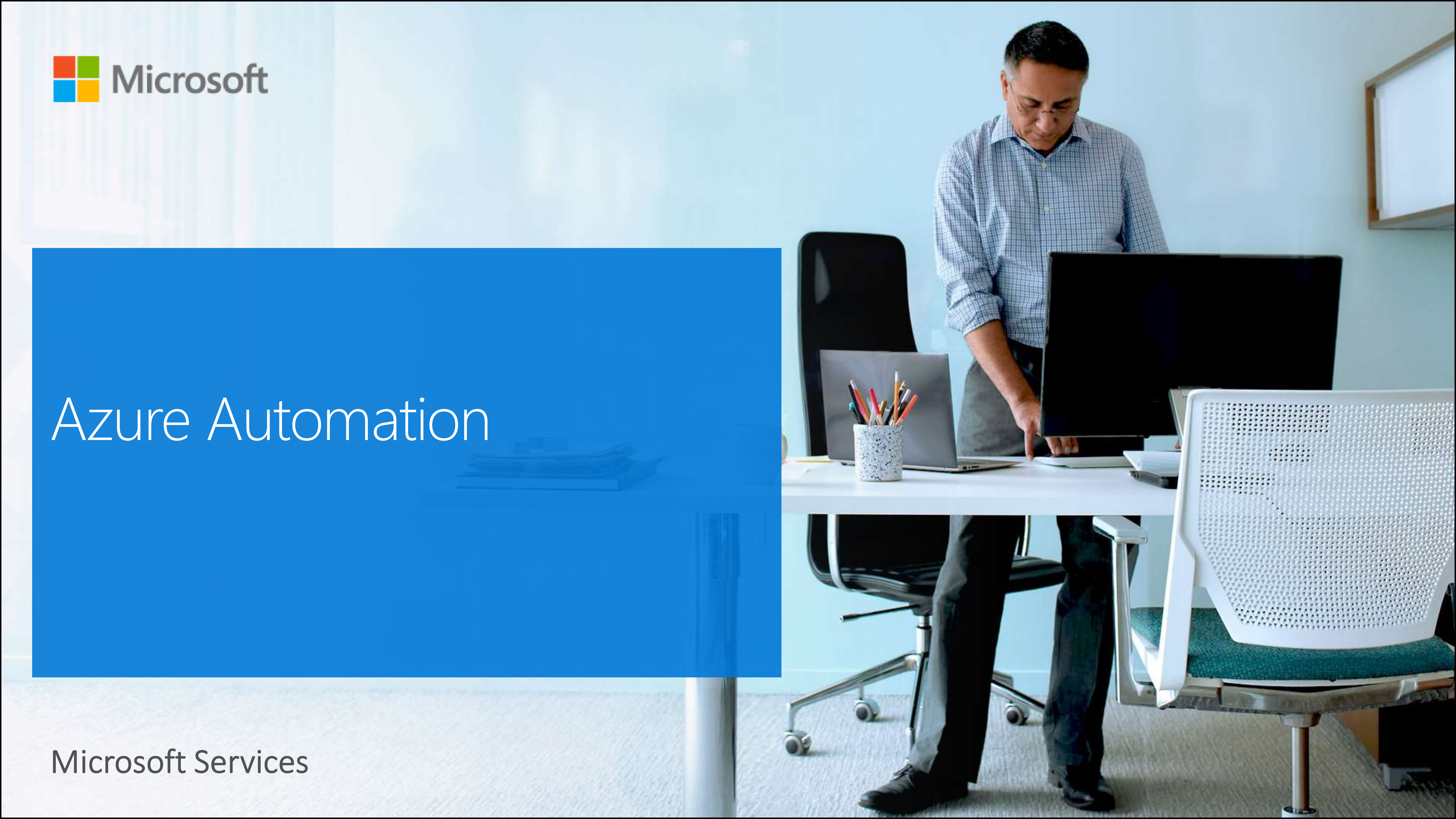
Microsoft



Azure Automation

Microsoft Services

Agenda

- Shared Resources (Assets)
- Runbooks
- Author Runbooks
- Desired State Configuration
- Monitoring & Troubleshooting

Automation Today

Automation today is often manual, error-prone, and frequently repeated tasks.

Some of the challenges with automation today are:

- Unable to manage enterprise wide automation tasks from a single location.
- Inconsistent credential usage for automated tasks.
- Command line only option for scheduled task management on remote servers.
- Laborious to monitor automated tasks.

What is Azure Automation

Azure Automation delivers a cloud-based automation and configuration service that provides:

- Consistent management across your Azure and non-Azure environments.
- Consists of process automation, update management, and configuration features.
- Provides complete control during deployment, operations, and decommissioning of workloads and resources.

Azure Automation capabilities

Process Automation

Orchestrate processes using graphical, PowerShell, and Python runbooks

Configuration Management

Collect inventory
Track changes
Configure desired state

Update Management

Assess compliance
Schedule update installation

Shared capabilities

Role based access control
Secure, global store for variables, credentials, certificates, connections
Flexible scheduling
Shared modules
Source control support
Auditing
Tags

Heterogenous

Windows & Linux
Azure and on-premises

Benefits of Azure Automation

- Automate time consuming, error prone operational tasks across Azure & 3rd party systems.
- Manage enterprise wide automation tasks from a single location.
- Ensure consistent credential usage for automated tasks.
- Improved monitoring of automated tasks.
- Lower operational costs by reduced management time.
- Integrate with existing systems.

Pricing

Current pricing posted at: http://azure.microsoft.com/en-us/pricing/details/automation/

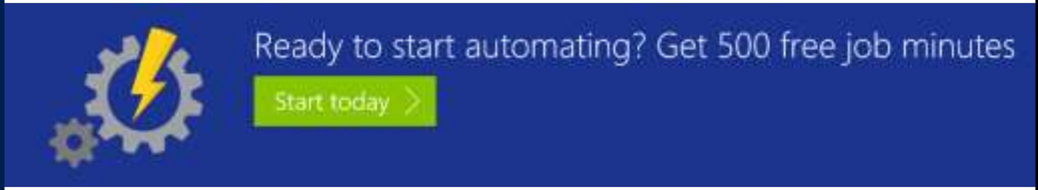
Process Automation

- Billed by the minute according to actual run time of your jobs
- Free tier gives you 500 minutes of job run time per month
- Basic tier costs $0.12 per hour and gives you unlimited job run time

Desired State Configuration (DSC)

- Free tier has 5 nodes
- Basic tier costs $6 a node per month & number of nodes is unlimited

SLA 99.9% guarantees that planned jobs start within 30 minutes of their planned start times.



Microsoft

Shared Resources (Assets)

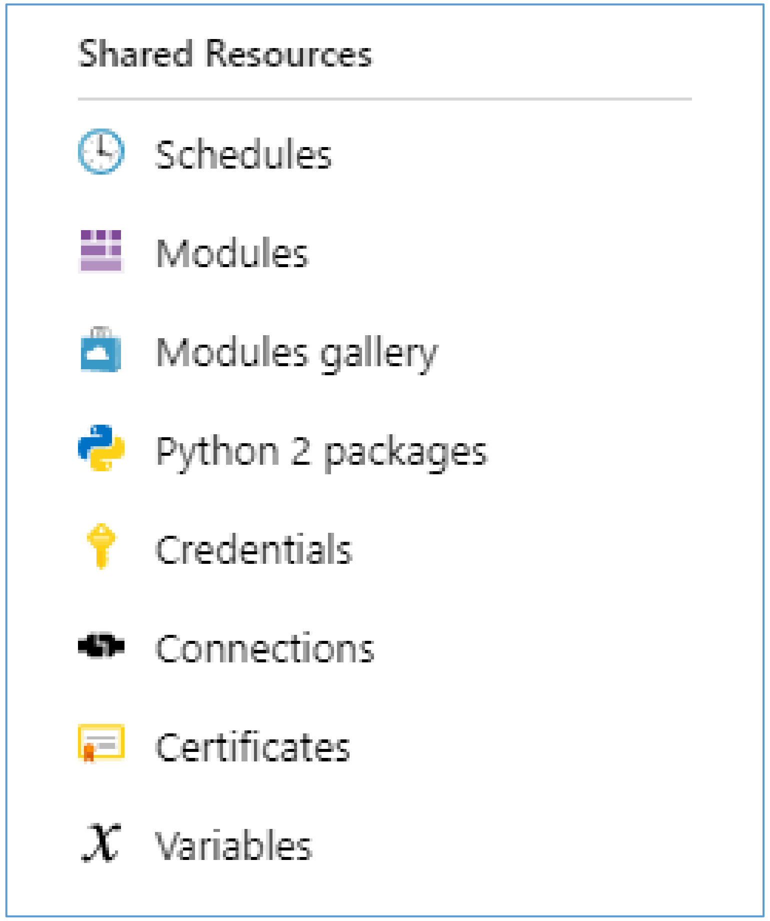
Microsoft Services

Shared Resources

- Azure Automation consists of a set of shared resources that make it easier to automate and configure your environments at scale.
- Assets are encrypted and stored in Azure Automation using a unique key that is generated for each Automation account.
- The unique key is encrypted by a master certificate and stored in Azure Automation.

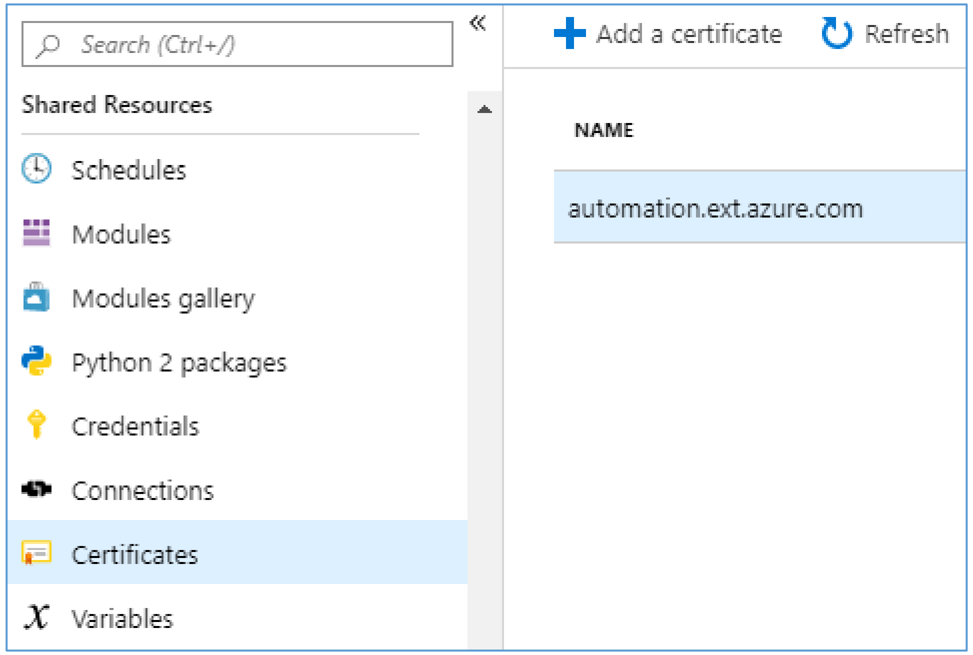
There are eight different types of Shared Resources:

Schedules	Modules	Modules gallery	Python 2 packages
Credentials	Connections	Certificates	Variables



Certificates

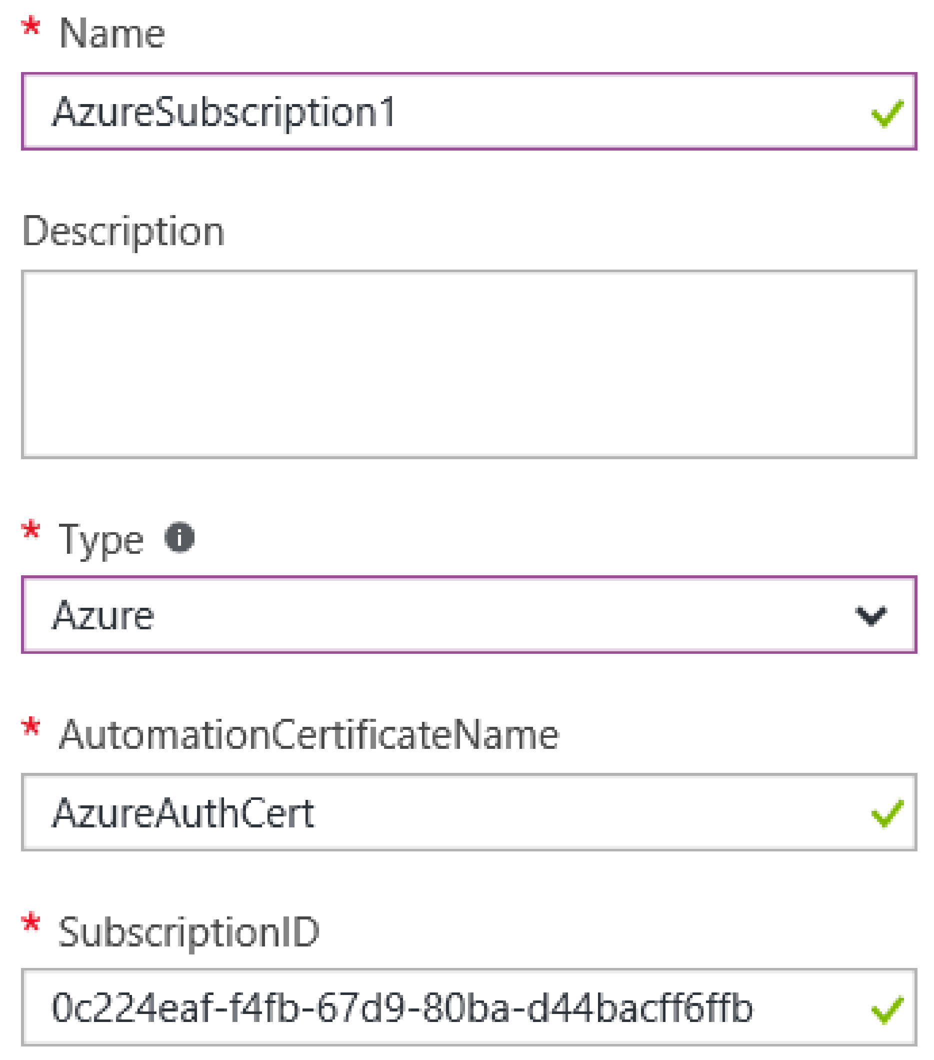
- Certificates can be securely stored in Azure Automation so they can be used by Runbooks or DSC configurations to encrypt sensitive information.
- This allows you to create Runbooks and DSC configurations that use certificates for authentication or adds them to Azure or third party resources.
- Import and export both .cer and .pfx files to Azure Automation using PowerShell or the Azure portal.
- Certificate must be marked as exportable during import if it is to be exported at another time.



