# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>3</td>
</tr>
<tr>
<td>Pre-Installation Information</td>
<td>4</td>
</tr>
<tr>
<td>Before you begin</td>
<td>4</td>
</tr>
<tr>
<td>Kelverion Licensing</td>
<td>5</td>
</tr>
<tr>
<td>Kelverion Runbook Studio License</td>
<td>5</td>
</tr>
<tr>
<td>Kelverion Integration Module Licenses</td>
<td>6</td>
</tr>
<tr>
<td>Kelverion Runbook Studio Layout</td>
<td>7</td>
</tr>
<tr>
<td>The Toolbar</td>
<td>7</td>
</tr>
<tr>
<td>File menu</td>
<td>7</td>
</tr>
<tr>
<td>The Resources Pane</td>
<td>7</td>
</tr>
<tr>
<td>The Runbook Canvas</td>
<td>8</td>
</tr>
<tr>
<td>Runbook Properties</td>
<td>8</td>
</tr>
<tr>
<td>Managing Assets</td>
<td>9</td>
</tr>
<tr>
<td>Setup Azure Connection</td>
<td>9</td>
</tr>
<tr>
<td>Connect to Azure Subscription</td>
<td>10</td>
</tr>
<tr>
<td>Create the “AutomationData” Design-time Connection</td>
<td>12</td>
</tr>
<tr>
<td>Create the “AutomationData” Run-time Connection</td>
<td>12</td>
</tr>
<tr>
<td>Create Your First Runbook</td>
<td>15</td>
</tr>
<tr>
<td>Testing your runbook</td>
<td>24</td>
</tr>
<tr>
<td>Deeper Discovery Example</td>
<td>27</td>
</tr>
<tr>
<td>Appendix A – Creating an Azure SQL instance and Database</td>
<td>34</td>
</tr>
<tr>
<td>Database configuration</td>
<td>34</td>
</tr>
<tr>
<td>Create an Azure SQL Server</td>
<td>34</td>
</tr>
<tr>
<td>Configure the SQL Firewall to allow client Access</td>
<td>35</td>
</tr>
<tr>
<td>Create an Azure SQL Database</td>
<td>37</td>
</tr>
</tbody>
</table>
Overview

The Kelverion Runbook Studio is a graphical runbook authoring environment, that can be used to simplify runbook authoring for Azure Automation, as well as manage all your Azure Automation assets across multiple subscriptions.

Graphical runbook authoring within the Runbook Studio allows you to focus on the logic flow of your automation without getting bogged down with syntax, this change of focus opens the doors for many more people to exploit the power of Azure Automation. To further simplify the design and maintenance of your runbooks, the Runbook Studio provides “Smart Discovery” when used in conjunction with many cmdlets. Smart discovery allows you to use the Runbook Studio GUI to explore the application, cmdlets, and API that you are interacting with and generate runbooks with a few quick clicks.

The Runbook Studio can offer the greatest acceleration to your runbook development when used in conjunction with the Kelverion integration modules. These PowerShell modules can be used in any PowerShell environment and provide a rich integration experience while developing your runbooks and provide a supported method to integrate into many applications.

This guide will take you through the creation of your first runbook with the Runbook Studio and show you how to build a graphical runbook and exploit the power of smart discovery.
Pre-Installation Information

Before you begin
This tutorial assumes that you have already configured the following.

1) An Azure subscription.

2) The Runbook Studio is installed and connected to your Azure Subscription.

3) The Kelverion Integration Module for SQL Server installed on your machine and imported into your Automation Account. See the integration module user guide for more help.

4) An Azure SQL database called AutomationData complete with example data (The TestDrive VMs come with this pre-installed). For creating your own Azure SQL database, see Appendix A – Creating an Azure SQL instance and Database.

For help with the initial installation of the Runbook Studio please see our website(www.kelverion.com) or our YouTube channel (https://www.youtube.com/user/Kelverion)
**Kelverion Licensing**

The Kelverion Runbook Studio and the Kelverion Integration Modules that it uses both require licensing. Please request evaluation licenses by sending an email using this link [Evaluation License Request](mailto:EvaluationLicenseRequest) or send an email to “info@kelverion.com” with the subject “Evaluation License Request”.

