Docker and Docker Compose

Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers. The service has both free and premium tiers.

Resources:

* <https://www.youtube.com/watch?v=3c-iBn73dDE&t=1485s>
* <https://www.youtube.com/watch?v=WmcdMiyqfZs>
* <https://www.youtube.com/watch?v=EoY1i8Ids1w>
* <https://www.youtube.com/watch?v=MVIcrmeV_6c>
* <https://www.youtube.com/watch?v=Qw9zlE3t8Ko>
* <https://www.youtube.com/playlist?list=PL4cUxeGkcC9hxjeEtdHFNYMtCpjNBm3h7>

Links:

* <https://docs.docker.com/>
* <https://docs.docker.com/compose/>
* <https://medium.com/@kmdkhadeer/docker-get-started-9aa7ee662cea>
* <https://towardsdatascience.com/docker-101-all-you-wanted-to-know-about-docker-2dd0cb476f03>

Courses:

* <https://kodekloud.com/courses/docker-for-the-absolute-beginner/?utm_source=google&utm_medium=&utm_id=16890563714&utm_content=&utm_term=&creativeId=>

Getting Started:

* [Docker overview](https://docs.docker.com/get-started/overview/)
* [Docker Installation](https://docs.docker.com/engine/install/)
* [Docker compose installation](https://docs.docker.com/compose/install/)
* [Docker Volumes](https://docs.docker.com/storage/volumes/)
* [Docker Networking](https://docs.docker.com/network/)

Certification:

* <https://training.linuxfoundation.org/training/containers-fundamentals/>

**Docker and Docker Composer:**

* Install docker and docker compose in your local machine linux
* Use docker cli to run a nginx server locally and expose it on port 8080
* Docker push an image to docker hub
* Run docker commands without sudo
* Learn all basic docker cli commands
* Write your first dockerfile to create a custom nginx server to output “hello World”
* Create a docker compose file which has 3 nginx services which outputs hello-world 1, 2, 3 respectively in a network.
* Attach docker volume and read the file dynamically in the container from outside
* Shell into a running container and execute basic commands
* Create 2 docker files of nginx with CMD and ENTRYPOINT respectively.
* Create a multi-stage build dockerfile for nginx

### ● **Install docker and docker compose in your local machine linux:**

Running Following Commands:

1. Update package list:

| sudo apt update |
| --- |

1. Install dependencies:

| sudo apt update |
| --- |

1. Add Docker's GPG key:

| curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add - |
| --- |

1. Add Docker repository:

| sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable" |
| --- |

1. Update package list again:

| sudo apt update |
| --- |

1. Install Docker Engine:

| sudo apt install -y docker-ce |
| --- |

1. Verify Docker installation:

| sudo systemctl status docker |
| --- |

1. Add user to Docker group:

| sudo usermod -aG docker $USER |
| --- |

1. Download Docker Compose:

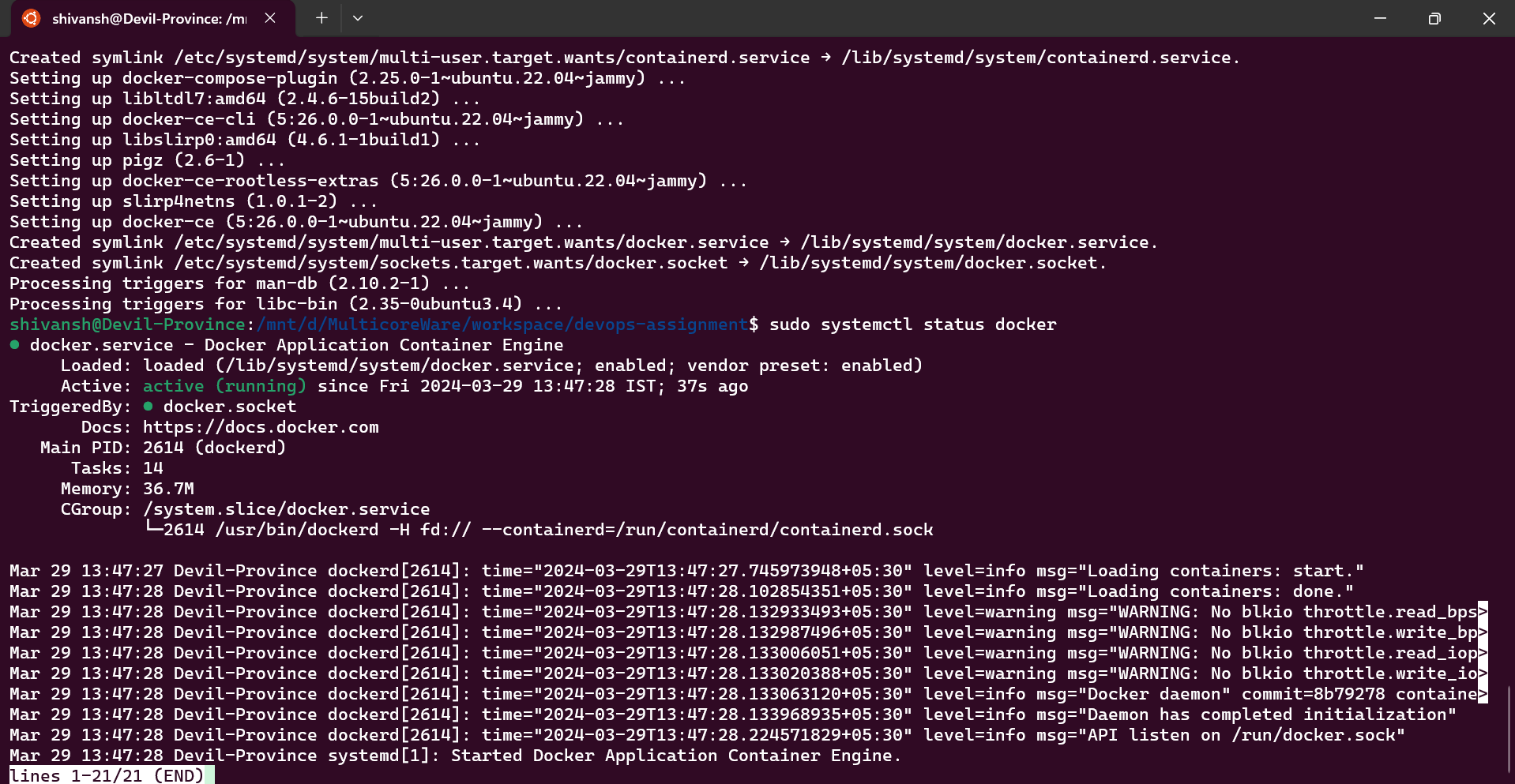
| sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose |
| --- |

1. Set executable permissions for Docker Compose:

| sudo chmod +x /usr/local/bin/docker-compose |
| --- |

1. Verify Docker Compose installation:

| docker-compose --version |
| --- |

Docker Installed Successfully: 

