# Docker

Great! Let's start by breaking down the fundamental concepts and then proceed to a hands-on Docker installation and container setup on an EC2 instance.

**What is Docker?**

Docker is a platform that simplifies the process of building, running, managing, and distributing applications in containers. It ensures that your application behaves the same, regardless of where it's run: your laptop, a cloud server, or a CI/CD pipeline.

**What is a Container?**

A container is a lightweight, standalone, and executable package that includes everything needed to run a piece of software: code, runtime, system tools, libraries, and settings. Containers use the host system’s kernel, which makes them much lighter and faster than traditional virtual machines (VMs).

**Containers vs. Virtual Machines**

**Virtual Machines**

* **Isolation**: Each VM runs a full OS.
* **Resources**: VMs are heavier since they require their own OS.
* **Boot Time**: Slower to start because of the full OS boot.
* **Portability**: VMs are portable but large and cumbersome to transfer.

**Containers**

* **Isolation**: Containers share the host OS kernel.
* **Resources**: Containers are lightweight and use fewer resources.
* **Boot Time**: Very fast startup as they do not need to boot an OS.
* **Portability**: Containers are highly portable and small.

**Docker Components**

1. **Docker Engine**: The runtime that allows you to build and run containers.
2. **Docker Images**: Read-only templates that define a container.
3. **Docker Containers**: Running instances of Docker images.
4. **Docker Hub**: A registry of Docker images where you can find and share containers.

**Setting up Docker on an EC2 Instance**

Let's go through the steps to set up Docker on an EC2 instance:

**Step 1: Launch an EC2 Instance**

1. **Log in to AWS Console**.
2. **Navigate to EC2 Dashboard**.
3. **Launch an Instance**.
   * Choose an Amazon Machine Image (AMI): Select "Ubuntu Server 20.04 LTS".
   * Choose an Instance Type: t2.micro (sufficient for this demo and eligible for free tier).
   * Configure Instance: Default settings.
   * Add Storage: Default settings.
   * Add Tags: Optional.
   * Configure Security Group: Create a new security group that allows SSH (port 22).
   * Review and Launch: Click "Launch", then choose or create a new key pair.

**Step 2: Connect to Your EC2 Instance**

Use SSH to connect to your instance. If you're using a local terminal:

sh

Copy code

ssh -i "path\_to\_your\_key\_pair.pem" ubuntu@your\_ec2\_public\_dns

**Step 3: Install Docker on Ubuntu**

Once connected to your EC2 instance, install Docker by following these steps:

1. **Update your package index**:

sh

Copy code

sudo apt-get update

1. **Install required packages**:

sh

Copy code

sudo apt-get install apt-transport-https ca-certificates curl software-properties-common

1. **Add Docker’s official GPG key**:

sh

Copy code

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

1. **Add Docker’s official APT repository**:

sh

Copy code

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

1. **Update your package index again**:

sh

Copy code

sudo apt-get update

1. **Install Docker**:

sh

Copy code

sudo apt-get install docker-ce

1. **Start and enable Docker**:

sh

Copy code

sudo systemctl start docker

sudo systemctl enable docker

1. **Verify Docker installation**:

sh

Copy code

sudo docker --version

**Step 4: Run Your First Container**

1. **Run a simple hello-world container**:

sh

Copy code

sudo docker run hello-world

This command downloads the hello-world image from Docker Hub (if not already available locally) and runs it in a container.

1. **List running containers**:

sh

Copy code

sudo docker ps

1. **List all containers (running and stopped)**:

sh

Copy code

sudo docker ps -a

**Conclusion**

Docker is a powerful tool that simplifies the process of application deployment by using containers. Containers are lightweight and efficient compared to traditional VMs, making them ideal for modern cloud-native applications.

By setting up Docker on an EC2 instance, you can start experimenting with containerization and experience its benefits firsthand. You can further explore Docker features like networking, volumes, and Docker Compose for orchestrating multi-container applications.

# Up Docker

**Setting Up Docker on an EC2 Instance Running Ubuntu 18.04**

Let's go through the detailed steps to launch an EC2 instance, install Docker, and verify the setup. Follow these instructions to get Docker up and running on your EC2 instance.

**Step 1: Launch an EC2 Instance**

1. **Log in to AWS Management Console**.
2. **Navigate to EC2 Dashboard**.
3. **Launch an Instance**:
   * **Choose an Amazon Machine Image (AMI)**: Select "Ubuntu Server 18.04 LTS".
   * **Choose an Instance Type**: t2.micro (eligible for free tier).
   * **Configure Instance Details**: Accept the default settings.
   * **Add Storage**: Accept the default storage settings.
   * **Add Tags**: Add a tag for easier identification, e.g., Name = docker-engine.
   * **Configure Security Group**:
     + Create a new security group named docker-security-group.
     + Add the following rules:
       - **SSH**: Port 22, Source: My IP
       - **All Traffic**: Source: My IP (for accessing Docker containers on different ports).
   * **Review and Launch**: Click "Launch", then select or create a new key pair for SSH access. Download the key pair file (.pem) and keep it safe.

**Step 2: Connect to Your EC2 Instance**

Use SSH to connect to your EC2 instance. Replace path\_to\_your\_key\_pair.pem with the path to your downloaded key pair file, and your\_ec2\_public\_dns with the public DNS of your EC2 instance.

sh

Copy code

ssh -i "path\_to\_your\_key\_pair.pem" ubuntu@your\_ec2\_public\_dns

**Step 3: Install Docker on Ubuntu 18.04**

Once connected to your EC2 instance, follow these steps to install Docker:

1. **Update your package index**:

sh

Copy code

sudo apt-get update

1. **Install required packages**:

sh

Copy code

sudo apt-get install apt-transport-https ca-certificates curl software-properties-common

1. **Add Docker’s official GPG key**:

sh

Copy code

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

1. **Add Docker’s official APT repository**:

sh

Copy code

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

1. **Update your package index again**:

sh

Copy code

sudo apt-get update

1. **Install Docker**:

sh

Copy code

sudo apt-get install docker-ce

1. **Start and enable Docker**:

sh

Copy code

sudo systemctl start docker

sudo systemctl enable docker

1. **Verify Docker installation**:

sh

Copy code

sudo docker --version

**Step 4: Manage Docker as a Non-Root User**

By default, only the root user can run Docker commands. To enable your user (ubuntu) to run Docker commands without sudo, follow these steps:

1. **Add your user to the docker group**:

sh

Copy code

sudo usermod -aG docker ${USER}

1. **Apply the new group membership**:

sh

Copy code

su - ${USER}

1. **Verify that your user can run Docker commands**:

sh

Copy code

docker images

If you see a list of images (or an empty list if no images are downloaded yet), the setup is successful.

**Step 5: Run Your First Docker Container**

1. **Run the hello-world container to test your setup**:

sh

Copy code

docker run hello-world

This command will download the hello-world image from Docker Hub (if it's not already downloaded) and run it in a container. The container prints a "Hello from Docker!" message and then exits.

1. **Check running containers**:

sh

Copy code

docker ps

1. **Check all containers (running and stopped)**:

sh

Copy code

docker ps -a

You should see the hello-world container listed with a status of Exited.

**Conclusion**

You've successfully launched an EC2 instance, installed Docker, and verified the installation by running a test container. This setup allows you to start experimenting with Docker containers and further explore Docker's capabilities, such as networking, volumes, and orchestration with Docker Compose.