**Configuring Azure Traffic Manager, Application Gateway and App Services with Terraform**

*Azure App Service* is a great choice for a Platform As A Service (PaaS) option to host Web and Api applications.

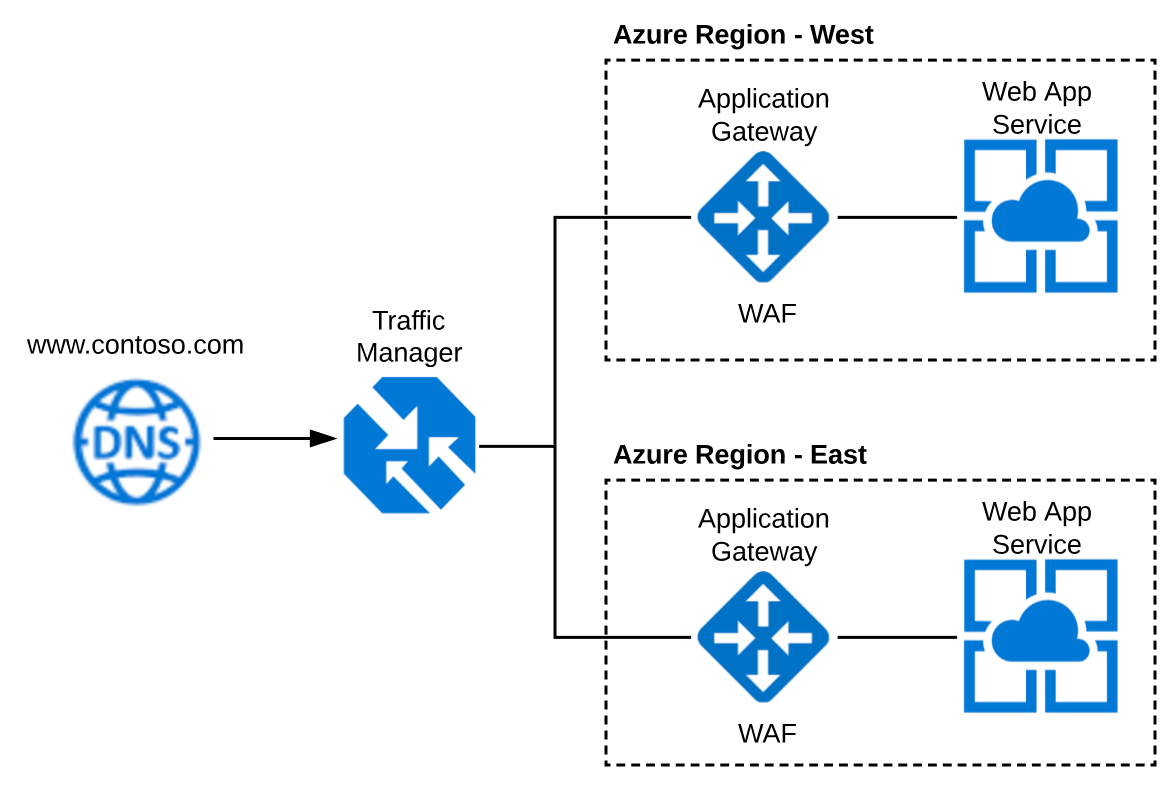
The service is fully managed, scales vertically, horizontally and runs platforms such as:

* .NET
* .NET Core
* Java
* Ruby
* Node.JS
* PHP
* Python
* Docker

When the service is created it provides a publicly accessible *Virtual IP Address* and a *\*.azurewebites.net* URL.

For a production deployment you are going to want to protect your App Services from potential attacks such as Denial of Service(DDOS), SQL Injection and Cross Side Scripting(XSS). To do this you can use *Azure Application Gateway* with a *Web Application Firewall* SKU.

Typically you are also going to want to support a  
multi-region deployment for high availability, redundancy or fail over reasons. To do this you can use *Azure Traffic Manager* to route traffic as needed.



**Terraform**

I like using Terraform to create my cloud infrastructure. Most of the GA Azure services are supported but there are always limitations which will need to be supplemented using some post deployment Powershell scripts.

Installation instructions can be found here

**Creating a Terraform Azure Principal**

Follow the instructions on the Terraform Website to create the principal account. This account will have *Contributor* access to your Azure Subscription so that it can be used to create / modify / remove resources.

The instructions can be found here.

**Terraform Script**

mkdir /c/code/contoso.com/infrastructure

cd /c/code/contoso.com/infrastructure

Create an **environment.tf** file it should look like the sample below with the Azure Principal configuration and the creation of 3 resource groups.

|  |
| --- |
| variable "subscription\_id" {} |
| variable "client\_id" {} |
| variable "client\_secret" {} |
| variable "tenant\_id" {} |
| variable "environment" {} |
|  |
| # Configure the Azure Provider |
| provider "azurerm" { |
| subscription\_id = "${var.subscription\_id}" |
| client\_id = "${var.client\_id}" |
| client\_secret = "${var.client\_secret}" |
| tenant\_id = "${var.tenant\_id}" |
| } |
|  |
| # Create Resource Groups |
| resource "azurerm\_resource\_group" "resource-group-global" { |
| name = "rgp-global-contoso-${var.environment}" |
| location = "Central US" |
| } |
|  |
| resource "azurerm\_resource\_group" "resource-group-westus" { |
| name = "rgp-westus-contoso-${var.environment}" |
| location = "West US" |
| } |
|  |
| resource "azurerm\_resource\_group" "resource-group-eastus" { |
| name = "rgp-eastus-contoso-${var.environment}" |
| location = "East US" |
| } |

[**view raw**](https://gist.github.com/stephenweinrich/eda886adf5b9501948a54d899e7ee754/raw/5e1aa3a4d8540a49ba92c50334e4ae894f53d0db/environment.tf)[**environment.tf**](https://gist.github.com/stephenweinrich/eda886adf5b9501948a54d899e7ee754#file-environment-tf) hosted with ❤ by [**GitHub**](https://github.com/)

Create a **config.dev01.tfvars** file it should look like the sample below.

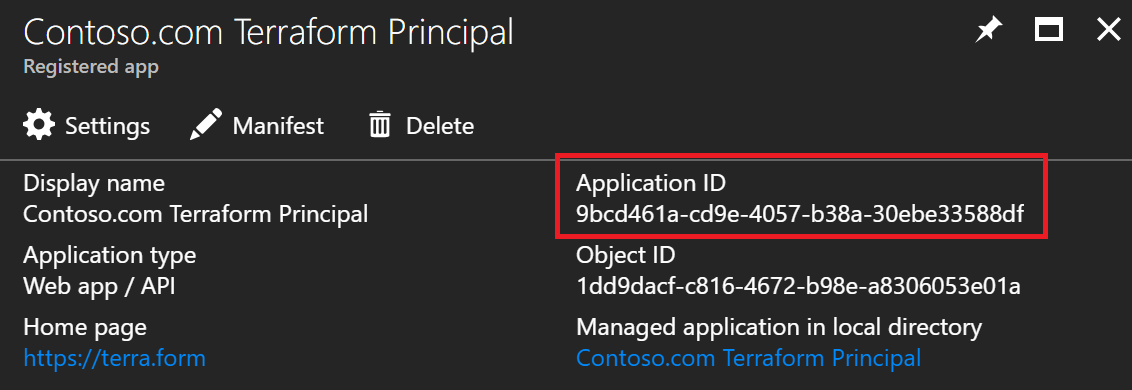
**NOTE:** Please set the environment variable to something unique to avoid resource naming conflicts.

