CHAPTER 4

Scenario-Based Examples: Cloud Only

In cloud first architectures, the monitoring perimeters are restricted to components in the cloud. The requirements for capacity, performance, and security monitoring can be met, to a large extent, using solutions available in Azure. The monitoring methodologies and parameters, however, could vary depending on the use cases or scenarios. Depending on whether the environment is greenfield or brownfield, the steps involved in designing the monitoring approach and implementation varies. Hence it is important to have a thorough planning exercise with applied due diligence before we set out to implement the identified solution.

This chapter will provide you a design and implementation reference guide for common cloud-only monitoring scenarios. We will also cover the configuration samples and templates for some of the use cases, including IaaS and PaaS components, which will help you with the implementations.

Design and Implantation Reference

Come of the common architectures for applications using cloud-only services in Azure can be broadly classified as shown in Table 4-1.

B. Chakraborty and S. A. Karthikeyan, *Understanding Azure Monitoring*,
https://doi.org/10.1007/978-1-4842-5130-0_4

Table 4-1. *Architectures in Our Scenario*

Architecture	Description
IaaS Only	These are "traditional" deployments in Azure, preferred by organizations adopting cloud for the first time. It is also used from on-premises environments that are migrated to cloud using a "Lift and Shift" approach. All application tiers would be front end, middle tier, and databases deployed and managed in VMs, providing customers more control over them.
PaaS Only	This approach is followed by "born in the cloud" organizations. It is also adopted when organizations can claim a certain maturity level in Azure. PaaS offerings, when used at different application tiers, enables customers to focus on the application rather than the nitty-gritty of infrastructure management.
IaaS and PaaS	This mix and match approach is commonly used when there is a need for balance between control and flexibility. IaaS services enable organizations to exercise end-to-end control on how they manage the services, while PaaS services offers the flexibility of outsourcing the management to the cloud platform where it deems fit.

Considering the popularity of microservices architecture, we should also add containers to the mix and the monitoring paradigm related to it. The choice of service for hosting containers could be Azure container instances, Kubernetes Services, and associated services like a Azure container registry. The monitoring strategy for a cloud-only environment is done over the following phases – Evaluation, Planning, and Implementation. Let us explore these phases in detail and activities to be covered in each of these phases.

For the ease of understanding the process, let us consider an environment that includes multiple PaaS and IaaS components to be monitored, for example, Virtual machines, webapps, App service

environments, AKS, Azure SQl DB, Azure storage, application gateways, Azure load balancers, NSG, etc. We will go through each of these phases, covering the evaluation, design, and implementation processes.

Evaluation

The entry point of designing the monitoring strategy is taking stock of the existing environment in case of Brownfield deployments or evaluating the future state in case of Greenfield deployments. You can use the following pointers to develop a questionnaire for evaluating the environment and to identify the monitoring parameters. Note that the questions are not exhaustive, and you might have to collect more information at the evaluation phase depending on the application architecture.

What are the Azure components needed for monitoring of the application?
For this use case, we have identified the following components – Virtual machines, webapps, App service environment, AKS, Azure SQl DB, Azure storage, application gateway, Azure load balancer, and NSG.

How does the interdependency of components impact monitoring?
Identify dependency of the components being monitored so that false alarms or duplications can be eliminated. For example, a health probe-related error of a load balancer could be a duplicate of a VM health error.

Is it relevant to monitor network connectivity between components or to external endpoints?
When there are many moving parts and dependencies in the application, network connectivity monitoring between components becomes relevant. Network watcher offers extensive network monitoring capabilities using different built-in tools. During the evaluation phase, it is recommended to

identify the tools in the network monitor that can be leveraged to meet the interconnectivity monitoring needs. For example, if your application needs access to an external endpoint, the connection monitor tool can be used to monitor the latency to this endpoint over time.

What components are relevant for monitoring from a business system perspective?

There are multiple metrics available for each of the identified components. Enabling the right metrics and alerts are important to avoid information overload. For example, it makes more sense to monitor metrics of App Service Plans rather than App Service Environment metrics, as App Service plans have a 1:1 mapping to hosts allocated to them.

How do we identity the right metrics for the components?

Now that we have identified the right components to be monitored, the next step is to drill down and finalize the right metrics for monitoring. For example, if you are running a web application in VM with DB hosted in another VM or Azure PaaS services (mySQL, SQL, etc.), it is not relevant to monitor the disk i/o metrics. However, it could be the most important metrics to be monitored in a VM that hosts a database in an IaaS model.

What alerts should be generated and on what is the severity to be assigned?

The goal of monitoring is to flag any anomalies in the system and alert settings are the crucial last mile in this configuration. The specific alert settings such as condition type, threshold, sensitivity, etc., should be considered to assign the right severity to an alert rule.

What are the relevant actions to be taken in response to the generated alerts?

Azure action groups can be used to inform customers about a potential issue through email, SMS, etc. It can also be used to trigger remedial actions through automation runbooks called via webhooks or by

invoking logic apps. It is in the evaluation phase that we identity the right course of action and identify the runbooks and logical apps to be created to execute it.

Planning

After the evaluation phase, the next step is to develop the implementation plan. This is not just a technical process but is closely tied to other organizational processes like change management as well. It is one level deeper than the evaluation process, where the parameters of monitoring are locked down and signed off by respective stakeholders. With respect to the different components in the use case, the monitoring configuration would be finalized during the planning phase. A sample outcome of this planning exercise is listed next.

Note: The planning outcome is dependent on the application architecture and the components to be monitored

Components	Monitoring Configuration
Virtual Machines	Performance metrics – CPU, Memory and Disk usage Disk I/O for SQL IaaS VM Guest OS Diagnostic data
App Service	Average response time Data in and Data out Http server errors CPU percentage Memory percentage
Azure SQl DB	Basic metrics Errors QueryStoreRuntimeStatistics QueryStoreWaitStatistics

(*continued*)

Components	Monitoring Configuration
Azure Storage	UsedCapacity ResponseType: AuthorizationError ResponseType: ServerBusyError ResponseType: ServerTimeoutError
Application Gateway	Metrics: Failed Requests Metrics: Throughput Metrics: Healthy Host Count Performance log monitoring Firewall log monitoring
Azure Loadbalancer	Data path availability Health probe status
NSG	NSG flow log in Network watcher
AKS	Health status Performance: Node CPU & Memory utilization Container performance monitoring through Azure Monitor for containers
Network Connection	Monitor connection to dependent component IP address through connection monitor of network watcher
Azure Traffic Manager	Queries by endpoint returned Endpoint status by EndPoint

Implementation

The outcome of the planning exercise is a detailed deployment plan for each of those identified monitoring components, which will be executed during the implementation phase. The following, additional pointers should be considered during the implementation phase for a smooth transition to operations.

How can the effectiveness of the implementation plan be tested before a rollout?

It is recommended to have a proof-of-concept period to test the settings and analyze the monitoring data for its effectiveness. This can be leveraged to fine-tune the configuration and lock down the final settings for implementation.

What roles and responsibilities should be assigned for managing the configurations?

For ongoing maintenance and updating of configurations, the right roles and responsibilities should be identified and assigned during the implementation phase. For example, while the ownership might largely remain with the IT team, application teams would need access to services like log analytics for troubleshooting purposes.

What role-based access control settings should be configured?

Once the roles and responsibilities are finalized, it would give a fair idea of the RBAC permissions to be configured. Any custom roles to be used should also be implemented during this phase

What services and configurations should be used for auto-remediation?

Options like runbooks and logic apps are available to be incorporated with alerts for auto-remediating issues. These runbooks and logic apps will be developed, tested, and deployed during the implementation phase.

What are the plans for operations training and handover?

The operations team should be given adequate training and documentation so that they can take over the monitoring of the components in the cloud. This effort required in the handover depends to a great extent on the expertise of the operations team in handling cloud monitoring.

In the next section we will cover the implementation details more closely for a few sample use cases.

Monitoring a Simple WebApplication on Azure

Let us consider the following use case of a three-tier web application with the sample architecture given in Figure 4-1.

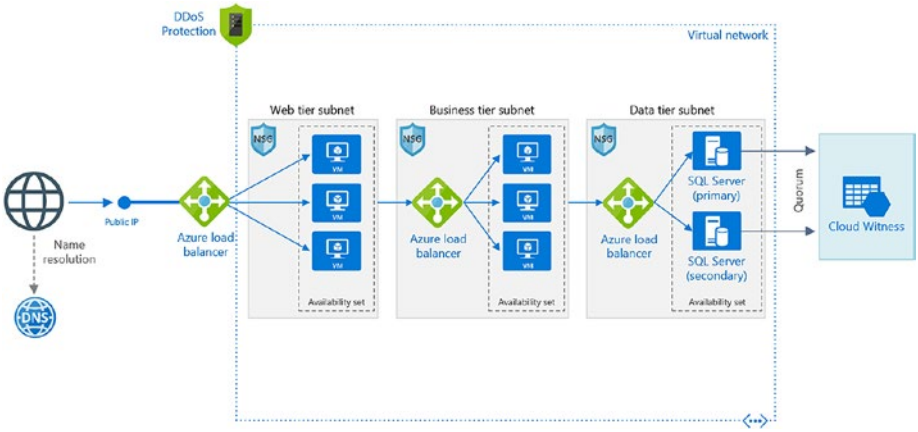

Figure 4-1. *Sample three-tier application*

As established in the planning section, monitoring will be configured for the following components.

Components	Monitoring Configuration
Virtual Machines	Performance metrics – CPU, Memory and Disk usage Disk I/O for SQL IaaS VM Guest OS Diagnostic data
Azure Loadbalancer	Data path availability Health probe status Load balancer health status

Components	Monitoring Configuration
NSG	NSG flow log in Network watcher
Network connection	Monitor connection to dependent component IP address through connection monitor of network watcher

Virtual Machines

The parameters to be monitored here are CPU, memory, and disk usage.

CPU Monitoring

CPU monitoring is available through the basic metrics of a VM without any additional configuration. This information can be accessed from VM settings ➤ Monitoring ➤ Metrics (Figure 4-2).

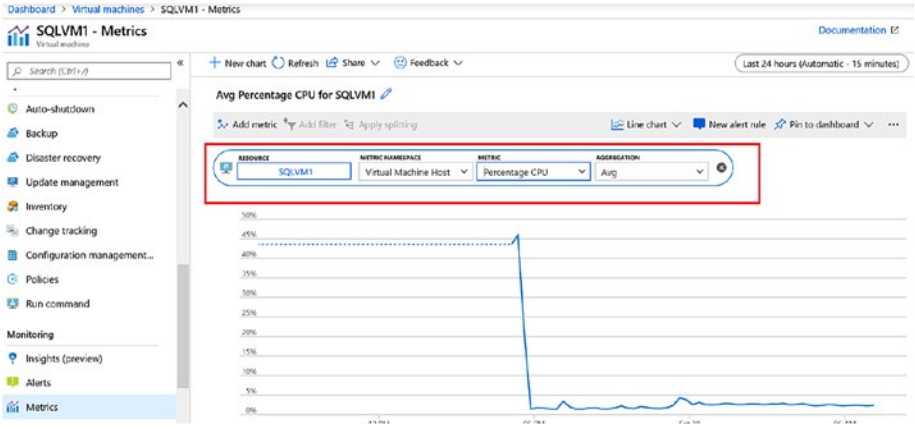

Figure 4-2. *VM metrics*

Memory Percentage

To get memory usage details, you will have to enable VM insights from VM Settings ➤ Monitoring ➤ Insights (preview). Provide the workspace subscription and name and click enabled (Figure 4-3).

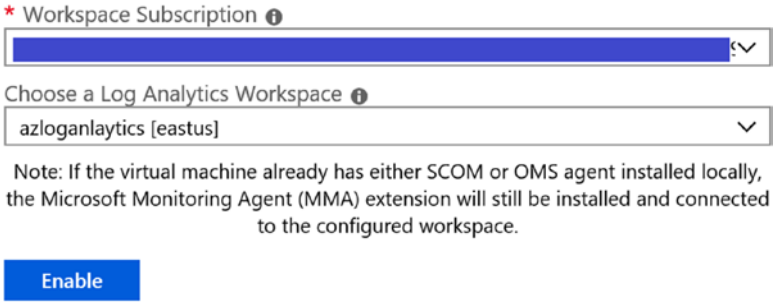

Figure 4-3. *Workspace details*

Once enabled, you see a notification that insights is being enabled. Note that it could take some time for the monitoring data to be collected and available (Figure 4-4).

