|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | C:\Users\kevinm.NORTHAMERICA\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Microsoft Logo.jpg | | |
|  |  |  |  | |
| Migrating to Microsoft Azure and leveraging RDS |  | Azure IaaS  Hackathon Guide  RDS |  |  |
|  |  |  |  |
|  |  |  | **Hands-on Experience** |
|  |  |  |  | |
| Created by:  Rob Kuehfus  Cloud Solutions Architect - OCP  Last Updated: October 5, 2017 |  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  | | |

This document supports a preliminary release of a software product that may be changed substantially prior to final commercial release. This document is provided for informational purposes only and Microsoft makes no warranties, either express or implied, in this document. Information in this document, including URL and other Internet Web site references, is subject to change without notice. The entire risk of the use or the results from the use of this document remains with the user. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in examples herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Copyright 2017 © Microsoft Corporation. All rights reserved.

Microsoft are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.

# Table of Contents

[Table of Contents 3](#_Toc495261402)

[Introduction 4](#_Toc495261403)

[Exercise 1: Architectural Design 6](#_Toc495261404)

[Exercise 2: Using Azure ARM Templates 9](#_Toc495261405)

[Exercise 3: Configure Runtime Automation 14](#_Toc495261406)

[Exercise 4: Monitoring with Azure Log Analytics 19](#_Toc495261407)

# Introduction

The Azure IaaS RDS Hackathon is intended to guide an individual or team through the process of taking an on-prem or datacenter hosted customer deployment and migrate it to Azure. This includes both the solution design, deployment and migration to Azure. This hackathon is structured to help guide you through the solution but not provide a step-by-step walk through. We want “**you”** to be hands-on figuring this out not clicking through a lab. During this hackathon we plan on touching the following Azure technologies:

* Virtual Machines
* Storage
* Azure Site Recovery
* Azure Monitor (Log Analytics \ OMS)
* Azure Automation
* Azure Backup
* Azure Networking
* Remote Desktop Services

This hackathon assumes you have a basic understanding of Azure and its concepts. If you need a quick refresh, please review the following:

Datacenter transformation – a roadmap for platforms, processes, and people  
<https://myignite.microsoft.com/sessions/55827>

Master VMs in Microsoft Azure Resource Manager  
<https://myignite.microsoft.com/videos/54967>

Resilient, cost effective backup and disaster recovery with Azure Blob Storage  
<https://myignite.microsoft.com/videos/55337>

Business continuity for your business running in IaaS  
<https://myignite.microsoft.com/videos/54804>

Azure Disks: Scale, performance, and ease of use  
<https://myignite.microsoft.com/videos/55123>

Microsoft Azure IaaS Deep Dive Jump Start  
<https://mva.microsoft.com/en-US/training-courses/microsoft-azure-iaas-deep-dive-jump-start-8287>

Dive Deep into Networking Storage and Disaster Recovery Scenarios

<https://mva.microsoft.com/en-US/training-courses/dive-deep-into-networking-storage-and-disaster-recovery-scenarios-8901>

## Prerequisites

Before working on this lab, you must have:

* Internet access
* [Free Trial Azure Subscription](https://azure.microsoft.com/en-us/free/?v=17.39a)
* [Install Visual Studio Code](https://code.visualstudio.com/download)
* [Azure PowerShell](https://docs.microsoft.com/en-us/powershell/azure/install-azurerm-ps?view=azurermps-4.4.0)
* Optional: [Install Git client](https://git-scm.com/book/en/v2/Getting-Started-Installing-Git)

## Notes

1. We will be using that credentials
   1. User Name: vmAdmin
   2. Password: G@me0fThrones
2. All hackathon content is published in the following GitHub repository  
   [Azure-IaaS-RDS-Hackathon](https://github.com/rkuehfus/Azure-IaaS-RDS-Hackathon)

# Exercise 1: Architectural Design

## Estimated time to complete this hands-on hack experience

120-180 minutes

## Overview

Design an Azure Infrastructure, deployment and migration.

## Objectives

After completing this hack, you will be able to:

* How to use design an Azure IaaS architecture
* How to estimate the Azure consumption costs
* How to plan out what the on-prem migration would look like
* How to plan the backup and disaster recover options
* Analyze what automation could be used to reduce consumption costs

## Stretch Objectives

Additional scenarios included at the end of the hack include:

* What would this design look like if the customer required high availability?
* TODO: Diagram out an HA solution for the sample scenario

