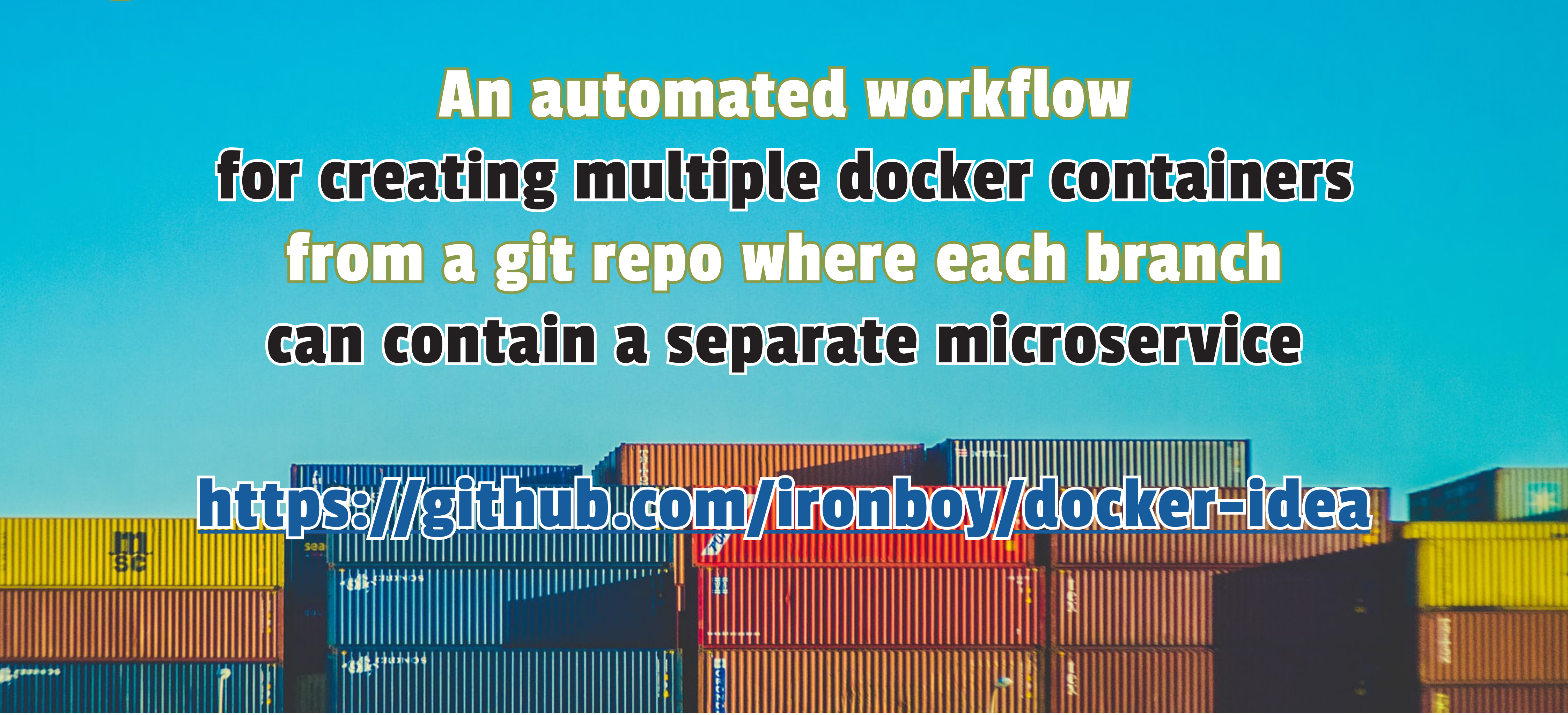
Thomas Frank, CEO, Node Hill AB, 2022



An automated workflow for creating multiple docker containers from a git repo where each branch can contain a separate microservice