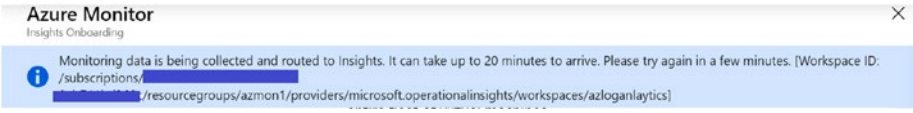

Figure 4-4. *Notification on enabling Azure Monitor*

Once enabled, a confirmation message will be displayed (Figure 4-5).

Figure 4-5. *Confirmation message*

From the VM insights dashboard, you can now view the Available memory usage and also pin the chart to the Dashboard (Figure 4-6).

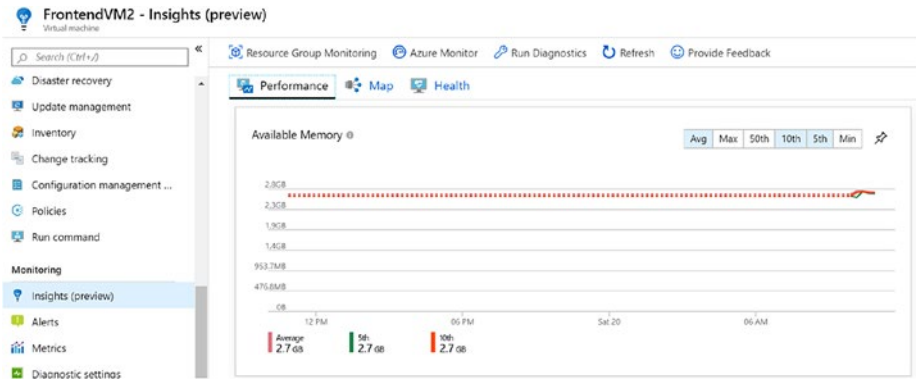

Figure 4-6. *Available memory*

You can also collect the memory usage metrics by enabling the guest OS diagnostics. To enable Guest-level monitoring, browse to VM ➤ Diagnostic settings ➤ "Enable guest-level monitoring" (Figure 4-7).

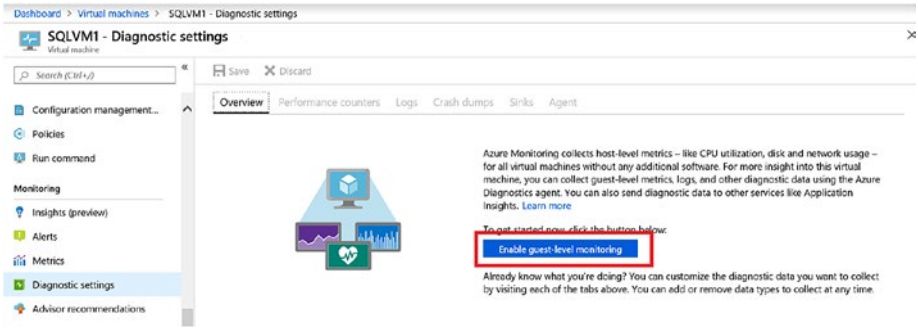

Figure 4-7. *Enable guest-level monitoring*

By default, the guest VM performance counters for CPU, Memory, Disk, and Network will be enabled (Figure 4-8).

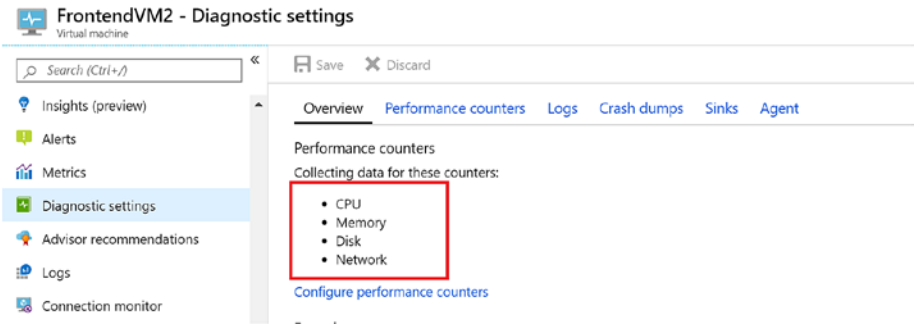

Figure 4-8. *Guest VM performance counters*

Now the memory metrics will be available from VM ➤ monitoring ➤ Metrics. Select the metrics namespace as Guest (Classic) and the "Available Bytes" memory counter (Figure 4-9).

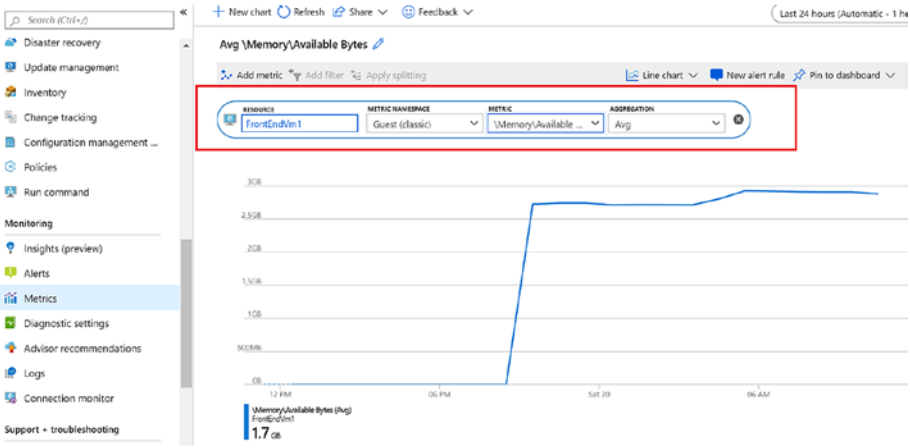

Figure 4-9. *VM memory metrics*

Disk Monitoring

To monitor free disk space in a VM, the "%Free space" counter can be used (Figure 4-10).

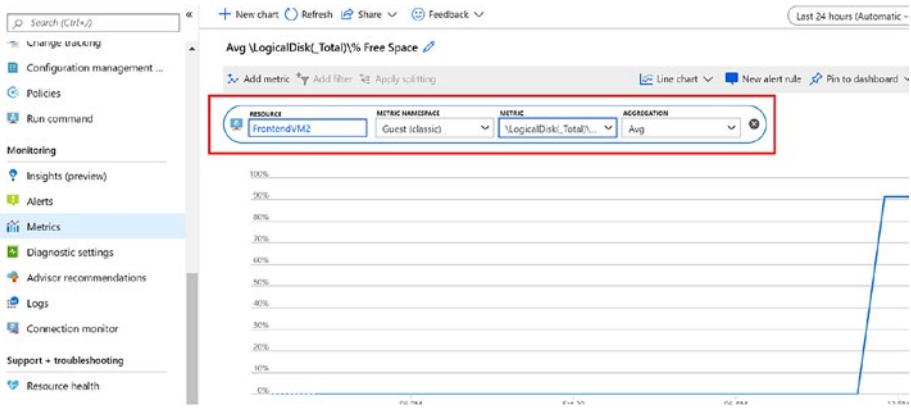

Figure 4-10. *Free disk space metrics*

You can fine-tune the settings if you click on "Configure performance counters" from monitoring ➤ Diagnostics settings of the VM and select only the required counters. For example, for SQL VM in the architecture, you need only Disk I/O-related counters. From the diagnostics settings, click on the performance counters tab. Click on the Custom switch and select the disk i/o related counters and click save (Figure 4-11).

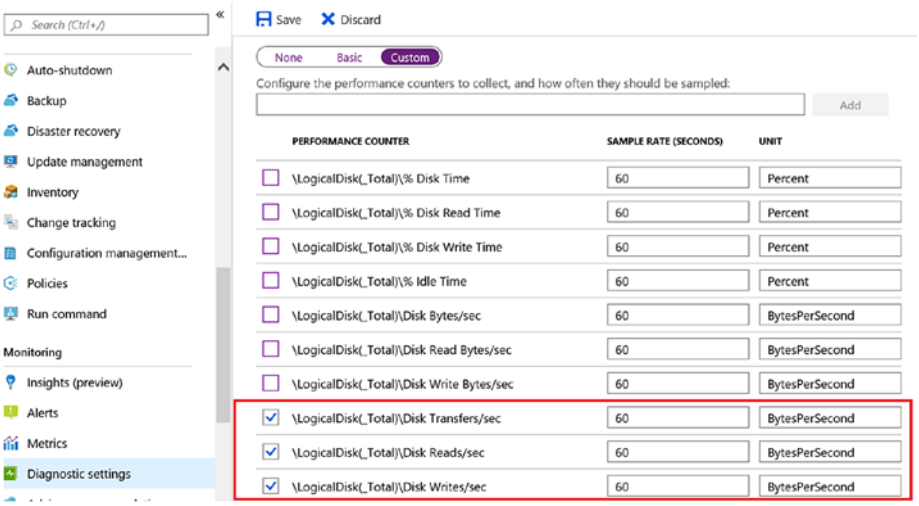

Figure 4-11. *Custom Switch configuration*

In the VM settings ➤ metrics, select Guest (classic) from the drop-down and select the disk counter (Figure 4-12). Now you can create the alert based on the threshold to be set for Disk I/O operation.

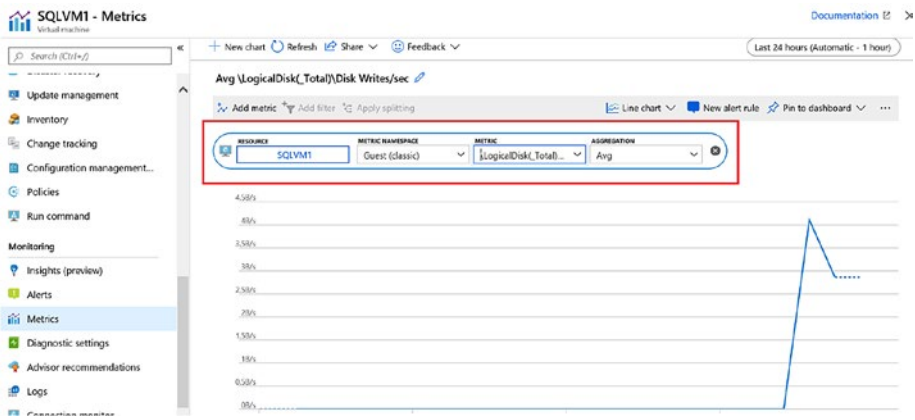

Figure 4-12. *Disk counter Metrics*

Azure Load Balancer

Monitoring configurations are slightly different for basic and standard load balancers. For basic load balancers, the settings are to be enabled from load balancer ➤ Configure monitoring ➤ Diagnostic settings ➤ Add Diagnostic settings.

In the Diagnostic settings, select the storage account, Event Hub, or log analytics workspace where the diagnostics data should be forwarded to. Since data from all components are being sent to log analytics, the same workspace is selected in this example. The diagnostics data for the load balancer will now be available in Log Analytics for review (Figure 4-13).

