**Usecases**

1. **Create an EC2 instance in a public subnet**

Write a Terraform script to provision a simple EC2 instance in a public subnet. The EC2 instance should be based on a specific AMI (Amazon Machine Image) and should be associated with a security group that allows SSH access. Also, attach an EBS volume to the instance.

1. Write a **Create a VPC with Public and Private Subnets**

Terraform script to create a VPC with two subnets: one public subnet and one private subnet. The script should configure the public subnet with an Internet Gateway for internet access and the private subnet without internet access. Additionally, the public subnet should have an associated route table that routes traffic to the Internet Gateway.

1. **Launch EC2 Instance in Private Subnet with NAT Gateway for Internet Access**

Write a Terraform script to launch an EC2 instance in a private subnet within a VPC. The script should also set up a NAT Gateway in the public subnet to provide internet access to the private subnet. The route table for the private subnet should route internet-bound traffic to the NAT Gateway.

1. **Create an Auto Scaling Group with EC2 Instances**

Write a Terraform script to create an Auto Scaling Group (ASG) that automatically launches EC2 instances based on demand. Set a minimum, maximum, and desired capacity for the ASG, and associate it with an Application Load Balancer (ALB) for traffic distribution.

1. **Provision an Application Load Balancer (ALB) and Route Traffic to EC2 Instances**

Write a Terraform script to create an ALB in front of EC2 instances running in a public subnet. Set up listener rules to route HTTP traffic to the EC2 instances in the target group.

1. **EC2 Instance with S3 Access Using IAM Role**

Write a Terraform script provision an **EC2 instance** that generates log files (e.g., application logs or system logs). The logs need to be securely uploaded to an **S3 bucket**. To achieve this, the EC2 instance must have permissions to read and write to the S3 bucket, but without exposing any sensitive credentials.

1. **Provision an ECS Cluster with EC2 Instances**

Write a Terraform script to provision an ECS (Elastic Container Service) cluster using EC2 instances as the compute resource. The script should launch EC2 instances within a private subnet and automatically register them with the ECS cluster. Additionally, configure an Auto Scaling group to scale the number of EC2 instances based on demand.

1. **Deploy ECS Service with Fargate Launch Type**

Write a Terraform script to deploy a containerized application on ECS using the Fargate launch type. The script should define a task definition that specifies the Docker image stored in ECR (Elastic Container Registry), and create an ECS service that runs the containerized application on Fargate within a specified VPC and subnet.

1. **Monitor EC2 CPU Utilization with CloudWatch and Trigger Lambda Function**

Write a Terraform script to monitor the CPU utilization of an EC2 instance using AWS CloudWatch, and automatically trigger Lambda function when the CPU utilization exceeds a specific threshold (e.g., 80%) for a defined period.

commands to install docker in terminal

**ChatGPT said:**

Here are the commands to install Docker on different operating systems via terminal:

**🐧 For Ubuntu (20.04/22.04+)**

bash

Copy code

# 1. Update packages

sudo apt update

# 2. Install dependencies

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

# 3. Add Docker’s official GPG key

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

# 4. Add Docker repo

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] \

https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# 5. Update again and install Docker

sudo apt update

sudo apt install docker-ce -y

# 6. Start Docker and enable on boot

sudo systemctl start docker

sudo systemctl enable docker

# 7. (Optional) Run Docker without sudo

sudo usermod -aG docker $USER

newgrp docker

**🧑‍💻 Check Docker Installation**

bash

Copy code

docker --version

docker run hello-world

Would you like the same instructions for **CentOS**, **Amazon Linux**, or **Windows WSL**?

4o

**You said:**

[ec2-user@ip-172-31-9-43 ~]$ sudo apt update sudo: apt: command not found

**ChatGPT said:**

You're seeing this error because your EC2 instance is likely running **Amazon Linux**, **Amazon Linux 2**, or **a CentOS-based** AMI — which uses yum or dnf, **not** apt.

