EXAMTOPICS

-Expert Verified, Online, **Free**.



CERTIFICATION TEST

-CertificationTest.net - Cheap & Quality Resources With Best Support

Question #1 *Topic 1*

What is the difference between the ADD and COPY Dockerfile instructions? (Choose two.)

A. ADD supports regular expression handling while COPY does not.

B. COPY supports compression format handling while ADD does not.

C. ADD supports compression format handling while COPY does not.

D. COPY supports regular expression handling while ADD does not.

E. ADD supports remote URL handling while COPY does not.

Suggested Answer: *C, E*

Official Answer: CE

Community vote distribution

CE (100%)

doudou123 Most Recent 10 months, 1 week ago

copy add

upvoted 1 times

Oxybisx 1 year, 2 months ago

Does anyone know if these questions are still valid in June 2025?

upvoted 1 times

forgaoqiang 1 year, 11 months ago

that's right, ADD support compressed and remote URL

upvoted 1 times

sandro781623 2 years, 9 months ago

- ADD - supports compression format handling - supports remote URL handling

upvoted 4 times

Question #2 *Topic 1*

When an application being managed by UCP fails, you would like a summary of all requests made to the UCP API in the hours leading up to the failure.
What must be configured correctly beforehand for this to be possible?

A. UCP audit logs must be set to the 'metadata' or 'request' level.

B. All engines in the cluster must have their log driver set to the 'metadata' or 'request' level.

C. UCP logging levels must be set to the 'info' or 'debug' level.

D. Set the logging level in the config object for the ucp-kube-api-server container to warning or higher.

Suggested Answer: *A*
Official Answer: A

Community vote distribution
A (78%) C (22%)

badalchavan11 Highly Voted 1 year, 6 months ago
The correct answer is: A. UCP audit logs must be set to the 'metadata' or 'request' level. Explanation: To capture a summary of all requests made to the UCP (Universal Control Plane) API leading up to a failure, UCP audit logs must be enabled at the appropriate level. The metadata level records request metadata, while the request level provides detailed information about each API request. These logs enable you to trace API activity and diagnose the issue. B. Incorrect: Log driver settings for the engines in the cluster are unrelated to UCP audit logs, as they are for container logs, not API logs. C. Incorrect: UCP logging levels (e.g., info or debug) control the verbosity of general logs but do not provide detailed API request traces. D. Incorrect: Changing the logging level for the ucp-kube-api-server affects Kubernetes API logs, not the UCP API logs.
upvoted 5 times

Raji82 Most Recent 1 year, 6 months ago
Should be C 'metadata' and 'request' are not log levels
upvoted 1 times

__rajan__ 1 year, 7 months ago
UCP (Universal Control Plane) audit logging is a feature that records all API requests made to the UCP controllers. To be able to review API requests made before a failure: You need audit logging enabled The logging level must be set to either 'metadata' or 'request' 'Request' level provides the most detailed information
upvoted 2 times

LavaPup 1 year, 8 months ago
Summary of UCP Logging Levels: debug: Very detailed logs for troubleshooting. info: General information, default for most environments. warn: Warnings about potential issues. error: Errors that affect functionality. fatal: Critical failures preventing operation. as per ChatGPT
upvoted 1 times

Question #3 *Topic 1*

A user's attempts to set the system time from inside a Docker container are unsuccessful.
Could this be blocking this operation?
SELinux

A. Yes

B. No

Suggested Answer: *B*
Official Answer: B

Community vote distribution
B (73%) A (27%)

ogogundare Most Recent 4 months, 3 weeks ago
Explanation: SELinux (Security-Enhanced Linux) can block attempts to change the system time from within a Docker container. Why Changing system time requires the CAP_SYS_TIME capability Even if the container has this capability: SELinux policies may still deny the action Containers run under restricted SELinux contexts by default Key Point: SELinux enforces mandatory access control It can block privileged operations like setting system time
upvoted 1 times

Jay87J 1 year, 5 months ago
• Setting system time requires privileged access because system time is a kernel-level setting. • Docker containers run with limited permissions by default, so a non-privileged container cannot change the system time. • SELinux does not directly block this operation; rather, it is restricted due to the container's unprivileged nature. • To change the system time inside a container, you would need to run it with --privileged mode, which grants elevated permissions.
upvoted 2 times

__rajan__ 1 year, 7 months ago
Setting system time inside a Docker container is controlled by Linux capabilities, not SELinux. Specifically, it requires the CAP_SYS_TIME capability. By default, Docker drops most capabilities, including CAP_SYS_TIME, for security reasons. This means that containers cannot modify the system time unless explicitly granted this capability. While SELinux can restrict various container operations, the ability to set system time is specifically controlled by Linux capabilities, not SELinux policies.
upvoted 3 times

AmenFlux 1 year, 8 months ago
SELinux (Security-Enhanced Linux) is a security mechanism that enforces access control policies. However, SELinux is not responsible for blocking the ability to set the system time from inside a Docker container. The actual reason for this restriction lies in Docker's default isolation mechanisms and Linux capabilities. * Docker's Security Model: • Docker containers operate in a sandboxed environment and lack access to host-level kernel features, including system-wide time changes. • This restriction is in place to prevent containers from interfering with the host system or other containers.
upvoted 3 times

bsalama 1 year, 10 months ago
SELinux (Security-Enhanced Linux) could potentially block a user's attempts to set the system time from inside a Docker container. SELinux is a security feature that provides mandatory access control policies, and it can restrict certain system operations, including actions taken within a container.
upvoted 2 times

Question #4 *Topic 1*

One of several containers in a pod is marked as unhealthy after failing its livenessProbe many times.
Is this the action taken by the orchestrator to fix the unhealthy container?
The unhealthy container is restarted.