### Prerequisites

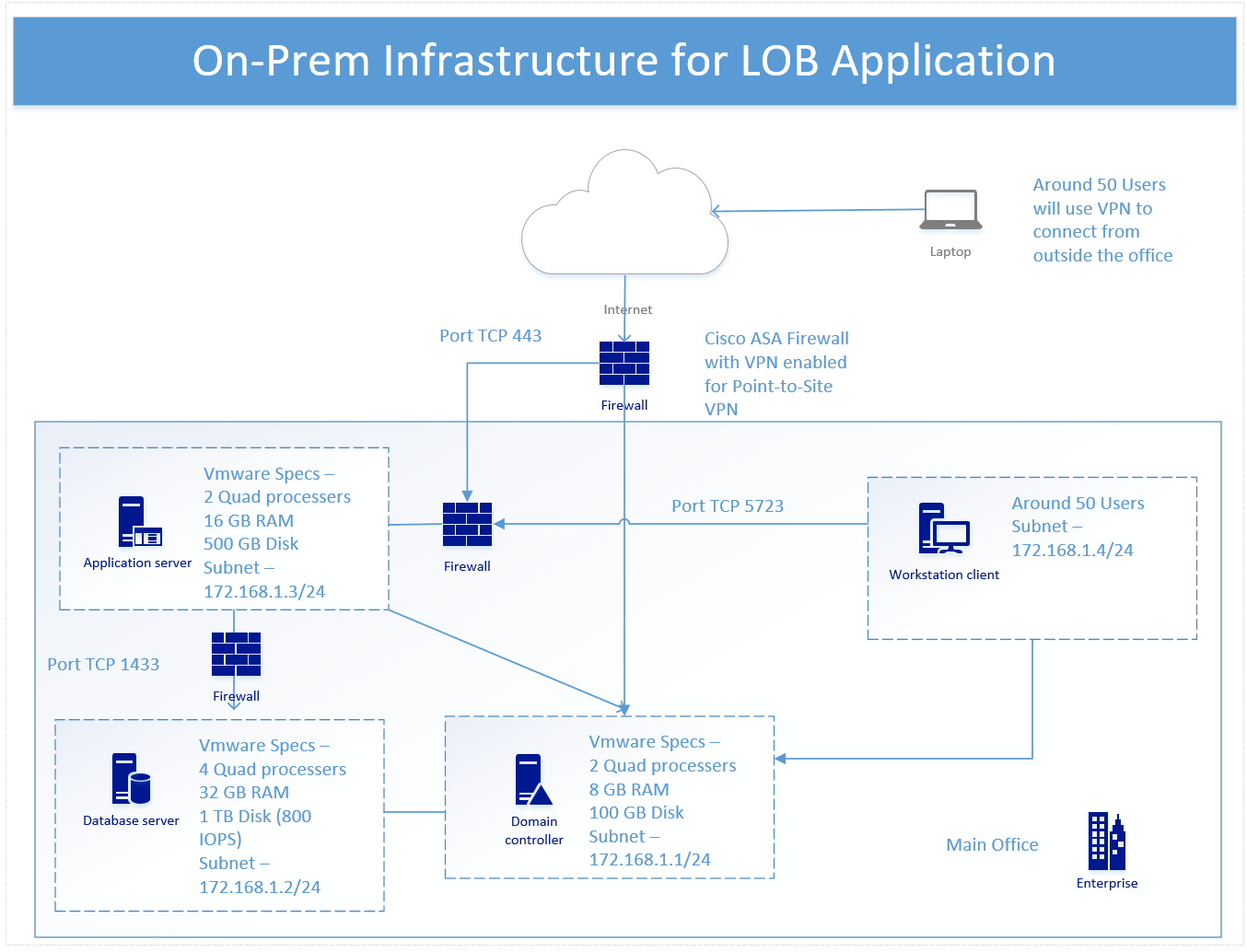
Before working on this lab, you must have:

* Internet access
* Foundational understanding of Azure Infrastructure

## Scenario Sample

Note: It is highly encouraged you bring your own real-world example with customer requirements to the Hackathon but in case you don’t have one now we have drafted a simple one below you may run into.

Customer is would like to move their core LOB application to the cloud to take advantage cloud services as they are currently installed on aging hardware and no longer wish to maintain infrastructure on premise. Due to the way this application is architected they will need to access this application through Remote Desktop Services as it cannot be re-written. This customer requires this application is backed up and accessible Monday – Friday during business hours. Today their current architecture looks like the following:



Please design a solution in Azure. Think through the following items.

* User access
* Security (Access control, patching, auditing, firewalls)
* Pricing (What will the monthly costs look like? How can you lower the cost?)
* Monitoring (How will you know when something is not working?)
* Backup and Disaster Recovery
* Deployment Automation

## Useful Tools and Links

Azure Pricing Calculator - <https://azure.microsoft.com/en-us/pricing/calculator/?&WT.srch=1&WT.mc_id=AID559320_SEM_AeXQozke>

Azure Pricing Calculator CSP - <https://azure.microsoft.com/en-us/pricing/calculator/channel/>

Azure Reference Architectures - <https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/>

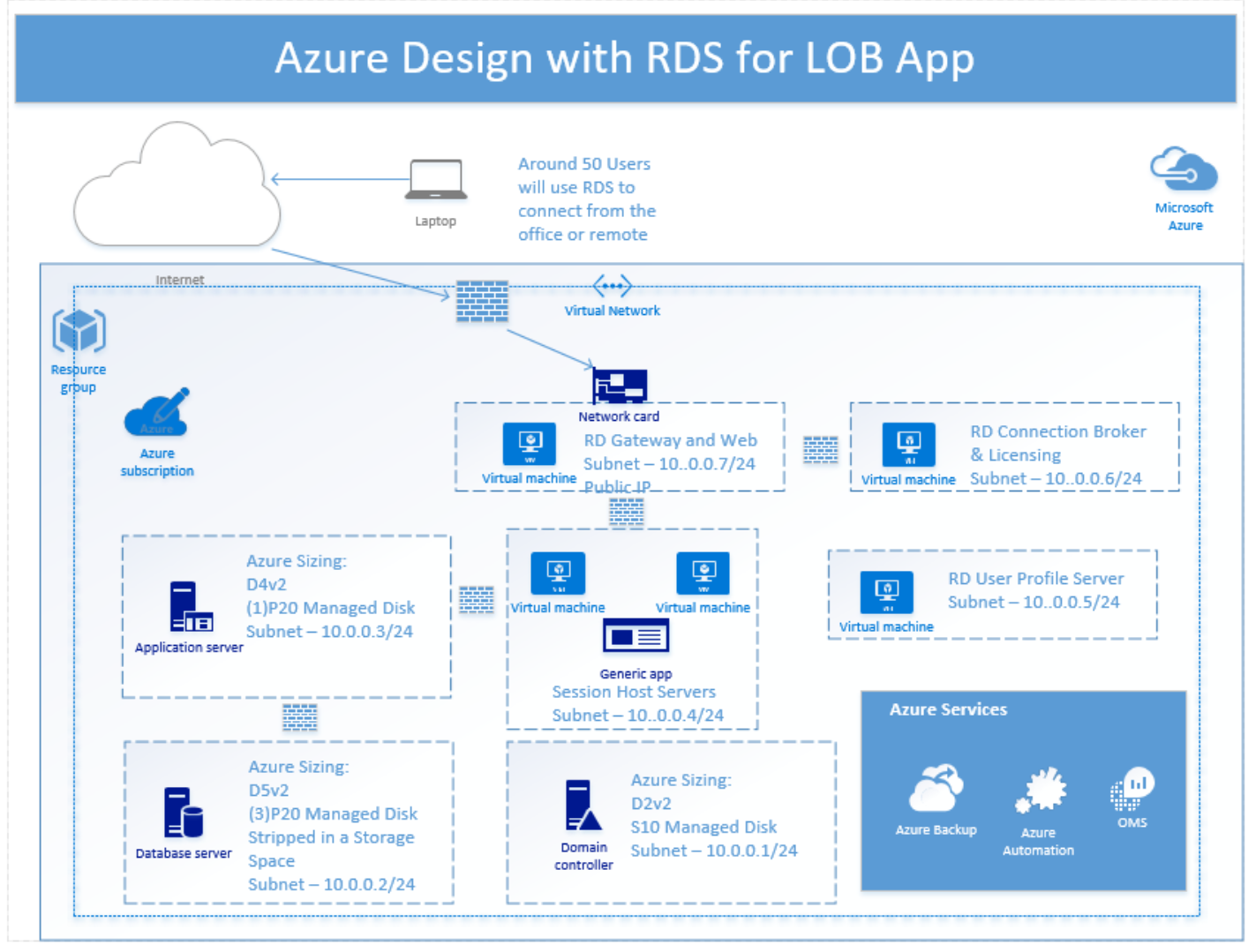
Azure Interactives - <http://azureinteractives.azurewebsites.net/>

