Microsoft

Network Bandwidth Measurement & Monitoring Solution using Azure Monitoring (Service Map)

Intended for planning or monitoring SaaS/PaaS network connectivity.

Published: December 2019
Version: 3.4

This document provides the steps to implement a solution which allows customers to remotely capture information from network connections so as to provide insights into connectivity and also monitor bandwidth utilization of machines or individual processes/services

Table of Contents

1 Introduction

As the world moves to the cloud, a critical piece of this transformation is around the network which connects users and businesses to their data and services. The path changes from a relatively simple line between the location of the users and the data/services, to a distributed model where data is in multiple places, and users are accessing it from a variety of locations and devices.

Planning physical connectivity for Microsoft cloud users should revolve around local egress as close to the user as possible so that the traffic can reach Microsoft's global network as quickly as possible. More detail on this specific to Office 365 can be found here.

Another major part of this transformation is planning how much bandwidth is required on the circuits which connect out of the enterprise. As user's data and services move from being hosted on the internal network, to being hosted in Microsoft datacenters, the path this data takes changes. in addition, the amount of data users send and receive changes as their available services increases or changes.

However, planning the amount of bandwidth can be a major challenge as every user and enterprise is different so it's impossible to give an average figure, what will be correct for one enterprise will be insufficient for another customer and far too much for another.

The traditional approach to solving this problem is a combination of monitoring pilot users and using process specific calculators to give an estimate based on assumed information or that obtained from pilot users. There are however multiple challenges with this model

a. It's not always easy to obtain accurate data from pilot users
b. Calculators provide an estimated answer, if incorrect information is entered, the result is therefore wrong. As the investment in bandwidth is normally a lengthy process which has an associated cost, it is imperative this information is correct so that the right long-term planning decisions can be taken.
c. Calculators focus on a single service where the planning needs to be done at a higher level covering all services.

With these challenges in mind we have created a more scalable, simpler solution to allow the accurate collection and display of the data required to make informed decisions in this space, and to enable easier monitoring of connectivity of our clients as we move to the cloud.

This solution will allow you to monitor and analyse the following example scenarios:

- Bandwidth used for a particular process or set of processes over a set period of time
- Bandwidth used by the machine over a set period of time
- Bandwidth used in connections to a specific port
- Bandwidth used to a specific IP address or range of addresses
- IP geolocation of the endpoints connected to

The approach in this document can be used for:

1. Measuring network bandwidth usage for pilot users on-boarded to Office 365 or network bandwidth usage of on-premises users.
2. Endpoint monitoring dashboard post on-boarding users to Office 365

You can apply this concept for measuring any SaaS/PaaS traffic, not just Office 365.

2 Prerequisites

The solution in this document requires a number of components bought together to provide the desired outcome, the good news is they are all easy to implement and should take 30-60 minutes in total to set up. For reference, the requirements are:

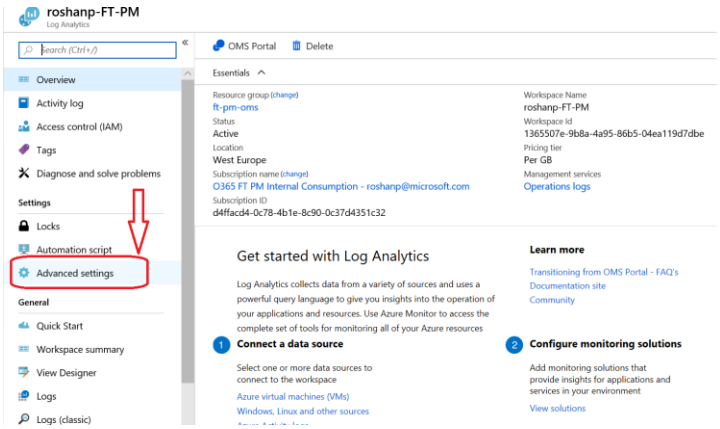
1. **An Azure Subscription**

   As the solution has Azure at its core, a valid Azure subscription with permissions to add components to that subscription (or the ability to ask an administrator to add them) is required. You can use your work or personal account to login to Azure, if this is for your organization then its recommended you use your work account and a subscription associate with your work account.

2. **Azure Log analytics workspace**

   You could opt to use Log Analytics as an individual service and in this case, billing is based on the data ingested into the workspace (Per GB pricing). This features cloud-friendly, consumption-based pricing that includes a Free tier for testing. You only pay for what you use, here is the pricing.

3. **Agent installs**
   a. Microsoft monitoring agent can be installed from the Azure portal itself by going to Log Analytics workspace and then the Advanced settings blade.



b. You will also need Dependency agent for Windows or Linux (see section 4.4 Client-Side configuration)
c. **UDP is now supported to measure network bandwidth usage for Teams**, just ensure you have the latest version of the Dependency agent installed.

4. Scaling this solution & connecting via OMS Gateway

Enterprise customers have scaled the agent deployment to thousands of workstations/servers by using SCCM. Enterprise customers also tend to use OMS Gateway as a gateway to ingest data to log analytics workspace if direct connection to the workspace from workstations/server is not an option.

5. Resources at Github

All resources referred in this document like scripts, templates are available at Github, there is also an INTRO pdf document for a high-level summary of this solution with visuals.

3 Anticipated Usage Scenarios

The usage scenarios for this solution depends on the network setup in place, or that which is possible for use.

Below are some anticipated scenarios:

A. Client has direct network connectivity (i.e. no proxy)

This is the simplest of scenarios, where the client/s being monitored connects directly connect to internet resources, i.e. they do a DNS lookup and directly connect to the public IP address in question. in this case we simply need to install the agents on the client and point them at our log analytics workspace. This method also works the same way when a transparent proxy is in place.

The Microsoft monitoring agent and Dependency agent is installed on the clients directly.

B. Client uses a proxy where the administrator can install the Agents on the proxy OS

If the clients to be monitored connect to the internet via a managed Linux proxy where the agents can be installed, for example a Squid proxy running on CentOS then we can simply monitor the clients by analysing the traffic leaving the proxy. Supported Linux versions can be found here.

It may be feasible for an organization to set up a temporary proxy of this type in Azure or on Premises simply for the monitoring of Bandwidth to specific endpoints, leaving standard browsing to traverse the standard web proxy.

C. Client uses a proxy where the administrator cannot install the agents.

In this scenario, where the user's only connection method to the internet is through a proxy solution where the agents cannot be installed to the host OS, then we need to get a little creative. One method is to open up secondary/tertiary ports on the existing proxy and edit the PAC file to send specific traffic we wish to monitor to that port.

For example:

Port 8080 – Default Web traffic port

Port 8081 – Exchange Online URLs

Port 8082 – SharePoint/OneDrive URLs

We can then monitor the bandwidth of traffic sent to the specific ports in Log analytics to give us an accurate view of traffic to the particular service. This usage scenario requires the agents to be installed on the clients we wish to monitor, as per scenario A.

D. Monitoring connectivity based on process vs destination IP's

One option which may be used is to monitor connectivity which originates from a single process. The above scenarios are necessary as some cloud services will have data accessed from a multitude of processes, it is therefore necessary to look at the destination address to capture the information for the service in question, regardless of the source process.

When an explicit proxy is used, the destination IP will always be the proxy IP for web bound traffic, thus we need to either capture on the proxy itself where we can see the destination IP to differentiate traffic, or we need to send known endpoints to a unique destination, such as a specific proxy port.

An example would be OneDrive for Business where access to data could come from the OneDrive Sync client, or any Office application, or the browser. We therefore need to filter based on the published SharePoint/OneDrive IP addresses to capture access from all sources.

However, some services will only be used by a single process in most cases, such as Outlook client being used for accessing Exchange. Technically OWA could be used in the browser but it would be known in advance if this is likely.

Therefore, for scenarios like this, the egress model does not matter as filtering can be performed against the Outlook.exe process and not the destination address which will differ based on the network egress model. This can also be a useful method for examples such as monitoring bandwidth usage to Exchange when on premises before any cloud deployment to give a view of current usage by the process, which also may aid in planning.

4 Step by Step Guide

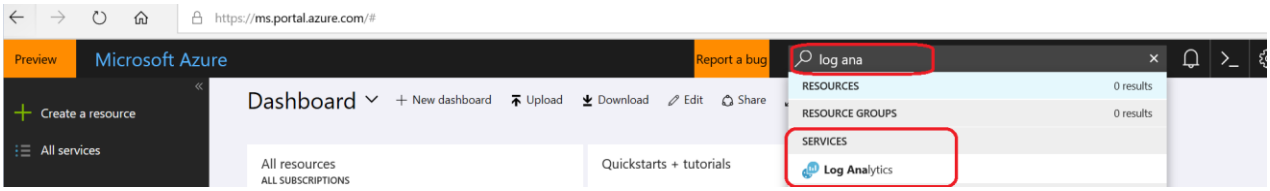
4.1 Login to Azure portal

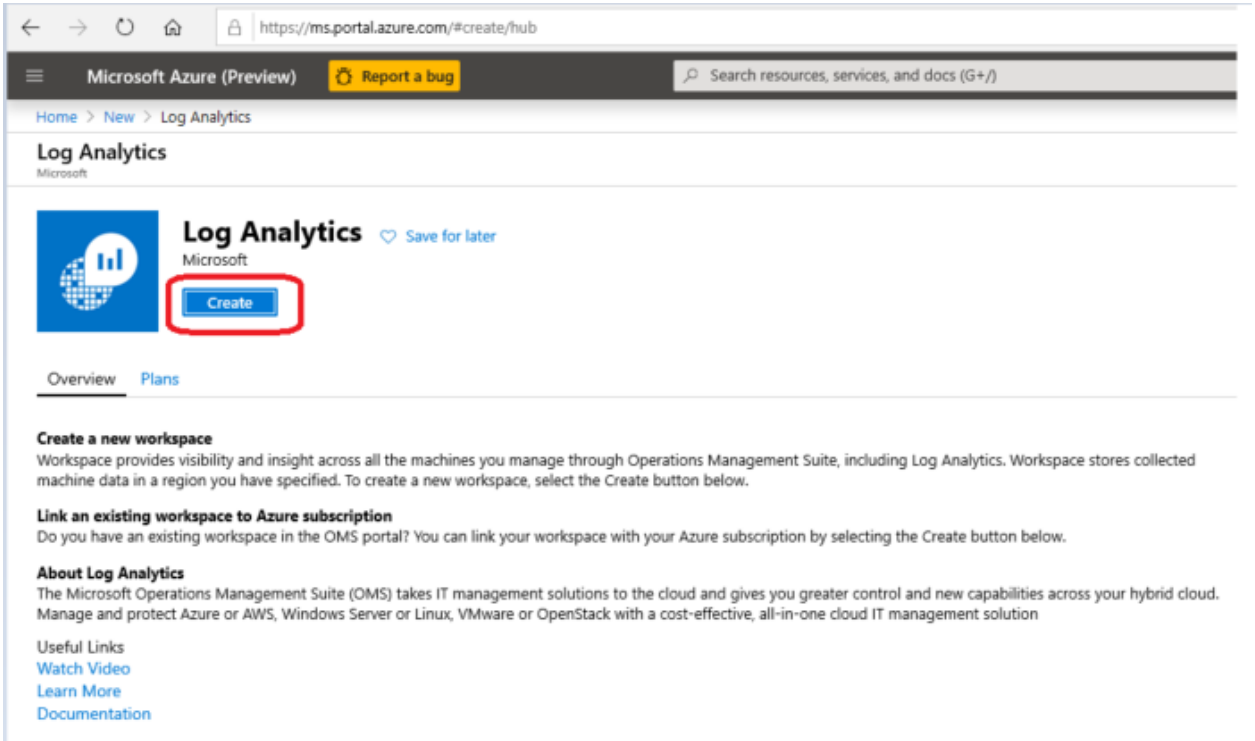
The first step is to login to the Azure portal for the account where you wish to host this solution. If you already have a tenant, simply login at https://portal.azure.com

If you don't already have one, or would like to test this solution in a new Azure instance, you can set up a free trial here.

