

Active Azure Academy

Application Hosting - Lab



Application Hosting

Background

Today your users expect your application to be available 24/7 with low latency, independent on where they live or where they happen to be on vacation. With the power of Azure we can easily enable our applications to work on a global scale with zero downtime. In this lab we will look at some of the core concepts and services to enable that.

In the lab you will use services such as Azure Traffic Manager, Azure CDN, Azure App Service and Azure App Service for Linux. The code deployed will be built with ASP.NET Core 2.

Not that this lab instruction primarily focuses on how to do the setup on your own and it assumes you have taken part in the sessions at Active Azure Academy to get the theory and details as to why we do certain things.

Prerequisites

- Visual Studio 2017 or Visual Studio 2019
 - Latest updates installed
 - "ASP.NET and Web Development" workload installed
 - "Azure development" workload installed
- An Azure Subscription

Note: VS Code will work to edit the code, but the lab instructions will be written for full Visual Studio

Manual steps

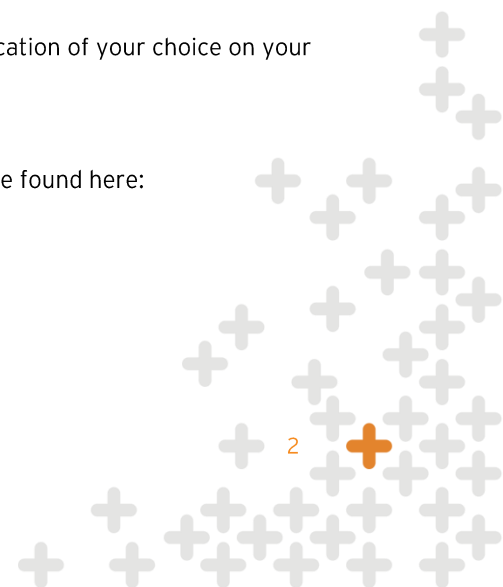
During the lab we will do a lot of manual steps, such as "Right click and deploy" from within Visual Studio as well as creating and configuring the resources in Azure manually using the portal.

In your daily development it's highly recommended to automate these processes in your CI/CD pipeline using, for example VSTS and Azure ARM templates. In the module about DevOps that will come later in Azure Academy we have information about how to do such automations, but to learn as much as possible about the services themselves we have concentrated the lab to the manual parts.

