

# Introduction

2 minutes

Autoscaling enables a system to adjust the resources required to meet the varying demand from users, while controlling the costs associated with these resources. You can use autoscaling with many Azure services, including web applications. Autoscaling requires you to configure autoscale rules that specify the conditions under which resources should be added or removed.

Imagine that you work for a large chain of hotels. You have a web site that customers can visit to make bookings, and retrieve the details of bookings that they have previously made.

When special events occur, such as a concert or sports event, hotels near the venue can experience short-term spikes in booking requests. Similar events can occur with minimal notice, making it difficult to plan in advance for the potential increase in traffic. The unpredictability of these events means manual scaling isn't an option, and it could be prohibitively expensive to keep website resources available on standby for such an eventuality.

The hotel reservation system must provide a timely response back to users, even during periods of excessive demand. You also need to keep running costs down during more quiescent periods.

This module shows you how to use autoscaling with App Service to scale a web app to meet demand at peak times.

## Learning objectives

In this module, you will:

- Identify scenarios for which autoscaling is an appropriate solution
- Create autoscaling rules for a web app
- Monitor the effects of autoscaling

## Prerequisites

- Experience using the Azure portal to create and manage App Service web apps
- Basic familiarity with manually scaling an App Service

**Next unit: Identify suitable scenarios for autoscaling**

Continue T

# Identify suitable scenarios for autoscaling

5 minutes

Autoscaling can be triggered according to a schedule, or by assessing whether the system is running short on resources. For example, autoscaling could be triggered if CPU utilization grows, memory occupancy increases, the number of incoming requests to a service appears to be surging, or some combination of factors.

In the hotel reservation system, autoscaling is useful for handling short-term spikes in the number of booking requests. You can also use scheduled autoscaling to provide additional resources at peak times.

## What is autoscaling?

Autoscaling is a cloud system or process that adjusts available resources based on the current demand. Autoscaling performs *scaling up and out*, as opposed to *scaling up and down*.

## Azure App Service Autoscaling

Autoscaling in Azure App Service monitors the resource metrics of a web app as it runs. It detects situations where additional resources are required to handle an increasing workload, and ensures those resources are available before the system becomes overloaded.

Autoscaling responds to changes in the environment by adding or removing web servers and balancing the load between them. Autoscaling doesn't have any effect on the CPU power, memory, or storage capacity of the web servers powering the app, it only changes the number of web servers.

### Autoscaling rules

Autoscaling makes its decisions based on rules that you define. A rule specifies the threshold for a metric, and triggers an autoscale event when this threshold is crossed. Autoscaling can also deallocate resources when the workload has diminished.

Define your autoscaling rules carefully. For example, a Denial of Service attack will likely result in a large-scale influx of incoming traffic. Trying to handle a surge in requests caused by a DoS attack would be fruitless and expensive. These requests aren't genuine, and should be discarded rather than processed. A better solution is to implement detection and filtering of requests that occur during such an attack before they reach your service.

## When should you consider autoscaling?

Autoscaling provides elasticity for your services. It's a suitable solution when hosting any application when you can't easily predict the workload in advance, or when the workload is likely to vary by date or time. For example, you might expect increased/reduced activity for a business app during holidays.

Autoscaling improves availability and fault tolerance. It can help ensure that client requests to a service won't be denied because an instance is either not able to acknowledge the request in a timely manner, or because an overloaded instance has crashed.

Autoscaling works by adding or removing web servers. If your web apps perform resource-intensive processing as part of each request, then autoscaling might not be an effective approach. In these situations, manually scaling up may be necessary. For example, if a request sent to a web app involves performing complex processing over a large dataset, depending on the instance size, this single request could exhaust the processing and memory capacity of the instance.

Autoscaling isn't the best approach to handling long-term growth. You might have a web app that starts with a small number of users, but increases in popularity over time. Autoscaling has an overhead associated with monitoring resources and determining whether to trigger a scaling event. In this scenario, if you can anticipate the rate of growth, manually scaling the system over time may be a more cost effective approach.

The number of instances of a service is also a factor. You might expect to run only a few instances of a service most of the time. However, in this situation, your service will always be susceptible to downtime or lack of availability whether autoscaling is enabled or not. The fewer the number of instances initially, the less capacity you have to handle an increasing workload while autoscaling spins up additional instances.

Next unit: Knowledge Check - Identify suitable scenarios for autoscaling

Continue

# Knowledge Check - Identify suitable scenarios for autoscaling

3 minutes

## Check your knowledge

1. Which of these statements best describes autoscaling?

- ☐ Autoscaling is a scale up/scale down solution. It migrates an application to more powerful hardware during periods of high demand, and moves it back to less powerful hardware when demand drops.
- ☐ Autoscaling requires an administrator to actively monitor the workload on a system. If the workload increases and response times start to drop, the administrator can trigger autoscaling to help increase the throughput of the system.
- ☒ Autoscaling is a scale out/scale in solution. The system can scale out when specified resource metrics indicate increasing usage, and scale in when these metrics drop.

**This answer is correct**

- ☐ Autoscaling is an ideal solution for handling sudden bursts of activity that last for a few minutes.
- ☐ Scaling in and out provides better availability than autoscaling.

2. Autoscaling enables a system to predict the resources that will be required to handle an increasing workload.

- ☐ True
- ☒ False

**This answer is correct**

3. Which of these scenarios is a suitable candidate for autoscaling?

- ☒ The number of users requiring access to an application varies according to a regular schedule. For example, more users use the system on a Friday than other days of the week.

**This answer is correct**

- ☐ The system is subject to a sudden influx of requests that grinds your system to a halt. The workload has increased exponentially and there appears to be no reason for this surge of activity.
- ☐ Over time, your service is becoming more popular, and you need to handle the increasing traffic.
- ☐ Your organization is running a promotion and expects to see increased traffic to their web site for the next couple of weeks. Ensure sufficient resources are available to handle the demand expected when the promotion starts.

**Next unit: Identify factors for implementing autoscaling**

Continue

# Identify factors for implementing autoscaling

7 minutes

Autoscaling enables you to specify the conditions under which a web app should be scaled out, and back in again. Effective autoscaling ensures sufficient resources are available to handle large volumes of requests at peak times, while managing costs when the demand drops.

You can configure autoscaling to detect when to scale in and out according to a combination of factors, based on resource usage. You can also configure autoscaling to occur according to a schedule.


In this unit, you'll learn how to specify the factors that can be used to autoscale a service.

## Autoscaling and the App Service Plan

When you create a web app, you also create an App Service Plan. The App Service Plan defines the operating system (Windows or Linux) used to host the web app. The Pricing Tier of the plan specifies the hardware available (memory capacity, CPU processing capacity, disk storage), and other services, such as regular backups for each instance of the web app.

Autoscaling is a feature of the App Service Plan used by the web app. When the web app scales out, Azure starts new instances of the hardware defined by the App Service Plan to the app.

To prevent runaway autoscaling, an App Service Plan has an instance limit. Plans in more expensive pricing tiers have a higher limit. Autoscaling cannot create more instances than this limit. As an example, the Standard series of service plans support up to 10 instances, and cost up to 0.40/hour for each instance. The Premium series of service plans enable 20 instances, at a price of up to 0.80/hour for each instance. The Isolated service plans allow 100 instances, and cost \$1.60/hour for each instance.

 **Note**

Not all App Service Plan pricing tiers support autoscaling.

## Autoscale conditions

You indicate how to autoscale by creating autoscale conditions. Azure provides two options for autoscaling:

- Scale based on a metric, such as the length of the disk queue, or the number of HTTP requests awaiting processing,
- Scale to a specific instance count according to a schedule. For example, you can arrange to scale out at a particular time of day, or on a specific date or day of the week. You also specify an end date, and the system will scale back in at this time.

Scaling to a specific instance count only enables you to scale out to a defined number of instances. If you need to scale out incrementally, you can combine metric and schedule-based autoscaling in the same autocal condition. So, you could arrange for the system to scale out if the number of HTTP requests exceeds some threshold, but only between certain hours of the day.