**✅ Use These Commands for Amazon Linux 2 or RHEL/CentOS-like AMIs:**

bash

Copy code

# 1. Update existing packages

sudo yum update -y

# 2. Install required packages

sudo yum install -y yum-utils

# 3. Add Docker repo

sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

# 4. Install Docker

sudo yum install -y docker-ce docker-ce-cli containerd.io

# 5. Start Docker service

sudo systemctl start docker

# 6. Enable Docker to start on boot

sudo systemctl enable docker

# 7. (Optional) Run Docker without sudo

sudo usermod -aG docker ec2-user

newgrp docker

**🔎 Test Docker**

bash

Copy code

docker --version

docker run hello-world

Let me know if your EC2 instance is based on **Ubuntu** or another distro — I’ll tailor the commands accordingly.

4o

**You said:**

[ec2-user@ip-172-31-9-43 ~]$ sudo yum install -y docker-ce docker-ce-cli containerd.io Docker CE Stable - x86\_64 459 B/s | 397 B 00:00 Errors during downloading metadata for repository 'docker-ce-stable': - Status code: 404 for https://download.docker.com/linux/centos/2023.7.20250428/x86\_64/stable/repodata/repomd.xml (IP: 3.164.240.105) Error: Failed to download metadata for repo 'docker-ce-stable': Cannot download repomd.xml: Cannot download repodata/repomd.xml: All mirrors were tried Ignoring repositories: docker-ce-stable Last metadata expiration check: 0:00:54 ago on Tue May 6 07:24:50 2025. No match for argument: docker-ce No match for argument: docker-ce-cli No match for argument: containerd.io Error: Unable to find a match: docker-ce docker-ce-cli containerd.io

**ChatGPT said:**

Thanks for the detailed error — this is a known issue when using **Amazon Linux 2023**, which **does not support Docker directly from Docker’s CentOS repo** (as it’s based on Fedora and uses dnf instead of yum).

**✅ 🔧 Solution: Use Amazon Linux Extras or install docker from the official Amazon repo (if using AL2)**

But since you're on **Amazon Linux 2023**, here's how to install Docker properly:

**✅ Install Docker on Amazon Linux 2023**

bash

Copy code

# 1. Update package list

sudo dnf update -y

# 2. Install Docker

sudo dnf install -y docker

# 3. Start the Docker service

sudo systemctl start docker

# 4. Enable Docker on boot

sudo systemctl enable docker

# 5. (Optional) Run Docker without sudo

sudo usermod -aG docker ec2-user

newgrp docker

**🧪 Test Docker**

bash

Copy code

docker version

docker run hello-world

Let me know if it still fails — I can give you an alternate approach using **Docker’s install script** or help with **containerd** as an alternative.

4o

**You said:**

○ docker.service - Docker Application Container Engine Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; preset: disabled) Active: inactive (dead) TriggeredBy: ○ docker.socket Docs: https://docs.docker.com

**ChatGPT said:**

That message means **Docker is installed**, but the service is not yet **running** or **enabled**.

Let’s fix that:

**✅ Start Docker and Enable It on Boot**

Run the following commands:

bash

Copy code

# Start Docker service

sudo systemctl start docker

# Enable Docker to run at boot

sudo systemctl enable docker

# Check status

sudo systemctl status docker

If the last command shows active (running), Docker is working correctly.

**🧪 Then Test It**

Run:

bash

Copy code

docker run hello-world

You should see a message confirming that Docker is installed and working.

Let me know if it still shows "inactive" or throws any error — I’ll help debug further.

4o

**You said:**

[ec2-user@ip-172-31-9-43 ~]$ docker run hello-world docker: permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Head "http://%2Fvar%2Frun%2Fdocker.sock/\_ping": dial unix /var/run/docker.sock: connect: permission denied. See 'docker run --help'

**ChatGPT said:**

That error means your current user (ec2-user) does **not have permission** to access the Docker daemon (which runs as root).

