

# Kelverion Automation

## VM Provisioning and Management

### Application for Azure Automation

## Deployment Guide

Version 2.3

Email: [info@kelverion.com](mailto:info@kelverion.com)

Web: [www.kelverion.com](http://www.kelverion.com)

# 1 Table of Contents

<b>2</b>	<b>Overview.....</b>	<b>3</b>
<b>3</b>	<b>General Configuration Steps .....</b>	<b>3</b>
3.1	Pre-Installation Information .....	3
3.1.1	<i>Kelverion Items Required .....</i>	<i>3</i>
3.1.2	<i>Other Products Required .....</i>	<i>3</i>
3.1.3	<i>Installation Steps.....</i>	<i>4</i>
3.2	Persistent Data Store.....	5
3.3	Configure the Automation Portal.....	5
3.4	Create an Azure user for starting runbooks .....	5
3.5	Load Integration Modules .....	6
3.6	Configure your Automation account using the Runbook Studio.....	7
3.6.1	<i>Design Time (Smart Connections).....</i>	<i>7</i>
3.6.2	<i>Azure Variables.....</i>	<i>9</i>
3.6.3	<i>Azure Credentials.....</i>	<i>12</i>
3.6.4	<i>Azure Runtime connections.....</i>	<i>13</i>
3.7	Import Runbooks using the Runbook Studio .....	15
3.8	Runbook Customisation.....	16
3.8.1	<i>High Level Overview.....</i>	<i>16</i>
3.9	Runbook Process Flow .....	17
3.9.1	<i>Request Data Storage (PDS Design).....</i>	<i>17</i>
3.9.2	<i>VM Provisioning and Management (PDS Design).....</i>	<i>18</i>
3.9.3	<i>Gathering the Data.....</i>	<i>21</i>
3.9.4	<i>Transposing the Data.....</i>	<i>22</i>
3.9.5	<i>Using the Data .....</i>	<i>23</i>
3.9.6	<i>Returning the Results.....</i>	<i>24</i>
3.9.7	<i>Scheduled Runbooks .....</i>	<i>25</i>
3.9.8	<i>Using Hybrid Workers.....</i>	<i>26</i>
3.10	Logic Apps - Scheduling runbook execution.....	27
3.10.1	<i>Gather Request Logic App.....</i>	<i>27</i>
3.10.2	<i>Return Status Logic App.....</i>	<i>31</i>
3.11	Testing.....	35

## 2 Overview

The VM Provisioning and Management application for Azure Automation, it allows your users to provision and manage virtual machines without the need to provide your users any direct access to your virtual machine administrator tools.

The application provides a simple and flexible interface that prompts the user for the required information to provision a virtual machine, while remaining tightly constrained so that the users can only deploy the resources that you want in a configuration that is approved. It comes preconfigured to run through the Keverion Automation Portal, although it can easily be modified to accept other service desk portals. The simplicity of the portal means that the users are not faced with dealing with understanding the complexity of the resources that are required before they deploy the virtual machines.

The Runbooks have been written using the Runbook Studio authoring application and leverage the integration and smart discovery capabilities provided by the Integration Module for SQL Server. These Integration Modules are also available in the PowerShell Gallery.

## 3 General Configuration Steps

### 3.1 Pre-Installation Information

The VM Provisioning and Management application package contains the following elements:

- Persistent Data Store (PDS) SQL configuration script
- VM Provisioning and Management Azure Automation Runbooks
- Azure Automation Service Export

#### 3.1.1 Keverion Items Required

The application requires the following Keverion products:

- Keverion Runbook Studio
- Keverion Integration Module for SQL Server
- Keverion Automation Portal

If you do not already have Keverion Integration Modules, Keverion Automation Portal, or the Keverion Runbook Studio they can be downloaded for evaluation from our website.

This guide assumes that you have already installed the Runbook Studio and the Automation Portal. If you have not yet installed those products, then please do so before you continue. Each of the product downloads contains its own documentation to guide you through the initial configuration.

#### 3.1.2 Other Products Required

The following Microsoft PowerShell modules are required:

- Az

- Az.Accounts
- Az.Automation
- Az.Compute
- Az.Network
- Az.Resources
- Az.Storage

These modules are available from the PowerShell Gallery

### 3.1.3 Installation Steps

As a guide the steps taken are as followed:

1. Configure the PDS database
2. Import and configure the Portal Service (If using the Keverion Automation Portal)
3. Configure the Service Offerings (If not using the Keverion Automation Portal)
4. Create the Azure components
  - a. Resource Group
  - b. Automation Account
  - c. Create Managed System Identity
  - d. Load Integration Modules
  - e. Import Runbooks
  - f. Create Smart Connections
  - g. Create Azure Variables
5. Create the Gather Runbook 90-XX (If not using the Keverion Automation Portal)
6. Create the Return Runbook 97-XX (If not using the Keverion Automation Portal)
7. Configure the Convert Runbook 95-XX
8. Create the Logic Apps

### 3.2 Persistent Data Store

The Persistent Data Store or PDS is a SQL Server database that is used by these runbooks to allow all the actions that the runbooks take to be carried out in a robust way. The use of the database at each “step” allows us to design the runbooks such that each runbook is simple and can be considered a discrete unit. In programming terms, it allows the runbooks to be modular.

To best exploit the power and flexibility of Azure, the PDS should be deployed to a SQL instance within your Azure subscription.

We will be using a PDS on Azure using Azure's SQL offerings, rather than building a VM and installing SQL. This allows us to deploy the SQL instance and PDS database quickly and with the minimum of maintenance.

1. Create a new Azure SQL Server. Create the SQL Server in a New Resource Group "Automation", ensure that the "Allow azure services to access server" check box is ticked. This means that ALL Azure resources will be allowed to access the SQL Server through the firewall
2. Give your desktop access to the SQL Server through the firewall
3. Create a new Database "AutomationData". It's important to use this name for the PDS, as it is held within the runbooks. The **BASIC** tier is ample performance for testing and evaluation of the application. As it is trivial to scale up or down the databases this should also be the starting point for your deployments unless you know that there is going to be a high volume of alerts right from the outset.
4. Connect to the database using SQL Management Studio
5. Execute the SQL script (*PDS\_VMPM.sql*) from the package to Create the database tables and views.

### 3.3 Configure the Automation Portal

Import the service request definition `Services_VMPM.export`

The following queries must be updated to point to the customers PDS

- VM Provisioning (Azure) - Infrastructure
- VM Provisioning (Azure) - Location
- VM Provisioning (Azure) - Zone
- VM Provisioning (Azure) - VM Platform
- VM Provisioning (Azure) - VM Images
- VM Provisioning (Azure) - Get Azure Template Types
- VM Provisioning (Azure) - List Existing VMs

The SQL instance AND authentication details will need to be updated for each query.

### 3.4 Create an Azure user for starting runbooks

In Azure AD create a "local" Azure AD User.

You will need to login to the portal using the user account to set the password, as the password change flag is set when the account is created.

The account should be set as an "Automation Operator" for the Automation account (Access Control (IAM) blade under the automation account). The username and password for this account will need to be configured within your Logic App so make a note of these values.

### 3.5 Load Integration Modules

The Integration modules will need to be installed in the following locations

- The Automation Account ("Assets" > "Modules")
- The machine where you are running the Runbook Studio.

The modules can easily be installed on the machine from the PowerShell Gallery.

Visit <https://docs.microsoft.com/en-us/azure/automation/automation-update-azure-modules> for more information about loading Integration modules into your Automation Account.


### 3.6 Configure your Automation account using the Runbook Studio


Connect the Runbook studio to your target Azure subscription. You can find more information on the initial configuration of the runbook studio on pages 5 and 6 of the User's Guide.

#### 3.6.1 Design Time (Smart Connections)

The runbook studio needs to have connections configured for use at design time, these smart connections are used by the discovery process within the Runbook Studio to allow the Runbook Studio to discover information about your target systems. This discovery process accelerates the process of runbook development.

Create the following smart connections within the Runbook Studio.