NAME
automation.ext.azure.com

Connections

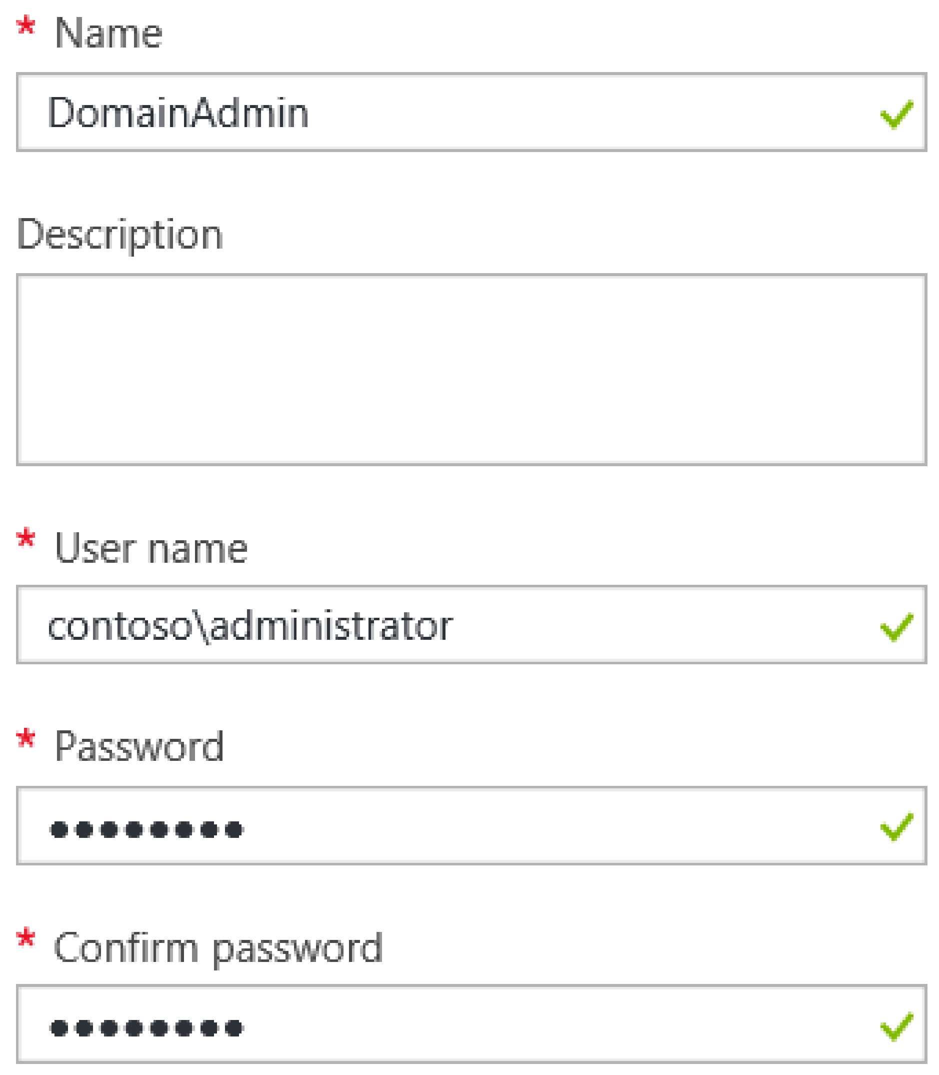
- Connections define the information required to connect to a service or application from a Runbook or DSC configuration.
- May include information required for authentication such as a username and password in addition to connection information such as a URL or a port.
- Connection types are defined by the PowerShell integration modules that are installed e.g. installing the Azure PowerShell module allows you to create a connection to an Azure subscription.
- Connections can be created using PowerShell or the Azure portal.



* Name	AzureSubscription1 ✓
Description	
* Type	Azure
* AutomationCertificateName	AzureAuthCert ✓
* SubscriptionID	0c224eaf-f4fb-67d9-80ba-d44bacff6ffb ✓

Credentials

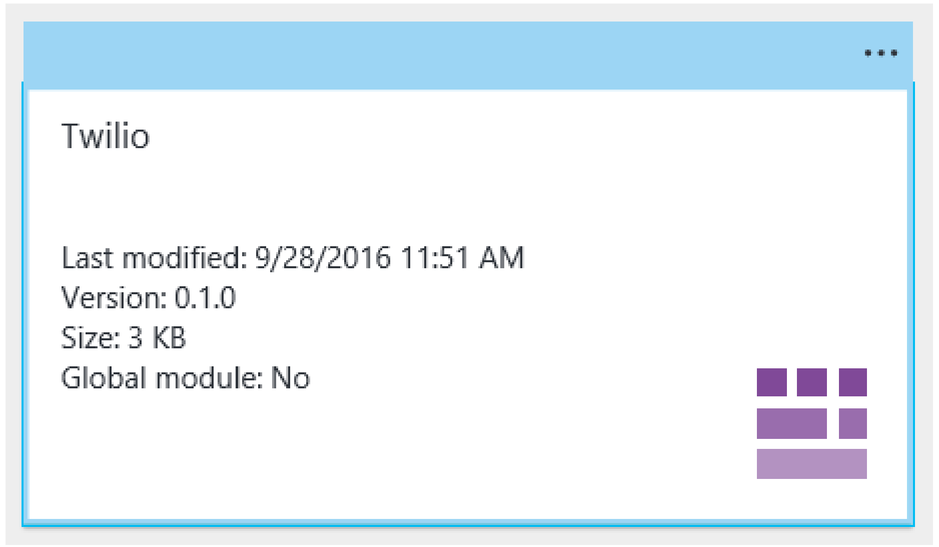
- An Automation credential asset holds a PSCredential object which contains security credentials such as a username and password.
- Simplifies Runbook and DSC configurations that may use cmdlets that accept a PSCredential object for authentication.
- Credentials can be created using PowerShell or the Azure portal.



* Name	DomainAdmin ✓
Description	
* User name	contoso\administrator ✓
* Password	●●●●●●●●● ✓
* Confirm password	●●●●●●●●● ✓

Integration Modules

- An Integration Module is a PowerShell module that is imported into an Azure Automation account.
- Cmdlets from integration modules can be used in Runbooks or DSC configurations.
- Import PowerShell modules from a local machine or from the Automation Integration Modules gallery.
- Imported modules from a local machine must be compressed with a .zip extension and be smaller than 100MB.
- Integration modules are imported via the Azure portal.

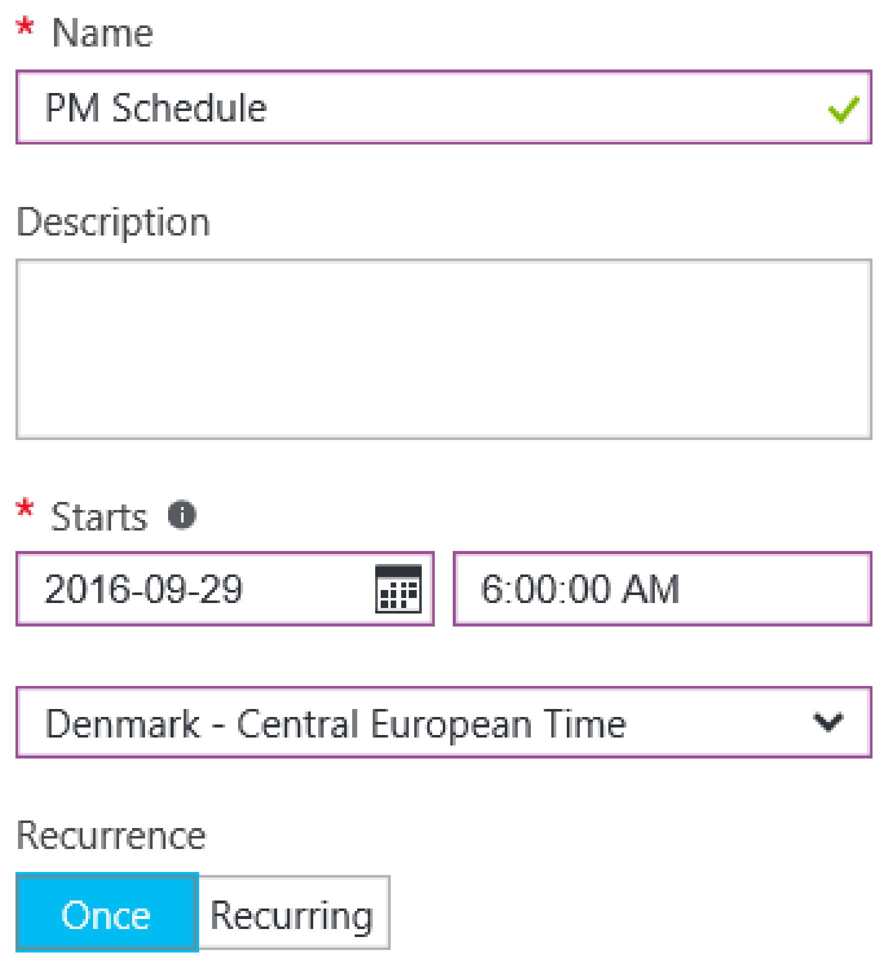


Twilio

Last modified: 9/28/2016 11:51 AM
Version: 0.1.0
Size: 3 KB
Global module: No

Schedules

- Azure Automation Schedules are used to schedule Runbooks to run automatically.
- Could be either a single date and time or it could be a recurring hourly, daily, weekly, or monthly schedule to start the Runbook multiple times.
- Schedules can be created using PowerShell or the Azure portal.
- Schedules do not currently support Azure Automation DSC configurations.



* Name: PM Schedule

Description:

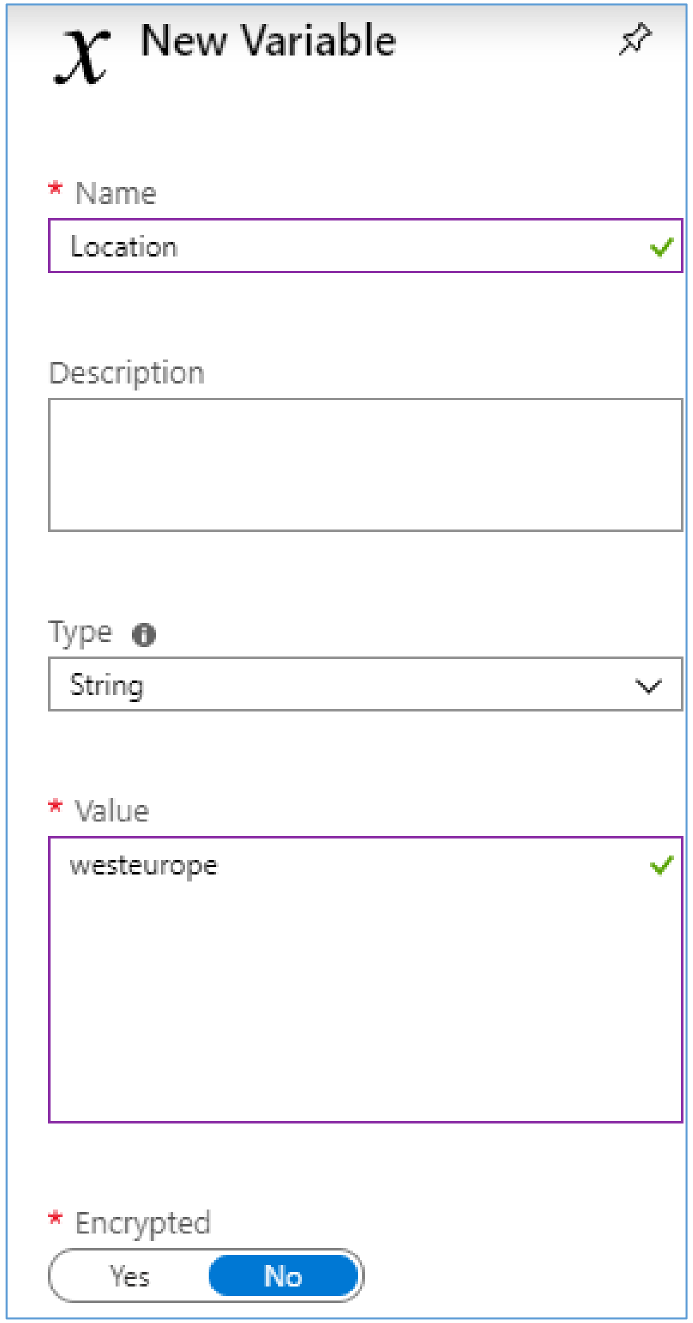
* Starts: 2016-09-29 6:00:00 AM

Denmark - Central European Time

Recurrence: [✓ Once] Recurring

Variables

- Variable assets are values that are available to all Runbooks and DSC configurations in your automation account.
- Can be created, modified, and retrieved from the Azure portal, PowerShell, and from within a Runbook or DSC configuration.
- Automation variables are useful for:
  - Sharing a value between multiple Runbooks or DSC configurations.
  - Sharing a value between multiple jobs from the same Runbook or DSC configuration.
  - Managing a value from the portal or from a PowerShell command line that is used by Runbooks or DSC configurations.
- Variables can be stored encrypted.



New Variable

* Name: Location

Description:

Type: String

* Value: westeurope

* Encrypted: Yes [✓ No]

Python 2 packages

- Azure Automation allows you to run Python 2 runbooks on Azure and on Linux Hybrid Runbook Workers.
- This includes the ability to upload Python 2 packages to be imported by Python 2 runbooks running serverlessly in Azure.
- It also includes the ability to use other Automation resources, such as schedules, variables, connections, and credentials.
- Supported package types include Python wheel packages and packages that have been source distribution compressed in the *.tar.gz format.

Demo: Automation Assets

Microsoft

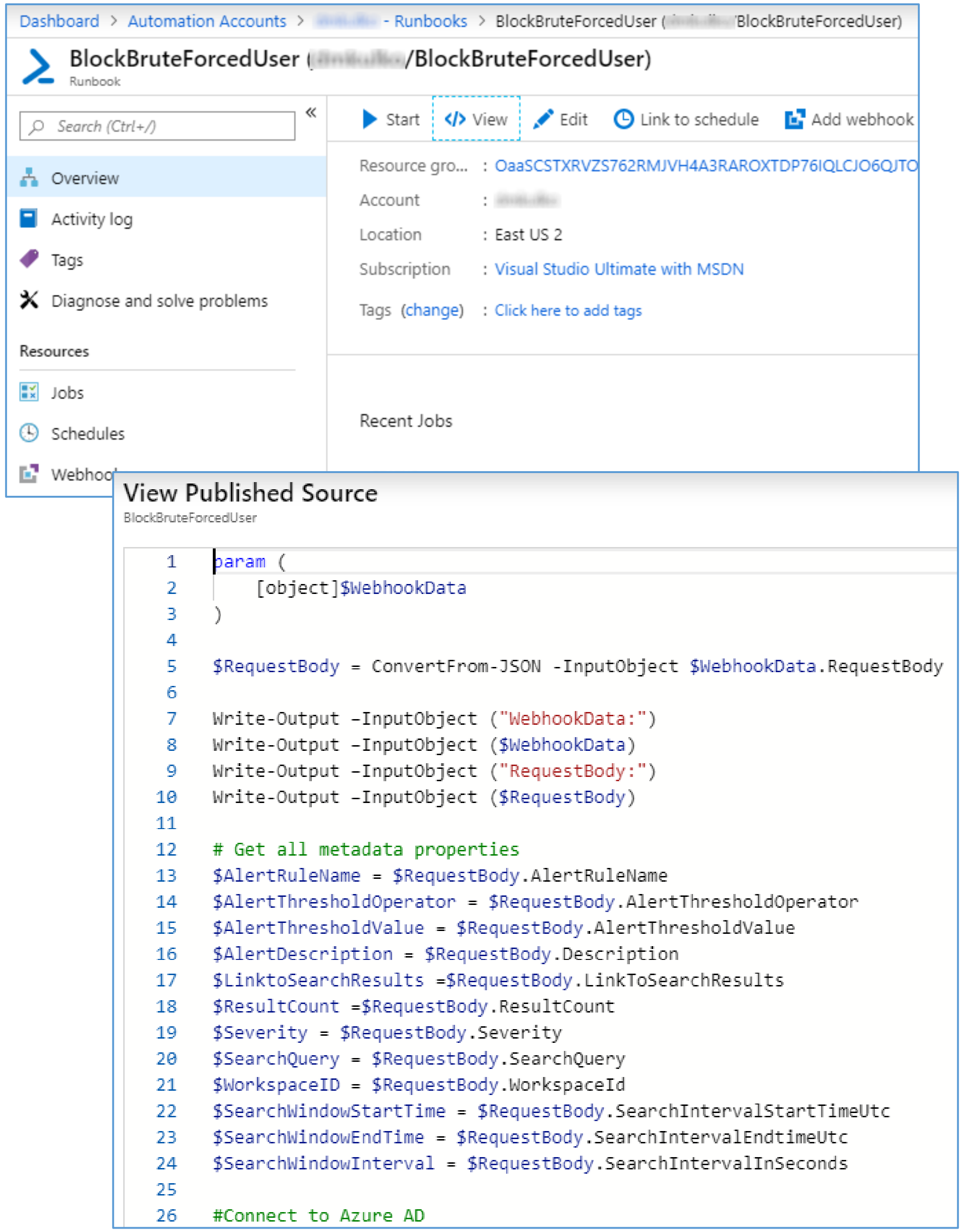
Runbooks

Microsoft Services

What is a Runbook

In Azure Automation, a Runbook is a file that contains a set of procedures and operations.