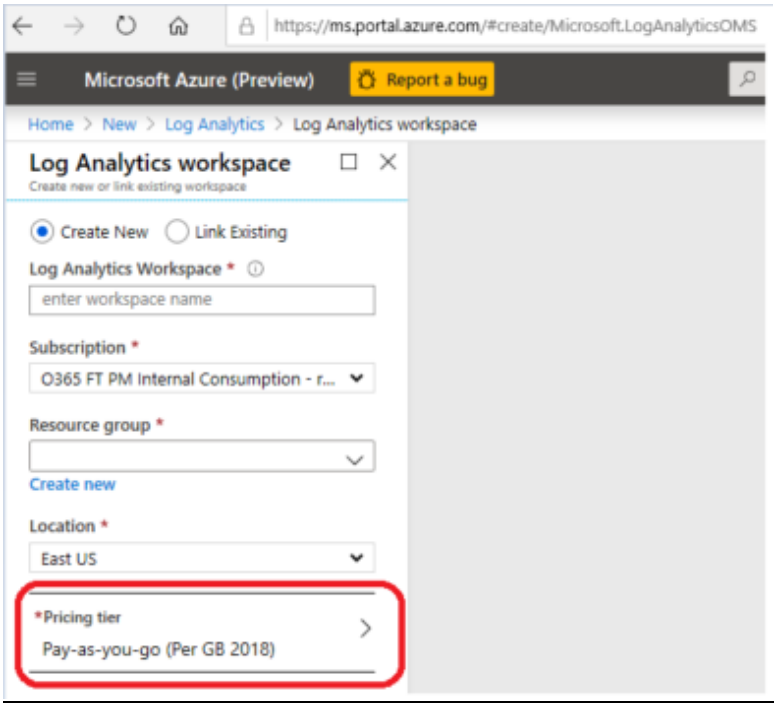
4.2 Creating the Log Analytics workspace

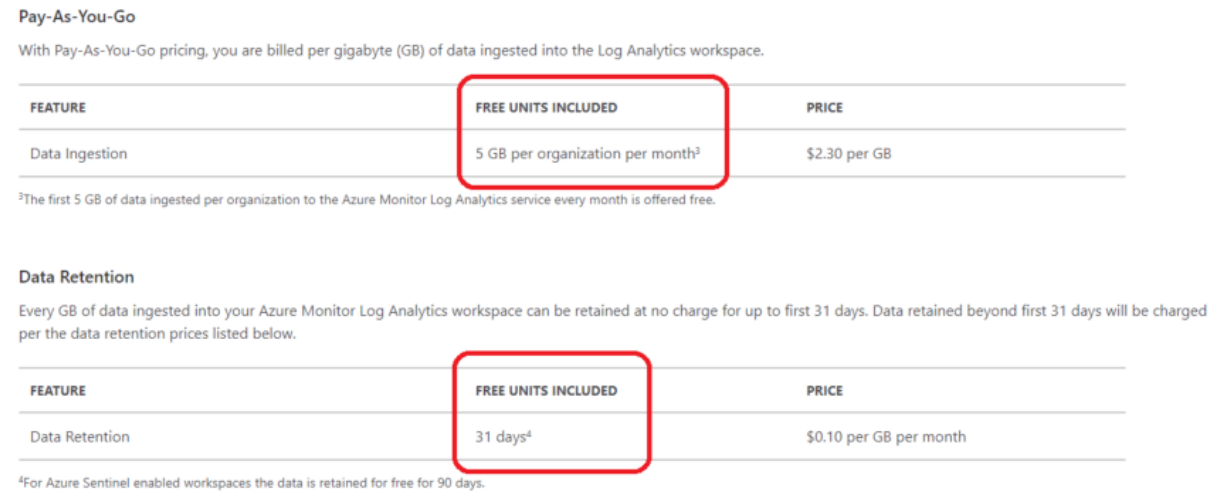
a. Once logged into the Azure Portal https://portal.azure.com/
b. Start typing 'Log Analytics' in the search bar at the top, select 'Log Analytics' under Services
c. Add a workspace if you don't have one already created
    - Give the workspace an appropriate name
    - Subscription is the one you wish to use for the workspace (as you may have more than one available)
    - Resource group is simply a collection of resources, you can either use an existing one, or create a new one.
    - Location is which Azure region you host the workspace in.





Notice the options for pricing tier, you can select Pay-as-you-go or another option depending your subscription, typically Pay-as-you-go is preferred since you get first 5GB data ingestion free along with 31days for data retention.





Pay-As-You-Go

With Pay-As-You-Go pricing, you are billed per gigabyte (GB) of data ingested into the Log Analytics workspace.

FEATURE	FREE UNITS INCLUDED	PRICE
Data Ingestion	5 GB per organization per month[3]	$2.30 per GB

[3]The first 5 GB of data ingested per organization to the Azure Monitor Log Analytics service every month is offered free.

Data Retention

Every GB of data ingested into your Azure Monitor Log Analytics workspace can be retained at no charge for up to first 31 days. Data retained beyond first 31 days will be charged per the data retention prices listed below.

FEATURE	FREE UNITS INCLUDED	PRICE
Data Retention	31 days[4]	$0.10 per GB per month

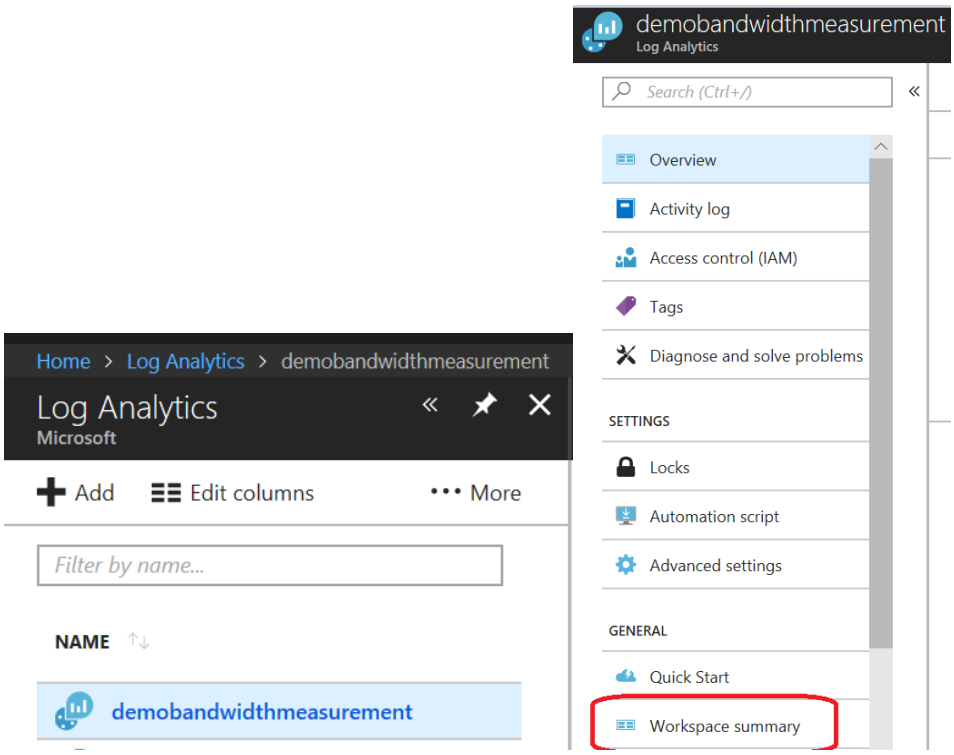
[4]For Azure Sentinel enabled workspaces the data is retained for free for 90 days.

If you need more information regarding pricing for Azure Monitoring/Log analytics workspace please refer here or use the Azure Pricing calculator.

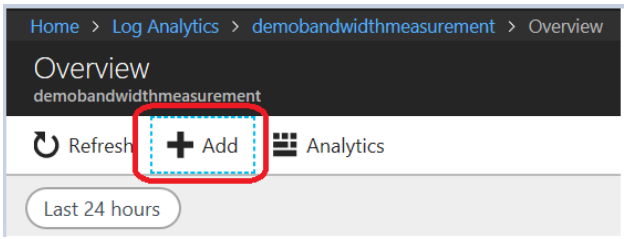
4.3 Configure Log Analytics workspace for Service Map

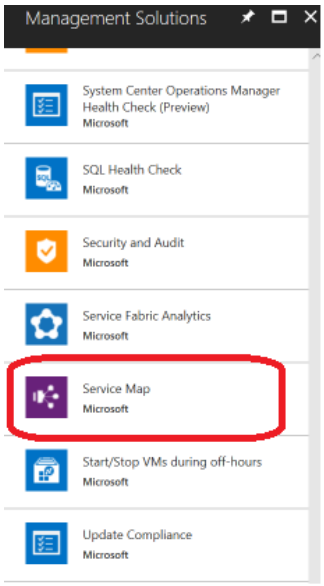
We will now configure Service Map in the workspace.

a. Go to the workspace you created in Log Analytics and click on 'workspace summary' on the left blade.

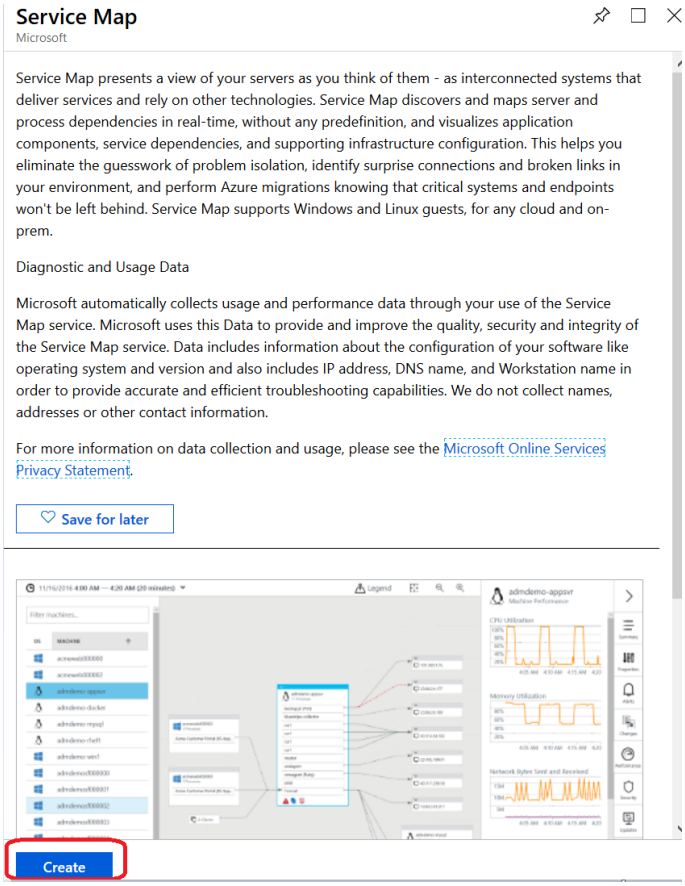


b. The workspace will be blank, click on 'Add' and select Service Map from the 'Management Solutions'.

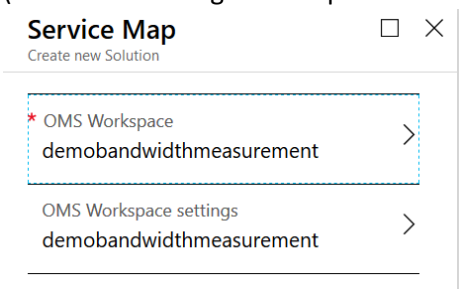




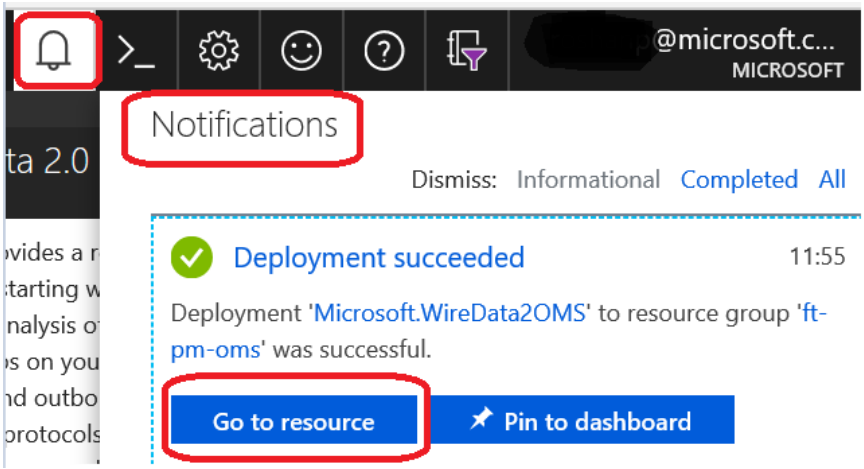
c. Select Service Map and click ‘Create’, ensure you have the right workspace selected



(ensure the the right workspace is selected)

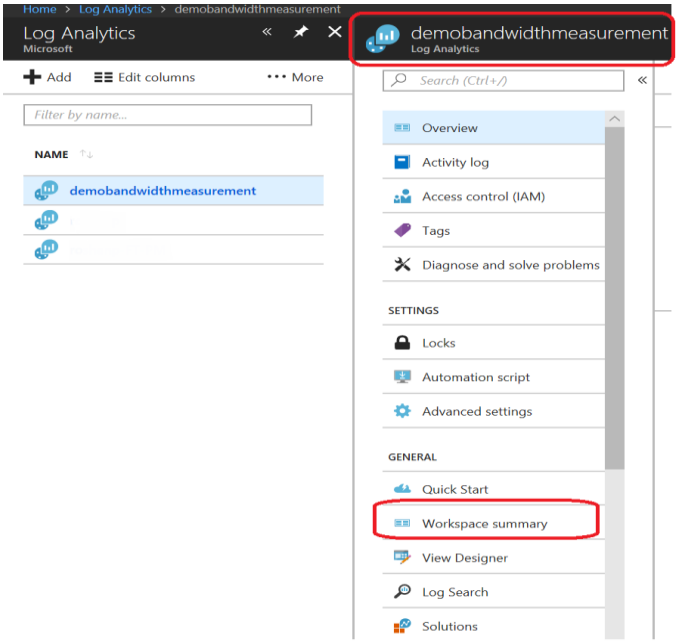


Once the deployment succeeds, go to 'Service Map' by clicking on 'Go to resource' from the Notifications pane

