# Free Questions for DCA by dumpshq

## Shared by Pate on 29-01-2024

**For More Free Questions and Preparation Resources**

**Check the Links on Last Page**

# Question 1

**Question Type:** MultipleChoice

A company's security policy specifies that development and production containers must run on separate nodes in a given Swarm cluster. Can this be used to schedule containers to meet the security policy requirements?

Solution. label constraints

## Options:

**A-** Yes

**B-** No

## Answer:

A

## Explanation:

Label constraints can be used to schedule containers to meet the security policy requirements. Label constraints are expressions that match node labels to service labels. Node labels are key-value pairs that can be attached to nodes to identify them by certain characteristics, such as role, environment, region, etc. Service labels are key-value pairs that can be attached to services to specify

scheduling preferences or requirements, such as node.role == manager or environment != production. Label constraints can be used with the --constraint flag when creating or updating a service to ensure that the service's containers are scheduled on nodes that match the specified criteria. Reference: https://docs.docker.com/engine/swarm/services/#specify-service-constraints, https://docs.docker.com/engine/swarm/manage-nodes/#add-or-remove-label-metadata

# Question 2

**Question Type:** **MultipleChoice**

You created a new service named 'http* and discover it is not registering as healthy. Will this command enable you to view the list of historical tasks for this service?

Solution. 'docker inspect http"

## Options:

**A-** Yes

**B-** No

## Answer:

B

### Explanation:

Using docker inspect http does not enable you to view the list of historical tasks for this service. 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 services or tasks. To view the list of historical tasks for this service, you need to use docker service ps http --no-trunc. Reference: https://docs.docker.com/engine/reference/commandline/inspect/, https://docs.docker.com/engine/reference/commandline/service_ps/

## Question 3

**Question Type:** MultipleChoice

Two development teams in your organization use Kubernetes and want to deploy their applications while ensuring that Kubernetes-specific resources, such as secrets, are grouped together for each application.

Is this a way to accomplish this?

Solution. Create a collection for for each application.

### Options:

**A-** Yes

**B-** No

### Answer:

B

### Explanation:

Creating a collection for each application is not a way to deploy applications using Kubernetes and group Kubernetes-specific resources for each application. A collection is a concept in Universal Control Plane (UCP) that defines a set of resources that users or teams can access. A collection can contain swarm services, Kubernetes namespaces, or volumes. A collection does not deploy applications or group resources; it only controls access to them. To deploy applications using Kubernetes and group Kubernetes-specific resources for each application, you can use namespaces or labels. A namespace is a way of isolating and organizing objects in a cluster. A label is a key-value pair that can be attached to any object for grouping or selecting purposes. Reference: https://docs.docker.com/ee/ucp/authorization/concepts/, https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/, https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/

## Question 4

**Question Type:** MultipleChoice

Will this sequence of steps completely delete an image from disk in the Docker Trusted Registry?

Solution. Delete the image and delete the image repository from Docker Trusted Registry.

## Options:

**A-** Yes

**B-** No

## Answer:

B

## Explanation:

Deleting the image and deleting the image repository from Docker Trusted Registry (DTR) does not completely delete an image from disk in DTR. Deleting an image only removes its tag and association with a repository, but does not delete its underlying layers from disk. Deleting a repository only removes its metadata and tags, but does not delete its underlying layers from disk either. To completely delete an image from disk in DTR, you need to run a garbage collection job after deleting the image or the repository. A garbage collection job scans the DTR storage and removes any unused layers that are not referenced by any images or repositories. Reference: https://docs.docker.com/ee/dtr/user/manage-images/delete-images/, https://docs.docker.com/ee/dtr/admin/configure/garbage-collection/

# Question 5

**Question Type:** MultipleChoice

An application image runs in multiple environments, with each environment using different certificates and ports. Is this a way to provision configuration to containers at runtime?

Solution. Create a Dockerfile for each environment, specifying ports and Docker secrets for certificates.

## Options:

**A-** Yes

**B-** No

## Answer:

B

## Explanation:

Creating a Dockerfile for each environment, specifying ports and Docker secrets for certificates, is not a way to provision configuration to containers at runtime. A Dockerfile defines the configuration of an image at build time, not at runtime. Creating a different Dockerfile for each environment is not a good practice, as it introduces duplication and inconsistency. To provision configuration to containers at

runtime, you can use environment variables, config objects, or command-line arguments. Reference: https://docs.docker.com/engine/reference/builder/, https://docs.docker.com/engine/swarm/configs/, https://docs.docker.com/engine/reference/commandline/run/#set-environment-variables--e---env---env-file

# Question 6

**Question Type:** MultipleChoice

Will this command ensure that overlay traffic between service tasks is encrypted?

