Docker.DCA.v2023-09-11.q65

Exam Code:	DCA
Exam Name:	Docker Certified Associate (DCA) Exam
Certification Provider:	Docker
Free Question Number:	65
Version:	v2023-09-11
# of views:	2407
# of Questions views:	650
https://www.freepdfdumps.com/Docker.DCA.v2023-09-11.q65.html	

NEW QUESTION: 1
Is this a supported user authentication method for Universal Control Plane?
Solution: PAM
A. Yes
B. No
Answer: (SHOW ANSWER)
Explanation
PAM is not a supported user authentication method for Universal Control Plane. According to the official documentation, the supported methods are LDAP, Active Directory, SAML 2.0, and local users.
References: https://docs.docker.com/ee/ucp/admin/configure/external-auth/

NEW QUESTION: 2
The following Docker Compose file is deployed as a stack:

Is this statement correct about this health check definition?
Solution: Health checks test for app health ten seconds apart. Three failed health checks transition the container into "unhealthy" status.
A. Yes
B. No
Answer: A (LEAVE A REPLY)

NEW QUESTION: 3
Your organization has a centralized logging solution, such as Sptunk.
Will this configure a Docker container to export container logs to the logging solution?
Solution. docker system events- -filter splunk
A. Yes
B. No
Answer: B (LEAVE A REPLY)
Explanation
This does not configure a Docker container to export container logs to the logging solution. The docker system events command shows information about real-time events in the Docker daemon. The --filter flag allows you to filter the output by various criteria, such as type, action, image, container, etc. However, splunk is not a valid filter value and will cause an error. To configure a Docker container to export container logs to the logging solution, you need to use the --log-driver and --log-opt flags when creating or running the container.
These flags allow you to specify which logging driver and options to use for the container. For example, to use Splunk as the logging driver, you can use --log-driver splunk and provide the Splunk URL, token, and other options using --log-opt. References:
https://docs.docker.com/engine/reference/commandline/system_events/,
https://docs.docker.com/config/containers/logging/configure/,
https://docs.docker.com/config/containers/logging/splunk/

NEW QUESTION: 4
Does this command display all the pods in the cluster that are labeled as 'env: development'?
Solution: 'kubectl get pods --all-namespaces -l env=development'
A. Yes
B. No
Answer: A (LEAVE A REPLY)

NEW QUESTION: 5
Does this command display all the pods in the cluster that are labeled as 'env: development'?
Solution: 'kubectl get pods --all-namespaces -label env=development'
A. Yes
B. No
Answer: B (LEAVE A REPLY)

NEW QUESTION: 6
You are troubleshooting a Kubernetes deployment called api, and want to see the events table for this object.
Does this command display it?
Solution: kubectl events deployment api

A. Yes
B. No
Answer: B (LEAVE A REPLY)
Explanation
Using kubectl events deployment api does not display the events table for this object. The kubectl events command shows cluster-level events, but it does not accept a resource name as an argument. To see the events table for this object, you need to use kubectl describe deployment api. References:
https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#events,
https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#describe

NEW QUESTION: 7
Is this the purpose of Docker Content Trust?
Solution. Sign and verify image tags.
A. Yes
B. No
Answer: A (LEAVE A REPLY)
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. References: https://docs.docker.com/engine/security/trust/

NEW QUESTION: 8
Will This command list all nodes in a swarm cluster from the command line?
Solution. 'docker swarm nodes'
A. Yes
B. No
Answer: B (LEAVE A REPLY)
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. References:
https://docs.docker.com/engine/reference/commandline/swarm/,
https://docs.docker.com/engine/reference/commandline/node_ls/

NEW QUESTION: 9
You want to create a container that is reachable from its host's network. Does this action accomplish this?
Solution: Use --link to access the container on the bridge network.
A. Yes
B. No
Answer: B (LEAVE A REPLY)
Explanation
Using --link to access the container on the bridge network does not make the container reachable from its host's network. The --link option allows containers to communicate with each other using a private network created by Docker. To make a container reachable from its host's network, you need to use either EXPOSE or
--publish to access the containers on the bridge network. References:
https://docs.docker.com/network/links/,
https://docs.docker.com/network/bridge/

NEW QUESTION: 10
Seven managers are in a swarm cluster.
Is this how should they be distributed across three datacenters or availability zones?
Solution: 5-1-1
A. Yes
B. No
Answer: (SHOW ANSWER)
Explanation
This is not how they should be distributed across three datacenters or availability zones, because having one manager in two datacenters or availability zones creates a risk of losing quorum if those datacenters or availability zones become unavailable. According to the official documentation, managers should be distributed evenly across datacenters or availability zones to ensure that the swarm can survive the loss of any one datacenter or availability zone.
References: https://docs.docker.com/engine/swarm/admin_guide/#add-manager-nodes-for-fault-tolerance

NEW QUESTION: 11
Seven managers are in a swarm cluster.
Is this how should they be distributed across three datacenters or availability zones?
Solution: 3-3-1
A. Yes
B. No
Answer: B (LEAVE A REPLY)
Explanation

This is not how they should be distributed across three datacenters or availability zones, because having one manager in one datacenter or availability zone creates a single point of failure and reduces the fault tolerance of the swarm. According to the official documentation, managers should be distributed evenly across datacenters or availability zones to ensure that the swarm can survive the loss of any one datacenter or availability zone.
References: https://docs.docker.com/engine/swarm/admin_guide/#add-manager-nodes-for-fault-tolerance

NEW QUESTION: 12
The Kubernetes yaml shown below describes a networkPolicy.