Azure Speed Test - <http://azurespeedtest.azurewebsites.net/>

Microsoft Azure Icon Set - [https://buildazure.com/2017/08/22/microsoft-azure-icon-set-download/](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fbuildazure.com%2F2017%2F08%2F22%2Fmicrosoft-azure-icon-set-download%2F&data=02%7C01%7CRobert.Kuehfus%40microsoft.com%7Cff0dfd545beb41f60d4d08d50c4469ec%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C636428410332937429&sdata=BOdsGqiCVpruBXseuf201VY0ckX%2FpShbK4w%2BBtSFEcM%3D&reserved=0)

Azure Migrate (New in Preview) - <https://azure.microsoft.com/en-us/campaigns/azure-migrate/>

## One Possible Design (simple)



# Exercise 2: Using Azure ARM Templates

### Estimated time to complete this hands-on experience

120 minutes

## Overview

This hack is intended to provide you guidance on how to setup Visual Studio Code for creating, editing and deploying Azure ARM templates.

## Objectives

After completing this hack, you will be able to:

* How to use Visual Studio Code to deploy an Azure ARM template
* How to customize an ARM template from GitHub
* How to deploy an ARM template using PowerShell
* How to configure RDS in Azure for testing
* How to Tag your resources post deployment

## Stretch Objectives

Additional scenarios included at the end of the hack include:

* How to configure ARM template for existing deployments
* How to use Azure Key Vault for ARM template password management  
  TODO: Build out these to additional scenarios

## Prerequisites

Before working on this hack, you must have:

* Internet access
* Visual Studio Code
* Azure PowerShell

### Task 1: Configure Visual Studio Code for Azure ARM Templates

Configure Visual Studio Code - <https://blogs.msdn.microsoft.com/azuredev/2017/04/08/iac-on-azure-developing-arm-template-using-vscode-efficiently/>

Also, check out this - <https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-manager-vscode-extension>

**Pro Tip:**

* As you edit your templates dropdowns appear with suggestions about the available values. Once you’ve selected a value you can hit Ctrl+Space to see the list of options. To get this to work, you’ll need to:
  + Open a double bracket "
  + Press Ctrl+Space
  + Select an option from the dropdown
  + For the value, open a double bracket "
  + Press Ctrl+Space
  + Select an option from the dropdown

### Task 2: Download and edit RDS template from GitHub

Download the templates from here to your local machine - https://github.com/Azure/azure-quickstart-templates

You want to start with this ARM template - <https://github.com/Azure/azure-quickstart-templates/tree/master/rds-deployment>

Open this template and parameter file in VS Code

Change the following on the parameter file:

* gwdnsLabelPrefix (make sure it’s something unique)
* adminPassword
* Add numberOfRdshInstances to the parameter file and change its value to 3

Change the following on the template file

* Update rdshVmSize allowed values with what was scoped in your design and update the default value
* Change the adDomainName to something you prefer
* Change the AD VM Size

Review and update the rest of the template how you see fit.

### Task 3: Deploy ARM Template using PowerShell

In VS Code open a PowerShell Terminal and deploy your ARM template to a resource group.

This may help - <https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-template-deploy>

Make sure the VM sizes you use are available in the region the resource group is located in.

* **TIP:** You may need to set the path for the Azure commands to work in Visual Studio Code  
  **$Env:Path += ';C:\Program Files (x86)\Microsoft SDKs\Azure\CLI2\wbin'**

If everything went well you should see an output from the PowerShell deployment:

DeploymentName : WesterosDeployment1

ResourceGroupName : RDSHackathon

ProvisioningState : Succeeded

Timestamp : 10/8/2017 8:10:25 PM

Mode : Incremental

TemplateLink :

Parameters :

