Lab:
Building Docker Images

Dockerfile

- Open your codespace if it has closed
- The following `Dockerfile` will work for both the api and website Services
- Create a file called **`Dockerfile`** in both the `events-api` and `events-website` folders
  - You can use the codespaces visual editor to create these files
  - Or with nano or vi

```
# Use aws public image
FROM public.ecr.aws/docker/library/node:slim

# Copy application code.
COPY . /app/

# Change the working directory
WORKDIR /app

# Install dependencies.
RUN npm install

# Start the Express app
CMD ["node", "server.js"]
```

`.dockerignore`

- Create a file called **`.dockerignore`** in both the `events-api` and `events-website` folders:

```
node_modules
npm-debug.log
```

  - Be sure the name starts with a “`.`” and is all lowercase

Build

- To build `events-api` from the terminal in the **`events-api`** folder:

  ```
  cd /workspaces/eventsapp/events-api/
  docker build . -t events-api:v1.0
  ```

- To build `events-website` from a terminal in the **`events-website`** folder:

  ```
  cd /workspaces/eventsapp/events-website/
  docker build . -t events-website:v1.0
  ```

- To view the Docker images just built:

  ```
  docker images
  ```

Run Locally

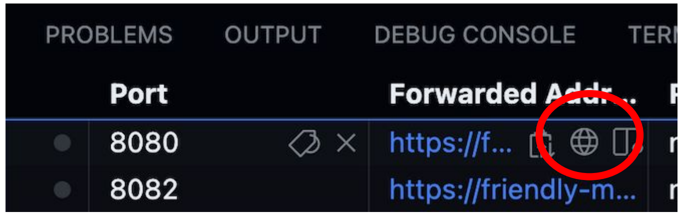
- To run `events-api`:

```
docker run -d -p 8082:8082 events-api:v1.0
```

- To run `events-website`:

```
docker run -d -p 8080:8080 -e SERVER='http://localhost:8082' --network="host" events-website:v1.0
```

  - To allow the website service to connect to the api service, we had to put the Docker networking into host mode
    - This allows the container to share the network namespace of the host
    - Otherwise it would not be allowed to connect to the api service
    - This also requires us to mannually open the port
- Click the **Ports** tab, click **Add Port**, type **`8080`** as the port, and press **ENTER**
  - Hover over the port 8080 line and click the globe to open in a browser
  - Test the app to ensure it will works



Port	Forwarded Addr...
8080	https://f...
8082	https://friendly-m...

Run Locally

- Other commands to try:
  - `docker images`
  - `docker ps -a`
  - `docker stop <ContainerID>`
  - `docker rm <ContainerID> --force`

Syncing the Changes to Git

- Commit these changes to your Git repository
  - On the left side, click the **Source Control** button
  - Be sure ALL changes are staged by clicking in the **+** button
  - Type a commit message of: **`Added Events app start code`** and click the **Commit** button
  - Press the **Sync Changes** button and press **OK** to push the changes
- The code has now been saved to your **eventsapp** Git repository created earlier

Success!

- **Congratulations**! You have successfully containerized the Events app