Hi! My name is Drew Shirts and I'll be sharing how to run an Angular app locally with Docker and Nginx ("Engine X").

A little about me: You can find me on Twitter @drewshirts. I'm a Software Engineer, currently working at the Church of Jesus Christ of Latter-Day Saints.

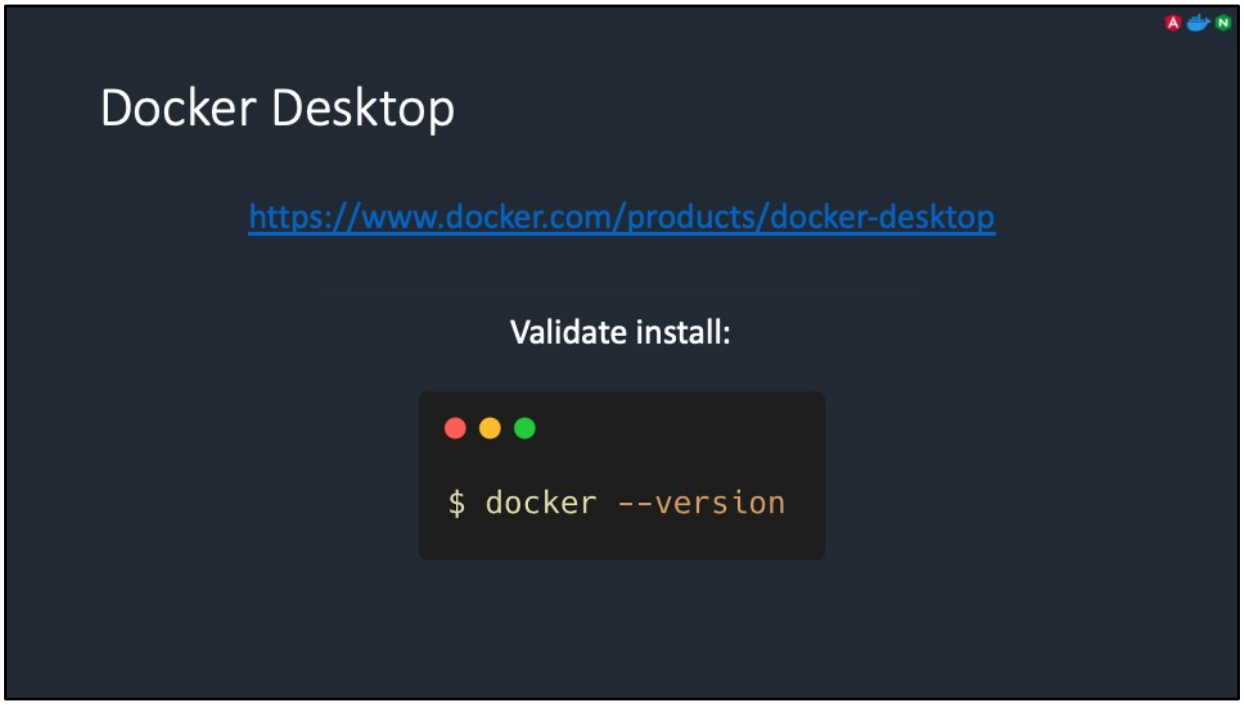
Tools

- Node.js
- NPM
- Visual Studio Code (Or your preferred editor)
- Docker Desktop
- Angular CLI

There are a few tools we'll be using for this presentation: Node, NPM, Visual Studio Code, Docker, and Angular CLI.

We won't be covering the install of Node and NPM in this tutorial, but you can download both together at nodejs.org.

I'll be using VS Code as the editor of choice for this tutorial.



Installing Docker Desktop will provide you with the CLI and the UI executables.

You can install the Angular CLI using this global npm command.