- Runbooks are used as input for the Azure Automation service to process.
- The contents of a Runbook file is written using PowerShell or PowerShell Workflow commands.
- PowerShell Workflow is a PowerShell extension that allows you to run a PowerShell script on multiple devices in parallel with added functionality such as checkpoints, suspend & restart.


Dashboard > Automation Accounts > [redacted] - Runbooks > BlockBruteForcedUser ([redacted]/BlockBruteForcedUser)
BlockBruteForcedUser ([redacted]/BlockBruteForcedUser)
Runbook
Search (Ctrl+/)
Start
View
Edit
Link to schedule
Add webhook
Overview
Activity log
Tags
Diagnose and solve problems
Resources
Jobs
Schedules
Webhoo
Recent Jobs

Resource gro... : OaaSCSTXRVZS762RMJVH4A3RAROXTDP76IQLCJO6QJTO
Account : [redacted]
Location : East US 2
Subscription : Visual Studio Ultimate with MSDN
Tags (change) : Click here to add tags

View Published Source
BlockBruteForcedUser

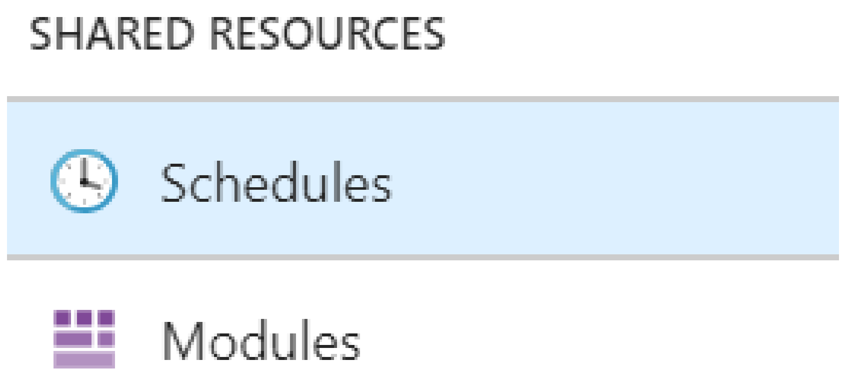
```
param (
    [object]$WebhookData
)

$RequestBody = ConvertFrom-JSON -InputObject $WebhookData.RequestBody

Write-Output -InputObject ("WebhookData:")
Write-Output -InputObject ($WebhookData)
Write-Output -InputObject ("RequestBody:")
Write-Output -InputObject ($RequestBody)

# Get all metadata properties
$AlertRuleName = $RequestBody.AlertRuleName
$AlertThresholdOperator = $RequestBody.AlertThresholdOperator
$AlertThresholdValue = $RequestBody.AlertThresholdValue
$AlertDescription = $RequestBody.Description
$LinktoSearchResults =$RequestBody.LinkToSearchResults
$ResultCount =$RequestBody.ResultCount
$Severity = $RequestBody.Severity
$SearchQuery = $RequestBody.SearchQuery
$WorkspaceID = $RequestBody.WorkspaceId
$SearchWindowStartTime = $RequestBody.SearchIntervalStartTimeUtc
$SearchWindowEndTime = $RequestBody.SearchIntervalEndtimeUtc
$SearchWindowInterval = $RequestBody.SearchIntervalInSeconds

#Connect to Azure AD
```

Start a Runbook

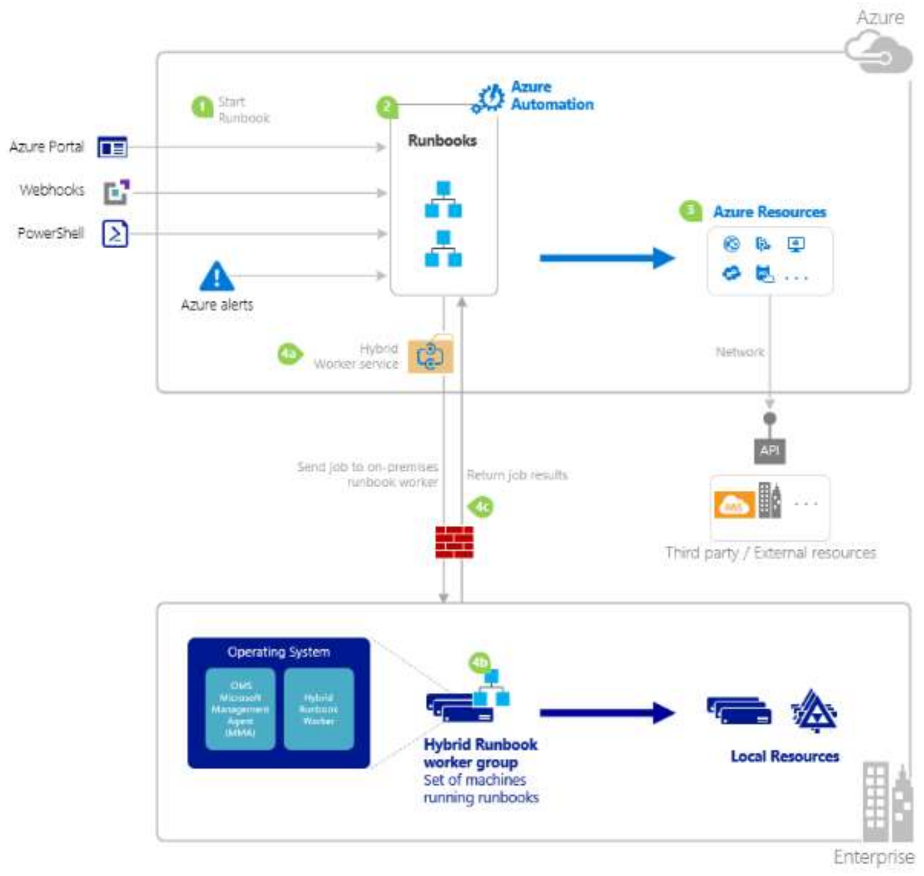
There are 7 different ways in which you can start a Runbook.

1. Azure Portal
2. PowerShell
3. Azure Automation API
4. Webhooks
5. Respond to Azure Alert
6. Schedule
7. From Another Runbook

The most common methods in use today are Schedule, PowerShell, the Azure Portal and Webhooks.



Runbook Processing



1. An actor starts a runbook

2. Azure Automation notes that the runbook should be started

3. Cloud resources – Runbook acts on local Azure resources or other external resources reachable via the network

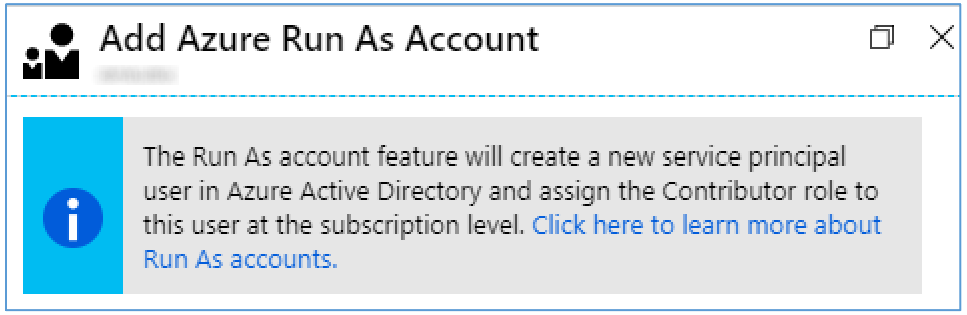
4a. On-Premises – Hybrid runbook group sends the runbook to an on-premises machine to run

4b. Runbook acts on its local networked resources

4c. Job results are returned from on-premises

Runbook Authentication

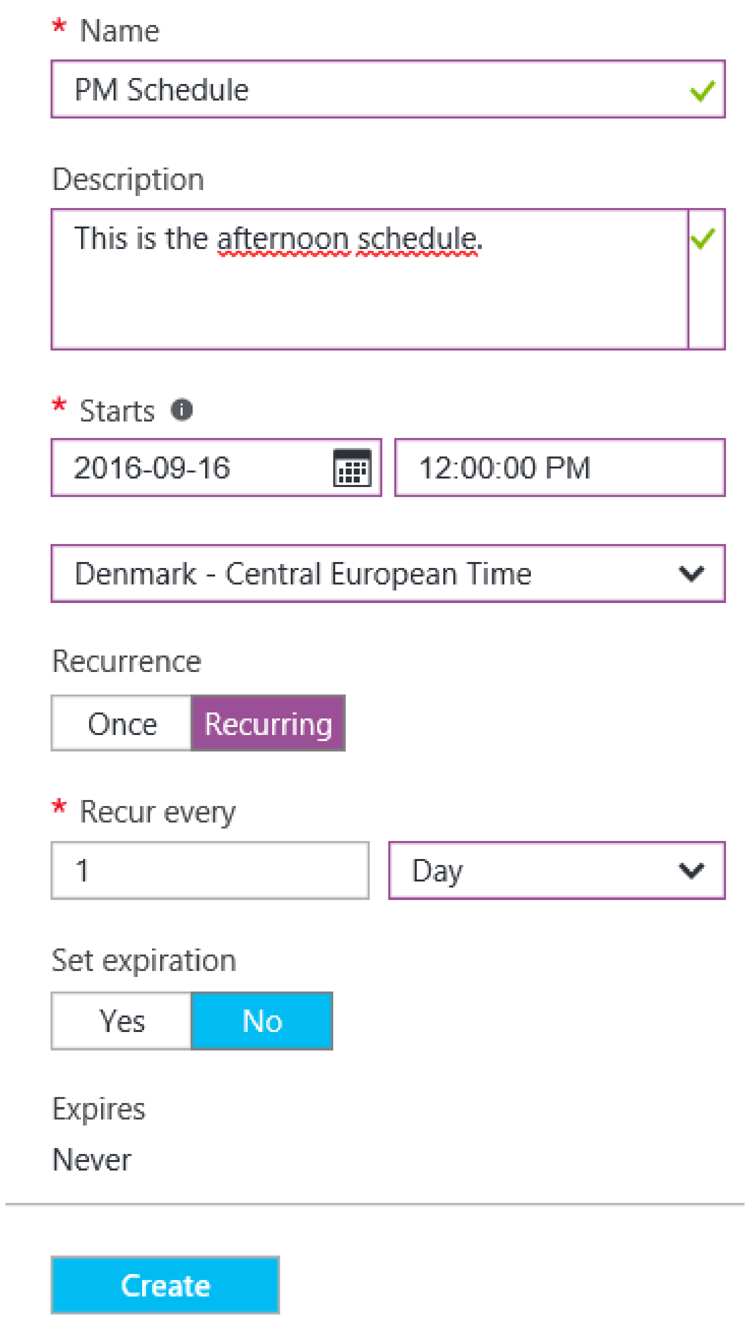
- Runbooks are authenticated using a Run As account, this allows the Runbook to execute it's tasks under this accounts security context.
- A Run As account can be created during the creation of an Azure Automation account or an existing account can be added later using PowerShell.
- The Run As account that is created during the creation of an Azure Automation account is granted the Contributor Role for the Azure subscription.
- Existing accounts that are used must be granted the appropriate permissions in order for the Runbook to complete its tasks.


Add Azure Run As Account

The Run As account feature will create a new service principal user in Azure Active Directory and assign the Contributor role to this user at the subscription level. Click here to learn more about Run As accounts.

Start a Runbook using a Schedule

- Automatically start a Runbook on an hourly, daily, or weekly schedule.
- Manipulate the schedule through the Azure portal, PowerShell cmdlets, or Azure API.
- Provide parameter values to be used with the schedule.



* Name
PM Schedule ✓

Description
This is the afternoon schedule. ✓

* Starts
2016-09-16 | 12:00:00 PM

Denmark - Central European Time

Recurrence
Once [✓ Recurring]

* Recur every
1 | Day

Set expiration
Yes [✓ No]

Expires
Never

Start a Runbook using PowerShell

- Runbooks can be called from a command line with Windows PowerShell cmdlets.
- The call can be included in an automated solution with multiple steps.
- The request is authenticated with a certificate or OAuth user principal / service principal.
- Provide simple and complex parameter values to start the Runbook.
- Track job state.
- Client required to support PowerShell cmdlets.

```
PS C:\> Start-AzureRmAutomationRunbook -Name AzureAutomationTutorialScript -ResourceGroupName cbauto -AutomationAccountName cbauto
```

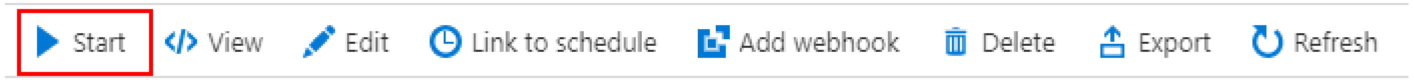
Start a Runbook using the Azure Portal

Is the simplest method with an interactive user interface.

Forms based to provide simple parameter values.

Easily track job state.

Access authenticated with Azure logon.



