


# Refactor a Linux app to multiple regions using Azure App Service, Traffic Manager, and Azure Database for MySQL

10/11/2018 • 11 minutes to read • Contributors 

## In this article

[Business drivers](#)

[Migration goals](#)

[Solution design](#)

[Current architecture](#)

[Proposed architecture](#)

[Migration process](#)

[Prerequisites](#)

[Scenario steps](#)

[Step 1: Provision Azure App Service](#)

[Step 2: Set up Traffic Manager](#)

[Step 3: Provision Azure Database for MySQL](#)

[Step 4: Migrate the database](#)

[Step 5: Set up GitHub](#)

[Step 6: Configure the web apps](#)

[Clean up after migration](#)

[Review the deployment](#)

This article shows how the fictional company Contoso refactors a two-tier Linux-based Apache MySQL PHP (LAMP) app, migrating it from on-premises to Azure using Azure App Service with GitHub integration and Azure Database for MySQL.

osTicket, the service desk app used in this example is provided as open source. If you'd like to use it for your own testing purposes, you can download it from [GitHub](#).

## Business drivers

The IT Leadership team has worked closely with business partners to understand what they want to achieve:

- **Address business growth.** Contoso is growing and moving into new markets. It needs additional customer service agents.
- **Scale.** The solution should be built so that Contoso can add more customer service agents as the business scales.
- **Improve resiliency.** In the past issues with the system affected internal users only. With the new business model, external users will be affected, and Contoso need the app up and running at all times.

## Migration goals

The Contoso cloud team has pinned down goals for this migration, in order to determine the best migration method:

- The application should scale beyond current on-premises capacity and performance. Contoso is moving the application to take advantage of Azure's on-demand scaling.

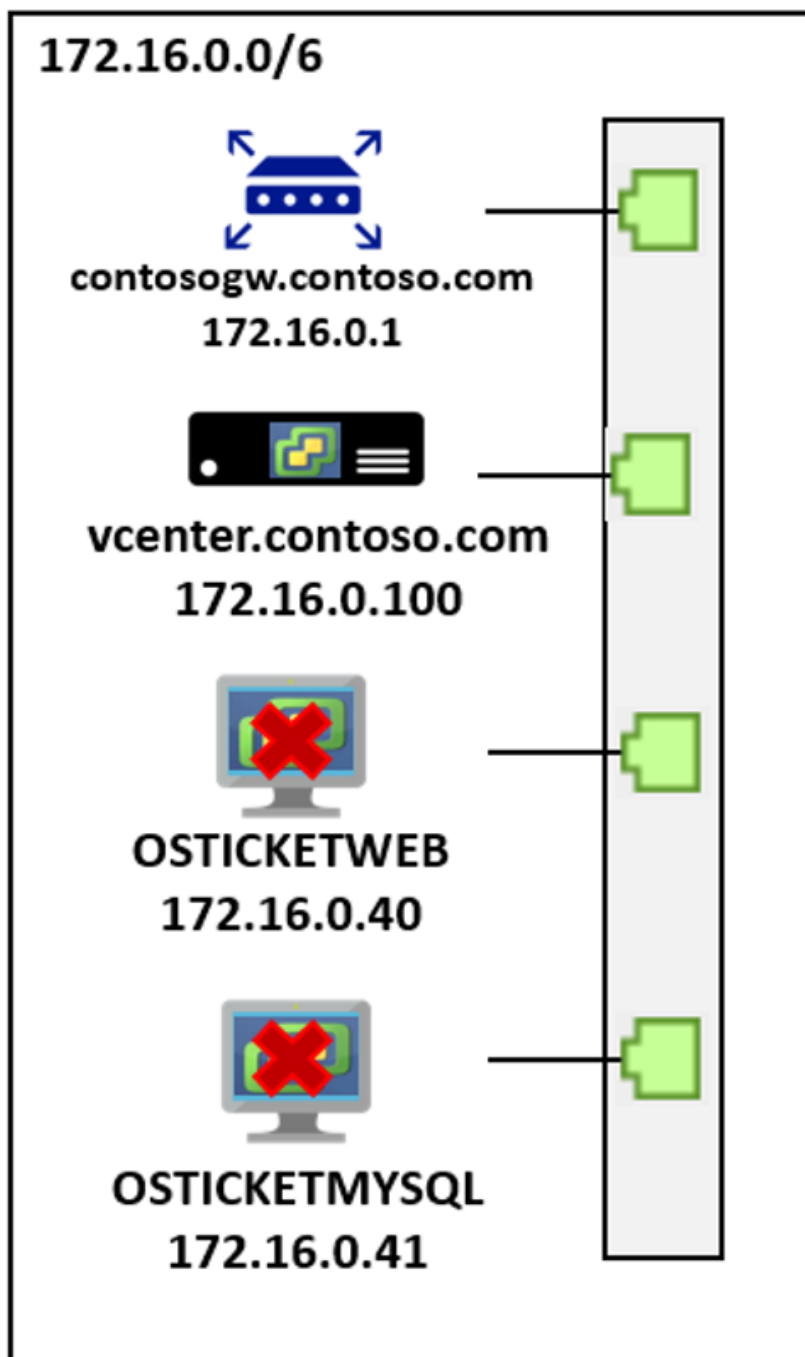
- Contoso wants to move the app code base to a continuous delivery pipeline. As app changes are pushed to GitHub, Contoso wants to deploy those changes without tasks for operations staff.
- The application must be resilient with capabilities for growth and failover. Contoso wants to deploy the app in two different Azure regions, and set it up to scale automatically.
- Contoso wants to minimize database admin tasks after the app is moved to the cloud.

## Solution design

After pinning down their goals and requirements, Contoso designs and reviews a deployment solution, and identifies the migration process, including the Azure services that will be used for the migration.

## Current architecture

- The app is tiered across two VMs (OSTICKETWEB and OSTICKETMYSQL).
- The VMs are located on VMware ESXi host **contosohost1.contoso.com** (version 6.5).
- The VMware environment is managed by vCenter Server 6.5 (**vcenter.contoso.com**), running on a VM.
- Contoso has an on-premises datacenter (contoso-datacenter), with an on-premises domain controller (**contosodc1**).



