Contents

[AZURE APP SERVICE 2](#_Toc155823929)

[WHY USE APP SERVICE? 2](#_Toc155823930)

[CREATING AN APP SERVICE 2](#_Toc155823931)

[SIMPLE CHANGES IN THE WEB APP 3](#_Toc155823932)

[AZURE APP SERVICE PLAN 4](#_Toc155823933)

[UNDERDTANDING THE BASIC APP SERVICE PLAN 4](#_Toc155823934)

[TYPES OF APP SERVICE PLAN 5](#_Toc155823935)

[APP SERVICE PLAN - CONFIGURATION 5](#_Toc155823936)

[SCALING WEB APP 6](#_Toc155823937)

[AUTOSCALING THE AZURE WEB APP 7](#_Toc155823938)

[IMPLEMENTING AUTOSCALING THE AZURE WEB APP 7](#_Toc155823939)

[DEPLOYMENT 9](#_Toc155823940)

[DEPLOYMENT SLOTS 9](#_Toc155823941)

[DEPLOYMENT CENTER 11](#_Toc155823942)

[CUSTOM DOMAIN 13](#_Toc155823943)

[AZURE WEB APP LOGGING 17](#_Toc155823944)

[SETTING UP LOGGING FOR WEB APP 17](#_Toc155823945)

[LIVE LOGS 18](#_Toc155823946)

[AZURE WEB APP - VIRTUAL NETWORK INTEGRATION FEATURE 19](#_Toc155823947)

[SETTING UP VIRTUAL NETWORK INTERGRATION 20](#_Toc155823948)

[AZURE APP SERVICE BACKUP 20](#_Toc155823949)

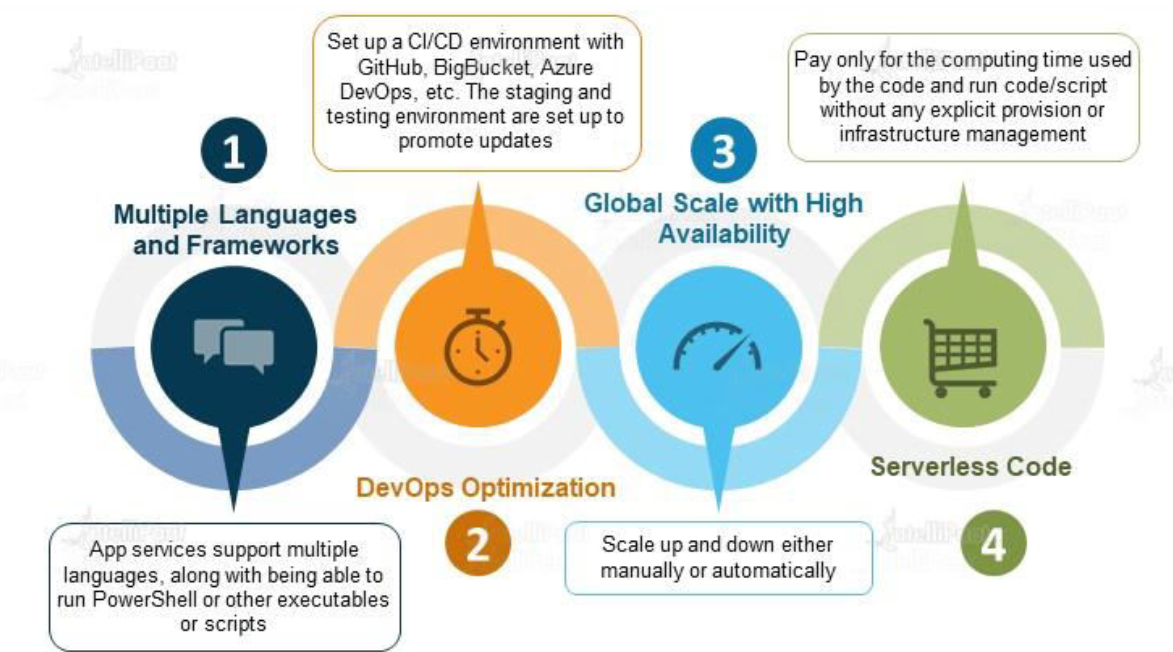
[AZURE APP SERVICE USE CASE 21](#_Toc155823950)

[TRAINING SCREEN SHOTS 21](#_Toc155823951)

# AZURE APP SERVICE

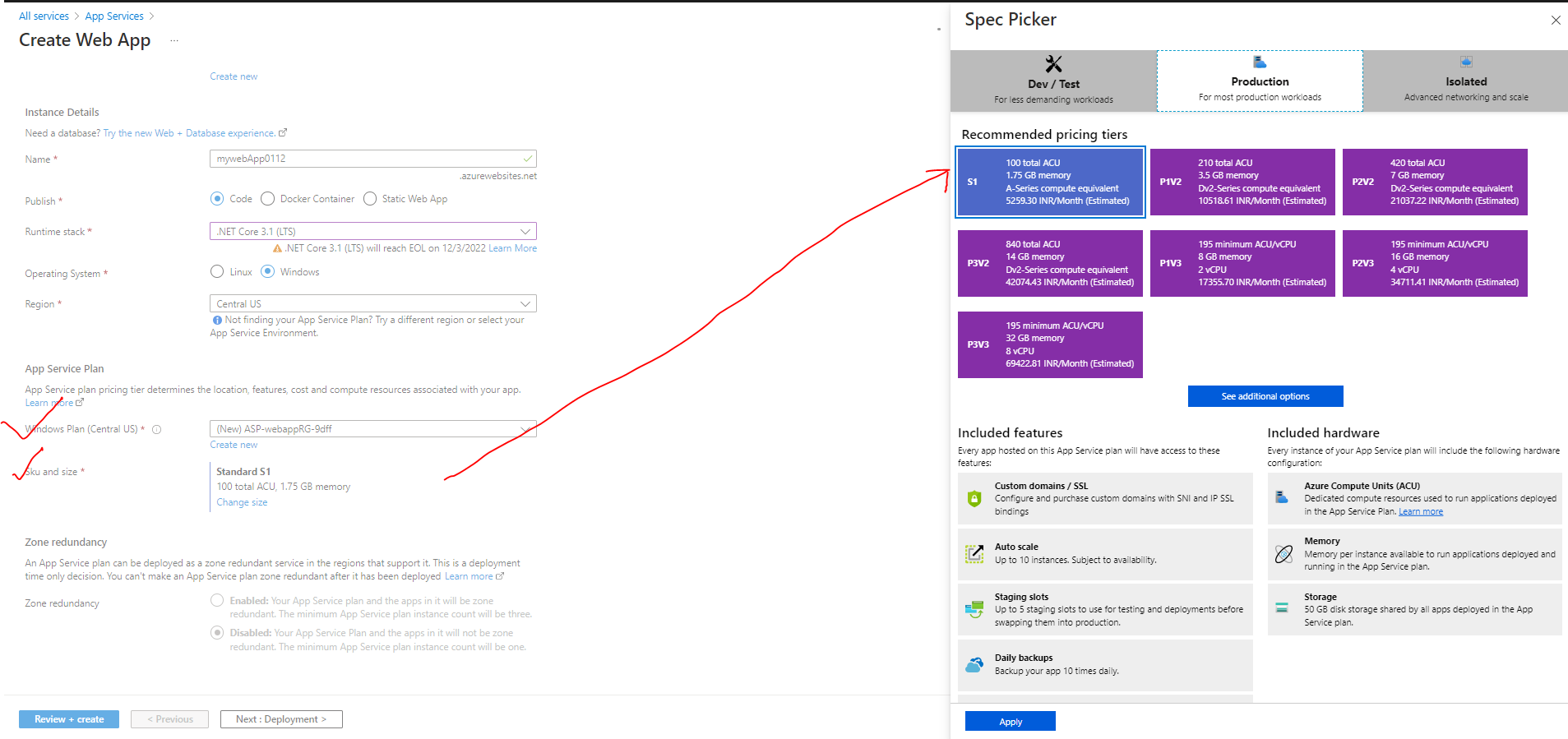
|  |  |
| --- | --- |
| * Web App Service is a **platform-as-a-service(PAAS)** where the undelying infrastructure is managed any Azure itself. * This is an HTTP-based service used for hosting web applications. * The web application can developed in languages like .NET, .NET Core, Java, Ruby, Node.js or Python which can run on Windows or Linux-based platforms. * Since is a PAAS solution, the application gets deployed on an underlying compute infrastructure (VM). The entire infrastructure is managed by Azure i.e., by the Azure App Service. * Azure webapps has a feature such as auto scaling and security and DevOps capabilities |  |

## WHY USE APP SERVICE?



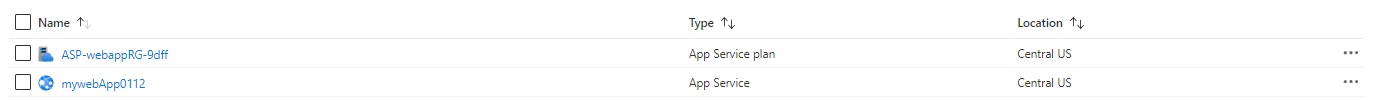
## CREATING AN APP SERVICE

* Select “Web App” Service
* This wizard will create 2 resources
  + The Web App itself
  + App Service Plan



|  |  |
| --- | --- |
| NAME | * Unique name of the app. By default, azure append a default DNS name to the web app. We can have custom DNS name as well. |
| PUBLISH | * Type of App |
| RUNTIME STACK | * Runtime of the of application is going to be hosted |
| OPERATING SYSTEM | * The operating system that will be installed on the underlying VM which will host the webapp |
| WINDOWS PLAN | * While create a web app – an app service plan is created. * We can also to link it the web app to the existing app service plan |
| SKU / SIZE | * Here we select the type of SKU/ Size available for a given app service plan |

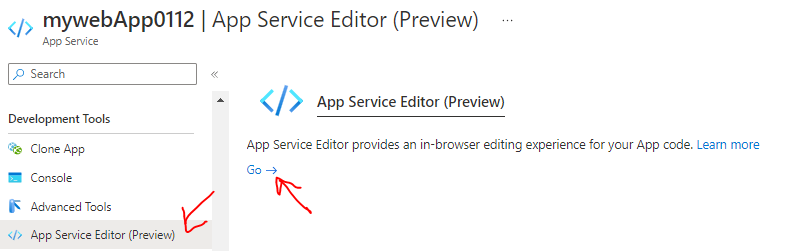
* This creates 2 resources i.e the App service and App service plan. Now if we want to create a another web app (which will use the same underlying OS ), then we can make use of same app service plan