A. Yes

B. No

Suggested Answer: *A*
Official Answer: A

Community vote distribution
A (100%)

doudou123 Most Recent 10 months, 1 week ago
Un pod peut contenir plusieurs conteneurs. Chaque conteneur peut avoir une livenessProbe, qui est une vérification périodique pour s'assurer que le conteneur est toujours vivant (ex. : une requête HTTP, une commande shell, etc.). Si cette probe échoue plusieurs fois (selon les paramètres failureThreshold, periodSeconds, etc.), le conteneur est considéré comme unhealthy. Action de l'orchestrateur : Kubernetes redémarre automatiquement le conteneur défaillant pour tenter de restaurer son état sain. Cela ne redémarre pas tout le pod, uniquement le conteneur concerné.
upvoted 1 times

__rajan__ 1 year, 7 months ago
This is the default behavior for failed liveness probes. The container is restarted with a back-off delay. The delay increases exponentially with each failure (10s, 20s, 40s, etc.). The delay is capped at 5 minutes.
upvoted 1 times

Question #5 *Topic 1*

Is this a way to configure the Docker engine to use a registry without a trusted TLS certificate?
Set INSECURE_REGISTRY in the '/etc/docker/default' configuration file.

A. Yes

B. No

Suggested Answer: *B*
Official Answer: B

Community vote distribution
B (100%)

__rajan__ Most Recent 1 year, 7 months ago
The Correct answer is No /etc/docker/default is not a valid Docker configuration file it should be "/etc/docker/daemon.json" INSECURE_REGISTRY is not the correct parameter name (it should be "insecure-registries") The configuration should be in JSON format in daemon.json
upvoted 1 times

vladcp 1 year, 10 months ago
I believe this should be setting 'insecure-registries' in /etc/docker/daemon.json
upvoted 1 times

forgaoqiang 1 year, 11 months ago
I check the docker document, it's "insecure-registries" not "insecure-registry"
upvoted 1 times

forgaoqiang 1 year, 11 months ago
setting INSECURE_REGISTRY in the /etc/docker/default configuration file is a way to configure the Docker engine to use a registry without a trusted TLS certificate The Answer should be YES, it's not recommended,but it works
upvoted 1 times

Question #6 *Topic 1*

Is this a way to configure the Docker engine to use a registry without a trusted TLS certificate?
Set and export the IGNORE_TLS environment variable on the command line.

A. Yes

B. No

Suggested Answer: *B*
Official Answer: B

Community vote distribution
B (100%)

__rajan__ Most Recent 1 year, 7 months ago
The correct ways to configure Docker to use an insecure registry are: a. Using the daemon.json file: { "insecure-registries" : ["myregistry.example.com:5000"] } b. Using the Docker daemon flag: --insecure-registry myregistry.example.com:5000
upvoted 1 times

LavaPup 1 year, 8 months ago
No, Docker does not have an environment variable like IGNORE_TLS to bypass TLS verification for a registry. If you need to interact with a Docker registry that does not have a trusted TLS certificate, you must configure Docker explicitly to allow the use of an insecure registry.
upvoted 1 times

Question #7 *Topic 1*

Is this a way to configure the Docker engine to use a registry without a trusted TLS certificate?
Pass the '--insecure registry' flag to the daemon at run time.

A. Yes

B. No

Suggested Answer: *A*
Official Answer: A

Community vote distribution
A (75%) B (25%)

Raji82 Most Recent 1 year, 6 months ago
It's A ish... should be --insecure-registry not --insecure registry
upvoted 1 times

badalchavan11 1 year, 6 months ago
A. Yes Explanation: Passing the --insecure-registry flag to the Docker daemon at runtime is a valid way to configure the Docker engine to use a registry without a trusted TLS certificate. Details: The --insecure-registry flag tells the Docker daemon to allow communication with a specific registry without verifying its TLS certificate. However, this flag must be passed when starting the Docker daemon. Example: If starting the Docker daemon manually, you can use: dockerd --insecure-registry :
upvoted 1 times

__rajan__ 1 year, 7 months ago
Using the --insecure-registry flag when starting the Docker daemon is a valid and supported method to configure Docker to accept an untrusted registry.
upvoted 1 times

LavaPup 1 year, 8 months ago
No, you cannot pass the --insecure-registry flag to the Docker daemon at runtime. This is a configuration option that must be set before starting the Docker daemon. Any changes to this setting require restarting the Docker service.
upvoted 1 times

vladcp 1 year, 10 months ago
I think the answer is B. "--insecure" is not a valid flag.
upvoted 1 times

Question #8 *Topic 1*

Is this an advantage of multi-stage builds?
better logical separation of Dockerfile instructions for increased readability

A. Yes

B. No

Suggested Answer: *A*
Official Answer: A

Community vote distribution
A (100%)

__rajan__ Most Recent 1 year, 7 months ago
This logical separation is particularly valuable in complex applications where you need different tools and dependencies for building versus running the application.
upvoted 1 times

LavaPup 1 year, 8 months ago
Yes, multi-stage builds in a Dockerfile offer significant advantages, particularly when optimizing Docker image size and improving build efficiency.
upvoted 1 times

walid1009 2 years, 7 months ago
Yes, as you can use one first stage for installing packages and dependencies and a second stage to copy specified binaries. This separation makes to Dockerfile more organized and readable.
upvoted 2 times

Question #9 *Topic 1*

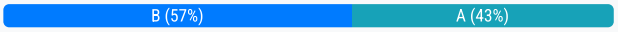
Is this an advantage of multi-stage builds?
better caching when building Docker images