Lab instructions

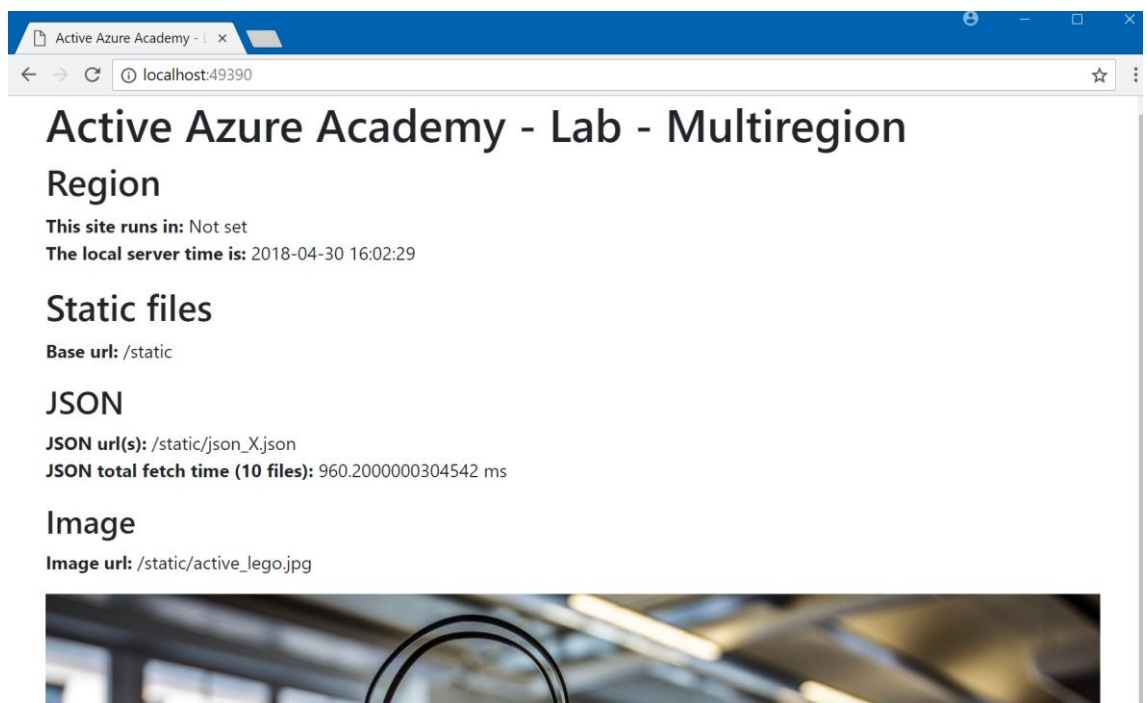
Local

1. Start by cloning the source code of the pre-built application to a location of your choice on your computer. The source code is available at GitHub at this location:
<https://github.com/ActiveSolution/azureacademy-lab-multiregion>
 - a. Instructions on how to clone a GIT repo from GitHub can be found here:
<https://help.github.com/articles/cloning-a-repository/>



```
Windows PowerShell
PS C:\Dev\Code> git clone https://github.com/ActiveSolution/azureacademy-lab-multiregion.git
Cloning into 'azureacademy-lab-multiregion'...
remote: Counting objects: 86, done.
remote: Compressing objects: 100% (56/56), done.
remote: Total 86 (delta 25), reused 76 (delta 19), pack-reused 0
Unpacking objects: 100% (86/86), done.
PS C:\Dev\Code>
```

2. Open the solution *ActiveAzureAcademy.Lab.Multiregion.sln* found in the root of the cloned repo.
3. Launch the web project and your browser should look something like this:



The app is a simple one built specifically to show how a multi-region setup can boost performance and allow for high availability.

- a. The *Region*-section will display in what datacenter the application is running. Locally this is configured in the file *appsettings.json* and set to "Not set". Once deployed into Azure we will override this using Environment variables to whatever region we deploy into.
- b. In the static files section, we load multiple static files, one images and a set if JSON-files. The JSON files are dynamically loaded using a JavaScript. The time it takes to fetch them all is displayed on the page.

Please take note during the lab on how this number changes depending on your configuration and where the app is deployed.

In one of the challenge labs, you will be able to use Azure CDN to speed this up even more.

4. Please take some time to get to know the code, especially how the configuration value for `RegionName` flows. In reversed order, this is a hint on how to look:
 - a. `/Views/Home/Index.cshtml`
 - b. `/Controllers/HomeController.cs`
 - c. `/Configuration/RegionConfiguration.cs`
 - d. `/Startup.cs`
 - e. `/Program.cs`
 - i. Note that `WebHost.CreateDefaultBuilder(args)`, since 2.0, will register these configuration providers by default. Therefore, any environment variable we provide in Azure will override our local config.
`config.AddJsonFile("appsettings.json").AddEnvironmentVariables();`

```
@using ActiveAzureAcademy.Lab.Multiregion.Configuration
@using Microsoft.Extensions.Options
@model ActiveAzureAcademy.Lab.Multiregion.Models.HomeViewModel
@inject IOptions<StaticFilesConfiguration> StaticFilesConfig

@{
    Layout = "_Layout";
}

<h1>Active Azure Academy - Lab - Multiregion</h1>

<h2>Region</h2>
<p>
    <strong>This site runs in: </strong> @Model.Location<br />
    <strong>The local server time is: </strong> @Model.LocalTime
</p>
```

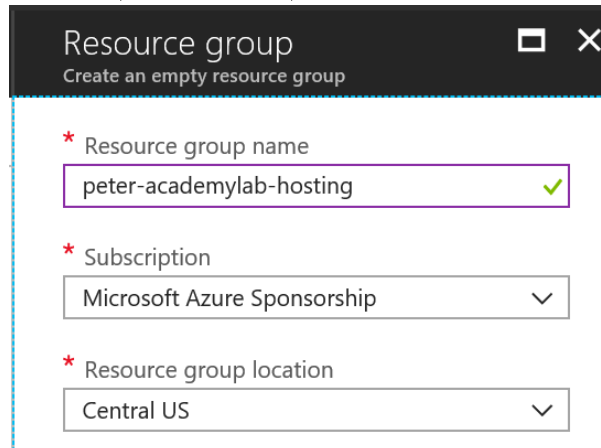
Azure

Now that you've got the application up and running locally and understands how it works we are ready to deploy into Azure.

