Fibonacci Sequence API

**Step 1: Python Script using Flask framework**

GitHub link for accessing the files:

<https://github.com/lisaelsa/Fibonacci/tree/main>

Prerequisites to run the Fibonacci.py file in your machine:

* Install Python
* Install Flask

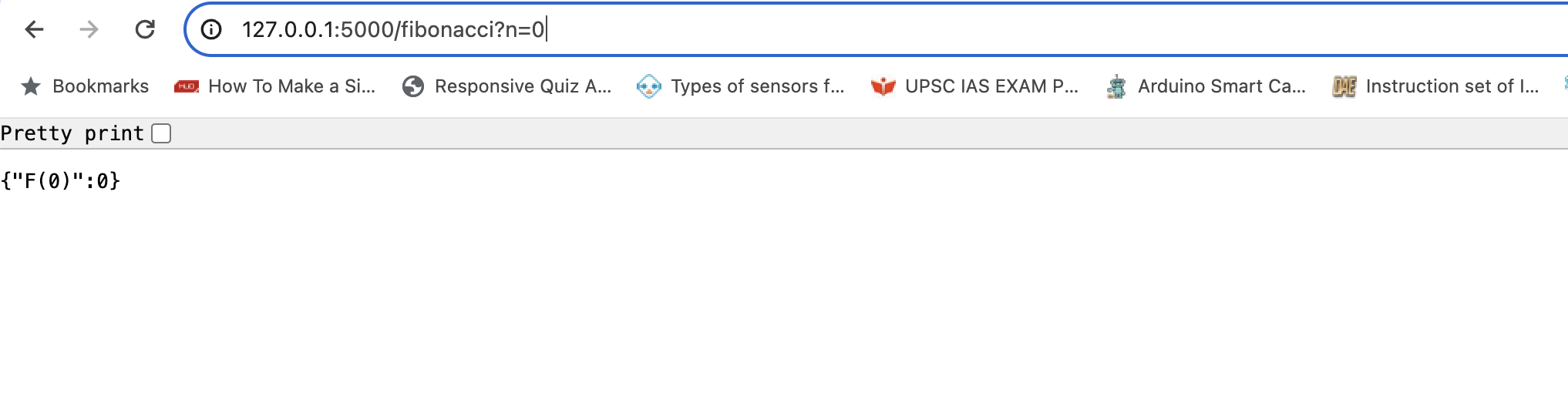
To run the script: **python Fibonacci.py**

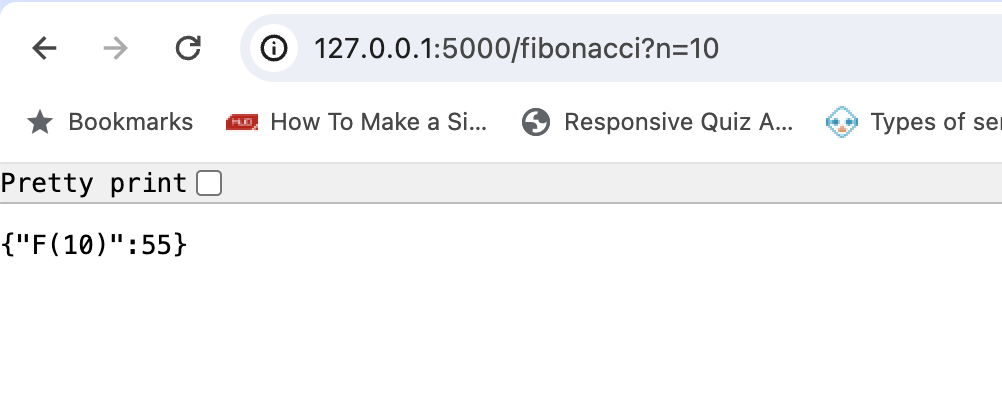
API to access: <http://127.0.0.1:5000/fibonacci?n=3> [Make sure 5000 port is free]

OR

Docker container can be run to test which includes all the necessary dependencies and configurations. Please refer Step 2. Dockerfile used for building the image can be referred in github repository.

