Q. What are the differences between the ‘docker run’ and the docker create’?

The most important difference that can be noted is that, by using the ‘docker create’ command we can create a Docker container in the Stopped state. We can also provide it with an ID that can be stored for later usages as well.  
This can be achieved by using the command ‘docker run’ with the option –cidfile FILE\_NAME as like this:  
‘docker run –cidfile FILE\_NAME’

Q. What is Dockerfile used for?

Dockerfile is nothing but a set of instructions that have to be passed on to Docker itself, so that it can build images automatically reading these instructions from that specified Dockerfile. 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 users can create an automated build that executes several command-line instructions in succession.

Q. Can I use JSON instead of YAML for my compose file in Docker?

YES, you can very comfortably use JSON instead of the default YAML for your Docker compose file. In order to use JSON file with compose, you need to specify the filename to use as the following:  
docker-compose -f docker-compose.json up

Q. Tell us how you have used Docker in your past position?

This is a question that you could bring upon your whole experience with Docker and if you have used any other Container technologies before Docker. You could also explain the ease that this technology has brought in the automation of the development to production lifecycle management. You can also discuss about any other integrations that you might have worked along with Docker such as Puppet, Chef or even the most popular of all technologies – Jenkins. If you do not have any experience with Docker itself but similar tools from this space, you could convey the same and also show in your interest towards learning this leading containerization technology.

Q. How to create Docker container?

You can create a Docker container out of any specific Docker image of your choice and the same can be achieved using the command given below:  
docker run -t -i command name  
The command above will create the container and also starts it for you. In order to check whether the Docker container is created and whether it is running or not, you could make use of the following command. This command will list out all the Docker containers along with its statuses on the host that the Docker container runs.  
docker ps -a

Q. How to stop and restart the Docker container?

The following command can be used to stop a certain Docker container with the container id as

CONTAINER\_ID:  
docker stop CONTAINER\_ID  
The following command can be used to restart a certain Docker container with the container id as

CONTAINER\_ID:  
docker restart CONTAINER\_ID

Q. What is the preferred way of removing containers - ‘docker rm -f’ or ‘docker stop’ then followed by a ‘docker rm’?

The best and the preferred way of removing containers from Docker is to use the ‘docker stop’, as it will allow sending a SIG\_HUP signal to its recipients giving them the time that is required to perform all the finalization and cleanup tasks. Once this activity is completed, we can then comfortably remove the container using the ‘docker rm’ command from Docker and thereby updating the docker registry as well.

Q. Difference between Docker Image and container?

Docker container is the runtime instance of docker image.

Docker Image doesnot have a state and its state never changes as it is just set of files whereas docker container has its execution state.

Q. Why do my services take 10 seconds to recreate or stop?

Docker compose stop will attempt to stop a specific Docker container by sending a SIGTERM message. Once this message is delivered, it waits for the default timeout period of 10 seconds and once the timeout period is crossed, it then sends out a SIGKILL message to the container – in order to kill it forcefully. If you are actually waiting for the timeout period, then it means that the containers are not shutting down on receiving SIGTERM signals / messages.

In an attempt to solve this issue, the following is what you can do:

• You can ensure that you are using the JSON form of the CMD and also the ENTRYPOINT in your dockerfile.  
• Use [“program”, “argument1”, “argument2”] instead of sending it as a plain string as like this – “program argument1 argument2”.  
• Using the string form, makes Docker run the process using bash that can’t handle signals properly. Compose always uses the JSON form.  
• If it is possible then modify the application which you intend to run by adding an explicit signal handler for the SIGTERM signal  
• Also set the stop\_signal to a proper signal that the application can understand and also know how to handle it

Q. How do I run multiple copies of a Compose file on the same host?

Docker’s compose makes use of the Project name to create unique identifiers for all of the project’s containers and resources. In order to run multiple copies of the same project, you will need to set a custom project name using the –p command line option or you could use the COMPOSE\_PROJECT\_NAME environment variable for this purpose.

Q. What’s the difference between up, run, and start?

On any given scenario, you would always want your docker-compose up. Using the command UP, you can start or restart all the services that are defined in a docker-compose.yml file. In the “attached” mode, which is also the default mode – we will be able to see all the log files from all the containers. In the “detached” mode, it exits after starting all the containers, which continue to run in the background showing nothing over in the foreground.