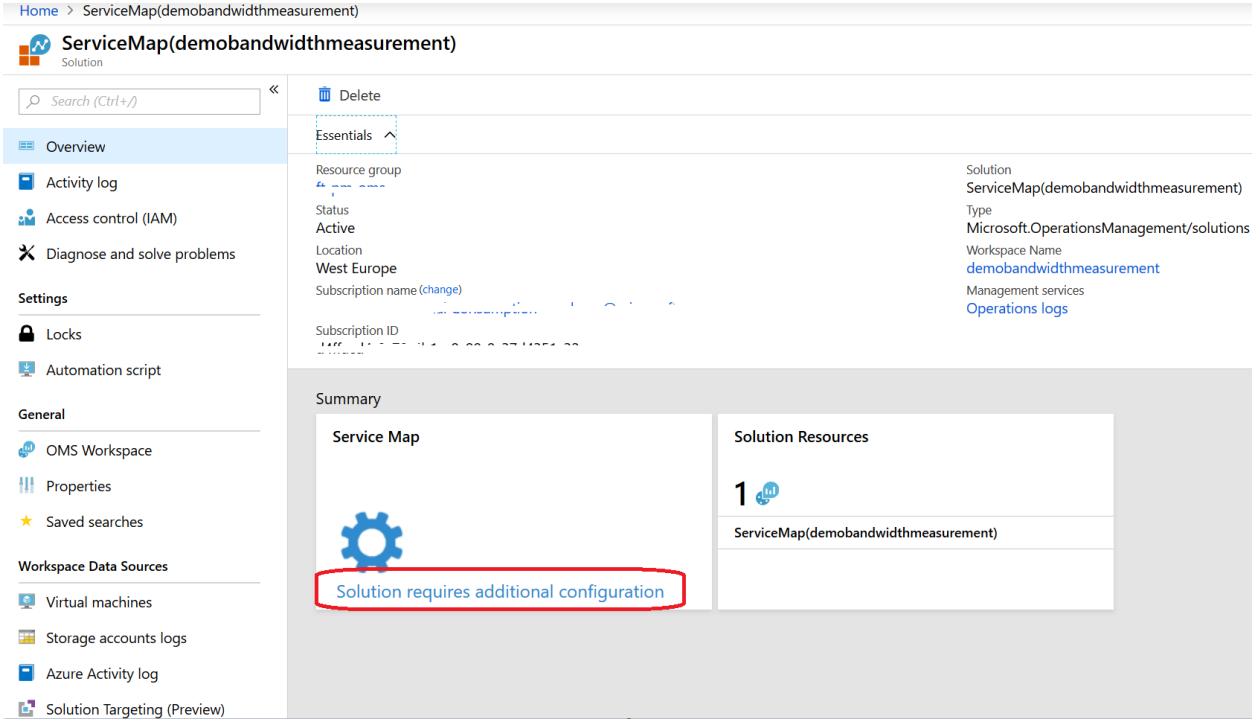


- Find and add 'Network Performance Monitor'
- Find and add 'Service Map'

d. You can also navigate to the resource (Service Map) by going to Log Analytics, selecting the workspace and clicking on 'workspace summary'



e. Your Solutions will then indicate they require further config, this is mainly the need for client setup and will change once it detects a client connected.

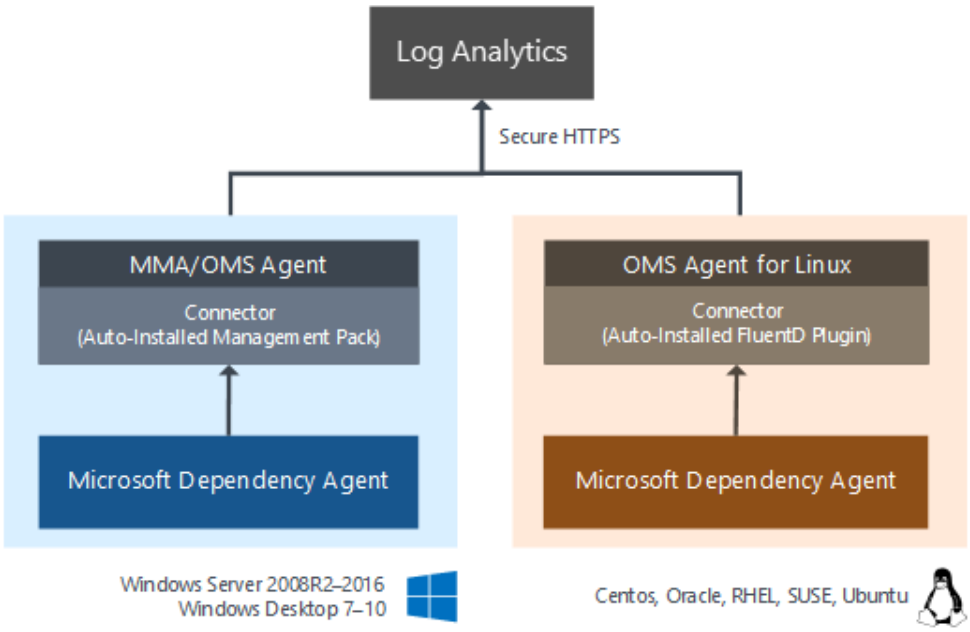


And that's it for the server-side setup. Now we need to configure some clients to provide data to the solution.

4.4 Client-Side Configuration

To send the data into our new solution, we need to install two client-side applications '**Microsoft Dependency Agent**' and '**Microsoft Monitoring Agent**'

The dependency agent collects the data we need and sends it to the Monitoring agent for transmission to Log analytics workspace. You have access to the ingested data from your Log analytics workspace in Azure.



Microsoft Dependency Agent install links for 64-bit Windows and Linux Machines:

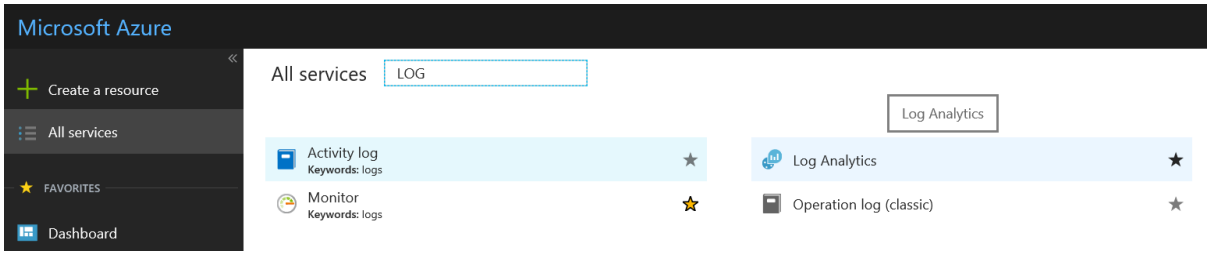
https://aka.ms/dependencyagentwindows

https://aka.ms/dependencyagentlinux

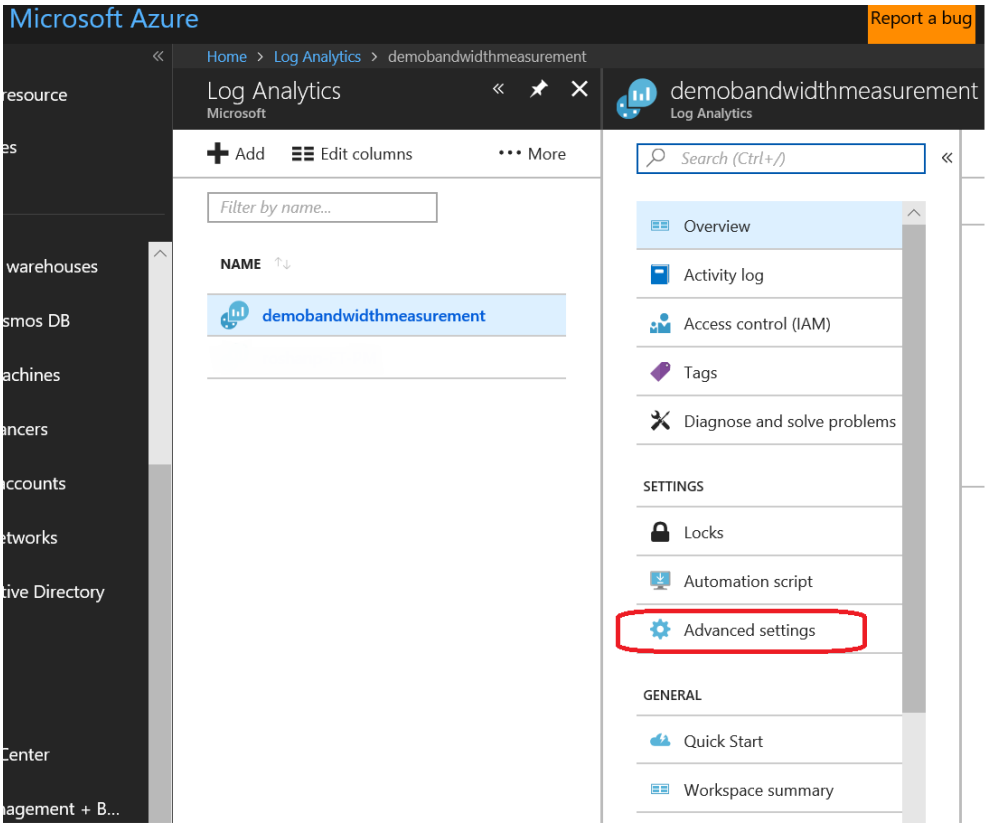
More information on scripting this install can be found here.

1. To obtain the correct **Microsoft Monitoring Agent** you'll need to go through the Azure Portal.

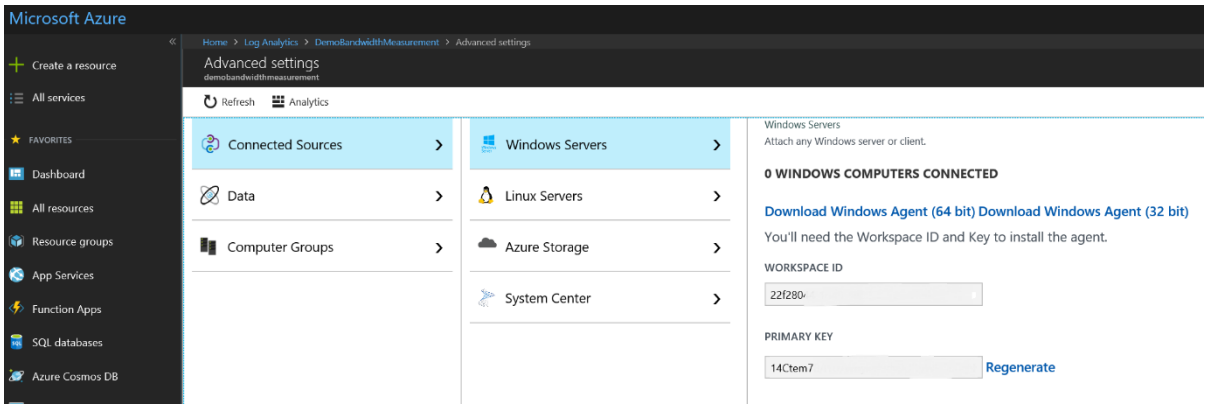
a. In the Azure Portal click on "All Services" then search for 'Log Analytics' and then click on it



b. Select your workspace, click on "Advanced Settings" from the workspace blade



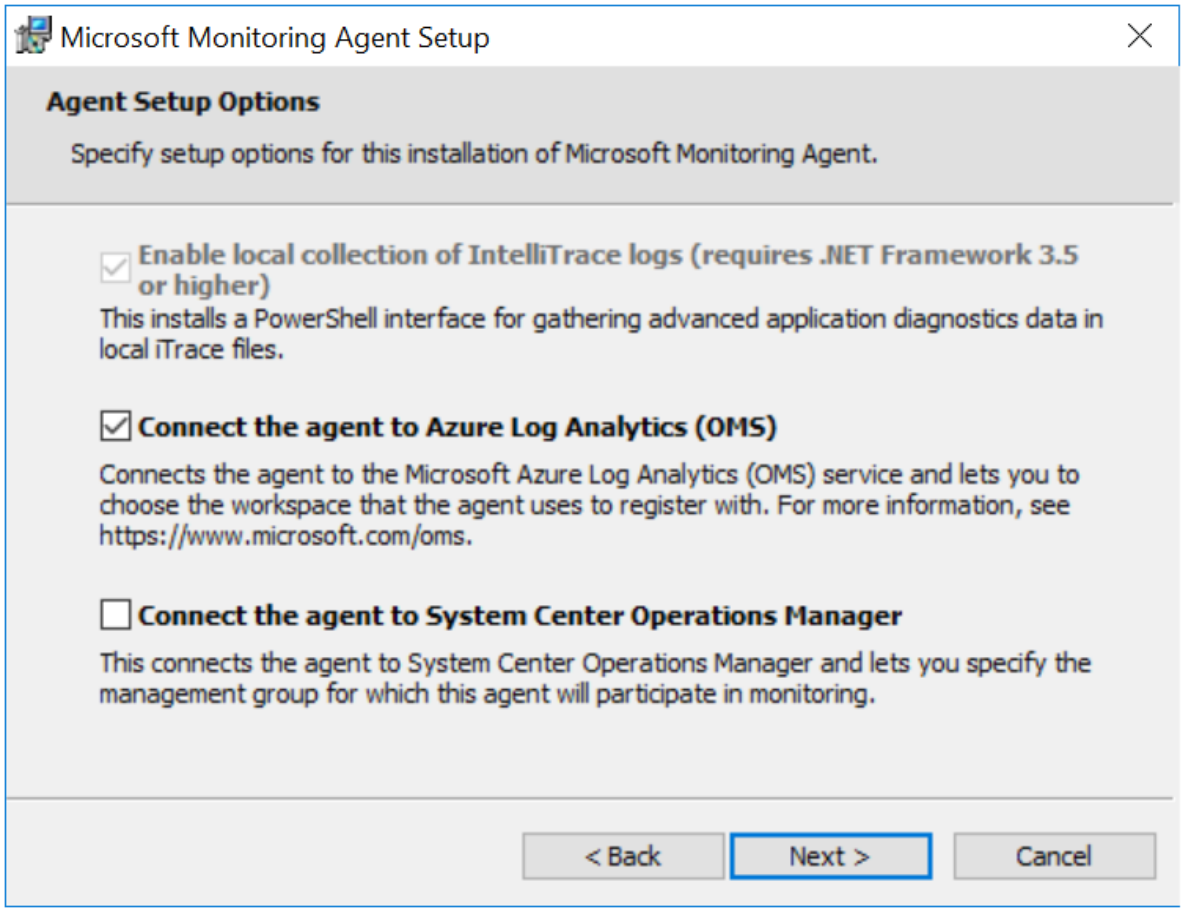
c. Here we have both the install link for the Monitoring Agent we need, and also some key information to connect it to our Log analytics worksapce.



d. Download and install the appropriate Windows Agent (64 bit) and install it on the client/s you wish to monitor

e. Also copy the Workspace ID and Primary Key.

