

# DOCKER

CONTAINER TEKNOLOJISI

Müberra ÇATAK

# İÇERİK

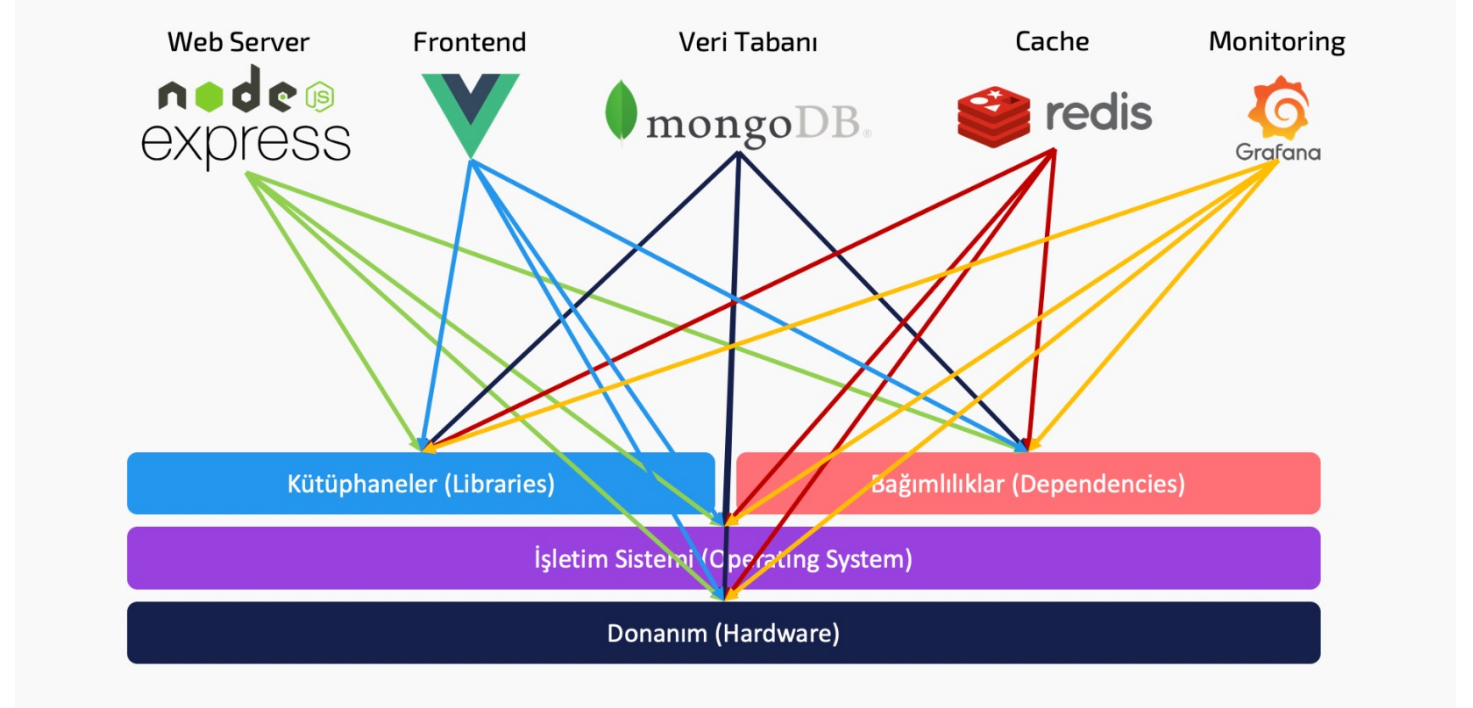
- Docker nedir ?
- Container nedir ?
- Image nedir ?
- Docker ve sanal makine arasındaki farklar nelerdir ?
- Docker komutları nelerdir ?
- Node.js projesini Dockerize etmek