https://github.com/ironboy/docker-idea

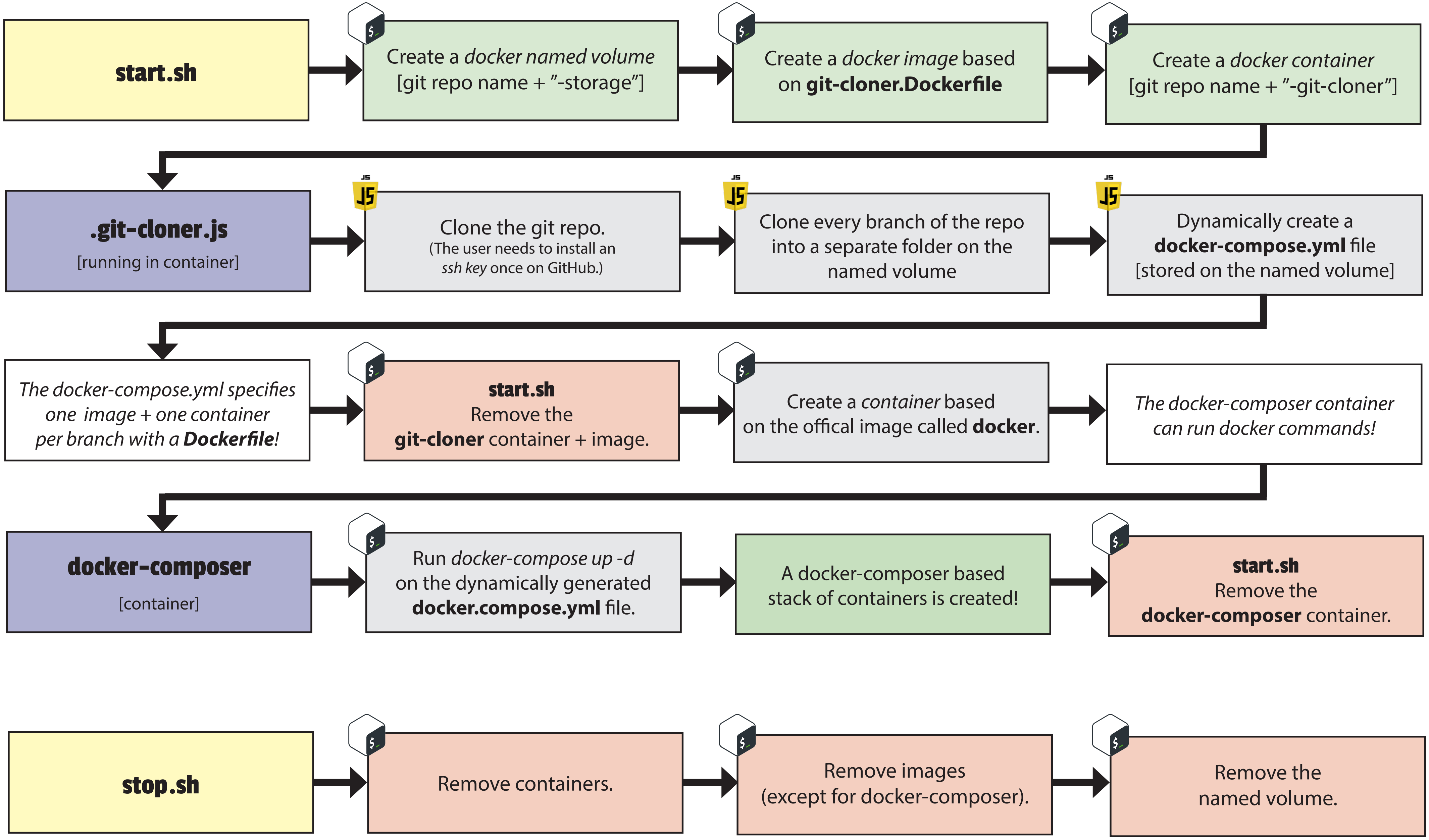
docker

JS

docker Compose

An automated workflow for creating multiple docker containers from a git repo where each branch can contain a separate microservice

NODEHILL



start.sh

Create a *docker named volume* [git repo name + ”-storage”]

Create a *docker image* based on **git-cloner.Dockerfile**

Create a *docker container* [git repo name + ”-git-cloner”]

.git-cloner.js
[running in container]

Clone the git repo.
(The user needs to install an *ssh key* once on GitHub.)