|  |
| --- |
| subscription\_id = "<--paste-your-subscription-id-here-->" |
|  |
| client\_id = "<--paste-your-client-id-here-->" |
|  |
| tenant\_id = "<--paste-your-tenant-id-here-->" |
|  |
| client\_secret = "<--paste-your-secret-here-->" |
|  |
| environment = "dev01" |

[**view raw**](https://gist.github.com/stephenweinrich/a966310a9157ac6e076e2a19144e69f1/raw/5bc5fd6e3b4a4de239b5470a9668a49048e72f3a/config.dev01.tfvars)[**config.dev01.tfvars**](https://gist.github.com/stephenweinrich/a966310a9157ac6e076e2a19144e69f1#file-config-dev01-tfvars) hosted with ❤ by [**GitHub**](https://github.com/)

**subscription\_id** - navigate to Subscriptions in Azure Portal, copy/paste Subscription ID.

**client*id****- navigate to your Azure Active Directory, App Registrations and search for your App. Application ID = client*id



**tenant*id****- navigate to the Azure Active Directory blade and select Properties. Directory ID = tenant*id.

**secret\_key** - this would have been created while following the Terraform Principal creation guidance mentioned above. Keep it in a safe place.

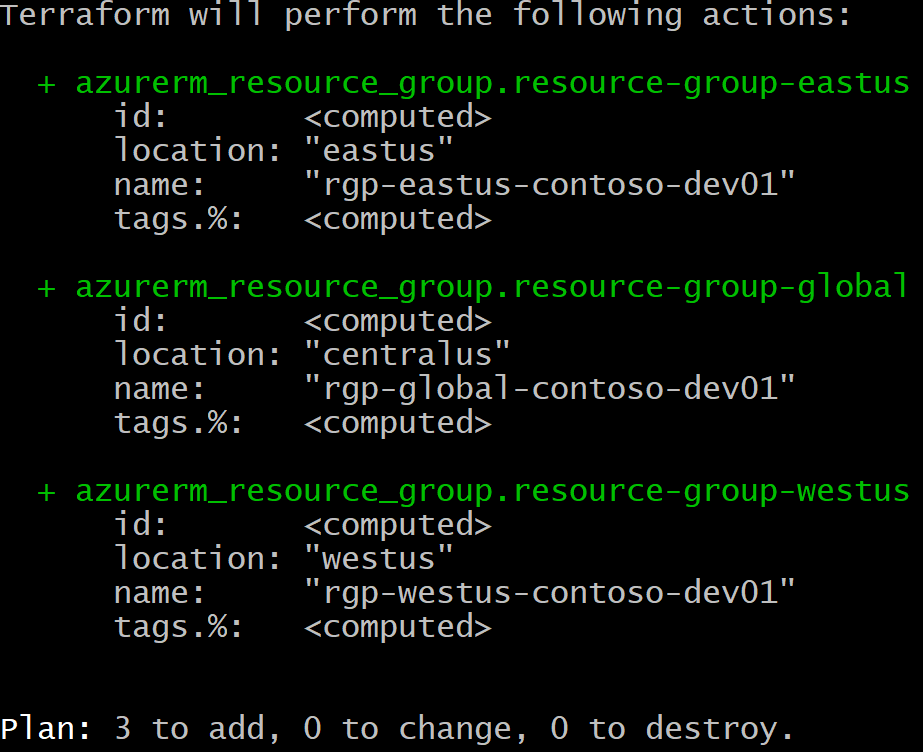
Initialize a terraform environment. For now we will keep the state file local. Normally I'd centralize the state file using a blob storage container or something similar.

terraform init

terraform workspace new dev01

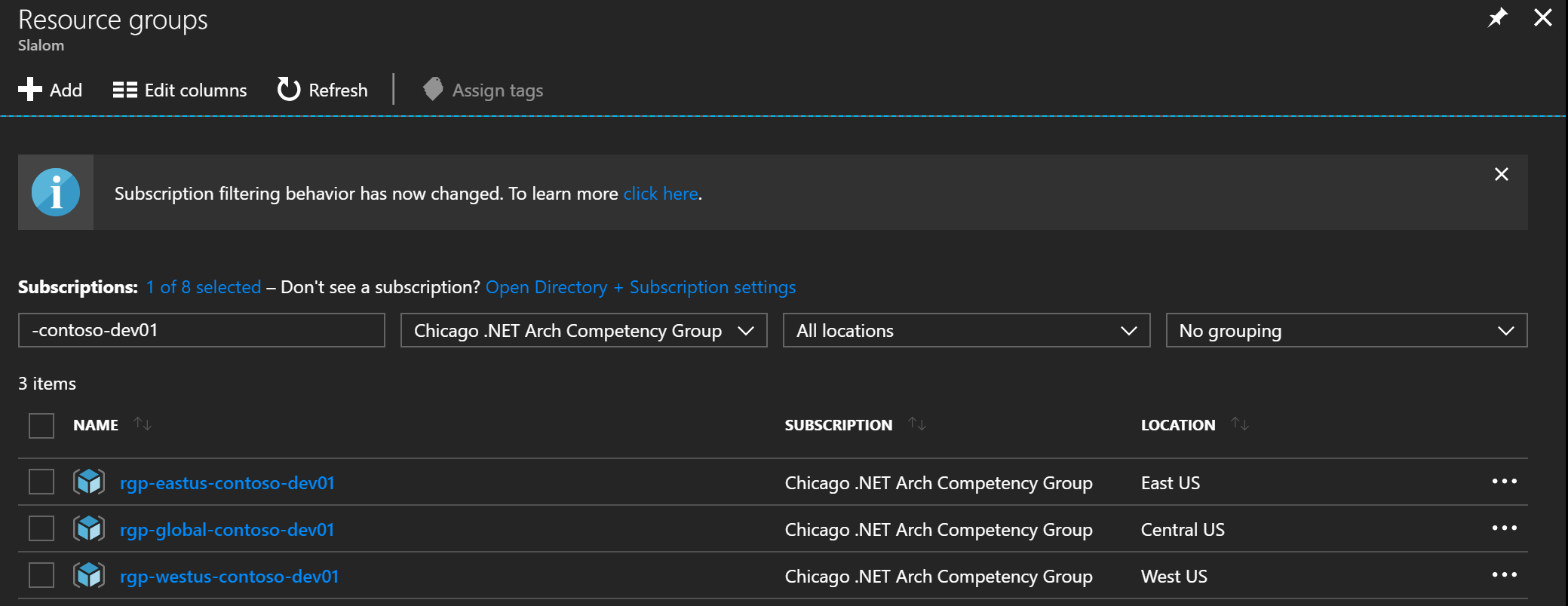
terraform plan -var-file=config.dev01.tfvars

If everything has been configured correctly you should have the following plan output.



terraform apply -var-file=config.dev01.tfvars -auto-approve

If everything is working as planned your 3 Resource Groups will now be visible for your Subscription in the Azure Portal.



Now that we have our infrastructure as code automation working, our next step is to create the App Services.

**Web App Services**

Add this terraform code to your environment.tf file to create a web app service in the East and West resource groups. We will create an app service plan for each too using the Free tier.

