Ready Pre-day Azure Monitoring Workshop Proctor Guide

A screenshot of a cell phone

Description automatically generated

Version 1.0

Author: Rob Kuehfus

Please send feedback to here - https://github.com/rkuehfus/pre-ready-2019-H1

Contents

[Hack Setup 3](#_Toc531674743)

[Guidance 3](#_Toc531674744)

[Day 1: Monitoring and Alert Rule Challenge 4](#_Toc531674745)

[Day 1: Monitoring and Alert Rule Automation Challenge 26](#_Toc531674746)

[Day 2: Log Analytics Setup Challenge 30](#_Toc531674747)

[Day 2: Log Analytics Dashboard Challenge 46](#_Toc531674748)

[Day 3: Application Insights Challenges 55](#_Toc531674749)

# Workshop Setup

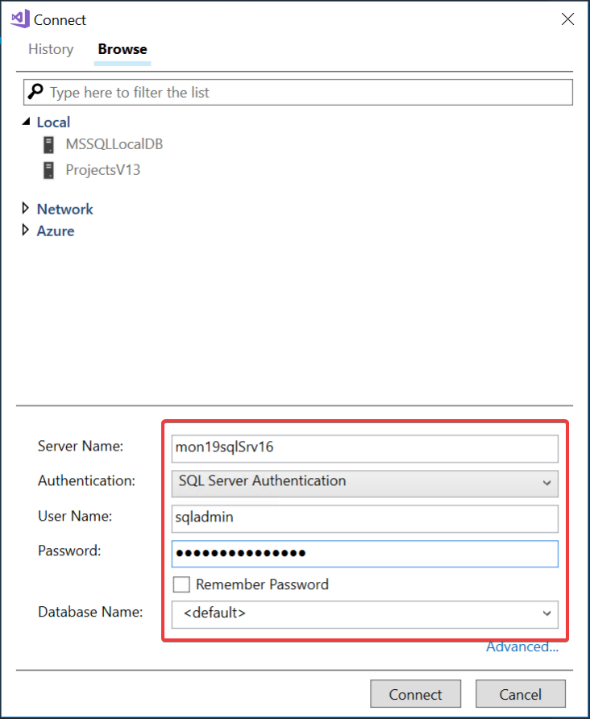
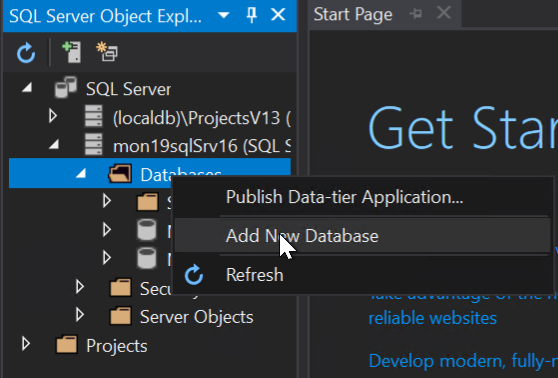
* Deploy Infra using Bash Cloud Shell and Azure CLI with an ARM Template
  + Setup Azure CLI  
    <https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest>
  + Install Visual Studio Code and Extensions (depending on your tool of choice)
    - Azure Resource Manager Tools - <https://marketplace.visualstudio.com/items?itemName=msazurermtools.azurerm-vscode-tools>
    - Azure Account and Sign-In (adds the Azure Cloud Shell for Bash) - <https://marketplace.visualstudio.com/items?itemName=ms-vscode.azure-account>  
      Azure CLI Tools –   
      <https://marketplace.visualstudio.com/items?itemName=ms-vscode.azurecli>
  + Download Azure Monitoring Workshop Setup Guide and follow the instructions to deploy.  
    <https://github.com/rkuehfus/pre-ready-2019-H1/blob/master/Student/Guides/Deployment%20Setup%20Guide.docx?raw=true>

## Guidance

Biggest issues I have seen when deploying the workshop are around compute quotas, changing the region to one that does not support the Azure Monitor preview, not having contributor role membership to the subscription and local machine issues. If you run into a student who cannot seem to get her or his computer working with the Azure CLI, have them use the Azure Cloud Shell. Remember to copy up the ARM template and parameters JSON files before kicking off the deployment.

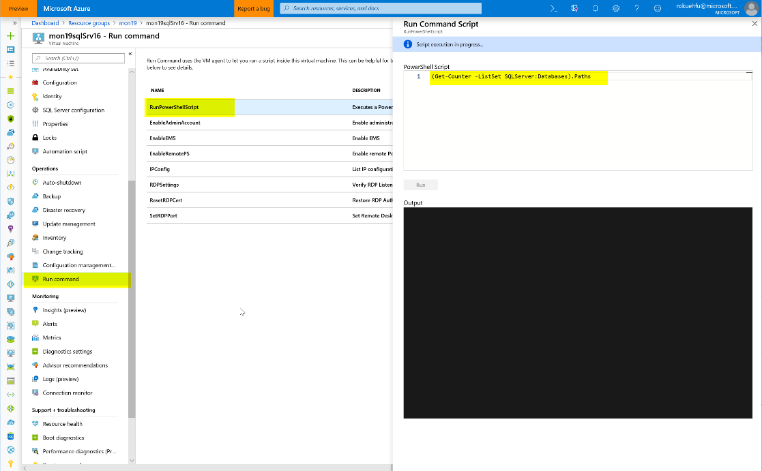
I highly recommend your students make sure they have an Azure Subscription they have contributor role access to before day one of the hack.

# Challenge 1: Monitoring and Alert Rule

* Create an empty database called “tpcc” on the SQL Server  
  Note: Use SQL Auth with the username being sqladmin and password being whatever you used during deployment
  + Connect (RDP) to the Visual Studio Server (xxxxxVSSrv17) using its public IP address and open Visual Studio. Create an account if you don’t have one.
  + VS has view called SQL Server Object Explore that can be used to create and delete SQL databases on the SQL server  
    
  + Connect to the database server, make sure to use sqladmin and the password you stored in the key vault during deployment  
      
      
    Once connected create a new database called “tpcc”



* From the ARM template, send the below guest OS metric to Azure Monitor for the SQL Server
  + Add a Performance Counter Metric for
    - Object: SQLServer:Databases
    - Counter: Active Transactions
    - Instance:tpcc
  + Hint: <https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/metrics-store-custom-guestos-resource-manager-vm>
  + First, figure out the correct format for the counter use the run command on the SQL Server in the Azure portal and run –



Run the command - (Get-Counter -ListSet SQLServer:Databases).Paths

Once its finished, review the results (scroll up) and copy the output for the SQLServer:Databases counter.  


\SQLServer:Databases(\*)\Active Transactions

Then change it to target just your specific database

\SQLServer:Databases(tpcc)\Active Transactions

**Tip:** Share the following link to help lead them to how to find the counter

<https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.diagnostics/get-counter?view=powershell-5.1>

Next, once you have the counter you need to modify the ARM template for the SQL Server to add the collection of this counter that sends it to Azure Monitor using the Azure monitor data sink

Add this JSON to the SQL Server code

Verify the identity is present in the template (add it if its missing)

"identity": {

"type": "SystemAssigned"

},

And the missing counter below

"resources":[

{

"type": "extensions",

"name": "Microsoft.Insights.VMDiagnosticsSettings",

"apiVersion": "2015-05-01-preview",

"location": "[resourceGroup().location]",

"dependsOn": [

"[concat('Microsoft.Compute/virtualMachines/', concat(parameters('envPrefixName'), 'sqlSrv16'))]"

],

"properties": {

"publisher": "Microsoft.Azure.Diagnostics",

"type": "IaaSDiagnostics",

"typeHandlerVersion": "1.5",

"autoUpgradeMinorVersion": true,

"settings": {

"WadCfg": {

"DiagnosticMonitorConfiguration": {

"overallQuotaInMB": 4096,

"DiagnosticInfrastructureLogs": {

"scheduledTransferLogLevelFilter": "Error"

},

"Directories": {

"scheduledTransferPeriod": "PT1M",

"IISLogs": {

"containerName": "wad-iis-logfiles"

},

"FailedRequestLogs": {

"containerName": "wad-failedrequestlogs"

}

},

"PerformanceCounters": {

"scheduledTransferPeriod": "PT1M",

"sinks": "AzMonSink",

"PerformanceCounterConfiguration": [

{

"counterSpecifier": "\\Memory\\Available Bytes",

"sampleRate": "PT15S"

},

{

"counterSpecifier": "\\Memory\\% Committed Bytes In Use",

"sampleRate": "PT15S"

},

{

"counterSpecifier": "\\Memory\\Committed Bytes",

"sampleRate": "PT15S"

},

{

"counterSpecifier": "\\SQLServer:Databases(tpcc)\\Active Transactions",

"sampleRate": "PT15S"

}

]

},

"WindowsEventLog": {

"scheduledTransferPeriod": "PT1M",

"DataSource": [

{

"name": "Application!\*"

}

]

},

"Logs": {

"scheduledTransferPeriod": "PT1M",

"scheduledTransferLogLevelFilter": "Error"

}

},

"SinksConfig": {

"Sink": [

{

"name": "AzMonSink",

"AzureMonitor": {}

}

]

}

},

"StorageAccount": "[variables('storageAccountName')]"

},

"protectedSettings": {

"storageAccountName": "[variables('storageAccountName')]",

"storageAccountKey": "[listKeys(variables('accountid'),'2015-06-15').key1]",

"storageAccountEndPoint": "https://core.windows.net/"

}

}

},

{

"type": "extensions",

"name": "WADExtensionSetup",

"apiVersion": "2015-05-01-preview",

"location": "[resourceGroup().location]",

"dependsOn": [

"[concat('Microsoft.Compute/virtualMachines/', concat(parameters('envPrefixName'), 'sqlSrv16'))]"

