|  |
| --- |
| Infrastrcuture as Code |
| 6. øving i faget IaC |
| Forfattere:  Tor Ivar Melling  Eiere:  Forfatterne og Forskningsstiftelsen TISIP  Versjon: 1.0  Dato: 1. oktober 2020 |

# praktisk øving

I denne øvingen skal dere benytte Visual Studio Code til å jobbe videre med ARM Templates. Det ligger en videoer som dere skal benytte som ressurser for denne leksjonen.

*I denne øvingen skal dere vise hvordan en kan deploye flere ressurser til Auzre ved bruk av dependsOn blocks i Visual Studio Code. Test først med en enkel .json-fil med Storage Account og Blob Container som er avhengig av Storage Account. Forsøk deretter og utforsk mulighetene ved bruk av child resources og med egne parameterfiler for forskjellige miljø som vist i leksjon 5, eksempelvis produksjon og utvikling for Storage Account, Blob Container, Server Farm og Web Sites. Forsøk også å opprette avhengighet mellom Web Sites og Storage Acocunt.*

*Forsøk først med dependsOn ved bruk av navn og endre deretter til bruk av ResourceID.*

*Øvingsfiler ligger i egen mappe i its learning. Merk at det er både PowerShell og Bash-filer (Azure CLI).*

Det er ingen skriftlig leksjon denne gangen, men jeg anbefaler alle å søke opp ord eller utrykk som en er usikker på hva betyr. F.eks. Google: «what is azure cloud shell» om en lurer på hva dette er for noe.

For å kunne gjennomføre denne øvingen trenger dere en service principal som har rettigheter til å opprette ressurser i Azure tenanten som vi benytter og en bruker som har read only pålogging til Azure Portal – <https://portal.azure.com>

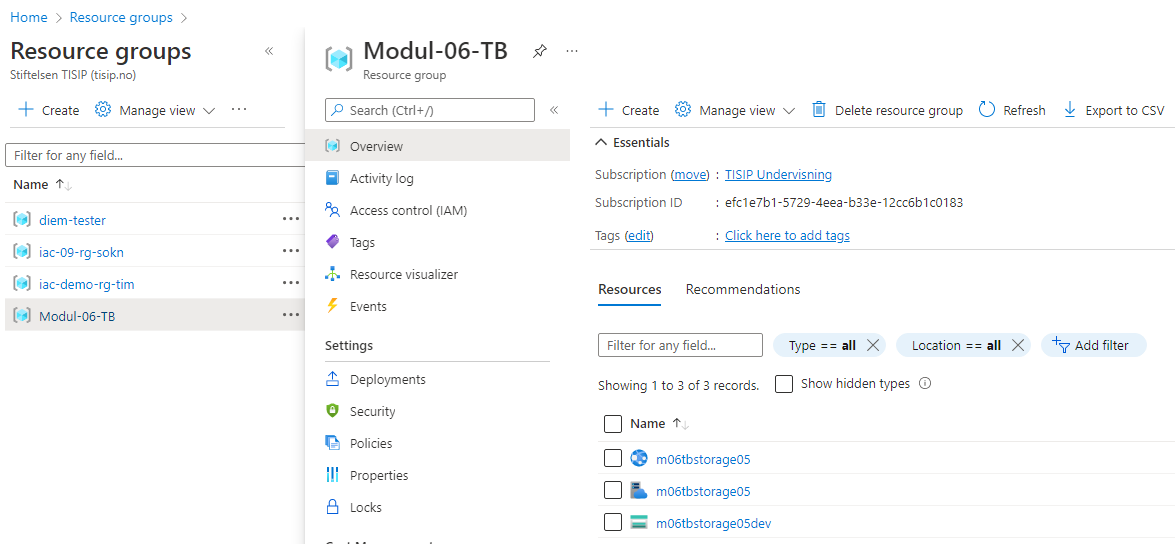
Service principal inneholder en ID og en Secret som er to lange strenger

Bruker med read only er en e-postadresse og passord.

Som levering på denne øvingen kan en levere en print screen av Azure Portal samt innholdet fra arm templaten og powershell-filen i et word dokument.

**HUSK**: Slett alltid ressursene etter at dere ferdig.

## Azure portal



## ARM-template

{

    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",

    "contentVersion": "1.0.0.0",

        "parameters": {

            "accountName": {

                "type": "string",

                "minLength": 3,

                "maxLength": 24,

                "metadata": {

                    "description": "Account name for parent storage account"

                }

            },

            "environment": {

                "type": "string",

                "allowedValues": ["dev","prod","qa","test"],

                "metadata": {

                    "description": "Select environments for creating resources"

                }

            },

            "apiVersion": {

                "type": "string",

                "allowedValues": ["2015-05-01-preview", "2015-06-15", "2016-01-01", "2016-05-01", "2016-12-01", "2017-06-01", "2017-10-01", "2018-02-01", "2018-03-01-preview", "2018-07-01", "2018-11-01", "2019-04-01", "2019-06-01", "2020-08-01-preview", "2021-01-01", "2021-02-01", "2021-04-01", "2021-06-01", "2021-08-01"],

                "defaultValue": "2019-06-01",

                "metadata": {

                    "description": "Selection for Azure API-version"

                }

            },

            //Defines possible storage configurations

            "storageSKU": {

                "type": "string",

                "defaultValue": "Standard\_LRS",

                "allowedValues": ["Standard\_LRS","Standard\_GRS","Standard\_ZRS","Premium\_LRS"],