To allow us to process your evaluation license request promptly please include the following information:

First Name:
Last Name:
Company name:
Job Title:
Country:

In response to your email you will receive the evaluation licenses for the runbook studio, and the integration modules as attachments. Save these attachments to the machine where the runbook studio is installed so they can be imported in the next section.

**Kelverion Runbook Studio License**

When the Kelverion Runbook Studio first starts it requests that you supply a valid license.

Click the “Add License” button and browse to the location that the KASL file is stored.

Ensure that the License Status changes to **green** with a valid date.
Kelverion Integration Module Licenses

The Integration Modules are Licensed from this same window by clicking on the “Integration Modules” section.

If you have closed this window, then you can open it again from “File \ About \ License Information”

Click on “Add License” and browse to the location where your evaluation licenses are saved.

Select the KAML file and click on OK

Ensure that the license has a green tick and a valid date.
Kelverion Runbook Studio Layout

In this section we will examine the parts of the Kelverion Runbook Studio. The image below has the key areas highlighted and are described in more detail below.

The Toolbar

The area highlighted in yellow as you would expect it allows quick access to many of the key functions. The tool bar also displays the default subscription, and default Automation Account

File menu

The file menu can be found just above the tool bar, it allows access to manage local copies of the runbooks that you create, as well as the menus to configure the runbook studio.

The Resources Pane

The area highlighted in blue is the Resources Pane. There are 2 modes for the Resources pane, Toolbox and Azure:

The Azure category is for managing the resources that exist within Azure. These are the Azure Assets such as variables, connections, credentials, and certificates along with Runbooks.

The Toolbox category provides access to the Activities that we will add to our Runbooks.

Switching between the “Toolbox” category or the “Azure” category toggles between those 2 modes.

You may also have noticed the search box at the top of the Resources pane. Both the Toolbox and Azure modes can be searched and filtered.
This example shows the list of cmdlets being filtered using the search facility.

The Runbook Canvas
The Green area in the middle of the screenshot above is the Runbook canvas, this is where we will place the Activities that go to make up our Runbook. You will see that a new empty Runbook is created on the Runbook canvas.
Right clicking on the Canvas brings up a context menu (shown highlighted in red) shows Use commands depending on when and where you right click on an area of the Kelverion Runbook Studio.

Runbook Properties
The purple area to the right of the image above is the properties pane, which is also context sensitive, containing the Activity properties or the Runbook properties.

Tip: Click an empty area in the Runbook canvas to get back to the Runbook properties after you have configured an activity.
Managing Assets
The Kelverion Runbook Studio can work in either an on-line or off-line mode. Here we see how to work connected to your Azure subscription by setting up the configuration needs to build a Runbook.
We must configure the Connections that are used at design time and those used at runtime separately. The “Connection Assets” within your Azure Automation account are used at the time of Runbook execution.
The “Smart Connections” used within the Kelverion Runbook Studio are used at Runbook design time to discover the properties of the target system and to assist your configuration of the Activities.

Setup Azure Connection
These steps only need to be carried out the first time you connect to an Azure Subscription.

In the Runbook Studio click on “File” and then “Azure AD”.

Click on “Manage Tenants” to open the Tenants window. Then click on “Add a Tenant” to enter a new Azure Tenant.
Enter the name of your Active Directory Tenant and the Active Directory Tenant ID.

The Active Directory tenant ID can be found either in the Azure Portal ("Directory ID" in Azure Active Directory > Properties) Or by logging in to your subscription with PowerShell. More information is available in the Runbook Studio User Guide.

N.B. If you running the Test Drive then you will find the Tenant ID in C:\__TestDrive__\Configinfo.txt

Click OK and OK to close these windows.

Ensure you make the tenant active by selecting it.

Connect to Azure Subscription
When you want to update resources in the cloud it is necessary to log in to Azure. You will notice that periodically you are required to reauthenticate as all Azure authorization tokens expire to enhance security. When the Kelverion Runbook Studio starts up it will not be connected to your Azure subscription.

Click on “Sign In” on the toolbar to start the login process
You will see the familiar Microsoft prompt to login to your Azure subscription.