# DOCKER NEDİR ?

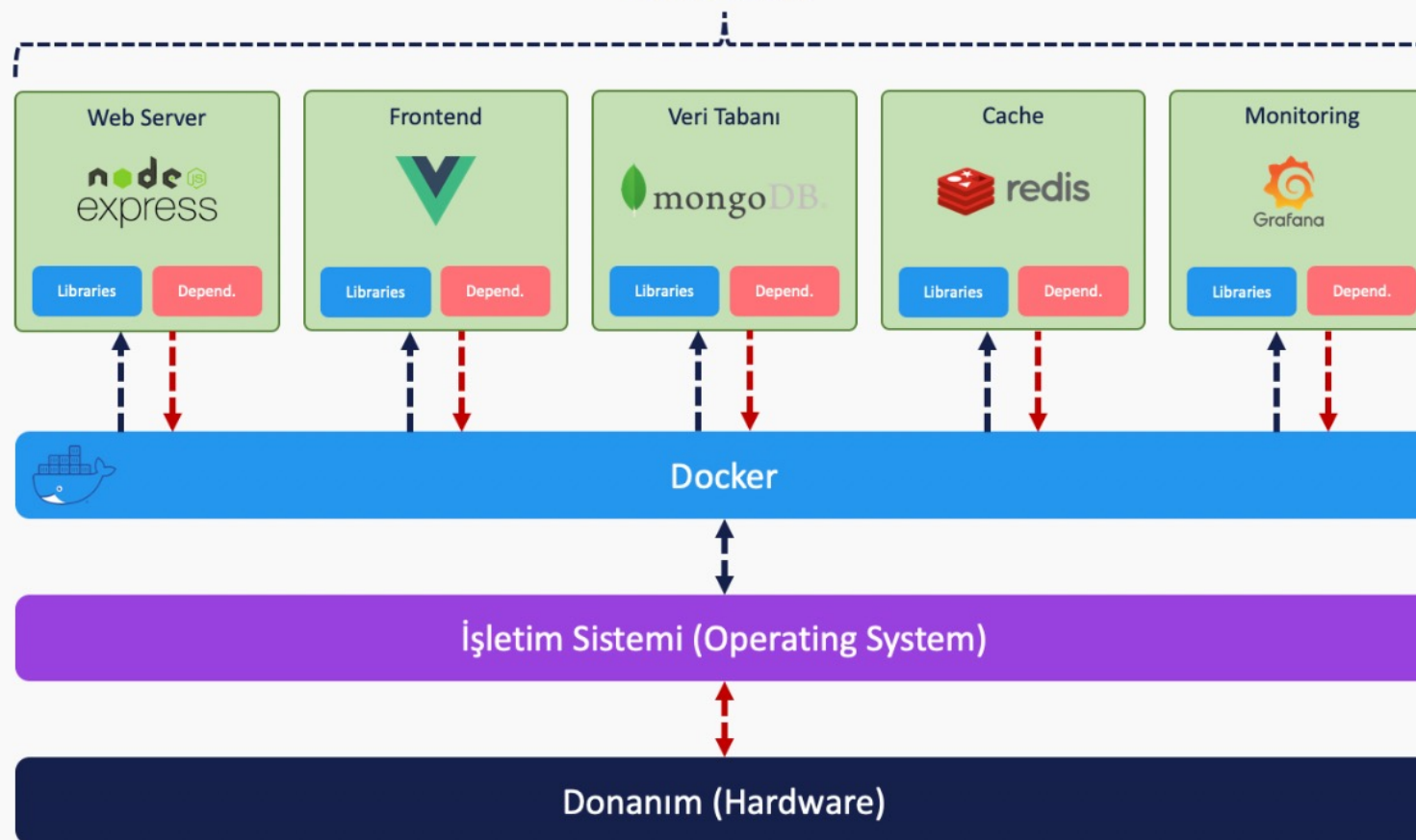
- Open source bir 'container' teknolojisidir.
- Aynı işletim sistemi üzerinde, yüzlerce hatta binlerce birbirinden izole ve bağımsız containerlar sayesinde sanallaştırma sağlayan bir teknolojidir.
- Web uygulamalarımızın kolayca kurulumunu, testini, çalışmasını ve deploymentını sağlar.
- Bunun yanında sunucu maliyetlerini önemli ölçüde azaltır.
- Docker, geliştirme ekiplerinin her yerde uygulamalar oluşturup yönetmesine ve güvenli hale getirmesine olanak tanıyan bir teknolojidir.