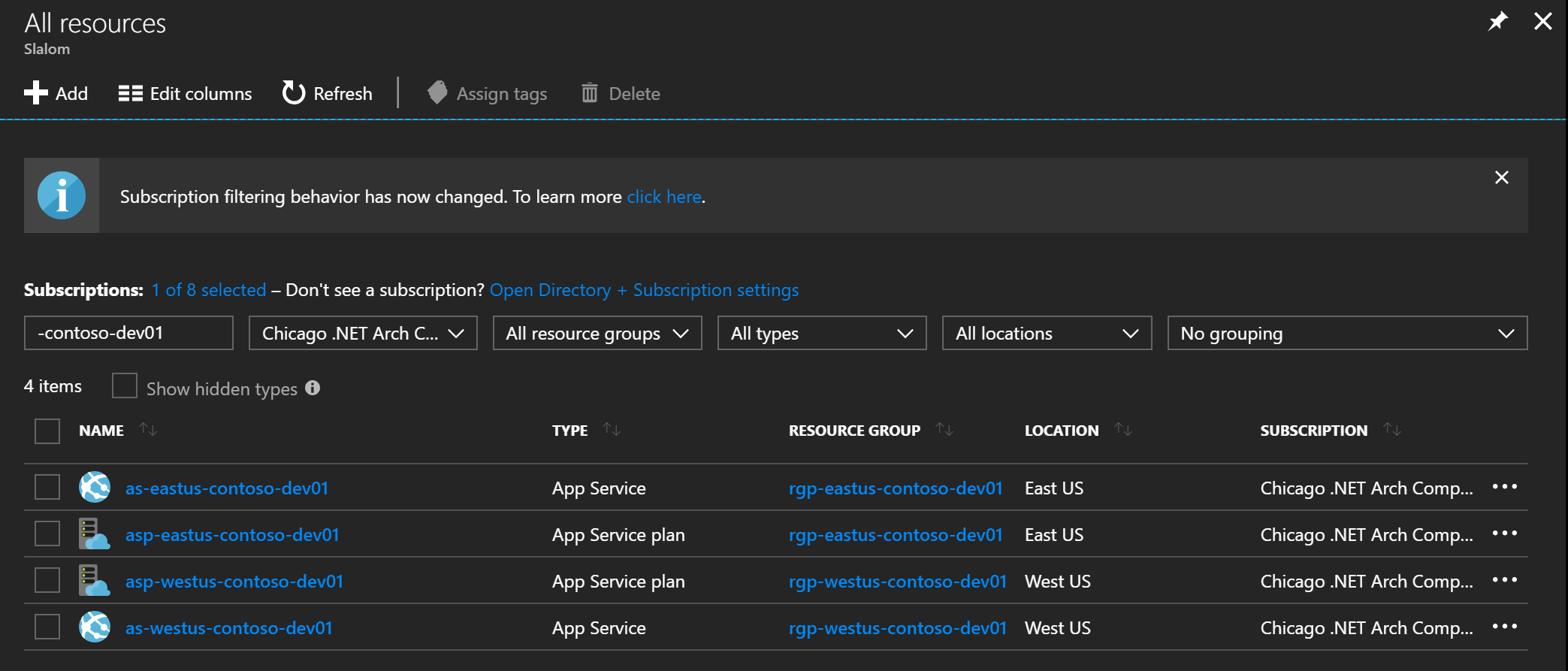
|  |
| --- |
| # Create App Service Plans |
| resource "azurerm\_app\_service\_plan" "app-service-plan-westus" { |
| name = "asp-westus-contoso-${var.environment}" |
| location = "${azurerm\_resource\_group.resource-group-westus.location}" |
| resource\_group\_name = "${azurerm\_resource\_group.resource-group-westus.name}" |
|  |
| sku { |
| tier = "Free" |
| size = "F1" |
| } |
| } |
|  |
| resource "azurerm\_app\_service\_plan" "app-service-plan-eastus" { |
| name = "asp-eastus-contoso-${var.environment}" |
| location = "${azurerm\_resource\_group.resource-group-eastus.location}" |
| resource\_group\_name = "${azurerm\_resource\_group.resource-group-eastus.name}" |
|  |
| sku { |
| tier = "Free" |
| size = "F1" |
| } |
| } |
|  |
| # Create App Services |
| resource "azurerm\_app\_service" "app-service-westus" { |
| name = "as-westus-contoso-${var.environment}" |
| location = "${azurerm\_resource\_group.resource-group-westus.location}" |
| resource\_group\_name = "${azurerm\_resource\_group.resource-group-westus.name}" |
| app\_service\_plan\_id = "${azurerm\_app\_service\_plan.app-service-plan-westus.id}" |
| } |
|  |
| resource "azurerm\_app\_service" "app-service-eastus" { |
| name = "as-eastus-contoso-${var.environment}" |
| location = "${azurerm\_resource\_group.resource-group-eastus.location}" |
| resource\_group\_name = "${azurerm\_resource\_group.resource-group-eastus.name}" |
| app\_service\_plan\_id = "${azurerm\_app\_service\_plan.app-service-plan-eastus.id}" |
| } |

[**view raw**](https://gist.github.com/stephenweinrich/32d380e0b8dcab58b5b48e215d374d29/raw/f801e0d3531748afb302accfb06e07cba9c323e7/environment.tf)[**environment.tf**](https://gist.github.com/stephenweinrich/32d380e0b8dcab58b5b48e215d374d29#file-environment-tf) hosted with ❤ by [**GitHub**](https://github.com/)

terraform workspace select dev01

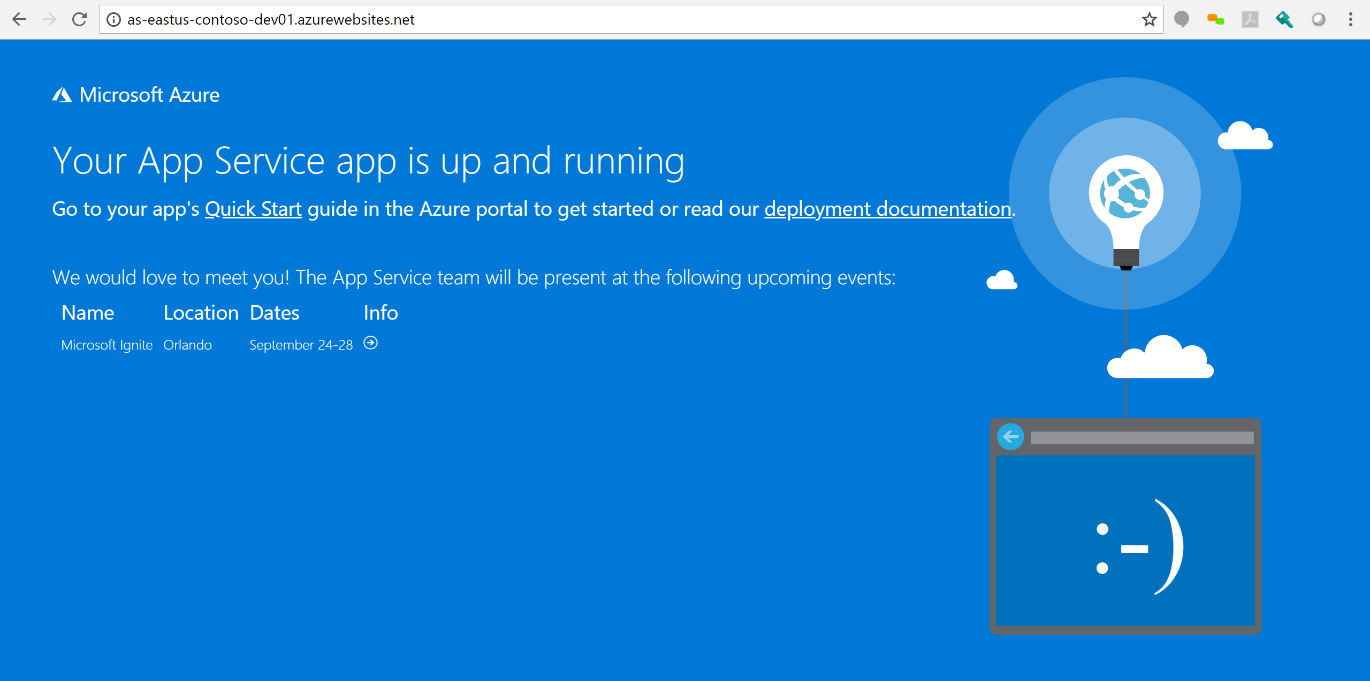
terraform plan -var-file=config.dev01.tfvars

