# Introduction to Docker and Docker Compose

Christoff Visser

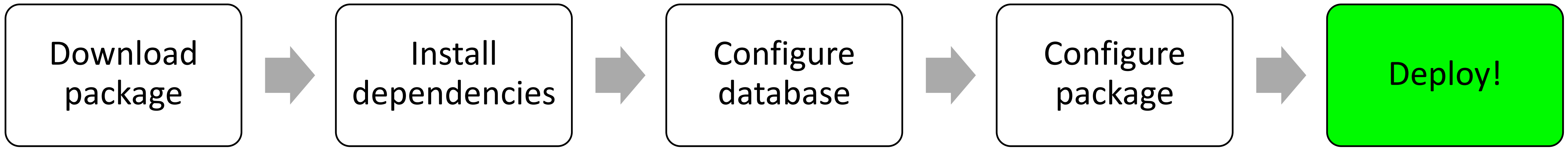
christoff@iij.ad.jp

Internet Initiative Japan

## Objectives

- Understand Docker and Docker Compose concepts
- Learn to install Docker and Docker Compose
- Create and manage Docker images and containers
- Understand how Docker Compose can simplify multi-container application deployments

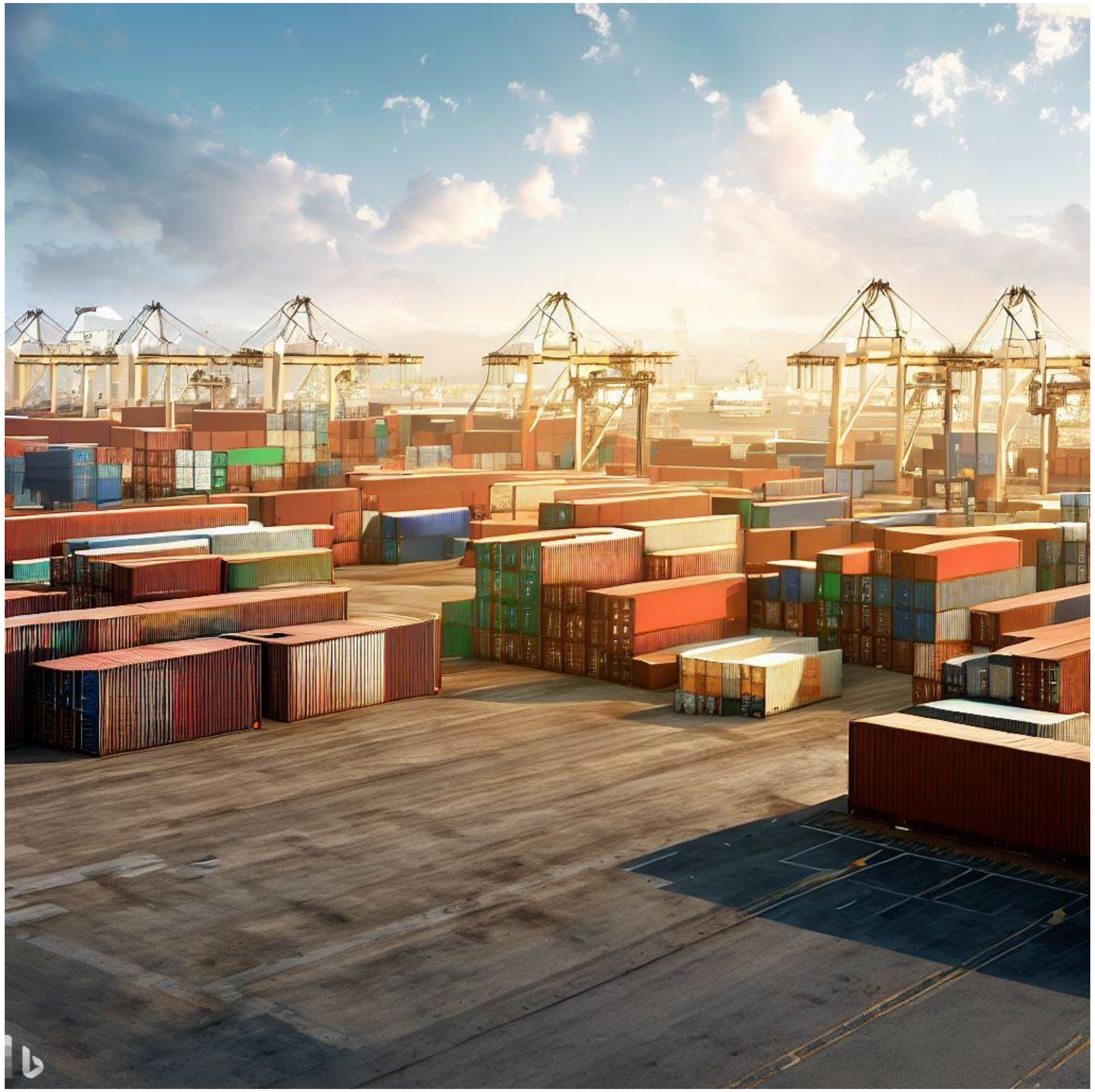
## Typical software deployment workflow



## Challenges with traditional software deployment

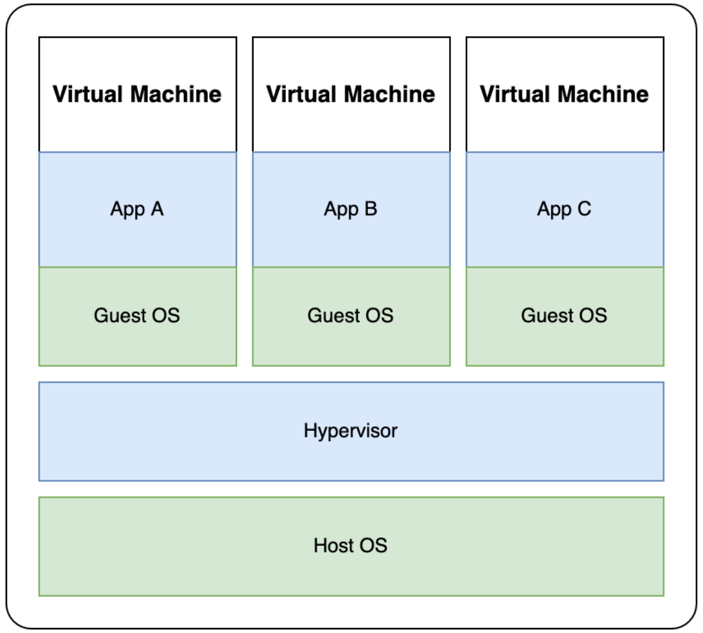
1. Takes time to go through documentation, install package and maintain it
2. Time consuming process to transfer software to a different server
3. Prone to errors and mistakes
4. “Works on my system”
5. Dependency conflicts

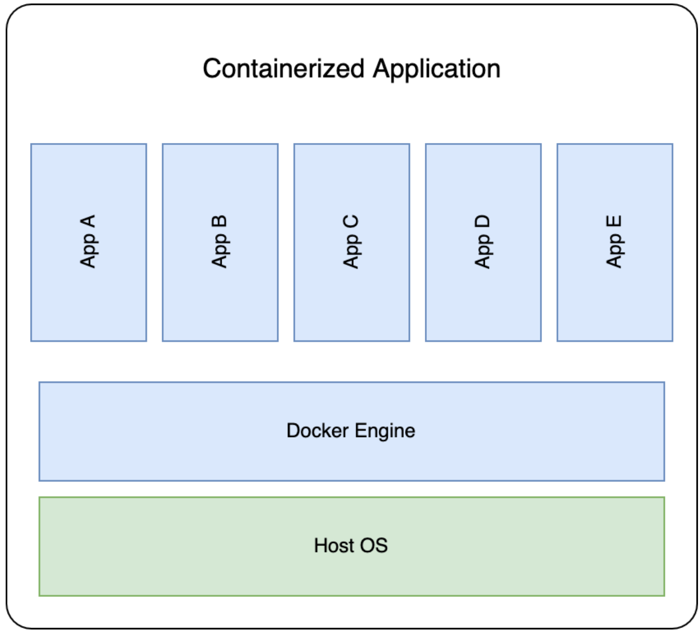