## NEDEN DOCKER'A İHTİYACIMIZ VAR ?

- Tüm bu uygulamalar OS ile uyumlu olacak şekilde çalışır. Bizim bu uyumu OS sürümüne göre her bir servis için ayrı ayrı sağlamamız gerekir.

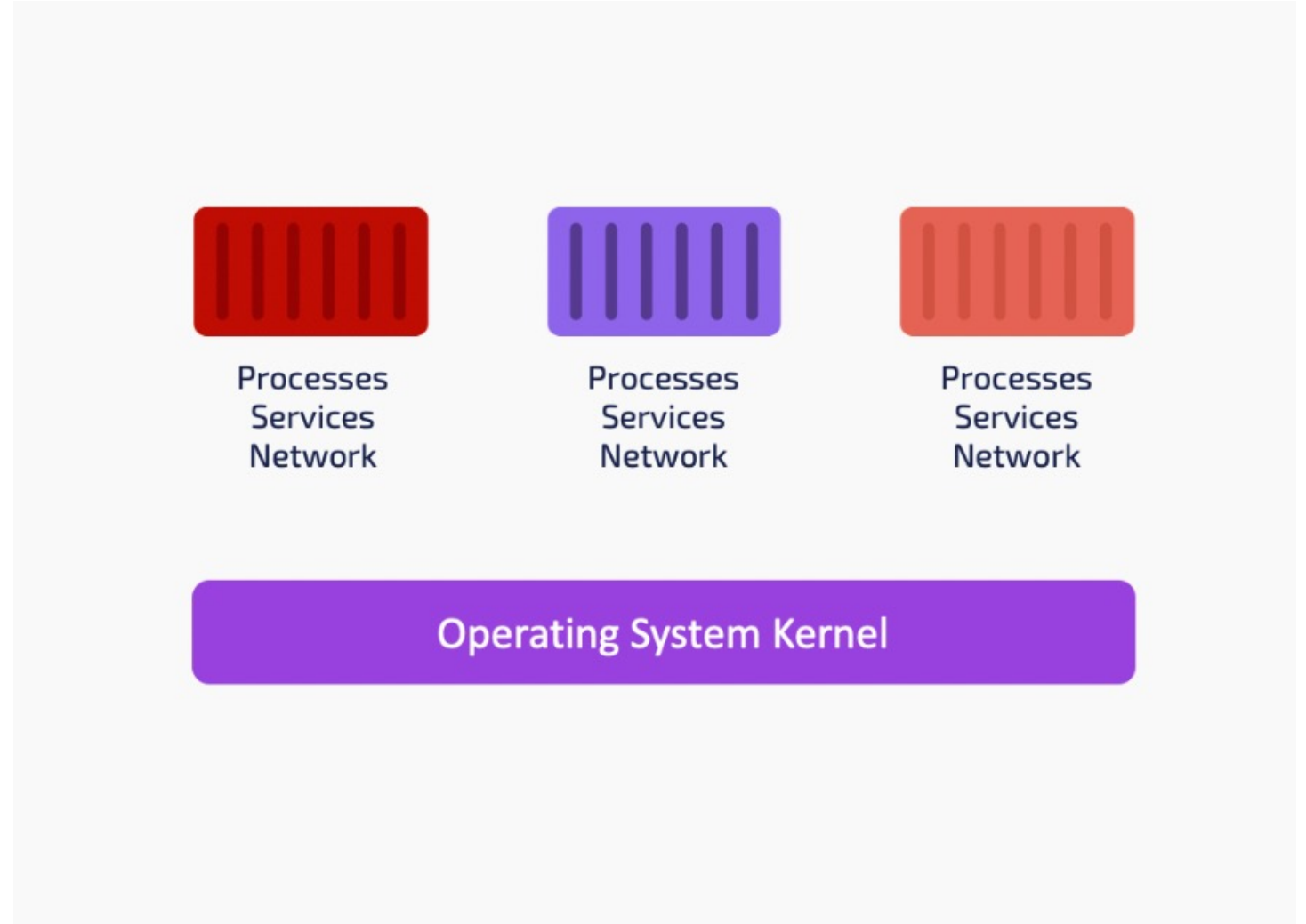


## Container




## CONTAINER NEDİR ?

- Container, kendilerine ait prosesleri, servisleri, networkleri bulunan tamamen izole edilmiş ortamlardır.
- Tıpkı VM gibi fakat her bir VM kendisine ait bir OS barındırırken her bir Container OS kernel'i paylaşmaktadır.



# IMAGE NEDİR ?

- Containerlar layer halindeki Image'lerden oluşur.
- Docker Image ise containerlara kurulacak ve run edilecek olan uygulamaların veya OS'lerin image dosyalarıdır.
