# Building your own Kubernetes and Docker

Erik Bernhardsson

# I’m Erik Bernhardsson

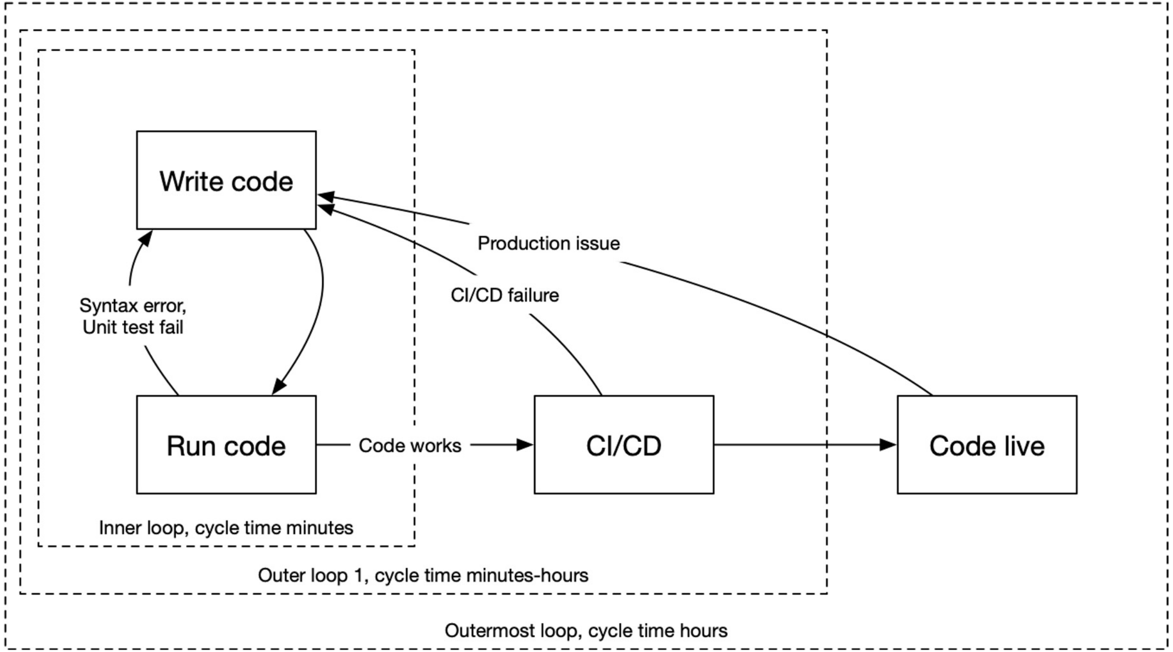
- Founder of Modal Labs
- Built the music recommendation system at Spotify
- I tweet sometimes: @bernhardsson
- I blog very occasionally: https://erikbern.com

# All just wanted to make data teams more productive!

- How to productionize jobs
- How to scale things out
- Scheduling things
- How to use GPUs and other hardware

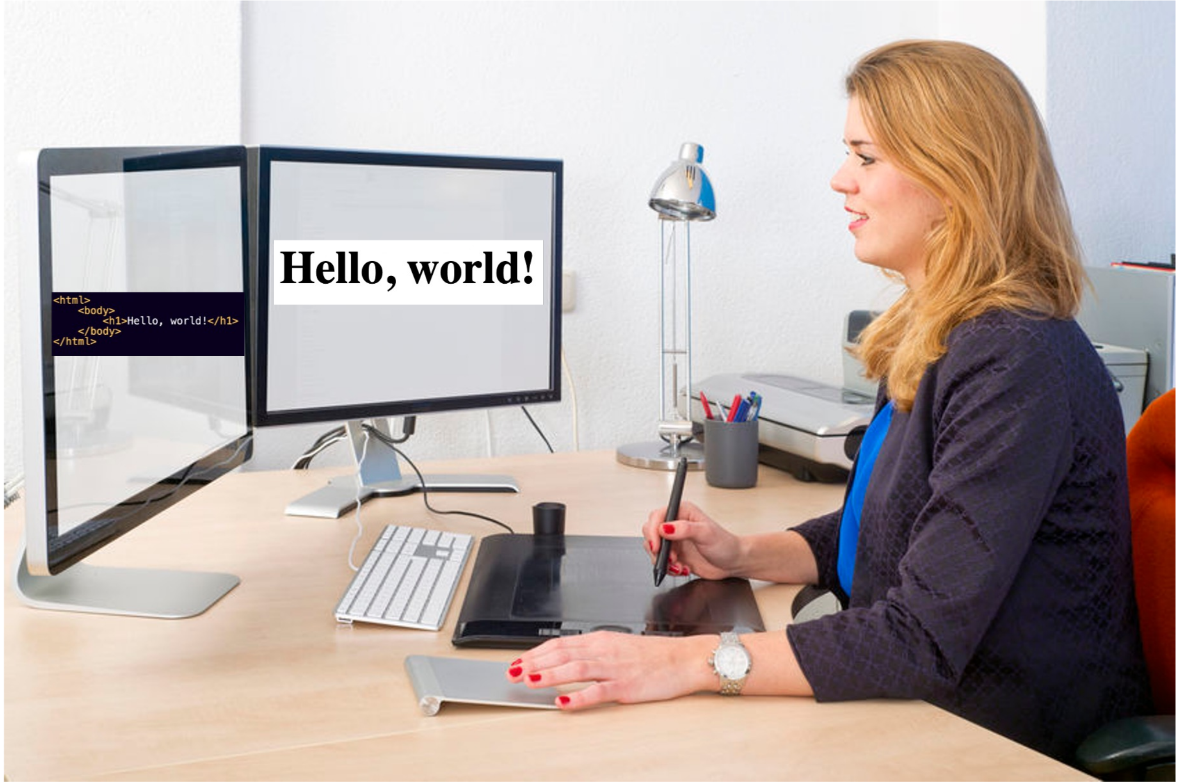
# What do I mean by eng productivity

A set of nested for-loops of writing code


Write code
Production issue
CI/CD failure
Syntax error,
Unit test fail
Run code
Code works
CI/CD
Code live
Inner loop, cycle time minutes
Outer loop 1, cycle time minutes-hours
Outermost loop, cycle time hours

# Frontend

- Write code in one window
- Look at the website in another window

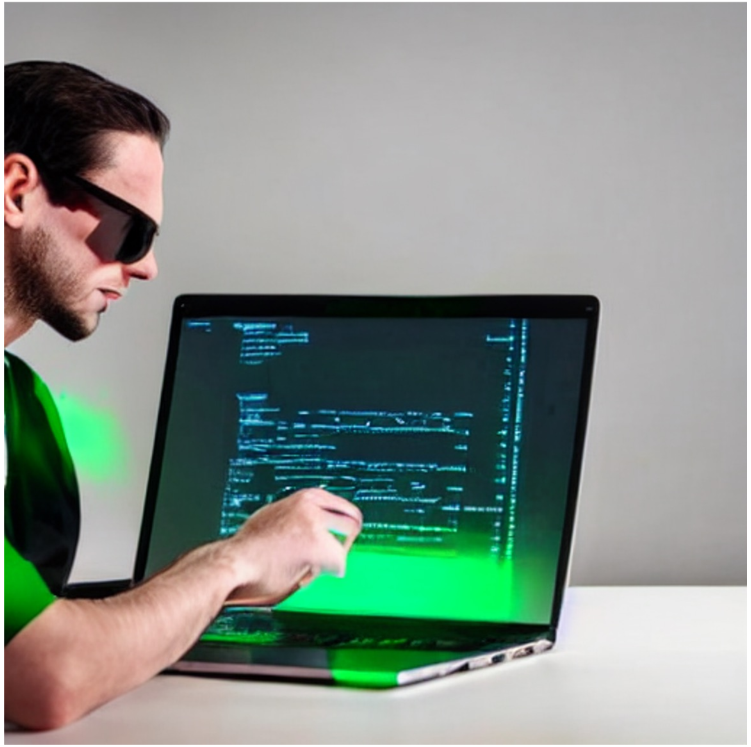


```
<html>
    <body>
        <h1>Hello, world!</h1>
    </body>
</html>
```

