Tutorial 4: Docker Compose and Docker Swarm

Question Set:

Q1. What are Microservices?

ANSWER:

Microservices and their related architectures are gaining traction in enterprises of all sizes. However, many IT decision makers are still in the dark as to what microservices actually are and what a microservices framework means to the development of IT solutions in today's enterprises. For DevOps, Microservices are also having a major impact. The agile development cycle is bringing together what were two separate IT entities, development and operations, into a commonly executed project ideology.

At a very high level, microservices can be conceptualized as a new way to create corporate applications, where applications are broken down into smaller, independent services, that are not dependent upon a specific coding language. In other words, development teams are able to use the language tools they are most comfortable with and still achieve synergy in the application development process. Using the ideology of microservices, large complex applications can be divvied up into smaller building blocks of executables, that when recomposed offer all of the functionality of a large scale, highly complex application.

At MuleSoft's Connect conference in 2016, Katharina Probst from Netflix and Uri Sarid, CTO of MuleSoft, made the point that the most successful businesses now, e.g. McDonalds, Under Armour, Amazon, Tesla, and Netflix, are not only competing on the goods and services they sell, but also on their ability to be digital platforms, providing innovative ways of getting those goods, services, and experiences to customers at scale and in real-time. They are able to do this by creating small services that compose themselves into processes, that evolve independently, can be built as quickly as they are needed, and are optimized for change and reuse.

Adopting microservices allows organizations to achieve greater agility and realize lower costs, thanks to the inherent granularity and reusability of what constitutes a microservice. A plethora of supporting technologies and ideologies have made the concept of microservice-based architectures a reality to businesses of any size, bringing agility and efficiencies to application development and deployment once deemed impossible.

Those concepts and technologies include the increased availability of containers, tools and processes that make SOA-based (Service Oriented Architecture) processes more attainable, as well as domain-driven design processes. Combining these creates a development environment that focuses more on collaboration than on code governance, resulting in improved productivity.

Q2. What is Docker Compose?

ANSWER:

Docker Compose is a tool for running multi-container applications on Docker defined using the Compose file format. A Compose file is used to define how the one or more containers that make up your application are configured. Once you have a Compose file, you can create and start your application with a single command: docker-compose up. Compose files can be used to deploy applications locally, or to the cloud on Amazon ECS or Microsoft ACI using the Docker CLI.

Q3. What is the fundamental role of the Raft consensus algorithm in a Docker Swarm cluster?
In Docker Swarm, the Raft consensus algorithm is used exclusively by the manager nodes to manage the cluster's state. The primary goal is to ensure that all manager nodes maintain an identical, consistent internal state of the entire swarm. This shared state is stored in a replicated log and includes all critical cluster information, such as:

- Node membership (which nodes have joined or left the swarm)
- The services running on the swarm
- The desired state of applications (e.g., number of replicas)
- Any configurations and secrets

By using Raft, the swarm guarantees that this state is replicated safely across a majority of managers. This consistency is the foundation of Swarm's fault tolerance, as it ensures that if the lead manager fails, another manager has an up-to-date copy of the state and can take over to continue managing the cluster.

Q4. How does Docker Swarm handle the failure of a leader manager node?
Docker Swarm is designed to be resilient to the failure of its leader manager node. If the leader fails or becomes unreachable, the remaining manager nodes automatically use the Raft consensus algorithm to elect a new leader.
The process works as follows:

1. The leader manager periodically sends out "heartbeat" signals to the other managers (followers) to assert its authority.
2. If the follower managers stop receiving these heartbeats for a certain period (an election timeout), they assume the leader has failed.
3. One or more followers will transition to a "candidate" state and start a new election. They request votes from the other available managers.
4. The first candidate to receive votes from a majority of the available managers (a quorum) is elected as the new leader.
5. Once elected, the new leader takes over all orchestration and cluster management responsibilities, ensuring that swarm operations can continue without interruption.

This automatic failover process ensures the swarm remains highly available and can continue to manage services even if the original leader goes down.

Q5. Why does Docker Swarm recommend an odd number of manager nodes, and how does this relate to fault tolerance?

Docker Swarm recommends an odd number of managers (typically three, five, or seven) to ensure high availability and prevent "split-brain" scenarios during network partitions. This recommendation is directly tied to how Raft's concept of a
quorum provides fault tolerance.
A quorum is the minimum number of managers that must be available and agree on any change to the cluster state. This is always a majority of the total managers ((N/2) + 1).

- Fault Tolerance: A swarm with N managers can tolerate the loss of (N-1)/2 managers.
    - With 3 managers, the quorum is 2. The cluster can tolerate the failure of 1 manager.
    - With 5 managers, the quorum is 3. The cluster can tolerate the failure of 2 managers.
    - With 7 managers, the quorum is 4. The cluster can tolerate the failure of 3 managers.
- Why an Odd Number is Better: Having an odd number of managers makes the cluster more resilient. If a network partition occurs that splits the managers into two groups, an odd number makes it more likely that one of the groups will still contain a majority (the quorum). The partition with the quorum can continue to operate and elect a leader, while the other partition without a quorum cannot make any changes. This prevents the two groups from making conflicting decisions, thus avoiding a split-brain scenario. Adding an even-numbered manager (e.g., going from 3 to 4 managers) does not increase the number of failures the cluster can tolerate; you can still only lose 1 manager in both cases.

Q6. Discuss the benefits of Docker Compose.

ANSWER:

Benefits of Docker Compose:

- Single host deployment - This means you can run everything on a single piece of hardware
- Quick and easy configuration - Due to YAML scripts
- High productivity - Docker Compose reduces the time it takes to perform tasks
- Security - All the containers are isolated from each other, reducing the threat landscape

Q7. Discuss the key features of Docker Swarm.

ANSWER:

Docker Swarm is an orchestration management tool that runs on Docker applications. It helps end-users in creating and deploying a cluster of Docker nodes. Each node of a Docker Swarm is a Docker daemon, and all Docker daemons interact using the Docker API. Each container within the Swarm can be deployed and accessed by nodes of the same cluster.

The key features of Docker Swarm are:

- Cluster management integrated with Docker Engine: Easy to set up. The Docker Engine CLI can simply be used to create a swarm of Docker Engines where you can deploy application services. There is no need for additional orchestration software to create or manage a swarm.

- Decentralized Design: Swarm makes it very easy for teams to access and manage the environment

- Scaling: For each service, you can declare the number of tasks you want to run. When you scale up or down, the swarm manager automatically adapts by adding or removing tasks to maintain the desired state.

- Load Balancing: Docker swarm also allows users to expose ports for services to an external load balancer. Internally, the swarm lets you specify how to distribute service containers between nodes.

- Secure by Default: It is also highly secure, of course, as it is a fundamental feature. Each node in the swarm enforces TLS mutual authentication and encryption to secure communications between itself and all other nodes.

- Rolling Updates: The swarm manager also lets you control the delay between service deployment to different sets of nodes. If anything goes wrong, you can roll back to a previous version of the service.

References

[1] what-are-microservices https://www.mulesoft.com/resources/api/what-are-microservices
[2] Docker Compose https://www.simplilearn.com/tutorials/docker-tutorial/docker-compose