Will the networkPolicy BLOCK this traffic?
Solution: a request issued from a pod bearing the tier: backend label, to a pod bearing the tier: frontend label
A. Yes
B. No
Answer: A (LEAVE A REPLY)
Explanation
The networkPolicy will block this traffic because it only allows ingress traffic from pods that have the tier: api label. Pods that have the tier: backend label do not match this selector and are therefore denied access to pods that have the tier: frontend label.
References:
https://kubernetes.io/docs/concepts/services-networking/network-policies/#the-networkpolicy-resource

NEW QUESTION: 13
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 one pod and add all the resources needed for each application

A. Yes
B. No
Answer: B (LEAVE A REPLY)
Explanation
his is not a way to accomplish this, because creating one pod and adding all the resources needed for each application is not a good practice for deploying applications in Kubernetes. According to the official documentation, pods are not intended to run multiple instances of an application or different applications that are tightly coupled. Pods are also not meant to hold resources that are shared across applications, such as secrets or configMaps.
References: https://kubernetes.io/docs/concepts/workloads/pods/#what-is-a-pod

NEW QUESTION: 14
In Docker Trusted Registry, is this how a user can prevent an image, such as 'nginx:latest', from being overwritten by another user with push access to the repository?
Solution: Use the DTR web UI to make all tags in the repository immutable.
A. Yes
B. No
Answer: B (LEAVE A REPLY)
Explanation
This is not how a user can prevent an image from being overwritten by another user with push access to the repository, because making all tags in the repository immutable will also prevent the user from updating their own image tags. According to the official documentation, a better way to prevent an image from being overwritten is to use promotion policies that only allow certain users or teams to push images with specific tags.
References: https://docs.docker.com/ee/dtr/user/promotion-policies/overview/