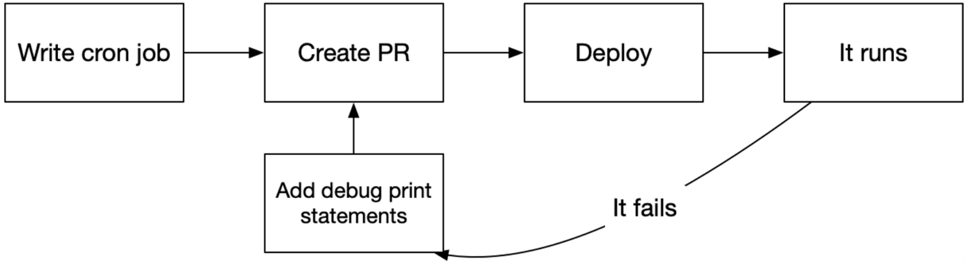
Hello, world!

# Backend

1. Write code
2. Does it compile?
3. Does it pass unit tests?
4. Ship it



# Data has super long feedback loops



Write cron job

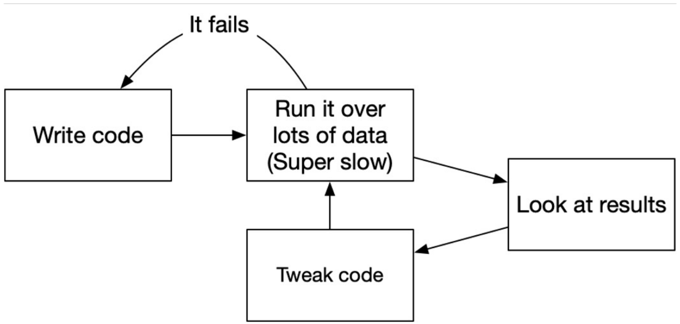
Create PR

Deploy

It runs

It fails

Add debug print statements



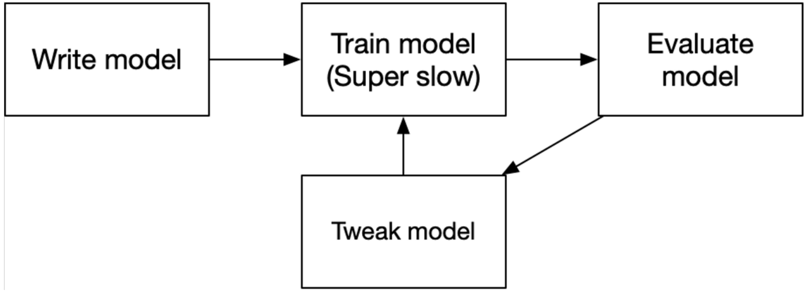
It fails

Write code

Run it over lots of data (Super slow)

Look at results

Tweak code



Write model

Train model (Super slow)

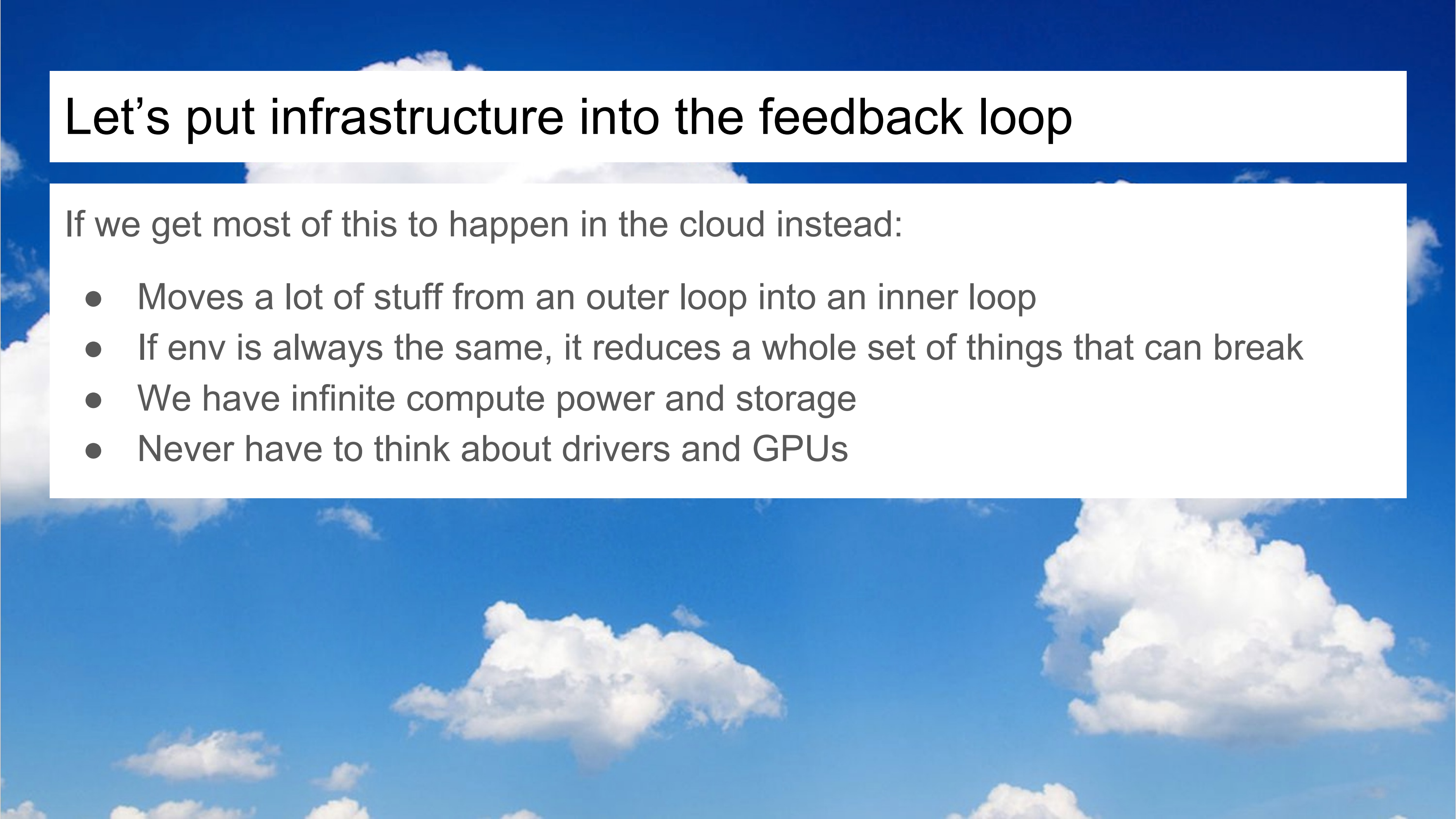
Evaluate model

Tweak model

# Let’s put infrastructure into the feedback loop

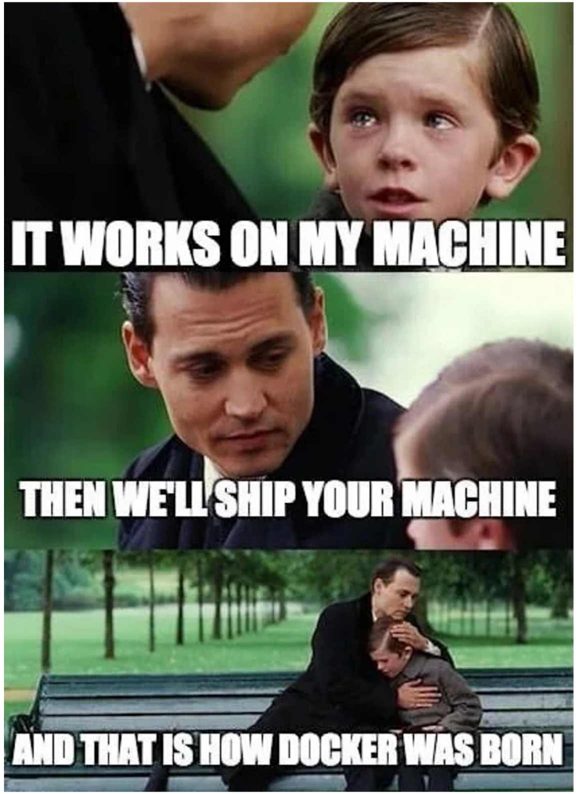
If we get most of this to happen in the cloud instead:

- Moves a lot of stuff from an outer loop into an inner loop
- If env is always the same, it reduces a whole set of things that can break
- We have infinite compute power and storage
- Never have to think about drivers and GPUs



# What are containers?

- Represent all dependencies as a Linux root filesystem
- Have a bunch of stuff for resource management (and to a limited extent, security)



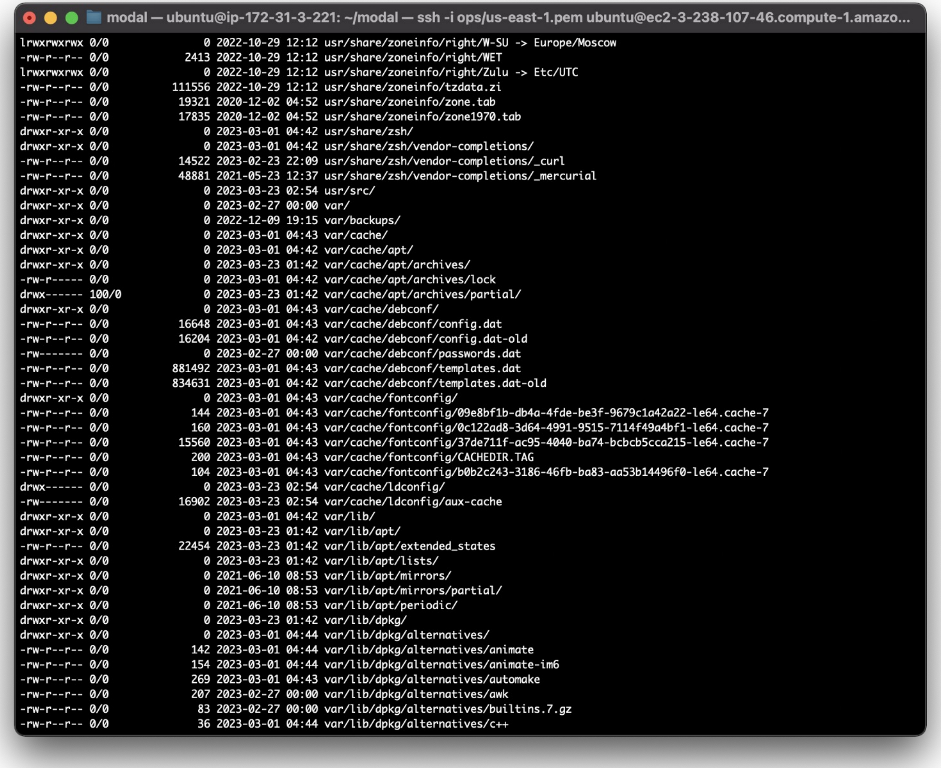
# Cracking open a Docker container

```
$ docker pull python

$ docker run -d python
sleep infinity

$ docker export
b0aa33209370 > python.tar

$ tar tvf python.tar
```

