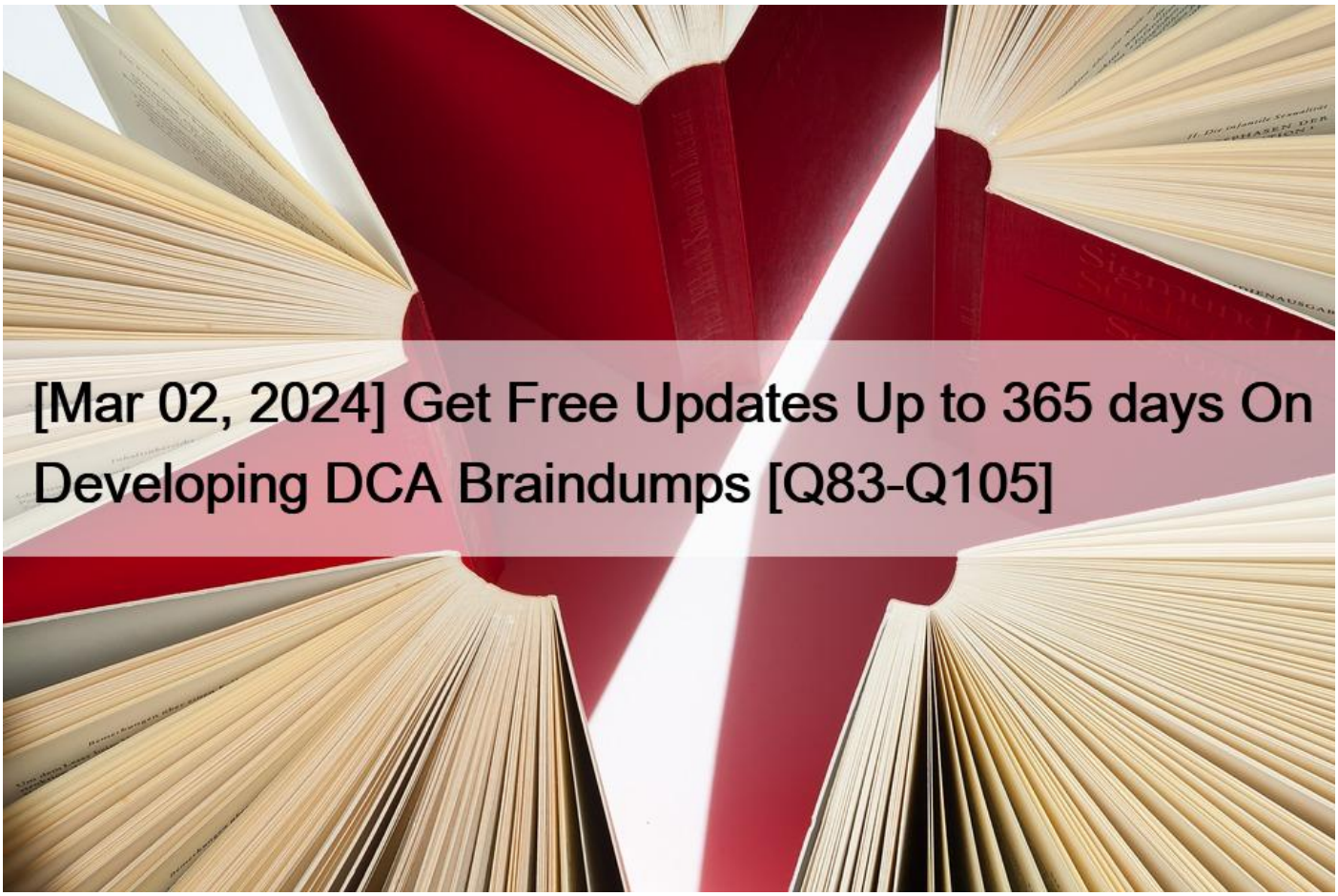
[Mar 02, 2024 Get Free Updates Up to 365 days On Developing DCA Braindumps [Q83-Q105



[Mar 02, 2024] Get Free Updates Up to 365 days On Developing DCA Braindumps
Best Quality Docker DCA Exam Questions

Docker Certified Associate (DCA) exam is a certification program offered by Docker that tests the skills and knowledge required to effectively use the Docker platform. DCA exam is designed to validate the competency of IT professionals in deploying, managing, and scaling applications using Docker containers. The DCA certification is an industry-standard credential that is recognized globally, and it is highly sought after by IT professionals who work with Docker.

Docker Certified Associate (DCA) exam is a valuable certification for IT professionals who want to demonstrate their skills and knowledge in Docker technology. Docker Certified Associate (DCA) Exam certification program is recognized by industry leaders and can help individuals advance their careers and increase their earning potential. By passing the DCA exam, IT professionals can show that they are proficient in Docker technology and are capable of working with Docker technology in real-world scenarios.

NO.83 Is this statement correct?

Solution: A Dockerfile provides instructions for building a Docker image
* Yes
* No

NO.84 During development of an application meant to be orchestrated by Kubemetes, you want to mount the /data directory on your laptop into a container.

