Account

VSTS:

Globaldevopsbootcamp-nsa.visualstudio.com

Azure: <https://portal.azure.com/#@gdbc2018.onmicrosoft.com/dashboard/private/23271ad0-b420-4cd3-a0b0-a5ac3d4d7010>

G-ObjectSharp-User037@ gdbc2018.onmicrosoft.com

Pswd:Welcome2GDBC2018

Project:GDBC2018-ObjectSharp-Team08

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

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]

<need VS2017 or GIT/.NET core for LInux>

<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>

[https://docs.microsoft.com/en-us/powershell/azure/create-azure-service-principal-azureps?view=azurermhttps://docs.microsoft.com/en-us/vsts/pipelines/library/service-endpoints?view=vstsps-5.7.0](https://docs.microsoft.com/en-us/powershell/azure/create-azure-service-principal-azureps?view=azurermps-5.7.0)

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

create service principle account

az ad sp create-for-rbac --name ServicePrincipalName --password PASSWORD

manage service principle roles

az role assignment create --assignee APP\_ID --role Reader

az role assignment delete --assignee APP\_ID --role Contributor

az role assignment list --assignee APP\_ID

Sign in using service principle

az login --service-principal --username APP\_ID --password PASSWORD --tenant TENANT\_ID

If you plan to manage your app or service with Azure PowerShell, you should run it under an Azure Active Directory (AAD) service principal, rather than your own credentials

Get-AzureRmADApplication -DisplayNameStartWith MyDemoWebApp

Create service principal for your app

Add-Type -Assembly System.Web

$password = [System.Web.Security.Membership]::GeneratePassword(16,3)

$securePassword = ConvertTo-SecureString -Force -AsPlainText -String $password

New-AzureRmADServicePrincipal -ApplicationId 00c01aaa-1603-49fc-b6df-b78c4e5138b4 -Password $securePassword

CH003-AC003 Use secrets in your pipeline

To ensure no secrets end up in Source Control or in plain text, secret variables or a keyvault can be used. Sensitive data that is required for deployment must be inserted in the scripts at deployment time.

* A SQL Script that creates a SQL user and assigns the user the db\_datareader role
* Script should contain tokens for secrets
* Replacement of secrets with pipeline variables
* Optional; Execution of SQL Script against the created database

<https://docs.microsoft.com/en-us/vsts/pipelines/release/variables?view=vsts&tabs=shell>

<https://marketplace.visualstudio.com/items?itemName=geeklearningio.gl-vsts-tasks-azure&targetId=6b2bf7bd-efaa-4180-bfb6-507d37b53ce1&utm_source=vstsproduct&utm_medium=ExtHubManageList>

CH003-AC004 Create a keyvault in your resource groups

When deploying your software, you most probably have to use some secrets here and there. Think of connection string password, username, and passwords for functional users or SQL users. VSTS has the possibility to store your secrets but once saved this can never be retrieved again. Keyvault has this possibility.

* Keyvault created in all resource groups
* Service Principal has Get, List permissions on all keyvaults
* Security group has Get, List permissions on Test
* Security group has List permissions on Production
* Random string added as secret in all keyvault with the name [admin-password]
* Release Pipeline uses variable group linked to Keyvault

<https://docs.microsoft.com/en-us/azure/key-vault/key-vault-get-started>

Get-AzureRmSubscription (get subscription info)

Set-AzureRmContext -SubscriptionId <subscription ID>

New-AzureRmResourceGroup –Name 'ContosoResourceGroup' –Location 'East US'

New-AzureRmKeyVault -VaultName 'secret-KeyVault' -ResourceGroupName 'secret-rg' -Location 'East US'

### Azure Key Vault generates a software protected key

$key = Add-AzureKeyVaultKey -VaultName 'secret-KeyVault' -Name 'secret-FirstKey' -Destination 'Software'

Get-AzureKeyVaultKey –VaultName 'secret-KeyVault'

$Key.id

$secretvalue = ConvertTo-SecureString 'Pa$$w0rd' -AsPlainText -Force

$secret = Set-AzureKeyVaultSecret -VaultName 'secret-KeyVault' -Name 'SQLPassword' -SecretValue $secretvalue

Applications that use a key vault must authenticate by using a token from Azure Active Directory. To do this, the owner of the application must first register the application in their Azure Active Directory

| **DISPLAY NAME** | | **APPLICATION TYPE** | | **APPLICATION ID** |
| --- | --- | --- | --- | --- |
| **webapp-os08-dev** | **Web app / API** | | **e62afec4-db6c-4d5d-bb14-952d7b71f0a3** | |

There are two ways to authorize the application to access the key or secret in the vault.

https://docs.microsoft.com/en-us/cli/azure/keyvault?view=azure-cli-latest

Set-AzureRmKeyVaultAccessPolicy -VaultName 'ContosoKeyVault' -ServicePrincipalName 8f8c4bbd-485b-45fd-98f7-ec6300b7b4ed -PermissionsToKeys decrypt,sign

CH003-AC005 Use keyvault secrets in your pipeline

To ensure no secrets end up in Source Control or in plain text, a keyvault is used. Sensitive data that is required for deployment must be inserted in the scripts at deployment time.

* A variable group is created for all environments which is linked to the keyvaults of the same environments
* Replace the variables with variables from keyvault

<https://docs.microsoft.com/en-us/vsts/pipelines/library/variable-groups?view=vsts>

Use a variable group to store values that you want to make available across multiple build and release definitions. Variable groups are defined and managed in the **Library** tab of the **Build & Release** hub.

Link an existing Azure key vault to a variable group and map selective vault secrets to the variable group.