NEW QUESTION: 15
The Kubernetes yaml shown below describes a networkPolicy.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: dbm
  namespace: default
spec:
  podSelector:
    matchLabels:
      tier: backend
  ingress:
  - from:
    - podSelector:
        matchLabels:
          tier: api
...
```

Will the networkPolicy BLOCK this trafftc?
Solution. a request issued from a pod bearing the tier: backend label, to a pod bearing the tier: frontend label

A. Yes
B. No
Answer: A (LEAVE A REPLY)
Explanation
The networkPolicy will block this traffic. A networkPolicy is a Kubernetes resource that defines how pods are allowed to communicate with each other and with other network endpoints. A networkPolicy has two main sections: podSelector and policyTypes. The podSelector selects which pods the networkPolicy applies to. The policyTypes specifies whether the networkPolicy affects ingress (incoming) traffic, egress (outgoing) traffic, or both. In this case, the networkPolicy applies to pods that have a label app: webserver and affects both ingress and egress traffic. The networkPolicy also has two optional sections: ingress and egress. The ingress section defines the rules for allowing ingress traffic to the selected pods. The egress section defines the rules for allowing egress traffic from the selected pods. If either section is missing or empty, it means that no traffic of that type is allowed. In this case, the networkPolicy has an empty ingress section, which means that no ingress traffic is allowed to the pods that have a label app: webserver. Therefore, a request issued from a pod bearing the tier: backend label, to a pod bearing the tier: frontend label will be blocked by this networkPolicy, since it is an ingress traffic to a pod that has a label app: webserver. References:
https://kubernetes.io/docs/concepts/services-networking/network-policies/,
https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.17/#networkpolicy-v1-networking-k8s-io

NEW QUESTION: 16
Will this configuration achieve fault tolerance for managers in a swarm?
Solution: only two managers, one active and one passive.
A. Yes
B. No
Answer: (SHOW ANSWER)
Explanation
This configuration will not achieve fault tolerance for managers in a swarm, because having only two managers creates a risk of losing quorum if one manager fails or becomes unavailable. According to the official documentation, having two managers also does not provide any benefits over having one manager, since both managers must be available for any management operations.
References: https://docs.docker.com/engine/swarm/admin_guide/#add-manager-nodes-for-fault-tolerance

Valid DCA Dumps shared by Actual4test.com for Helping Passing DCA Exam!
Actual4test.com now offer the **newest DCA exam dumps**, the Actual4test.com DCA exam

questions have been updated and **answers have been corrected** get the **newest** Actual4test.com DCA dumps with Test Engine here: https://www.actual4test.com/DCA_examcollection.html (**189** Q&As Dumps, **30%OFF** **Special Discount:** **Freepdfdumps**)

NEW QUESTION: 17
Is this a type of Linux kernel namespace that provides container isolation?
Solution: Storage
A. Yes
B. No
Answer: B (LEAVE A REPLY)
Explanation
Storage is not a type of Linux kernel namespace that provides container isolation. Namespaces are a Linux kernel feature that provide isolation and virtualization of system resources for processes. They can be used to create isolated environments for containers that have their own view of system resources, such as process IDs, user IDs, network interfaces, etc. However, there is no storage namespace in Linux. The types of namespaces that exist are mount (mnt), process ID (pid), network (net), interprocess communication (ipc), user ID (user), control group (cgroup), time (time), and user namespace (uts). References:
https://docs.docker.com/engine/security/userns-remap/,
https://man7.org/linux/man-pages/man7/namespaces.7.html

NEW QUESTION: 18
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
A. Yes
B. No
Answer: (SHOW ANSWER)
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. References:
https://docs.docker.com/engine/swarm/services/#specify-service-constraints,
https://docs.docker.com/engine/swarm/manage-nodes/#add-or-remove-label-metadata

NEW QUESTION: 19
You add a new user to the engineering organization in DTR.
Will this action grant them read/write access to the engineering/api repository?
Solution: Add the user directly to the list of users with read/write access under the repository's Permissions tab.
A. Yes
B. No
Answer: A (LEAVE A REPLY)
Explanation
This action will grant them read/write access to the engineering/api repository, because adding a user directly to the list of users with read/write access under the repository's Permissions tab is one way to grant permissions to a user in DTR. According to the official documentation, this is an example of using fine-grained permissions for a repository.
References: https://docs.docker.com/ee/dtr/user/manage-repos/permission-levels/

NEW QUESTION: 20
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: Add all the resources to the default namespace.
A. Yes
B. No
Answer: B (LEAVE A REPLY)
Explanation
This is not a way to accomplish this, because adding all the resources to the default namespace is not a good practice for isolating applications in Kubernetes. According to the official documentation, namespaces are used to group resources into logical units that correspond to different projects, teams, or environments. Using namespaces can help avoid naming collisions and enforce resource quotas and access policies.
References: https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/

NEW QUESTION: 21
Does this command create a swarm service that only listens on port 53 using the UDP protocol?
Solution. 'docker service create -name dns-cache -p 53:53 -constraint networking.protocol.udp=true dns-cache"
A. Yes
B. No
Answer: B (LEAVE A REPLY)
Explanation

This command does not create a swarm service that only listens on port 53 using the UDP protocol. The docker service create command creates a new service in a swarm cluster. The --name flag sets the name of the service. The -p or --publish flag publishes a port or a range of ports to the swarm. The --constraint flag applies a constraint to limit the set of nodes where the task can be scheduled. The networking.protocol.udp is not a valid constraint and will cause an error. To create a swarm service that only listens on port 53 using the UDP protocol, you need to use -p 53:53/udp instead. References:
https://docs.docker.com/engine/reference/commandline/service_create/,
https://docs.docker.com/engine/reference/commandline/service_create/#publish-service-ports-externally-to-the-s
https://docs.docker.com/engine/reference/commandline/service_create/#specify-service-constraints-constraint

NEW QUESTION: 22
Is this a type of Linux kernel namespace that provides container isolation?
Solution: Authentication
A. Yes
B. No
Answer: B (LEAVE A REPLY)
Explanation
Authentication is not a type of Linux kernel namespace that provides container isolation. Namespaces are a Linux kernel feature that provide isolation and virtualization of system resources for processes. They can be used to create isolated environments for containers that have their own view of system resources, such as process IDs, user IDs, network interfaces, etc. However, there is no authentication namespace in Linux. The types of namespaces that exist are mount (mnt), process ID (pid), network (net), interprocess communication (ipc), user ID (user), control group (cgroup), time (time), and user namespace (uts). References:
https://docs.docker.com/engine/security/userns-remap/,
https://man7.org/linux/man-pages/man7/namespaces.7.html

NEW QUESTION: 23
Is this the purpose of Docker Content Trust?
Solution: Enable mutual TLS between the Docker client and server.
A. Yes
B. No
Answer: B (LEAVE A REPLY)
Explanation
Enabling mutual TLS between the Docker client and server is not the purpose of Docker Content Trust.
According to the official documentation, the purpose of Docker Content Trust is to verify the integrity and publisher of all data received from a registry over any channel.

References: https://docs.docker.com/engine/security/trust/content_trust/

NEW QUESTION: 24
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'
A. Yes
B. No
Answer: B (LEAVE A REPLY)
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.
References: https://docs.docker.com/storage/bind-mounts/,
https://docs.docker.com/engine/reference/run/#volume-shared-filesystems

NEW QUESTION: 25
Is this a Linux kernel namespace that is disabled by default and must be enabled at Docker engine runtime to be used?
Solution: mnt
A. Yes
B. No
Answer: (SHOW ANSWER)
Explanation
mnt is not a Linux kernel namespace that is disabled by default and must be enabled at Docker engine runtime to be used. According to the official documentation, mnt is one of the namespaces that are enabled by default when using namespaces for isolation.
References: https://docs.docker.com/engine/security/userns-remap/#user-namespace-known-limitations

NEW QUESTION: 26
Can this set of commands identify the published port(s) for a container?
Solution. 'docker container inspect", docker port'
A. Yes
B. No
Answer: (SHOW ANSWER)
Explanation

This set of commands can identify the published port(s) for a container. The docker container inspect command shows low-level information about a container in JSON format. This information includes the network settings of the container, such as the port bindings and exposed ports. The docker port command shows the public port(s) that are mapped to a private port inside the container. By using these two commands, you can identify the published port(s) for a container. References:
https://docs.docker.com/engine/reference/commandline/container_inspect/,
https://docs.docker.com/engine/reference/commandline/port/

NEW QUESTION: 27
You configure a local Docker engine to enforce content trust by setting the environment variable DOCKER_C0NTENT_TRUST=l. If myorg/myimage: 1.0 is unsigned, does Docker block this command?
Solution. docker image build, from a Dockeflle that begins FROM myorg/myimage: l1.0
A. Yes
B. No
Answer: A (LEAVE A REPLY)
Explanation
Docker blocks this command if you configure a local Docker engine to enforce content trust by setting the environment variable DOCKER_CONTENT_TRUST=1 and if myorg/myimage:1.0 is unsigned. Content trust is a feature that allows you to use digital signatures to verify the integrity and publisher of specific image tags.
When you enable content trust, you can only pull, run, or build with trusted images. If an image tag is unsigned, Docker will block any command that attempts to use it. This includes docker image build, from a Dockerfile that begins FROM myorg/myimage:1.0, if myorg/myimage:1.0 is unsigned. References:
https://docs.docker.com/engine/security/trust/,
https://docs.docker.com/engine/reference/commandline/image_build/

NEW QUESTION: 28
Will this command display a list of volumes for a specific container?
Solution: 'docker container inspect nginx'
A. Yes
B. No
Answer: A (LEAVE A REPLY)
Explanation
This command will display a list of volumes for a specific container, because it uses docker container inspect to show detailed information about a container, including its volumes.
According to the official documentation, docker container inspect will show the Mounts section that contains information about all volumes mounted by the container.
References: https://docs.docker.com/engine/reference/commandline/container_inspect/

https://docs.docker.com/storage/volumes/#start-a-container-with-a-volume

NEW QUESTION: 29
Is this a supported user authentication method for Universal Control Plane?
Solution. SAML
A. Yes
B. No
Answer: (SHOW ANSWER)
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. References:
https://docs.docker.com/ee/ucp/admin/configure/external-auth/,
https://docs.docker.com/ee/ucp/admin/configure/saml/

NEW QUESTION: 30
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?
Solution: The PVC remains unbound until a persistentVolume that matches all requirements of the PVC becomes available.
A. Yes
B. No
Answer: A (LEAVE A REPLY)
Explanation
The PVC remains unbound until a persistentVolume that matches all requirements of the PVC becomes available is an action Kubernetes takes in this situation. A persistentVolumeClaim (PVC) is a request for storage by a user. A PVC specifies the desired size, access modes, and storage class of the storage. A persistentVolume (PV) is a piece of storage in the cluster that has been provisioned by an administrator. A PV has a lifecycle independent of any individual pod that uses it. A PVC is automatically bound to a suitable PV based on its requirements. If no PV exists that matches the PVC, and dynamic provisioning is not enabled, then the PVC remains unbound indefinitely until a matching PV is created. References:
https://kubernetes.io/docs/concepts/storage/persistent-volumes/,
