# Practice M5: Azure and DevOps

For this module we will need a PC running recent version of **Windows**, **Linux**, or **macOS**. It can be either physical or virtual machine

For this practice you will need VS Code with the following extensions – Azure Tools, Azure CLI Tools, etc.

Additionally, you will need access to a Docker instance, local docker and git clients installed

## Part 1: Azure Kubernetes Service

### Azure Portal: Cluster

Navigate to <https://portal.azure.com>

Enter your credentials

#### Resource group

Search for **Resource groups** in the main search bar and hit **Enter**

Click on **+ Add**

Check the **Subscription**

For **Resource group** enter **RG-Kubernetes**

Select **West Europe** for **Region**

Click on **Review + create** and then on **Create**

#### Container registry

Search for **Container registries** in the main search bar and hit **Enter**

Click on **+ Add**

For **Registry name** enter **azesu**

Check the **Subscription**

For **Resource group** select **RG-Kubernetes** created earlier

Adjust the **Location**

Change **SKU** to **Basic**

Click **Review + create**

Click **Create**

#### Kubernetes cluster

Search for **Kubernetes services** in the main search bar and hit **Enter**

Click on **+ Add** and select **Add Kubernetes cluster**

Check that the correct **Subscription** is selected

For the **Resource group** select the **RG-Kubernetes** created earlier

In the **Kubernetes cluster name** field enter **aze-kubernetes**

Change **Region** to **West Europe**

Change **Node size** to **Standard B2s**

Set **Node count** to **1**

Click on **Next: Node pools**

Click on **Next: Authentication**

Click on **Next: Networking**

Click on **Next: Integrations**

Select the registry created earlier in the **Container registry** drop-down list

Click on **Create new** under the **Log Analytics workspace**

Leave the **Region** to **West Europe**

For **Log Analytics workspace** enter **LA-Kubernetes**

Click **Create**

Click **Review + create** and then **Create**

Once the deployment is ready, click on the **Go to resource** button

### Azure CLI: Cluster

Similar result can be achieved with the help of **Azure CLI**

Firs, we need to login

**az login**

Then create a resource group:

**az group create -n RG-K8S -l westeurope**

Next, create a container registry:

**az acr create -g RG-K8S -n azesucli --sku Basic**

Finally, we can create the cluster:

**az aks create -g RG-K8S -n k8s-demo --node-count 1 --node-vm-size Standard\_B2s --enable-addons monitoring --generate-ssh-keys --attach-acr azesucli**

### Command-line: Explore the cluster

If not already logged in, execute this command first:

**az login**

In order to get and store the credentials needed for communication with the cluster, you must execute:

**az aks get-credentials --resource-group RG-Kubernetes --name aze-kubernetes**

Now, you can use the **kubectl** tool (you should download it first from here: <https://kubernetes.io/docs/tasks/tools/>) to manage the cluster. For example, get information about the cluster and nodes:

**kubectl cluster-info**

**kubectl get nodes**

List all namespaces and all pods in every namespace:

**kubectl get namespaces**

**kubectl get pods --all-namespaces**

### Portal: Explore the cluster

Your first stop is the **Overview**

Then, you can visit the **Node pools**

Next stop is the **Insights**, **Metrics**, and **Logs** sections

### Command-line: Run an application

Extract the file **web-app-php.zip**

Build the image:

**docker build . -t aze-web-app-php**

Test the app locally:

**docker run -d --name webapp -p 8000:80 aze-web-app-php**

Open a browser and navigate to <http://localhost:8000>

After we are sure that everything is working as expected, we can continue further

Login to the ACR

**az acr login --name azesu**

Check the login server:

**az acr list --resource-group RG-Kubernetes --query "[].{acrLoginServer:loginServer}" --output table**

Tag the image:

**docker tag aze-web-app-php azesu.azurecr.io/aze-web-app-php:v1**

Push the image to our ACR:

**docker push azesu.azurecr.io/aze-web-app-php:v1**

Check the list of images available on our ACR:

**az acr repository list --name azesu --output table**

And all tags for an image:

**az acr repository show-tags --name azesu --repository aze-web-app-php --output table**

Integrate our existing ACR with our existing AKS cluster (if not done):

**az aks update -n aze-kubernetes -g RG-Kubernetes --attach-acr azesu**

Enter the **manifests** folder

Deploy both the service and application simultaneously:

**kubectl apply -f service.yaml -f deployment.yaml**

We can check periodically how it is going:

**kubectl get svc,pod**

Get the load balancer IP address and paste it into a new browser window to check if the application is working

Should we want to delete the application (do not do it for now), we can use:

**kubectl delete -f service.yaml -f deployment.yaml**

### Scale an application

Let’s first check what do we have:

**kubectl get pods**

Get detailed information about the deployment:

**kubectl describe deployment phpapp-deployment**

Scale up to 5 replicas:

**kubectl scale --replicas=5 deployment.apps/phpapp-deployment**

Check what is going on with the pods:

**kubectl get pods**

Scale once again, this time to 10 replicas:

**kubectl scale --replicas=10 deployment.apps/phpapp-deployment**

Check that the new pods are created:

**kubectl get pods**

Now, we can go to the app opened in the browser and refresh a few times to see that it is served by different pods

Let’s scale down a bit:

**kubectl scale --replicas=2 deployment.apps/phpapp-deployment**

And check what is going on with the pods:

**kubectl get pods**

### Portal: Scale the cluster

Go to **Node pools** under **Settings**

Select the pool and from the context menu select **Scale**

Change **Node count** to **2**

Click **Apply**

After a while you will notice that the number of nodes has changed to two

You can ask for nodes’ status on the command line with:

**kubectl get nodes**

If we scale now to 5 replicas:

**kubectl scale --replicas=5 deployment.apps/phpapp-deployment**

We will notice that some of the pods are on the first node, and others on the second:

**kubectl get pods -o wide**

### Command-line: Scale the cluster

We can add or remove nodes on the command line as well

Let’s add one more node:

**az aks scale --resource-group RG-Kubernetes --name aze-kubernetes --node-count 3 --nodepool-name agentpool**

After a while, a new node will appear:

**kubectl get nodes -o wide**

### Update and redeploy the application

Let’s modify our application

Remember to stop it first if it is still running. This can be done with

**docker container rm webapp --force**

For example, change the title to **Top 10 cities in Bulgaria**, change the **H3** tag to **H2**, and add a border to the table

Save the file

Build the new image:

**docker build . -t aze-web-app-php**

Test the app locally:

**docker run -d --name webapp -p 8000:80 aze-web-app-php**

Open a browser and navigate to[**http://localhost:8000**](http://localhost:8000)

After we are sure that everything is working as expected, we can continue further

Tag the image:

**docker tag aze-web-app-php azesu.azurecr.io/aze-web-app-php:v2**

Push the image to our **ACR**:

**docker push azesu.azurecr.io/aze-web-app-php:v2**

Check the list of images available on our ACR:

**az acr repository list --name azesu --output table**

And all tags for an image:

**az acr repository show-tags --name azesu --repository aze-web-app-php --output table**

Change the image version in the **deployment.yaml** file as well

Deploy both the service and application simultaneously:

**kubectl apply -f service.yaml -f deployment.yaml**

We can check periodically how it is going:

**kubectl get svc,pod**

We will notice that extra replicas will be terminated and once, the new pod is up and running, the last of the old version will be terminated

We can go to the browser and check the result

## Part 2: Azure DevOps Boards

You can reach the **Azure DevOps** services in two ways

#### Option 1: Azure Portal

Navigate to[**https://portal.azure.com**](https://portal.azure.com)

Enter your credentials

Search for **Azure DevOps** in the main search bar and hit **Enter**

Then click on **My Azure DevOps Organizations**

#### Option 2: Azure DevOps portal

Navigate to [**https://dev.azure.com**](https://dev.azure.com)

Enter your credentials

### Create new organization

Once you’re in, click on the **Create new organization** button

Click on **Continue**

Enter name for the organization, for example **azedo202104**

Then select a region, for example **West Europe**

Enter the code

Click on the **Continue** button

### Set up a project

In the **Project name** field enter **Demo1** for example

You can leave the **Visibility** setting to **Private**

Click on **Create project**

Our brand-new empty project with **Basic process** is set up

### Project settings

Click on **Project settings** (bottom-left)

Here, we may change many things

Let’s go to **General / Overview** and add **Description**. For example, **This is our first Azure DevOps project**