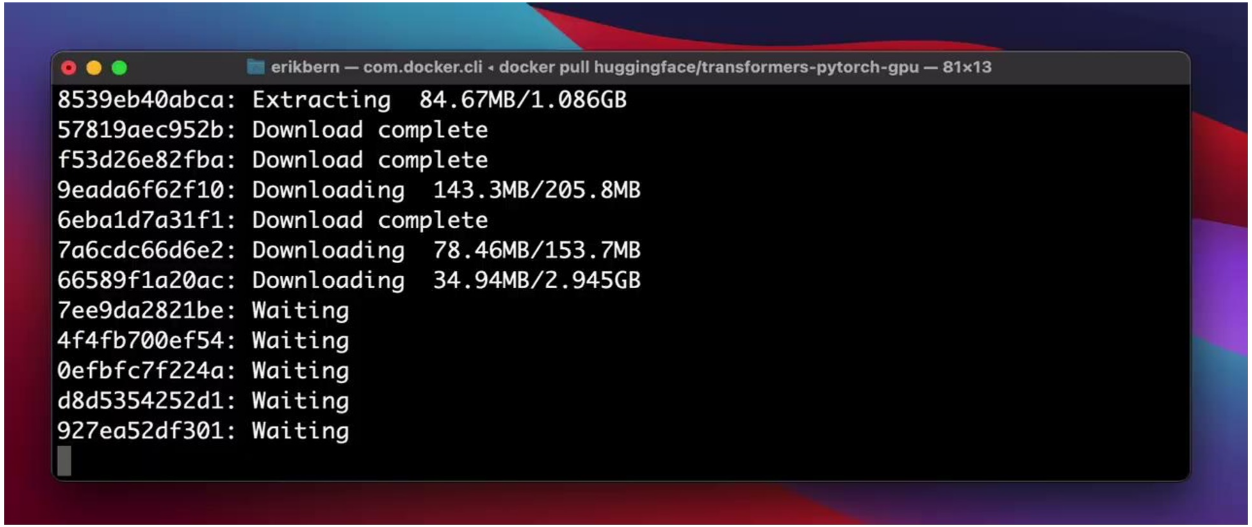


modal — ubuntu@ip-172-31-3-221: ~/modal — ssh -i ops/us-east-1.pem ubuntu@ec2-3-238-107-46.compute-1.amazo...

```
lrwxrwxrwx 0/0               0 2022-10-29 12:12 usr/share/zoneinfo/right/W-SU -> Europe/Moscow
-rw-r--r-- 0/0            2413 2022-10-29 12:12 usr/share/zoneinfo/right/WET
lrwxrwxrwx 0/0               0 2022-10-29 12:12 usr/share/zoneinfo/right/Zulu -> Etc/UTC
-rw-r--r-- 0/0          111556 2022-10-29 12:12 usr/share/zoneinfo/tzdata.zi
-rw-r--r-- 0/0           19321 2020-12-02 04:52 usr/share/zoneinfo/zone.tab
-rw-r--r-- 0/0           17835 2020-12-02 04:52 usr/share/zoneinfo/zone1970.tab
drwxr-xr-x 0/0               0 2023-03-01 04:42 usr/share/zsh/
drwxr-xr-x 0/0               0 2023-03-01 04:42 usr/share/zsh/vendor-completions/
-rw-r--r-- 0/0           14522 2023-02-23 22:09 usr/share/zsh/vendor-completions/_curl
-rw-r--r-- 0/0           48881 2021-05-23 12:37 usr/share/zsh/vendor-completions/_mercurial
drwxr-xr-x 0/0               0 2023-03-23 02:54 usr/src/
drwxr-xr-x 0/0               0 2023-02-27 00:00 var/
drwxr-xr-x 0/0               0 2022-12-09 19:15 var/backups/
drwxr-xr-x 0/0               0 2023-03-01 04:43 var/cache/
drwxr-xr-x 0/0               0 2023-03-01 04:42 var/cache/apt/
drwxr-xr-x 0/0               0 2023-03-23 01:42 var/cache/apt/archives/
-rw-r----- 0/0               0 2023-03-01 04:42 var/cache/apt/archives/lock
drwx------ 100/0             0 2023-03-23 01:42 var/cache/apt/archives/partial/
drwxr-xr-x 0/0               0 2023-03-01 04:43 var/cache/debconf/
-rw-r--r-- 0/0           16648 2023-03-01 04:43 var/cache/debconf/config.dat
-rw-r--r-- 0/0           16204 2023-03-01 04:42 var/cache/debconf/config.dat-old
-rw------- 0/0               0 2023-02-27 00:00 var/cache/debconf/passwords.dat
-rw-r--r-- 0/0          881492 2023-03-01 04:43 var/cache/debconf/templates.dat
-rw-r--r-- 0/0          834631 2023-03-01 04:42 var/cache/debconf/templates.dat-old
drwxr-xr-x 0/0               0 2023-03-01 04:43 var/cache/fontconfig/
-rw-r--r-- 0/0             144 2023-03-01 04:43 var/cache/fontconfig/09e8bf1b-db4a-4fde-be3f-9679c1a42a22-le64.cache-7
-rw-r--r-- 0/0             160 2023-03-01 04:43 var/cache/fontconfig/0c122ad8-3d64-4991-9515-7114f49a4bf1-le64.cache-7
-rw-r--r-- 0/0           15560 2023-03-01 04:43 var/cache/fontconfig/37de711f-ac95-4040-ba74-bcbcb5cca215-le64.cache-7
-rw-r--r-- 0/0             200 2023-03-01 04:43 var/cache/fontconfig/CACHEDIR.TAG
-rw-r--r-- 0/0             104 2023-03-01 04:43 var/cache/fontconfig/b0b2c243-3186-46fb-ba83-aa53b14496f0-le64.cache-7
drwx------ 0/0               0 2023-03-23 02:54 var/cache/ldconfig/
-rw------- 0/0           16902 2023-03-23 02:54 var/cache/ldconfig/aux-cache
drwxr-xr-x 0/0               0 2023-03-01 04:42 var/lib/
drwxr-xr-x 0/0               0 2023-03-23 01:42 var/lib/apt/
-rw-r--r-- 0/0           22454 2023-03-23 01:42 var/lib/apt/extended_states
drwxr-xr-x 0/0               0 2023-03-23 01:42 var/lib/apt/lists/
drwxr-xr-x 0/0               0 2021-06-10 08:53 var/lib/apt/mirrors/
drwxr-xr-x 0/0               0 2021-06-10 08:53 var/lib/apt/mirrors/partial/
drwxr-xr-x 0/0               0 2021-06-10 08:53 var/lib/apt/periodic/
drwxr-xr-x 0/0               0 2023-03-23 01:42 var/lib/dpkg/
drwxr-xr-x 0/0               0 2023-03-01 04:44 var/lib/dpkg/alternatives/
-rw-r--r-- 0/0             142 2023-03-01 04:44 var/lib/dpkg/alternatives/animate
-rw-r--r-- 0/0             154 2023-03-01 04:44 var/lib/dpkg/alternatives/animate-im6
-rw-r--r-- 0/0             269 2023-03-01 04:43 var/lib/dpkg/alternatives/automake
-rw-r--r-- 0/0             207 2023-02-27 00:00 var/lib/dpkg/alternatives/awk
-rw-r--r-- 0/0              83 2023-02-27 00:00 var/lib/dpkg/alternatives/builtins.7.gz
-rw-r--r-- 0/0              36 2023-03-01 04:44 var/lib/dpkg/alternatives/c++
```

# How to launch a container on a remote host

1. Pull down an image: a few sec to a few minutes
2. Start the image: a couple of seconds


erikbern — com.docker.cli ◂ docker pull huggingface/transformers-pytorch-gpu — 81×13

```
8539eb40abca: Extracting  84.67MB/1.086GB
57819aec952b: Download complete
f53d26e82fba: Download complete
9eada6f62f10: Downloading  143.3MB/205.8MB
6eba1d7a31f1: Download complete
7a6cdc66d6e2: Downloading  78.46MB/153.7MB
66589f1a20ac: Downloading  34.94MB/2.945GB
7ee9da2821be: Waiting
4f4fb700ef54: Waiting
0efbfc7f224a: Waiting
d8d5354252d1: Waiting
927ea52df301: Waiting
```

# The average container image has a lot of junk

Eg the **`python`** container from Dockerhub:

- 870MB large
- 29,772 files
  - /usr/share/locale: 1,553 files
  - /usr/share/doc: 3,210 files
  - /usr/share/perl: 1,389 files
  - /usr/share/man: 3,050 files

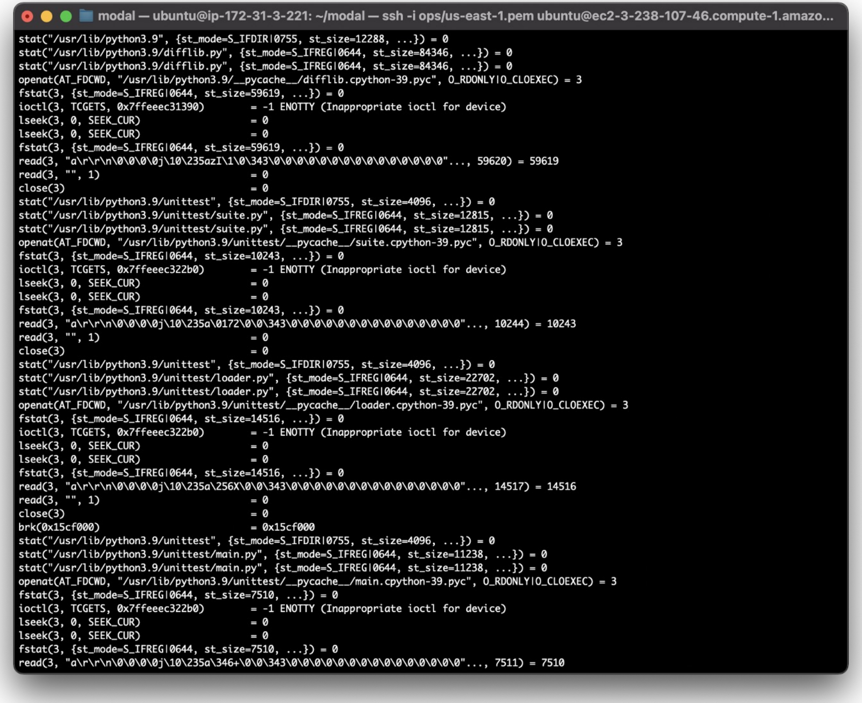
# What do we actually need to run something?

**`$ python3 -c 'import sklearn'`**

- 3,043 calls to **`stat`**
- 1,073 calls to **`openat`**

Lots of file system operations!

But only a small number of unique files are accessed.