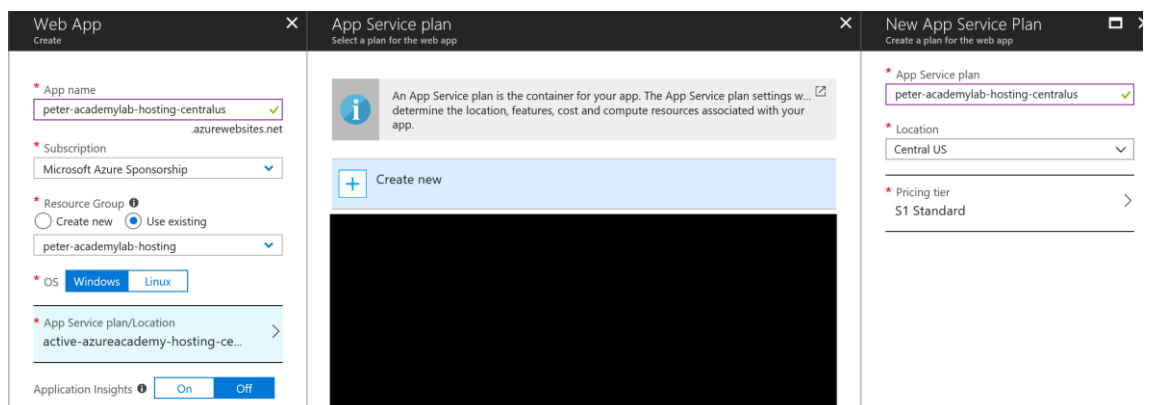
1. Start by creating a resource group in your subscription. The resource group should be located in *Central US*. Instructions on how to create a resource group using the portal can be found here: <https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-portal>
 - a. Feel free to follow your own naming convention, in these instructions we will use this:
`[name]-academylab-hosting`

For someone called Peter, that would be: *peter-academylab-hosting*

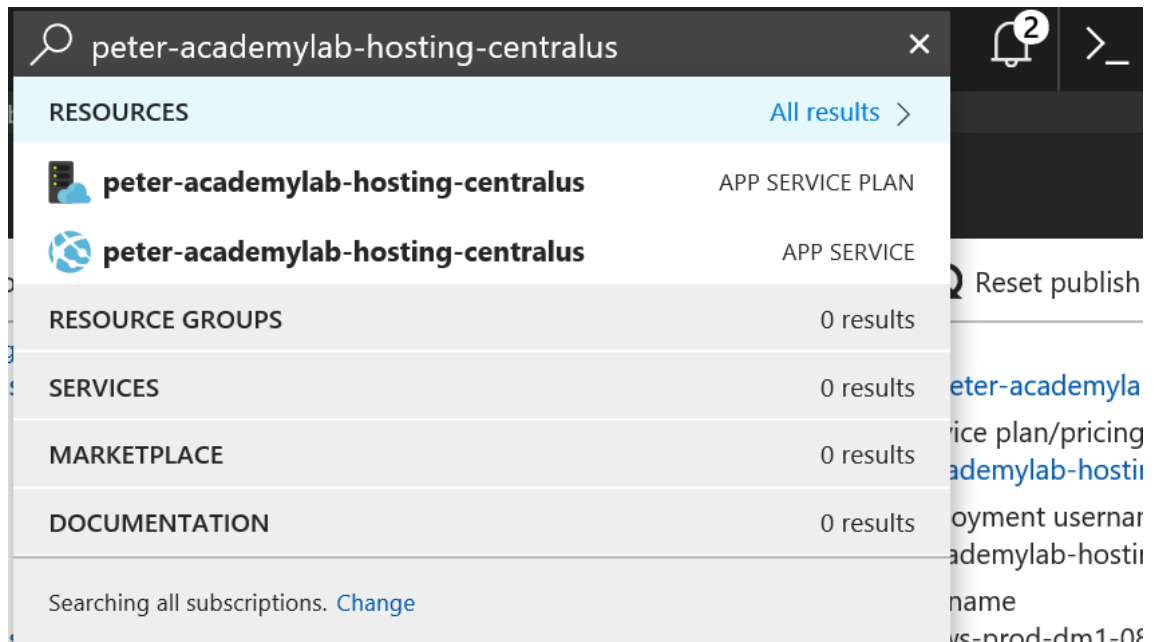
Not all, but some, services in azure requires a globally unique name and therefore using your name will be an important part. The reason some services requires a unique name is that they will get their DNS/hostname based on the name, and such hostname is by its very nature required to be unique.