## Introducing containers



# Virtual Machines vs Containers

- Containers are very lightweight and run the bare metal to run an app
- Unlike VMs, containers come preloaded with the application and dependencies
- One container == One application
- Applications can be spread over multiple containers
- Database - Application– Redis – Web Frontend
- Developers build images that contain applications

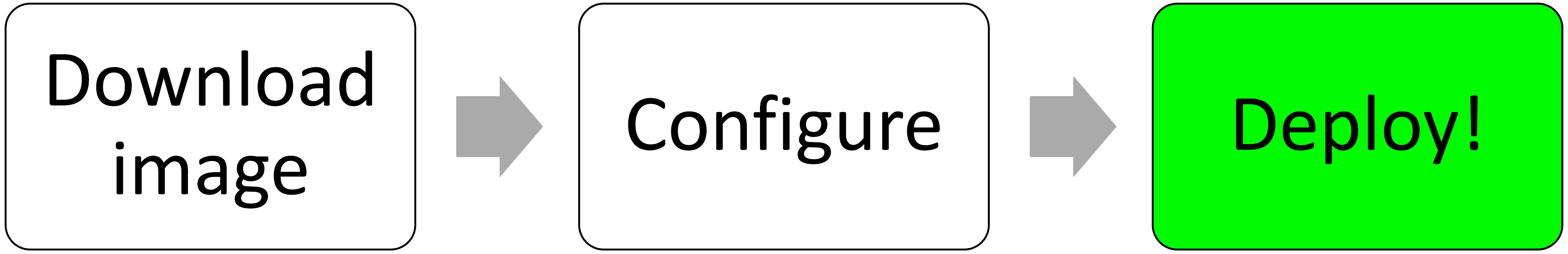




## What is Docker?

- Lightweight platform to isolate applications using containers
- Docker containers contain all the needed binaries, dependancies, configurations, etc
- Greatly simplifies deploying, managing and updating applications
- Containers are self contained, making them very portable with only configuration and data needing backups and migration