],

"properties": {

"publisher": "Microsoft.ManagedIdentity",

"type": "ManagedIdentityExtensionForWindows",

"typeHandlerVersion": "1.0",

"autoUpgradeMinorVersion": true,

"settings": {

"port": 50342

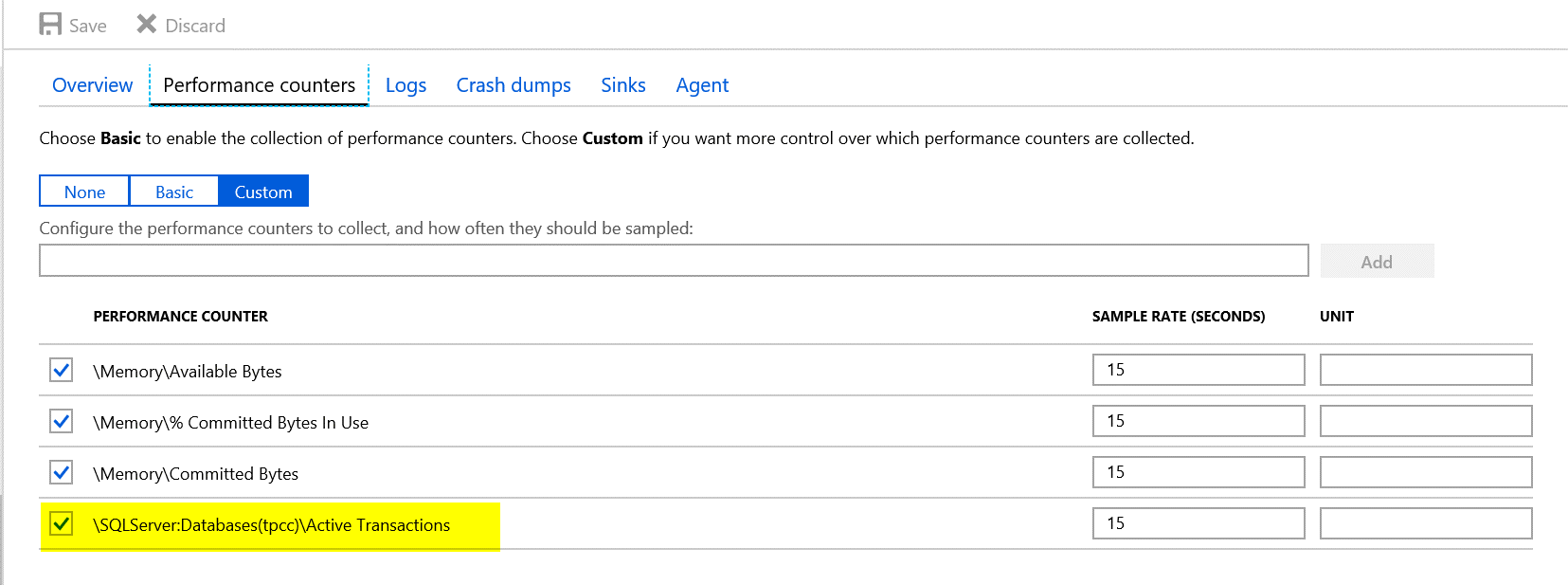
}

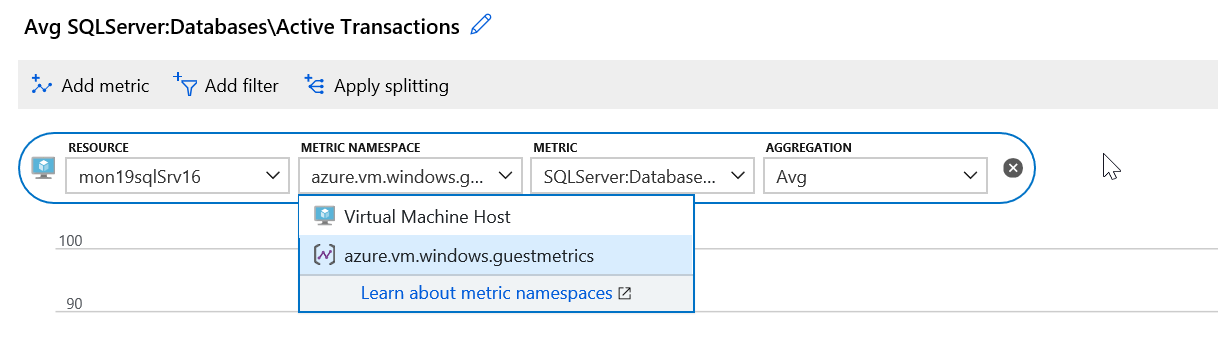
}

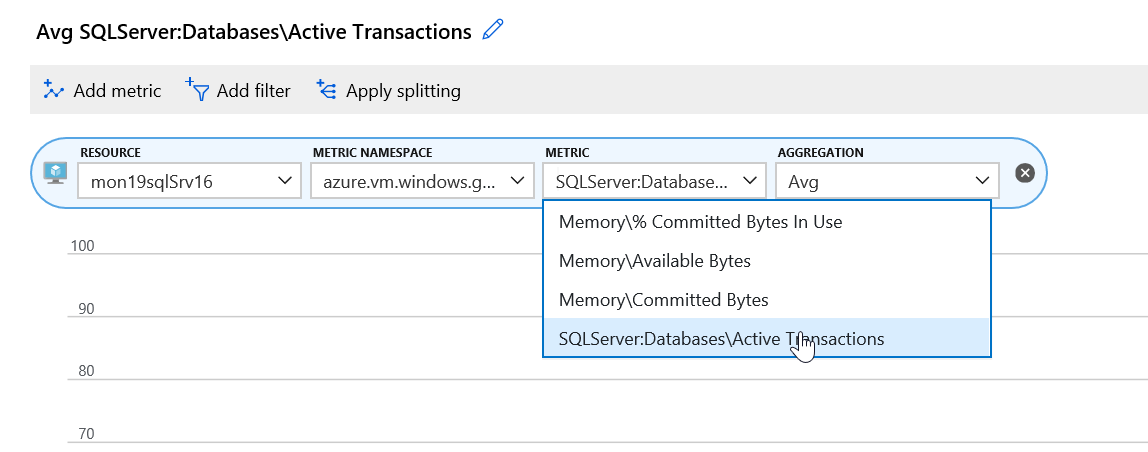
}

]

Save and redeploy.

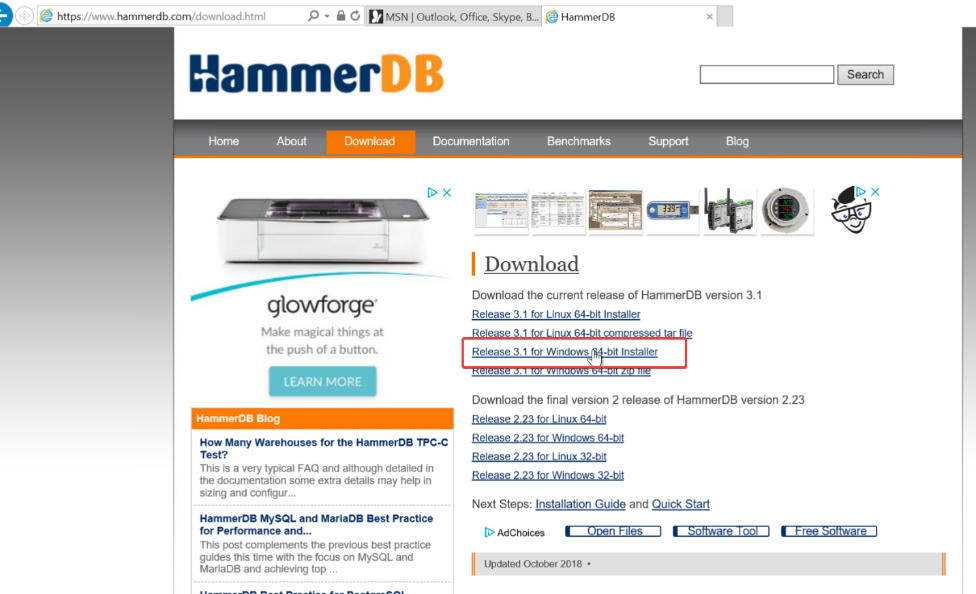
Once redeployed, go to metrics and check to make sure you are seeing the new metrics.   


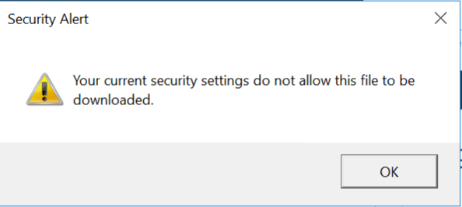




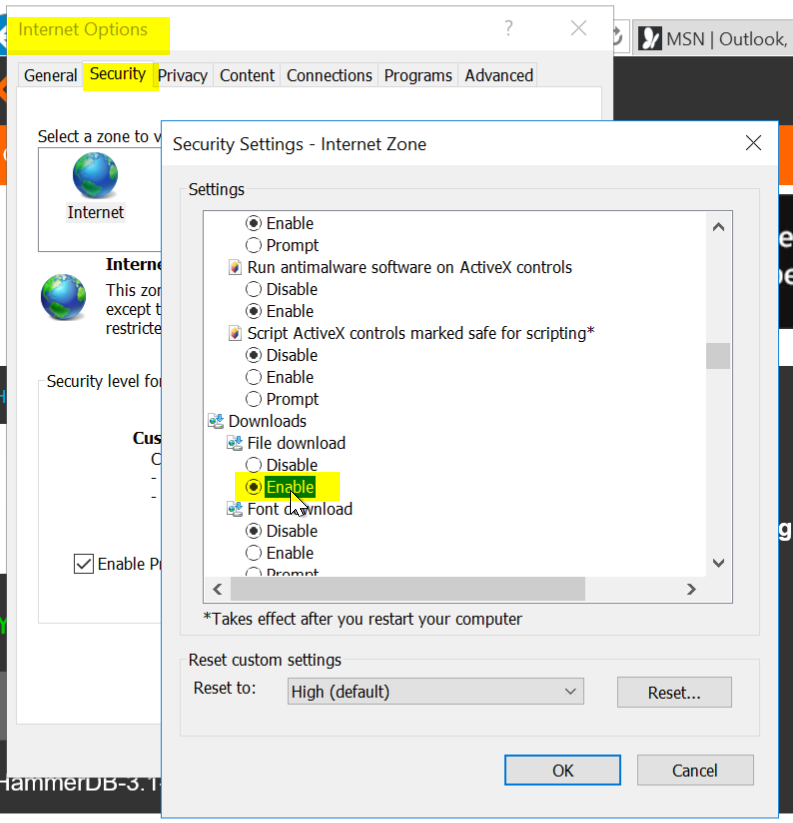
**Tip:** A bunch of OS metrics are configured already under the scale set as a sample.

* Download and Install HammerDB tool on the Visual Studio VM. Note: I copy of these instructions are in the student folder under “AzureMonitoringHackathon\Student\Guides\Day-1”
  + [www.hammerdb.com](http://www.hammerdb.com/)

From the Visual Studio Server, download the latest version of HammerDB  




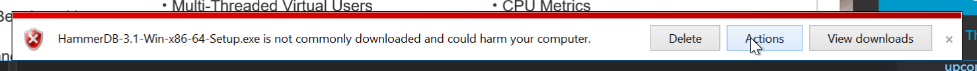
If you get this Security Warning, go to Internet Options.



Security \ Security Settings \ Downloads \ File download \ Enable

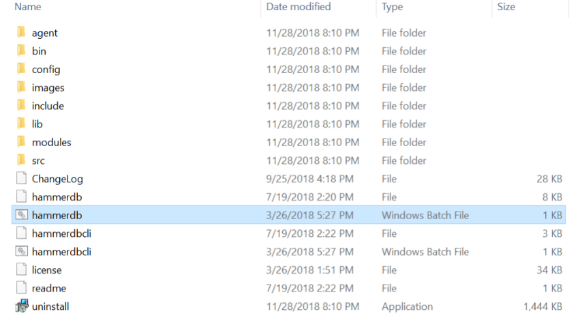
Click OK

Try again

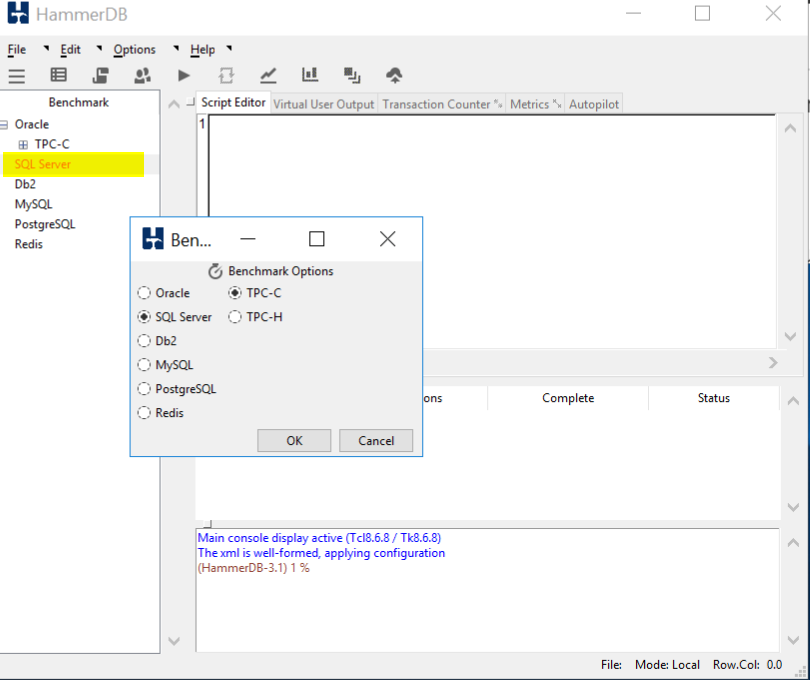


Click Actions and accept the warnings

Tip: If you end up closing HammerDB you have to go to C:\Program Files\HammerDB-3.1 and run the batch file