AutomationData	<div><div><div><div><div></div><div>Edit Smart Connection</div><div><span>✕</span></div></div><div><div>Name</div><div><input type="text" value="AutomationData"/></div></div><div><div>Description</div><div><input type="text"/></div></div><div><div>Connection type</div><div><input type="text" value="Kelverion.SqlServer"/></div></div><div><div>ServerName ⓘ</div><div><input type="text" value="database.windows.net"/></div></div><div><div>UserName ⓘ</div><div><input type="text" value="localadmin"/></div></div><div><div>Password ⓘ</div><div><input type="password" value="....."/></div></div><div><div>UseWindowsAuthentication ⓘ</div><div><input type="text" value="False"/></div></div><div><div>ConnectTimeout ⓘ</div><div><input type="text" value="60"/></div></div><div><div>OK</div><div>Cancel</div></div></div></div></div>
----------------	--

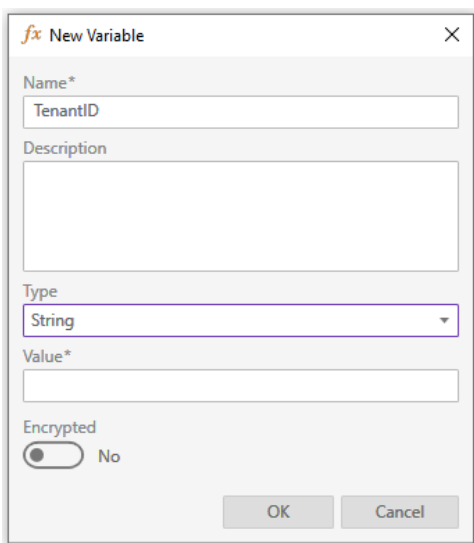
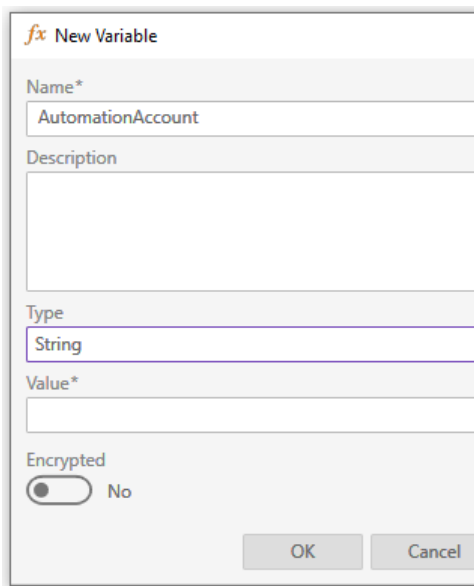
<p>AutomationPortal</p> <p>N.B. Only if you are using the Kolverion Automation Portal</p>	<div><div> Edit Smart Connection <span>✕</span></div><div><div>Name *</div><div>AutomationPortal</div></div><div><div>Description</div><div>Smart Connections created via manual consolidation</div></div><div><div>Connection type</div><div>Kolverion.SqlServer</div></div><div><div>ServerName * ⓘ</div><div><div></div>.database.windows.net</div></div><div><div>UserName ⓘ</div><div></div></div></div>
---	--

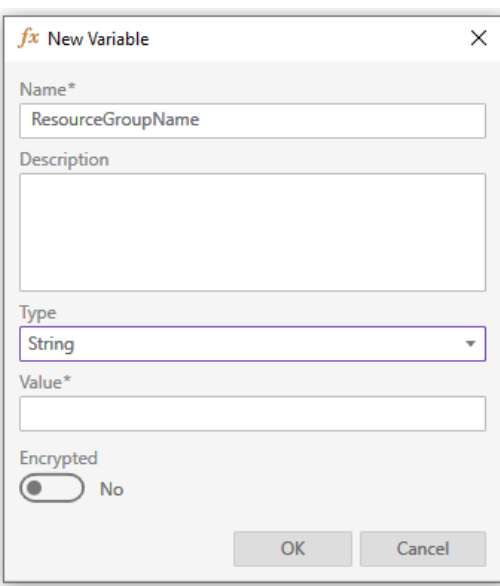
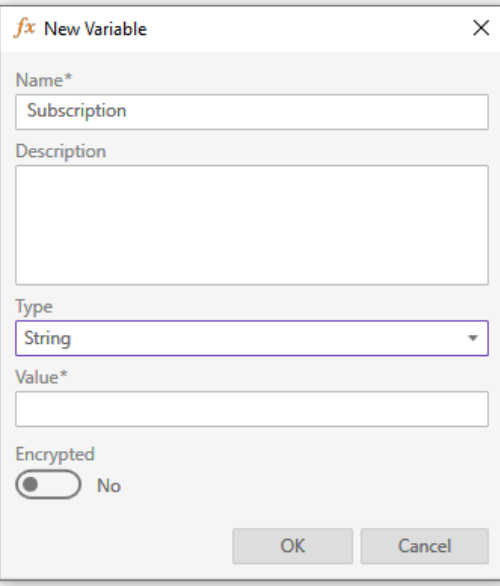


### 3.6.2 Azure Variables

Azure variables allow us to remove static configuration information from the code in our runbooks and store the information in an easy to access place. This helps us to build consistent configuration between all our runbooks. These values are accessed at design time, and at runtime by both the Hybrid workers, and the Azure Worker sandboxes.

Create the following variables in the Automation Account using the Runbook Studio.

TenantId	The Tenant Id of your Azure Subscription	
AutomationAccount	The name of the automation account to run the child runbooks	

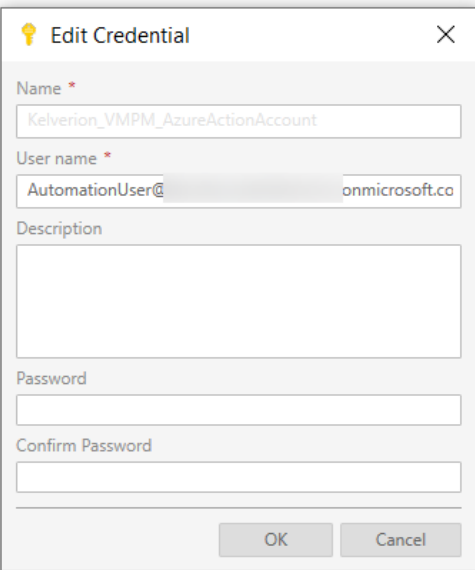
ResourceGroupName	The name of the resource group that has the runbooks	
Subscription	The subscription that the runbooks are launching in	



### 3.6.3 Azure Credentials

Azure Credential assets allow us to create and manage credential objects that can be utilised throughout our runbooks. These credentials are accessed by our runbooks at runtime.

Create the following Credential Assets using the Runbook Studio.

<p>Kelverion_VMPM_AzureActionAccount</p>	<p>This credential will be used to connect to Azure to build the virtual machines. It will require the applicable permissions to any subscriptions \ resource groups that are associated with the virtual machines.</p>	
--	---	--

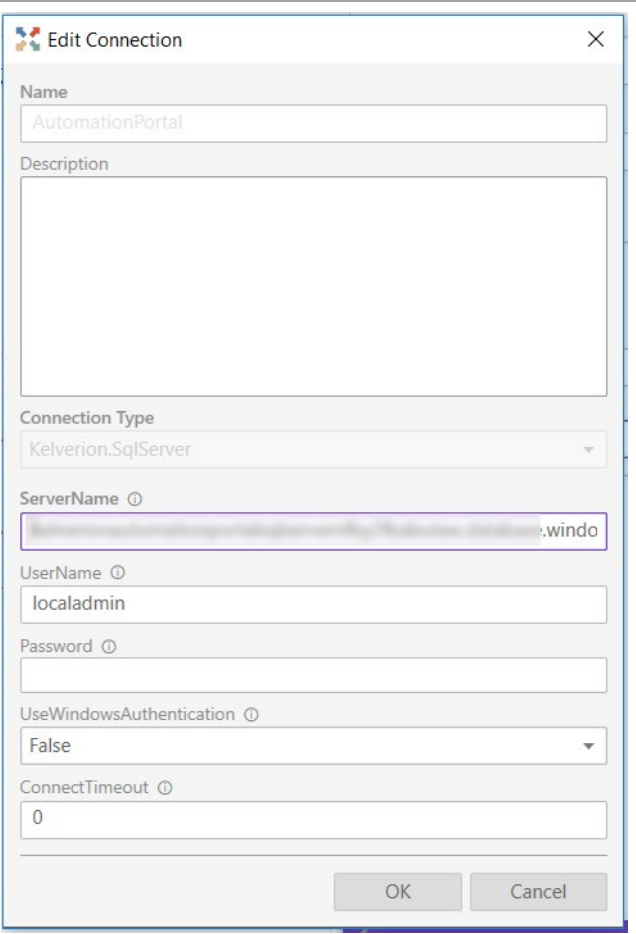
N.B. If you are building the application for on-premise HyperV (VMM) then you will require a Hybrid Worker account to connect your Hybrid Worker.

3.6.4 Azure Runtime connections

Azure connection assets are used at runtime to define a reusable connection configuration. The connection types available are dependent upon module that have been loaded into your Automation Account. If you cannot see the connection types listed below when you attempt to create the connection assets, then this indicates an issue with the modules that are loaded into your automation account. Please verify that all the required modules are loaded.

Create the following Connection in Azure using the Runbook Studio.