You can create multiple autoscale conditions to handle different schedules and metrics. Azure will autoscale your service when any of these conditions apply. An App Service Plan also has a default condition that will be used if none of the other conditions are applicable. This condition is always active and doesn't have a schedule.

## Metrics for autoscale rules

Autoscaling by metric requires that you define one or more autoscale rules. An autoscale rule specifies a metric to monitor, and how autoscaling should respond when this metric crosses a defined threshold. The metrics you can monitor for a web app are:

- **CPU Percentage** This metric is an indication of the CPU utilization across all instances. A high value shows that instances are becoming CPU-bound, which could cause delays in processing client requests.
- **Memory Percentage** This metric captures the memory occupancy of the application across all instances. A high value indicates that free memory could be running low, and could cause one or more instances to fail.
- **Disk Queue Length** This metric is a measure of the number of outstanding I/O requests across all instances. A high value means that disk contention could be occurring.
- **Http Queue Length** This metric shows how many client requests are waiting for processing by the web app. If this number is large, client requests might fail with HTTP 408 (Timeout) errors.

- **Data In** This metric is the number of bytes received across all instances.
- **Data Out** This metric is the number of bytes sent by all instances.

You can also scale based on metrics for other Azure services. For example, if the web app processes requests received from a Service Bus Queue, you might want to spin up additional instances of a web app if the number of items held in an Azure Service Bus Queue exceeds a critical length.

## How an autoscale rule analyzes metrics

Autoscaling works by analyzing trends in metric values over time across all instances. Analysis is a multistep process.

In the first step, an autoscale rule aggregates the values retrieved for a metric for all instances across a period of time known as the *time grain*. Each metric has its own intrinsic time grain, but in most cases this period is 1 minute. The aggregated value is known as the *time aggregation*. The options available are *Average*, *Minimum*, *Maximum*, *Total*, *Last*, and *Count*.

An interval of one minute is a very short interval in which to determine whether any change in metric is long-lasting enough to make autoscaling worthwhile. So an autoscale rule performs a second step that performs a further aggregation of the value calculated by the *time aggregation* over a longer, user-specified period, known as the *Duration*. The minimum *Duration* is 5 minutes. If the *Duration* is set to 10 minutes for example, the autoscale rule will aggregate the 10 values calculated for the *time grain*.

The aggregation calculation for the *Duration* can be different for that of the *time grain*. For example, if the *time aggregation* is *Average* and the statistic gathered is *CPU Percentage* across a one-minute *time grain*, each minute the average CPU percentage utilization across all instances for that minute will be calculated. If the *time grain statistic* is set to *Maximum*, and the *Duration* of the rule is set to 10 minutes, the maximum of the 10 average values for the CPU percentage utilization will be used to determine whether the rule threshold has been crossed.

## Autoscale actions

When an autoscale rule detects that a metric has crossed a threshold, it can perform an autoscale action. An autoscale action can *scale-out* or *scale-in*. A scale-out action increases the number of instances, and a scale-in action reduces the instance count. An autoscale action uses an operator (such as *less than*, *greater than*, *equal to*, and so on) to determine how to react to the threshold. Scale-out actions typically use the *greater than* operator to compare the metric value to the threshold. Scale-in actions tend to compare the metric value to the threshold with the *less than* operator. An autoscale action can also set the instance count to a specific level, rather than incrementing or decrementing the number available.

An autoscale action has a *cool down* period, specified in minutes. During this interval, the scale rule won't be triggered again. This is to allow the system to stabilize between autoscale events. Remember that it takes time to start up or shut down instances, and so any metrics gathered might not show any significant changes for several minutes. The minimum cool down period is five minutes.

## Pairing autoscale rules

You should plan for scaling-in when a workload decreases. Consider defining autoscale rules in pairs in the same autoscale condition. One autoscale rule should indicate how to scale the system out when a metric exceeds an upper threshold. The other rule should define how to scale the system back in again when the same metric drops below a lower threshold.

## Combining autoscale rules

A single autoscale condition can contain several autoscale rules (for example, a scale-out rule and the corresponding scale-in rule). However, the autoscale rules in an autoscale condition don't have to be directly related. You could define the following four rules in the same autoscale condition:

- If the HTTP queue length exceeds 10, scale out by 1
- If the CPU utilization exceeds 70%, scale out by 1
- If the HTTP queue length is zero, scale in by 1
- If the CPU utilization drops below 50%, scale in by 1

When determining whether to scale out, the autoscale action will be performed **any** of the scale-out rules are met (HTTP queue length exceeds 10 **or** CPU utilization exceeds 70%). When scaling in, the autoscale action will run **only if all** of the scale-in rules are met (HTTP queue length drops to zero **and** CPU utilization falls below 50%). If you need to scale in if only one the scale-in rules are met, you must define the rules in separate autoscale conditions.

**Next unit: Knowledge Check - Identify factors for implementing autoscaling**

# Knowledge Check - Identify factors for implementing autoscaling

3 minutes

## Check your knowledge

1. You have defined an autoscale condition with four autoscale rules. The first rule scales out when the CPU utilization reaches 70 percent. The second rule scales back in when the CPU utilization drops below 50 percent. The third rule scales out if memory occupancy exceeds 75 percent. The fourth rule scales back in when memory occupancy falls below 50 percent. When will the system scale-out?

☒ When CPU utilization reaches 70 percent **or** memory occupancy exceeds 75 percent

This answer is correct

☐ When CPU utilization reaches 70 percent **and** memory occupancy exceeds 75 percent

☐ You can't do this with a single autoscale condition. An autoscale condition can only contain autoscale rules that use the same metric

2. An autoscale rule defines a scale-out action that increases the instance count when the disk queue length exceeds 10. The system scaled out two minutes ago but the disk queue length is still over 10. When will the system scale-out again?

☐ The autoscale rule will trigger another autoscale action immediately, and will continue doing so until the disk queue length drops below 10, or the maximum number of instances have been created.

☐ The autoscale rule will not run again until the system has scaled back in.

☒ The autoscale rule will not trigger the action again until the **cool down** period for the rule has expired. If the disk queue length is still over 10 at this time, the action will be performed.

This answer is correct

Next unit: Autoscale a web app

Continue

# Autoscale a web app

5 minutes

By default, an App Service Plan only implements manual scaling. The hotel reservation system in the scenario requires you to configure autoscaling.

In this unit, you will see how to enable autoscaling, create autoscale rules, and monitor autoscaling activity