Name Type Value

=============== ========================= ==========

gwdnsLabelPrefix String westeroskue

gwpublicIPAddressName String gwpip

adDomainName String westeros.com

adminUsername String vmadmin

adminPassword SecureString

imageSKU String 2016-Datacenter

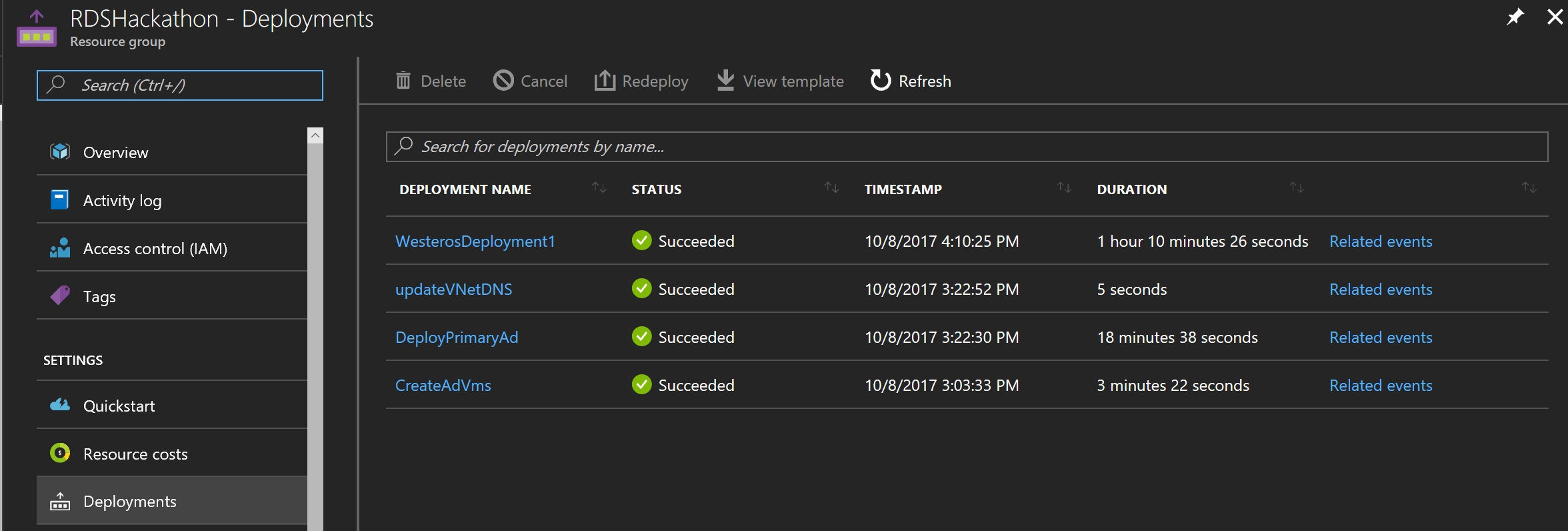
numberOfRdshInstances Int 3

rdshVmSize String Standard\_DS4\_v2

Outputs :

DeploymentDebugLogLevel :

You can check on your deployment status in the Azure Portal by navigating to your Resource Group \ Settings \ Deployments



### Task 4: Configure and Test the RDS deployment

Use the following PowerShell Script to configure your local machine with a cert you can use to test RDS - <https://gallery.technet.microsoft.com/Azure-Resource-Manager-4ea7e328>

Or

Note: I have found the manual instructions to work better

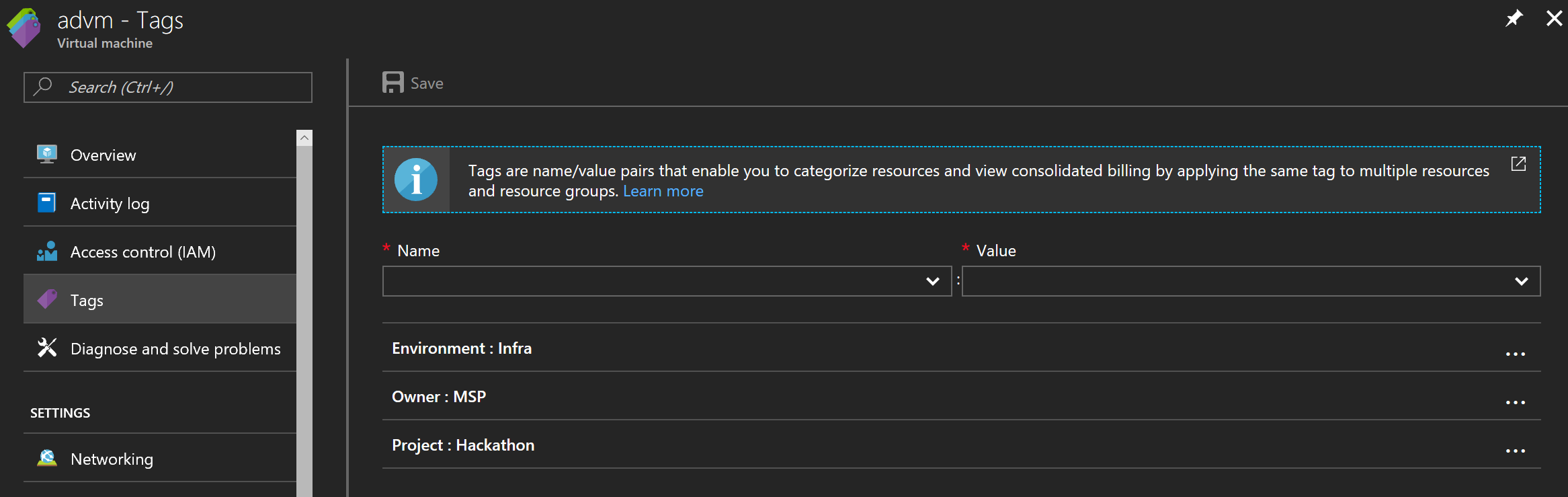
1. Create and install self-signed certificates for the RD Gateway and RD Connection Broker servers. > [!NOTE] > If you are providing and installing certificates from a trusted certificate authority, perform the procedures from step h to step k for each role. You will need to have the .pfx file available for each of these certificates.
   1. In Server Manager, click **Remote Desktop Services > Overview > Tasks > Edit Deployment Properties**.
   2. Expand **Certificates**, and then scroll down to the table. Click **RD Gateway > Create new certificate**.
   3. Enter the certificate name, using the external FQDN of the RD Gateway server (for example, contoso.westus.cloudapp.azure.com) and then enter the password.
   4. Select **Store this certificate** and then browse to the shared folder you created for certificates in a previous step. (For example,\Contoso-Cb1\Certificates.)
   5. Enter a file name for the certificate (for example, ContosoRdGwCert), and then click **Save**.
   6. Select **Allow the certificate to be added to the Trusted Root Certificate Authorities certificate store on the destination computers**, and then click **OK**.
   7. Click **Apply**, and then wait for the certificate to be successfully applied to the RD Gateway server.
   8. Click **RD Web Access > Select existing certificate**.
   9. Browse to the certificate created for the RD Gateway server (for example, ContosoRdGwCert), and then click **Open**.
   10. Enter the password for the certificate, select **Allow the certificate to be added to the Trusted Root Certificate store on the destination computers**, and then click **OK**.
   11. Click **Apply**, and then wait for the certificate to be successfully applied to the RD Web Access server.
   12. Repeat substeps 1-11 for the **RD Connection Broker - Enable Single Sign On** and **RD Connection Broker - Publishing services**, using the internal FQDN of the RD Connection Broker server for the new certificate's name (for example, Contoso-Cb1.Contoso.com).
2. Export self-signed public certificates and copy them to a client computer. If you are using certificates from a trusted certificate authority, you can skip this step.
   1. Launch certlm.msc.
   2. Expand **Personal**, and then click **Certificates**.
   3. In the right-hand pane right-click the RD Connection Broker certificate intended for client authentication, for example **Contoso-Cb1.Contoso.com**.
   4. Click **All Tasks > Export**.
   5. Accept the default options in the Certificate Export Wizard accept defaults until you reach the **File to Export** page.
   6. Browse to the shared folder you created for certificates, for example \Contoso-Cb1\Certificates.
   7. Enter a File name, for example ContosoCbClientCert, and then click **Save**.
   8. Click **Next**, and then click **Finish**.
   9. Repeat substeps 1-8 for the RD Gateway and Web certificate, (for example contoso.westus.cloudapp.azure.com), giving the exported certificate an appropriate file name, for example **ContosoWebGwClientCert**.
   10. In File Explorer, navigate to the folder where the certificates are stored, for example \Contoso-Cb1\Certificates.
   11. Select the two exported client certificates, then right-click them, and click **Copy**.
   12. Paste the certifcates on the local client computer.