Solution. docker network create -d overlay --secure

## Options:

**A-** Yes

**B-** No

## Answer:

B

### Explanation:

Using docker network create -d overlay --secure <network-name> does not ensure that overlay traffic between service tasks is encrypted. The --secure flag is not a valid option for this command and will cause an error. To ensure that overlay traffic between service tasks is encrypted, you need to use --opt encrypted flag instead. This flag enables IPsec encryption at the level of the vxlan overlay driver. Reference: https://docs.docker.com/engine/reference/commandline/network_create/#options, https://docs.docker.com/network/overlay/#encryption-and-overlay-networks

## Question 7

**Question Type:** MultipleChoice

Is this statement correct?

Solution. A Dockerfile stores persistent data between deployments of a container

### Options:

**A-** Yes

**B-** No

## Answer:

B

## Explanation:

A Dockerfile does not store persistent data between deployments of a container. A Dockerfile is a text document that contains instructions for building an image from a base image and other components. A Dockerfile does not store any data itself; it only defines how an image should be built. Persistent data between deployments of a container can be stored using volumes or bind mounts, which are ways of attaching external storage to containers. Reference: https://docs.docker.com/engine/reference/builder/, https://docs.docker.com/storage/

# Question 8

**Question Type:** MultipleChoice

Is this a supported user authentication method for Universal Control Plane?

Solution. x.500

### Options:

**A-** Yes

**B-** No

### Answer:

B

### Explanation:

x.500 is not a supported user authentication method for Universal Control Plane (UCP). x.500 is a series of standards for directory services that define how distributed directory information can be accessed and managed over a network. x.500 is not an external authentication backend that UCP supports. UCP supports LDAP, Active Directory, and SAML as external authentication backends.
Reference: https://docs.docker.com/ee/ucp/admin/configure/external-auth/, https://en.wikipedia.org/wiki/X.500

# Question 9

**Question Type:** MultipleChoice

Is this a supported user authentication method for Universal Control Plane?

Solution. SAML

### Options:

**A-** Yes

**B-** No

### Answer:

A

### Explanation:

SAML is a supported user authentication method for Universal Control Plane (UCP). SAML (Security Assertion Markup Language) is an open standard for exchanging authentication and authorization data between parties, such as an identity provider and a service provider. UCP supports SAML as an external authentication backend, which allows users to log in to UCP using their existing credentials from a SAML identity provider, such as Okta, Ping Identity, OneLogin, etc. UCP also supports other external authentication backends, such as LDAP and Active Directory. Reference: https://docs.docker.com/ee/ucp/admin/configure/external-auth/, https://docs.docker.com/ee/ucp/admin/configure/saml/

## Question 10

**Question Type:** MultipleChoice

In the context of a swarm mode cluster, does this describe a node?

Solution. an instance of the Docker CLI connected to the swarm

## Options:

**A-** Yes

**B-** No

## Answer:

B

## Explanation:

An instance of the Docker CLI connected to the swarm does not describe a node in the context of a swarm mode cluster. A node is a physical or virtual machine that runs the Docker Engine and participates in the swarm. A node can have one of two roles: manager or worker. Manager nodes maintain the cluster state and orchestrate tasks. Worker nodes execute tasks assigned by manager nodes. An instance of the Docker CLI connected to the swarm is a client that can interact with the swarm using commands such as docker service, docker node, docker stack, etc. A client can connect to any manager node in the swarm using the --host or -H flag. Reference: https://docs.docker.com/engine/swarm/key-concepts/#nodes-and-services, https://docs.docker.com/engine/swarm/swarm-tutorial/#use-docker-for-mac-or-docker-for-windows

# Question 11

**Question Type:** MultipleChoice

Is this the purpose of Docker Content Trust?

Solution. Indicate an image on Docker Hub is an official image.

## Options:

**A-** Yes

**B-** No

## Answer:

B

## Explanation:

Indicating an image on Docker Hub is an official image is not the purpose of Docker Content Trust. Official images are curated images that are provided and maintained by Docker, Inc. or by upstream software maintainers. Official images have a special badge on Docker Hub that indicates their status. Official images are not necessarily signed by Docker Content Trust, although some of them are. The purpose of Docker Content Trust is to sign and verify image tags, not to indicate official images. Reference: https://docs.docker.com/docker-hub/official_images/, https://docs.docker.com/engine/security/trust/

# Question 12

**Question Type:** **MultipleChoice**

Is this the purpose of Docker Content Trust?

Solution. Sign and verify image tags.

## Options:

**A-** Yes

**B-** No

## Answer:

A

## Explanation:

Signing and verifying image tags is the purpose of Docker Content Trust. Docker Content Trust (DCT) is a feature that allows you to use digital signatures for data sent to and received from remote Docker registries. These signatures allow client-side or runtime verification of the integrity and publisher of specific image tags. With DCT, image publishers can sign their images and image consumers can ensure that the images they pull are signed. Reference: https://docs.docker.com/engine/security/trust/