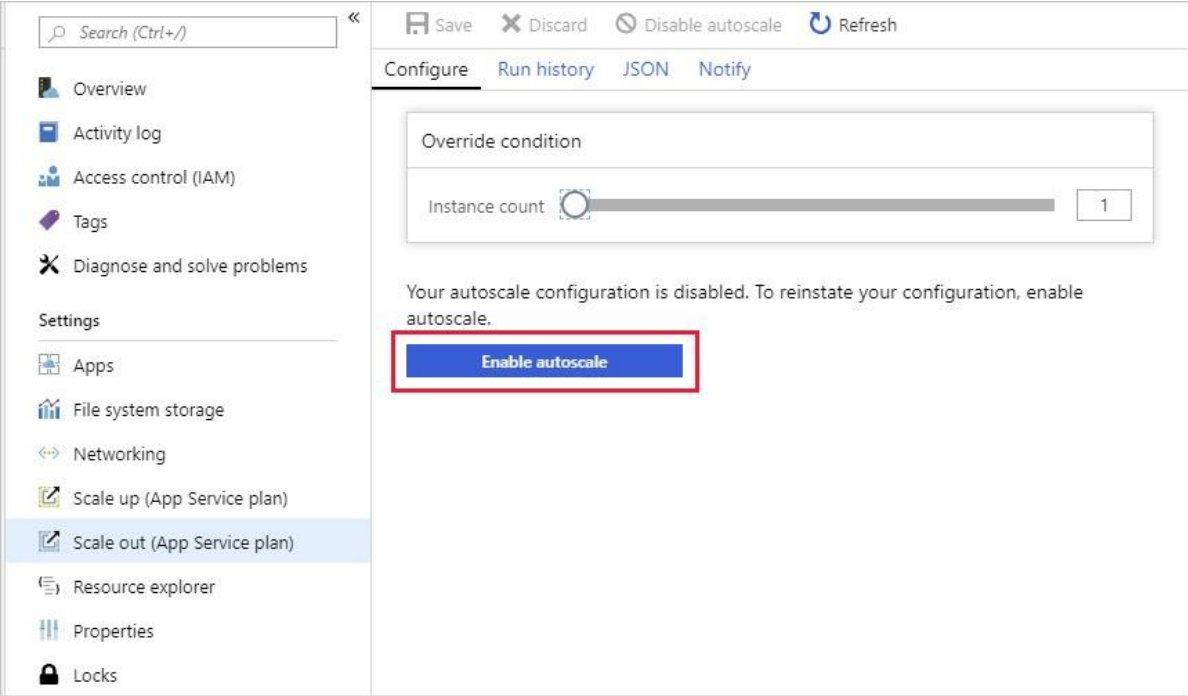
## Enable autoscaling

You modify the App Service Plan for a web app to implement autoscaling. An App Service Plan has scale-out settings that you use to enable autoscaling, add autoscaling conditions, and define autoscale rules.

### Note

Not all pricing tiers support autoscaling. The development pricing tiers are either limited to a single instance (the **D1** tier), or they only provide manual scaling (the **B1** tier). If you've selected one of these tiers, you must first scale up to the **S1** or any of the **P** level production tiers.

You enable autoscaling with the **Enable autoscale** button on the **Scale out** page for an App Service Plan.



## Add scale conditions

Once you enable autoscaling, you can edit the default scale condition, and you can add your own custom scale conditions. Remember that each scale condition can either scale based on a metric, or scale to a specific instance count.

Search (Ctrl+/)

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Settings

Apps

File system storage

Networking

Scale up (App Service plan)

Scale out (App Service plan)

Resource explorer

Properties

Locks

Export template

Monitoring

Alerts

Metrics

SaveDiscardDisable autoscaleRefresh

ConfigureRun historyJSONNotify

\* Autoscale setting name

Resource group

DefaultAuto created scale condition

Delete warning

Scale mode

Rules

Instance limits

Schedule

The very last or default recurrence rule cannot be deleted. Instead, you can disable autoscale to turn off autoscale.

Scale based on a metric

Scale to a specific instance count

Scale out and scale in your instances based on metric. For example: 'Add a rule that increases instance count by 1 when CPU percentage is above 70%'

It is recommended to have at least one scale in rule

+ Add a rule

Minimum

Maximum

Default

1

2

1

This scale condition is executed when none of the other scale condition(s) match

+ Add a scale condition

A metric-based scale condition can also specify the minimum and maximum number of instances to create. The maximum number can't exceed the limits defined by the pricing tier. Additionally, all scale conditions other than the default may include a schedule indicating when the condition should be applied.

## Create scale rules

A metric-based scale condition contains one or more scale rules. Initially, a scale condition contains only a default rule. You use **Add a rule** link to add your own custom rules. You define the criteria that indicate when a rule should trigger an autoscale action, and the autoscale action to be performed (scale out or scale in) using the metrics, aggregations, operators, and thresholds described earlier.



Home > hotelsystemplan - Scale out (App Service plan)

App Service plan

Search (Ctrl+ /)

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Settings

Apps

File system storage

Networking

Scale up (App Service plan)

Scale out (App Service plan)

Resource explorer

Properties

Locks

Export template

Monitoring

Alerts

Metrics

Save Discard Disable autoscale Refresh

Configure Run history JSON Notify

\* Autoscale setting name Resource group

Default Auto created scale condition

Delete warning The very last or default recurrence rule cannot be deleted. Instead, you can disable

Scale mode ☒ Scale based on a metric ☐ Scale to a specific instance count

Scale out and scale in your instances based on metric. For example: 'Add a rule that incre percentage is above 70%'

Rules It is recommended to have at least one scale in rule

+ Add a rule

Instance limits Minimum 1 Maximum 2 Default 1

Schedule This scale condition is executed when none of the other scale condition(s) match

+ Add a scale condition

Scale rule

Metric source Current resource (hotelsystemplan)