* Use HammerDB to create transaction load



Double click on SQL Server and click OK, and OK on the confirm popup



Drill into SQL Server \ TPC-C \ Schema Build and double click on Options

Modify the Build Options for the following:

SQL Server: Name of your SQL Server

SQL Server ODBC Driver: SQL Server

Authentication: SQL Server Authentication

SQL Server User ID: sqladmin

SQL Server User Password: <password you stored in the Key Vault>

SQL Server Database: tpcc

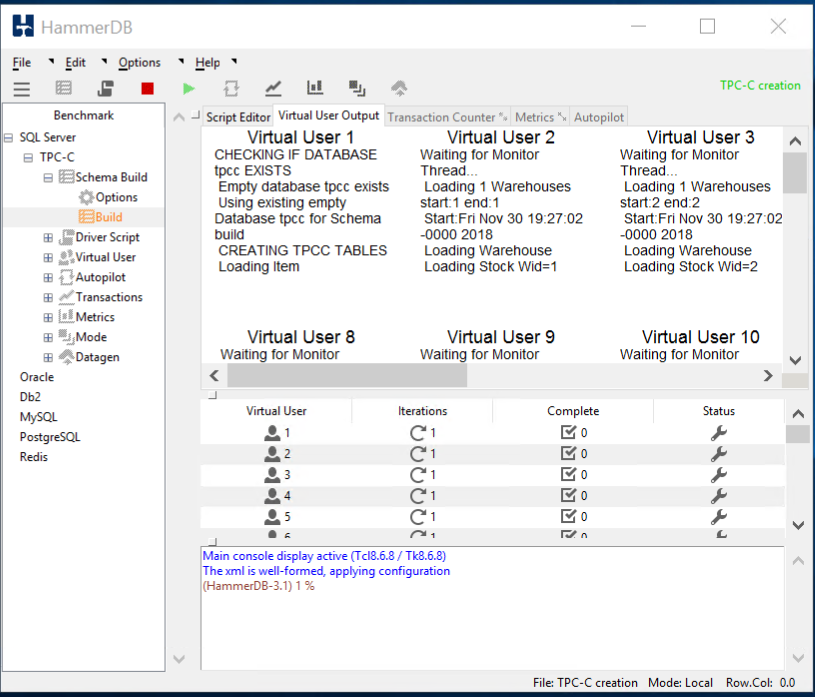
Number of Warehouses: 50

Virtual Users to Build Schema: 50

Note: Setting the last two at 50 should generate enough load to trip a threshold and run long enough for you to graph

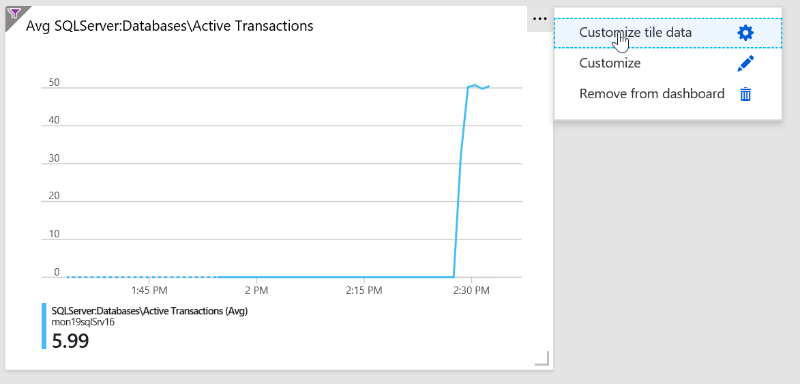


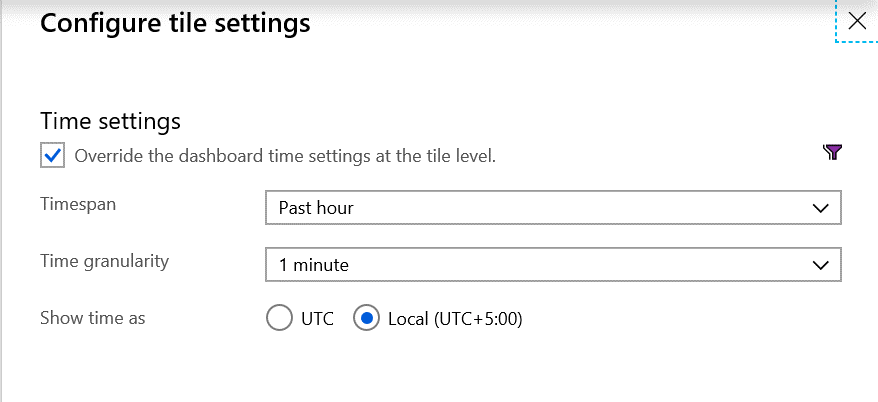
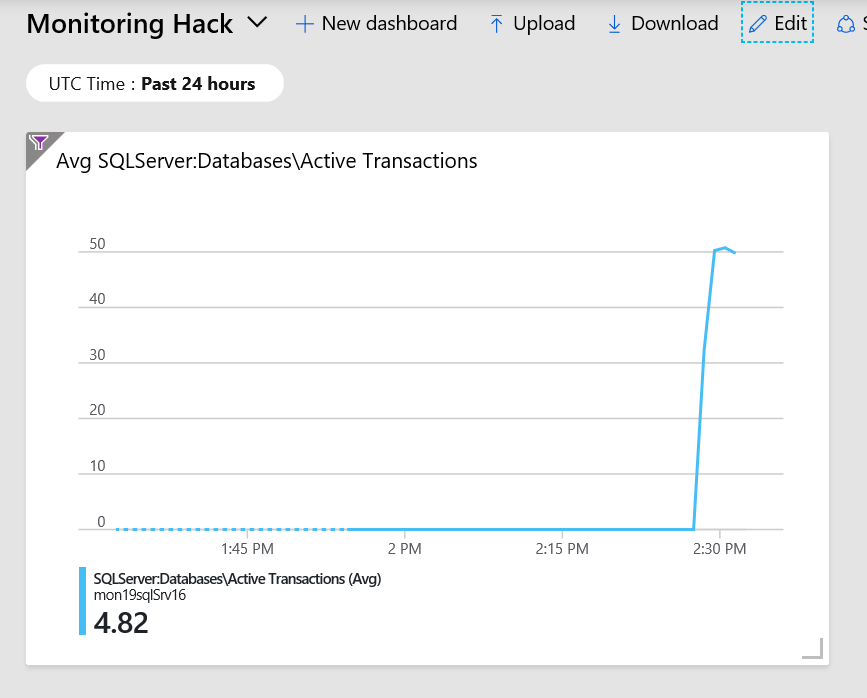
Double click on Build and Click Yes to kick of a load test.



When the test is running it should like this.

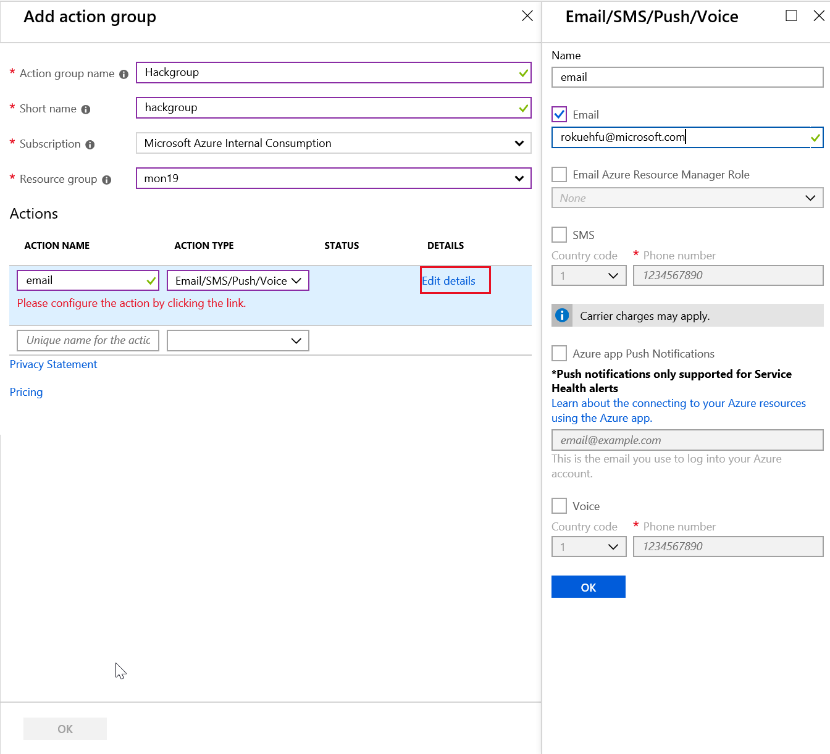
TIP: If you would like to run a second test you **must** first delete the database you created and recreate it. HammerDB will not run a test against a database that has data in it. When you run a test is fills the database with a bunch of sample data.

* From Azure Monitor,
  + Create a graph for the SQL Server Active Transactions and Percent CPU and pin to your Azure Dashboard
  + Note: I had to customize the dashboard once I pinned it to a new Azure Dashboard  
    

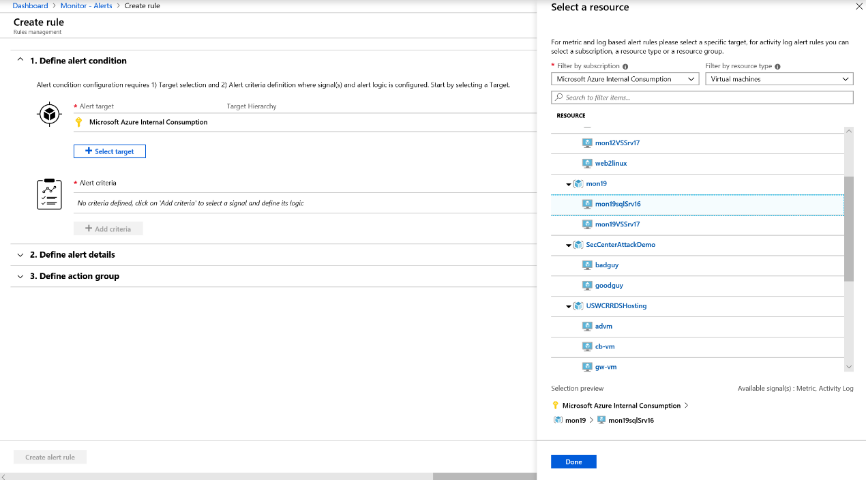
  


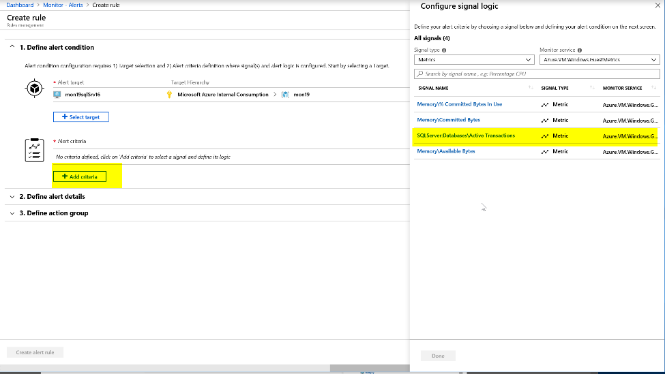
Dashboard should look something like this.