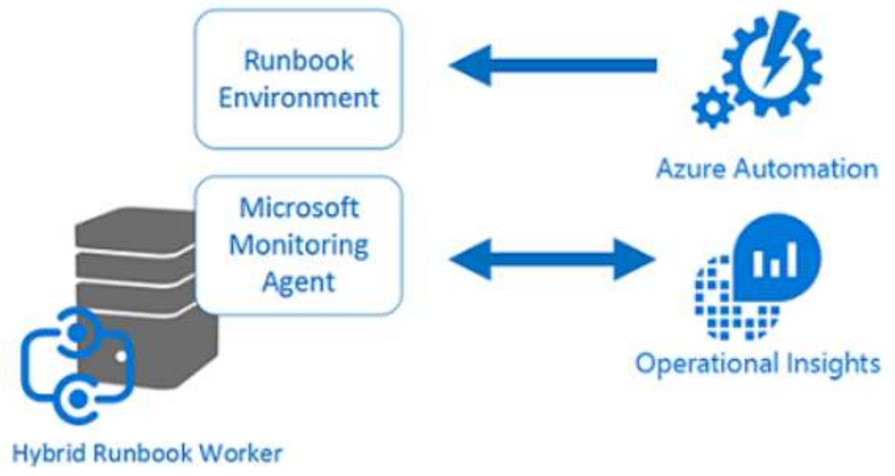
Start a Runbook using Webhooks

- A Webhook is a HTTP POST request that is sent to a specific URL, on receiving the POST request, some action is taken i.e. start a Runbook.
- The URL is generated during the creation of a Webhook and includes a built in security token that is used to authenticate the request when it is received by the Azure Automation service.
- The Webhook URL must be specified in the application that will be making the request.
- You cannot specify a custom URL and the URL expiration date cannot be changed after the Webhook as been created.
- No ability to track job state through a Webhook URL.

```
https://s2events.azure-automation.net/webhooks?token=HiQsFa4HqNbrmAhbD4jxwvGbcWVkOwI5E%2bsIQ3rKmdk%3d
```

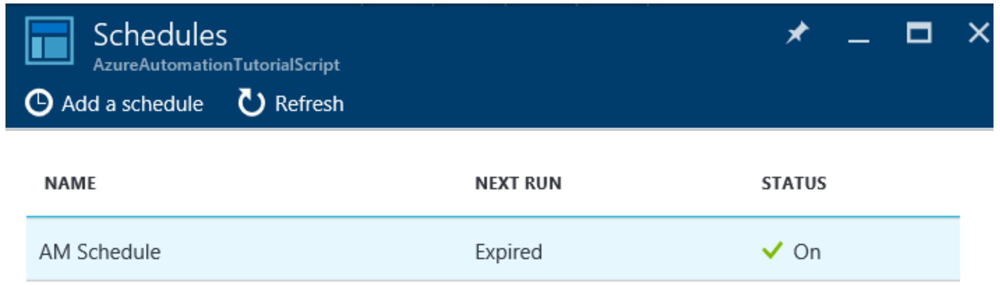
Hybrid Runbook Workers

- Hybrid Runbook Workers allow you to run Runbooks on machines located in your local data center in order to manage local resources.
- Runbooks are stored and managed in Azure Automation and downloaded by one or more designated on-premises machines via an agent.
- Outbound TCP 443 is required since the agent on the local computer initiates all communication with Azure Automation.
- Whitelist URL: *.azure-automation.net
- Use the RunOn option in the Azure portal to select the name of the Hybrid Runbook Worker to start a Runbook.



Schedule a Runbook

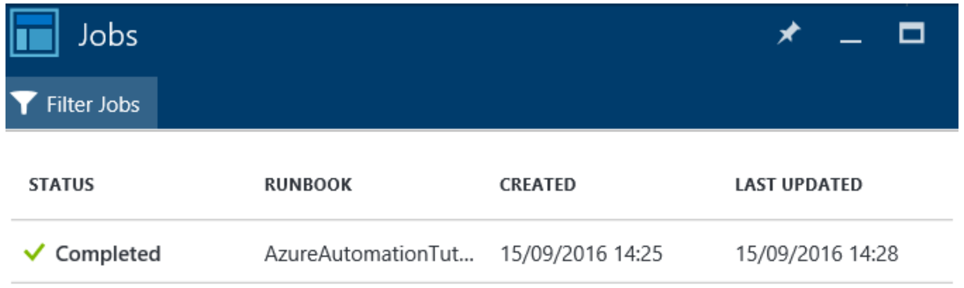
- To schedule a Runbook in Azure Automation, it must be linked to a pre-existing schedule.
- A schedule can be configured to either run once or on a reoccurring hourly or daily basis.
- A schedule can also be configured to run weekly, monthly, specific days of the week or days of the month, or a particular day of the month.
- A Runbook can be linked to multiple schedules, and a schedule can have multiple Runbooks linked to it.



NAME	NEXT RUN	STATUS
AM Schedule	Expired	✓ On

Runbook Job Status

- Runbook job status can be monitored using the Jobs tile in the Azure Portal or using Get-AzAutomationJob in PowerShell.
- Runbook jobs running for more than 3 hours will be temporarily unloaded and resumed from their last checkpoint, this is known as the Fair Share Limit.
- PowerShell Runbooks will be started from the beginning since they don't support checkpoints.
- Runbooks are terminated with a status of 'Failed, waiting for resources' if they are restarted 3 consecutive times.

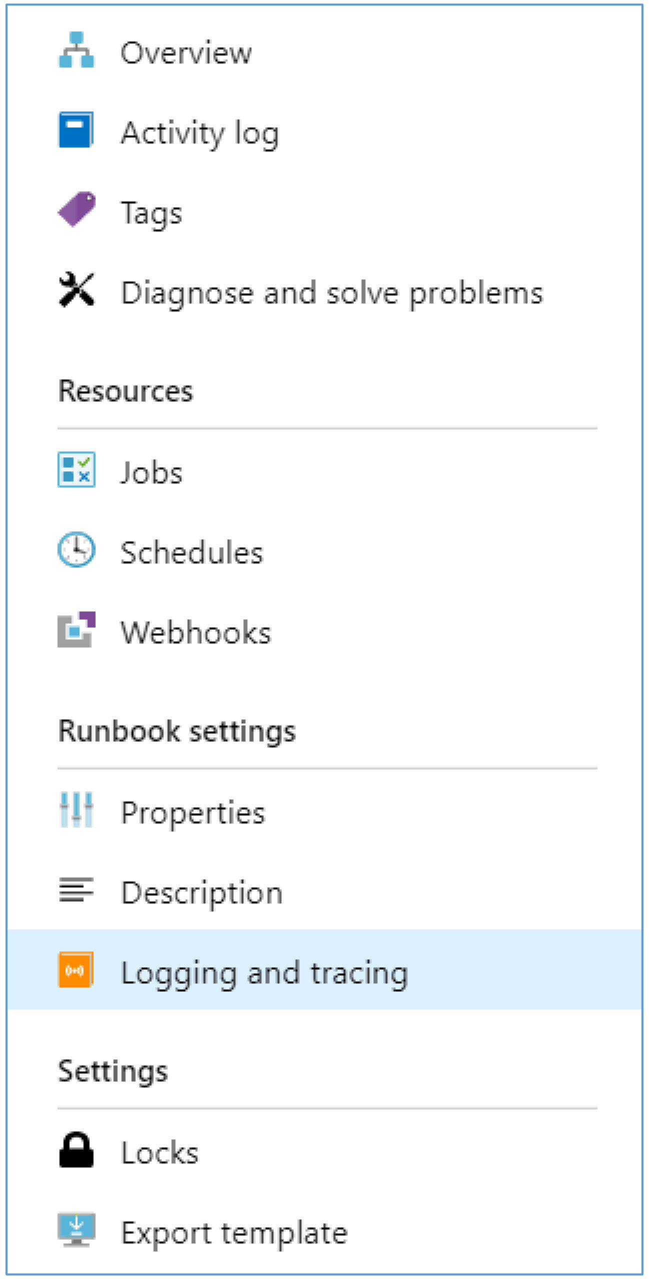


STATUS	RUNBOOK	CREATED	LAST UPDATED
✓ Completed	AzureAutomationTut...	15/09/2016 14:25	15/09/2016 14:28

Runbook Settings

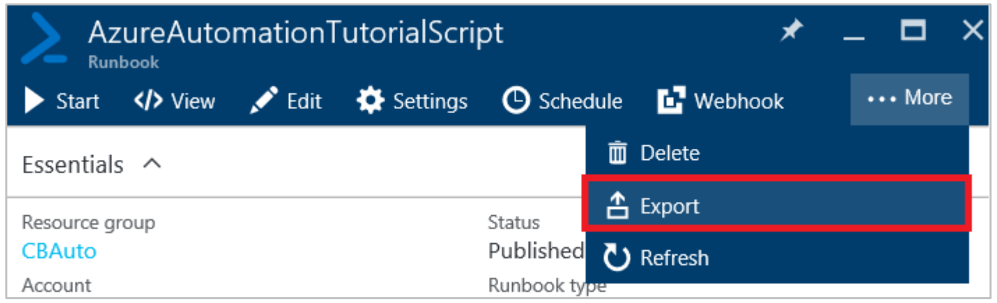
A Runbook in Azure Automation has multiple settings that help it to be identified and to change its logging behavior.

- Name & Description - You cannot change the name of a Runbook after it has been created. The Description is optional and can be up to 512 characters.
- Tags - Tags allow you to assign distinct words and phrases to help identify a runbook.
- Logging - By default, Verbose and Progress records are not written to job history. You can change the settings for a particular runbook to log these records.



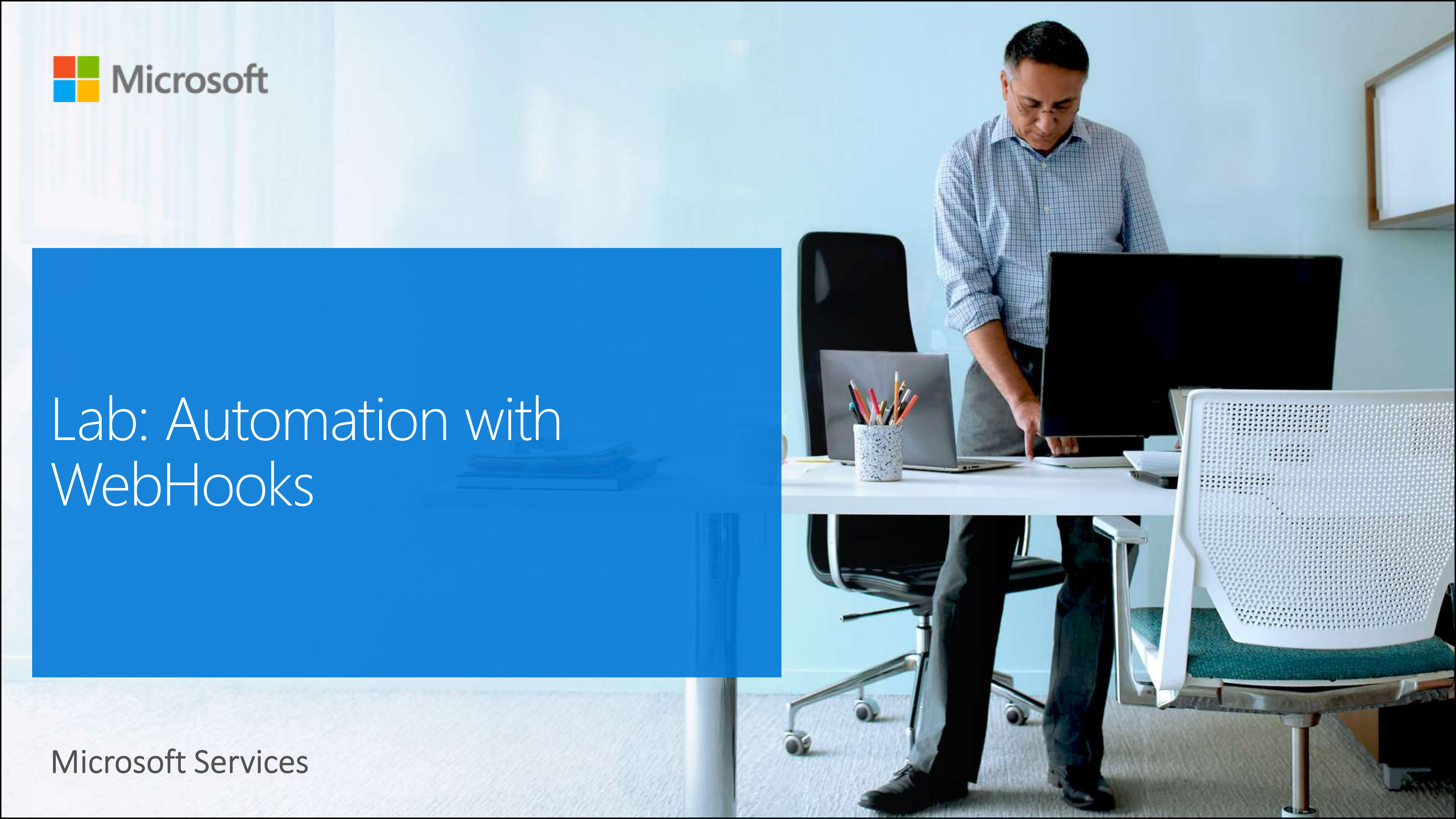
Manage Azure Automation Data

- When a resource is deleted in Azure Automation, it is retained for 90 days for auditing purposes before being permanently removed.
- When you delete an automation account in Microsoft Azure, all objects in the account are deleted including runbooks, modules, configurations, settings, jobs, and assets.
- Azure Automation data should be manually backed up e.g. export Runbooks, DSC configuration etc.
- Geo-Replication in Azure Automation is used by default to ensure data recovery in the event of an Azure datacenter outage.



Resource group
CBAuto
Account
Status
Published
Runbook type

Demo: Creating & Running a Runbook



Microsoft

Lab: Automation with WebHooks

Microsoft Services

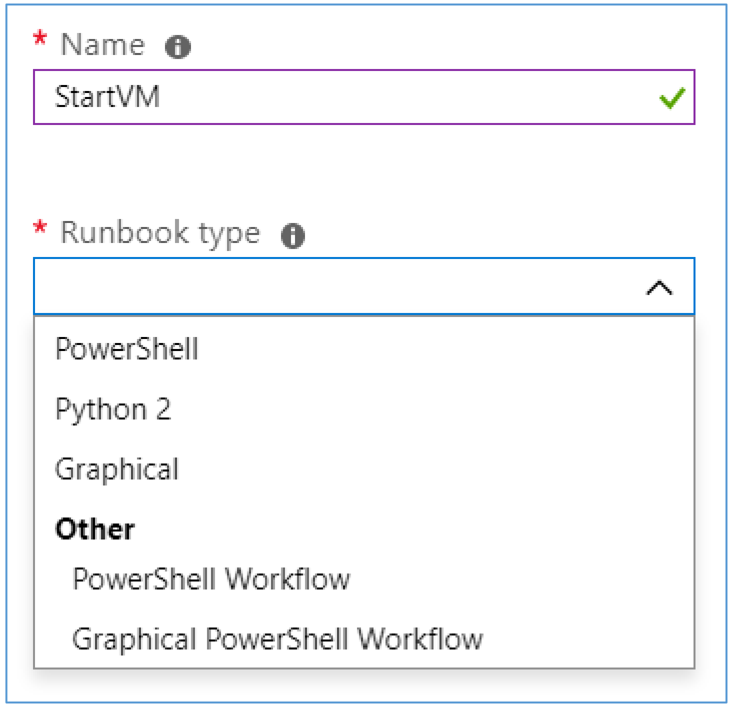
Microsoft

Author Runbooks

Microsoft Services

Runbook Types

There are five types of Runbooks in Azure Automation


* Name

StartVM

* Runbook type

PowerShell

Python 2

Graphical

Other

PowerShell Workflow

Graphical PowerShell Workflow

Graphical.
Based on Windows PowerShell and created and edited completely in graphical editor in Azure portal.

Graphical PowerShell Workflow.
Based on Windows PowerShell Workflow and created and edited completely in the graphical editor.

PowerShell.
Text runbook based on Windows PowerShell script.

PowerShell Workflow.
Text runbook based on Windows PowerShell Workflow.

Python.
Text runbook based on Python.

Graphical Runbooks

- Graphical Runbooks are created and edited with the graphical editor only in the Azure portal.
- Can be exported to a file and then imported into another automation account.
- Generate PowerShell code which cannot be directly viewed or modified
- Cannot be converted to one of the text formats, nor can a text Runbook be converted to graphical format.
- Can be converted to Graphical PowerShell Workflow Runbooks during import and vice-versa.