terraform apply -var-file=config.dev01.tfvars -auto-approve



Navigate to you App Service endpoints to verify they are "up and running".

* <https://as-eastus-contoso-dev01.azurewebsites.net/>
* <https://as-westus-contoso-dev01.azurewebsites.net/>



**Application Gateway**

Next we will add the following terraform code to create the Azure Application Gateway. We will be adding the Web Application Firewall (OWASP 3.0) and we will be enabling HTTP2 which it now supports.

Application Gateway requires several other services namely:

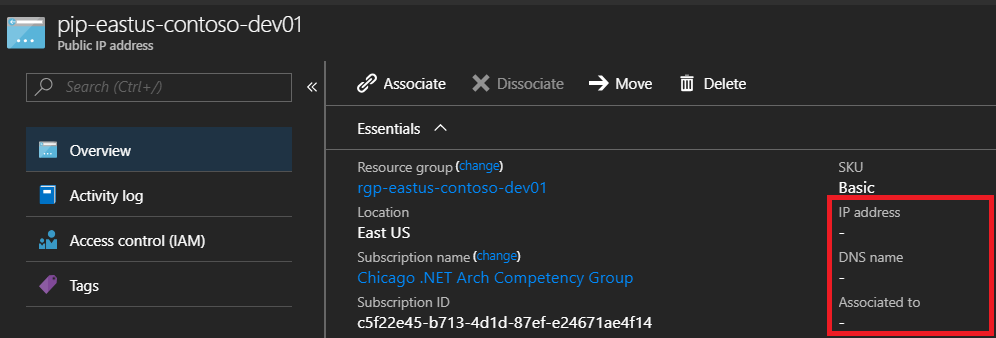
* Virtual Network (VNET)
* Subnet
* Dynamic Public IP

Use the following code to create these resources:

|  |
| --- |
| # Create Vnets |
| resource "azurerm\_virtual\_network" "vnet\_west" { |
| name = "vnet-westus-contoso-${var.environment}" |
| resource\_group\_name = "${azurerm\_resource\_group.resource-group-westus.name}" |
| location = "${azurerm\_resource\_group.resource-group-westus.location}" |
| address\_space = ["10.0.0.0/16"] |
| } |
|  |
| resource "azurerm\_virtual\_network" "vnet\_east" { |
| name = "vnet-eastus-contoso-${var.environment}" |
| resource\_group\_name = "${azurerm\_resource\_group.resource-group-eastus.name}" |
| location = "${azurerm\_resource\_group.resource-group-eastus.location}" |
| address\_space = ["10.0.0.0/16"] |
| } |
|  |
| # Create Subnets |
| resource "azurerm\_subnet" "subnet\_west" { |
| name = "default" |
| resource\_group\_name = "${azurerm\_resource\_group.resource-group-westus.name}" |
| virtual\_network\_name = "${azurerm\_virtual\_network.vnet\_west.name}" |
| address\_prefix = "10.0.0.0/24" |
| } |
|  |
| resource "azurerm\_subnet" "subnet\_east" { |
| name = "default" |
| resource\_group\_name = "${azurerm\_resource\_group.resource-group-eastus.name}" |
| virtual\_network\_name = "${azurerm\_virtual\_network.vnet\_east.name}" |
| address\_prefix = "10.0.0.0/24" |
| } |
|  |
| # Create Dynamic Public IP Addresses |
| resource "azurerm\_public\_ip" "pip\_west" { |
| name = "pip-westus-contoso-${var.environment}" |
| location = "${azurerm\_resource\_group.resource-group-westus.location}" |
| resource\_group\_name = "${azurerm\_resource\_group.resource-group-westus.name}" |
| public\_ip\_address\_allocation = "dynamic" |
| } |
|  |
| resource "azurerm\_public\_ip" "pip\_east" { |
| name = "pip-eastus-contoso-${var.environment}" |
| location = "${azurerm\_resource\_group.resource-group-eastus.location}" |
| resource\_group\_name = "${azurerm\_resource\_group.resource-group-eastus.name}" |
| public\_ip\_address\_allocation = "dynamic" |
| } |

[**view raw**](https://gist.github.com/stephenweinrich/be974d406539a21da5fdc620ec584015/raw/6a768e4cc1c2c327ac8e1f625c251b764ee867f2/environment.tf)[**environment.tf**](https://gist.github.com/stephenweinrich/be974d406539a21da5fdc620ec584015#file-environment-tf) hosted with ❤ by [**GitHub**](https://github.com/)

Notice that the IP and DNS have NOT been assign to the Dynamic Public IP Address. This is because there is nothing attached to it. Once we create the gateway and attache the Public IP it will generate an address and DNS for us.



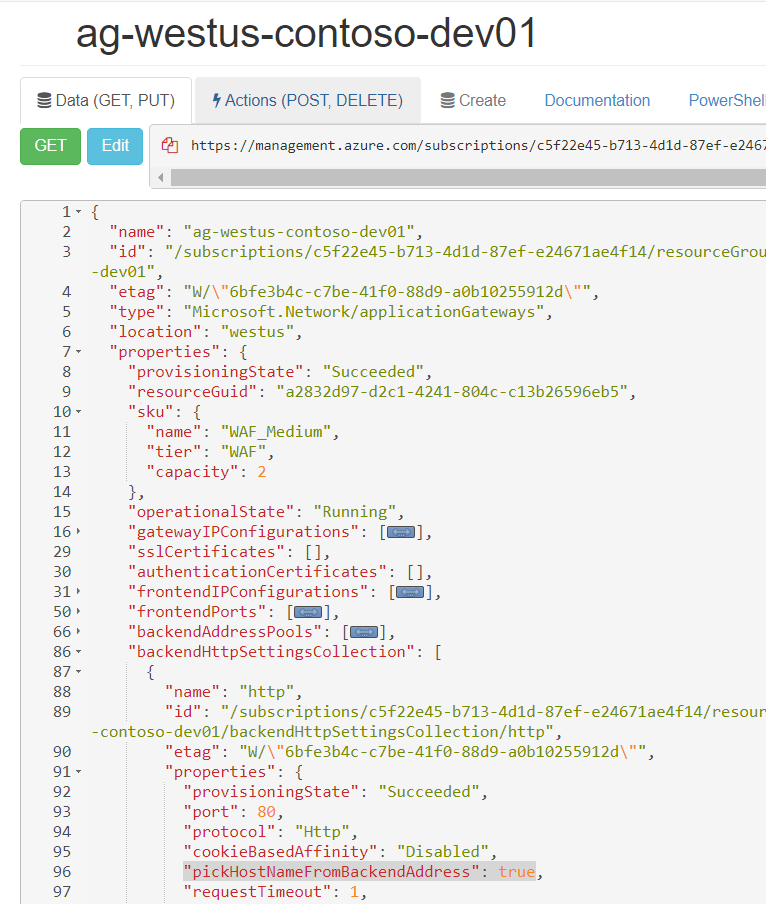
Use the following terraform code to create the Application Gateways:

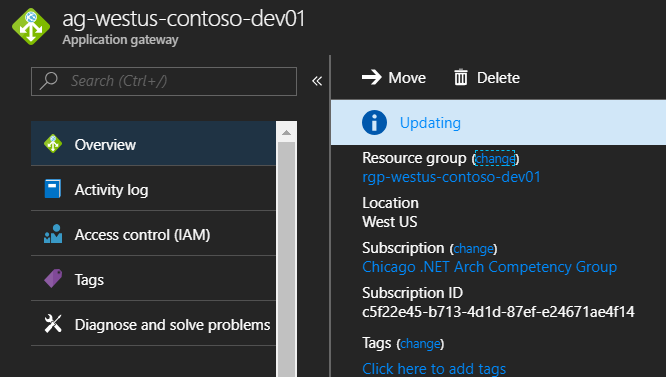
|  |
| --- |
| # Create Application Gateways |
| resource "azurerm\_application\_gateway" "application-gateway-west" { |
| name = "ag-westus-contoso-${var.environment}" |
| resource\_group\_name = "${azurerm\_resource\_group.resource-group-westus.name}" |
| location = "${azurerm\_resource\_group.resource-group-westus.location}" |
|  |
| sku { |
| name = "WAF\_Medium" |
| tier = "WAF" |
| capacity = 2 |
| } |
|  |
| waf\_configuration { |
| enabled = "true" |
| firewall\_mode = "Detection" |
| rule\_set\_type = "OWASP" |
| rule\_set\_version = "3.0" |
| } |
|  |
| gateway\_ip\_configuration { |
| name = "subnet" |
| subnet\_id = "${azurerm\_virtual\_network.vnet\_west.id}/subnets/${azurerm\_subnet.subnet\_west.name}" |
| } |
|  |
| frontend\_port { |
| name = "http" |
| port = 80 |
| } |
|  |
| frontend\_ip\_configuration { |
| name = "frontend" |
| public\_ip\_address\_id = "${azurerm\_public\_ip.pip\_west.id}" |
| } |
|  |
| backend\_address\_pool { |
| name = "AppService" |
| "fqdn\_list" = ["${azurerm\_app\_service.app-service-westus.name}.azurewebsites.net"] |
| } |
|  |
| http\_listener { |
| name = "http" |
| frontend\_ip\_configuration\_name = "frontend" |
| frontend\_port\_name = "http" |
| protocol = "Http" |
| } |
|  |
| probe { |
| name = "probe" |
| protocol = "http" |
| path = "/" |
| host = "${azurerm\_app\_service.app-service-westus.name}.azurewebsites.net" |
| interval = "30" |
| timeout = "30" |
| unhealthy\_threshold = "3" |
| } |
|  |
| backend\_http\_settings { |
| name = "http" |
| cookie\_based\_affinity = "Disabled" |
| port = 80 |
| protocol = "Http" |
| request\_timeout = 1 |
| probe\_name = "probe" |
| } |
|  |
| request\_routing\_rule { |
| name = "http" |
| rule\_type = "Basic" |
| http\_listener\_name = "http" |
| backend\_address\_pool\_name = "AppService" |
| backend\_http\_settings\_name = "http" |
| } |
| } |
|  |
| resource "azurerm\_application\_gateway" "application-gateway-east" { |
| name = "ag-eastus-contoso-${var.environment}" |
| resource\_group\_name = "${azurerm\_resource\_group.resource-group-eastus.name}" |
| location = "${azurerm\_resource\_group.resource-group-eastus.location}" |
|  |
| sku { |
| name = "WAF\_Medium" |
| tier = "WAF" |
| capacity = 2 |
| } |
|  |
| waf\_configuration { |
| enabled = "true" |
| firewall\_mode = "Detection" |
| rule\_set\_type = "OWASP" |
| rule\_set\_version = "3.0" |
| } |
|  |
| gateway\_ip\_configuration { |
| name = "subnet" |
| subnet\_id = "${azurerm\_virtual\_network.vnet\_east.id}/subnets/${azurerm\_subnet.subnet\_east.name}" |
| } |
|  |
| frontend\_port { |
| name = "http" |
| port = 80 |
| } |
|  |
| frontend\_ip\_configuration { |
| name = "frontend" |
| public\_ip\_address\_id = "${azurerm\_public\_ip.pip\_east.id}" |
| } |
|  |
| backend\_address\_pool { |
| name = "AppService" |
| "fqdn\_list" = ["${azurerm\_app\_service.app-service-eastus.name}.azurewebsites.net"] |
| } |
|  |
| http\_listener { |
| name = "http" |
| frontend\_ip\_configuration\_name = "frontend" |
| frontend\_port\_name = "http" |
| protocol = "Http" |
| } |
|  |
| probe { |
| name = "probe" |
| protocol = "http" |
| path = "/" |
| host = "${azurerm\_app\_service.app-service-eastus.name}.azurewebsites.net" |
| interval = "30" |
| timeout = "30" |
| unhealthy\_threshold = "3" |
| } |
|  |
| backend\_http\_settings { |
| name = "http" |
| cookie\_based\_affinity = "Disabled" |
| port = 80 |
| protocol = "Http" |
| request\_timeout = 1 |
| probe\_name = "probe" |
| } |
|  |
| request\_routing\_rule { |
| name = "http" |
| rule\_type = "Basic" |
| http\_listener\_name = "http" |
| backend\_address\_pool\_name = "AppService" |
| backend\_http\_settings\_name = "http" |
| } |
| } |

Unfortunately, at the time of writing, App Services and Application Gateway are not fully supported. There is an additional boolean flag that needs to be set. This can only be done via powershell or resource explorer (api), it is not possible to set this flag through terraform or the azure portal. If you know an alternative please correct me. I've spent hours trying to figure this out.