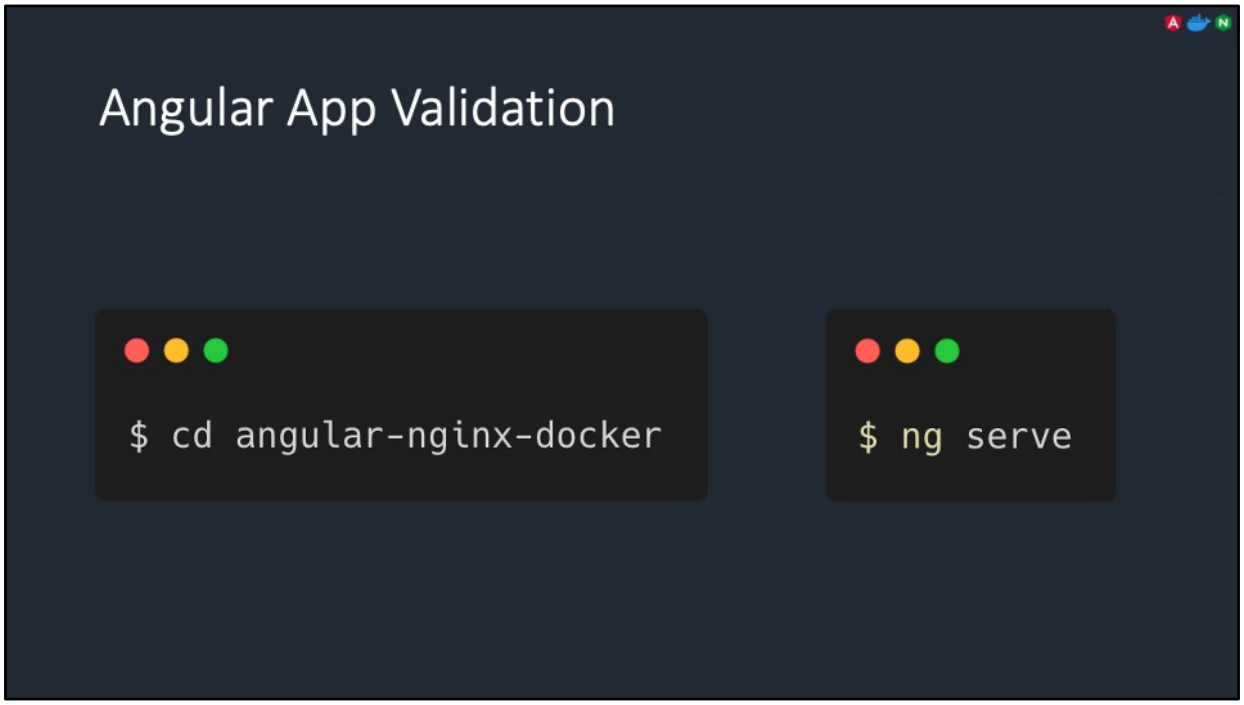
Angular App

```
$ ng new angular-nginx-docker --defaults
```