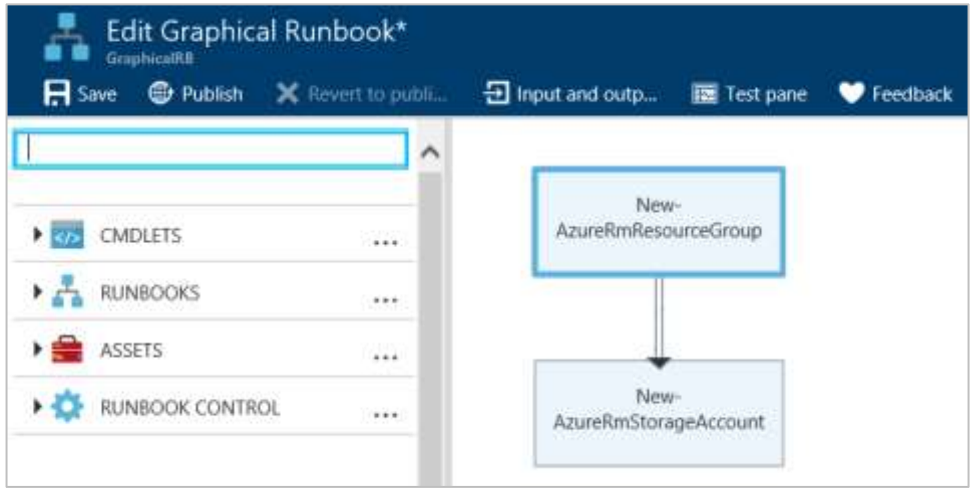
Graphical Runbooks

Advantages

- Create Runbooks with minimal knowledge of PowerShell.
- Visually represent management processes.
- Include other Runbooks as child Runbooks to create high level workflows.

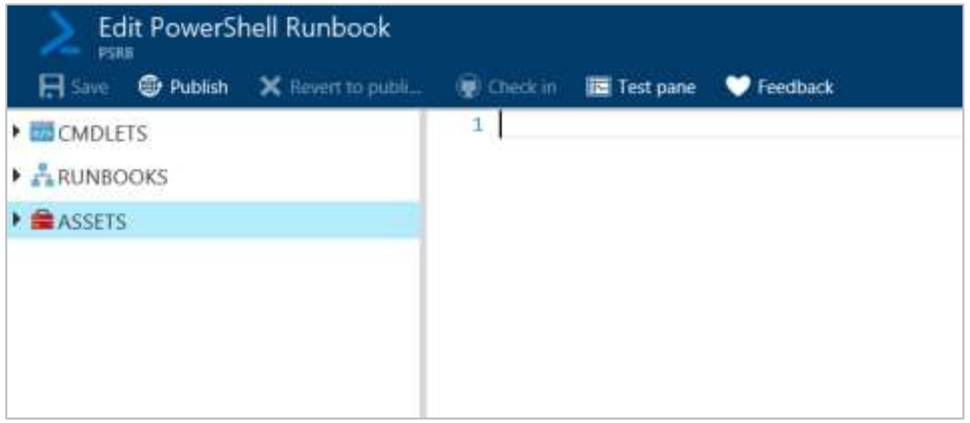
Limitations

- Can't edit a Runbook outside of Azure portal.
- May require PowerShell code to perform complex logic.
- Can't view or directly edit the PowerShell code that is created by the graphical workflow.
- Can't be ran on a Linux Hybrid Runbook Worker



Powershell Runbooks

- PowerShell Runbooks are based on PowerShell.
- Directly edit the code of the Runbook using the text editor in the Azure portal.
- Can also use any offline text editor and import the Runbook into Azure Automation



PowerShell Runbooks

Advantages

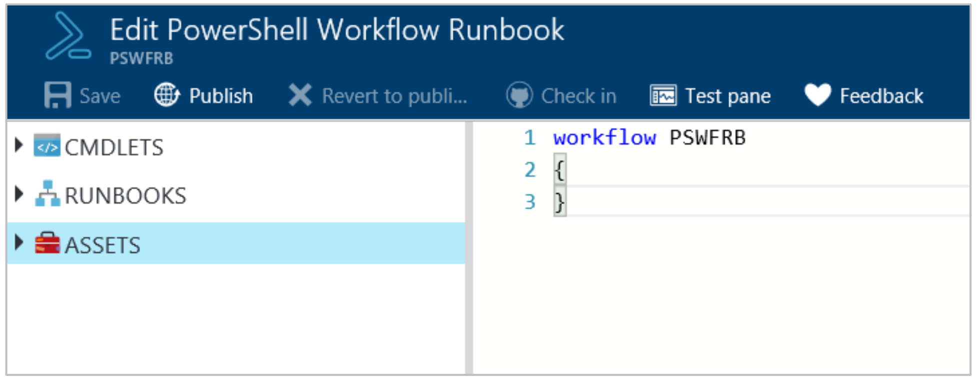
- Implement all complex logic with PowerShell code without the additional complexities of PowerShell Workflow.
- Runbook starts faster than Graphical or PowerShell Workflow Runbooks since it doesn't need to be compiled before running.
- Can be ran in Azure or on both Linux and Windows Hybrid Runbook Workers

Limitations

- Must be familiar with PowerShell scripting.
- Can't use parallel processing to perform multiple actions in parallel.
- Can't use checkpoints to resume Runbook in case of error.

PowerShell Workflow Runbooks

- PowerShell Workflow Runbooks are text Runbooks based on PowerShell Workflow.
- Directly edit the code of the Runbook using the text editor in the Azure portal.
- Can also use any offline text editor and import the Runbook into Azure Automation



```
workflow PSWFRB
{
}
```

PowerShell Workflow Runbooks

Advantages

- Implement all complex logic with PowerShell Workflow code.
- Use checkpoints to resume Runbook in case of error.
- Use parallel processing to perform multiple actions in parallel.
- Include other Graphical Runbooks and PowerShell Workflow Runbooks as child Runbooks to create high level workflows.

Limitations

- Author must be familiar with PowerShell Workflow.
- Runbook must deal with the additional complexity of PowerShell Workflow such as deserialized objects.
- Runbook takes longer to start than PowerShell Runbooks since it needs to be compiled before running.
- Can't be ran on a Linux Hybrid Runbook Worker

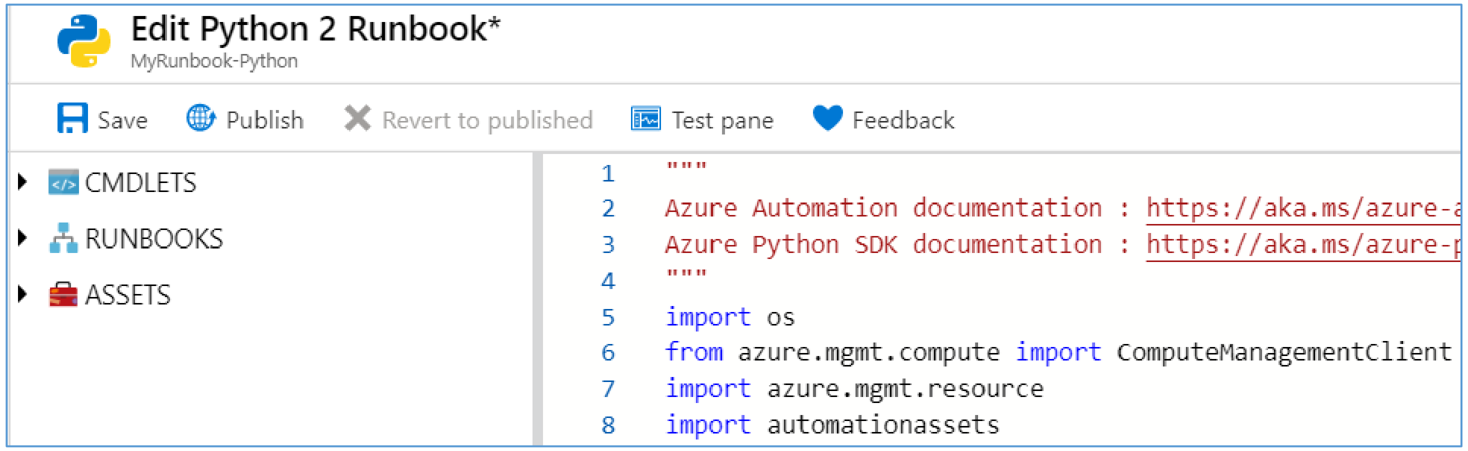
Python runbooks

Advantages

- Utilize the robust Python libraries.
- Can be ran in Azure or on both Linux Hybrid Runbook Workers. Windows Hybrid Runbook Workers are supported with python2.7 installed.

Limitations

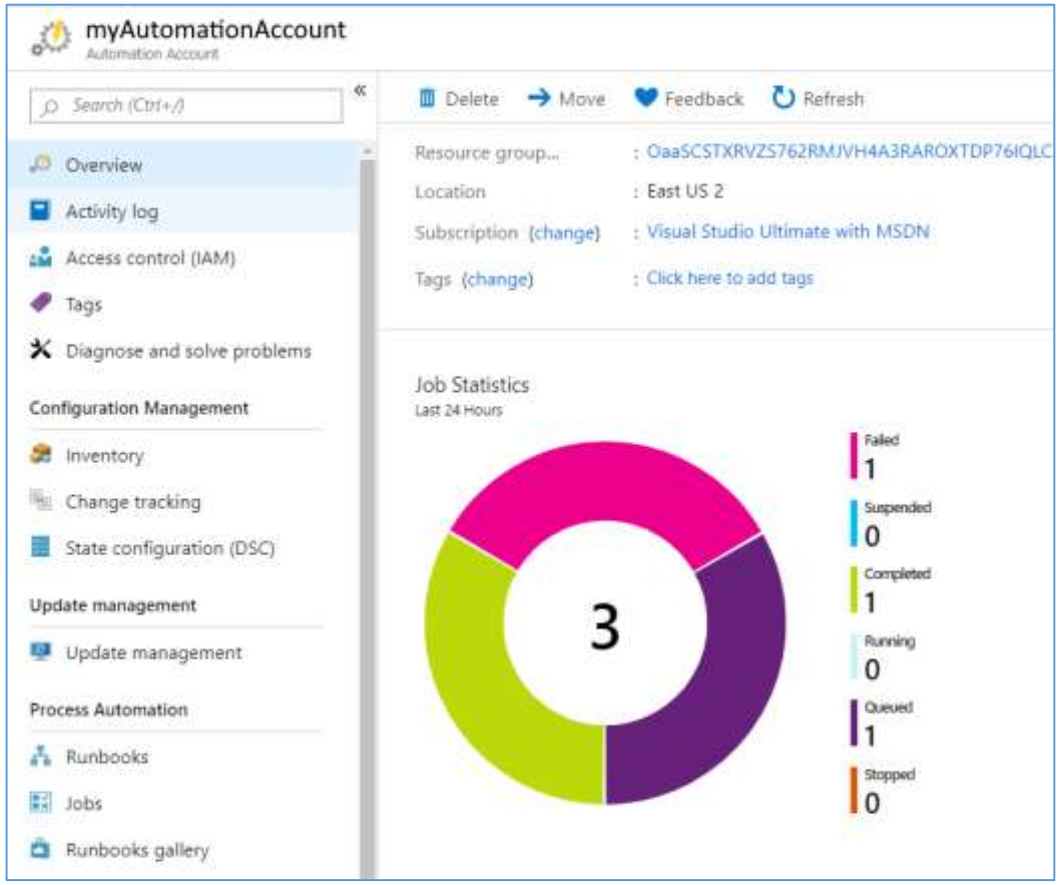
- Must be familiar with Python scripting.
- Only Python 2 is supported now, meaning Python 3 specific functions will fail.
- To use third-party libraries, you must import the package into the Automation Account for it to be used.



```
"""
Azure Automation documentation : https://aka.ms/azure-a
Azure Python SDK documentation : https://aka.ms/azure-p
"""
import os
from azure.mgmt.compute import ComputeManagementClient
import azure.mgmt.resource
import automationassets
```

Automation Accounts

- All Runbooks, Hybrid Worker Groups, Assets and DSC configuration are stored in an Automation Account
- Security and Automation management boundary for Azure Automation
- Scope for RBAC
- Location ensures that all Automation components are stored in a particular Azure region
- Specifies the Azure subscription to be billed for Automation usage
- There is no limit for a maximum number of Automation accounts in a subscription


myAutomationAccount
Automation Account

Resource group... : OaaSCSTXRVZS762RMJVH4A3RAROXTDP76IQLC
Location : East US 2
Subscription (change) : Visual Studio Ultimate with MSDN
Tags (change) : Click here to add tags

Job Statistics
Last 24 Hours

Failed	1
Suspended	0
Completed	1
Running	0
Queued	1
Stopped	0

AL

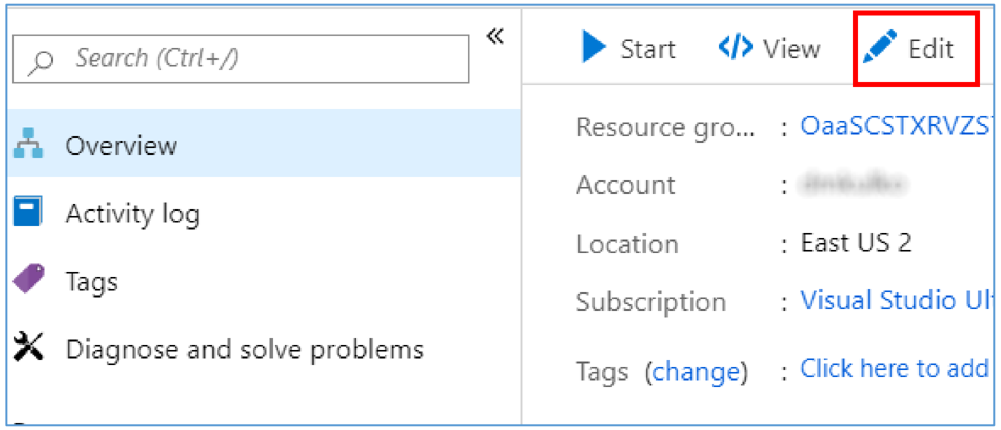
Create & Import Runbooks

- You can add a Runbook to Azure Automation by either creating a new one or by importing an existing Runbook from a file or from the Runbook Gallery.
- Create a new Runbook using the Azure portal or Windows PowerShell.
- Use New-AzAutomationRunbook to create an empty Runbook.
- When you create or import a new Runbook, it must be published before you can run it.

```
New-AzureRmAutomationRunbook -AutomationAccountName MyAccount `
-Name NewRunbook -ResourceGroupName MyResourceGroup -Type PowerShell
```

Edit Text Runbooks

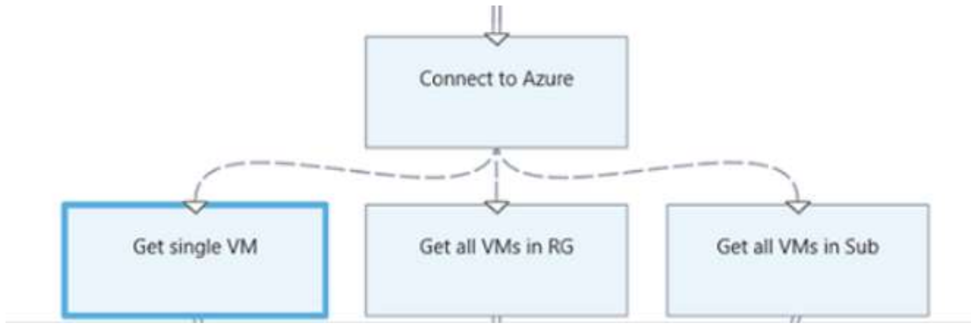
- Textual Runbooks can be edited using the text editor in Azure Automation.
- It has the typical features of other code editors such as intellisense and color coding.
- Allows you to select from a list of pre-existing resources instead of typing code in.
- Each Runbook in Azure Automation has two versions, Draft and Published, you edit the Draft version of the Runbook and then publish it so it can be executed.
- To edit a Runbook with Windows PowerShell, use the editor of your choice and save it as a .ps1 file.