CH004-AC001 Run Unit Test in your build

Make sure that your app still works after every check-in and build using VSTS. Find problems earlier by running tests automatically with each build.

When your build is done, you can review your test results to start resolving the problems that you find.

**Achievement**

In this exercise you will add running unit tests to your VSTS build. A good continuous integration process includes running tests to validate that the code always is in good shape. With code coverage data you can also monitor the trend to ensure that the developers keep adding tests in proportion to added code.

* Configure your build to run unit tests and collect code coverage
* Run a build and inspect the test run information

<https://docs.microsoft.com/en-us/vsts/pipelines/test/getting-started-with-continuous-testing?view=vsts>

CH005-AC002 Setup branch policies for code review

After the audit, it became clear that a developer that makes a change to the code is also the reviewer of that same code. to tackle this the IT department of GDBC Inc. wants to setup a structural code review process.

This code review process will not only solve the compliancy audit issue. Since GDBC Inc. is adopting a DevOps way of working the IT department wants to catch bugs or issues as early as possible. Reviewing the code will make catching these bugs or issues shift to the left of the software delivery process.

Also the DevOps culture of sharing knowledge will be stimulated, because developers will see each others solutions to a given requirement and educate each other in delivering higher quality code.

**Achievement**

In this challenge you will configure a branch policy that will enforce that a code review is required before a pull request can be completed.

Setting up a code review process will achieve the following goals:

* Reduce the risk of low quality/malicious/insecure code getting into production that can cause damage to the company in any way
* Minimize the amount of technical debt introduced into a code base
* Increase team collaboration and sharing of knowledge

**ACCEPTANCE CRITERIA**

* Before a pull request can be completed the code must be reviewed by at least one other developer.
* It is not allowed for a developer to accept his/her own changes
* All reviewers must accept the proposed code change before it can be completed

CH005-AC003 Force traceability of work items

The internal audit also listed a traceability issue. It is not clear which requirement or issue was the reason for the code change. End-to-end traceability was setup, but apparently not all developers make sure the administration to get the audit trail is done.

The GDBC Inc. IT department needs to find a way to remember developers about this when they tend to forget and to ensure that traceability is always available.

### Achievement

In this challenge you will make sure that a code change is always linked to at least one work item by configuring a branching policy.

**ACCEPTANCE CRITERIA**

* A pull request must be linked to at least one work item otherwise it will be blocked
* Completing the pull request without a linked workitem fails

CH005-AC004 Set up traceability of every change to production

One of the things that needs to be provided as evidence in almost any audit is end-to-end traceability. Who made this change, when was this change made and how did this change end up in the code deployed to production. And is this change really on production and has it not been modified during the deployment process.

**Achievement**

In this achievement you will show that compliancy is not as hard as it seems. You will proof to the regulatory office that any requirement can be traced. For that, you are going to implement a change and deploy this to production.

* Create a PBI that describes a change in your backlog
* Write some code (or dummy code) and link this change to the requirement
* Create a build that contains this change
* Run tests in the build
* Create a release that contains this build
* Run tests in the release

**ACCEPTANCE CRITERIA**

* The release report contains information about the requirement, code changes, tests and builds

CH005-AC005 Link test cases to requirements

If your test suites include requirements, link these to your test results and view the results on your team's dashboard. This enables end-to-end traceability of requirements for agile teams. For example, when teams do not use planned testing (by creating test plans or test case work items), and instead choose to simply write automated tests that run in the CI/CD pipeline, associating test results with requirements provides an easy way to monitor test results and ensure requirements are met.

**Achievement**

In this step you will visualize the quality of developed features using requirement to test association.

* Associate tests with requirements
* Add a Requirements quality widget to the dashboard that shows the test quality for relevant requirements

**ACCEPTANCE CRITERIA**

* Tests have been associated with requirements
* Requirements with associated tests shown on the dashboard using the Requirements quality widget

<https://docs.microsoft.com/en-us/vsts/pipelines/tasks/test/run-functional-tests?view=vsts>

**CH006-AC001 Manually create a deployment slot in your web app and deploy to this**

In order to improve the release quality and reduce downtime the team wants to move to deployment slots. The slots help to improve quality by allowing to test the release in a staged environment before it goes into production. When the system is released to production, downtime is reduced by simply swapping the current production environment with the staged environment.

**Achievement**

In this challenge you will add deployment slots to the infrastructure to improve the release process.

* Upgrade the app service hosting plan to a tier that supports deployment slots Minimal type "Production S1"
* Add a deployment slot to the web app
* Change the release pipeline to deploy to a slot
* Web app has deployment slots
* A release has been run that deploys to the deployment slot
* The web app has been tested from the deployment slot

<https://docs.microsoft.com/en-us/azure/app-service/web-sites-staged-publishing>

Set up staging environments

web-asp-os08 small D1

Get-AzureRmResourceGroupDeployment -ResourceGroupName RG-ManualDeploy-os08

How to deploy app in azure

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

Create a deployment user: < his deployment user is required for FTP and local Git deployment to a web app>

az webapp deployment user set --user-name deployment\_icecream --password icecream08

create resource group

az group create --name rg\_py\_qli --location "east us"

create app service plan

az appservice plan create --name asp\_py\_qli --resource-group rg\_py\_qli --sku FREE

create web app

az webapp list-runtimes |grep -i python

az webapp create --resource-group rg\_py\_qli --plan asp\_py\_qli --name app-py-qli --runtime "python|3.4" --deployment-local-git

< git remote add azure <https://deployment_icecream@app-py-qli.scm.azurewebsites.net/app-py-qli.git>>