Resource type App Service plans

Resource hotelsystemplan

Criteria

\* Time aggregation Average

\* Metric name CPU Percentage 1 minute time grain

\* Time grain statistic Average

\* Operator Greater than

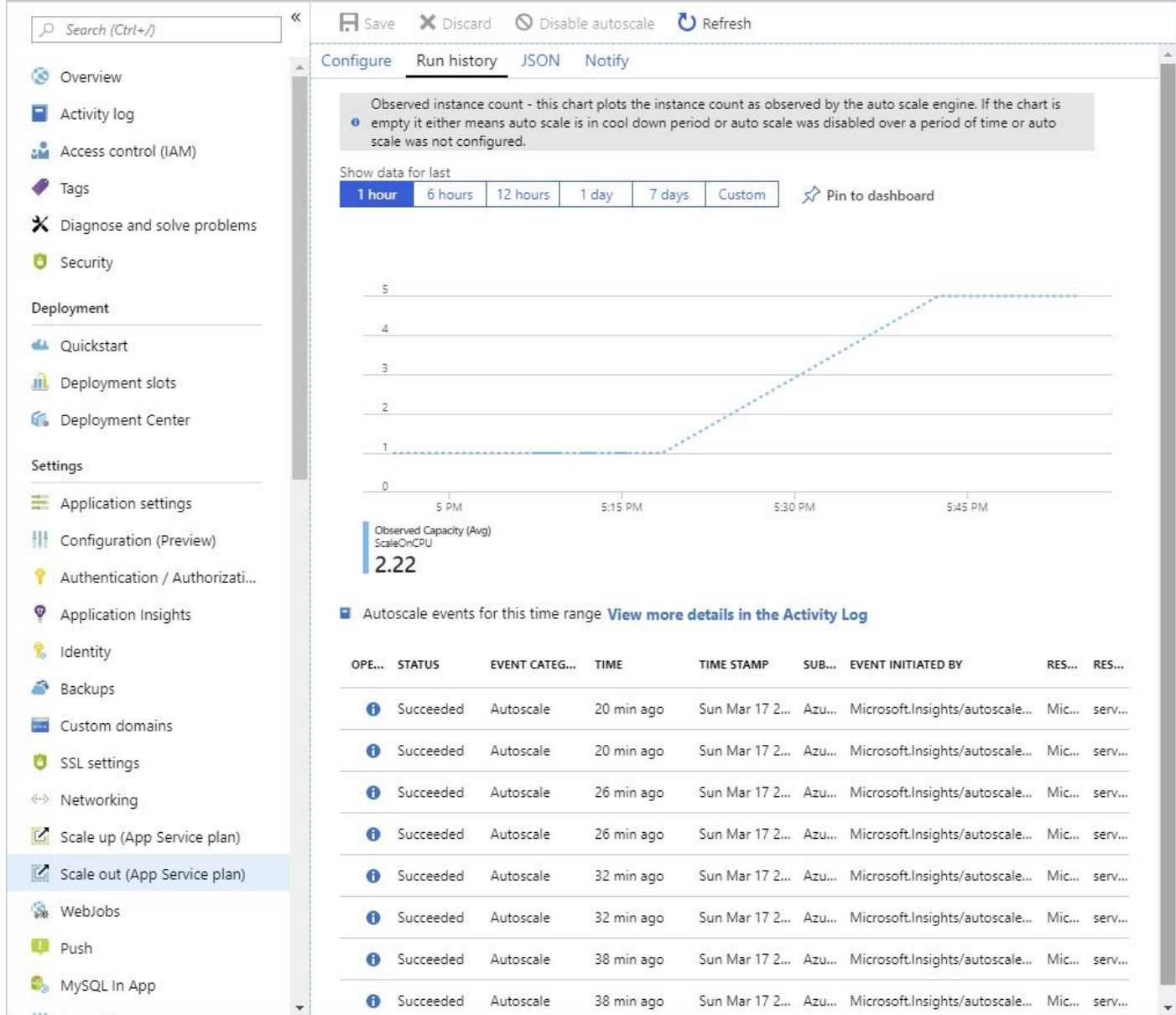
\* Threshold 70

\* Duration (in minutes) 10

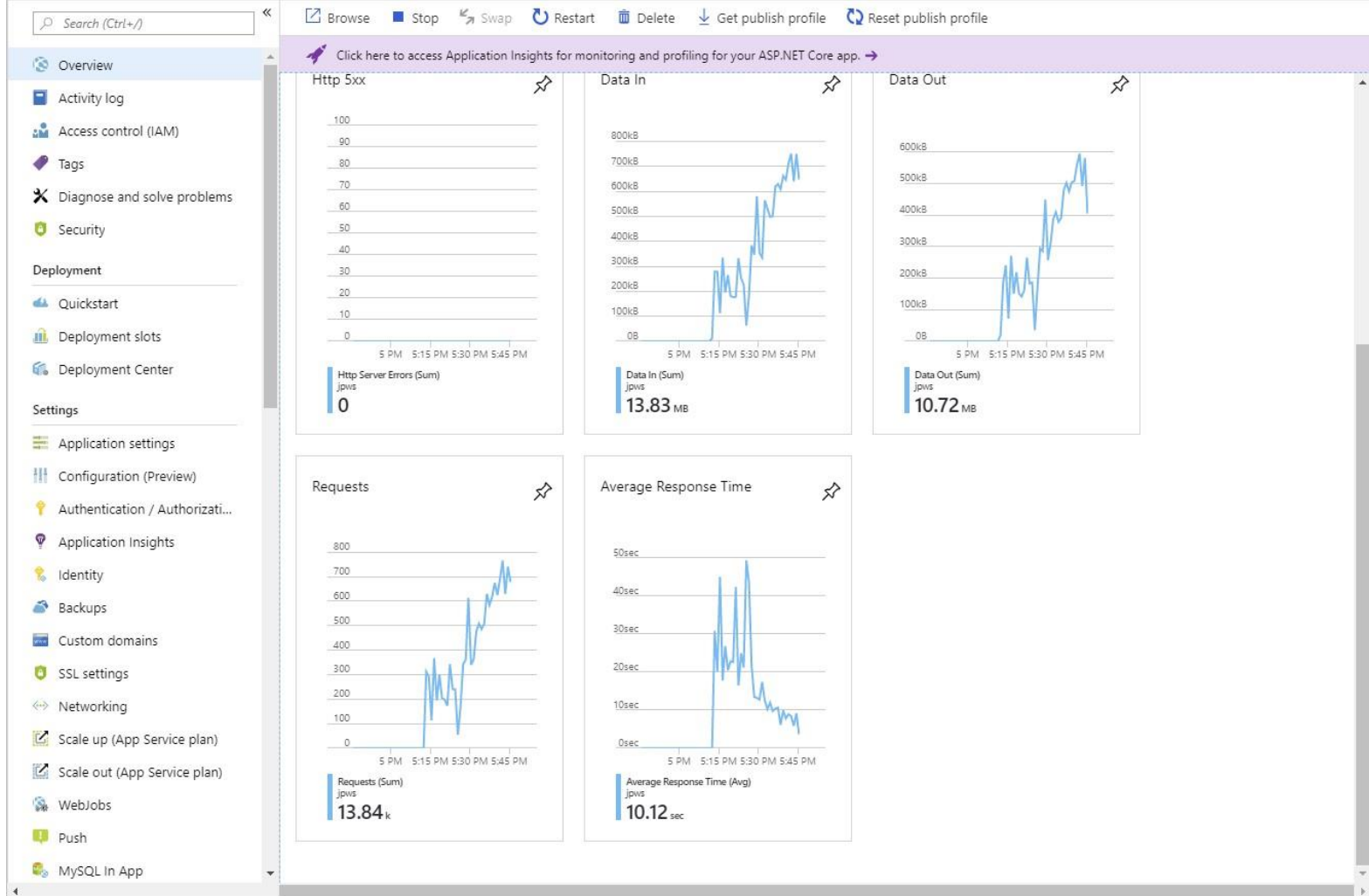
Add

## Monitor autoscaling activity

The Azure portal enables you to track when autoscaling has occurred through the **Run history** chart. This chart shows how the number of instances varies over time, and which autoscale conditions caused each change.



You can use the **Run history** chart in conjunction with the metrics shown on the **Overview** page to correlate the autoscaling events with resource utilization.



## Disable autoscaling

To disable autoscaling, you can select the **Disable autoscale** option on the **Scale out** page of the App Service Plan. Autoscaling is also disabled if you delete all of the autoscale conditions, including the default condition.

Save Discard **Disable autoscale** Refresh

Configure Run history JSON Notify

Autoscale setting name ScaleOnCPU

Resource group testrg

Instance count 1

**Default** Auto created scale condition

Delete warning The very last or default recurrence rule cannot be deleted. Instead, you can disa

Scale mode ☒ Scale based on a metric ☐ Scale to a specific instance count

Next unit: Exercise - Autoscale and monitor a web app

Continue

# Exercise - Autoscale and monitor a web app

20 minutes

Autoscaling is a key part of ensuring that a system remains available and responsive.

You want to implement autoscaling for the hotel reservation system web app, based on the CPU usage of the host. When the CPU utilization rises over a specific threshold, the web app will scale out. If the CPU usage drops, the web app will scale back in again.

In this unit, you'll set up the web app and run a test client application that imposes a load on the web app. You'll see the types of errors that can occur when the web host becomes overloaded. Next, you'll configure autoscaling for the web app and run the test client again. You'll monitor the autoscale events that occur, and examine how the system responds to the workload.

## Setup

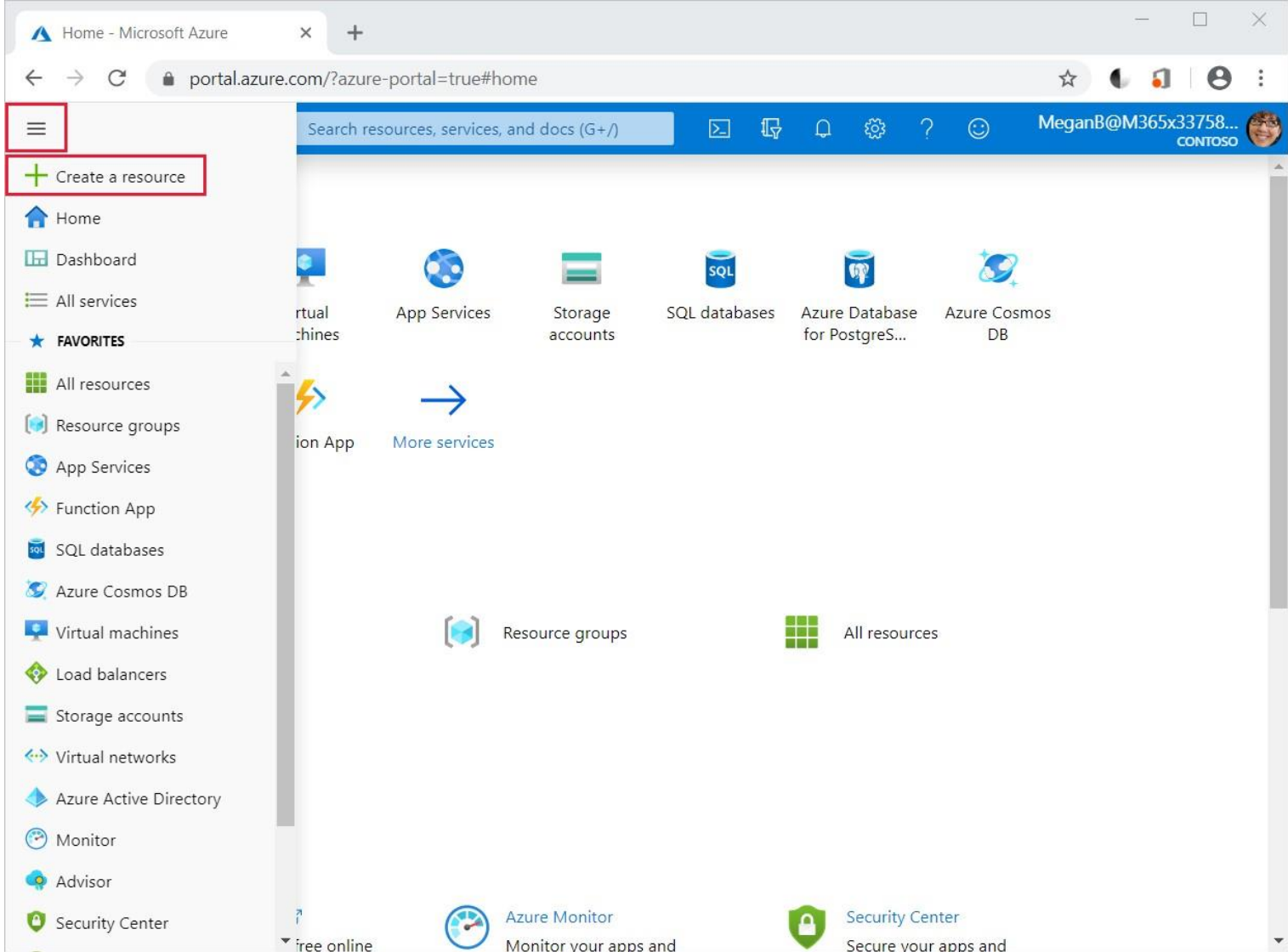
The web app for the hotel reservation system implements a web API. The web API exposes HTTP POST and GET operations that create and retrieve customer's bookings. In this exercise, the bookings aren't saved, and the GET operation simply retrieves dummy data.