Enter your Username and Password, then click “Sign in”

Once you have logged in you will notice that the subscription and account dropdowns are populated in the Toolbar.

*Note: If you have multiple Automation Accounts associated with your subscription, then you can easily manage them all with the Kelverion Runbook Studio.*

In the Kelverion runbook Studio click on the Azure component in the Resources Pane and expand the configuration.

Here you will see all the Accounts and Assets that are contained in the Automation Account.
Create the “AutomationData” Design-time Connection
The steps below show how to configure the design-time connection. This enables the Runbook Studio to make the connection to the databases in SQL Server during the development process.

Click on “Smart Connections” on the toolbar to open the “Smart Connections” window.

Click on “Add a Connection”

Add the “Name” “AutomationData” This should represent the type of data that the Connection represents

Select: “Kelverion. SqlServer” as the Connection type.

Enter the “ServerName”, “UserName” and “Password” for your Azure SQL server.

Click OK

Click OK on the “Smart Connections” screen to complete the process.

Create the “AutomationData” Run-time Connection
When runbooks are executing they use connection assets that are stored within Azure. These connections are different to the Smart connections that you have created for design time use, although they can point at the same targets.

This demonstration Runbook Studio comes with the AutomationData connection asset preconfigured. However, the connection in Azure will need the latest Integration Module license updated. To update the license on the connection, do the following:
Ensure that the runbook studio is connected to your subscription.

*Check the Status bar for confirmation. If it shows “Disconnected” then follow the instructions above to sign in.*

In the resources view

Expand Subscriptions > Automation accounts

Right Click on “Connections” to access the context menu

Click “Update License Keys”

You should get a pop up telling you that the connection assets were updated.

Click on OK to close the window.

If your TestDrive VM does not have an AutomationData connection asset then you will need to create one. To create an Azure connection asset for the “AutomationData” database do the following:

Ensure that the runbook studio is connected to your subscription.

*Check the Status bar for confirmation. If it shows “Disconnected” then follow the instructions above to sign in.*

In the resources view

Expand Subscriptions > Automation accounts

Right Click on “Connections” to access the context menu

Click “New Connection”
Enter the Connection Name

“AutomationData”

And Select “Kelverion.SqlServer” as the connection type.

**NOTE:** Using the same connection names for your Smart Connections, and your Azure connections simplifies management.

**NOTE:** if you do not see “Kelverion.SqlServer” as the connection type, then the integration module is not loaded in your Automation Account. Please see the integration pack user guide for instructions on loading integration packs.

After you select the Connection Type you will see that inputs appropriate to this connection type become available.

Enter the ServerName, UserName and Password for your Azure SQL Server

Click on OK to commit your changes.
Create Your First Runbook
In this section we will create a simple runbook and examine some of the elements of the runbook studio.

Click the “New Runbook” button on the toolbar.
Select “Graphical Runbook” from the dropdown.

In the Runbook Properties enter the name “MyFirstRunbook”

Optionaly: Enter a description.

Ensure that the Resources pane is in Toolbox mode
Expand **Smart Modules** to display the Kelverion Integration Modules that are installed on your system.

From the Kelverion.SqlServer section, drag and drop an “Insert-SqlRow” activity onto the runbook canvas.

Drag and drop and “Select-SqlRow” activity onto the runbook canvas.
Point to the “connection node” at the bottom of the “Insert-SqlRow” activity, you will notice that the mouse pointer changes.

Click on the “connection node” and drag a link to the “connection node” at the top of the “Select-SqlRow” activity.

**NOTE:** Each of the activities as has an exclamation point to alert you to the fact that its configuration is not complete yet.
Select the Insert-SqlRow activity and notice that the properties pane updates to allow you to access the properties of the activity (or whichever object is currently in focus)

**NOTE:** the notice explains in more detail what actions must be taken to complete the configuration of the activity so that the exclamation point is removed, and the runbook can be successfully published.

Click on the Discovery tab of the activity properties.

Select the “AutomationData” Smart connection (design time) that we created earlier.

You will see that at each stage, discovery prompts you with appropriate choices for the next level.

Select the AutomationData database.