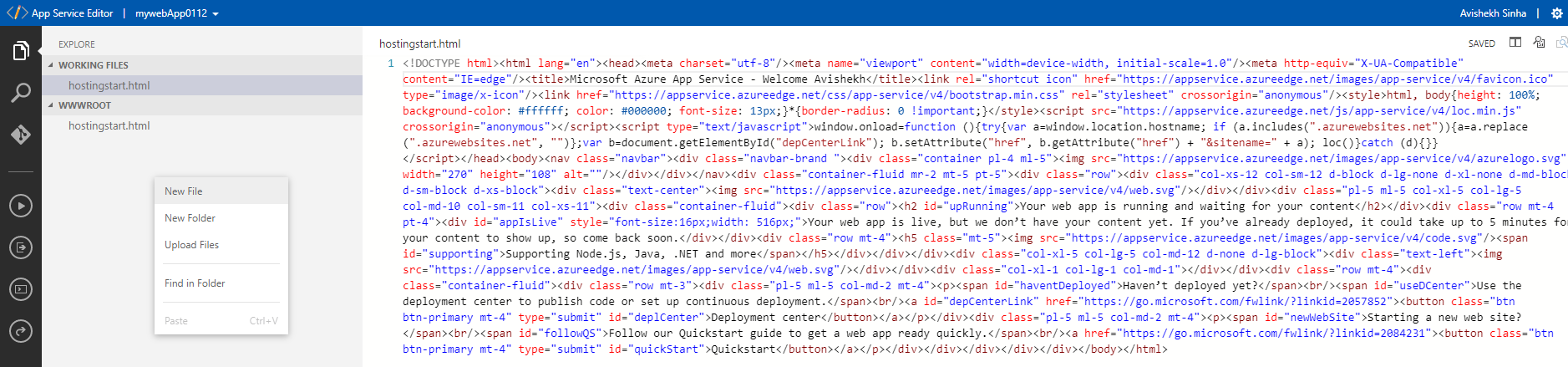


|  |  |
| --- | --- |
|  | * Once the setup is done. Azure will host a default “.NET Core” based application on Azure webapp service. For our case it can be accessed at - <https://mywebapp0112.azurewebsites.net> |

### SIMPLE CHANGES IN THE WEB APP

* For making small scale changes we can make use of App Service Editor. For example – updating an­­­­­­­ existing HTML, adding new HTML files etc.





## AZURE APP SERVICE PLAN

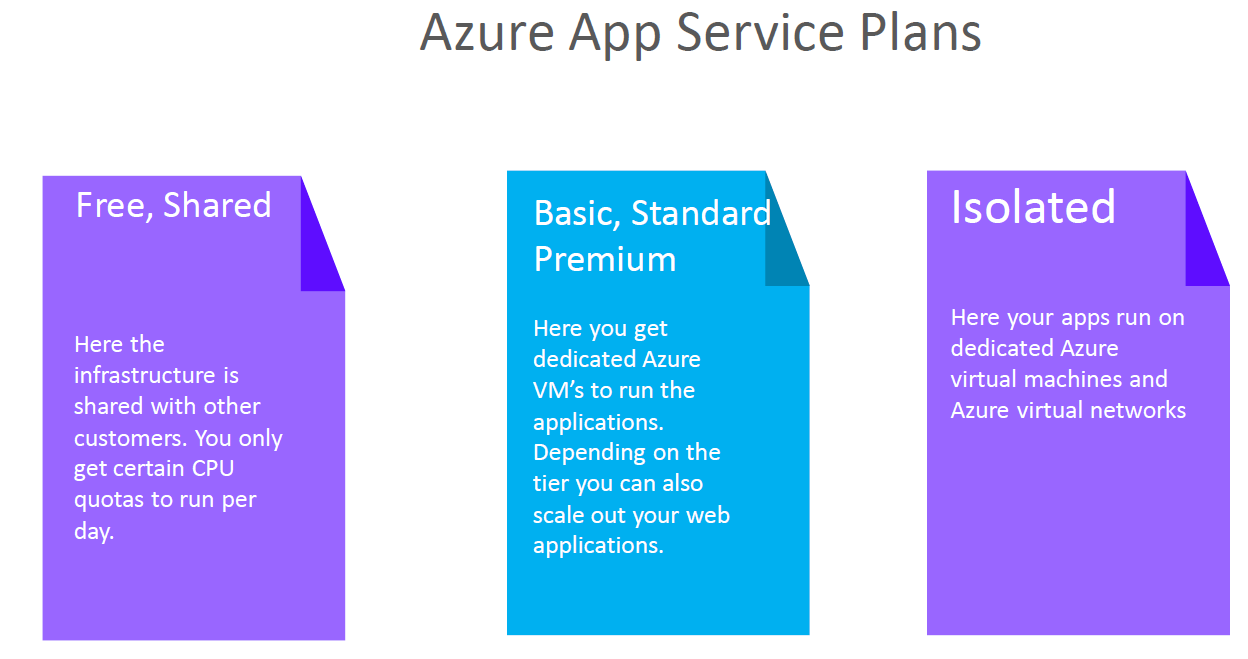


* Azure Web app service are always linked to an app service plan. For the compute resources needed for the App Service are specified within the App Service Plan. Hence - The billing of an app service depends upon the type of app service plan that we choose ( <https://azure.microsoft.com/en-us/pricing/details/app-service/windows/#pricing> )

### UNDERDTANDING THE BASIC APP SERVICE PLAN

|  |  |
| --- | --- |
|  | * In Basic plan we can deploy unlimited web app and storage capacity of 10 GB in total. We can be able to scale up to 3 VM instances. * Let's say we create an Azure app service plan (basic pricing tier). * In this plan - we can create unlimited Azure Web applications and link it to the app service plan, * As Azure Web app service is a PAAS so the infrastructure that is required for hosting the Web applications get created as well   **USE CASE**   * Let's say that the app service plan created one compute instance to host two Web applications, * Now after that - we add a third Web application. If the compute instance is not enough to withstand the load of all the three web applications. * As per the basic plan – the underlying VM can scale up to 3 instances. Hence now application can be distributed across these compute instances and the load gets distributed accordingly.   **NOTE**   * ***We don't have to create multiple app service plans – we can have one App service plan and have multiple Web applications as part of that app service plan.*** * But this decision also depends on what is the underlying Programming language for that Web application. For example, we can create an app service plan that spins up Windows based virtual machines. This Windows based virtual machine will not be able to run applications that are dependent on Linux, if we want to host Linux based web applications, then we need to create another separate app service plan. |

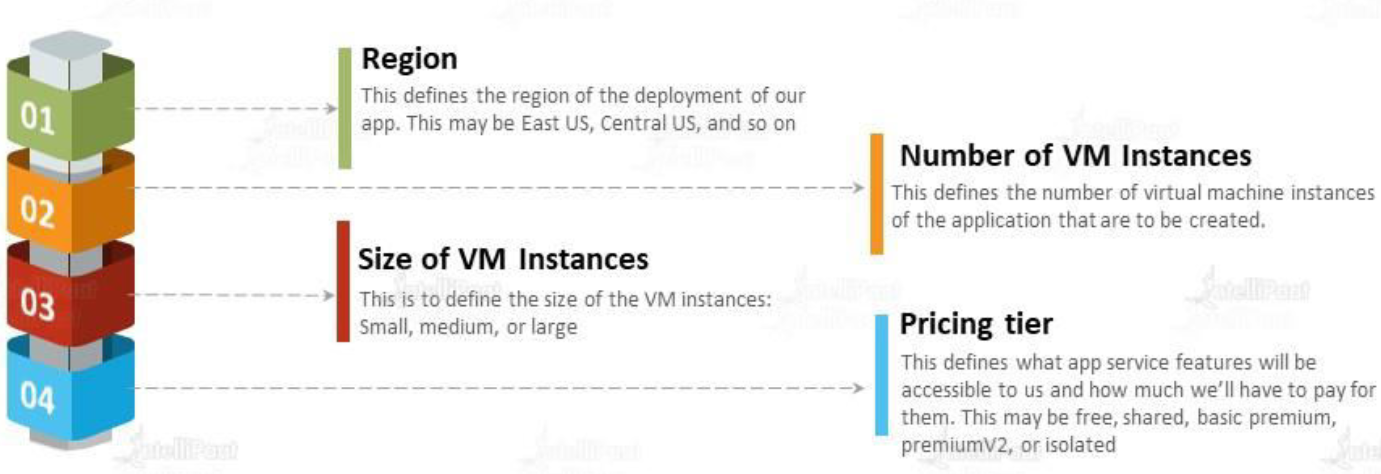
### TYPES OF APP SERVICE PLAN





* In the above pricing table of app service plan show different features associated with different App service plans.