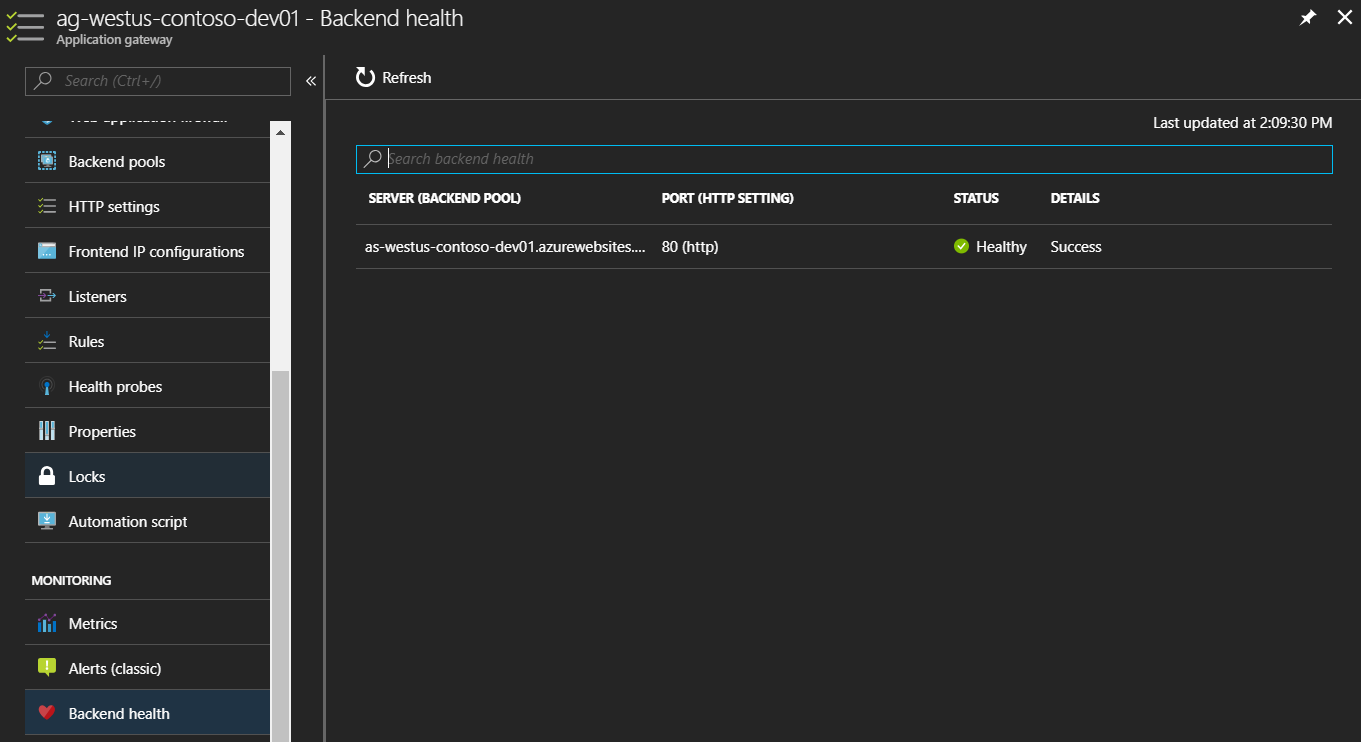
For my example I used Azure Resource Explorer which can be used to edit specific values on your resources. Navigate to your subscription --> resource group --> providers --> Microsoft.Network --> your gateway

Look for the **pickHostNameFromBackendAddress** click Edit and set the value to true, click PUT to update. Browse back to application gateway on the portal and wait for the update to complete.

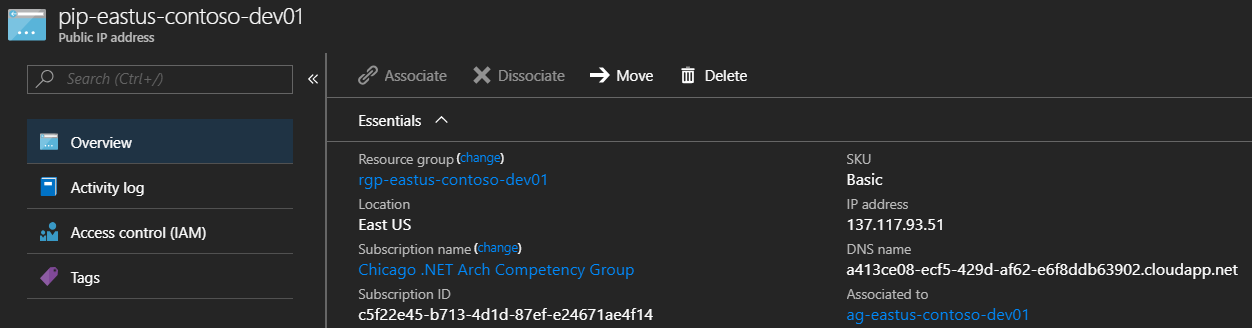




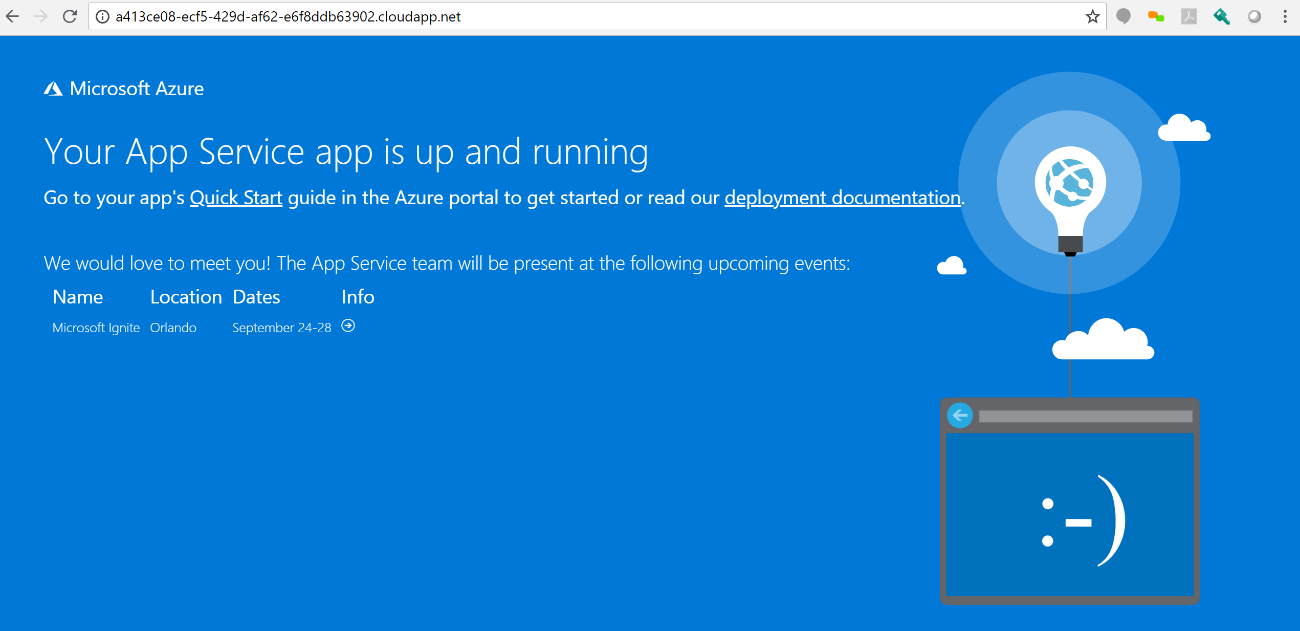
Select Backend Health on the Application Gateway blade, you should now see a healthy status.



Navigate back to the Dynamic IP address that we were looking at earlier, the one that is attached to the Frontend off your Application Gateway. You will now notice it has an IP address and a generated DNS name. This is because it is not attached to the gateway.