Click on **Save**

Then, go to **Boards / Project configuration**

Select **Demo 1** and click on **Set dates**

Select **today** as **Start date** and **31.05.2021** as **End date**

Click on **Save and close**

Select **Sprint 1** and click on **Set dates**

Select **today** as **Start date** and **30.04.2021** as **End date**

Click on **Save and close**

Click on **New**

For **Iteration name** enter **Sprint 2**

Select **01.05.2021** as **Start date** and **31.05.2021** as **End date**

Click on **Save and close**

We can manage **Teams** from here as well

Go to **General / Teams**

Click on the **Demo 1 Team**

Now, you can see the members

Click on the **Add** button to add some more

Once you are done, click **Save**

Click on the project’s name **Demo 1** in the navigation bar

### Manage work items

We will add a few work items on different levels – epics, issues, and tasks

Navigate to **Boards / Work items**

Let’s start with the epics

Click on **New Work Item** and select **Epic**

Fortitle enter **User interface**

For the **Iteration** select **Demo 1\Sprint 1**

Enter a **Description.** For example, **Align the interface with the specification**

Set the **Priority** to **3**

Click on **Save** and then **Back to Work Items**

Click on **New Work Item** and select **Epic**

Fortitle enter **Data presentation**

For the **Iteration** select **Demo 1\Sprint 1**

Enter a **Description.** For example, **Nice formatted table**

Set the **Priority** to **1**

Click on **Save** and then **Back to Work Items**

Now, let’s add a few **Issues** to both **Epics** but we will use another place to do this

Navigate to **Boards / Backlogs**

Change the view from **Issues** to **Epics**

Click on the **+** sign in front of the first epic

Enter **Title style** and click on **Save & Close**

Click on the **+** sign in front of the first epic

Enter **Data table** and click on **Save & Close**

Click on the **+** sign in front of the second epic

Enter **Row numbering** and click on **Save & Close**

Finally, let’s add a few **Tasks** to each **Issue**

Click on the **+** sign in front of the **Title style** issue

Enter **Element type H1**, for **Remaining work** enter **1**, and click on **Save & Close**

Click on the **+** sign in front of the **Title style** issue

Enter **Underline**, for **Remaining work** enter **1**, and click on **Save & Close**

Click on the **+** sign in front of the **Data table** issue

Enter **Show border**, for **Remaining work** enter **1**, and click on **Save & Close**

Click on the **+** sign in front of the **Row numbering** issue

Enter **Each row numbered from 1 to 10**, for **Remaining work** enter **2**, and click on **Save & Close**

Click on the **+** sign in front of the **Row numbering** issue

Enter **Order from biggest to smallest**, for **Remaining work** enter **2**, and click on **Save & Close**

### Working with the items

Switch to **Boards / Sprints**

Check the four different views – **Taskboard**, **Backlog**, **Capacity**, and **Analytics**

Explore **Boards** and **Backlogs** modes as well

Go to **Boards** and switch the view between **Epics** and **Issues**

Make sure that your are in **Boards** and the view is set to **Issues**

Drag the **Title style** issue from **To Do** to **Doing**

Check how the statuses are reflected in different views

### Azure DevOps Repos

Let’s go to **Repos**

There we have an empty repository

#### Prepare a repository

Click on the **Initialize** button at the end of the page to initialize the repository with a read me file

Select the **README.md** file if not selected and click on **Contents**

Click on **Edit** and delete everything

Enter some descriptive text

For example, on the first line enter **# Top cities in Bulgaria**

And on the second – **Simple web application that shows top 10 cities by population in Bulgaria**

Hit **Commit** and then again **Commit**

#### Upload files

Extract the **web-app-html.zip** archive to a folder

Return to the browser

Click on the three dots next to the repository’s name and select **Clone**

In the **IDE** drop-down list select **Clone in VS Code**

Confirm in the pop-up window that you want to open the link with **VS Code**

When asked, click on **Open**

Navigate to a folder and click on **Select Repository Location**

If prompted for credentials, enter them

Once all is set up, you will be asked if you want to open the folder in **VS Code**, confirm with **Open**

Now, copy all extracted files into this new folder

Next steps can be done either visually or on the command line. Let’s go with the command-line approach

Open either a separate terminal and login to Azure or use the integrated one – **View / Terminal**

First, we must stage the changes (plus the new files):

**git add .\Dockerfile .\web\index.html**

**git status**

Next, we can commit the changes:

**git commit -m "Initial upload"**

And finally, push them:

**git push**

Return to the browser, you must see the changes there as well

### Azure DevOps Pipelines

Click on the **Demo 1** repository

Then click on **Set up build**

Click on **Show more** to explore the available options

Click on **Deploy to Azure Kubernetes Service**

Select your subscription and click on **Continue**

Enter your Microsoft account credentials

In the **Cluster** drop-down select the cluster created earlier

For **Namespace** select **New** and enter **devops-pipeline**

In the **Container registry** drop-down select the registry created earlier

For **Image Name** enter **webapp**

For **Service Port** enter **80**

You may select the **Enable Review App flow for Pull Requests** option if you like so

For this exercise we do not want to review pull requests, so let’s leave it unselected

Click **Validate and configure**

Click **Save and run**

Accept the proposed defaults for the commit operation and click **Save and run**

Click on **Build stage** and watch the progress

Then select **Deploy** to monitor the deployment progress

Once the pipeline finishes successfully, click on the **Deploy to Kubernetes cluster** step and scroll to the bottom

Close to the end you will notice a string like **service webapp external IP is** xxx.xxx.xxx.xxx

Copy the IP address, open a new browser tab and paste it there

You must see the web application

There is plenty to be done here. Some of the pending tasks are on **Azure DevOps Boards**. Let’s get to work

### Modify the application #1

Return to **Boards** and move the **Title style** issue from **To Do** to **Doing** if not there already

Open the item in editing mode

Click on **create a branch** under the **Add link** in the **Development** section

In **Name** enter **branch-title** and click on **Create branch**

Switch to the **VS Code** to take care for the task

Be sure that the local copy of the repo is aligned with the remote one:

**git pull**

**git switch branch-title**

Open the **index.html** file for editing

Change the **<h3>** tags to **<h1>**

Add a pair of **<u>** and **</u>** tags around the title

Save the file

Test locally that the result is as expected

Publish the changes:

**git add .\web\index.html**

**git commit -m "Title corrected"**

**git push**

Return to **Repos**

You will see a message that the **branch-title** has been updated

Next to it there will be a **Create a pull request** button

Click on it

Examine the information on the next screen

Click on **Create**

Explore the overview of the pull request and click on **Approve**

And finally, on **Complete**

Click on **Complete merge**

Switch to **Pipelines**

Click on **Demo 1**

Click on the running pipeline

Click on the stages to monitor the progress

Now, if you visit the app, you should see the changes

Switch to **Boards / Boards**

The item that was under **Doing** now should in **Done** state

Open the tasks and mark them as done as well

Now, switch to **Boards / Sprints**

Open the **Title style** and check if the tasks are marked as done

### Query the Boards

Go to **Boards / Query**

Click on **New query**

Change **Value** of **Work Item Type** to **Issue**

Change **Value** of **State** to **Done**

Click on **Run query**

Then, click on **Save query**

For **Name** enter **Done Items** and click **OK**

Based on the query results, we can create charts

Let’s continue with the modifications

### Modify the application #2

Being in **Boards /** **Boards**, move both the **Data table** and **Row numbering** issues from **To Do** to **Doing**

Open the **Data table** issue in editing mode

Click on **create a branch** under the **Add link** in the **Development** section

In **Name** enter **branch-data**, select **Demo 1** in **Based on** if not selected, and click on **Create branch**

Return to **Boards** and click on the **Row numbering** issue

Click on **Add link**

Select **Branch** in **Link type**

Select **branch-data** in **Branch**

Click **OK**

Click **Save & Close**

Switch to **VS Code** to take care for the tasks

Be sure that the local copy of the repo is aligned with the remote one:

**git pull**

**git switch branch-data**

Open the **index.html** file for editing

Add **border="1"** in the **table** tag

Rearrange the rows based on their population column

Add additional column with numbers from 1 to 10

Save the file

Test locally that the result is as expected

Publish the changes:

**git add .\web\index.html**

**git commit -m "Table and data corrected"**

**git push**

Return to **Repos**

You will see a message that the **branch-data** has been updated

Next to it there will be a **Create a pull request** button

Click on it

Examine the information on the next screen

Click on **Create**

Examine the information, skip the **Approve**, and click on **Complete** directly

Click on **Complete merge**

Switch to **Pipelines**

Click on **Demo 1**

Click on the running pipeline

Click on the stages to monitor the progress

Now, if you visit the app, you should see the changes

Switch to **Boards / Boards**

Both issues now should in **Done** state

Open the tasks and mark them as done as well

Now, switch to **Boards / Sprints**

Open each of the issues and check if the tasks are marked as done

### Final touches

Let’s modify the **trigger** section of our pipeline

We want it not to be triggered by changes committed to the **README.md** file for example