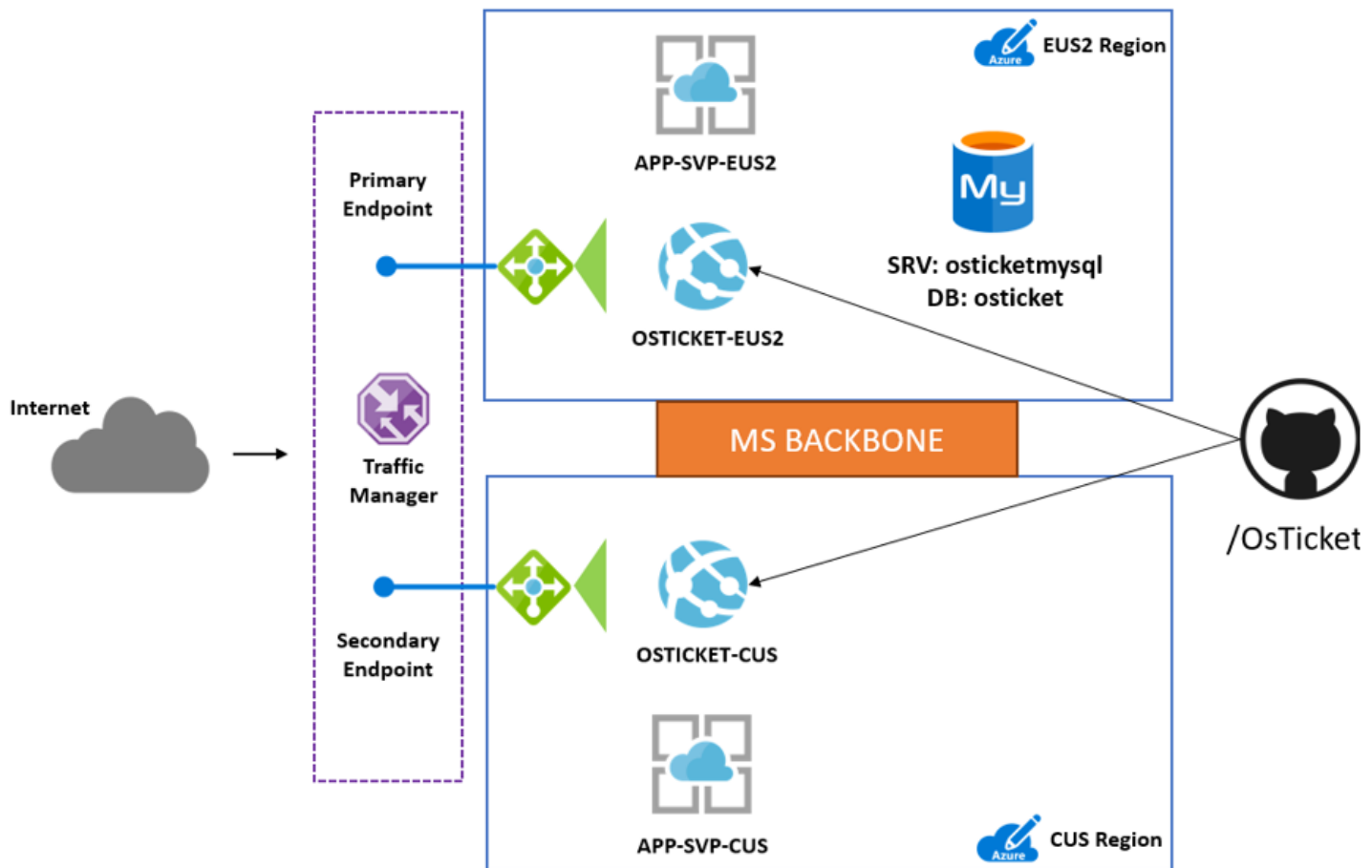
On-premises

## Proposed architecture

Here's the proposed architecture:

- The web tier app on OSTICKETWEB will be migrated by building an Azure App Service in two Azure regions. Azure App Service for Linux will be implemented using the PHP 7.0 Docker container.
- The app code will be moved to GitHub, and the Azure App Service web app will be configured for continuous delivery with GitHub.
- Azure App Servers will be deployed in both the primary (East US 2) and secondary (Central US) region.
- Traffic Manager will be set up in front of the two web apps in both regions.
- Traffic Manager will be configured in priority mode to force the traffic through East US 2.
- If the Azure App Server in East US 2 goes offline, users can access the failed over app in Central US.

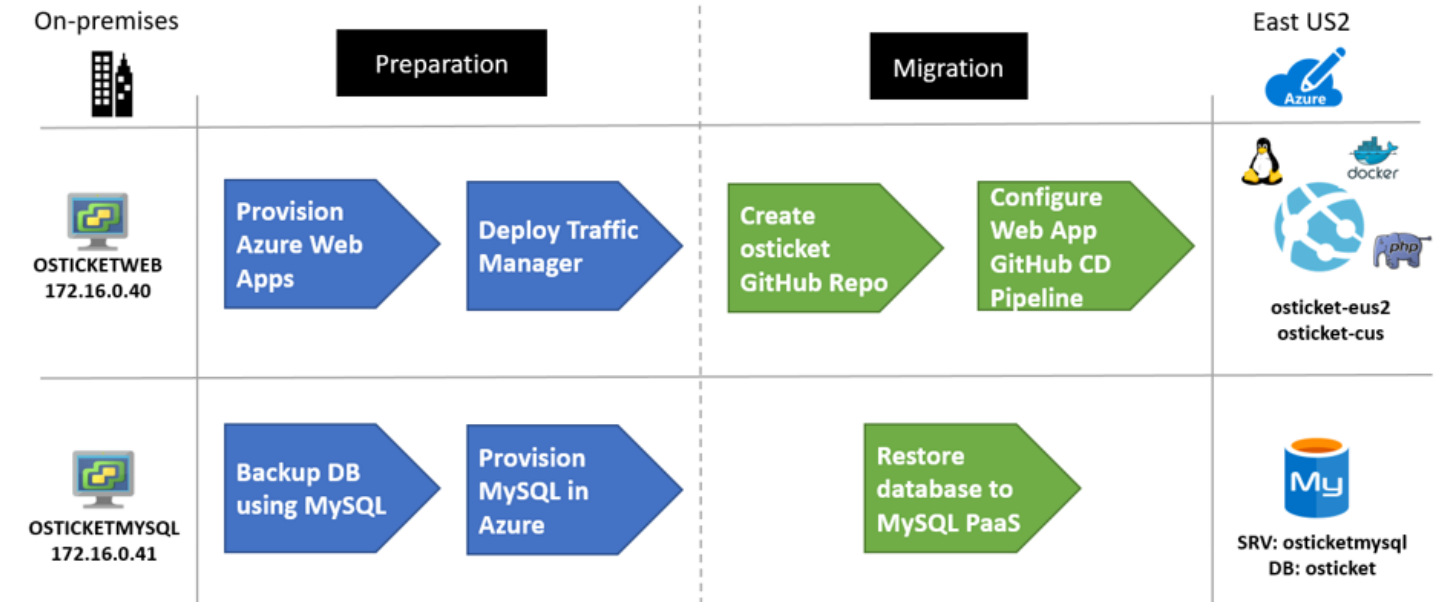
- The app database will be migrated to the Azure Database for MySQL service using MySQL Workbench tools. The on-premises database will be backed up locally, and restored directly to Azure Database for MySQL.
- The database will reside in the primary East US 2 region, in the database subnet (PROD-DB-EUS2) in the production network (VNET-PROD-EUS2):
- Since they're migrating a production workload, Azure resources for the app will reside in the production resource group **ContosoRG**.
- The Traffic Manager resource will be deployed in Contoso's infrastructure resource group **ContosoInfraRG**.
- The on-premises VMs in the Contoso datacenter will be decommissioned after the migration is done.