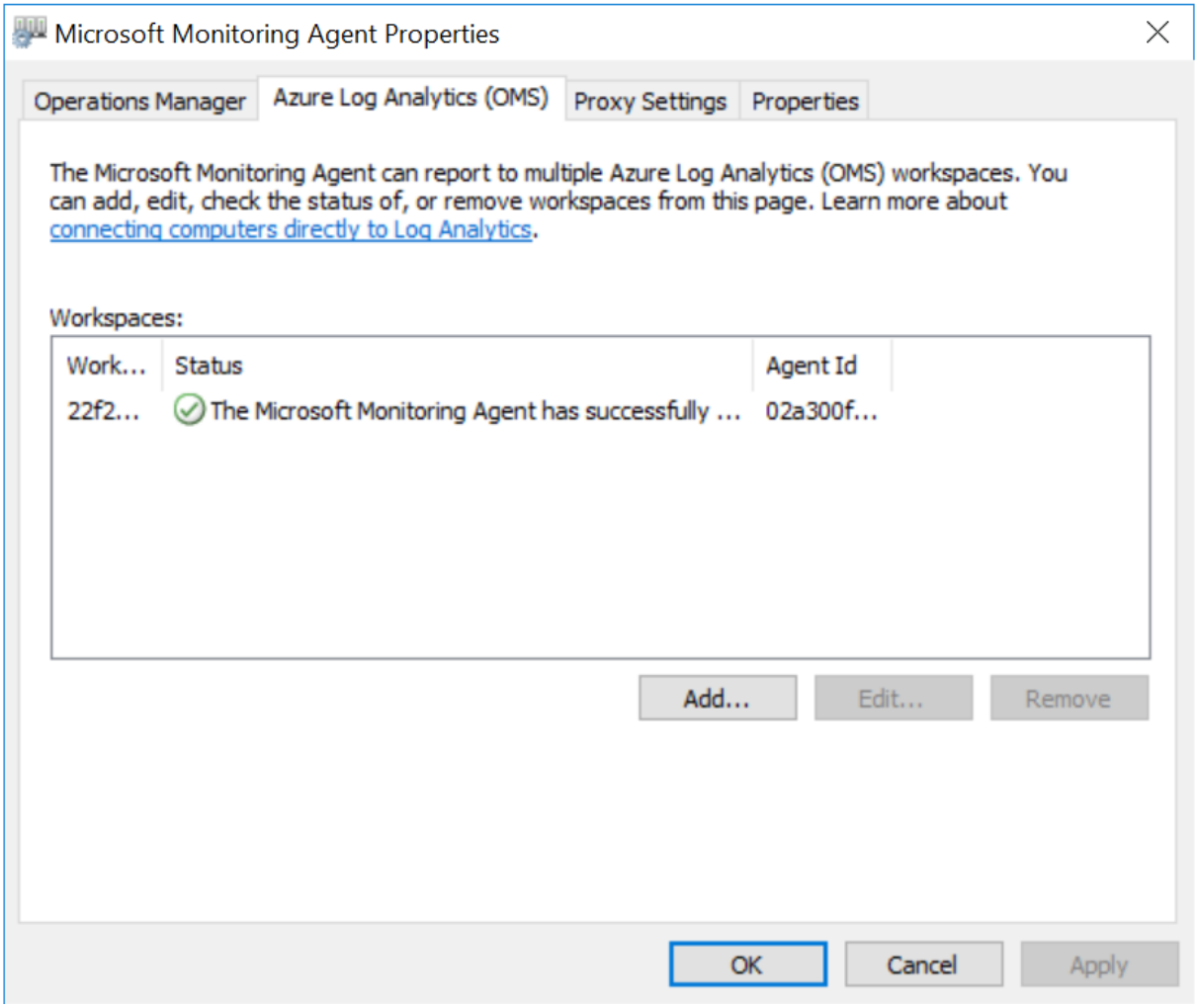
f. Once the monitoring agent is installed, a window will pop up. We require the middle option to connect to our Azure Log Analytics workspace. Select and click next.



g. Next, we'll be asked for our Workspace ID and Primary Key we copied from the Azure Portal above. Most customers will require the Azure Commercial instance. If the client uses a proxy to connect to the internet, then this need configuring by clicking the advanced button.

h. Click next and select whether to use Windows Update for the agent (recommended) then next again.

i. To check installation succeeded, open the control panel and select Microsoft Monitoring Agent. If successfully connected you should see a green tick in the status bar on the 'Azure Log Analytics' Tab.

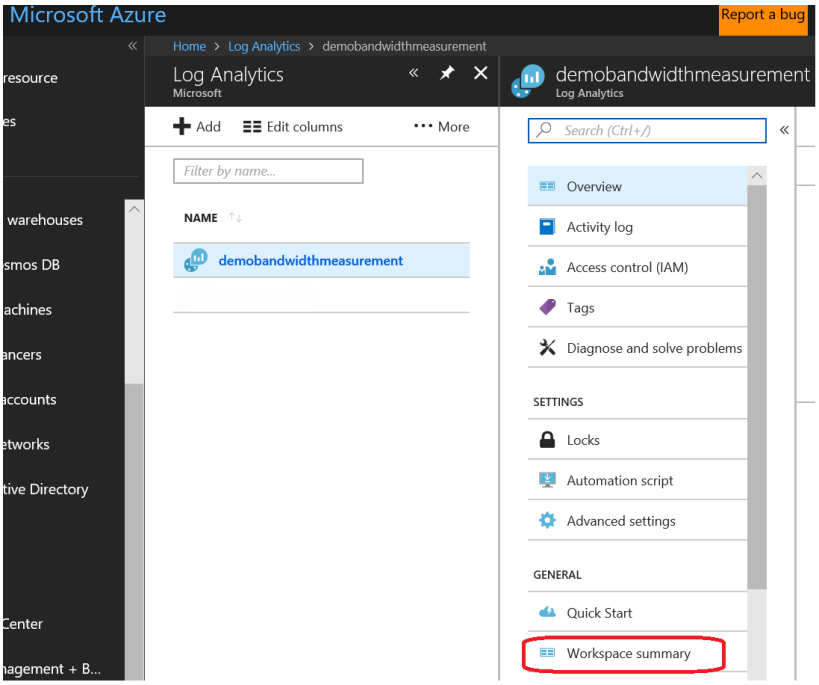


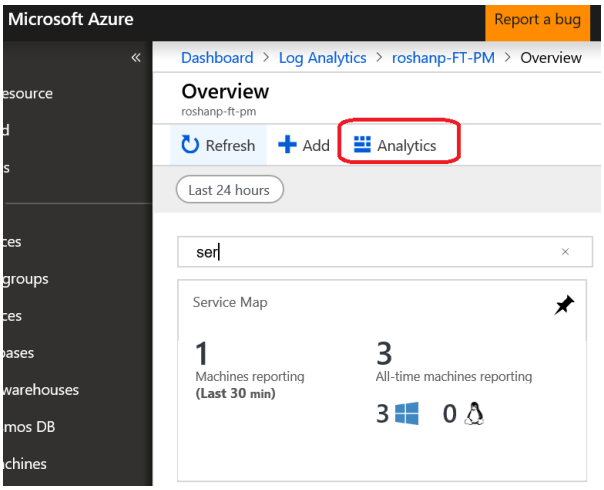
j. And that's it! Sit back and wait for your client to talk to Log analytics workspace and ingest data which should normally be within 5-10 minutes.

4.5 Analyzing Data in Log Analytics

After about 5 minutes you should be able to see your machine is reporting into Log analytics by logging into your portal. You can do this in two ways.

a. Log into your Azure Portal at https://portal.azure.com click on 'All Resources' and click on your Log Analytics workspace, if you go to the 'Workspace Summary' you will notice that Service Map has been configured and collecting data

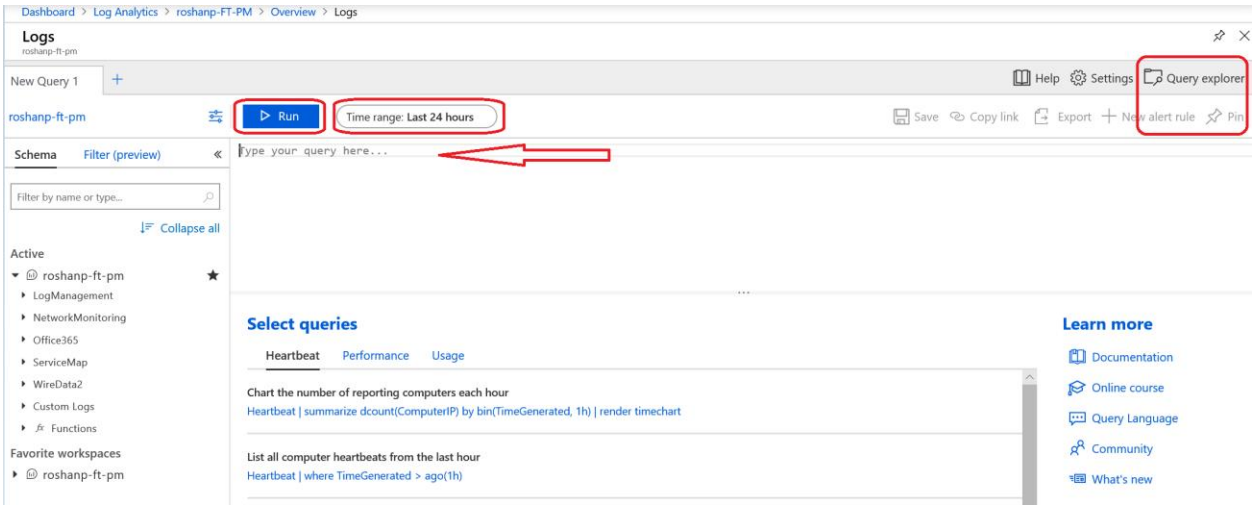




b. Then click on 'Analytics' at the top of the page which opens up in a new tab. If you click on Service Map tab it will show you a summary of network traffic information collected. We will use Analytics so you can run queries against the workspace to extract specific information we need for Bandwidth.

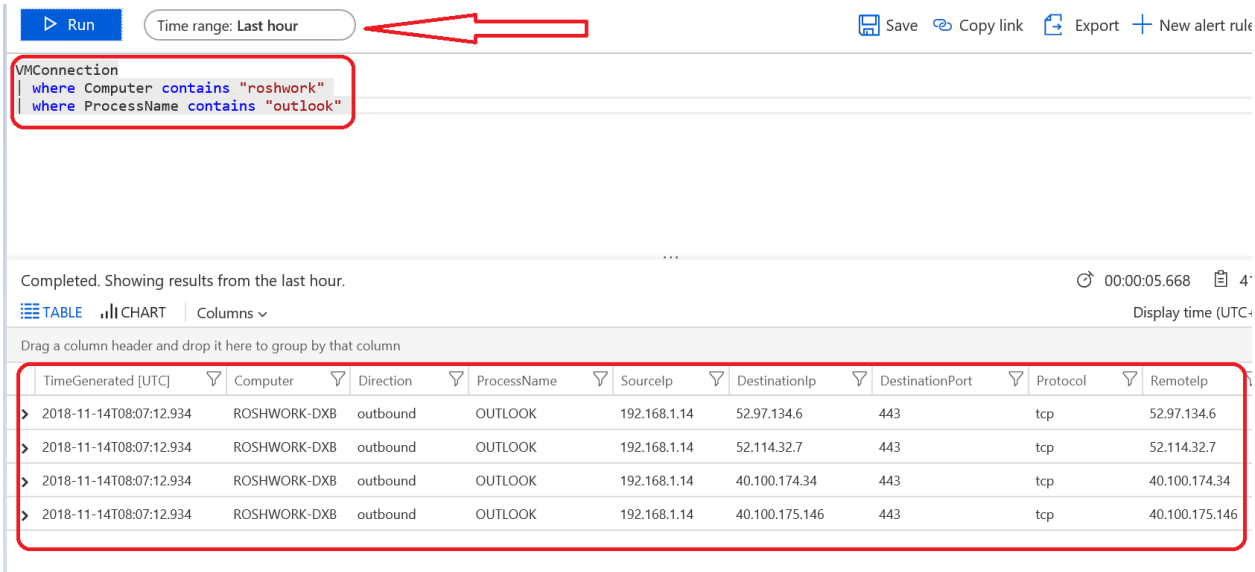
c. There are a couple of things to take note there if you are using Log analytics for the first time.