https://kubernetes.io/docs/concepts/storage/persistent-volumes/#lifecycle-of-a-volume-and-claim

NEW QUESTION: 31

Will this sequence of steps completely delete an image from disk in the Docker Trusted Registry?
Solution: Delete the image and run garbage collection on the Docker Trusted Registry.
A. Yes
B. No
Answer: A (LEAVE A REPLY)
Explanation
Deleting the image and running garbage collection on the Docker Trusted Registry will completely delete the image from disk. According to the official documentation, this is the recommended way to remove images and reclaim disk space.
References: https://docs.docker.com/ee/dtr/admin/manage-images/garbage-collection/

Valid DCA Dumps shared by Actual4test.com for Helping Passing DCA Exam! Actual4test.com now offer the **newest DCA exam dumps**, the Actual4test.com DCA exam **questions have been updated** and **answers have been corrected** get the **newest** Actual4test.com DCA dumps with Test Engine here: https://www.actual4test.com/DCA_examcollection.html (**189** Q&As Dumps, **30%OFF** **Special Discount:** **Freepdfdumps**)

NEW QUESTION: 32
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 service ps http'
A. Yes
B. No
Answer: A (LEAVE A REPLY)
Explanation
Using 'docker service ps http' enables you to view the list of historical tasks for this service. The docker service ps command shows information about tasks associated with one or more services. A task is a slot where a container runs to execute a service's commands. A task can have different states in its lifecycle, such as new, running, complete, failed, etc. The docker service ps command shows all tasks by default, including historical ones. References: https://docs.docker.com/engine/reference/commandline/service_ps/, https://docs.docker.com/engine/swarm/how-swarm-mode-works/services/

NEW QUESTION: 33
During development of an application meant to be orchestrated by Kubernetes, you want to mount the /data directory on your laptop into a container.
Will this strategy successfully accomplish this?

Solution: Add a volume to the pod that sets hostPath.path: /data, and then mount this volume into the pod's containers as desired.

A. Yes

B. No

Answer: A (LEAVE A REPLY)

Explanation

Adding a volume to the pod that sets hostPath.path: /data, and then mounting this volume into the pod's containers as desired is a strategy that successfully accomplishes this. A hostPath volume mounts a file or directory from the host node's filesystem into a pod. It can be used to access files on the host from a container, such as configuration files, logs, binaries, etc. However, this type of volume is not portable across nodes and should be used with caution.

References: https://kubernetes.io/docs/concepts/storage/volumes/#hostpath, https://kubernetes.io/docs/tasks/configure-pod-container/configure-volume-storage/

NEW QUESTION: 34

Is this a function of UCP?

Solution: scans images to detect any security vulnerability

A. Yes

B. No

Answer: B (LEAVE A REPLY)

Explanation

Scanning images to detect any security vulnerability is not a function of UCP. UCP stands for Universal Control Plane, which is a web-based user interface for managing Docker Enterprise clusters and applications.

UCP does not provide image scanning capabilities, but it integrates with Docker Trusted Registry (DTR), which does offer image scanning as part of its security features. References: https://docs.docker.com/ee/ucp/, https://docs.docker.com/ee/dtr/user/manage-images/scan-images-for-vulnerabilities/

NEW QUESTION: 35

Is this an advantage of multi-stage builds?

Solution: better caching when building Docker images

A. Yes

B. No

Answer: B (LEAVE A REPLY)

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. References: https://docs.docker.com/develop/develop-images/multistage-build/, https://docs.docker.com/develop/develop-images/dockerfile_best-practices/#leverage-build-cache

NEW QUESTION: 36
Will this command ensure that overlay traffic between service tasks is encrypted?
Solution: docker network create -d overlay -o encrypted=true <network-name>
A. Yes
B. No
Answer: A (LEAVE A REPLY)
Explanation
This command will ensure that overlay traffic between service tasks is encrypted, because it uses the -o encrypted=true option to enable encryption for the overlay network. According to the official documentation, when you enable overlay encryption, Docker creates IPSEC tunnels between all the nodes where tasks are scheduled for services attached to the overlay network. These tunnels also use the AES algorithm in GCM mode and manager nodes automatically rotate the keys every 12 hours.
References: https://docs.docker.com/network/drivers/overlay/#encryption https://docs.docker.com/network/network-tutorial-overlay/#use-an-overlay-network-for-standalone-containers

NEW QUESTION: 37
Is this an advantage of multi-stage builds?
Solution: optimizes Images by copying artifacts selectively from previous stages
A. Yes
B. No
Answer: A (LEAVE A REPLY)
Explanation
Optimizing images by copying artifacts selectively from previous stages is an advantage of multi-stage builds.

Multi-stage builds allow you to use multiple FROM statements in your Dockerfile, each starting a new stage of the build. You can selectively copy artifacts from one stage to another, leaving behind everything you don't want in the final image. This reduces the size and complexity of your images, and improves security and performance. References:
https://docs.docker.com/build/building/multi-stage/,
https://docs.docker.com/engine/reference/builder/#copy

NEW QUESTION: 38
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'
A. Yes
B. No
Answer: B (LEAVE A REPLY)
Explanation
This command will not mount the host's '/data' directory to the ubuntu container in read-only mode, because it has several syntax errors and missing options. According to the official documentation, the correct command should be:
docker run --volume /data:/mydata:ro --read-only ubuntu
The errors and missing options are:
The --add-volume flag does not exist and should be replaced by --volume or -v to mount a host directory as a data volume.
The --volume flag requires a colon (:) to separate the host directory and the container directory, not a space.
The --read-only flag applies to the whole container's file system, not just the mounted volume. To make the volume read-only, a :ro suffix should be added to the --volume flag.
The ubuntu image name should be the last argument of the command.
References: https://docs.docker.com/engine/reference/commandline/run/#mount-volume-v-read-only