modal — ubuntu@ip-172-31-3-221: ~/modal — ssh -i ops/us-east-1.pem ubuntu@ec2-3-238-107-46.compute-1.amazo...

```
stat("/usr/lib/python3.9", {st_mode=S_IFDIR|0755, st_size=12288, ...}) = 0
stat("/usr/lib/python3.9/difflib.py", {st_mode=S_IFREG|0644, st_size=84346, ...}) = 0
stat("/usr/lib/python3.9/difflib.py", {st_mode=S_IFREG|0644, st_size=84346, ...}) = 0
openat(AT_FDCWD, "/usr/lib/python3.9/__pycache__/difflib.cpython-39.pyc", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=59619, ...}) = 0
ioctl(3, TCGETS, 0x7ffeeec31390)        = -1 ENOTTY (Inappropriate ioctl for device)
lseek(3, 0, SEEK_CUR)                   = 0
lseek(3, 0, SEEK_CUR)                   = 0
fstat(3, {st_mode=S_IFREG|0644, st_size=59619, ...}) = 0
read(3, "a\r\r\n\0\0\0\0j\10\235azI\1\0\343\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 59620) = 59619
read(3, "", 1)                          = 0
close(3)                                = 0
stat("/usr/lib/python3.9/unittest", {st_mode=S_IFDIR|0755, st_size=4096, ...}) = 0
stat("/usr/lib/python3.9/unittest/suite.py", {st_mode=S_IFREG|0644, st_size=12815, ...}) = 0
stat("/usr/lib/python3.9/unittest/suite.py", {st_mode=S_IFREG|0644, st_size=12815, ...}) = 0
openat(AT_FDCWD, "/usr/lib/python3.9/unittest/__pycache__/suite.cpython-39.pyc", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=10243, ...}) = 0
ioctl(3, TCGETS, 0x7ffeeec322b0)        = -1 ENOTTY (Inappropriate ioctl for device)
lseek(3, 0, SEEK_CUR)                   = 0
lseek(3, 0, SEEK_CUR)                   = 0
fstat(3, {st_mode=S_IFREG|0644, st_size=10243, ...}) = 0
read(3, "a\r\r\n\0\0\0\0j\10\235a\0172\0\0\343\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 10244) = 10243
read(3, "", 1)                          = 0
close(3)                                = 0
stat("/usr/lib/python3.9/unittest", {st_mode=S_IFDIR|0755, st_size=4096, ...}) = 0
stat("/usr/lib/python3.9/unittest/loader.py", {st_mode=S_IFREG|0644, st_size=22702, ...}) = 0
stat("/usr/lib/python3.9/unittest/loader.py", {st_mode=S_IFREG|0644, st_size=22702, ...}) = 0
openat(AT_FDCWD, "/usr/lib/python3.9/unittest/__pycache__/loader.cpython-39.pyc", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=14516, ...}) = 0
ioctl(3, TCGETS, 0x7ffeeec322b0)        = -1 ENOTTY (Inappropriate ioctl for device)
lseek(3, 0, SEEK_CUR)                   = 0
lseek(3, 0, SEEK_CUR)                   = 0
fstat(3, {st_mode=S_IFREG|0644, st_size=14516, ...}) = 0
read(3, "a\r\r\n\0\0\0\0j\10\235a\256X\0\0\343\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 14517) = 14516
read(3, "", 1)                          = 0
close(3)                                = 0
brk(0x15cf000)                          = 0x15cf000
stat("/usr/lib/python3.9/unittest", {st_mode=S_IFDIR|0755, st_size=4096, ...}) = 0
stat("/usr/lib/python3.9/unittest/main.py", {st_mode=S_IFREG|0644, st_size=11238, ...}) = 0
stat("/usr/lib/python3.9/unittest/main.py", {st_mode=S_IFREG|0644, st_size=11238, ...}) = 0
openat(AT_FDCWD, "/usr/lib/python3.9/unittest/__pycache__/main.cpython-39.pyc", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=7510, ...}) = 0
ioctl(3, TCGETS, 0x7ffeeec322b0)        = -1 ENOTTY (Inappropriate ioctl for device)
lseek(3, 0, SEEK_CUR)                   = 0
lseek(3, 0, SEEK_CUR)                   = 0
fstat(3, {st_mode=S_IFREG|0644, st_size=7510, ...}) = 0
read(3, "a\r\r\n\0\0\0\0j\10\235a\346+\0\0\343\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 7511) = 7510
```

# It would be nice to avoid Docker

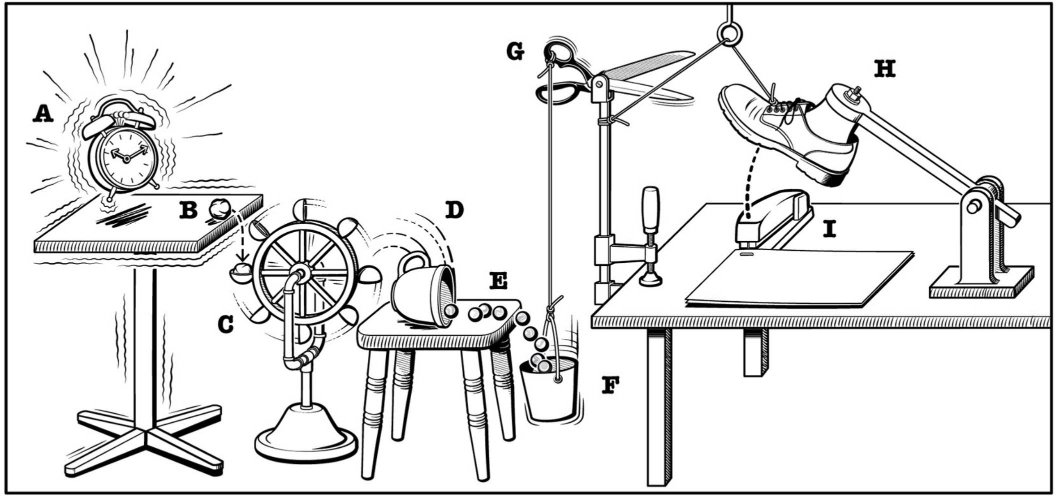
**`runc`** is a nice utility:

- Point it at a root file system
- It runs a container!
- Not absurdly complex (~50k lines of Go)

# Basic container runner that avoids docker pull:

After building the image: **`docker save`** to a network drive

When running the container: **`runc`** with a root filesystem over the network



© Vernier Software & Technology

# This is still pretty slow though!

- Python does thousands of file system operations sequentially
- NFS latency is a few milliseconds!

This adds up to like 10 seconds!

If we want to do this in seconds, we have a fraction of a millisecond for each file system operation.

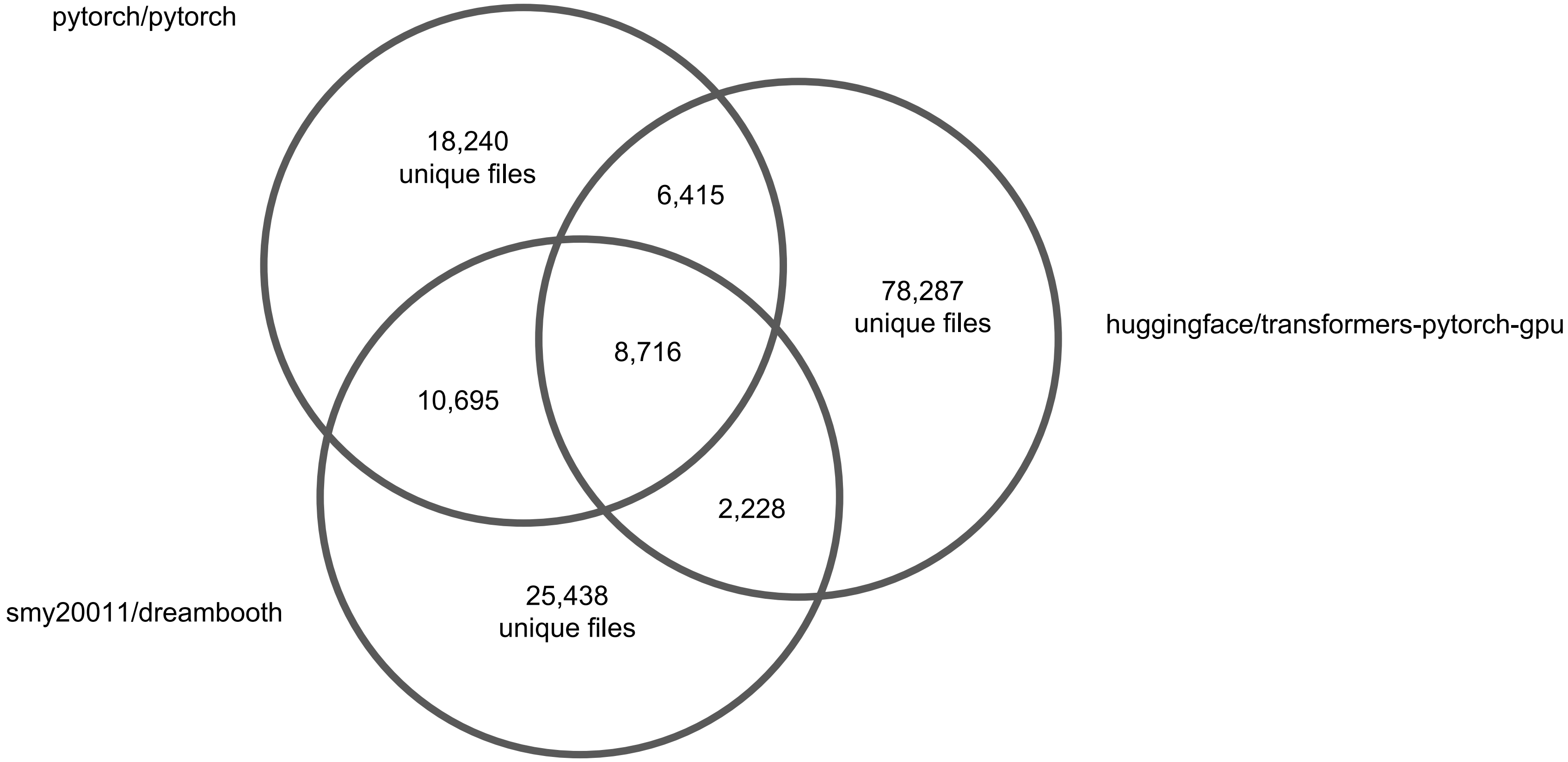
Rough latency numbers:

- S3: 10-20ms
- NFS: 1-2ms
- EBS: 0.5-1ms
- SSD: 100-200 µs

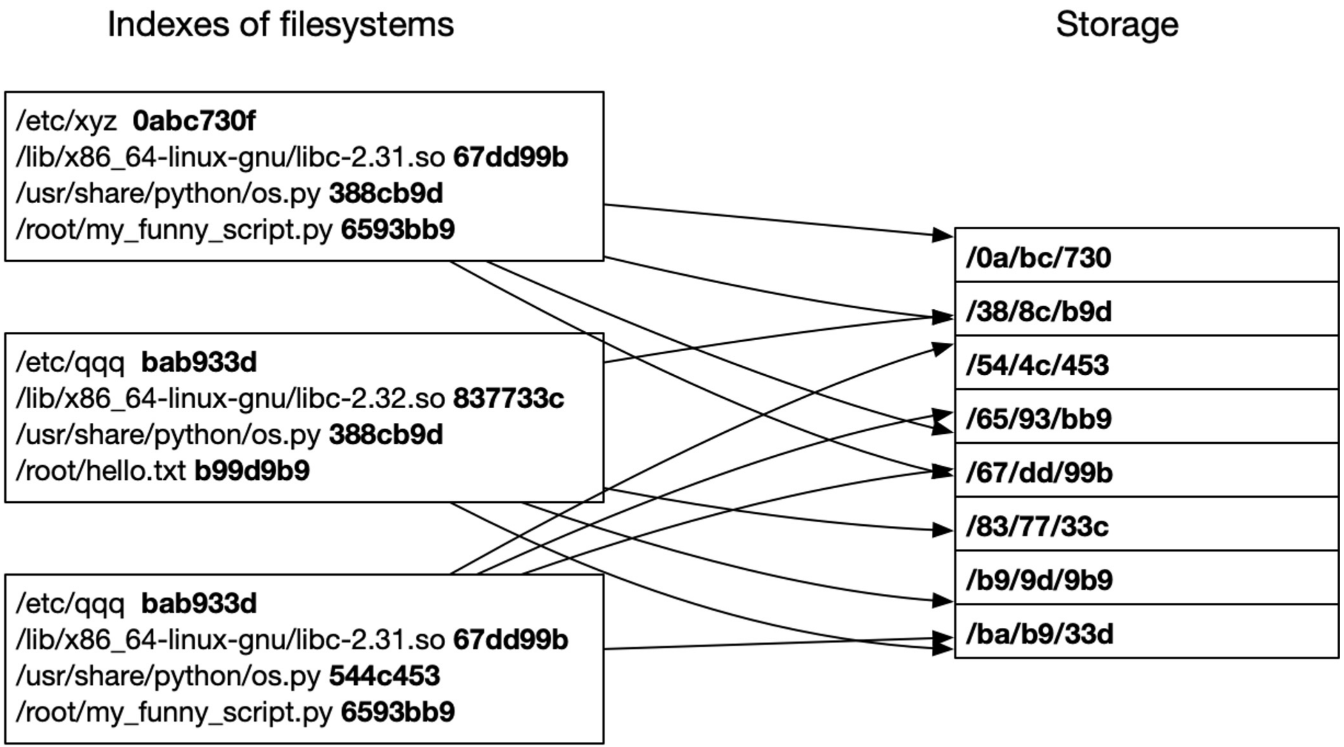
# Can we cache things locally?

- SSD latency: ~100 μs (0.1ms)
- Same image: almost the same files are read every time
- Different image: still almost the same files every time!

# Unrelated images have a lot of overlap!



# How to cache efficiently: content-addressing



Indexes of filesystems

/etc/xyz **0abc730f**
/lib/x86_64-linux-gnu/libc-2.31.so **67dd99b**
/usr/share/python/os.py **388cb9d**
/root/my_funny_script.py **6593bb9**

/etc/qqq **bab933d**
/lib/x86_64-linux-gnu/libc-2.32.so **837733c**
/usr/share/python/os.py **388cb9d**
/root/hello.txt **b99d9b9**

/etc/qqq **bab933d**
/lib/x86_64-linux-gnu/libc-2.31.so **67dd99b**
/usr/share/python/os.py **544c453**
/root/my_funny_script.py **6593bb9**

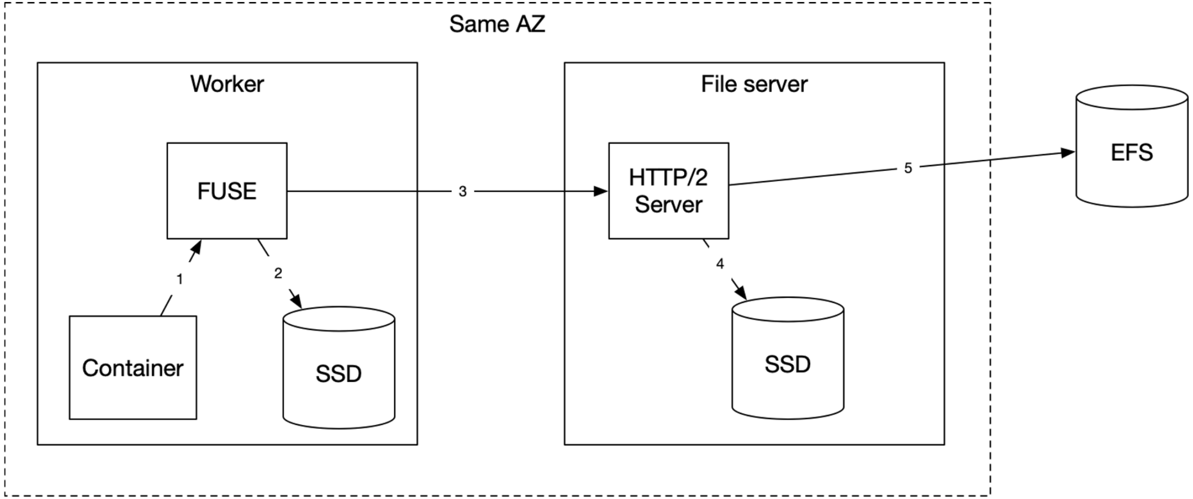
Storage

**/0a/bc/730**
**/38/8c/b9d**
**/54/4c/453**
**/65/93/bb9**
**/67/dd/99b**
**/83/77/33c**
**/b9/9d/9b9**
**/ba/b9/33d**

# How do we make this work with containers?

Build our own file system:

- Not super hard with FUSE!
- You can even do it in Python
- A lot easier if the file system is read-only



# FUSE operations we need to implement

`open`

`read`

`release`

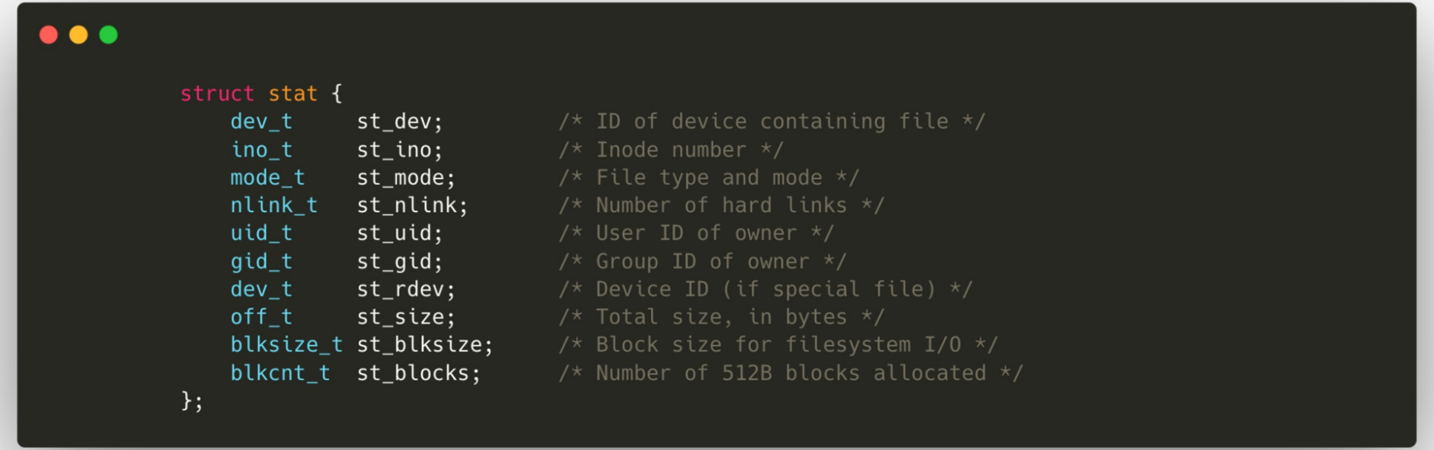
`readdir`

`readdirplus`

# Handle the indirection when reading files

Keep an index in memory that maps file system paths to

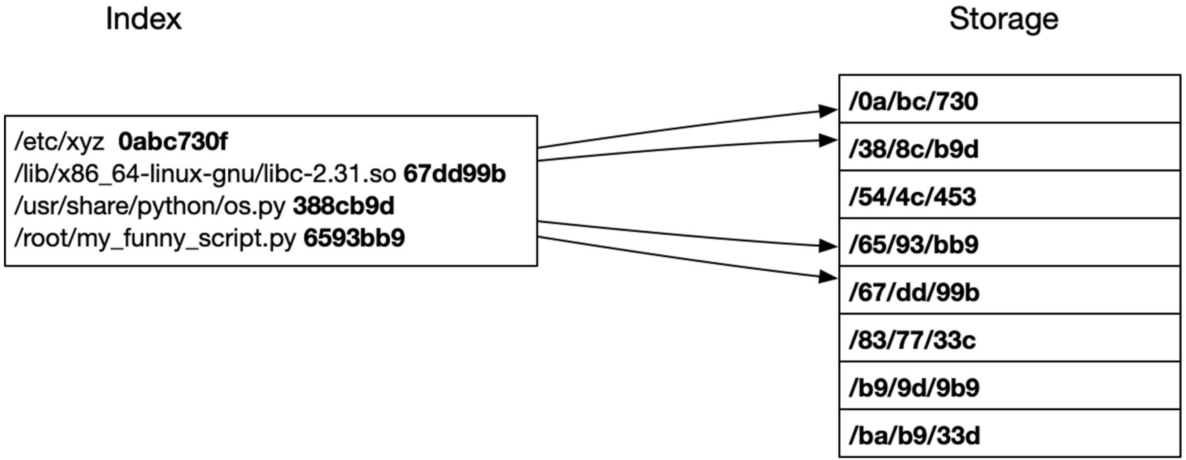
1. The hash of the content
2. A **`struct stat`**



```
struct stat {
    dev_t     st_dev;         /* ID of device containing file */
    ino_t     st_ino;         /* Inode number */
    mode_t    st_mode;        /* File type and mode */
    nlink_t   st_nlink;       /* Number of hard links */
    uid_t     st_uid;         /* User ID of owner */
    gid_t     st_gid;         /* Group ID of owner */
    dev_t     st_rdev;        /* Device ID (if special file) */
    off_t     st_size;        /* Total size, in bytes */
    blksize_t st_blksize;     /* Block size for filesystem I/O */
    blkcnt_t  st_blocks;      /* Number of 512B blocks allocated */
};
```

# When reading a file

1. Look up its hash in the index
2. See if it exists on local disk
   a. If not, fetch it, return its content, and store the file on local disk
   b. If it does exist, just return it



Index

/etc/xyz **0abc730f**
/lib/x86_64-linux-gnu/libc-2.31.so **67dd99b**
/usr/share/python/os.py **388cb9d**
/root/my_funny_script.py **6593bb9**

Storage

**/0a/bc/730**
**/38/8c/b9d**
**/54/4c/453**
**/65/93/bb9**
**/67/dd/99b**
**/83/77/33c**
**/b9/9d/9b9**
**/ba/b9/33d**

# Ok but how do we get the images into this?

We already build the containers in the cloud so that’s a good starting point!

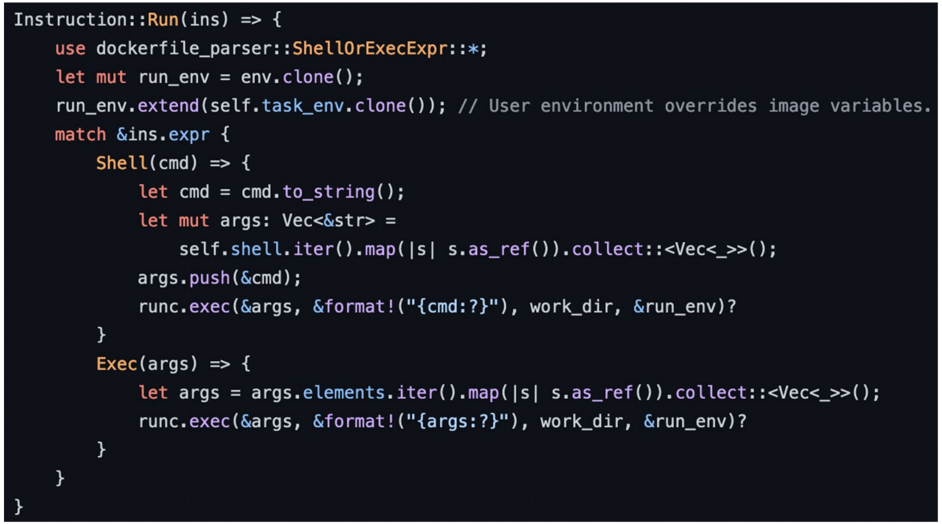
Super janky idea:

- Build images using Docker
- Then **`docker save`** to a temporary directory
- Then checksum of every file
    - Upload any file to NFS that we didn’t have already
- Then build an index of path → (checksum, struct stat)
- Store the index on NFS too