Notice you are now in Log analytics playground, you can use this console to run log analytics queries and review data that is stored in the workspace. You also have the ability to specify time range or save queries for frequent use in 'Query Explorer'



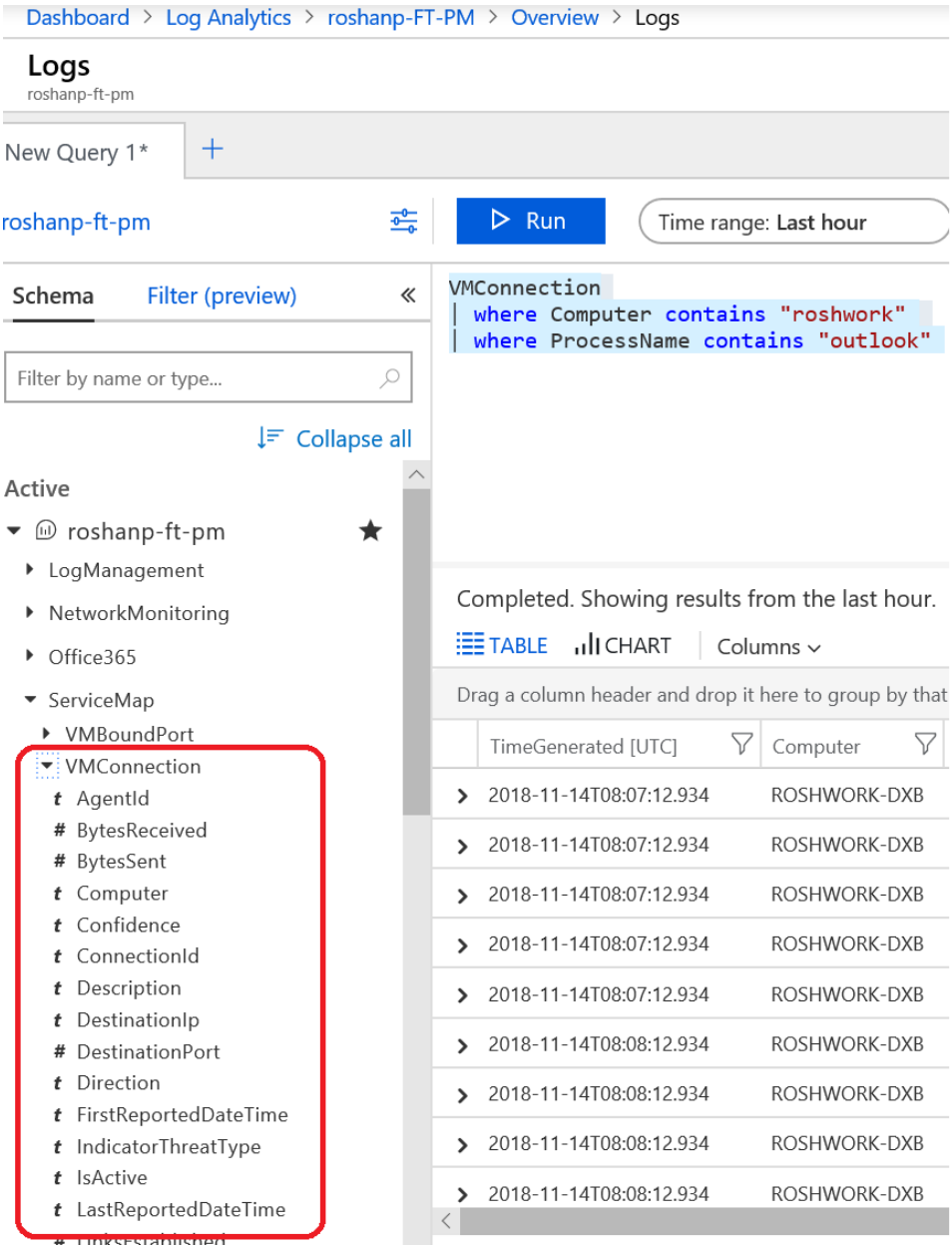
d. Run a simple query like the following, replace 'DESKTOP' with your computer name. Notice the time range by default is set to Last 24 hours.

```
VMConnection
| where Computer contains "DESKTOP"
| where ProcessName contains "outlook"
```



If the query runs successfully you should see the results showing the data table and records. In the example above we are also setting the time range for the last one hour.

e. Service Map has the following records we can search from, you can see this in the Schema on the left hand side, expand the Service Map, VMConnection table



Service Map has this information which is aggregated in 1 minute intervals. For example, a single TCP session will have the bytes sent/received/total for the previous 1 minute logged as an entry. This enables us to query the data for information such as the following examples:

- Bandwidth used for a particular process or set of processes over a set period of time
- Bandwidth used by the machine over a set period of time
- Bandwidth used in connections to a specific port
- Bandwidth used to a specific IP address or range of addresses
- Number of TCP connections a process has open at any one time
- IP geolocation of the endpoints connected to

Essentially, if we know the process name or the destination IP address we can easily monitor the endpoints used for any of the fields above.

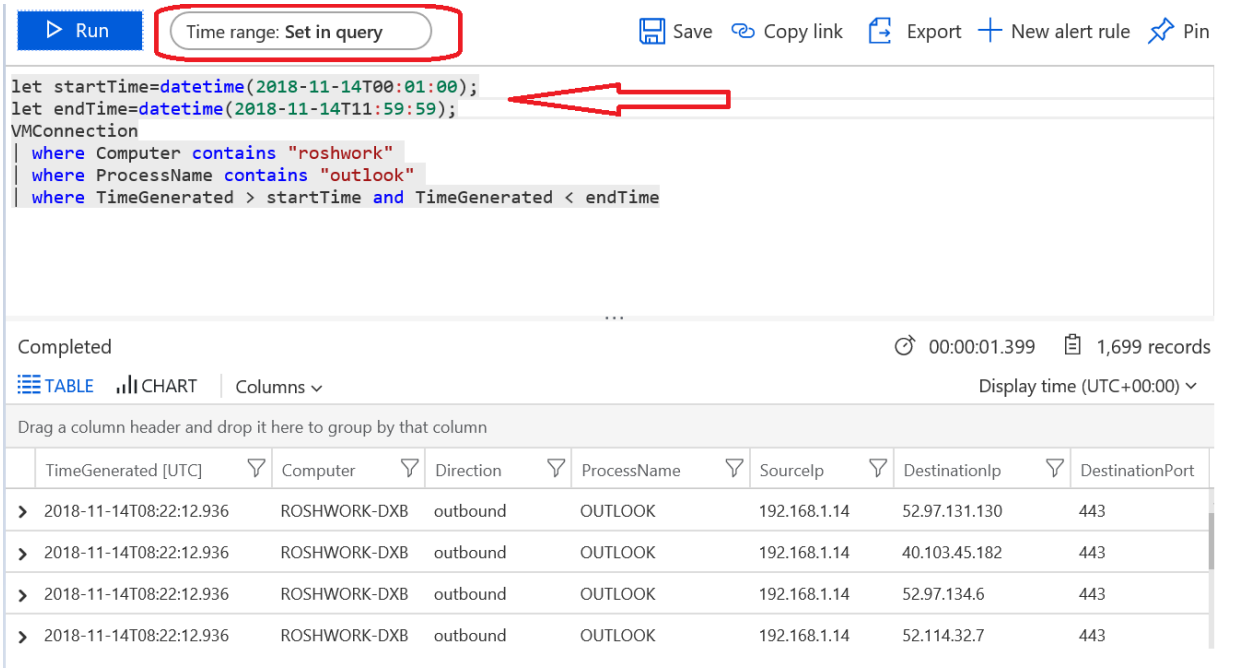
From an Office 365 perspective this allows us to collect very accurate data for bandwidth usage for pilot users to gain an accurate figure for planning for a wider rollout.

For example, you may wish to see the bandwidth usage of the Outlook client over the course of 24 hours to see how much bandwidth your Exchange connectivity may require.

To do this we query Service Map for a specific PC (or multiple if required) in this case a machine called 'DESKTOP' where the process name contains 'Outlook', where the TCP session start time was in the last day and output the total bytes used in each sample of 1 minute (TimeGenerated).

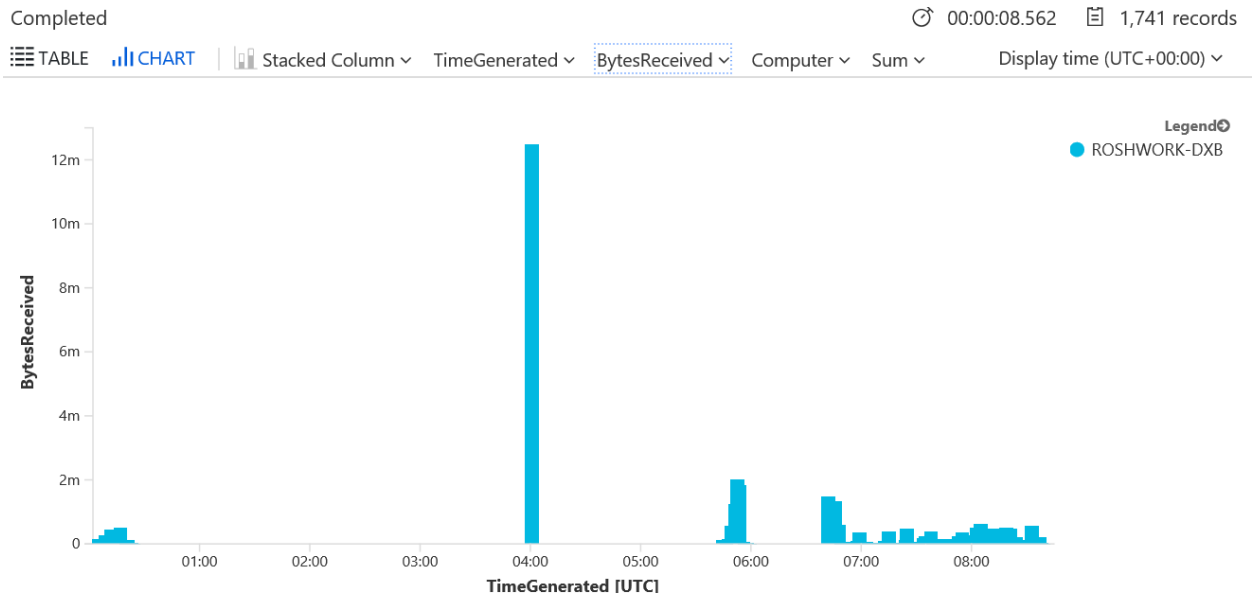
Add the above to the query then hit 'Run'. You need to ensure the timeframe for the query is set correctly at the top, in this case as we're specifying one day, it needs to be at least 'Last 24 Hours' which allows us to search through the data for the last 24 hours or you can specify the time range in the query itself as shown below

```
let startTime=datetime(2018-11-14T00:01:00);
let endTime=datetime(2018-11-14T11:59:59);
VMConnection
| where Computer contains "roshwork"
| where ProcessName contains "outlook"
| where TimeGenerated > startTime and TimeGenerated < endTime
```



If the query worked, you'll see a table of data output. We can then turn this into a graph within Analytics by clicking 'Chart' then when rendered, click 'Stacked Column' and change to Line or any other graph you prefer.

In the graph produced, we can clearly see the bytes used by the Outlook process for the time range specified on this machine. This allows us to see the peaks and normal usage and thus allow us to plan for bandwidth across all our clients.



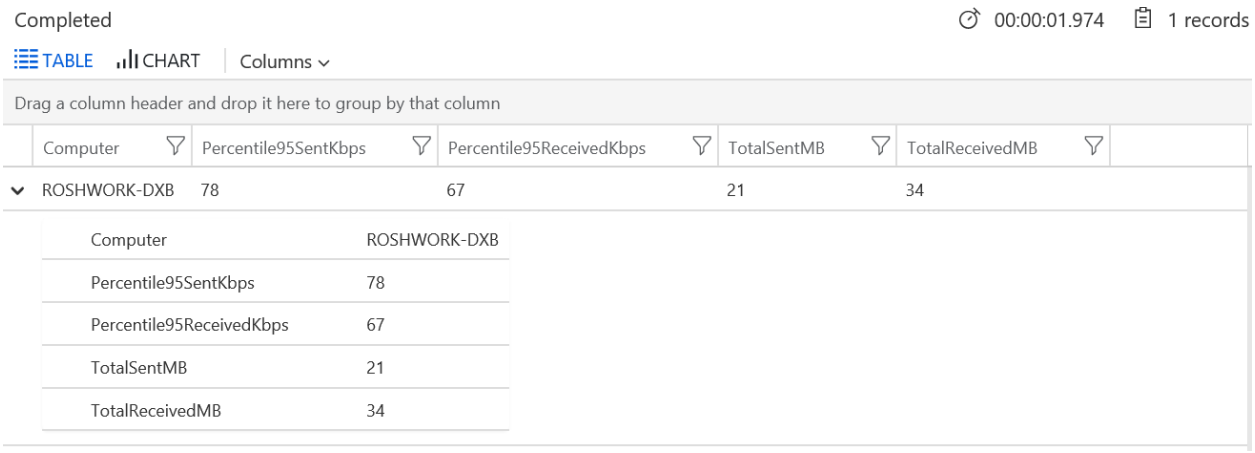
If we just want to find out the key points, i.e. 95th percentile of sent/received Kbps and Total sent/received MB over a time period, we can run the following query:

```
let startTime=datetime(2018-11-14T00:01:00);
let endTime=datetime(2018-11-14T11:59:59);
VMConnection
| where Computer contains "desktop"
| where ProcessName contains "outlook"
| where TimeGenerated > startTime and TimeGenerated < endTime

| summarize SentKbps = (sum(BytesSent)/1024*8/60), ReceivedKbps = (sum(BytesReceived)/1024*8/60) by bin(TimeGenerated, 1m), Computer
| summarize Percentile95SentKbps = round(percentile(SentKbps, 95),2), Percentile95ReceivedKbps = round(percentile(ReceivedKbps, 95),2) by Computer
| join (
  VMConnection
   | where Computer contains "desktop"
   | where ProcessName contains "outlook"

   | where TimeGenerated > startTime and TimeGenerated < endTime

   | summarize TotalSentMB = sum(BytesSent)/1024/1024, TotalReceivedMB = sum(BytesReceived)/1024/1024 by Computer
) on Computer
| project Computer, Percentile95SentKbps, Percentile95ReceivedKbps, TotalSentMB, TotalReceivedMB
```

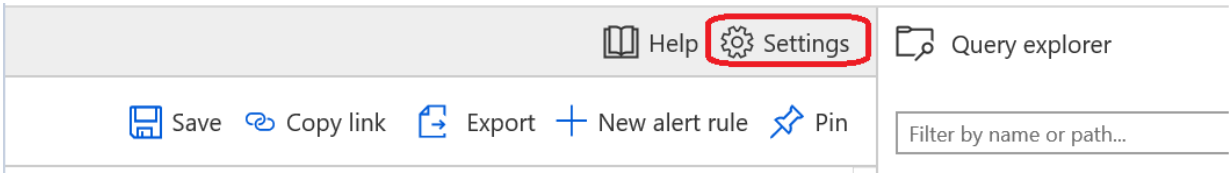


This method can be used for any process running on the machine and will work regardless of whether a proxy is used or not because we're monitoring the process, not the IP address of the endpoint (which will always be the proxy IP if one is used for all non-internal traffic).

