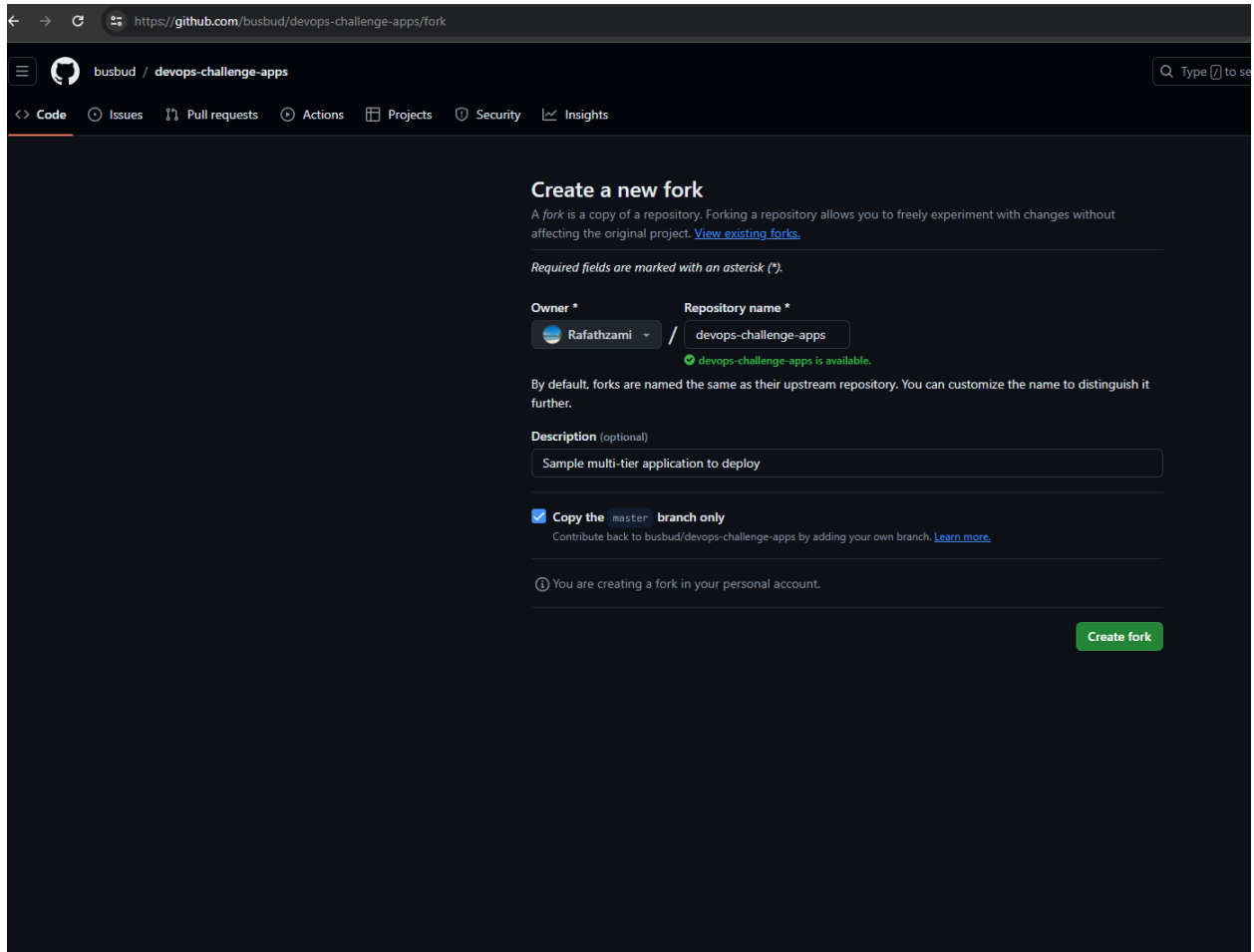


This application consists of a web app and an API that is connected to a database.

1. Create a fork from this repository.



The screenshot shows the GitHub interface for creating a new fork of the repository `busbud / devops-challenge-apps`. The page title is "Create a new fork". Below the title, a description states: "A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)". A note indicates "Required fields are marked with an asterisk (*)".

The form contains the following fields and options:

- Owner ***: A dropdown menu showing the user `Rafathzami`.
- Repository name ***: A text input field containing `devops-challenge-apps`. A green checkmark and message below the field state: "devops-challenge-apps is available."
- Description (optional)**: A text input field containing the text "Sample multi-tier application to deploy".
- Copy the master branch only**: A checked checkbox. Below it, a note says: "Contribute back to busbud/devops-challenge-apps by adding your own branch. [Learn more.](#)"

At the bottom of the form, there is a note: "You are creating a fork in your personal account." and a green button labeled "Create fork".

