Microsoft

AZ-203.5
Module 02: Develop for code scalability

Prashanth Kumar

Microsoft

Topics

- Implement Autoscale
- Implement code that addresses singleton instances
- Implement code that handles transient faults

Lesson 01: Implement Autoscale

Autoscale

- A primary advantage of the cloud is elastic scaling (the ability to use as much capacity as you need)
  - Scaling out as load increases
  - Scaling in when the extra capacity is not needed
- Many Microsoft Azure services provide the capability to scale both manually and automatically
- Autoscale refers to the capability of many of these services to monitor the application instances and automatically scale appropriately to handle the current usage of the application
  - Using autoscale, your cloud service can scale out and in to exactly match the amount of instances needed for your specific computing pattern

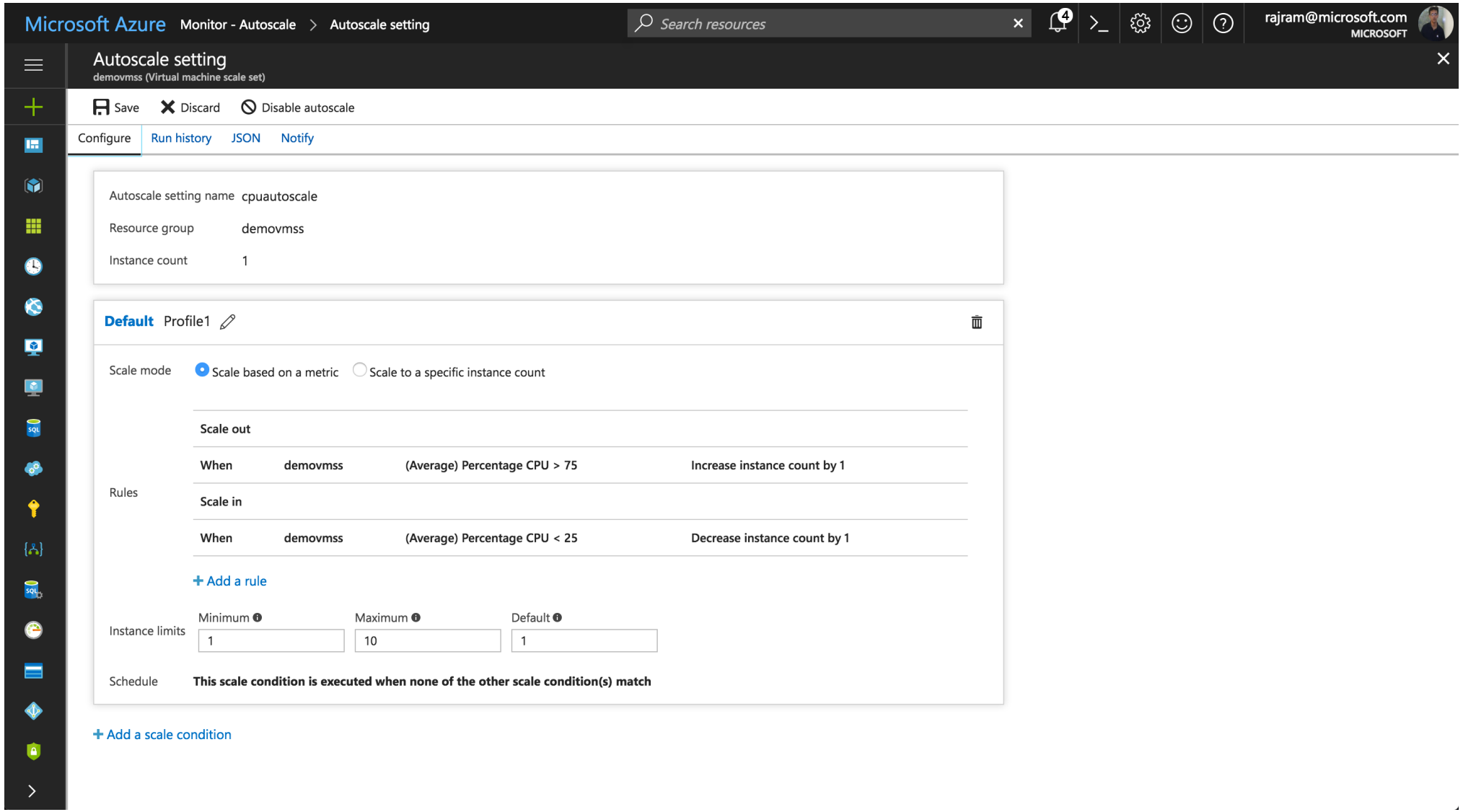
Autoscale metrics

Metric	Metric identifier	Description
CPU	CpuPercentage	The average amount of CPU time used across all instances of the plan
Memory	MemoryPercentage	The average amount of memory used across all instances of the plan
Data in	BytesReceived	The average incoming bandwidth used across all instances of the plan
Data out	BytesSent	The average outgoing bandwidth used across all instances of the plan
HTTP queue	HttpQueueLength	The average number of both read and write requests that were queued on storage. A high disk queue length is an indication of an application that might be slowing down due to excessive disk I/O.
Disk queue	DiskQueueLength	The average number of HTTP requests that had to sit in the queue before being fulfilled. A high or increasing HTTP queue length is a symptom of a plan under a heavy load.