4.6 Example Queries

The data can be analyzed to a limited extent within the Analytics portal itself. For more comprehensive analysis we can use Power BI which is covered in the next session. Regardless, the Power BI query can be obtained within Azure Analytics via the example queries below.

*Time is defined in UTC by default, you can change this by going to the 'Settings' tab



*Full Query = Give me all the data and I will perform my own query to filter it

*Key Points Query = Filter the data and display only the key points, for example SentKbps, ReceivedKbps, Total Sent and Received in MB. For Sent/Received Kbps the query is set to use 95th percentile function you can modify (increase/decrease) this as per your requirement.

EXO Full Query (Exchange Online)

```
let startTime=datetime(2020-01-03T00:01:00);
let endTime=datetime(2020-01-07T23:59:59);
VMConnection
| where ProcessName contains "outlook"
| where TimeGenerated > startTime and TimeGenerated < endTime
```

EXO Key Points Query

```
let startTime=datetime(2020-01-03T00:01:00);
let endTime=datetime(2020-01-07T23:59:59);
VMConnection
| where ProcessName contains "outlook"
```

```
| where TimeGenerated > startTime and TimeGenerated < endTime

| summarize SentKbps = (sum(BytesSent)/1024*8/60), ReceivedKbps = (sum(BytesReceived)/1024*8/60) by bin(TimeGenerated, 1m), Computer
| summarize Percentile95SentKbps = round(percentile(SentKbps, 95),2), Percentile95ReceivedKbps = round(percentile(ReceivedKbps, 95),2) by Computer
| join (
  VMConnection
   | where ProcessName contains "outlook"

   | where TimeGenerated > startTime and TimeGenerated < endTime

   | summarize TotalSentMB = sum(BytesSent)/1024/1024, TotalReceivedMB = sum(BytesReceived)/1024/1024 by Computer
) on Computer
| project Computer, Percentile95SentKbps, Percentile95ReceivedKbps, TotalSentMB, TotalReceivedMB
```

If you need filter for traffic from a specific computer, you can always add a filter like:

```
| where Computer contains "XXXX"
```

The query logic is the same for Skype for Business and Teams too but just replaced ‘outlook’ with ‘lync’ or ‘teams’

The query logic for **SPO (Sharepoint Online)** and **OD4B (OneDrive for Business)** is as follows since it is based on destination IP ranges:

SPO/OD4B Full Query (Sharepoint Online / OneDrive for Business)

```
let startTime=datetime(2020-01-02T01:00:00);
let endTime=datetime(2020-01-07T23:30:00);
VMConnection
| where Computer contains "XXXX"
| where TimeGenerated > startTime and TimeGenerated < endTime
| where (parse_ipv4(DestinationIp) >= parse_ipv4('13.107.136.0') and parse_ipv4(DestinationIp) <= parse_ipv4('13.107.139.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('40.108.128.0') and parse_ipv4(DestinationIp) <= parse_ipv4('40.108.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('52.104.0.0') and parse_ipv4(DestinationIp) <= parse_ipv4('52.107.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('104.146.128.0') and parse_ipv4(DestinationIp) <= parse_ipv4('104.146.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('150.171.40.0') and parse_ipv4(DestinationIp) <= parse_ipv4('150.171.43.255'))
```

SPO/OD4B Key Points Query

```
let startTime=datetime(2020-01-02T01:00:00);
let endTime=datetime(2020-01-07T23:30:00);
VMConnection
| where Computer contains "XXXX"
| where TimeGenerated > startTime and TimeGenerated < endTime
| where (parse_ipv4(DestinationIp) >= parse_ipv4('13.107.136.0') and parse_ipv4(DestinationIp) <= parse_ipv4('13.107.139.255'))
```

```
or (parse_ipv4(DestinationIp) >= parse_ipv4('40.108.128.0') and parse_ipv4(DestinationIp) <= parse_ipv4('40.108.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('52.104.0.0') and parse_ipv4(DestinationIp) <= parse_ipv4('52.107.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('104.146.128.0') and parse_ipv4(DestinationIp) <= parse_ipv4('104.146.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('150.171.40.0') and parse_ipv4(DestinationIp) <= parse_ipv4('150.171.43.255'))
| summarize SentKbps = (sum(BytesSent)/1024*8/60), ReceivedKbps = (sum(BytesReceived)/1024*8/60) by bin(TimeGenerated, 1m), Computer
| summarize Percentile95SentKbps = round(percentile(SentKbps, 95),2), Percentile95ReceivedKbps = round(percentile(ReceivedKbps, 95),2) by Computer
| join (
  VMConnection
   | where Computer contains "XXXX"
   | where TimeGenerated > startTime and TimeGenerated < endTime
| where (parse_ipv4(DestinationIp) >= parse_ipv4('13.107.136.0') and parse_ipv4(DestinationIp) <= parse_ipv4('13.107.139.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('40.108.128.0') and parse_ipv4(DestinationIp) <= parse_ipv4('40.108.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('52.104.0.0') and parse_ipv4(DestinationIp) <= parse_ipv4('52.107.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('104.146.128.0') and parse_ipv4(DestinationIp) <= parse_ipv4('104.146.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('150.171.40.0') and parse_ipv4(DestinationIp) <= parse_ipv4('150.171.43.255'))
| summarize TotalSentMB = sum(BytesSent)/1024/1024, TotalReceivedMB = sum(BytesReceived)/1024/1024 by Computer
) on Computer
| project Computer, Percentile95SentKbps, Percentile95ReceivedKbps, TotalSentMB, TotalReceivedMB
```

Please note that the following filter for computer name is optional, comment it or remove it if you want to view the results for all computers connected to your workspace.

```
Computer contains "XXXX"
```

Only for SPO/OD4B the highlighted section in orange may need to be replaced if the IP ranges used for SPO/OD4B changes, the ranges are published in https://aka.ms/o365ip

You can use the Powershell script Generate-LAQuery-v3.ps1 in Github to generate the query with up to date IP ranges for the highlighted orange section only (copy/paste the output from Generate-LAQuery-v3.ps1 into your log analytics query). Syntax examples for Generate-LAQuery-v3.ps1 are available in the comments section of the file.

```
PS C:\temp> .\Generate-LAQuery-v3.ps1

cmdlet Generate-LAQuery-v3.ps1 at command pipeline position 1
Supply values for the following parameters:
services: Sharepoint
startdate: 2019-12-01T09:00
enddate: 2019-12-23T09:00
Log Analytics query written to 'C:\Users\[redacted]\AppData\Local\Temp\LogAnalyticsquery_Sharepoint_25-12-2019_15-27.txt'
PS C:\temp>
```

Teams Full Query (Microsoft Teams, including UDP for media traffic)

```
let startTime=datetime(2020-01-03T00:01:00);
let endTime=datetime(2020-01-07T23:59:59);
VMConnection
| where ProcessName contains "teams"
| where TimeGenerated > startTime and TimeGenerated < endTime
```

Teams Key Points Query

```
let startTime=datetime(2020-01-03T00:01:00);
let endTime=datetime(2020-01-07T23:59:59);
VMConnection
| where ProcessName contains "teams"
| where TimeGenerated > startTime and TimeGenerated < endTime

| summarize SentKbps = (sum(BytesSent)/1024*8/60), ReceivedKbps = (sum(BytesReceived)/1024*8/60) by bin(TimeGenerated, 1m), Computer
| summarize Percentile95SentKbps = round(percentile(SentKbps, 95),2), Percentile95ReceivedKbps = round(percentile(ReceivedKbps, 95),2) by Computer
| join (
  VMConnection
   | where ProcessName contains "teams"

   | where TimeGenerated > startTime and TimeGenerated < endTime

   | summarize TotalSentMB = sum(BytesSent)/1024/1024, TotalReceivedMB = sum(BytesReceived)/1024/1024 by Computer
) on Computer
| project Computer, Percentile95SentKbps, Percentile95ReceivedKbps, TotalSentMB, TotalReceivedMB
```

Query for 'Optimize' category endpoints

This query is useful when you would like to have a consolidated view of network bandwidth usage for all endpoints (IP's) that belong to the Optimize category. Refer here for Office 365 endpoint categories

```
let startTime=datetime(2020-01-02T01:00:00);
let endTime=datetime(2020-01-07T23:30:00);
VMConnection
| where Computer contains "XXXX"
| where TimeGenerated > startTime and TimeGenerated < endTime
| where (parse_ipv4(DestinationIp) >= parse_ipv4('13.107.6.152') and parse_ipv4(DestinationIp) <= parse_ipv4('13.107.6.153'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('13.107.18.10') and parse_ipv4(DestinationIp) <= parse_ipv4('13.107.18.11'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('13.107.128.0') and parse_ipv4(DestinationIp) <= parse_ipv4('13.107.131.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('23.103.160.0') and parse_ipv4(DestinationIp) <= parse_ipv4('23.103.175.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('40.96.0.0') and parse_ipv4(DestinationIp) <= parse_ipv4('40.103.255.255'))
```

```
or (parse_ipv4(DestinationIp) >= parse_ipv4('40.104.0.0') and parse_ipv4(DestinationIp) <= parse_ipv4('40.105.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('52.96.0.0') and parse_ipv4(DestinationIp) <= parse_ipv4('52.99.255.255'))
or (parse_ipv4(DestinationIp) == parse_ipv4('131.253.33.215'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('132.245.0.0') and parse_ipv4(DestinationIp) <= parse_ipv4('132.245.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('150.171.32.0') and parse_ipv4(DestinationIp) <= parse_ipv4('150.171.35.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('191.234.140.0') and parse_ipv4(DestinationIp) <= parse_ipv4('191.234.143.255'))
or (parse_ipv4(DestinationIp) == parse_ipv4('204.79.197.215'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('13.107.64.0') and parse_ipv4(DestinationIp) <= parse_ipv4('13.107.127.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('52.112.0.0') and parse_ipv4(DestinationIp) <= parse_ipv4('52.115.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('13.107.136.0') and parse_ipv4(DestinationIp) <= parse_ipv4('13.107.139.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('40.108.128.0') and parse_ipv4(DestinationIp) <= parse_ipv4('40.108.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('52.104.0.0') and parse_ipv4(DestinationIp) <= parse_ipv4('52.107.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('104.146.128.0') and parse_ipv4(DestinationIp) <= parse_ipv4('104.146.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('150.171.40.0') and parse_ipv4(DestinationIp) <= parse_ipv4('150.171.43.255'))
| summarize SentKbps = (sum(BytesSent)/1024*8/60), ReceivedKbps = (sum(BytesReceived)/1024*8/60) by bin(TimeGenerated, 1m), Computer
| summarize Percentile95SentKbps = round(percentile(SentKbps, 95),2), Percentile95ReceivedKbps = round(percentile(ReceivedKbps, 95),2) by Computer
| join (
  VMConnection
   | where Computer contains "XXXX"
   | where TimeGenerated > startTime and TimeGenerated < endTime
| where (parse_ipv4(DestinationIp) >= parse_ipv4('13.107.6.152') and parse_ipv4(DestinationIp) <= parse_ipv4('13.107.6.153'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('13.107.18.10') and parse_ipv4(DestinationIp) <= parse_ipv4('13.107.18.11'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('13.107.128.0') and parse_ipv4(DestinationIp) <= parse_ipv4('13.107.131.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('23.103.160.0') and parse_ipv4(DestinationIp) <= parse_ipv4('23.103.175.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('40.96.0.0') and parse_ipv4(DestinationIp) <= parse_ipv4('40.103.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('40.104.0.0') and parse_ipv4(DestinationIp) <= parse_ipv4('40.105.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('52.96.0.0') and parse_ipv4(DestinationIp) <= parse_ipv4('52.99.255.255'))
or (parse_ipv4(DestinationIp) == parse_ipv4('131.253.33.215'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('132.245.0.0') and parse_ipv4(DestinationIp) <= parse_ipv4('132.245.255.255'))
```

```
or (parse_ipv4(DestinationIp) >= parse_ipv4('150.171.32.0') and parse_ipv4(DestinationIp) <= parse_ipv4('150.171.35.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('191.234.140.0') and parse_ipv4(DestinationIp) <= parse_ipv4('191.234.143.255'))
or (parse_ipv4(DestinationIp) == parse_ipv4('204.79.197.215'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('13.107.64.0') and parse_ipv4(DestinationIp) <= parse_ipv4('13.107.127.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('52.112.0.0') and parse_ipv4(DestinationIp) <= parse_ipv4('52.115.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('13.107.136.0') and parse_ipv4(DestinationIp) <= parse_ipv4('13.107.139.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('40.108.128.0') and parse_ipv4(DestinationIp) <= parse_ipv4('40.108.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('52.104.0.0') and parse_ipv4(DestinationIp) <= parse_ipv4('52.107.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('104.146.128.0') and parse_ipv4(DestinationIp) <= parse_ipv4('104.146.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('150.171.40.0') and parse_ipv4(DestinationIp) <= parse_ipv4('150.171.43.255'))
| summarize TotalSentMB = sum(BytesSent)/1024/1024, TotalReceivedMB = sum(BytesReceived)/1024/1024 by Computer
) on Computer
| project Computer, Percentile95SentKbps, Percentile95ReceivedKbps, TotalSentMB, TotalReceivedMB
```