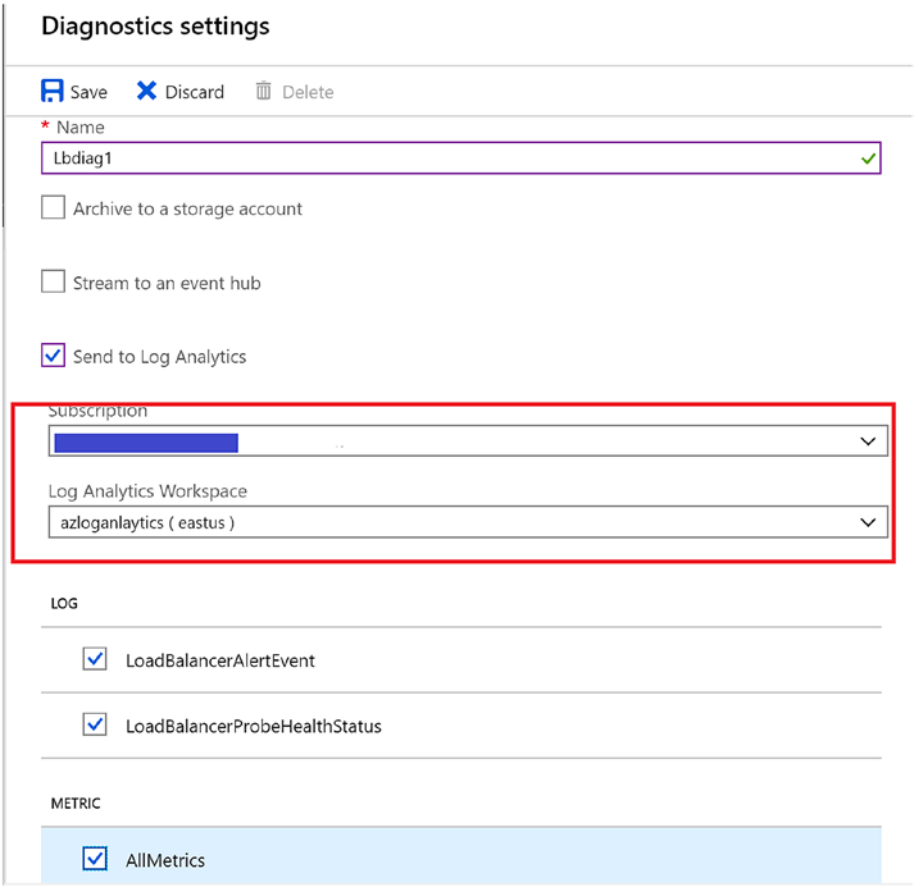

Figure 4-13. *Load balancer diagnostics configuration*

A standard load balancer is the recommended one for all new deployments as it supports features like availability zones, HA ports, HTTPS health probes, as well as the flexibility to add a combination of virtual machines, availability sets, and VMSS in the back-end pools. With respect to diagnostics, standard load balancers support multidimensional metrics for health probe status, byte, and packet counters, outbound connection health, etc.

Data Path Availability

This metrics provides visibility on whether the services are available externally, by testing the datapath from within an Azure region to the back-end Azure virtual machine. It helps to identify Azure infrastructure-related issues, if any. A traffic matching the front-end rule is automatically generated by the platform to create the metrics. The failure could occur if there are no healthy VMs in the back end or if an infrastructure outage has occurred. The recommended aggregate to be used for these metrics is "Average" (Figure 4-14).

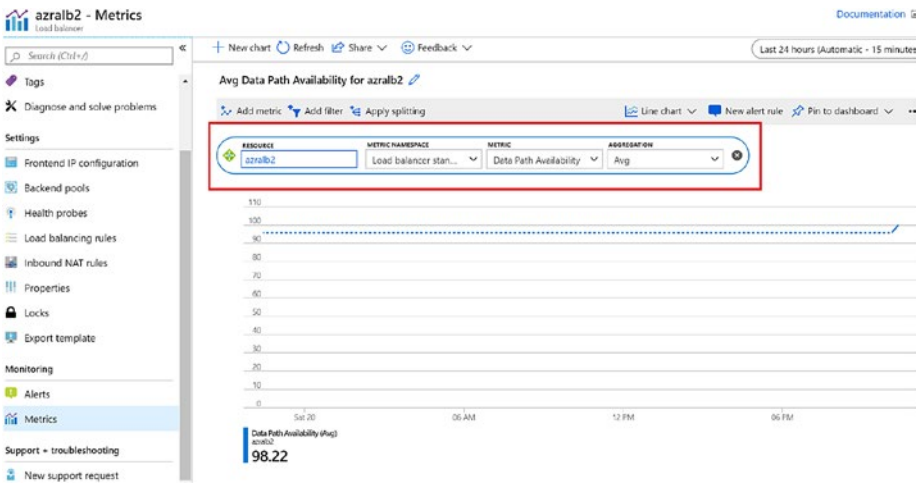

Figure 4-14. *Data path availability Metrics*

You can also select apply Splitting and choose the Frontend IP address or Frontend port as an additional dimension for monitoring (Figure 4-15)

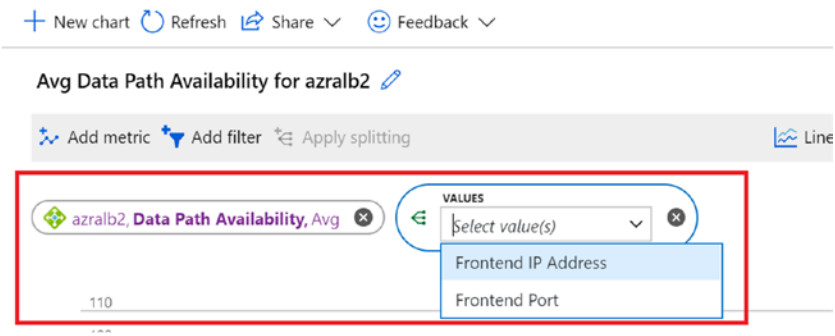

Figure 4-15. *Data path availability Metrics Values*

Health Probe Status

This metrics gives an overview of the health probe of the application, based on the instance endpoints configured. This metrics will show downtime if the service is unavailable or if there is any configuration like NSG or firewall blocking access to the endpoints (Figure 4-16).

Figure 4-16. *Health probe status Metrics*

Click on "Apply splitting" and select "Backend IP Address" from the listed options. This is optimal as all other values would be the same for all the endpoints in the back-end pool (Figure 4-17).

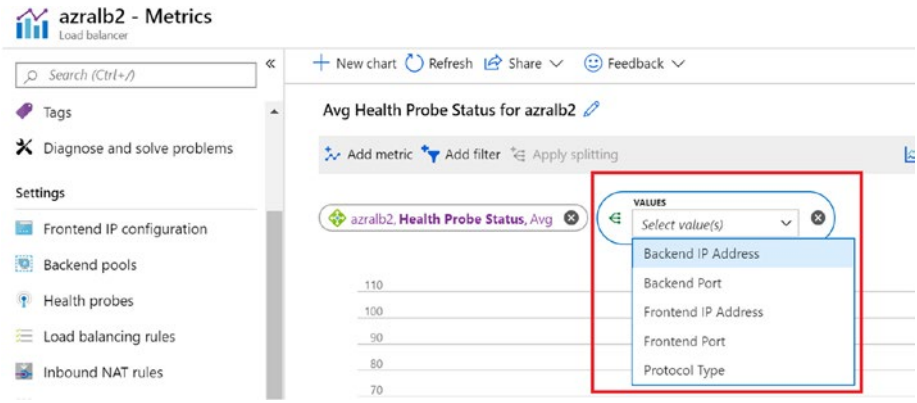

Figure 4-17. *Health probe status Metrics values*

As can be seen in the next example, one of the endpoints was down, while the other had higher availability (Figure 4-18).

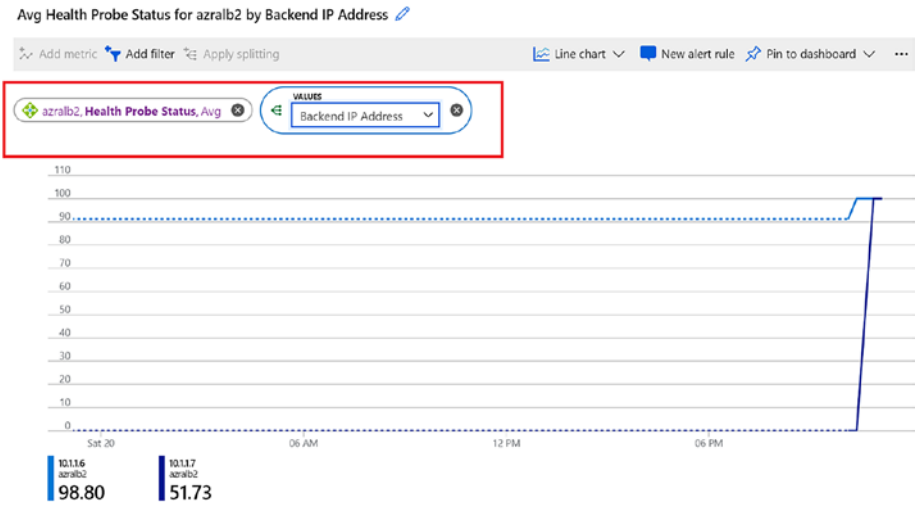

Figure 4-18. *Backend IP address status*

NSG

For monitoring NSG flow, Azure Network watcher should be enabled for the region. Search for network watcher from all services in the Azure portal. In the overview tab, expand regions and make sure that service is enabled in the target region where the NSG is created. The next step is to enable the NSG flow logs.

From Network watcher ➤ Logs ➤ NSG flow logs, filter down to the NSGs where the flow logs need to be enabled. Click on the NSG to open the flow log settings.

Select from between Version 1 and Version 2. Version 1 provides ingress and egress traffic information for packets that have been allowed and denied. Version 2 provides additional information on the flow state starting from when the flow is initiated to continuation and termination of the flow along with the traffic bandwidth details.

Also, select the storage accounts to which the flow logs will be stored. The raw data can be downloaded from the storage account, or it can be analyzed using a traffic analytics solution (Figure 4-19).

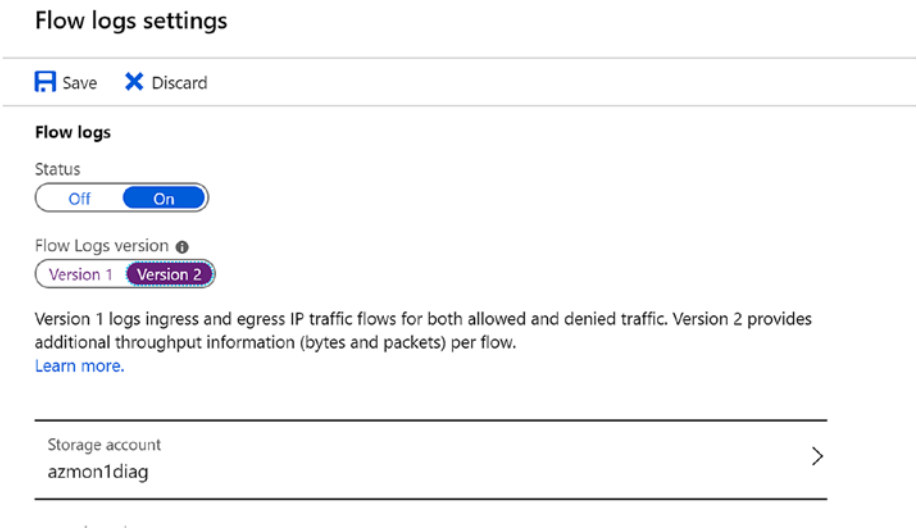

Figure 4-19. *Flow log settings*

In the same window you can configure the number of days for which the logs will be stored. You can also enable Traffic analytics to visualize the network activity by leveraging the NSG flow logs. It also helps to identify the traffic flow patterns within Azure and also from the internet, giving valuable insights to optimize performance and capacity.

Note that this configuration needs a log analytics workspace where a network performance monitor solution will be installed. The processing interval can be configured to be either 10 minutes or 1 hour (Figure 4-20).