AutomationData	The "AutomationData" connection asset in Azure defines the connection that will be used at runtime for the runbooks to connect to the PDS database.	
----------------	---	--

AutomationPortal	The "AutomationPortal" connection asset in Azure defines the connection that will be used at runtime for the runbooks to connect to the Keverion Automation Portal Database.	
------------------	--	---

### 3.7 Import Runbooks using the Runbook Studio

Connect to the Automation Account using the Runbook Studio.

Check that you have the correct Automation Account set as the default target (if there is more than one Automation Account associated with the subscription)

Open each of the following runbooks, then "publish draft" and then "publish" the runbooks.

Runbook Name	Brief Description
Kelverion_VMPM_20-0_Util-GetRunbookData	This runbook will gather Request data from the Automation Portal
Kelverion_VMPM_30-1-1_Worker_AzureDeployStandardVM	This will deploy a virtual machine to Azure
Kelverion_VMPM_30-1-2_Worker_HyperVDeployStandardVM	This will deploy a virtual machine to HyperV (SCVMM)
Kelverion_VMPM_30-2-1-1_Worker_AzureStartVM	This will start an Azure virtual machine
Kelverion_VMPM_30-2-1-2_Worker_AzureStopVM	This will stop an Azure virtual machine
Kelverion_VMPM_30-2-1-3_Worker_AzureRestartVM	This will restart an Azure virtual machine
Kelverion_VMPM_30-2-2-1_Worker_HyperVStartVM	This will start a HyperV virtual machine
Kelverion_VMPM_30-2-2-2_Worker_HyperVStopVM	This will stop a HyperV virtual machine
Kelverion_VMPM_30-2-2-3_Worker_HyperVRestartVM	This will restart a HyperV virtual machine
Kelverion_VMPM_90-01_KAP_Get_Request	Gathers the request data from the Kelverion Automation Portal. Requires a Logic App to trigger
Kelverion_VMPM_95-01_ConvertToRequestData	Converts the gathered data to the correct format for the runbooks
Kelverion_VMPM_97-01_KAP_Return_Status	Collects processed rows of the PDS for returning data back to the Kelverion Automation Portal. Requires a Logic App to trigger

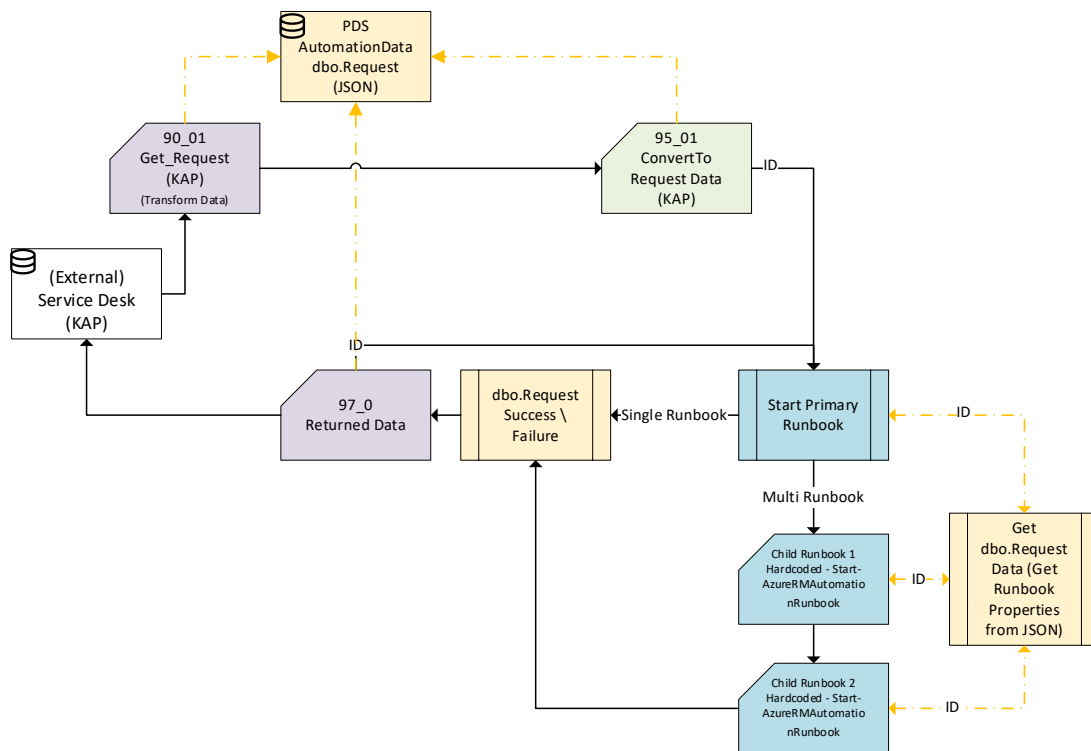
### 3.8 Runbook Customisation

This application is designed to allow flexibility across multiple service desk applications. The initial runbooks are configured for using the Keverion Automation Portal as the main user portal for initiating any requests and receiving the status updates.

If you are not using the Keverion Automation Portal, then the installation engineer will be required to create a gathering runbook (90\_XX) and a return runbook (97\_XX).

#### 3.8.1 High Level Overview

The main process flow is as follows:





### 3.9 Runbook Process Flow

This section covers a more detailed description of how the runbook logic fits together. You should be able to use it to configure and customise the application.

#### 3.9.1 Request Data Storage (PDS Design)

The applications use a SQL database (PDS) to store the request \ offerings from the Service Desk. The table dbo.Request stores data per request from the source Service Desk. Each row has a unique ID that is used as a reference for the worker runbooks.

RunbookOwner	Data	Message	ServiceName	OfferingName	ExternalId	State	OutputData
Kelverion_STSK...	{ "First Nam...	User **Orchestrator....	Business Us...	New Joiner	306	Complete	{ "Runbook": "Ke
Kelverion_O365_...	{ "Reason fo...	User disabled	Office 365	Disable Login	305	Complete	{ "ObjectId": "240
Kelverion_O365_...	{ "User": { ...	User enabled	Office 365	Enable User	304	Complete	{ "ObjectId": "240

##### 3.9.1.1 Request Table

Description of the column use in the dbo.Request table.

Column	Description
ID	Unique column ID. Created when data is inserted.
Created	Create time stamp
Updated	Updated time stamp
RequestedBy	Service Offering Requestor
Data	JSON Data dump from the request offering
Message	Return message to the service desk
Deleted	(Bit) 1 = request deleted
ServiceID	Numerical ID of the Service Name (Optional and service desk dependent)
ServiceName	Service Name from the service desk request
OfferingID	Numerical ID of the Offering Name (Optional and service desk dependent)
OfferingName	Offering Name from the service desk request
ExternalId	Service Desk reference ID
State	Worker Runbook state
RowVersion	SQL Row version. Generated automatically on data entry
ServiceDesk	Name of the service desk being used
OutputData	JSON Data used by the worker runbooks

### 3.9.1.2 Request Schedule Table

This table is used for scheduling any runbook deployments. The use of it is covered in the section “Scheduling Runbooks”

Description of the column use in the dbo.RequestSchedule table.

Column	Description
ID	Unique column ID. Created when data is inserted.
_created	Create time stamp
_updated	Updated time stamp
_state	Standard Kelverion PDS column. E.g. Unprocessed entries are “New”
Request_ID	ID of the request offering from the dbo.Request table
Runbook	The name of the runbook that will be called
StartDate	The date \ time of when the runbook will be actioned
EndDate	The date \ time of when the runbook will be actioned
ExternalId	To store a 3 <sup>rd</sup> party request reference

### 3.9.2 VM Provisioning and Management (PDS Design)

The following tables are used to allow the portal to provide data to the end user. Only the VM Assets table is written to by the runbooks. All other tables will need to be populated if you are using the Kelverion Automation Portal. Example data is included in the PDS script.

#### 3.9.2.1 Infrastructure Images

This table is used to store the available virtual machine images that are available to the portal offering.

Column	Description
_id	Unique column ID. Created when data is inserted.
_created	Create time stamp
_updated	Updated time stamp
_state	Standard Kelverion PDS column. E.g. Unprocessed entries are “New”
_owner	Standard Kelverion PDS column. Defines the owner or person who has updated the data
ImageName	VM Image name
ImageDescription	VM Image description
Publisher	Published of the Image. E.g. Amazon, Microsoft Windows Server
Offer	e.g. WindowsServer
Sku	e.g. 2012-R2-Datacenter
Architecture	x64 etc
Platform	Linux \ Windows
ObjectStatus	Valid values: Active, Disabled
AssetStatus	Valid values: Active, Disabled
Notes	For any descriptive notes
Infrastructure	Microsoft Azure \ Microsoft HyperV etc.

