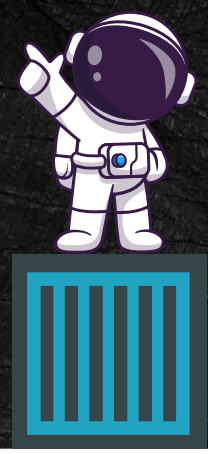
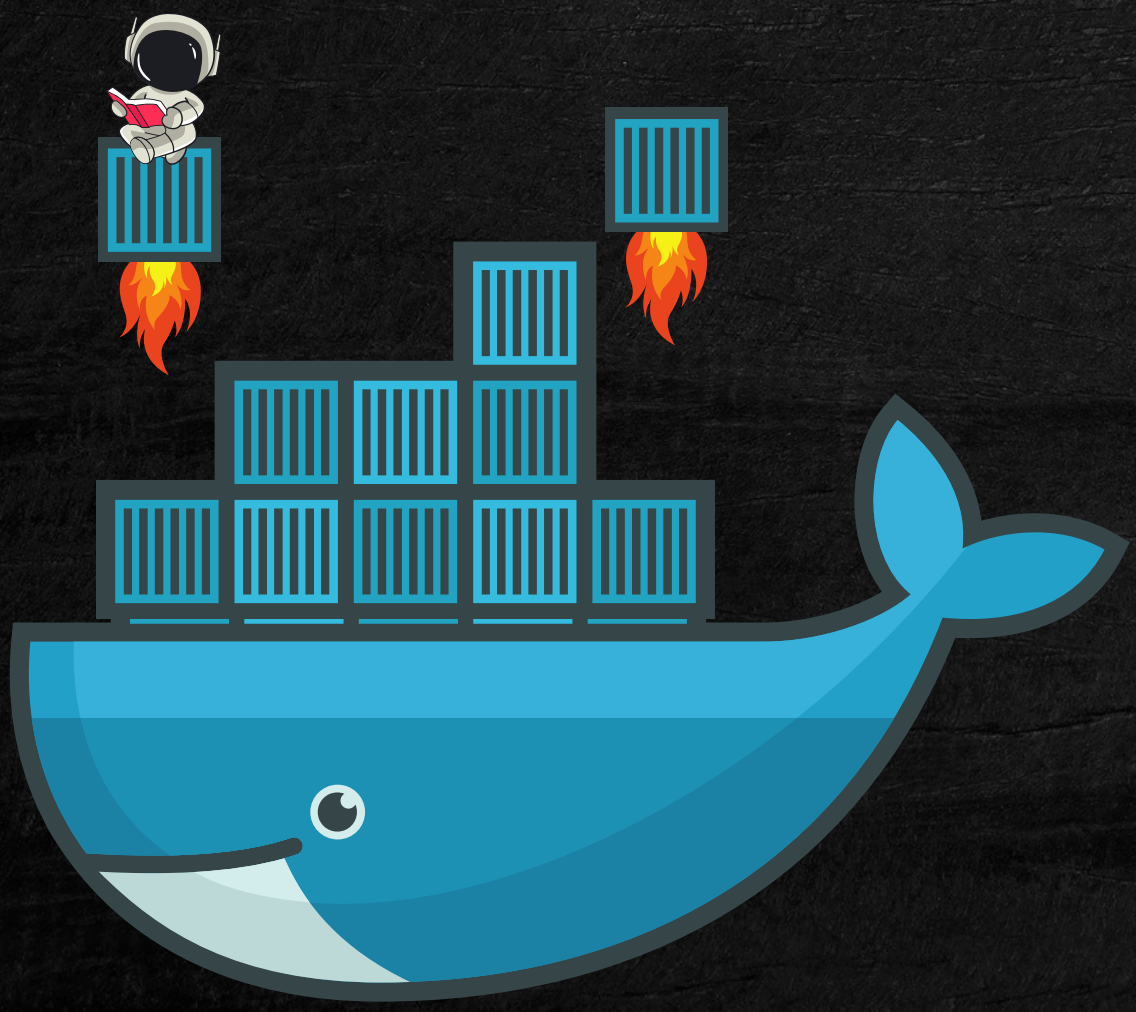
Rajani Ekunde