With these tools in place, let's create an Angular app with the default settings.

```
drewshirts ng-conf-2021 %ng new angular-nginx-docker --defaults
CREATE angular-nginx-docker/README.md (1027 bytes)
CREATE angular-nginx-docker/.editorconfig (274 bytes)
CREATE angular-nginx-docker/.gitignore (631 bytes)
CREATE angular-nginx-docker/angular.json (3647 bytes)
CREATE angular-nginx-docker/package.json (1210 bytes)
CREATE angular-nginx-docker/tsconfig.json (538 bytes)
CREATE angular-nginx-docker/tslint.json (3185 bytes)
CREATE angular-nginx-docker/.browserslistrc (703 bytes)
CREATE angular-nginx-docker/karma.conf.js (1437 bytes)
CREATE angular-nginx-docker/tsconfig.app.json (287 bytes)
CREATE angular-nginx-docker/tsconfig.spec.json (333 bytes)
CREATE angular-nginx-docker/src/favicon.ico (948 bytes)
CREATE angular-nginx-docker/src/index.html (304 bytes)
CREATE angular-nginx-docker/src/main.ts (372 bytes)
CREATE angular-nginx-docker/src/polyfills.ts (2830 bytes)
CREATE angular-nginx-docker/src/styles.css (80 bytes)
CREATE angular-nginx-docker/src/test.ts (753 bytes)
CREATE angular-nginx-docker/src/assets/.gitkeep (0 bytes)
CREATE angular-nginx-docker/src/environments/environment.prod.ts (51 bytes)
CREATE angular-nginx-docker/src/environments/environment.ts (662 bytes)
CREATE angular-nginx-docker/src/app/app.module.ts (314 bytes)
CREATE angular-nginx-docker/src/app/app.component.css (0 bytes)
CREATE angular-nginx-docker/src/app/app.component.html (25725 bytes)
CREATE angular-nginx-docker/src/app/app.component.spec.ts (982 bytes)
CREATE angular-nginx-docker/src/app/app.component.ts (224 bytes)
CREATE angular-nginx-docker/e2e/protractor.conf.js (904 bytes)
CREATE angular-nginx-docker/e2e/tsconfig.json (274 bytes)
CREATE angular-nginx-docker/e2e/src/app.e2e-spec.ts (671 bytes)
CREATE angular-nginx-docker/e2e/src/app.po.ts (274 bytes)
✔ Packages installed successfully.
   Successfully initialized git.
drewshirts ng-conf-2021 %
```

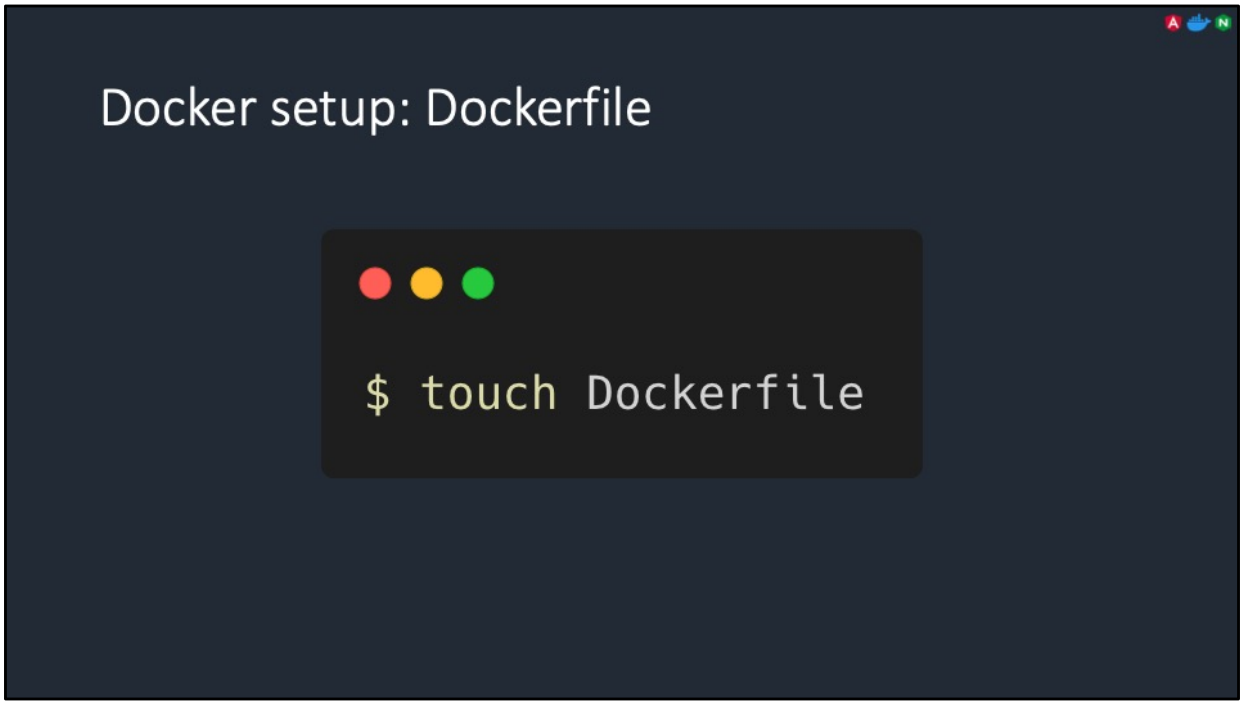
After creating the default project, it will run npm install for you. Depending on your internet connection, it can take a while.