### 3.9.2.2 Infrastructure Instances

This table is a summary of the virtual machine images, that allows information to be fed into the portal offering.

Column	Description
_id	Unique column ID. Created when data is inserted.
_created	Create time stamp
_updated	Updated time stamp
_state	Standard Keverion PDS column. E.g. Unprocessed entries are "New"
_owner	Standard Keverion PDS column. Defines the owner or person who has updated the data
CPUs	
MemoryGB	
InstanceName	
HddSizeGB	
OsName	
InstanceId	
ObjectStatus	Valid values: Active, Disabled
AssetStatus	Valid values: Active, Disabled
Notes	
DisplayName	Name of Infrastructure. E.g. Amazon EC2, Microsoft Azure

### 3.9.2.3 Infrastructure Regions

This table stores data about the virtualisation platform and applicable locations \ networks.

Column	Description
_id	Unique column ID. Created when data is inserted.
_created	Create time stamp
_updated	Updated time stamp
_state	Standard Keverion PDS column. E.g. Unprocessed entries are "New"
_owner	Standard Keverion PDS column. Defines the owner or person who has updated the data
Infrastructure	Name of Infrastructure. E.g. Amazon EC2, Microsoft Azure
Region	For Azure: West Europe etc.
RegionRunbookPath	Not used
RegionZone1	Can be used to further define a region or resource group in azure
RegionZone2	
RegionZone3	
RegionZone4	
ObjectStatus	Valid values: Active, Disabled
AssetStatus	Valid values: Active, Disabled
Notes	
DisplayName	Name of Region. E.g. Microsoft Azure - Prod

#### 3.9.2.4 VM Assets

This table stores any virtual machines that have been built using the application. It is also used as a reference for queries for the Keverion Automation Portal. Any device stored in this table will be available for the Start \ Stop \ Restart offerings.

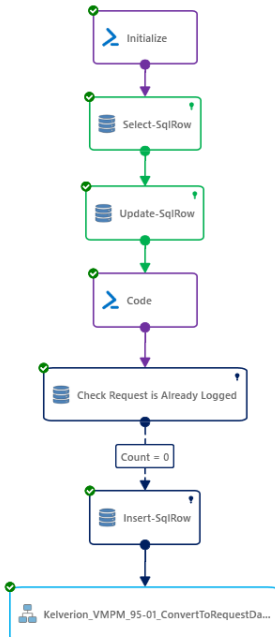
**N.B.** This table is directly referenced by the runbooks so it can keep track of available virtual machines.

Column	Description
_id	Unique column ID. Created when data is inserted.
_created	Create time stamp
_updated	Updated time stamp
_state	Standard Keverion PDS column. E.g. Unprocessed entries are "New"
_owner	Standard Keverion PDS column. Defines the owner or person who has updated the data
AssetStatus	Valid values: Deployed \ Deleted
InstanceID	
FQDN	FQDN
Description	Description field
DeploymentType	e.g. Standard_A2 etc
VMTemplateUsed	Template Name
DeploymentDate	Date time
ExpiryDate	Date time
AssignedMemory	Memory in GB
NumberCPUs	CPU count
SysHddSizeGB	
DynamicMemory	
NetworkName	
NetworkID	
Infrastructure	Name of Infrastructure. E.g. Amazon EC2, Microsoft Azure. Matched from Region Table. Check the entry there.
Region	West Europe \ UK South etc
SubRegion	
Location	
InfrastructureGroup	Resource Group if Azure
Notes	

### 3.9.3 Gathering the Data

The base applications come with a runbook designed to use the Kelverion Automation Portal.

Other Service Desk portals with available Kelverion Integration Modules can be used too, but a gathering runbook will need to be created.



The runbooks are driven from an Azure Logic App, to either:

- Monitor the Service Desk portals database (SQL)
- or
- Launch the gather runbook every 'x' minute(s)

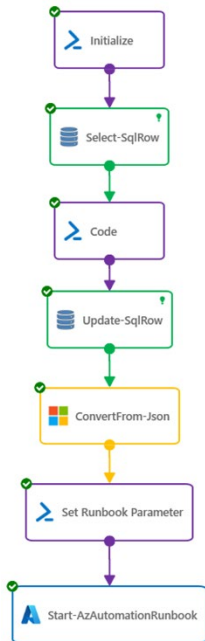
Each application has a 90-0x\_XXX\_Get\_Request runbook that can be configured for the required Service Desk portal.

This runbook has a code block that will transpose the Automation Portal request into JSON format and store it into the PDS dbo.Request table. Other service desks will require different activities to store the code in the corresponding JSON format.

RunbookOwner	Data	ServiceName	OfferingName	ExternalId	State	OutputData
Kelverion_STSK_21-1_Work...	{ "First Name": "Bob", ...	Business User...	New Joiner	306	Complete	{ "Runbook": "Kelverion_JML_30-...
Kelverion_O365_10-4_Worke...	{ "Reason for Blocking ...	Office 365	Disable Login	305	Complete	{ "ObjectId": "240ba31e-6a1a-46d...
Kelverion_O365_10-2_Worke...	{ "User": { "..."	Office 365	Enable User	304	Complete	{ "ObjectId": "240ba31e-6a1a-46d...

### 3.9.4 Transposing the Data

The modular runbooks are designed to use JSON. However, they require the JSON to be in a specific format. Each application has a runbook (Kolverion\_XXX\_95-01\_ConvertToRequestData) to transpose the data into the required format for the worker runbooks.



These runbooks will all look similar. This depends on if the worker runbooks require the use of a Hybrid Worker. In this case you may see additional logic at the end of the Runbook.

The only activity that will require modification is the “Code” activity. This PowerShell activity has an input of JSON (Data) and transposes it to JSON (OutputData).

Example Code for an offering with a single worker runbook.

```

'Delete User' {
    Service Desk $out = [PSCustomObject]@{
    Offering Name Runbook = 'Kolverion_0365_10-3_Worker-Delete-User'
    Worker Runbook 'Delete User' = [PSCustomObject]@{
    Offering Name UserPrincipalName = $inputConverted.'User'.UserPrincipalName
    Worker Runbook Inputs ObjectId = $inputConverted.'User'.ObjectId
    }
}
  
```

Annotations in the image point to: 'Delete User' (Runbook Name), \$out (Service Desk), Runbook (Offering Name), 'Delete User' (Worker Runbook), UserPrincipalName (Offering Name), and ObjectId (Worker Runbook Inputs).

Each service desk offering must pass through the name of the ‘Parent Worker Runbook’. This can be a hidden field in the service desk portal.

The activity Set Runbook Parameter passes the ID column, from the Request table, and feeds it into the Start-AzureRMAutomationRunbook activity.

Start-AzureRMAutomationRunbook launches the Parent Worker Runbook defined from the Service Desk offering.

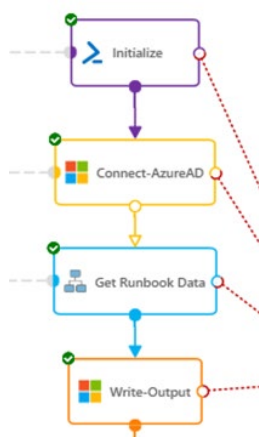
RunbookOwner	Data	ServiceName	OfferingName	ExternalId	State	OutputData
Kolverion_STSK_21-1_Work...	{ "First Name": "Bob", ...	Business User...	New Joiner	306	Complete	{ "Runbook": "Kolverion_JML_30-...
Kolverion_O365_10-4_Worke...	{ "Reason for Blocking ...	Office 365	Disable Login	305	Complete	{ "ObjectId": "240ba31e-6a1a-46d...
Kolverion_O365_10-2_Worke...	{ "User": { "..."	Office 365	Enable User	304	Complete	{ "ObjectId": "240ba31e-6a1a-46d...

### 3.9.5 Using the Data

Each worker runbook has been designed with the following:

- Single input of ID (from the dbo.Request table)
- Calls a child runbook Kelverion\_XXXX\_20-0\_Util-GetRunbookData to gather the required runbook inputs

This allows the correct input data to be gathered back from the PDS.



Here is an example start of a worker runbook.

It will:

- Set some initial variables (Initialize)
- Connect to Azure (Connect-AzureAD)
- Call 'Get Runbook Data' runbook
- Output the returned data (Write-Output) for use in the main runbook activities

N.B. More complex array outputs may require a PowerShell code block to process them, rather than a Write-Output.

