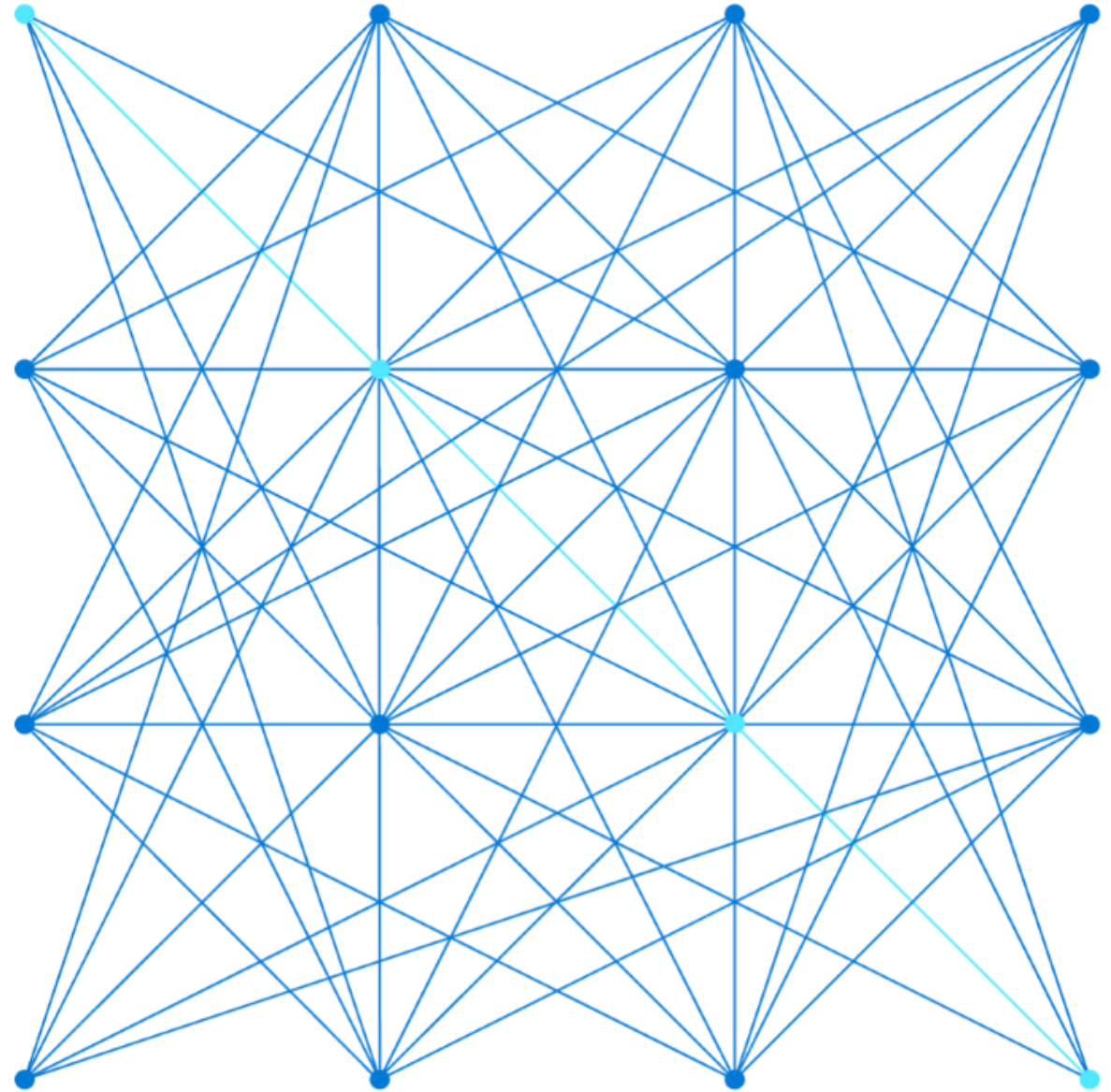


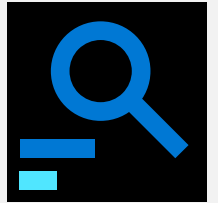
AZ-400.00

Module 13: Managing Infrastructure and Configuration using Azure Tools

az-p-REST



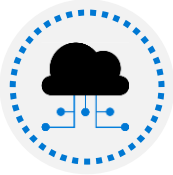
Lesson 01: Module overview



Module overview



Lesson 1: Module Overview



Lesson 2: Infrastructure as code and configuration management



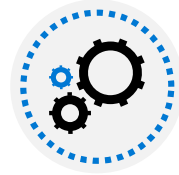
Lesson 3: Create Azure resources using ARM templates



