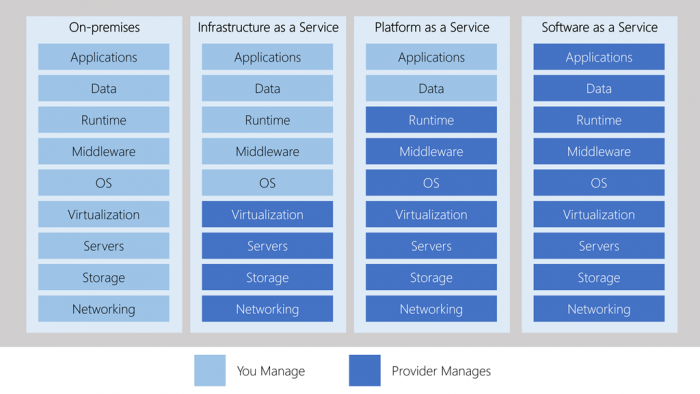
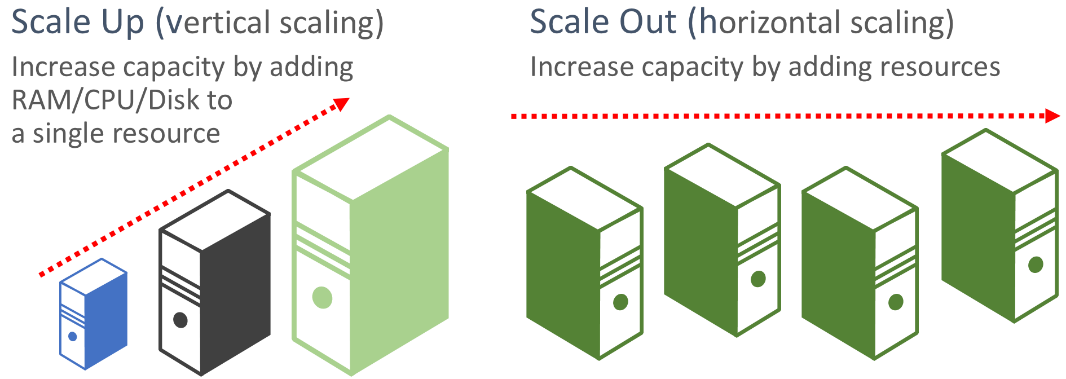
**Azure Containers**

**28.09.2023**

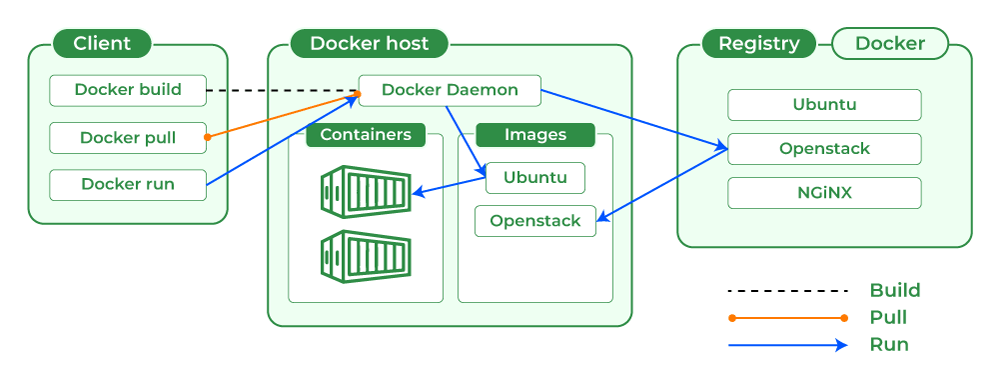
* **IaaS vs PaaS vs SaaS**
  + **Infrastructure as a Service:**
    - Everything is under customer’s control from OS to Storage and others. For e.g. VMs
  + **Platform as a Service:** 
    - The user only contains Data and the Applications. E.g. Azure SQL, Databricks, Dynamo DB
  + **Software as a Service:**
    - The user is responsible only for using the software. Microsoft Office 365, PowerBI, Dynamic 365



