# Containerization

Elisaveta Manasieva – Senior Software Engineer at ATOSS Software AG

# AGENDA

- What is CONTAINERIZATION?
- History of Containerization
- What is Docker and Docker Containers?
- Docker terminologies
- Benefits of using containers
- Demo (Build, Tag, Push)
- ATOSS Containers (ASES, AMIS, ABC, HELP)

# What is CONTAINERIZATION?

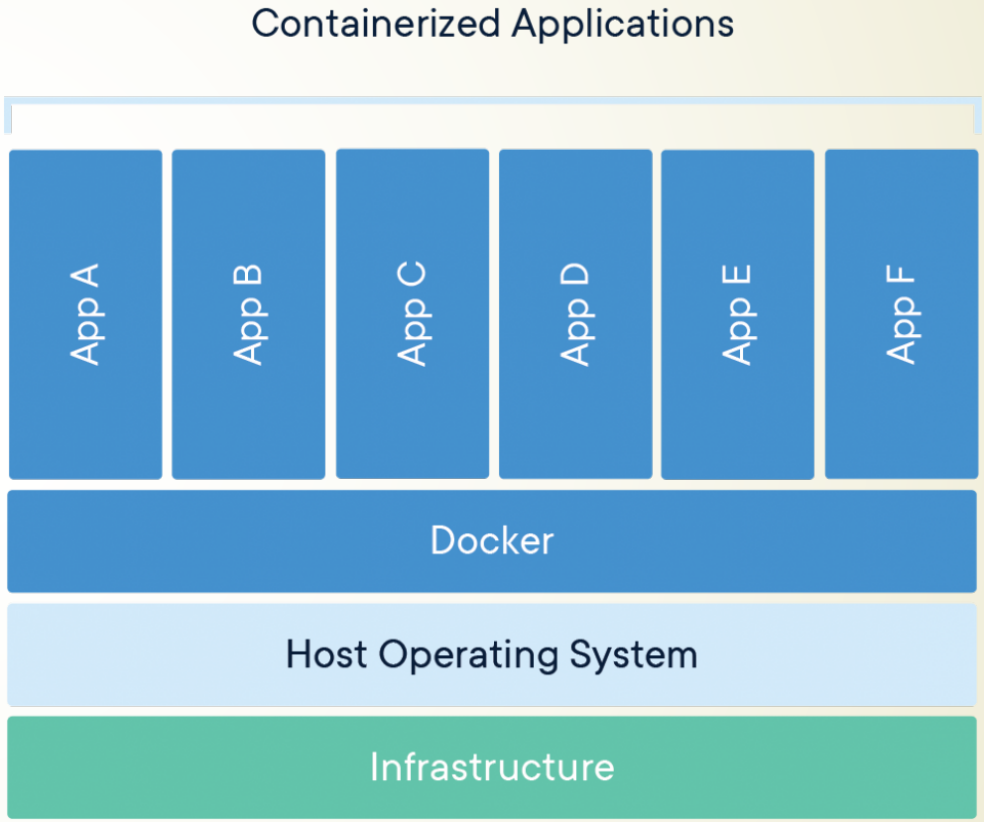
- **Containerization** is the bundling of application code and dependencies into a single, virtual package. A containerized application typically sits alongside other applications, and runs via a shared operating system on a computer, server, or cloud.

# History of Containerization

- Before Containers
- Introduction of Virtual Machines
- Introduction and Evolution of Container Technologies

# What is Docker and Docker Container?

- **Docker** is the containerization platform which is used to package your application and all its dependencies together in the form of containers so to make sure that your application works seamlessly in any environment which can be development or test or production. Docker is a tool designed to make it easier to create, deploy, run applications by using 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.



# Docker Terminologies

- **Container image** – A package that provides all the dependencies and information one needs to create a container.
- **Docker Hub** – A public image-hosting registry where you can upload images and work on them.
- **Dockerfile** – A text file containing instructions on how to build a Docker image.
- **Repository** – A network-based or internet-based service that stores Docker images. There are both private and public Docker repositories.
- **Registry** – A service that stores repositories from multiple sources. It can be both public as well as private.
- **Azure Container Registry** – A registry provider for storing Docker images.

# Dockerfile

```
FROM golang:1.16.4-stretch as builder
ENV GOPATH /go
ARG DOCK_PKG_DIR=/
WORKDIR /
COPY ./ $DOCK_PKG_DIR

RUN go build ./main.go

FROM debian:stretch
COPY --from=0 /main .
COPY --from=0 / /

ENTRYPOINT ["./main"]
```

# Docker Registry

- http://adeimages:8081/#browse/browse:dockerasesimages
- http://nexus.apps.lab.ocp.lan/#browse/browse:dockerimages
- http://nexus-mirror-rnd.micro-okd.atoss.com/#browse/search/docker
- https://hub.docker.com/

# BENEFITS OF USING CONTAINERS

**1. DevOps-friendly -** Containerization packages the application along with its environmental dependencies, which ensures that an application developed in one environment works in another. This helps developers and testers work collaboratively on the application, which is exactly what DevOps culture is all about.

**2. Multiple Cloud Platform -** Containers can be run on multiple cloud platforms like GCS, Amazon ECS (Elastic Container Service), Amazon DevOps Server.

**3. Portable in Nature -** Containers offer easy portability. A container image can be deployed to a new system easily, which can then be shared in the form of a file.

**4. Faster Scalability -** As environments are packaged into isolated containers, they can be scaled up faster, which is extremely helpful for a distributed application.

**5. No Separate OS Needed -** In the VM system, the bare-metal server has a different host OS from the VM. On the contrary, in containers, the Docker image can utilize the kernel of the host OS of the bare-metal physical server. Therefore, containers are comparatively more work-efficient than VMs.

**6. Maximum Utilization of Resources -** Containerization makes maximum utilization of computing resources like memory and CPU, and utilize far fewer resources than VMs.

**7. Fast-Spinning of Apps -** With the quick spinning of apps, the delivery takes place in less time, making the platform convenient for performing more development of systems. The machine does not need to restart to change resources.

With the help of automated scaling of containers, CPU usage and machine memory optimization can be done taking the current load into consideration. And unlike the scaling of Virtual Machines, the machine does not need to be restarted to modify the resource limit.

**8. Simplified Security Updates -** As containers provide process isolation, maintaining the security of applications becomes a lot more convenient to handle.

**9. Value for Money -** Containerization is advantageous in terms of supporting multiple containers on a singular infrastructure. So, despite investing in tools, CPU, memory, and storage, it is still a cost-effective solution for many enterprises.

A complete DevOps workflow, with containers implemented, can be advantageous for the software development team in the following ways:

Offers automation of tests in every little step to detect errors, so there are fewer chances of defects in the end product.

Faster and more convenient delivery of features and changes.

- Nature of the software is more user-friendly than VM-based solutions.
- Reliable and changeable environment.
- Promotes collaboration and transparency among the team members.
- Cost-efficient in nature.
- Ensures proper utilization of resources and limits wastage.

# Demo (Build, Tag and Push a docker image)

- docker build -t demo-image:1 .
- docker images
- docker tag <image_id> <tag>:<version>
- docker push <image>

# ATOSS Containers

- ASES
- AMIS
- HELP
- ABC

# Questions?

# Reference links

- https://www.docker.com/resources/what-container
- https://docs.docker.com/engine/reference/commandline/container/
- https://www.geeksforgeeks.org/containerization-using-docker/#:~:text=Docker%20is%20the%20containerization%20platform%20which%20is%20used,which%20can%20be%20development%20or%20test%20or%20production.