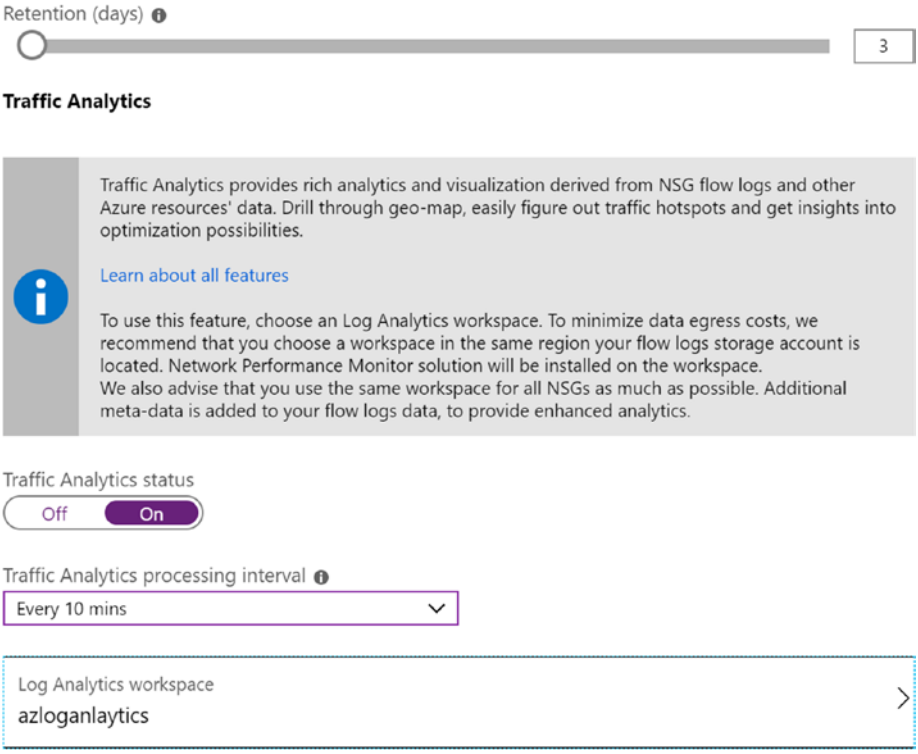

Figure 4-20. *Traffic Analytics processing interval*

After the configured processing interval, Open Network Watcher ➤ Logs ➤ Traffic Analytics to view the NSG flow information (Figure 4-21).

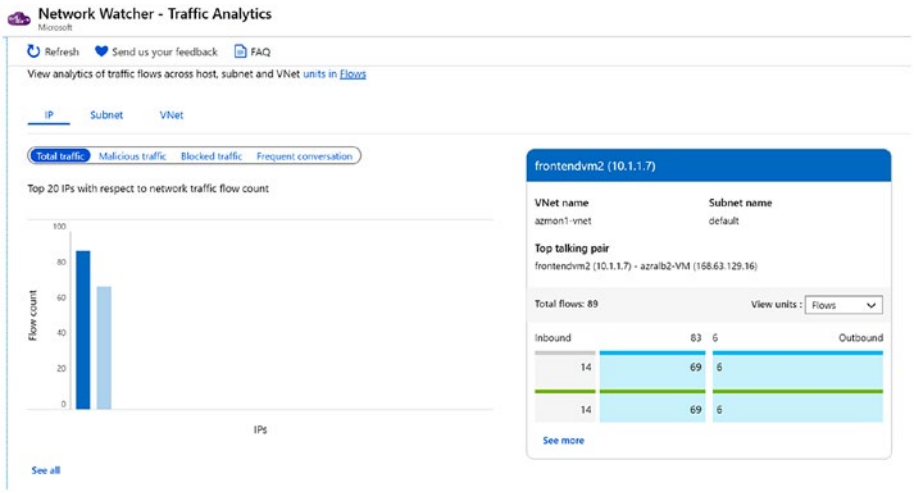

Figure 4-21. *NSG Flow information*

You can drill down to additional details such as Malicious traffic, Blocked traffic, or Frequent conversation from the collected NSG flow information at IP/Subnet/VNet level.

Network Connection

The connection watcher service of network watcher enables connectivity monitoring from a virtual machine to another VM, FQDN, URI, or IP address. This is helpful in monitoring dependent application components and to identify if the network traffic is getting blocked. The connection monitor is also capable of providing potential reasons for a connectivity issue such as DNS resolution issue, custom route-related issue, VM security rules, etc. The data is available over a period of time in terms of minimum, maximum, and average latency observed between the VM and the endpoint.

To monitor a network connection using a connection watcher, browse to network watcher ➤ Monitoring ➤ Connection monitor and Click on +Add.

Give a name to the connection monitor, select the source subscription, source virtual machine, destination virtual machine, and port (Figure 4-22).

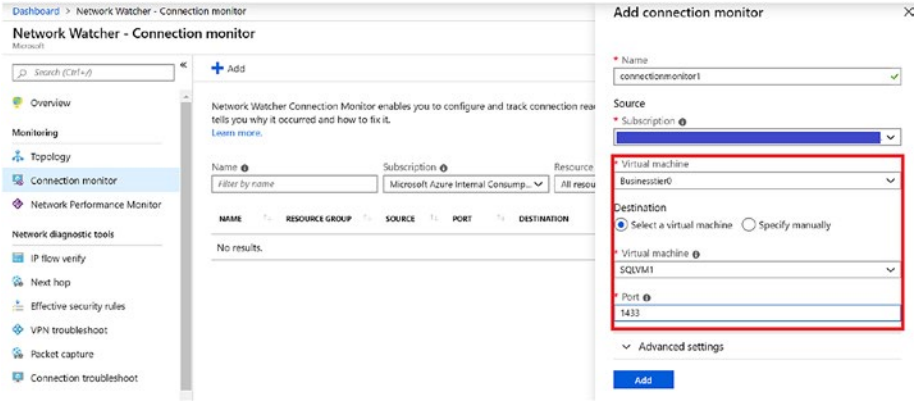

Figure 4-22. *Connection monitor configuration*

If you click on the connection monitor, the details pane will be displayed on the bottom pane. You can click to view the graph in a new window (Figure 4-23).

Figure 4-23. *Connection monitor graph*

You can also see the grid view and topology view from the connection monitor (Figure 4-24).

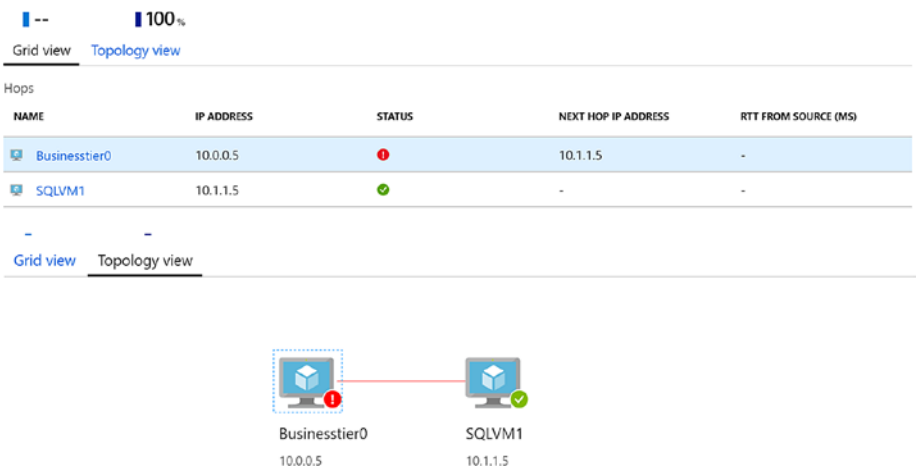

Figure 4-24. *Connection monitor grid view*

Monitoring Application That Includes PaaS and IaaS Services

Let us consider the sample architecture of a two-tier web application that uses webapp in the front end, connecting to SQL VMs in the back end (Figure 4-25).

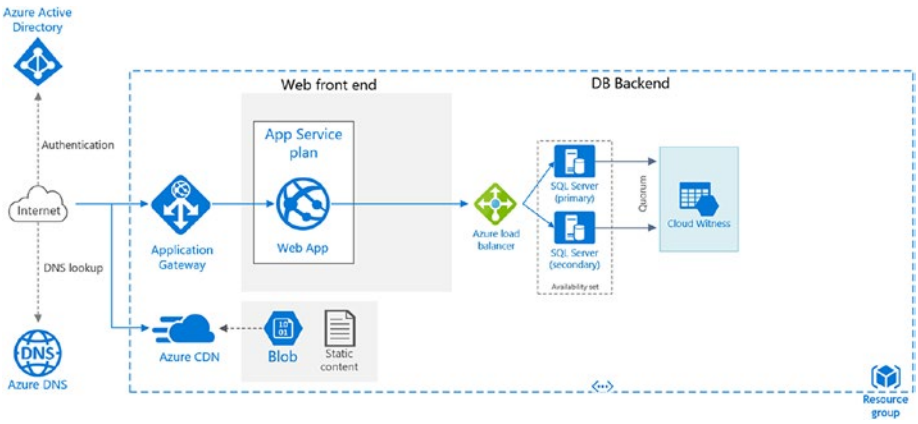

Figure 4-25. *Sample two-tier application*

As established in the planning section, monitoring will be configured for the following components.

Components	Monitoring Configuration
App service environment	Average response time Data in and Data out Http server errors CPU percentage Memory percentage
Azure Storage	UsedCapacity ResponseType: AuthorizationError ResponseType: ServerBusyError ResponseType: ServerTimeoutError
Application Gateway	Metrics: Failed Requests Metrics: Throughput Metrics: Healthy Host Count Performance log monitoring Firewall log monitoring

Azure Storage

UsedCapacity: Azure storage capacity metrics can be leveraged here to monitor the used capacity. While using standard storage, the value is the sum of capacity consumed by all tables, blobs, and queues in the storage. For premium storage and blob storage, values will be equivalent to the capacity used by all blobs in the storage.

In this example, we are using standard storage. From the Storage setting ➤ Monitoring ➤ Metrics, select metrics "Used Capacity" listed in the drop-down under Capacity (Figure 4-26).

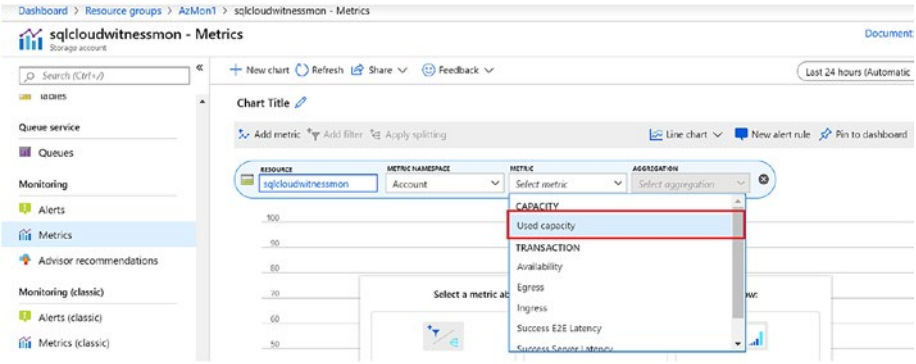

Figure 4-26. *Storage metrics selection*

ResponseType:"Authorization error" helps to track any unauthorized access of data in the storage account and is important to monitor from a data security perspective. To monitor the ResponseType, select the metrics type as Transactions (Figure 4-27).

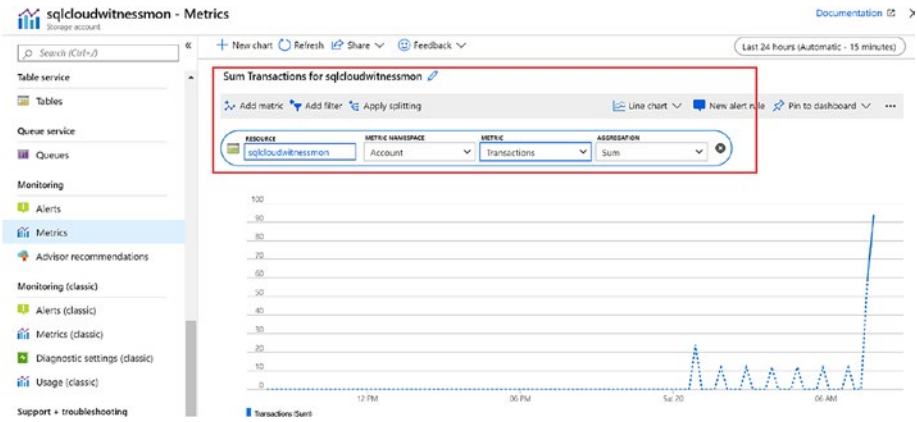

Figure 4-27. *Storage Transactions metrics*

Click on "Add Filter" (Figure 4-28).

