**Rapid Docs**

v1.0.0

This document provide information about RapidDocs app deployment, cloud resources, database, configuration files and others.

1. **Prerequisites** 
   1. Docker for Windows
   2. Azure CLI 2.0
2. **Cloud Resources**
   1. Resource Group
      1. Create a Resource Group <RapidDocsRG>
   2. Storage Account
      1. Create a Storage Account resource <RapidDocsSA> in the <RapidDocsRG> resource group
      2. In this storage account create two queues :
         1. <pdf-queue-items> queue
         2. <pdf-poison-queue-items> queue
   3. Service Bus
      1. Create a Service Bus resource <RapidDocsSB> in the <RapidDocsRG> resource group
         1. When you create this resource, the pricing tier must be at least “Standard”, otherwise you cannot create topics.
      2. In this resource, create two topics :
         1. <text-document-items> topic
         2. <text-training-document-items> topic
   4. Container Registry
      1. Create a container registry resource <RapidDocsACR> in the <RapidDocsRG> resource group
3. **Database**
   1. Create database <RapidDocsDB>
      1. The database should be empty. Tables will be generated when the application will start (using migration from Entity Framework Core)
      2. Add an application user, you’ll need this later in the appSettings.json file.
4. **Other Web Applications**
   1. StorageProviderAPI
      1. Before publishing, edit Web.config – you need to edit <StorageProvider> section

<StorageProvider … defaultProvider="DatabaseStorage" storageInfoConnectionString="<DB\_Conn>">

<providers>

…

<add name="DatabaseStorage" … connectionString="<DB\_Conn>" />

…

</providers>

</StorageProvider>

* + 1. The database that we received is called RapidAdvanceStorage

1. **Azure Func App** 
   1. The Func app will be published from Cloud/RapidDocs.AzureFunctions project
   2. Before you publish the app, edit local.settings.json file

{

"IsEncrypted": false,

"Values": {

"AzureWebJobsStorage":<ConnectionString from <RapidDocsSA> storage account>,

"AzureWebJobsDashboard":<ConnectionString from <RapidDocsSA> storage account>,

"ServiceBusConnection": <ConnectionString from <RapidDocsSB> service bus>,

"StorageProviderConnection": <Url to StorageProviderAPI>,

"DocumentsTopic": "text-document-items",

"TrainingDocumentsTopic": "text-training-document-items"

}

}

* 1. Publish the project to Azure using:
     1. <RapidDocsRG> resource group
     2. <RapidDocsSA> storage account

d. After publishing the Azure Function project, make sure that all the settings from the local.settings.json file exists in the Azure Application Settings. To do this, browse to the Azure Portal -> Function Apps -> Click on the newly created function and navigate to Application Settings sections. If there are missing settings, add them as in the json file (with the same name and value).

1. **Rapid Docs API & Silo host**
   1. Before publishing the app, you need to provide values for appSettings.json for both RapidDocs.Api and RapidDocs.SiloHost projects:
      1. ConnectionStrings

"ConnectionStrings": {

"DefaultConnection": "<Connection string to <RapidDocsDB> >",

"AzureServiceBusConnection":<Connection string to <RapidDocsSB> >,

"AzureWebJobsStorage":<Connection string to <RapidDocsSA> >

}

* + 1. AppSettings

"AppSettings": {

"CommonSettings": {

"ApiName": "RapidDocs.API",

},

"AuthSettings": {

"SystemApiKey": "<System api key>",

"TestApiKey": "<Test api key>"