A. Yes

B. No

Suggested Answer: *A*
Official Answer: A

Community vote distribution



metal Most Recent 2 months, 2 weeks ago

The confusion arises because multi-stage builds can indirectly help you write Dockerfiles that cache more efficiently, but the advantage is not the caching itself. "Use multi-stage builds" as a related resource, it does not state that multi-stage builds provide "better caching." he Docker documentation explains: Each instruction creates a layer. When a layer changes, that layer and all downstream layers must rebuild. This rule applies equally to single-stage and multi-stage builds. Multi-stage builds help you design your Dockerfile so that: Infrequently changed steps (like installing compilers or dependencies) go in early stages. Frequently changed steps (like copying source code) go in later stages. However, the caching mechanism itself is not better. It is the same mechanism. You can achieve identical cache efficiency in a single-stage build by carefully ordering instructions.

upvoted 1 times

ogogundare 4 months, 4 weeks ago

Explanation: Multi-stage builds are mainly used for: - Reducing image size - Separating build and runtime stages - Improving Dockerfile organization Why This Is Incorrect - Docker layer caching works the same regardless of multi-stage builds - Caching efficiency depends on: Instruction order File changes - Multi-stage builds do not inherently improve caching Key Point - Multi-stage builds ≠ better caching - They help with image optimization and structure

upvoted 1 times

dai_5_leva 10 months ago

In essence, multi-stage builds help you organize your Dockerfile so that the slow-to-change parts (like dependencies) are built early and cached effectively, while the fast-to-change parts (like your application code) trigger rebuilds only for the necessary steps.

upvoted 1 times

cea2cbe 11 months ago

Multi-stage builds allow for more efficient use of Docker's layer caching mechanism.

upvoted 1 times

1twoMany 1 year ago

Docker's Caching Mechanism: The Backbone of Optimization Each Dockerfile instruction creates an image layer, making Docker images a stack of these layers. Docker stores these intermediate images in its cache to accelerate future builds. When building an image, Docker checks if there's a cached layer corresponding to each instruction. If an identical layer exists and the context hasn't changed, Docker uses the cached layer instead of executing the instruction anew. This caching mechanism significantly speeds up image builds.

upvoted 1 times

radurocio 1 year, 5 months ago

Caching can help you improve build times and performance because you only build the layer that changed, not the entire project.

upvoted 1 times

__rajan__ 1 year, 7 months ago

Multi-stage builds are not primarily designed to improve caching. Docker's layer caching works the same way in both single-stage and multi-stage builds.

upvoted 1 times

b_script 2 years ago

A. Multi-stage builds in Docker offer better caching mechanisms when building Docker images, which is indeed an advantage. This approach allows Docker to cache the results of individual stages in the build process. If a particular stage hasn't changed (for example, the dependencies installation step), Docker can reuse the cached layer from a previous build instead of executing the instructions again. This can significantly reduce build times,

especially for complex builds where certain steps (like installing dependencies) do not change often but take a considerable amount of time to complete.

upvoted 1 times

michele91 2 years, 1 month ago

I am not sure that's correct. Documentation seems to hint that multi-stage build help with caching: https://docs.docker.com/build/cache/#use-multi-stage-builds

upvoted 2 times

Question #10 *Topic 1*

You add a new user to the engineering organization in DTR.
Will this action grant them read/write access to the engineering/api repository?
Add the user directly to the list of users with read/write access under the repository's Permissions tab.

A. Yes

B. No

Suggested Answer: *A*
Official Answer: A

Community vote distribution
A (100%)

__rajan__ Most Recent 1 year, 7 months ago
Adding a user to the engineering organization in DTR (Docker Trusted Registry) does not automatically grant them read/write access to specific repositories within that organization. To grant repository access, you need to either: a. Add the user directly to the repository's permission list b. Add the user to a team that has permissions to the repository
upvoted 1 times

LavaPup 1 year, 8 months ago
As per Docker Documentation - https://docs.oldtimes.me/docker/datacenter/dtr/2.4/guides/admin/manage-users/permission-levels/index.html
upvoted 1 times

Question #11 *Topic 1*

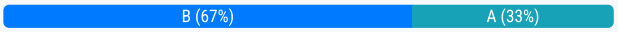
You add a new user to the engineering organization in DTR.
Will this action grant them read/write access to the engineering/api repository?
Mirror the engineering/api repository to one of the user's own private repositories.

A. Yes

B. No

Suggested Answer: *B*
Official Answer: B

Community vote distribution



Pepsi16 Most Recent 1 year, 5 months ago

Mirror the engineering/api repository to one of the user's own private repositories. individual users can only work on their own individual repositories.

upvoted 1 times

__rajan__ 1 year, 7 months ago

Mirroring the engineering/api repository to a user's private repository is NOT a way to automatically grant them read/write access to the original engineering/api repository.

upvoted 2 times

Question #12 *Topic 1*

You add a new user to the engineering organization in DTR.
Will this action grant them read/write access to the engineering/api repository?
Add them to a team in the engineering organization that has read/write access to the engineering/api repository.