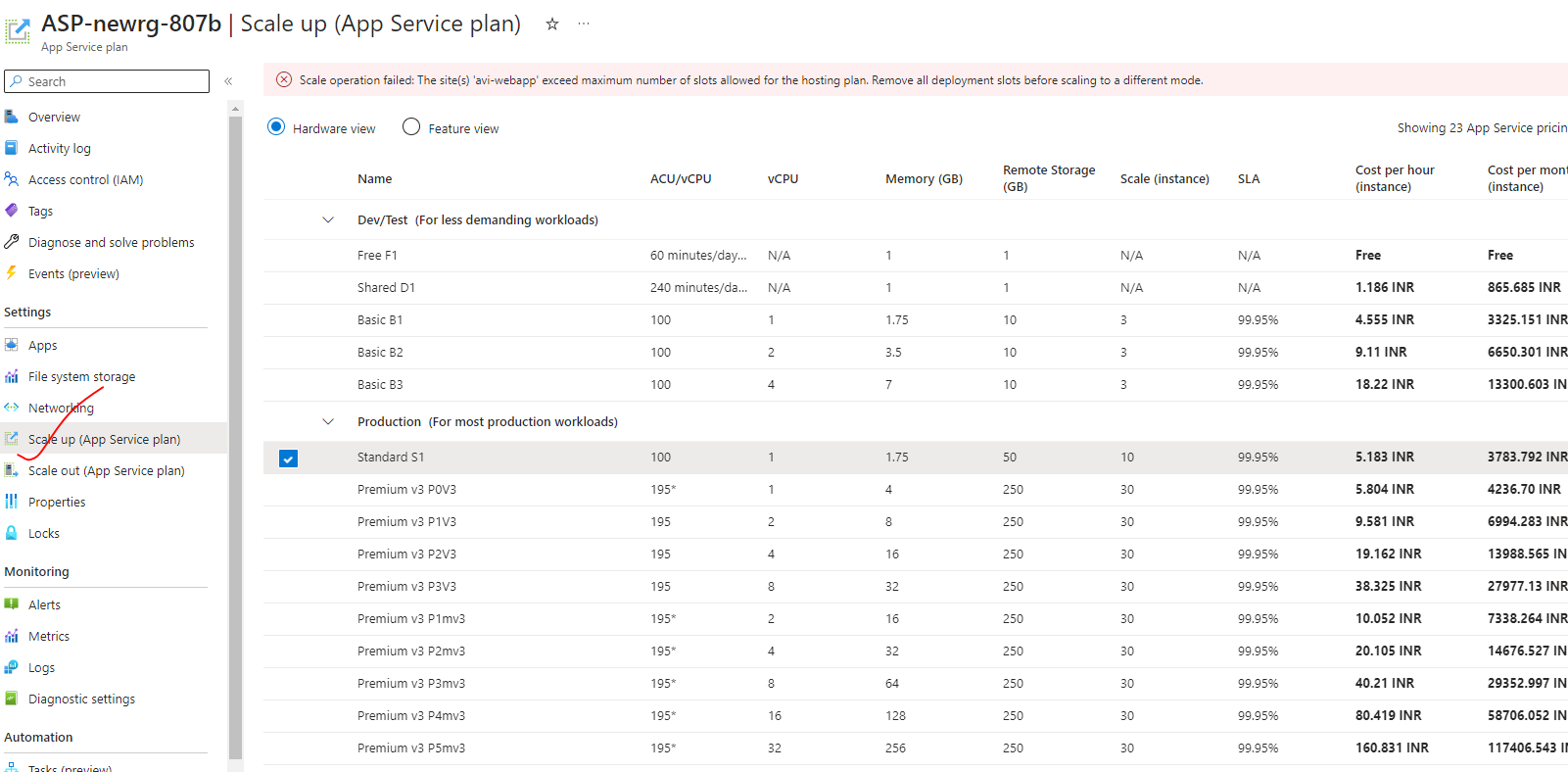
### APP SERVICE PLAN - CONFIGURATION



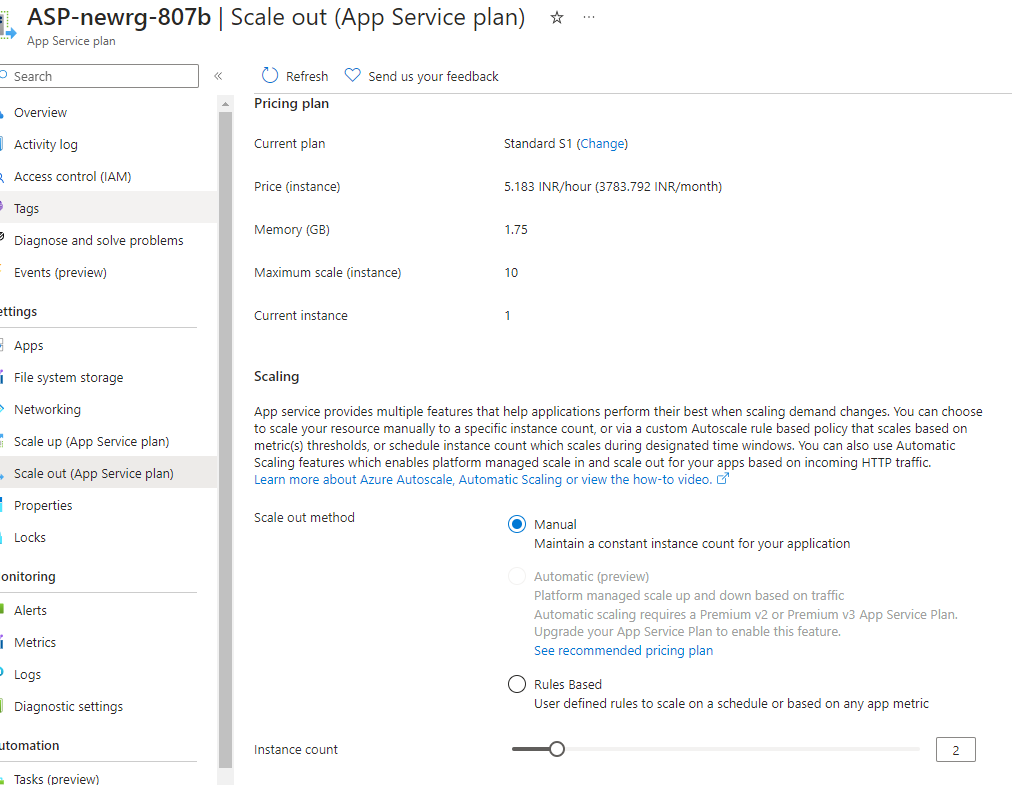
### SCALING WEB APP

|  |  |
| --- | --- |
|  | * For scale up/ down in web app – we update / downgrade the app service * In case of scale out /in – we need to update the number of instance of compute resources.   ***Note – the compute resource of the Azure web app is VMSS behind the scene.*** |

#### SCALE UP CONFIGURATION



#### SCALE OUT CONFIGURATION

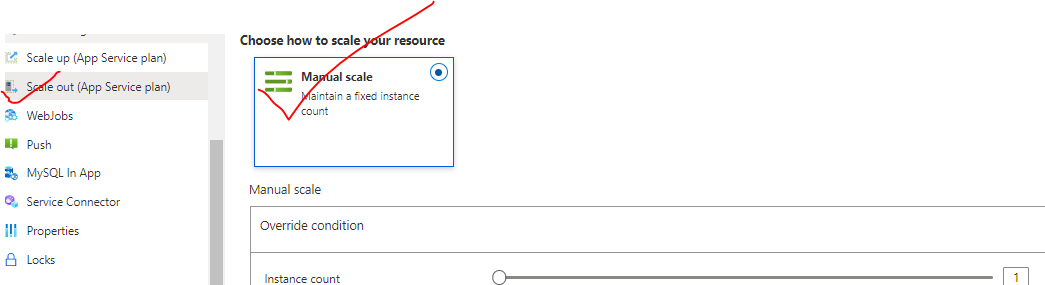


## AUTOSCALING THE AZURE WEB APP

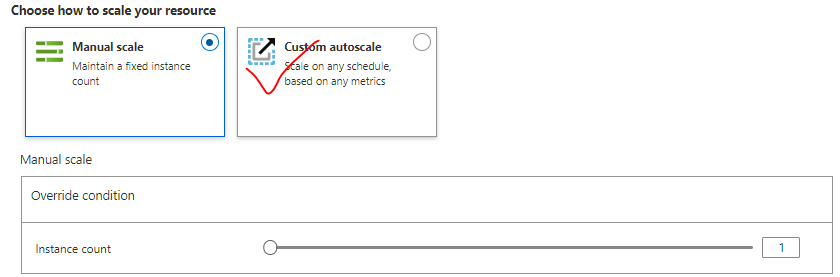
* We get the autoscaling feature if the app service plan is *Standard App Service plan or higher*.
* The autoscaling feature is useful when the traffic increases to the webapp.
* Scaling out add more compute resources which help in distributing the load across these virtual machines.
* Note –
  + The basic up service plan, we scale out to three virtual machines but it’s a manual procedure.
  + For auto scaling we need to choose the standard service plan or higher.
* We need to create rules and conditions to trigger the scaling process. For example, we can create a condition based on the percentage of the underlying compute infrastructure.
  + Example - if the C.P.U goes beyond a particular limit, we can scale out the number of machines automatically.
* If the load decreases, it should scale in the number of virtual machines.

### IMPLEMENTING AUTOSCALING THE AZURE WEB APP

* For basic app plan – We can do only manual scaling

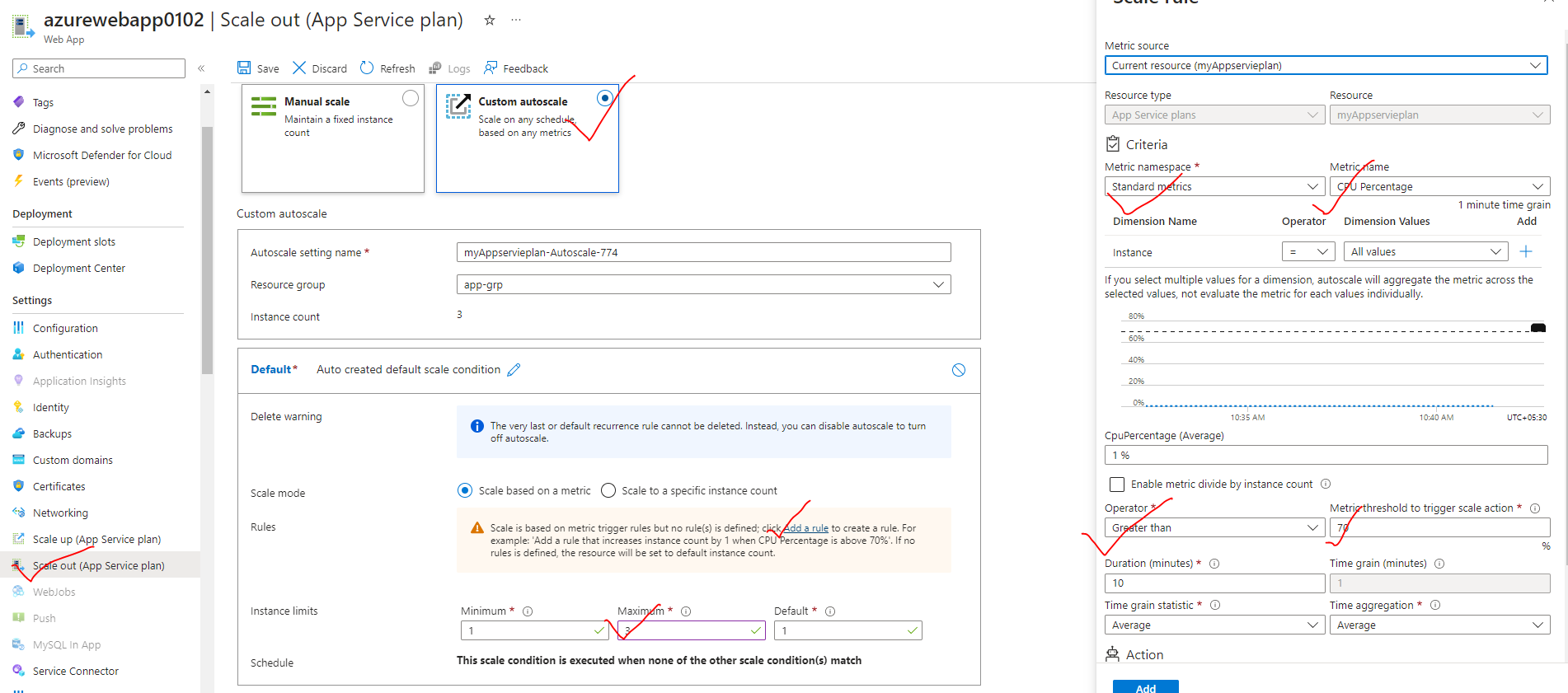


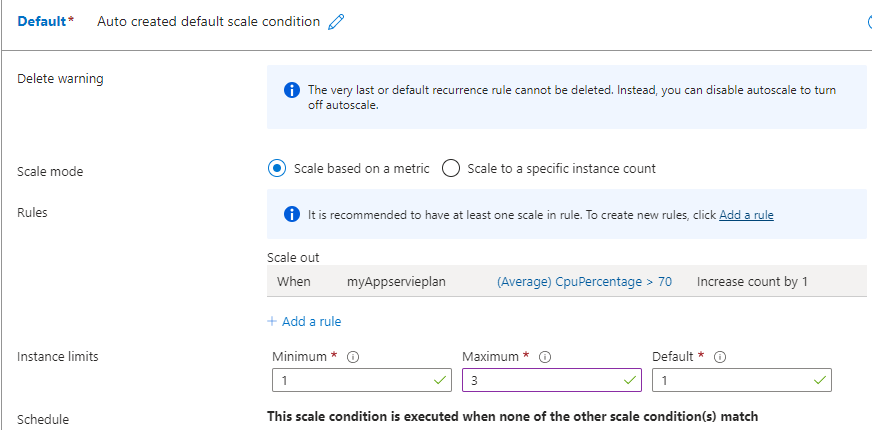
* For Standard and higher app plans – auto scaling feature is available.