"enabledHostNames": [

"app-py-qli.azurewebsites.net"

"deploymentLocalGitUrl": <https://deployment_icecream@app-py-qli.scm.azurewebsites.net/app-py-qli.git>

URL: https://app-py-qli.azurewebsites.net

Push to Azure from Git

git remote add azure <deploymentLocalGitUrl-from-create-step>

git remote add azure <https://deployment_icecream@app-py-qli.scm.azurewebsites.net/app-py-qli.git>

git push azure master (prompt for deploymentuser id/pswd)

< <https://app-py-qli.scm.azurewebsites.net/>>

web-aps-os08 - Deployment slots – S1

automate slotswap

* 1. create webapp

New-AzureRmWebApp -ResourceGroupName [resource group name] -Name [app name] -Location [location] -AppServicePlan [app service plan name]

* 1. create deployment slot

New-AzureRmWebAppSlot -ResourceGroupName [resource group name] -Name [app name] -Slot [deployment slot name] -AppServicePlan [app service plan name]

* 1. initiate swap with previous (multi-phase swap) and apply destination slot config t source slot

$ParametersObject = @{targetSlot = "[slot name – e.g. “production”]"}

Invoke-AzureRmResourceAction -ResourceGroupName [resource group name] -ResourceType Microsoft.Web/sites/slots -ResourceName [app name]/[slot name] -Action applySlotConfig -Parameters $ParametersObject -ApiVersion 2015-07-01

* 1. cancel pending swap

Invoke-AzureRmResourceAction -ResourceGroupName [resource group name] -ResourceType Microsoft.Web/sites/slots -ResourceName [app name]/[slot name] -Action resetSlotConfig -ApiVersion 2015-07-01

* 1. swap deployment slot

$ParametersObject = @{targetSlot = "[slot name – e.g. “production”]"}

Invoke-AzureRmResourceAction -ResourceGroupName [resource group name] -ResourceType Microsoft.Web/sites/slots -ResourceName [app name]/[slot name] -Action slotsswap -Parameters $ParametersObject -ApiVersion 2015-07-01

* 1. monitor swap events in activity log

Get-AzureRmLog -ResourceGroup [resource group name] -StartTime 2018-03-07 -Caller SlotSwapJobProcessor

* 1. delete deployment slot

Remove-AzureRmResource -ResourceGroupName [resource group name] -ResourceType Microsoft.Web/sites/slots –Name [app name]/[slot name] -ApiVersion 2015-07-01

**CH006-AC002 Automatically create a deployment slot in your web app and deploy to this**

The manual creation of Deployment slots stresses some of the developers. They don't want to forget this important stuff. In order to make this part of the release process, the creation of deployment slots needs to be automated.

**Achievement**

In this challenge you will add a script to automatically create deployment slots to your web application

**ACCEPTANCE CRITERIA**

* Web app has deployment slots
* A release has been run that deploys to the deployment slot
* The web app has been tested from the deployment slot
* The deployment slot is created as part of the release pipeline

https://github.com/Microsoft/vsts-tasks/blob/master/Tasks/AzureRmWebAppDeploymentV4/README.md

# app Service Deployment: ARM

the task is used to deploy a Web project to an existing Azure Web App. To deploy to Azure, an Azure subscription has to be linked to VSTS in account admin section

Create the [ARM](https://azure.microsoft.com/en-in/documentation/articles/resource-group-overview/) service endpoint, use **'Azure Resource Manager'** endpoint type

Deloyment - Based on the type of Azure App Service and VSTS agent, the task chooses a suitable deployment technology: web deploy, kudu REST API, container registry

az webapp deployment slot create --name

--resource-group

--slot

[--configuration-source]

List existing deployment slot

az webapp deployment slot list --name web-aps-os08 --resource-group RG-ManualDeploy-os08

< **RG-ManualDeploy-os08> which had deployment before; app nanme <web-aps-os08>**

az webapp deployment slot create --name web-aps-os08 --resource-group RG-ManualDeploy-os08 –-slot pre-staging

swap slots

az webapp deployment slot swap -g RG-ManualDeploy-os08 -n web-aps-os08 --slot staging --target-slot pre-staging

**CH006-AC003 Update your pipeline to swap slots when ready**

When a system can be deployed to an Azure app service deployment slot it's natural to integrate that with the release management system so that the system first is deployed to a staging environment. The deployment can then be tested in that environment and when it is confirmed to be working the staged environment can be swapped with current production version so that the stage becomes the production system.

**Achievement**

In this achievement you will integrate the swapping of Azure web apps from a staged environment (slot) to the production version.

* Add a release phase that swaps the staged environment with the live version
* Add an approval step that waits for the staged environment to be validated before it is swapped to production

**ACCEPTANCE CRITERIA**

* A release has been run that first deploys to the staged environment, then waits for approval and then swaps the stage to production

<https://github.com/Microsoft/vsts-tasks/blob/master/Tasks/AzureAppServiceManageV0/README.md>

**ARM:** The Azure App Service Management task is used to Start/Stop/Restart App Service, Swap Slot, Install Extentions, Enable [Continuous Monitoring](https://go.microsoft.com/fwlink/?linkid=859946) or Start/Stop all continuous WebJobs on an [Azure App Service](https://azure.microsoft.com/en-in/documentation/articles/app-service-web-overview/).