A. Yes

B. No

Suggested Answer: *A*
Official Answer: A

Community vote distribution
A (100%)

__rajan__ Most Recent 1 year, 7 months ago
Adding user to a team with repository access will grant them read/write access.
upvoted 1 times

LavaPup 1 year, 8 months ago
Answer - A (Yes) Team member: Everything a user can do, plus the permissions granted by the teams the user is member of. As per Docker Documentation - https://docs.oldtimes.me/docker/datacenter/dtr/2.4/guides/admin/manage-users/permission-levels/index.html#overall-permissions
upvoted 1 times

Question #13 *Topic 1*

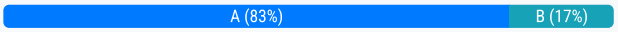
Seven managers are in a swarm cluster.
Is this how should they be distributed across three datacenters or availability zones?
3-2-2

A. Yes

B. No

Suggested Answer: *A*
Official Answer: A

Community vote distribution



OmarFadl Highly Voted 2 years, 7 months ago
A https://docs.docker.com/engine/swarm/admin_guide/#distribute-manager-nodes
upvoted 6 times

2ca247d Most Recent 6 months, 1 week ago
A https://docs.docker.com/engine/swarm/admin_guide/#distribute-manager-nodes
upvoted 1 times

Pepsi16 1 year, 5 months ago
true Seven Manager
upvoted 1 times

__rajan__ 1 year, 7 months ago
Swarm Manager Quorum Requirements, requires a majority (N/2 + 1) of managers to maintain quorum.
upvoted 2 times

LavaPup 1 year, 8 months ago
B. (No) I believe the correct answer would be 3-3-1
upvoted 1 times

bes_i 2 years ago
A right answer
upvoted 1 times

Archana_25 2 years, 8 months ago
A FYR - https://docs.docker.com/engine/swarm/admin_guide/
upvoted 3 times

Question #14 *Topic 1*

Seven managers are in a swarm cluster.

Is this how should they be distributed across three datacenters or availability zones?

5-1-1

A. Yes

B. No

Suggested Answer: *B*

Official Answer: B

Community vote distribution

B (100%)

fonro Most Recent 1 year, 5 months ago

(1-1-1) + (1-1-1) + (1-0-0) = 3-2-2

upvoted 1 times

__rajan__ 1 year, 7 months ago

Swarm Manager Quorum Requirements, requires a majority (N/2 + 1) of managers to maintain quorum.

upvoted 1 times

OmarFadl 2 years, 1 month ago

B https://docs.docker.com/engine/swarm/admin_guide/#distribute-manager-nodes

upvoted 2 times

Question #15 *Topic 1*

Is this a function of UCP?
image role-based access control

A. Yes

B. No

Suggested Answer: *B*
Official Answer: B

Community vote distribution
B (50%) A (50%)

dreagle Highly Voted 1 year, 10 months ago
It's RBAC 'role-based access control' not image role-based access control
upvoted 5 times

Pepsi16 Most Recent 1 year, 4 months ago
I'm not correct, but the answer is true, Why look at This from form rols of UCP Image operations All Image operations Image Build Image Commit Image Create Image Export Image History Image Load Image Prune Image Push Image Remove Image Remove parameters Image Search Image Tag Image View
upvoted 1 times

Pepsi16 1 year, 4 months ago
false ucp --> RBAC
upvoted 1 times

Pepsi16 1 year, 4 months ago
UCP just role-based access control (RBAC) not image
upvoted 1 times

diazed 1 year, 5 months ago
Its B. UCP has role-based access control (RBAC), so that you can control who can access and make changes to your cluster and applications. Not Image level
upvoted 1 times

__rajan__ 1 year, 7 months ago
UCP (Universal Control Plane) does provide image role-based access control (RBAC) as one of its core functions. This allows: * Fine-grained control over image access * Control over who can pull/push images * Management of image permissions at user and team levels
upvoted 3 times

LavaPup 1 year, 8 months ago
B is correct. Built-in security and access control Docker UCP has its own built-in authentication mechanism and integrates with LDAP services. It also has role-based access control (RBAC), so that you can control who can access and make changes to your cluster and applications.
https://dockerlabs.collabnix.com/advanced/Docker-UCP-overview.html
upvoted 1 times

Question #16 *Topic 1*

Is this a function of UCP?
enforces the deployment of signed images to the cluster

A. Yes

B. No

Suggested Answer: *A*
Official Answer: A

Community vote distribution
A (100%)

__rajan__ Most Recent 1 year, 7 months ago
Key Content Trust Features in UCP: * Can require images to be signed before deployment * Integrates with Docker Content Trust (DCT) * Enforces signature verification at the cluster level * Works with DTR (Docker Trusted Registry) for image signing
upvoted 1 times

LavaPup 1 year, 8 months ago
A is correct. Yes, Docker Universal Control Plane (UCP) supports enforcing the deployment of signed images to the cluster through Content Trust, which is part of Docker's security features. By enabling content trust in UCP, you can ensure that only images signed by trusted parties can be deployed in your swarm cluster or Kubernetes environment.
upvoted 1 times

Question #17 *Topic 1*

Is this the purpose of Docker Content Trust?
Enable mutual TLS between the Docker client and server.

A. Yes

B. No

Suggested Answer: *B*
Official Answer: B

Community vote distribution
B (100%)

__rajan__ Most Recent 1 year, 7 months ago
Docker Content Trust (DCT) is NOT used for enabling mutual TLS between Docker client and server. Its actual purpose is: * Provides digital signing and verification of container images * Ensures image integrity and authenticity * Verifies the publisher of container images * Protects against image tampering
upvoted 1 times

walid1009 2 years, 1 month ago
No, the DCT manages the signature of images so that they become authorized to be pulled / pushed. TLS connection needs digital certificates to secure the main communication between the server and the client.
upvoted 3 times

Question #18 *Topic 1*

Is this the purpose of Docker Content Trust?
Sign and verify image tags.