#### AUTOSCALING CONDITON

Condition to Scale out when CPU percentage exceeds 70% utilization

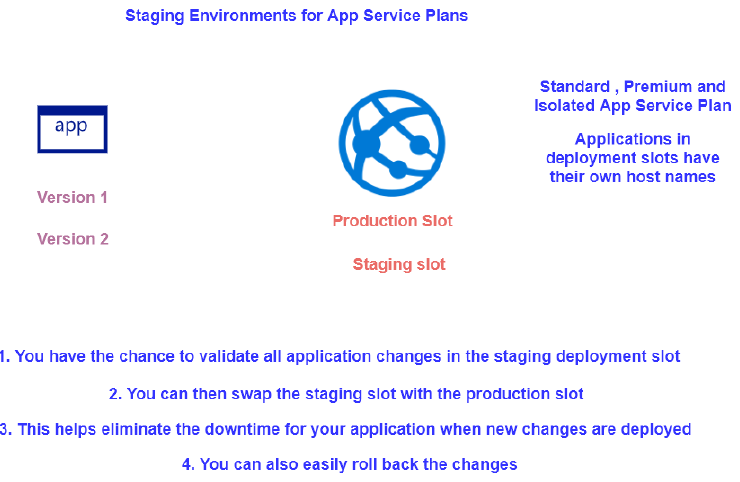




## DEPLOYMENT

### DEPLOYMENT SLOTS

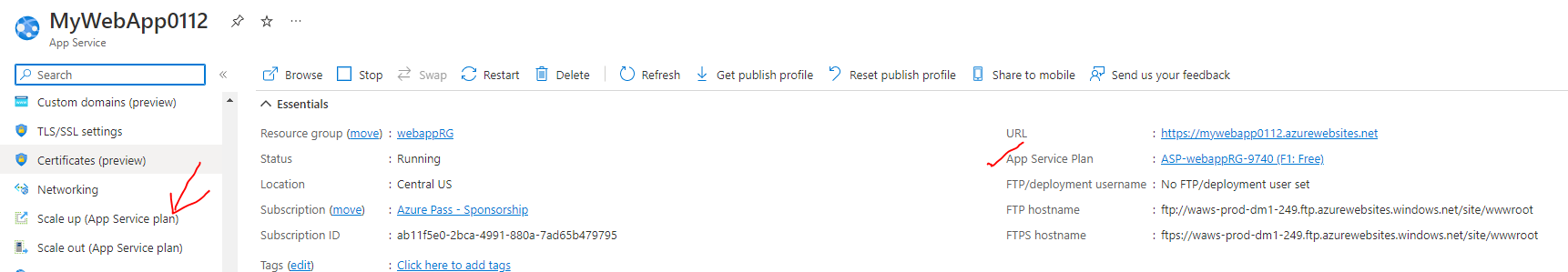
* Azure deployment slots allow us to run multiple versions of the application code on the same infrastructure.
* For example- Let's say we have a web application (*MyWebApp*) deployed in azure. By default, it gets an URL (mywebapp.azurewebsites.net). Considering it as our production version of the application then users can access and use it.
* Now, let's say we need to make a change to the application. (Maybe, there is a bug or adding a new feature). After the change we want to test these changes in production like environment.

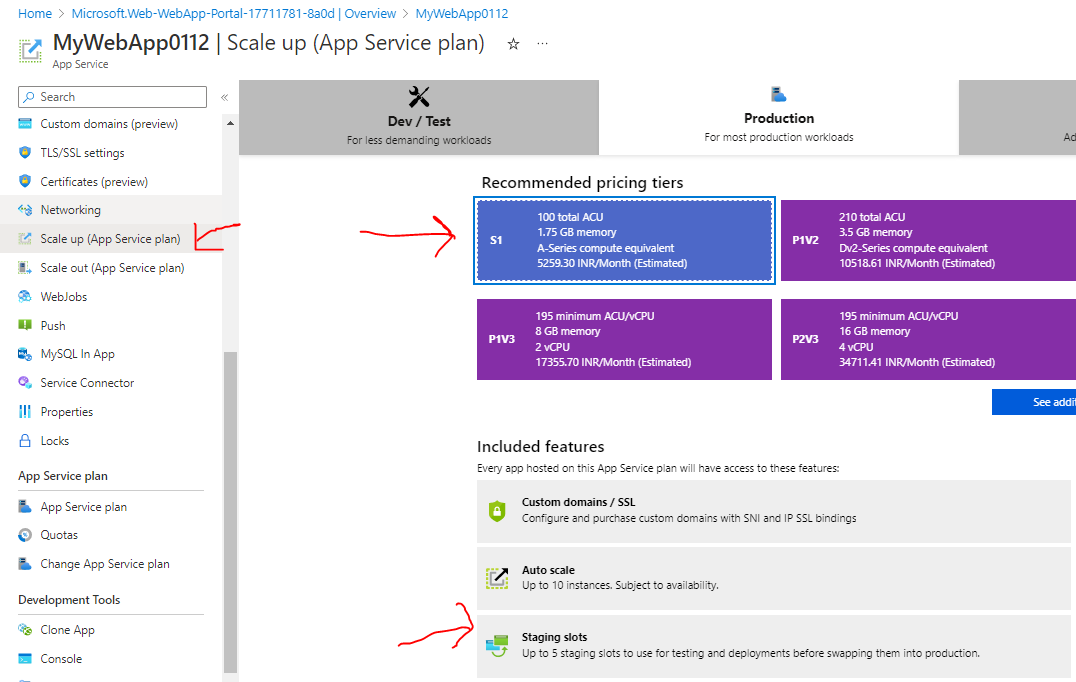


|  |  |
| --- | --- |
| azure staging slots | * We can create a deployment slot (e.g., **staging**) and deploy the changed version of our application. For example, if we name the deployment slot - staging, then we get the following URL. - mywebapp-staging.azurewebsites.net * The new version of the app is now in the staging slot. It gives you an opportunity to test our application. We all test looks good, we can swap the staging deployment slot with the production deployment slot. |

#### CREATING A DEPLOYMENT SLOTS

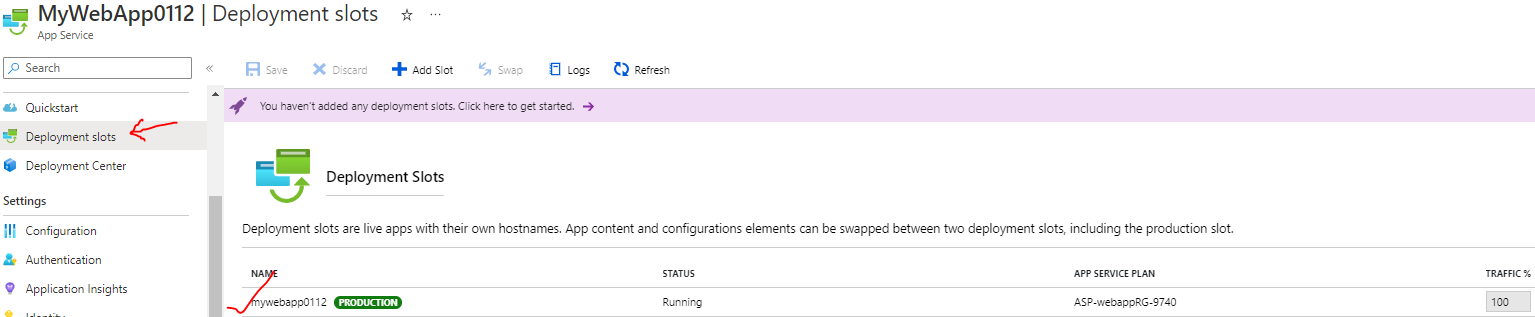
* To make use of deployment slot, the app service plan attached to the web app service should be Standard or higher. Note – we can be able to update the existing app service plan using “Scale Up” option.
* Let’s say we already have a web app tied to a Free app service plan, to update the app service plan – we can make use of Scale up option.





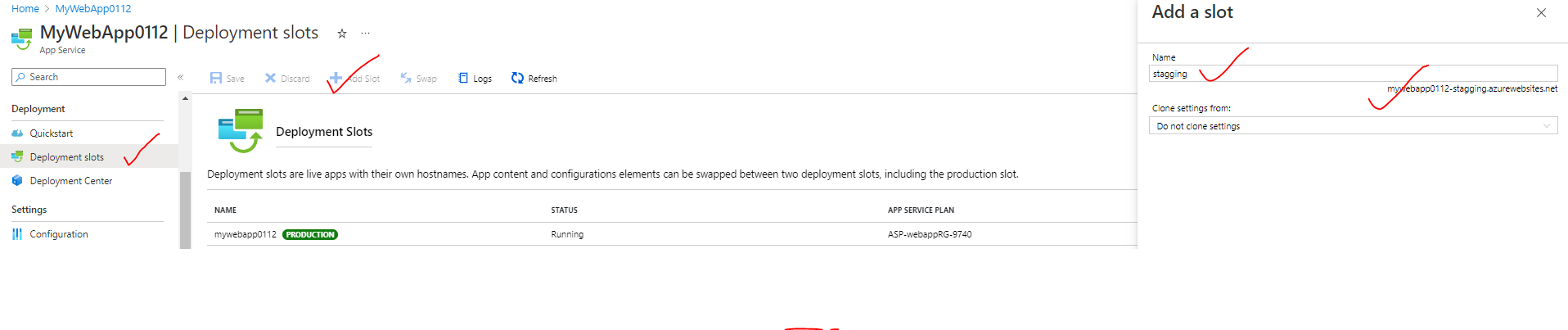
PROD VERSION OF APPLICATION

* ***PROD URL (prod version)***- <https://mywebapp0112.azurewebsites.net/>



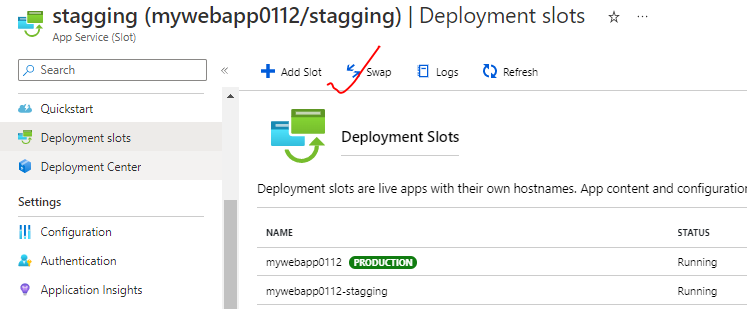
STAGE VERSION

* To deploy the newer version of the application as a new web app – In this app itself, we can create a deployment slot
* ***STAGE URL(new version)*** - <https://mywebapp0112-stagging.azurewebsites.net/>



SWAPING

* Once the testing is done of the newer version. We can now swap the deployment slots . Hence the PROD version will become STAGE version and vice versa.