8

DOCKER BEST PRACTICES YOU SHOULD USE





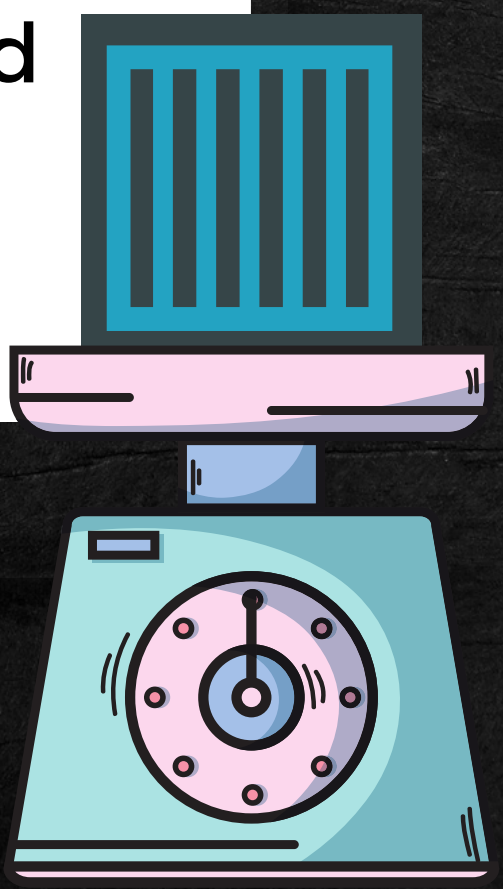
DevOpsOnSteroids

Rajani Ekunde

1

Keep Containers Lightweight

Create lightweight containers. Use minimal base images like Alpine Linux instead of larger distributions. Avoid installing unnecessary packages and libraries in your containers. Additionally, leverage multi-stage builds to separate build and runtime environments.



DevOpsOnSteroids

Rajani Ekunde

2

Use Official Images and Trusted Repositories

Always rely on official Docker images from trusted repositories like Docker Hub. These images are maintained and regularly updated by the official teams, ensuring security patches and bug fixes are promptly delivered. Additionally, avoid using images from untrusted sources to prevent potential security risks. When necessary, create custom images.

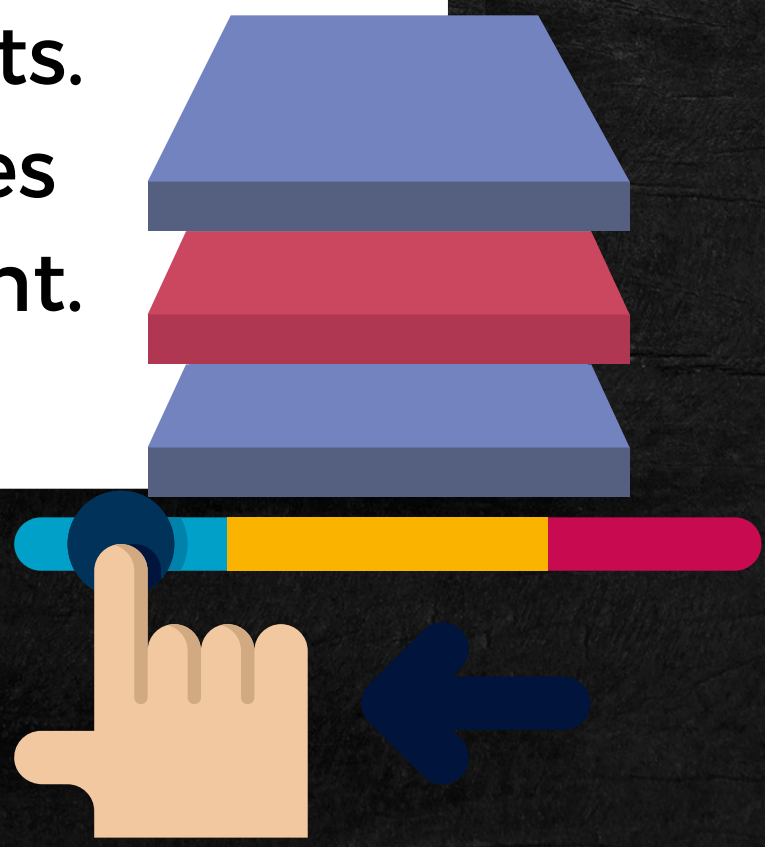
DevOpsOnSteroids

Rajani Ekunde

3

Minimize the Number of Layers

Docker images are built from multiple layers, and each layer introduces some overhead. Minimize the number of layers in your images to reduce image size and build times. Combine multiple commands in a single RUN instruction and use multi-stage builds to clean up unnecessary files, dependencies, or build artifacts. Fewer layers make your images more manageable and efficient.



DevOpsOnSteroids

Rajani Ekunde

4

Limit Container Privileges

By default, Docker containers run with root privileges inside the container, which can pose security risks. To mitigate this, avoid running containers as the root user. Instead, use a less privileged user within the container, and drop unnecessary capabilities. This principle of least privilege ensures that if an attacker gains access to the container, they'll have limited capabilities outside the container.