**CH006-AC004 Implement Zero Downtime for Database**

GDBC Inc sees that the vinyl records are great but there is also a big interest for separate songs (streaming), CD's and merchandise. They decide to create the possibility to also buy these items. Unfortunately the database does not support the desired changes and needs an update that will break existing code. Downtime cannot be afforded, so a plan is created to update the code and database in multiple steps to do a zero-downtime deployment of the database

**Achievement**

In this challenge you will implement a strategy for handling zero downtime updates to the database.

* Update the binaries to support multiple database schemas
* Update the database schema with new table
* Migrate existing data to new data structure
* Remove old table
* Change code to support one schema

**ACCEPTANCE CRITERIA**

* A change to the database has been deployed without breaking existing clients
* A change to the database has been deployed without affecting performance of the running application

**https://docs.microsoft.com/en-us/azure/sql-database/sql-database-designing-cloud-solutions-for-disaster-recovery**

**CH007-AC001 Add App Insights to your WebApp**

The GDBC Inc. development team would like to understand their users better and drive business investments and prioritization based on evidence and data instead of guessing. Azure has a Application Performance Monitoring service called Azure Application Insights which allows to gather telemetry to better understand how the business applications are being used.

**Achievement**

In this achievement you are going to create application insights resource through Azure Portal and add it to your web application so your team starts receiving telemetry about how the application is being used by users. You are going to enable usage telemetry and integrate that into our web application code.

**ACCEPTANCE CRITERIA**

* Application Insights resource created in the resource group
* Application code changed to include Application Insights Telemetry

[**https://docs.microsoft.com/en-us/azure/application-insights/app-insights-asp-net**](https://docs.microsoft.com/en-us/azure/application-insights/app-insights-asp-net)

**CH007-AC002 Set up Application Insights resource in Azure From Pipeline**

Deploying your infrastructure as code in source control, written and runnable in repeatable and desired state manner guarantees that the all environments created are consistently the same, they can always be recreated, would allow validation that the current state of the environment is according to the definition and if its not, the Azure Resource Manager would know what to amend and how to make it so.

All resources part of the infrastructure should be deployed in the same manner and Application Insights is merely another azure resource to be deployed.

Since GDBC Inc. wants to use application insights to collect data to understand how their application is being used by users, it is important to configure the application insights usage analytics within the application to send information to the newly deployed azure resource

The goal is to:

* Have one place where infrastructure is described
* Have one way of deploying infrastructure changes to all environments
* Be able to validate that the infrastructure matches the desired state
* To be able to collect usage analytics to the new application insights resource

**Achievement**

GDBC Inc development team already has automation (scripts/ARM template) to create the infrastructure in Azure. Application Insights is a new Azure resource that should be included in the automation so it also gets deployed in the same way. In this achievement you will:

* Change the automation and add the application insight resource to it
* Deploy the new resource using the changed template through the pipeline
* Add usage analytics to pages and configure correct instrumentation key into the application from pipeline

**ACCEPTANCE CRITERIA**

* Automation that can deploy application insights resource that is configured to be used by the application
* Completed release that has deployed application insights resource to one environment

[**https://docs.microsoft.com/en-us/azure/application-insights/**](https://docs.microsoft.com/en-us/azure/application-insights/)

**Free Pluralsight Video Training**

[**https://docs.microsoft.com/en-us/azure/application-insights/app-insights-powershell**](https://docs.microsoft.com/en-us/azure/application-insights/app-insights-powershell)

[**https://github.com/Azure/azure-quickstart-templates**](https://github.com/Azure/azure-quickstart-templates)

**CH007-AC003 Add Application Insights Usage Dashboard**

Now that the application insights resource is being deployed and configured as part of the release pipeline we want to see the usage data that is being collected in application insight. To see the usage data we need to either use the existing Usage Analysis views within application insights or create our own Usages dashboard in azure dashboards.

GDBC Inc. business wants to quickly get an overview of relevant usage data from application insights. A great way to bring out only the relevant data and important metrics to business is to create a custom Azure Dashboard that shows just that data and is easy to monitor by the business.

### Achievement

Create a custom azure dashboard that shows users statistics and active sessions / requests / users. Experiment and see what other usages data you can have on dashboard.

**ACCEPTANCE CRITERIA**

* Custom dashboard has been added to Azure Portal that shows active users / requests / sessions currently and other usage data

[**https://docs.microsoft.com/en-us/azure/application-insights/app-insights-dashboards**](https://docs.microsoft.com/en-us/azure/application-insights/app-insights-dashboards)

**CH007-AC004 Add Application Insights Custom Events for every Menu Item**

The business starts seeing the befenits of having the telemetry in place. Business would like to see additional data on menu items being clicked so they can easily identify what navigation paths and thus features are used.

**Achievement**

Track custom event data for every menu item click and make sure its shown on the new Azure Usages Dashboard

**ACCEPTANCE CRITERIA**

* Menu item clicks are tracked as custom events in application insights
* Custom events are shown on azure usages dashboard

[**https://docs.microsoft.com/en-us/azure/application-insights/app-insights-usage-overview**](https://docs.microsoft.com/en-us/azure/application-insights/app-insights-usage-overview)

**CH007-AC005 Add telemetry to measure customer behavior on deleting items in shopping basket**

GDBC Inc marketing team is constantly striving to improve conversion. the team would like to know if there is a correlation between the fact that people are deleting items from the shopping basket and the total amount of goods they have in the basket.

**Achievement**

Therefore they need telemetry data when a user deletes an item from the basket. The moment a customer deletes an item, they want to track the following information:

* Name of the album
* Name of the Artist
* Genre of the album
* Total Amount of the shopping basket

Since the amount in the basket can vary a lot, they want that to be bucketed into the following buckets:

* < 100 $
* between 100 and 200 $
* between 200 and 500 $
* over 500 $

The marketing team would like to see the data gathered in a graph and table so they can try to draw conclusions out of the data.