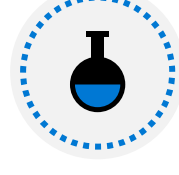
Lesson 4: Create Azure resources by using Azure CLI



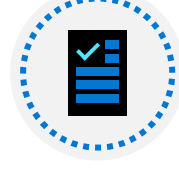
Lesson 5: Azure Automation with DevOps



Lesson 6: Desired State Configuration (DSC)



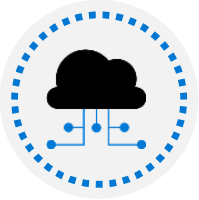
Lesson 7: Lab



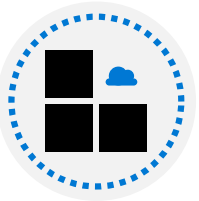
Lesson 8: Module review and takeaways

Learning objectives

After completing this module, students will be able to:

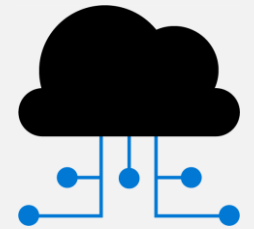


Apply infrastructure and configuration as code principles



Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, Azure CLI, DSC and Azure Automation

Lesson 02: Infrastructure as code and configuration management



Environment **deployment** (Compare manual Vs IaC) 1 of 2

Manual **deployment**:

Snowflake servers

Deployment steps vary by environment

More verification steps and more elaborate manual processes

Increased documentation to account for differences

Deployment on weekends to allow time to recover from errors

Slower release cadence to minimize pain and long weekends

Infrastructure as code:

Consistent servers between environments

Environments created or scaled easily

Fully automate creation and updates of environments

Transition to immutable infrastructure

Use blue/green deployments

Treat servers as cattle, not pets

Environment **configuration** (Compare manual Vs **CaC**) 2 of 2

Manual **configuration**:

- Configuration bugs difficult to Identify
- Error prone
- More verification steps and more elaborate manual processes
- Increased documentation
- Deployment on weekends to allow time to recover from errors
- Slower release cadence to minimize requirement for long weekends

Configuration as code:

- Bugs easily reproducible
- Consistent configuration
- Increase deployment cadence to reduce amount of incremental change
- Treat environment and configuration as executable documentation

Imperative versus declarative configuration

Approaches to implementing infrastructure and configuration as code

Declarative:

Functional

Defines **what** the final state should be



Imperative:

Procedural

Defines **how** to achieve that final state



Idempotent configuration

Idempotence – definition:

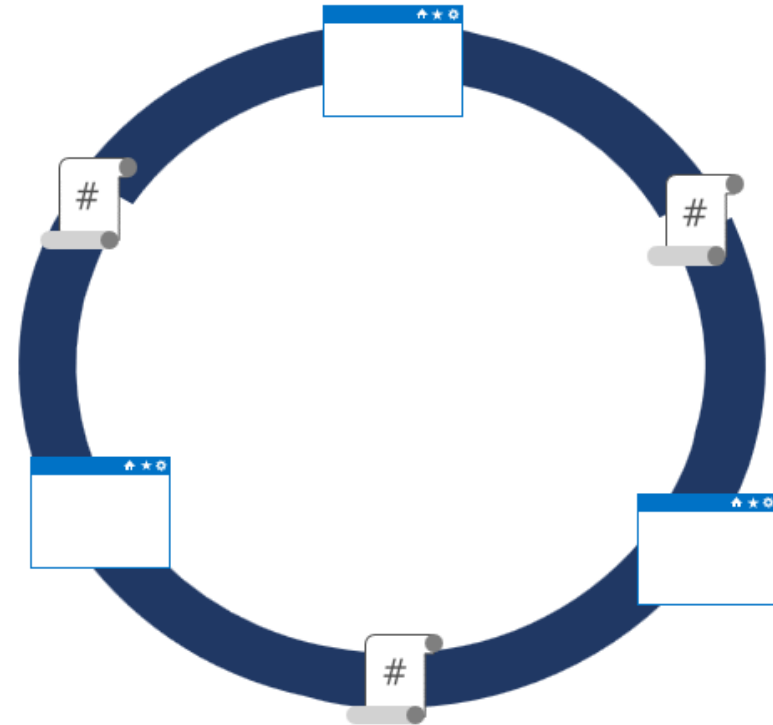
Mathematical term used in the context of infrastructure and configuration as code

Ability to apply one or more operations against a resource, resulting in the same outcome

To attain idempotence:

Automatically configure and reconfigure an existing set of resources, or

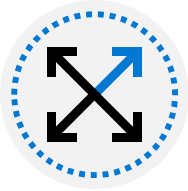
Discard existing resources and spin up a fresh environment



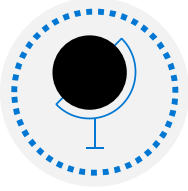
Lesson 03: Create Azure resources using ARM templates



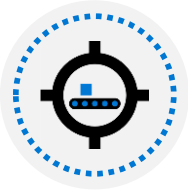
Why use ARM templates?