Please note that the following filter for computer name is optional, comment it or remove it if you want to view the results for all computers connected to your workspace.

```
Computer contains "XXXX"
```

The highlighted section in orange may need to be replaced if the IP ranges used for Optimize category changes, the ranges are published in https://aka.ms/o365ip

You can use the Powershell script Generate-LAQuery-optimize-category-v3.ps1 in Github to generate the query with up to date IP ranges for the highlighted orange section only (copy/paste the output from Generate-LAQuery-optimize-category-v3.ps1 into your log analytics query). Syntax examples for Generate-LAQuery-optimize-category-v3.ps1 are available in the comments section of the file.

```
PS C:\temp> .\Generate-LAQuery-optimize-category-v3.ps1

cmdlet Generate-LAQuery-optimize-category-v3.ps1 at command pipeline position 1
Supply values for the following parameters:
Category: optimize
startdate: 2019-12-01T09:00
enddate: 2019-12-23T09:00
Log Analytics query written to 'C:\Users\[redacted]\AppData\Local\Temp\LogAnalyticsquery_optimize_25-12-2019_15-48.txt'
PS C:\temp>
```

Query for LinksLive (TCP Connections) per Process

This query is useful when you would like to measure the number of TCP connections that were used by a specific process during the defined time period. The query is set to use 95th percentile function you can modify (increase/decrease) this as per your requirement.

```
let startTime=datetime(2019-12-01T00:01:00);
let endTime=datetime(2019-12-23T23:59:00);
```

```
VMConnection
| where TimeGenerated > startTime and TimeGenerated < endTime
| where Direction == "outbound"
| where Protocol == "tcp"
| where ProcessName == "OUTLOOK"
| summarize sum(LinksLive) by TimeGenerated, Computer
| summarize percentile(sum_LinksLive, 95) by Computer
```

Query for LinksLive (TCP Connections) for Optimize category endpoints

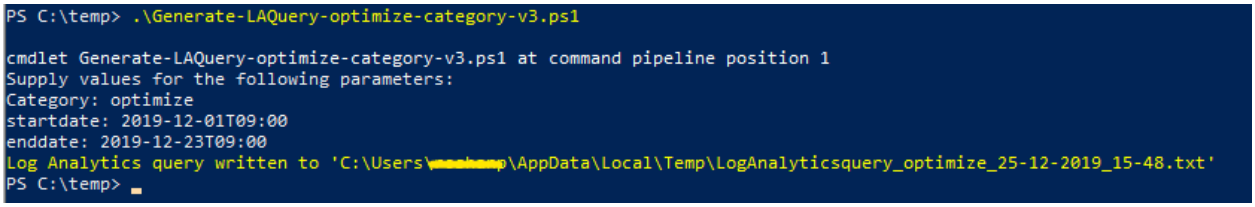
This query is useful when you would like to measure the number of TCP connections that were used by any process connecting to an IP endpoint part of 'Optimize' category during the defined time period. The query is set to use 95th percentile function you can modify (increase/decrease) this as per your requirement.

```
let startTime=datetime(2019-12-01T00:01:00);
let endTime=datetime(2019-12-23T23:59:00);
VMConnection
| where TimeGenerated > startTime and TimeGenerated < endTime
| where (parse_ipv4(DestinationIp) >= parse_ipv4('13.107.6.152') and parse_ipv4(DestinationIp) <= parse_ipv4('13.107.6.153'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('13.107.18.10') and parse_ipv4(DestinationIp) <= parse_ipv4('13.107.18.11'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('13.107.128.0') and parse_ipv4(DestinationIp) <= parse_ipv4('13.107.131.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('23.103.160.0') and parse_ipv4(DestinationIp) <= parse_ipv4('23.103.175.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('40.96.0.0') and parse_ipv4(DestinationIp) <= parse_ipv4('40.103.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('40.104.0.0') and parse_ipv4(DestinationIp) <= parse_ipv4('40.105.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('52.96.0.0') and parse_ipv4(DestinationIp) <= parse_ipv4('52.99.255.255'))
or (parse_ipv4(DestinationIp) == parse_ipv4('131.253.33.215'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('132.245.0.0') and parse_ipv4(DestinationIp) <= parse_ipv4('132.245.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('150.171.32.0') and parse_ipv4(DestinationIp) <= parse_ipv4('150.171.35.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('191.234.140.0') and parse_ipv4(DestinationIp) <= parse_ipv4('191.234.143.255'))
or (parse_ipv4(DestinationIp) == parse_ipv4('204.79.197.215'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('13.107.64.0') and parse_ipv4(DestinationIp) <= parse_ipv4('13.107.127.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('52.112.0.0') and parse_ipv4(DestinationIp) <= parse_ipv4('52.115.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('13.107.136.0') and parse_ipv4(DestinationIp) <= parse_ipv4('13.107.139.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('40.108.128.0') and parse_ipv4(DestinationIp) <= parse_ipv4('40.108.255.255'))
```

```
or (parse_ipv4(DestinationIp) >= parse_ipv4('52.104.0.0') and parse_ipv4(DestinationIp) <= parse_ipv4('52.107.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('104.146.128.0') and parse_ipv4(DestinationIp) <= parse_ipv4('104.146.255.255'))
or (parse_ipv4(DestinationIp) >= parse_ipv4('150.171.40.0') and parse_ipv4(DestinationIp) <= parse_ipv4('150.171.43.255'))
| summarize sum(LinksLive) by TimeGenerated, Computer
| summarize percentile(sum_LinksLive, 95) by Computer
```

The highlighted section in orange may need to be replaced if the IP ranges used for Optimize category changes, the ranges are published in https://aka.ms/o365ip

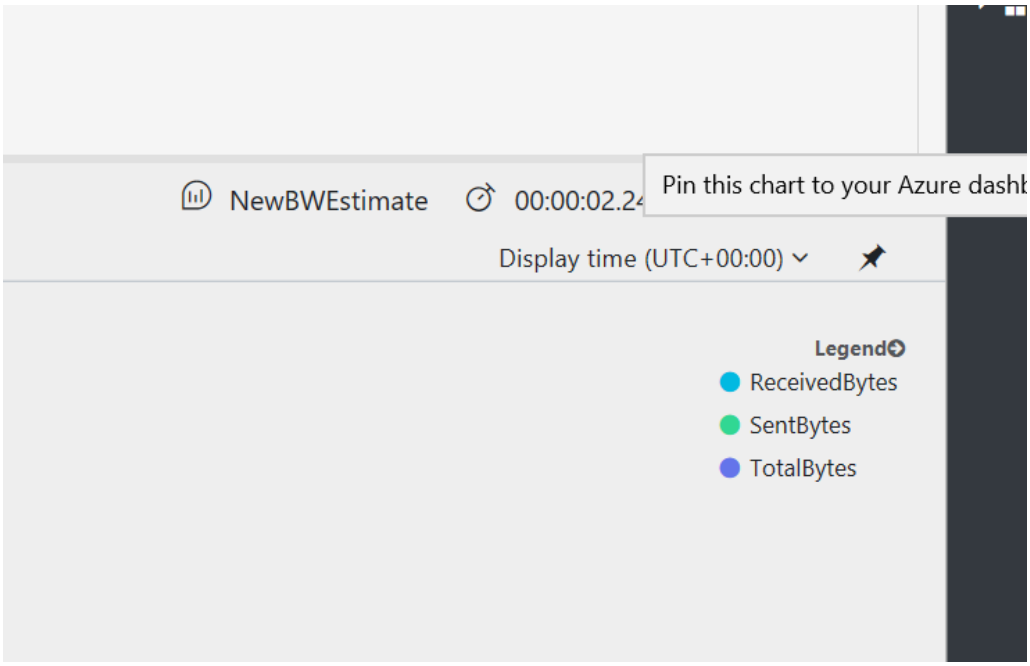
You can use the Powershell script Generate-LAQuery-optimize-category-v3.ps1 in Github to generate the query with up to date IP ranges for the highlighted orange section only (copy/paste the output from Generate-LAQuery-optimize-category-v3.ps1 into your log analytics query). Syntax examples for Generate-LAQuery-optimize-category-v3.ps1 are available in the comments section of the file.



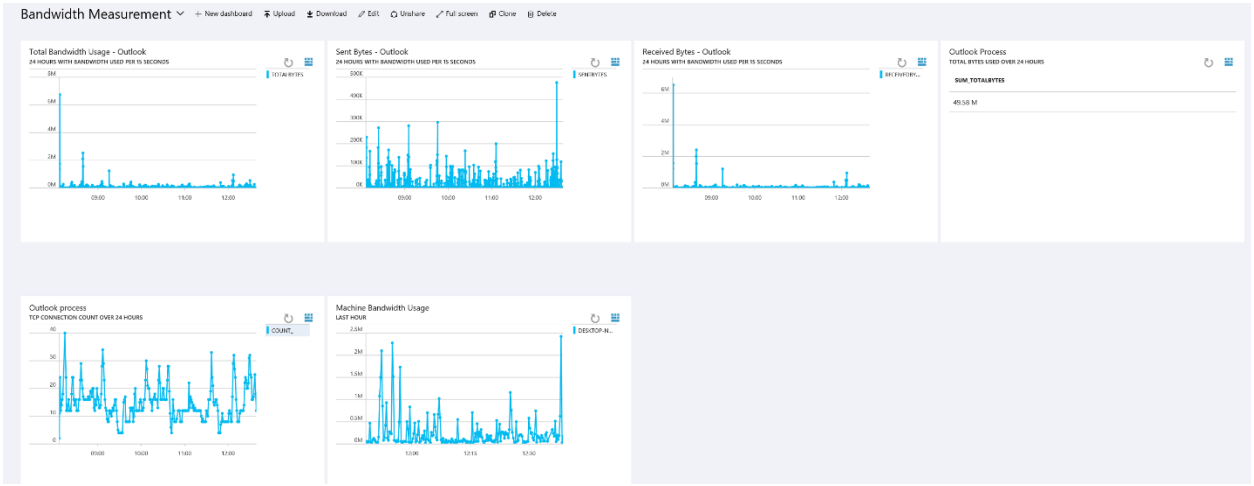
4.7 Adding graphs to an Azure Dashboard

With each of the above queries, when you create a graph to show the information, you have the option to pin the graph to an Azure dashboard

Once the table is displayed, click 'Chart' then select the view you wish. Once the graph is displayed, simply click on the pin icon in the top right-hand corner of the graph



You can add multiple graphs to create a dashboard to show the data you need to view as soon as you log into Azure. These graphs can easily be refreshed to keep them current without having to run the query in Azure Analytics again.



4.8 Building a Dashboard in Power BI

Whilst Azure Analytics provides some great features to quickly analyse the data, it does have some limitations as it is not intended for extensive analysis. Firstly the number of rows returned in a query is limited to 10000 which may cause issues with larger sample sets.

Also, the graph options are fairly limited so may not contain the type you require.

Thankfully the Analytics portal makes it easy to export the data for analysis elsewhere. Once you enter a query there is an option to export the data to CSV for loading into Microsoft Excel, or it will re-write the query into a Power BI ready version.

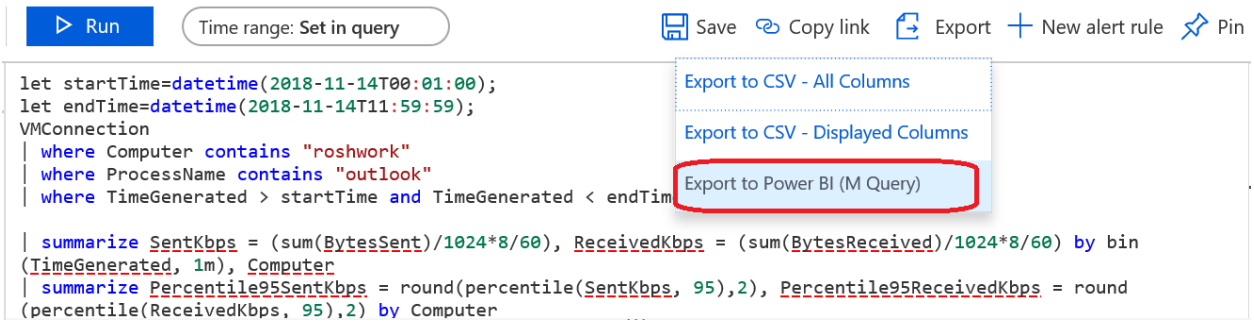
Here we'll walk through the steps to create a Power BI dashboard and use the Outlook process as an example.