Only problem: this is super slow

# Much better idea

- Building an image is basically just running containers
- Use OverlayFS to make the image writable
- This lets us build content indexes very easily
- “Only” need to implement a Dockerfile parser



```
Instruction::Run(ins) => {
    use dockerfile_parser::ShellOrExecExpr::*;
    let mut run_env = env.clone();
    run_env.extend(self.task_env.clone()); // User environment overrides image variables.
    match &ins.expr {
        Shell(cmd) => {
            let cmd = cmd.to_string();
            let mut args: Vec<&str> =
                self.shell.iter().map(|s| s.as_ref()).collect::<Vec<_>>();
            args.push(&cmd);
            runc.exec(&args, &format!("{cmd:?}"), work_dir, &run_env)?
        }
        Exec(args) => {
            let args = args.elements.iter().map(|s| s.as_ref()).collect::<Vec<_>>();
            runc.exec(&args, &format!("{args:?}"), work_dir, &run_env)?
        }
    }
}
```

# What about scheduling?

What did we build so far:

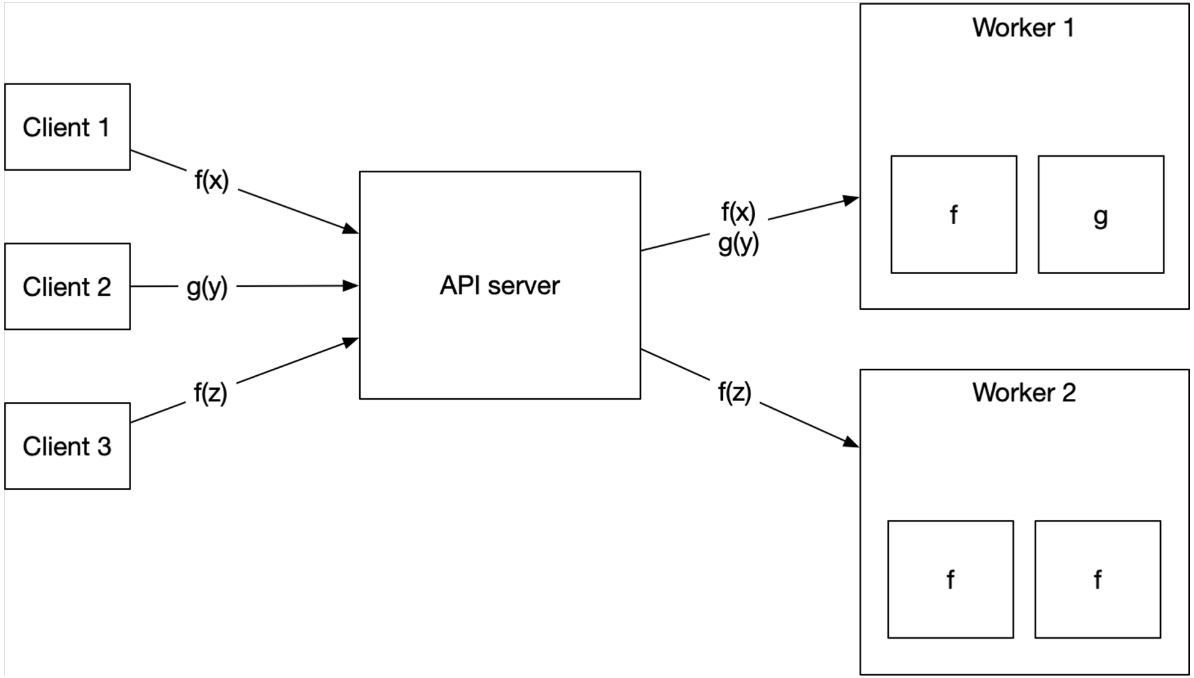
- ~~Run custom images very fast~~
- ~~Build custom images very fast~~
- Maintain a pool of worker instances
- Allocate jobs to workers

# Let’s run our own resource pool

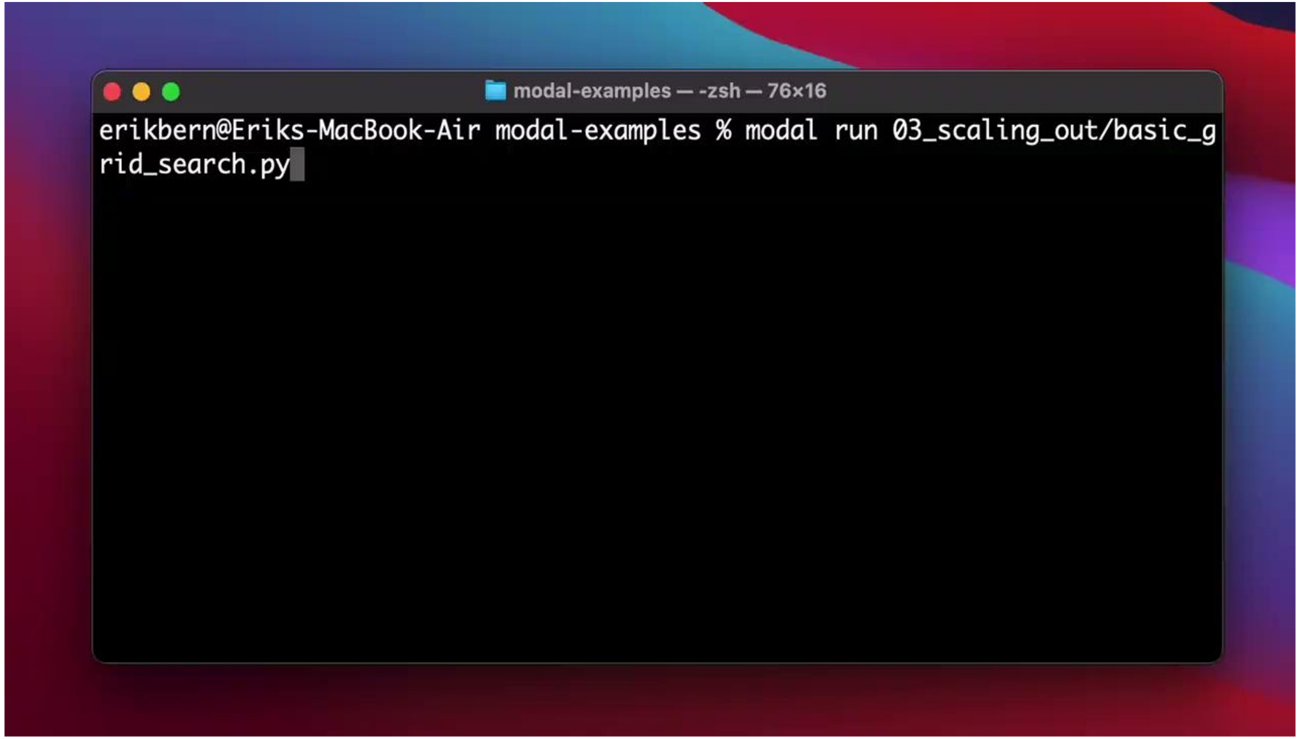
- Launch & terminate instances on AWS & GCP
- We can launch an instance in about 40s
- “Overprovision” so we always have a bit of spare capacity
- We benefit from multi-tenancy
- Every worker reports available CPU & memory every 2s

# Turning this into a function-as-a-service platform

- Main trick: reuse the same container for multiple function calls
- Autoscale on-demand, scale down to zero quickly
- Super useful for GPUs
- Need fast cold start



# What does this let us do?



```
erikbern@Eriks-MacBook-Air modal-examples % modal run 03_scaling_out/basic_g
rid_search.py
```

# What are some use cases?

- Lots of Stable Diffusion and Dreambooth
- Also computational biotech, web scraping, data pipelines, and many other things

# Was it dumb to build this in-house?

Maybe? But

- Docker is too slow & limited for what we needed
- It would have taken too much work getting Kubernetes to do this
- AWS Lambda is too expensive and limited

# Thanks!

Questions?
erik@modal.com
@bernhardsson