Measure What Matters - Deployment Docs

Glossary:

1. **VPS:** Virtual private server
2. **Node:** Server side version of Javascript, this is what the server runs on and also builds the frontend components
3. **NPM:** Node package manager, installs packages for node.
4. **Yarn:** Essentially npm with a new name.
5. **Sudo:** Is the command used when you would like to do things that require admin privileges
6. **PM2:** Is what keeps the project processes running on the server, even if the server is rebooted.
7. **Docker:** is a software layer that makes it easy to run pre-packaged apps in a process that is isolated from the main machine.
8. **Nginx:** Provides a reverse proxy to our processes and exposes them on port 80.
9. **Pipeline:** Refers to an automated process on Github that can build and update the application when new versions are released to master.

Before you begin:

1. Determine where you would like to install the project. We will cover deploying on a VPS (private server).
2. Setup proper firewall rules. Ports that should be open include: 22, 80, 443. All others should be closed. This will be different depending on your VPS provider.

Setting up your VPS:

1. Connect to your VPS (see: How to Connect to an SSH Server)
    a. Ensure you are connected as a non-root user who can run commands with `sudo`. Enter command: `groups | grep sudo` and ensure the group name “sudo” is printed.
2. Install build-essentials
    a. Run command `**sudo apt-get install build-essential -y**`
3. Install docker
    a. Complete steps 1 & 2 from the following article: How to install Docker on Ubuntu
    b. When you get to the step that says “su - ${USER}”, this will not work as you may not know the password for the current user. (If the account was

part of the image from your VPS provider). Instead, logout of the ssh session and reconnect.

   c. Ensure you can run docker commands without sudo, run command `docker run hello-world`. If this was successful then you are good to go!

4. Install node
   a. Run command `**curl -L https://git.io/n-install | bash**`, type 'y' when prompted
   b. Logout of your ssh session and reconnect again. Run node -v and ensure a version is printed.
5. Install pm2
   a. This is what keeps the server running if it crashes
   b. Run command: `npm install -g pm2`
6. Install yarn
   a. npm i -g yarn
7. Install nginx
   a. Run command: `sudo apt install nginx -y`
8. Your VPS should now be ready to deploy the application.

Deploy using git:

1. Clone the project
   a. Run the following commands:
      i. `cd`
         1. This should take you to the home directory
      ii. `git clone [https://github.com/UAlberta-CMPUT401/teamsnap-game-observation.git](https://github.com/UAlberta-CMPUT401/teamsnap-game-observation.git) mwm`
         1. Provide your github credentials, this should clone the repo into a directory called "mwm"
         2. If it fails, please ensure that you have permission to view the repository on Github
2. Install and build backend:
   a. Run commands:
      i. `cd mwm/backend`
      ii. `yarn install`
      iii. `yarn build`
3. Install and build frontend:
   a. Run commands:
      i. `cd ../frontend`
      ii. `npm install`
      iii. `npm run build`

4. Install and run the database:
    a. Run commands:
        i. `cd`
            1. This should take you to the home directory
        ii. `mkdir postgresData`
        iii. For some peace of mind and to make sure all is going well
            1. Enter command: `ls`, this should list “mwm n postgresData”.
            2. If they are all there, then you’re doing a great job.
        iv. First, pick a password. This will be used to lock down the postgres database in the next command
        v. Secondly determine your user path, type `cd postgresData && pwd && cd ../`
            1. This should print a directory path, save this for the next command
        vi. `docker run --name mwm-psql -e POSTGRES_DB=mwm -e POSTGRES_USER=mwm -e POSTGRES_PASSWORD=**<YOUR_PASSWORD_HERE>** -p 5432:5432 -v **<YOUR_DIRECTORY_PATH>**:/var/lib/postgresql/data --restart unless-stopped -d postgres:12`
            1. Replace the password with the one you chose
            2. Replace the directory path with the path from the command ran previously. (with pwd).
            3. An example command could be: `docker run --name mwm-psql -e POSTGRES_DB=mwm -e POSTGRES_USER=mwm -e POSTGRES_PASSWORD=VOtgTCqCIA3IpLEGqtV6GCVp -p 5432:5432  -v /home/ubuntu/postgresData:/var/lib/postgresql/data --restart unless-stopped -d postgres:12`
            4. Run command: `docker ps` and ensure there is a container with the name “mwm-psql”.
5. Install nginx configuration
    a. Run commands:
        i. `cd mwm`
        ii. `sudo rm -rf /etc/nginx/sites-enabled/default`
        iii. `sudo ./bin/nginx/create_lnk.sh`
        iv. `sudo systemctl restart nginx`
        v. Nginx should now be listening on the ports defined in the ecosystem file.
6. Start the app processes
    a. Run pm2 command:

        i. `pm2 restart ecosystem.config.js --update-env`
            1. This will start the apps based on the definitions in the ecosystem.config.js file.
        ii. `pm2 save`
            1. This saves the state of your PM2 instances so that pm2 will reboot them if the server is restarted.
    b. Navigate the the IP of your VPS in the browser. If IPV4 this will be something like: http://192.168.1.254/. If your VPS is running on the flashy new IPV6 then it will be similar to: http://[2605:fd00:4:1001:f816:3eff:fe34:d1f3]/.
7. Congratulations! The app is deployed on your vps. You can now hook up a domain name and you'll be off to the races.