- Örnek olarak mysql, mongodb, redis, ubuntu, mariadb verilebilir.

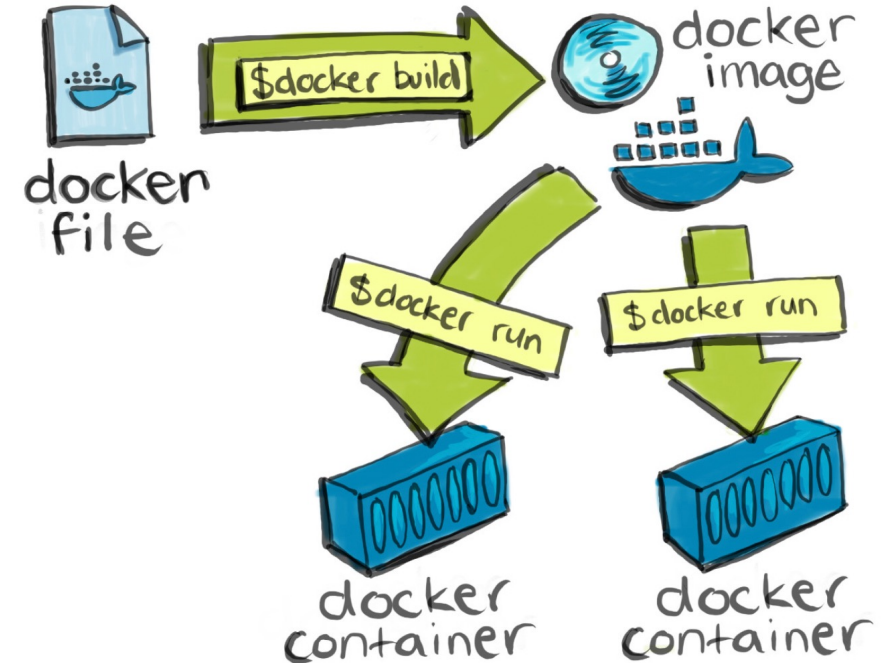
- 
- **Docker Image** sizin projeniz gibi düşünebilirsiniz. Projelerinizin dosyaları tüm ayarlar yani bir paket, template, plan gibi.
  - **Docker Container** ise bu template'in çalıştırılmış hali bir instance'ı gibi. Bir Docker Image'den birden fazla Docker Container çalıştırabilirsiniz.
  - Ayrıca siz de kendinize ait **image** dosyalarınızı üretebilir ve bunu Docker Hub Repository'e gönderebilirsiniz. Böylece Public ya da Private olarak diğer geliştiriciler bundan yararlanabilirler.