*Output Snapshots:*





**Step 2: Containerization**

Container is a self-contained sealed unit of software which contains everything required to run the code.

To build the docker image: [Already done and the final image can be used for testing]

* Clone the gihub repository: git clone <https://github.com/lisaelsa/Fibonacci.git>
* Run: docker build -t <tagname> .

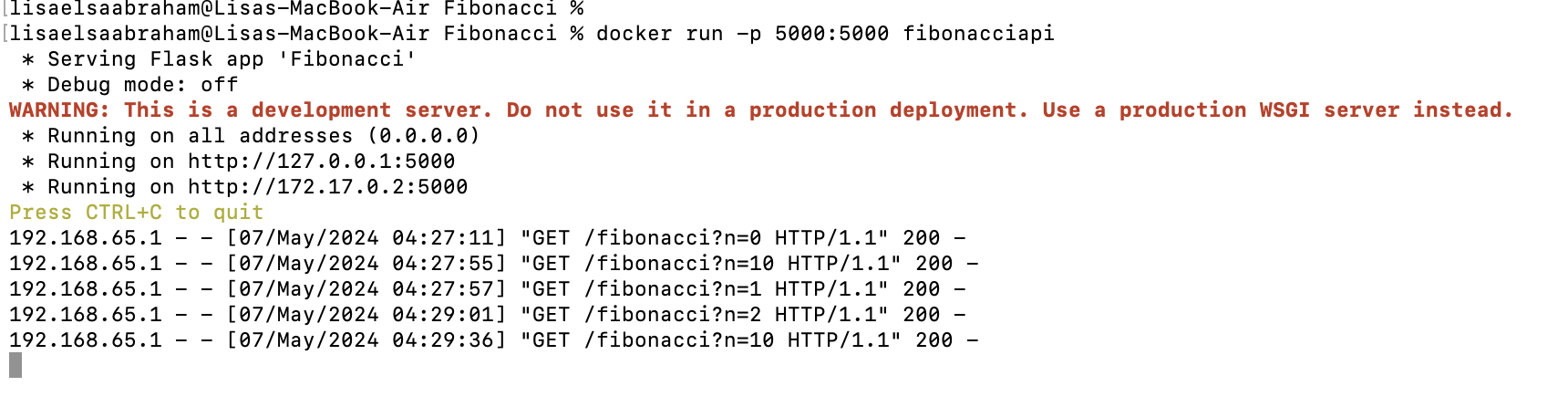
***Note: Make sure docker is installed in the machine***

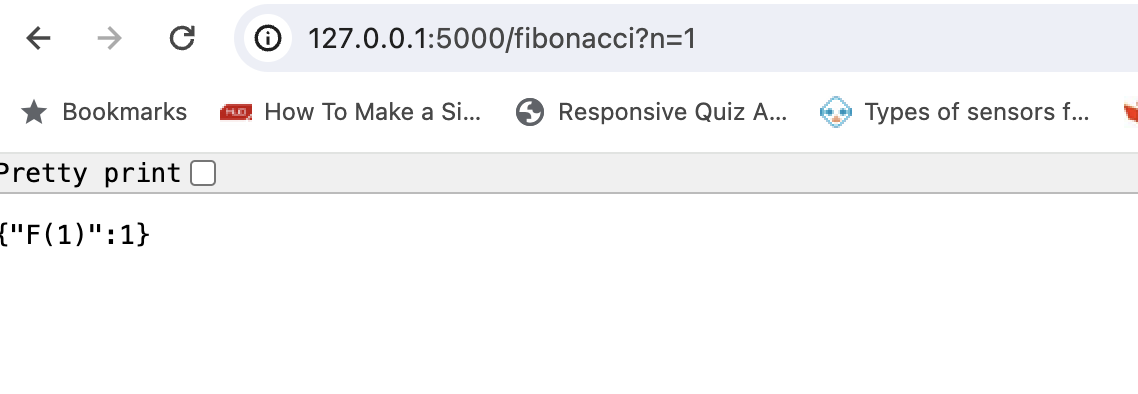
**Docker Image Link** **for testing**: <https://hub.docker.com/repository/docker/lisaelsa/fibonacciapi/general>