In order to get the data, you need to add some C# code to the shopping basket controller. The moment the delete method is called you need to log the additional data using the app insights c# SDK

Using the help function will provide guidance on using Application Insights Analytics, yes this comes at a price!

**ACCEPTANCE CRITERIA**

* On the app insights dashboard there is a graph that shows the genre and amount bucket of deleted records
* On the app insights dashboard there is a table that shows the top 5 genres that get deleted

[**https://docs.microsoft.com/en-us/azure/application-insights/app-insights-analytics**](https://docs.microsoft.com/en-us/azure/application-insights/app-insights-analytics)

**AKA.MS/AIGETSTRATED**

[**https://docs.microsoft.com/en-us/azure/application-insights/app-insights-api-custom-events-metrics#trackevent**](https://docs.microsoft.com/en-us/azure/application-insights/app-insights-api-custom-events-metrics#trackevent)

[**https://github.com/Microsoft/ApplicationInsights-Python**](https://github.com/Microsoft/ApplicationInsights-Python)

[**https://docs.microsoft.com/en-us/azure/application-insights/app-insights-usage-overview**](https://docs.microsoft.com/en-us/azure/application-insights/app-insights-usage-overview)

**CH008-AC001 Release only when all requirements are done**

Now that the CI/CD pipeline is working, every check-in from a developer is a potential release candidate. Although it is possible, the Product Owner wants to have some control over what is released to production. He wants to release a set of requirements at once. Work items are nicely tracked in VSTS and he wants to release the software once these items are all set to done

**Achievement**

In this achievement you are going to create a Deployment gate based on a Work Item Query. If the query results match your criteria, the release will be automatically approved and deployed.

* Create three (3) work items and tag them with "vNext"
* Create a Work Item query that lists all the "vNext" work items that are NOT done
* Create a pre-deployment gate that approves when this query does not get any results

**ACCEPTANCE CRITERIA**

* Query for vNext items that are not done
* Query Work Items Deployment gate defined for Production Environment to approve when there are 0 results
* Automatically approved release by Deployment Gate

[**https://docs.microsoft.com/en-us/vsts/pipelines/release/approvals/gates?view=vsts**](https://docs.microsoft.com/en-us/vsts/pipelines/release/approvals/gates?view=vsts)

**CH008-AC002 Create an Azure Function**

A situation occurred where there were known issues with some Azure features. The fully automated CI/CD release pipeline, still did a release. Because of this release the application crashed and things got worse. The GDBC Inc. development team want to have the possibility to "stop the line" when there are known issues. They want to address this by creating a check that returns OK or NOK before releasing to production.

**Achievement**

In this achievement you will create an Azure function. The purpose of the function is to return status for a GO or NO GO. If the function returns OK then all is working as expected and the deployment can continue to the next stage of the deployment.

* Create an Azure function that returns OK or NOK
* Test your Azure function
* (optional) Automate the deployment of the Azure function
* (optional) Automate a test for the Azure function

**ACCEPTANCE CRITERIA**

* Azure function operational
* Azure function verified in for example Postman

[**https://docs.microsoft.com/en-us/azure/azure-functions/functions-create-first-azure-function**](https://docs.microsoft.com/en-us/azure/azure-functions/functions-create-first-azure-function)

https://docs.microsoft.com/en-us/azure/azure-functions/functions-overview

Create a function app

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-create-first-azure-function-azure-cli-linux>

function: web-asp-os08

app: web-asp-os08

windows, resource group: RG-ManualDeploy-os08

**https://web-asp-os08.azurewebsites.net/api/HttpTriggerJS1?code=R8nCjwap2AoXq0abPauFgUyVaFLvk2p4Duca/Gs7OamAbrMuR7juYQ==**&name=<Cocacola>

az storage account create --name <storage\_name> --location westeurope --resource-group myResourceGroup --sku Standard\_LRS

Storage a/c: webaspos088f2e

RG: RG-ManualDeploy-os08

Create a Linux App Service plan

az appservice plan create --name asp-py-lx --resource-group rg\_py\_qli --sku S1 --is-linux

az functionapp create --name func-web-asp-os08 --storage-account webaspos088f2e --resource-group RG-ManualDeploy-os08 --plan ServicePlan9dbf68bf-9ddc --deployment-source-url https://github.com/Azure-Samples/functions-quickstart-linux

<https://func-web-asp-os08.azurewebsites.net>

The sample repository currently includes two scripting files, [**deploy.sh**](https://github.com/Azure-Samples/functions-quickstart-linux/blob/master/deploy.sh) and [**.deployment**](https://github.com/Azure-Samples/functions-quickstart-linux/blob/master/.deployment). The .deployment file tells the deployment process to use deploy.sh as the [**custom deployment script**](https://github.com/projectkudu/kudu/wiki/Custom-Deployment-Script). In the current preview release, scripts are required to deploy the function app on a Linux image.