1. Install Power BI Desktop

Firstly, you'll need to install Power BI desktop from here. Power BI desktop is the place where we'll create our dashboard which can then be published for viewing via a browser.

2. Obtain the Power BI query from Azure Log Analytics

Once Power BI Desktop is installed we'll first have to obtain a query to use from Azure Log Analytics. To export a query, we simply enter the query as normal into Azure Analytics, and click the page icon with an arrow pointing to the right, then click on 'Export to Power BI (M Query)



In this instance, to build an Outlook dashboard we don't have to be very specific, the following query will export all data related to Outlook for all users. For a specific machine uncomment the bottom line and add the machine name.

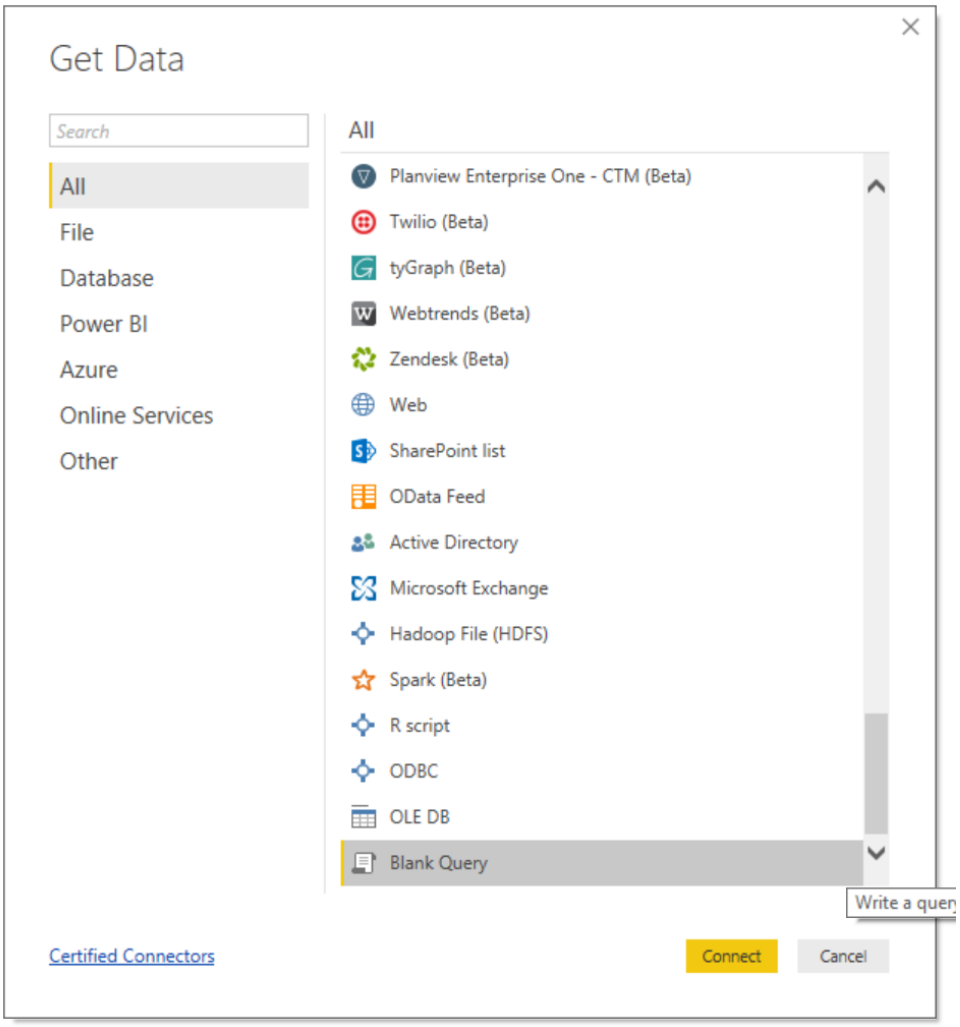
```
let startTime=datetime(2018-11-14T00:01:00);
let endTime=datetime(2018-11-14T11:59:59);
VMConnection
| where ProcessName contains "outlook"
| where TimeGenerated > startTime and TimeGenerated < endTime
```

By exporting all the data, we can choose what to display from within Power BI in different graphs in the same dashboard. Ensure the timeframe set at the top is what you wish to view in Power BI e.g. 24 Hours.

2. Create a new Dashboard in Power BI Desktop

a. Open Power BI Desktop and you should see a start-up screen displayed. Select 'Get Data' from the options on the left

b. On the next window which opens, scroll to the bottom of the options and select 'Blank Query' then click 'Connect'



c. In the window which opens we'll import the Power BI query we exported from Azure Log Analytics. Click 'Get Data' from the menu at the top, then 'Blank Query'.

d. In the new window which opens, click 'Advanced Editor' delete what is in the window and paste your query from Log Analytics. If all is well, a green tick should appear showing no syntax errors. If this does not occur, it's likely the copy/paste has copied some of the instructions at the top of the export.

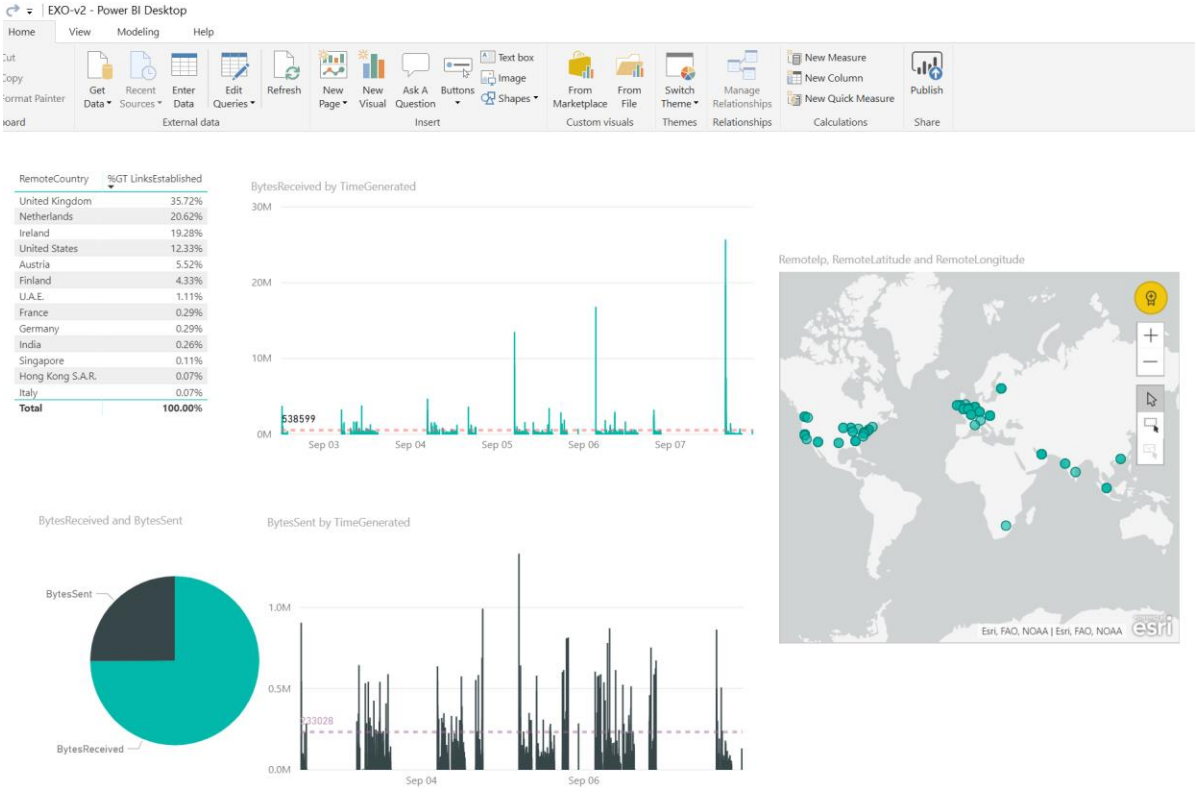
e. This will load the table of data into the window and when complete, click "Close and Apply" at the top to import into our blank dashboard.

Tip: Ensure you are logged in to PowerBI with the same Organizational account that you used to login to Azure and Log Analytics. Otherwise PowerBI will throw an Authentication warning since you won't have permissions to access the workspace.

f. Now we have the data loaded into the dashboard we can begin to create graphs showing the data we wish to see.

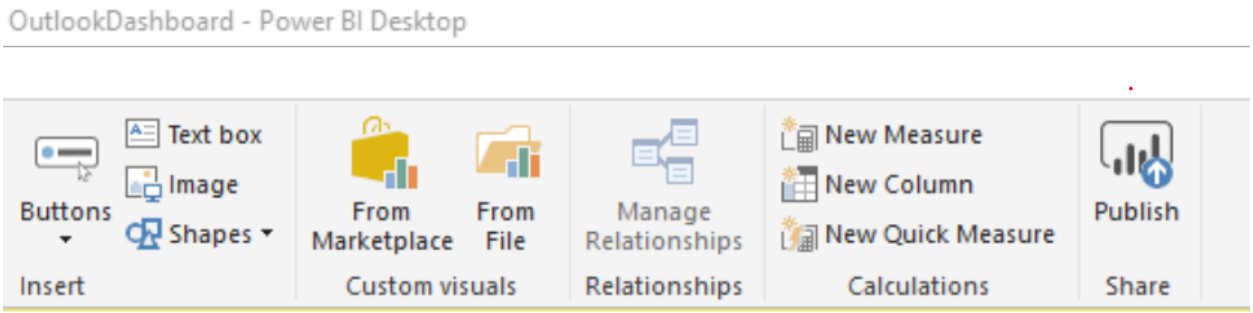
3. Create graphs in the Power BI Dashboard

You can create your own graphs/visuals in Power BI if you are familiar within otherwise, we have provided example templates for EXO, SPO, SFBO and Teams to help you with getting started. Dashboard templates (Dashboards-v2.zip) is available for download at Github



4.9 Publish a Power BI Dashboard

Once you're happy with your dashboard it's time to publish it. Simply click the Publish button on the top taskbar.



Once saved, you'll be presented a list of possible places to publish the dashboard. Select wherever you wish, and you should receive confirmation it's been published.

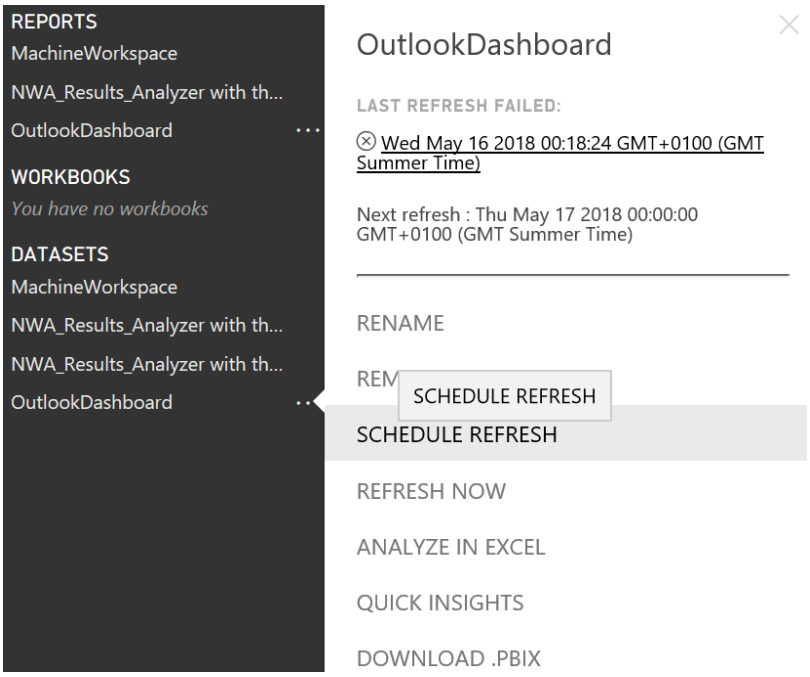
Once complete, login to https://powerbi.com via a browser and navigate to where you published the dashboard and you should see what you created in Power BI Desktop.

When logged in, to share the dashboard, simply click share in the top right-hand corner and enter the users or groups you wish to share it with.

4.10 AutoRefresh Power BI Dashboard

To refresh the dashboard with new data every day, we need to follow these steps. In the browser, when in your Power BI dashboard:

- Click on 'My Workspace' if this is where you published it.
- Next select the workspace under 'DATASETS'.
- Click the three dots to launch the menu then select "schedule refresh' from the list.



- Select 'Schedule Refresh' (again) from the next menu and enable and configure the feature to your requirements.

5 Summary

The approach in this document can be used for:

3. Measuring network bandwidth usage for pilot users on-boarded to Office 365 or network bandwidth usage of on-premises users.
4. Endpoint monitoring dashboards post on-boarding users to Office 365

You can apply this concept for measuring any SaaS/PaaS traffic, not just Office 365. Future improvements to this project can be tracked at Github