The exercise also runs a client app that simulates a number of users issuing POST and GET operations simultaneously. This app provides the workload for testing how the web app autoscales.

### Important

You need your own Azure subscription to run this exercise and you may incur charges. If you don't already have an Azure subscription, create a [free account](#) before you begin.

1. Sign in to the [Azure portal](#).
2. From the Azure portal menu, select **Create a resource**.



3. Select **Web**, and then select **Web App**

Azure Marketplace

See all

Featured

See all

Get started

Recently created

AI + Machine Learning

Analytics

Blockchain

Compute

Containers

Databases

Developer Tools

DevOps

Identity

Integration

Internet of Things

Media

Mixed Reality

IT & Management Tools

Networking

Software as a Service (SaaS)

Security

Storage

Web

Web App

Quickstart tutorial

Logic App

Quickstart tutorial

App Service Plan

Learn more

App Service Environment

Learn more

API Management

Quickstart tutorial

CDN

Quickstart tutorial

Media Services

Quickstart tutorial

Azure Cognitive Search

Quickstart tutorial

Web App for Containers

Quickstart tutorial

Sitecore® Experience Cloud

Learn more

4. Specify the values in the following table for the properties of the web app.

#### Note

The web app must have a unique name. We suggest using something like **<your name or initials>hotelsystem**. Use this name wherever you see **<your-webapp-name>** in this exercise.

Property	Value
Name	<your-webapp-name >
Subscription	Select the Azure subscription you'd like to use for this exercise
Resource Group	Create a new resource group called <b>mslearn-autoscale</b>
OS	Windows
Publish	Code

5. Click **Create**.

6. Open the Cloud Shell in the Azure portal and run the following command to download the source code for the hotel reservation system:

```
bash
git clone https://github.com/MicrosoftDocs/mslearn-hotel-reservation-system.git
```

7. Move to the **mslearn-hotel-reservation-system/src** folder:

```
bash
cd mslearn-hotel-reservation-system/src
```

8. Build the apps for the hotel system. There are two apps; a web app that implements the web API for the system, and a client app that you'll use for load testing the web app:

```
bash
dotnet build
```

9. Prepare the HotelReservationSystem web app for publishing:

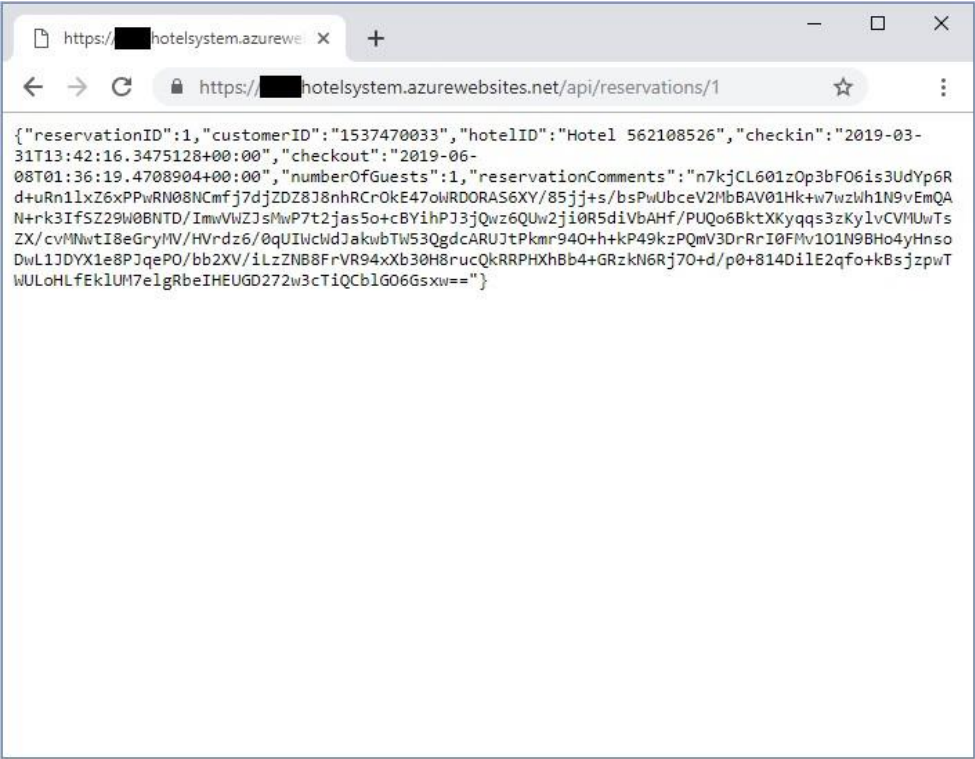
```
bash
cd HotelReservationSystem
dotnet publish -o website
```

10. Go to the **website** folder containing the published files, zip them up, and deploy them to the web app you created in the previous task. Replace `<your-webapp-name>` with the name of your web app.

```
bash
cd website
zip website.zip *
az webapp deployment source config-zip --src website.zip --name <your-webapp-name> --resource-group mslearn-autoscale
```

## Test the web app before configuring autoscaling

1. Using a web browser, go to `https://<your-webapp-name>.azurewebsites.net/api/reservations/1`. Visiting this URL sends a GET request to the web API to retrieve the details of reservation number 1. You should see a result similar to the one shown below. The response contains a JSON document with the details of the booking. Remember that this is dummy data:




2. Return to the Cloud Shell and move to the `~/mslearn-hotel-reservation-system/src/HotelReservationSystemTestClient` folder:

```
bash
cd ~/mslearn-hotel-reservation-system/src/HotelReservationSystemTestClient
```

3. Edit the App.config file in this folder using the **code** editor:


```
bash
code App.config
```

4. Uncomment the line that specifies the **ReservationsServiceURI** and replace the value with the URL of your web app. The file should like the example shown below:


text	
<pre>&lt;?xml version="1.0" encoding="utf-8" ? &gt; &lt;configuration &gt;   &lt;appSettings&gt;     &lt;add key="NumClients" value="100" /&gt;     &lt;add key="ReservationsServiceURI" value="https://&lt;your-webapp-name&gt;.azurewebsites.net/" /&gt;     &lt;add key="ReservationsServiceCollection" value="api/reservations" /&gt;   &lt;/appSettings&gt; &lt;/configuration &gt;</pre>	

The **NumClients** setting in this file specifies the number of simultaneous clients that will attempt to connect to the web app and perform work. The work consists of creating a reservation, and then running a query to fetch the details of a reservation – all of the data used is fake and is not actually persisted anywhere. Leave this value set to 100.

