

EASWARI ENGINEERING COLLEGE
(Autonomous)



Internship Report

Submitted in Partial Fulfilment of for the award of degree of

BACHELOR OF ENGINEERING

In

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by

HIRESH R -310620106044

Internship carried out at

LARSON & TOUBRO Ltd, CONSTRUCTION

Department of Electronics and Communication Engineering

EASWARI Engineering College, Chennai

(2022 - 2023)

**TABLE OF CONTENT**

S.NO	DESCRIPTION	PAGE
	ABSTRACT	<u>2</u>
1	OBJECTIVE OF THE PROJECT	<u>3</u>
2	NECESSITY	<u>3</u>
3	PROJECT OVERVIEW	<u>3</u>
4	PROJECT WORK FLOW DIAGRAM	<u>4</u>
5	RESOURCE & USE	<u>5</u>
5.1	AZURE CONATINER REGISTRY (ACR)	<u>5</u>
5.2	AZURE APP SERVICE	<u>5</u>
5.3	AZURE DEVOPS BUILD PIPELINE	<u>6</u>
5.4	AZURE DEVOPS RELEASE PIPELINE	<u>6</u>
6	SYSTEM ARCHITECTURE AND RESOURCES	<u>7</u>
7	STEPS	<u>10</u>
7.1	CREATE CONTAINER REGISTRY & COSMOS DB	<u>10</u>
7.2	MOVE CODE TO AZURE REPO	<u>11</u>
7.3	BUILDING AND PUSHING THE IMAGE TO AZURE APP SERVICE	<u>13</u>
7.4	CREATING AND DEPLOYING WEB APP IN AZURE APP SERVICE	<u>16</u>
7.5	CONFIGURING THE AZURE APP SERVICE	<u>17</u>
7.6	ADD AZURE WEB APP CONFIGURATION INTO AZURE DEVOPS	<u>18</u>
7.7	DEPLOYMENT OF THE WEB PAGE	<u>19</u>
7.8	ADDING NETWORK RESTRICTION	<u>20</u>
	SUMMARY	<u>21</u>



ABSTRACT

In today's software development landscape, Continuous Integration (CI) and Continuous Delivery (CD) have become essential practices for maintaining the quality and efficiency of software projects. Docker containers offer a standardized and efficient way to package applications and their dependencies, while Azure App Service provides a scalable and reliable platform for hosting web applications. This internship project aims to enhance the CI/CD pipeline in Azure DevOps for deploying Docker images to Azure App Service, thereby automating the deployment process and ensuring rapid, reliable, and consistent application delivery.



1. OBJECTIVE OF THE PROJECT

In today's modern software development landscape, our goal is to establish an agile and efficient CI/CD pipeline. Leveraging Azure DevOps and Azure Container Registry, we aim to seamlessly deploy Docker images to an Azure App Service. This approach not only enhances our development workflow but also ensures scalability and security. With automation at the core, we're poised to accelerate innovation and maintain a competitive edge in the ever-evolving tech industry.

2. NECESSITY

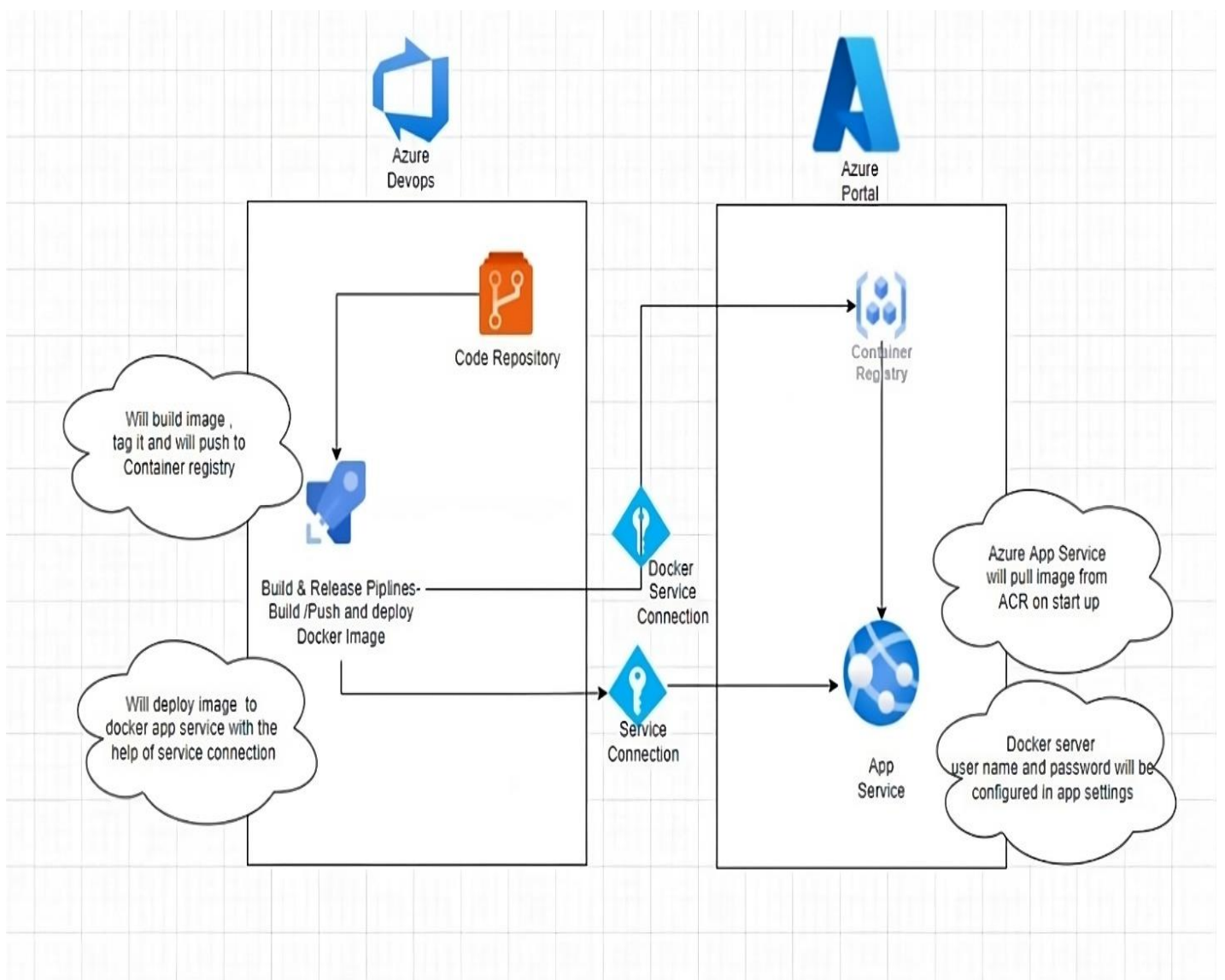
The necessity for this project is driven by the imperative need for efficiency and agility in today's software development ecosystem. Modern software development demands streamlined CI/CD pipelines to accelerate software delivery, respond to market dynamics, and maintain a competitive edge. By integrating Azure DevOps and Azure Container Registry, we meet this demand by automating deployment, ensuring scalability, and fortifying security measures. This approach aligns us with contemporary industry standards and fosters innovation in our software development processes