Resource gro...	:	OaaSCSTXRVZS
Account	:	[redacted]
Location	:	East US 2
Subscription	:	Visual Studio Ul
Tags (change)	:	Click here to add

Edit Graphic Runbooks

- Graphic Runbooks can be edited using the graphic editor in Azure Automation.
- Select activities from the library control and place them onto the canvas, then edit their parameters in the configuration pane and save and publish.
- Use links to connect two activities in your canvas.
- Sequence link – destination activity runs after source activity completes
- Pipeline link – destination activity runs once an output object is received from the source activity
- Conditions can also be specified on a link so that destination activity will run only if true.
- Junctions allow you to run multiple activities in parallel and ensure that all have completed before moving on.
- Use Runbook input & output to provide and receive data to and from a Runbook e.g. VM name.



PowerShell Workflow

PowerShell Workflow is a PowerShell extension that allows you to run a PowerShell script on multiple devices in parallel with added functionality such as checkpoints, suspend & restart.

You can convert a PowerShell script to a PowerShell workflow by enclosing it with the Workflow keyword and naming it.

Consider the following script that gets all running services.

```
Get-Service | Where-Object {$_.Status -eq "Running"}
```

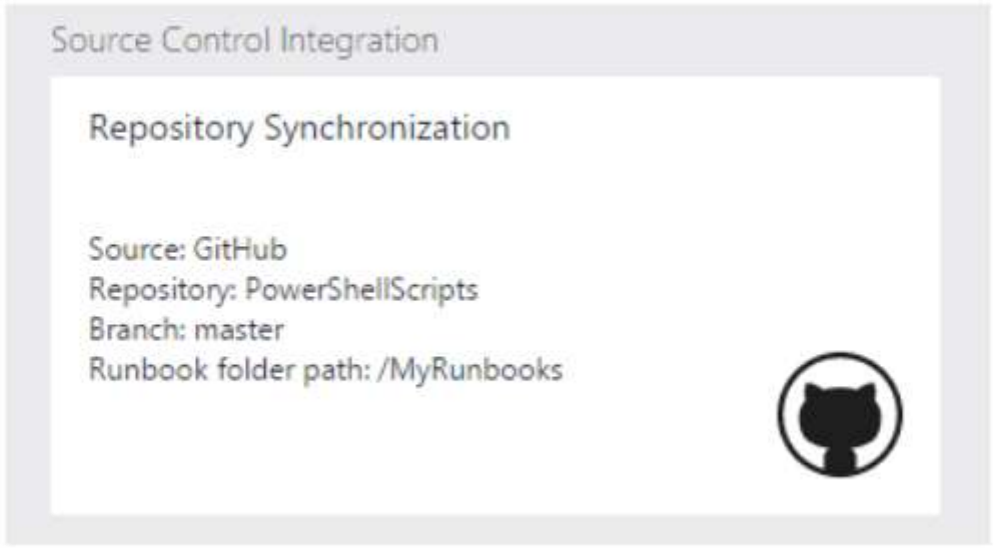
Converted to a PowerShell Workflow, it is:

```
Workflow Get-RunningServices
{
    Get-Service | Where-Object -FilterScript {$_.Status -eq "Running"}
}
```

This syntax allows you to benefit from the PowerShell Workflow functionality such as checkpoints, suspend and restart.

Source Control Integration

- Source control integration allows you to push & pull Runbooks from Azure Automation to and from GitHub.
- Create a new or add an existing GitHub account to Azure Automation.
- Configure the GitHub folder path that will be synchronized.
- A check-in from Azure Automation will overwrite code that currently exists in GitHub.
- Synchronization is a manual process and syncs Runbooks only and not ARM templates.


Source Control Integration

Repository Synchronization

Source: GitHub
Repository: PowerShellScripts
Branch: master
Runbook folder path: /MyRunbooks

Demo: Graphical Runbook Authoring

Microsoft

Desired State Configuration

Microsoft Services

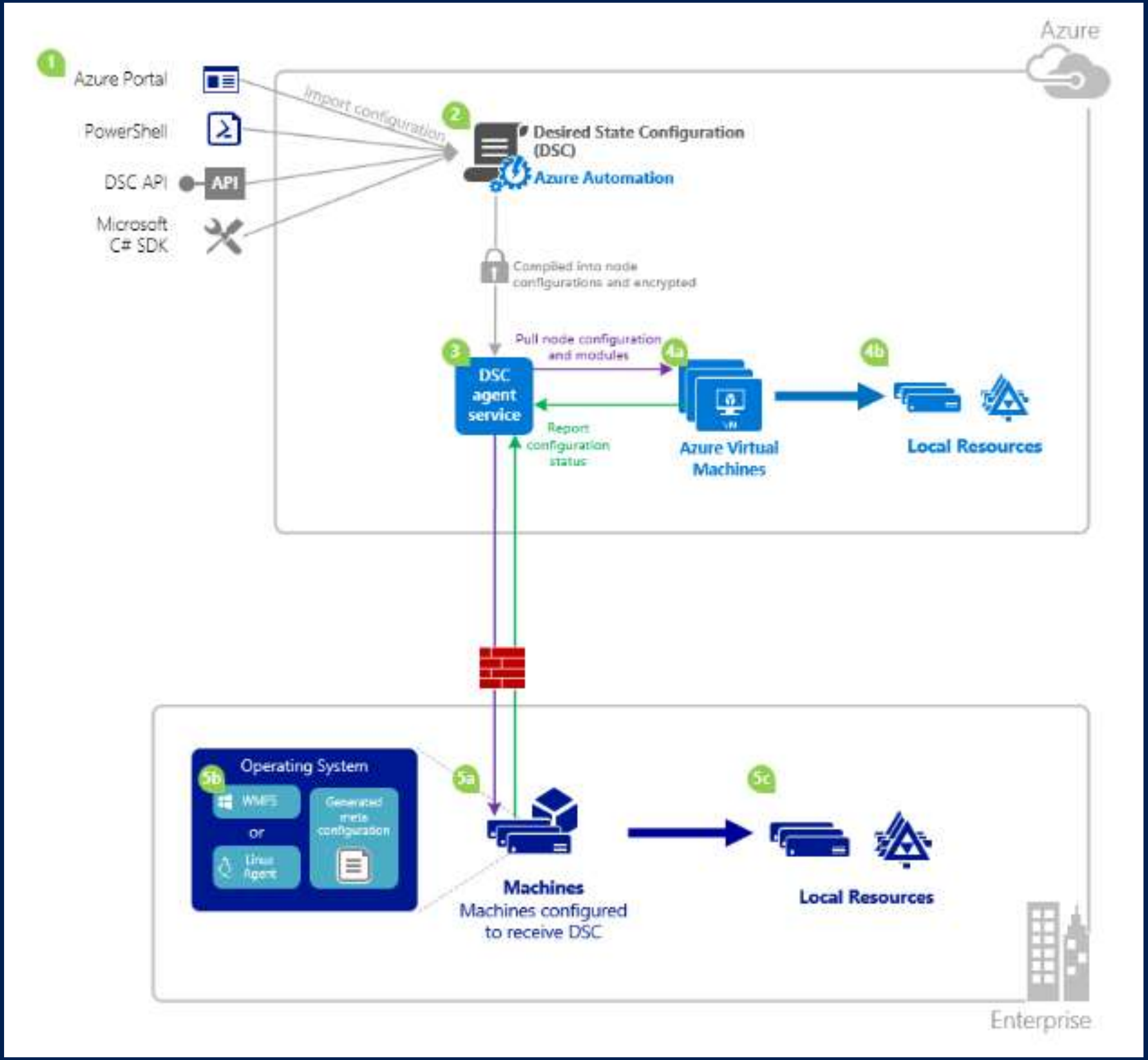
Automation Desired State Configuration (DSC)

- Azure Automation Desired State Configuration (DSC) allows you to consistently deploy, monitor, and automatically update the desired state of all your IT resources, at scale from the cloud.
- Built on PowerShell DSC and can align machine configuration with a specific state across physical and virtual machines, using Windows or Linux, in the cloud or on-premises
- Builds on top of PowerShell DSC to provide an easier configuration management experience.
- Automation DSC includes:
  - Author and manage PowerShell DSC configurations
  - Import DSC Resources
  - Generate DSC Node configurations (MOF documents)
- DSC configuration files are stored on an Azure Automation DSC Pull server so that target nodes can download and apply them.

Automation DSC Terms

- Configuration – Introduced in PowerShell DSC and allows you to define the desired state of your environment using PowerShell syntax.
- Node Configuration – Is a file that is produced when a DSC configuration is compiled, this is typically the configuration document that nodes will apply.
- Node – Any machine that has its configuration managed by DSC.
- Resource – A PowerShell module that is used to define a DSC configuration. They are seen as the building blocks of DSC configuration.
- Compilation Job – An instance of compilation of a configuration to create a node configuration.
  - Similar to Azure Automation Runbook jobs, but they do not perform any task, only compile configurations
  - Automatically stored on an Azure DSC pull server
  - Overwrites previous versions of node configurations

Automation DSC Process



1 An actor imports a DSC configuration to Azure Automation

2 An actor compiles the DSC configuration into node configuration and it's encrypted by Azure Automation

3 The node configuration is placed in the DSC agent service and is then pulled by cloud or on-premises machines

4a Cloud – Node configurations or modules are pulled by the DSC on-boarded Azure VM's and the VM's report on their configuration status and compliance back to the service

4b Azure VMs conform to the desired state

5a On-premises – node configuration is pulled on to local machine configured to receive DSC.

5b On-premises machines must have either Windows Management Framework 5 installed if it's a Windows node or the Linux Agent installed for a Linux node. Those machines act on local resources as instructed

5c Local machines conform to the desired state

Onboarding Nodes for Management

- Azure Automation DSC allows onboarding of the following machines:
  - Azure Classic VM’s
  - Azure Resource Manager VM’s
  - Amazon Web Services VM’s
  - Physical or virtual Windows machines on-premises, or in a cloud other than Azure or AWS
  - Physical or virtual Linux machines on-premises, in Azure, or in a cloud other than Azure

- Azure classic VM’s can be onboarded via the new Azure portal or PowerShell.
- Azure Resource Manager VM’s can be onboarded via the new Azure portal, ARM templates or PowerShell.
- Amazon Web Services VM’s can be onboarded using the AWS DSC Toolkit.
- Windows machines on-premises or in a cloud other than Azure or AWS must have WMF 5.0 installed and have the PowerShell DSC metaconfiguration applied.
- Linux machines on-premises, in Azure or in another cloud must have the latest DSC Linux agent installed and have the PowerShell DSC metaconfiguration applied.

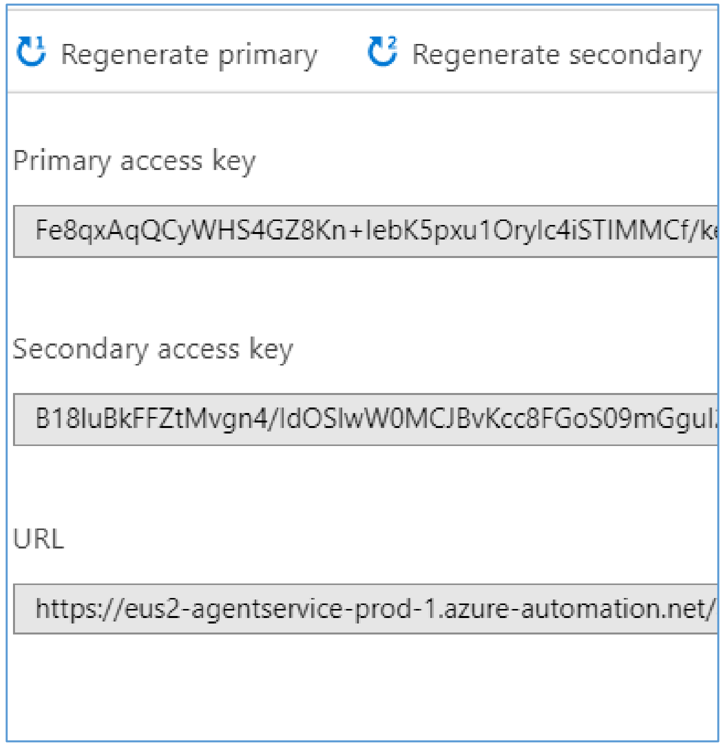
DSC Node Registration Parameters

- **Registration key** – Specifies which Automation account Access Key to use for the DSC node to authenticate with.
- **Node Configuration Name** – Specifies the name of the DSC configuration file to be used.
- **Refresh Frequency** – Specifies how often the DSC node will contact the Pull server and download the latest node configuration.
- **Configuration Mode Frequency** – Specifies how often the downloaded node configuration will be applied.
- **Configuration Mode** – Specifies the mode of configuration e.g. ApplyAndMonitor, ApplyOnly.
- **Allow Module Override** – Specifies whether or not newer modules downloaded from the Pull server are allowed to overwrite older modules on the DSC node.
- **Reboot Node if Needed** – Specifies whether or not to reboot following a configuration update.
- **Action after Reboot** – Specifies actions to take following a reboot e.g. ContinueConfiguration or StopConfiguration.

DSC Metaconfiguration & Secure Registration

- DSC Metaconfiguration is a script that consists of the DSC engine settings that will be used to connect a node to a DSC pull server and keep it updated.
- DSC metaconfigurations for Azure Automation DSC can be generated using either a PowerShell DSC configuration, or the Azure Automation PowerShell cmdlets.
- Must be applied to on-premises or non Azure virtual machines in order to onboard the server to Automation DSC.

- Secure Registration is a registration protocol, allows a DSC node to authenticate to a PowerShell DSC V2 Pull server (including Azure Automation DSC).
- The node registers to the DSC Pull server at a Registration URL, and authenticates using a Registration key specified in the DSC Metaconfiguration.
- A certificate is generated and used for future communication between the node and the DSC Pull server.



Primary access key

Fe8qxAqQCyWHS4GZ8Kn+lebK5pxu1OryIc4iSTIMMCf/k

Secondary access key

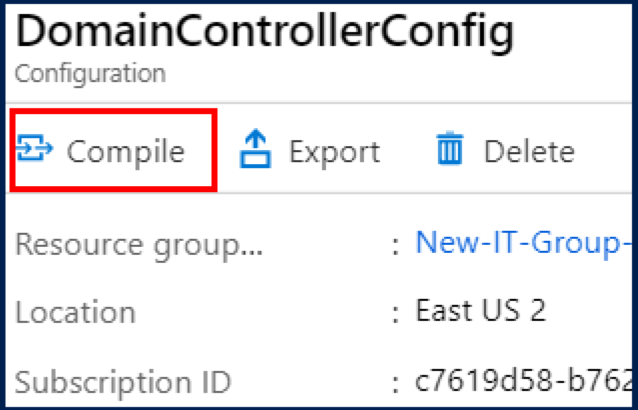
B18luBkFFZtMvgn4/ldOSlwW0MCJBvKcc8FGoS09mGgul

URL

https://eus2-agentservice-prod-1.azure-automation.net/

Compiling Configuration

DSC configuration can be compiled in two ways with Azure Automation, the Azure portal or with Windows PowerShell.