You will notice that you can type in the table name box to filter the list of tables.

Try typing “dbo.O” in the box and see how the list is filtered.

Select **TestDrive.Computers**

*Now that discovery has completed for this table the runbook studio understands the structure of the table, which columns are*
mandatory, and the data types of each of the columns. It uses this information to help you configure the parameters for the activity correctly, this reduce errors and time spent troubleshooting. The graphical approach to building the queries also eliminates syntax errors in your queries.

Switch to the Parameters tab of the activity properties.

You will notice that the connection property (run time connection) is mandatory and has not yet been set.

You will notice that there are 10 optional properties for this insert activity too.

Click on the connection

And select “connection asset” as the value type (this says that we will use an Azure Connection Asset, for this parameter)

The drop down now shows you a list of the connection assets that you can select from.

Select “AutomationData”
You will notice that now a valid connection has been set, the exclamation points for the connection property have gone.

You will also notice that now all the mandatory properties are populated, the exclamation point on the activity within the runbook canvas has also been cleared.

*If the table was defined with “mandatory” fields they would be listed with the connection (an “optional field” in the database is one that allows NULL’s). The Runbook Studio is already helping validate your configuration here to ensure that you supply values for all the fields that do not allow NULL’s.*

*For example, changing the discovery to a different table (e.g. dbo.ACTIVITY_TRACE) you can see that there are several mandatory fields, along with some optional fields.*

Before moving on, set the discovery back to `TestDrive.Computers`

Click on the “Optional” button to show a list of the optional parameters for this activity

Use the search button to search for the “Name” field

*Note:* if you mouseover the info icon (/button) you will see a tooltip which shows that the field expects to consume a string and is not mandatory.
Set the data type to be “Constant”

You will see that there is a warning until you have entered a value.

Enter the value “TEST”

Well Done. Your first Activity is complete!

Now Select the “Select-SqlRow” activity in the runbook canvas.

Switch to the Discovery tab

Select the AutomationData connection, then the AutomationData database.

Finally select the table “TestDrive.Computers”

Select the Parameters Tab

Set the connection to use an Azure Connection Asset

Use the “AutomationData” Asset.
We don’t need to set any optional properties, but feel free to take a peek

Select the Filters tab

Click on Add

*This is where we build the “where clause” for our select statement.*

In the first drop down, you will see a list of the fields that exist within the database.

Select the ID Field

The second drop down “operation” shows the comparison operators that are appropriate for this field.

*For example, this is a numeric field, so we see Greater Than and Less Than. If we had selected a text field we would see contains, matches, ends with etc.*
In the Data Source drop down select "Activity Output"

*Using Activity Output from a previous activity allows us to “subscribe” to the data that it has “published” onto the pipeline. It’s to core of building a complex multi-activity runbook.*

When you selected “Activity Output” as the data source an additional drop down is added that allows you to select from all the activities that preceded this one within the pipeline.

Select “Insert-SqlRow”

You will notice that a further input “Field Path” is added. If the selected data source has complex output, this will allow you to browse the published properties, however the “Insert-SqlRow” simply published the ID of the record that is inserted. As a single unnamed attribute is published there is no browser, and we simply leave this input empty.

Congratulations! Your first runbook is complete.

Click the “Upload” button on the toolbar to upload it to Azure. In the next section we will see how to test the runbook.

In the resources pane switch back to the Azure View.

Expand your subscription and Automation Account.
Expand “runbooks” and you should see that your runbook now exists within the cloud.

Testing your runbook

The following instructions show how to change the logging level on the runbook and perform a test run. It should be noted that testing a runbook is NOT a simulated run. Any side effects, or changes made during the test are real changes applied to the target system. The only difference between a test run and a “real” execution is that the test run is carried out with the DRAFT version of the runbook rather than the published version. This is important as it allows you to make many small incremental changes to the DRAFT before you commit you changes over the top of the existing published version.

Expand Subscription > Automation Account > Runbooks

If you mouseover your runbook you will see that the icon is a yellow asterisk this indicates that the runbook is “new”.