Make deployments faster and more repeatable



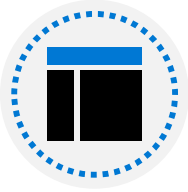
Improve consistency by providing a common language



Enable you to deploy multiple resources in the correct order by mapping out resource dependencies



Reduce manual, error-prone tasks



Templates can be linked together to provide a modular solution (might build on Quickstart Templates)

Template components

JSON data stored as an object in text
Collection of key-value pairs

Templates can contain the following sections:

Parameters

Variables

Functions

Resources

Outputs

Manage dependencies

Some resources will depend on other resources before you can deploy them

Define this relationship by marking the dependency with the **dependsOn** element

```
129     "type": "Microsoft.Compute/virtualMachines",
130     "name": "[variables('vmName')]",
131     "location": "[parameters('location')]",
132     "apiVersion": "2018-10-01",
133     "dependsOn": [
134         "[resourceId('Microsoft.Storage/storageAccounts/', variables('storageAccountName'))]",
135         "[resourceId('Microsoft.Network/networkInterfaces/', variables('nicName'))]"
136     ],
```

Modularize templates ([see this first](#))

Best practice: Modularize templates into individual components:

Use linked templates to break the solution into individual pieces

Reuse those elements across different deployments

```
"resources": [  
  {  
    "name": "linkedTemplate",  
    "type": "Microsoft.Resources/deployments",  
    "apiVersion": "2018-05-01",  
    "properties": {  
      "mode": "Incremental",  
      "templateLink": {  
        "uri": "https://linkedtemplateek1store.blob.core.windows.net/linkedtemplates/linkedStorageAccount.json?sv=2se=2018-12-31T14%3A32%3A29Z&sp=r"  
      },  
      "parameters": {  
        "storageAccountName": {"value": "[variables('storageAccountName')]"},  
        "location": {"value": "[parameters('location')]" }  
      }  
    }  
  },  
],
```

Managing secrets in templates

(#mw: Also Bicep)

When passing a secure value (e.g., a password) as a parameter during deployment:

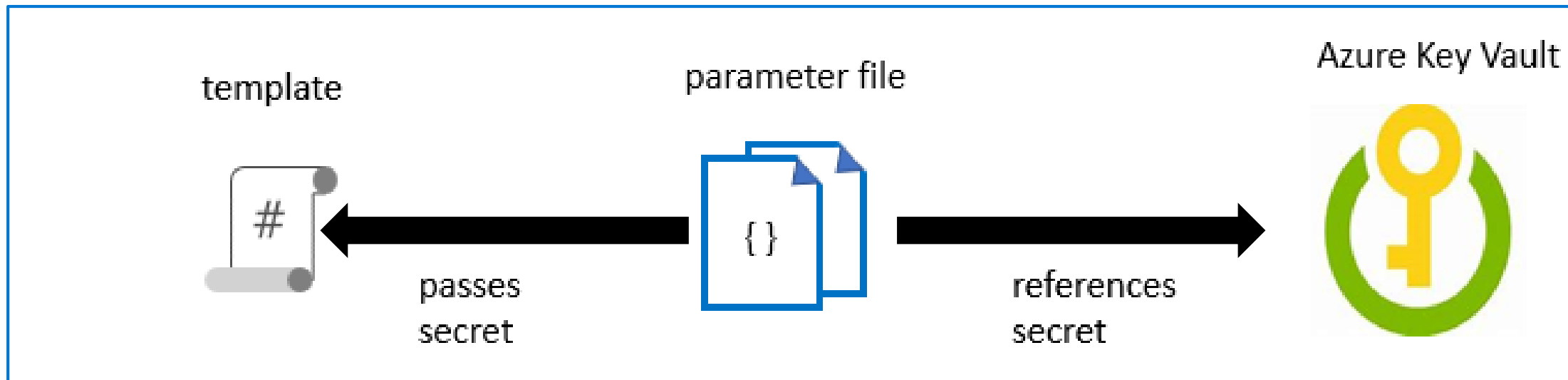
Create a key vault and secret using Azure CLI or PowerShell

Enable Azure Resource Manager access for template deployment

Reference the key pair in the parameter file, **not** the template

Enable access to the secret.

