תמונה שמכילה טקסט, צילום מסך, מולטימדיה, גופן

התיאור נוצר באופן אוטומטיThis specifies the version of the Docker Compose syntax being used. The "3" corresponds to version 3 of the syntax.

תמונה שמכילה טקסט, צילום מסך, מולטימדיה, גאדג'ט

התיאור נוצר באופן אוטומטיThis section defines the different services (containers) you want to run.

תמונה שמכילה חשמל, טקסט, צילום מסך, מולטימדיה

התיאור נוצר באופן אוטומטיThis defines the "guardian" service. Here's what each attribute means:

- `image`: Specifies the Docker image to be used for the container.

- `container\_name`: Assigns a custom name to the container.

- `tty`: Allocates a pseudo-TTY, which allows you to interact with the container's terminal.

- `cap\_add`: Grants additional Linux capabilities to the container.

- `privileged`: Runs the container in privileged mode, giving it more access to the host system's resources.

- `volumes`: Mounts a directory from the host into the container.

- `network\_mode`: Sets the network mode to "host," which allows the container to use the host's networking stack.

תמונה שמכילה טקסט, גופן, מולטימדיה, תוכנה

התיאור נוצר באופן אוטומטיThis section defines the "sender" service, which is similar to the "guardian" service but with differences like the container name and the network configuration.

תמונה שמכילה טקסט, גופן, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטיThis section defines the "proxyA" service, following the same pattern as the previous services.

תמונה שמכילה טקסט, צילום מסך, מולטימדיה, גופן

התיאור נוצר באופן אוטומטיThis section defines the "proxyB" service, following the same pattern as the previous services.

תמונה שמכילה טקסט, גופן, מולטימדיה, צילום מסך

התיאור נוצר באופן אוטומטיThis section defines the "endUser" service, following the same pattern as the previous services.

תמונה שמכילה טקסט, צילום מסך, מולטימדיה, תוכנה

התיאור נוצר באופן אוטומטיThis section defines a custom network named "net-10.9.0.0" and configures its IP addressing. Here's what each attribute means:

- `name`: Assigns a custom name to the network.

- `ipam`: Specifies IP address management (IPAM) configuration for the network.

- `config`: Defines the IP subnet that containers on this network will use.

In summary, the YAML file defines multiple services (containers) that use the `handsonsecurity/seed-ubuntu:large` image. Each service has its own attributes such as the container name, networking configurations, volumes, and commands to run. Additionally, a custom network is defined with a specified IP subnet. The provided configurations set up a collection of services with distinct roles and networking settings.

The difference between docker imag and container:

**Docker Image:** An image in Docker is a lightweight, stand-alone, and executable software package that contains all the necessary code, runtime, libraries, and configurations to run a piece of software. Images are the building blocks of containers. They are created based on a Dockerfile, which contains instructions for building the image layer by layer. An image can include the operating system, application code, dependencies, and any configuration files needed to run your application.

Docker images are designed to be immutable, meaning they don't change once they're created. If you need to make changes to an image, you typically create a new image with the desired changes. Images are versioned using tags, such as **latest**, **v1.0**, or custom names, to help manage different versions of the same image.

Images are stored in a registry, which is a repository for Docker images. Docker Hub is a popular public registry, but you can also set up private registries to store your own images.

**Docker Container:** A container is a running instance of a Docker image. It's a lightweight, isolated, and executable environment that encapsulates your application and its dependencies. Containers are designed to be consistent across different environments, making it easier to develop, test, and deploy applications without worrying about differences in the underlying infrastructure.

Containers provide process isolation, filesystem isolation, and networking isolation. Each container runs as an independent process on the host system, with its own isolated filesystem and network stack. This isolation allows you to run multiple containers on the same host without conflicts between applications.

Containers are created from Docker images using the **docker run** command. When you run a container, an instance of the image's filesystem and configuration is created, and the application defined in the image's **CMD** instruction is executed. You can interact with containers, view their logs, and manage their lifecycle using Docker commands.

In summary, images are the blueprints that define the environment and software, while containers are the running instances of those images that allow you to execute and manage your applications in an isolated and consistent manner. Docker's image-container model enables efficient software packaging, distribution, and deployment, making it easier to develop and manage applications across different environments.