https://docs.docker.com/storage/volumes/#use-a-read-only-volume

NEW QUESTION: 39
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?
Solution: The controller managing the pod is autoscaled back to delete the unhealthy pod and alleviate load.
A. Yes
B. No
Answer: B (LEAVE A REPLY)
Explanation
The controller managing the pod is not autoscaled back to delete the unhealthy pod and alleviate load, because this is not how Kubernetes handles pod failures. According to the official

documentation, Kubernetes will try to maintain the desired number of pods for each controller, and will not scale down or up based on pod health.
References:
https://kubernetes.io/docs/concepts/workloads/controllers/replicationcontroller/#how-a-replicationcontroller-wor

NEW QUESTION: 40
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. environment variables
A. Yes
B. No
Answer: (SHOW ANSWER)
Explanation
Environment variables cannot be used to schedule containers to meet the security policy requirements.
Environment variables are key-value pairs that can be passed to containers when they are created or run.
Environment variables can be used to configure the behavior of the containerized application or provide runtime information, such as database credentials, API keys, etc. Environment variables do not affect how containers are scheduled on nodes in a swarm mode cluster. References:
https://docs.docker.com/engine/reference/commandline/run/#set-environment-variables-e-env-env-file,
https://docs.docker.com/engine/swarm/services/#create-a-service

NEW QUESTION: 41
Will this configuration achieve fault tolerance for managers in a swarm?
Solution: an odd number of manager nodes, totaling more than two
A. Yes
B. No
Answer: A (LEAVE A REPLY)
Explanation
This configuration will achieve fault tolerance for managers in a swarm, because an odd number of manager nodes allows for quorum-based consensus among managers and avoids split-brain scenarios. According to the official documentation, having more than two manager nodes ensures that there is always at least one manager available in case of failures.
References: https://docs.docker.com/engine/swarm/admin_guide/#add-manager-nodes-for-fault-tolerance

NEW QUESTION: 42

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

A. Yes

B. No

Answer: B (LEAVE A REPLY)

Explanation

This does not describe the role of Control Groups (cgroups) when used with a Docker container, because user authorization to the Docker API is not related to cgroups. According to the official documentation, cgroups are a Linux kernel feature that limits and isolates the resource usage of a group of processes, such as CPU, memory, disk I/O, network, etc. Docker can use cgroups to share available hardware resources to containers and optionally enforce limits and constraints.
References: https://docs.docker.com/config/containers/runmetrics/
https://bikramat.medium.com/namespace-vs-cgroup-60c832c6b8c8

NEW QUESTION: 43

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

A. Yes

B. No

Answer: B (LEAVE A REPLY)

Explanation

Disabling the Docker service via chkconfig or systemctl does not upgrade Docker Engine CE to Docker Engine EE. Disabling the Docker service only stops and prevents Docker from starting automatically on system boot. It does not change or upgrade the version of Docker Engine installed on the system. To upgrade Docker Engine CE to Docker Engine EE, you need to uninstall Docker Engine CE and install Docker Engine EE following the official instructions for your operating system. References:
https://docs.docker.com/engine/install/linux-postinstall/#disable-docker-from-starting-automatically-on-boot,
https://docs.docker.com/engine/install/centos/#uninstall-old-versions,
https://docs.docker.com/engine/install/ubuntu/#uninstall-old-versions,
https://docs.docker.com/engine/install/debian/#uninstall-old-versions,
https://docs.docker.com/ee/docker-ee/linux-install/

NEW QUESTION: 44

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

A. Yes

B. No

Answer: (SHOW ANSWER)

Explanation
net is not a Linux kernel namespace that is disabled by default and must be enabled at Docker engine runtime to be used. According to the official documentation, net is one of the namespaces that are enabled by default when using namespaces for isolation.
References: https://docs.docker.com/engine/security/userns-remap/#user-namespace-known-limitations

NEW QUESTION: 45
You want to provide a configuration file to a container at runtime. Does this set of Kubernetes tools and steps accomplish this?
Solution: Mount the configuration file directly into the appropriate pod and container using the .spec.containers.configMounts key.
A. Yes
B. No
Answer: B (LEAVE A REPLY)
Explanation
This set of Kubernetes tools and steps does not accomplish this, because there is no such key as
.spec.containers.configMounts in the pod specification. According to the official documentation, the correct key to use for mounting a configuration file directly into a container
is .spec.containers.volumeMounts, which requires a corresponding volume definition
in .spec.volumes.
References:
https://kubernetes.io/docs/tasks/configure-pod-container/configure-pod-configmap/#add-configmap-data-to-a-vo
https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#container-v1-core

NEW QUESTION: 46
Is this a type of Linux kernel namespace that provides container isolation?
Solution. Process ID
A. Yes
B. No
Answer: A (LEAVE A REPLY)
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. References:

https://docs.docker.com/engine/reference/run/#pid-settings-pid,
https://en.wikipedia.org/wiki/Linux_namespaces#Process_ID_(pid)

Valid DCA Dumps shared by Actual4test.com for Helping Passing DCA Exam!
Actual4test.com now offer the **newest DCA exam dumps**, the Actual4test.com DCA exam **questions have been updated** and **answers have been corrected** get the **newest** Actual4test.com DCA dumps with Test Engine here:
https://www.actual4test.com/DCA_examcollection.html (**189** Q&As Dumps, **30%OFF** **Special Discount:** **Freepdfdumps**)

NEW QUESTION: 47
During development of an application meant to be orchestrated by Kubernetes, you want to mount the /data directory on your laptop into a container.
Will this strategy successfully accomplish this?
Solution: Create a PersistentVolume with storageciass: "" and hostPath: /data, and a persistentVolumeClaim requesting this PV. Then use that PVC to populate a volume in a pod
A. Yes
B. No
Answer: B (LEAVE A REPLY)
Explanation