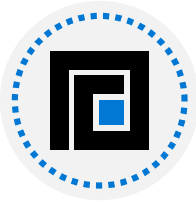
Deploy the template and pass in the parameter file



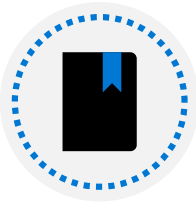
Lesson 04: Create Azure resources using Azure CLI



What is Azure CLI?



Command-line program to connect to Azure (Azure Cloud Shell, PowerShell, or Bash)



Execute administrative commands on Azure resources through a terminal, command-line prompt, or script, instead of a web browser:

For example, to restart a VM use the command:

```
az vm restart -g MyResourceGroup -n MyVm
```

Can be installed on Linux, macOS, or Windows computers, and added as a module to PowerShell



Can be used interactively or scripted:

Interactive. Issue commands directly at the shell prompt

Scripted. Assemble the CLI commands into a shell script and then execute the script

Working with Azure CLI

Commands in the CLI are structured in groups and subgroups:

Use `az find` to find commands you need

`az find blob`

Use the `help` argument to get more detail about the command

`az storage blob --help`

Creating a new Azure resource typically involves the following process:



Demonstration: Run templates using Azure CLI

Process to deploy and verify a sample Azure deployment template with custom script extension using Azure CLI

- 1 Create a resource group to deploy your resources to
- 2 Run **curl** to download the GitHub template
- 3 Validate the template
- 4 Deploy the resource
- 5 Obtain the IP address
- 6 Run **curl** to access your web server and verify the successful deployment and running of the custom script extension

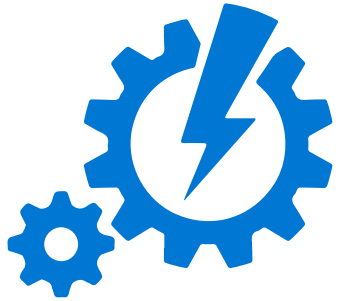


Always remember to delete any resources you deployed when no longer needed to avoid incurring additional costs on them

Lesson 05: Azure Automation with DevOps



What is **Azure** Automation?



An Automation service integrated with Microsoft Azure for automating and simplifying the creation, deployment, monitoring and maintenance of Azure resources and resources external to Azure

Azure Automation Capabilities include:

Manage Shared resources

State configuration

Integration with GitHub, Azure DevOps Git/TFVC

Update management

Can automate Windows or Linux environments

Can apply to any system that exposes an API over internet protocols

Automation accounts

To use Azure Automation you must create an Automation account

Automation account acts as a container in which you store, manage and use automation artifacts

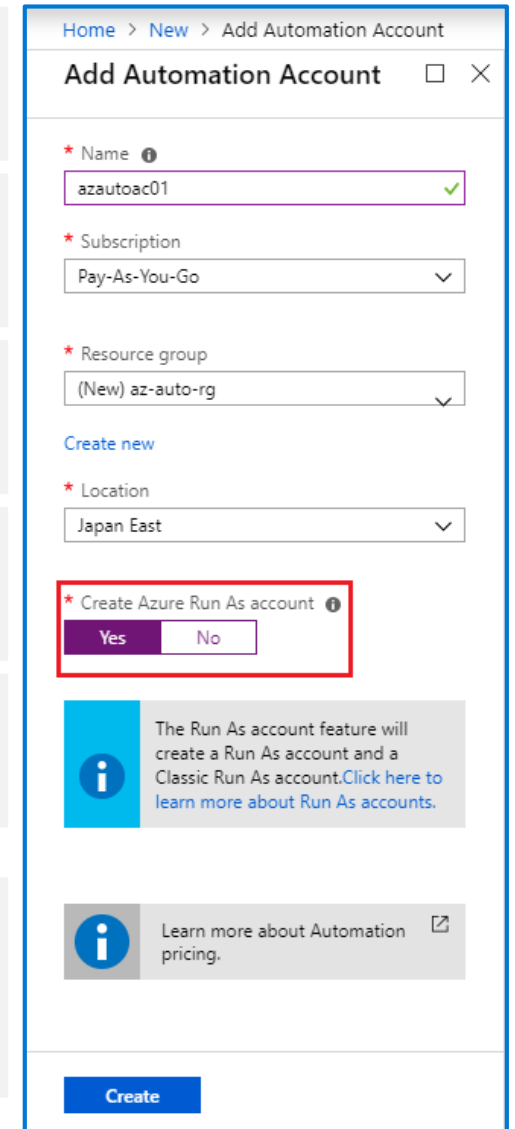
Provides a way to separate your environments or further organize your Automation workflows and resources.