### ● Use docker cli to run a nginx server locally and expose it on port 8080

1. Pull Nginx Image

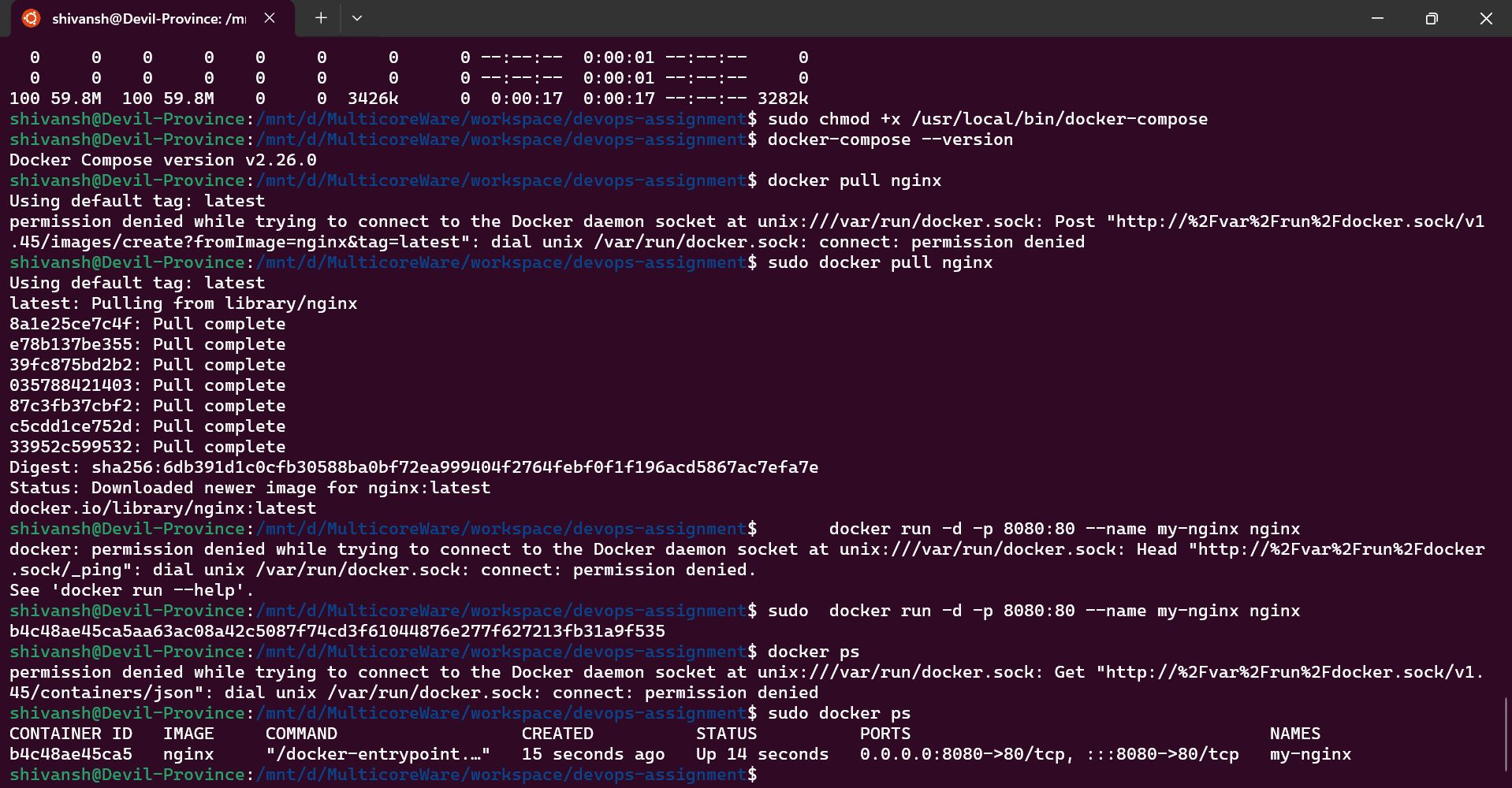
Sudo docker pull nginx

1. Run Nginx Container

docker run -d -p 8080:80 --name my-nginx nginx

1. Verify Nginx Container Status:

docker ps

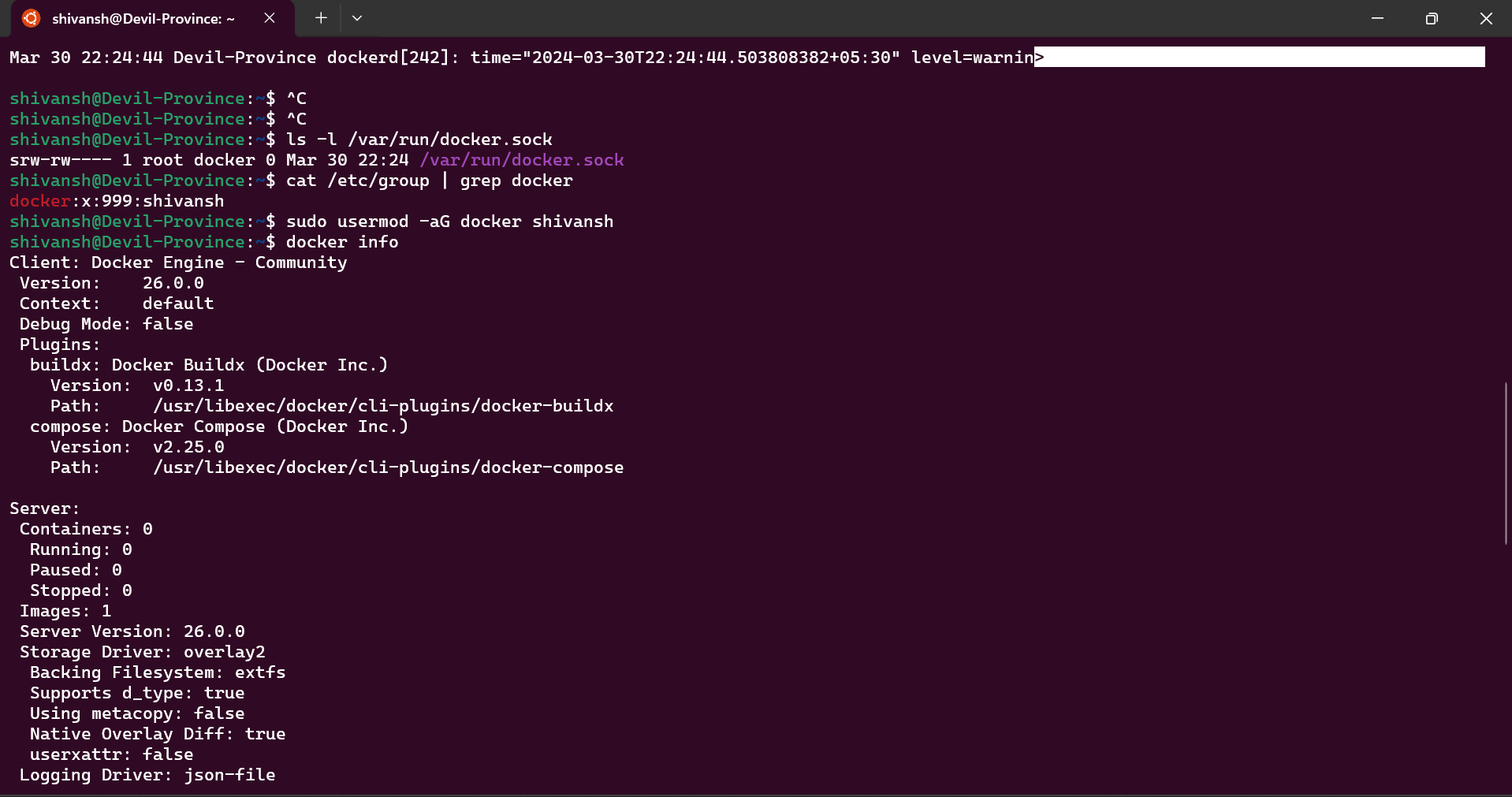


● Docker push an image to docker hub

| docker tag my-image:latest myusername/my-image:latest docker login docker push myusername/my-image:latest |
| --- |

● Run docker commands without sudo

| cat /etc/group | grep docker sudo usermod -aG docker <username> logout groups docker info |
| --- |



● Learn all basic docker cli commands

| # Display Docker version information docker version  # Show detailed Docker system information docker info  # List all Docker images docker images  # List running containers docker ps  # Pull a Docker image from a registry (e.g., Docker Hub) docker pull <image-name>  # Create and start a new container based on an image docker run <image-name>  # Create and start a container in detached mode (background) docker run -d <image-name>  # Stop a running container docker stop <container-id>  # Start a stopped container docker start <container-id>  # Restart a container docker