Creating a PersistentVolume with storageClass: "" and hostPath: /data, and a persistentVolumeClaim requesting this PV, and then using that PVC to populate a volume in a pod is not a strategy that successfully accomplishes this. A PersistentVolume is an API object that represents a piece of storage in the cluster that has been provisioned by an administrator. A PersistentVolumeClaim is a request for storage by a user. A hostPath PersistentVolume uses a file or directory on the node to emulate network-attached storage. However, this type of volume only works on a single node cluster, and it does not work on Windows nodes. Therefore, it cannot be used to mount the /data directory on your laptop into a container. References:
https://kubernetes.io/docs/concepts/storage/persistent-volumes/,
https://kubernetes.io/docs/concepts/storage/volumes/#hostpath

NEW QUESTION: 48
The Kubernetes yaml shown below describes a clusterIP service.

Is this a correct statement about how this service routes requests?
Solution: 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.

A. Yes

B. No

Answer: B (LEAVE A REPLY)

Explanation

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 is not a correct statement about how this service routes requests. A clusterIP service does not route requests based on the pod IP, but on the service IP. A clusterIP service also does not forward requests from one port on the pod IP to another port on the same pod IP, but from one port on the service IP to another port on the pod IP. Therefore, this statement is incorrect and does not describe how this service routes requests. References: https://kubernetes.io/docs/concepts/services-networking/service/#defining-a-service, https://kubernetes.io/docs/concepts/services-networking/service/#virtual-ips-and-service-proxies

NEW QUESTION: 49

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

A. Yes

B. No

Answer: (SHOW ANSWER)

Explanation

This statement is correct. A Dockerfile provides instructions for building a Docker image. A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.
Using docker build, you can create an automated build that executes several command-line instructions in succession. References: https://docs.docker.com/engine/reference/builder/, https://docs.docker.com/engine/reference/commandline/build/

NEW QUESTION: 50

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
A. Yes
B. No
Answer: (SHOW ANSWER)
Explanation
Deleting the image and deleting the image repository from Docker Trusted Registry will not completely delete the image from disk. According to the official documentation, you also need to run garbage collection on the Docker Trusted Registry to reclaim disk space.
References: https://docs.docker.com/ee/dtr/admin/manage-images/garbage-collection/

NEW QUESTION: 51
Does this command create a swarm service that only listens on port 53 using the UDP protocol?
Solution. 'docker service create -name dns-cache -p 53:53 -udp dns-cache'
A. Yes
B. No
Answer: (SHOW ANSWER)
Explanation
This command does create a swarm service that only listens on port 53 using the UDP protocol. The docker service create command creates a new service in a swarm cluster. The --name flag sets the name of the service.
The -p or --publish flag publishes a port or a range of ports to the swarm. The syntax for publishing a port is -p
<published-port>:<target-port>[/protocol]. The protocol can be tcp or udp. If no protocol is specified, tcp is used by default. In this case, the command publishes port 53 on the swarm and maps it to port 53 in the container using the udp protocol. Therefore, this command creates a swarm service that only listens on port 53 using the udp protocol. References: https://docs.docker.com/engine/reference/commandline/service_create/, https://docs.docker.com/engine/reference/commandline/service_create/#publish-service-ports-externally-to-the-s

NEW QUESTION: 52
The Kubernetes yaml shown below describes a networkPolicy.

Will the networkPolicy BLOCK this traffic?

Solution: a request issued from a pod lacking the tier: api label, to a pod bearing the tier: backend label

A. Yes

B. No

Answer: B (LEAVE A REPLY)

Explanation

The networkPolicy will not block this traffic because it does not apply to pods that have the tier: backend label. The networkPolicy only applies to pods that have the tier: frontend label, as specified by the podSelector field. Pods that have the tier: backend label are not affected by this networkPolicy and can receive traffic from any source.

References:

https://kubernetes.io/docs/concepts/services-networking/network-policies/#the-networkpolicy-resource

NEW QUESTION: 53

You are troubleshooting a Kubernetes deployment called api, and want to see the events table for this object.

Does this command display it?

Solution: kubectl logs deployment api

A. Yes

B. No

Answer: (SHOW ANSWER)

Explanation

Using kubectl logs deployment api does not display the events table for this object. The kubectl logs command shows the logs of a pod or a container in a pod, but it does not show the events related to the deployment object. To see the events table for this object, you need to use kubectl describe deployment api. References:

https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#logs,

https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#describe

NEW QUESTION: 54
The following Docker Compose file is deployed as a stack:

freepdfdumps.com

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.
A. Yes
B. No
Answer: A (LEAVE A REPLY)
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.
References:
https://docs.docker.com/engine/reference/builder/#healthcheck,
https://docs.docker.com/engine/reference/run/#healthcheck

NEW QUESTION: 55
Are these conditions sufficient for Kubernetes to dynamically provision a persistentVolume, assuming there are no limitations on the amount and type of available external storage?
Solution: A default storageClass is specified, and subsequently a persistentVolumeClaim is created.
A. Yes
B. No
Answer: A (LEAVE A REPLY)
Explanation

These conditions are sufficient for Kubernetes to dynamically provision a persistentVolume, because a default storageClass provides the information needed to create a persistentVolume that matches the persistentVolumeClaim. According to the official documentation, if no storageClass is specified or requested, then the default one is used.
References: https://kubernetes.io/docs/concepts/storage/dynamic-provisioning/#defaulting-behavior

NEW QUESTION: 56