## Container based software deployment



# Installing Docker

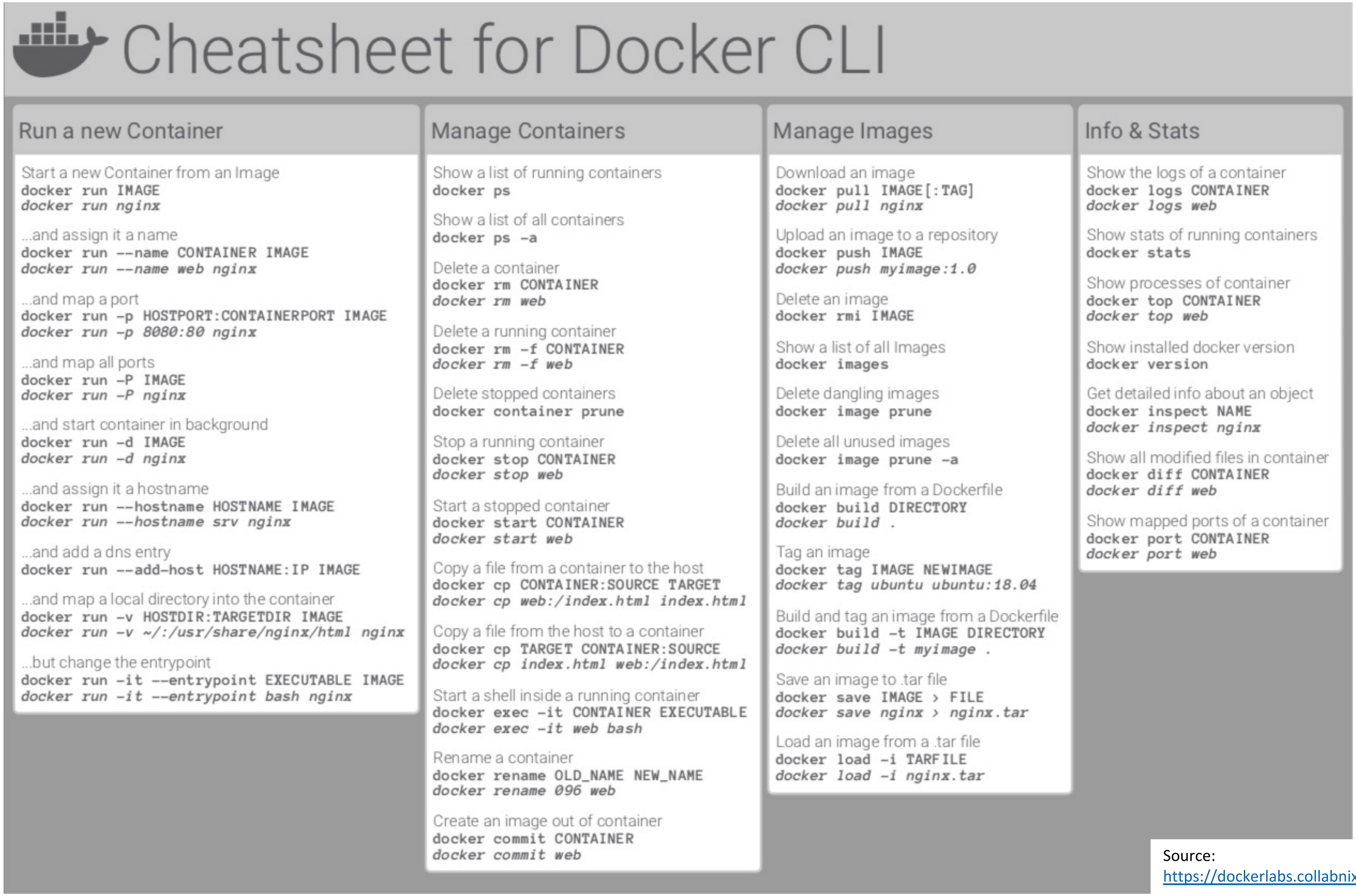
- Docker Desktop
  - Runs a small linux VM on your machine to run containers in
  - Available for Linux, Windows and MacOS (Intel + Apple Silicon)

- Docker Server
  - Runs natively
  - Available only on linux

- Install instructions: https://docs.docker.com/engine/install/

## Docker terms

- Docker Engine - The ”engine” used to interact with docker containers.
- Docker Containers – Self contained applications. Typically 1 container is only running 1 application
- Images – Pre-packaged applications that are used to spin up containers. A running image is simply a container
- Container Registry – A registry service used to host and share images. These can be pulled and used
- Networks – Docker will create it’s own internal docker network on a new bridge called docker0. One can create more networks. The ports of a container can also be mapped to ports on the host, e.g. port 80 from the container to port 8080 on the host.
- Volumes – Volumes can be created and attached to containers
- Bind mounts – Used to mount an existing location to a container