Now, let's validate the angular app runs correctly with ng-serve.

```
drewshirts ng-conf-2021 %cd angular-nginx-docker
drewshirts angular-nginx-docker %ng serve
✔ Browser application bundle generation complete.

Initial Chunk Files | Names         |      Size
vendor.js           | vendor        |   2.43 MB
polyfills.js        | polyfills     | 128.80 kB
main.js             | main          |  56.03 kB
runtime.js          | runtime       |   6.15 kB
styles.css          | styles        | 119 bytes

                    | Initial Total |   2.62 MB

Build at: 2021-04-22T06:07:32.210Z - Hash: b217119e71be0af757fa - Time: 8060ms

** Angular Live Development Server is listening on localhost:4200, open your browser
on http://localhost:4200/ **


✔ Compiled successfully.
```

Once the build and compilation complete, it will start the local server on localhost, port 4200. Let's open up the browser and take a look.



Create a file named ‘Dockerfile’. The init-cap and no extension are on purpose.

Docker setup: .dockerignore

```
$ echo "node_modules" > .dockerignore
```

Create a file named '.dockerignore' and add the folder name 'node_modules'. This allows us to ignore our dependencies there just like we do with a .gitignore file.

Docker setup: Dockerfile Node Stage

Dockerfile > ...

```
# Create the node stage with the name "builder"
FROM node:latest as builder
# Set the working directory
WORKDIR /app
# Copy files from current directory to working directory
COPY . .
# Run npm install and build all assets
RUN npm i && npm run ng build
```

We'll be using a two-stage process for building out our Dockerfile. Our first stage will add a node image, copy all the Angular files to our working directory, run npm install to add project dependencies, and finally will build the app with npm.

Docker setup: Dockerfile Nginx Stage

Dockerfile > ...

```
# Create the nginx stage for serving content
FROM nginx:alpine
# Set the working directory to nginx asset directory
WORKDIR /usr/share/nginx/html
# Remove default nginx static assets
RUN rm -rf ./*
# Copy static assets from builder stage
COPY --from=builder /app/dist/angular-nginx-docker .
# Containers run nginx with global directives and daemon off
ENTRYPOINT ["nginx", "-g", "daemon off;"]
```

Our second stage for our Dockerfile will use an nginx image, set the working directory for nginx assets, remove any default assets from the nginx image, copy our node static assets from the builder image we just created, and specify the entrypoint for our Docker container to run nginx. You can add an additional configuration file for nginx to match your local settings with those of lower test lanes and production, but for this exercise, I'll keep it simple.