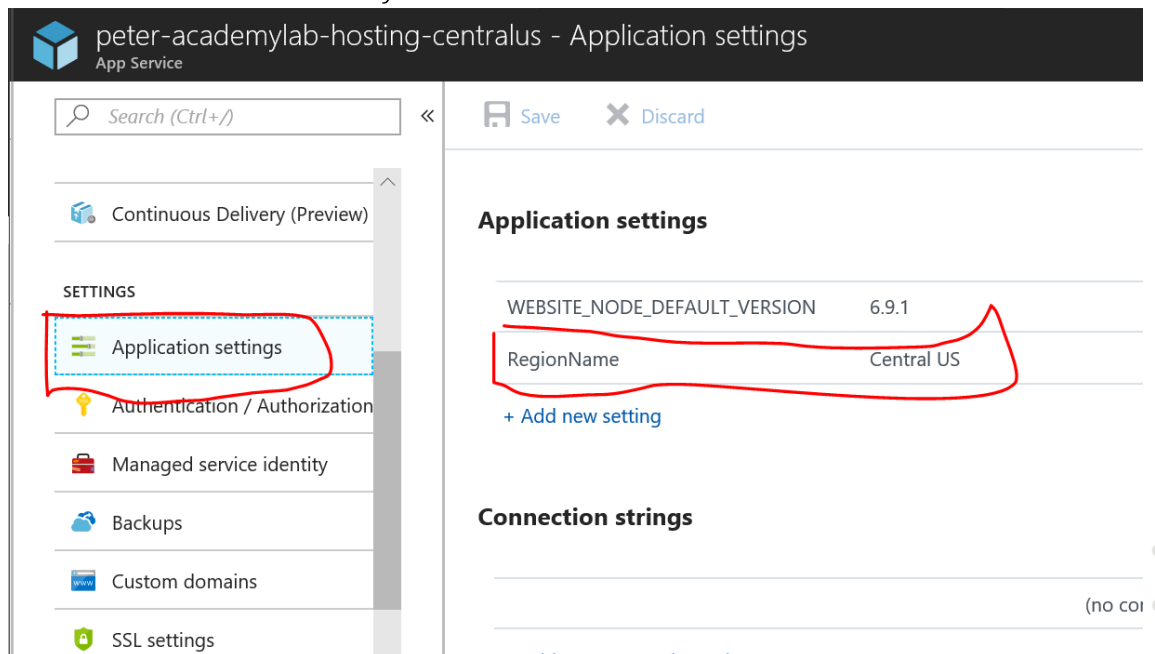
2. Create an App Service in your resource group and make sure to create a new App Service plan at the same time. Instructions on how to do so can be found here: <https://docs.microsoft.com/en-us/azure/app-service/app-service-plan-manage>
 - a. The App Service should use Windows
 - b. The App Service and App Service Plan should be named: *[name]-academylab-hosting-centralus*
 - c. The App Service Plan should be located in Central US and be using the S1 Pricing tier as this is the cheapest one that allows us to use Traffic Manager.



5. Once your App Service is ready, open it up. Hint: you could locate it using the global search located in the top right corner:





6. Under *Application settings*, add a new setting with the name *RegionName* and give it the value *Central US*. These settings will be presented as environment variables to the application and therefore override our local settings as described above.





7. Deploy the web application from Visual Studio to your Azure App service running in the US. Instructions on how to do so can be found here: <https://docs.microsoft.com/en-us/aspnet/core/tutorials/publish-to-azure-webapp-using-vs?view=aspnetcore-2.1#deploy-the-app-to-azure>. Make sure to use *Select existing* and navigate to the App Service you just created.

Pick a publish target

 App Service

 Azure Virtual Machines

 IIS, FTP, etc

 Folder

Azure App Service

Fully managed, and highly scalable cloud environment

☐ Create New

☒ Select Existing

App Service

Host your web and mobile applications, REST APIs, and more in Azure

Subscription


Microsoft Azure Sponsorship


View

Resource Group

Search

peter

 peter-academylab-hosting

 peter-academylab-hosting-centralus

Active Solution Sverige AB

petero@activesolution.se

8. Once the app is deployed a browser will launch and you can give it a try. Refresh a couple of times (to ensure its warmed up) and then compare the numbers with your local deployment. Note that even with the speed of light, the distance from Europe to the US adds up on the milliseconds.
9. Now we will create another app service, running in a different region (West Europe). This time however, we will create it using Web App for containers, meaning that we will create an App Service that hosts the application running in a container.