Requires subscription-owner level access as provides access to all Azure resources via an API

Need at least one automation account but should have multiple for access control

Run As account:

Creates a service principal in Azure AD which allows access to Azure resources when running automation



The screenshot shows the 'Add Automation Account' form in the Azure portal. The breadcrumb navigation at the top reads 'Home > New > Add Automation Account'. The form title is 'Add Automation Account' with a close button. The form contains the following fields:

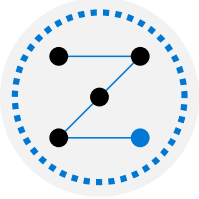
- Name:** A text input field containing 'azautoac01' with a green checkmark icon to its right.
- Subscription:** A dropdown menu showing 'Pay-As-You-Go'.
- Resource group:** A dropdown menu showing '(New) az-auto-rg'.
- Location:** A dropdown menu showing 'Japan East'.
- Create Azure Run As account:** A section with two radio buttons, 'Yes' (selected) and 'No'.

Below the form, there are two informational messages:

- A message with an information icon stating: 'The Run As account feature will create a Run As account and a Classic Run As account. [Click here to learn more about Run As accounts.](#)'
- A message with an information icon stating: 'Learn more about Automation pricing.' with an external link icon.

At the bottom of the form is a blue 'Create' button.

What is a runbook?



A *runbook* is a set of tasks that perform some automated process in Azure Automation



Runbooks serve as repositories for your custom scripts and workflows



Can create your own or import and modify from community via Runbook Gallery



Runbook Types available:

Graphical runbook

PowerShell runbooks

PowerShell Workflow runbooks

Python runbooks

Azure Automation shared resources

Azure Automation contains shared (global) resources to be used in, or with runbooks

Currently Eight Categories:

Schedules

Modules

Modules gallery

Python 2 packages

Credentials

Connections

Certificates

Variables

Shared Resources



Schedules



Modules



Modules gallery



Python 2 packages



Credentials



Connections



Certificates



Variables

Runbook gallery



Can import pre-existing runbooks from the runbook gallery at the **Microsoft Script Center**



Runbooks provided to help eliminate the time it takes to build custom solutions



Already been built by Microsoft and the Microsoft community



Can be used with or without modification



Can review the code or a visualization of the runbook code on the gallery as well as see source projects, rating, etc.



Considerations:

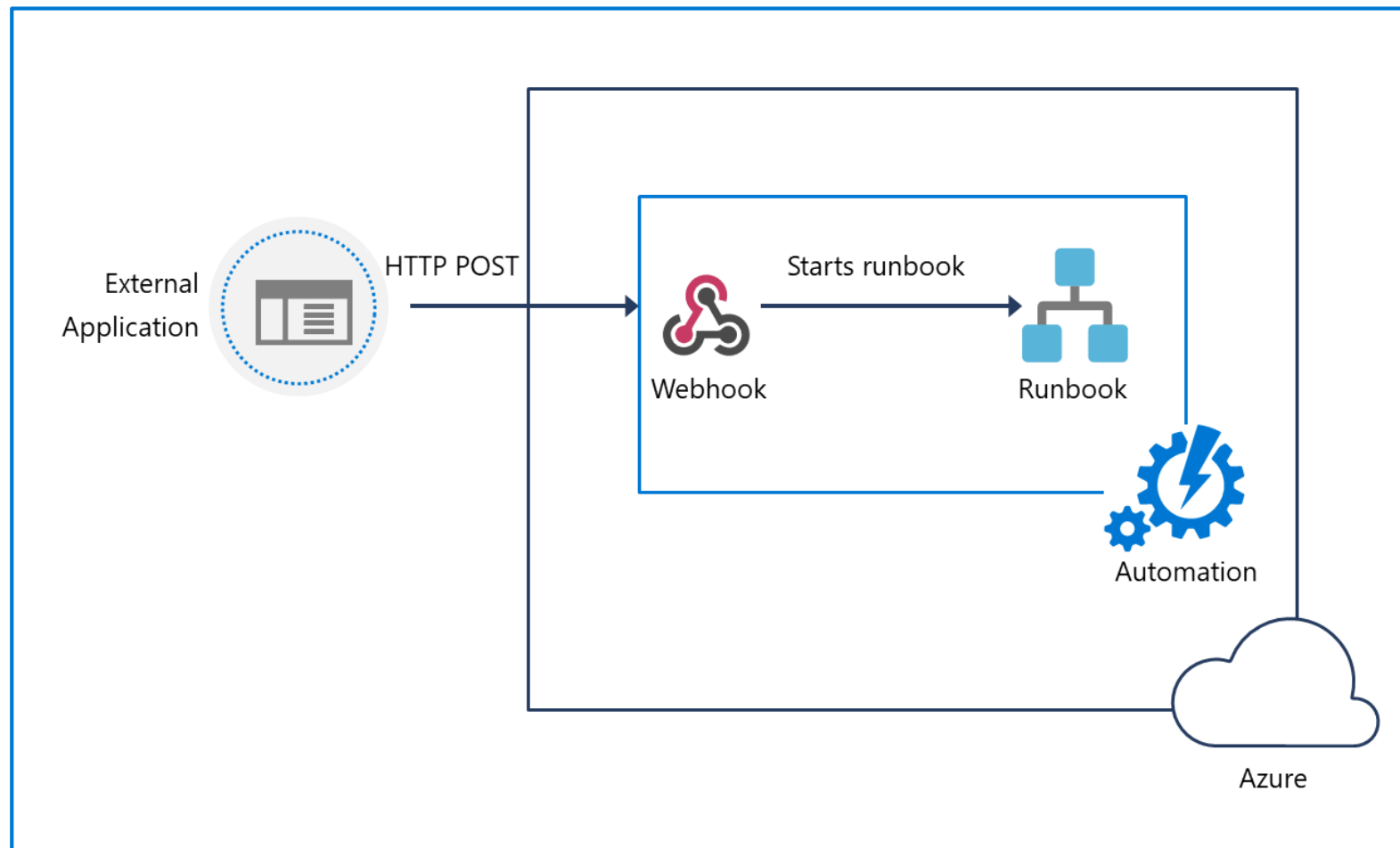
Python runbooks are also available from the script center gallery. To find them, filter by language and select **Python**
You cannot use PowerShell to import directly from the Runbook Gallery

