resource group: RG-ManualDeploy-os08

Security group: GDBC2018\_os08\_sg

user:g-objectsharp-user037@gbcd2018.onmicrosoft.com

#2

**Achievement**

In the Azure portal you will create a new Azure Web App with a simple App Service Plan. You will also create an Azure SQL Server that can later be used to contain the data of your Web application.

Using the Azure Portal you can connect to the SQL Database to create a table and add some data. Create a table called TestArtist containing an ID and names of artists. Attached you will find a script called create-artists.sql.

ACCEPTANCE CRITERIA

* When you browse to the Azure website, you will see the welcome screen of web apps where they explain how to deploy applications
* A SQL Database has been created containing 1 table TestArtist

Webapps: os08-web-asp

SQL DB: os08-web-sqldb

SQL server: os08-web-sqlserver.database.windows.net

Admin: admin08

Pswd: Icecream08

CREATE TABLE TestArtist ( Id int primary key, Name varchar(255) not null )

#3

The development team experimented with the manual creation of Resource Groups, but soon found that the manual steps are not the way to go. Everybody needs to be able to create a resource group and this need to be a predictable and reliable process. With the true DevOps Mindset they want to start automating everything.

### Achievement

In this achievement you will create an automation script to automatically create Resource Groups. You will also grant your security group the needed permissions to these groups. This should be done in an automated fashion (Azure CLI / Powershell or...) To get you familiar with Azure and the concept of Resource Groups / Security and Access you need to do the following things.

* Create an automation script to create a resource group for your team
  + RG-Playground-Teamname
* Create an automation script to add your AAD Security group to the resource groups
  + Security group (Teamname) – Contributor

**ACCEPTANCE CRITERIA**

* 1 resource groups is created in an automated fashion
* Security Group (Teamname) has Contributor access on the Resource Group. This has been granted in an automated fashion
* Users in the security group should be able to access the Resource Group and be able to create resources (e.g. a Web App)

<http://hadzimahmutovic.com/azure/azure-cli/cheatsheet/2018/03/14/azure-cli-cheat-sheet.html>

az group create --name RG-cli-os08 --location eastus

az ad group list --display-name GDBC2018\_os08\_sg <get object id of security group)

az role assignment create --role Contributor --assignee-object-id 4bb5175d-dfc0-407c-9ee8-e3c65c8481bc --resource-group RG-cli-os08

#4

The team created the required Azure resources to publish their website. But they want to create more environments that are exactly the same so they can use this for testing purposes. One of the benefits of the cloud is they can create new resources fast and discard them when not needed anymore. The team wants to be able to run an automation script that can create the previously created resources (resource group, Web application, SQL server)

### Achievement

In this achievement you will create a Web App and SQL Database in a fully automated fashion. The deployment of the web application is still done from Visual Studio, but at least the creation of the required resources are done automatically. Create a new resource group or use the existing resource groups from previous challenges.

* A script that can be run from the commandline that creates a Web Application and SQL Server **Note:** you should choose a complex password for the SQL Server admin password otherwise it fails setting the password without specific reason
* The Resources are named uniquely based on a provided parameter in the script "Environment\_name". For example when running the script with the parameter "Dev", the Web App is called WebApp-TeamName-Dev.
* After running the script the resources are available in Azure
* The SQL Server firewall settings allows access to Azure services

az appservice plan create -g "RG-cli-os08" -n web-asp-os08

az webapp create -g RG-cli-os08 -p web-asp-os08 -n webapp-os08-dev

az sql server create --admin-password Icecream08 --admin-user admin08 --location EastUS --name sql-server-os08-dev --resource-group RG-cli-os08

az sql server firewall-rule create --start-ip-address "0.0.0.0" --end-ip-address "0.0.0.0" --name sql-server-os08-dev-firewall --resource-group RG-cli-os08 --server sql-server-os08-dev

az sql db create -g RG-cli-os08 -s sql-server-os08-dev -n sql-db-os08-dev --service-objective S0

#5

Now that automation scripts are created, the move to DevOps has really started. The automation scripts are available to be used, but we want to keep our sources safe, track the changes we made and be able to use the scripts in our build and release pipeline. This way the team can ensure a reliable and stable way of rolling out new environments on demand. The team wants to use Git as the version control repository and they want to keep the automation separate from the application sources. They create a new Git Repo in the same team project, so the sources are kept separate.

### Achievement

In this achievement you will create a new Git Repository in your VSTS Team project that holds all the automation scripts and Infrastructure as Code files. Call this Git Repo Team-IaC.

**ACCEPTANCE CRITERIA**

* A new Git Repository that only contains the automation scripts to create the required Azure resources
* Git Repository contains the automation scripts for Resource Group creation and Resources creation
* Team members are able to clone the Git Repo
* Team members can make changes to the Git Repo

Git-repo-os08

#ch001-ac006

Automation can be done in many ways. In Azure the templated approach which is also idempotent is by using ARM templates. An ARM template can be deployed using a pipeline or on the command line

### Achievement

In this achievement you will create ARM templates to deploy the Web Application and SQL Server.

* An ARM template for SQL Server which is triggered from command line and creates SQL Server.
* An ARM template for Web App which is triggered from command line and creates Web App.

#CH001-AC007 Manually deploy the web application

In a previous achievement you have created a Web App and SQL Database in a fully automated fashion. The web application can now be manually deployed from your development environment to the Azure App Service.

* You have a clone of the GDBC-website repo on your development environment
* The GDBC-website application makes use of the SQL database you have automatically created
* The application builds and runs in your development environment
* The application is manually deployed to Azure from within Visual Studio and runs in Azure.

