**DEVOPS**

1. CREATE AZURE WEBAPP

Follow the below steps to create a Web App in Azure:

**Step 1:** Log in to [Azure Portal](https://portal.azure.com/).

**Step 2:** Access App Service from azure global search and Click **Create.**

**Step 3:**Select **Basic >>**under **Project Details**select your active azure**subscription**and **resource group.**

**Step 4:**Under **Instance Details:**

* **App Name:** Enter unique **https://<app\_name>. azurewebsites.net** app name.
* **publish type:** Select your type as code.
* Select your **runtime stack**: for example, python 3.10.
* **Region:** Select your app deployment region (**Note:** Select your region where your Azure Service Plan deployed.).
* Create or Select the **App Service Plan.**

Create a new one:

1. Under **Pricing plan**, select Explore pricing plans to select a different App Service plan.

The App Service plan controls how many resources (CPU/memory) are available to your app and the cost of those resources. For this example, under Production, select the **Premium V3 P1V3** plan. The plan will incur a charge against your Azure account but is recommended for better performance.

1. When finished, select **Select** to apply your changes.

**Step 5:** Click on **Review + create** then verify all your deployment details and Click **Create**

Once the deployment is done, you can open the deployment section from there you can click on browse to view the default home page.

1. AZURE DEVOPS: SERVICE CONNECTION

2.1: AZURE RESOURCE MANAGER

There are two options for authentication: use a managed identity or use a service principal. The advantage of the managed identity option is that you can use it if you don't have permissions to create service principals or if you're using a different Microsoft Entra tenant than your Azure DevOps user.

**2.1.1 Create a managed identity.**

1. Sign in Azure Portal
2. In the search box, enter **Managed Identities**.
3. Select **Create**.
4. In the **Create User Assigned Managed Identity** pane, enter or select values for the following items:
5. **Subscription**: Select the subscription in which to create the user-assigned managed identity.
6. **Resource group**: Select a resource group to create the user-assigned managed identity in or select **Create new** to create a new resource group.
7. **Region**: Select a region to deploy the user-assigned managed identity
8. **Name**: Enter the name for your user-assigned managed identity
9. Copy the **Subscription ID** and **Client ID** values for your managed identity to use later.
10. Go to **Settings** > **Properties**.
11. Copy the **Tenant Id** value to use later.
12. Go to **Settings** > **Federated credentials**.
13. Select **Add credentials**.
14. Select the **Other issuer** scenario.
15. Enter values for **Issuer** and **Subject identifier**. You will replace these values later.
16. Select **Save**.

**2.1.2 Grant permissions to the managed identity**

1. In the Azure portal, go to the Azure resource that you want to grant permissions for (for example, a resource group).
2. Select **Access control (IAM)**.
3. Select **Add role assignment**. Assign the required role to your managed identity (for example, Contributor).
4. Select **Review and assign**.

**2.1.3 Create a service connection for managed identity authentication.**

1. In Azure DevOps, open your project and go to  > **Pipelines** > **Service connections**.
2. Select **New service connection**.
3. Select **Azure Resource Manager**, and then select **Next**.
4. Select **Workload Identity federation (manual)**, and then select **Next**.
5. For **Service connection name**, enter the value that you used for **Subject identifier** when you created your federated credentials.
6. For **Subscription ID** and **Subscription Name**, enter the values for the subscription in your Azure portal account.
7. In the authentication section:
8. For **Service Principal Id**, enter the value of **Client Id** from your managed identity.
9. For **Tenant ID**, enter the value of **Tenant Id** from your managed identity.

* In Azure DevOps, copy the generated values for **Issuer** and **Subject identifier**.
* In the Azure portal, return to your app registration federated credentials.

1. Paste the values for **Issuer** and **Subject identifier** that you copied from your Azure DevOps project into your federated credentials in the Azure portal.
2. In the Azure portal, select **Update** to save the updated credentials.
3. In Azure DevOps, select **Verify and save**.
   1. GITHUB INTEGRATION WITH AZURE PIPELINE

**Step 1: Install Azure DevOps Pipeline in GitHub**

1. Go to marketplace of your repository.
2. Install Azure Pipeline and authorize it.

**Step 2: Enable GitHub Integration in Azure Pipeline**

1. Go to Project Settings in your Azure DevOps project.
2. Under Pipeline, select Service Connections.
3. Connect your GitHub using PAT (Personal Access Token)
4. Choose your GitHub repository and authorize.

*Integration is completed.*

1. DEPLOY TO APP SERVICE USING AZURE PIPELINES

**Create a pipeline for your stack:**

1. Sign into your Azure DevOps organization and navigate to your project.
2. Go to **Pipelines**, and then select **New Pipeline**.
3. When prompted, select the location of your source code: **GitHub**.

You might be redirected to GitHub to sign in. If so, enter your GitHub credentials.

