The Main Components of Docker Explained.  
  
Building and deploying applications has its complexities. Software inconsistencies across different environments lead to significant issues including deployment failures, increased development and testing complexity, and more.  
  
Docker solves the "it worked on my machine" problem, and streamlines application deployment by encapsulating applications and their dependencies into standardized, scalable, and isolated containers (containerization).  
  
Docker has a handful of core components powering the technology. By understanding them you will have a strong fundamental understanding — Let's dive in!  
  
𝗜𝗺𝗮𝗴𝗲𝘀:  
Read-only templates that are used to build containers. Images are created with Dockerfile instructions or can be downloaded from a Docker registry like Docker Hub.  
  
𝗖𝗼𝗻𝘁𝗮𝗶𝗻𝗲𝗿:  
An instance of an image. It's a lightweight, standalone package that includes everything needed to run an application.  
  
𝗗𝗼𝗰𝗸𝗲𝗿𝗳𝗶𝗹𝗲:  
A script-like file that defines the steps to create a Docker image.  
  
𝗗𝗼𝗰𝗸𝗲𝗿 𝗲𝗻𝗴𝗶𝗻𝗲:  
The Docker engine is responsible for running and managing containers. It's composed of the Docker daemon and the Docker CLI that communicates through REST API.  
  
𝗗𝗼𝗰𝗸𝗲𝗿 𝗱𝗮𝗲𝗺𝗼𝗻:  
The daemon is a persistent background service responsible for managing objects. It does so via listening for API requests. Docker objects include images, containers, networks, and storage volumes.  
  
𝗗𝗼𝗰𝗸𝗲𝗿 𝗿𝗲𝗴𝗶𝘀𝘁𝗿𝘆:  
Are repositories where Docker images are stored and can be distributed from. Docker registries can be public or private. Docker Hub is the default public registry that Docker is configured with.  
  
𝗗𝗼𝗰𝗸𝗲𝗿 𝗻𝗲𝘁𝘄𝗼𝗿𝗸:  
Containers run on networks allowing them to communicate with each other and the outside world. The network provides the communication gateway between containers running on the same or different hosts.  
  
𝗩𝗼𝗹𝘂𝗺𝗲𝘀:  
Allow data to persist outside of a container and to be shared between container instances, even after a container is deleted. Volumes decouple data life from the container lifecycle.  
  
The components listed above all tie together to produce a simple system for developers to automate the deployment, scaling, and management of applications. This has led Docker to become a powerful and integral tool in modern software development.  
  
——  
  
𝗔 𝗯𝗶𝗴 𝘁𝗵𝗮𝗻𝗸 𝘆𝗼𝘂 𝘁𝗼 𝗼𝘂𝗿 𝗽𝗮𝗿𝘁𝗻𝗲𝗿 𝗣𝗼𝘀𝘁𝗺𝗮𝗻 𝘄𝗵𝗼 𝗸𝗲𝗲𝗽𝘀 𝗼𝘂𝗿 𝗰𝗼𝗻𝘁𝗲𝗻𝘁 𝗳𝗿𝗲𝗲 𝘁𝗼 𝘁𝗵𝗲 𝗰𝗼𝗺𝗺𝘂𝗻𝗶𝘁𝘆.  
  
The power of AI has found its way to API management, with Postman leading the charge through their AI assistant, Postbot. Check it out: <https://lnkd.in/gRs7qtqQ>

Activate to view larger image,