DomainControllerConfig
Configuration

Resource group... : New-IT-Group-
Location : East US 2
Subscription ID : c7619d58-b762

Azure portal

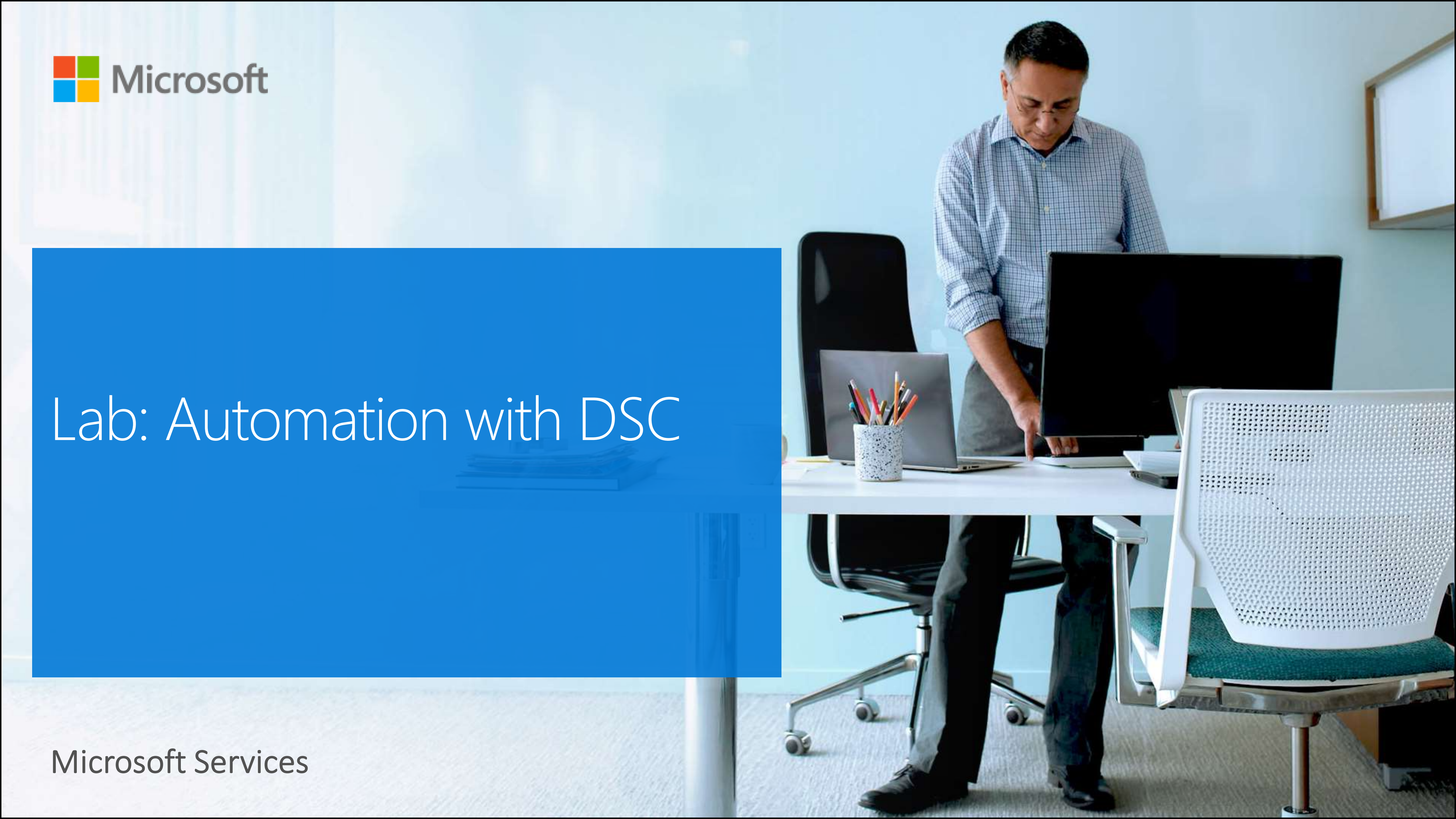
- Simplest method with interactive user interface
- Form to provide simple parameter values
- Easily track job state
- Access authenticated with Azure logon

Windows PowerShell

- Call from command line with PowerShell cmdlets
- Can be included in automated solution with multiple steps
- Provide simple and complex parameter values
- Track job state
- Client required to support PowerShell cmdlets
- Pass ConfigurationData
- Compile configurations that use credentials

Demo: Desired State Configuration

Microsoft



Lab: Automation with DSC

Microsoft Services

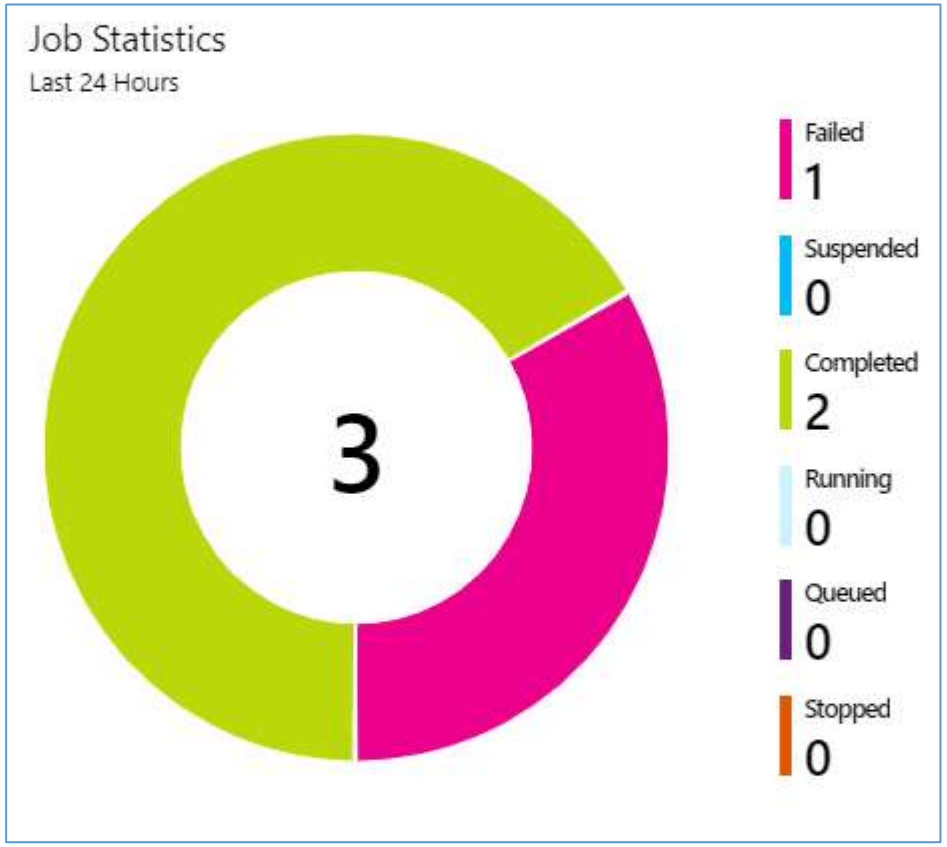
Microsoft

Monitoring & Troubleshooting

Microsoft Services

Automation Monitoring

- Azure Automation provides monitoring on the status of Runbooks and DSC configurations.
- Job status logs can be viewed in the Azure portal or using the PowerShell cmdlet Get-AzAutomationJob
- Advanced logging scenarios requires you to create custom PowerShell scripts or send job logs to Azure Log Analytics Log Analytics.

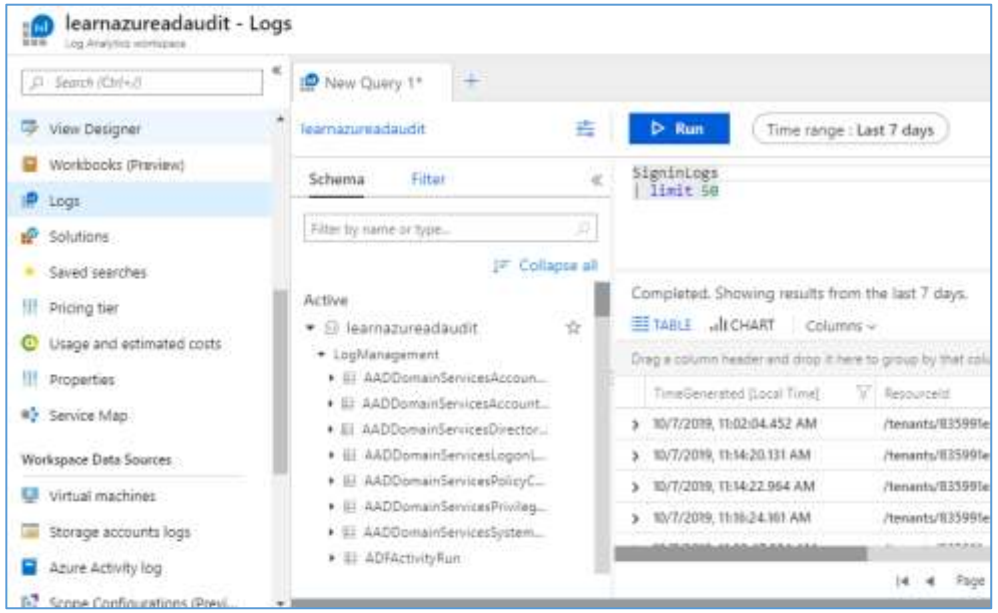


Azure Log Analytics

- Log Analytics workspace is a unique environment for Azure Monitor log data. Each workspace has its own data repository and configuration, and data sources and solutions are configured to store their data in a workspace. You require a Log Analytics workspace if you intend on collecting data.

- Provides real-time insights using integrated search and custom dashboards to readily analyze records across all of your workloads and servers regardless of their physical location.

- Log Analytics allows you to:
    - Get insight on your Automation jobs
    - Trigger an email or alert based on your Runbook job status (e.g. failed or suspended)
    - Write advanced queries across your job streams
    - Correlate jobs across Automation accounts
    - Visualize your job history over time



```
SigninLogs
| limit 50
```

Completed. Showing results from the last 7 days.

Drag a column header and drop it here to group by that colu

TimeGenerated [Local Time]	ResourceId
10/7/2019, 11:02:04.452 AM	/tenants/835991e
10/7/2019, 11:14:20.131 AM	/tenants/835991e
10/7/2019, 11:14:22.964 AM	/tenants/835991e
10/7/2019, 11:16:24.161 AM	/tenants/835991e

Automation & Log Analytics

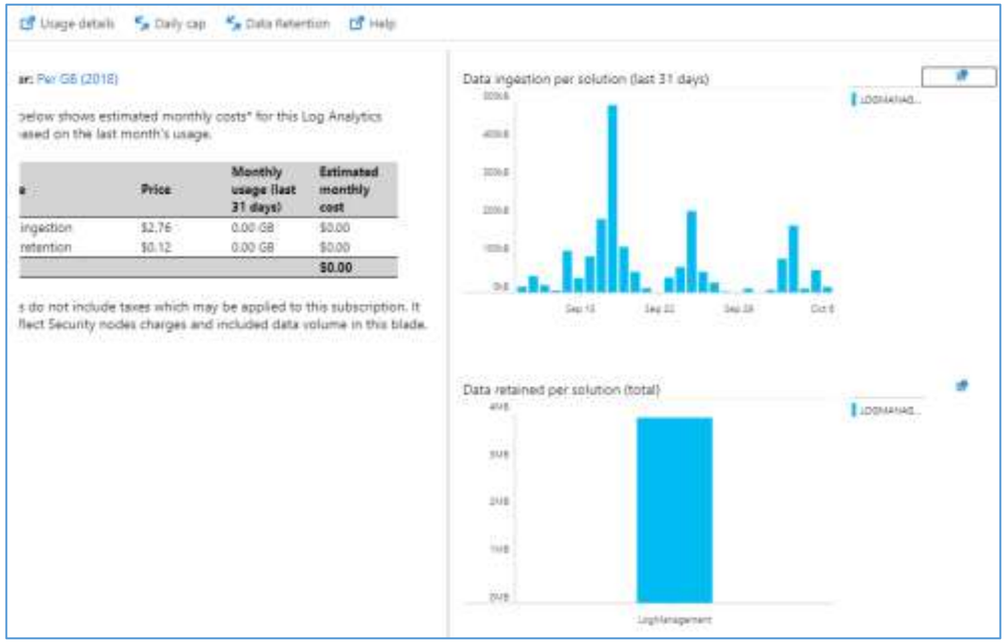
- Log analytics is billed per GB of data uploaded into the service, and offers a free data plan with up to 500MB of data per day with a retention period of 7 days.

- If the 500MB daily limit is reached, data analyzing will stop and resume at the start of the next day.

- Requirements to use Log Analytics with Automation:
  - The latest release of Azure PowerShell.
  - A Log Analytics workspace.
  - The ResourceId for your Azure Automation account.

- Set up integration with Azure Monitor logs:

```
$workspaceId = "[resource id of the log analytics workspace]"

$automationAccountId = "[resource id of your automation account]"

Set-AzDiagnosticSetting -ResourceId $automationAccountId `
            -WorkspaceId $workspaceId -Enabled 1
```


Usage details
Daily cap
Data Retention
Help
Data ingestion per solution (last 31 days)
Data retained per solution (total)

ar: Per GB (2018)

below shows estimated monthly costs* for this Log Analytics ased on the last month's usage.

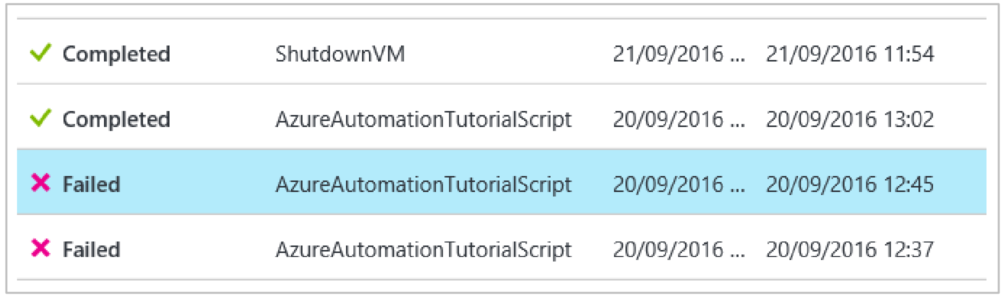
e	Price	Monthly usage (last 31 days)	Estimated monthly cost
ingestion	$2.76	0.00 GB	$0.00
retention	$0.12	0.00 GB	$0.00
			$0.00

s do not include taxes which may be applied to this subscription. It flect Security nodes charges and included data volume in this blade.

Demo: Automation Monitoring

Automation Troubleshooting

- Automation logs are useful during troubleshooting Runbook and DSC configuration errors.
- Error logs can be viewed in the Azure portal or by using the PowerShell cmdlet Get-AzAutomationJob and can be used as a starting point to help isolate the problem.
- Some of the most common troubleshooting scenarios include:
  - Authentication errors when working with Azure Automation Runbooks
  - Errors when working with Runbooks
  - Errors when importing modules
  - Errors when working with Desired State Configuration (DSC)

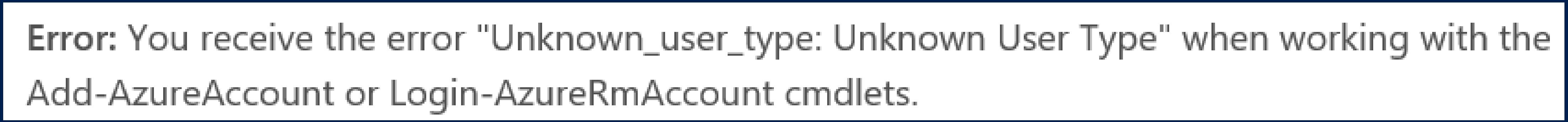


✓ Completed	ShutdownVM	21/09/2016 ...	21/09/2016 11:54
✓ Completed	AzureAutomationTutorialScript	20/09/2016 ...	20/09/2016 13:02
✗ Failed	AzureAutomationTutorialScript	20/09/2016 ...	20/09/2016 12:45
✗ Failed	AzureAutomationTutorialScript	20/09/2016 ...	20/09/2016 12:37

Sign in to Azure Account failed

Cause:

This error occurs if the credential asset name is not valid or if the username and password that you used to setup the Automation credential asset are not valid.