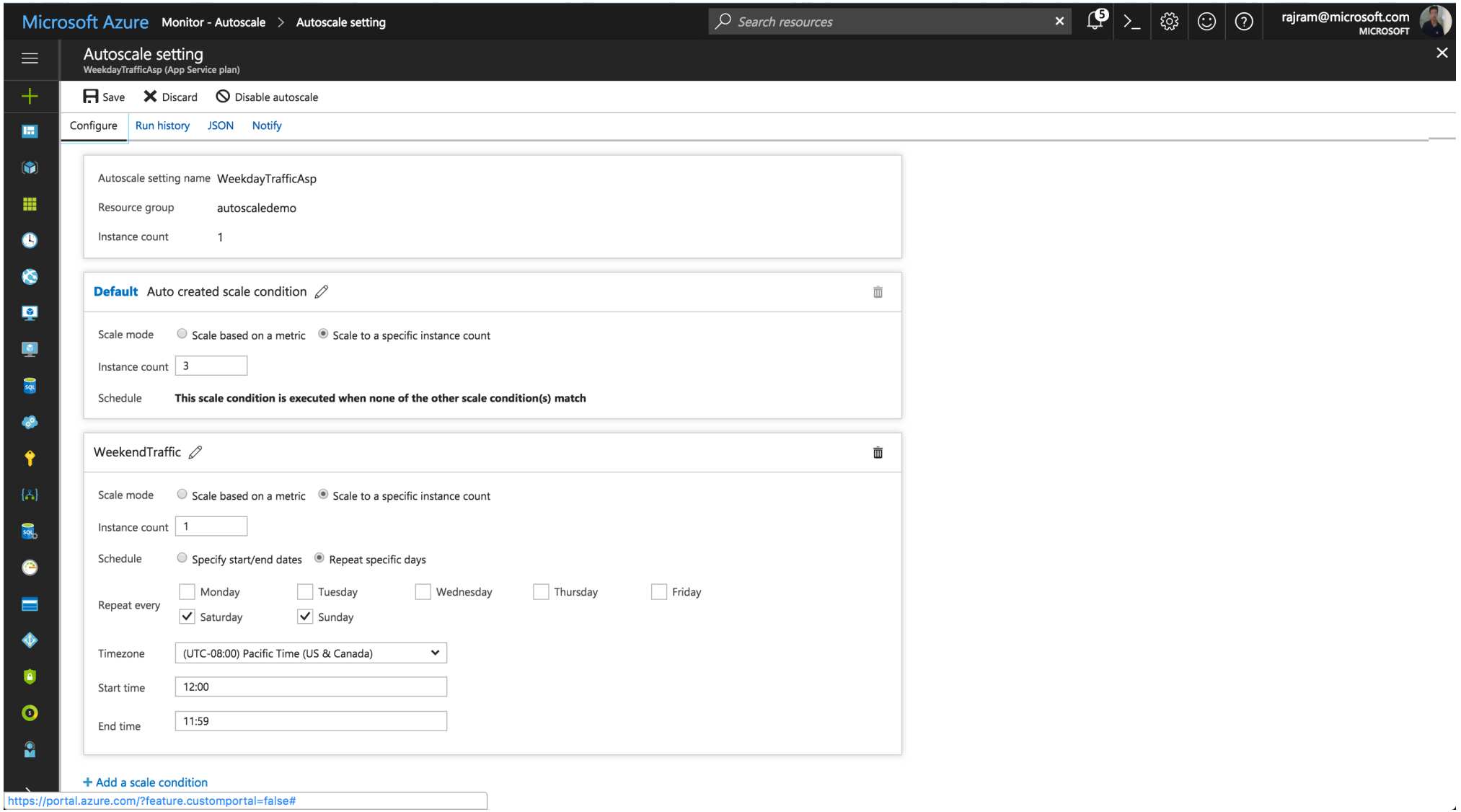
Autoscale patterns

- Scale based on CPU
- Scale differently on weekdays vs. weekends
- Scale differently during holidays
- Scale based on custom metric