### Task 5: Configure Azure Tags on Resources deployed

Using Tags helps to keep track of your deployments and apply some useful automation.

Use the PowerShell Script “T agAllResourcesInRGorSingleVM.ps1 “published in this hackathon’s GitHub repository to configure Azure tags on newly deployed resources.

Once the script has completed have a look at one of the resources in the Resource Group and it should look something like this -



# Exercise 3: Configure Runtime Automation

## Estimated time to complete this hands-on experience

90 minutes

## Overview

This hack is intended to help you configure automation that starts and stops the VMs on a schedule. This will help you to keep consumptions costs down.

## Objectives

After completing this hack, you will be able to:

* How to create an Azure Automation account with Runas account
* How to create a runbook and schedule

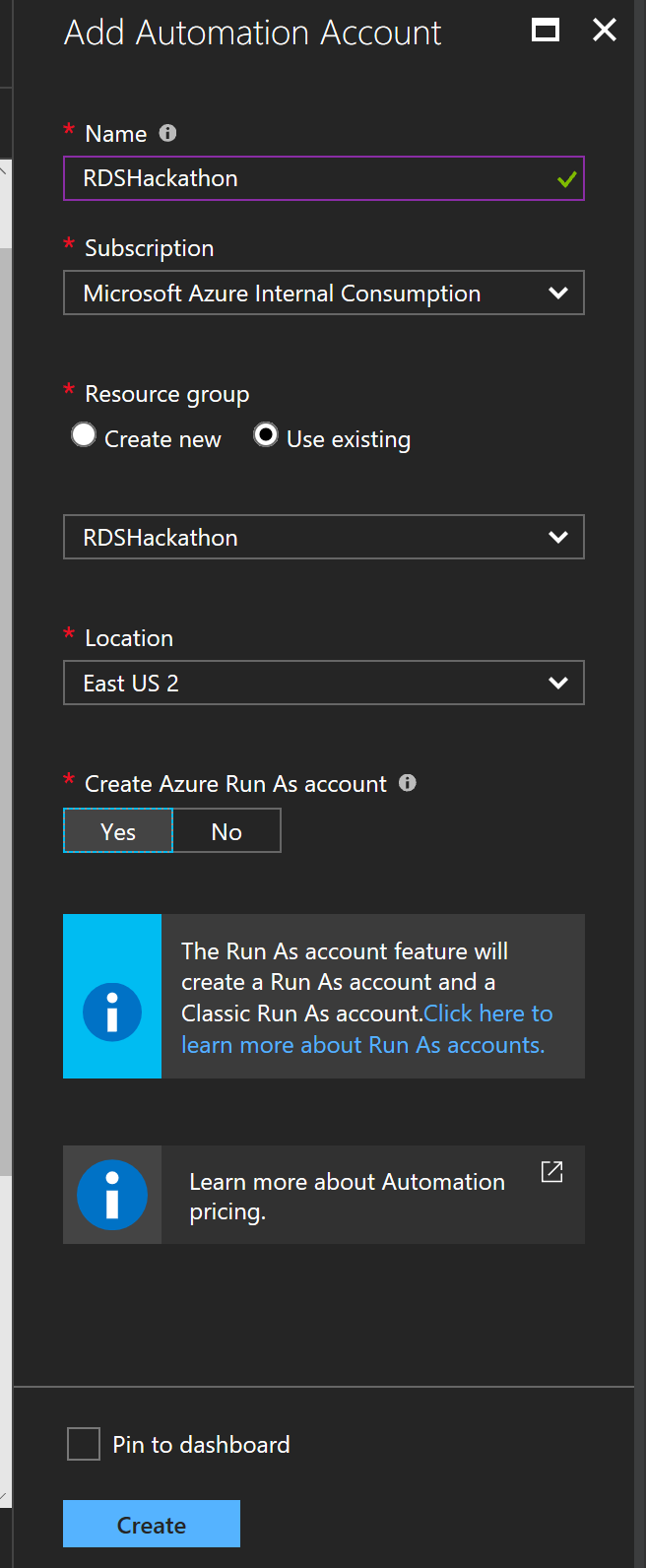
## Stretch Objectives

Additional scenarios included at the end of the hack include:

* None

### Task 1: Create an Automation Account

In the Azure portal create a new Automaton account in the resource group you are using for this hack.

  
Make sure to create an Azure Run As account. This Run As account will be used in our PowerShell runbooks to control our infrastructure.