Clone every branch of the repo into a separate folder on the named volume

Dynamically create a **docker-compose.yml** file [stored on the named volume]

The docker-compose.yml specifies one image + one container per branch with a ***Dockerfile****!*

start.sh
Remove the **git-cloner** container + image.

Create a *container* based on the offical image called **docker**.

The docker-composer container can run docker commands!

docker-composer
[container]

Run *docker-compose up -d* on the dynamically generated **docker.compose.yml** file.

A docker-composer based stack of containers is created!

start.sh
Remove the **docker-composer** container.

stop.sh

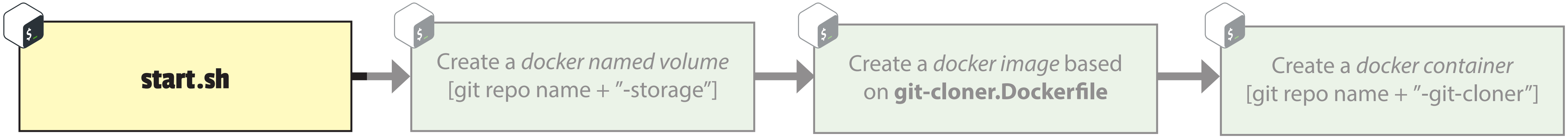
Remove containers.

Remove images (except for docker-composer).

Remove the named volume.

An automated workflow for creating multiple docker containers from a git repo where each branch can contain a separate microservice

NODEHILL



start.sh

Create a *docker named volume* [git repo name + "-storage"]

Create a *docker image* based on **git-cloner.Dockerfile**

Create a *docker container* [git repo name + "-git-cloner"]

Some initial setup

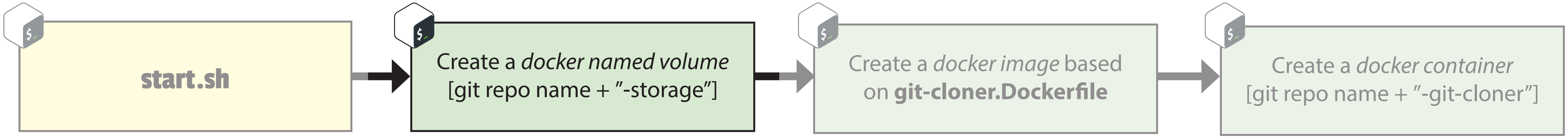
Getting som directory paths and the name of the repository and the current branch:

```
#!/bin/sh

# Get the dir path of the dir where this script is located
DIRNAME=$( cd "$(dirname "${BASH_SOURCE[0]}")" ; pwd -P )

# Get the repo dir path
REPO_DIR="$(dirname "$DIRNAME")"

# Cd to the dir where this script is located
cd $DIRNAME

### get the name of the repository
REPO_NAME=$(basename -s .git `git config --get remote.origin.url`)

## get the name of the checked out branch
BRANCH_NAME=$(git rev-parse --abbrev-ref HEAD)
```

~~And then, *not so important,* we don't run the script if we are in the docker branch. We *could* but it seems meaningsless since the deveoper is supposed to run it in a branch that contains the code for one of the microservices:~~

```

if [[ "$BRANCH_NAME" == "docker" ]]
then
  echo ""
  echo "I WOULD LOVE TO START YOUR DOCKER COMPOSE STACK BUT:"
  echo "Do not start from the docker branch!"
  echo "Instead start from the branch you want to bind mount!"
  echo ""
  echo "Run ./create-docker-tools.sh in the docker branch"
  echo "to create start and stop scripts available everywhere!"
  echo ""
  exit 1
fi