2. Write Docker files for Web app and API.

Web app docker file

FROM node:16-alpine

WORKDIR /app

#if we need to copy the build files we can uncomment below

#COPY dist/app.js ./

#please comment the below line if you're copying from the direct build artifacts

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 5000

CMD ["npm", "start"]

Api Docker file

FROM node:latest

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 5000

CMD ["npm", "start"]

3. Identify resources that you need and develop a deployment diagram.

Resources:

Servers / AZURE VM

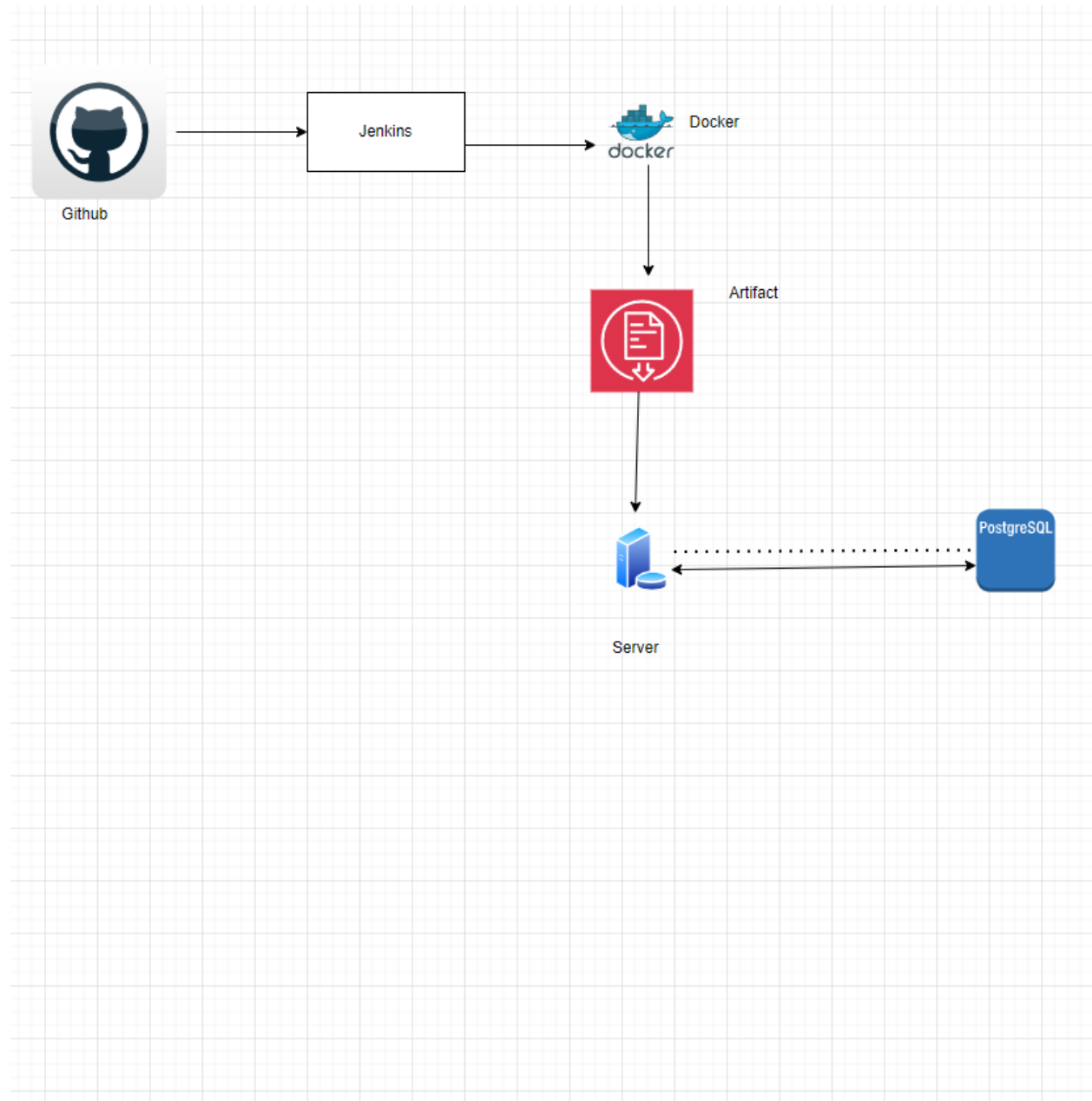
Web Server: - Nginx, Apache / Azure App Service, Application Gateway

Application containers: - Docker runtime, Web app & API docker images / Azure Container Registry