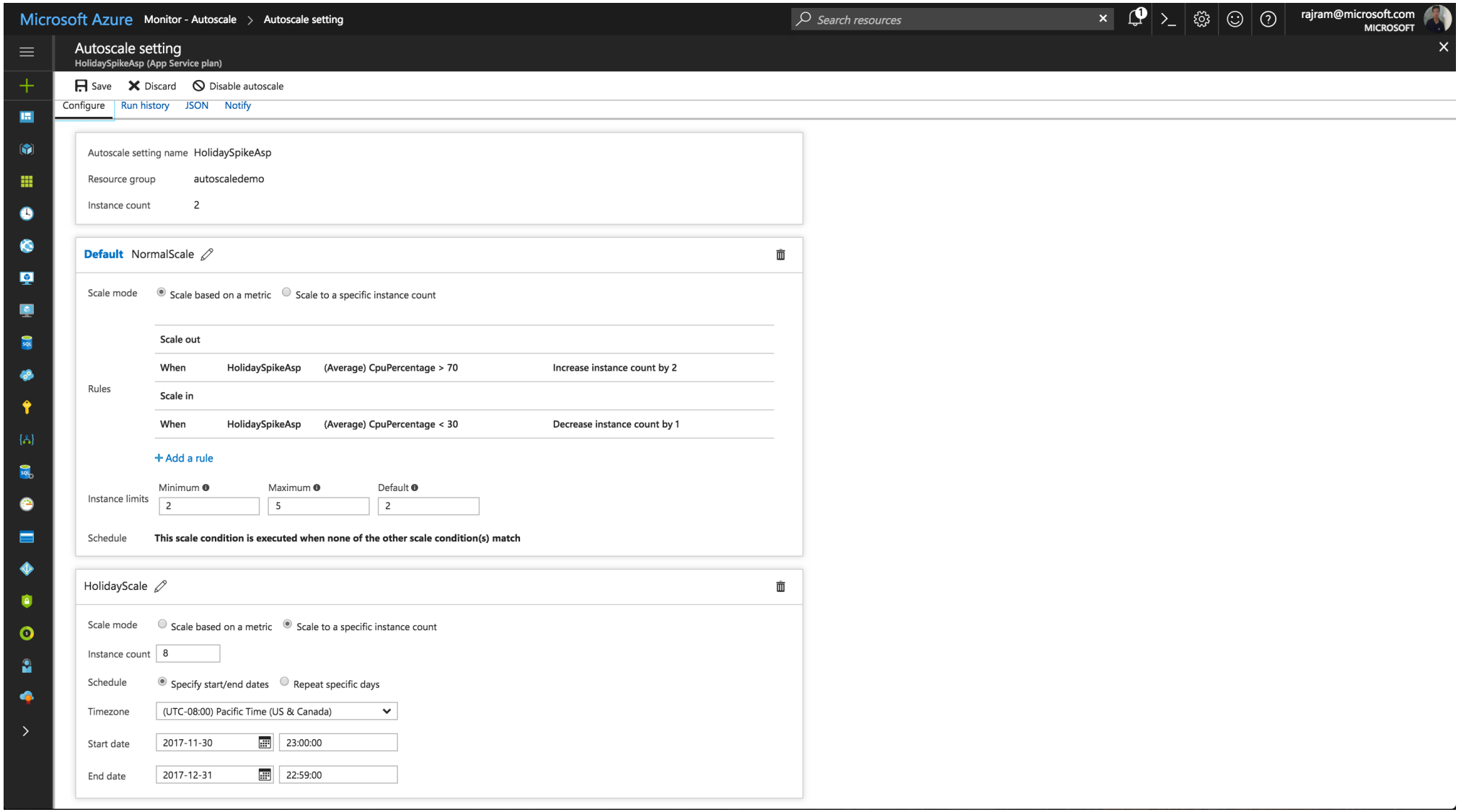
Scale based on CPU



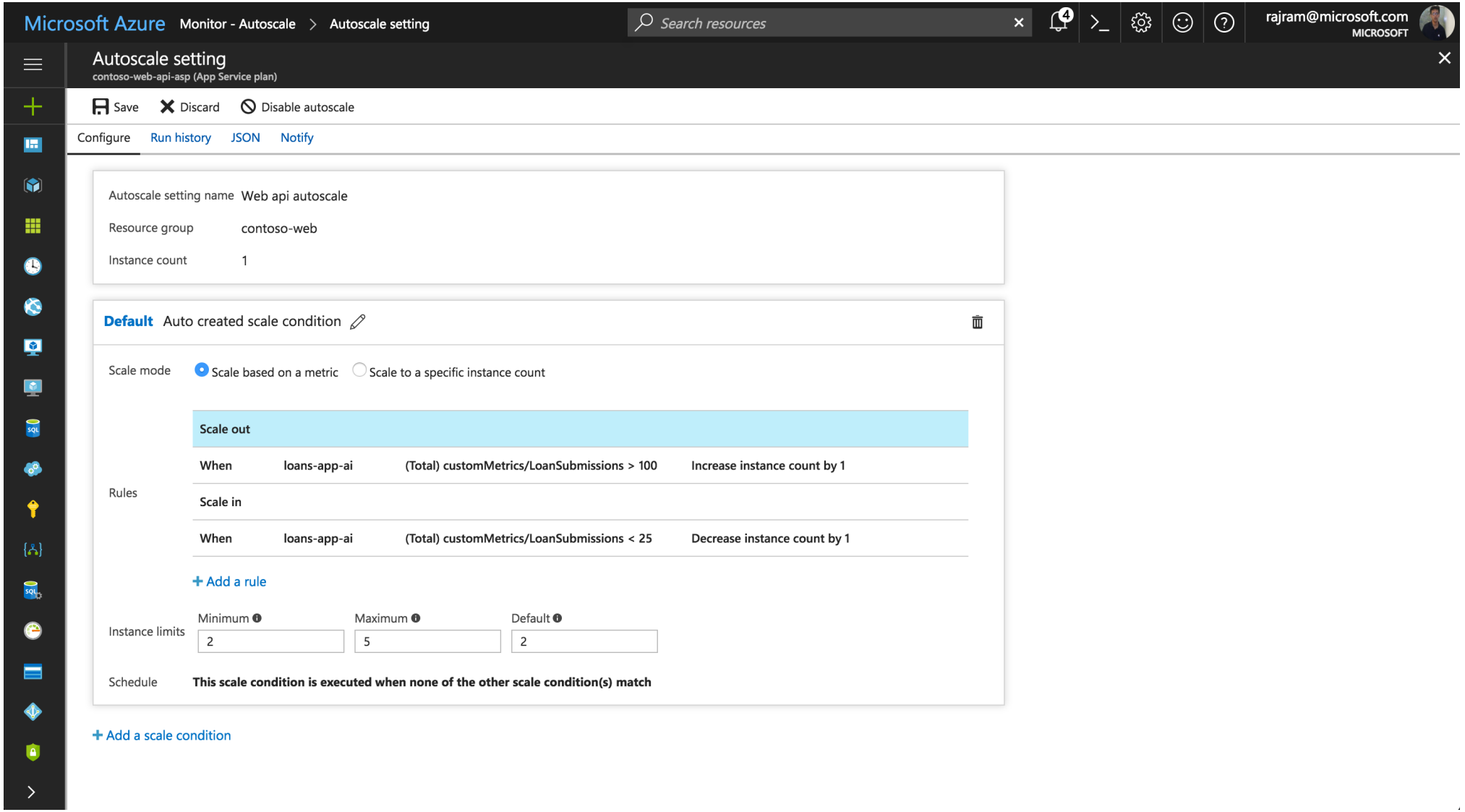
Scale differently on weekdays vs. weekends



