Free Questions for DCA by dumpssheet

Shared by Witt on 12-12-2023

Question 1

Question Type: MultipleChoice

Is this an advantage of multi-stage builds?

Solution: better caching when building Docker images

Options:

A- Yes

B- No

Answer:

B

Explanation:

Better caching when building Docker images is not an advantage of multi-stage builds. Multi-stage builds are a feature that allows you to use multiple FROM statements in a single Dockerfile. Each FROM statement begins a new stage of the build, with its own base image and instructions. You can selectively copy artifacts from one stage to another, leaving behind everything you don't want in the final image. The advantages of multi-stage builds are:

Reducing the size of the final image by removing unnecessary dependencies or intermediate files.

Improving the security of the final image by minimizing the attack surface and avoiding leaking secrets.

Simplifying the development workflow by using different tools or environments in different stages. Better caching when building Docker images is not an advantage of multi-stage builds, as it depends on other factors, such as the order and content of the instructions in each stage, the availability and freshness of the base images and intermediate layers, and the use of build arguments or environment variables that may invalidate the cache. Reference: https://docs.docker.com/develop/develop-images/multistage-build/, https://docs.docker.com/develop/develop-images/dockerfile_best-practices/#leverage-build-cache

Question 2

Question Type: MultipleChoice

Will this Linux kernel facility limit a Docker container's access to host resources, such as CPU or memory?

Solution. capabilities

Options:

A- Yes

B- No

Answer:

B

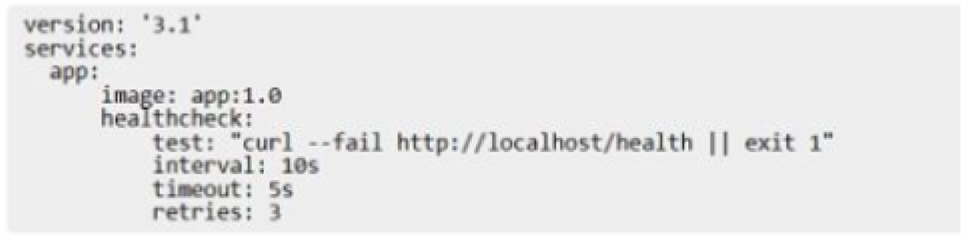
Explanation:

Capabilities are not a Linux kernel facility that limit a Docker container's access to host resources, such as CPU or memory. Capabilities are a Linux kernel feature that divide the privileges of the root user into distinct units, called capabilities, which can be independently enabled or disabled for each process. Capabilities allow fine-grained control over the operations that a process can perform on the system, such as binding to a privileged port, changing the system time, loading kernel modules, etc. Docker uses capabilities to restrict the default set of capabilities available to the root user inside a container, following the principle of least privilege. However, capabilities do not affect how much CPU or memory a container can use on the host system. Reference:
https://docs.docker.com/engine/reference/run/#runtime-privilege-and-linux-capabilities, https://man7.org/linux/man-pages/man7/capabilities.7.html

Question 3

Question Type: MultipleChoice

The following Docker Compose file is deployed as a stack:



```
version: '3.1'
services:
  app:
      image: app:1.0
      healthcheck:
          test: "curl --fail http://localhost/health || exit 1"
          interval: 10s
          timeout: 5s
          retries: 3
```

Is this statement correct about this health check definition?

Solution. Health checks lest for app health ten seconds apart. Three failed health checks transition the container into "unhealthy" status.

Options:

A- Yes

B- No

Answer:

A

Explanation:

This statement is correct about this health check definition. A health check is a feature that allows Docker to monitor the health status of a container by running a command in the container periodically. A health check has three parameters: test, interval, and retries. The test parameter specifies the command to run to check the health of the container. The interval parameter specifies how often to run the health check. The retries parameter specifies how many consecutive failures of the health check are needed to mark the container as unhealthy. In this case, the health check runs curl -f http://localhost/ || exit 1 every 10 seconds to test for app health. If this command fails three times in a row, the container is marked as unhealthy. Reference: https://docs.docker.com/engine/reference/builder/#healthcheck, https://docs.docker.com/engine/reference/run/#healthcheck

Question 4

Question Type: MultipleChoice

Is this a type of Linux kernel namespace that provides container isolation?