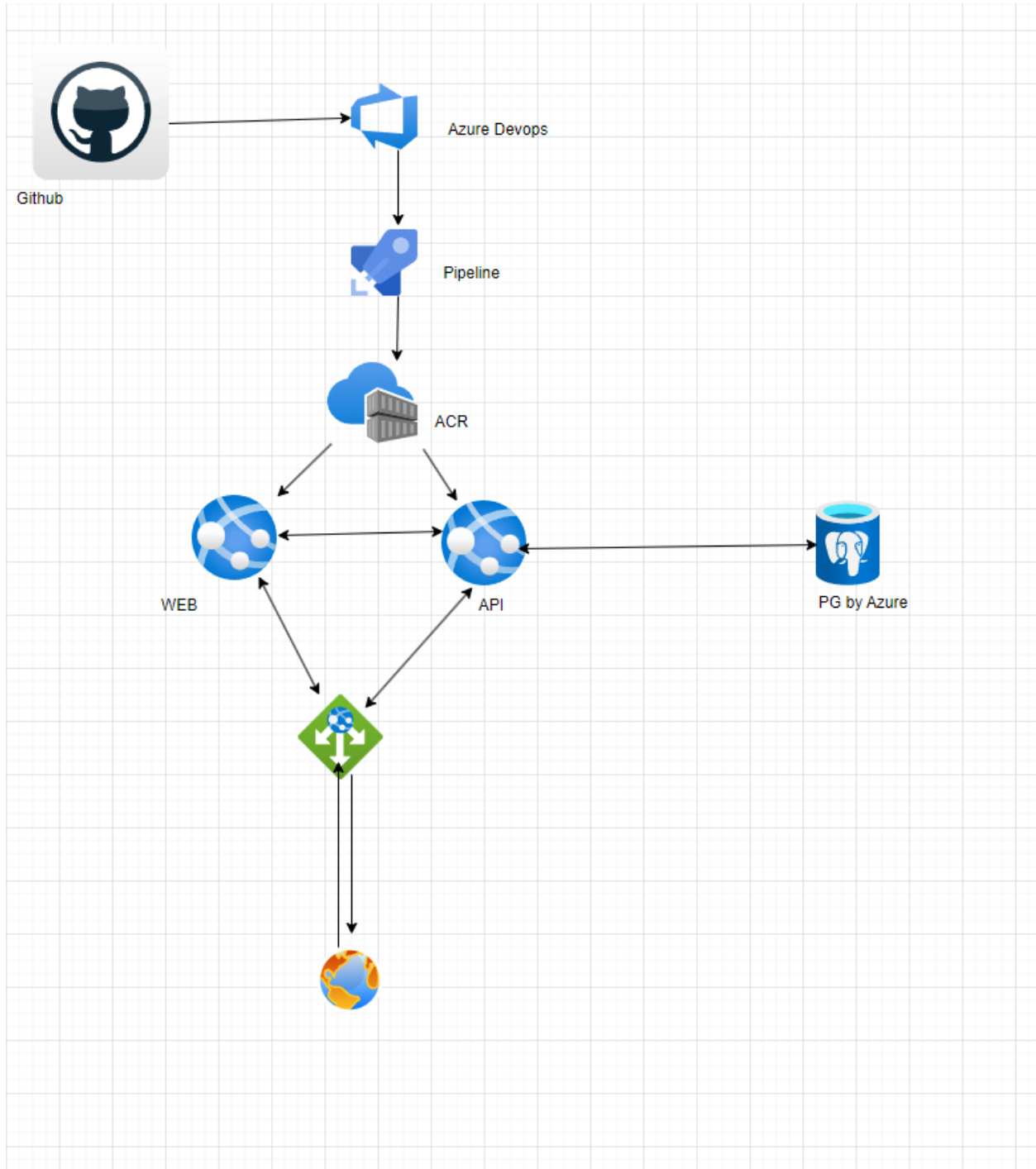
Database instance: Postgres / Azure Database for PostgreSQL

CI/CD : Jenkins / Azure Devops

Method 1 :



Method 2 :



4. Write the Terraform code to provision the infrastructure on a cloud of your choice.

Note : Terraform files are in the Terraform directory

5. Deploy Web app and API to the cloud. Make sure that you are using containerized technologies provided in the cloud platform

- I. We using GitHub for version control and Maintain our codes respectively
- II. Whenever a change happens into GitHub repos Azure DevOps will trigger a pipeline
- III. After that pipelines will build the docker images and Push it to ACR in Azure .
- IV. And from the pipeline itself we can configure to deploy our containers into App service.
- V. We can configure to use database connections from pipelines and we can configure from app service environment variables

6. Make sure all application components should be appropriately communicated.

- i. DNS Configuration: Make sure Dns is pointed to correct Ip address
- ii. Firewall and LB rules: Make sure firewall and LB rules are perfectly enabled so there won't be any blockages.
- iii. Application configs: Need to check the application ports and database, other URLs configured properly
- iv. Configure application insights and check the application logs to further troubleshoot or analyze the root cause

7. Write a CI/CD pipeline using GitHub Actions to automate the deployment process of the Web app and the API

Files are added to GitHub folder