### Task 2: Setup Runbooks for Starting and Stopping VMs

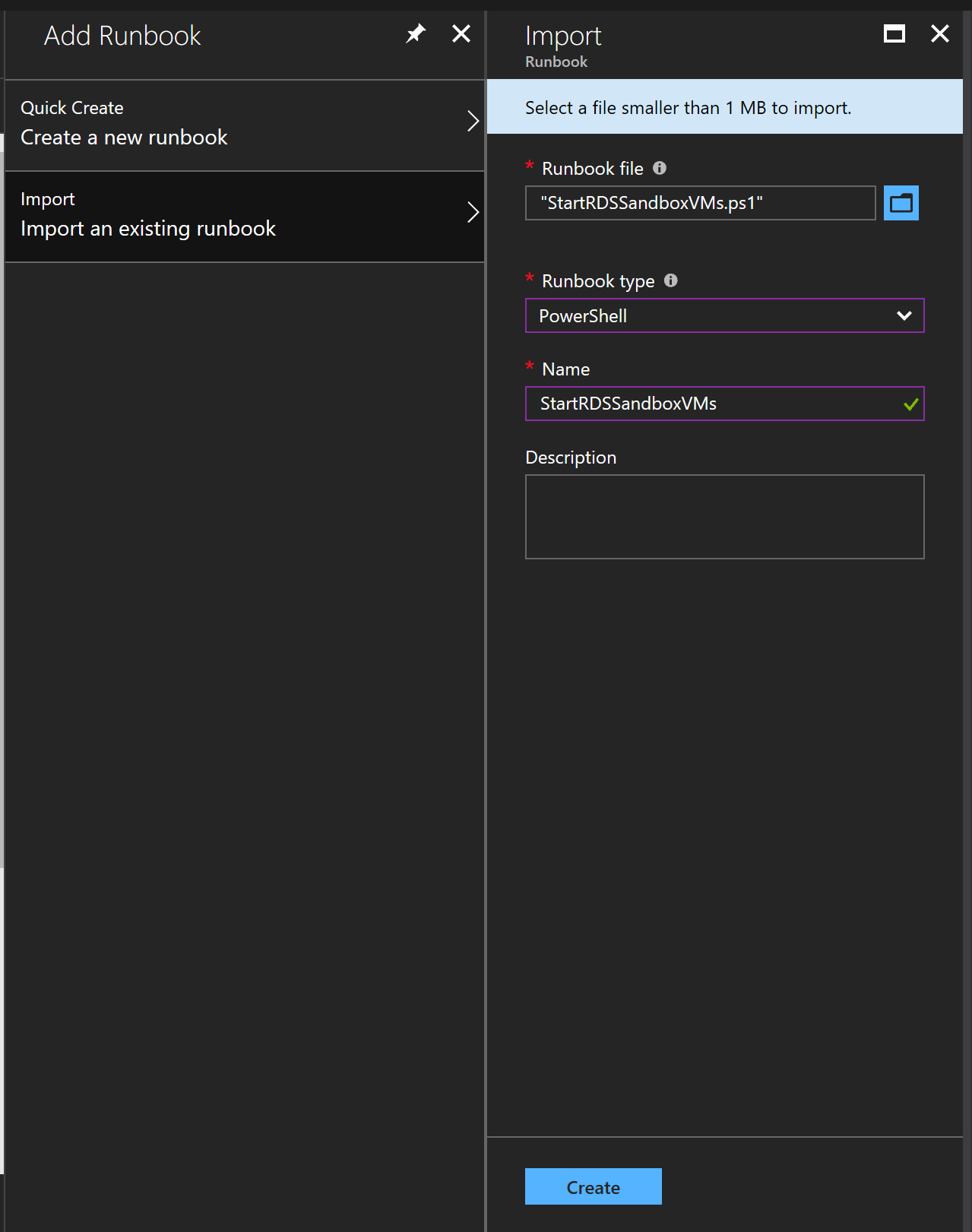
Download the following runbooks from our GitHub repository:

