Week 1 Readings and Resources

With the first week of our course on containers being primarily focused on grasping the basics of containerization through the use of Docker. The additional readings for this week will continue to help you dive into better understanding those basics so that you have a strong foundation as you learn about container management platforms.

**Docker Basics**

To fully understand the Docker engine, familiarize yourself with the Docker architecture and the technology underlying it. Docker uses a client-server architecture. The main components that make up this architecture is the Docker daemon, the Docker client, and Docker registries.

You will use Docker to create and manage images, containers, networks, volumes, and more.

For a brief overview of some of the core components that make up Docker, check out this link:

<https://docs.docker.com/engine/docker-overview/>

**Dockerfiles**

In order to have an executable container image, you need to create a text file known as a Dockerfile. Dockerfiles contain instructions on how to build a Docker image step by step. There are many different keywords that you will use to write instructions. When the Dockerfile is built, each instruction is executed in order and the end result is an executable Docker image.

For the full list of keywords and more information on Dockerfiles, click here: <https://docs.docker.com/engine/reference/builder/>

**Docker Images**

 Docker uses Union File Systems and the copy on write strategy for managing files and layers.

The use of the copy on write strategy helps keep the top writable layer small, copying files into the top writable layer only when needed. To read more about copy on write and union files systems click here: <https://docs.docker.com/storage/storagedriver/>

**Docker CLI and Logging**

The Docker CLI is a tool that developers can use to interact with the Docker engine. It allows you to build containers, run containers, view container logs, and stop your containers, all through a terminal.

When you use the CLI, you’ll type a command into your terminal that starts with docker. After the base command, you can use whichever child command you’d like, such as docker **build**, docker **logs**, or docker **run**.

For a full list of Docker CLI commands, click on the following link:

<https://docs.docker.com/engine/reference/commandline/docker/>

**AWS Elastic Beanstalk**

AWS Elastic Beanstalk provisions the needed infrastructure for you, installs and manages the runtime of your application, and places your running application onto an EC2 instance(s). AWS Elastic Beanstalk supports containerized applications in addition to various supported languages and runtimes. AWS Elastic Beanstalk supports multiple containers for a multi-container environment by provisioning and managing an Amazon Elastic Container Service cluster on your behalf.

Read more about AWS Elastic Beanstalk and Docker containers here: <https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create_deploy_docker.html>

**Amazon Elastic Container Registry**

A strong understanding of container image registry utilization is going to be very important as you learn about, and begin your use of, containers. Amazon Elastic Container Registry provides a lot of useful features and great benefits as an image registry, and you should familiarize yourself with at least its basic concepts as soon as possible.

To dive more into the parts and functions of Amazon ECR, you can start with the documentation here:

<https://docs.aws.amazon.com/AmazonECR/latest/userguide/what-is-ecr.html>