Will this strategy successfully accomplish this?

Solution. Create a Persistent VolumeClaim requesting storageClass:”” (which defaults to local storage) and hostPath: /data, and use this to populate a volume in a pod.
* Yes
* No
Explanation

= This strategy will not successfully accomplish this. A PersistentVolumeClaim (PVC) is a request for storage by a user that is automatically bound to a suitable PersistentVolume (PV) by Kubernetes1. A PV is a piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using StorageClasses1. A hostPath is a type of volume that mounts a file or directory from the host node’s filesystem into a pod2. It is mainly used for development and testing on a single-node cluster, and not recommended for production use2.

The problem with this strategy is that it assumes that the hostPath /data on the node is the same as the /data directory on your laptop. This is not necessarily true, as the node may be a different machine than your laptop, or it may have a different filesystem layout. Also, the hostPath volume is not portable across nodes, so if your pod is scheduledon a different node, it will not have access to the same /data directory2. Furthermore, the storageClass parameter is not applicable for hostPath volumes, as they are not dynamically provisioned3.

To mount the /data directory on your laptop into a container, you need to use a different type of volume that supports remote access, such as NFS, Ceph, or GlusterFS4. You also need to make sure that your laptop is accessible from the cluster network and that it has the appropriate permissions to share the /data directory. Alternatively, you can use a tool like Skaffold or Telepresence to sync your local files with your cluster56. References:

* Persistent Volumes | Kubernetes

* Volumes | Kubernetes

* Storage Classes | Kubernetes

* Kubernetes Storage Options | Kubernetes Academy

* Skaffold | Easy and Repeatable Kubernetes Development

* Telepresence: fast, local development for Kubernetes and OpenShift microservices

NO.85 What is the docker command to setup a swarm?
* docker swarm init
* docker swarm create
* docker init swarm
* docker create swarm

https://docs.docker.com/engine/reference/commandline/swarm/

NO.86 Is this statement correct?

Solution. A Dockerfile stores persistent data between deployments of a container
* Yes
* No
Explanation

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

https://docs.docker.com/storage/

NO.87 Can this set of commands identify the published port(s) for a container?

Solution: docker container inspect’, ‘docker port’
* Yes
* No

NO.88 In the context of a swarm mode cluster, does this describe a node?

Solution: a virtual machine participating in the swarm
* Yes
* No
Explanation

A virtual machine participating in the swarm is a node in the context of a swarm mode cluster. A node is an instance of the Docker engine participating in the swarm. A node can be either a physical machine or a virtual machine. Nodes are either managers or workers. Managers maintain cluster state and manage cluster tasks.

Workers execute tasks assigned by managers. References:

https://docs.docker.com/engine/swarm/key-concepts/#nodes-and-services,

https://docs.docker.com/engine/swarm/how-swarm-mode-works/nodes/

NO.89 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 containers on the bridge network
* Yes
* No

NO.90 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. Mirror the engineering/api repository to one of the user’s own private repositories.

* Yes
* No

Explanation

Mirroring the engineering/api repository to one of the user’s own private repositories does not grant them read/write access to the engineering/api repository. Mirroring is a feature that allows you to automatically replicate images from one repository to another, either within the same DTR or across different DTRs.

Mirroring does not change the permissions or access levels of the source or destination repositories. It only copies the images and tags from one repository to another. To grant a user read/write access to the engineering/api repository, you need to add them as a collaborator with read/write role on that repository, or add them to a team that has read/write role on that repository. References:

https://docs.docker.com/ee/dtr/user/manage-images/mirror-repository-images/,

https://docs.docker.com/ee/dtr/user/manage-repositories/set-repository-permissions/

NO.91 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

* Yes
* No

Explanation

Label constraints can be used to schedule containers to meet the security policy requirements. Label constraints are a way to specify which nodes a service can run on based on the labels assigned to the nodes.

Labels are key-value pairs that can be attached to any node in the swarm. For example, you can label nodes as development or production depending on their intended use. Then, you can use the –constraint option when creating or updating a service to filter the nodes based on their labels. For example, to run a service only on development nodes, you can use:

docker service create –constraint’node.labels.environment == development’…

To run a service only on production nodes, you can use:

docker service create –constraint’node.labels.environment == production’…

This way, you can ensure that development and production containers run on separate nodes in the swarm, as required by the security policy. References:

* Using placement constraints with Docker Swarm

* Multiple label placement constraints in docker swarm

* Machine constraints in Docker swarm

* How can set service constraint to multiple value

NO.92 Will this configuration achieve fault tolerance for managers in a swarm?

Solution: an odd number of manager nodes, totaling more than two
* Yes
* No

NO.93 What is the purpose of multi-stage builds?
* Better logical separation of Dockerfile instructions for better readability
* Optimizing images by copying artifacts selectively from previous stages
* Better caching when building Docker images
* Faster image builds by allowing parallel execution of Docker builds

NO.94 Which ‘docker run’ flag lifts cgroup limitations?
* ‘docker run –isolation’
* ‘docker run –cap-drop’
* ‘docker run –privileged’
* ‘docker run –cpu-period’

NO.95 Is this a way to configure the Docker engine to use a registry without a trusted TLS certificate?

