docker intervirw questions:

**what is the difference b/w docker image and docker container**

Docker Image:  
Definition: A Docker image is a lightweight, standalone, and executable package   
that includes everything needed to run a piece of software, including the code, runtime, libraries, and system tools.   
It is a snapshot of a filesystem and a set of parameters that define how to run a container based on that filesystem.  
what tool in multicontainer deployment

Docker Container:  
Definition: A Docker container is a runnable instance of a Docker image.   
It is an encapsulated, isolated environment that runs applications and services.   
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  
how do yo achieve multicontainer deployment

------------------------  
docker compose  
+++++++++++++++++++++++++++++++++++++++++++  
what is the docker file   
---------------------------------------------------------------------  
Base Image Layer:  
The first layer is the base image layer, specified with the FROM instruction.   
This layer provides the starting point for your image and includes the operating system and basic dependencies.

Layer for Each RUN Instruction:  
Each RUN instruction in the Dockerfile creates a new layer.   
This layer captures the changes made to the file system during the execution of the specified command.

Layer for Each COPY or ADD Instruction:  
The COPY and ADD instructions copy files or directories into the image.   
Each instruction creates a new layer that captures the added files.

Layer for Each WORKDIR Instruction:  
The WORKDIR instruction sets the working directory for subsequent instructions.   
It creates a new layer indicating the change in the working directory.

Layer for Each EXPOSE Instruction:  
The EXPOSE instruction informs Docker that the container listens on specified network ports at runtime.   
It does not actually publish the ports but documents them.

Layer for Each CMD or ENTRYPOINT Instruction:  
The CMD and ENTRYPOINT instructions specify the default command to run when the container starts.   
Each instruction creates a new layer.  
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  
what is the difference between CMD and ENTRYPOINT  
-----------------------------------------------  
CMD Instruction:  
Purpose: Specifies the default command and/or parameters to be executed when a container is run.   
It is often used to define the main application that runs in the container.  
Usage: You can specify the CMD instruction with or without parameters.   
If the Dockerfile contains multiple CMD instructions, only the last one takes effect.

ENTRYPOINT Instruction:  
Purpose: Configures a container to run as an executable.   
It provides a way to set a fixed command and parameters that are not easily overridden when the container is run.   
It is often used for defining the main executable for the container.  
Usage: Similar to CMD, you can specify the ENTRYPOINT instruction with or without parameters.   
If the Dockerfile contains multiple ENTRYPOINT instructions, only the last one takes effect.  
+++++++++++++++++++++++++++++++++++++++++++++++++++++++  
what is the command to run the dockercompose file  
--------------------------------------------------

docker build -t <image\_name>:<tag> <path\_to\_Dockerfile\_directory>

++++++++++++++++++++++++++++++++++++++++++++++++++  
docker-compose -f docker-compose.prod.yml up  
+++++++++++++++++++++++++++++++++++++++++++++++++++  
what is the difference between docker compose file and docker file  
----------------------------------------------  
Dockerfile:  
Purpose: The Dockerfile is used to define the structure and configuration of an individual Docker image.   
It contains a set of instructions for building a Docker image,   
specifying the base image, installing dependencies, setting up configurations,   
and defining the entry point for the container.

Docker Compose file:  
Purpose: The Docker Compose file is used to define and manage multi-container Docker applications.   
It allows you to define services, networks, and volumes for multiple containers and specify their configurations.   
Docker Compose simplifies the process of managing complex applications with multiple interconnected services.  
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

how to interact with the running docker container  
----------------------------------------------------  
docker exec -it [container\_name or container\_id] /bin/bash