# DOCKER NASIL ÇALIŞIR ?

- LXC (Linux Container) üzerine kurulu bir teknoloji olan Docker, aynı linux işletim sistemi üzerinde birbirinden izole bir biçimde çalışan konteynerler oluşturmaktadır. Her bir konteyner bir process kullanmakta ve bu sayede konteynerler kolaylıkla çoğaltılabilmektedir. Ayrıca Docker, klasik sanal makinelerden (vmware, virtualbox vb.) farklı olarak bir Hypervisor katmanına sahip değildir. Bunun yerine Docker Engine üzerinden konak işletim sistemine erişerek sistem kaynaklarını paylaşımlı kullanır.

- İlk olarak, oluşturulmuş olan programın gerekli özelliklerini tanımlayan bir “Dockerfile” oluşturulur. (Örneğin; **FROM** komutu ile birlikte node.js isin hangi sürümünün kullanıldığı, **WORKDIR** ile ilgili çalışma dizinin adresi, **RUN** komutu ile de gerekli paketlerin kurulumu için “npm install” komutu belirtilebilir)
- Daha sonra terminal üzerinden “**docker build**” komutu ile birlikte, Dockerfile da belirtilen komutlar baz alınarak bir **Image** oluşturulmuş olunur.
- Oluşturulmuş olan **Image**, “**docker run**” komutu ile birlikte container halini alır.Ve ilgili işlemlerin gerçekleştirilmesi için Docker Deamon motoruna iletilir. Bu sayede ilgili uygulama container’lar üzerinden çalıştırılmış olur.



 package-lock.json

 Dockerfile 

 Dockerfile

```
1 FROM node:10
2 WORKDIR /home/node/app
3 COPY rest-api /home/node/app
4 RUN npm install
5 CMD npm run start
6 EXPOSE 3000
```

## AVANTAJLARI NELERDİR ?

- Docker, Hypervisor kullanmaz ve tam yüklü bir işletim sistemi barındırmaz bu sayede saniyeler içinde çalışır ve kullanıma hazır hale gelir.
- Docker, yazılımlarınızın tüm altyapı gereksinimlerini kod olarak saklar. (versiyonlama) Bu Docker'ın en önemli özelliklerinden biridir. Bu özellik sayesinde yazılımınızı farklı servis sağlayıcıları üzerinde kolaylıkla gezindirebilir, çoğaltabilir veya paylaşabilirsiniz.
- Docker, çok az kaynakla büyük işler yapabilmektedir. Bunun ana sebebi kullandığı konteyner teknolojisidir.
- Uygulamalarınıza beklenmedik bir şekilde yüksek trafik geldiğinde saniyeler içerisinde 1000'lerce konteyner hazır hale gelerek yükünüzü omuzlar.
- Docker, uygulamalarınızı standart bir zemine oturtarak her platformda aynı şekilde çalışmasını sağlar.

# DOCKER VE SANAL MAKİNE ARASINDAKİ FARKLAR

## VM (Virtual Machine)

- OS : **Tam işletim sistemi**
- İzolasyon : **Yüksek**
- Çalışır hale gelmesi : **Dakikalar**
- Versiyonlama : **Yok**
- Kolay paylaşılabilirlik : **Düşük**

## Docker

- OS : **Küçültülmüş işletim sistemi imajı**
- İzolasyon : **Daha düşük**
- Çalışır hale gelmesi : **Saniyeler**
- Versiyonlama : **Yüksek**
- Kolay paylaşılabilirlik : **Yüksek**

- Bir çok şirket kendisine ait bir çok uygulamayı containerized edip **DockerHub** üzerinden bunu public ya da private olarak sunuyor. Siz bunlardan birini kullanmak istediğinizde bunun sadece

```
docker run imageName
```