To test the docker container:

* Pull image to your machine: docker pull lisaelsa/fibonacciapi:latest
* Run docker image: docker run -d -p 5000:5000 lisaelsa/fibonacciapi

**Output Snapshot**



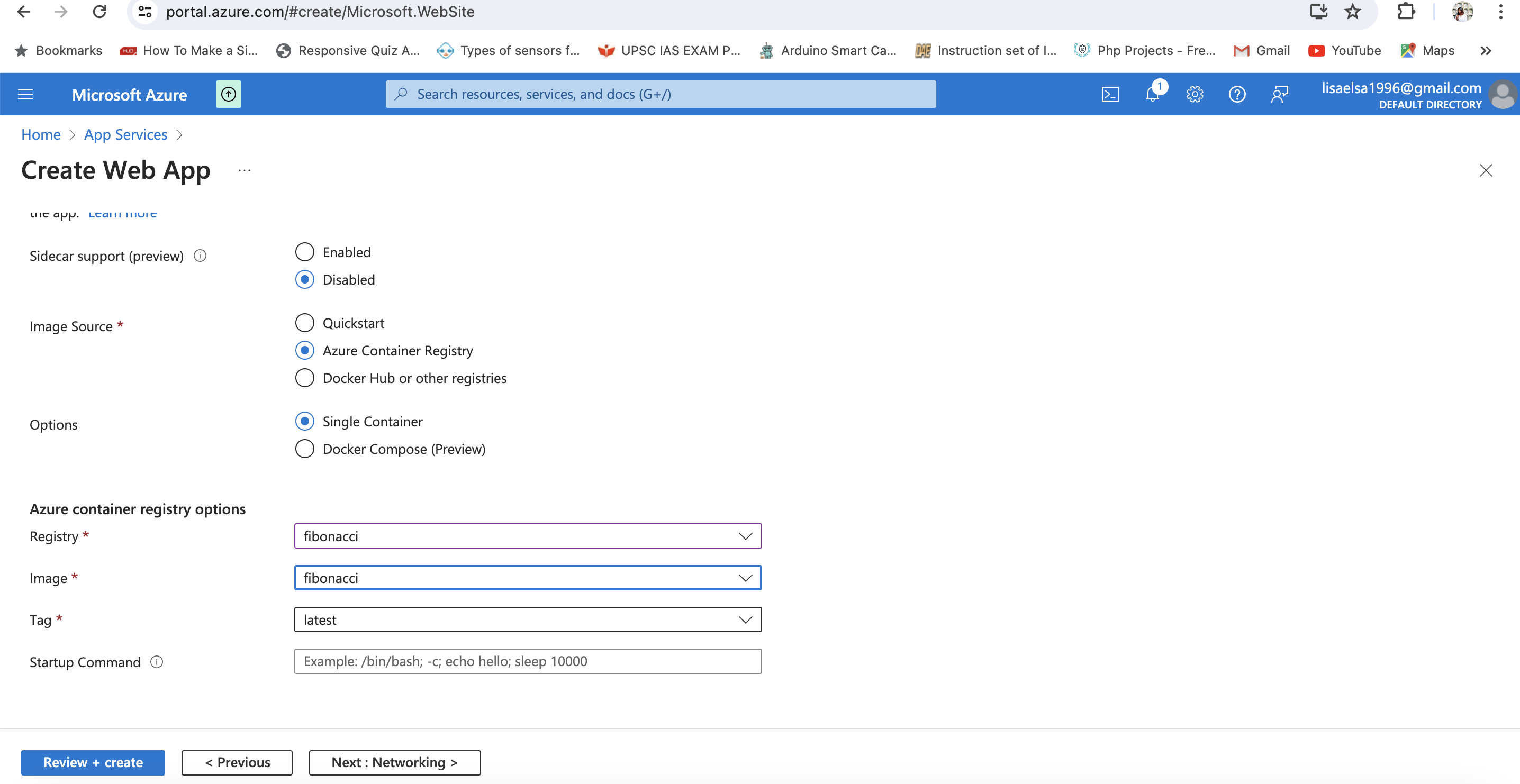


Step 3: Continuous Integration/Continuous Deployment (CI/CD) processes

