_resource 'Microsoft.ContainerInstance/containerGroups@2022-10-01-preview' = {

**name**: name

**location**: location

**tags**: {

'docker-compose-application': 'docker-compose-application'

}

**properties**: {

**sku**: 'Standard'

**containers**: [

{

**name**: 'app'

**properties**: {

**image**: '${registry}/${image}:latest'

**ports**: [

{

**protocol**: 'TCP'

**port**: 80

}

]

**environmentVariables**: [

{

**name**: 'SERVER\_PORT'

**value**: '80'

}

{

**name**: 'SPRING\_DATASOURCE\_URL'

**value**: 'jdbc:mariadb://db:3306/studle'

}

]

**resources**: {

**requests**: {

**memoryInGB**: 1

**cpu**: 1

}

**limits**: {

**memoryInGB**: 1

**cpu**: 1

}

}

}

}

{

**name**: 'db'

**properties**: {

**image**: 'mariadb:10.9'

**ports**: []

**environmentVariables**: [

{

**name**: 'MARIADB\_ROOT\_PASSWORD'

**value**: 'password'

}

{

**name**: 'MARIADB\_DATABASE'

**value**: 'studle'

}

]

**resources**: {

**requests**: {

**memoryInGB**: 1

**cpu**: 1

}

**limits**: {

**memoryInGB**: 1

**cpu**: 1

}

}

}

}

{

**name**: 'aci--dns--sidecar'

**properties**: {

**image**: 'docker/aci-hostnames-sidecar:1.0'

**command**: [

'/hosts'

'app'

'db'

]

**ports**: []

**environmentVariables**: []

**resources**: {

**requests**: {

**memoryInGB**: '0.1'

**cpu**: '0.01'

}

}

}

}

]

**initContainers**: []

**imageRegistryCredentials**: [

{

**server**: registry

**username**: username

**password**: password

}

]

**restartPolicy**: 'Always'

**ipAddress**: {

**ports**: [

{

**protocol**: 'TCP'

**port**: 80

}

]

**type**: 'Public'

}

**osType**: 'Linux'

}

}

* az deployment group create --resource-group
* [--aux-subs]
* [--aux-tenants]
* [--confirm-with-what-if]
* [--handle-extended-json-format]
* [--mode {Complete, Incremental}]
* [--name]
* [--no-prompt {false, true}]
* [--no-wait]
* [--parameters]
* [--proceed-if-no-change]
* [--query-string]
* [--rollback-on-error]
* [--template-file]
* [--template-spec]
* [--template-uri]
* [--what-if]
* [--what-if-exclude-change-types {Create, Delete, Deploy, Ignore, Modify, NoChange, Unsupported}]
* [--what-if-result-format {FullResourcePayloads, ResourceIdOnly}]

Folgende Parameter musst du dabei zwingend übergeben.

* image Name deines Image Builds mit korrektem Tag
* registry URL zu deiner Registry, z.B. xyz.azurecr.io
* username Service Principal APP ID
* password Service Principal PASSWORT

## Tipps & Tricks:

* Keine Tabs erlaubt => 2 Leerschläge == 1 Tab
* $CI\_MERGE\_REQUEST\_SOURCE\_BRANCH\_NAME == aktueller Branchname
* When-Attribute :
  + on\_success

default, wenn alle Jobs erfolgreich durchlaufen

* + manual

nur wenn Job manuell getriggert wird

* + always

immer, Status vom Job egal (auch wenn er failed)

* + on\_failure

nur wenn min. ein Job failt

* + delayed

verzögert um eine angegebene Zeit

* + never

niemals

* equals (==)
* not equals (!=)
* and (&&)
* or (||)

Das funktioniert nur mit absolut leeren Reps:

cd existing\_folder

git init --initial-branch=main

git remote add origin PFAD\_ZU\_DEINEM\_GITLAB\_REPO

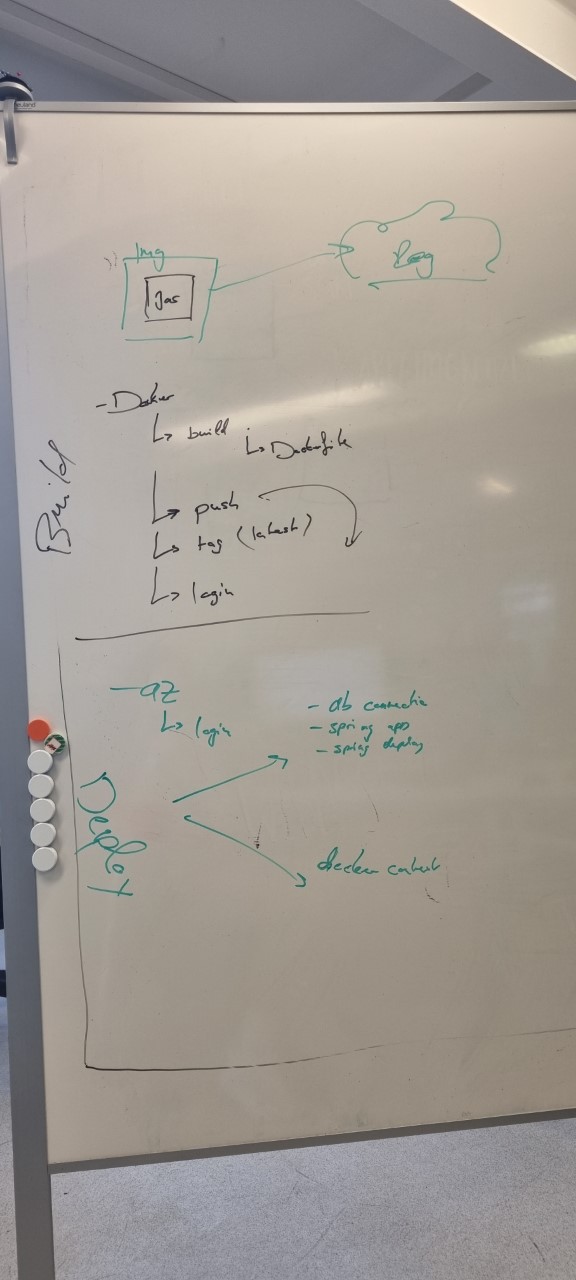
git add .

git commit -m "Initial commit"

git push -u origin main

für möglichst fehlerfreies Ergebnis:

die remote branch sollte im GUI noch nicht existieren (nur bei first commits)



Azure Functions Core Tool installieren:

wget -q [https://packages.microsoft.com/config/ubuntu/20.04/packages-microsoft-prod.deb sudo dpkg -i packages-microsoft-prod.deb](https://packages.microsoft.com/config/ubuntu/20.04/packages-microsoft-prod.deb%20sudo%20dpkg%20-i%20packages-microsoft-prod.deb)

# Vorlage für Namenskonzept beim Erstellen von bsw. Apps

<resource-group-name> Resource group-Name

<app-service-name> App-Service Name (Spring Applikation)

<app-name> App Name

<path-to-jar-file> Jar-File

<connection\_name> Connection-/Netzwerk-Name

<server-name> Server-Name

<database-name> Datenbank-Name

<db-admin-user-name> DB-Admin

<db-admin-password> DB-Admin-Password

<mysql-server-name> MySQL-Name (Erstellt in Task 1)