A. Yes

B. No

Suggested Answer: *A*
Official Answer: A

Community vote distribution
A (100%)

__rajan__ Most Recent 1 year, 7 months ago
Docker Content Trust (DCT): • Provides digital signing and verification of container images • Ensures image integrity and authenticity • Verifies the publisher of container images • Protects against image tampering
upvoted 1 times

Anderson23 2 years ago
A. Yes, Docker Content Trust (DCT) is a security feature in Docker that provides image signing and verification. Its main purpose is to ensure the integrity and authenticity of Docker images. When Docker Content Trust is enabled, Docker signs the images with digital signatures, and users can verify the authenticity of the images before pulling and running them on their systems.
upvoted 2 times

Question #19 *Topic 1*

Is this the purpose of Docker Content Trust?
Indicate an image on Docker Hub is an official image.

A. Yes

B. No

Suggested Answer: *B*
Official Answer: B

Community vote distribution
B (100%)

__rajan__ Most Recent 1 year, 7 months ago
Docker Content Trust (DCT) is not used to indicate official images on Docker Hub. Official Images on Docker Hub: * Are managed and maintained by Docker, Inc. * Are identified by their namespace location (no username prefix) * Have a "Official Image" badge on Docker Hub * Go through Docker's verification process * Example: nginx, ubuntu, mysql (vs. username/nginx)
upvoted 1 times

LavaPup 1 year, 8 months ago
B is correct. No, Docker Content Trust (DCT) does not inherently indicate whether an image on Docker Hub is an official image. Docker Content Trust is focused on ensuring the authenticity and integrity of an image through cryptographic signatures. It guarantees that the image being pulled or pushed has been signed by a trusted party, providing confidence that the image has not been tampered with.
https://docs.docker.com/engine/security/trust/
upvoted 1 times

Question #20 *Topic 1*

In the context of a swarm mode cluster, does this describe a node? an instance of the Docker CLI connected to the swarm

A. Yes

B. No

Suggested Answer: *B*

Official Answer: B

Community vote distribution

B (100%)

__rajan__ Most Recent 1 year, 7 months ago

This is because in the context of a Docker swarm mode cluster, a node is NOT simply an instance of the Docker CLI connected to the swarm. Instead, a node is an actual Docker Engine instance participating in the swarm.

upvoted 2 times

LavaPup 1 year, 8 months ago

B is correct. The Docker CLI can be used on any machine to interact with the Docker Engine and manage Swarm clusters. However, the CLI is just a client that interacts with the Docker daemon. It does not, by itself, create or register the machine as a node in the Swarm cluster.

upvoted 1 times

vladcp 1 year, 10 months ago

Technically, you'd need to have the docker daemon running on a node, having the Docker CLI is not enough.

upvoted 1 times

ubdubdoo 2 years, 2 months ago

youre not correct. someone using an instance of the CLI does not automatically make that instance a node.

upvoted 1 times

walid1009 2 years, 7 months ago

yes ! It could be physical or virtual and is called "node" in Docker Swarm. The node can join the swarm (manager) as a worker using the token generated from this command : $ docker swarm join-token worker

upvoted 1 times

Question #21 *Topic 1*

In the context of a swarm mode cluster, does this describe a node? a virtual machine participating in the swarm

A. Yes

B. No

Suggested Answer: *A*

Official Answer: A

Community vote distribution

A (100%)

__rajan__ Most Recent 1 year, 7 months ago

In the context of a Docker swarm mode cluster, a virtual machine participating in the swarm is indeed considered a node.

upvoted 1 times

LavaPup 1 year, 8 months ago

A is correct. A node is an instance of the Docker engine that participates in a swarm. A swarm is a cluster of one or more nodes, which can be physical or virtual machines. https://docs.docker.com/engine/swarm/key-concepts

upvoted 1 times

Question #22 *Topic 1*

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

A. Yes

B. No

Suggested Answer: *A*
Official Answer: A

Community vote distribution
A (100%)

__rajan__ Most Recent 1 year, 7 months ago
UCP supports several authentication methods, including: SAML 2.0 LDAP Active Directory Built-in authentication
upvoted 1 times

LavaPup 1 year, 8 months ago
A seems to be correct. https://docs.mirantis.com/mke/3.4/ops/administer-cluster/enable-saml-authentication.html
upvoted 1 times

Question #23 *Topic 1*

Is this a supported user authentication method for Universal Control Plane?
X.500

A. Yes

B. No

Suggested Answer: *B*
Official Answer: B

Community vote distribution
B (100%)

__rajan__ Most Recent 1 year, 7 months ago
X 500 is not supported.
upvoted 1 times

Question #24 *Topic 1*

Is this statement correct?
A Dockerfile provides instructions for building a Docker image.

A. Yes

B. No

Suggested Answer: *A*
Official Answer: A

Currently there are no comments in this discussion, be the first to comment!

Question #25 *Topic 1*

Is this statement correct?

A Dockerfile stores persistent data between deployments of a container.

A. Yes

B. No

Suggested Answer: *B*

Official Answer: B

Community vote distribution

B (100%)

__rajan__ Most Recent 1 year, 7 months ago

Docker file contains the instruction for building an image.

upvoted 1 times

LavaPup 1 year, 8 months ago

B is correct. No, a Dockerfile does not store persistent data between deployments of a container. The Dockerfile is used to define the instructions for building a Docker image, not for managing persistent data across container deployments.

upvoted 1 times

Question #26 *Topic 1*

Will this command ensure that overlay traffic between service tasks is encrypted? docker network create -d overlay --secure

