Building a Scalable Application on Azure

This blog post was written with the assistance of ChatGPT-4o.

Table of Contents

Introduction

In the world of cloud computing, Microsoft Azure stands out as a robust platform for building, deploying, and managing applications. In our recent project, we leveraged several Azure services, including Azure Subscription, Azure Kubernetes Service (AKS), Application Insights, Virtual Machines (VMs), Event Hubs, API Management Services, and SQL Databases to create a scalable and monitored application infrastructure. This blog post outlines our approach, tools used, best practices, and detailed steps for managing clusters, fetching logs, and querying logs.

Getting Started with Azure Subscription

An Azure Subscription is your gateway to accessing Azure services. It acts as a container that holds all your resources, such as virtual machines, databases, and Kubernetes clusters.

1. **Setting Up Azure Subscription**:
   - **Sign Up**: If you don’ t have an Azure account, start by signing up at the Azure portal.
   - **Create a Subscription**: Navigate to the “Subscriptions” section and create a new subscription. This will be your billing and management container.

2. **Resource Organization**:
    - **Resource Groups**: Organize your resources into resource groups based on their lifecycle and management criteria.
    - **Tags**: Use tags for additional metadata and easier resource management and billing.

Deploying Applications with Azure Kubernetes Service (AKS)

Azure Kubernetes Service (AKS) is a managed Kubernetes service that simplifies deploying, managing, and scaling containerized applications.

Creating and Managing an AKS Cluster

1. **Creating an AKS Cluster in the Azure Portal**:
    - **Setup**: In the Azure portal, search for AKS and create a new Kubernetes cluster.
    - **Configuration**: Choose your cluster size, configure node pools, and set up networking.
    - **Authentication**: Use Azure Active Directory (AAD) for secure access control.
    - **Monitoring**: Enable monitoring and logging during the setup process.
2. **Creating an AKS Cluster with Azure CLI**:

```
az aks create \
  --resource-group myResourceGroup \
  --name myAKSCluster \
  --node-count 3 \
  --enable-addons monitoring \
  --generate-ssh-keys
```

3. **Managing Your AKS Cluster**:
    - **Scaling the Cluster**:

```
az aks scale \
  --resource-group myResourceGroup \
  --name myAKSCluster \
  --node-count 5
```

    - **Upgrading the Cluster**:

```
az aks upgrade \
  --resource-group myResourceGroup \
  --name myAKSCluster \
  --kubernetes-version 1.21.2
```

Deploying Applications

1. **Using Kubernetes Manifests**: Write YAML files for your deployments, services, and other Kubernetes objects.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp
spec:
  replicas: 2
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
      - name: myapp
        image: myregistry.azurecr.io/myapp:latest
        ports:
        - containerPort: 80
```

2. **Deploying with kubectl**:

```
kubectl apply -f myapp-deployment.yaml
```

3. **Helm Charts**: Use Helm for managing Kubernetes applications and version control.

```
helm install myapp ./mychart
```

Fetching Logs from Pods

1. **Attaching to a Pod and Fetching Logs**:

```
kubectl logs <pod-name>
```

   - To stream logs:

```
kubectl logs <pod-name> -f
```

2. **Using a Sidecar for Logging**:

- Create a logging sidecar container in your pod specification to ship logs to a centralized logging service.