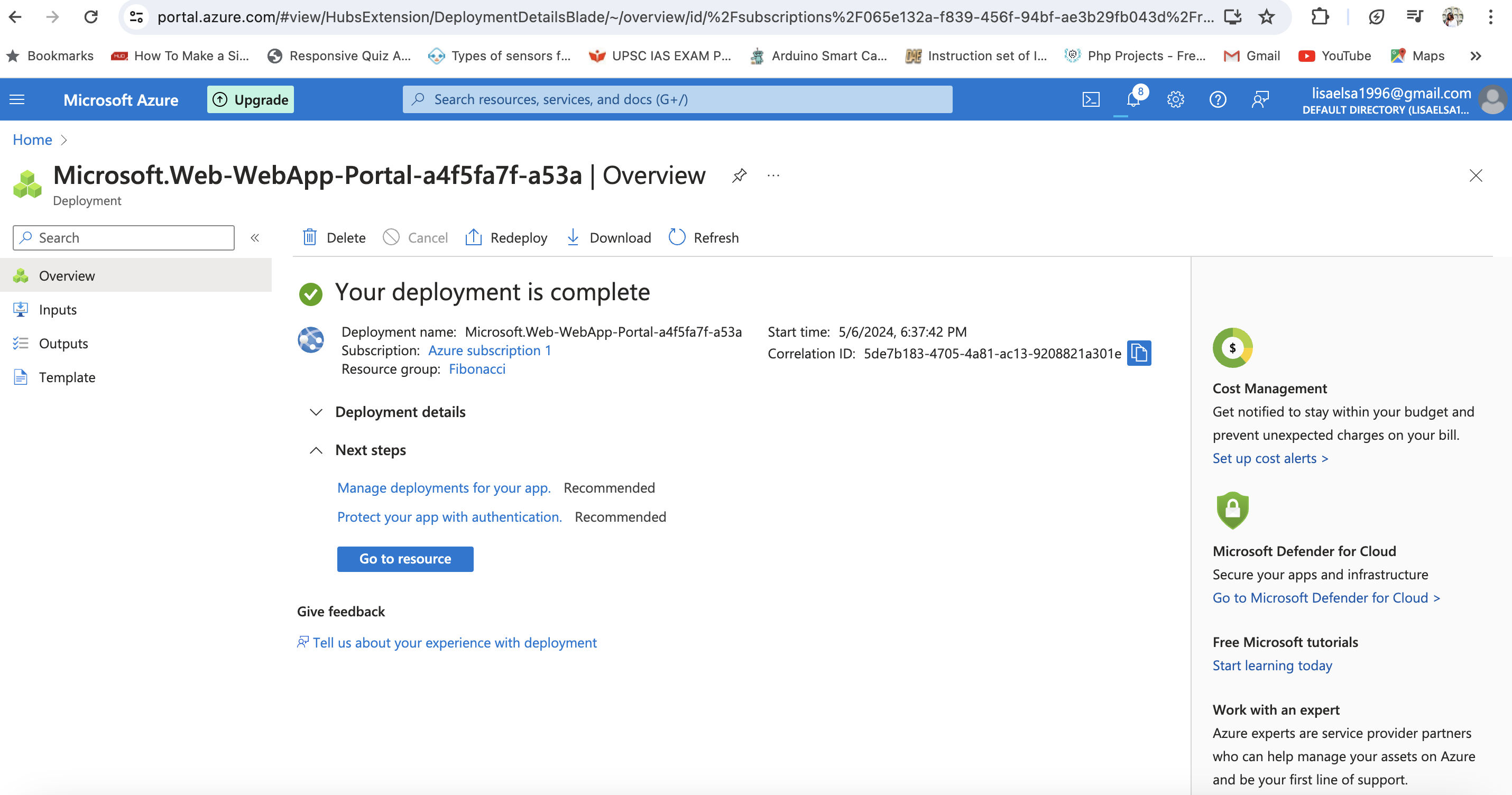
Azure Web App service used for deploying the project as it’s a great Azure service to spin up the app/container by pulling it either from Azure Container Registries or from other registries(Docker..)