StartTaggedVMs.ps1

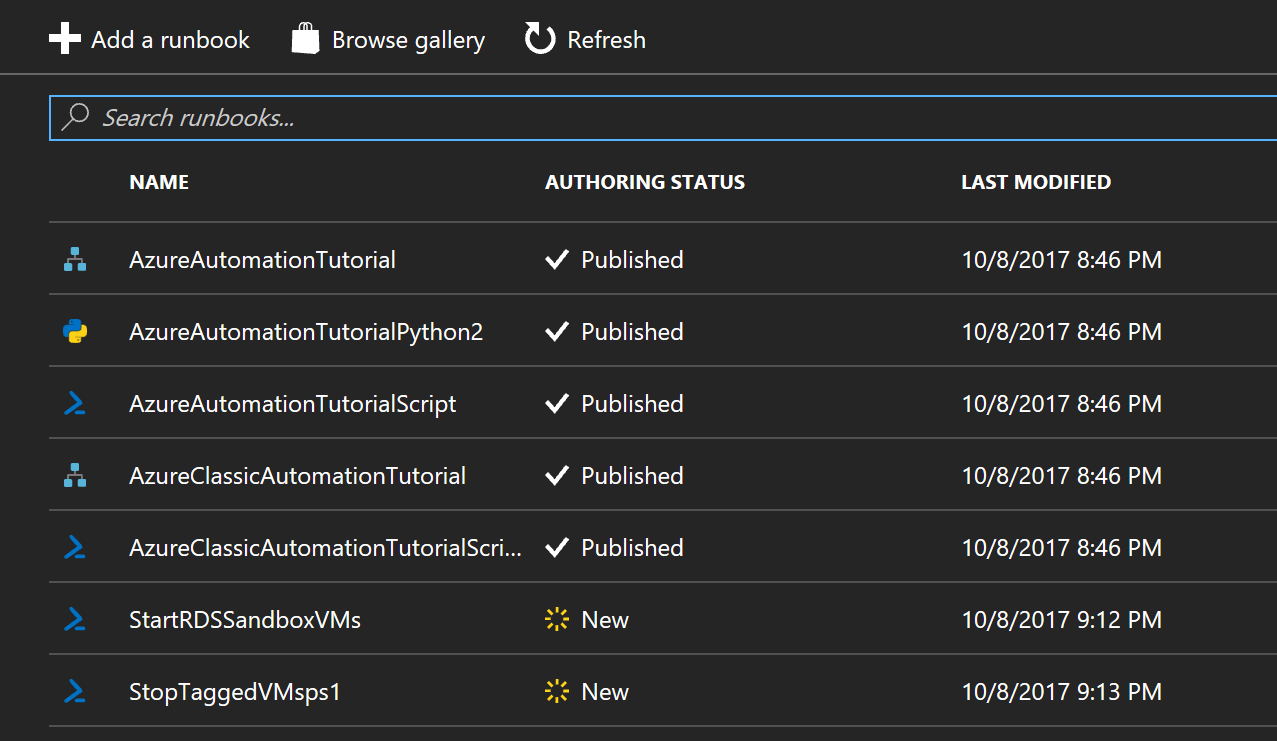
StopTaggedVMs.ps1

Edit each PowerShell script’s Resource Group and Subscription Id variable with yours.

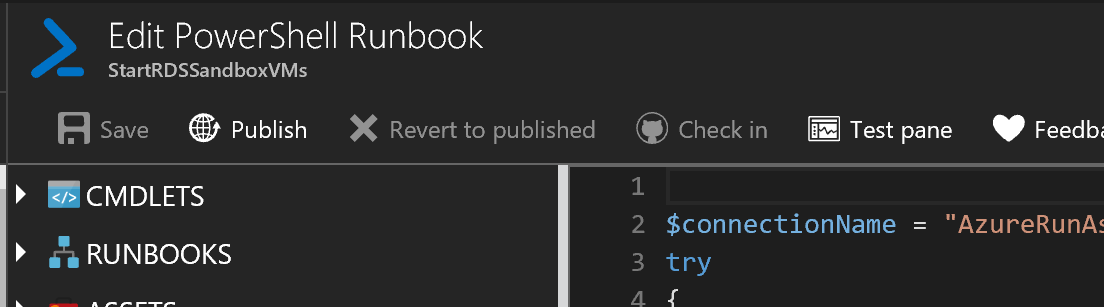
From your Azure Automation account add these two scripts as runbooks



You should have both runbooks listed like so,



Last, we need to publish these by going to edit and clicking on publish.