},

"AzureWebJobsStorage": {

"PdfInputQueue": "pdf-queue-items",

"PdfPoisonQueue": "pdf-poison-queue-items",

"CountDequeueMessages": 10,

"CountDequeueTries": 1

},

"AzureServiceBus": {

"TenantId": "<Azure Tenant Id>",

"ClientId": "<Client Id from Azure AD>",

"ClientSecret": "<Client secret from Azure AD>",

"SubscriptionId": "<Azure subscription Id>",

"ResourceGroup": "<RapidDocsRG>",

"Namespace": "<RapidDocsSB>",

"DataCenterLocation": "<RapidDocsSB> Location (e.g. Central US)",

"ServiceBusSku": "Standard",

"DocumentsTopic": "text-document-items",

"TrainingDocumentsTopic": "text-training-document-items",

"ValidItemsSubscription": "valid-items",

"InvalidItemsSubscription": "invalid-items"

},

"ProvidersUrls": {

"PdfProcessorFuncApp":"<Azure Function App url>",

"StorageProvider": "<Storage Provider url>"

},

"OrleansSettings": {

"ClusterId": "RapidDocsOrleansCluster",

"ServiceId": "RapidDocsOrleansService",

"SilePort": 11111,

"GatewayPort": 30000

}

}

For the AzureServiceBus settings, you can find information in [this](https://docs.bmc.com/docs/cloudlifecyclemanagement/46/setting-up-a-tenant-id-client-id-and-client-secret-for-azure-resource-manager-provisioning-669202145.html) article.

* 1. Create Docker containers:
     1. In the root folder of the solution you can find a docker-compose.yml file
     2. Create a Docker image. You can do this from Visual Studio or with CMD:

docker-compose build

* + 1. To see a list of your current local images, use the “docker images” command. There should be two images :
       1. rapiddocs.api
       2. rapiddocs.silohost
  1. Tag container images
     1. Connect to the container registry

az acr login --name <RapidDocsACR>

* + 1. To get the login server address, use the [az acr list](https://docs.microsoft.com/cli/azure/acr" \l "az-acr-list) command and query for the loginServer as follows

az acr list --resource-group <RapidDocsRG> --query "[].{acrLoginServer:loginServer}" --output table

* + 1. Tag both images

docker tag rapiddocs.api <acrLoginServer>/rapiddocs.api:v1

docker tag rapiddocs.silohost <acrLoginServer>/rapiddocs.silohost:v1

* + 1. To verify the tags are applied, run “docker images” again
  1. Push images to registry

docker push <acrLoginServer>/rapiddocs.api:v1

docker push <acrLoginServer>/rapiddocs.silohost:v1

* 1. List images in registry

az acr repository list --name <RapidDocsACR> --output table

1. **Create Kubernetes cluster**
   1. Create a service principal

az ad sp create-for-rbac --skip-assignment

The output is similar to the following example:

{

"appId": "e7596ae3-6864-4cb8-94fc-20164b1588a9",

"displayName": "azure-cli-2018-06-29-19-14-37",

"name": "http://azure-cli-2018-06-29-19-14-37",

"password": "52c95f25-bd1e-4314-bd31-d8112b293521",

"tenant": "72f988bf-86f1-41af-91ab-2d7cd011db48"

}

* 1. Configure ACR authentication
     1. Get ACR id

az acr show --resource-group <RapidDocsRG> --name <RapidDocsACR> --query "id" --output tsv

* + 1. Grant Access

az role assignment create --assignee <appId> --scope <acrId> --role Reader

* 1. Create a Kubernetes cluster

az aks create \

--resource-group <RapidDocsRG>\

--name <RapidDocsAKSCluster> \

--node-count 1 \

--service-principal <appId> \

--client-secret <password> \

--generate-ssh-keys

* 1. Create Secret

az aks get-credentials --resource-group <RapidDocsRG> --name <RapidDocsAKSCluster>

kubectl create secret docker-registry acr-auth --docker-server <acr-login-server> --docker-username <acr-login-server-username> --docker-password <acr-server-login-password> --docker-email <any-email-address>

* 1. Deploy the applications

kubectl apply -f rapiddocs-aspcore.yaml

* 1. Test applications

kubectl get service rapiddocs-api –watch

kubectl get service rapiddocs-silohost –watch

az aks browse --resource-group <RapidDocsRG> --name <RapidDocsAKSCluster>