Test the function

curl http://<app\_name>.azurewebsites.net/api/HttpTriggerJS1?name=<yourname>

or with function name

curl http://<function\_name>.azurewebsites.net/api/HttpTriggerJS1?name=<yourname>

curl https:// func-web-asp-os08.azurewebsites.net/api/HttpTriggerJS1?name=Kenya

az group delete --name myResourceGroup

[**https://docs.microsoft.com/en-us/azure/azure-functions/functions-run-local**](https://docs.microsoft.com/en-us/azure/azure-functions/functions-run-local)

# Work with Azure Functions Core Tools

develop and test your functions on your local computer from the command prompt or termin

[**https://docs.microsoft.com/en-us/azure/azure-functions/functions-create-your-first-function-visual-studio**](https://docs.microsoft.com/en-us/azure/azure-functions/functions-create-your-first-function-visual-studio)

[**https://docs.microsoft.com/en-us/azure/azure-functions/functions-test-a-function**](https://docs.microsoft.com/en-us/azure/azure-functions/functions-test-a-function) [**https://www.jan-v.nl/post/automate-deploying-azure-functions-with-vsts**](https://www.jan-v.nl/post/automate-deploying-azure-functions-with-vsts)

**CH008-AC003 Add a release gate that triggers the Azure function**

The Azure Function that returns the status if the deployment can proceed works great. GDBC Inc. wants to make this a part of the automated deployment process so that no manual interaction is needed.

**Achievement**

In this achievement you will add a release gate to your release pipeline. The release gate will call an Azure function to decide if the deployment can continue forward.

**Note:** The Azure function to be used here should have been created in an earlier challenge.

* Add a deployment gate to your release definition that calls an Azure function
* Run a new release and see how gates will speed up the release cycle

**ACCEPTANCE CRITERIA**

* Deployment gate that calls the Azure function added to pipeline
* Completed release that continued deployment based on the gate evaluation

**https://docs.microsoft.com/en-us/vsts/pipelines/release/approvals/gates?view=vsts**

**CH008-AC004 Add a Release gate on app performance**

The last deployment resulted in a panic situation at GDBC Inc. after a few hours the site went down. A thorough root cause analysis showed that a simple code change resulted in a memory leak, which eventually brought down the site. The problem was reproducable in the test environment.

The CTO wants that these things are checked before going to production. The Lead Architect knows how to use Azure Monitor Alerts. He wants to use the status of this monitor to decide if the release can proceed.

**Achievement**

In this exercise you will create an Azure alert that monitors the behaviour and performance of your application. Next you will add this insight into the release pipeline so that it can continue or halt the release process.

* Create an Azure alert
* Add deployment gate that uses this alert
* Run a new release and validate that the Azure alert is called

**ACCEPTANCE CRITERIA**

* Azure Alert created
* Azure Alert added to deployment gate
* Completed release that continued deployment based on the gate

[**https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/monitoring-overview-unified-alerts**](https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/monitoring-overview-unified-alerts)

**https://docs.microsoft.com/en-us/vsts/pipelines/release/approvals/gates?view=vsts**

**Release deployment control using gates**

**CH009-AC001 Internally - Post to Slack/Teams**

To get the word out internally about new feature releases, GDBC want to post a message to one of their Slack channels for every new release to the test environment.

**Achievement**

In this achievement you will extend your release definition to post a Slack message after a successful deployment to the test environment.

* GDBC is cannot provide a Slack Workspace, use an existing (at your own risk!) or create a new Slack Workspace

**ACCEPTANCE CRITERIA**

* A message in a Slack channel or MS Teams channel (use your own or create one) for every successful deployment to the test environment

[**https://docs.microsoft.com/en-us/vsts/service-hooks/overview?view=vsts**](https://docs.microsoft.com/en-us/vsts/service-hooks/overview?view=vsts)

**https://docs.microsoft.com/en-us/vsts/service-hooks/services/slack?view=vsts**

**CH009-AC002 Externally - Post to Twitter.. With some GDBC details**

Every new release to production is worthy of some publicity. GDBC. Inc wants to create some buzz by publishing a message to Twitter for every new release to production.

**Achievement**

In this achievement you will extend your release definition to send a tweet every time a new deployment has made it all the way to production.

**ACCEPTANCE CRITERIA**

* A tweet with the hash tags #gdbc and #releasetweet is sent when a new release has been deployed into production

[**https://marketplace.visualstudio.com/items?itemName=petergroenewegen.PeterGroenewegen-Xpirit-Vsts-Release-Twitter**](https://marketplace.visualstudio.com/items?itemName=petergroenewegen.PeterGroenewegen-Xpirit-Vsts-Release-Twitter)

**CH010-AC001 Hide a discount banner and sales funnel with a feature toggle**

The marketing and sales department of GDBC Inc. are working on a global marketing campaign during which a few top selling products will be offered with significant price reductions during a limited time window. The exact date at which the campaign will be broadcasted on television isn't known yet. It is important that the discount banner will be displayed on the website together with or shortly after the television broadcast.

The goal is to:

* Get feedback on functionality that has been activated
* Decouple deployment and exposure of new functionality

**Achievement**

In this achievement you will prepare the application for the upcoming marketing campaign.

* Add code to the application that is required to support feature flags
* Test if the banner is displayed based on the feature flag setting.

**ACCEPTANCE CRITERIA**

* The discount text/banner on the homepage can be (de)activated with a feature flag
* The feature flag setting can be set through configuration

**[https://blog.launchdarkly.com/](https://blog.launchdarkly.com/ http://featureflags.io/  feature-flag-driven-development/)****[http://featureflags.io/](https://blog.launchdarkly.com/ http://featureflags.io/  feature-flag-driven-development/)**

**[feature-flag-driven-development/](https://blog.launchdarkly.com/ http://featureflags.io/  feature-flag-driven-development/)**

[**https://microsoft.github.io/PartsUnlimited/advanced/FeatureFlagWeb.html**](https://microsoft.github.io/PartsUnlimited/advanced/FeatureFlagWeb.html)

[**https://msdn.microsoft.com/en-us/library/610xe886.aspx**](https://msdn.microsoft.com/en-us/library/610xe886.aspx)

**CH010-AC002 Create a feature toggle switch page to (de)activate a feature**

The campaign was launched, but some unexpected errors occurred. Solving those errors took a lot of time and the lead-time for these changes to be solved in production environment was quit long. In the mean time errors kept occurring until someone of the IT staff, who where not available in the weekend the campaign launched and during the week where all intensively working on problem solving, had some time to change the web application configuration. For the future, the marketing and sales department of GDBC Inc. wants to be able to quickly (de)activate the functionality when required. Because of this they want an admin user to be able to (de)activate the feature flag through the administration page of the web application.