DevOpsOnSteroids

Rajani Ekunde

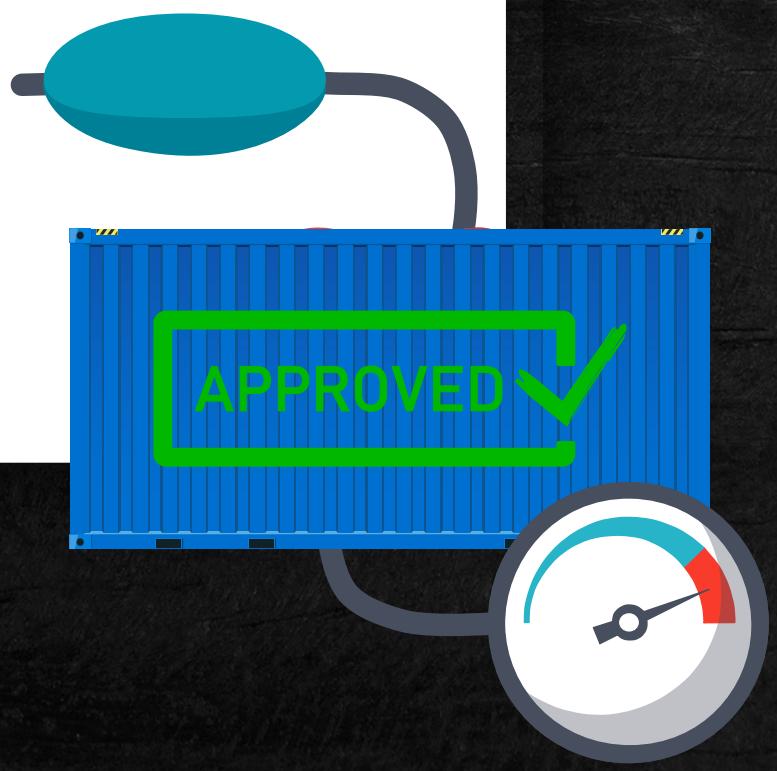
Expose Only Necessary Ports

Minimize the attack surface by exposing only the ports that your application genuinely needs to communicate with the outside world. Exposing unnecessary ports can introduce security vulnerabilities. Use the EXPOSE instruction in your Dockerfile to document the ports required by your application and be cautious while publishing ports during runtime.

Rajani Ekunde

Implement Health Checks

Health checks are essential for ensuring the availability and reliability of your containers. Define health checks in your Dockerfile or using the `HEALTHCHECK` instruction to periodically test the status of your application inside the container. This way, Docker can take corrective actions, like restarting the container, if the application becomes unresponsive.

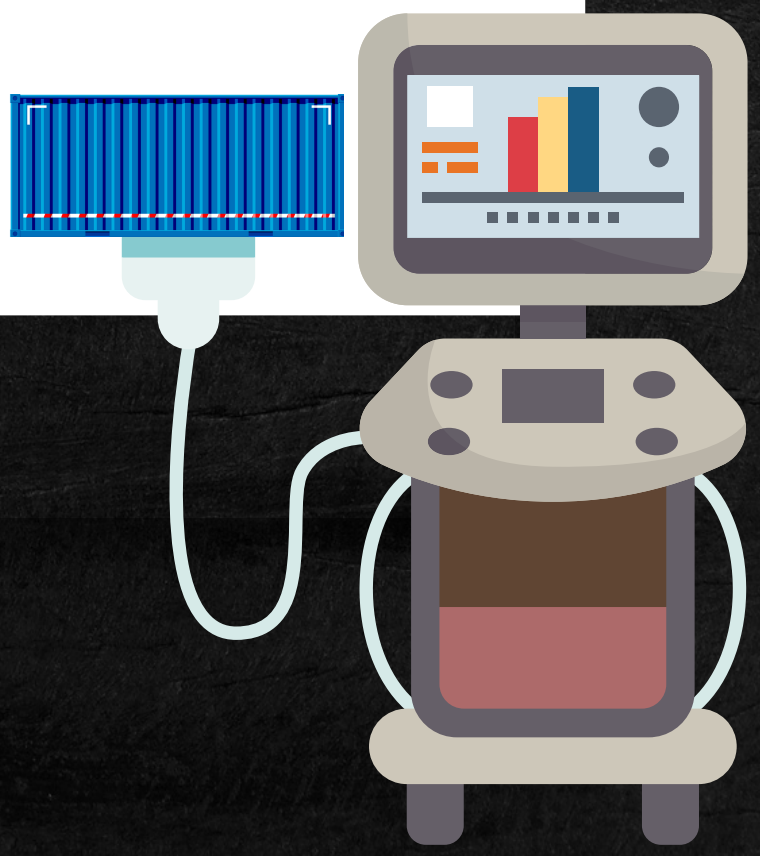


DevOpsOnSteroids

Rajani Ekunde

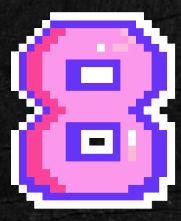
Monitor and Log Containers

Implement monitoring and logging mechanisms to gain insights into your containerized applications' performance and behavior. Docker provides built-in logging, or you can configure container logging to be redirected to external tools like ELK stack, Fluentd, or Splunk.



DevOpsOnSteroids

Rajani Ekunde



Regularly Update Containers

Stay up-to-date with the latest security patches and software updates for your container images. Regularly check for updates on your base images and application dependencies. Create a process to rebuild and redeploy containers with updated images, ensuring you have the latest bug fixes and security improvements.

DevOpsOnSteroids

Rajani Ekunde

DEVOPSONSTEROIDS

DO FOLLOW FOR MORE SUCH CONTENT AROUND CLOUD AND DEVOPS.

DevOpsOnSteroids