Mark Angelo D. Siazon IV-ACSAD

Docker and Containerization

Overview

Learn container fundamentals with Docker through a focused plan from basics to deployment. You will build, run, and ship containers, then apply security and best practices with small weekly projects.

Learning Outcomes

- Explain containers and how they isolate processes (namespaces, cgroups)
- Use Docker to run, inspect, network, and persist containers
- Write efficient Dockerfiles; use multi-stage builds
- Manage multi-container apps with Docker Compose
- Apply security, performance, and image hygiene best practices
- Publish images to a registry; add Docker to CI pipelines

Short History

- 1979: chroot on Unix
- 2002–2013: Linux namespaces and cgroups mature
- 2004: Solaris Zones (OS-level virtualization)
- 2008: LXC (Linux Containers)
- 2013: Docker launches
- 2015: Open Container Initiative (OCI) standards
- 2016+: containerd, runc, Kubernetes ecosystem growth

Best Practices

- Use small base images (alpine, distroless) when possible
- Pin versions; use .dockerignore to shrink context
- Create a non-root user; drop privileges
- Use multi-stage builds; keep only runtime artifacts
- Add HEALTHCHECK to critical services
- Do not bake secrets into images; mount or use a secret manager
- Persist data in volumes, not container FS
- Scan images for CVEs and patch regularly
- Set resource limits and restart policies
- Tag images consistently (e.g., app:1.2.0)
- Log to stdout and collect centrally

Two-Week Plan

Week 1 Foundations

- Install Docker Desktop or Docker Engine
- Run containers: hello-world, nginx, redis
- Explore commands: run, ps, logs, exec, stop, rm, images, rmi, inspect
- Volumes and bind mounts for data persistence
- Docker networks and port mapping
- Lab: build a simple static site container served by nginx

Week 2 Build, Compose, Ship

- Dockerfile basics: FROM, COPY, RUN, EXPOSE, CMD, ENTRYPOINT
- Layer caching, .dockerignore, and tagging
- Multi-stage builds for small images
- Docker Compose for web app + database
- Add HEALTHCHECK, non-root user, and env vars
- Push to a registry and pull on another machine
- Lab: containerize a small API with database using Compose

Milestone Project

Containerize a small service (e.g., Python FastAPI or Node app) with a database. Write a Dockerfile with multi-stage build, add a Compose file for the app, db, and reverse proxy, publish the image to a registry, then run it on another machine.

Command Cheatsheet

- docker run -d -p 8080:80 nginx   run nginx and map port 80 to 8080
- docker ps -a   list containers
- docker logs -f <id>   follow logs
- docker exec -it <id> sh   open a shell
- docker build -t app:dev .   build an image
- docker image prune   remove unused images
- docker volume ls   list volumes
- docker network ls   list networks
- docker compose up -d   start multi-container app

Assessment Checklist

- Explains containers and images in plain language
- Builds and runs a container locally
- Uses a Dockerfile with multi-stage builds
- Stores persistent data in a volume
- Runs a multi-service stack with Compose
- Implements a non-root user and a HEALTHCHECK
- Publishes and pulls an image from a registry

Resources

- Docker docs: docs.docker.com
- Awesome Compose: github.com/docker/awesome-compose
- OCI specifications: opencontainers.org