The goal is to:

* Turn off new functionality quickly in case something goes wrong and problem solving takes a long time.
* Make these changes without redeployment of the web application

**Achievement**

In this achievement you will prepare the application so that an admin user can (de)activate the campaign functionality for the upcoming marketing campaign.

* Add the option (de)activate the discount and sales functionality through the administration page replacing (de)activation through web.config
* Change the code that uses the web.config value to using the value set through the administration page
* Test if the feature is (de)activated when the setting is changed on the administration page

**ACCEPTANCE CRITERIA**

* The feature flag setting can be set through the administration page by an admin user instead of changing the application configuration

[**https://github.com/jason-roberts/FeatureToggle**](https://github.com/jason-roberts/FeatureToggle)

**CH010-AC003 Configure the release pipeline to gradually roll out the application**

This first marketing campaign was a huge success for GDBC Inc., but the journey had some hiccups.

The goal is to:

* Get feedback early
* Limit the impact if there is an error in brand new functionality or to test if customers like newly added functionality.

**Achievement**

In this achievement you will setup the release pipeline and the app service in such a way that new functionality is exposed to an incrementally growing group of users. In other words your are applying the concept of a canary release.

* Deploy the web application to a deployment slot and route different percentages of traffic to this application by using the Testing In Production (TIP) feature of an App Service.
* Implement deployment rings in your release pipeline and automatically increase the percentage of users after approval has been given for each ring
* After successfully passing your approval stages swap the deployment slot and send all traffic to it

**ACCEPTANCE CRITERIA**

* There are one or more rings in the release pipeline to which you the application is deployed after approval has been given.
* For each ring the amount of traffic that is directed to this ring increases.
* In the final ring all traffic is directed to the newly deployed application.

[**https://channel9.msdn.com/Events/Ignite/Microsoft-Ignite-Orlando-2017/THR2155**](https://channel9.msdn.com/Events/Ignite/Microsoft-Ignite-Orlando-2017/THR2155)

[**https://docs.microsoft.com/en-us/vsts/articles/phase-rollout-with-rings?view=vsts**](https://docs.microsoft.com/en-us/vsts/articles/phase-rollout-with-rings?view=vsts)

[**https://martinfowler.com/bliki/CanaryRelease.html**](https://martinfowler.com/bliki/CanaryRelease.html)

**CH011-AC002 Add Release Annotations to your pipeline to show when a release has been done**

The GDBC Inc. development team feels pretty much on top of performance and application troubleshooting since they know when performance issues occur and they have ways to diagnose and investigate them all because of collecting necessary telemetry regarding the running application components and dependencies. It feels pretty good to be proactive. Lately though some of the performance issues seem to be correlated always with releases, but no-one is quite sure. Deployments happen quite often now and its hard to know if the latest deployment could have affected the performance or not.

Both the CTO and Product Management is eager to get some clarity if the fast and frequent deployments could be the culprit and in those cases extra performance tests and quality gates / verification should be implemented to have more stable versions of software.

Application Insights has a feature called Release Annotations. These are markers in application insights regarding important milestones or changes that happened to resources (application components) that send telemetry. These markers are visible on performance and metrics dashboards thus allowing to correlate change in metric with an important change such as new version deployment.

**Achievement**

Add release annotation creation to your release pipeline so that there is a marker in application insights for every new version that gets deployed.

**ACCEPTANCE CRITERIA**

* Release annotation creation step in the pipeline
* A release has been run that has created a marker in application insights

[**https://docs.microsoft.com/en-us/azure/application-insights/app-insights-annotations**](https://docs.microsoft.com/en-us/azure/application-insights/app-insights-annotations)

[**https://martinfowler.com/bliki/CanaryRelease.html**](https://martinfowler.com/bliki/CanaryRelease.html)

[**https://docs.microsoft.com/en-us/azure/application-insights/app-insights-annotations**](https://docs.microsoft.com/en-us/azure/application-insights/app-insights-annotations)

**CH011-AC003 Set up a availability and load test for the application**

Some users report some pages that are not available on specific times. After some investigation this had various reasons. To be more proactive in mitigating problems before they are deployed to production, the Lead Architect wants to make sure that the most important parts of the application are checked on a regular bases. He also want to hit the application with some load to see if this changes behaviour.

Application Insights has a feature called "availability" that offers the functionality to check the availability of your application by running a test on those pages on a regular interval from multiple regions. These web tests can be either simple ping tests from multiple locations or multi-step tests that go through a sequence of business functionality (such as going to home page, logging in, navigating to x).

**Achievement**

Setup single step availability test called URL ping test that would ping your application and continuously monitor the availability and responsiveness. Set it up with 5 minute interval and from 4 locations that are relevant to GDBC Inc application (your choice).