# Cheatsheet for Docker CLI

## Run a new Container

Start a new Container from an Image
```
docker run IMAGE
docker run nginx
```
...and assign it a name
```
docker run --name CONTAINER IMAGE
docker run --name web nginx
```
...and map a port
```
docker run -p HOSTPORT:CONTAINERPORT IMAGE
docker run -p 8080:80 nginx
```
...and map all ports
```
docker run -P IMAGE
docker run -P nginx
```
...and start container in background
```
docker run -d IMAGE
docker run -d nginx
```
...and assign it a hostname
```
docker run --hostname HOSTNAME IMAGE
docker run --hostname srv nginx
```
...and add a dns entry
```
docker run --add-host HOSTNAME:IP IMAGE
```
...and map a local directory into the container
```
docker run -v HOSTDIR:TARGETDIR IMAGE
docker run -v ~/:/usr/share/nginx/html nginx
```
...but change the entrypoint
```
docker run -it --entrypoint EXECUTABLE IMAGE
docker run -it --entrypoint bash nginx
```

## Manage Containers

Show a list of running containers
```
docker ps
```
Show a list of all containers
```
docker ps -a
```
Delete a container
```
docker rm CONTAINER
docker rm web
```
Delete a running container
```
docker rm -f CONTAINER
docker rm -f web
```
Delete stopped containers
```
docker container prune
```
Stop a running container
```
docker stop CONTAINER
docker stop web
```
Start a stopped container
```
docker start CONTAINER
docker start web
```
Copy a file from a container to the host
```
docker cp CONTAINER:SOURCE TARGET
docker cp web:/index.html index.html
```
Copy a file from the host to a container
```
docker cp TARGET CONTAINER:SOURCE
docker cp index.html web:/index.html
```
Start a shell inside a running container
```
docker exec -it CONTAINER EXECUTABLE
docker exec -it web bash
```
Rename a container
```
docker rename OLD_NAME NEW_NAME
docker rename 096 web
```
Create an image out of container
```
docker commit CONTAINER
docker commit web
```

## Manage Images

Download an image
```
docker pull IMAGE[:TAG]
docker pull nginx
```
Upload an image to a repository
```
docker push IMAGE
docker push myimage:1.0
```
Delete an image
```
docker rmi IMAGE
```
Show a list of all Images
```
docker images
```
Delete dangling images
```
docker image prune
```
Delete all unused images
```
docker image prune -a
```
Build an image from a Dockerfile
```
docker build DIRECTORY
docker build .
```
Tag an image
```
docker tag IMAGE NEWIMAGE
docker tag ubuntu ubuntu:18.04
```
Build and tag an image from a Dockerfile
```
docker build -t IMAGE DIRECTORY
docker build -t myimage .
```
Save an image to .tar file
```
docker save IMAGE > FILE
docker save nginx > nginx.tar
```
Load an image from a .tar file
```
docker load -i TARFILE
docker load -i nginx.tar
```

## Info & Stats

Show the logs of a container
```
docker logs CONTAINER
docker logs web
```
Show stats of running containers
```
docker stats
```
Show processes of container
```
docker top CONTAINER
docker top web
```
Show installed docker version
```
docker version
```
Get detailed info about an object
```
docker inspect NAME
docker inspect nginx
```
Show all modified files in container
```
docker diff CONTAINER
docker diff web
```
Show mapped ports of a container
```
docker port CONTAINER
docker port web
```

Source:
https://dockerlabs.collabnix.com/docker/cheatsheet/

## Running containers demo

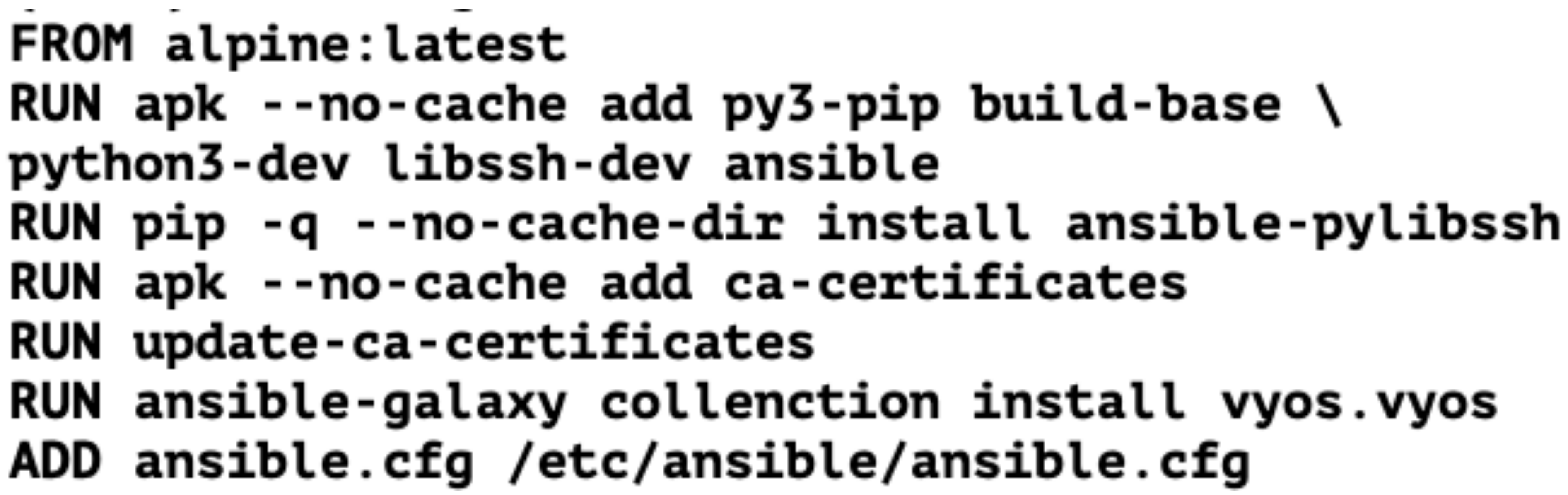
## Dockerfile - simple

```
FROM alpine
CMD [“echo”, “Hello World”]
```

