Docker Swarm Command Reference

1. Swarm Initialization and Setup

- `docker swarm init`

- Description: Initializes the current Docker host as the Swarm manager.

- Example: `docker swarm init --advertise-addr <IP>`

- Initializes a Swarm and advertises the manager’s IP.

- `docker swarm join --token <token> <manager-IP>:<port>`

- Description: Joins a node to an existing Swarm as a worker or manager.

- Example: `docker swarm join --token SWMTKN-1-xyz 192.168.1.10:2377`

- `docker swarm leave`

- Description: Removes the node from the Swarm.

- Example: `docker swarm leave`

- Forces the node to leave the Swarm.

- `docker swarm join-token [worker|manager]`

- Description: Retrieves the join token for adding nodes to the Swarm as either a worker or manager.

- Example: `docker swarm join-token worker`

---

2. Node Management

- `docker node ls`

- Description: Lists all nodes in the Swarm.

- Example: `docker node ls`

- `docker node inspect [node-ID]`

- Description: Provides detailed information about a specific node.

- Example: `docker node inspect node1`

- `docker node update [options] [node-ID]`

- Description: Updates the configuration of a node (e.g., role, availability).

- Example: `docker node update --availability drain node1`

- Puts `node1` into maintenance mode.

- `docker node promote [node-ID]`

- Description: Promotes a node from worker to manager.

- Example: `docker node promote node1`

- `docker node demote [node-ID]`

- Description: Demotes a manager node to a worker.

- Example: `docker node demote node1`

---

3. Service Management

- `docker service create [options] [image]`

- Description: Creates and deploys a new service in the Swarm.

- Example: `docker service create --name webserver -p 80:80 nginx`

- Creates a service named `webserver` with the nginx image.

- `docker service ls`

- Description: Lists all services running in the Swarm.

- Example: `docker service ls`

- `docker service ps [service-name]`

- Description: Shows the tasks (individual containers) associated with a service.

- Example: `docker service ps webserver`

- `docker service scale [service-name]=[replicas]`

- Description: Scales the service to the specified number of replicas.

- Example: `docker service scale webserver=5`

- Scales the `webserver` service to 5 instances.

- `docker service update [options] [service-name]`

- Description: Updates an existing service’s configuration (e.g., image, replicas).

- Example: `docker service update --image nginx:latest webserver`

- `docker service rm [service-name]`

- Description: Removes a service from the Swarm.

- Example: `docker service rm webserver`

---

4. Task Management

- `docker task ls`

- Description: Lists all tasks running in the Swarm.

- Example: `docker task ls`

- `docker task inspect [task-ID]`

- Description: Provides detailed information about a specific task.

- Example: `docker task inspect task1`

- `docker task logs [task-ID]`

- Description: Shows the logs for a specific task.

- Example: `docker task logs task1`

---

5. Stack Management

- `docker stack deploy -c [compose-file] [stack-name]`

- Description: Deploys a new stack using a Docker Compose file.

- Example: `docker stack deploy -c docker-compose.yml mystack`

- `docker stack ls`

- Description: Lists all deployed stacks.

- Example: `docker stack ls`

- `docker stack ps [stack-name]`

- Description: Lists all tasks in a specific stack.

- Example: `docker stack ps mystack`

- `docker stack services [stack-name]`

- Description: Lists all services in a specific stack.

- Example: `docker stack services mystack`

- `docker stack rm [stack-name]`

- Description: Removes a stack.

- Example: `docker stack rm mystack`

---

6. Swarm Management

- `docker config create [config-name] [file]`

- Description: Creates a configuration file to be used by services.

- Example: `docker config create myconfig config.txt`

- `docker secret create [secret-name] [file]`

- Description: Creates a secret to be used securely by services.

- Example: `docker secret create mysecret secret.txt`

- `docker config ls`

- Description: Lists all configurations available to the Swarm.

- Example: `docker config ls`

- `docker secret ls`

- Description: Lists all secrets available to the Swarm.

- Example: `docker secret ls`

- `docker config rm [config-name]`

- Description: Removes a configuration.

- Example: `docker config rm myconfig`

- `docker secret rm [secret-name]`

- Description: Removes a secret.

- Example: `docker secret rm mysecret`

---

7. Swarm Troubleshooting and Monitoring

- `docker service logs [service-name]`

- Description: Shows logs for a specific service.

- Example: `docker service logs webserver`

- `docker node ps [node-ID]`

- Description: Lists all tasks running on a specific node.

- Example: `docker node ps node1`

- `docker service inspect [service-name]`

- Description: Provides detailed JSON information about a service.

- Example: `docker service inspect webserver`

- `docker stack inspect [stack-name]`

- Description: Provides detailed JSON information about a stack.

- Example: `docker stack inspect mystack`

- `docker swarm update [options]`

- Description: Updates Swarm configuration, such as the dispatcher heartbeat period.

- Example: `docker swarm update --dispatcher-heartbeat 10s`

Certainly! Here’s a section specifically for Docker daemon debugging commands:

---

8. Docker Daemon Debugging

- `dockerd --debug`

- Description: Starts the Docker daemon in debug mode to provide detailed logging.

- Example: `dockerd --debug`

- Starts the daemon with debug logging, which can be helpful for troubleshooting startup or runtime issues.

- `docker info`

- Description: Provides detailed information about the Docker installation, including storage driver, logging driver, and network configurations.

- Example: `docker info`

- Useful for understanding the system environment and configuration details.

- `docker logs [container-ID]`

- Description: Shows logs from a specific container.

- Example: `docker logs my\_container`

- Useful for debugging issues within running containers.

- `docker inspect [resource]`

- Description: Provides low-level details about Docker objects (containers, images, volumes, networks).

- Example: `docker inspect my\_container`

- Helps identify configuration and state details for resources, including environment variables and runtime configurations.

- `docker events`

- Description: Streams real-time events from the Docker daemon, which can include container start, stop, creation, deletion, etc.

- Example: `docker events`

- Useful for monitoring Docker activity in real time to trace potential issues.

- `docker daemon --log-level debug`

- Description: Configures the Docker daemon’s log level to debug, capturing more detailed logging information.

- Example: `docker daemon --log-level debug`

- Setting a higher log level provides insights into daemon operations and error messages.

- `sudo journalctl -u docker.service`

- Description: Fetches logs specifically related to the Docker daemon service on systems using `systemd` (e.g., Ubuntu).

- Example: `sudo journalctl -u docker.service -n 100`

- Displays the last 100 lines of Docker daemon logs, making it easier to identify recent issues.

- `docker network inspect [network-name]`

- Description: Provides details on a specific Docker network, showing connected containers, IP addresses, and network configurations.

- Example: `docker network inspect bridge`

- Useful for debugging network connectivity issues between containers.

- `docker volume inspect [volume-name]`

- Description: Shows configuration and mount point details for a specified volume.

- Example: `docker volume inspect my\_volume`

- Useful for verifying volume mounts and troubleshooting data persistence issues.

- `strace -p $(pidof dockerd)` (Linux-only)

- Description: Attaches to the Docker daemon process and outputs system calls in real time.

- Example: `strace -p $(pidof dockerd)`

- Helps in identifying system-level issues affecting the Docker daemon, such as file access or network calls. (Requires `strace` installed.)

- `docker stats`

- Description: Provides a real-time view of resource usage (CPU, memory, network) by each container.

- Example: `docker stats`

- Useful for identifying resource-intensive containers or bottlenecks.

- `docker system df`

- Description: Shows Docker disk usage, including total space used by images, containers, volumes, and build cache.

- Example: `docker system df`

- Helps diagnose issues related to disk space or excessive unused data.

- `docker system prune`

- Description: Removes unused data, including stopped containers, dangling images, and unused volumes.

- Example: `docker system prune -f`

- Useful for clearing up space and resolving storage issues affecting the Docker daemon.

- `docker-compose logs`

- Description: Shows combined logs of all services managed by Docker Compose.

- Example: `docker-compose logs -f`

- Helpful for debugging applications managed with Docker Compose, where multiple containers may be interdependent.