**Containerization with Docker - Essential Commands and Scripts**

**Installing Docker on Rocky Linux**

**Prerequisites**

Before installing Docker, ensure:

* Rocky Linux 9 is installed
* Root or sudo privileges
* Internet connectivity
* System is updated

**System Update**

sudo dnf update -y

**Enable Docker Repository**

sudo dnf config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

Verify the repository:

sudo dnf repolist | grep docker

**Install Docker**

sudo dnf install -y docker-ce docker-ce-cli containerd.io

Confirm installation:

docker --version

**Enable and Start Docker**

sudo systemctl enable --now docker

Check status:

sudo systemctl status docker

**Add User to Docker Group (Optional)**

sudo usermod -aG docker $USER

sudo newgrp docker

Log out and log back in for changes to take effect.

**Test Docker Installation**

docker run hello-world

**Running an Nginx Container (Example)**

docker run -d --name mynginx -p 8080:80 nginx

Access via browser: http://localhost:8080

**Security Best Practices**

* Use non-root users for containers.
* Enable firewall rules.
* Regularly update Docker.
* Monitor logs: journalctl -u docker
* Limit privileged containers.

**Creating and Managing Docker Containers**

**Pulling a Docker Image**

sudo docker pull nginx

Verify image download:

sudo docker images

**Running a Docker Container**

sudo docker run -d --name mynginx -p 8080:80 nginx

**Inspecting and Managing Containers**

sudo docker ps # List running containers

sudo docker ps -a # List all containers

sudo docker inspect mynginx # Get details

**Restarting, Stopping, and Removing Containers**

sudo docker restart mynginx

sudo docker stop mynginx

sudo docker rm mynginx

sudo docker container prune # Remove all stopped containers

**Running Interactive Containers**

sudo docker run -it ubuntu

Exit interactive session:

exit

**Viewing Logs**

sudo docker logs mynginx

sudo docker logs -f mynginx # Real-time logs (Exit with Ctrl + C)

**Container Networking and Storage**

**Creating a Custom Bridge Network**

docker network create my\_bridge\_network

docker run -d --name web\_container --network my\_bridge\_network nginx

**Creating a Volume**

docker volume create my\_volume

docker run -d --name db\_container -v my\_volume:/var/lib/mysql mysql

**Using Bind Mounts**

docker run -d --name app\_container -v /host/path:/container/path nginx

**Using tmpfs Mounts**

docker run -d --name cache\_container --tmpfs /tmp:rw,size=64m,mode=1777 nginx

**Docker Compose for Multi-Container Applications**

**Installing Docker Compose**

sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

Verify installation:

docker compose version

**Creating a Multi-Container Application**

version: "3.8"

services:

web:

image: nginx:latest

ports:

- "8080:80"

depends\_on:

- redis

redis:

image: redis:latest

Start the application:

docker compose up -d

Stop the application:

docker compose down

**AWS ECS and EKS for Container Orchestration**

**Setting Up AWS ECS**

1. Create an ECS cluster in AWS Console.
2. Define a Task Definition specifying:
   * Docker image
   * CPU & Memory
   * IAM role
3. Run a task from ECS Console.

**Setting Up AWS EKS**

aws eks --region <region> update-kubeconfig --name <cluster\_name>

**Deploying an Application on EKS**

apiVersion: apps/v1

kind: Deployment

metadata:

name: my-app

spec:

replicas: 3

template:

spec:

containers:

- name: my-app

image: my-docker-image:latest

ports:

- containerPort: 80

Apply the deployment:

kubectl apply -f deployment.yaml

Expose the application:

apiVersion: v1

kind: Service

metadata:

name: my-service

spec:

type: LoadBalancer

ports:

- port: 80

selector:

app: my-app

Apply service configuration:

kubectl apply -f service.yaml

This document compiles all essential Docker and AWS ECS/EKS commands, enabling efficient containerized application deployment and management on Rocky Linux.