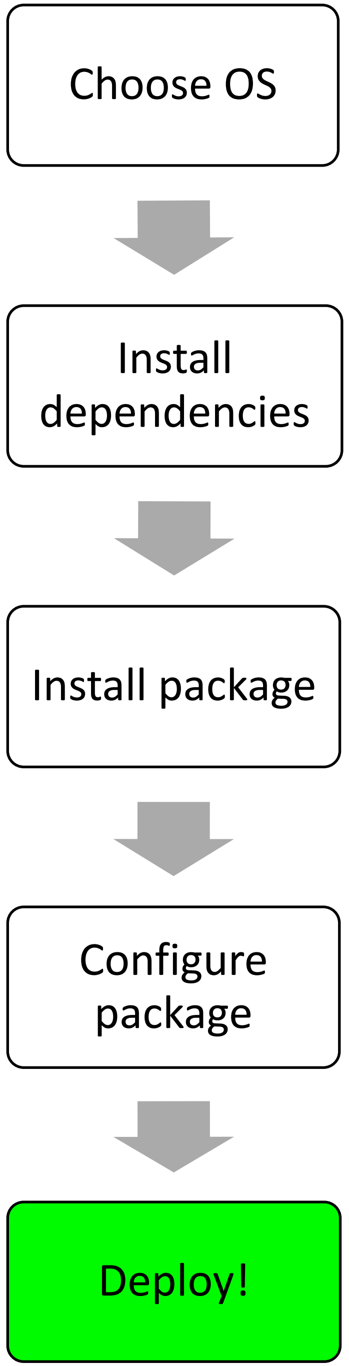
Pull a base Alpine image to build FROM
Runs the CMD echo Hello World

After this command is run it will exit

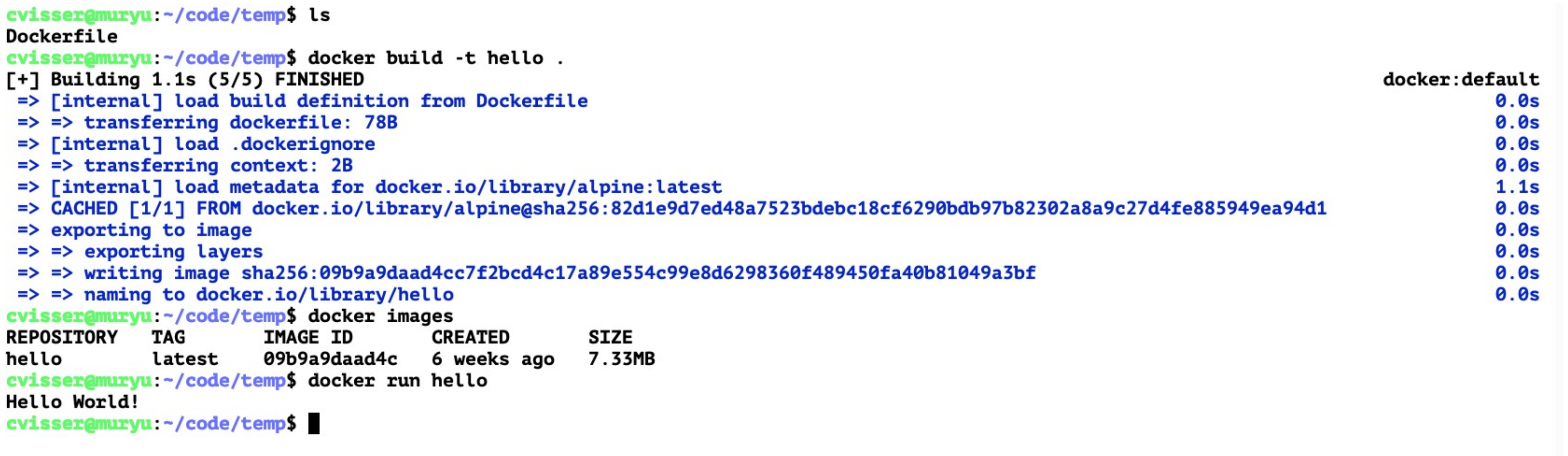
# Building a Container - ansible



```
FROM alpine:latest
RUN apk --no-cache add py3-pip build-base \
python3-dev libssh-dev ansible
RUN pip -q --no-cache-dir install ansible-pylibssh
RUN apk --no-cache add ca-certificates
RUN update-ca-certificates
RUN ansible-galaxy collenction install vyos.vyos
ADD ansible.cfg /etc/ansible/ansible.cfg
```