Steps followed for creation of web app

* Go to Azure Portal
* Uploaded the docker image to the Azure Container registry
* Create Web app by configuring the details (Subscription, Resource group, app name, Region …)
* Select publish option as Container
* Add Azure Container registry details in the container section



* Enable Application Insights for monitoring in Monitoring section
* Review and create



Deployment slots can be configured in Azure Web Apps to deploy different versions of the application to separate instances of our web app. This feature enables us to perform staged deployments, testing new versions application in a production-like environment before swapping them into production.

Steps:

* **Create deployment slots:** You can create multiple deployment slots for a single Azure Web App. Each deployment slot has its own URL and separate configuration settings. We can name deployment slots as staging, testing, production or any other suitable name as per our requirement
* **Deploy to deployment slots:** After creating deployment slots, we can deploy our application to each slot independently. We can deploy directly from our CI/CD pipeline or manually through Azure Portal, CLI, or Azure DevOps. Deployed applications in slots run on separate instances from the production instance of our web app.
* **Testing and Validation:**Deployment slots provide a production-like environment where we can validate application behaviour, monitor performance, and troubleshoot issues before promoting changes to production.
* **Swap with Production:**Once tested and validated the deployment slot, we can swap the slot with the production slot to make the changes live.The swap operation swaps the IP address and hostnames of the slots, effectively directing traffic from the production slot to the previously staged slot. If any issues occur after the swap, we can quickly swap back to the previous version or perform additional troubleshooting in the staging slot.

Also we can enable the deployment creating build and release pipelines in Azure.