'Get Runbook Data' has 2 required inputs:

1. ID
2. OfferingName (Worker Runbook Offering Name)

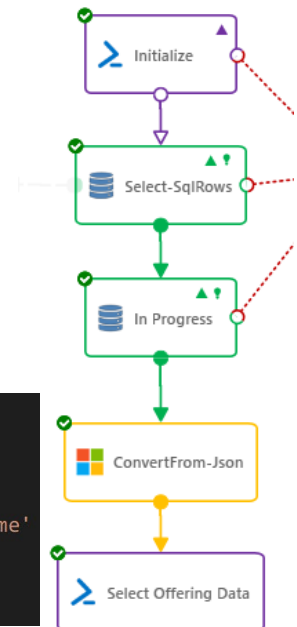
The activity 'Get Runbook Data' uses the ID input to go and retrieve the OutputData from the PDS.

The data is converted back from JSON and the 'Select Offering Data' filters out the required data based on the worker runbook offering name.

```

'Delete User' {
  Service Desk      $out = [PSCustomObject]@{
  Offering Name     Runbook = 'Kelverion_0365_10-3_Worker-Delete-User'
  Worker Runbook    'Delete User' = [PSCustomObject]@{
  Offering Name     UserPrincipalName = $inputConverted.'User'.UserPrincipalName
  Worker Runbook Inputs ObjectId = $inputConverted.'User'.ObjectId
  }
}
  
```

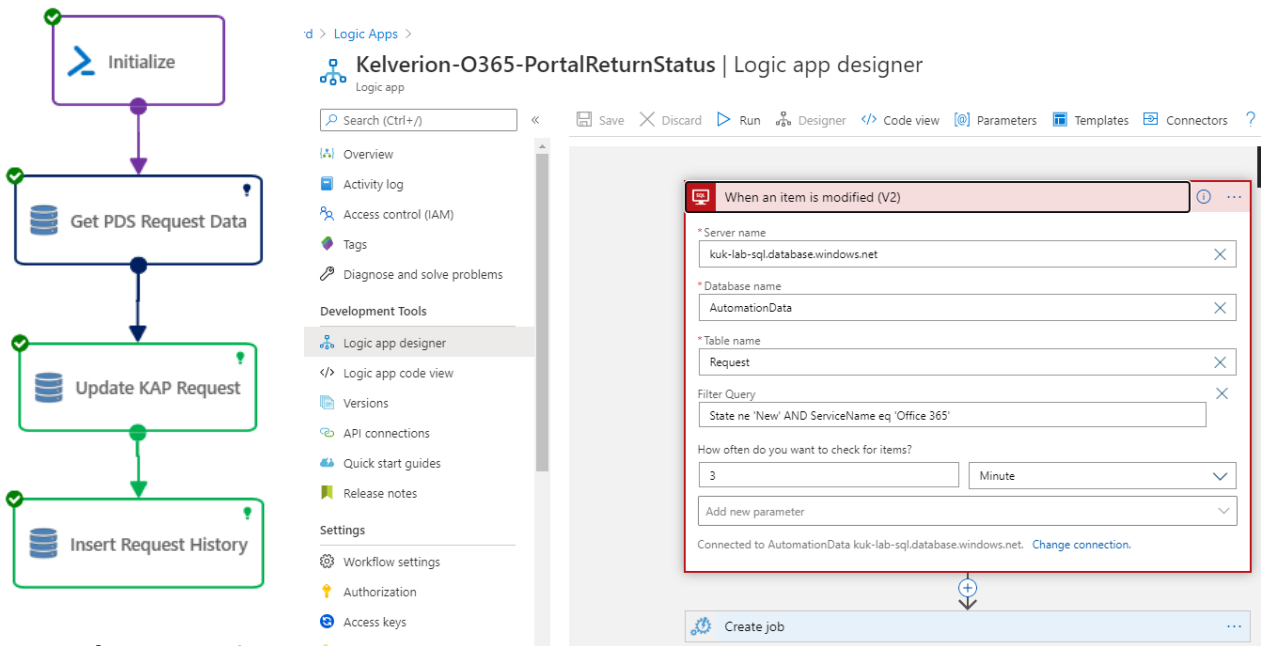
Labels in the diagram point to: Parent Runbook Name (Runbook), Worker Runbook Offering Name ('Delete User'), and Worker Runbook Inputs (UserPrincipalName, ObjectId).



### 3.9.6 Returning the Results

Each application has a return runbook that gathers updates to the PDS table and returns the message and state back to the required Service Desk portal.

For the Kelderion created applications, this uses the Kelderion Automation Portal. The runbook will be launched via a Logic App in Azure that detects the changes in the dbo.Request table.

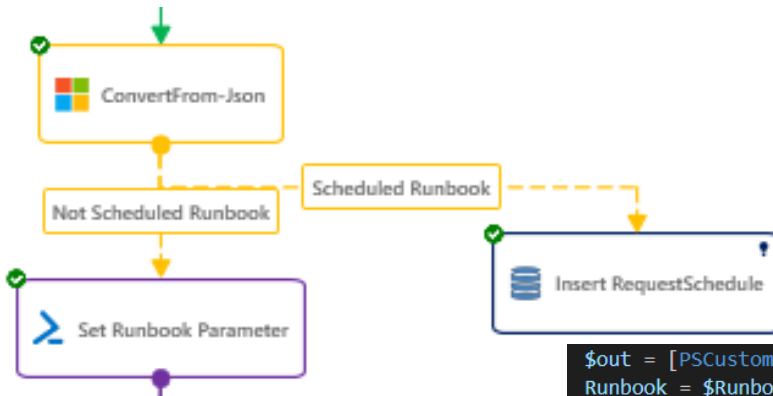


N.B. The Logic App filter can be modified if more than one ServiceName exists so that a single Logic App can be used for multiple applications.



### 3.9.7 Scheduled Runbooks

The offerings come with the option of specifying a Date to start \ expire a virtual machine. To handle scheduled requests some additional logic is added to the Kelverion\_XXX\_95-01\_ConvertToRequestData runbook. Additionally an extra PDS table dbo.RequestSchedule is used.



The additional logic is checking for a value added to **ActionDate** in the request data. If you do not have a field called this, then the runbook will move on and process the first runbook immediately.

The VM Provisioning and Management application uses **ActionDate** and **EndDate**.

Only **ActionDate** is required to add the entry to the RequestSchedule table. If you only require EndDate, then you will need to customise the link conditions on the runbook.

The runbook can handle a 'Start Time'. Use the value **ActionTime** in the Code block. Final written value to the PDS is combination of **ActionDate** and **ActionTime** in the **Insert RequestSchedule** activity.

```

$DateTime = $ActivityOutput
['ConvertFrom-Json'].$Offering.ActionDate
+ " " + $ActivityOutput
['ConvertFrom-Json'].$Offering.ActionTime
  
```

```

$out = [PSCustomObject]@{
  Runbook = $Runbook
  Hybrid = $Hybrid
  Wait = $Wait
  'Deploy Standard VM' = [PSCustomObject]@{
    VMFQDN = $inputConverted.'VM FQDN'
    Description = $inputConverted.'Description'
    Location = $inputConverted.'Location'.Location
    Infrastructure = $inputConverted.Infrastructure
    Network = $inputConverted.'Virtual Network'.NetworkName
    StorageAccount = $inputConverted.'Virtual Network'.StorageAccount
    ResourceGroup = $inputConverted.'Virtual Network'.ResourceGroup
    VMOSType = $inputConverted.'VMOSType'.Platform
    VMTemplate = $inputConverted.'Virtual Machine Template'
    VMInstanceType = $inputConverted.'VM Instance Type'
    ActionDate = $inputConverted.'Deployment Date'
    EndDate = $inputConverted.'Expiry Date'
    TemplateName = $inputConverted.'TemplateName'
  }
}
  
```

#### 3.9.7.1 Triggering the Scheduled Runbook

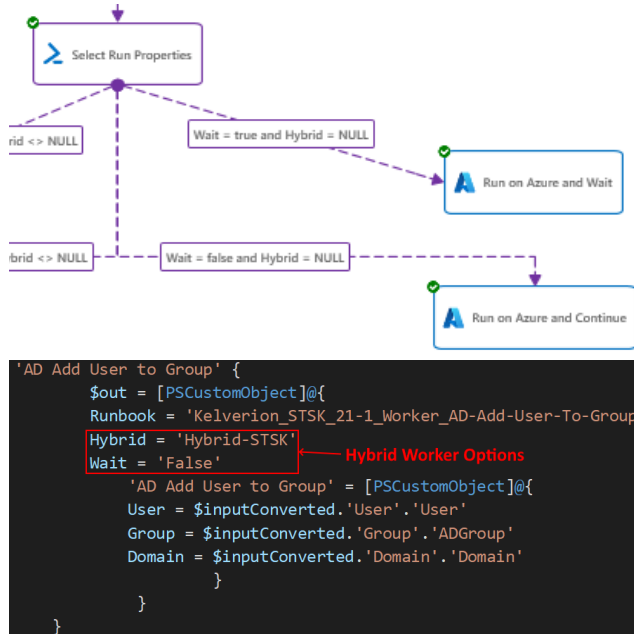
To trigger the scheduled runbook, you will need to build a Logic App that checks the dbo.RequestSchedule table

	_state	Request_ID	Runbook	StartDate	EndDate
1	New	928	Kelverion_VMPM_30-1-1_Worker_AzureDeployStandardVM	2021-10-14 00:00:00.000	2022-03-10 00:00:00.000

You will require the Logic App to launch based on the StartDate or EndDate. The table stores the runbook name to launch and the Request\_ID, that is the required input parameter.