++++++++++++++++++++++++++++++++++++++++++++++++++++++++  
after successfully executing the docker container (docker run) the container is terminating automatically what might be the reason?  
-------------------------------------------------------------------------  
container will be terminated when there is no process running in the container make use of CMD or ENTRYPOINT to set up a background brocess  
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  
How do you maintain relese in your previous projects  
-------------------------------------------------  
maintaining releases typically involved a structured process to ensure the stability, quality, and controlled deployment of software.   
Here's a general outline of how release management was handled:  
Version Control:Utilized a version control system (e.g., Git) to manage source code. Each release was associated with a specific tag or branch, allowing for easy reference and rollback.  
Semantic Versioning:Followed semantic versioning (SemVer) conventions for version numbers. This helped communicate the nature of changes in each release (major, minor, patch) and ensure compatibility.  
Release Planning:Conducted release planning to define the scope of each release, including new features, enhancements, bug fixes, and any breaking changes.  
Feature Branches:Used feature branches in Git for developing new features or making significant changes. Pull requests were used for code reviews before merging into the main branch.  
Continuous Integration (CI):Implemented a CI/CD pipeline to automate the build, test, and deployment processes. This helped catch issues early and ensured that code was always in a deployable state.  
Pre-Release Testing:Conducted thorough testing before each release, including unit tests, integration tests, and, if applicable, user acceptance testing (UAT). Automated testing was prioritized to speed up the testing process.

+++++++++++++++++++++++++++++++++  
how do you check  logs in docker   
---------------------------------  
docker logs [container\_name or container\_id]

+++++++++++++++++++++++++++++++++++++++++  
what aws codebuild how it is different from other build tools  
------------------------------------------------  
Fully Managed Service:  
Serverless Architecture:  
Integration with AWS Ecosystem  
Wide Range of Build Environment Options:  
Buildspec.yml Configuration:  
Artifact Management:  
Parallel and Concurrent Builds:  
Flexible Source Control Integration  
Built-in Security:  
++++++++++++++++++++++++++++++++++++++++++++++++++  
explain buildspec.yml and significance in aws code build  
-------------------------------------  
The buildspec.yml file is a configuration file used in AWS CodeBuild to define the build steps, commands, and settings for your build process.   
It serves as a blueprint for AWS CodeBuild to execute your build, test, and deployment tasks.   
The buildspec.yml file is a key component of your project in CodeBuild,   
and its structure and content determine how CodeBuild performs the build.  
++++++++++++++++++++++++++++++++++++++++++++  
How can integrate aws code build with version control system like git hub  
-------------------------------------------  
Create a CodeBuild Project:

Open the AWS CodeBuild console.  
Choose "Create build project."  
Configure your build project settings, including the project name, source provider (GitHub), and other details.

+++++++++++++++++++++++++++++++++++++++  
what are artifacts in the context of aws codebuild  
----------------------------------------  
artifacts refer to the files and data produced as a result of the build process.   
These artifacts can include compiled code, binaries, application packages, and any other output generated during the build.  
AWS CodeBuild allows you to define and specify these artifacts so that they can be further used in   
subsequent stages of your CI/CD (Continuous Integration/Continuous Deployment) pipeline.

+++++++++++++++++++++++++++++++++++++++++++  
how do you secure sensitive information such as api keys or credentials in aws code build  
--------------------------------  
Parameter Store or Secrets Manager:  
Environment Variables:  
IAM Roles and Policies:  
KMS Encryption:  
Secure Build Artifacts:  
Use Temporary Credentials:

++++++++++++++++++++++++++++++++++++++++++++  
share  the experience troubleshooting common issues in aws codebuild  
-------------------------------  
Build Failure:  
Timeouts  
Permissions Issues:  
Network Issues:  
Artifact Management:  
Custom Docker Images:  
Environment Configuration:

++++++++++++++++++++++++++++++++++++++  
what is the artiact promotion in aws code pipeline workflow  
---------------------------------------------------  
  artifacts are generated in the build stage and then promoted to the test stage for testing. If the tests pass, the artifacts are further promoted to the deploy stage for deployment to production.

This promotion process ensures a controlled and automated flow of artifacts through different stages of the software delivery lifecycle. It helps maintain consistency and reliability in the deployment process.

AWS CodePipeline provides a vi