Webhooks

Automate the process of starting a runbook either by scheduling it, or by using a *webhook*.

Uses a HTTPS request to start a runbook

Reduces complexity and allows external services such as Azure DevOps, GitHub, or custom applications to use webhooks



Webhook Syntax: **http://< Webhook Server >/token?= < Token Value >**

Source control integration

(#mw i.e You should add your RunBooks to source control)

Azure Automation supports source control integration

Easier collaboration

Increased auditing and traceability

Roll back to earlier versions of your runbooks

Can push code from Azure Automation to source control or pull your runbooks from source control to Azure Automation

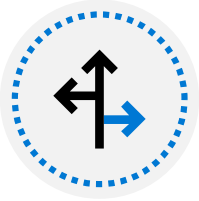
Azure Automation supports the following source Control options:

GitHub

Azure DevOps (Git)

Azure DevOps (TFVC)

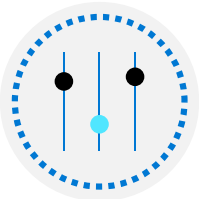
PowerShell workflows



Allows automation and orchestration of multi-environment tasks



Built on PowerShell and based on Windows Workflow Foundation



Characteristics:

Contain *Activities* – which are core a component of a workflow, are a specific task in a workflow

Tasks can be run in parallel

Can be long-running and repeated over and over (idempotent)

Be interrupted—can be stopped and restarted, suspended and resumed.

Continue after an unexpected interruption, such as a network outage or computer/server restart

Creating a PowerShell workflow

There are syntax differences between PowerShell scripts and Workflows

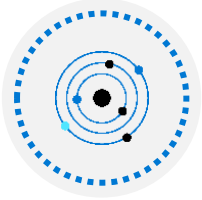
Requires keyword ***workflow*** to identify a workflow command

Add parameter values using keyword ***Param***

```
workflow MyFirstRunbook-Workflow

{
    Param(
        [string]$VMName,
        [string]$ResourceGroupName
    )
    ....
    Start-AzureRmVM -Name $VMName -ResourceGroupName $ResourceGroupName
}
```

Demonstration: Create and run a workflow runbook



This demonstration will create a new PowerShell workflow runbook, test, publish and then run the runbook



You can complete this walkthrough task by completing the steps outlined below, or you can simply read through them, depending on your available time

Checkpoint and parallel processing



Checkpoints:

If a workflow ends in an error or is suspended, it will start from its last checkpoint the next time it runs

You can set a checkpoint in a workflow with the [Checkpoint-Workflow](#) activity

A checkpoint is a snapshot of the current state of the workflow



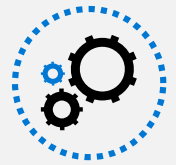
Parallel Processing:

Use the [Parallel](#) keyword to create a script block with multiple commands that run concurrently

Tasks within this script block will be run *concurrently* or in *parallel*

Can use *For Each* with *Parallel* keyword for more granular control on parallelism