## Migration process

Contoso will complete the migration process as follows:

1. As a first step, Contoso admins set up the Azure infrastructure, including provisioning Azure App Service, setting up Traffic Manager, and provisioning an Azure Database for MySQL instance.
2. After preparing the Azure, they migrate the database using MySQL Workbench.
3. After the database is running in Azure, they up a GitHub private repository for Azure App Service with continuous delivery, and load it with the osTicket app.
4. In the Azure portal, they load the app from GitHub to the Docker container running Azure App Service.
5. They tweak DNS settings, and configure autoscaling for the app.



## Azure services

Service	Description	Cost
<a href="#">Azure App Service</a>	The service runs and scales applications using the Azure PaaS service for websites.	Pricing is based on the size of the instances, and the features required. <a href="#">Learn more.</a>
<a href="#">Traffic Manager</a>	A load balancer that uses DNS to direct users to Azure, or external websites and services.	Pricing is based on the number of DNS queries received, and the number of monitored endpoints. <a href="#">Learn more.</a>
<a href="#">Azure Database for MySQL</a>	The database is based on the open-source MySQL Server engine. It provides a fully managed, enterprise-ready community MySQL database, as a service for app development and deployment.	Pricing based on compute, storage, and backup requirements. <a href="#">Learn more.</a>

## Prerequisites

Here's what Contoso needs to run this scenario.

Requirements	Details
<b>Azure subscription</b>	<p>Contoso created subscriptions earlier in this article series. If you don't have an Azure subscription, create a <a href="#">free account</a>.</p> <p>If you create a free account, you're the administrator of your subscription and can perform all actions.</p> <p>If you use an existing subscription and you're not the administrator, you need to work with the admin to assign you Owner or Contributor permissions.</p>
<b>Azure infrastructure</b>	Contoso set up their Azure infrastructure as described in <a href="#">Azure infrastructure for migration</a> .

## Scenario steps

Here's how Contoso will complete the migration:

- ✓ **Step 1: Provision Azure App Service.** Contoso admins will provision web apps in the primary and secondary regions.
- ✓ **Step 2: Set up Traffic Manager.** They set up Traffic Manager in front of the web apps, for routing and load balancing traffic.
- ✓ **Step 3: Provision MySQL.** In Azure, they provision an instance of Azure Database for MySQL.
- ✓ **Step 4: Migrate the database.** They migrate the database using MySQL Workbench.
- ✓ **Step 5: Set up GitHub.** They set up a local GitHub repository for the app web sites/code.
- ✓ **Step 6: Deploy the web apps.** They deploy the web apps from GitHub.

## Step 1: Provision Azure App Service

Contoso admins provision two web apps (one in each region) using Azure App Service.

1. They create a web App resource in the primary East US 2 region (**osticket-eus2**) from the Azure Marketplace.
2. They put the resource in the production resource group **ContosoRG**.

**Web App**  
Create

\* App name  
osticket-eus2 ✓  
.azurewebsites.net

\* Subscription  
▼

\* Resource Group ⓘ  
☐ Create new ☒ Use existing  
ContosoRG ▼

\* OS

3. They create a new App Service plan in the primary region (**APP-SVP-EUS2**), using the standard size.

**New App Service Plan**  
Create a plan for the web app

\* App Service plan  
APP-SVP-EUS2 ✓

\* Location  
East US 2 ▼

\* Pricing tier  
S2 Standard >

4. They select a Linux OS with PHP 7.0 runtime stack, which is a Docker container.

Web App Create

\* App name  
osticket-eus2 ✓  
.azurewebsites.net

\* Subscription  
▼

\* Resource Group ⓘ  
☐ Create new ☒ Use existing  
ContosoRG ▼

\* OS Windows Linux Docker

\* App Service plan/Location  
APP-SVP-EUS2(East US 2) >

\* Runtime Stack  
PHP 7.0 ▼

5. They create a second web app (osticket-cus), and Azure App Service plan for the Central US region.

Web App Create

\* App name  
osticket-cus ✓  
.azurewebsites.net

\* Subscription  
▼

\* Resource Group ⓘ  
☐ Create new ☒ Use existing  
ContosoRG ▼

\* OS Windows Linux Docker

\* App Service plan/Location  
APP-SVP-CUS(Central US) >

\* Runtime Stack  
PHP 7.0 ▼

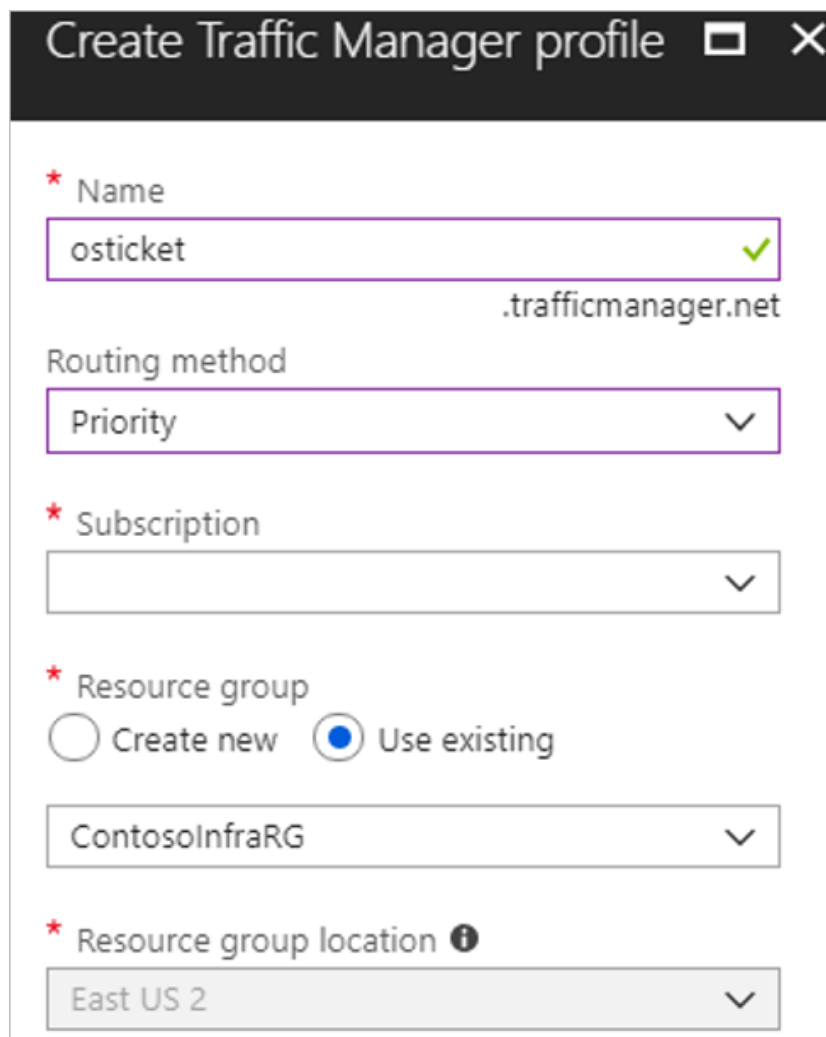
Need more help?