### 3.9.8 Using Hybrid Workers

For on premise hypervisors (VMM-2016) you will be required to use a Hybrid Worker. Runbook Kelverion\_VMPM\_95-01\_ConvertToRequestData has the logic already set to allow a runbook to be called using a Hybrid Worker.



To configure this, you will need to adjust the Code activity as shown below.

If you leave the Hybrid option empty then the runbook will assume that it is to run in Azure.

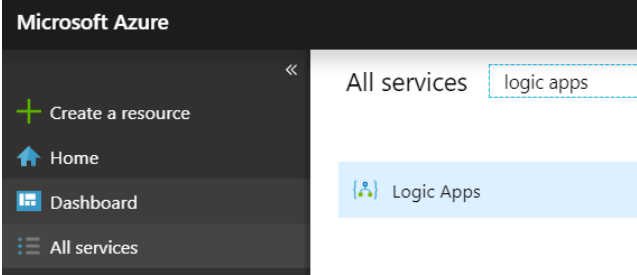
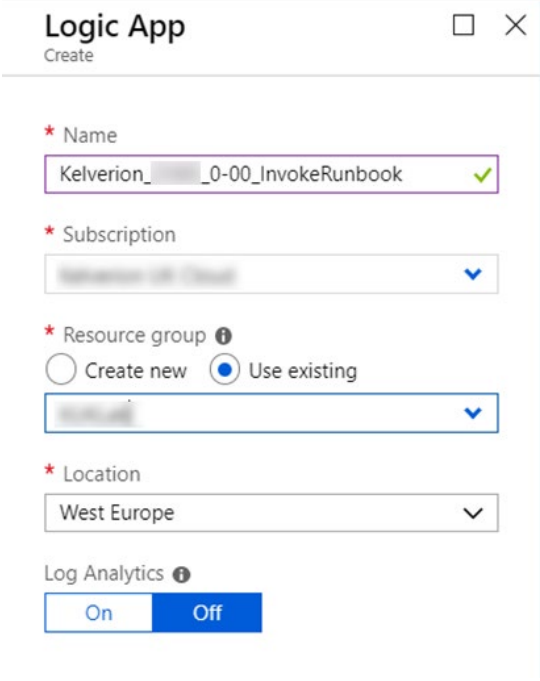
### 3.10 Logic Apps - Scheduling runbook execution

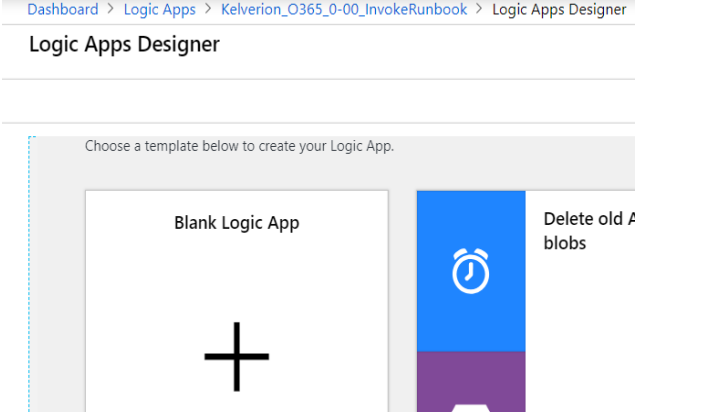
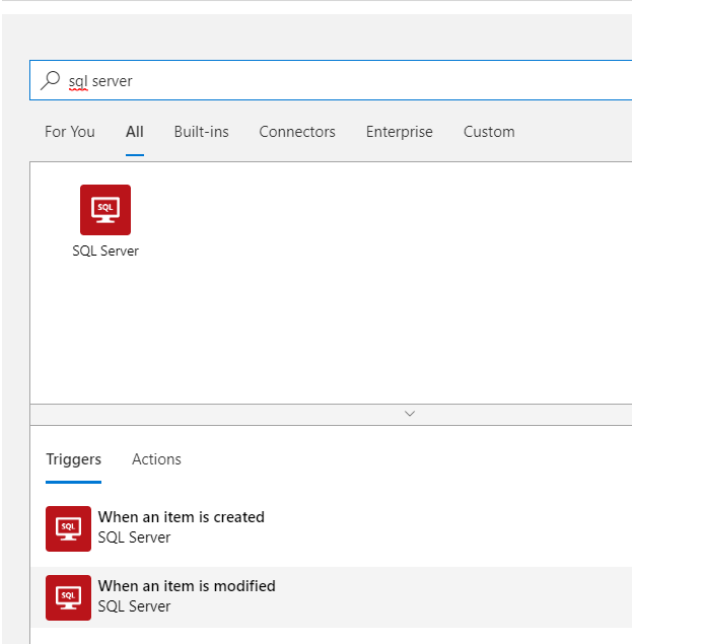
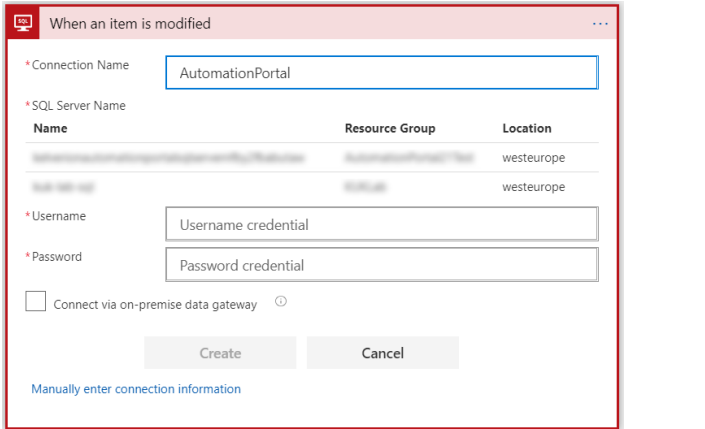
Once all the runbooks (and other assets) are in place and the runbooks have been tested you will need to schedule the 9X runbooks for repeated execution.

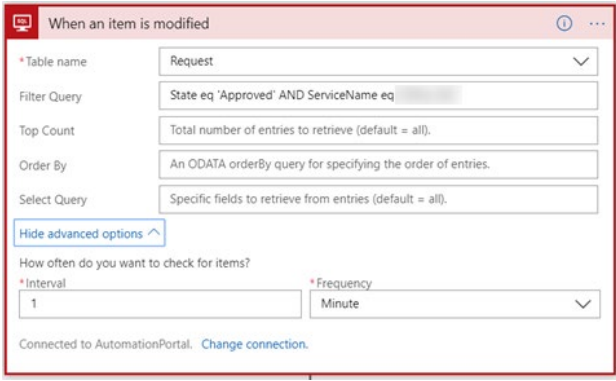
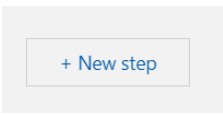
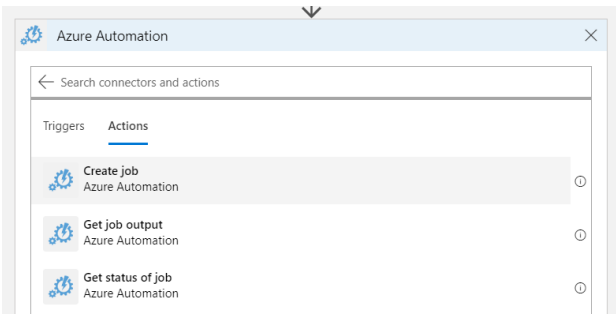
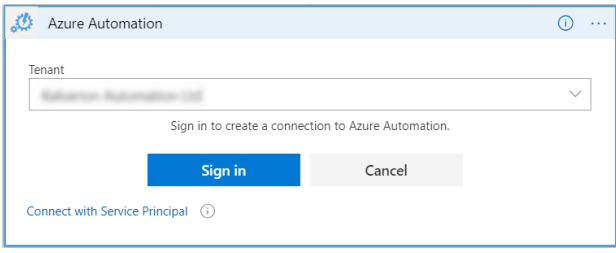
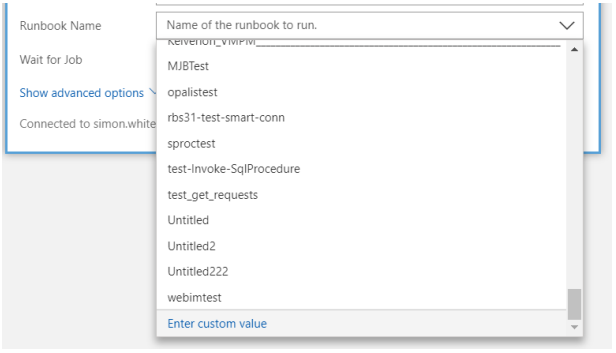
Two Logic Apps are required. One to gather the information from the required Service Desk application (in this case the Automation Portal) and one for returning the status of the runbook to the Service Desk application.