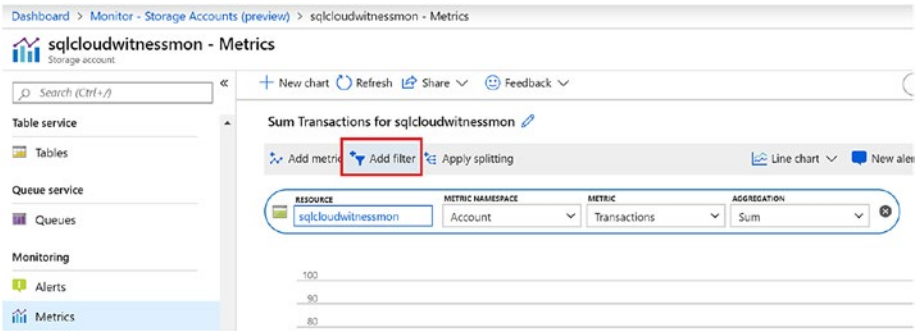

Figure 4-28. *Storage Transactions metrics filter*

Select Property as ResponseType and value as "AuthorizationError" (Figure 4-29).

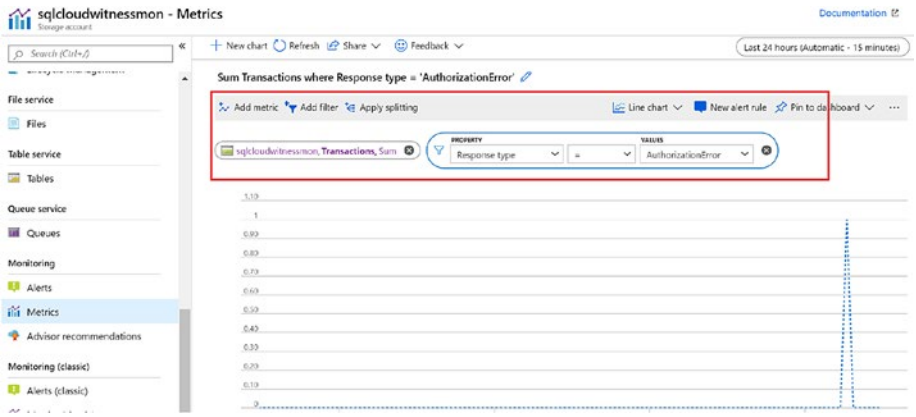

Figure 4-29. *"AuthorizationError" filter*

App Service

The identified metrics for App Service are the following:

- Average response time: It is the time taken in milliseconds by the app to respond to serve requests.
- Data in and Data out: Monitors the data in and out of the bandwidth consumed by the application in MiB.
- Http server errors: Count of requests that result in an Http status code greater than or equal to 500 but less than 600.
- CPU percentage: This metric will be used at the app service plan level to monitor the CPU usage across all instances of the plan.
- Memory percentage: This metric is also used for the app service plan, to monitor the memory usage across all instances.

The first four metrics, that is, the Average response time, Data in, Data out, and Http server errors can be configured from the app service ➤ monitoring ➤ metrics (Figure 4-30).

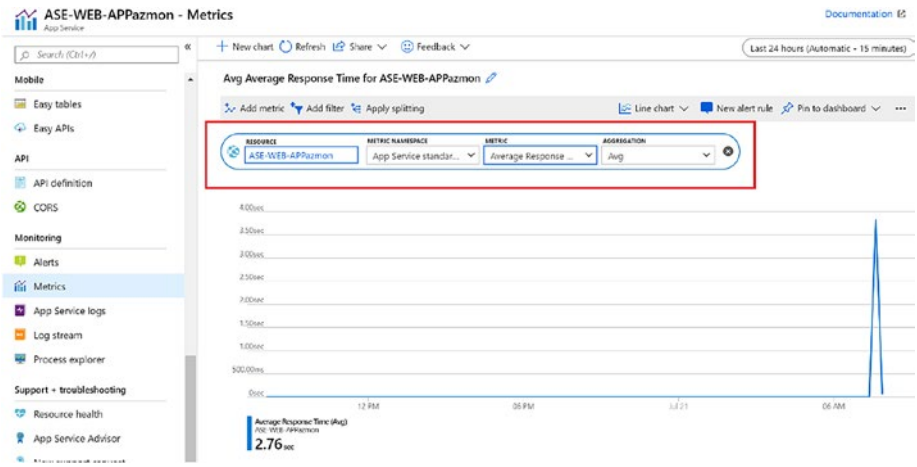

Figure 4-30. *App Service metrics*

For CPU and memory percentage monitoring for all instances, browse to app service plan ➤ Monitoring ➤ Metrics and select the metrics (Figure 4-31).

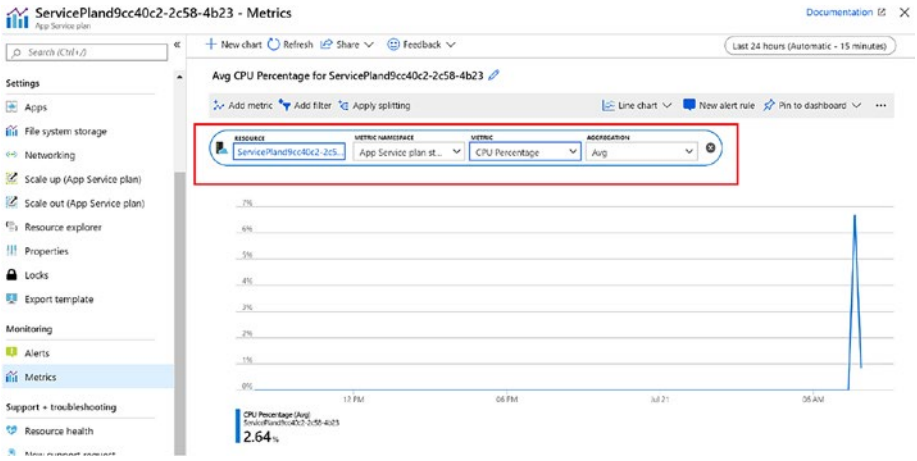

Figure 4-31. *CPU Percentage metrics*

Application Gateway

Both metrics and log files should be monitored for an application gateway. The metrics can be monitored the same way as in other components, that is, from Application gateway ➤ Monitoring ➤ Metrics (Figure 4-32).

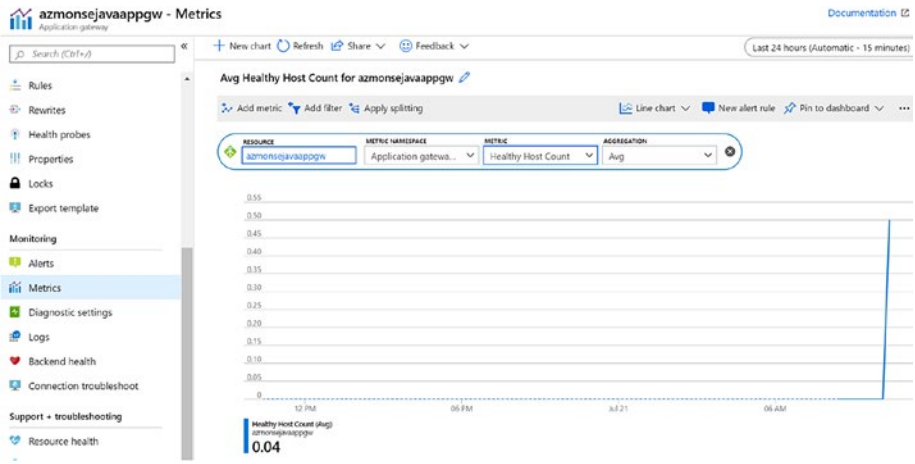

Figure 4-32. *Healthy host count metrics*

The healthy host count and unhealthy host count information can be further drilled down to a specific back-end pool. This is useful when the application gateway is used for multiple applications and you want to monitor the status of one specific application back-end pool (Figure 4-33).

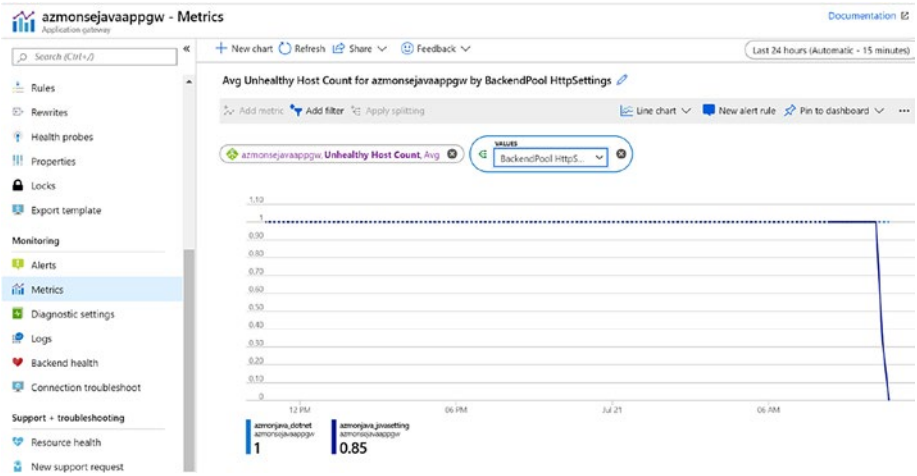

Figure 4-33. *Unhealthy host count metrics*

To enable performance and firewall log monitoring, configure the diagnostics settings from Application gateway ➤ Monitoring ➤ diagnostic setting ➤ Add diagnostic setting (Figure 4-34).

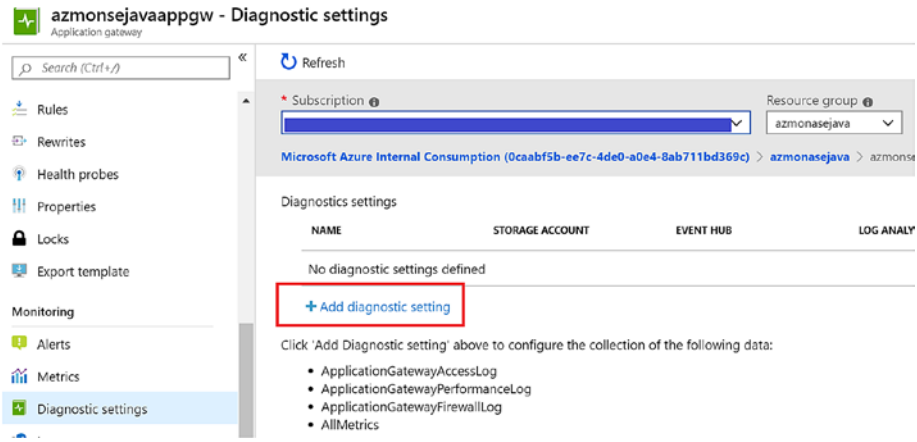

Figure 4-34. *Add diagnostic settings*

Configure the diagnostics information to be sent to a storage account, stream it to Event Hub, or send to a log analytics workspace. In this example, we are sending the Performance log and Firewall log data to the log analytics workspace (Figure 4-35).

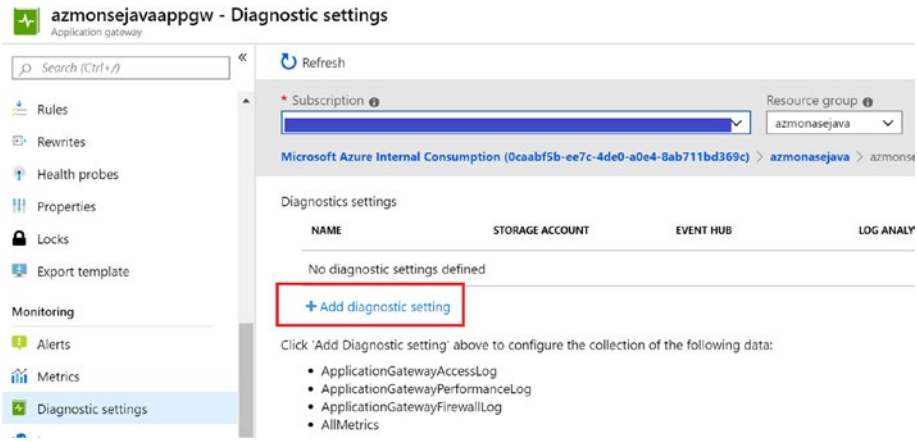

Figure 4-35. *Application gateway diagnostic settings*

Application Insights

Application Insights should be included in the architecture for monitoring of your app service. It can be done using the Application Insights Agent, which is from the Azure portal and ensures a minimum level of monitoring.

Browse to the application ➤ settings ➤ Application Insights and click on "Turn on Application Insights" (Figure 4-36).