Scale differently during holidays

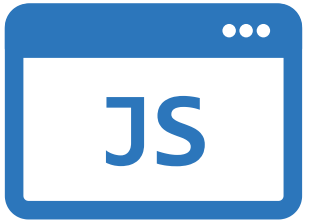


Scale based on custom metric



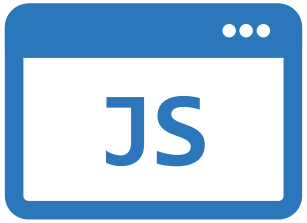
Autoscale setting schema

```
{
    "id": "...", "name": "demoSetting", "type": "Microsoft.Insights/autoscaleSettings",
    "location": "East US",
    "properties": {
        "enabled": true, "targetResourceUri": "...",
        "profiles": [
            {
                "name": "mainProfile",
                "capacity": {
                    "minimum": "1", "maximum": "4", "default": "1"
                },
                "rules": [ ... ]
            }
        ]
    }
}
```



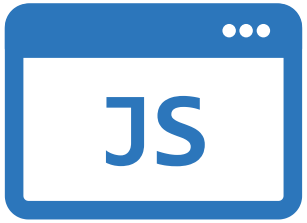
Autoscale setting schema – scale-out rule

```
{
    "metricTrigger": {
        "metricName": "Percentage CPU", "metricResourceUri": "...",
        "timeGrain": "PT1M",
        "statistic": "Average",
        "timeWindow": "PT10M",
        "timeAggregation": "Average",
        "operator": "GreaterThan",
        "threshold": 85
    },
    "scaleAction": {
        "direction": "Increase", "type": "ChangeCount",
        "value": "1",
        "cooldown": "PT5M"
    }
}
```



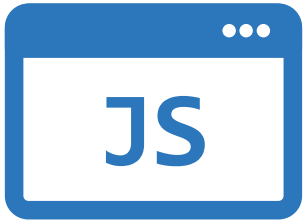
Autoscale setting schema – scale-in rule

```
{
    "metricTrigger": {
        "metricName": "Percentage CPU", "metricResourceUri": "...",
        "timeGrain": "PT1M",
        "statistic": "Average",
        "timeWindow": "PT10M",
        "timeAggregation": "Average",
        "operator": "LessThan",
        "threshold": 60
    },
    "scaleAction": {
        "direction": "Decrease", "type": "ChangeCount",
        "value": "1",
        "cooldown": "PT5M"
    }
}
```



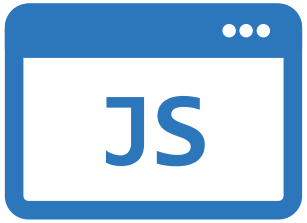
Autoscale setting schema – fixed date profile

```
"fixedDate": {
    "timeZone": "Pacific Standard Time",
    "start": "2020-12-26T00:00:00",
    "end": "2020-12-26T23:59:00"
}
```



Autoscale setting schema – weekdays and weekends

```
"recurrence": {
    "frequency": "Week",
    "schedule": {
        "timeZone": "Pacific Standard Time",
        "days": [
            "Saturday"
        ],
        "hours": [
            0
        ],
        "minutes": [
            0
        ]
    }
}
```