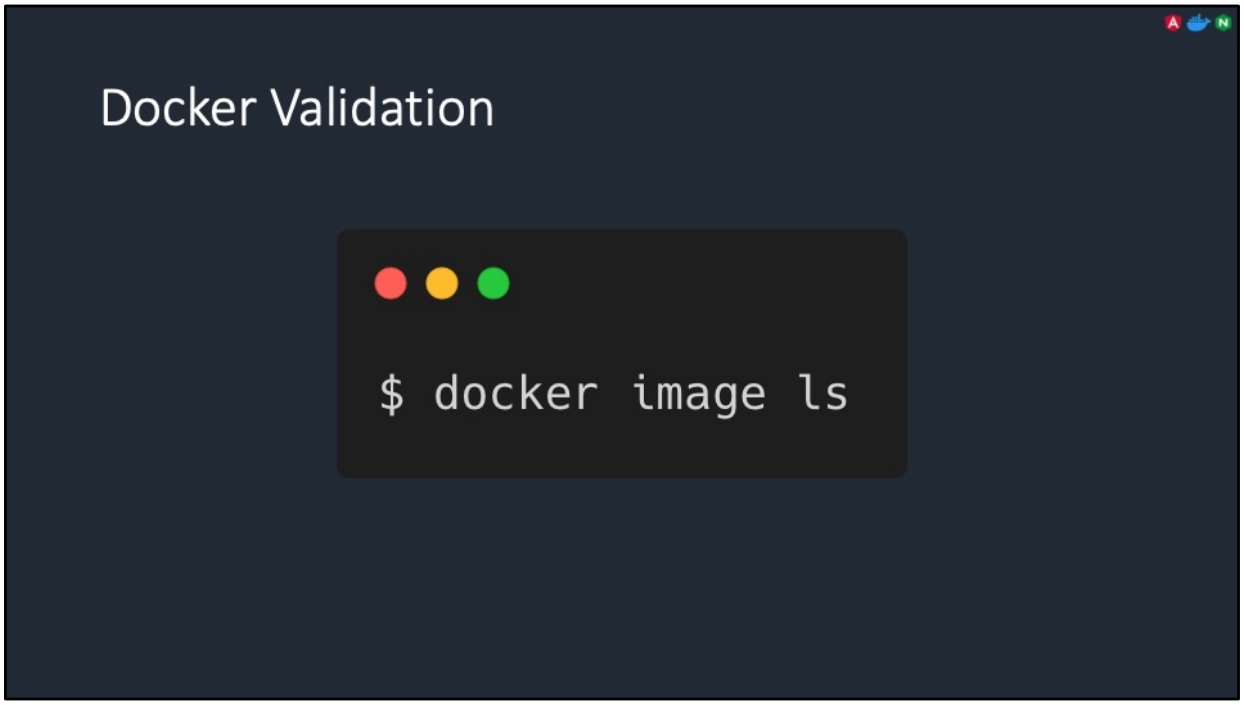
Build Docker Image

```
$ docker build -t angular-nginx-docker .
```

With these two stages added to our Dockerfile, we are ready to build an image called 'angular-nginx-docker'

```
drewshirts angular-nginx-docker %docker build -t angular-nginx-docker .
[+] Building 1.5s (14/14) FINISHED
 => [internal] load build definition from Dockerfile                                                    0.0s
 => => transferring dockerfile: 37B                                                                     0.0s
 => [internal] load .dockerignore                                                                       0.0s
 => => transferring context: 34B                                                                        0.0s
 => [internal] load metadata for docker.io/library/node:latest                                          1.4s
 => [internal] load metadata for docker.io/library/nginx:alpine                                         0.0s
 => [builder 1/4] FROM docker.io/library/node:latest@sha256:6cbc150709d59d2667f5d34cbf03fb4594dc        0.0s
 => [stage-1 1/4] FROM docker.io/library/nginx:alpine                                                   0.0s
 => [internal] load build context                                                                       0.0s
 => => transferring context: 6.61kB                                                                     0.0s
 => CACHED [stage-1 2/4] WORKDIR /usr/share/nginx/html                                                  0.0s
 => CACHED [stage-1 3/4] RUN rm -rf ./*                                                                 0.0s
 => CACHED [builder 2/4] WORKDIR /app                                                                   0.0s
 => CACHED [builder 3/4] COPY . .                                                                       0.0s
 => CACHED [builder 4/4] RUN npm i && npm run ng build                                                  0.0s
 => CACHED [stage-1 4/4] COPY --from=builder /app/dist/angular-nginx-docker .                           0.0s
 => exporting to image                                                                                  0.0s
 => => exporting layers                                                                                 0.0s
 => => writing image sha256:af7561fc75d33832d291e07f8e1023614d1fbf895b05c984c63512b22270cb3f            0.0s
 => => naming to docker.io/library/angular-nginx-docker                                                 0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
drewshirts angular-nginx-docker %
```

The build completed will look like this.

