README

Here's the general process to get your AWS backend up and running:
1. Install Docker on your AWS Lightsail instance
2. Transfer the appropriate files to your Lightsail instance
3. Build and launch your Docker image

Sounds easy enough, but this is quite detailed!

Let's first get docker installed on the AWS machine.

1. Log onto your AWS lightsail instance (click the >_ button to launch the virtual terminal)
2. Docker Intallation (last updated: 01/07/2026 Ref: https://docs.docker.com/engine/install/debian/):
    a. Set up Docker's apt repository. (COPY everything between the ```, and PASTE it in your server console)

```
# Add Docker's official GPG key:
sudo apt update
sudo apt install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/debian/gpg -o
/etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
sudo tee /etc/apt/sources.list.d/docker.sources <<EOF
Types: deb
URIs: https://download.docker.com/linux/debian
Suites: $(. /etc/os-release && echo "$VERSION_CODENAME")
Components: stable
Signed-By: /etc/apt/keyrings/docker.asc
EOF

sudo apt update
```

    b. install the latest version, run

```
sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-
plugin docker-compose-plugin
```

    c. Type ```sudo docker run hello-world```

If line c runs and says 'Hello from Docker!' and some more text about things you can do with Docker, then you have it running correctly.

Once docker is installed you can now transfer your files to the AWS machine using FileZilla.

First, create a new folder inside of /home/bitnami/ on the AWS machine, using either FileZilla or the AWS console, name it whatever you want.

Transfer the following files to your aws machine: docker-compose.yml, Dockerfile, requirements.txt
These should be in the same folder as your .py file and your .h5 models
All of this should be in some folder that you create inside of /home/bitnami/ on the aws machine

Before you do that though, you need to edit these 3 files a bit. (right click / open with / your favorite plain text editor)

in the Dockerfile and docker-compose.yml file there are references to FOLDERNAME
You can name this whatever you want, but it will be the home folder on your docker image
So, if you need to load a .h5 file in your code, then in your code you would write something like:

```
my_model = tf.keras.models.load_model('/FOLDERNAME/bitnami/path_to_model_on_aws_machine/tensorflow_model.h5')
```

There is also a reference to PYTHONFILE.py - This can be any name, I'm just using this as a placeholder name. It is the .py file that has all your anvil backend code and needs to be in the same directory as the requirements.txt, Dockerfile, and docker-compose.yml files on the aws machine

requirements.txt is a list of all the packages that need to be installed for your code to run - this may change for your code

Once all this is loaded to the right place on the aws machine via filezilla, to get your code up and running follow these instructions:

1. log into your aws account and launch the virtual lightsail terminal (the >_ button on your instance)
2. cd to the folder where all of these files are saved on the aws machine
3. Run the following 3 lines of code in the aws terminal
    a. sudo docker compose build
    b. sudo docker compose up -d
    c. sudo docker compose logs
4. line c just checks that your code is up and running. if anything gets printed in your python file it will be displayed here...a good check to make sure everything's working is on the line above anvil.server.wait_forever(), have a simple print command like print('everything is working!') as a gut check to make sure it's running correctly
5. if you want to make any changes to your code, you need to take down your docker image. to do that, in the terminal, after you cd to the correct folder run: sudo docker compose down
6. once you have fixed your code, and put the new version onto the aws machine, go back to step 2-3
7. Every docker image you launch and take down is saved on the aws machine, this can be a lot of data, to delete old docker image files your

can run the following line in the terminal: sudo docker system prune -a --filter â□□until=10mâ□□