Go to **Pipelines / Pipelines**

Click on **Demo1**

Click **Edit**

Change the **trigger** section to be like this:

**trigger:**

**branches:**

**include:**

**- master**

**paths:**

**exclude:**

**- README.md**

Click on **Save** and then again on **Save**

Now a build will be triggered

Meanwhile, we can continue our work

Let’s add two badges to our **README.md** file

The first badge will reflect our pipeline status

Navigate to the pipeline

Click on the button with three vertical dots

Click on **Status badge**

Click on the **Copy** button next to the **Sample markdown** field

Close the dialog box

Go to **Repos / Files**

Click on the **README.md** file

Click **Edit** and paste the copied string at the end of the file

Click **Commit** and for **Comment** add **Added pipeline badge**

Click **Commit**

Now, we must see the badge

Let’s add one more badge but this time it will reflect the status of the work items

Go to **Boards / Boards**

Click on **Settings**

Click on **Status badge** under **Board**

Click on the **Copy** button next to the **Sample Markdown** field (Please note, that this will reflect the current board mode – what is selected **Epics** or **Issues**)

Close the dialog box

Go to **Repos / Files**

Click on the **README.md** file

Click **Edit** and paste the copied string at the end of the file

Click **Commit** and for **Comment** add **Added boards badge**

Click **Commit**

Now, we must see the second badge as well

## Part 3: DevOps Projects (DevOps Starter)

**NOTE:** *For this task you must either fork the* ***shekeriev/aze-web-app-php*** *GitHub repository or create your own either from the* ***web-app-php.zip*** *archive or from other source*

Navigate to <https://portal.azure.com>

Enter your credentials

### Create a project

In the main search bar enter **DevOps Starter** (used to be **DevOps Projects**)and hit the **Enter** key

Click on **Create DevOps starter**

Select **PHP**

Click **Next: Framework >**

Select **Simple PHP**

Click **Next: Service >**

Select **Web App for Containers**

Click **Next: Create >**

Click the **Authorize** button

Click **Authorize** **AzureGithubActions**

Enter a **Repository** name, for example **aze-demo**

Select the right **Subscription**

Select the right **Location**

Click **Additional settings** link

Change **Container Registry SKU** to **Basic**

Change **Container Registry Location** to **West Europe**

Click **OK**

Click **Review + create**

Click **Create**

Once the resource is ready, click on **Go to resource**

Click the **Authorize** button

Click the **Authorize** button again

Click the link to the **GitHub** repositoryto explore the files

Return to the portal and watch the progress

Click on the stages to see their progress

Periodically, click on the **Refresh** button if you think that the status if not updated

You can click on different stages and jobs to explore and watch the progress

Once the whole processes is done, explore the **Azure Resources** area

Click on the **Browse** button next to the **Application endpoint**

### Make a change

Go to the repository and make a change directly in the master branch

The pipeline should be triggered just after the commit

Monitor its execution

Once finished, check the end result

## A Reminder (to clean up)

Don’t forget to clean up the resources

You can delete those resource groups that you are not going to need

For Azure DevOps, one of the most important tasks is to either disable all pipelines that you do not need or delete them

### Disable a Pipeline

In order to disable a pipeline, go to **Pipelines / Pipelines**

Click on the button with the three vertical dots next to the pipeline

Select the **Edit** option

Switch to **Options**

Select **Disabled** option

Select **Save** in the **Save & queue** button drop-down list

### Delete a Pipeline

To delete a pipeline, go to **Pipelines / Pipelines**

Click on the button with the three vertical dots next to the pipeline

Click on **Delete**

Type the name of the pipeline and click **Delete**

Please note that you may need to delete any releases upfront

### Delete a DevOps Project

Use the **Delete** functionality on the **DevOps Projects** dashboard

### Delete an Azure Kubernetes Service

Navigate to the resource group where you created the AKS resource and click on it

Click on the **Delete** button and confirm

This will delete all related resources

After the process is finished, you must take care to delete the ACR (if you created one) and the resource group

### Leftovers

Check for any leftovers like empty resource groups

Of course, they won’t cost you money, but they will clutter your workspace