- Learn about [Azure App Service web apps](#).
- Learn about [Azure App Service on Linux](#).

## Step 2: Set up Traffic Manager

Contoso admins set up Traffic Manager to direct inbound web requests to the web apps running on the osTicket web tier.

1. They create a Traffic Manager resource (**osticket.trafficmanager.net**) from the Azure Marketplace. They use priority routing so that East US 2 is the primary site. They place the resource in their infrastructure resource group (**ContosoInfraRG**). Note that Traffic Manager is global and not bound to a specific location.




The screenshot shows the 'Create Traffic Manager profile' form. The form has a title bar with the text 'Create Traffic Manager profile' and window control icons. The form fields are as follows:

- Name:** A text input field containing 'osticket' with a green checkmark icon to its right. Below the input field, the text '.trafficmanager.net' is displayed.
- Routing method:** A dropdown menu with 'Priority' selected.
- Subscription:** A dropdown menu that is currently empty.
- Resource group:** A section with two radio buttons: 'Create new' (unselected) and 'Use existing' (selected). Below the radio buttons is a dropdown menu containing 'ContosoInfraRG'.
- Resource group location:** A dropdown menu with 'East US 2' selected. An information icon (i) is located to the right of the label.

2. Now, they configure Traffic Manager with endpoints. They add the East US 2 web app as the primary site (**osticket-eus2**), and the Central US app as secondary (**osticket-cus**).





Add endpoint
osticket

Type ⓘ

Azure endpoint

\* Name

osticket-eus2

Target resource type

App Service


\* Target resource

osticket-eus2

\* Priority

1

3. After adding the endpoints, they can monitor them.



osticket - Endpoints
Traffic Manager profile

Search (Ctrl+/)
Add Refresh

Overview
Activity log
Access control (IAM)
Tags

NAME	STATUS	MONITOR STATUS	TYPE	PRIORITY
osticket-eus2	Enabled	Checking endpoint	Azure endpoint	1
osticket-cus	Enabled	Checking endpoint	Azure endpoint	2

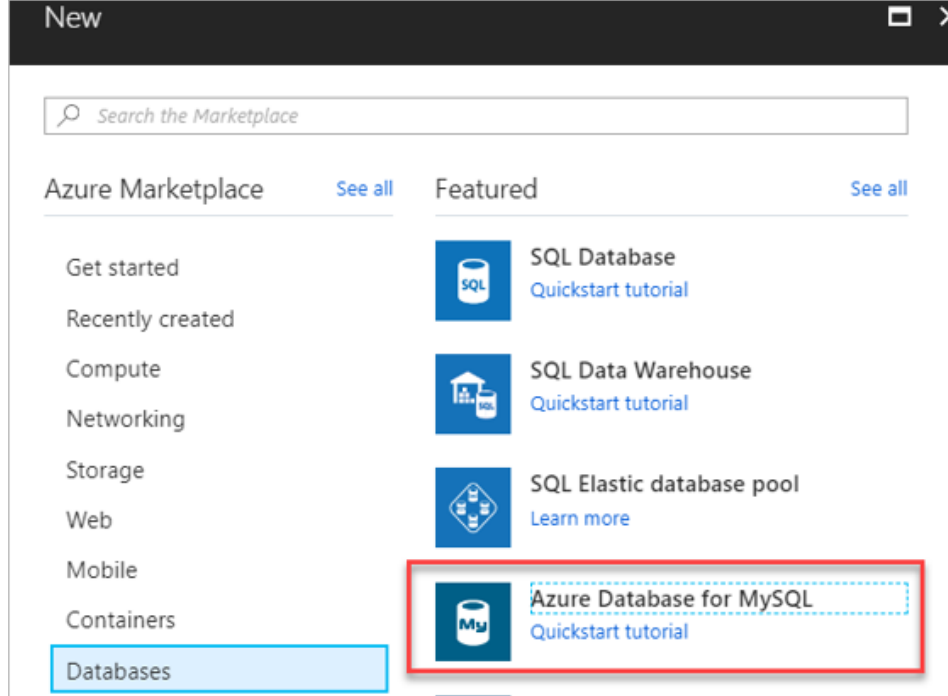
### Need more help?

- Learn about [Traffic Manager](#).
- Learn about [routing traffic to a priority endpoint](#).

## Step 3: Provision Azure Database for MySQL

Contoso admins provision a MySQL database instance in the primary East US 2 region.

1. In the Azure portal, they create an Azure Database for MySQL resource.



2. They add the name **contosoosticket** for the Azure database. They add the database to the production resource group **ContosoRG**, and specify credentials for it.
3. The on-premises MySQL database is version 5.7, so they select this version for compatibility. They use the default sizes, which match their database requirements.

4. For **Backup Redundancy Options**, they select to use **Geo-Redundant**. This option allows them to restore the database in their secondary Central US region if an outage occurs. They can only configure this option when they provision the database.

## Backup Redundancy Options - [Learn more details](#)

### Locally Redundant

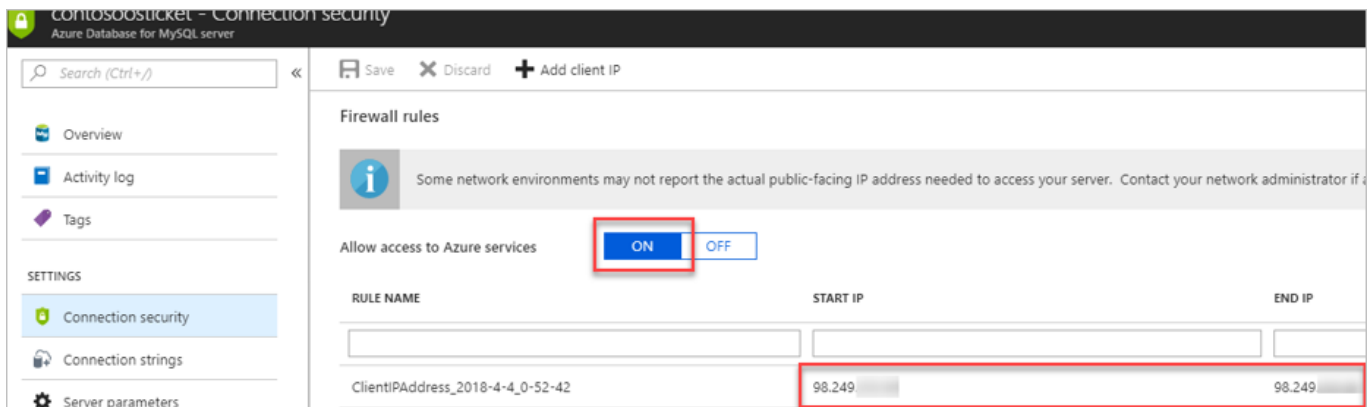
Recover from data loss  
within region

### Geo-Redundant

Recover from regional  
outage or disaster



5. They set up connection security. In the database > **Connection Security**, they set up Firewall rules to allow the database to access Azure services.
6. They add the local workstation client IP address to the start and end IP addresses. This allows the web apps to access the MySQL database, along with the database client that's performing the migration.