5. Save the file and close the code editor.
6. Rebuild the test client app with the new configuration:

bash	
<pre>dotnet build</pre>	

7. Run the client app. You'll see a number of messages appear as the clients start running, make reservations, and run queries. Allow the system to run for a couple of minutes. The responses will be slow, and soon the client requests will start to fail with HTTP 408 (Timeout) errors:

bash	
<pre>dotnet run</pre>	



```
Bash
pv274e4QIf3Qa0btYyDyhT5mik1G1IqpygmZNT/UAwIHnJ+hwey9RyWEnw80QWMEsNF-iMI5CV495MavXQVBN8=
Client Client45 making reservation 1109836230
Client Client53 querying reservation: Reservation 351998021, CustomerID 795210174, HotelID Hotel 182
Comments: j7EEtVYwG0LUFhvEG022xSuPL42v1aCxFKqNbmBJMGMO0LXpCJ0upi3LMFbF+c7iKTHYZJBVOVgqOMkmoUoGnnEdrKfF
4sTb8Z4CjR7s73oX/Bc6Qj1vY2H29DkiE7eEyoZkNeX5SGodZ/gZJNveuBC0G91fvUKqyC2ywwXGNZFD15Y0VApjd4RKk+3S6N/ol
eXWtVnDKSfxxzrq/s/eP3NivpzK806cAA3A+2tNBuE0k7QvWqf8ltU+SIImNZ7IBY+E2erlGloAtqBLY3N8MdgxDys015AFTaaJuqA
ghHB7yONiFDj+KUMqWjtlp13cZuFox3ysaRUP9uqBk9hvtD/iMmY6ugz0Kh09MoP3Qqe4Cwfj9lH6y4Ha0inGnVp4hk/45iYKy6V
offswPL5noI8qjozoxnnah13mo4PEgBmz+95gZATqxDvyyUxEGzk07/hTfqjQg6PeHVL5h41GYIfCdgiAaFyrhFmJQuKZeQzIqCz
Client Client53 making reservation 1685198217
Client Client33 querying reservation: Reservation 589443144, CustomerID 1034084455, HotelID Hotel 21
Comments: LtR9w1xGqiyzdmOROY3qh80EJSD+qVkud7P57++DtDoUzwnNwRskAJyT78joFsvCzH2JuiEdYJNewAdx8+T0zS5dv9L
6VLHocD1Po1HEBMoi5Xng/jellYON7BPYMBM2teXa2Feq0t3mC4Rw4Hx/x6hu2cw90THSEGGcvQZJA/sfzvYYBayjOgmJuQMbrn2
UTiaf+vBQ8jzbTeLYoeSHf6V07U+PXu9u3Wt7yd0YkdCbJN/np1LEfWBEKiGnDakyXv9+QowUgiWC+6mRIR9qy5bDFjS1JYc6Tt
Client Client33 making reservation 2102474681
Client Client43 querying reservation: Reservation 160354436, CustomerID 766230361, HotelID Hotel 345
Comments: oWps80CRnI4hGcYT6KyF3jdSSG4Y5Y516ebMqfwZonueMoazeGhzB+eFwy/qObRlbdqdz4UiyhNM03imtDptkTiZVf
Drq+x4/BlgQDmiQhLGeccmQd5KAP7+0EDBKfYu1Mncregj22qHakPJiyqUk1lnAvtxvhkuFq79wNFPQOWN07hJigVMHwmEcm3mu
WYCuE34SOJGCEsb3obuUtDabXezRVPDiZuprW9fk+SV+AjmEj4xkPxJZKTOjGYslw9otdk7Z+0dUq+9K/JuPhkN/hv54p10mtf40s
Client Client43 making reservation 1480437344
Client Client16 querying reservation: Reservation 272106587, CustomerID 1539781140, HotelID Hotel 13
Comments: xNsgr8mm/sdluXI3LzjfifbvW9Qpmjr8fYdjjeoPKOzD+O4TMKRWailioS4/yBREJkwqmucoAtXoRGA/HtaYTrcx9xwcl
p2cx6eAaya0TwbttNsSiBh0afPHtjwWlByPseQvCvwCDFAwHtQqHeXCIxEXU/n3Io0Fy5ZXBm001QGKYYKn3LbE+tf+f++EQ20p0
q9ks8M/fq8F+biBvpQtKdNkVtMrja106+AA+p3WYK4A0mZWQYPhLv4eIbA6Rif3Bng3R6LDW7bh2nLyhxM5dWls8R7sn9et9tuB6d
SRF7/woT3yltq8P3GCq1vmm0jBdzzLGB+XfH0Fr+LPAZL1QuI0TB1QzD9mK711L5Uelwh5cin3mMEsNiss+fuT0kKjQdY5XqRyzvg
1lVsJjH7ecUKyYrZNHtNjVq67gyv6emsCRGICRuQwyJWPQ1rjMo0N3wMCyCnD6VRK2hQCWabFSG+9BxiET5w9HRj5XYTXPF1RhyL
Client Client16 making reservation 1072786692
Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
Client Client8 querying reservation: Reservation 798222214, CustomerID 609864951, HotelID Hotel 8613
Comments: DK6pMVElQvM81tVef2zxN2v3Gy1Jve3jotCDZkoKmxgYDATOz+R0h63Us0CPghD40Pg9sCgmDGFuEhkVERBc8zSjTY
6900svN0kjyGzRC8zwiUVsPB0JJiD0QpMnFIWag8MwTZY+DUakUoypaP5i0uHi2XRqK8BZ0k3TBImLmuVPEVWHORxkc+Lkm2Qc
7ii6KpvfRaf
Client Client8 making reservation 993123983
Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
```

8. Press Enter to stop the client application.

## Enable autoscaling and create a scale condition

1. Return to your web app in the Azure portal.
2. Under **Settings** click **Scale out (App Service plan)** and then click **Enable autoscale**
3. In the default autoscale rule, verify that the scale mode is set to **Scale based on a metric** and then click **Add a rule**

Search (Ctrl+/)

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Settings

Apps

File system storage

Networking

Scale up (App Service plan)

Scale out (App Service plan)

Resource explorer

Properties

Locks

Export template

Monitoring

Alerts

Metrics

SaveDiscardDisable autoscaleRefresh

ConfigureRun historyJSONNotify

\* Autoscale setting name

Resource group

Default

Auto created scale condition

Delete warning

The very last or default recurrence rule cannot be deleted. Instead, you can disable autoscale to turn off autoscale.

Scale mode

☒ Scale based on a metric

☐ Scale to a specific instance count

Scale out and scale in your instances based on metric. For example: 'Add a rule that increases instance count by 1 when CPU percentage is above 70%'

Rules

It is recommended to have at least one scale in rule

+ Add a rule

Instance limits

Minimum

Maximum

Default

1

2

1

Schedule

This scale condition is executed when none of the other scale condition(s) match

+ Add a scale condition

4. Add a rule that increases the instance count by one if the average CPU utilization across all instances in the web site exceeds 50 percent in the preceding five minutes. This is a scale-out rule.

Scale rule

Metric source

Current resource (hotelsystemplan)

Resource type

App Service plans

Resource

hotelsystemplan

Criteria

\* Time aggregation ⓘ

Average

\* Metric name

CPU Percentage

1 minute time grain

\* Time grain statistic ⓘ

Average

\* Operator

Greater than

\* Threshold

50

\* Duration (in minutes) ⓘ

5

Action

\* Operation

Increase count by

\* Instance count

1

\* Cool down (minutes) ⓘ

5

Add

5. Click **Add a rule** again. Add a rule that reduces the instance count by one if the average CPU utilization across all instances in the web site drops below 30 percent in the preceding five minutes. This is a scale-in rule. Remember that it's good practice to define scale rules in pairs.