- şeklinde yazarak kendi ortamınıza bir kopyasını (instance) alabiliyorsunuz.
- Mesela;

```
docker run nodejs  
docker run redis  
docker mongodb
```

# DOCKER KOMUTLARI

- `docker run` : **run** komutu bir image'ı çalıştırmayı sağlar.
- `docker ps` : **ps** komutu çalışan tüm containerların listesini bilgileriyle beraber döker
- `docker build --tag node-rest-api .` : image oluşturur.
- `docker rm <container_id>` : ilgili containerı kaldırır.
- `docker rmi <image_id|image_name>` : ilgili imageı kaldırır.

# NODE.JS PROJESİNİ DOCKERIZE ETMEK

```
tubanurcatak@Tubanur-MacBook-Pro dockerize-nodejs-app-master % docker images
REPOSITORY          TAG          IMAGE ID       CREATED        SIZE
<none>              <none>       9a98326206b3  7 days ago    861MB
node-todoapp         latest       86e103df86b3  7 days ago    864MB
mongo               latest       595bfa82a067  2 weeks ago   656MB
docker/getting-started latest       157095baba98  8 weeks ago   27.4MB
hello-world         latest       46331d942d63  2 months ago  9.14kB
mongoclient/mongoclient latest       16ff4e68d176  20 months ago 1.18GB
tubanurcatak@Tubanur-MacBook-Pro dockerize-nodejs-app-master % docker build --tag node-rest-api .
[+] Building 0.0s (1/2)
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 2B                                                  0.0s
failed to solve with frontend dockerfile.v0: failed to read dockerfile: open /var/lib/docker/tmp/buildkit-mount757841260/Dockerfile: no such file or directory
tubanurcatak@Tubanur-MacBook-Pro dockerize-nodejs-app-master % docker build --tag node-rest-api .
[+] Building 13.8s (9/9) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 152B                                                0.0s
=> [internal] load .dockerignore                                                    0.0s
=> => transferring context: 2B                                                      0.0s
=> [internal] load metadata for docker.io/library/node:10                        2.3s
=> [internal] load build context                                                    0.0s
=> => transferring context: 56.10kB                                                 0.0s
=> [1/4] FROM docker.io/library/node:10@sha256:59531d2835edd5161c8f951          0.0s
=> CACHED [2/4] WORKDIR /home/node/app                                             0.0s
=> [3/4] COPY rest-api /home/node/app                                              0.0s
=> [4/4] RUN npm install                                                            11.2s
=> exporting to image                                                              0.2s
=> => exporting layers                                                              0.2s
=> => writing image sha256:c3bdb7243aa1c9827ce2ff873d4ab9bdf90bf1a09a          0.0s
=> => naming to docker.io/library/node-rest-api                                    0.0s
```