```

Bash-script, cheat-sheet:
https://devhints.io/bash



start.sh

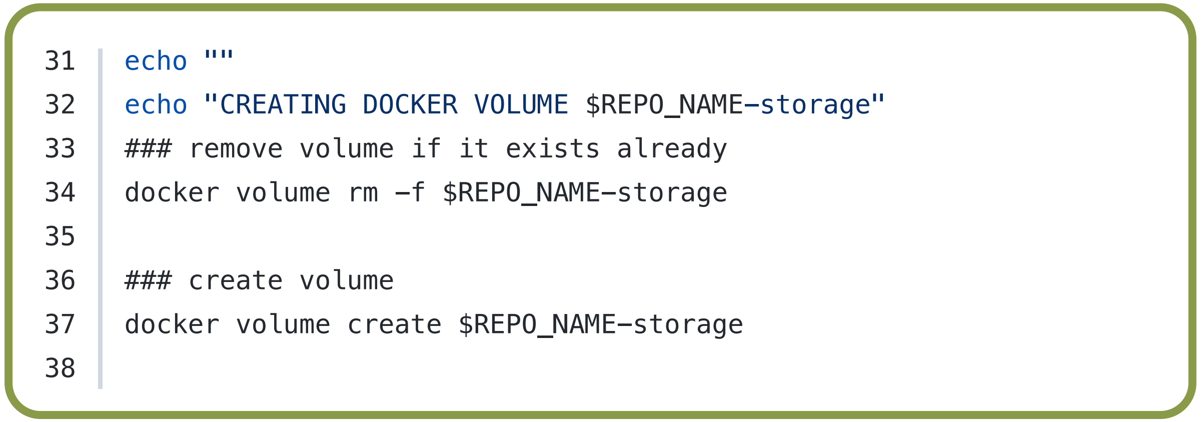
Create a *docker named volume* [git repo name + ”-storage”]

Create a *docker image* based on **git-cloner.Dockerfile**

Create a *docker container* [git repo name + ”-git-cloner”]

Create a named volume

Remove the volume if it exists since earlier runs, then create the named volume:



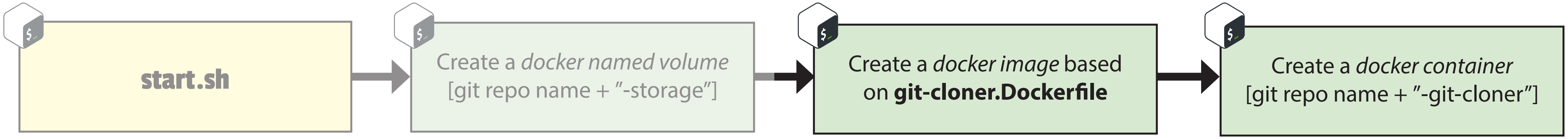
```
echo ""
echo "CREATING DOCKER VOLUME $REPO_NAME-storage"
### remove volume if it exists already
docker volume rm -f $REPO_NAME-storage

### create volume
docker volume create $REPO_NAME-storage

```

https://docs.docker.com/storage/volumes

https://www.baeldung.com/ops/docker-volumes

An automated workflow for creating multiple docker containers from a git repo where each branch can contain a separate microservice

NODEHILL


start.sh

Create a *docker named volume* [git repo name + ”-storage”]

Create a *docker image* based on **git-cloner.Dockerfile**

Create a *docker container* [git repo name + ”-git-cloner”]

Create an image – ”git-cloner” – and run as a container

Create an image, based on an official Node.js Buster (Debian) image that will run our the JS-part of our automation, git-cloner.js. Start it as a container, connect it to our named volume and send som important environment variables to it.

```
# build image from Dockerfile
docker build -f git-cloner.Dockerfile -t $REPO_NAME-git-cloner .

### run image as container
docker run \
--name $REPO_NAME-git-cloner \
-v $REPO_NAME-storage:/storage \
-e GIT_REPO_URL=$(git remote get-url origin) \
-e GIT_USERNAME=$(git config --global user.name) \
-e GIT_EMAIL=$(git config --global user.email) \
-e GIT_REPO_NAME=$REPO_NAME \
-e GIT_BRANCH_NAME=$BRANCH_NAME \
-e HOST_REPO_PATH=$REPO_DIR \
$REPO_NAME-git-cloner