Will this action upgrade Docker Engine CE to Docker Engine EE?
Solution: Uninstall 'docker-ce' package before installing 'docker-ee' package.
A. Yes
B. No
Answer: A (LEAVE A REPLY)
Explanation
Uninstalling 'docker-ce' package before installing 'docker-ee' package does upgrade Docker Engine CE to Docker Engine EE. Docker Engine CE is the free and open source edition of Docker Engine, while Docker Engine EE is the enterprise-ready edition that includes additional features and support. To upgrade from CE to EE, you need to uninstall the 'docker-ce' package and its dependencies, and then install the 'docker-ee' package from the Docker repository.
References:
https://docs.docker.com/engine/install/centos/#upgrade-docker-after-using-the-convenience-script,
https://docs.docker.com/engine/install/centos/#install-using-the-repository

NEW QUESTION: 57
Will a DTR security scan detect this?
Solution: licenses for known third party binary components
A. Yes
B. No
Answer: B (LEAVE A REPLY)

NEW QUESTION: 58
Will this command list all nodes in a swarm cluster from the command line?
Solution: 'docker ls -a'
A. Yes
B. No
Answer: B (LEAVE A REPLY)
Explanation
Using 'docker ls -a' does not list all nodes in a swarm cluster from the command line. The docker ls command is not a valid command. To list containers, you need to use 'docker container ls' or

'docker ps'. To list images, you need to use 'docker image ls' or 'docker images'. To list nodes in a swarm cluster, you need to use
'docker node ls'. References:
https://docs.docker.com/engine/reference/commandline/container_ls/,
https://docs.docker.com/engine/reference/commandline/image_ls/,
https://docs.docker.com/engine/reference/commandline/node_ls/

NEW QUESTION: 59
Is this a Linux kernel namespace that is disabled by default and must be enabled at Docker engine runtime to be used?
Solution. pid
A. Yes
B. No
Answer: (SHOW ANSWER)
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. References:
https://docs.docker.com/engine/reference/run/#pid-settings-pid,
https://en.wikipedia.org/wiki/Linux_namespaces#Process_ID_(pid)

NEW QUESTION: 60
Your organization has a centralized logging solution, such as Splunk.
Will this configure a Docker container to export container logs to the logging solution?
Solution: docker system events --filter splunk
A. Yes
B. No
Answer: B (LEAVE A REPLY)
Explanation
Using docker system events --filter splunk does not configure a Docker container to export container logs to the logging solution. The docker system events command shows real-time events from the server, such as container creation, image deletion, network connection, etc. It does not show or export container logs to any external service. The --filter option allows filtering events by type, action, image, container, etc., but it does not accept splunk as a valid filter value. References:
https://docs.docker.com/engine/reference/commandline/system_events/,

https://docs.docker.com/config/containers/logging/

NEW QUESTION: 61
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: node taints
A. Yes
B. No
Answer: B (LEAVE A REPLY)
Explanation
Node taints cannot be used to schedule containers to meet the security policy requirements, because node taints are a Kubernetes concept and not a Swarm concept. According to the official documentation, node taints are used to mark nodes with certain attributes that prevent pods from being scheduled on them unless they have matching tolerations.
References: https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration/

Valid DCA Dumps shared by Actual4test.com for Helping Passing DCA Exam!
Actual4test.com now offer the **newest DCA exam dumps**, the Actual4test.com DCA exam **questions have been updated** and **answers have been corrected** get the **newest** Actual4test.com DCA dumps with Test Engine here:
https://www.actual4test.com/DCA_examcollection.html (**189** Q&As Dumps, **30%OFF** **Special Discount:** **Freepdfdumps**)

NEW QUESTION: 62
In the context of a swarm mode cluster, does this describe a node?
Solution: an instance of the Docker engine participating in the swarm
A. No
B. Yes
Answer: B (LEAVE A REPLY)

NEW QUESTION: 63
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"
A. Yes
B. No
Answer: (SHOW ANSWER)
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.
References:
https://docs.docker.com/engine/reference/commandline/inspect/,
https://docs.docker.com/engine/reference/commandline/service_ps/

NEW QUESTION: 64
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 contraints
A. Yes
B. No
Answer: A (LEAVE A REPLY)
Explanation
Label constraints can be used to schedule containers to meet the security policy requirements, because label constraints allow you to specify which nodes a service can run on based on node labels. According to the official documentation, label constraints are used to filter nodes based on their role, availability, or any other criteria.
References: https://docs.docker.com/engine/swarm/services/#specify-service-constraints

NEW QUESTION: 65
You want to create a container that is reachable from its host's network.
Does this action accomplish this?
Solution. Use either EXPOSE or -publish to access the container on the bridge network.
A. Yes
B. No
Answer: (SHOW ANSWER)
Explanation
Using either EXPOSE or --publish to access the container on the bridge network does not create a container that is reachable from its host's network. EXPOSE and --publish are options that specify which ports on the container should be exposed or published to the outside world. They do not affect which network the container is connected to. By default, Docker creates and connects containers to a bridge network, which is an internal network that isolates containers from each other and from the host. To create a container that is reachable from its host's network, you need to use --network host option, which connects the container to the host's network stack. References: https://docs.docker.com/engine/reference/builder/#expose, https://docs.docker.com/engine/reference/run/#expose-incoming-ports, https://docs.docker.com/network/bridge/, https://docs.docker.com/network/host/

Valid DCA Dumps shared by Actual4test.com for Helping Passing DCA Exam! Actual4test.com now offer the **newest DCA exam dumps**, the Actual4test.com DCA exam **questions have been updated** and **answers have been corrected** get the **newest** Actual4test.com DCA dumps with Test Engine here: https://www.actual4test.com/DCA_examcollection.html (**189** Q&As Dumps, **30%OFF** **Special Discount:** **Freepdfdumps**)