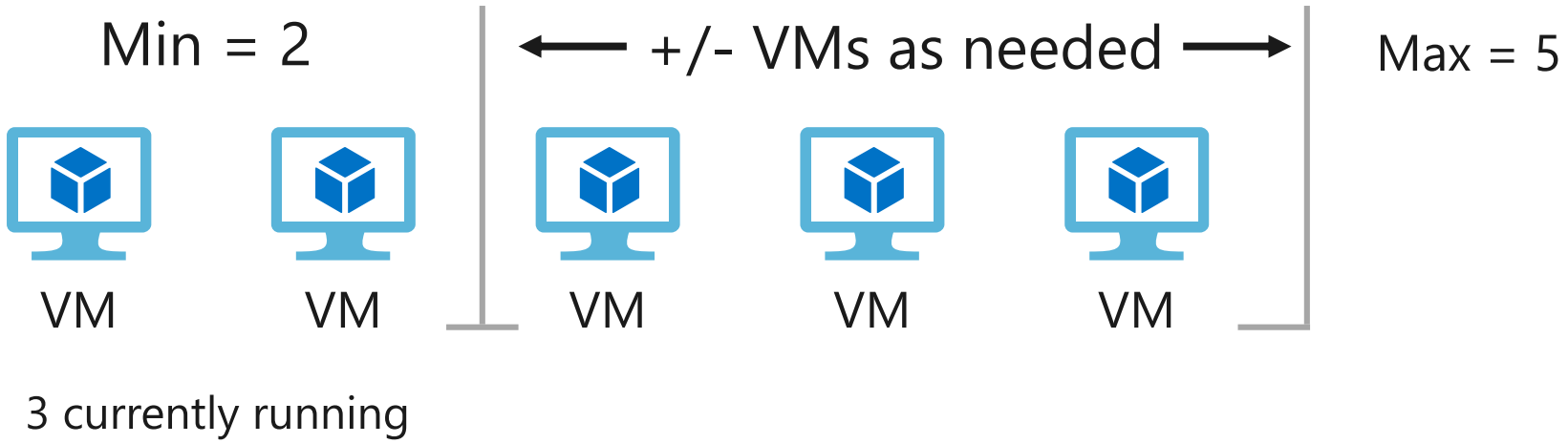


Autoscale concepts

- Each resource can have one autoscale setting
  - Autoscale settings can have one-to-many profiles
  - Profiles can have one-to-many rules
- Autoscale increases instances horizontally within bounds
  - Bounds are set by using the minimum, maximum, and default values
- Thresholds are calculated at an instance level
- Autoscale successful actions and failures are logged to the Activity Log

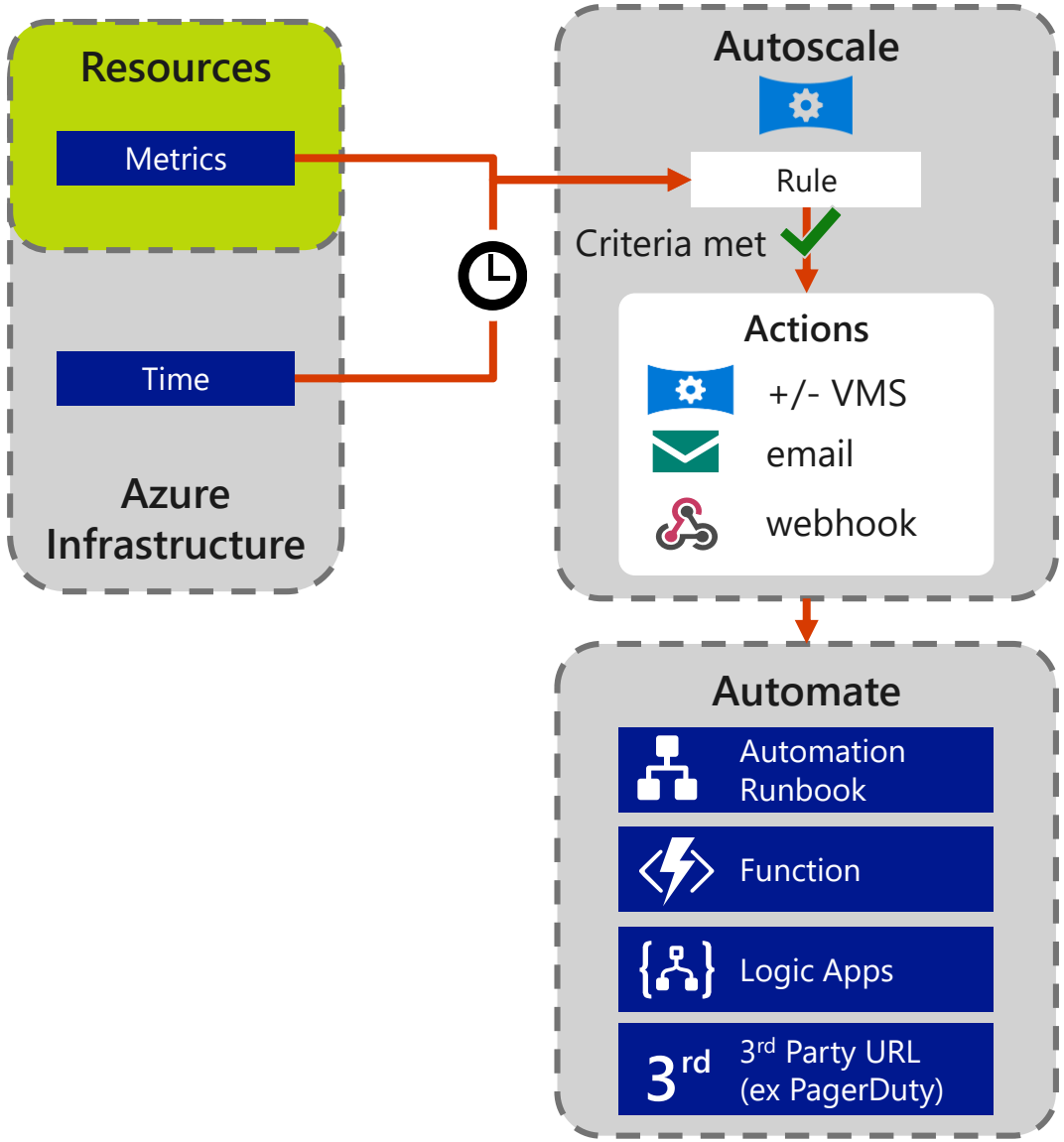
Autoscale thresholds

- Scale is constrained to a minimum and maximum
  - Your current instance count must be between the minimum and maximum
    - Minimum can help guarantee availability
    - Maximum can help control costs