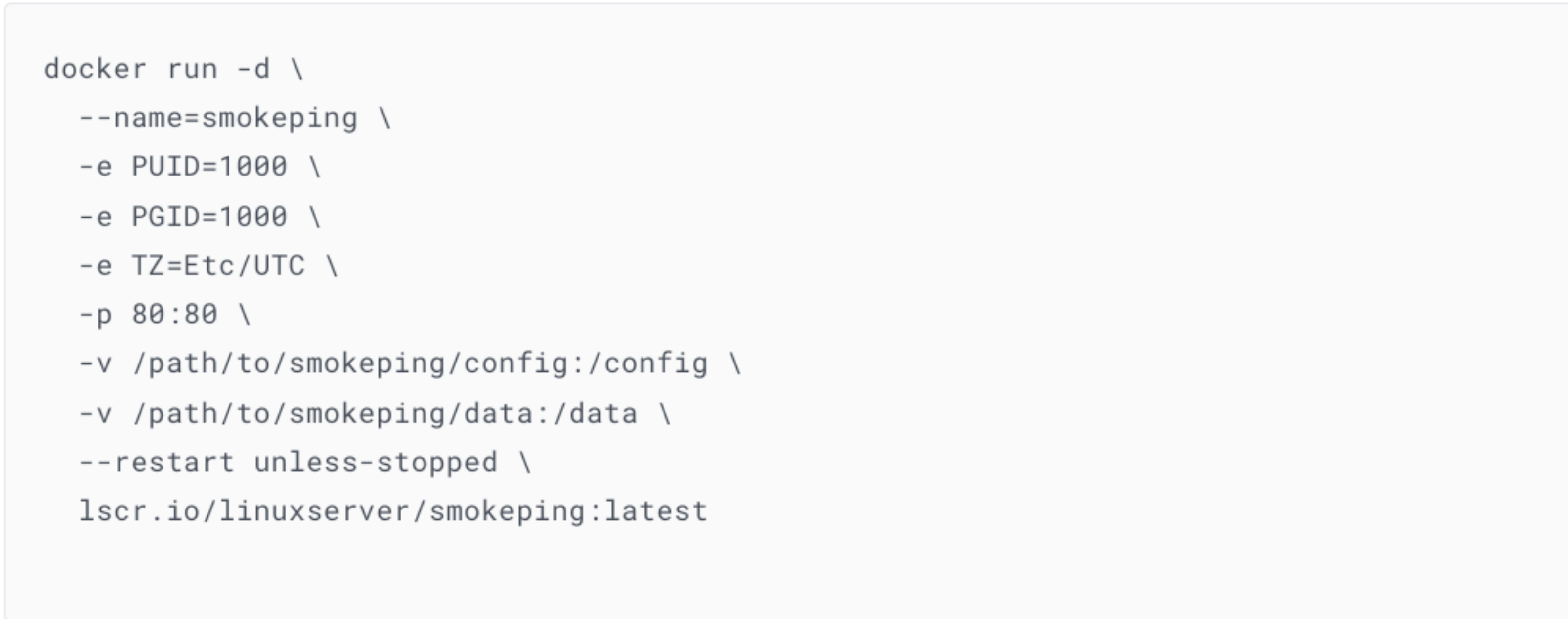


## How to build the image



```
cvisser@muryu:~/code/temp$ ls
Dockerfile
cvisser@muryu:~/code/temp$ docker build -t hello .
[+] Building 1.1s (5/5) FINISHED                                                                                            docker:default
 => [internal] load build definition from Dockerfile                                                                                  0.0s
 => => transferring dockerfile: 78B                                                                                                   0.0s
 => [internal] load .dockerignore                                                                                                     0.0s
 => => transferring context: 2B                                                                                                       0.0s
 => [internal] load metadata for docker.io/library/alpine:latest                                                                      1.1s
 => CACHED [1/1] FROM docker.io/library/alpine@sha256:82d1e9d7ed48a7523bdebc18cf6290bdb97b82302a8a9c27d4fe885949ea94d1                0.0s
 => exporting to image                                                                                                                0.0s
 => => exporting layers                                                                                                               0.0s
 => => writing image sha256:09b9a9daad4cc7f2bcd4c17a89e554c99e8d6298360f489450fa40b81049a3bf                                          0.0s
 => => naming to docker.io/library/hello                                                                                              0.0s
cvisser@muryu:~/code/temp$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
hello        latest    09b9a9daad4c   6 weeks ago   7.33MB
cvisser@muryu:~/code/temp$ docker run hello
Hello World!
cvisser@muryu:~/code/temp$ █
```

## Running docker ad-hoc



```
docker run -d \
  --name=smokeping \
  -e PUID=1000 \
  -e PGID=1000 \
  -e TZ=Etc/UTC \
  -p 80:80 \
  -v /path/to/smokeping/config:/config \
  -v /path/to/smokeping/data:/data \
  --restart unless-stopped \
  lscr.io/linuxserver/smokeping:latest
```