A. Yes

B. No

Suggested Answer: *B*

Official Answer: B

Community vote distribution



ogogundare Most Recent 4 months, 4 weeks ago

Explanation: The command: docker network create -d overlay --secure creates an overlay network with encryption enabled. What --secure Does Enables encryption (IPsec) for traffic between containers (service tasks) across nodes Protects data in transit within the swarm Key Point Encryption is configured at the network level All services attached to this network will have encrypted communication

upvoted 1 times

__rajan__ 1 year, 7 months ago

The correct flag to enable encryption for overlay network traffic is --opt encrypted (or -o encrypted), not --secure. The proper command would be: $ docker network create -d overlay --opt encrypted

upvoted 1 times

LavaPup 1 year, 8 months ago

B is correct. https://stackoverflow.com/questions/47589489/docker-communication-encryption-between-container-in-swarm-mode

upvoted 1 times

ProfessorJayy 1 year, 10 months ago

b. in order to secure overlay do --opt encrypted

upvoted 1 times

InversaRadice 1 year, 10 months ago

absolutely B is correct!

upvoted 1 times

bes_i 2 years ago

A is correct

upvoted 1 times

ubdubdoo 2 years, 2 months ago

docker network create -d overlay --opt encrypted

upvoted 1 times

OmarFadl 2 years, 7 months ago

B https://docs.datadoghq.com/security/default_rules/cis-docker-1.2.0-7.4/ -- opt encrypted

upvoted 2 times

Question #27 *Topic 1*

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

A. Yes

B. No

Suggested Answer: *B*
Official Answer: B

Community vote distribution
B (100%)

__rajan__ Most Recent 1 year, 7 months ago
Creating separate Dockerfiles for each environment: * Violates the "build once, run anywhere" principle * Makes maintenance more difficult * Increases the risk of configuration drift * Requires rebuilding images for configuration changes * Goes against Docker best practices The better approaches for handling environment-specific configurations - Using Docker Configs and Secrets, Using Environment Variables, Using Docker Volumes.
upvoted 1 times

LavaPup 1 year, 8 months ago
B is correct.
upvoted 1 times

ubdubdoo 2 years, 2 months ago
B While technically possible, creating a separate Dockerfile for each environment is not the most efficient or scalable way to manage configurations that vary across environments.
upvoted 2 times

Question #28 *Topic 1*

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

A. Yes

B. No

Suggested Answer: *B*
Official Answer: B

Community vote distribution
B (100%)

__rajan__ Most Recent 1 year, 7 months ago
The complete process to fully delete an image from DTR requires additional steps: Delete the image through the DTR interface or API Delete the repository Run garbage collection The garbage collection step is crucial because: * When you delete images/repositories from DTR, the actual image files remain on disk * These files are marked for deletion but are not immediately removed * Only after running garbage collection will the actual disk space be reclaimed * This is a safety mechanism to prevent accidental data loss
upvoted 1 times

LavaPup 1 year, 8 months ago
B is correct. https://stackoverflow.com/questions/25436742/how-to-delete-images-from-a-private-docker-registry
upvoted 1 times

Question #29 *Topic 1*

Will this sequence of steps completely delete an image from disk in the Docker Trusted Registry?
Delete the image and run garbage collection on the Docker Trusted Registry.

A. Yes

B. No

Suggested Answer: *A*
Official Answer: A

Community vote distribution
A (100%)

__rajan__ Most Recent 1 year, 7 months ago
The complete process to fully delete an image from DTR requires additional steps: 1. Delete the image through the DTR interface or API 2. Delete the repository 3. Run garbage collection
upvoted 1 times

ubdubdoo 1 year, 8 months ago
A: DTR's garbage collection process is responsible for actually removing the image data from disk. You'll need to trigger this process manually after deleting the tags.
upvoted 2 times

Question #30 *Topic 1*

Does this describe the role of Control Groups (cgroups) when used with a Docker container? user authorization to the Docker API

A. Yes

B. No

Suggested Answer: *B*

Official Answer: B

Community vote distribution

B (100%)

__rajan__ Most Recent 1 year, 7 months ago

Control Groups (cgroups) are NOT used for user authorization to the Docker API.

upvoted 1 times

LavaPup 1 year, 8 months ago

B is correct. Control Groups (cgroups) are used for Limit Resource Usage, Isolate Containers, Monitor Resource Consumption.

https://medium.com/@maheshwar.ramkrushna/managing-docker-resources-with-cgroups-a-practical-guide-169289c80451

upvoted 1 times

Question #31 *Topic 1*

Does this describe the role of Control Groups (cgroups) when used with a Docker container? role-based access control to clustered resources

A. Yes

B. No

Suggested Answer: *B*

Official Answer: B

Community vote distribution

B (100%)

__rajan__ Most Recent 1 year, 7 months ago

B is correct. Control Groups (cgroups) are used to: • Limit and monitor system resources used by containers • Control CPU usage • Manage memory allocation • Restrict disk I/O • Control network bandwidth • Set process priorities

upvoted 1 times

LavaPup 1 year, 8 months ago

B is correct. Control Groups (cgroups) are used for Limit Resource Usage, Isolate Containers, Monitor Resource Consumption.

https://medium.com/@maheshwar.ramkrushna/managing-docker-resources-with-cgroups-a-practical-guide-169289c80451

upvoted 1 times

Question #32 *Topic 1*

Does this describe the role of Control Groups (cgroups) when used with a Docker container? accounting and limiting of resources

A. Yes

B. No

Suggested Answer: *A*

Official Answer: A

Community vote distribution

A (100%)

LavaPup Most Recent 1 year, 8 months ago

A (Yes) is correct. Control Groups (cgroups) are used for Limit Resource Usage, Isolate Containers, Monitor Resource Consumption.

https://medium.com/@maheshwar.ramkrushna/managing-docker-resources-with-cgroups-a-practical-guide-169289c80451

upvoted 1 times

Question #33 *Topic 1*

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?
Create one pod and add all the resources needed for each application.