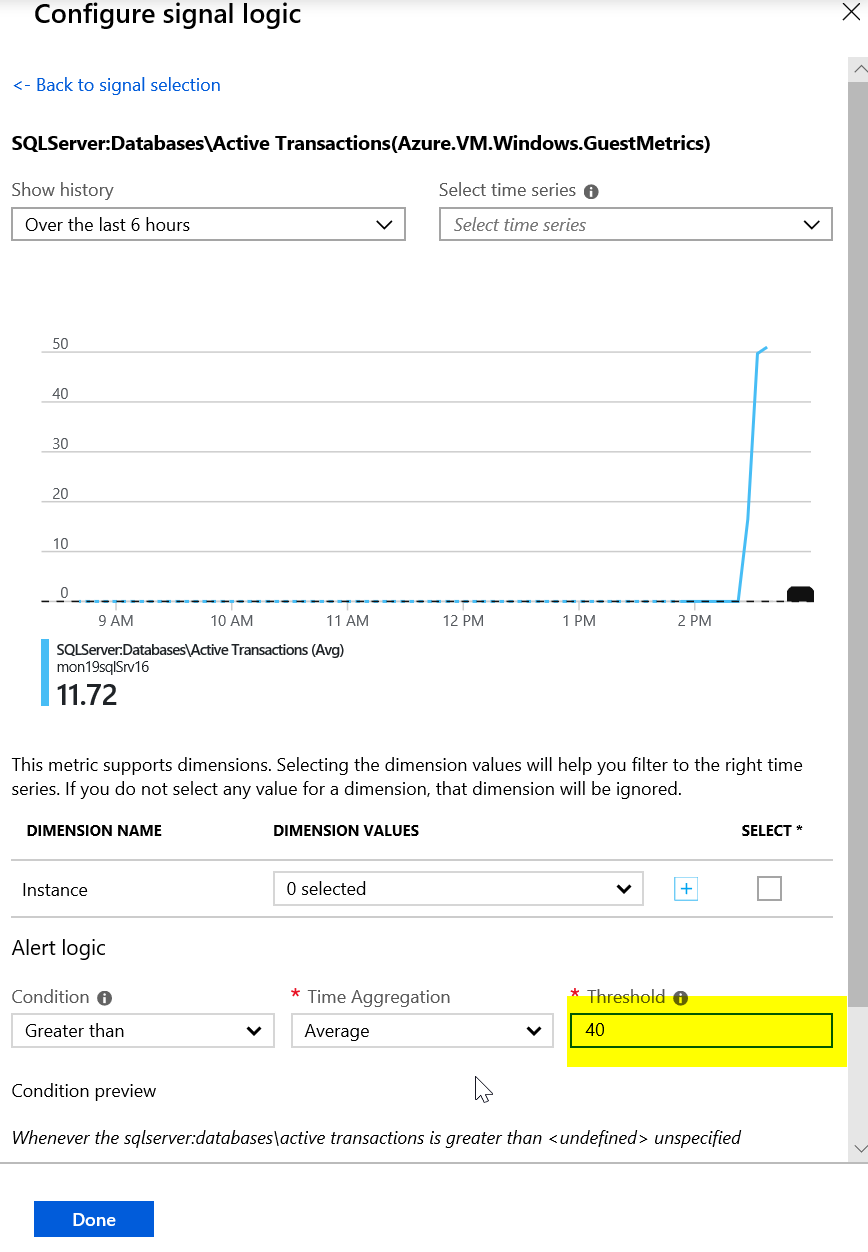
* + From Azure Monitor, create an Action group, to send email to your address

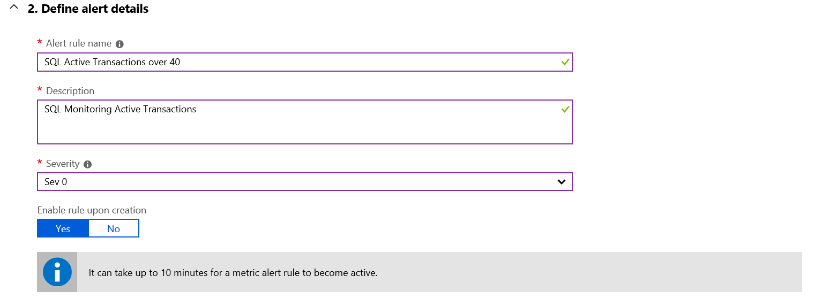


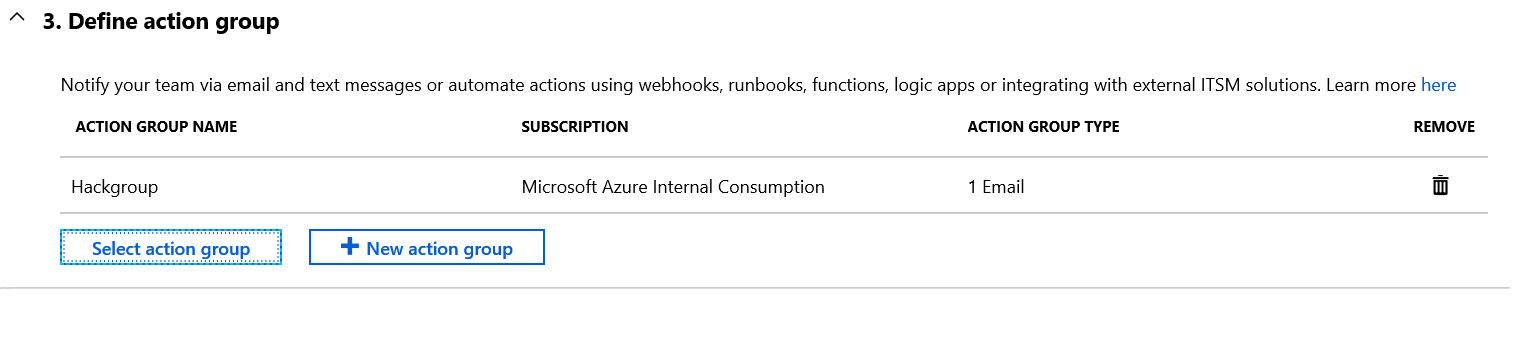
* + Create an Alert if Active Transactions goes over 40 on the SQL Server tpcc database.



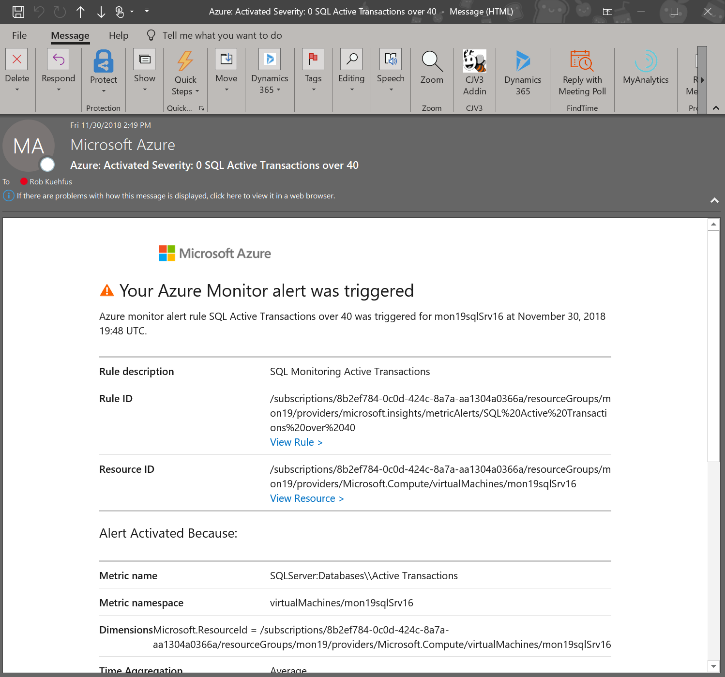






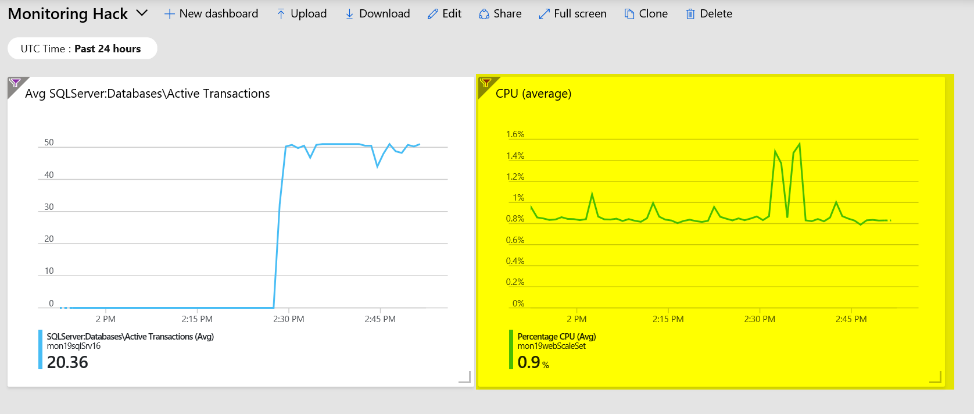






* Create an Alert Rule for CPU over 75% on the Virtual Scale Set that emails me when you go over the threshold.

First create a dashboard to watch the Scale Set CPU



Navigate to the folder called “Loadscripts” under the Resources folder in Student and copy the cpuGenLoadwithPS.ps1 script to both instances running in the Scale Set and run them.

This may be a bit of a challenge to those not used to working with a scale set. If your student just grabs the public IP address and then RDP to it. They will end up on one of the instances but because they are going through the Load Balancer, they cannot control which one. Or can they? 😊

If you look at the configuration of the LB it is configured with an inbound NAT rule that will map starting at port 50000 to each instance in the Scale Set. So if they RDP using the PIP:50000 for instance 1 and PIP:50001 for instance 2.