3. PROJECT OVERVIEW

Our project's core objective is to modernize our software development process with a strong focus on efficiency and agility. We'll achieve this by implementing a streamlined CI/CD pipeline using Azure DevOps and Azure Container Registry. This initiative will automate Docker image deployments to an Azure App Service, leading to faster software delivery, enhanced collaboration, and improved reliability. By embracing these modern technologies, we're poised to respond swiftly to market changes and maintain a secure and competitive edge.



4. PROJECT WORK FLOW DIAGRAM



5. RESOURCES & USE

5.1 AZURE CONTAINER REGISTRY (ACR)

- ACR is a fundamental component for storing Docker images securely within the Azure ecosystem.
- It serves as a centralized repository where Docker images are stored and versioned, ensuring easy access and management.
- ACR provides robust security features, such as access control and image scanning, to safeguard the container images.



- It allows for image replication across Azure regions, enhancing redundancy and global accessibility.
- ACR ensures the reliability and availability of container images for deployment, making it a critical part of a containerized application workflow.

5.2 AZURE APP SERVICE

- Azure App Service is a fully managed platform-as-a-service (PaaS) offering that simplifies the deployment and management of web applications.
- It supports hosting containerized applications, making it an ideal target for deploying Docker images.
- Azure App Service offers features like automatic scaling, load balancing, and integration with Azure services, reducing operational overhead.
- It supports various programming languages and frameworks, allowing developers to choose the most suitable environment for their applications.
- With built-in monitoring and diagnostics tools, Azure App Service enhances the management and monitoring of containerized application

5.3 AZURE DEVOPS BUILD PIPELINE

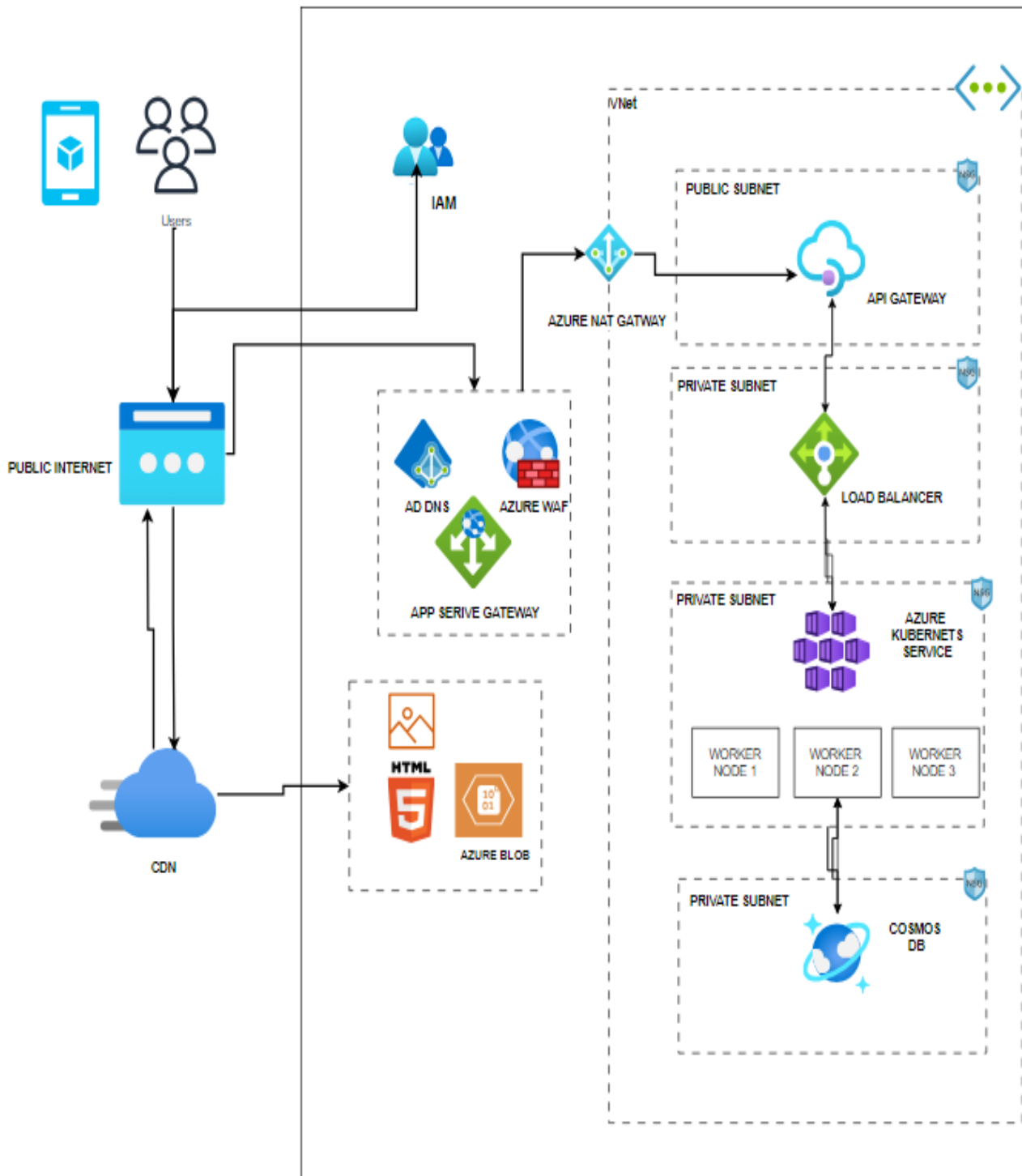
- The build pipeline automates the process of creating Docker images from source code and Docker files.
- It integrates seamlessly with source code repositories like GitHub or Azure Repos, enabling continuous integration.
- By automating image building, it ensures consistency and repeatability in the creation of container images.
- The build pipeline can also execute tests and validation steps before packaging the application into a Docker image.
- This component plays a pivotal role in preparing container images for deployment, maintaining code quality, and streamlining the development workflow.