Lesson 06: Desired State Configuration (DSC)



Configuration drift



Configuration drift:

Process whereby a set of resources change their state over time
Can occur from changes made by people, processes, or programs



Potential security risks introduced by configuration drift:

Open ports that should have been closed
Inconsistent patching across environments
Software that doesn't meet compliance requirements



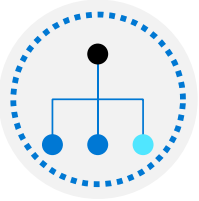
Solutions that can help:

Windows PowerShell Desired State Configuration
Azure Policy
Many non-Microsoft solutions integrated with Azure

Desired State Configuration (DSC)



Ensures that an environment is maintained in a state that you specify (*defined state*), to eliminate configuration drift, and no deviation from that *defined state*

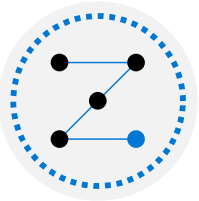


Components:

Configurations: Idempotent declarative PowerShell scripts

Resources: PowerShell scripts compiled into .mof format files to ensure state

Local Configuration Manager (LCM): Client engine to ensure configuration



Methods of Implementing DSC:

Push mode: A user actively pushed out a configuration to environments

Pull mode: Clients get state from remote *pull service* automatically. Remote service is provided by a pull server, which acts as a central control and manager for the configurations

Azure Automation State Configuration



A cloud-based implementation of PowerShell DSC, available as part of Azure Automation



Characteristics:

Built-in pull server: Built-in pull server in Azure Automation eliminates the need to set up and maintain your own pull server

Management of all DSC artifacts: Manage all DSC configurations, resources, and target nodes in single instance in the Azure portal or PowerShell

Ability to Import Reporting Data directly into Azure Log Analytics: Can send this data to your Log Analytics workspace

DSC configuration file

DSC configurations are PowerShell scripts that define a special type of function.

Config File Elements:

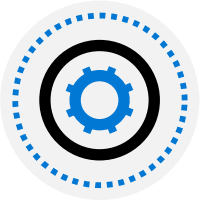
Configuration block: Name of the configuration

Node block: Define nodes being configured e.g. VMs and servers

Resource blocks: Defines the actual configuration state for the nodes

```
configuration LabConfig
{
    Node WebServer
    {
        WindowsFeature IIS
        {
            Ensure = 'Present'
            Name = 'Web-Server'
            IncludeAllSubFeature = $true
        }
    }
}
```

Demonstration: Import and compile

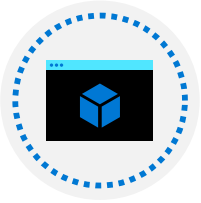


This walkthrough will create a configuration file, will then import that configuration to Azure Automation State configuration (DSC) and then compile the configuration file to create the MOF file



You can complete this walkthrough task by completing the steps outlined below, or you can simply read through them, depending on your available time

Demonstration: Onboarding machines for management



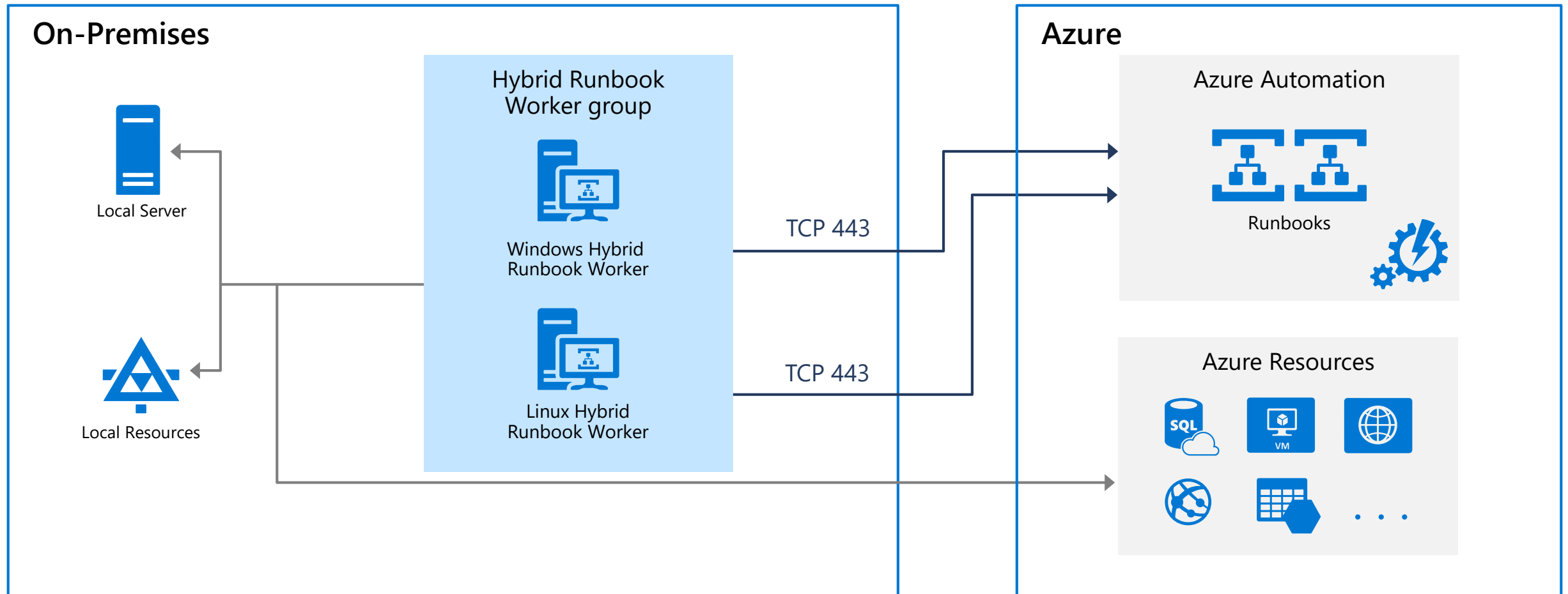
This walkthrough will follow on from the last walkthrough and will on-board virtual machines for management