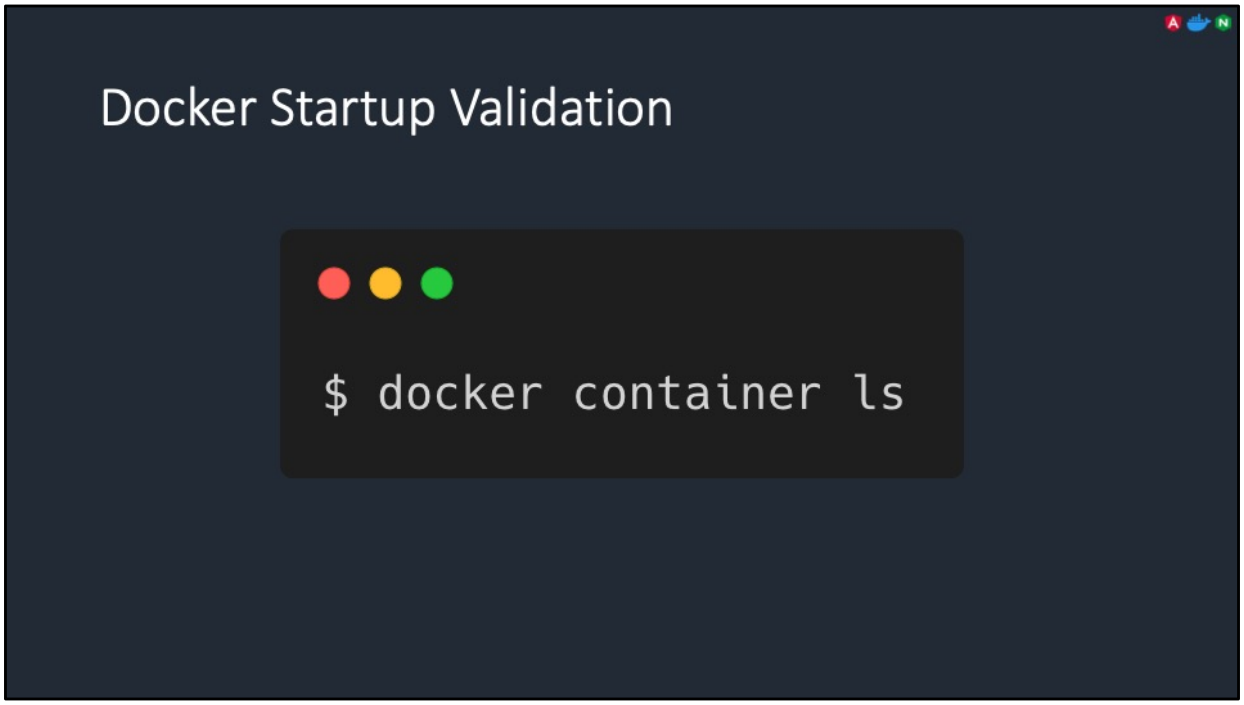


When the build is complete, you can validate the image was created with “docker image ls”. It will show all the Docker images you have on your computer.

Docker Startup

```
$ docker run --rm -it -p 8080:80 angular-nginx
```

Building complete, enter when ready. Now we can start a container, serving up our app on localhost, port 8080.



After the container is up and running you can see which containers are up with “docker container ls”.

Docker Management

```
$ docker container stop af756
$ docker container rm af756
```

When you make changes to your Angular app, you will need to create a new docker image. Use these when the container is still running. The hash at the end of these commands is the sha256 hash created when the container is built.

Thank you for your time today! I'm glad I've had this opportunity to share. It has helped me learn a lot about using Docker and Nginx. I plan on using this with my current projects at work.