# Project architecture

**Grundsätzliche Unterteilung (1-Tier):**

*Client-App:*

* Darstellung
* Business-Logik
* Daten

**2-Tier:**

*Client-App:*

* Darstellung
* Business-Logik

*DB:*

* Daten

**2-Tier:**

*Client-App:*

* Darstellung

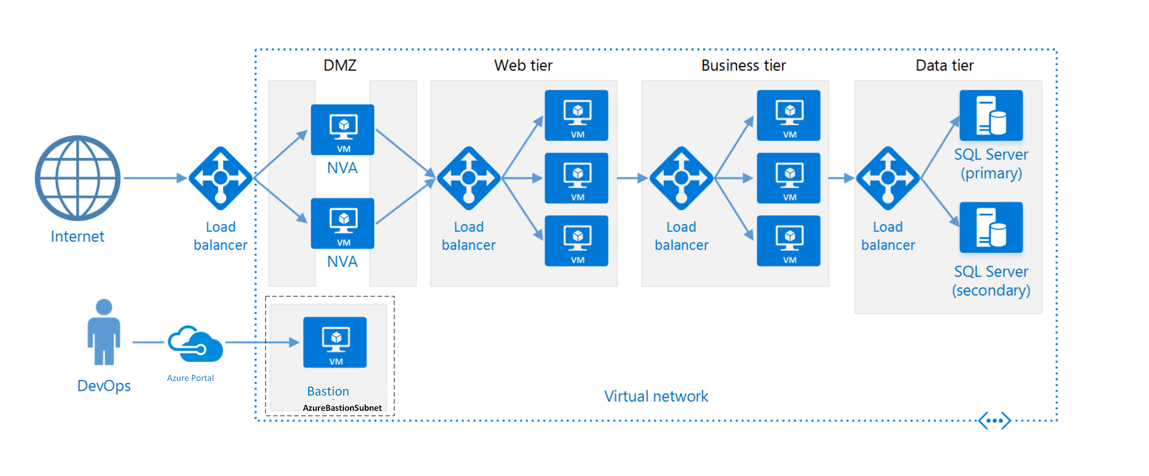
*Middleware:*

* Business-Logik

*DB:*

* Daten

**n-Tier:**

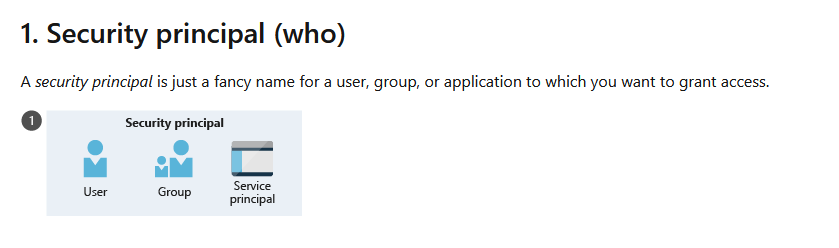


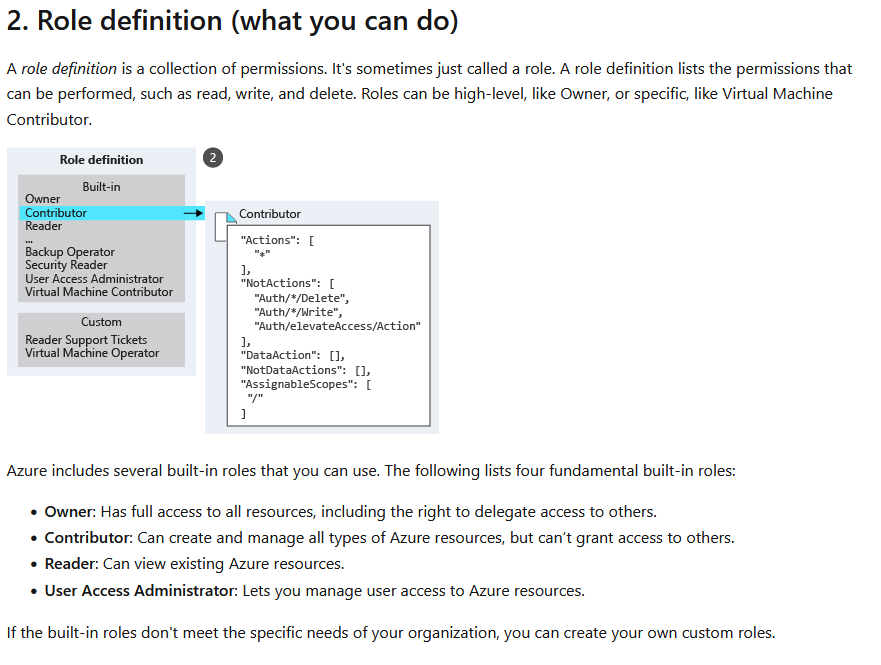
# RBAC

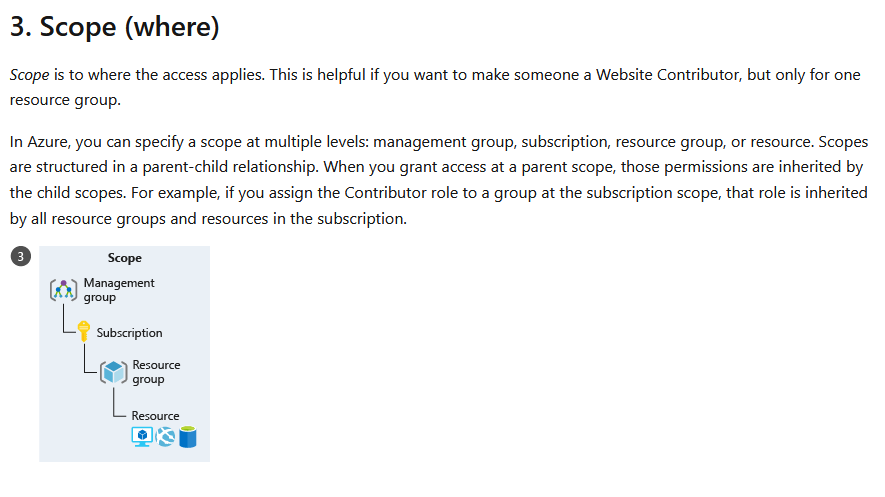
Azure role-based access control (Azure RBAC) is an authorization system that provides fine-grained access management of resources.