"inboundNatPools": [

{

"name": "natpool",

"properties": {

"frontendIPConfiguration": {

"id": "[concat(variables('webLbId'), '/frontendIPConfigurations/loadBalancerFrontEnd')]"

},

"protocol": "Tcp",

"frontendPortRangeStart": 50000,

"frontendPortRangeEnd": 50119,

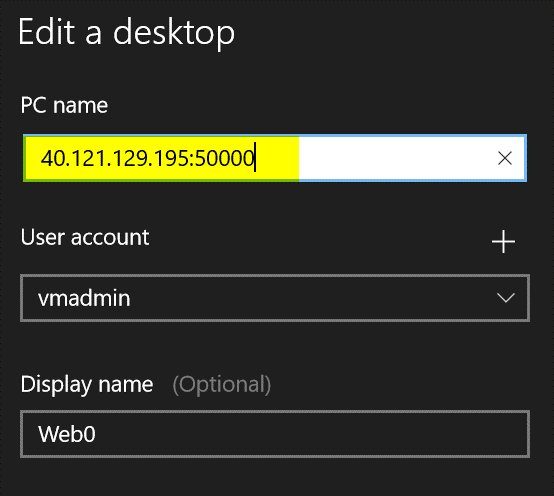
"backendPort": 3389

}

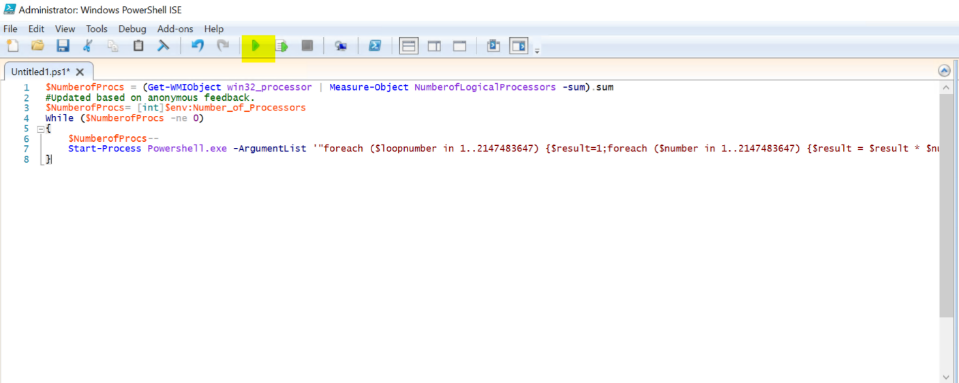
}

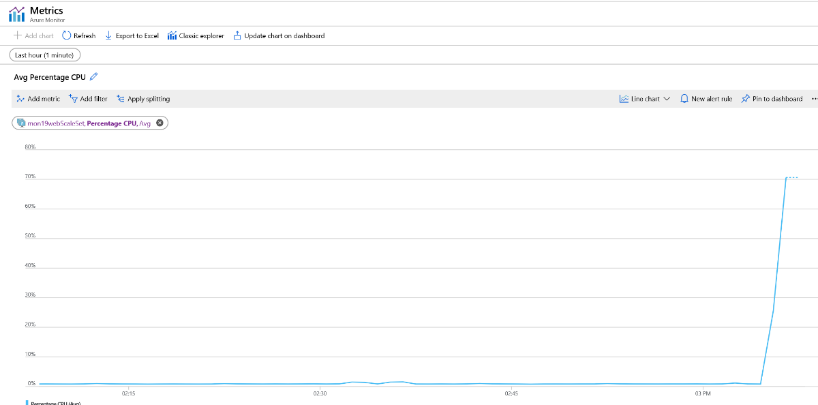
],

For Example,

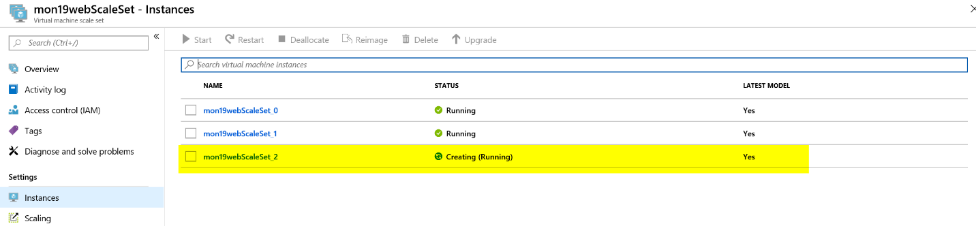


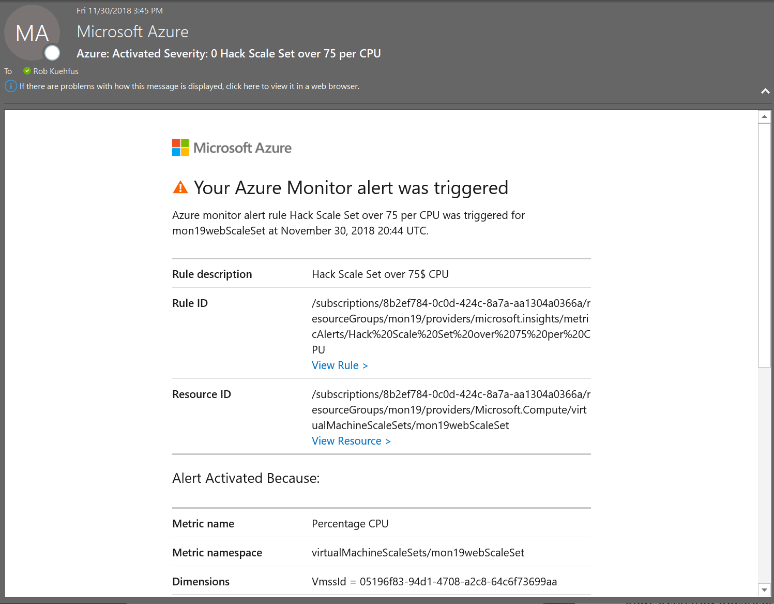
Jump on to both VMs in the Scale Set, Open the PowerShell ISE, Copy the script in the window and run it. You may need to run it more then once to really add the pressure. This script will pin each core on the VM no matter how many you have.





The trick to getting the alert to fire is to pin both instances at the same time as the CPU metric is an aggregate of the scale set. If you just max the CPU on one server to 100% the Scale Set is only at 50% and till not trip the alert threshold of 75%. Also, if they run the script and then setup the Alert Rule then to back to run another test to trip the alert, they have scaled out to a third instance and not realized it. They may need to jump on that box and max it out as well.



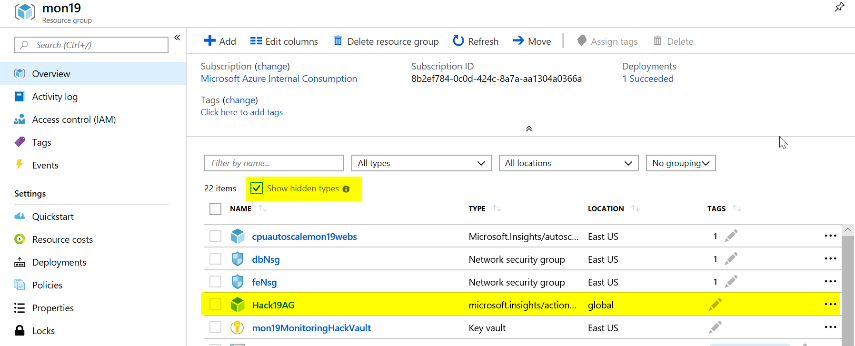


* First team to send me both alerts wins the challenge!!
* Good luck!

# Challenge 2: Monitoring and Alert Rule Automation

* Update the parameters file and deployment script for the GenerateAlertRules.json template located in the AlertTemplates folder
  + Add the names of your VMs and ResouceId for your Action Group

To find the ResourceId for your Action group navigate to the Resource Group where you are stored the action group and make sure to check off “Show hidden types”.



Click on your Action Group and copy the ResourceId



Then update the deployAlertRules.parameters.json file as it shows below.

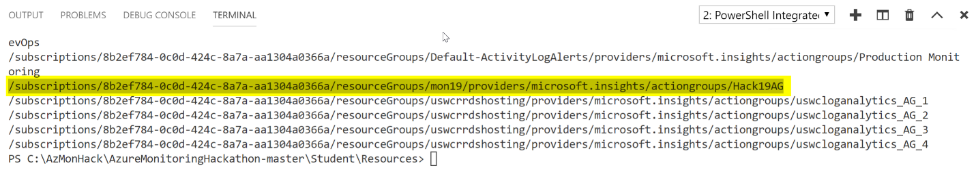
Or

In the deployAlertRulesTemplates.ps1 script update the resourcegroup and run the first few lines then run the code to get the Azure Monitor Action Group

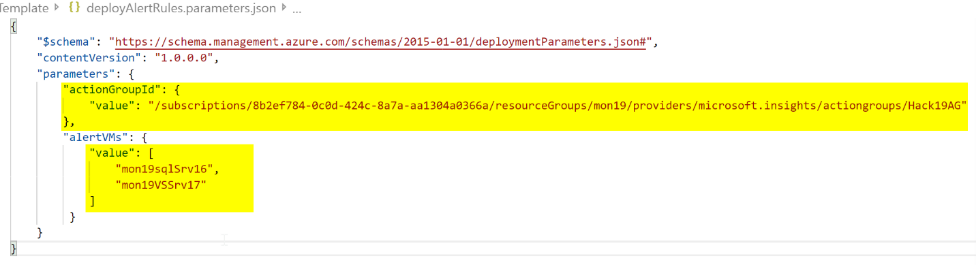
#Get Azure Monitor Action Group

Get-AzureRmResource -ResourceType 'Microsoft.Insights/actiongroups' | Select-Object Name, ResourceId

Copy and paste the resource Id for the Action Group you would like to use.



Save the parameters file and update the deployAlertRulesTemplate.ps1 file with the name of your Resource Group (and save it).



* Deploy the GenerateAlertRules.json template using the sample PowerShell script (deployAlertRulesTemplate.ps1) or create a Bash script (look at the example from the initial deployment)

#Update Path to files as needed

#Update the parameters file with the names of your VMs and the ResourceId of your Action Group (use command above to find ResourceId)

$template=".\AlertsTemplate\GenerateAlertRules.json"

$para=".\AlertsTemplate\deployAlertRules.parameters.json"

$job = 'job.' + ((Get-Date).ToUniversalTime()).tostring("MMddyy.HHmm")

New-AzureRmResourceGroupDeployment `

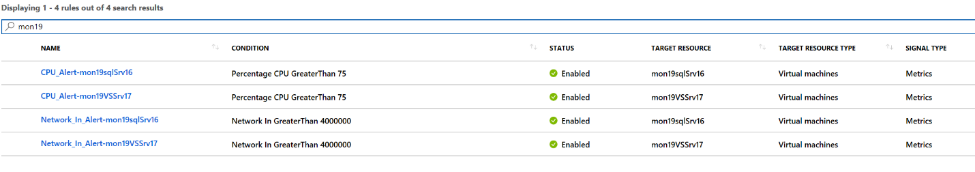
-Name $job `

-ResourceGroupName $rg.ResourceGroupName `

-TemplateFile $template `

-TemplateParameterFile $para

* Verify you have new Monitor Alert Rules in the Portal or from the command line (sample command is in the deployment script)



* Modify the GenerateAlertsRules.json to include “Disk Write Operations/Sec” and set a threshold of 10  
  **Tip:** Go here to view the list of metrics available by resource type - <https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/monitoring-supported-metrics#microsoftcomputevirtualmachines>

Use this link to see the ARM schema - <https://docs.microsoft.com/en-us/rest/api/monitor/metricalerts/update>

{

"type": "Microsoft.Insights/metricAlerts",

"name": "[concat('Disk\_Write\_Alert','-',parameters('alertVMs')[copyIndex()])]",

"copy": {

"name": "iterator",

"count": "[length(parameters('alertVMs'))]"

},

"apiVersion": "2018-03-01",

"location": "global",

"tags": {},

"scale": null,

"properties": {

"description": "Disk Write metric has detected a large amount of disk operations",

"severity": "[parameters('alertSeverity')]",

"enabled": "[parameters('isEnabled')]",

"scopes": [

"[resourceId('Microsoft.Compute/virtualMachines', parameters('alertVMs')[copyIndex()])]"

],

"evaluationFrequency": "PT5M",

"windowSize": "PT5M",

"criteria": {

"odata.type": "Microsoft.Azure.Monitor.SingleResourceMultipleMetricCriteria",

"allOf": [

{

"name": "MetricDiskWriteOper",

"metricName": "Disk Write Operations/Sec",

"dimensions": [],

"operator": "GreaterThan",

"threshold": 10,

"timeAggregation": "Average"

}

]

},

"actions": [

{

"actionGroupId": "[parameters('actionGroupId')]",

"webHookProperties": {}

}

]

},

"dependsOn": []

},

* Rerun your template and verify your new Alert Rules are created for each of your VMs



* First team to me a screenshot of the new Alert Rules wins the challenge!!
* Good luck!