We have prepared a pre-baked image for you, but as a challenge you can extend the project and add Docker support to it.

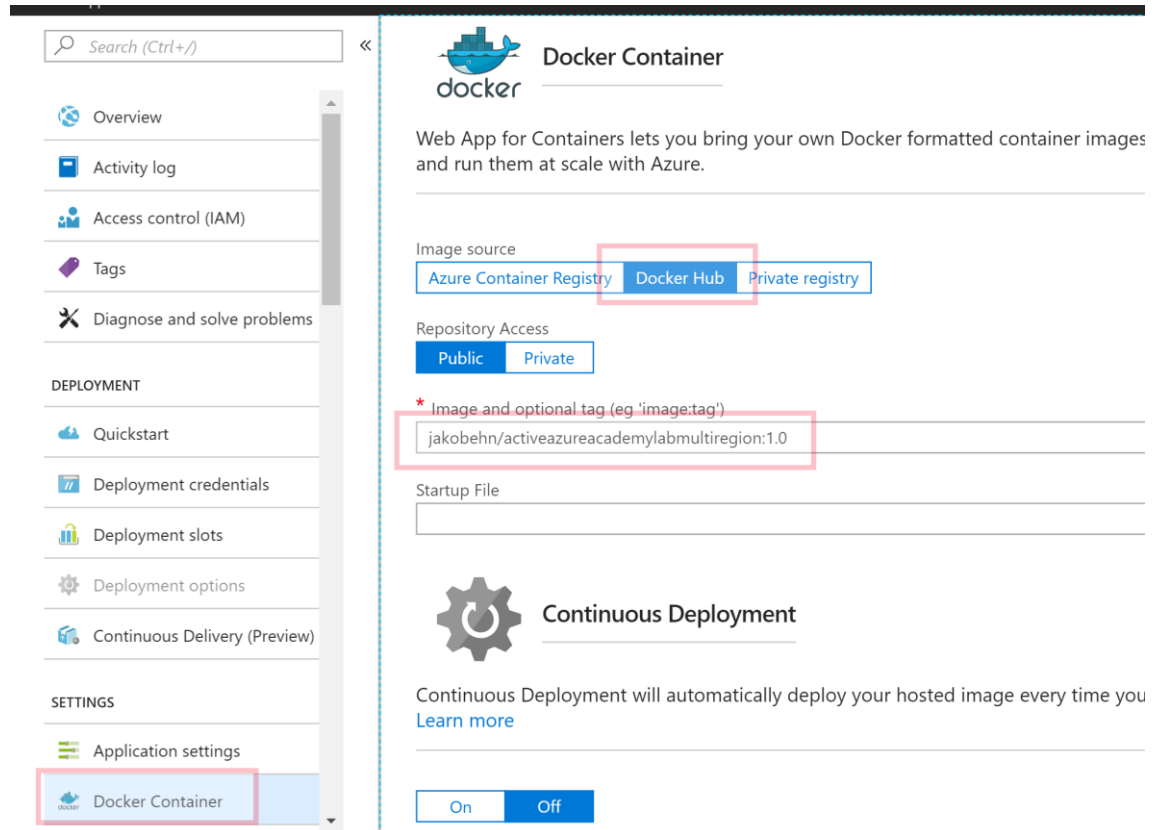
See <https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/docker/visual-studio-tools-for-docker?view=aspnetcore-2.1> for information about how to add Docker support to an ASP.NET Core project.

10. In the same resource group as before, add a *Web app for containers* app service and an app service plan called *[name]-academylab-hosting-west europe*.

Make sure that you select West Europe as the region

11. For the containers configuration, either select your own container from the previous step, or select the pre-baked image that is located at Docker Hub in this URL:

[jakobehn/activeazureacademylabmultiregion:1.0](https://hub.docker.com/r/jakobehn/activeazureacademylabmultiregion/1.0)



Docker Container

Web App for Containers lets you bring your own Docker formatted container images and run them at scale with Azure.