You can complete this walkthrough task by completing the steps outlined below, or you can simply read through them, depending on your available time

Hybrid management

Hybrid Runbook Worker feature of Azure Automation allows you to run runbooks that manage local resources in your private datacenter, on machines located in your datacenter.



DSC and Linux Automation on Azure



The following Linux operating system versions are currently supported by both PowerShell DSC and Azure Automation DSC:

CentOS 5, 6, and 7 (x86/x64)

Debian GNU/Linux 6, 7 and 8 (x86/x64)

Oracle Linux 5, 6 and 7 (x86/x64)

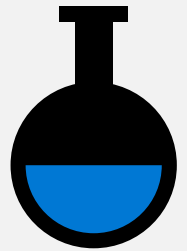
Red Hat Enterprise Linux Server 5, 6 and 7 (x86/x64)

SUSE Linux Enterprise Server 10, 11 and 12 (x86/x64)

Ubuntu Server 12.04 LTS, 14.04 LTS, 16.04 LTS, 18.04 (x86/x64)

Lesson 07: Lab

We're skipping this.
Nothing specific to Azure DevOps here.
Can do outside class time



Deployments using Azure Resource Manager templates



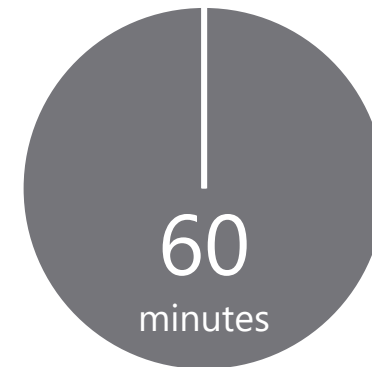
Lab overview:

In this lab, you will create an Azure Resource Manager template and modularize it by using a linked template. You will then modify the main deployment template to call the linked template and updated dependencies, and finally deploy the templates to Azure.

Objectives:

- Create Resource Manager template
- Create a Linked template for storage resources
- Upload Linked Template to Azure Blob Storage and generate SAS token
- Modify the main template to call Linked template
- Modify main template to update dependencies
- Deploy resources to Azure using linked templates

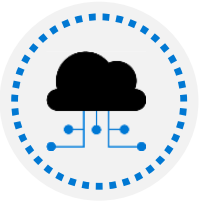
Duration:



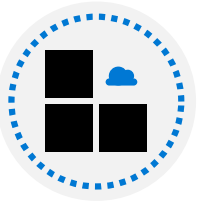
Lesson 8: Module review and takeaways



What did you learn?



Apply infrastructure and configuration as code principles



Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, Azure CLI, DSC and Azure Automation

Module review questions

1

What benefits can you achieve by modularizing your infrastructure and configuration resources?

2

Which method or approach for implementing Infrastructure as Code states what the final state of an environment should be without defining how it should be achieved?

3

Which term defines the ability to apply one or more operations against a resource, resulting in the same outcome every time?

4

Which term is the process whereby a set of resources change their state over time from their original state in which they were deployed?

5

Which Resource Manager deployment mode only deploys whatever is defined in the template, and does not remove or modify any other resources not defined in the template?