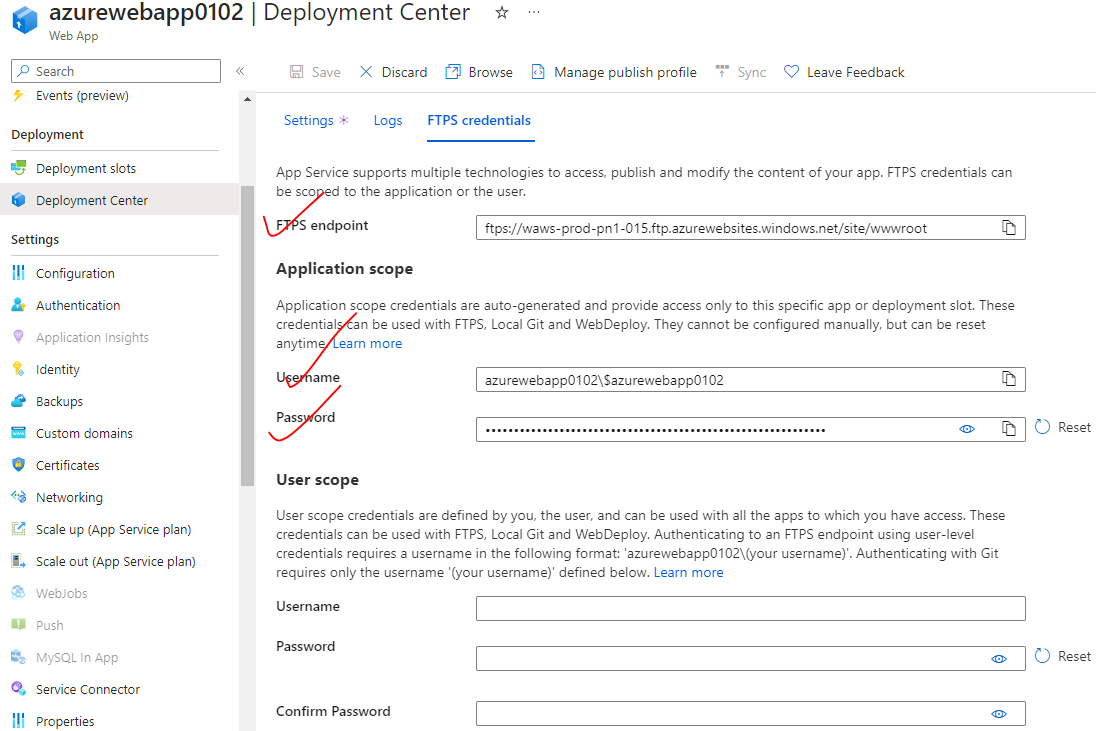


### DEPLOYMENT CENTER

For the deployment of the Web App, we make use of Deployment Center. The deployment can be automated or manually

#### MANUAL DEPLOYMENT

* The manual deployment can be done using FTP (tools like winSCP)

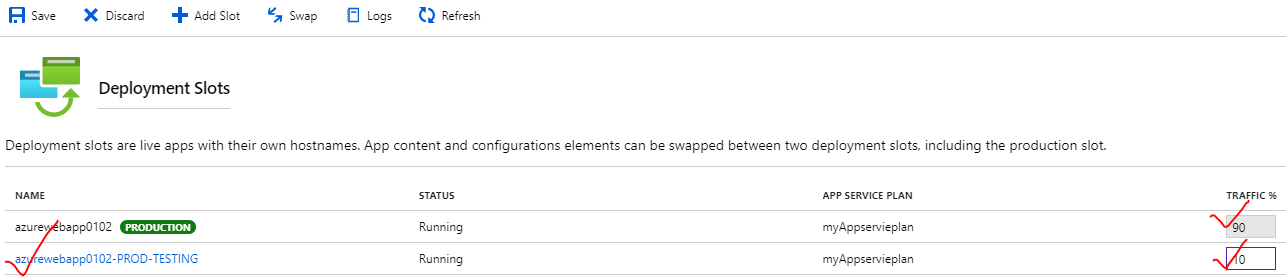


#### AUTOMATED DEPLOYMENT

|  |  |
| --- | --- |
|  | * For automated deployment- We can apply CI/CD to “PROD-TESTING” slot. Once the testing is done – It can swap to the Production slot. * Note – Setting up CI /CD directly on PROD slot is not recommended |

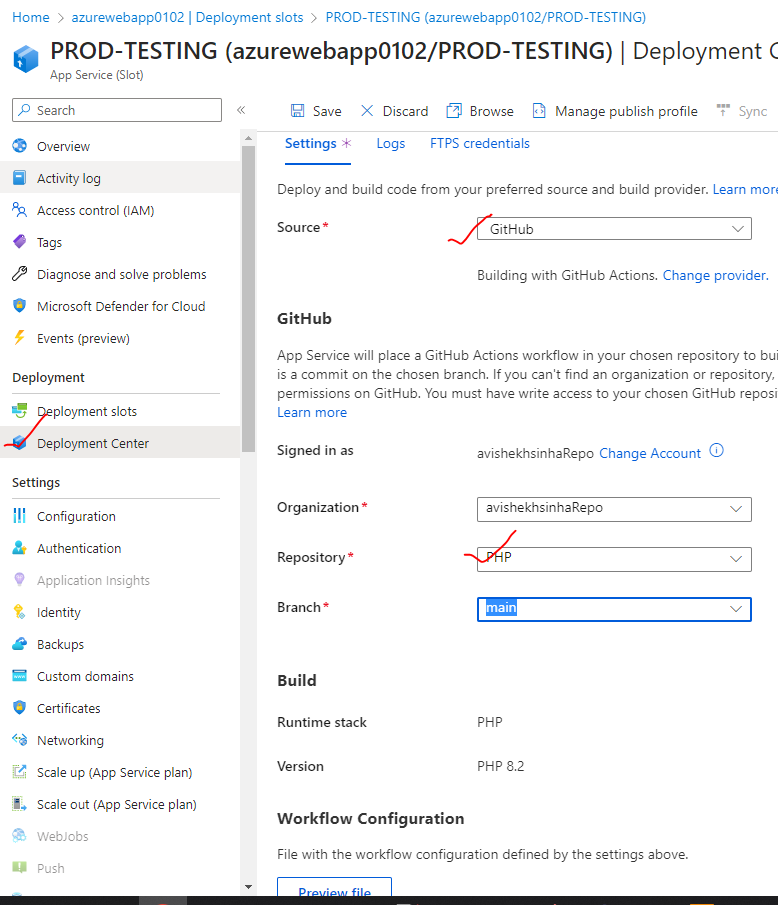
STEP 1: CREATE A PROD\_TESTING DEPLOYMENT SLOT

1. Note –Deployment slot can also distribute the traffic between slots. This can help in testing the application with actual traffic.

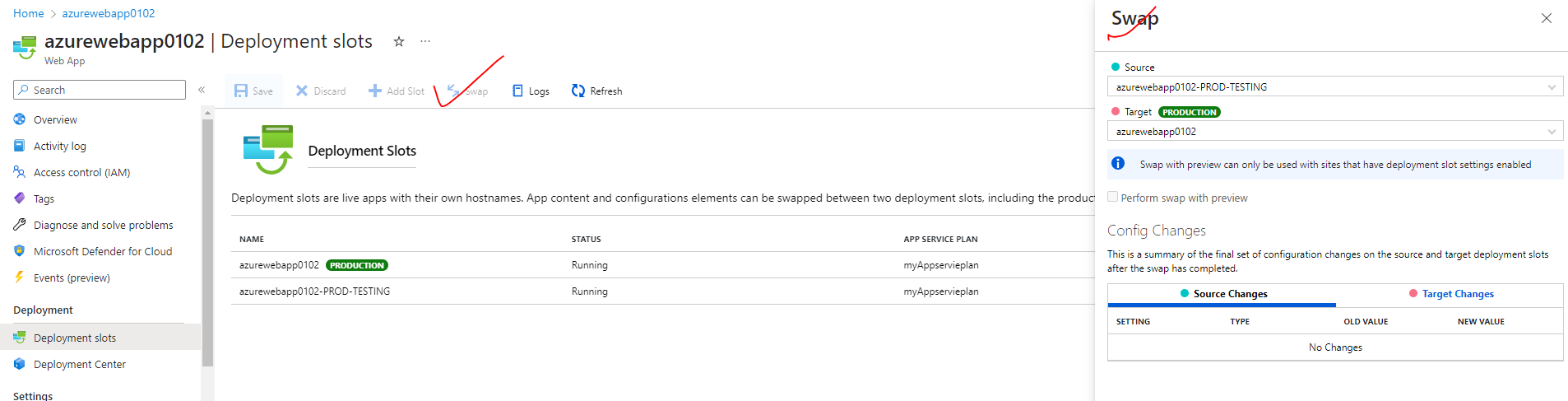


STEP 2: APPLYING CI/CD TO PROD-TESTING SLOT

* Click and navigate to PROD-TESTING Deployment slot 🡪 Deployment Slot
* Connect to the GitHub/Bitbucket etc. account 🡪 Select the repository 🡪 Save
* REPO LINK - <https://github.com/avishekhsinhaRepo/PHP.git>



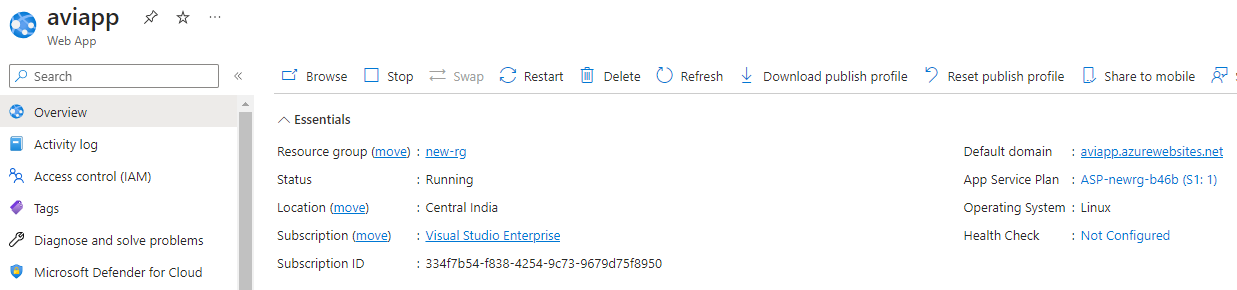
STEP 3: SWAP THE DEPLOYMENT SLOT

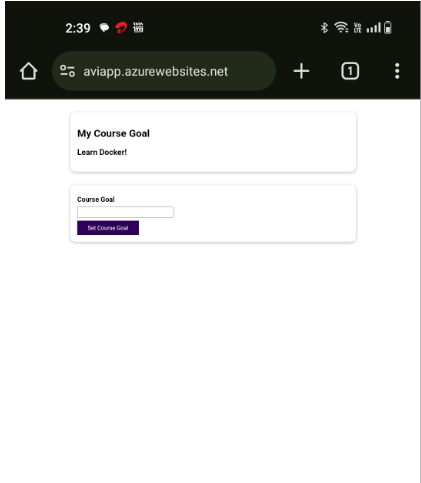


## CUSTOM DOMAIN

To set up the custom domain.