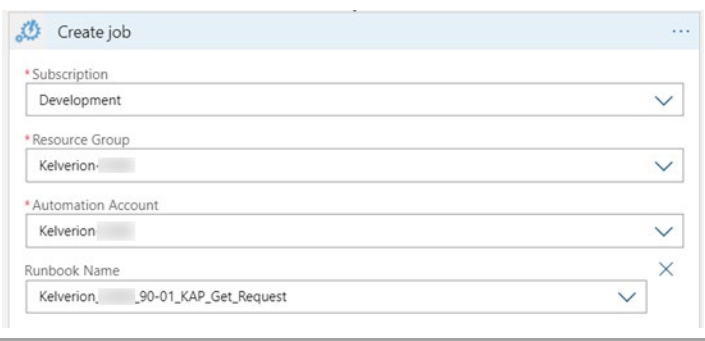
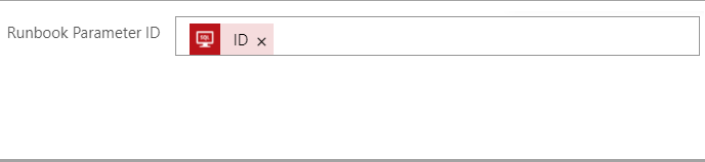
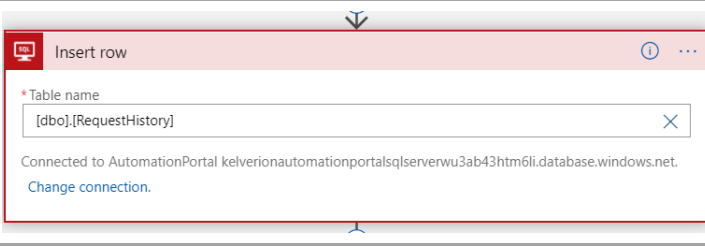
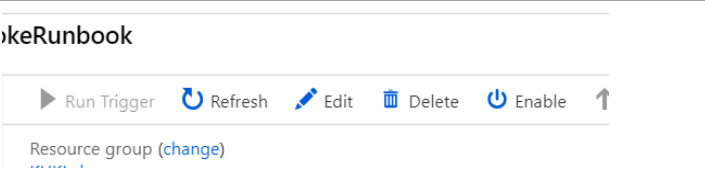
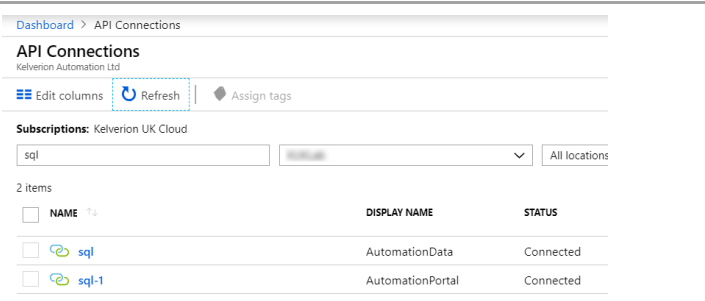
#### 3.10.1 Gather Request Logic App

This Logic App will call the runbook Kolverion\_VMPM\_90-01\_KAP\_Get\_Request.

Step	
Login to the Azure Portal and go to "Logic Apps"	
<p>Create a New Logic App and name it appropriately for this application. e.g. Kolverion_VMPM_0-00_InvokeRunbook</p> <p>Add it to the same Resource Group that you have deployed the runbooks too</p>	

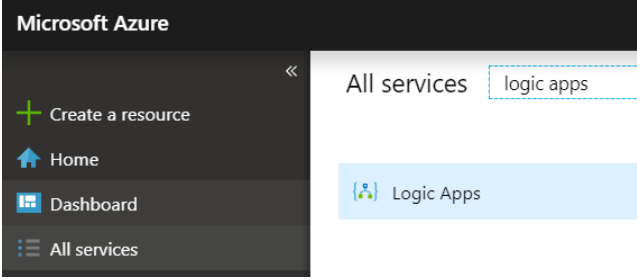
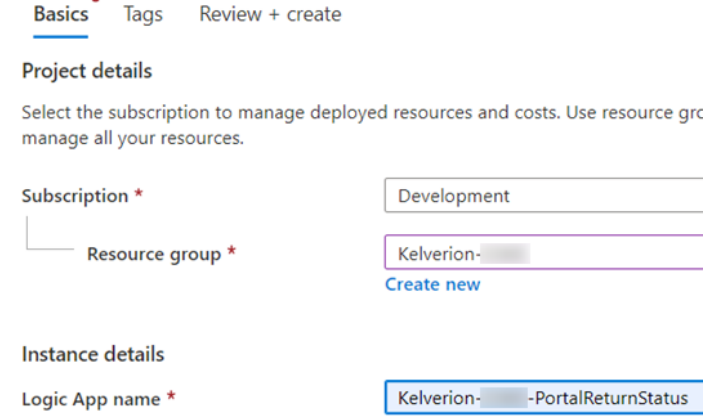
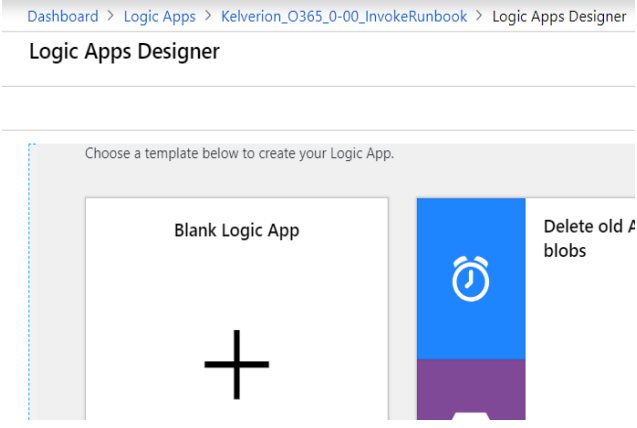
<p>Open the newly created Logic App and select “Blank Logic App”</p>										
<p>Search for “Sql Server” and select the Trigger “When an item is modified”</p>										
<p>If you do not already have a connection (API Connection) then you will need to set one up for your Automation Portal database.</p> <p>Enter the appropriate connection details for the database.</p>	 <table border="1" data-bbox="690 1480 1258 1564"> <thead> <tr> <th>Name</th> <th>Resource Group</th> <th>Location</th> </tr> </thead> <tbody> <tr> <td>AutomationPortalDB</td> <td>AutomationPortalRG</td> <td>westeurope</td> </tr> <tr> <td>AutomationPortalDB</td> <td>AutomationPortalRG</td> <td>westeurope</td> </tr> </tbody> </table>	Name	Resource Group	Location	AutomationPortalDB	AutomationPortalRG	westeurope	AutomationPortalDB	AutomationPortalRG	westeurope
Name	Resource Group	Location								
AutomationPortalDB	AutomationPortalRG	westeurope								
AutomationPortalDB	AutomationPortalRG	westeurope								