1. When the list of repositories appears, select your repository.
2. You might be redirected to GitHub to install the Azure Pipelines app. If so, select **Approve & install**.
3. When the **Configure** tab appears, select **starter pipeline.**
4. When your new pipeline appears, create the YAML from start. When it is ready, select **Save and run**.
5. Once the build is successful, you will get a URL of your webapp. Check if everything is working well.
6. CONFIGURE AZURE WEBAPP
7. **Webapp configurations:** *application settings*

* Enable WEBSITE\_RUN\_FROM\_PACKAGE and disable SCM\_DO\_BUILD\_DURING\_DEPLOYMENT. This setting ensures that the application runs directly from the package deployed to the App Service, improving performance and reliability.

Note: please make sure your package is less than 1GB. In case it is greater than 1GB go with SCM\_DO\_BUILD\_DURING\_DEPLOYMENT = 1

1. **Deployment logs:**

Use the following steps to access the deployment logs:

1. On the Azure portal for your web app, select **Deployment** > **Deployment Center** on the left menu.
2. On the **Logs** tab, select the **Commit ID** for the most recent commit.
3. On the **Log details** page that appears, select the **Show Logs...** link that appears next to "Running oryx build...".

Build issues such as incorrect dependencies in requirements.txt and errors in pre- or post-build scripts will appear in these logs. Errors also appear if your requirements file isn't exactly named requirements.txt or doesn't appear in the root folder of your project.

1. **Scaling and Instances:**

* Scale out method = automatic *(platform managed scale up and down based on traffic)*

Configured the web application to run on two instances to ensure high availability and load balancing.

* This setup allows the application to manage increased traffic and supplies redundancy in case one instance fails.

1. **File Access and Logging with Kudu:**

* Utilized Kudu for accessing files and logs directly within the Azure WebApp environment.
* Kudu is integrated with Azure WebApp, making it a convenient tool for logging and debugging the application. It enables viewing, downloading and streaming the logs.
* **Log stream** can be used for live log streaming.

1. **App backup:**

* By default, App backups happen automatically every hour. To restore, select a backup.
* We can setup the custom backup, and we will need to set up a separate storage account for that.

1. **Azure Monitor:** *performance monitoring*

* Metrics: Provides metrics for app service, i.e., infrastructure level metrics
* Health Checks:
* Enable health checks on the Azure App Service to check the availability and responsiveness of the web application.
* It increases application's availability by removing unhealthy instances from the load balancer. If the instance stays unhealthy, it will be replaced.
* *Set Load balancing threshold = 10 min*
* App services logs: Enable app service logs.

1. Navigate to your Linux Web App and select the **"App Service Logs"** option under the "Monitoring" section in the left pane.
2. Toggle the "Application Logging" button to **"File System"**, and optionally change the "Quota" and "Retention Period" as desired.
3. Click **"Save"** to enable the feature.
4. **Security:**

* Locks: Add Lock to the app service. Select **DELETE** as the lock type. if you want to do modifications to webapp. Otherwise go with **READ-ONLY**
* Use **Storage accounts** to store the logs for longer period.

1. TROUBLESHOOTING
2. In general, the first step in troubleshooting is to use App Service Diagnostics:

* In the Azure portal for your web app, select **Diagnose and solve problems** from the left menu.
* Select **Availability and Performance**.
* Examine the information in the **Application Logs** and **Container Issues** options, where the most common issues will appear.

1. Next, examine both the [deployment logs](https://learn.microsoft.com/en-us/azure/app-service/configure-language-python#access-deployment-logs) and the [app logs](https://learn.microsoft.com/en-us/azure/app-service/configure-language-python#access-diagnostic-logs) for any error messages. These logs often find specific issues that can prevent app deployment or app startup.

***Reference:***

[1] [Set a Resource Manager workload identity service connection - Azure Pipelines | Microsoft Learn](https://learn.microsoft.com/en-us/azure/devops/pipelines/release/configure-workload-identity?view=azure-devops)

[2] [Quickstart: Deploy a Python (Django or Flask) web app to Azure - Azure App Service | Microsoft Learn](https://learn.microsoft.com/en-us/azure/app-service/quickstart-python?tabs=flask%2Cwindows%2Cazure-cli%2Cazure-cli-deploy%2Cdeploy-instructions-azportal%2Cterminal-bash%2Cdeploy-instructions-zip-azcli)

[3] [Configure Linux Python apps - Azure App Service | Microsoft Learn](https://learn.microsoft.com/en-us/azure/app-service/configure-language-python)

[4] [Run your functions from a package file in Azure | Microsoft Learn](https://learn.microsoft.com/en-us/azure/azure-functions/run-functions-from-deployment-package)

[5] [Monitor Azure App Service - Azure App Service | Microsoft Learn](https://learn.microsoft.com/en-us/azure/app-service/monitor-app-service)

[6] [Azure App Service Logging: How to Monitor Your Web Apps in Real-Time (microsoft.com)](https://techcommunity.microsoft.com/t5/apps-on-azure-blog/azure-app-service-logging-how-to-monitor-your-web-apps-in-real/ba-p/3800390#:~:text=To%20enable%20App%20Service%20Logs%20and%20Log%20Stream,desired%203%20Click%20%22Save%22%20to%20enable%20the%20feature.)

[7] [Diagnostics and solve tool - Azure App Service | Microsoft Learn](https://learn.microsoft.com/en-us/azure/app-service/overview-diagnostics)