You have two options to fix this:

**✅ Option 1: Run Docker with sudo (Works immediately)**

bash

Copy code

sudo docker run hello-world

**✅ Option 2: Add Your User to the Docker Group (Preferred for convenience)**

This lets you run Docker commands **without sudo**:

bash

Copy code

# Add ec2-user to the docker group

sudo usermod -aG docker ec2-user

# Apply the group change (option 1: log out and back in, or)

newgrp docker

Then test again:

bash

Copy code

docker run hello-world

🧠 **Why this happens**:  
Docker’s daemon (dockerd) runs as root, and only users in the docker group can interact with it without sudo.

Let me know if you want me to set up a full Docker + Compose environment next.

4o

**You said:**

[ec2-user@ip-172-31-9-43 ~]$ git clone https://github.com/iam-veeramalla/Docker-Zero-to-Hero.git bash: git: command not found

**ChatGPT said:**

You're seeing bash: git: command not found because **Git is not installed** on your EC2 instance.

**✅ Install Git on Amazon Linux**

For **Amazon Linux 2023** or **Amazon Linux 2**, run:

bash

Copy code

sudo dnf install -y git

If you're using Amazon Linux 2 and get an error with dnf, try:

bash

Copy code

sudo yum install -y git

After installation, verify with:

bash

Copy code

git --version

Then re-run your command:

bash

Copy code

git clone https://github.com/iam-veeramalla/Docker-Zero-to-Hero.git

Let me know if you'd like help navigating that repo or running a specific project inside it.

4o

**You said:**

PS C:\Users\VijayPrasanthBurgula\Docker-Zero-to-Hero\examples\first-docker-file> sudo apt update sudo : The term 'sudo' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again. At line:1 char:1 + sudo apt update + ~~~~ + CategoryInfo : ObjectNotFound: (sudo:String) [] , CommandNotFoundException + FullyQualifiedErrorId : CommandNotFoundException

**ChatGPT said:**

You're running the command:

powershell

Copy code

sudo apt update

in **PowerShell on Windows**, which is why you're getting:

csharp

Copy code

'sudo' is not recognized...

**⚠️ Here's Why:**

* sudo and apt are **Linux commands**.
* **PowerShell** on Windows does **not** recognize them.
* If you're trying to install Linux packages or work with Docker in a Linux environment, you need to use a **Linux shell**, such as:

**✅ Your Options:**

**1. Use WSL (Windows Subsystem for Linux)**

Install and use Ubuntu on Windows:

* Open PowerShell as Administrator and run:

powershell

Copy code

wsl --install

* After it installs and reboots, open **Ubuntu** and run:

bash

Copy code

sudo apt update

This is the best way to run Linux commands on Windows.

**2. Use EC2 Terminal (SSH)**

If you're working on a remote EC2 instance:

* Keep using the terminal inside your SSH session — not PowerShell.

**3. Use Docker Desktop (for Windows)**

If you're trying to run Docker locally:

* Install **Docker Desktop for Windows**.
* Then you can use PowerShell or CMD to run commands like:

powershell

Copy code

docker run hello-world

Let me know if you'd like help setting up WSL or Docker Desktop on your machine.

4o

**You said:**

C:\Users\VijayPrasanthBurgula\Docker-Zero-to-Hero\examples\first-docker-file>sudo yum update -y 'sudo' is not recognized as an internal or external command, operable program or batch file.

**ChatGPT said:**

You're still running **Linux commands (sudo yum update -y)** in a **Windows Command Prompt or PowerShell**, which doesn't recognize them. That's why you're seeing:

kotlin

Copy code

'sudo' is not recognized as an internal or external command

**🧭 Let's clarify your environment:**

You are in this path:

vbnet

Copy code

C:\Users\VijayPrasanthBurgula\Docker-Zero-to-Hero\examples\first-docker-file>

This is clearly **Windows**, not Linux or a cloud instance.

**✅ What You Can Do Instead**

**🛠️ If you want to run Docker commands:**

You **do not need sudo or yum on Windows**.

Just make sure you:

1. Have **Docker Desktop installed**
2. Open **PowerShell** or **CMD**
3. Run Docker directly:

powershell

Copy code

docker build -t first-docker-image .

docker run first-docker-image

**🧪 To test if Docker is installed on your Windows:**

powershell

Copy code

docker --version