Using docker-compose run command, we will be able to run the one-off or the ad-hoc tasks that are required to be run as per the Business needs and requirements. This requires the service name to be provided which you would want to run and based on that, it will only start those containers for the services that the running service depends on. Using the run command, you can run your tests or perform any of the administrative tasks as like removing / adding data to the data volume container. It is also very similar to the docker run –ti command, which opens up an interactive terminal to the containers an exit status that matches with the exit status of the process in the container.

Using the docker-compose start command, you can only restart the containers that were previously created and were stopped. This command never creates any new Docker containers on its own.

Q. What’s the benefit of “Dockerizing?”

Dockerizing enterprise environments helps teams to leverage over the Docker containers to form a service platform as like a CaaS (Container as a Service). It gives teams that necessary agility, portability and also lets them control staying within their own network / environment.

Most of the developers opt to use Docker and Docker alone because of the flexibility and also the ability that it provides to quickly build and ship applications to the rest of the world. Docker containers are portable and these can run on any environment without making any additional changes when the application developers have to move between Developer, Staging and Production environments. This whole process is seamlessly implemented without the need of performing any recoding activities for any of the environments. These not only helps reduce the time between these lifecycle states, but also ensures that the whole process is performed with utmost efficiency. There is every possibility for the Developers to debug any certain issue, fix it and also update the application with it and propagate this fix to the higher environments with utmost ease.

The operations teams can handle the security of the environments while also allowing the developers build and ship the applications in an independent manner. The CaaS platform that is provided by Docker framework can deploy on-premise and is also loaded with full of enterprise level security features such as role-based access control, integration with LDAP or any Active Directory, image signing and etc. Operations teams have heavily rely on the scalability provided by Docker and can also leverage over the Dockerized applications across any environments.

Docker containers are so portable that it allows teams to migrate workloads that run on an Amazon’s AWS environment to Microsoft Azure without even having to change its code and also with no downtime at all. Docker allows teams to migrate these workloads from their cloud environments to their physical datacenters and vice versa. This also enables the Organizations to focus on the infrastructure from the gained advantages both monetarily and also the self-reliability over Docker. The lightweight nature of Docker containers compared to traditional tools like virtualization, combined with the ability for Docker containers to run within VMs, allowing teams to optimize their infrastructure by 20X, and save money in the process.

Q. How many containers can run per host?

Depending on the environment where Docker is going to host the containers, there can be as many containers as the environment supports. The application size, available resources (like CPU, memory) will decide on the number of containers that can run on an environment. Though containers create newer CPU on their own but they can definitely provide efficient ways of utilizing the resources. The containers themselves are super lightweight and only last as long as the process they are running.

Q. Is there a possibility to include specific code with COPY/ADD or a volume?

This can be easily achieved by adding either the COPY or the ADD directives in your dockerfile. This will count to be useful if you want to move your code along with any of your Docker images, example, sending your code an environment up the ladder – Development environment to the Staging environment or from the Staging environment to the Production environment.

Having said that, you might come across situations where you’ll need to use both the approaches. You can have the image include the code using a COPY, and use a volume in your Compose file to include the code from the host during development. The volume overrides the directory contents of the image.

Q. Will cloud automation overtake containerization any sooner?

Docker containers are gaining the popularity each passing day and definitely will be a quintessential part of any professional Continuous Integration / Continuous Development pipelines. Having said that there is equal responsibility on all the key stakeholders at each Organization to take up the challenge of weighing the risks and gains on adopting technologies that are budding up on a daily basis. In my humble opinion, Docker will be extremely effective in Organizations that appreciate the consequences of Containerization.

Q. Is there a way to identify the status of a Docker container?

We can identify the status of a Docker container by running the command ‘docker ps –a’, which will in turn list down all the available docker containers with its corresponding statuses on the host. From there we can easily identify the container of interest to check its status correspondingly.

Q. Do I lose my data when the Docker container exits?

There is no loss of data when any of your Docker containers exits as any of the data that your application writes to the disk in order to preserve it. This will be done until the container is explicitly deleted. The file system for the Docker container persists even after the Docker container is halted.