Image source: Azure Container Registry **Docker Hub** Private registry

Repository Access: **Public** Private

* Image and optional tag (eg 'image:tag')

Startup File

Continuous Deployment

Continuous Deployment will automatically deploy your hosted image every time you [Learn more](#)

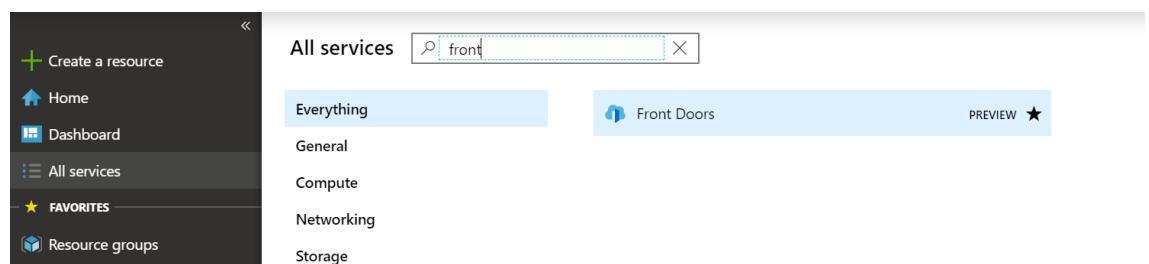
On **Off**

12. Create an application setting for **RegionName** and set it to *West Europe*
13. Save the app service and browse to the URL. Note that the first time it will take some time before it starts. This is because Azure app service needs to pull the image the first time before it can start the app service.
14. Refresh the page a couple of times, note that the JSON fetch time should now be shorted compared to the previous one, since it's now served from the West Europe data center.

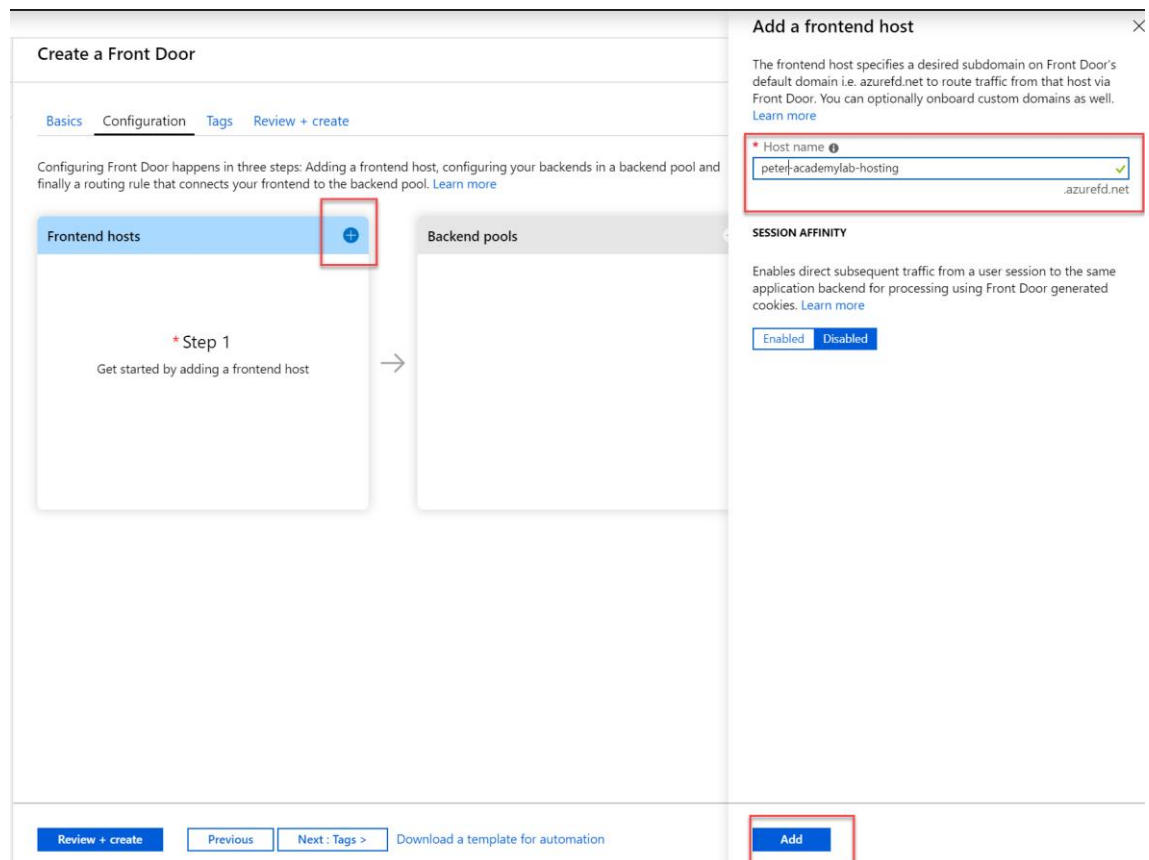
15. Now that we have two web apps running, one in Central US and one in West Europe, we basically have two different URLs to access your website, and that's not especially user friendly. To get a unified URL we can deploy either a Traffic Manager or Front Door. They have different use cases and characteristics, but Front Door is probably what you want in most cases, so that we will use here. It will enable you to have a unified URL and still ensure that the users use the closest datacenter.