5.4 AZURE DEVOPS RELEASE PIPELINE

- The release pipeline orchestrates the deployment of Docker images from Azure Container Registry to Azure App Service.
- It automates the entire deployment process, from image retrieval to application launch, ensuring consistency and reliability.
- The release pipeline supports continuous integration and delivery (CI/CD) workflows, enabling rapid and frequent deployments.
- It can trigger deployments automatically when new images become available in the container registry.
- By automating the deployment process, it reduces the risk of human errors and accelerates the delivery of new application features and updates.

6. SYSTEM ARCHITECTURE AND RESOURCE USED



AKS (Azure Kubernetes Service)

- Orchestrates and manages container deployments.
- Provides scalability and container orchestration capabilities.

Cosmos DB



- A globally distributed database for storing data generated by containers.
- Ensures data availability and scalability.

VNet (Virtual Network)

- Offers network isolation for secure communication between containers and resources.
- Controls and secures network traffic.

API Gateway

- Exposes containerized services securely to external clients.
- Provides features like authentication, rate limiting, and API management.

Load Balancer

- Distributes traffic evenly among container instances.
- Ensures high availability and reliability of container deployments.

Azure Blob Storage

- Stores container images, configuration files, and data.
- Essential for managing data within containerized applications.

Azure WAF (Web Application Firewall)

- Enhances security for containerized web applications.
- Protects against security threats and attack

NAT Gateway (Network Address Translation)

- Allows secure outbound internet access for containers.
- Hides internal IP addresses.

Active Directory DNS

- Provides authentication and authorization for containerized applications.



- Enhances security by controlling access.

IAM (Identity and Access Management)

- Controls permissions and access to container resources.
- Ensures secure authentication and authorization.

CDN (Content Delivery Network)

- Accelerates content delivery for containerized applications.
- Improves performance and user experience by caching and serving content efficiently

7. STEPS:

7.1 CREATE CONTAINER REGISTRY & COSMOS DB

- Open Azure Portal
- Search for “Container Registry” in Azure home
- Now configure The Container Registry

The screenshot shows the 'Create container registry' page in the Azure Portal. The page has a blue header with the Microsoft Azure logo and a search bar. Below the header, there's a breadcrumb trail: Home > Container registries > Create container registry. The page is divided into tabs: Basics, Networking, Encryption, Tags, and Review + create. The 'Basics' tab is active. The page contains the following fields and options:

- Project details:**
 - Subscription ***: Visual Studio Professional Subscription (dropdown)
 - Resource group ***: rg-demo (dropdown) with a 'Create new' link below it.
- Instance details:**
 - Registry name ***: Enter the name (text input) with a '.azurecr.io' suffix.
 - Location ***: Central India (dropdown)
 - Availability zones**: ☐ Enabled

At the bottom, there are three buttons: 'Review + create' (blue), '< Previous' (grey), and 'Next: Networking >' (grey).



- Search for “Cosmos DB” in Azure home
- Now configure The Cosmos DB

Home > New > Create Azure Cosmos DB Account

Create Azure Cosmos DB Account

For a limited time, create a new Azure Cosmos DB account with multi-region writes in any region, and receive up to 33% off for the life of your account. Restrictions apply.

Basics Networking Tags Review + create

Azure Cosmos DB is a globally distributed, multi-model, fully managed database service. [Try it for free](#), for 30 days with unlimited renewals. Go to production starting at \$24/month per database, multiple containers included. [Learn more](#)

Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * Visual Studio Professional

Resource Group * [Create new](#)

Instance Details

Account Name * validacosmosdbnotebooks

API * Core (SQL)

Notebooks (Preview) On Off

Location * (Asia Pacific) Australia East

Account Type Production Non-Production

Geo-Redundancy Enable Disable

Multi-region Writes Enable Disable

*Up to 33% off multi-region writes is available to qualifying new accounts only. Offer limited to accounts with both account locations and geo-redundancy, and applies only to multi-region writes in those same regions. Both Geo-Redundancy and Multi-region Writes must be enabled in account settings. Actual discount will vary based on number of qualifying regions selected.

[Review + create](#) [Previous](#) [Next: Networking](#)

7.2 MOVE CODE TO AZURE REPO

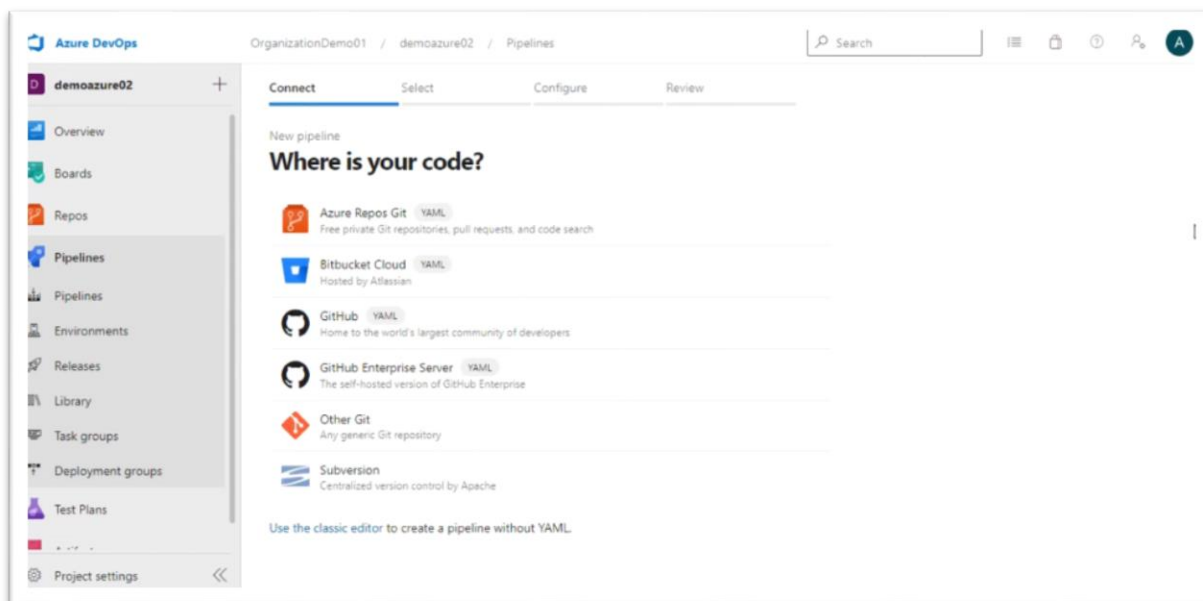
- Add the program to GITHUB repository
- To make it easy to access code by azure pipeline