```
spec:
  containers:
  - name: myapp
    image: myregistry.azurecr.io/myapp:latest
    ...
  - name: log-shipper
    image: log-shipper:latest
    ...
```

Monitoring and Diagnostics with Azure Application Insights

Application Insights provides powerful monitoring and diagnostic capabilities for your applications.

1. **Setting Up Application Insights**:
   - **Integration**: Add the Application Insights SDK to your application code.
   - **Instrumentation Key**: Configure your application with the instrumentation key from your Application Insights resource.
2. **Tracking Performance**:
   - **Metrics**: Monitor response times, failure rates, and application dependencies.
   - **Live Metrics Stream**: View real-time performance metrics for immediate insights.
3. **Diagnostics and Troubleshooting**:
   - **Application Map**: Visualize dependencies and identify performance bottlenecks.
   - **Transaction Diagnostics**: Use distributed tracing to trace requests across services.

Utilizing Azure Virtual Machines (VMs)

Azure VMs offer the flexibility to run custom applications and services that are not containerized.

1. **Provisioning Virtual Machines**:
   - **Create VMs**: In the Azure portal, create new virtual machines and choose the appropriate size and operating system.
   - **Network Configuration**: Set up virtual networks, subnets, and security groups to control traffic.
2. **Configuring VMs**:
   - **Software Installation**: Install required software and dependencies.
   - **Security**: Apply patches and updates regularly, configure firewalls, and use Network Security Groups (NSGs).
3. **Managing VMs**:

- **Backup and Restore**: Use Azure Backup for VM backups.
- **Monitoring**: Monitor VM performance using Azure Monitor.

Real-Time Data Ingestion with Azure Event Hubs

Azure Event Hubs is a big data streaming platform and event ingestion service capable of receiving and processing millions of events per second.

1. **Setting Up Event Hubs**:
   - **Create an Event Hub Namespace**: In the Azure portal, create an Event Hub namespace to house your Event Hubs.
   - **Create Event Hubs**: Within the namespace, create one or more Event Hubs to capture your data streams.
2. **Ingesting Data**:
   - **Producers**: Configure your application or services to send events to Event Hubs using SDKs available for multiple languages (e.g., .NET, Java, Python).
   - **Partitions**: Use partitions to scale event processing, ensuring high throughput and parallelism.
3. **Processing Events**:
   - **Consumers**: Use consumer groups to read and process events. Azure provides several options for processing, including Azure Stream Analytics, Azure Functions, and custom processing using the Event Hubs SDK.
4. **Monitoring Event Hubs**:
   - **Metrics**: Monitor throughput, latency, and event processing metrics through the Azure portal.
   - **Alerts**: Set up alerts to notify you of any issues, such as high latency or dropped messages.

Managing APIs with Azure API Management Services

Azure API Management Services provide a way to create consistent and modern API gateways for existing back-end services.

1. **Setting Up API Management**:
   - **Create an API Management Service**: In the Azure portal, search for API Management and create a new service.
   - **Configure APIs**: Define and import APIs from OpenAPI specifications, Azure Functions, or other backends.
2. **Securing APIs**:
   - **Authentication and Authorization**: Use OAuth2, JWT validation, and other mechanisms to secure your APIs.
   - **Rate Limiting and Throttling**: Implement policies to protect your APIs from abuse.

3. **Monitoring and Analytics**:
    - **API Insights**: Track usage, monitor performance, and analyze logs.
    - **Developer Portal**: Provide a portal for developers to discover and use your APIs.
4. **Managing Lifecycle**:
    - **Versioning and Revisions**: Manage different versions and revisions of your APIs seamlessly.
    - **Policy Management**: Apply policies for transformation, validation, and routing of requests

and responses.

Utilizing Azure SQL Databases

Azure SQL Database is a fully managed relational database with built-in intelligence, high availability, and scalability.

1. **Setting Up Azure SQL Database**:
    - **Create a SQL Database**: In the Azure portal, navigate to SQL Databases and create a new database.
    - **Configure Database**: Set the database size, performance level, and configure networking settings.
2. **Connecting to SQL Database**:
    - **Connection Strings**: Use the provided connection strings to connect your application to the SQL database.
    - **Firewall Rules**: Configure firewall rules to allow access from your application or local machine.
3. **Managing Database**:
    - **Backup and Restore**: Use automated backups and point-in-time restore to protect your data.
    - **Scaling**: Scale the database up or down based on your performance needs.
4. **Monitoring and Performance Tuning**:
    - **Query Performance Insights**: Monitor and optimize query performance.
    - **Automatic Tuning**: Enable automatic tuning features to improve performance.

Querying Logs with Kusto Query Language (KQL)

Kusto Query Language (KQL) is used to query Azure Monitor Logs, providing powerful insights into your log data.

1. **Basic KQL Query**:

```
// Retrieve records from a specific table
LogTableName
| where TimeGenerated > ago(1h)
| project TimeGenerated, Level, Message
```

2. **Filtering and Aggregating Data**:

```
LogTableName
| where TimeGenerated > ago(1h) and Level == "Error"
| summarize Count=count() by bin(TimeGenerated, 5m)
```

3. **Joining Tables**:

```
Table1
| join kind=inner (Table2) on $left.UserId == $right.UserId
| project Table1.TimeGenerated, Table1.Message, Table2.AdditionalInfo
```

4. **Creating Alerts Based on Queries**:
   - In the Azure portal, navigate to the Log Analytics workspace.
   - Click on `Logs` and enter your KQL query.
   - Click `New alert rule` to create an alert based on the query results.

Setting Up Alerts for Proactive Monitoring

Azure Alerts help you stay informed about the health and performance of your resources.

1. **Creating Alerts**:
   - **Metric Alerts**: Set up alerts based on metrics such as CPU usage, memory usage, and response times.
   - **Log Alerts**: Create alerts based on log search queries using KQL.
2. **Configuring Actions**:
   - **Action Groups**: Define action groups to specify who gets notified and how (email, SMS, webhook).
   - **Integration**: Integrate with ITSM tools like ServiceNow for automated incident management.
3. **Responding to Alerts**:
   - **Dashboards**: Set up Azure dashboards to provide a centralized view of alerts.
   - **Automation**: Use Azure Automation to automatically respond to certain alerts.

Conclusion

By leveraging Azure Subscription, AKS, Application Insights, Virtual Machines, Event Hubs, API Management Services, and SQL Databases, we built a scalable, robust, and monitored application infrastructure. Azure's comprehensive suite of tools ensured that we could deploy, manage, and monitor our applications efficiently. This setup not only improved our application performance but also provided us with the insights needed to maintain and optimize our resources proactively.