## Step 4: Migrate the database

Contoso admins migrate the database using backup and restore, with MySQL tools. They install MySQL Workbench, back up the database from OSTICKETMYSQL, and then restore it to Azure Database for MySQL Server.

### Install MySQL Workbench

1. They check the [prerequisites and downloads MySQL Workbench](#).
2. They install MySQL Workbench for Windows in accordance with the [installation instructions](#). The machine on which they install must be accessible to the OSTICKETMYSQL VM, and Azure via the internet.
3. In MySQL Workbench, they create a MySQL connection to OSTICKETMYSQL.

Connection Name: osticketmysql

Connection Remote Management System Profile

Connection Method: Standard TCP/IP over SSH Method to use to connect to the RDBMS

Parameters SSL Advanced

SSH Hostname: 172.16.0.43:22 SSH server hostname, with optional port number.

SSH Username: contosoadmin Name of the SSH user to connect with.

SSH Password: Store in Vault ... Clear SSH user password to connect to the SSH tunnel.

SSH Key File: ... Path to SSH private key file.

MySQL Hostname: 127.0.0.1 MySQL server host relative to the SSH server.

MySQL Server Port: 3306 TCP/IP port of the MySQL server.

Username: osticket Name of the user to connect with.

Password: Store in Vault ... Clear The MySQL user's password. Will be requested later if not set.

Default Schema: The schema to use as default schema. Leave blank to select it later.

4. They export the database as **osticket**, to a local self-contained file.

osticketmysql Data Export Advanced Options...

Object Selection Export Progress

Tables to Export

Exp	Schema
<input checked="" type="checkbox"/>	osticket
<input type="checkbox"/>	sys

Refresh Dump Structure and Dat Select Views Select Tables Unselect All

Objects to Export

☐ Dump Stored Procedures and Functions ☐ Dump Events ☐ Dump Triggers

Export Options

☐ Export to Dump Project Folder C:\Users\dan\OneDrive\Documents\dumps\Dump20180531 ...

Each table will be exported into a separate file. This allows a selective restore, but may be slower.

☒ Export to Self-Contained File C:\Users\...osticket.sql ...

All selected database objects will be exported into a single, self-contained file.

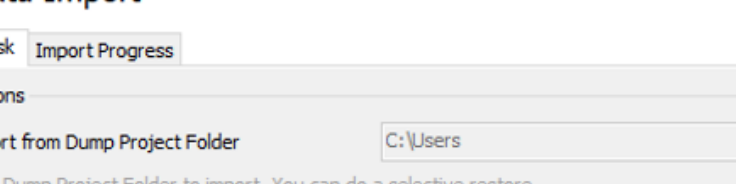
☐ Create Dump in a Single Transaction (self-contained file only) ☐ Include Create Schema

Export Completed Start Export

5. After the database has been backed up locally, they create a connection to the Azure Database for MySQL instance.

The screenshot shows the 'Setup New Connection' dialog box in MySQL Workbench. The 'Connection Name' is 'contosoosticket.mysql.database.azure.com'. The 'Connection Method' is 'Standard (TCP/IP)'. The 'Parameters' tab is selected, showing 'Hostname' as 'ntosoosticket.mysql.database.azure.com', 'Port' as '3306', 'Username' as 'contosoadmin@contosoosticket', and 'Password' as 'Store in Vault ...'. The 'Default Schema' is blank. A success message overlay reads: 'MySQL Workbench Successfully made the MySQL connection. Information related to this connection: Host: contosoosticket.mysql.database.azure.com, Port: 3306, User: contosoadmin@contosoosticket, SSL: enabled with AES256-SHA'.

6. Now, they can import (restore) the database in the Azure Database for MySQL instance, from the self-contained file. A new schema (osticket) is created for the instance.



contosoosticket.mysql.database.azure.com

## Data Import

Import from Disk | Import Progress

### Import Options

☐ Import from Dump Project Folder C:\Users

Select the Dump Project Folder to import. You can do a selective restore.

Load Folder Contents

☒ Import from Self-Contained File C:\Users\dan\osticket.sql

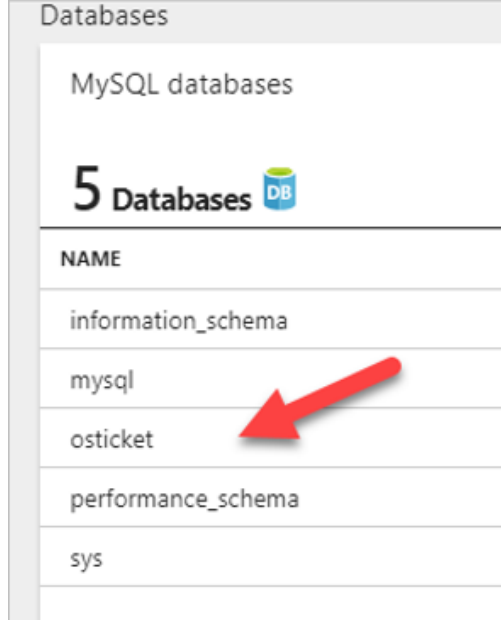
Select the SQL/dump file to import. Please note that the whole file will be imported.

### Default Schema to be Imported To

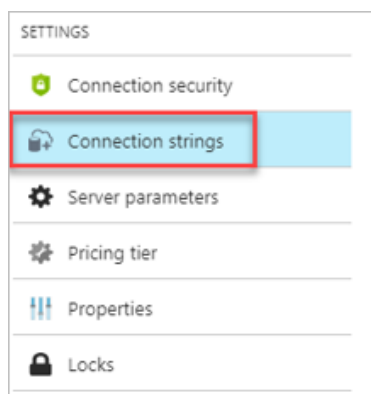
Default Target Schema: osticket

7. After data is restored, it can be queried using Workbench, and appears in the Azure portal.

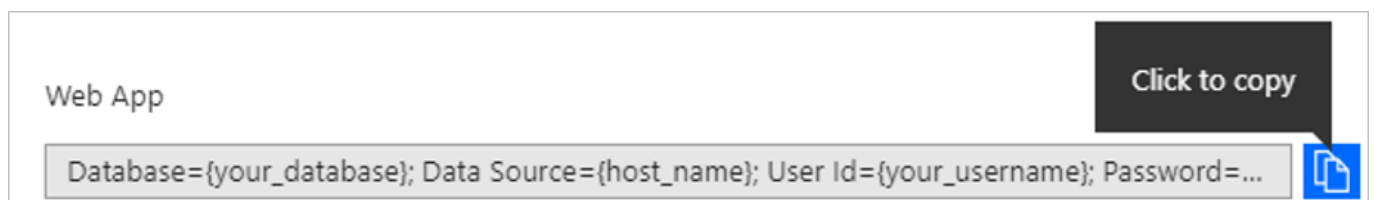
[illegible]



8. Finally, they need to update the database information on the web apps. On the MySQL instance, they open **Connection Strings**.