A. Yes

B. No

Suggested Answer: *B*
Official Answer: B

Community vote distribution
B (100%)

__rajan__ Most Recent 1 year, 7 months ago
The correct approach would be to use Kubernetes Namespaces because: Namespaces provide logical isolation between different applications They allow you to: Group related resources (pods, secrets, configmaps, etc.) together Apply resource quotas per application/team Set up role-based access control (RBAC) specific to each team Avoid naming conflicts between different applications
upvoted 1 times

ProfessorJayy 1 year, 10 months ago
given answer is correct. You'd use namespaces to logically separate and group resources for each application, ensuring that the application's secrets, config maps, and other resources are isolated from others.
upvoted 1 times

Question #34 *Topic 1*

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?
Create one namespace for each application and add all the resources to it.

A. Yes

B. No

Suggested Answer: *A*
Official Answer: A

Currently there are no comments in this discussion, be the first to comment!

Question #35 *Topic 1*

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?
Create a collection for each application.

A. Yes

B. No

Suggested Answer: *B*
Official Answer: B

Currently there are no comments in this discussion, be the first to comment!

Question #36 *Topic 1*

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?
‘docker inspect http’

A. Yes

B. No

Suggested Answer: *B*
Official Answer: B

Community vote distribution
B (100%)

Pepsi16 Most Recent 1 year, 4 months ago
docker logs
upvoted 2 times

ubdubdoo 1 year, 8 months ago
docker service inspect is specific to services in a Docker Swarm.
upvoted 1 times