Scale rule

Metric source

Current resource (hotelsystemplan)

Resource type

App Service plans

Resource

hotelsystemplan

Criteria

\* Time aggregation ⓘ

Average

\* Metric name

CPU Percentage

1 minute time grain

\* Time grain statistic ⓘ

Average

\* Operator

Less than

\* Threshold

30

\* Duration (in minutes) ⓘ

5

Action

\* Operation

Decrease count by

\* Instance count

1

\* Cool down (minutes) ⓘ

5

Add

6. In the **Default** auto scale condition window, in the **Instance limits** section, set the **Maximum** instance count to five. Name the Autoscale setting **ScaleOnCPU**, and then click **Save**.

Save Discard Disable autoscale Refresh

Configure Run history JSON Notify

Autoscale setting name **ScaleOnCPU**

Resource group [REDACTED]

Instance count 1

**Default** Auto created scale condition

Delete warning ⓘ The very last or default recurrence rule cannot be deleted. Instead, you can disable autoscale to turn off autoscale.

Scale mode ☒ Scale based on a metric ☐ Scale to a specific instance count

Scale out

When	Condition	Action
hotelsystemplan	(Average) CpuPercentage > 50	Increase instance count by 1

Scale in

When	Condition	Action
hotelsystemplan	(Average) CpuPercentage < 30	Decrease instance count by 1

+ Add a rule

Instance limits Minimum ⓘ Maximum ⓘ Default ⓘ

1 5 1

Schedule **This scale condition is executed when none of the other scale condition(s) match**

+ Add a scale condition

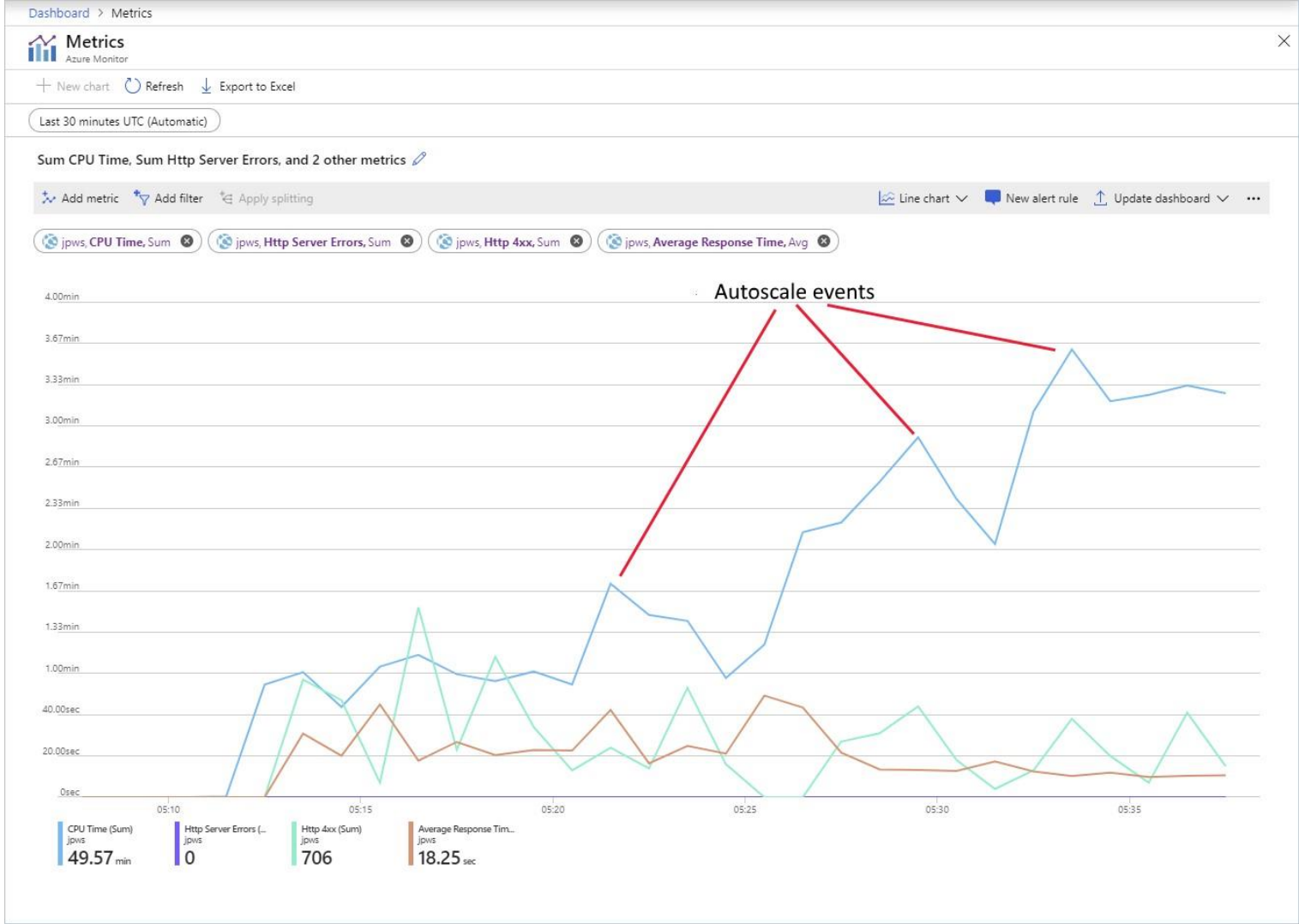
## Monitor autoscale events

1. Return to the Cloud Shell, go to the `~/hotelsystem/HotelReservationSystemTestClient` folder, and run the test client again:

```
bash

cd ~/hotelsystem/HotelReservationSystemTestClient
dotnet run
```

2. While the client app is running, switch back to the Azure portal showing the autoscale settings for the web app, and click **Run history**. Under **show data for last**, click **1 hour**. Initially, the chart will be empty as it will take several minutes for autoscaling to kick in.
3. While you're waiting for autoscaling events to occur, go to the pane for your web service (not the service plan), and under **Monitoring** click **Metrics**.
4. Add the following metrics to the chart, set the time range to **Last 30 minutes** and then pin the chart to the current dashboard:
  - CPU Time. Select the Sum aggregation
  - Http Server Errors. Select the Sum aggregation.
  - Http 4.xx. Select the Sum aggregation.
  - Average Response Time. Select the Avg aggregation.
5. Allow the system to stabilize, and note the CPU Time, the number of HTTP 4.xx errors, and the average response time. Before the system autoscales, you should see a significant number of HTTP 4.xx errors (these are HTTP 408 Timeout errors), and that the average response time is several seconds. There may be the occasional HTTP Server Error, depending on how the web server is coping with the burden.
6. After 10 minutes or so you should see the following trends in this chart:
  - The CPU Time jumps up.
  - The number of HTTP 4.xx errors diminishes.
  - The average response time drops.