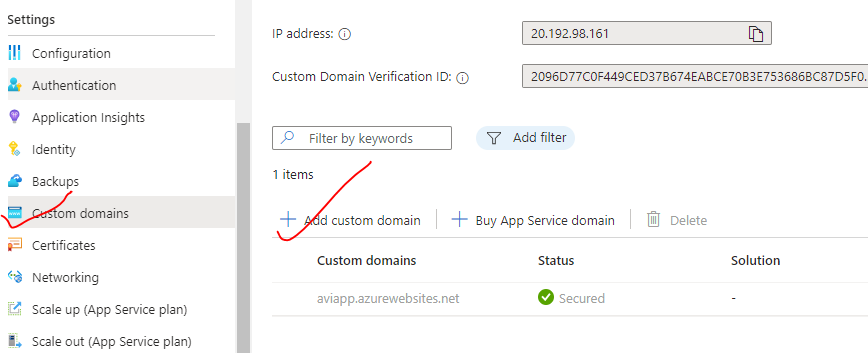
**Step 1: SET UP THE WEB APP**





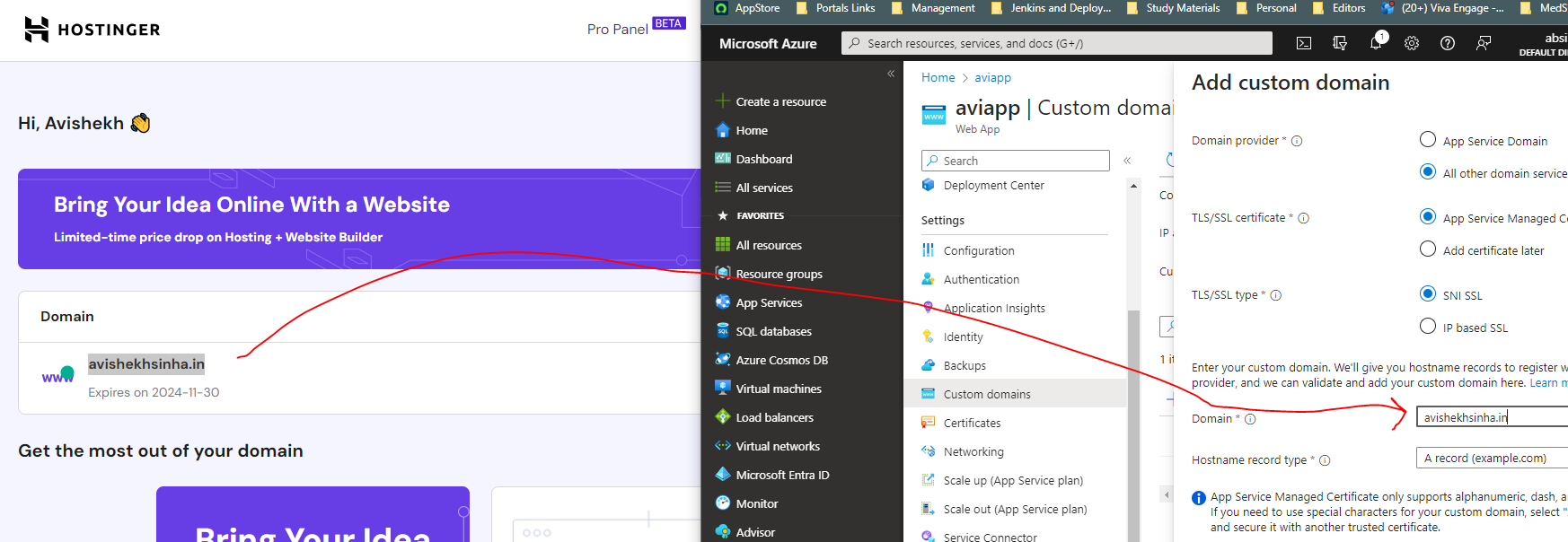
**STEP 2 : ADD THE CUSTOM DOMAIN**

Navigate to the console of any domain provider



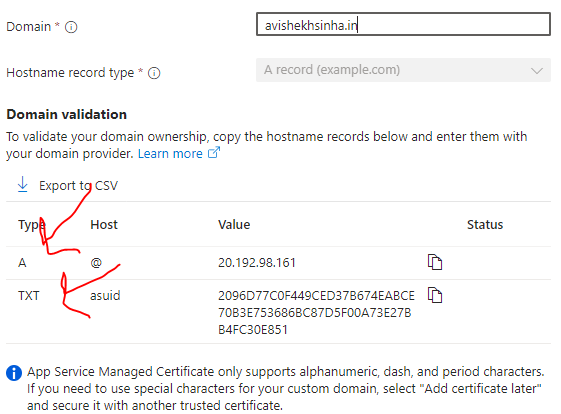
**STEP 3 : ADD THE CUSTOM DOMAIN**

Configure the domain puchased from the Domain Provider

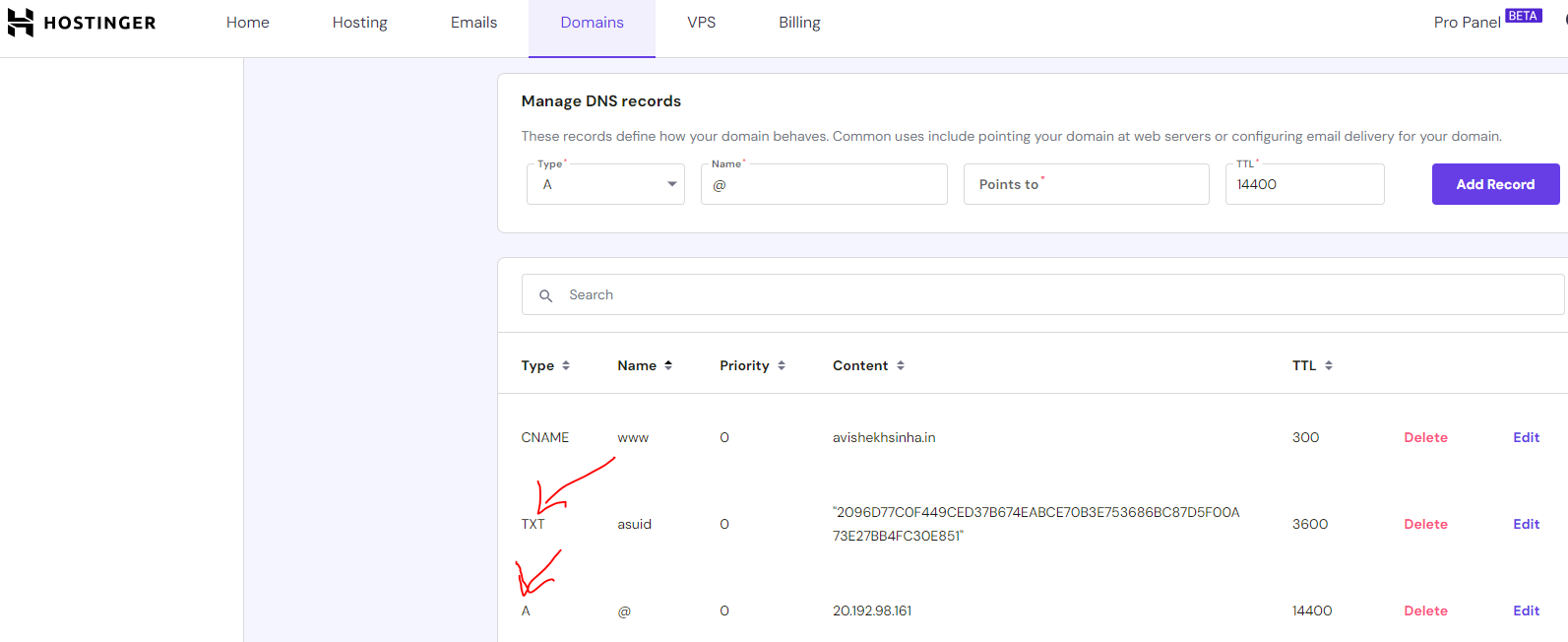


**STEP 4 : DOMAIN VALIDATION**

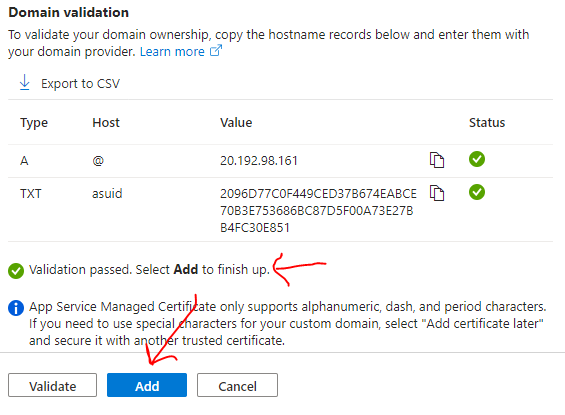
Copy the “A” and “TXT” record and add the configure in the DNS records of the Domain Provider



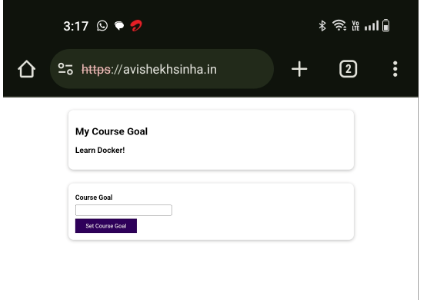
DNS RECORDS



**STEP 4 : VALIDATE THE DOMAIN AND ADD THE CUSTOM DOMAIN**



* ***Note : We might get binding issue – while mapping the custom domain – Restart the Web app might resolve the binding issues.***