<https://docs.microsoft.com/en-us/visualstudio/deployment/quickstart-deploy-to-azure>

<https://docs.microsoft.com/en-us/vsts/git/tutorial/clone?view=vsts&tabs=command-line#clone-from-visual-studio-team-services--team-foundation-server>

CH002-AC001 Set up a private build agent

In order to build and release the software using the VSTS tooling, you will need a build agent. Next to the hosted agents, provisioned by VSTS, you have the ability to provision your own private build agents.

* The build agent is installed and configured using the instructions provided by Microsoft.
* Build agent is linked to the already provisioned agent queue for your team.

<https://docs.microsoft.com/en-gb/vsts/pipelines/agents/agents?view=vsts>

<https://docs.microsoft.com/en-gb/vsts/pipelines/agents/v2-windows?view=vsts>

CH002-AC002 Set up a Continuous Integration Build (CI)

The GDBC Inc. wants to be able to create the installation package for the website every time a change in source control is pushed. Currently, only one developer can build the software on his own developer machine and this creates a dependency with that person in order to deliver new features. Furthermore the team does not have a validation if the software is still in a deliverable state.

* When you commit and push new changes to the master branch a build is triggered.
* The build has at least the steps to compile and deliver the WebDeploy package, run the available unit tests and publish this to the artifact store
* After the build is finished you can browse the artifact store and see the zipfile that can be deployed to the webserver.

<https://docs.microsoft.com/en-us/vsts/pipelines/get-started-designer?view=vsts#a-quick-introduction-to-cicd>

CH002-AC003 Set up a Continuous Deployment (CD)

GDBC Inc. has a CI build but still has to manually deploy their site to the various test and production environments. They have word documents that can be used as a step by step guide to do the installation, but this has proven to be very error prone. Almost every deployment goes wrong because things are missing. This results in delays or even production incidents.

The Management team of GDBC Inc. has asked the team to improve this and the team has proposed to further automate the deployment of the software using a Continuous Deployment Pipeline. To save some costs management has asked to make use on-demand environments

* When a CI build is finished, the new version of the software is automatically deployed to the Test environment. This test environment is in Azure. **Note:** a script is provided which you can use to create a Azure Service Principal. This is needed when creating a Azure Resource Manager service endpoint that allows you to connect VSTS to Azure]
* A new environment is created automatically at the beginning of the deployment. **Note:** a script is provided to create a resource group with a web app]
* The release pipeline contains a way to break down the environment when deployment was done. **Note:** a script is provided to delete the resource group]

<https://docs.microsoft.com/en-us/azure/app-service/app-service-web-get-started-dotnet>

<https://damianbrady.com.au/2018/02/01/friends-dont-let-friends-right-click-publish/>

<https://docs.microsoft.com/en-us/vsts/pipelines/apps/cd/deploy-webdeploy-webapps?view=vsts#cd>

<https://docs.microsoft.com/en-us/vsts/pipelines/library/service-endpoints?view=vsts#create-a-service-endpoint>

CH003-AC001 Set up separate Playground Test and Production resource groups

GDBC Inc. experienced an outage last week, because someone in the team made a mistake with cleaning up resources in Azure. Instead of cleaning the test environment resources, by accident, the production resources got deleted. Luckily the automation scripts and CI/CD pipelines were there to redeploy quite easily, but still, the outage took over one-hour to overcome.

GDBC Inc. management wants to get more assurance that this can not happen again and appropriate measures are taken.

The team decided to split the various environments also in Azure. For this they want to create separate resource groups and provide access to only the service principal, so creation and clean-up are part of the deployment pipelines. This removes any manual intervention and hence makes the deployments more stable and reliable.

* There is an automated script to create the 3 resource groups
* One service principal is created that can be used to deploy to all Azure Resource Groups (that the team owns)
* Scripts are committed to the Git Repository so it can be used by the automated deployment pipelines.

<https://docs.microsoft.com/en-us/azure/azure-resource-manager/vs-azure-tools-resource-groups-deployment-projects-create-deploy>

[https://docs.microsoft.com/en-us/cli/azure/create-an-azure-service-principal-azure-cli?view=azure-cli-latest](https://docs.microsoft.com/en-us/cli/azure/create-an-azure-service-principal-azure-cli?view=azure-cli-latest https://docs.microsoft.com/en-us/cli/azure/group?view=azure-cli-latest)

[https://docs.microsoft.com/en-us/cli/azure/group?view=azure-cli-latest](https://docs.microsoft.com/en-us/cli/azure/create-an-azure-service-principal-azure-cli?view=azure-cli-latest https://docs.microsoft.com/en-us/cli/azure/group?view=azure-cli-latest)

<https://docs.microsoft.com/en-us/powershell/azure/create-azure-service-principal-azureps?view=azurermps-5.7.0>

<https://docs.microsoft.com/en-us/powershell/module/azurerm.resources/new-azurermresourcegroup?view=azurermps-5.7.0>

CH003-AC002 Extend Release pipeline with new environments

The outage made people aware that is better to roll out gradually before moving to production. The rollout to different environments should be embedded in the pipeline and be triggered every time a change is committed.

* Release Pipeline creates Azure Resources for Dev, Test and Production
* Release Pipeline contains three environments
* Each environment targets it own Azure resource group
* VSTS Endpoint is configured and used in the Release Pipeline

<https://docs.microsoft.com/en-us/cli/azure/create-an-azure-service-principal-azure-cli?view=azure-cli-latesthttps://docs.microsoft.com/en-us/vsts/pipelines/apps/cd/deploy-webdeploy-webapps?view=vsts>