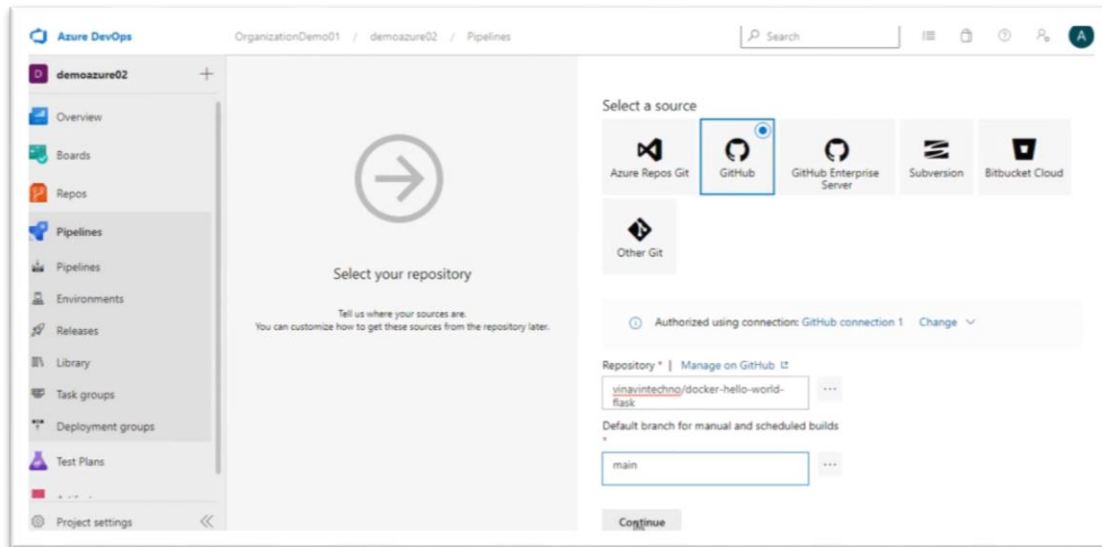
The screenshot shows a GitHub repository named 'docker-hello-world-flask' with a file named 'server.py'. The file content is as follows:

```
1 from flask import Flask
2
3 PORT = 8000
4 MESSAGE = "Hello, world! \n This is flask app(v-2.0.11). \n"
5
6 app = Flask(__name__)
7
8
9 @app.route("/")
10 def root():
11     result = MESSAGE.encode("utf-8")
12     return result
13
14
15 if __name__ == "__main__":
16     app.run(debug=True, host="0.0.0.0", port=PORT)
```

- Go to Azure DevOps
- Click pipeline and create new pipeline
- And choose classic editor to code without YAML