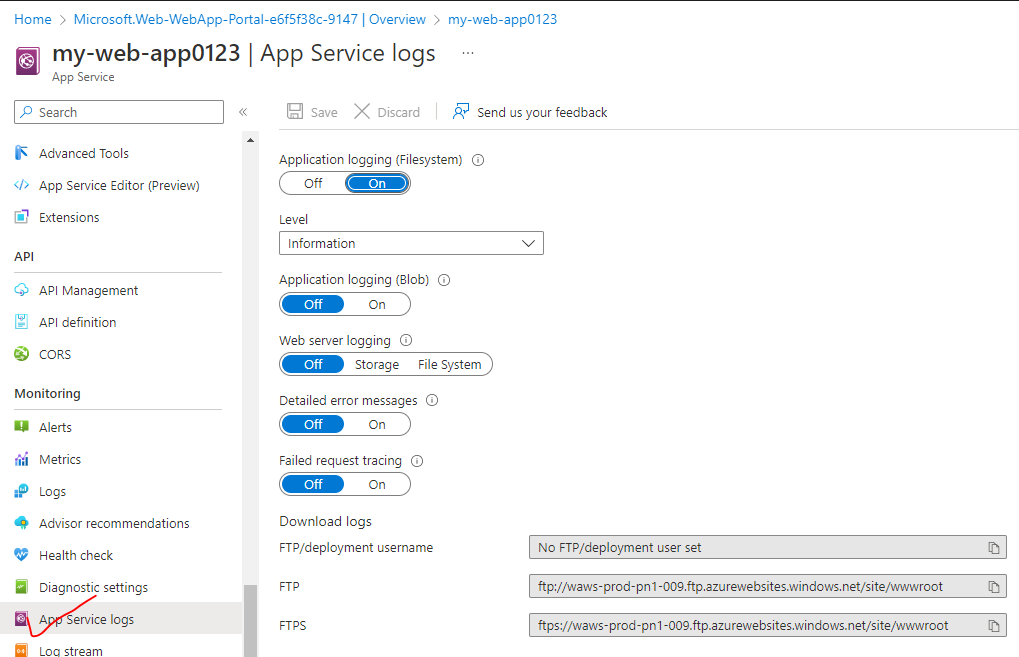
## AZURE WEB APP LOGGING

There are different types of logging feature available for Azure Web App

|  |  |
| --- | --- |
| APPLICATION LOGGING | This captures log messages that are generated from application code |
| WEBSERVER LOGGING | This records raw HTTP request data |
| DETAILED ERROR MESSGES | This stores the copies of .htm error pages that would have been sent to the client browser |
| DEPLOYMENT LOGGING | These are the logs when we publish content to an application |

### SETTING UP LOGGING FOR WEB APP

* Go to Webapp 🡪 App Service logs



#### APPLICATION-LEVEL LOGGING

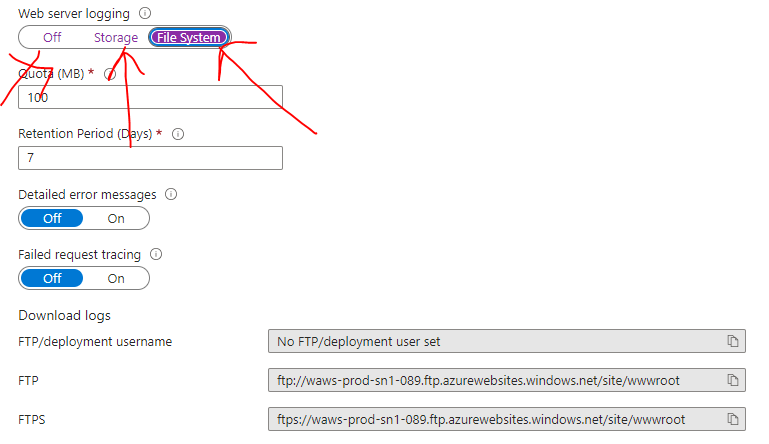
If the application-level logging is enabled for a web-app

* By default- It will store on the underlying compute infrastructure (the virtual machines filesystem) which we can FTP and download the logs.
* If the application logging turned on at the Blob level, then we can then store the logs offline onto an Azure Storage account. We can configure the retention period of the logs as well.



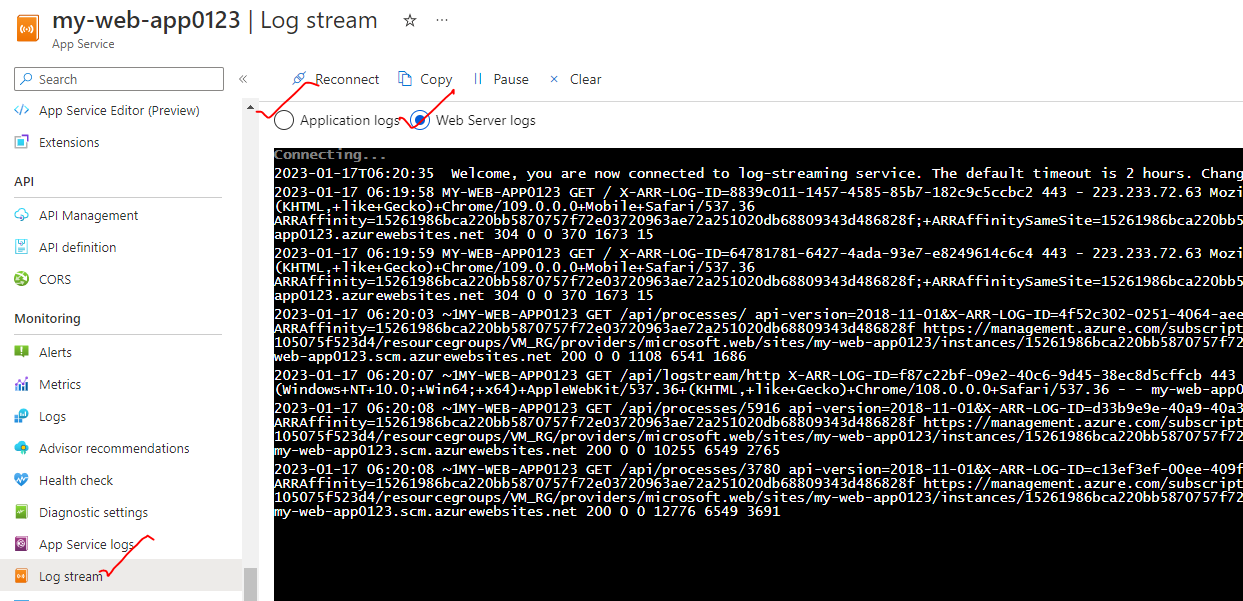
#### WEB SERVER LOGGING

* The webserver logging can be done in file system (with a retention period)or in storage account container.
* The logs can be downloaded from the below FTP location



### LIVE LOGS

* The live logs can be captured from the ***Log Stream***

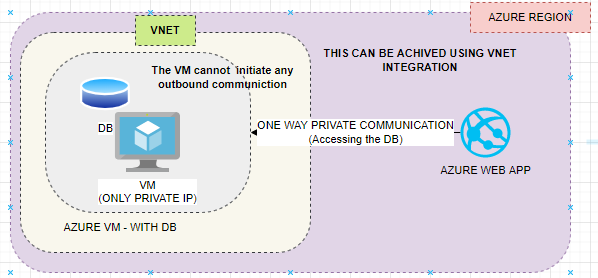


## AZURE WEB APP - VIRTUAL NETWORK INTEGRATION FEATURE

* **VNet integration allows us to securely connect our Azure Web App to a virtual network (VNet) in Azure.**
* By integrating the web app with a VNet, we can access resources within the VNet, such as databases, virtual machines, or other services.
* **VNet integration provides a secure and private connection between the web app and the resources in the VNet, without exposing it to the public internet.**
* It allows your web app to communicate with resources in the VNet using their private IP addresses.
* VNet integration requires a subnet within the VNet dedicated to hosting the integration.
* Once the integration is set up, your web app can securely access resources in the VNet using their private IP addresses and vice versa.
* VNet integration is particularly useful when you want to connect your web app to on-premises resources or other Azure services within a secure network environment.

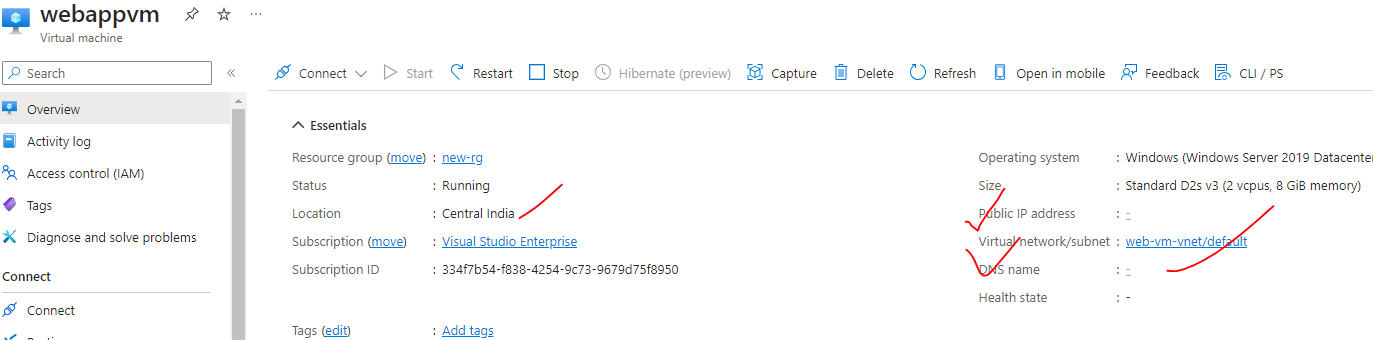
USE CASE

* Let's say we have an Azure web app in place that needs to interact with DB. The DB has been set up in Azure VM
* Then VM is a part of a VNET. Since DB is set up in the VM- hence only need private communication between the web app and the VM. So, the VM will have only has a private IP address (no public IP).
* Note: The Azure web app is a public service which has public IP address (possibly it might have DNS name)
* **The private communication between Web App and VM can be done using VNET integration**.