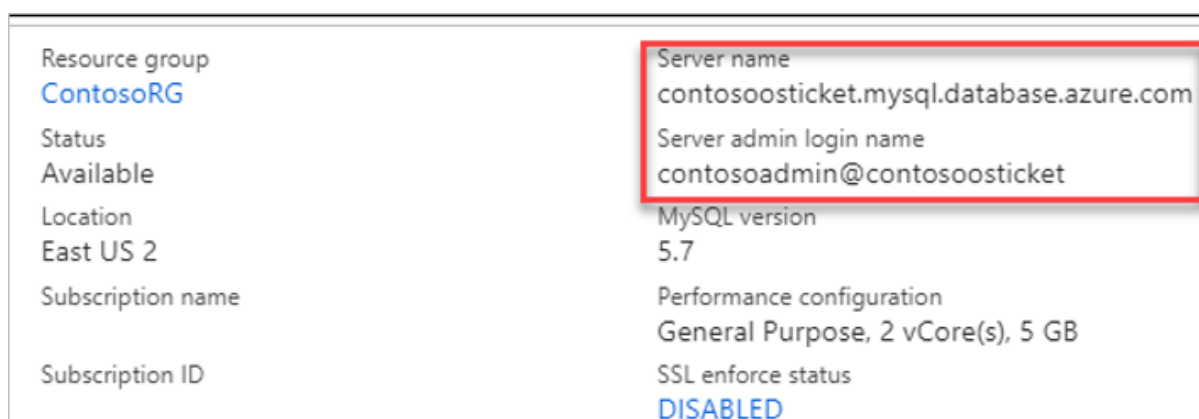
9. In the strings list, they locate the web app settings, and select to copy them.



10. They open a Notepad window and paste the string into a new file, and update it to match the osticket database, MySQL instance, and credentials settings.



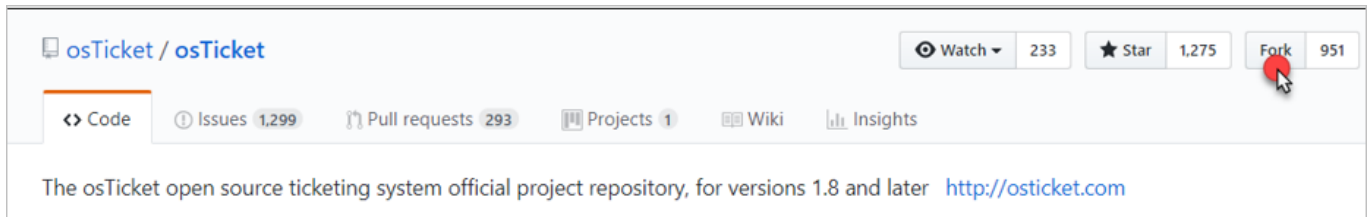
11. They can verify the server name and login from **Overview** in the MySQL instance in the Azure portal.



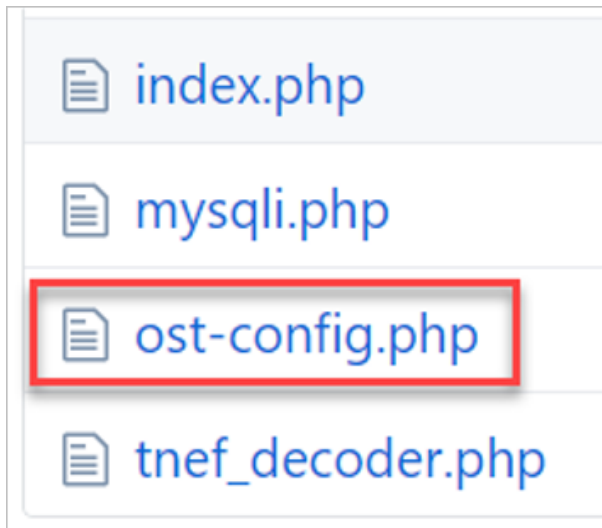
## Step 5: Set up GitHub

Contoso admins create a new private GitHub repo, and sets up a connection to the osTicket database in Azure Database for MySQL. Then, they load the web app into Azure App Service.

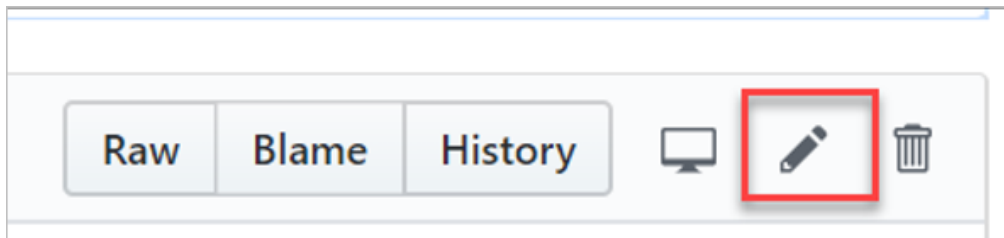
1. They browse to the OsTicket software public GitHub repo, and fork it to the Contoso GitHub account.



2. After forking, they navigate to the **include** folder, and find the **ost-config.php** file.



3. The file opens in the browser and they edit it.



4. In the editor, they update the database details, specifically **DBHOST** and **DBUSER**.

```
38  # Database Options
39  # -----
40  # Mysql Login info
41  define('DBTYPE','mysql');
42  define('DBHOST','contosoosticket.mysql.database.azure.com');
43  define('DBNAME','osticket');
44  define('DBUSER','contosoadmin@contosoosticket');
45  define('DBPASS','[REDACTED]');
```

5. Then they commit the changes.

**Commit changes**

Updated MySQL Server Settings

Add an optional extended description...

☒ Commit directly to the **master** branch.

☐ Create a new branch for this commit and switch to it.

**Commit changes** **Cancel**

6. For each web app (**osticket-eus2** and **osticket-cus**), they modify the **Application settings** in the Azure portal.

**SETTINGS**

**Application settings**

Docker Container

Authentication / Authorization

Managed service identity

Backups

7. They enter the connection string with the name **osticket**, and copy the string from notepad into the **value area**. They select **MySQL** in the dropdown list next to the string, and save the settings.

**Connection strings**

osticket	Database=osticket; Data Source=osticketmysql.mys...	MySQL
----------	---	-------

## Step 6: Configure the web apps

As the final step in the migration process, Contoso admins configure the web apps with the osTicket web sites.