Error: You receive the error "Unknown_user_type: Unknown User Type" when working with the Add-AzureAccount or Login-AzureRmAccount cmdlets.

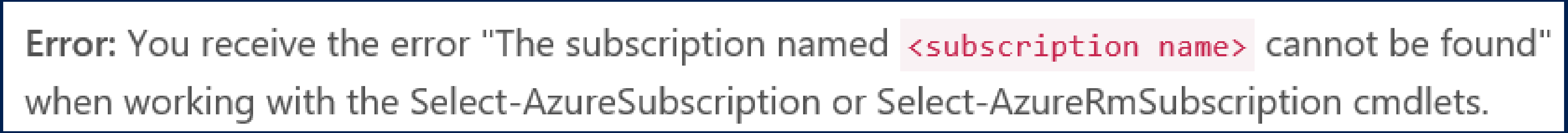
Resolution:

Make sure that you don't have any special characters, including the @ character in the Automation credential asset name that you are using to connect to Azure and check that you can use the username and password that are stored in the Azure Automation credential in your local PowerShell ISE editor.

Unable to find the Azure subscription

Cause:

This error occurs if the subscription name is not valid or if the Azure Active Directory user who is trying to get the subscription details is not configured as an admin of the subscription.



Error: You receive the error "The subscription named `<subscription name>` cannot be found" when working with the Select-AzureSubscription or Select-AzureRmSubscription cmdlets.

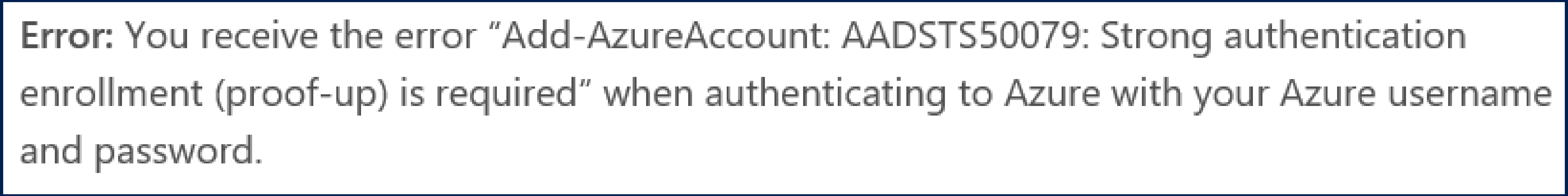
Resolution:

Make sure that you run the Add-AzureAccount before running the Select-AzureSubscription cmdlet and if you still see this error message, modify your code by adding the Get-AzureSubscription cmdlet following the Add-AzureAccount cmdlet and then execute the code. Then verify if the output of Get-AzureSubscription contains your subscription details.

Authentication to Azure failed due to MFA being enabled

Cause:

If you have multi-factor authentication on your Azure account, you can't use an Azure Active Directory user to authenticate to Azure. Instead, you need to use a certificate or a service principal to authenticate to Azure.



Error: You receive the error "Add-AzureAccount: AADSTS50079: Strong authentication enrollment (proof-up) is required" when authenticating to Azure with your Azure username and password.

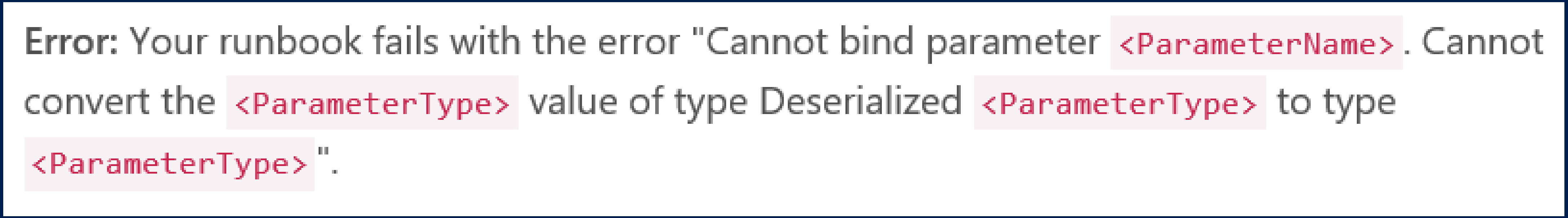
Resolution:

Use a certificate or a service principal to authenticate to Azure.

Runbook fails because of deserialized object

Cause:

If your Runbook is a PowerShell Workflow, it stores complex objects in a deserialized format in order to persist your Runbook state if the workflow is suspended.



Error: Your runbook fails with the error "Cannot bind parameter `<ParameterName>`. Cannot convert the `<ParameterType>` value of type Deserialized `<ParameterType>` to type `<ParameterType>`".

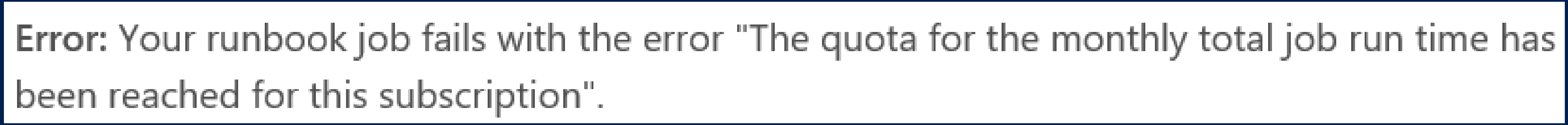
Resolution:

If you are piping complex objects from one cmdlet to another, wrap these cmdlets in an InlineScript or pass the name or value that you need from the complex object instead of passing the entire object or use a PowerShell Runbook instead of a PowerShell Workflow Runbook.

Runbook job failed because the allocated quota is exceeded

<u>Cause:</u>

This error occurs when the job execution exceeds the 500-minute free quota for your account and applies to all types of job execution tasks such as testing a job, starting a job from the portal, executing a job by using Webhooks and scheduling a job to execute by using either the Azure portal or in your datacenter.



Error: Your runbook job fails with the error "The quota for the monthly total job run time has been reached for this subscription".

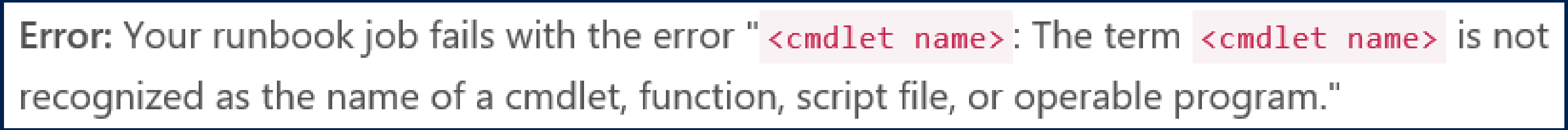
<u>Resolution:</u>

If you want to use more than 500 minutes of processing per month you will need to change your subscription from the Free tier to the Basic tier.

Cmdlet not recognized when executing a Runbook

Cause:

This error is caused when the PowerShell engine cannot find the cmdlet you are using in your Runbook. This could be because the module containing the cmdlet is missing from the account, there is a name conflict with a Runbook name, or the cmdlet also exists in another module and Automation cannot resolve the name.



Error: Your runbook job fails with the error "`<cmdlet name>`: The term `<cmdlet name>` is not recognized as the name of a cmdlet, function, script file, or operable program."

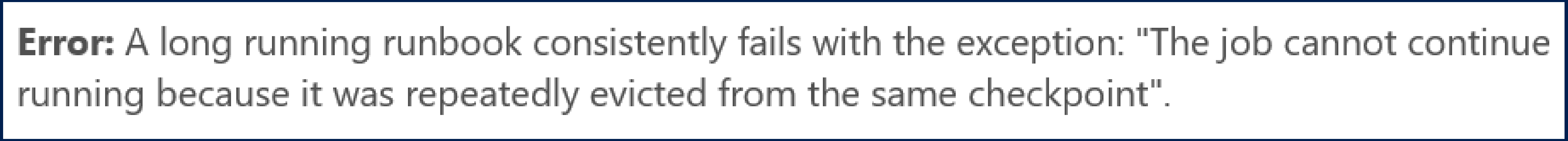
Resolution:

Check that you have entered the cmdlet name correctly or make sure the cmdlet exists in your Automation account and that there are no conflicts or if you do have a name conflict and the cmdlet is available in two different modules, you can resolve this by using the fully qualified name for the cmdlet.

Runbook job repeatedly evicted from the same checkpoint

Cause:

This is by design behavior due to the "Fair Share" monitoring of processes within Azure Automation, which automatically suspends a Runbook if it executes longer than 3 hours.



Error: A long running runbook consistently fails with the exception: "The job cannot continue running because it was repeatedly evicted from the same checkpoint".

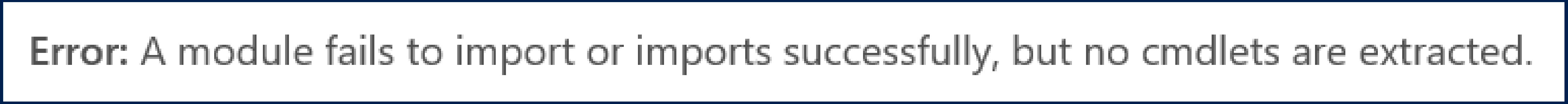
Resolution:

Use Checkpoints in a workflow

Module fails to import or cmdlets can't be executed after importing

Cause:

The structure does not match the structure that Automation needs it to be in or the module is dependent on another module that has not been deployed to your Automation account or the module is missing its dependencies in the folder.



Error: A module fails to import or imports successfully, but no cmdlets are extracted.

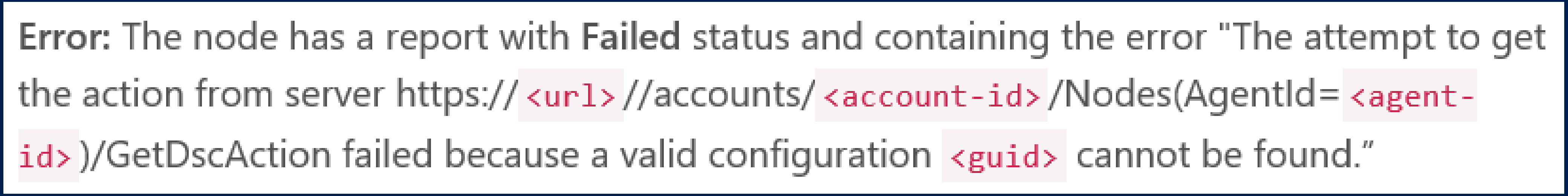
Resolution:

Make sure that the module follows the following format: ModuleName.Zip -> ModuleName or Version Number -> (ModuleName.psm1, ModuleName.psd1) or open the .psd1 file and see if the module has any dependencies. If it does, upload these modules to the Automation account or make sure that any referenced .dlls are present in the module folder.

DSC node is in failed status with a “Not found” error

Cause:

This error typically occurs when the node is assigned to a configuration name (e.g. ABC) instead of a node configuration name (e.g. ABC.WebServer).



Error: The node has a report with **Failed** status and containing the error "The attempt to get the action from server https://`<url>`//accounts/`<account-id>`/Nodes(AgentId=`<agent-id>`)/GetDscAction failed because a valid configuration `<guid>` cannot be found.”

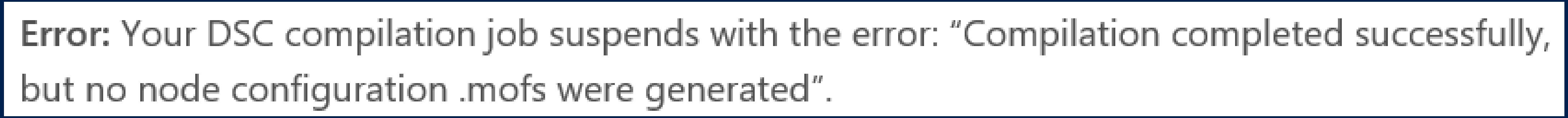
Resolution:

Make sure that you are assigning the node with "node configuration name" and not the "configuration name. You can assign a node configuration to a node using Azure portal or with a PowerShell cmdlet.

No node configurations produced when a configuration is compiled

Cause:

When the expression following the Node keyword in the DSC configuration evaluates to $null then no node configurations will be produced.



Error: Your DSC compilation job suspends with the error: "Compilation completed successfully, but no node configuration .mofs were generated".

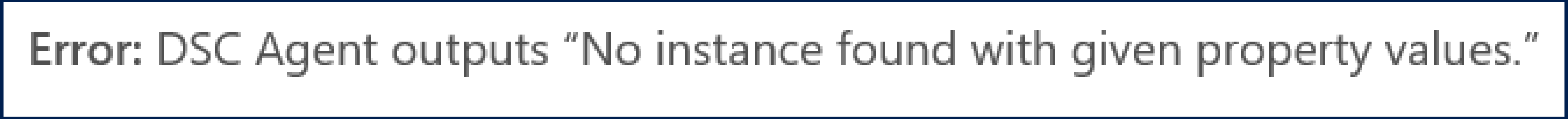
Resolution:

Make sure that the expression next to the Node keyword in the configuration definition is not evaluating to $null or if you are passing ConfigurationData when compiling the configuration, make sure that you are passing the expected values that the configuration requires from ConfigurationData.

The DSC node report becomes stuck “in progress” state

Cause:

You have upgraded your WMF version and have corrupted WMI.



Error: DSC Agent outputs “No instance found with given property values.”

Resolution:

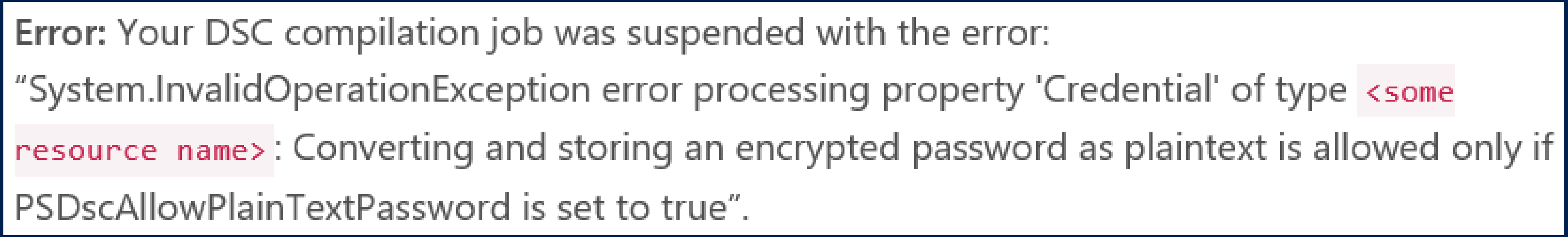
Delete DSCEngineCache.mof by running the following command in an elevated PowerShell session (Run as Administrator):

```
Remove-Item -Path $env:SystemRoot\system32\Configuration\DSCEngineCache.mof
```

Unable to use a credential in a DSC configuration

Cause:

You have used a credential in a configuration but didn't provide proper ConfigurationData to set PSDscAllowPlainTextPassword to true for each node configuration.



Error: Your DSC compilation job was suspended with the error: "System.InvalidOperationException error processing property 'Credential' of type `<some resource name>`: Converting and storing an encrypted password as plaintext is allowed only if PSDscAllowPlainTextPassword is set to true".

Resolution:

Make sure to pass in the proper ConfigurationData to set PSDscAllowPlainTextPassword to true for each node configuration mentioned in the configuration

Microsoft