```

The **git-cloner.Dockerfile**:

```
# start with a debian node container
FROM node:16.15-buster

# Install git
RUN apt update
RUN apt install git

# Set a work dir (working directory)
WORKDIR /app

# Copy the contents of copy-to-docker-container folder
# to the work dir
COPY ./copy-to-docker-container .

# Run the git-cloner.js file from copy-to-docker-container
CMD node git-cloner.js
```

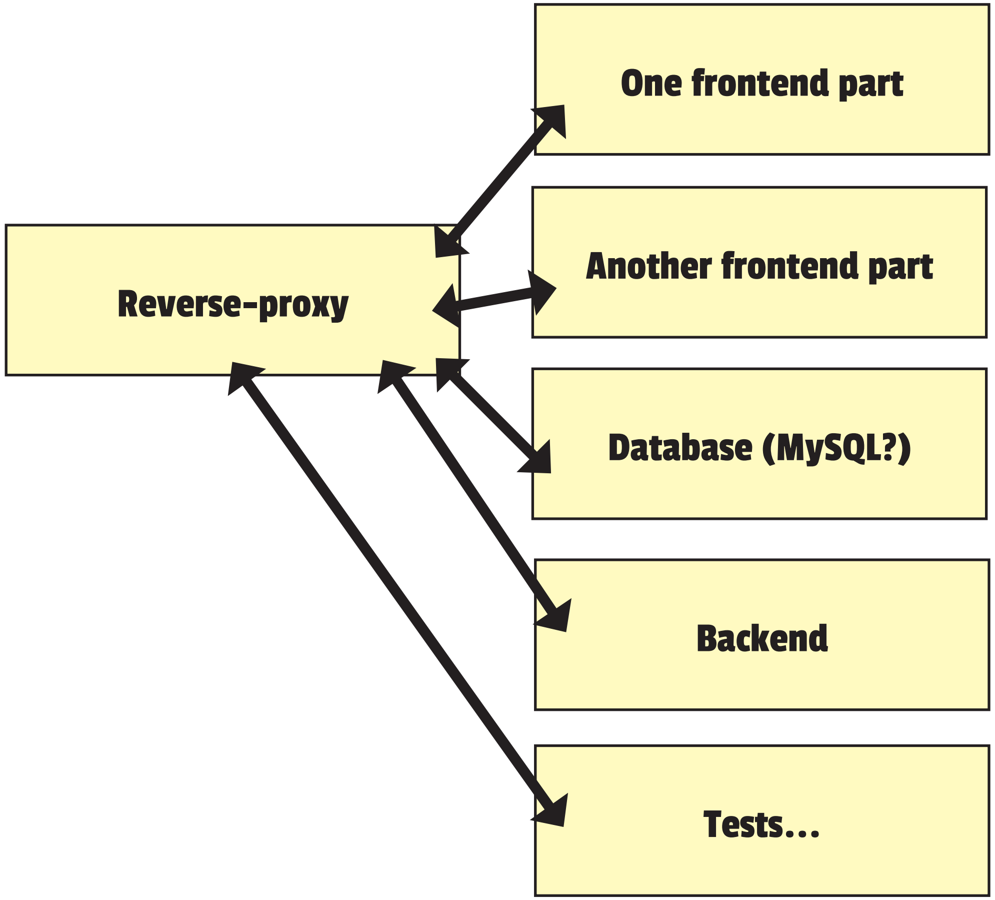
https://www.baeldung.com/ops/docker-container-environment-variables

https://dev.to/aws-builders/understanding-the-dockerfile-format-3cc6

Thomas, todo:

- Finish writing documentation!
- Create an example application consisting of several containers (microservices)…

Sketch for example application:


One frontend part
Reverse-proxy
Another frontend part
Database (MySQL?)
Backend
Tests...