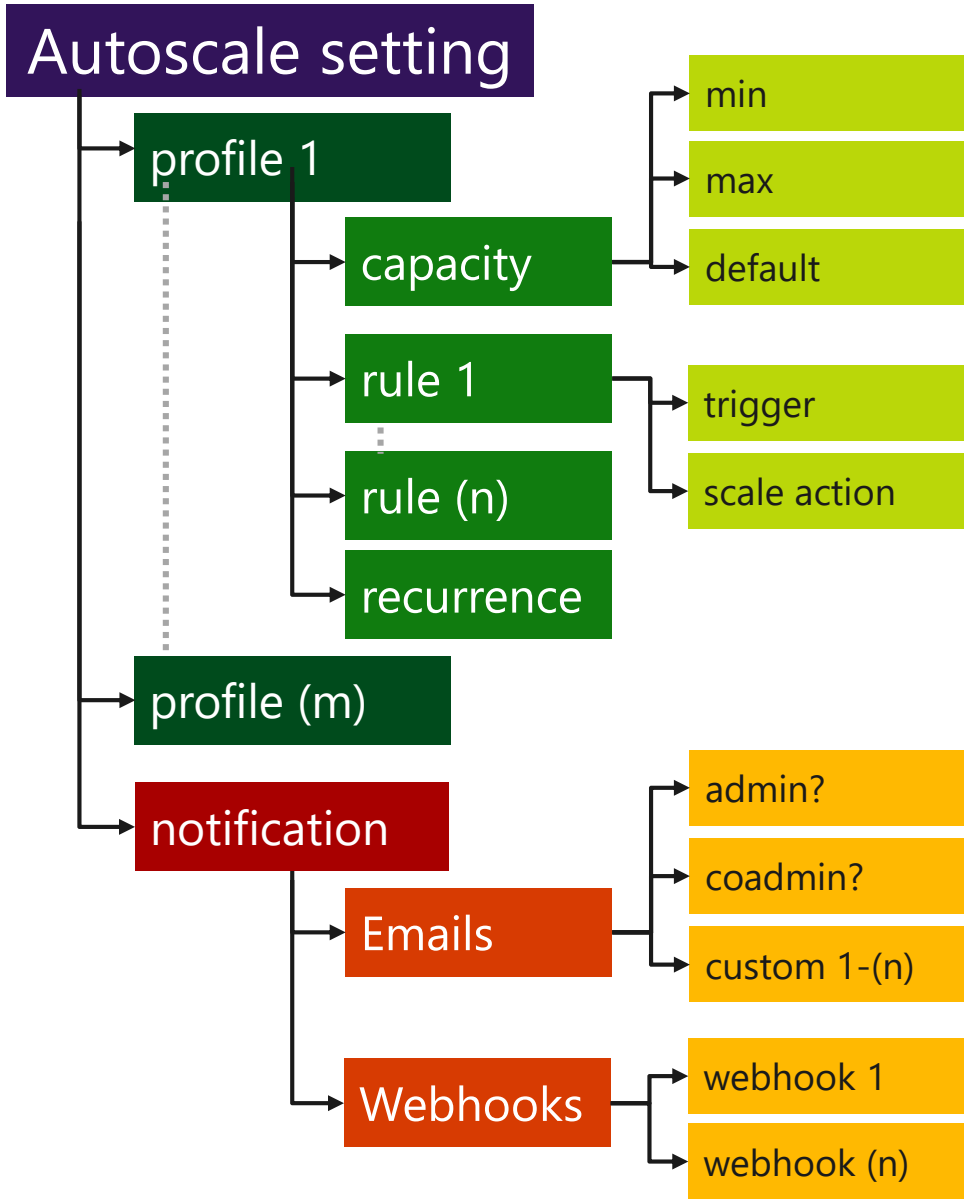
Autoscale workflow

1. Metrics are measured for a resource
2. When conditions are met (threshold surpassed), autoscale triggers
   a. Perform scale actions
   b. Send notifications (alerts)
   c. Send messages to webhooks for external automation



Autoscale hierarchy

- One autoscale setting
- Settings have one or more profiles
- Profiles have one or more rules
  - Profiles can also have recurrences and capacity settings
- Notifications can be directly associated with an autoscale setting



Best practices

- Ensure that the maximum and minimum values are different and have an adequate margin between them
- Manual scaling is reset by autoscale min and max
- Always use a scale-out and scale-in rule combination that performs an increase and decrease
- Choose the appropriate statistic for your diagnostics metric
- Choose the thresholds carefully for all metric types

Lesson 02: Implement code that addresses singleton instances

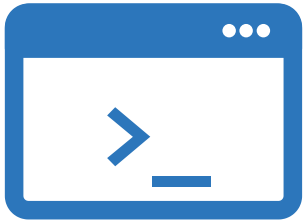
Querying resources by using Azure CLI

- Azure CLI
  - Microsoft cross-platform command-line experience for managing Azure resources
  - Can be installed on macOS, Linux, or Windows
  - Can be run in-browser by using the Azure Cloud Shell
- Supports JMESPath queries over the results of individual commands
  - JMESPath is a query language for JavaScript Object Notation (JSON) that gives you the ability to select and present data from JSON object or array output