Solution. Process ID

Options:

A- Yes

B- No

Answer:

A

Explanation:

Process ID is a type of Linux kernel namespace that provides container isolation. Process ID (pid) namespace isolates the process ID number space, which means that processes in different pid namespaces can have the same PID. This allows each container to have its own init process (PID 1), which is the first process to start in a container and the ancestor of all other processes in the container. Pid namespace also prevents processes in one container from seeing or signaling processes in another container or on the host system, unless they share the same pid namespace or have the CAP_SYS_PTRACE capability. Reference: https://docs.docker.com/engine/reference/run/#pid-settings---pid, https://en.wikipedia.org/wiki/Linux_namespaces#Process_ID_(pid)

Question 5

Question Type: **MultipleChoice**

Will This command list all nodes in a swarm cluster from the command line?

Solution. 'docker swarm nodes'

Options:

A- Yes

B- No

Answer:

B

Explanation:

This command does not list all nodes in a swarm cluster from the command line. The docker swarm command manages swarm operations, such as initializing or joining a swarm, updating the swarm configuration, etc. It does not show information about nodes or services. To list all nodes in a swarm cluster from the command line, you need to use docker node ls command. This command shows information about all the nodes that are part of the swarm, such as their ID, hostname, status, availability, etc. Reference: https://docs.docker.com/engine/reference/commandline/swarm/, https://docs.docker.com/engine/reference/commandline/node_ls/

Question 6

Question Type: **MultipleChoice**

Will this command list all nodes in a swarm cluster from the command line?

Solution. 'docker inspect nodes

Options:

A- Yes

B- No

Answer:

B

Explanation:

This command does not list all nodes in a swarm cluster from the command line. The docker inspect command shows low-level information about one or more objects, such as containers, images, networks, etc. It does not show information about nodes or services. To list all nodes in a swarm cluster from the command line, you need to use docker node ls command. This command shows information about all the nodes that are part of the swarm, such as their ID, hostname, status, availability, etc. Reference: https://docs.docker.com/engine/reference/commandline/inspect/, https://docs.docker.com/engine/reference/commandline/node_ls/

Question 7

Question Type: **MultipleChoice**

Will this command mount the host's '/data* directory to the ubuntu container in read-only mode?

Solution. 'docker run -add-volume /data /mydata -read-only ubuntu'

Options:

A- Yes

B- No

Answer:

B

Explanation:

This command does not mount the host's /data directory to the ubuntu container in read-only mode. The --add-volume and --read-only flags do not exist and will cause an error. To mount a host directory or a named volume to a container, you need to use the -v or --volume flag. To mount the host's /data directory to the ubuntu container in read-only mode, you need to use -v /data:/mydata:ro instead. Reference: https://docs.docker.com/storage/bind-mounts/, https://docs.docker.com/engine/reference/run/#volume-shared-filesystems

Question 8

Question Type: **MultipleChoice**

Will this command mount the host's '/data1 directory to the ubuntu container in read-only mode?

Solution. 'docker run -v /data:/mydata -mode readonly ubuntu'

Options:

A- Yes

B- No

Answer:

B

Explanation:

This command does not mount the host's /data directory to the ubuntu container in read-only mode. The -v or --volume flag mounts a host directory or a named volume to a container. The syntax for mounting a host directory is -v <host-path>:<container-

path>[:<options>]. The options can be ro for read-only mode, rw for read-write mode, z or Z for SELinux labels, etc. In this command, -mode readonly is not a valid option and will cause an error. To mount the host's /data directory to the ubuntu container in read-only mode, you need to use -v /data:/mydata:ro instead. Reference: https://docs.docker.com/storage/bind-mounts/, https://docs.docker.com/engine/reference/run/#volume-shared-filesystems

Question 9

Question Type: MultipleChoice

Is this a Linux kernel namespace that is disabled by default and must be enabled at Docker engine runtime to be used?

Solution. pid

Options:

A- Yes

B- No

Answer:

B

Explanation:

pid is not a Linux kernel namespace that is disabled by default and must be enabled at Docker engine runtime to be used. pid is a Linux kernel namespace that provides process isolation for containers. It ensures that processes in one container cannot see or signal processes in another container or on the host system. pid is enabled by default for Docker containers and does not require any special flag or option to be used. However, you can disable pid isolation for a container by using --pid host option when creating or running a container. This option connects the container to the host's pid namespace and allows the container to see and signal processes on the host system. Reference: https://docs.docker.com/engine/reference/run/#pid-settings---pid, https://en.wikipedia.org/wiki/Linux_namespaces#Process_ID_(pid)