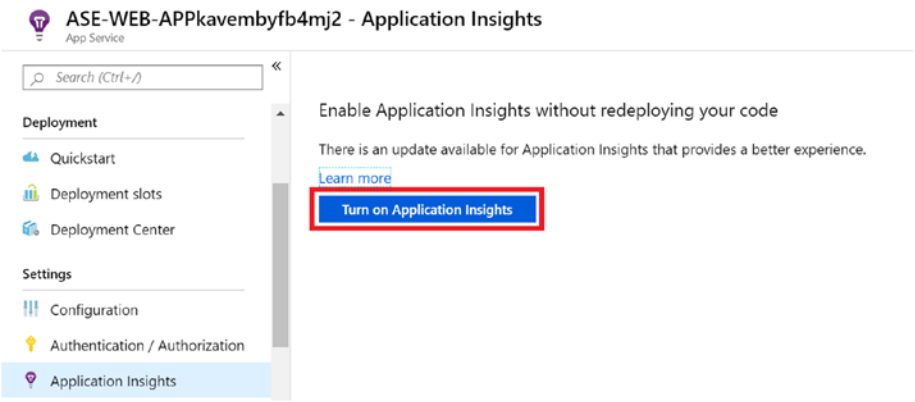

Figure 4-36. *Enable Application Insights*

Link to an existing application Insights or create a new Application Insights resource (Figure 4-37).

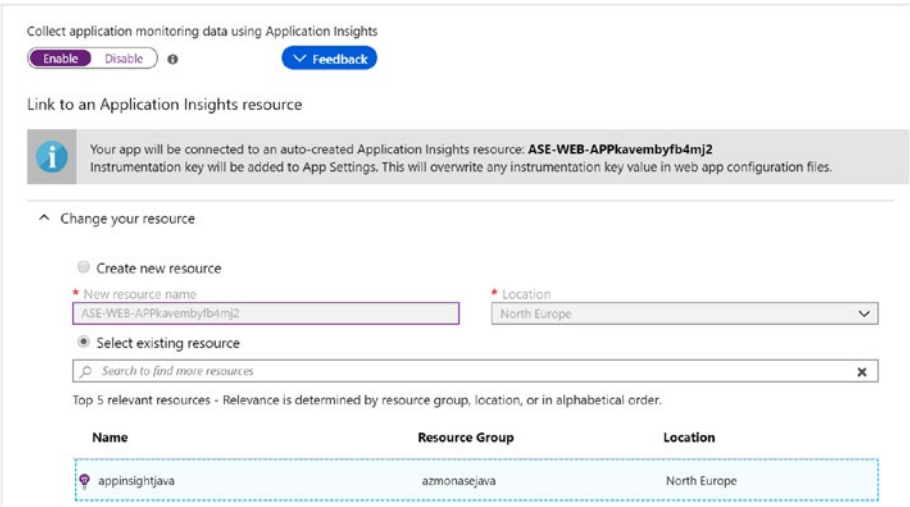

Figure 4-37. *Select/Create Application Insights*

Additional settings such as configuration of the collection level, Profiler, SnapShot debugger, etc., can also be configured from this window (Figure 4-38).

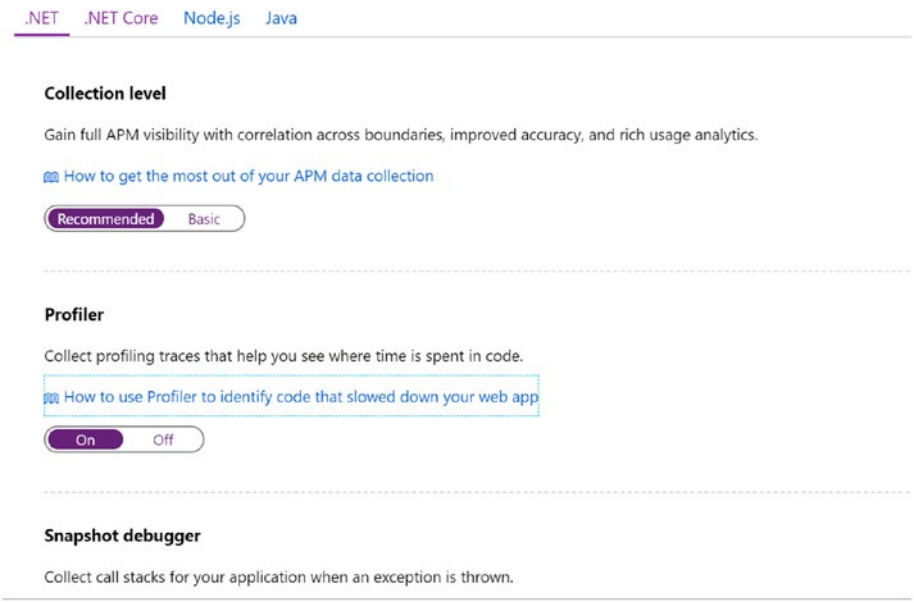

Figure 4-38. *Application Insights additional settings configuration*

Click apply to complete the configuration of Application Insights.

You can also manually instrument the code by using Application insights SDK from an IDE like Visual Studio. It is recommended if additional customizations are required where you want to monitor events or dependencies using custom API calls.

Live Metrics from Application Insights is another useful feature that helps you monitor the metrics and performance counters of applications in real time. It is a noninvasive method for live monitoring of your application. Live Metrics needs the latest version of Application Insights SDK and the Microsoft.ApplicationInsights.PerfCounterCollector package to be installed in your webapp. Unlike other metrics that are aggregated over minutes, Live Metrics data (Figure 4-39) is displayed within the duration of 1 second. It is an on-demand monitoring mechanism where the live streaming starts when you open the tab. It does not persist data in storage or log analytics and it is free of charge.

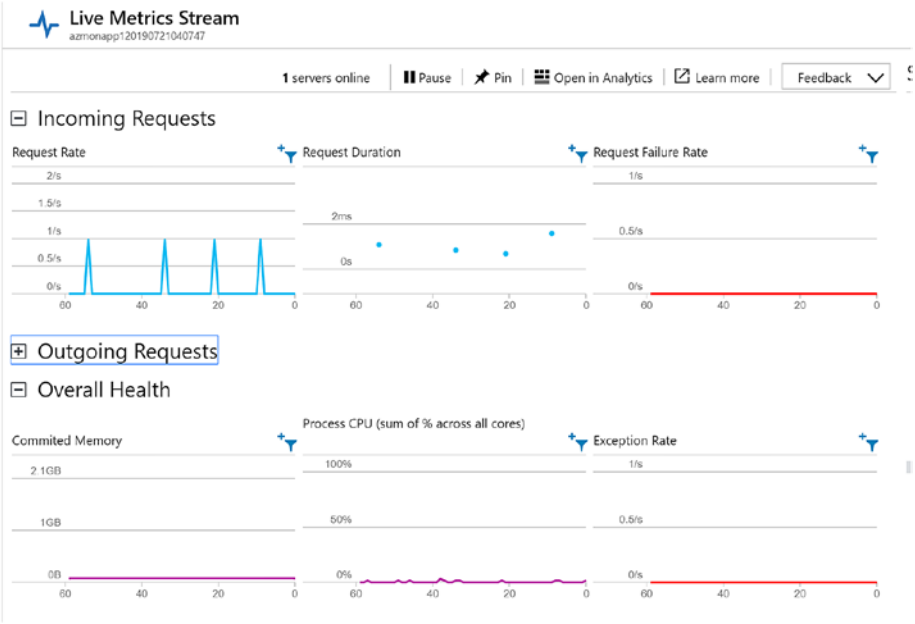

Figure 4-39. *Live Metrics Stream*

Monitoring Multi-Region WebApplication

Let us consider the sample architecture of a multi-region web application that uses Traffic manager for DNS-based load balancing (Figure 4-40).

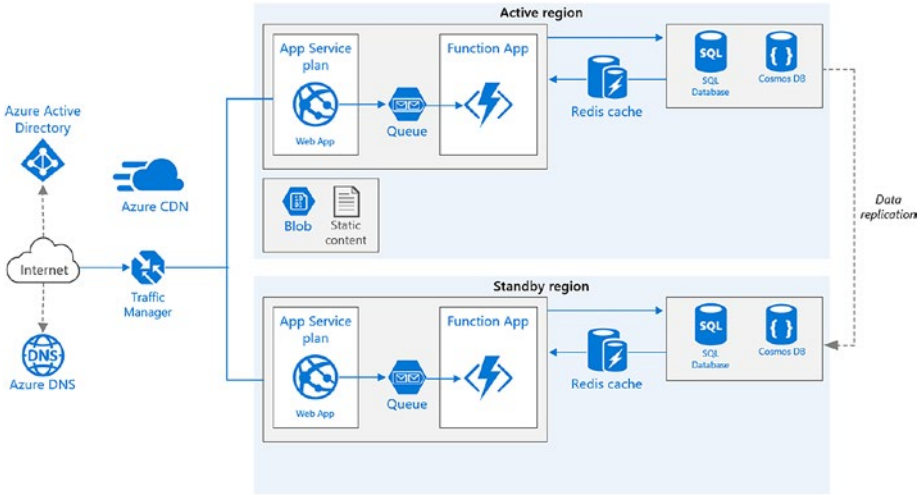

Figure 4-40. *Multi-region web application sample architecture*

As established in the planning section, monitoring will be configured for the following components. Other components like app service plan, storage, etc., are have already been covered in the previous section.

Components	Monitoring Configuration
Azure SQl DB	Basic metrics Errors QueryStoreRuntimeStatistics QueryStoreWaitStatistics
Azure Traffic Manager	Queries by endpoint returned Endpoint status by EndPoint

Azure SQl DB

The basic metrics for SQl DB are available from SQl DB ➤ Performance ➤ metrics (Figure 4-41).

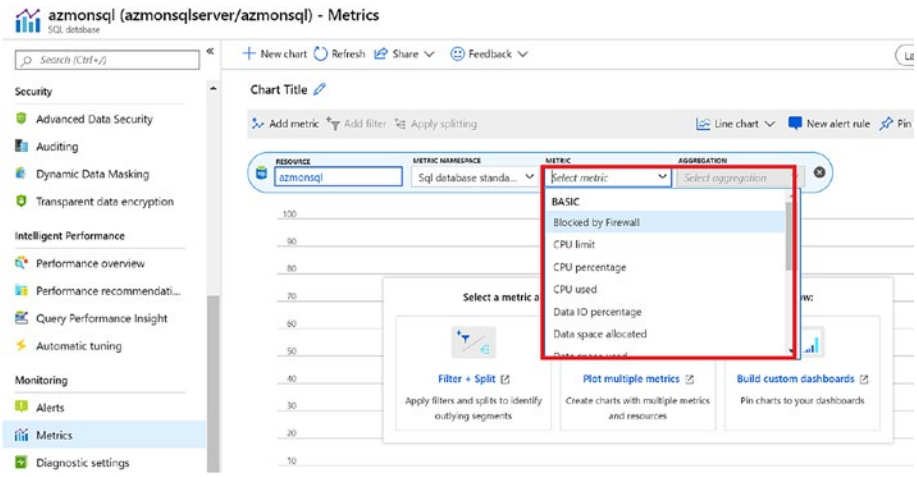

Figure 4-41. *Azure SQL DB Metrics*

We will also configure the SQL performance logs to be sent to the Log analytics workspace for centralized monitoring. From the SQL database settings ➤ Monitoring ➤ Diagnostic settings, add the Diagnostics settings and select the identified logs (Figure 4-42).

- Basic metrics: These metrics contain information like DTU/CPU percentage, limit, data read and log write percentage, firewall connections, etc.
- QueryStoreRuntimeStatistics: Information on CPU usage and query duration statistics during query runtime.
- QueryStoreWaitStatistics:Gives insights on what aspects the queries are being waited on, that is, CPU, Locking, log, etc.
- Errors: These metrics give insights on SQL errors that occur in the database.