1. In the primary web app (**osticket-eus2**) they open **Deployment option** and set the source to **GitHub**.

**Deployment option**  
Set up deployment option

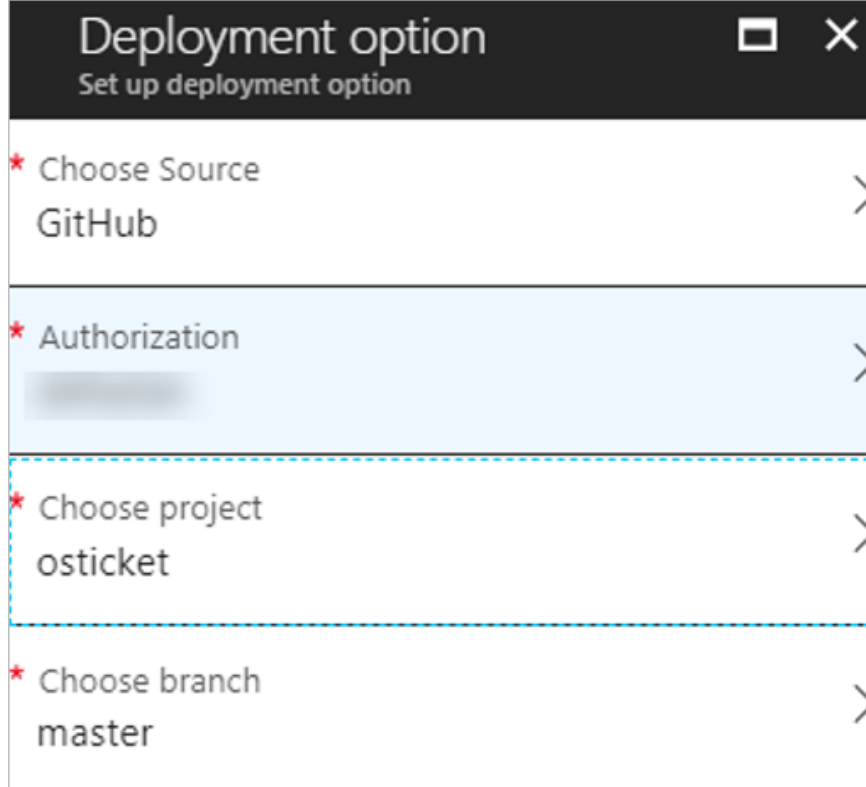
\* Choose Source  
Configure required settings

**Choose source**

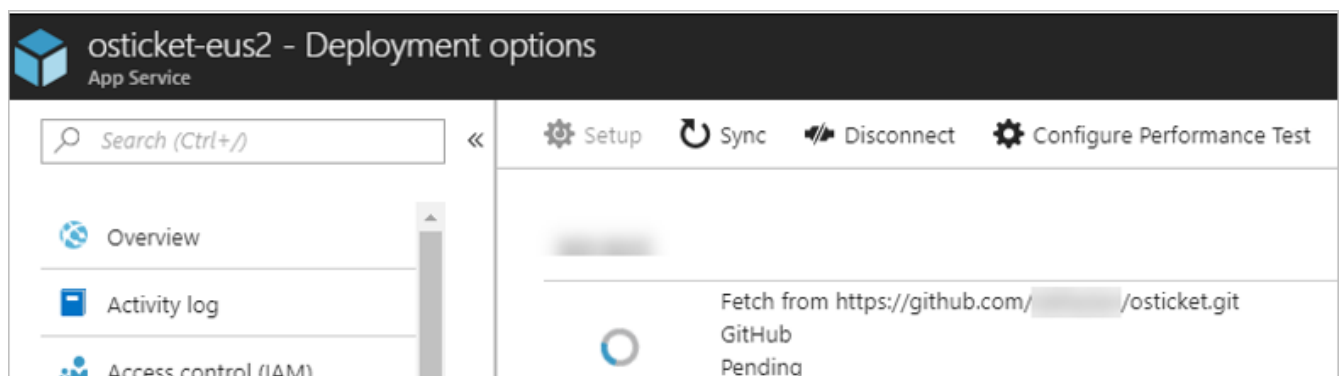
- Local Git Repository  
By Git
- GitHub**  
By GitHub
- Bitbucket  
By Atlassian