Getting Azure Virtual Machines

```
az vm list
```

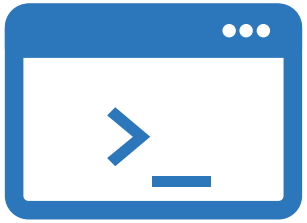
```
[
    {
        ...
        "hardwareProfile": {
            "vmSize": "Standard_B1s"
        },
        "location": "eastus",
        "name": "simple",
        ...
    }
]
```



Projecting virtual machine properties

```
az vm list --query '[].{name:name, image:storageProfile.imageReference.offer}'
```

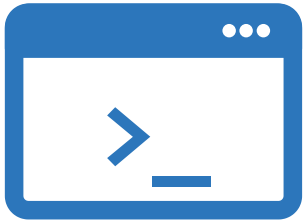
```
[
    {
        "image": "UbuntuServer",
        "name": "linuxvm"
    },
    {
        "image": "WindowsServer",
        "name": "winvm"
    }
]
```



Querying result set and projecting properties

```
az vm list --query "[?starts_with(storageProfile.imageReference.offer,
'Ubuntu')].{name:name, id:vmId}"
```

```
[
    {
        "name": "linuxvm",
        "id": "6aed2e80-64b2-401b-a8a0-b82ac8a6ed5c"
    }
]
```



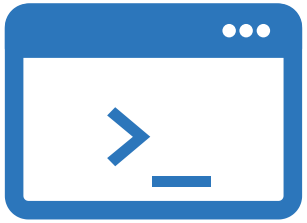
Querying resources by using the fluent Azure SDK

- Newest Azure SDK for .NET supports fluent queries
  - Fluent methods can be chained together to create easy-to-read queries
- Prior to using the fluent SDK, you had to create an authorization file that contains metadata about a service principal
- The authorization file is used to authenticate the SDK

Creating authorization file by using Azure CLI

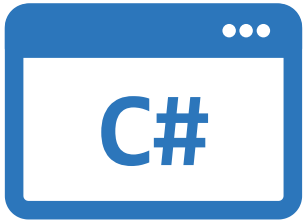
```
az ad sp create-for-rbac --sdk-auth > azure.auth
```

```
{
    "clientId": "b52dd125-9272-4b21-9862-0be667bdf6dc",
    "clientSecret": "ebc6e170-72b2-4b6f-9de2-99410964d2d0",
    "subscriptionId": "ffa52f27-be12-4cad-b1ea-c2c241b6cceb",
    "tenantId": "72f988bf-86f1-41af-91ab-2d7cd011db47",
    "activeDirectoryEndpointUrl": "https://login.microsoftonline.com",
    "resourceManagerEndpointUrl": "https://management.azure.com/",
    "activeDirectoryGraphResourceId": "https://graph.windows.net/",
    "sqlManagementEndpointUrl": "https://management.core.windows.net:8443/",
    "galleryEndpointUrl": "https://gallery.azure.com/",
    "managementEndpointUrl": "https://management.core.windows.net/"
}
```



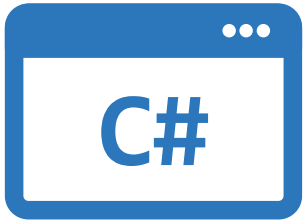
Authentication by using the fluent SDK

```
Azure azure = Azure.Authenticate("azure.auth")
    .WithDefaultSubscription();
```



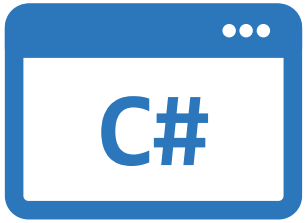
Listing virtual machines by using the fluent Azure SDK

```
var vms = await azure.VirtualMachines.ListAsync();

foreach(var vm in vms)
{
    Console.WriteLine(vm.Name);
}
```



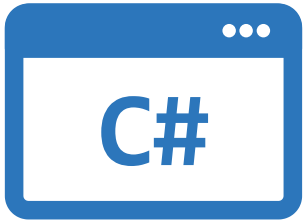
Listing virtual machines by using the fluent Azure SDK (continued)

```
var allvms = await azure.VirtualMachines.ListAsync();

IVirtualMachine targetvm = allvms
    .Where(vm => vm.Name == "simple")
    .SingleOrDefault();

Console.WriteLine(targetvm?.Id);
```

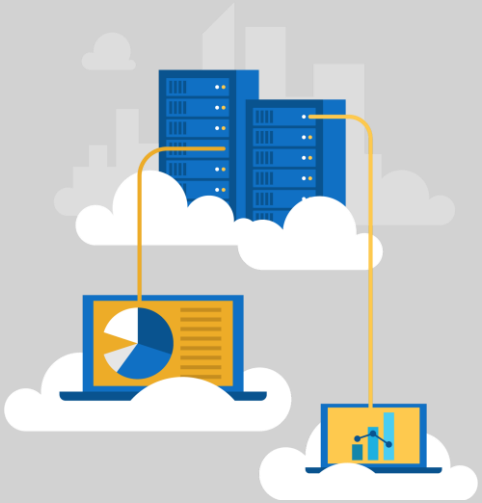


Access virtual machine metadata by using the fluent Azure SDK

```
INetworkInterface targetnic = targetvm
    .GetPrimaryNetworkInterface();

INicIPConfiguration targetipconfig = targetnic
    .PrimaryIPConfiguration;

IPublicIPAddress targetipaddress = targetipconfig
    .GetPublicIPAddress();

Console.WriteLine($"IP Address:\t{targetipaddress.IPAddress}");
```