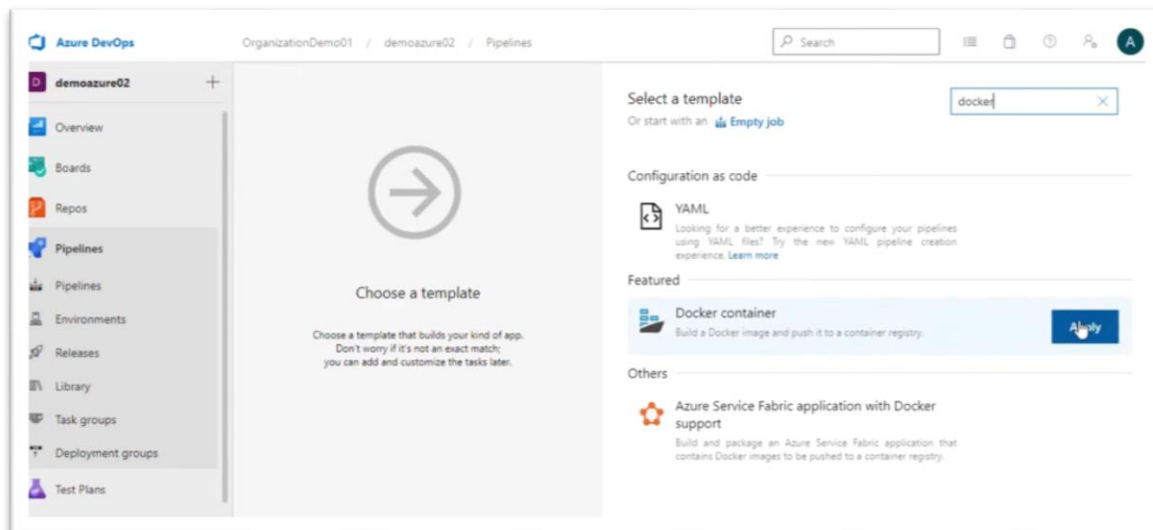


- Select GITHUB and fill the details required

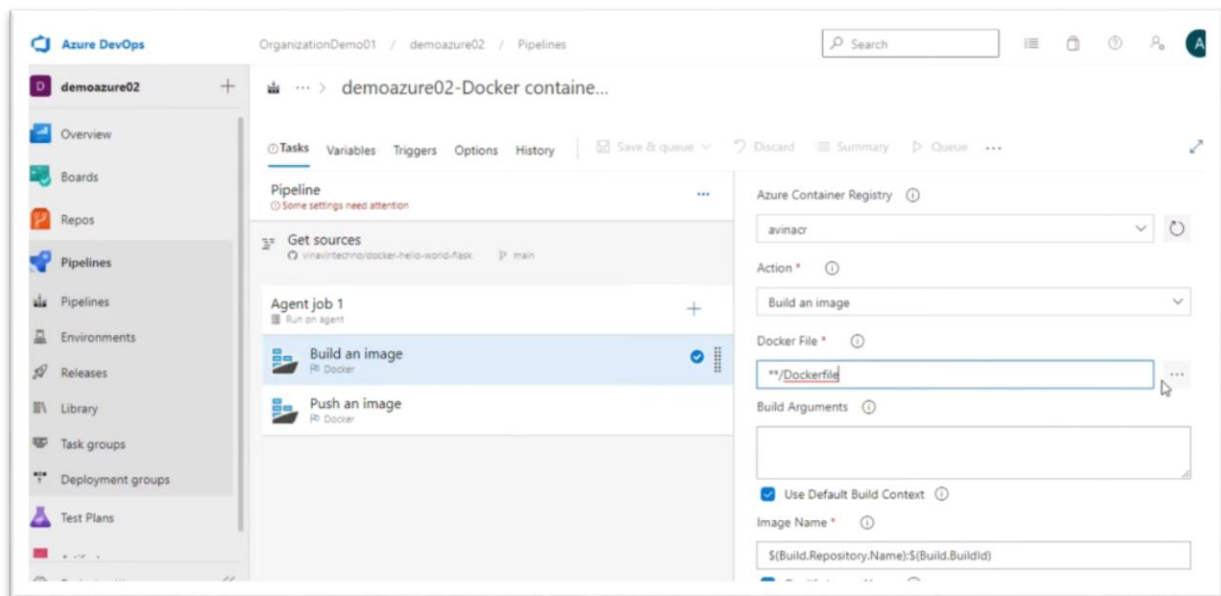


7.3 BUILDING AND PUSHING THE IMAGE TO AZURE APP SERVICE

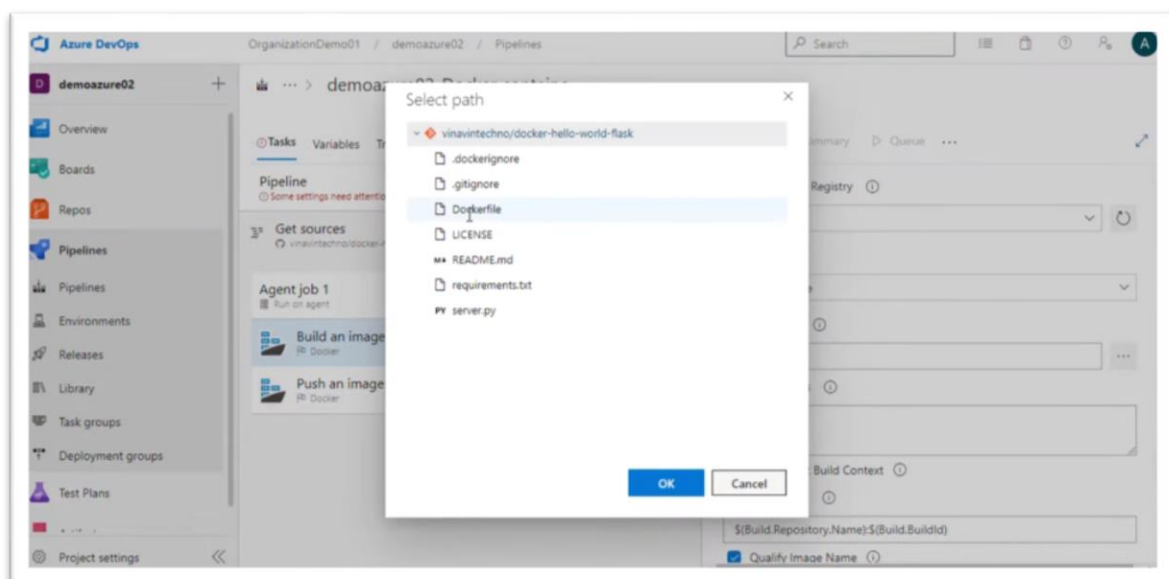
- Choose Docker Container as template



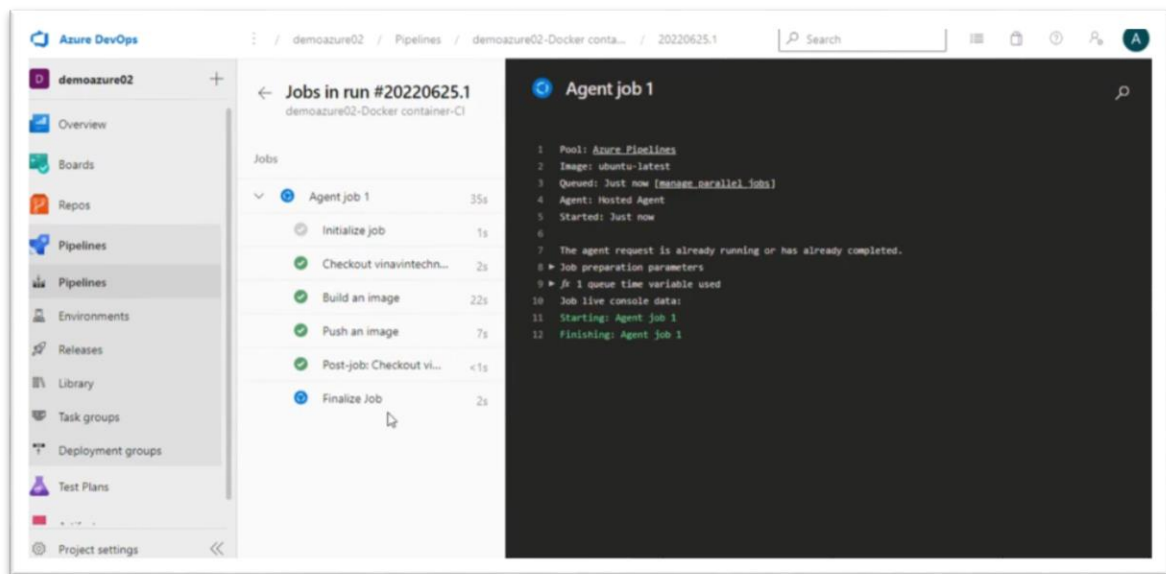
- Fill the details in the docker container
- There are two action settings Build and Push
- 1st lets configure Build Action



