Day 47

What is a container?

A container is a portable computing environment. It contains everything an application needs to run, from binaries to dependencies to configuration files.

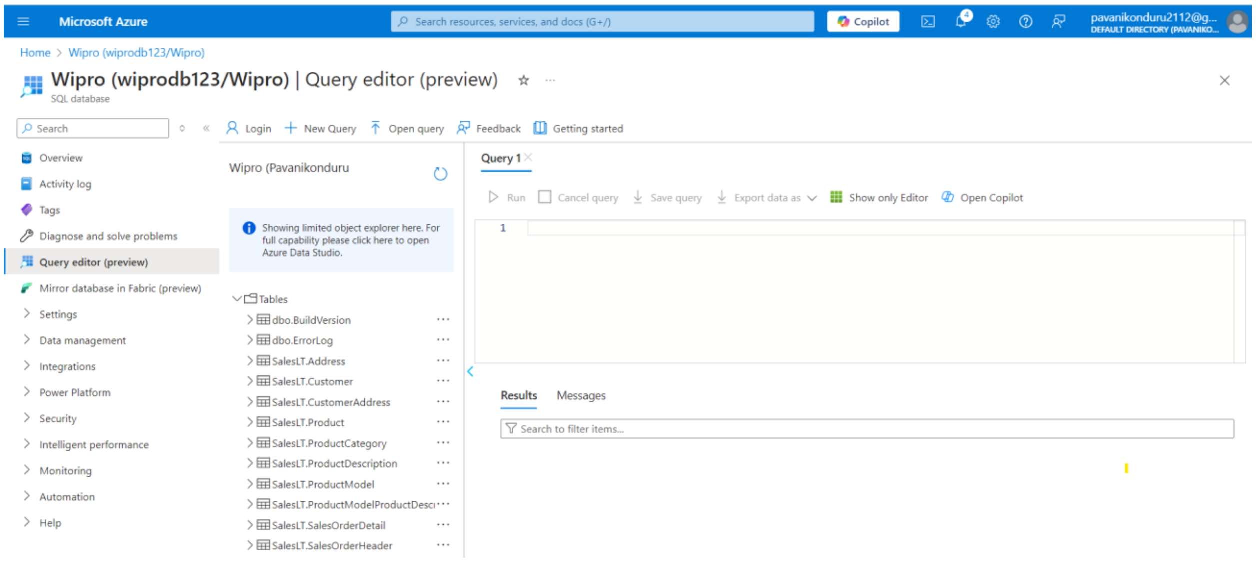
Containers operate on an abstracted layer above the underlying host operating system. Like virtual machines (VMs), they are isolated and have restricted access to system resources. Containers consume no virtual hardware, virtual kernel, or virtual operating system resources to run applications. So, containerization is a much leaner and more efficient method of virtualization.

Benefits of containerization

Every day, developers find new ways to put containerization to work to solve their challenges. Containerization can produce unique benefits for your applications. Here are some of the most common reasons developers decide to containerize:

- Portability
- Efficiency
- Agility
- Faster delivery
- Improved security
- Faster app startup
- Easier management
- Flexibility

Connecting to db. using login credentials:



Monolithic Deployment:

Key Characteristics

- **Single Codebase**: All components (UI, business logic, database access) are in one codebase.
- **Tightly Coupled**: Components are interconnected, making isolation challenging.
- **Single Deployment**: The entire application is deployed as one unit.
- **Simplified Development and Testing**: Easier to manage during initial development.

Advantages

- **Simplicity**: Easier to develop and manage, especially for smaller applications.
- **Performance**: Generally better performance due to reduced inter-service communication overhead.
- **Fewer Deployment Complexities**: Straightforward deployments without managing inter-service dependencies.

Disadvantages

- **Scalability Challenges**: Difficult to scale only parts of the application; scaling requires scaling the whole system.
- **Longer Deployment Times**: Redeploying the entire application can be time-consuming.

- **Difficult to Adopt New Technologies**: Upgrading parts may require extensive changes across the codebase.
- **Tightly Coupled Code**: Changes in one area can affect others, leading to increased bugs

Multi-service architecture using REST APIs:

1. **User Interaction:**
    - Users interact with a Web Server.
    - The server communicates with various services through REST APIs for different management domains.
2. **Supplier Management:**
    - Built with Microsoft .NET.
    - Uses MySQL database.
    - MySQL is described as easy to use, widely available, lightweight, and provides predictable load performance.
3. **Product Management:**
    - Built with Python.
    - Uses Cassandra as the database.
    - Cassandra offers high scalability, high availability, and high performance.
4. **Cost Management:**
    - Developed with Node.js.
    - Uses Couchbase as the database.
    - Couchbase provides low latency, simple administration, high availability, and a powerful query language.
5. **Price Management:**
    - Also built with Node.js.
    - Uses Couchbase as well.

    - Similar characteristics as the Cost Management module.

6. **Stock Management:**
    - Developed with Python.
    - Utilizes Redis as a cache layer and MongoDB for the main database schema.
    - This combination ensures super high performance with simplicity.
7. **Finance:**
    - Built with Java.
    - Uses MySQL as the database.
    - MySQL's ease of use, wide availability, and lightweight nature are emphasized, as well as predictable performance.

Each service module communicates through a REST API, allowing scalable, decoupled, and specialized microservices for different business functionalities.

Docker Images and Containers:

Docker Image:

- A **Docker Image** is a pre-configured environment that contains everything required to run an application, including the operating system, dependencies, and the application itself.
- It is a static blueprint for creating containers.
- **Example**: Ubuntu with ASP.NET Core and application code.

Docker Container:

- A **Docker Container** is a running instance of a Docker Image.
- It provides an isolated environment where your application runs.
- Containers can be started, stopped, or scaled independently from the image.

What is Kubernetes:

Kubernetes, often referred to as K8s (pronounced "kates"), is an open-source platform that brings order to this container chaos. It's like having a team of expert managers who take care of deploying, scaling, and managing all those containers, so you can focus on building amazing applications. It acts as a conductor, coordinating the complex interplay of containers, allowing them to work together harmoniously.

The Super Squad of Kubernetes:

1. Pods: The Atomic Units of Kubernetes

Pods are the fundamental units of deployment in Kubernetes. They are designed to encapsulate related application containers that are deployed together on a single worker node. Pods provide a cohesive environment for containers to interact and share resources.

2. Worker Nodes: The Muscle Behind the Scenes

Worker nodes, also known as minions, are the powerhouse of the Kubernetes cluster. These nodes are where the actual containers run and execute your application's code.

3. Master Node: The Strategic Mastermind

At the heart of every Kubernetes cluster resides the master node. Think of it as the strategic mastermind behind the scenes, overseeing and coordinating the entire cluster's activities.

4. Deployments: Orchestrating Application Lifecycles

Deployments are like the secret weapon in Kubernetes, empowering you to manage and control the lifecycle of your applications. They provide a declarative way to define how your application should be deployed and managed within the cluster.

5. Services: The Connectors of the Kubernetes Universe

Services play a crucial role in enabling communication and connectivity between different components. It abstracts the underlying IP addresses of pods, providing a consistent and reliable network interface. They act as the connectors, enabling load balancing and ensuring that other services or external clients can easily discover and interact with the pods.