

DOCKER

DOCKER

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. By using containers, developers can be sure that their applications will run on any other machine, regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code.

Docker is widely used in the development and deployment of modern applications, particularly in the context of microservices architectures. It allows developers to deploy their applications in a consistent and repeatable way, making it easier to develop and test applications and to deploy them to production.

CONCEPTS

Here are some key concepts related to Docker:

Images: A Docker image is a lightweight, stand-alone, executable package that includes everything needed to run a piece of software, including the application code, libraries, dependencies, and runtime.

Containers: A Docker container is a running instance of a Docker image. You can create, start, stop, move, or delete a container using the Docker API or CLI.

Registries: A Docker registry is a store for Docker images. You can think of a registry as a directory of all available Docker images. Docker Hub is a public registry that contains many pre-built images that you can use or you can set up your own private registry.

CONCEPTS


Dockerfile: A Dockerfile is a text file that contains the instructions for building a Docker image. You can use a Dockerfile to create a new image by specifying the base image to use, the dependencies to install, and other details required to set up the environment for your application.

Compose: Docker Compose is a tool for defining and running multi-container Docker applications. You can use Compose to define the services that make up your application in a YAML file, and then use a single command to create and start all of the services defined in the file.

INSTALL DOCKER IN WINDOWS

To install Docker on a machine running Windows, follow these steps:

1. Download the Docker Desktop installation package from the Docker website:
2. <https://www.docker.com/products/docker-desktop>
3. Double-click the downloaded .exe file to start the installation process.
4. Follow the prompts in the installation wizard to complete the installation.
5. Once the installation is complete, click the Docker icon in the system tray to open the Docker Desktop application.



6. Click the Sign In button in the upper right corner of the window and enter your Docker account credentials to log in.

7. When you are logged in, Docker Desktop will automatically download the necessary images and start the Docker daemon.

8. To verify that Docker is running correctly, open a terminal window and run the following command:

- Copy code
- `docker run hello-world`

9. This command will download the hello-world image and start a new container based on it. If the installation was successful, you should see a message saying "Hello from Docker!" printed to the console.


NEED FOR JENKINS WITH DOCKER

Jenkins is a popular open-source automation server that is often used for building, deploying, and managing applications. Docker is a containerization platform that allows you to package applications and their dependencies into lightweight, standalone containers that are easy to distribute and deploy.

There are several reasons why you might want to use Jenkins with Docker:

Ease of setup and configuration: Jenkins can be difficult to set up and configure, especially if you have a complex build pipeline. By using Docker, you can easily spin up a Jenkins instance and start building your applications without having to worry about installing and configuring all of the required dependencies.

Isolation and reproducibility: By using Docker containers, you can ensure that your builds are isolated and reproducible. This is especially important if you have multiple teams working on different projects, as it ensures that each team's builds are not impacted by changes made by other teams.




Ease of scaling: Jenkins can be resource-intensive, especially if you have a large number of builds running concurrently. By using Docker, you can easily scale your Jenkins environment up or down as needed, without having to worry about managing additional infrastructure.

Ease of deployment: Docker makes it easy to deploy your applications to different environments, whether it's on-premises or in the cloud. By using Jenkins with Docker, you can easily build and deploy your applications to different environments as part of your continuous delivery pipeline.

Overall, using Jenkins with Docker can help you streamline your application development and deployment process, and make it easier to manage and scale your Jenkins environment.

DOCKER PLAYGROUND

- Docker Playground is an online tool that allows you to run and experiment with Docker commands in a browser-based terminal. It is a convenient way to try out Docker without installing it on your local machine.
- To use Docker Playground, follow these steps:
- Go to the Docker Playground website:
- <https://labs.play-with-docker.com/>
- Click the Add New Instance button to create a new terminal session.



A new terminal window will open in the browser. You can use this terminal window to run Docker commands as if you were using a local installation of Docker.

To try out some basic Docker commands, you can run the following:

Copy code

View the version of Docker that is installed

```
docker --version
```

List the available Docker images

```
docker images
```

Run a simple "hello world" command in a new container


```
docker run hello-world
```

You can use Docker Playground to experiment with different Docker commands and see how they work. When you are finished, you can close the terminal window to end the session.

DOCKER COMMANDS

Here are some common Docker commands that you might find useful:

- **docker run**: Run a command in a new container.
- **docker start**: Start one or more stopped containers.
- **docker stop**: Stop one or more running containers.
- **docker build**: Build an image from a Dockerfile.
- **docker pull**: Download an image from a registry.
- **docker push**: Upload an image to a registry.
- **docker exec**: Run a command in an existing container.
- **docker search**: Search for an image in a registry.
- **docker logs**: Fetch the logs of a container.

- 
- `docker inspect`: Inspect the details of an image or container.
 - `docker network`: Manage networks.
 - `docker system`: Manage Docker.
 - You can use these commands to manage containers, images, and networks, and to view and modify the configuration of Docker. For more information about these commands and their options, you can refer to the Docker documentation or use the `--help` option with any of the commands.
 - For example, to view the help information for the `docker run` command, you can run:
 - Copy code
 - `docker run --help`