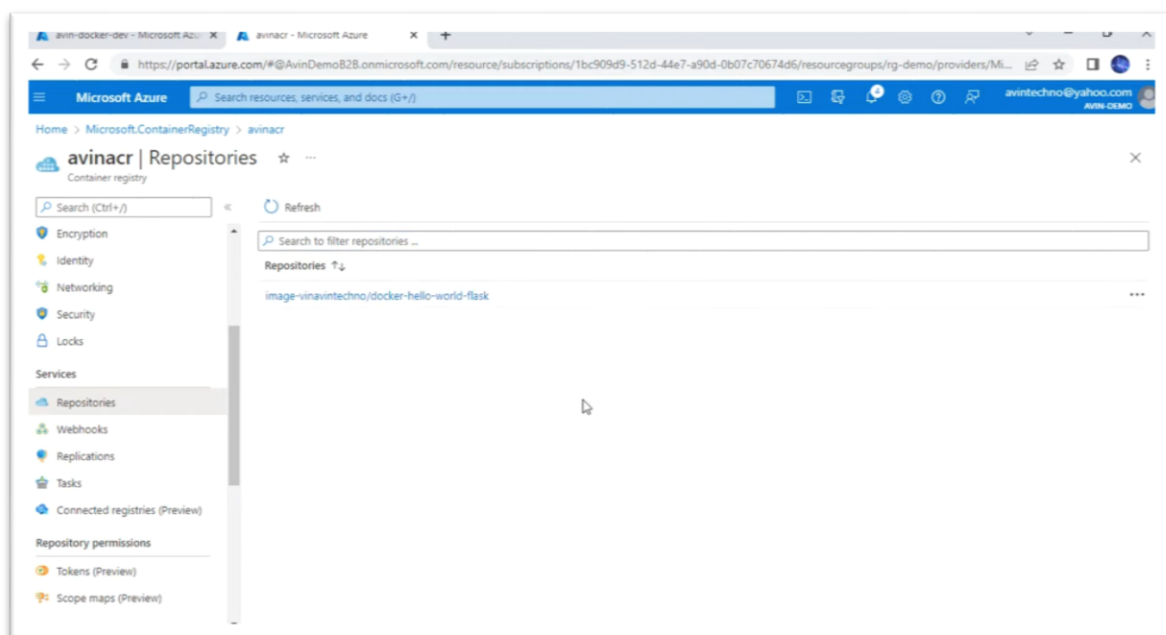
- Select the docker file from the GITHUB repository



- Configure Push Action settings next
- Fill same details as build action settings in the push action details
- After configuration, save & queue
- The task starts build and pushing process



- Check the repositories in the container registry to verify that image is pushed





7.4 CREATING AND DEPLOYING WEB APP IN AZURE APP SERVICE

- Create Web App in Azure App Service

Create Web App - Microsoft Azure

Home > App Services >

Create Web App

Operating System * ☒ Linux ☐ Windows

Region *
 Not finding your App Service Plan? Try a different region or select your App Service Environment.

App Service Plan

App Service plan pricing tier determines the location, features, cost and compute resources associated with your app. [Learn more](#)

Linux Plan (Central India) *
 [Create new](#)

Sku and size * **Basic B1**
 100 total ACU, 1.75 GB memory

Zone redundancy

An App Service plan can be deployed as a zone redundant service in the regions that support it. This is a deployment time only decision. You can't make an App Service plan zone redundant after it has been deployed. [Learn more](#)

Zone redundancy ☒ Enabled: Your App Service plan and the apps in it will be zone redundant. The minimum App Service plan instance count will be three.

[Review + create](#) [< Previous](#) [Next: Docker >](#)

- Configure the Docker information into Web App

Create Web App - Microsoft Azure

Home > App Services >

Create Web App

Basics Docker Networking (preview) Monitoring Tags Review + create

Pull container images from Azure Container Registry, Docker Hub or a private Docker repository. App Service will deploy the containerized app with your preferred dependencies to production in seconds.

Options

Image Source

Azure container registry options

Registry *

Image *

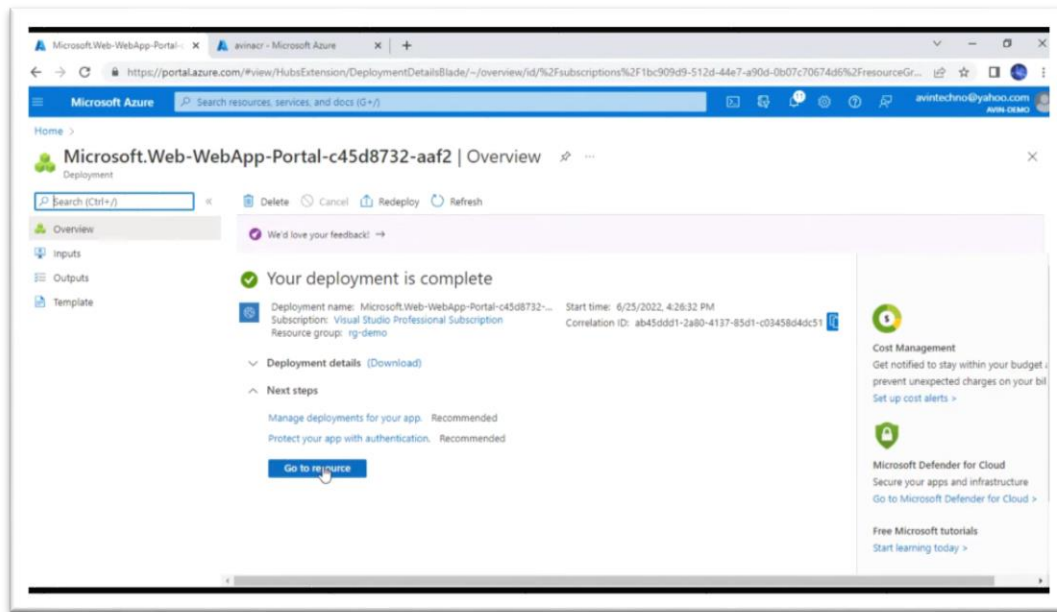
Tag *

Startup Command

[Review + create](#) [< Previous](#) [Next: Networking \(preview\) >](#)