* **Introduction to Docker Containers**
  + Docker is a set of PaaS products that use the OS level virtualization to deliver software in packages called containers.
  + Isolated from one another and bundle their own software, libraries, and configuration files. They can communicate with each other using well defined separate channels.
  + All containers are run by a single OS kernel and therefore use fewer resources. (Dynamic Link Library).
  + Docker containers contains binary files, libraries, and configuration files along with the application itself.
  + They don’t contain a guest OS for each container and rely on the underlying OS kernel, which makes the containers lightweight.
  + They also share resources with other containers in the same host OS and provide OS level isolation.
* **Virtual Machines**
  + Run on hypervisors, which allows multiple VMs to run on single machine along with its own OS.
  + Each VM has its own copy of an OS along with application and necessary binaries, which makes them significantly larger.
* **Features of Docker**
  + Lightweight and fast environment to run code.
  + Open-Source Platform.
  + Easy Lightweight, consistent way of delivering applications.
  + Fast and efficient development lifecycle
  + Segregation of duties
  + Service oriented architecture
  + Security
  + Scalability
    - Scale up, Scale down – vertical Scaling.
    - Scale in, scale out – Horizontal Scaling



* + Reduction in size
  + Image & Volume Management
  + Networking
* **Architecture of Docker**
  + Client server architecture is used.



* + **Docker Daemon:**

The Docker Daemon is a background service that manages Docker containers on a system. It is responsible for building, running, and managing Docker containers. It listens for Docker API requests and communicates with the Docker Client to carry out container-related tasks.

* + **Docker Client:**

The Docker Client is a command-line tool or a graphical user interface (GUI) application that allows users to interact with Docker. It sends commands to the Docker Daemon, which then processes these commands to manage Docker containers, images, and other resources.

* + **Docker Host:**

The Docker Host is the physical or virtual machine on which Docker is installed and where containers run. It provides the necessary infrastructure for running Docker containers, including CPU, memory, and storage resources.

* + **Docker Registry:**

A Docker Registry is a service used to store and distribute Docker images. It acts as a centralized repository for Docker images, making it easier to share and distribute containerized applications. Docker Hub is a popular public Docker Registry, but organizations can also set up private registries for security and control.

* + **Docker Images:**

Docker Images are lightweight, stand-alone, executable packages that contain everything needed to run a piece of software, including the code, runtime, system tools, and libraries. Images serve as the blueprints for Docker containers and can be shared and reused to create containers.