                "metadata": {

                    "description": "Gives select options for storage configuration on storage account"

                }

            },

            "serverfarmSKU": {

                "type": "string",

                "defaultValue": "S1",

                "allowedValues": ["S1"],

                "metadata": {

                    "description": "Define SKU for serverfarm"

                }

            },

            "planName": {

                "type": "string"

            },

            "webAppName": {

                "type": "string"

            }

        },

        "variables": {

            "accountName": "[concat(parameters('accountname'), parameters('environment'))]",

            "planName": "[concat(parameters('planName'), parameters('environment'))]",

            "webAppName": "[concat(parameters('webAppName'), parameters('environment'))]"

        },

        "resources": [

            {

                "name": "[variables('accountName')]",

                "type": "Microsoft.Storage/storageAccounts",

                "apiVersion": "[parameters('apiVersion')]",

                "sku": {

                    "name": "[parameters('storageSKU')]"

                },

                "location": "[resourceGroup().location]",

                "properties": {

                    "accountType": "[parameters('storageSKU')]"

                },

                "resources": [

                    {

                        "type": "blobServices/Containers",

                        "apiVersion": "[parameters('apiVersion')]",

                        "name": "default/output",

                        "dependsOn": [

                            "[resourceId('Microsoft.Storage/storageAccounts', variables('accountName'))]"

                        ]

                    }

                ]

            },

            {

                "type": "Microsoft.Web/serverfarms",

                "apiVersion": "2018-02-01",

                "name": "[parameters('planName')]",

                "location": "[resourceGroup().location]",

                "sku": {

                    "name": "[parameters('serverfarmSKU')]"

                }

            },

            {

                "type": "Microsoft.Web/Sites",

                "apiVersion": "2018-02-01",

                "name": "[parameters('webAppName')]",

                "location": "[resourceGroup().location]",

                "dependsOn": [

                    "[resourceId('Microsoft.Web/serverfarms', parameters('planName'))]",

                    "[resourceId('Microsoft.Storage/storageAccounts', variables('accountName'))]"

                ],

                "properties": {

                    "serverFarmId": "[resourceId('Microsoft.Web/serverfarms', parameters('planName'))]",

                    "siteConfig": {

                        "appSettings": [

                            {

                                "name": "storageAccessKey",

                                "value": "[listKeys(resourceId('Microsoft.Storage/storageAccounts', variables('accountName')), '2019-06-01').keys[0].value]"

                            }

                        ]

                    }

                }

            }

        ],

        "outputs": {

            "Name": {

                "type": "string",

                "value": "[variables('accountName')]"

            }

        }

}

## Powershell-skript

#Variables

$ResourceGroup = 'Modul-06-TB'

$ResourceDirectory = 'C:\IAC-TISIP-2022\Modul-06'

$UserInput = $null

#Create Azure resource group

New-AzResourceGroup `

    -Name $ResourceGroup `

    -Location 'West Europe' `

    -Force

#Create group deployment

New-AzResourceGroupDeployment `

    -ResourceGroupName $ResourceGroup `

    -TemplateFile "$ResourceDirectory\Modul-06.json" `

    -TemplateParameterFile "$ResourceDirectory\Modul-06-DEV.parameters.json"

#Production

New-AzResourceGroupDeployment `

    -ResourceGroupName $ResourceGroup `

    -TemplateFile "$ResourceDirectory\Modul-06.json" `

    -TemplateParameterFile "$ResourceDirectory\Modul-06-PROD.parameters.json"

#Gets user-input y/n for deletion of resource

Write-Host -ForegroundColor Cyan 'Resource Created. Verify resource before continuing...'

[String]$UserInput = Read-host -Prompt 'Enter anything and press enter when you want to remove resource'

if ($UserInput -eq 'y' -or 'yes') {

    Remove-AzResourceGroup -Name $ResourceGroup -Force -Verbose

}

## Parameter-file (PROD)

{

    "schema": "https://schema.management.azure.com/schemas/201/deploymentParameters.json#",

    "contentVersion": "1.0.0.0",

    "parameters": {

        "accountName": {

            "value": "m06tbstorage05"

        },

        "planName": {

            "value": "m06tbstorage05"

        },

        "webappName": {

            "value": "m06tbstorage05"

        },

        "environment": {

            "value": "prod"

        },

        "storageSKU": {

            "value": "Premium\_LRS"

        },

        "apiVersion": {

            "value": "2019-06-01"

        }

    }

}

# TEORETISK ØVING

* Hvilke ressursen kan en sette som Child Resources?
  + Man kan sette undertjenester til en ressurs som en «child resource». Om ressursen ikke er en undertjeneste, så vil ikke ARM gjenkjenne ressursen du ønsker å opprette under «parent resourcen».
* Hvis en setter en ressurs som Child Resource, er det da gitt at den er avhengig av Parent Resource?
  + Det er ikke gitt at den er avhgengig av parent resource. Om den skal ta hensyn til parent så må det fremdeles defineres en «dependsOn».
* Hvilke fordeler ser du ved bruk DependsOn blocks og utrulling av større komplekse miljøer?
  + Fordelene ved bruk av «dependsOn» gjelder alt fra automatisering av utrullingen på et avansert nivå, i tillegg til feilsikring hvor da en ressurs vil ta hensyn til en annen ressurs før opprettelse. Dette kan gjøre en utrulling mer sømløs, og i tillegg kreve mindre feedback fra bruker/bestiller.

Lykke til med øvingen og faget