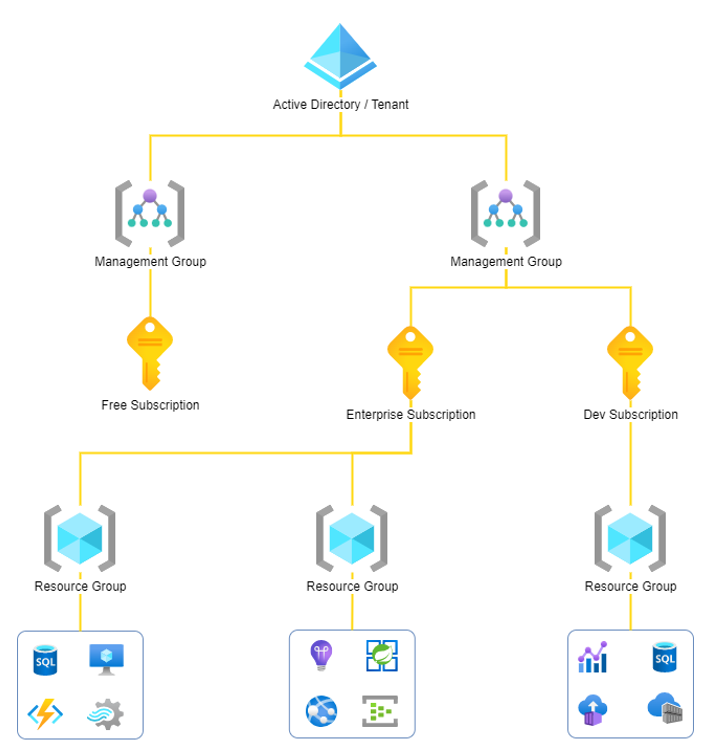
## Basuc features

* Allow one user to manage virtual machines in a subscription and another user to manage virtual networks
* Allow a database administrator group to manage SQL databases in a subscription
* Allow a user to manage all resources in a resource group, such as virtual machines, websites, and subnets
* Allow an application to access all resources in a resource group









## Lis role assignments

Profile -> (…) -> My permissions

## List role assignments for a resource group

Resource groups -> example-group -> Access control (IAM)-> role assignments

# Grant access

Resource group -> example-group -> Access control (IAM) -> Role assignments Add -> Add role assignment

# Remove access

Select user -> Remove -> Yes

# View activity logs

All services -> Activity log -> Timespan filter, operation filter

# Netzwerk und Firewall

|  |  |
| --- | --- |
| Begriff | Erklärung |
| Availability Zone | * Über min. 3 Zonen aufgebaut * Hohe Verfügbarkeit und Resistenz (bsw. Breakdown in einem datacenter) * Verschlüsselte Datenübermittlung |
| Region | * Traffic über Latenz-niedrige Applikationen * Supported VM’s, Container und eine Auswahl an Azure Services |
| Edge Zone | * Lässt Daten mit speziellem Schutzfaktor (bsw. Daten, die nach Schweizer Standart gespeichert werden müssen) angepasst speichern |
| Region Pair | * Paare in derselben Geographischen Region, um einen Serverausfall zu verkraften (bsw. North Central US und South Central US sind ein Region Pair) |

1. Welche fünf Adressen können in einem Subnetz nicht verwendet werden.   
   Begründe deine Antwort.

|  |
| --- |
| Reservierte IP-Adressen von Azure am Beispiel von der IP-Range 192.168.1.0/24:  192.168.1.0 Identifikation der Adresse des virtuellen Netzwerks  192.168.1.1 Adresse vom Default Gateway  192.168.1.2 + 192.168.1.2 werden von Azure zum Bereich des virtuellen Netzwerks zugeordnet  192.168.1.255 die Broadcastadresse des virtuellen Netzwerks |

1. Welche IP-Ranges können als Subnetz gebraucht werden? Wieso nicht alle?

|  |
| --- |
| Alle, die in die IP-Range des Bereiches des virtuellen Netzwerks fallen. Das Subnetz muss unique sein => innerhalb eines virtuellen Netzwerkes kann jede IP (-Range) nur einmal vergeben werden. Sie dürfen logischerweise somit auch nicht überlappen. |

# IaC (Infrastructure as Code)

az bicep install

az bicep version

## Automatisieren in Pipeline via Azure Bicep

(Hier: Projekt erstellen:)

targetScope = 'subscription'

param location **string** = **deployment**().location

param name **string**

resource rg 'Microsoft.Resources/resourceGroups@2018-05-01' = {

**location**: location

**name**: name

}

D ganze deployen:

az deployment sub create --location 'switzerlandnorth' --template-file rg.bicep   
--parameters name='uek-210'

## Behind the scenes:

