**Securing a Multi-Container Application with Docker Compose**

* **Goal**: Use Docker Compose to orchestrate a small web + database setup, with basic security considerations.
* **Tasks**:
  + Create a docker-compose.yml defining two services (web + DB).
  + Configure environment variables for DB credentials, ensuring they are not stored in plaintext within the code.
  + Run containers with minimal privileges (e.g., non-root user if possible).
  + Provide logs/screenshots verifying the services run and communicate.

**Docker Compose Secrets Management**

* **Goal**: Store sensitive data (e.g., passwords) using Docker Compose secrets (or an equivalent approach).
* **Tasks**:
  + Create a secrets file for credentials.
  + Update your docker-compose.yml to reference the secrets.
  + Demonstrate how the container reads the secret from a mounted file or environment variable without exposing it in logs.
  + Provide short documentation or screenshots showing the approach.

**Docker Swarm: Setting Up a 3-Node Cluster**

* **Goal**: Spin up 3 VMs (Vagrant or other tool) and configure 1 manager + 2 workers in Docker Swarm.
* **Tasks**:
  + Initialize Swarm on the manager node (docker swarm init).
  + Join the worker nodes using the provided token/command.
  + Show final docker node ls output confirming the cluster membership.
  + Provide a short reflection on any challenges faced.

**Deploying a Service on Docker Swarm**

* **Goal**: Deploy a simple service (e.g., NGINX) with multiple replicas across the Swarm.
* **Tasks**:
  + Use docker service create or a Compose stack (docker stack deploy) specifying replicas=3.
  + Verify the service is distributed across worker nodes.
  + Document logs or screenshots from each node.

**Performing a Rolling Update on Swarm**

* **Goal**: Update a running service to a new image version with minimal downtime.
* **Tasks**:
  + Deploy an initial version (e.g., nginx:1.21).
  + Update to a newer version (nginx:1.23) using docker service update.
  + Observe the rolling update, ensuring tasks are updated one at a time.
  + Provide final service status logs.

**Minikube Installation and First Pod**

* **Goal**: Install Minikube and run a simple pod in Kubernetes.
* **Tasks**:
  + Install Minikube on your machine (or VM).
  + Start Minikube, confirm with kubectl get nodes.
  + Create a pod (e.g., nginx), using a basic YAML.
  + Provide logs/screenshots verifying the pod is running.

**Deploying a Multi-Replica Service on Minikube**

* **Goal**: Create a Deployment with multiple replicas and expose it via a Service.
* **Tasks**:
  + Write a YAML for a Deployment (e.g., 3 replicas of nginx).
  + Create a Service (NodePort or ClusterIP) to expose it.
  + Verify you can access the service from inside Minikube (and from host if NodePort).
  + Document the YAML and logs from a successful test.

**Scaling a Deployment in Kubernetes**

* **Goal**: Demonstrate how to scale up or down a Deployment.
* **Tasks**:
  + Start with a Deployment of 2 replicas.
  + Use kubectl scale to increase to 5 replicas.
  + Observe the new pods spinning up, then scale down to 1.
  + Provide screenshots or logs of each scaling step.

**Persistent Storage in Minikube**

* **Goal**: Create a PersistentVolume (PV) and PersistentVolumeClaim (PVC) for a pod that needs data persistence.
* **Tasks**:
  + Write YAML for a PV and PVC, referencing them in a pod or deployment.
  + Store data in the pod’s volume, then delete the pod.
  + Re-create the pod to confirm data persists.
  + Show logs or screenshots verifying the volume usage.

**Rolling Updates and Rollbacks in Kubernetes**

* **Goal**: Update a Deployment to a new container image version, then roll it back if needed.
* **Tasks**:
  + Deploy an app (version 1).
  + Update to version 2 using kubectl set image or editing the YAML.
  + If needed, roll back using kubectl rollout undo.
  + Provide logs or screenshots showing the rollout steps.

**Docker Compose for a Simple Microservice Setup**

* **Goal**: Run at least two microservices (web + DB) using Docker Compose, focusing on secure defaults.
* **Tasks**:
  + Provide a docker-compose.yml that references environment variables for DB credentials.
  + Restrict DB access to the internal Compose network only.
  + Document the logs verifying both containers are healthy and able to communicate.

**Docker Compose with Secrets**

* **Goal**: Use Docker Compose secrets or an equivalent approach to manage a sensitive credential.
* **Tasks**:
  + Write a compose file that references a secret (e.g., a password).
  + Show how the container reads it from /run/secrets/<secret-name> or an environment variable.
  + Provide logs or screenshots verifying the secret is not logged in plaintext.

**Docker Swarm Overlay Network**

* **Goal**: Create a custom overlay network for services in a Swarm.
* **Tasks**:
  + Set up a 3-node Swarm (1 manager, 2 workers).
  + Create an overlay network (docker network create -d overlay).
  + Deploy a service connected to this overlay network, verifying container-to-container communication.
  + Document the final docker service ls and docker network ls outputs.

**Docker Swarm Secrets**

* **Goal**: Store a secret in Docker Swarm and reference it in a service.
* **Tasks**:
  + Create a secret (docker secret create <name> <file>).
  + Deploy a service (e.g., a small web app) that uses the secret from /run/secrets/<name>.
  + Show logs or environment output verifying the secret is accessible inside the container but not stored in plain text in the image.

**Upgrading a Swarm Service with Zero Downtime**

* **Goal**: Perform a rolling update in Docker Swarm.
* **Tasks**:
  + Deploy a service (e.g., nginx) with multiple replicas.
  + Update the service to a new image version using docker service update.
  + Observe tasks being updated incrementally, ensuring no downtime.
  + Provide logs showing the update steps.

**Minikube with a Custom Docker Image**

* **Goal**: Build a Docker image locally and deploy it on Minikube without pushing to a remote registry.
* **Tasks**:
  + Enable Minikube’s Docker daemon (eval $(minikube docker-env)).
  + Build your image so it’s available in Minikube’s local Docker environment.
  + Create a pod or deployment that uses this local image.
  + Document logs verifying the container runs successfully.

**Kubernetes Service Discovery**

* **Goal**: Demonstrate how pods communicate via a Service and DNS within Minikube.
* **Tasks**:
  + Create two Deployments: a front-end and a back-end.
  + Expose the back-end with a ClusterIP Service.
  + Update the front-end to call the back-end using the service name (e.g., http://backend-svc).
  + Show logs verifying successful communication via DNS.

**Kubernetes ConfigMaps**

* **Goal**: Use a ConfigMap to store non-sensitive configuration for a container.
* **Tasks**:
  + Create a ConfigMap with environment variables or a config file.
  + Deploy a pod or deployment that references the ConfigMap.
  + Confirm the container reads the config (e.g., log it at startup).
  + Provide YAML and logs verifying success.

**Kubernetes PersistentVolume and PVC**

* **Goal**: Provide persistent storage to a pod or deployment in Minikube.
* **Tasks**:
  + Create a PersistentVolume (PV) and a matching PersistentVolumeClaim (PVC).
  + Deploy a pod or deployment that mounts the PVC, storing data that persists across pod restarts.
  + Show logs or screenshots verifying data persistence.

**Deploying a Simple Multi-Container Stack on Kubernetes**

* **Goal**: Convert a small multi-container Docker Compose setup to Kubernetes manifests (manually or using a tool like Kompose).
* **Tasks**:
  + Take a Docker Compose file (e.g., web + DB).
  + Generate or write YAML for deployments and services.
  + Deploy on Minikube, verifying both containers run and communicate.
  + Provide final logs or screenshots showing success.