**ACCEPTANCE CRITERIA**

* Single url ping test setup and running / monitoring availability and responsiveness
* Test trial application unavailability to confirm good and bad scenario visibility
* Add site availability to your azure dashboard

[**https://docs.microsoft.com/en-us/azure/application-insights/app-insights-monitor-web-app-availability**](https://docs.microsoft.com/en-us/azure/application-insights/app-insights-monitor-web-app-availability)

**CH011-AC004 Add Performance Metrics to Azure Dashboard and alert on VSTS dashboard**

GDBC Inc. development team has gathered a ton of metrics, but the information is scattered in different dashboards within application insights.The CTO would like to get a more high level and informative dashboard regarding the current and last 7 days performance metrics of the application to use these to monitor the application. This will be useful for everyone - the support, live site / operations and development team and of course the CTO that will most likely take nice snapshots out of them to illustrate and explain situations to management.

**Achievement**

Create an azure dashboard that would pull together different performance metrics that are now collected regarding the application and its components and dependencies and offer high level and detailed view over those metrics.

The dashboard should cover:

* Application components / dependencies health state
* Response times
* Availability and responsiveness information
* Users / Requests / Sessions
* Resource consumption and availability (memory, cpu, io, etc. of relevant and important resources)
* Performance counters
* Exceptions / sec rate

**ACCEPTANCE CRITERIA**

* Created a dashboard with available and relevant metrics and views on them to provide high level and detail information
* Shared the dashboard within organization by adding metrics from Application Insights Analytics to VSTS dashboard

[**https://docs.microsoft.com/en-us/azure/application-insights/app-insights-dashboards**](https://docs.microsoft.com/en-us/azure/application-insights/app-insights-dashboards)

[**https://docs.microsoft.com/en-us/azure/application-insights/app-insights-live-stream**](https://docs.microsoft.com/en-us/azure/application-insights/app-insights-live-stream)

[**https://marketplace.visualstudio.com/items?itemName=ms-appinsights.ApplicationInsightsWidgets**](https://marketplace.visualstudio.com/items?itemName=ms-appinsights.ApplicationInsightsWidgets)

**CH012-AC002 Set up a SQL Server container**

Getting the database in a correct state, with the right version of the schema and the necessary metadata is something that many teams struggle with. Containers can help out a lot here, by packaging the necessary version of the SQL server and

**Achievement**

In this challenge you will add a SQL Server container to the solution, making it easy to develop locally against the correct version of the SQL Server.

**ACCEPTANCE CRITERIA**

* When the application is started, there should be two containers running. One with the web application, the other one with SQL Server
* The application should be working, running against the database in the other container
* Successfully connect to the running instance of SQL Server inside the container using SQL Management studio

[**https://fluentbytes.com/override-classic-asp-net-web-config-configuration-settings-when-using-docker-containers/**](https://fluentbytes.com/override-classic-asp-net-web-config-configuration-settings-when-using-docker-containers/)

[**https://hub.docker.com/r/microsoft/mssql-server-windows-developer/**](https://hub.docker.com/r/microsoft/mssql-server-windows-developer/)

app: container-webapp-os08

asp: asp-py-lx (Basic: 1 Small)

rg: RG-ManualDeploy-os08

alert: availability-container

alert url”: <https://container-webapp-os08.azurewebsites.net/> (testingurl)

insight: insight-os08

url: http://insight-os08.azurewebsites.net

rg: RG-ManualDeploy-os08

<https://container-webapp-os08.azurewebsites.net>

ssh: <https://container-webapp-os08.scm.azurewebsites.net/webssh/host>

**CH012-AC003 Create Prefilled Data in SQL Server container**

Creating a database with prefilled data by running migrations scripts with data can be time consuming, especially if we want to use it as a basis for automated unit testing. Having a prebuild container image with SQL Server and a database with all necessary data available for developers and testers can improve the overall delivery process.

**Achievement**

In this challenge you will use the running SQL container from the previous exercise as the basis for a new image. In the next challenge, you will push this image to ACR, together with the application image. You will also change the application so that is does not create a database when starting up.

**ACCEPTANCE CRITERIA**

* A container image with the MusicStore database
* MvcMusic store application should use this image instead of the default SQL Server image

[**https://docs.docker.com/engine/reference/commandline/commit/**](https://docs.docker.com/engine/reference/commandline/commit/)

**CH012-AC004 Set up Azure Container Registry**

When working with Docker containers, the artefacts that are produced are Docker images, that contains the application as well as everything needed to run that application, including the base OS. These images need to be published to a container registry, from which they can be pulled from different target environments and started as containers.

**Achievement**

In this challenge you will create an Azure Container Registry that you can use later on to publish the images of the application

**ACCEPTANCE CRITERIA**

* A generated Azure Container Registry visible in the Azure portal

**https://docs.microsoft.com/en-us/azure/container-registry/container-registry-get-started-portal**

registry: containerregos08

rg: RG-ManualDeploy-os08

login\_server: containerregos08.azurecr.io

Username : containerregos08

Pswd: Z0/VkBHWYoX12g=4hkvLJc4chnDywKtJ

Pswd2: qUgkU4wbQ=+805DLYWzw0SQen29SXn27

**CH012-AC005 Publish "development" Container to ACR from pipeline**

Even if we are not running the application using containers in production, they can still provide a lot of value for the development workflow. We can for example publish development images of the application that developers can pull and run on their local machine, without having to compile and configure them.

**Achievement**

In this challenge you publish container images that you have built locally to the Azure Container Registry that you created previously

**ACCEPTANCE CRITERIA**

* At least one container image that has been pushed to the container registry

[**https://docs.microsoft.com/en-us/azure/container-registry/container-registry-get-started-docker-cli**](https://docs.microsoft.com/en-us/azure/container-registry/container-registry-get-started-docker-cli)