Browse to the DNS name provided and your web site should be displayed.



Go ahead and test both the west and east gateway DNS to ensure both are working.

The next step would be to setup Traffic Manager to route traffic to the gateways.

**Traffic Manager**

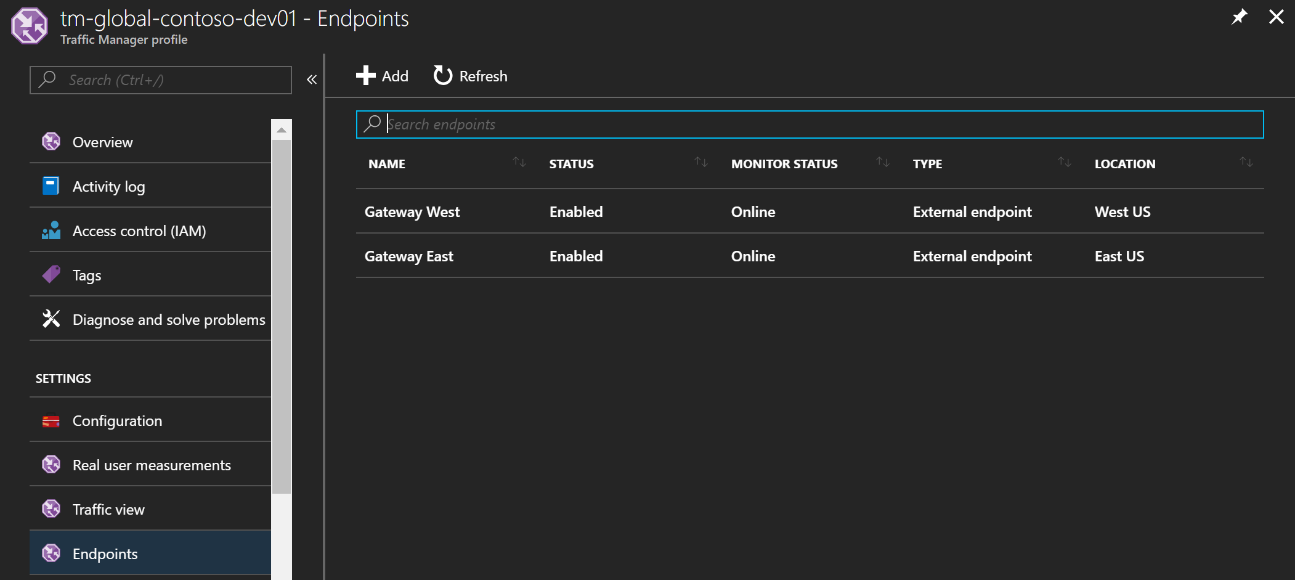
Use the following Terraform code to create the Traffic Manager Profile and an Endpoint for each Gateway.

|  |
| --- |
| # Create Traffic Manager API Profile |
| resource "azurerm\_traffic\_manager\_profile" "traffic-manager" { |
| name = "tm-global-contoso-${var.environment}" |
| resource\_group\_name = "${azurerm\_resource\_group.resource-group-global.name}" |
| traffic\_routing\_method = "Performance" |
|  |
| dns\_config { |
| relative\_name = "tm-global-contoso-${var.environment}" |
| ttl = 300 |
| } |
|  |
| monitor\_config { |
| protocol = "http" |
| port = 80 |
| path = "/" |
| } |
| } |
|  |
|  |
| # Create Traffic Manager - West End Point |
| resource "azurerm\_traffic\_manager\_endpoint" "tm-endpoint-west" { |
| name = "Gateway West" |
| resource\_group\_name = "${azurerm\_resource\_group.resource-group-global.name}" |
| profile\_name = "${azurerm\_traffic\_manager\_profile.traffic-manager.name}" |
| type = "externalEndpoints" |
| target = "${azurerm\_public\_ip.pip\_west.fqdn}" |
| endpoint\_location = "${azurerm\_public\_ip.pip\_west.location}" |
| } |
|  |
| # Create Traffic Manager - East End Point |
| resource "azurerm\_traffic\_manager\_endpoint" "tm-endpoint-east" { |
| name = "Gateway East" |
| resource\_group\_name = "${azurerm\_resource\_group.resource-group-global.name}" |
| profile\_name = "${azurerm\_traffic\_manager\_profile.traffic-manager.name}" |
| type = "externalEndpoints" |
| target = "${azurerm\_public\_ip.pip\_east.fqdn}" |
| endpoint\_location = "${azurerm\_public\_ip.pip\_east.location}" |
| } |

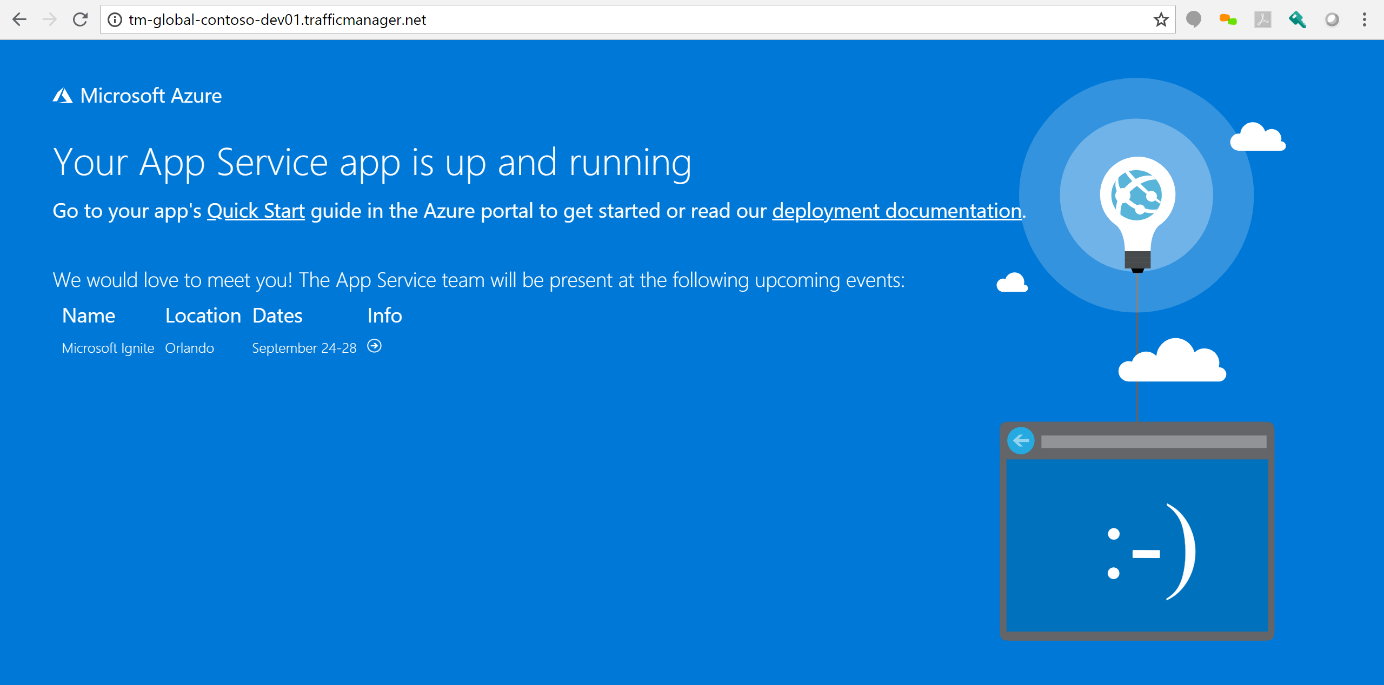
Traffic Manager Endpoints supports different types. I tried using *azureEndpoints* and then specifying the public IP resource but Terraform timed out with a 500 for some reason.