restart <container-id>  # Remove a stopped container docker rm <container-id>  # Remove a Docker image docker rmi <image-id>  # Execute a command inside a running container in interactive mode docker exec -it <container-id> <command>  # Display logs of a running container docker logs <container-id>  # Build a Docker image from a Dockerfile and tag it docker build -t <image-name> <path-to-dockerfile>  # Start services defined in a Docker Compose file docker-compose up  # Stop and remove services defined in a Docker Compose file docker-compose down |
| --- |

● Write your first dockerfile to create a custom nginx server to output “hello World”

Create Project Directory and Navigate:

| mkdir custom-nginx cd custom-nginx |
| --- |

Create Dockerfile:

| cat > Dockerfile <<EOF FROM nginx:latest COPY nginx.conf /etc/nginx/nginx.conf RUN mkdir -p /usr/share/nginx/html COPY index.html /usr/share/nginx/html/index.html EXPOSE 80 CMD ["nginx", "-g", "daemon off;"] EOF |
| --- |

Create nginx.conf:

| cat > nginx.conf <<EOF worker\_processes 1; events {  worker\_connections 1024; } http {  server {  listen 80;  server\_name localhost;  location / {  root /usr/share/nginx/html;  index index.html;  }  } } EOF |
| --- |

Create index.html:

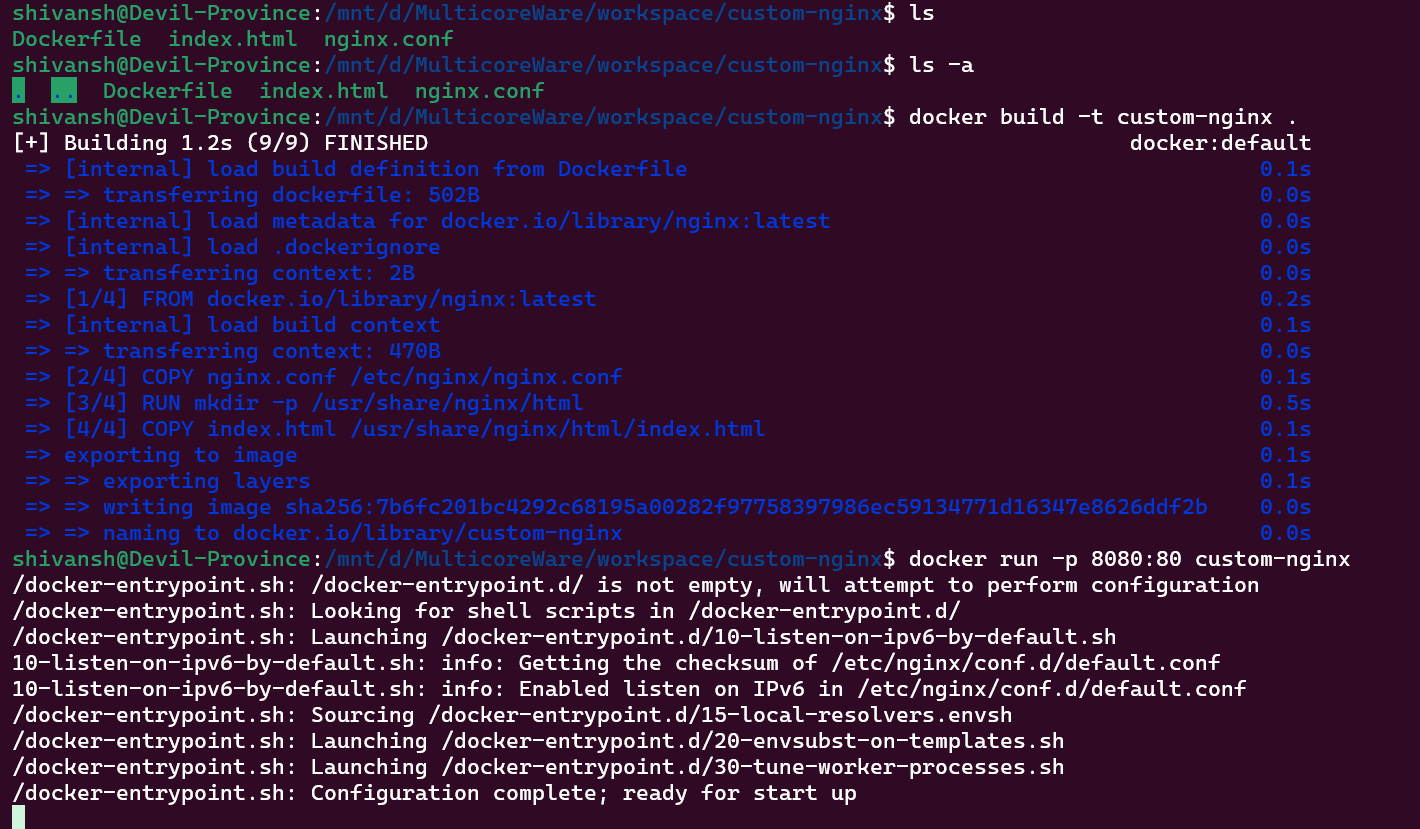
| cat > index.html <<EOF <!DOCTYPE html> <html> <head>  <title>Hello, World!</title> </head> <body>  <h1>Hello, World!</h1> </body> </html> EOF |
| --- |

Build Docker Image:

| docker build -t custom-nginx . |
| --- |

Run Docker Container:

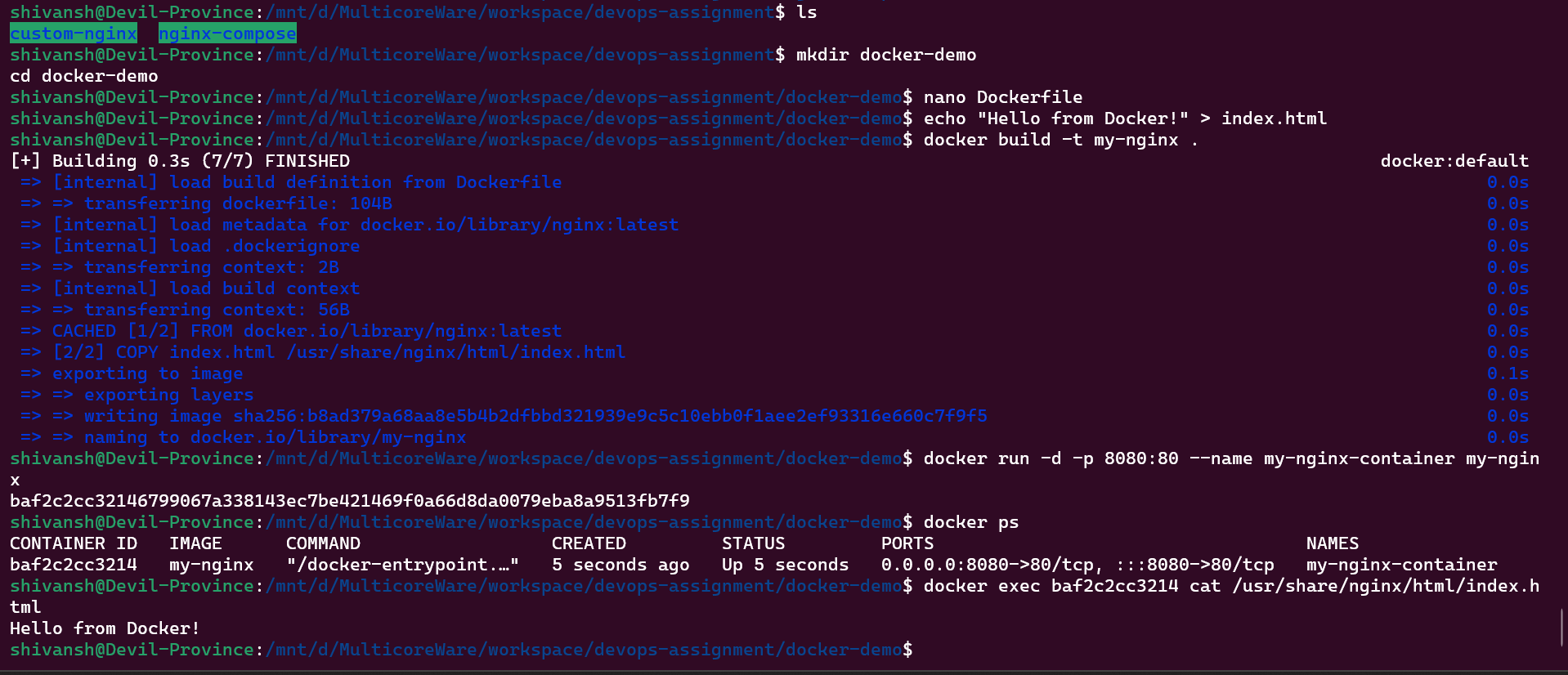
| docker run -p 8080:80 custom-nginx |
| --- |



● Create a docker compose file which has 3 nginx services which outputs hello-world 1, 2, 3

| respectively in a network.  cat > docker-compose.yml <<EOF version: '3.8'  services:  nginx1:  image: nginx:latest  ports:  - "8081:80"  environment:  - MESSAGE=hello-world 1  networks:  - my-network   nginx2:  image: nginx:latest  ports:  - "8082:80"  environment:  - MESSAGE=hello-world 2  networks:  - my-network   nginx3:  image: nginx:latest  ports:  - "8083:80"  environment:  - MESSAGE=hello-world 3  networks:  - my-network  networks:  my-network: EOF |
| --- |

● Attach docker volume and read the file dynamically in the container from outside



● Shell into a running container and execute basic commands

| # List running containers and find the ID or name of the container you want to shell into docker ps  # Shell into the container docker exec -it <container-id-or-name> /bin/bash  # Now you are inside the container's shell # You can execute any basic commands you need ls pwd whoami uname  # Exit the container's shell exit |
| --- |

● Create 2 docker files of nginx with CMD and ENTRYPOINT respectively.

Dockerfile using CMD:

| FROM nginx:latest COPY index.html /usr/share/nginx/html/index.html CMD ["nginx", "-g", "daemon off;"] |
| --- |

Dockerfile using ENTRYPOINT:

| FROM nginx:latest COPY index.html /usr/share/nginx/html/index.html ENTRYPOINT ["nginx", "-g", "daemon off;"] |
| --- |

● Create a multi-stage build dockerfile for nginx

| # Stage 1: Build stage FROM node:14 as build-stage  WORKDIR /app  COPY package\*.json ./ RUN npm install  COPY . . RUN npm run build  # Stage 2: Production stage FROM nginx:latest as production-stage  COPY --from=build-stage /app/build /usr/share/nginx/html  EXPOSE 80 CMD ["nginx", "-g", "daemon off;"] |
| --- |