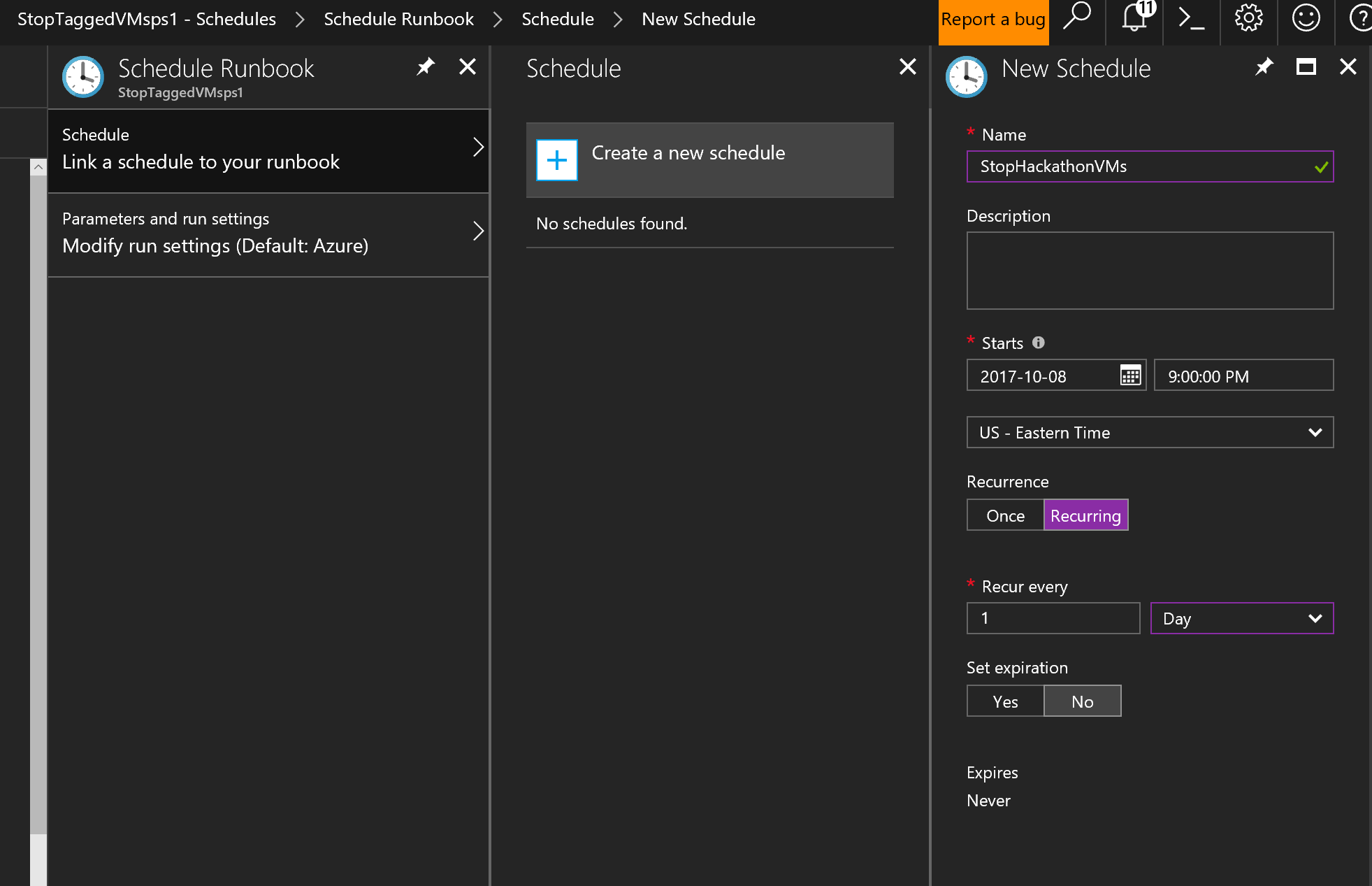


### Task 3: Schedule Runbooks to control consumption

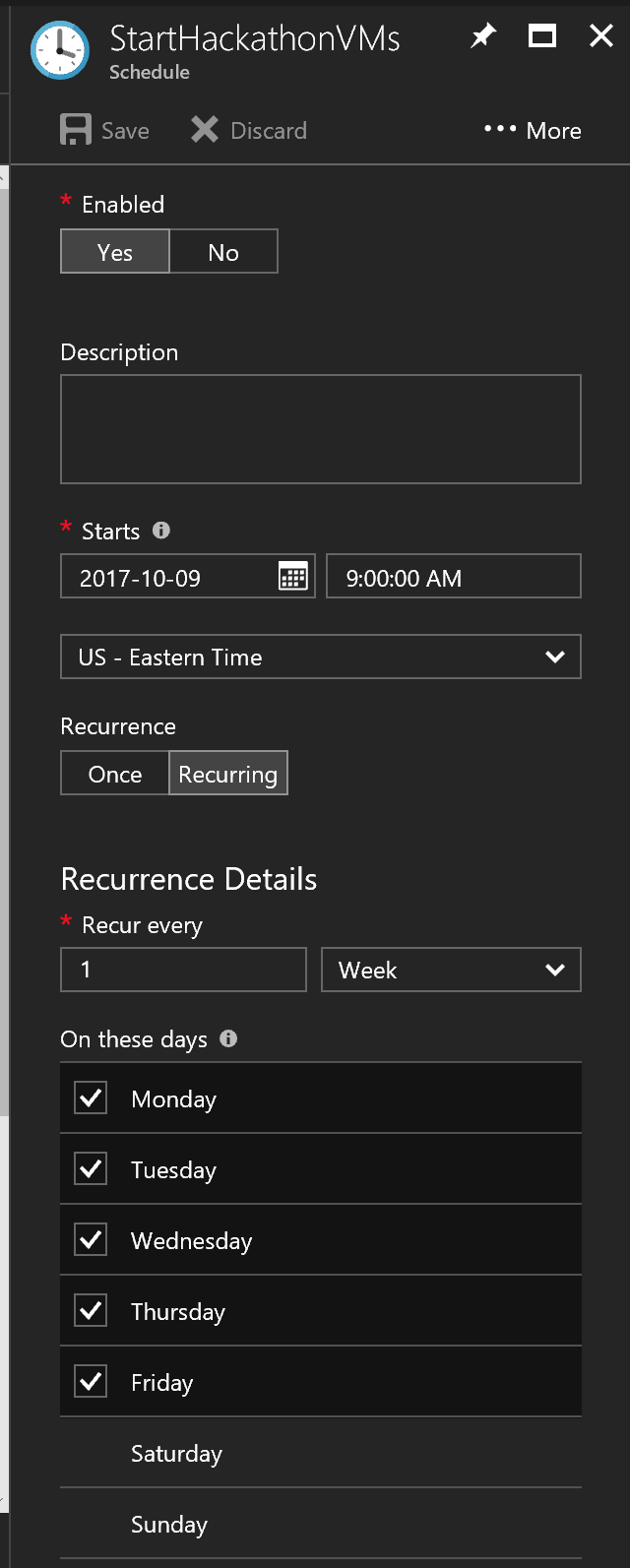
Now all we need to do is create a few schedules and let Azure take care of the rest for making sure we only have these running when we want.

For each runbook go to schedule runbook and create a new recurrence schedule based on your requirements. I have an example below.

StopTaggedVMs



StartTaggedVMs



You will notice I only start these on weekdays.

# Exercise 4: Monitoring with Azure Log Analytics

## Estimated time to complete this hands-on experience

120 minutes

## Overview

Now that we have an Azure deployment we need to monitor the environment to make sure we know if it’s healthy.

## Objectives

After completing this hack, you will be able to:

* How to create a Log Analytics Workspace
* How to deploy monitoring agents
* Which solutions should be enabled
* How to add new log and performance counts
* How to search the logs for valuable connection information
* How to create an Alert that kicks off automation to resolve the issue

## Stretch Objectives

Additional scenarios included at the end of the hack include:

* TBD

## Task 1: Create Log Analytics Workspace

Have a look here and create a Log Analytics workspace - <https://docs.microsoft.com/en-us/azure/log-analytics/>

Note: Free training can be found on Pluralsight - <https://www.pluralsight.com/courses/azure-iaas-monitoring-management-getting-started>

## Task 2: Deploy Monitoring Agents using what we learned above

Review the following documentation - <https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-azure-vm-extension>

Either create a PowerShell script that deploys the monitoring agent to all VMs tagged with Hackathon or update your ARM template to include the extension for deploying log analytics.

## Task 3: Deploy and configure common Solutions

Solution documentation can be found here - <https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-add-solutions>

Deploy and configure the following solutions to your workspace:

* Activity Log Analytics – Connect your storage account to your workspace to pull in more data
* Agent Health
* Alert Management
* Change Tracking
* Network Performance Monitor – Use different subnets to test this
* Security Audit
* Service Map – Requires you to deploy agents to the machines
* Update Management – Setup a patching cycle

## Task 4: Configure Workspace to collect performance and event logs for RDS

Configure Log Analytics to collect performance counter for the RDS Connection Brocker

|  |  |
| --- | --- |
| **Perf counter** | **Description** |
| Total Connection | The total number of connections handled by the Connection Brokers since the service started. |
| Successful Connections | The total number of successful connections since the service started. |
| Failed Connections | The total number of failed connections since the service started. |
| Pending connections | The total number of redirected connections pending completion from the Remote Desktop Session Host in the farm. |

<https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-data-sources-performance-counters>

## Task 4: Setup an Alert of the number of successful connections

For this hack we will use an absurdly low number, so we can test easily. Create an alert that send and email if the number of successful connections goes over 1 in a 5-min span of time.

Note: You should create a few test users in Active Directory.

<https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-alerts>

Also, it is recommended you get your query right first before setting up the alert.