Kofax RPA Serverless Computing with Microsoft Azure

Thursday 7th May 2020.

Version 0.2.

Ash Grant, Sales Engineer, Kofax.
## Contents

- Kofax RPA Docker Deployment .......................................................... 3
- Docker Registry ................................................................................ 3
- Docker Tools .................................................................................... 4
- To Tag or Not to Tag ........................................................................ 4
- Push It .................................................................................................. 5
- Create your Container Instance .......................................................... 5
  - Environmental Variables ................................................................. 6
- Job Done ............................................................................................ 7
Organisations around the globe are under pressure to deliver operational efficiencies and reduce costs. The rising cost of management and maintenance has encouraged organisations to evaluate solutions that will deliver a lower total cost of ownership, whilst ensuring rapid deployment. In addition, they want the flexibility to select their platform of choice without being locked in with a specific provider.

This article will walk through how Kofax enables organisations to achieve their goals by using efficient technologies such as Docker to achieve this within Kofax RPA (robotic process automation).

Several cloud providers enable customers to take advantage of serverless computing possibilities. That is, they provide an environment that dynamically manages the allocation of machine resources and removes the need to administer an operating system. Pricing is typically based upon the actual amount of resources consumed by the application, rather than on pre-purchased and pre-allocated units of capacity.

Serverless computing provides several key benefits to Kofax customers, including a reduction in the effort/cost of ownership, elasticity and productivity. The ability to rapidly spin-up new instances and release unused resources provides a significant benefit to ensure that costs are only incurred when there is demand. Cost-effective computing allows you to realise extended savings in addition to those provided through the implementation of Kofax RPA. It’s a win-win.

Kofax RPA is a cloud-neutral solution. With the Docker approach, you can easily switch between your preferred choice of platform provider, such as Microsoft Azure, Amazon Web Services and Google Cloud. Avoiding ‘cloud lock’ provides you with the ability to easily migrate between services as and when your business landscape or requirements change.

This article walks through the steps and best practices for deploying Kofax RPA to run in Microsoft Azure’s Docker service, with the approach being equally applicable to services like AWS’s Elastic Container Service or if you are using Docker in a Windows or Linux environment. Using Docker to deploy RPA allows you to use your cloud of choice, supporting multi-cloud or cloud neutrality strategies.

Docker Registry

When we create images, logically we need somewhere to store and manage them effectively, complete with versioning capabilities. Step one is the Azure Container Registry (or “ACR,” if you’re familiar with the abbreviation).

Of course, you’re more than welcome to use another private or public repository like DockerHub if that strikes your fancy.

Within our Azure console, let’s create a new ACR resource (you’ll find that within the “Containers” category).
Creating an ACR is easy. There’s nothing more to do here.
Alternatively, the same task can be performed using the Azure Command Line Interface (or Azure ‘CLI’). The Azure CLI can be installed from Microsoft and used to deploy docker images into Azure from a Powershell interface.

Docker Tools

Kofax RPA is shipped with both Windows- and Linux-based docker tools (since most cloud providers support both, you’ll have the best of both worlds!).

The first thing we need to do is to build our images. Kofax RPA provides docker-compose examples, providing you with a quick and easy method to create your first image, with all the necessary pre-requisites.

PowerShell is our friend here.

Navigate to your default Kofax RPA installation directory, and run the `docker build` command for the service(s) you wish to create. **Tip:** Replace `<ServiceNameHere>` with the name of the service you are creating.

Building images takes time. Why? Because we’re downloading a complete operating base from Microsoft, and we’re compiling all the Kofax RPA dependencies into a single docker image.

So, grab a coffee…

To Tag or Not to Tag

I’ll give you a tip – you must tag! It helps with versioning and identifying exactly what you’re pushing to your container registry.

List your new images using the `docker image ls` command:
Each image will be assigned a unique ID. Tag each image ID like this:

```bash
PS C:\Program Files\Kofax RPA 11.0.0.0 173 x64\ docker tag <ImageIDHere> kofaxrpa.azurecr.io/<ServiceNameHere>:v1
```

**Tip:** Replace `kofaxrpa.azurecr.io` with the name you chose when you created your ACR. Make good use of version tags too, such as in my example ‘:v1’.

Now, we’ll be able to tell which image is which within our repository.

**Push It**

Now we’re ready to push our images to the ACR. Use the `docker push` command.

Here’s an example command I used to upload my Management Console service.

```bash
PS C:\Program Files\Kofax RPA 11.0.0.0 173 x64\ docker push kofaxrpa.azurecr.io/managementconsole:v1
```

Time to grab another coffee…

Here is a screenshot of our first image uploaded to the ACR.

![Screenshot of first image uploaded to ACR](image)

**Create your Container Instance**

Now we can reference our uploaded image in creating our Azure container instance. You’ll find that within the Azure portal - **All Resources > Containers > Container Instance**.
Select your Image Source. That’s the repository where your docker image resides. If you used another repository, be sure to select it here.

We’ll go ahead with the ACR, and from there we can now select our newly uploaded Management Console image.

Next is Networking. Here, we specify the ports we wish to allow communication to/from. You’ll need to configure this to allow communication between different RPA services, such as that between the Management Console and Roboservers.

Finally…Advanced.

**Environmental Variables**

Kofax RPA services accept several variables that can be passed in to configure the service. This best-practice approach gives you the flexibility to quickly deploy ‘vanilla’ images that can be tokenized at runtime by providing input parameters. The input parameters will ultimately define the operation of your image, including database connections, credentials and licensing information. Refer to the Kofax RPA product documentation for a full list of variables.

Azure provides an easy method to provide input parameters on the Advanced tab:
Here, I’m providing my license details, which will allow the Management Console to start with my license applied.

On that topic, you’ll also need to configure your connection to your database in the same manner:

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_LICENSE_NAME</td>
<td>Sales Engineer</td>
</tr>
<tr>
<td>CONFIG_LICENSE_EMAIL</td>
<td><a href="mailto:Sales.Engineer@Kofax.com">Sales.Engineer@Kofax.com</a></td>
</tr>
<tr>
<td>CONFIG_LICENSE_COMPANY</td>
<td>Kofax</td>
</tr>
<tr>
<td>CONFIG_LICENSE_PRODUCTIONKEY</td>
<td>ProductionKeyHere</td>
</tr>
<tr>
<td>CONFIG_LICENSE_NONPRODUCTIONKEY</td>
<td>NonProductionKeyHere</td>
</tr>
</tbody>
</table>

You’ll need to specify the JDBC connection string to the database. In accordance with serverless computing, I’m using a Microsoft Azure SQL database service.

We’re all done. Let Azure create and deploy your new container instance.

Job Done

Once your container instance is deployed, access the resource within the Azure portal to find your URL/IP address:
Access the URL/IP address within your browser of choice, and you should now be looking at your Kofax RPA Management Console login screen.

**Tip:** Tap the required port number onto the end of your URL/IP address; the default is 8080.

Now that you’re familiar with creating, pushing and running Kofax RPA services as containers, you can go back and create the remaining services (Kapplets, File System and Roboserver).

Thanks for reading!