OmarFadl 2 years, 1 month ago
docker service inspect http
upvoted 1 times

Question #37 *Topic 1*

The Kubernetes yaml shown below describes a clusterIP service.
Is this a correct statement about how this service routes requests?
Traffic sent to the IP of any pod with the label app: nginx on port 8080 will be forwarded to port 80 in that pod.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: dca
spec:
  type: clusterIP
  selector:
      app: nginx
  ports:
  - port: 8080
    targetPort: 80
  - port: 4443
    targetPort: 443
```

A. Yes

B. No

Suggested Answer: *B*
Official Answer: B

Community vote distribution
B (67%) A (33%)

ogogundare Most Recent 4 months, 3 weeks ago
Explanation: In a ClusterIP Service, traffic is routed through the Service IP, not directly to pod IPs on the service port. How It Actually Works Client sends traffic to: ServiceIP:8080 Kubernetes then: Selects pods with label app: nginx Forwards traffic to: PodIP:80 Why the Statement is Incorrect The statement says: Traffic sent to the IP of any pod on port 8080... □ This is wrong because: Pods do not listen on the service port (8080) The Service handles the port mapping, not the pods directly Correct Behavior ServiceIP:8080 → PodIP:80 Key Point: Service abstracts access to pods You don't send traffic to pod IP on service port
upvoted 1 times

Pepsi16 1 year, 4 months ago
The question is indected the IP pod no first go the IP service then is forwarded to the IP pod
upvoted 1 times

walid1009 2 years, 7 months ago
A. Yes
upvoted 2 times

Lightfood 1 year, 12 months ago
Correct yaml for internal requests in a cluster
upvoted 1 times

walid1009 2 years, 7 months ago
A. Yesthe question mentions traffic sent to pod IP, not the cluster virtual IP so the correct answer is Bhttps://www.baeldung.com/ops/kubernetes-service-types#1-clusterip-servicesI think the correct answer is "A", because the question only states "requests", not requests from outside of the cluster. So intern request would be routed correctly -> A is the correct answer
upvoted 2 times

OmarFadl 2 years, 7 months ago
the question mentions traffic sent to pod IP, not the cluster virtual IP so the correct answer is B
upvoted 2 times

OmarFadl 2 years, 7 months ago
https://www.baeldung.com/ops/kubernetes-service-types#1-clusterip-services
upvoted 1 times

OmarFadl 2 years, 7 months ago
https://www.baeldung.com/ops/kubernetes-service-types#1-clusterip-servicesI think the correct answer is "A", because the question only states "requests", not requests from outside of the cluster. So intern request would be routed correctly -> A is the correct answer

upvoted 1 times

Lightfood 1 year, 12 months ago

I think the correct answer is "A", because the question only states "requests", not requests from outside of the cluster. So intern request would be routed correctly -> A is the correct answer

upvoted 2 times

Question #38 *Topic 1*

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? label constraints

A. Yes

B. No

Suggested Answer: *A*
Official Answer: A

Currently there are no comments in this discussion, be the first to comment!

Question #39 *Topic 1*

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? environment variables

A. Yes

B. No

Suggested Answer: *B*
Official Answer: B

Currently there are no comments in this discussion, be the first to comment!

Question #40 *Topic 1*

Will a DTR security scan detect this?
private keys copied to the image

A. Yes

B. No

Suggested Answer: *A*
Official Answer: A

Community vote distribution
B (100%)

ogogundare Most Recent 4 months, 3 weeks ago
Explanation: Docker Trusted Registry (DTR) security scanning focuses on: Known vulnerabilities (CVEs) Package dependencies License information Why This Is Incorrect It does not detect sensitive data exposure, such as: Private keys Credentials Secrets embedded in images Key Point: DTR scan = vulnerability & license scanning Not = secret detection
upvoted 1 times

bes_i 2 years ago
DTR's built-in security scans focus on detecting vulnerabilities and ensuring compliance with security policies
upvoted 1 times

ubdubdoo 2 years, 2 months ago
Aggressively searching for private keys could lead to many false positives, as certain patterns might resemble keys but not actually be them
upvoted 1 times

b_script 2 years, 6 months ago
B. No Docker Trusted Registry (DTR) security scanning is designed to detect vulnerabilities in the software and dependencies within container images, such as known security issues in packages or libraries. It is not specifically designed to detect the presence of private keys or other sensitive information that may have been copied into an image.
upvoted 1 times

Question #41 *Topic 1*

Will a DTR security scan detect this?
image configuration poor practices, such as exposed ports or inclusion of compilers in production images

A. Yes

B. No

Suggested Answer: *B*
Official Answer: B

ubdubdoo Most Recent 2 years, 2 months ago

A: DTR incorporates security best practice rules. These rules often check for common misconfigurations and flag them, even if they aren't directly tied to a known vulnerability.

upvoted 1 times

Question #42 *Topic 1*

You configure a local Docker engine to enforce content trust by setting the environment variable DOCKER_CONTENT_TRUST=1. If myorg/myimage:1.0 is unsigned, does Docker block this command? docker image build, from a Dockerfile that begins FROM myorg/myimage:1.0

A. Yes

B. No

Suggested Answer: *A*

Official Answer: A

Community vote distribution

A (100%)

bes_i Most Recent 2 years ago

A. yes

upvoted 1 times

bes_i 2 years ago

When DOCKER_CONTENT_TRUST=1 is set, Docker enforces the use of signed images.

upvoted 1 times

Question #43 *Topic 1*

Can this set of commands identify the published port(s) for a container?
'docker container inspect', 'docker port'

A. Yes

B. No

Suggested Answer: *A*
Official Answer: A

OmarFadl Most Recent 2 years, 1 month ago
A $ docker port sharp_austin 80/tcp -> 0.0.0.0:8080
upvoted 1 times

Question #44 *Topic 1*

Can this set of commands identify the published port(s) for a container?
'docker port inspect', 'docker container inspect'

A. Yes

B. No

Suggested Answer: *B*
Official Answer: B

ubdubdoo Most Recent 2 years, 2 months ago
docker port inspect is not a thing
upvoted 1 times

Question #45 *Topic 1*

A persistentVolumeClaim (PVC) is created with the specification storageClass: "", and size requirements that cannot be satisfied by any existing persistentVolume.
Is this an action Kubernetes takes in this situation?
The PVC remains unbound until a persistentVolume that matches all requirements of the PVC becomes available.

A. Yes

B. No

Suggested Answer: *A*
Official Answer: A

Currently there are no comments in this discussion, be the first to comment!

Question #46 *Topic 1*

Will this configuration achieve fault tolerance for managers in a swarm? an odd number of manager nodes, totaling more than two

A. Yes

B. No

Suggested Answer: *A*
Official Answer: A

Currently there are no comments in this discussion, be the first to comment!

Question #47 *Topic 1*

Will this configuration achieve fault tolerance for managers in a swarm? only two managers, one active and one passive

A. Yes

B. No

Suggested Answer: *B*
Official Answer: B

Currently there are no comments in this discussion, be the first to comment!

Question #48 *Topic 1*

You want to provide a configuration file to a container at runtime.
Does this set of Kubernetes tools and steps accomplish this?
Turn the configuration file into a configMap object, use it to populate a volume associated with the pod, and mount that file from the volume to the appropriate container and path.

A. Yes

B. No

Suggested Answer: *A*
Official Answer: A

Currently there are no comments in this discussion, be the first to comment!

Question #49 *Topic 1*

You want to create a container that is reachable from its host's network.
Does this action accomplish this?
Use either EXPOSE or --publish to access the container on the bridge network.

A. Yes

B. No

Suggested Answer: *B*
Official Answer: B

Community vote distribution
B (100%)

bes_i Most Recent 2 years ago
No, using EXPOSE alone will not make the container reachable from the host's network. Yes, using --publish (or -p) will accomplish this.
upvoted 1 times

Question #50 *Topic 1*

You want to create a container that is reachable from its host's network.
Does this action accomplish this?
Use network connect to access the container on the bridge network.

A. Yes

B. No

Suggested Answer: *B*
Official Answer: B

Currently there are no comments in this discussion, be the first to comment!

Question #51 *Topic 1*

Will this action upgrade Docker Engine CE to Docker Engine EE?
Disable the Docker service via ‘chkconfig’ or ‘systemctl’.

A. Yes

B. No

Suggested Answer: *B*
Official Answer: B

Currently there are no comments in this discussion, be the first to comment!

Question #52 *Topic 1*

Will this action upgrade Docker Engine CE to Docker Engine EE?
Run docker engine activate.

A. Yes

B. No

Suggested Answer: *B*
Official Answer: B

Community vote distribution
B (50%) A (50%)

ogogundare Most Recent 4 months, 3 weeks ago
Explanation: The command: docker engine activate is used to activate an Enterprise license on an already installed Docker Enterprise Engine (EE). Why This Does NOT Upgrade CE → EE It does not install Docker EE It does not convert Docker CE into EE It only enables enterprise features on an EE installation Correct Upgrade Process To upgrade from CE to EE: Uninstall Docker CE Install Docker EE packages Run docker engine activate (to apply license) Key Point: Activate ≠ Upgrade You must install EE first
upvoted 1 times

Pepsi16 1 year, 4 months ago
No, because you should be the docker CE deleted then install docker EE from Mirantis
upvoted 1 times

Lightfood 1 year, 12 months ago
If you are running a Community Edition engine, with "docker engine activate" the daemon will be updated to Enterprise, see https://manpages.ubuntu.com/manpages/xenial/man1/docker-engine-activate.1.html
upvoted 2 times