System Administration (LTAT.06.003)

Lab 9
Containers
What are containers?

• A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

• Container images become containers at runtime and in the case of Docker containers – images become containers when they run.

• A container image is an immutable, static file which includes everything a container needs to run - the container engine settings such as Docker or CoreOS, system libraries, utilities, configuration settings, and specific workloads that should run on the container.
Benefits of containers

- **Very little overhead** - Containers require less system resources than traditional or hardware virtual machine environments because they don’t include operating system images.

- **Good portability** - Applications running in containers can be deployed easily to multiple different operating systems and hardware platforms.

- **Easily automated** - The processes for building, testing and scaling containers is fairly simple and efficient, especially when compared to VMs.
Problems with containers (without external tooling)

- Very difficult to keep track of.
- Monitoring containers requires custom solutions or good practices.
- All images need to be carefully vetted beforehand, as they can contain malicious code that will happily run.
- Upgrading containers to new versions can break things, as it’s often difficult to keep track what changed between the versions and what the implications are.
- Updating libraries in containers is difficult, leading to increased attack surfaces.
- Publishing containers to the internet through conventional web servers and methods requires a lot of manual work.
Solutions to problems

- Need to do security scanning on your containers, to prevent security issues. (Will do in DevOps lab)
- Need some kind of container orchestration for easier management, like Docker Swarm or Kubernetes. (Will do in Kubernetes lab)
- Containers should be fairly short lived, built and deployed automatically through pipelines. (Will do in DevOps lab)
- Using dynamic web servers, service discovery and automatic orchestration processes is strongly recommended to avoid manual work. (Will do in DevOps + Kubernetes labs)
Containers lab

- Install and get familiar with Docker
- Run a Docker container, in both ad-hoc and daemon methods.
- Write a Dockerfile and build an image using it
- Try different methods of routing traffic to it