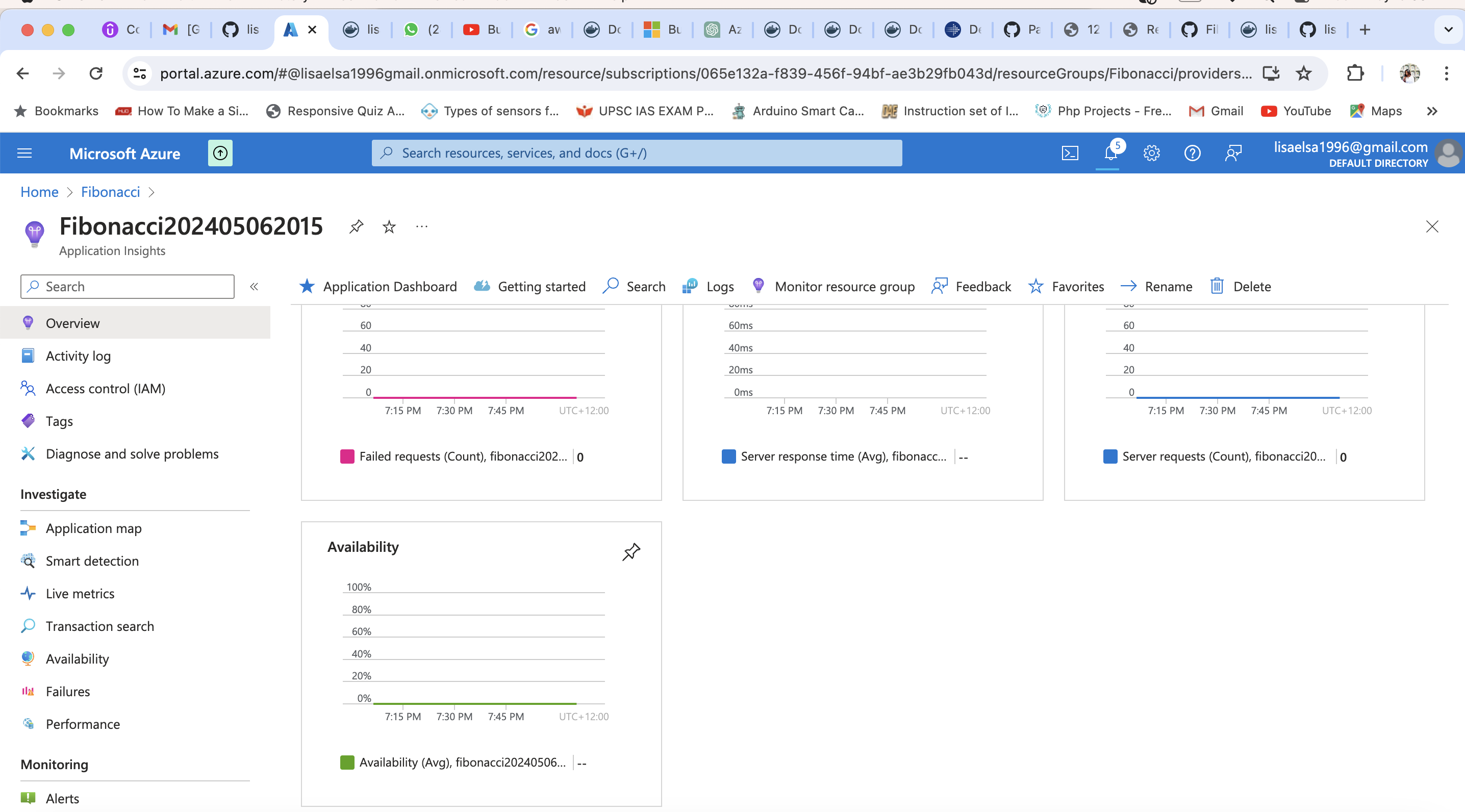
**Steps for configuring Build and Release Pipeline for enabling deployment through Azure Web App:**

* Sign into your Azure DevOps portal and select your organization and project
* Set up the source control and create a build pipeline with proper tasks to create the docker image
* create a new release pipeline by mapping the artifacts from build pipeline to deploy the app to Azure web app. We can use Azure app Service deploy task to deploy to Azure Web app
* Configure triggers to automatically run the build and release pipelines whenever changes are pushed to repository.

**Step 4: Monitoring**

Azure Monitor application insights is an Application Performance Management (APM) service for developers and DevOps professionals. Enable it while creating web app to automatically monitor our application. It will detect performance anomalies, and includes powerful analytics tools to help us diagnose issues and to understand what users actually do with our app.

While creating web app for Fibonacci API, I enabled the Application Insights under monitoring section. PFB the Snapshot of Application Insights generated for Fibonacci App deployed using web app.



Also azure provides many other monitor services which can be configured while creating web app (Diagnostic Logging, Log Analytics, Security logging etc.)

**Step 4: Scaling**

We can enable autoscaling to automatically adjust the number of instances (horizontal scaling) based on demand. We can define scaling rules based on metrics like CPU utilization, memory usage, or request count.

This feature is enabled for Standard or Premium tier.

If horizontal scaling is not enough to handle the increased load, then we need to upgrade the pricing tier of our Azure Web App to a higher tier with more resources (vertical scaling)