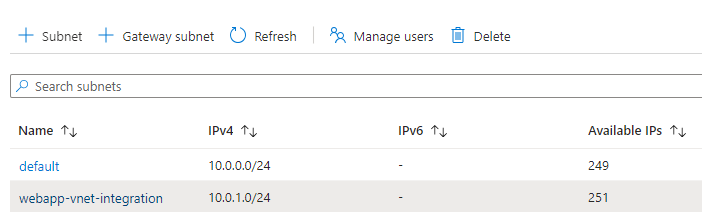


### SETTING UP VIRTUAL NETWORK INTERGRATION

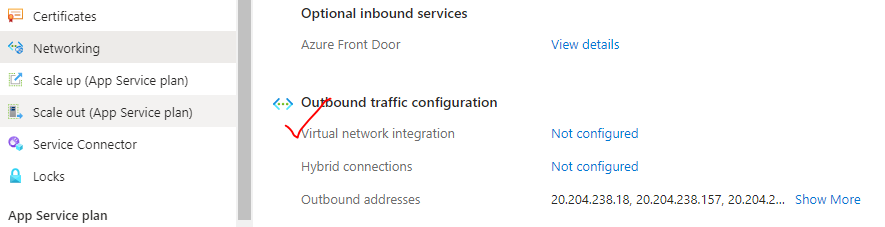
* Step 1 : Set Up the VM in a region (e.g Central India) without any public IP. Make sure the web app connecting to it should also lie in the same region.



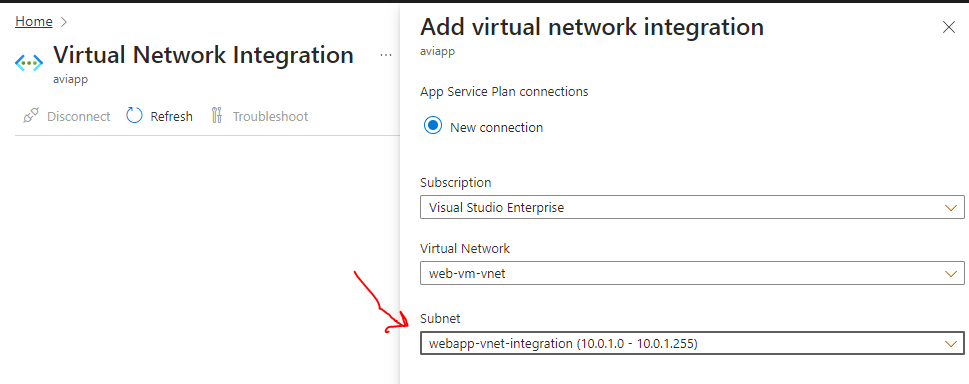
Step 2: We need to have an empty subnet in place for VNET integration. Hence navigate to VNET and add an empty Subnet

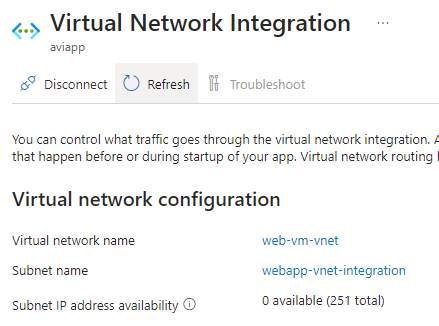


* Navigate to the webapp 🡪 Networking 🡪 Virtual Network Integration
* Select the VNET 🡪 The empty Subnet to establish the connection



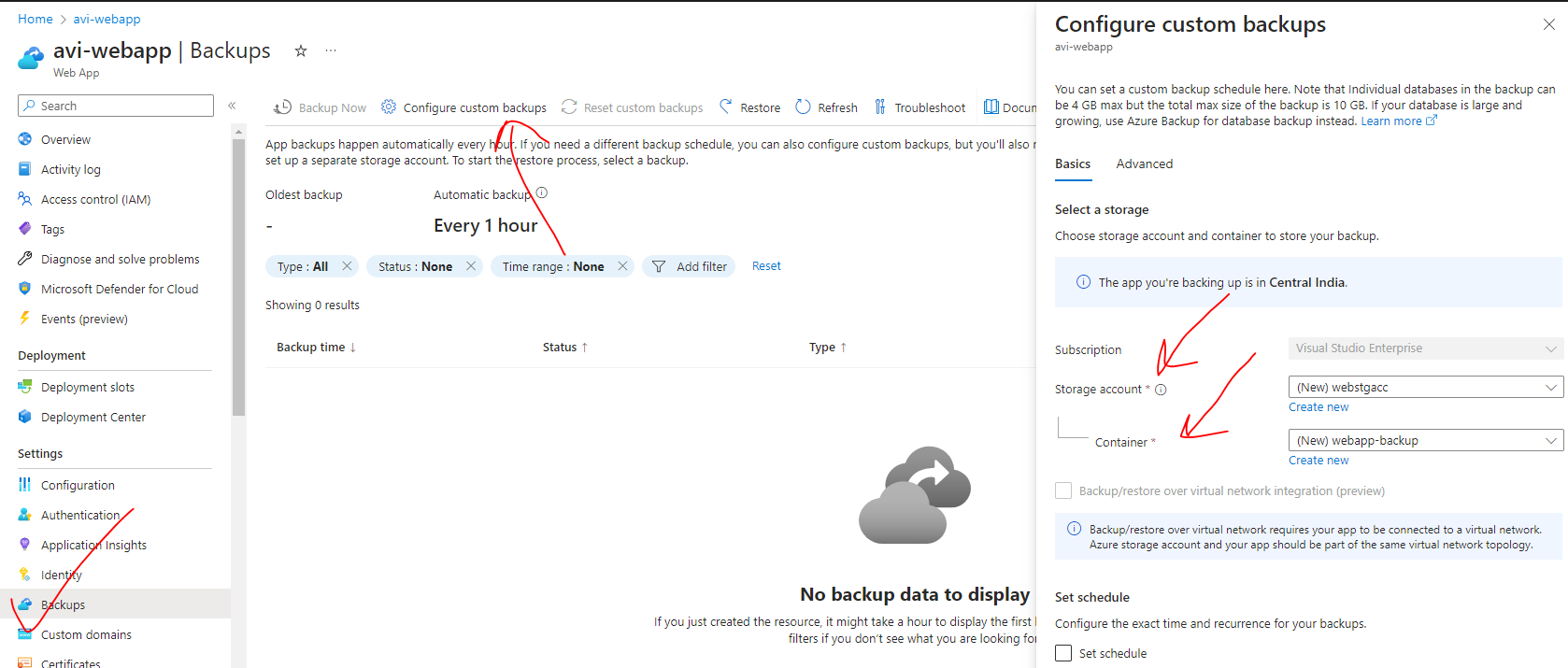
* After the connection is established the Webapp can able to connect to the VM using its private IP address





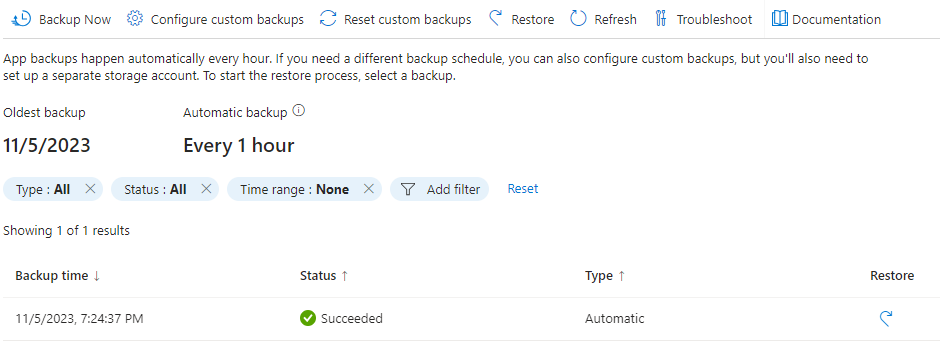
## AZURE APP SERVICE BACKUP

* The backup feature that is available with Azure Web App can be used to create backups of the web app.
* The backups are stored in an Azure storage account.
* Here the App configuration, the file content and the database connected to the application get backed up.
* To use the Backup and Restore feature, the App Service Plan needs to be in the **Standard, Premium or Isolated tier**.
* Backups of the app + database can be up to a maximum of 10 GB.



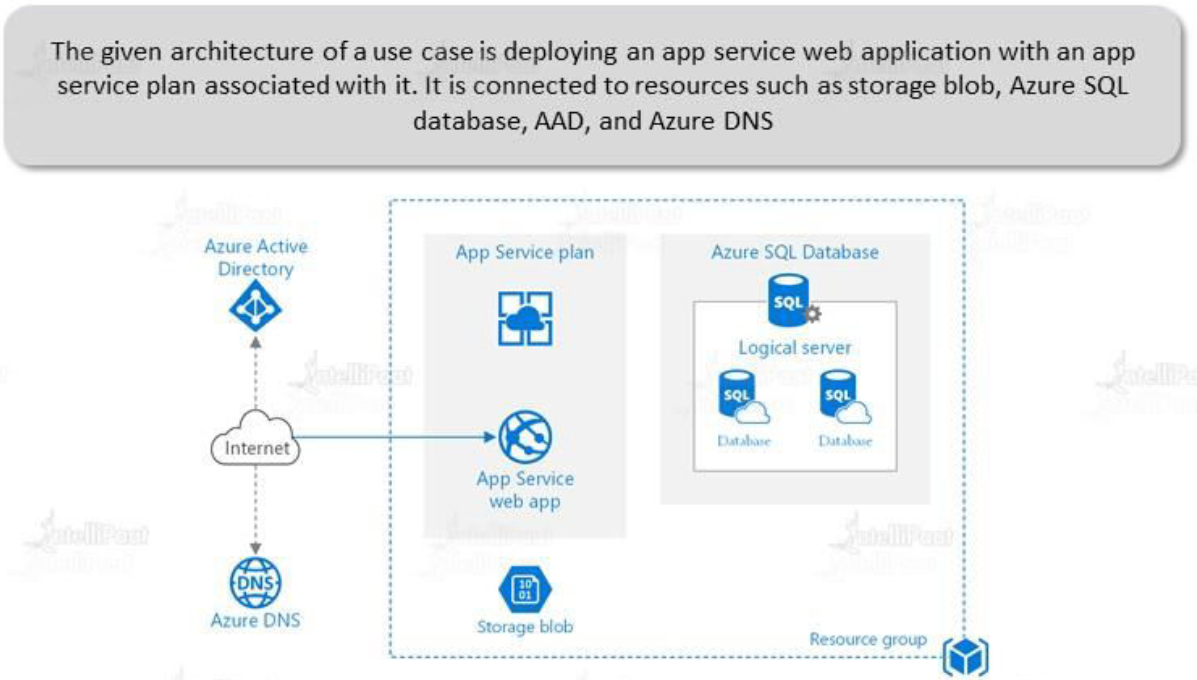
To configure custom backup. Navigate to webapp 🡪 Backup. We need to create following.

* **STORAGE ACCOUNT**
* **CONTAINER IN THE STORAGE ACCOUNT**
* **SET SCHEDULE FOR THE BACKUP**



* Restoring the webapp from the backup will create a deployment slot and the restored web app will be deployed in the newly created deployment slot.

## AZURE APP SERVICE USE CASE



## TRAINING SCREEN SHOTS

