# How to Run a Nuxt App with preinstall script in Elestio

When running a Nuxt.js application using runtime `NodeJs`, particularly with a `preinstall` script, you may encounter issues related to missing dependencies or commands not being found. In this guide, we'll walk you through how to set up a pipeline to run a Nuxt app, ensuring that all necessary dependencies (including Nuxt itself) are properly installed and that any `preinstall` scripts run smoothly.

We're considering your `package.json` to have something like this kind of script:

```
  "scripts": {
    "preinstall": "echo 'Running preinstall script...'",
    "build:icons": "tsx plugins/iconify/build-icons.ts",
    "build": "nuxt generate",
    "start": "nuxt start"
  },
```

If you run this type of script with our current `NodeJs` runtime, it may not work as expected. To resolve this issue, you need to switch to our `Docker` runtime and use the Dockerfile provided below. Before deploying the pipeline, please ensure that you place the Dockerfile in your code's root folder file named `Dockerfile`.

In this example, we assume the client(frontend) code is located in the `app` directory, so I reference `app`.

However, if your repository structure differs, make sure to adjust the paths accordingly.

```
FROM node:22

# Add a work directory
WORKDIR /app

# Copy entire application files before installing dependencies
COPY ./app/ .

# Install dependencies
```

```
RUN npm install --legacy-peer-deps

# .env Source destination argument
ARG source_file=./.env
ARG destination_dir=./app/.env

# .env copying management
RUN if [ -f "$source_file" ]; then \
        if [ "$source_file" != "$destination_dir" ]; then \
            echo "Copying $source_file to $destination_dir"; \
            cp "$source_file" "$destination_dir"; \
        else \
            echo "Source and destination paths are the same; skipping copy."; \
        fi \
    else \
        echo ".env has been added to dockerignore; skipping copy; if you want to copy it, remove it from
dockerignore."; \
    fi

# Change the working directory
# WORKDIR ./app/

# Build the app (Make sure to use the local Nuxt binary from node_modules)
RUN npx nuxt generate

# Expose port
EXPOSE 3000

# Start the app (Make sure to use the local Nuxt binary from node_modules)
CMD HOST=0.0.0.0 npx nuxt start
```

Once your Dockerfile is copied to the root directory of your repository, you can create a pipeline from our dashboard and select the `Docker` runtime for it.

---

Revision #5
Created 7 October 2024 11:01:44 by Amit
Updated 8 October 2024 09:50:32 by Amit