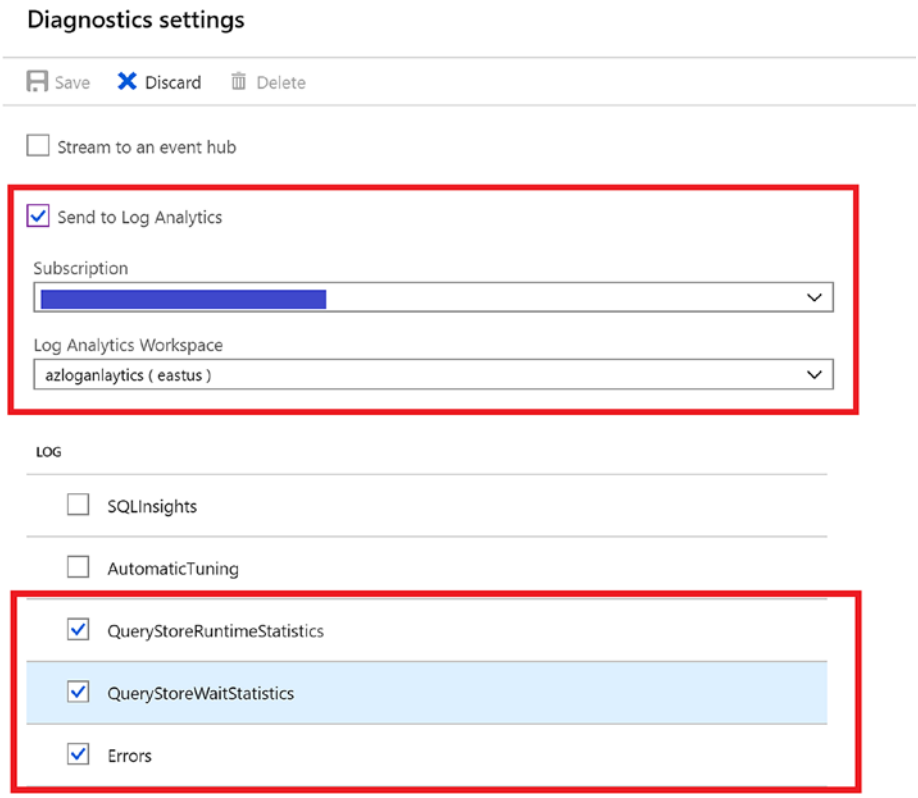

Figure 4-42. *Select Logs*

To view the data and gain additional insights, you can add the solution named Azure SQL Analytics (Preview) in Azure Log analytics (Figure 4-43).

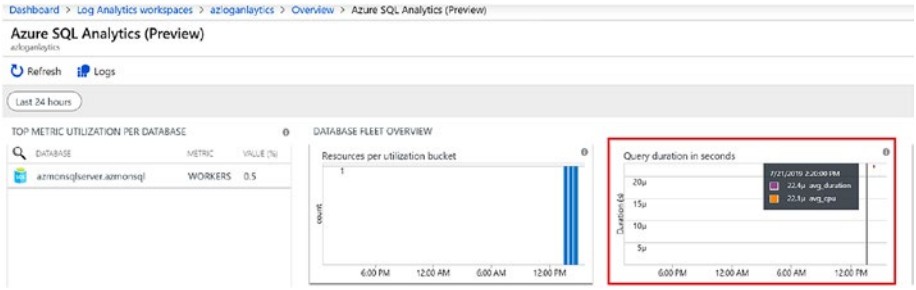

Figure 4-43. *Azure SQL Analytics*

Traffic Manager

Queries by Endpoint Returned

These metrics can be used to view the number of requests received by a traffic manager over a period of time. This information can also be split across endpoints (Figure 4-44).

Figure 4-44. *Queries by endpoint-returned metrics*

Endpoint Status by Endpoint

This metric is used to show the status of endpoints of a traffic manager. It has two values: 1 if the endpoint is up, and 0 if the endpoint is down (Figure 4-45).

Figure 4-45. *Endpoint status filtered by endpoint Name*

Monitoring of Container-Based Microservices

Let us consider a typical AKS deployment architecture as shown next for this use case (Figure 4-46).

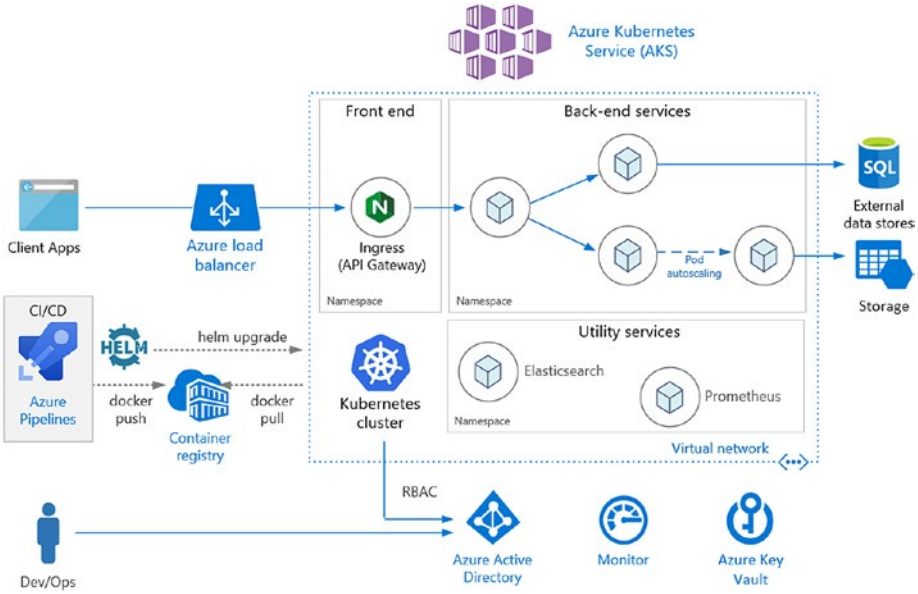

Figure 4-46. *AKS Architecture*

As established in the planning section, monitoring will be configured for the following components.

Components	Monitoring Configuration
AKS	Health status Performance: Node CPU and Memory utilization Container performance monitoring

Azure offers container monitoring through Azure Monitor for container solutions. It collects performance and memory metrics from nodes as well as containers deployed in those clusters. This information is collated and available for review in log analytics. Container monitoring information, which includes inventory, performance, logs, and events – along with

AKS control plane data – is also captured and made available in the Log Analytics workspace. The following information is available from Azure Monitor for containers:

Average processor and memory utilization of containers deployed in AKS clusters;

Identify the location of container and pod to get a better understanding of the container/pod performance;

Monitor cluster behavior under different loads, which helps in capacity planning and future expansion of the cluster;

Configure thresholds and alerts to notify administrators about potential resource bottlenecks. The functionality of Azure Monitor for containers is depicted in the next diagram (Figure 4-47).

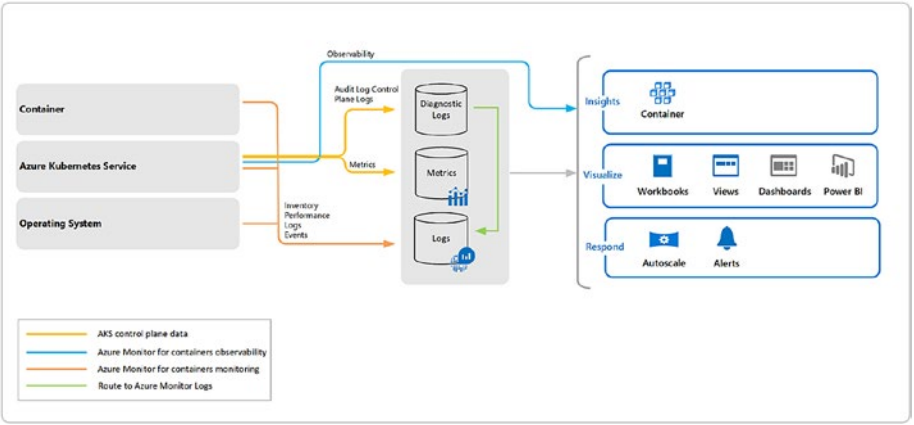

Figure 4-47. *Azure Monitor for containers*

To enable Azure Monitor for containers in your AKS cluster, browse to AKS cluster ➤ Monitoring ➤ Insights. Select the log analytics cluster to which you want to link the cluster and click enable (Figure 4-48).

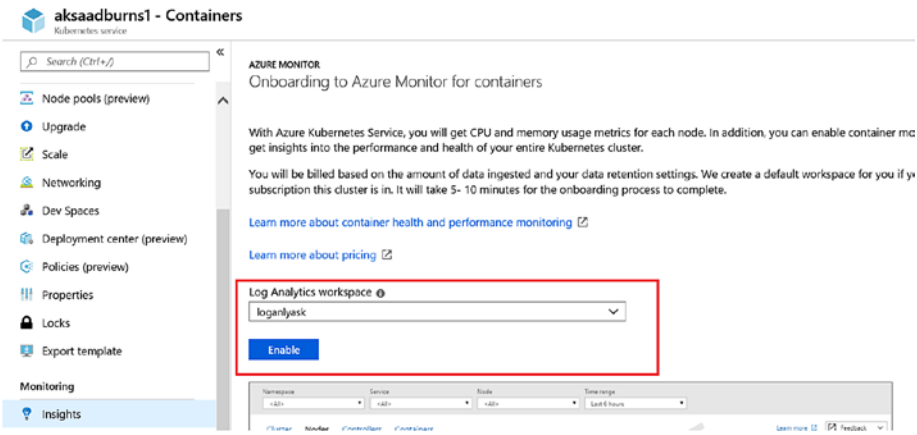

Figure 4-48. *Select Log Analytics workspace*

Once enabled, you can see a segregated view of information from Cluster, Nodes, Controllers, and containers in the cluster (Figure 4-49).

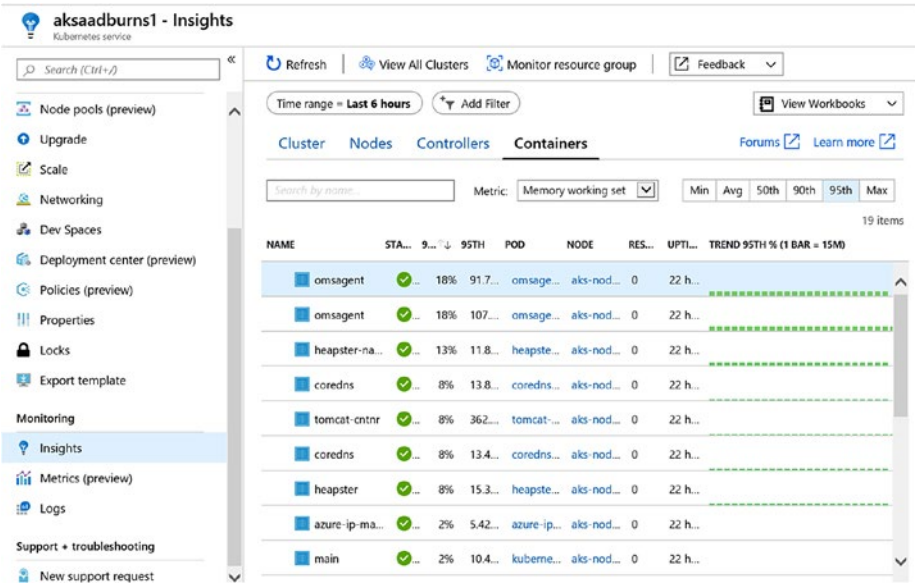

Figure 4-49. *Azure Monitor for containers sample output*

If you want to do a deeper analysis using container logs, browse to the log analytics workspace that the AKS cluster was linked to. Browse to general ➤ Logs and view the logs from "ContainerLog" (Figure 4-50).

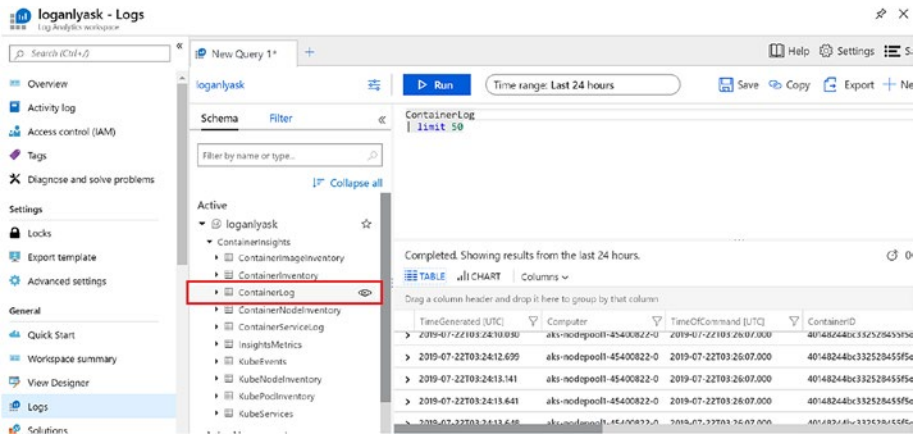

Figure 4-50. *View Container log*

Additional metrics information is available by default from Monitoring ➤ Metrics (preview) (Figure 4-51).