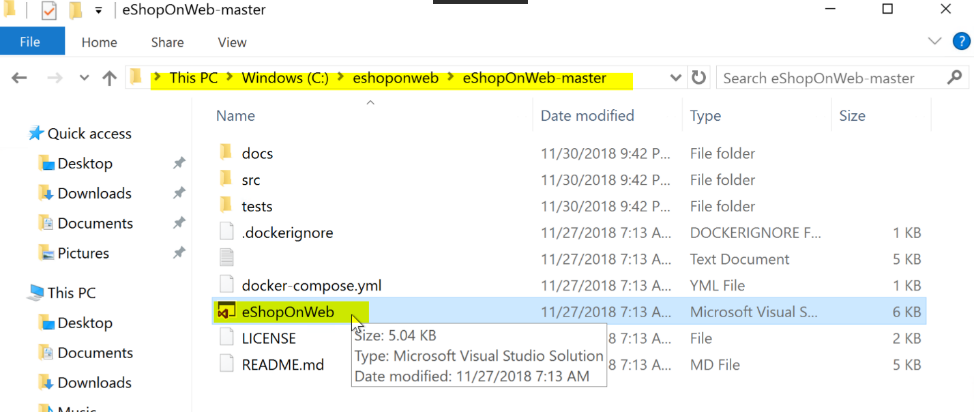
# Challenge 3: Azure Monitor for Containers

* From your Visual Studio Server, deploy the eShoponWeb application to AKS using Dev Spaces
  + Hint: <https://docs.microsoft.com/en-us/azure/dev-spaces/get-started-netcore-visualstudio>
* From Azure Monitor, locate the container running the eShoponWeb application
* Generate an exception in the eShoponWeb application  
  (Hint: Try to change your password)
* First person to send me a screen shop of the live log with the exception message wins the challenge

# Challenge 4: Application Insights

In Visual Studio, Install the Application Insights SDK in the eShopOnWeb Web Project in the Solution

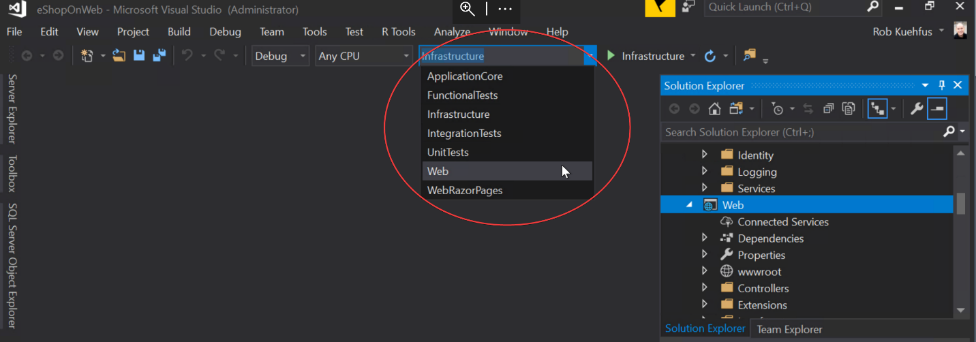
From the Visual Studio Server, navigate to C:\eshoponweb\eShopOnWeb-master and double-click on eShopOnWeb.sln



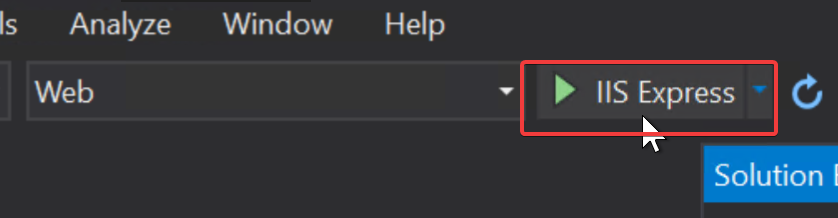


Select Visual Studio 2017 and click OK.

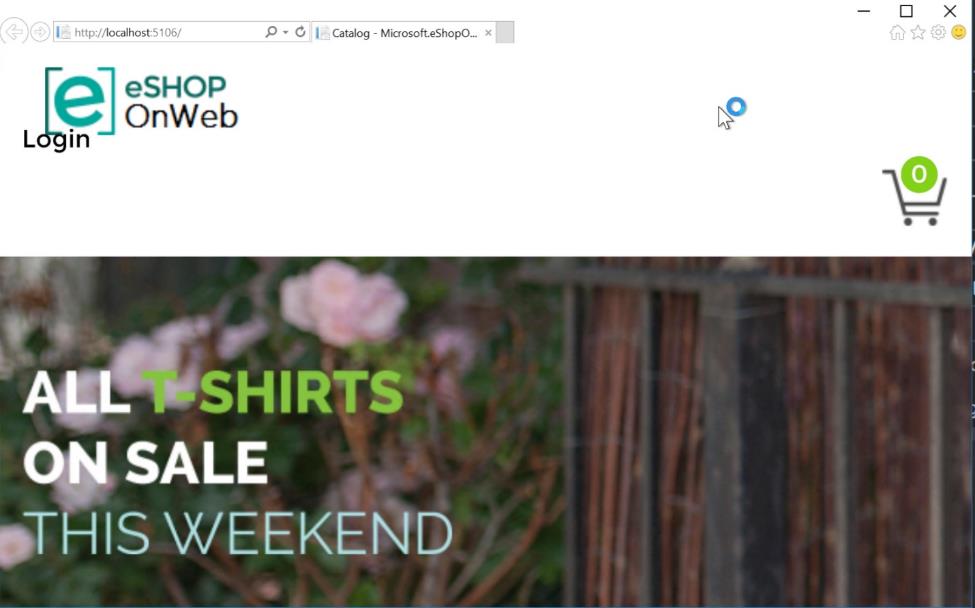
If this is the first time you are opening Visual Studio please log in or create an account and log in.



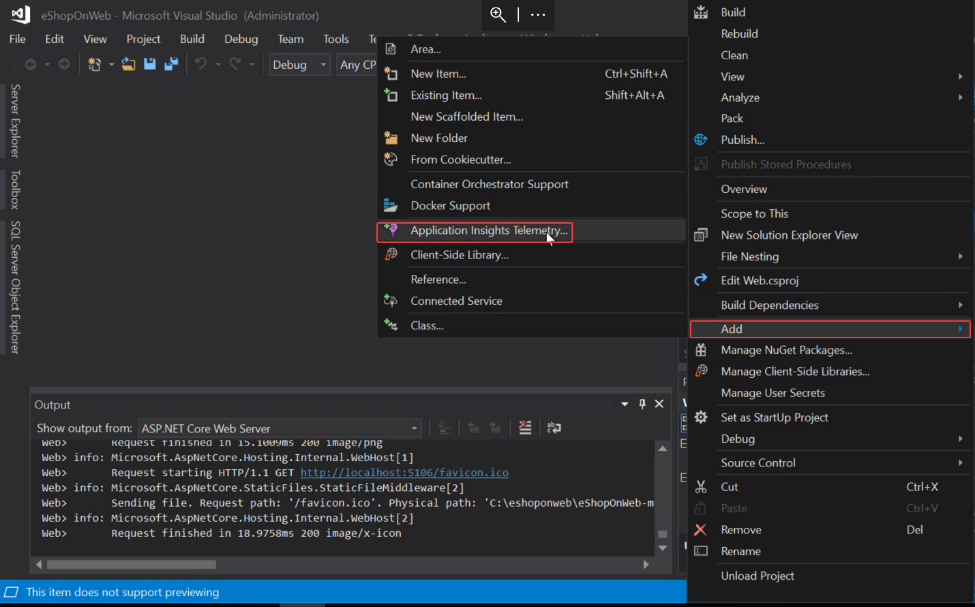
Select Web



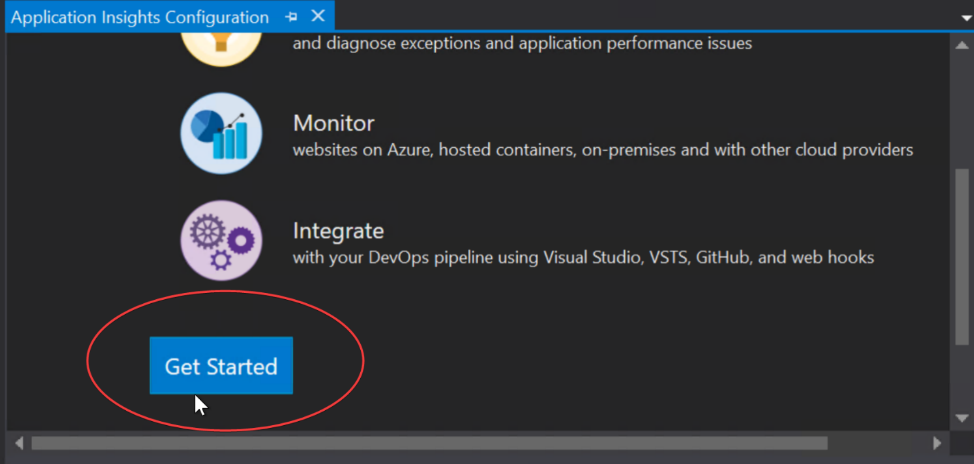
Click on IIS Express to test the eShop application



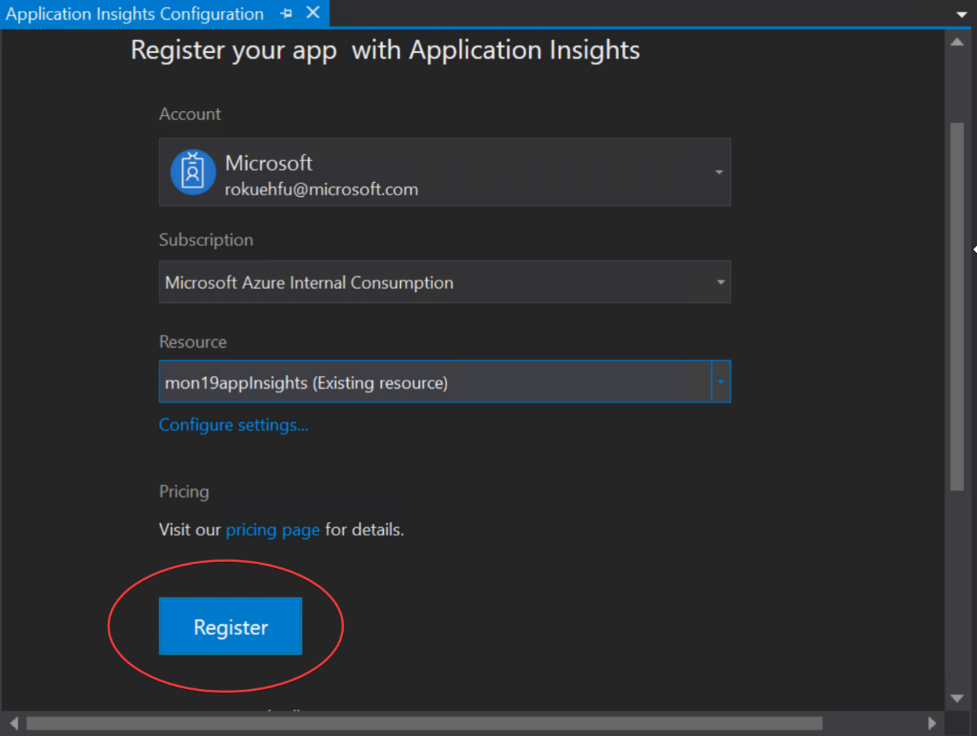
You should see the eShop app open locally. Close it and let’s add the Application Insights SDK