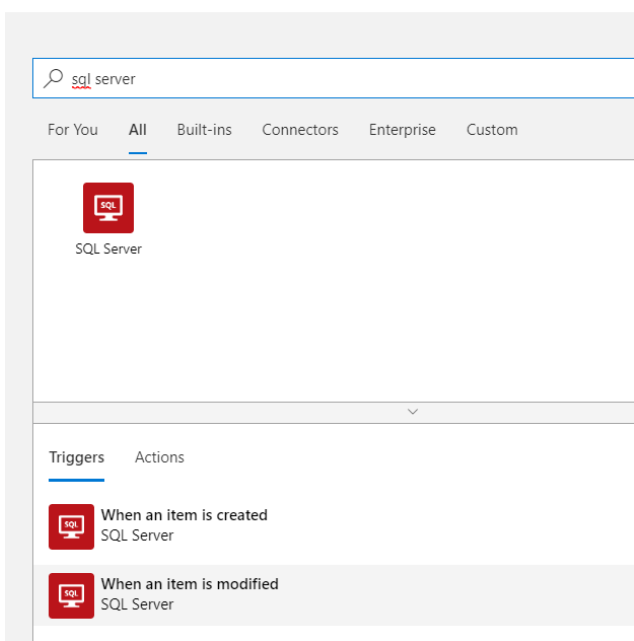
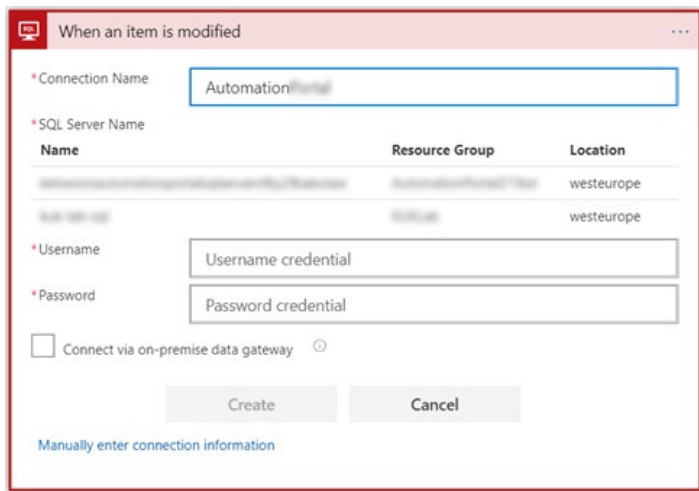
<p>Configure the activity as shown. Table = Request</p> <p>Filter Query: <b>State eq 'Approved' AND ServiceName eq 'VM Provisioning and Management - Azure Automation'</b></p> <p>Interval = 1 Minute</p>	
<p>Click on New Step to add another activity</p>	
<p>Search for “Azure automation” and select the Action “Create Job”</p>	
<p>If you have not done so before, you will need to create an API Connection to your tenant. Use the required Azure login details to make the connection.</p>	
<p>Enter the appropriate: Subscription \ Resource Group \ Automation Account For Runbook Name, scroll to the bottom and select “Custom Value”</p>	

<p>Select the Runbook from the drop down list as:</p> <p><b>Kelverion_VMPM_90-01_KAP_Get_Request</b></p>	
<p>Add Runbook Parameters. Where ID is from the List of Items from the previous activity</p>	
<p><b>Automation Portal Only:</b> Add a new activity to write a line to the RequestHistory table</p>	
<p>Ensure the Logic App is active by clicking on <b>Enable</b></p>	
<p>If you need to change connection details, you should be able to find your connection information in “API Connections”</p>	

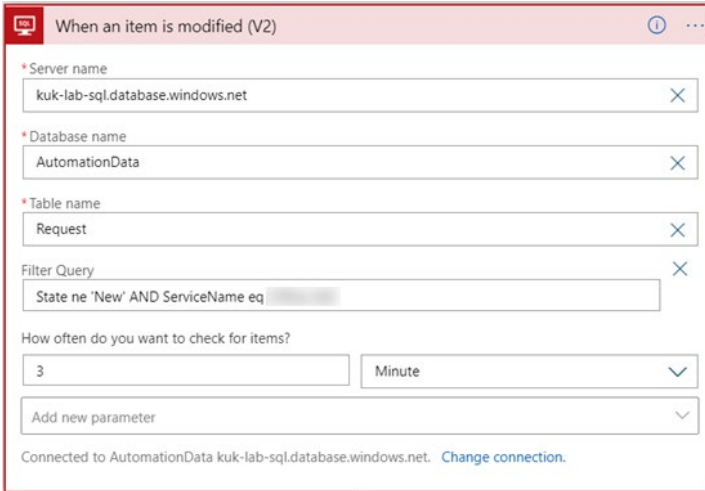
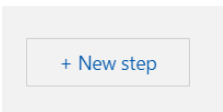
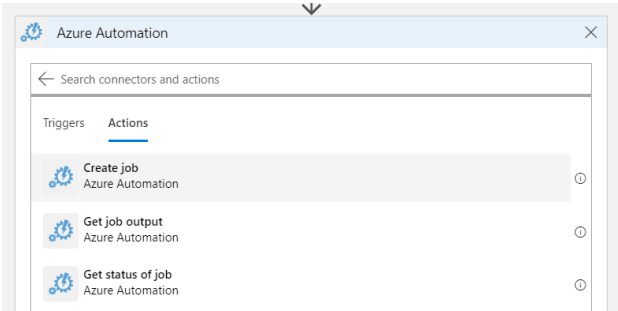
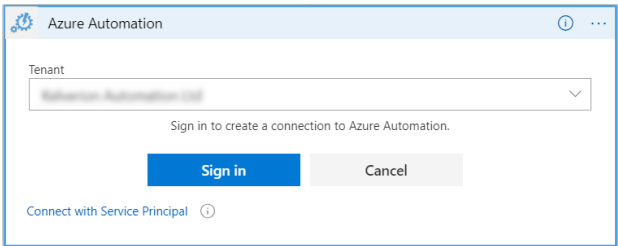
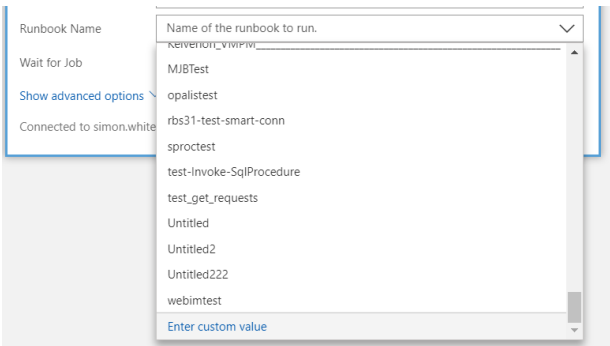
### 3.10.2 Return Status Logic App

This Logic App will call the runbook Kelverion\_VMPM\_97-01\_KAP\_Return\_Status.

Step	
Login to the Azure Portal and go to "Logic Apps"	
<p>Create a New Logic App and name it appropriately for this application. e.g. Kelverion-VMPM-PortalReturnStatus</p> <p>Add it to the same Resource Group that you have deployed the runbooks too</p>	
Open the newly created Logic App and select "Blank Logic App"	

<p>Search for “Sql Server” and select the Trigger “When an item is modified”</p>	
<p>If you do not already have a connection (API Connection) then you will need to set one up for your AutomationData database.</p> <p>Enter the appropriate connection details for the database.</p>	



<p>Configure the activity as shown. Table = Request</p> <p>Filter Query: <b>State ne 'New' AND ServiceName eq 'VM Provisioning and Management - Azure Automation'</b></p> <p>Interval = 1 Minute</p>	
<p>Click on New Step to add another activity</p>	
<p>Search for "Azure automation" and select the Action "Create Job"</p>	
<p>If you have not done so before, you will need to create an API Connection to your tenant. Use the required Azure login details to make the connection.</p>	
<p>Enter the appropriate: Subscription \ Resource Group \ Automation Account For Runbook Name, scroll to the bottom and select "Custom Value"</p>	

<p>Ensure that the activity is pointing at the correct Runbook:</p> <p><b>Kelverion_VMPM_97-01_KAP_Return_Status</b></p>										
<p>Add Runbook Parameters. Where ID is from the List of Items from the previous activity.</p>										
<p>Ensure the Logic App is active by clicking on <b>Enable</b></p>										
<p>If you need to change connection details, you should be able to find your connection information in “API Connections”</p>	<table><thead><tr><th>NAME</th><th>DISPLAY NAME</th><th>STATUS</th></tr></thead><tbody><tr><td>sql</td><td>AutomationData</td><td>Connected</td></tr><tr><td>sql-1</td><td>AutomationPortal</td><td>Connected</td></tr></tbody></table>	NAME	DISPLAY NAME	STATUS	sql	AutomationData	Connected	sql-1	AutomationPortal	Connected
NAME	DISPLAY NAME	STATUS								
sql	AutomationData	Connected								
sql-1	AutomationPortal	Connected								

### 3.11 Testing

The following steps allow you to prove that all the components have been configured correctly. Testing the components should take place before the runbooks are scheduled for repeated execution.

1. Using the Automation Portal, create a request for Deploy VM
2. Start the runbook Kolverion\_VMPM\_90-01\_KAP\_Get\_Request using the Azure Portal
3. Monitor the runbook to ensure it completes without errors
4. Check the resource group has created the virtual machine
5. Check the status of the request using the Automation portal

Kelverion UK Limited  
1 Lea Business Park,  
Lower Luton Road,  
Harpenden,  
Hertfordshire  
AL5 5EQ  
Email: [info@kelverion.com](mailto:info@kelverion.com)  
Web: [www.kelverion.com](http://www.kelverion.com)