Lesson 03: Implement code that handles transient faults



Transient errors

- Transient faults are **temporary** faults
  - Could be caused by environmental issues
    - Loss of network connectivity
    - Busy hardware components
    - Temporary unavailability of a connected service
    - Server timeouts
  - Typically are self-correcting

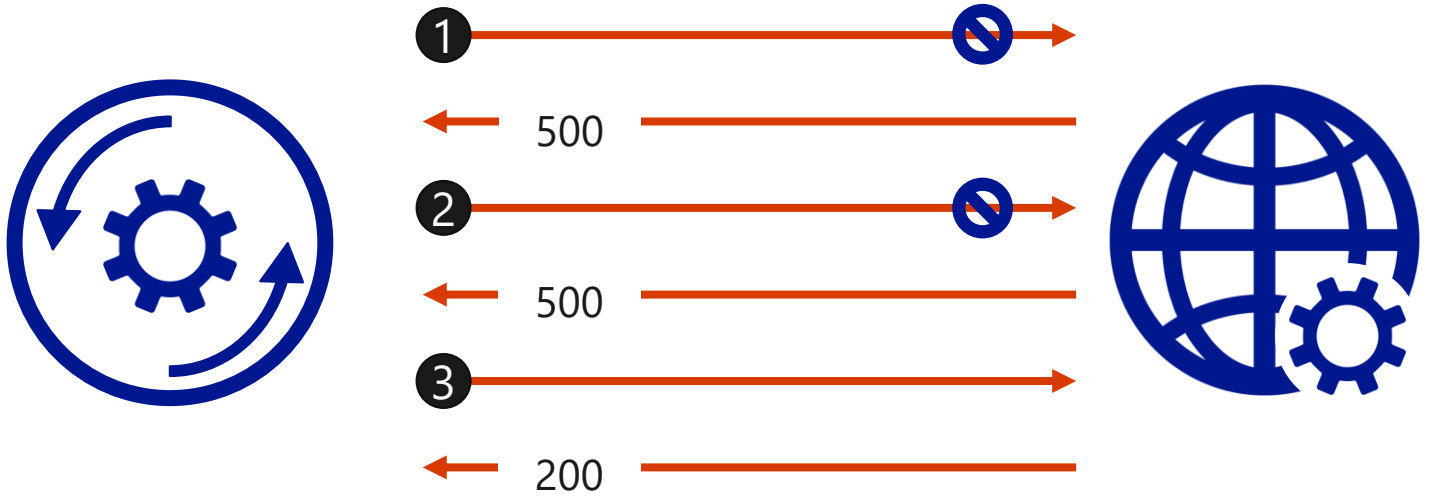
Handling transient errors

If an application detects a failure when it tries to send a request to a remote service, it can handle the failure by using the following strategies:

- Cancel
  - If the fault indicates that the failure isn't transient or is unlikely to be successful if repeated, the application should cancel the operation and report an exception. For example, an authentication failure caused by providing invalid credentials is not likely to succeed no matter how many times it's attempted.
- Retry
  - If the specific fault reported is unusual or rare, it might have been caused by unusual circumstances, such as a network packet becoming corrupted while it was being transmitted. In this case, the application could retry the failing request again immediately, because the same failure is unlikely to be repeated and the request will probably be successful.
- Retry after a delay
  - If the fault is caused by one of the more commonplace connectivity or busy failures, the network or service might need a short period of time while the connectivity issues are corrected or the backlog of work is cleared. The application should wait for a suitable amount of time before retrying the request.

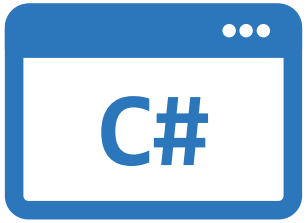
Retrying after a transient error

1. The application invokes an operation on a hosted service. The request fails, and the service host responds with HTTP response code 500 (internal server error).
2. The application waits for a short interval and tries again. The request still fails with HTTP response code 500.
3. The application waits for a longer interval and tries again. The request succeeds with HTTP response code 200 (OK).



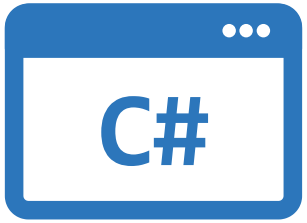
Handling transient errors in code

```
int retryCount = 3; readonly TimeSpan delay = TimeSpan.FromSeconds(5);
public async Task OperationWithBasicRetryAsync()
{
    int currentRetry = 0;
    for (;;)
    {
        try
        { await TransientOperationAsync(); break; }
        catch (Exception ex)
        {
            Trace.TraceError("Operation Exception");
            if (currentRetry++; > this.retryCount || !IsTransient(ex))
            { throw; }
        }
        await Task.Delay(delay);
    }
}
```



Detecting if an error is transient

```
private bool IsTransient(Exception ex)
{
    if (ex is OperationTransientException) return true;
    var webException = ex as WebException;
    if (webException != null)
    {
        return new[] {
            WebExceptionStatus.ConnectionClosed,
            WebExceptionStatus.Timeout,
            WebExceptionStatus.RequestCanceled
        }.Contains(webException.Status);
    }
    return false;
}
```



Microsoft

Review

- Implement autoscale
- Implement code that addresses singleton instances
- Implement code that handles transient faults