You can also test a runbook that is in “edit” mode (yellow pencil)

NOTE:
Any publish runbooks (green tick) will need to have a draft version pushed into Azure before they can be tested. Using the runbook studio, you can easily open a published version and then push a draft copy back to Azure when required.

Right Click on your runbook to access the context menu, and click on “Test Draft”

Alternatively, if the runbook is open in the canvas, click into the runbook canvas and use the test button on the toolbar
The test dialogue will open.

The run on drop down allows you to select a hybrid worker for execution.

For now, though we are happy to execute the runbook in an Azure sandbox.

Click on the Start button.

After a moment you will notice that the status changes to “Submitted”

*At this point Azure Automation is provisioning your sandbox and configuring it in preparation for execution. It may remain in a submitted state for a short time, before eventually switching briefly to queued.*

Eventually the state will change to “Running”

And finally, either “completed” or “failed”

When the runbook has completed you will see the output from the runbook

You can expand the Test pane to provide a better view of the runbook output.

Click OK to close the Runbook test dialogue.
Click an empty area of the runbook canvas so the properties pane switches to the runbook Properties.

You will notice that there a tab for “logging and tracing” at the bottom.

Click on Logging and Tracing.

Try different combinations of the logging and tracing levels, and retest your runbook after each change, so you can see the extra levels of detail that are available.

**N.B.** You will need to Upload your runbook each time you make a change to the logging.

Please be aware that with a complex runbook, the highest levels of logging and tracing can have an impact on execution time.
Deeper Discovery Example

In our first example runbook, we quickly explored Smart Discovery. Let’s take a deeper dive into discovery. This example assumes that you have a connection to the Adventure Works database. The Adventure Works database can be downloaded from GitHub (https://github.com/Microsoft/sql-server-samples/releases/tag/adventureworks) or deployed as an option when creating a new database in Azure or if you are using the Kelverion gallery VM then the Adventure Works database is already installed in the SQL Express instance. These instructions assume that you are using the SQL Express instance in the Gallery VM.

In this example, we’re going to create a more complex runbook with outputs from the activities used as inputs to the subsequent ones. Just as we would in the majority of the real runbooks we create. So, let’s give the hard-working engineering department some extra holiday time as they’ve done such a great job with our new products.

NOTE:
The SQL express database is not accessible from Azure, so the following examples only describe the design time process, and do not include publishing and executing the runbooks in Azure.

Click on the “Smart Connections” in the toolbar

Click on “Add a Connection”
Enter the Name LocalSQL

The ServerName: Localhost

Leave the username and password blank

Set UseWindowsAuthentication to True

Click on OK

Click on OK again to return to the main page.

Switch the resources pane to the Toolbox

Search for Kelverion.Sql

Drag 2 Select activities and an update activity on to the runbook canvas and join them up with pipeline links.
Rename the activities:
Select-SqlRow  ->  Select Department
Select-SqlRow1 ->  Select Members
Update-SqlRow  ->  Update Members

Select the “Select Department” activity and switch to the discovery tab

Click on the connection drop down, and then select the “LocalSQL” configuration. The Runbook Studio will discover the available databases. Select the “Adventure Works” Database. The Runbook studio will now discover all the tables that exist within the database. If you start to type “human resources” in the search box you will see that the table list is filtered dynamically to show the tables that you are interested in.

Select the HumanResources.EmployeeDepartmentHistory Table.
Switch to the parameters tab in the Properties pane. Set the connection to be any available connection (we're not going to execute the runbook from Azure, so the runtime connection is unimportant.)

*Once the connection is set the activity will no longer have a warning!* (as all the mandatory properties have values)

Switch to the filters tab in the properties pane and click on “Add Filter”. A configurable filter option will pop up. Click on the filter dropdown and you will see a list of the columns that exists within the table.

Select DepartmentID.

*In the operations field, you will see that we have text specific operators (contains, like, etc.) If you change the filter column to ModifiedDate and check those operations again you will see that the operators are now specific to Date’s and Times. That’s an important feature of the discovery, and part of what makes it Smart discovery. The Runbook Studio in conjunction with the integration modules know how to interrogate the system that you are interacting with and take care of the details so that you don’t have to worry about it. It helps you to get the right low-level syntax for whatever the target application is, so you don’t need to be a SQL, or Service-Now Guru. You can focus on the runbook logic instead.*
Switch the Filter field back to DepartmentID, use the Equal operations and sent the comparison to a constant “1”

Click on OK.

