**Docker Assignment 6**

1. What is the .dockerignore file's purpose?

* Similar to a .gitignore file, we also have a Dockerignore files which allows you to mention a list of files and/or directories which you might want to ignore while building the image. This would definitely reduce the size of the image and also help to speed up the docker build process.
* Before the docker CLI sends the context to the docker daemon, it looks for a file named .dockerignore in the root directory of the context. If this file exists, the CLI modifies the context to exclude files and directories that match patterns in it. This helps to avoid unnecessarily sending large or sensitive files and directories to the daemon and potentially adding them to images using ADD or COPY.

1. How necessary is it to create a cache in Docker?

* Docker will cache the results of the first build of a Dockerfile, allowing subsequent builds to be super fast.
* Every time we run command in the Dockerfile, Docker will create and commit a new layer to the image, which is just a set of tightly-coupled directories full of various file structure that comprise a Docker image.
* During a new build, all of these file structures have to be created and written to disk this is where Docker stores base images. Once created, the container (and subsequent new ones) will be stored in the folder in this same area.
* If the objects on the file system that Docker is about to produce are unchanged between builds, reusing a cache of a previous build on the host is a great time-saver.
* It makes building a new container really, really fast. None of those file structures have to be created and written to disk this time the reference to them is sufficient to locate and reuse the previously built structure

1. What is the significance of docker monitoring?

* Monitoring your containers in real time is essential to ensure peak app performance. When it comes to Docker containers, however, monitoring helps you to:
* Detect and solve issues early and proactively to avoid risks in production
* Implement changes safely as the entire environment is monitored
* Fine-tune applications to deliver improved performance and better user experience
* Optimize resource allocation

1. Differences between Windows and Hyper-V Containers/

* In the same way that Linux containers share the host operating system kernel files, Windows Server Containers make use of this sharing in order to make the containers efficient.
* if you were to log into the host operating system on your container server, you would be able to see the running processes of each container. The container is not able to see the host or other containers, and is still isolated from the host in various ways, but knowing that the host is able to view the processes within the container shows us that some interaction does exist with this level of sharing. Windows Server Containers are going to be most useful in circumstances where your container host server and the containers themselves are within the same trust boundary.
* If you trust both your host server and your containers, and are okay with those entities trusting each other, deploying regular Windows Server Containers is the most efficient use of your hardware resources.
* If you’re looking for **an increased amount of isolation** and **stronger boundaries**, that is where you will foray into Hyper-V Containers. Hyper-V Containers are more like a super-optimized version of a virtual machine. While kernel resources are still shared by Hyper-V Containers, so they are still much more performant than full virtual machines, each Hyper-V Container gets its own dedicated Windows shell within which a single container can run. This means you have isolation between Hyper-V Containers that is more on par with isolation between VMs

1. How do I use docker-compose to declare default environment variables?

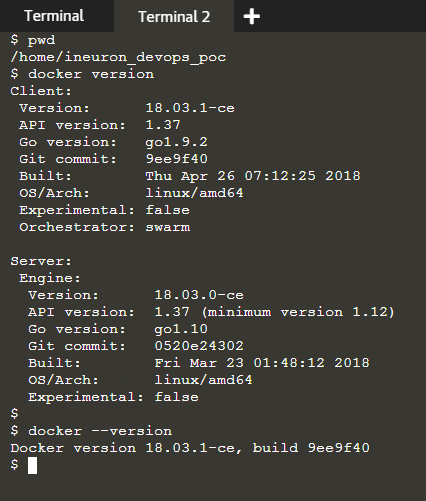
* Compose supports declaring default environment variables in an environment file named .env placed in the project directory. Docker Compose versions earlier than 1.28, load the .env file from the current working directory, where the command is executed, or from the project directory if this is explicitly set with the --project-directory option. This inconsistency has been addressed starting with +v1.28 by limiting the default .env file path to the project directory. You can use the --env-file commandline option to override the default .env and specify the path to a custom environment file.

1. What is the trusted registry for Docker?

* Docker Trusted Registry is the enterprise-grade image storage toll for Docker. You should install it after your firewall so that you can securely manage the Docker images you use in your applications.
* Docker Trusted Registry is an on-premises registry that allows enterprises to store and manage their Docker images on-premise or in their virtual private cloud (VPC) to meet security or regulatory compliance requirements.

1. How do you determine the version of the Docker client and server?

use command **docker version**



1. Can you tell the difference between the COPY and ADD commands in a Dockerfile?

* In Dockerfile it provides two ways to copy files from the source system into an image: the COPY and ADD
* COPY and ADD are both Dockerfile instructions that are almost same purposes. with this we can copy files from a specific location into a Docker image.
* COPY takes in a source and destination. It only allows you copy in a local or directory from your host (the machine building the Docker image) into the Docker image itself.
* ADD does that same job, but it also supports 2 other sources.
* First you can use a **URL** instead of a **local file/directory**.

1. Is it possible for a container to restart on its own?

* No By default the flag is set to false, but yes if we configure then yes restart is possible, docker provides restart policies after they exit. Docker policy will make sure that it started in correct order. Docker recommends that you use restart policies, and avoid using process managers to start containers.

1. What is the volume parameter in a docker run command used for?

* The syntax of docker run when using the volumes is:

**docker run -v host\_path:docker\_path <container\_name>**

* The volume parameter is used for syncing a directory of a container with any of the host directories. Consider the below command as an example:

**docker run -v /data/app:usr/src/app myapp**

* The above command mounts the directory /data/app in the host to the usr/src/app directory. We can sync the container with the data files from the host without having the need to restart it.
* This also ensures data security in cases of container deletion. This ensures that even if the container is deleted, the data of the container exists in the volume mapped host location making it the easiest way to store the container data.