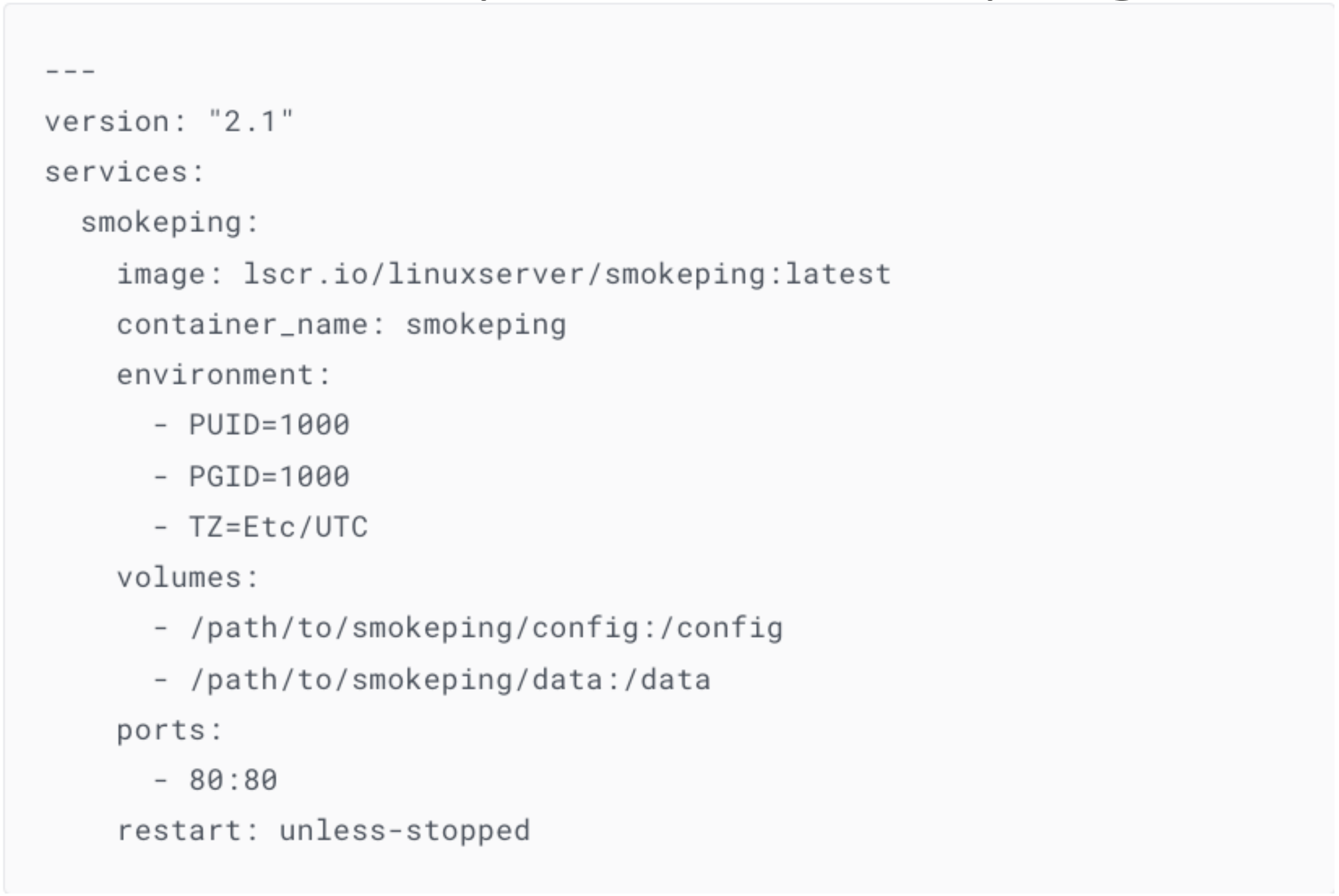
## Docker ad-hoc challenges

- Gets complicated the more arguments are passed
- Hard to remember all previously used arguments
- Easy to misconfigure when running multiple ad-hoc containers

## docker-compose.yml

- Single yaml text file for multiple containers
- Easier to read and includes all instructions for all containers
- Simplify creating/attaching volumes to bind to
- Ensure you’re exposing only what you need to
- Simplify upgrading and maintaining containers