* + **Docker Containers:**

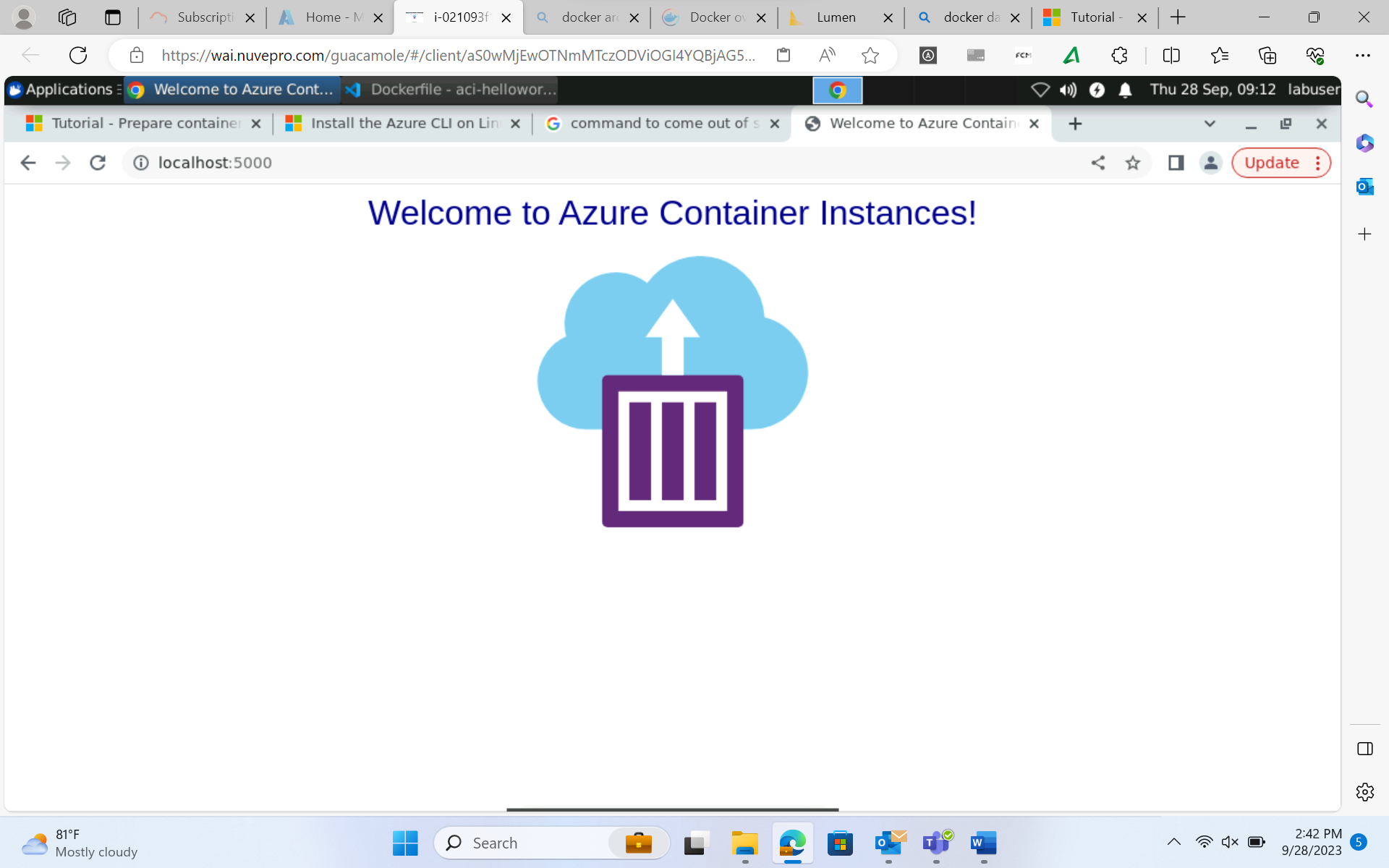
Docker Containers are instances of Docker images that are running as isolated processes on a Docker Host. They encapsulate an application and its dependencies, providing consistent and reproducible environments. Containers can be started, stopped, and managed using Docker commands.

* + **Docker Storage:**

Docker Storage refers to the storage options and mechanisms used by Docker to manage data within containers and images. Docker provides various storage drivers and storage types to accommodate different use cases, including local file systems, network-attached storage (NAS), and cloud-based storage solutions.

* **Azure Container Services:**
  + Azure Container Instances
    - Deploying Azure Container:

[Tutorial - Prepare container image for deployment - Azure Container Instances | Microsoft Learn](https://learn.microsoft.com/en-us/azure/container-instances/container-instances-tutorial-prepare-app)



* + Azure Container Apps
  + Azure Kubernetes Services
* **Introduction to Kubernetes Services**
  + **Features**
    - **Automatic Scaling**
      * Kubernetes provides an advanced scheduler to launch containers on cluster nodes. It performs resource optimization.
    - **Self-Healing Capabilities**
      * It provides rescheduling, replacing and restarting the containers which are dead.
    - **Automated Rollbacks and Rollouts**
    - **Horizontal Scaling & Load Balancing:**
      * It can scale up and scale down as per requirements.
    - **Resource Utilization**
      * Provides resource utilization monitoring and optimization, ensuring containers are using their resources efficiently.
    - **Support for multiple clouds & hybrid clouds**
    - **Extensibility:** 
      * It is very extensible and can be extended with custom plugins and controllers
    - **Community Support**
  + **Architecture of Kubernetes** 
    - **It follows client server architecture where we have the master installed on one machine and the node on the separate Linux machine.**
    - **It follows the master slave model which uses master to manage multiple containers across multiple nodes.**
    - **Master + worker nodes = cluster**