*We’re done with the configuration of the “Select Department” activity, and it will “publish” each of the fields onto the pipeline for the engineering department for us to “subscribe” to in the later activities.*

Let’s configure the “Select Members” activity to get all the members of the engineering department.

Configure the discovery within the properties pane to point to the “Adventure Works” database, and the “HumanResources.Employee” table (don’t forget the search box for the table name.)

In the “parameters” tab of the properties pane set the Connection again, and then switch to Filters, and add a new filter. When the “Edit Filter Condition” dialogue opens select the Filter Column “BusinessEntityID” and “Equals” as the operation.

Now we’re going to use data that’s output from the previous activity rather than a constant value as we did before.

In the Value field select “Activity output”.

The Activity drop down is added, and this allows you to select the activity that we want to consume output from, select the “select department” activity.
Finally, “field path” input will allow you to select the particular property that you want to use.

If you look at the fields available there and check the able definition you will notice that each of the fields in the database table have been made available to us.

Select the “BusinessEntityID” field in the field path.

When the runbook is executed now we will have published the details of all the members of department 1 onto the pipeline. Our final activity is going to update the properties of those employees.

Click on the final activity in our runbook “Update Employee” and configure the discovery tab to point at the “employee” table again.

Then in the filters tab set the RowGUID to equal the activity output from the select members, and the field path to the RowGUID.

Now switch to the parameters tab and set the runtime connection property.

Finally, we will set the new value for the VacationHours, because we want to perform a calculation as before we update the data we will use a PowerShell expression as the input type. This allows us a huge amount of flexibility, but it means...
that we cannot use the GUI to do *All* of the hard work.
Select the optional field “VacationHours”
(don’t forget you can type in the search box at the top to filter the list)
And set the input type to be “PowerShell expression”
Finally set the expression to be

\[
\text{ActivityOutput["Select Members"].VacationHours + 16}
\]

Which is the PowerShell way of subscribing to the “VacationHours” property for the “Select Members” activity.
Obviously when you type the PowerShell instead of letting the Runbook Studio do the hard work you need to be careful with the spelling’s and Syntax.

So, our Deeper Discovery example is complete. If you’re adventure works database is accessible from Azure, then you can publish the runbook and execute it and give all those hardworking engineers a well-earned rest.

Hopefully you can appreciate the huge time saving that the smart discovery can make when you’re developing runbooks.
Appendix A – Creating an Azure SQL instance and Database.

The process of creating a new SQL Server and database in your Azure subscription is very quick, however there are a number of steps that need to be completed in the correct order. These instructions take you through a basic deployment that will allow you complete the first tutorial.

**Database configuration**

To start we will create a new SQL instance in Azure, create a database within it and then add the table that the runbooks use to store the data. This will take about 15 minutes.

**Create an Azure SQL Server**

Login to the Azure portal and select the SQL Server resource type. Click on the Add Button.

Click on the Add Button
Populate each of the fields as shown above. You will need to enter a server name that is unique to you.
Once all the fields are populated Click on OK.
Your SQL Instance will be deployed this typically takes less than 5 minutes.

**Configure the SQL Firewall to allow client Access**  
In the SQL Servers blade, click on the database that has just been created
In the settings section click on “Firewalls and virtual networks”

**- Firewalls and virtual networks**

Click on “Add client IP” to add your current IP address to the list of addresses allowed through the firewall. Then click on “Save”

You can now close the SQL Server blades.
Create an Azure SQL Database
Once your SQL instance is deployed to Azure you can create a new SQL database

Select the SQL Databases resource type in the Azure portal

Once the Databases blade opens Click on the Add button
Populate the fields in the Add Database blade.
The runbooks that we will import later assume that the database name is entered as **AutomationData**, using the same database name will simplify the rest of the process, so you should stick with that for now.

Ensure that you use the resource group and server that were created in the previous step.  
For the purposes of the tutorial the "basic" pricing tier is recommended as we do not require any of the more advanced features that are available.