I ended up using *externalEndpoints* and then specifying the FQDN of the public IP resources which worked just fine.

Traffic Manager will monitor the health of your endpoints:



Browsing to the DNS name for the Traffic Manager should now display your site relative to your location:



**Enabling HTTP2**

Enabling HTTP2 on the Application Gateway is currently not support by Terraform.

You can use the following PowerShell script to enable HTTP2:

|  |
| --- |
| # Signing into azure with principal account |
| $clientID = "<--client-id-->" |
| $key= "<--secret-key-->" |
| $SecurePassword = $key | ConvertTo-SecureString -AsPlainText -Force |
| $cred = new-object -typename System.Management.Automation.PSCredential ` -argumentlist $clientID, $SecurePassword |
| Add-AzureRmAccount -Credential $cred -Tenant "<--tenant-id-->" -ServicePrincipal |
|  |
| # Selecting Subscription |
| Select-AzureRmSubscription "<--subscription-id-->" |
|  |
| # enabled HTTP2 on app gateway |
| $gw = Get-AzureRmApplicationGateway -Name "<--application-gateway-name-->" -ResourceGroupName "<--resource-group-name-->" |
| $gw.EnableHttp2 = $true |
| Set-AzureRmApplicationGateway -ApplicationGateway $gw |

[**view raw**](https://gist.github.com/stephenweinrich/1d22e0caed7c4b64d7c326f0bb95433d/raw/50fcd7fb347cb49a4b77c86f22e70d6d93084400/enable-http2-gateway.ps1)[**enable-http2-gateway.ps1**](https://gist.github.com/stephenweinrich/1d22e0caed7c4b64d7c326f0bb95433d#file-enable-http2-gateway-ps1) hosted with ❤ by [**GitHub**](https://github.com/)

Configure using the same Principal credentials as Terraform and execute for both East and West gateway.

This will only work once you have configured HTTPS since that is a requirement for HTTP2.

**Limitations**

I thought I'd talk about some of the limitations that I found during this build out as well as some points of interest.

**pickHostNameFromBackendAddress Flag**

This is an obvious limitation, I'm hoping that soon there will be a selection in the Azure Portal for an App Service when adding to your backend pool instead of using the FQDN. Selecting this option should then set the pickHostNameFromBackendAddress flag for you automatically.

Ideally Terraform could also provide this setting to make automation easier and more consistent.

The only recommendation Microsoft documentation currently has for doing this is to use powershell.

You can read about it here.

**Http vs Https**

In my example I have configured only for HTTP. In reality you are going to want to use HTTPS. This is easy enough to configure on the DNS and App Service side using a CName and adding a Custom Domain. Best practice then is to terminate SSL at the Application Gateway level by attaching your SSL Certificate to a HTTPS listener on port 443. You can then route traffic to http (port 80) on your App Service and change your Application Gateway HTTP listener to route to the HTTPS listener.

One limitation is that Terraform currently does not support the option to route from one listener to another. So in terms of automation you may need to write some powershell for that.

**IP Restrictions**

One of the reasons for using the Application Gateway is to prevent users from directly accessing your App Services. You can use IP Restriction on the Networking blade for this, providing the IP address of your Application Gateway.

A few things to note here is that if you have any CI/CD build servers deploying code you may need to allow that IP address access to your App Service too.

Currently neither Terraform nor Powershell support the addition of IP Restrictions for App Services. There is custom powershell to do this which I submitted as a PR to MSDN documentation. The response I received was that a Powershell module was coming out soon for this.

You can use the following PowerShell script to configure IP restrictions:

|  |
| --- |
| # Signing into azure with principal account |
| $clientID = "<--client-id-->" |
| $key= "<--secret-key-->" |
| $SecurePassword = $key | ConvertTo-SecureString -AsPlainText -Force |
| $cred = new-object -typename System.Management.Automation.PSCredential ` -argumentlist $clientID, $SecurePassword |
| Add-AzureRmAccount -Credential $cred -Tenant "<--tenant-id-->" -ServicePrincipal |
|  |
| # Selecting Subscription |
| Select-AzureRmSubscription "<--subscription-id-->" |
|  |
| $ResourceGroupName = "<-resource-group-name->" |
| $WebAppName = "<-web-app-name->" |
|  |
| $Resource = Get-AzureRmResource -ResourceGroupName $ResourceGroupName -ResourceType Microsoft.Web/sites/config -ResourceName $WebAppName/web -ApiVersion 2016-08-01 |
|  |
| $Properties = $Resource.Properties |
| $Properties.ipSecurityRestrictions = @() |
|  |
| # IP Restriction 1 |
| $Restriction = @{} |
| $Restriction.Add("ipAddress","127.0.0.1") |
| $Restriction.Add("subnetMask","255.255.255.255") |
| $Properties.ipSecurityRestrictions += $Restriction |
|  |
| # IP Restriction 2 |
| $Restriction = @{} |
| $Restriction.Add("ipAddress","127.0.0.2") |
| $Restriction.Add("subnetMask","255.255.255.255") |
| $Properties.ipSecurityRestrictions += $Restriction |
|  |
| Set-AzureRmResource -ResourceGroupName $ResourceGroupName -ResourceType Microsoft.Web/sites/config -ResourceName $WebAppName/web -ApiVersion 2016-08-01 -PropertyObject $Properties -force |

**WAF Detection vs Prevention**

In this example the WAF is set to *Detection* which will only log attacks (if configured) but not prevent access to the gateway. Ideally you want to set the WAF to *Prevention* mode. Doing this however prevents all traffic for me with my current setup and I still need to figure out why. There is most likely one or 2 of the OWASP 3.0 rules that are being broken. My plan is to configure logging which should identify the rules being broken. These rules could either be disabled (if that makes sense) or the root cause could be addressed.

**Traffic Manager**

When I was initially testing this setup I tried using Traffic Manager directly to my App Services (bypassing the Application Gateway). An interesting learning is that Traffic Manager will only support the Standard and above tiers. Using the Free tier will allow you to add the App Services as endpoints but traffic will not route correctly.

Another friendly tip is that during testing dns caching is a factor. Clear your cache and flush your DNS, use Chrome Incognito mode and test with multiple browsers. Often I had actually configured correctly but cache from previous requests were letting me down.

**Conclusion**

Traffic Manager, Application Gateway (WAF) with App Services can be used to provide a secure and efficient multi-region deployment option for your PaaS applications.

Using Terraform to help deploy your infrastructure is powerful but will only get you 80% of the way there. Be prepared to write post deployment Powershell scripts to supplement the limitation of Terraform.

Configure for HTTPS and take advantage of HTTP2 which can be enabled on the Application Gateway. Consider terminating HTTPS at the gateway level that way you wont need to upload a certificate for each App Service.

Set the WAF to Prevention instead of Detection to protect you from DDOS, XSS and SQL Injection attacks.

Restrict direct access to your App Services so that users cannot bypass the Application Gateway.