Mayank Singh

Master Docker with 100 Essential Commands and Interview Insights

Docker Basics

1. **docker --version**: Displays the installed Docker version.
2. **docker info**: Shows detailed information about the Docker installation.
3. **docker help**: Provides help for Docker commands.
4. **docker search <image_name>**: Searches for images on Docker Hub.
5. **docker pull <image_name>**: Pulls an image from Docker Hub.
6. **docker images**: Lists all the downloaded images.
7. **docker rmi <image_id>**: Removes an image by its ID.

Containers Management

8. **docker run <image_name>**: Runs a container from an image.
9. **docker ps**: Lists running containers.
10. **docker ps -a**: Lists all containers, including stopped ones.
11. **docker stop <container_id>**: Stops a running container.
12. **docker start <container_id>**: Starts a stopped container.
13. **docker restart <container_id>**: Restarts a container.
14. **docker rm <container_id>**: Removes a container.
15. **docker exec -it <container_id> bash**: Accesses the shell of a running container.
16. **docker kill <container_id>**: Immediately stops a container.

Networking

17. **docker network ls**: Lists all Docker networks.
18. **docker network create <network_name>**: Creates a new network.
19. **docker network rm <network_name>**: Removes a network.
20. **docker network inspect <network_name>**: Displays detailed information about a network.

21. **docker run --network <network_name> <image_name>**: Runs a container in a specific network.
22. **docker network connect <network_name> <container_id>**: Connects a container to a network.
23. **docker network disconnect <network_name> <container_id>**: Disconnects a container from a network.

Volumes

24. **docker volume ls**: Lists all Docker volumes.
25. **docker volume create <volume_name>**: Creates a new volume.
26. **docker volume rm <volume_name>**: Removes a volume.
27. **docker volume inspect <volume_name>**: Displays details of a volume.
28. **docker run -v <volume_name>:<container_path> <image_name>**: Mounts a volume to a container.

Images

29. **docker build -t <image_name>:<tag> .**: Builds an image from a Dockerfile.
30. **docker tag <image_id> <new_name>:<new_tag>**: Tags an image with a new name.
31. **docker push <image_name>:<tag>**: Pushes an image to Docker Hub.
32. **docker save -o <file_name>.tar <image_name>**: Saves an image as a tar file.
33. **docker load -i <file_name>.tar**: Loads an image from a tar file.

Container Logs

34. **docker logs <container_id>**: Displays logs of a container.
35. **docker logs -f <container_id>**: Streams logs of a container.
36. **docker logs --tail 50 <container_id>**: Shows the last 50 lines of logs.

Resource Management

37. **docker stats**: Shows real-time statistics of container resource usage.
38. **docker top <container_id>**: Displays processes running in a container.

39. **docker update --memory 512m <container_id>**: Updates the resource limits of a container.

Inspect and Debugging

40. **docker inspect <container_id>**: Shows detailed information about a container.
41. **docker inspect <image_id>**: Shows detailed information about an image.
42. **docker diff <container_id>**: Lists changes made to a container's filesystem.

Compose

43. **docker-compose up**: Starts services defined in a docker-compose.yml file.
44. **docker-compose down**: Stops and removes containers defined in a Compose file.
45. **docker-compose ps**: Lists services managed by Compose.
46. **docker-compose logs**: Displays logs for Compose-managed containers.

Security

47. **docker scan <image_name>**: Scans an image for vulnerabilities.
48. **docker trust inspect <image_name>**: Inspects the trust policy of an image.
49. **docker content trust enable**: Enables content trust for image signing.
50. **docker history <image_name>**: Shows the history of an image.

Swarm (Orchestration)

51. **docker swarm init**: Initializes a Docker Swarm.
52. **docker swarm join-token worker**: Displays a join token for adding a worker node.
53. **docker service create --name <service_name> <image_name>**: Creates a service in a swarm.
54. **docker service ls**: Lists services in the swarm.
55. **docker node ls**: Lists nodes in the swarm.
56. **docker stack deploy -c <file.yml> <stack_name>**: Deploys a stack.
57. **docker stack rm <stack_name>**: Removes a stack.

Advanced Commands

58. **docker system df**: Displays Docker disk usage.
59. **docker system prune**: Cleans up unused images, containers, and networks.
60. **docker export <container_id> > <file_name>.tar**: Exports a container's filesystem as a tar file.
61. **docker import <file_name>.tar**: Imports a tar file as an image.
62. **docker checkpoint create <container_id> <checkpoint_name>**: Creates a checkpoint for a container.
63. **docker checkpoint ls <container_id>**: Lists checkpoints of a container.
64. **docker checkpoint rm <container_id> <checkpoint_name>**: Removes a checkpoint.

Interview Questions

65. **What is Docker?**: Docker is a containerization platform that packages applications and dependencies into a lightweight, portable container.
66. **Difference between Docker and Virtual Machines?**: Docker uses OS-level virtualization, whereas VMs emulate hardware for isolation.
67. **What is a Dockerfile?**: A Dockerfile is a script containing instructions to build a Docker image.
68. **Explain Docker Compose.**: Docker Compose is a tool to define and manage multi-container applications using YAML.
69. **What is Docker Swarm?**: Swarm is Docker's native clustering and orchestration tool for managing containers.
70. **How is a container different from an image?**: An image is a template, while a container is a running instance of an image.
71. **What is the purpose of Docker volumes?**: Volumes provide persistent storage independent of the container lifecycle.
72. **What are Docker namespaces?**: Namespaces provide isolation for containers by separating resources like processes and networking.
73. **How do you secure Docker containers?**: Use signed images, enable content trust, scan for vulnerabilities, and apply least privilege principles.
74. **What are Docker labels?**: Labels are metadata for organizing and managing containers.

Advanced Docker Networking

75. **docker network prune**: Removes all unused networks.

- **Explanation**: Deletes networks that are not associated with any containers.

76. **docker run --name <container_name> --network <network_name> <image_name>**: Runs a container in a specific network with a custom name.
77. **docker network create --driver bridge <network_name>**: Creates a bridge network.
78. **docker network create --driver overlay <network_name>**: Creates an overlay network for multi-host communication.
79. **docker network connect --ip <custom_ip> <network_name> <container_id>**: Assigns a static IP to a container within a network.

Dockerfile Commands

80. **FROM <base_image>**: Specifies the base image for the Dockerfile.

- **Interview Tip**: This is the first instruction in a Dockerfile.

81. **RUN <command>**: Executes a command during image building.
82. **CMD ["command", "param1", "param2"]**: Sets the default command for a container.
83. **ENTRYPOINT ["command", "param1"]**: Sets a command that is always executed when the container starts.
84. **COPY <source> <destination>**: Copies files from the local system into the container.
85. **WORKDIR <path>**: Sets the working directory inside the container.
86. **EXPOSE <port_number>**: Informs Docker about the port the container listens on.
87. **ENV <key>=<value>**: Sets environment variables inside the container.
88. **ADD <source> <destination>**: Similar to COPY, but supports remote URLs and auto-extracting archives.
89. **LABEL <key>=<value>**: Adds metadata to an image.

Docker Container Lifecycle Management

90. **docker pause <container_id>**: Pauses all processes in a container.

- **Explanation**: Suspends the container without stopping it.

91. **docker unpause <container_id>**: Resumes a paused container.
92. **docker rename <current_name> <new_name>**: Renames a container.

93. **docker commit <container_id> <new_image_name>**: Creates a new image from a container's current state.
94. **docker cp <container_id>:<source_path> <local_path>**: Copies files from a container to the host.
95. **docker export <container_id> -o <file_name>.tar**: Exports a container's filesystem without its layers.

Resource Constraints

96. **docker run --cpus="1.5" <image_name>**: Limits the CPU usage of a container.
97. **docker run --memory="500m" <image_name>**: Restricts memory usage of a container.
98. **docker update --cpus="2" --memory="1g" <container_id>**: Updates resource limits for an existing container.

Multi-Stage Builds

99. **Dockerfile with Multi-Stage Builds**:

```
dockerfile
FROM golang:alpine AS builder
WORKDIR /app
COPY . .
RUN go build -o main .
FROM alpine:latest
WORKDIR /root/
COPY --from=builder /app/main .
CMD ["./main"]
```

- **Explanation**: This approach builds the application in a lightweight final image.

Docker Daemon

100. **dockerd**: Starts the Docker daemon manually. - **Explanation**: Used when troubleshooting or configuring the Docker service