2. They select the deployment options.

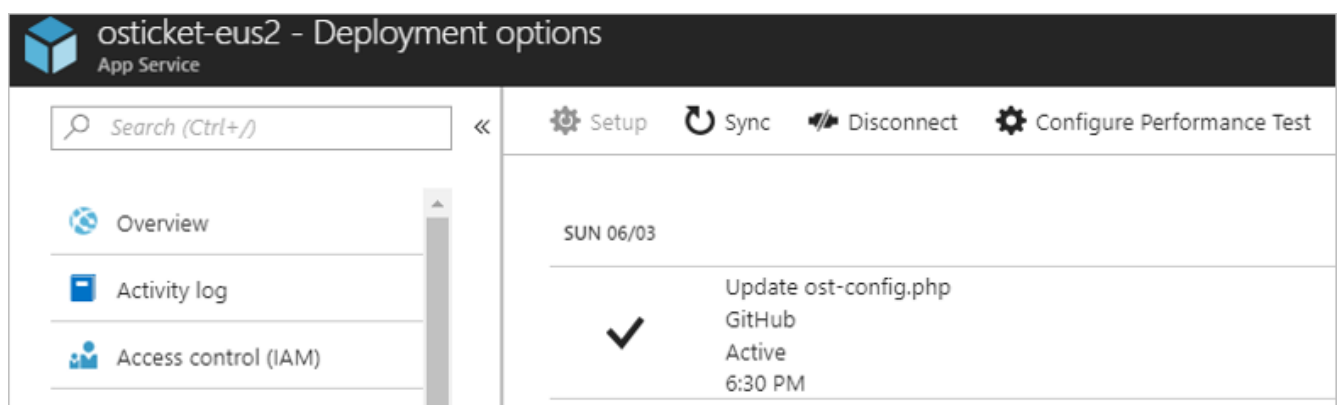




3. After setting the options, the configuration shows as pending in the Azure portal.

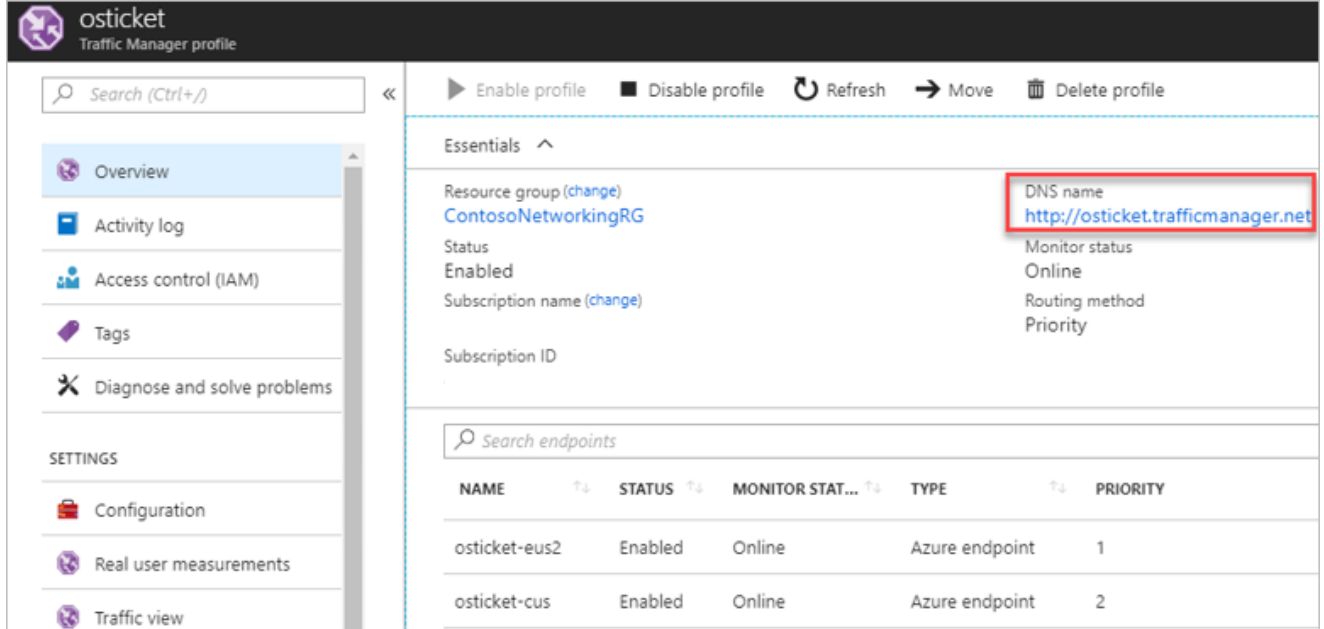


4. After the configuration is updated and the osTicket web app is loaded from GitHub to the Docket container running the Azure App Service, the site shows as Active.



5. They repeat the above steps for the secondary web app (**osticket-cus**).

6. After the site is configured, it's accessible via the Traffic Manager profile. The DNS name is the new location of the osTicket app. [Learn more](#).



7. Contoso wants a DNS name that's easy to remember. They create an alias record (CNAME) **osticket.contoso.com** which points to the Traffic Manager name, in the DNS on their domain controllers.

The screenshot shows the 'New Resource Record' dialog box in the DNS console. The 'Alias (CNAME)' tab is selected. It contains three text input fields: 'Alias name (uses parent domain if left blank):' with the value 'osticket', 'Fully qualified domain name (FQDN):' with the value 'osticket.contoso.com.', and 'Fully qualified domain name (FQDN) for target host:' with the value 'osticket.trafficmanager.net'. A 'Browse...' button is located next to the target host field.

8. They configure both the **osticket-eus2** and **osticket-cus** web apps to allow the custom hostnames.

The screenshot shows the 'Add hostname' dialog box for the 'osticket-eus2' web app. The 'Hostname' field contains 'osticket.contoso.com' and has a green checkmark icon to its right. Below the field is a 'Validate' button, which is highlighted with a red circle and a mouse cursor. At the bottom, the 'Hostname record type' is set to 'CNAME (www.example.com or any subdomain)' in a dropdown menu.

Finally, they set up automatic scaling for the app. This ensures that as agents use the app, the app instances increase and decrease according to business needs.

1. In App Service **APP-SRV-EUS2**, they open **Scale Unit**.
2. They configure a new autoscale setting with a single rule that increases the instance count by one when the CPU percentage for the current instance is above 70% for 10 minutes.

The screenshot shows the 'Scale Unit' configuration page for the App Service 'APP-SRV-EUS2'. At the top, there are buttons for 'Save', 'Discard', 'Disable autoscale', and 'Refresh'. Below these are tabs for 'Configure', 'Run history', 'JSON', and 'Notify'. The 'Configure' tab is active, showing the following settings:

- Autoscale setting name:** OsTicketAutoScale (with a green checkmark)
- Resource group:** ContosoRG (dropdown menu)
- Default Auto created scale condition:** (with a pencil icon and a lock icon)
- 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
- Rules:**
  - Scale out:**

When	APP-SRV-EUS2	(Average) CpuPercentage > 70	Increase instance count by 1
<a href="#">+ Add a rule</a>			
- Instance limits:**
  - Minimum:** 2 (with a green checkmark)
  - Maximum:** 10 (with a green checkmark)
  - Default:** 2 (with a green checkmark)
- Schedule:** This scale condition is executed when none of the other scale condition(s) match

3. They configure the same setting on **APP-SRV-CUS** to ensure that the same behavior applies if the app fails over to the secondary region. The only difference is that they set the instance limit to 1 since this is for failovers only.

The screenshot shows the 'Scale Unit' configuration page for the App Service 'APP-SRV-CUS'. The layout is identical to the previous screenshot, but with the following differences:

- Autoscale setting name:** OsTicketAutoScale-CUS (with a green checkmark)
- Resource group:** ContosoRG (dropdown menu)
- Scale mode:** ☒ Scale based on a metric, ☐ Scale to a specific instance count
- Rules:**
  - Scale out:**

When	APP-SRV-CUS	(Average) CpuPercentage > 70	Increase instance count by 1
<a href="#">+ Add a rule</a>			
- Instance limits:**
  - Minimum:** 1
  - Maximum:** 10 (with a green checkmark)
  - Default:** 1
- Schedule:** This scale condition is executed when none of the other scale condition(s) match

## Clean up after migration

With migration complete, the osTicket app is refactored to running in an Azure App Service web app with continuous delivery using a private GitHub repo. The app's running in two regions for increased resilience. The osTicket database is running in Azure database for MySQL after migration to the PaaS platform.

For clean up, Contoso needs to do the following:

- Remove the VMware VMs from the vCenter inventory.
- Remove the on-premises VMs from local backup jobs.
- Update internal documentation show new locations and IP addresses.
- Review any resources that interact with the on-premises VMs, and update any relevant settings or documentation to reflect the new configuration.
- Reconfigure monitoring to point at the osticket-trafficmanager.net URL, to track that the app is up and running.

## Review the deployment

With the app now running, Contoso need to fully operationalize and secure their new infrastructure.

### Security

The Contoso security team reviewed the app to determine any security issues. They identified that the communication between the osTicket app and the MySQL database instance isn't configured for SSL. They will need to do this to ensure that database traffic can't be hacked. [Learn more](#).

### Backups

- The osTicket web apps don't contain state data and thus don't need to be backed up.
- They don't need to configure backup for the database. Azure Database for MySQL automatically creates server backups and stores. They selected to use geo-redundancy for the database, so it's resilient and production-ready. Backups can be used to restore your server to a point-in-time. [Learn more](#).

### Licensing and cost optimization

- There are no licensing issues for the PaaS deployment.
- Contoso will enable Azure Cost Management licensed by Cloudyn, a Microsoft subsidiary. It's a multicloud cost management solution that helps you use and manage Azure and other cloud resources. [Learn more](#) about Azure Cost Management.