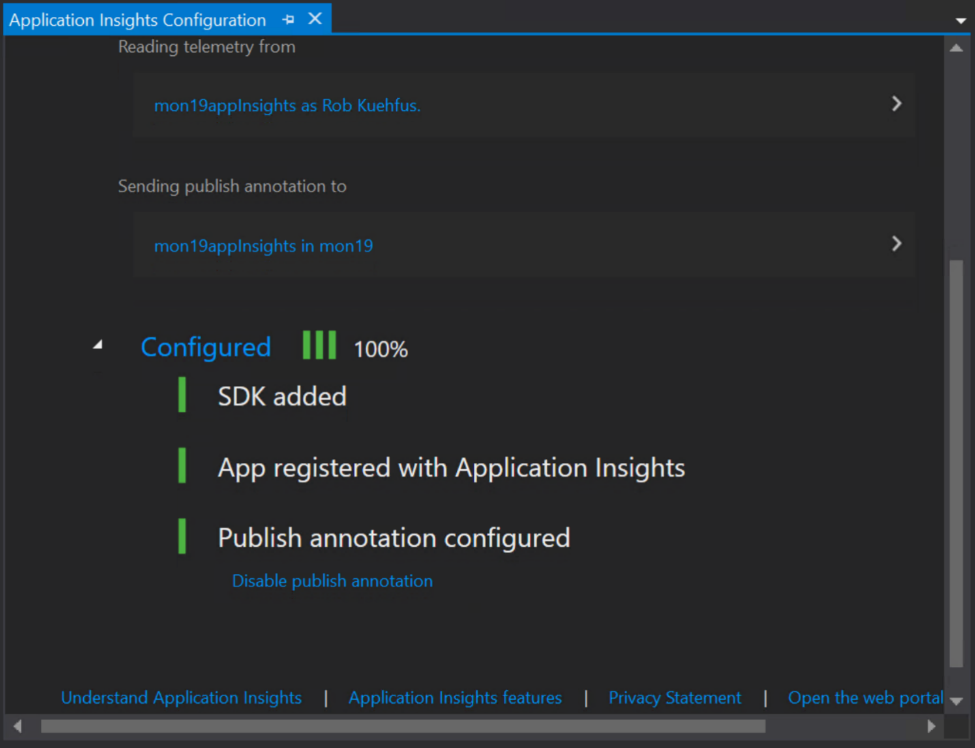
On the right hand side, find Web and right click, go to Add and select Application Insights Telemetry



Click get Started

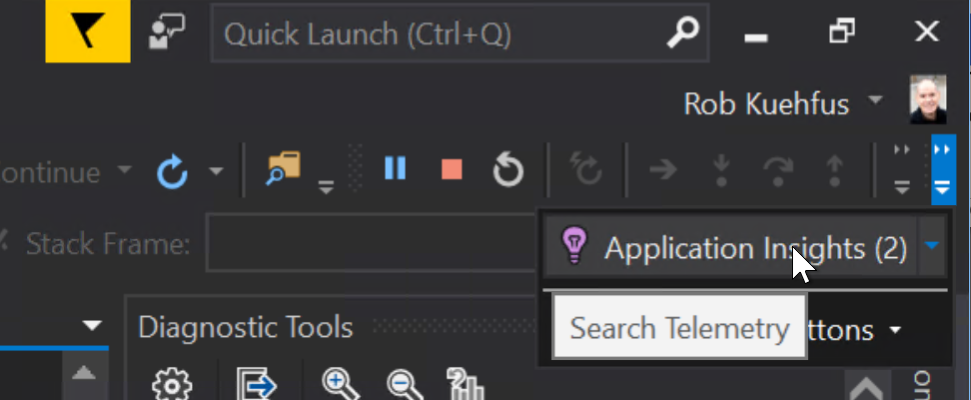


Select your subscription, Resource (name of your App Insights) and click Register.

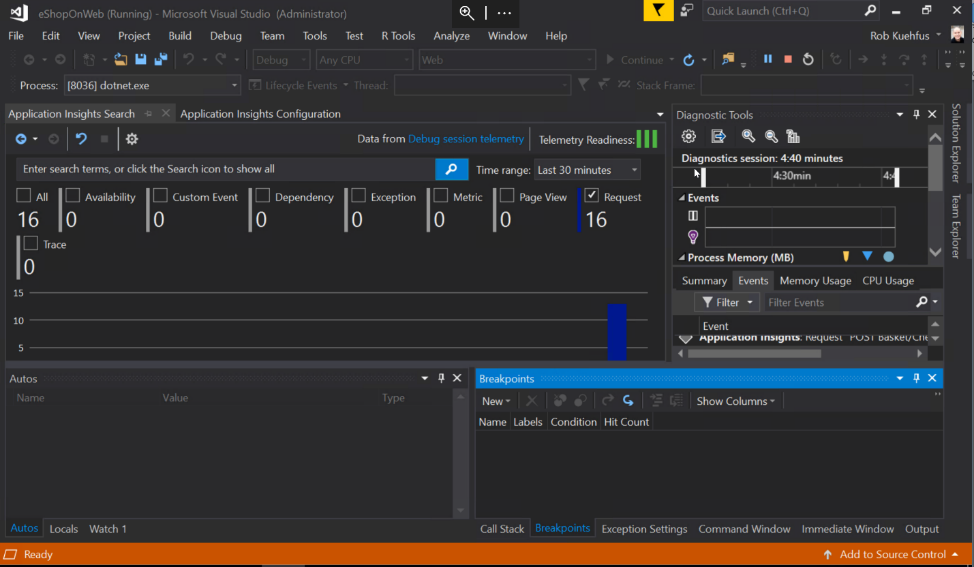


* + Run the eShopOnWeb Web project and check out the App Insights tooling

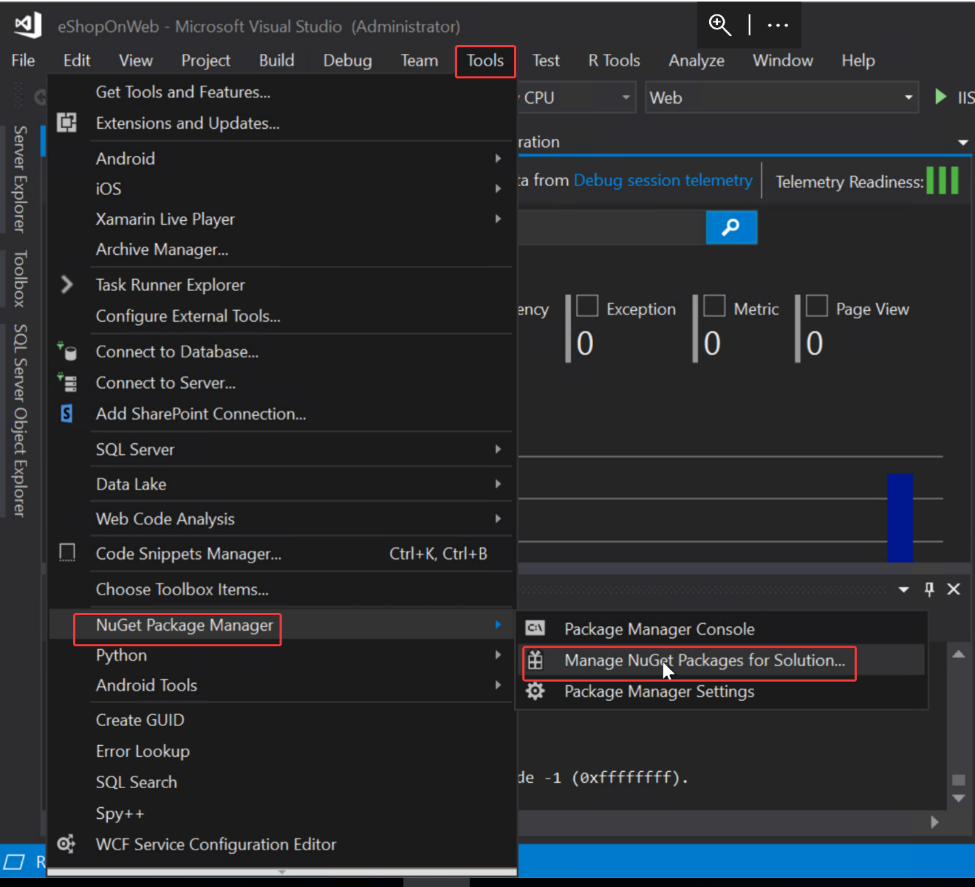
Test the application by running it and verify its working.



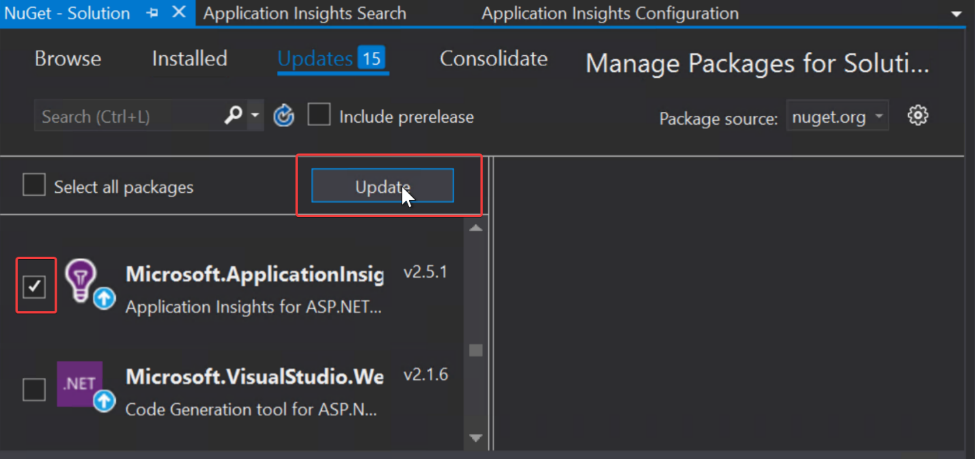
While its running you can navigate to Application Insights and view the telemetry while you are interacting with eShop running on the local machine. Add something to the shopping cart, log in and check out.



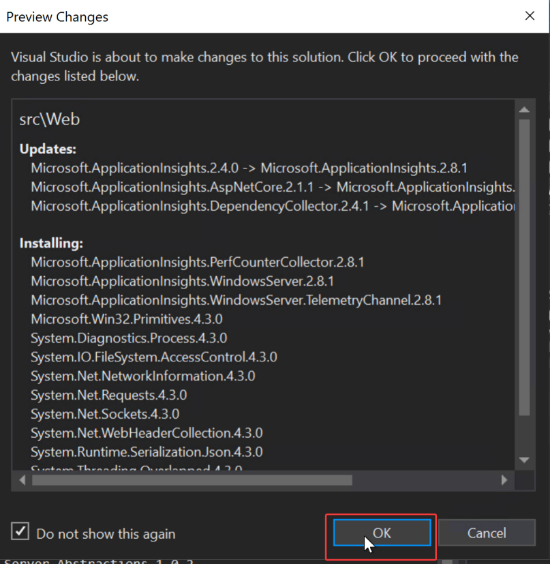
* + Add the updated Application Insights NuGet package to v2.5.1



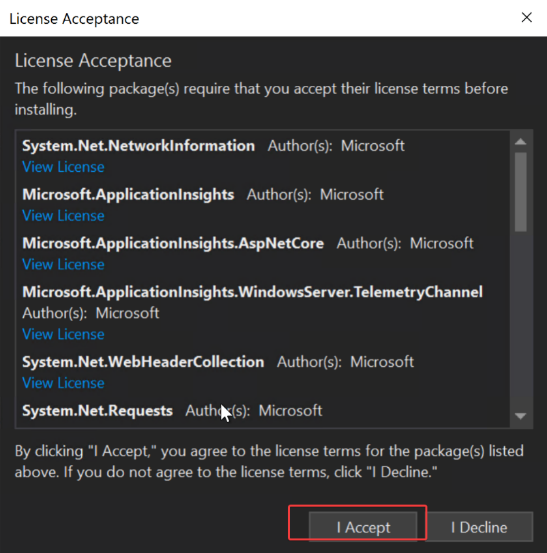
Go to Tools, NuGet Package Manager, Manage NuGet Packages for Solution



Check off the Microsoft.ApplicationInsights package (v2.5.1) and click Update



Click OK



Click I Accept. When finished run the eShop application again to make sure it’s working.