Create a Front Door, Instructions can be found here:

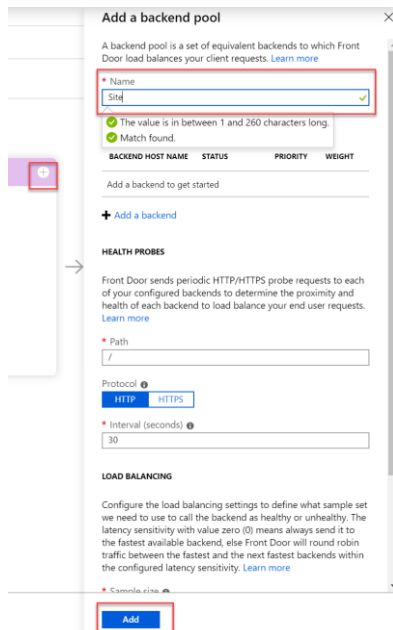
<https://docs.microsoft.com/en-us/azure/frontdoor/quickstart-create-front-door>



16. Add a frontend pool add set the Host name to: *[name]-academylab-hosting*



17. Add a backend pool, call it something, for example "Site":



Add a backend pool

A backend pool is a set of equivalent backends to which Front Door load balances your client requests. [Learn more](#)

* Name
Site1 ✓

✓ The value is in between 1 and 260 characters long.
✓ Match found.

BACKEND HOST NAME	STATUS	PRIORITY	WEIGHT
Add a backend to get started			

+ Add a backend

HEALTH PROBES

Front Door sends periodic HTTP/HTTPS probe requests to each of your configured backends to determine the proximity and health of each backend to load balance your end user requests. [Learn more](#)

* Path
/

Protocol
HTTP HTTPS

* Interval (seconds)
30

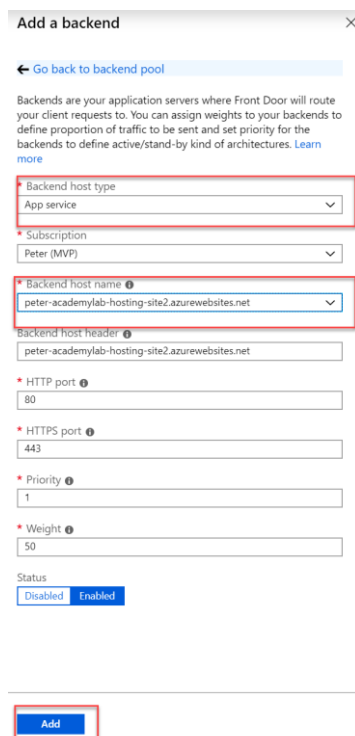
LOAD BALANCING

Configure the load balancing settings to define what sample set we need to use to call the backend as healthy or unhealthy. The latency sensitivity with value zero (0) means always send it to the fastest available backend, else Front Door will round robin traffic between the fastest and the next fastest backends within the configured latency sensitivity. [Learn more](#)

* Sample size

Add

18. Add a two backends, one per app service you have created (West Europe and Central US):



Add a backend

← Go back to backend pool

Backends are your application servers where Front Door will route your client requests to. You can assign weights to your backends to define proportion of traffic to be sent and set priority for the backends to define active/stand-by kind of architectures. [Learn more](#)

* Backend host type
App service

* Subscription
Peter (MVP)

* Backend host name
peter-academylab-hosting-site2.azurewebsites.net

Backend host header
peter-academylab-hosting-site2.azurewebsites.net

* HTTP port
80

* HTTPS port
443

* Priority
1

* Weight
50

Status
Disabled Enabled

Add

19. Add a routing rule and call it something of your choice, for example "Default". Also, go to "Advanced" and enable Caching.