Solution. Set INSECURE_REGISTRY in the’ /etc/docker/default’ configuration file.
* Yes
* No

Explanation

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 INSECURE_REGISTRY option allows you to specify one or more registries that do not have valid TLS certificates or use HTTP instead of HTTPS. This option bypasses the TLS verification for these registries and allows Docker to pull and push images from them without errors. However, this option is not recommended for production use as it exposes your registry communication to potential security risks. References: https://docs.docker.com/registry/insecure/,

https://docs.docker.com/engine/reference/commandline/dockerd/#insecure-registries

NO.96 Will this command list all nodes in a swarm cluster from the command line?

Solution: ‘docker node Is’
* Yes
* No

NO.97 Is this a supported user authentication method for Universal Control Plane?

Solution. SAML
* Yes
* No

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/

NO.98 Will a DTR security scan detect this?

Solution.image configuration poor practices, such as exposed ports or inclusion of compilers in production images
* Yes
* No
Explanation

A DTR security scan will not detect image configuration poor practices, such as exposed ports or inclusion of compilers in production images. A DTR security scan is designed to discover vulnerabilities in the images based on the MITRE CVE or NIST NVD databases1. It does not check the image configuration or best practices. To check the image configuration and best practices, you can use other tools, such as Dockerfile Linter) or Docker Bench for Security). References: Vulnerability scanning must be enabled for all repositories in the Docker Trusted Registry (DTR) component of Docker Enterprise), Dockerfile Linter), Docker Bench for Security)

NO.99 Will this configuration achieve fault tolerance for managers in a swarm?

Solution: an odd number of manager nodes, totaling more than two
* Yes
* No
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

NO.100 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 inspect http’
* Yes
* No
Explanation

Using ‘docker service inspect http’ does not enable you to view the list of historical tasks for this service. The docker service inspect command shows low-level information about one or more services, such as their configuration, replicas, networks, endpoints, etc. It does not show the history of tasks that have been run by the service. To view the list of historical tasks for this service, you need to use ‘docker service ps http’.

References: https://docs.docker.com/engine/reference/commandline/service_inspect/,

https://docs.docker.com/engine/reference/commandline/service_ps/

NO.101 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
* Yes
* No
Explanation

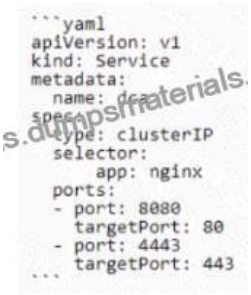
his is not a way to accomplish this, because creating one pod and adding all the resources needed for each application is not a good practice for deploying applications in Kubernetes. According to the official documentation, pods are not intended to run multiple instances of an application or different 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

NO.102 You are running only Kubernetes workloads on a worker node that requires maintenance, such as installing patches or an OS upgrade

Which command must be run on the node to gracefully terminate all pods on the node, while marking the node as unschedulable?
* ‘docker node update –availability drain <node name>’
* ‘docker swarm leave’
* ‘kubectl drain <node name>’
* ‘kubectl cordon <node name>’

NO.103 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 this service on port 8080 will be routed to port 80 in a random pod with the label aPP: nginx.
* Yes
* No
Explanation

Traffic sent to the IP of this service on port 8080 will be routed to port 80 in a random pod with the label app:

nginx is a correct statement about how this service routes requests. A clusterIP service is a type of service that exposes an internal IP address that is only reachable within the cluster. A clusterIP service routes requests to pods based on their labels and selectors. In this case, the service has a selector app: nginx, which means it will route requests to any pod that has the label app: nginx. The service also has a port mapping from 8080 to 80, which means it will forward requests from port 8080 on the service IP to port 80 on

the pod IP. The service will use a round-robin algorithm to distribute requests among the pods that match the selector. 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

NO.104 Will this command list all nodes in a swarm cluster from the command line?

Solution: ‘docker node Is’
* Yes
* No
Explanation

= (Please check the official Docker site for the comprehensive explanation) References: (Some possible references from the web search results are)

* [Docker Swarm Tutorial: Manage Multiple Docker Hosts – Edureka]

* [Docker Swarm – Docker Documentation]

* [Docker Swarm Tutorial: How to Manage Multiple Docker Hosts – Linux Hint]

* [Docker Swarm Tutorial: How to Manage Multiple Docker Hosts – YouTube]

* [Docker Swarm Tutorial: How to Manage Multiple Docker Hosts – Medium] I hope this helps you in your exam preparation. Good luck!

NO.105 Which of the following is true about using the ‘-P’ option when creating a new container?
* Docker binds each exposed container port to a random port on all the host’s interface
* Docker gives extended privileges to the container.
* Docker binds each exposed container port to a random port on a specified host interface
* Docker binds each exposed container port with the same port on the host

Docker Exam Practice Test To Gain Brilliante Result: https://www.dumpsmaterials.com/DCA-real-torrent.html]