* + Publish eShopOnWeb Web project to AKS

<need to add steps>

* + Generate some load and check out the results

From your laptop or the Visual Studio Server copy the code in the LoadScripts folder and modify it to your URL

for ($i = 0 ; $i -lt 100; $i++)

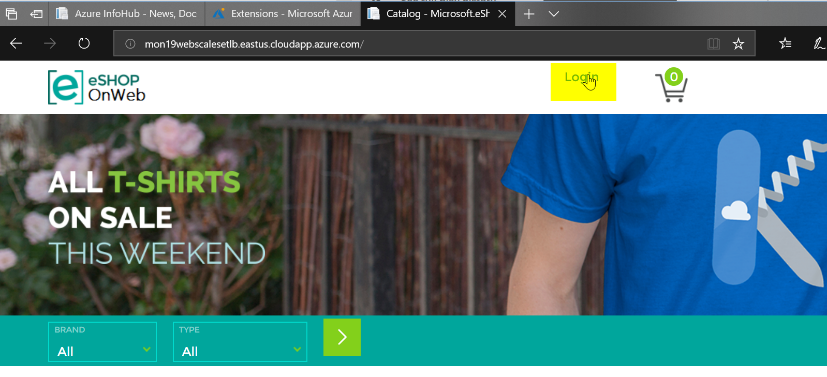
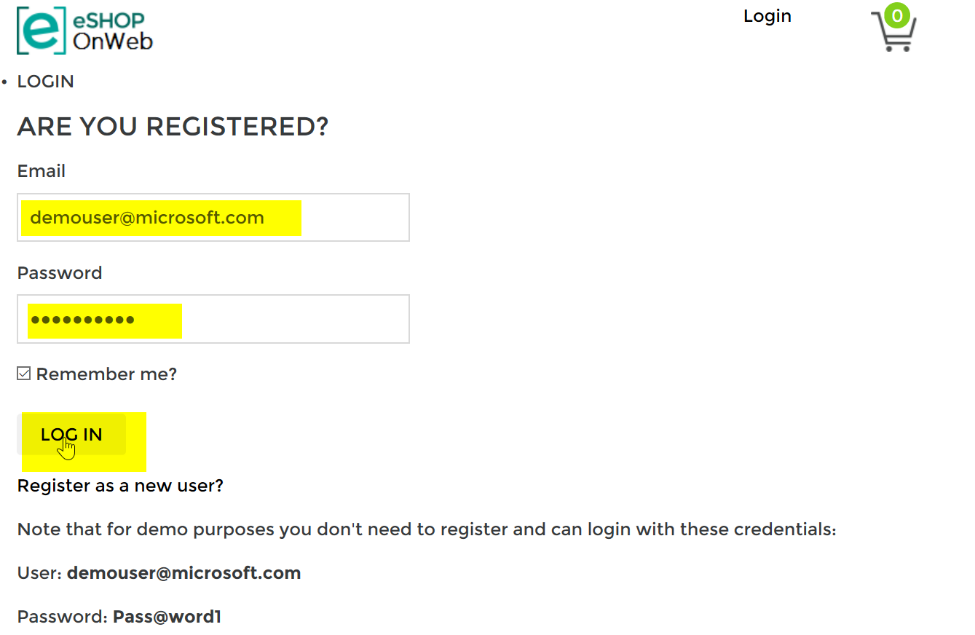
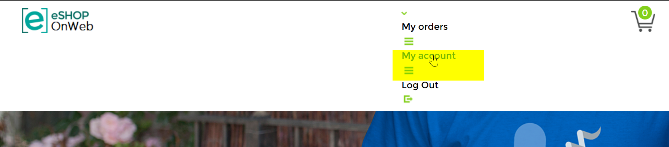
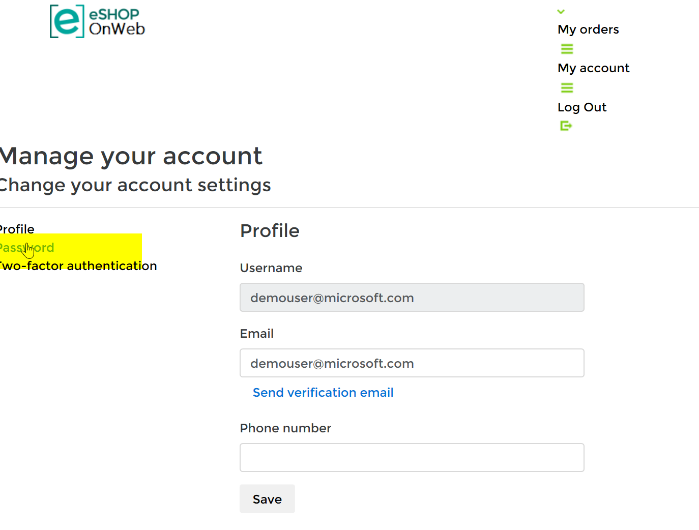
{

Invoke-WebRequest -uri http:// mon19webscalesetlb.eastus.cloudapp.azure.com/

}

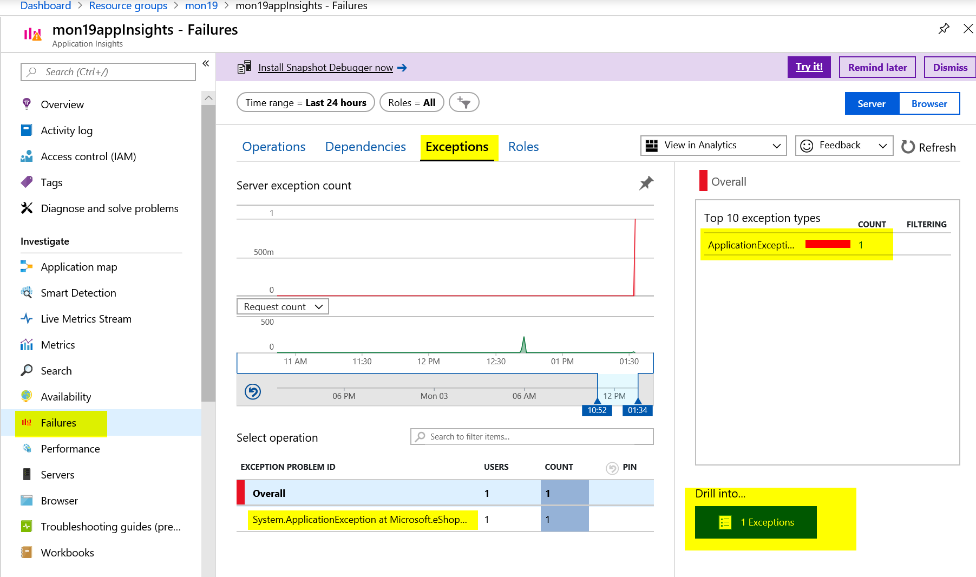
Run the code to generate some load on your eShopOnWeb site

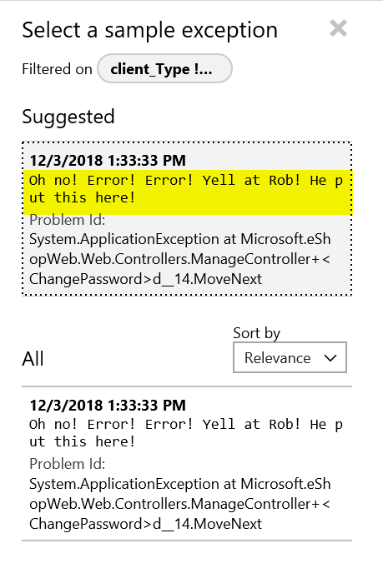
To trip the exception,

1. Open your eShop site in your browser and login to the site  
     
   
2. Try to change your password  
     
   

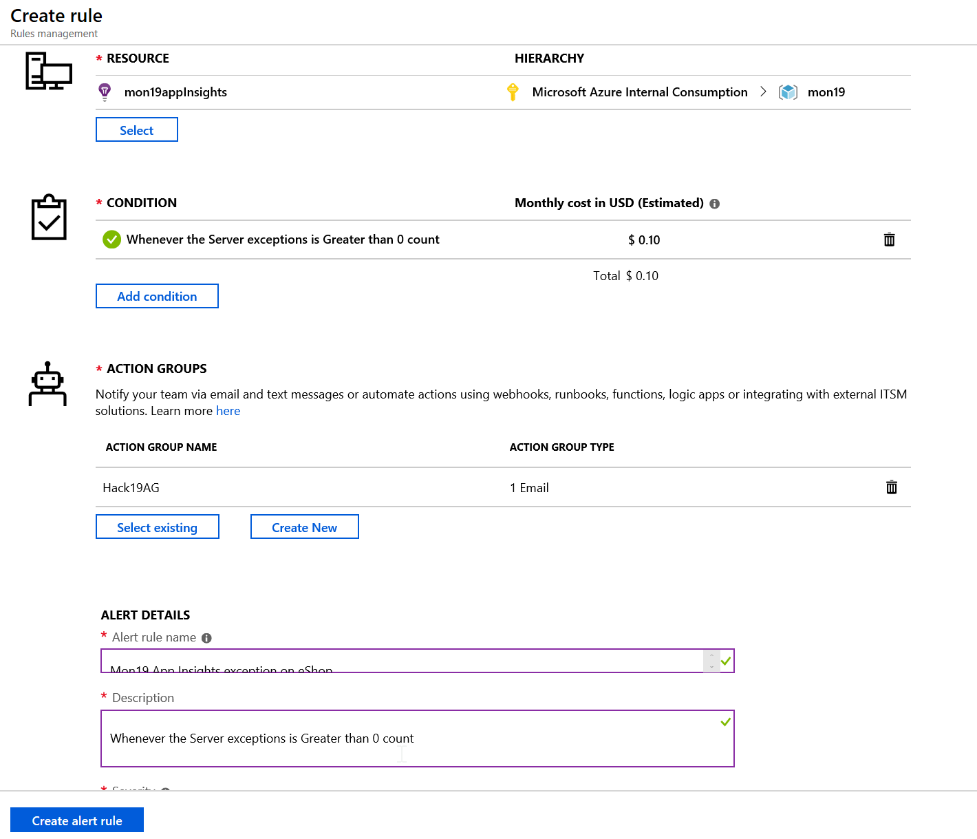


* + Find the exception in App Insights





* Create Alerts based on Availability and exceptions



* First Team to email me an alert of the exception and a screenshot with your scaleset scale out based on the App Insights metric wins the challenge. Good luck

# Challenge 5: Dashboard and Insights

* Deploy Grafana using Web App for Container
  + Hint: <http://docs.grafana.org/installation/docker/>
* Configure the Azure Monitor Data Source for Azure Monitor, Log Analytics and Application Insights
* Create a CPU Chart with a Grafana variable used to select Computer Name
* Add an Annotation to your chart overlaying Computer Heartbeat
* First Team to email me a screenshot with your chart wins the challenge. Good luck!