Advanced Docker Assignment (Comprehensive & In-Depth)

Overview:

This assignment is designed to provide a complete and highly advanced understanding of Docker, covering all major concepts including image creation, networking, storage, orchestration, security, plugin development, and CI/CD. Each task is hands-on and intended to challenge even experienced Docker users.

Section 1: Core Docker Concepts

Task 1: Building and Managing Images

- Write a Dockerfile for a Node.js application using best practices.
- Use `.dockerignore` to optimize build context.
- Build, tag, and push to Docker Hub.
- Create a multi-architecture image using `buildx`.
- Automate the build process using a Makefile or Shell script.

Deliverables:

- Dockerfile
- `.dockerignore`
- `Makefile` or script
- Image link on Docker Hub

Task 2: Docker Volumes and Storage Management

- Create a Flask logging app.

- Use named volumes to persist logs and database data.

- Use bind mounts for source code hot reload.

- Explore tmpfs mounts and explain use cases.

Deliverables:

- Flask app

- `docker-compose.yml` with all volume types

- Screenshots/logs showing persistence

Task 3: Docker Networking and Service Discovery

- Create a 3-tier app: frontend, backend (Flask), database (Postgres), and Redis.

- Use user-defined bridge and overlay networks.

- Demonstrate inter-container communication.

- Implement service discovery using container names and aliases.

Deliverables:

- `docker-compose.yml` with custom networks

- Ping/curl results for service discovery

Section 2: Image Optimization and Security

Task 4: Multi-Stage Builds and Image Slimming

- Use multi-stage builds for a React or Go application.

- Compare image sizes with/without multi-stage builds.

- Use distroless or Alpine base images.

Deliverables:

- Optimized Dockerfile

- Image size comparison table

- Explanation of slimming techniques used

Task 5: Docker Security Hardening

- Scan images with `trivy` or `docker scan`.

- Patch/mitigate at least 3 vulnerabilities.

- Use non-root user, `--read-only`, `--cap-drop`, `--no-new-privileges`.

- Apply AppArmor and Seccomp profiles.

Deliverables:

- Scan reports (before and after)

- Hardened Dockerfile and run commands

- Explanation of each security measure

Section 3: Advanced Docker Usage

Task 6: Docker Compose and Secret Management

- Create a full-stack app using Docker Compose.

- Use `.env` files and secrets management.

- Dynamically inject secrets without environment variables.

Deliverables:

- `docker-compose.yml`

- Secret files/configs

- Explanation of secret injection strategy

Task 7: Docker Swarm and Orchestration

- Deploy the full-stack app using Swarm.

- Use `docker stack deploy`.

- Scale services, perform rolling updates, and simulate failure recovery.

- Use placement constraints and resource limits.

Deliverables:

- `docker-compose.yml` for Swarm

- Logs/screenshots of scaling and updates

Task 8: Kubernetes Fundamentals (Optional but Recommended)

- Convert your app to Kubernetes YAML manifests.

- Use Deployments, Services, ConfigMaps, Secrets, and Ingress.

Deliverables:

- Kubernetes YAML files

- Explanation of architecture and resource definitions

Section 4: Plugins, CI/CD, and Infrastructure Automation

Task 9: Plugin Development

- Write a custom Docker volume or network plugin in Go or Python.

- Implement a simple logging plugin.

Deliverables:

- Plugin source code

- Instructions to build and install

Task 10: CI/CD with Docker

- Set up GitHub Actions or GitLab CI:

- Build and scan Docker image

- Run tests inside container

- Push to Docker Hub or GHCR

- Deploy to Docker Swarm or Kubernetes

Deliverables:

- CI/CD config file

- Screenshot of successful run

- Deployment script or config

Task 11: Docker Host Hardening and Infrastructure as Code

- Use Ansible or Terraform to:

- Set up a hardened Docker host

- Enforce Docker daemon config (e.g., rootless mode, TLS)

Deliverables:

- IaC scripts

- Documentation for the setup

Submission Guidelines:

- Organize code by task in folders

- Include markdown/explanation for each task

- Submit as a zipped directory or public GitHub repo

Expected Outcome:

You will gain expert-level Docker skills including image creation, security, networking, orchestration with Swarm/Kubernetes, plugin development, and DevOps CI/CD integration.

Good luck!