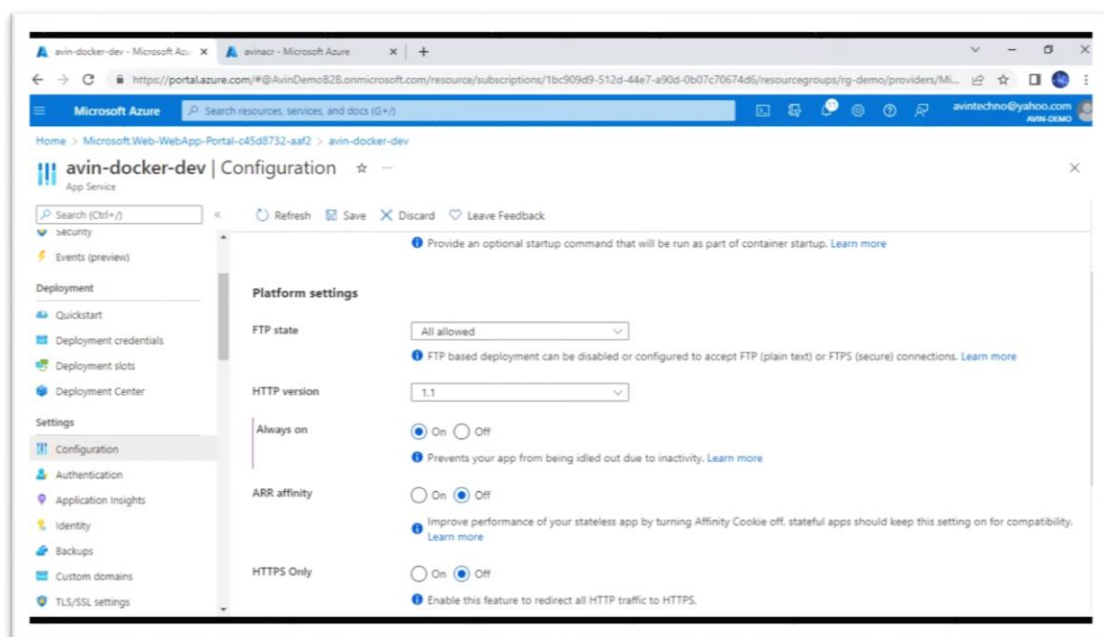


- Deploy after completing the configuration

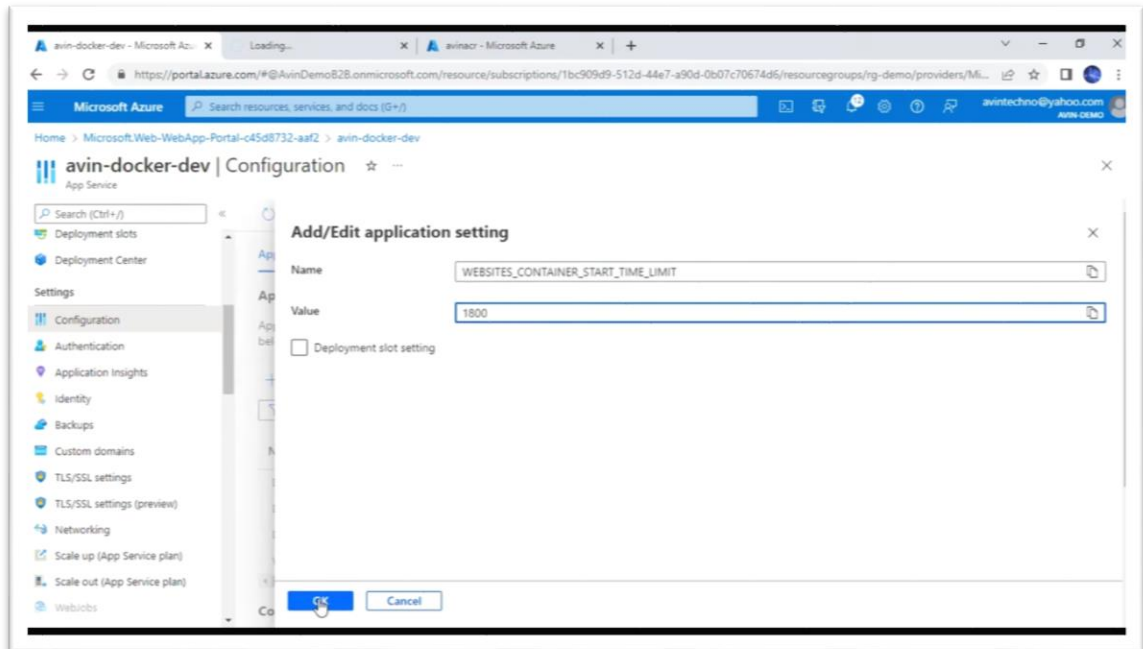


7.5 CONFIGURING THE AZURE APP SERVICE

- Configure the Web App using “Configuration” in App Service
- Visit General Settings and Turn On “Always On” under HTTP version

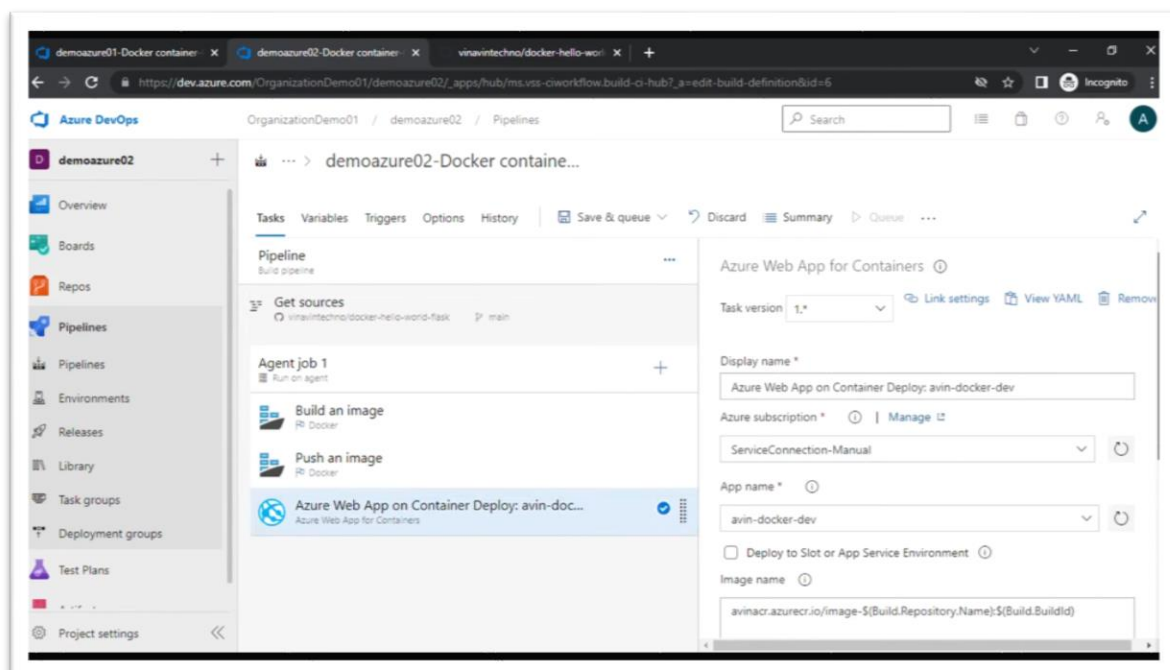


- In Application settings add new settings and also add `COSMOS_DB_CONNECTION_STRING` to link Cosmos DB to Azure App Service.