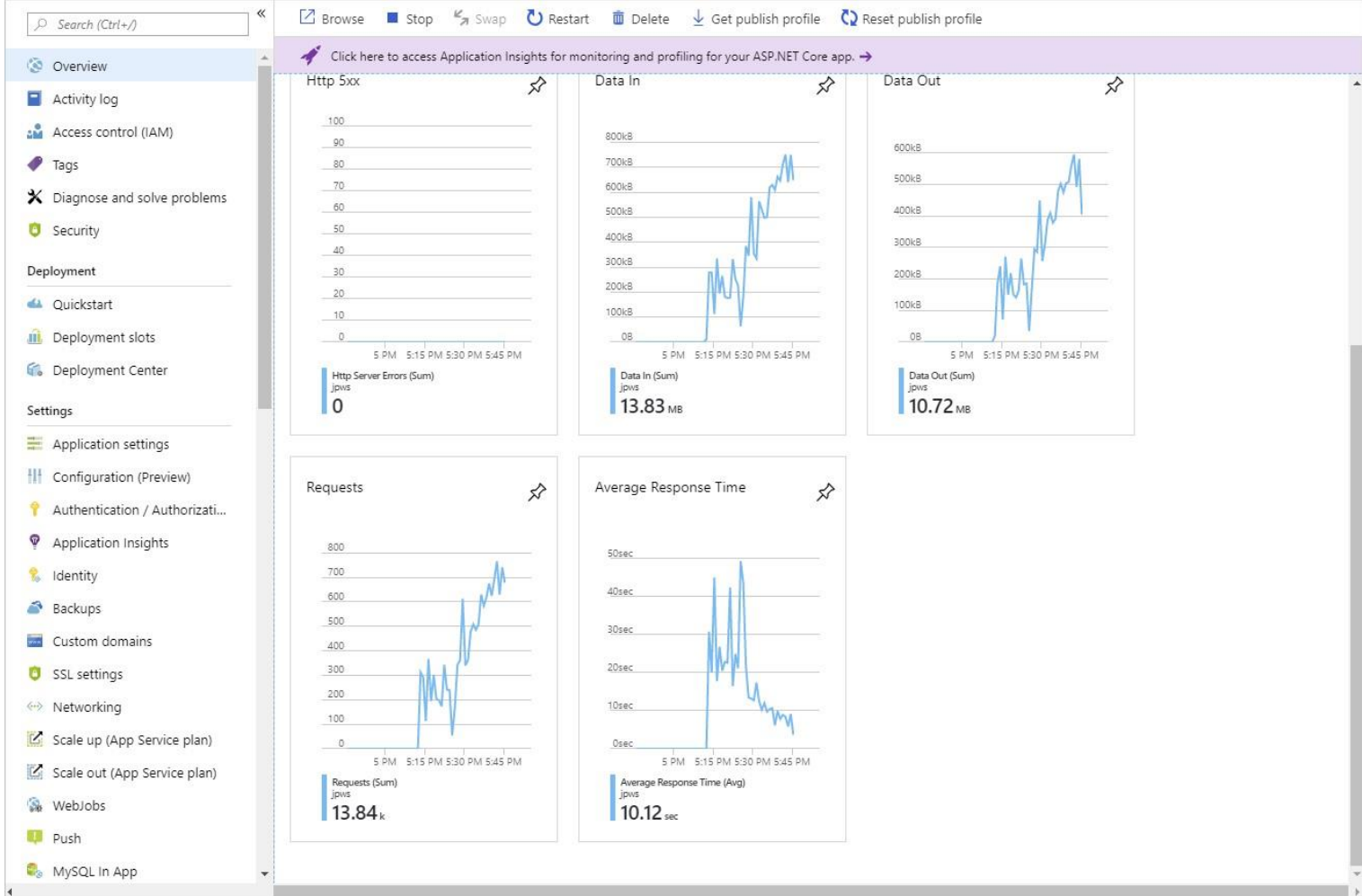


Each major spike in the CPU Time indicates that more CPU processing power has become available. This is a result of autoscaling.

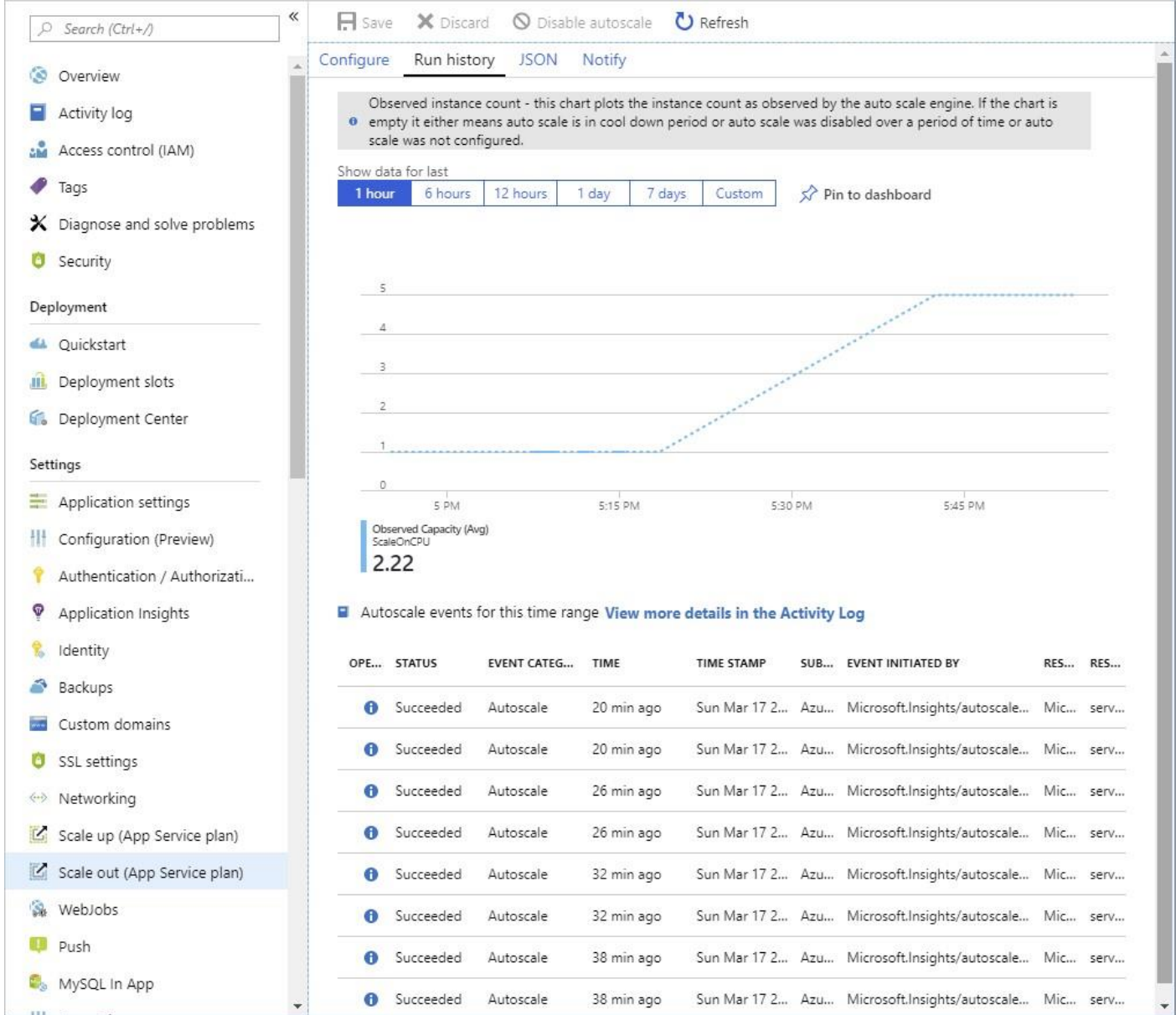
7. Return to the **Overview** page for the web app and examine the charts. These charts should indicate the following trends:

- The Data In, Data Out, and Requests metrics have increased.
- The Average Response Time has dropped.





8. Click **Scale out (App Service plan)** and then click **Run history**. Click **1 hour**. The graph should now indicate that autoscaling has occurred. The number of instances will have increased (it may have reached five, depending on how long the client app has been running), and you should see a number of autoscale events reported.



### Note

The autoscale events are reported in pairs. The first event occurs when autoscaling has triggered an increase in the number of instances. The second event occurs when autoscaling has completed.

9. Return to the Cloud Shell. You should see that the app is running more quickly, and far fewer requests are failing.

10. Press enter to stop the app.

11. After several minutes, if you examine the run history of the App Service Plan, you'll see the number of instances drop. This is the result of the scale-in rule releasing these resources.

You configured autoscaling for the hotel reservation system. The system scaled out when the total CPU usage across all instances hosting the web site exceeded 50 percent in a five-minute period. The system scaled back in when the total CPU utilization dropped below 30 percent, again for a five-minute period. You subjected the hotel reservation system to a test load, and monitored when autoscaling occurred.

## Next unit: Summary

Continue