Add a routing rule

A routing rule maps your frontend host and a matching URL path pattern to a specific backend pool. [Learn more](#)

Basic

Advanced

Name

Default

Frontend hosts

peter-academy-lab-hosting.azurefd.net

Backend pool

CentralUS

Accepted protocol

☒ HTTP
 ☒ HTTPS

PATTERNS TO MATCH

/*

/path

Add

Basic Advanced

Forwarding protocol ?

☒ Match request
 ☐ HTTP only
 ☐ HTTPS only

URL REWRITE

Path to use when constructing the request for URL rewrite to forward to the backend. [Learn more](#)

Enabled Disabled

CACHING

Enable caching for this routing rule. When enabled, Front Door will cache your static content. [Learn more](#)

Enabled Disabled

* Query string caching behavior ?

Cache every unique URL

* Dynamic compression

☒ Enabled
 ☐ Disabled

- Finish the creation, this can take a while, especially for all config to propagate to the edge nodes:

Create a Front Door

✓ Validation passed

Basics Configuration Tags **Review + create**

BASICS

Subscription	Peter (MVP)
Resource group	peter-academylab-hosting
Region	East Asia
Front Door name	peter-academylab-hosting
Host name	peter-academylab-hosting.azurefd.net

TAGS

None

Create Previous Next Download a template for automation

21. Once the Front Door is fully configured, visit the URL ([http://\[name\]-academylab-hosting.azurefd.net/](http://[name]-academylab-hosting.azurefd.net/)) and note that you will be served from the location nearest you. If not, wait a few minutes as the DNS might not have been updated.
22. As a last test, stop the App Service located in Europe. Front Door will continuously monitor your endpoints and make sure they are healthy. In case an instance of your web goes down, the load will be redistributed to a healthy site within a few minutes. In your case the traffic should be directed to Central US.
23. Send an email to peter.orneholm@activesolution.se with the link to your traffic manager profile to show that you have completed the lab.
24. You now have a multi-region web app running that will server your users from the nearest location. As a last step, don't forget to delete the resource group to stop all cost related to this lab.

Challenge labs

In case you've finished the standard lab, here are some suggestions on what steps to take next. No detailed instructions will be provided, only suggestions and relevant links.

- **Additional regions:** Add another region, maybe North Europe or any other located in Europe to see what location has the lowest response times.
 - <https://azure.microsoft.com/en-us/global-infrastructure/regions/>

- **CDN:** The application is prepared to serve static files, such as the JSON files and the image over a CDN. As a challenge, add a CDN endpoint that serves the files in /static/. Add an Application Setting named BaseUrl to set the base URL for static files. As there is a CDN node in Sweden, that will bring down the response times even further.
- <https://docs.microsoft.com/en-us/azure/cdn/cdn-create-new-endpoint>
- **Route static files from a blob storage:** Front door allows you to serve content for specific urls from other backends. Try to configure /static to serve content from blob storage accounts.
- **Deploy to Kubernetes in AKS (Azure Container Services):**
Running your containers in Kubernetes allows you easily scale up and down, and many other features.

Follow these instructions to create a Kubernetes cluster in Azure AKS:

<https://docs.microsoft.com/en-us/azure/aks/kubernetes-walkthrough>

When the cluster is up and running, and you have connected to it using kubectl, you can deploy this application using the yaml file that is shown below.

***Note:** The file is also available in the git repository of the sample application, in the `deploy\aks` folder*

```
apiVersion: v1
kind: Service
metadata:
  name: apphostinglab
  labels:
    app: apphostinglab
spec:
  selector:
    app: apphostinglab
  ports:
    - protocol: TCP
      port: 80
      nodePort: 30082
    type: LoadBalancer
---
apiVersion: apps/v1beta1
kind: Deployment
metadata:
```

```

name: apphostinglab
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: apphostinglab
    spec:
      containers:
        - name: apphostinglab
          image: jakobehn/activeazureacademylabmultiregion:1.0
          ports:
            - containerPort: 80
          env:
            - name: regionName
              value: "West Europe"

```

When the deployment is done, look at the Kubernetes service for the external endpoint, where the web application will be available (Note: Your IP addresses will be different compared to the screenshot below)

Services					
Name ↕	Labels	Cluster IP	Internal endpoints	External endpoints	Age ↕
✓ apphostinglab	app: apphostinglab	10.0.152.97	apphostinglab.apphosting... apphostinglab.apphosting...	52.166.17.44:80 ↗	a minute