Figure 4-51. *Additional container metrics*

You can view the following information from the Figure 4-51: Number of pods by phase, Number of pods in the ready state, Total amount of memory available in a managed cluster, and Total number of available CPU cores in a managed cluster.

You can even drill down to further details by applying dimensions. For example, for the metrics "Number of pods in ready state," you can drill down to the number of pods in a specific namespace (Figure 4-52).

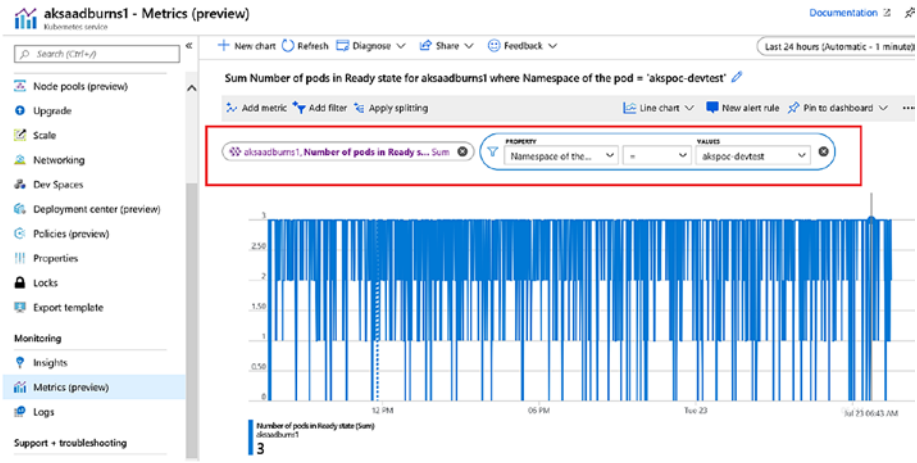

Figure 4-52. *Number of pods in ready state*

Action Groups and Alerts

Configuration of metrics and logs is useful only if you are able to generate alerts in case of any anomalies. Action groups are used to define the notification mechanism for alerts. Action groups consist of the following:

- **Name:** Unique name for the action group;
- **Action type:** Action group can be configured to notify stakeholders through emails, SMS, push notifications, or voice messages. It can also call a logic app, Function, Webhook, Automation runbook, or generate an ITSM ticket;

- **Details:** These are configurations specific to the Action type selected.

Action groups will be called during an alert configuration, thereby triggering a notification when an alert condition is met.

To configure Action groups, open Azure Monitor ➤ Alerts ➤ Manage actions ➤ Add action group.

Provide information such as Action group name, Short name to be included in notifications, and the subscription and resource group where the action group will be created. In this, for example, we will trigger an automation runbook using the action group, and hence "Automation Runbook" is selected as the Action type. Click on Edit details to configure the runbook to be used (Figure 4-53).

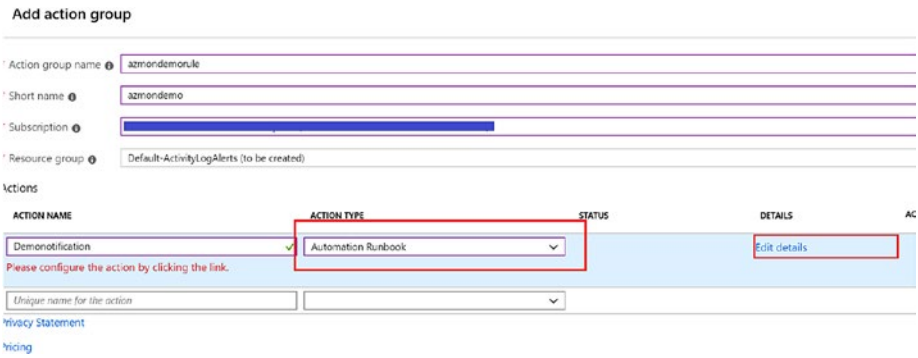

Figure 4-53. *Select Action Type*

In the Configure runbook tab, select the Runbook source. It can be built in runbook or runbooks created by users. There are built-in runbooks available to perform the following activities: Stop, Restart, Scale up, Scale down, and Remove VM. In this example, we have selected the Scale up VM option. Select a subscription and an automation account in the subscription. All other configurations can be default. Click on Ok. It will take you back to the action group configuration window. Click Ok there to create the action group (Figure 4-54).

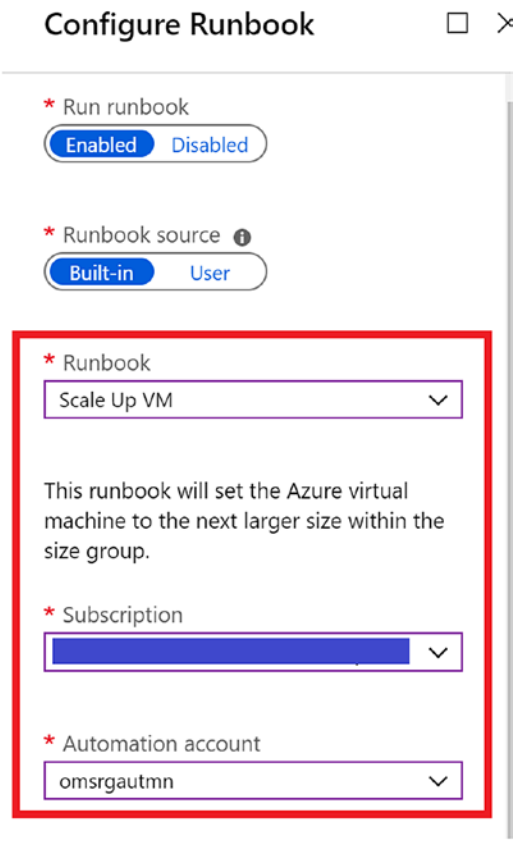

Figure 4-54. *Select Automation Runbook*

Now that an action group is created, let us see how to use the action group in alerts. To create an alert for a VM using collected metrics, browse to the VM ➤ Monitoring ➤ Alerts and Click on the new alert rule. Click on Add under Condition and click on the metrics to be used. In this example, let us create an alert rule based on the CPU percentage (Figure 4-55).

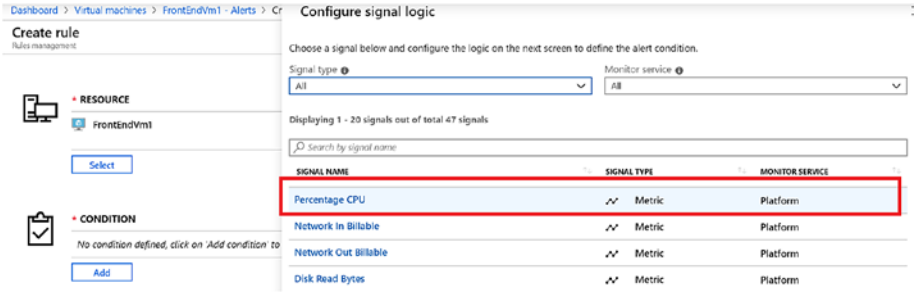

Figure 4-55. *Select signal logic for alert rule*

Configure the alert condition. Here the action group will be generated when the CPU utilization is greater than 80% for 5 minutes (Figure 4-56).

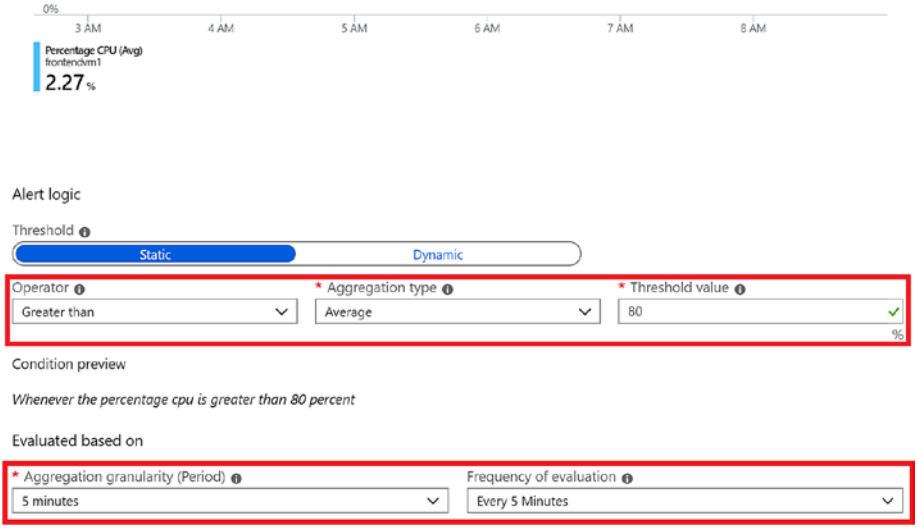

Figure 4-56. *Alert configuration*

The next step is to configure an action in the Action group. Click on Add and select the Action group that we created earlier. Essentially this alert setting scales up the VM to the next available size when the CPU percentage is higher than 80% for 5 minutes (Figure 4-57).

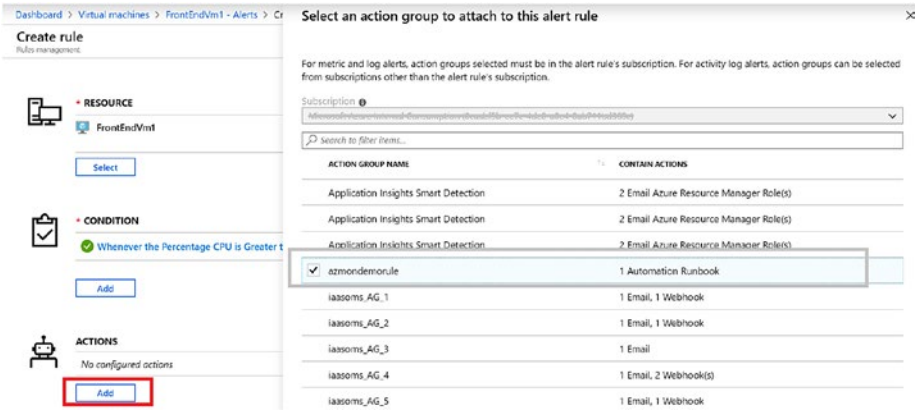

Figure 4-57. *Select action group*

Note You can add additional action groups to the same rule, for example, to send an email to the IT team so that when the configured condition matches, the IT team is alerted while the VM is also scaled up automatically.

The next step is to set an alert rule name, description, and severity; and then click on "Create Alert rule: to complete the configuration (Figure 4-58).

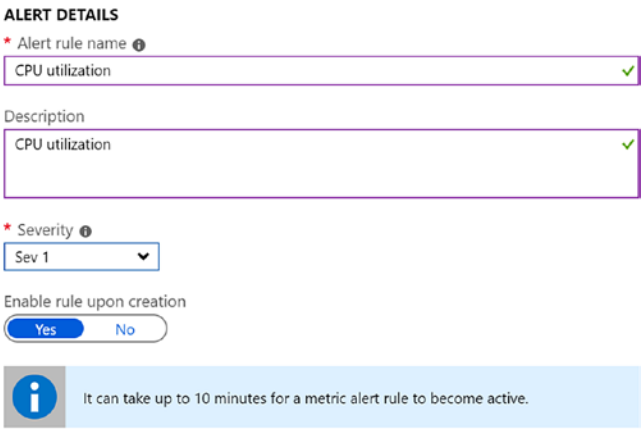

Figure 4-58. *Create Alert rule*

Summary

In this chapter, we have reviewed the different phases of implementation of monitoring for cloud-only scenarios: that is, evaluation, planning, and implementation. We also covered the different possible scenarios and implementation details of respective components. Azure Monitoring enables customers to configure alerts based on the monitoring configurations: that is, notifying stakeholders or taking actions such as executing runbooks. The configuration details of the same was also covered in this chapter.