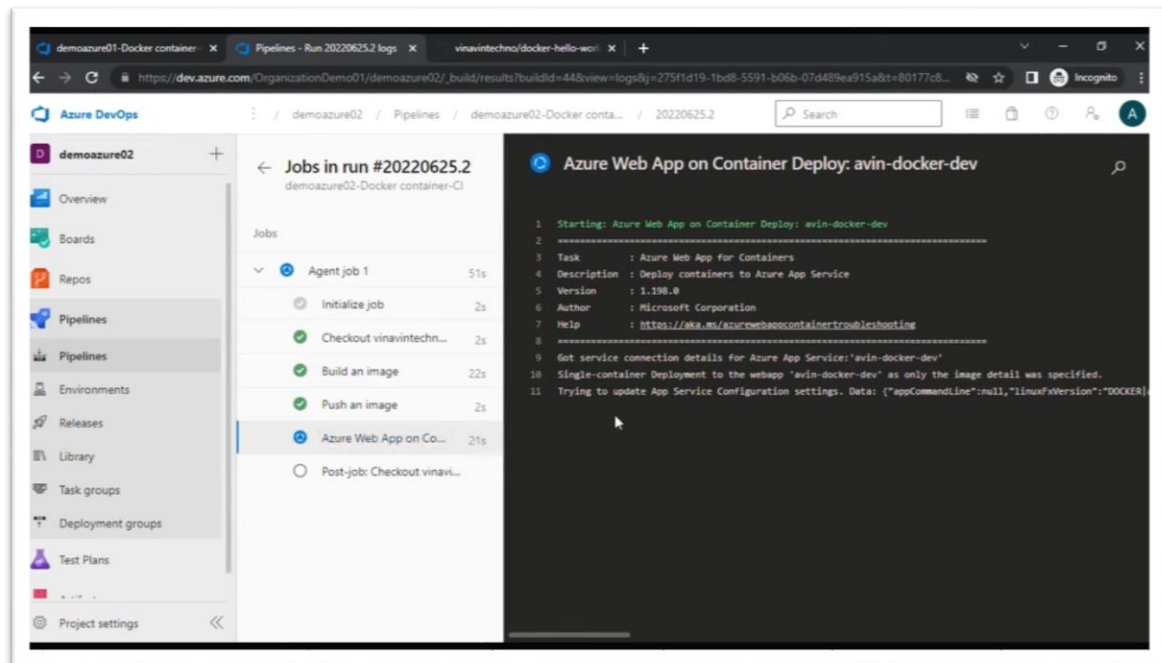
7.6 ADD AZURE WEB APP CONFIGURATION INTO AZURE DEVOPS

- Add “Azure Web App on Container Deploy” action in the Azure DevOps pipeline





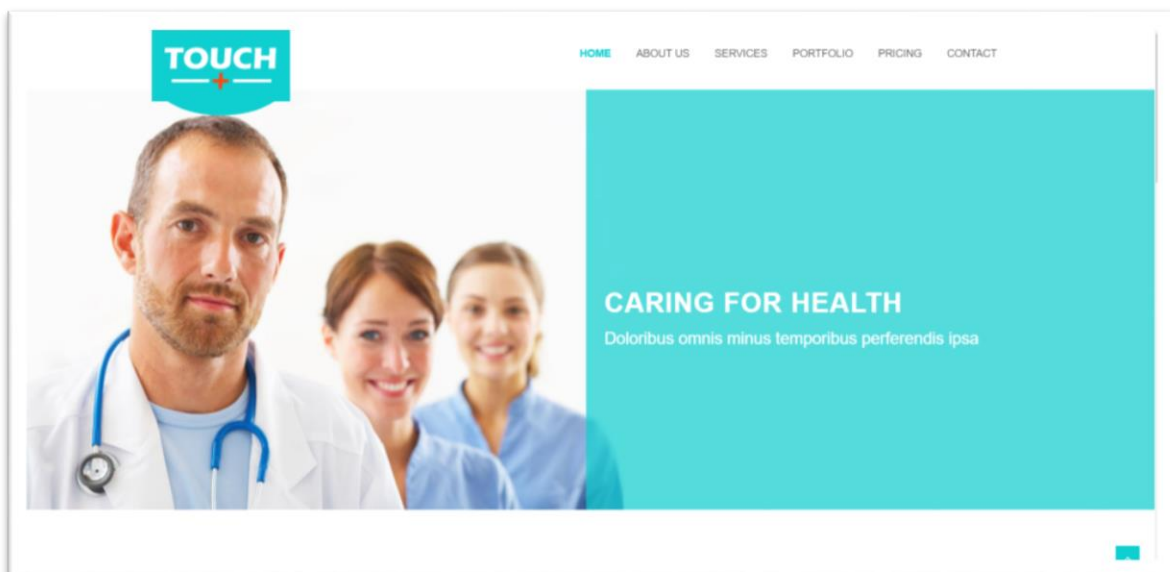
- Run the pipeline to see the new setting activate



7.7 DEPLOYMENT OF THE WEB PAGE

- Click on the default domain in the web app overview to view the web page hosted

DEMO PAGE





7.8 ADDING NETWORK RESTRICTION

- After Deploying Web page, we can use Networking option in webpage overview.
- We can configure Networking tool to restrict access based on IP address.

SUMMARY

In today's modern DevOps landscape, deploying a Docker image to Azure App Service using Azure DevOps involves creating a robust Build pipeline that efficiently containerizes and pushes the application to Azure Container Registry (ACR), all while adhering to Infrastructure as Code (IaC) principles, typically with tools like Azure Resource Manager (ARM) templates or terraform for streamlined resource management and scalability. To elevate the application's capabilities, integrating Cosmos DB, a cutting-edge NoSQL database solution, is essential. Cosmos DB offers global distribution, elastic scalability, and low-latency data access, aligning perfectly with the demands of contemporary cloud-native applications and positioning them to excel in the ever-evolving technology landscape