## Docker compose - smokeping



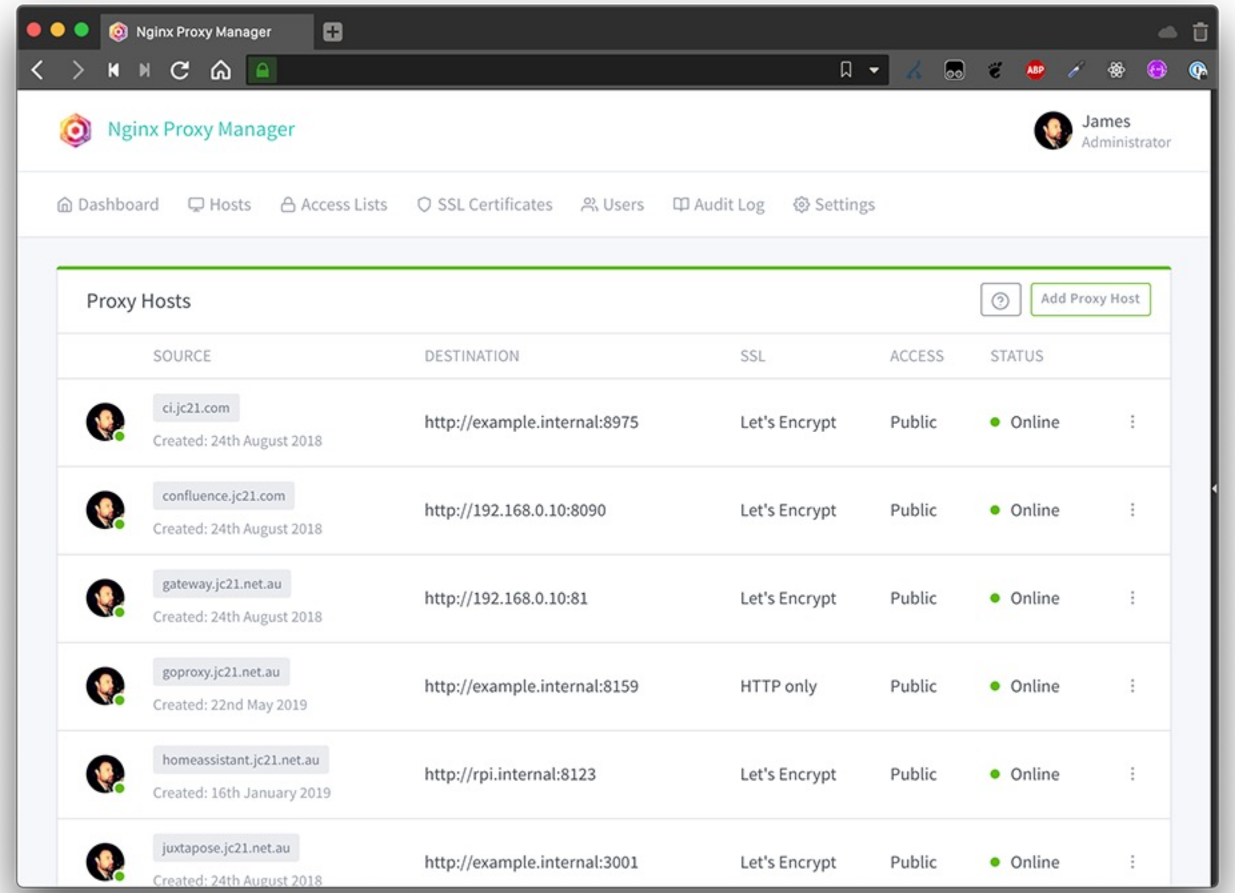
```yaml
---
version: "2.1"
services:
  smokeping:
    image: lscr.io/linuxserver/smokeping:latest
    container_name: smokeping
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Etc/UTC
    volumes:
      - /path/to/smokeping/config:/config
      - /path/to/smokeping/data:/data
    ports:
      - 80:80
    restart: unless-stopped
```

## How to deal with a lot of containers

- Introducing a reverse proxy
- Sits between outside world and your containers
- Map internal port numbers to a DNS hostname on port 80 and 443
- Simplifies deploying SSL certificates
- Simplify dual stacking, make your apps available on IPv6 as well
- Can be run as a docker container as well

# Nginx Proxy Manager

- Web interface for simple setup
- Automatically updates SSL certificates and forces HTTPS
- No need to expose ports, Nginx Proxy Manager will do it for you
- [https://nginxproxymanager.com/](https://nginxproxymanager.com/)



Proxy Hosts

| SOURCE | DESTINATION | SSL | ACCESS | STATUS |
|---|---|---|---|---|
| ci.jc21.com Created: 24th August 2018 | http://example.internal:8975 | Let's Encrypt | Public | ● Online |
| confluence.jc21.com Created: 24th August 2018 | http://192.168.0.10:8090 | Let's Encrypt | Public | ● Online |
| gateway.jc21.net.au Created: 24th August 2018 | http://192.168.0.10:81 | Let's Encrypt | Public | ● Online |
| goproxy.jc21.net.au Created: 22nd May 2019 | http://example.internal:8159 | HTTP only | Public | ● Online |
| homeassistant.jc21.net.au Created: 16th January 2019 | http://rpi.internal:8123 | Let's Encrypt | Public | ● Online |
| juxtapose.jc21.net.au Created: 24th August 2018 | http://example.internal:3001 | Let's Encrypt | Public | ● Online |

## Notes on backup

- Docker containers are reproducible. No need to backup
- User data is stored using volumes or bind mounts
    - Only these need to be backed up
- Popular tools like Restic Duplicati (can be run as docker container)
- Always encrypt data before storing it on the cloud

## Some containers to play with

- RIPE Atlas - https://hub.docker.com/r/jamesits/ripe-atlas
- HTML 5 speedtest - https://hub.docker.com/r/adolfintel/speedtest
- iperf3 - https://hub.docker.com/r/networkstatic/iperf3
- Nextcloud - https://hub.docker.com/_/nextcloud
- Docker-speedtest-grafana - https://github.com/frdmn/docker-speedtest-grafana
- Kerberos - https://doc.kerberos.io/opensource/installation#docker
- Linux-server.io - Many great images actively maintained by the open source community

## Questions?

christoff@iij.ad.jp

IIJ
Internet Initiative Japan