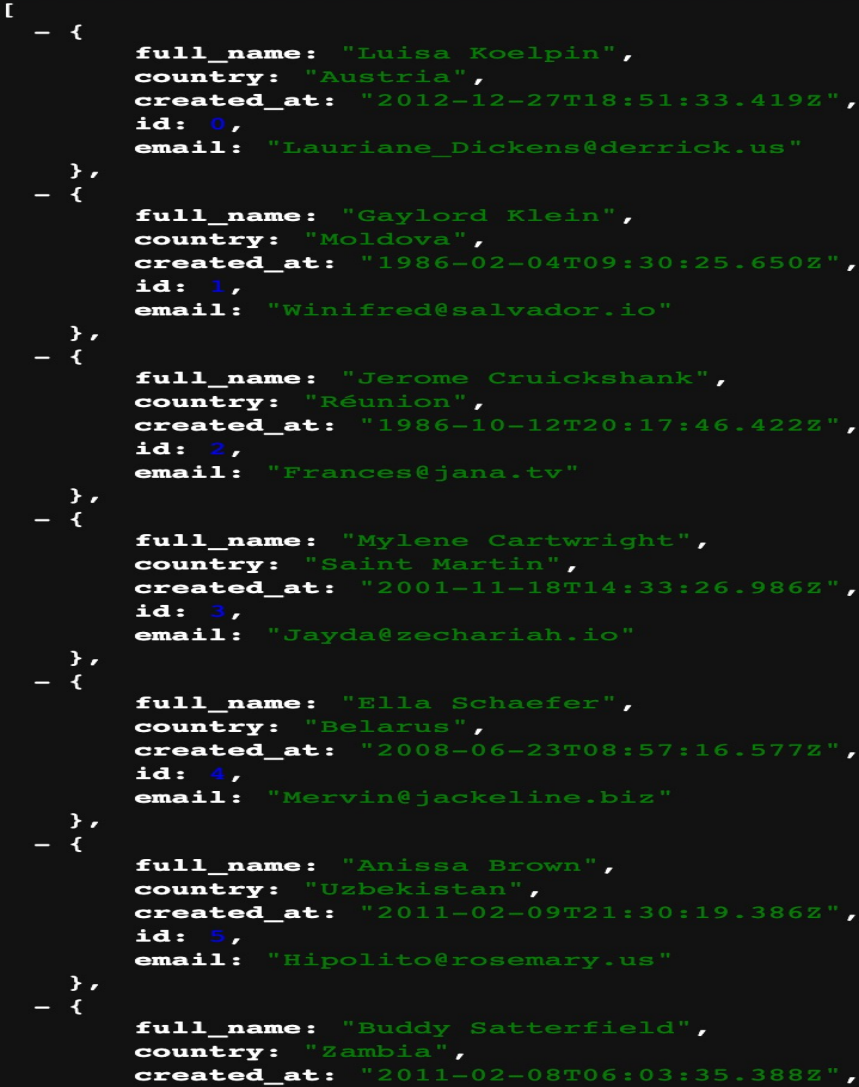
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them



```
tubanurcatak@Tubanur-MacBook-Pro dockerize-nodejs-app-master % docker images REPOSITORY TAG IMAGE ID CREATED SIZE
node-rest-api latest c3bdb7243aa1 14 seconds ago 869MB
<none> <none> 9a98326206b3 7 days ago 861MB
node-todoapp latest 86e103df86b3 7 days ago 864MB
mongo latest 595bfa82a067 2 weeks ago 656MB
docker/getting-started latest 157095baba98 8 weeks ago 27.4MB
hello-world latest 46331d942d63 2 months ago 9.14kB
mongoclient/mongoclient latest 16ff4e68d176 20 months ago 1.18GB
tubanurcatak@Tubanur-MacBook-Pro dockerize-nodejs-app-master % docker run node-rest-api

> node-js-rest-api-heroku-deployment@1.0.0 start /home/node/app
> node app.js

Sunucu ayaktaadır.. Çalışıyor...
```



```
tubanurcatak@Tubanur-MacBook-Pro dockerize-nodejs-app-master % docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
2e2bbc3c6c1f   node-rest-api  "docker-entrypoint.s..." 29 seconds ago Up 29 seconds  3000/tcp       jovial_satoshi
tubanurcatak@Tubanur-MacBook-Pro dockerize-nodejs-app-master % docker kill 2e2bbc3c6c1f
2e2bbc3c6c1f
```

```
tubanurcatak@Tubanur-MacBook-Pro dockerize-nodejs-app-master % docker run -p 3000:3000 node-rest-api
```

```
> node-js-rest-api-heroku-deployment@1.0.0 start /home/node/app
> node app.js
```

Sunucu ayaktaadır.. Çalışıyor...

Thu, 09 Jun 2022 21:55:45 GMT express deprecated res.send(status, body): Use res.status(status).send(body) instead at app.js:9:7



DİNLEDİĞİNİZ İÇİN TEŞEKKÜR EDERİM

Müberra ÇATAK