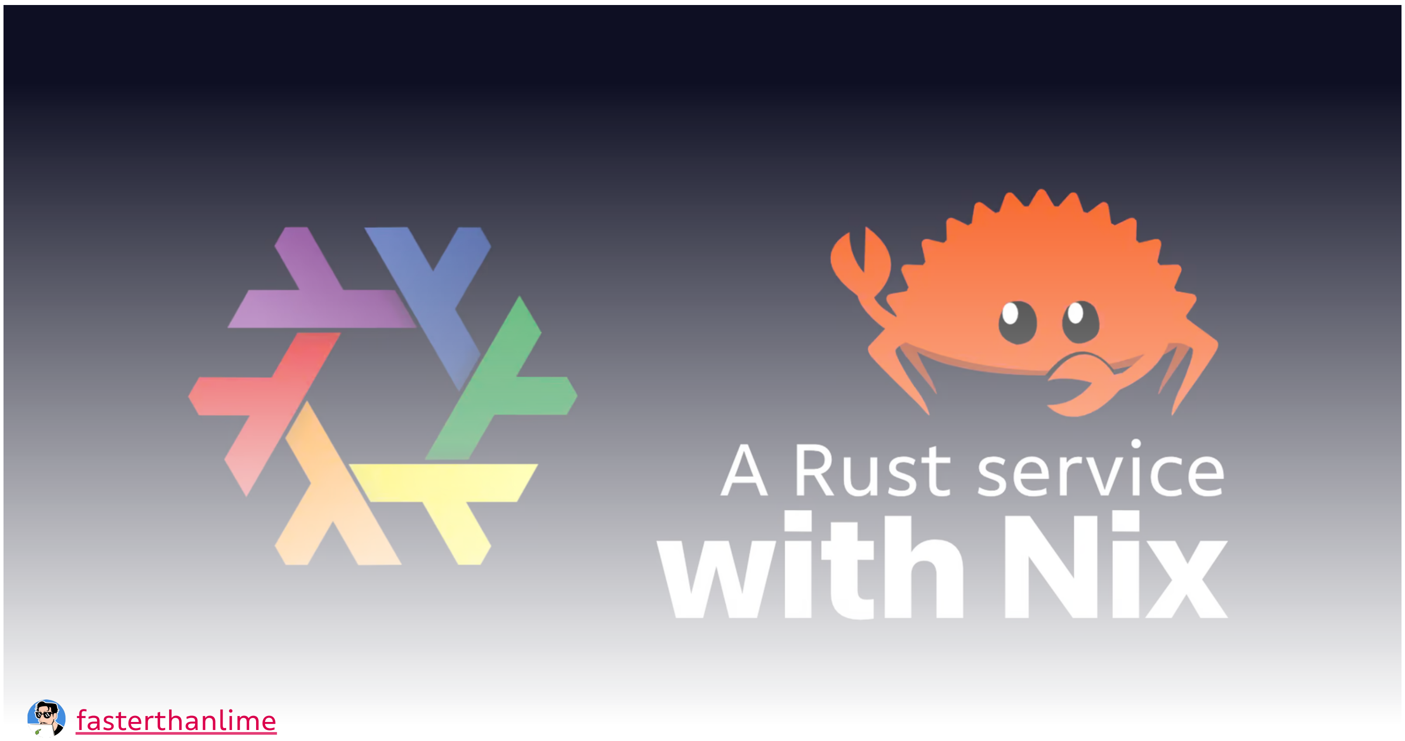


fasterthanlime

Mar 05, 2023 17 min #rust · #docker · #shipyard

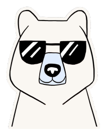
Using the Shipyard private crate registry with Docker

From the series Building a Rust service with Nix

1 2 3 4 5 6 7 8 9 10 11 12

👋 This page was last updated ~2 years ago. Just so you know.

Thanks to my sponsors: Colin VanDervoort, Max Heaton, Scott Steele, qrpth, Neil Blakey-Milner, Beat Scherrer, Marco Carmosino, Andy Gocke, Marty Penner, Tanner Muro, Luke Yue, Dom, Matt Jackson, prairiewolf, Gorazd Brumen, Lyssieth, Dylan Anthony, James Leitch, Herman J. Radtke III, Toon Willems and 266 more



Wait wait wait, so we're not talking about nix yet?

Well, no! The service we have is pretty simple, and I want to complicate things a bit, to show how things would work in both the Dockerfile and the nix scenario.

And because I don't like contrived examples, we're going to do something somewhat real-world: we're going to geo-locate visitors, and track how many visits we get from each country.

But we're not logging IP addresses, right?

No, that would be problematic re: a bunch of privacy protection laws. I feel like the country is broad enough that it's okay to store somewhere.

Very well, if we must.

Creating another crate

In that particular context, it would make sense to have a cargo workspace, with both our main (web server) crate, and our "geo-locate and count visitors by country" crate in the same Git repository.

But for demonstration purposes, we're going to make an entirely separate repository for it.

Let's call it... `locat`. For location. And cat.

```
$ cd ~
$ cargo new --lib locat
     Created library `locat` package
$ cd locat/
```

Let's stub out our API:

```
use std::net::IpAddr;

/// Allows geo-locating IPs and keeps analytics
pub struct Locat {}

impl Locat {
    pub fn new(_geoip_country_db_path: &str, _analytics_db_path: &str) -> Self {
        // TODO: read geoip db, create analytics db
        Self {}
    }

    /// Converts an address to an ISO 3166-1 alpha-2 country code
    pub fn ip_to_iso_code(&self, _addr: IpAddr) -> Option<&str> {
        None
    }

    /// Returns a map of country codes to number of requests
    pub fn get_analytics(&self) -> Vec<(String, u64)> {
        Default::default()
    }
}
```

Now's a good time to create a repository on GitHub (or wherever), do an initial commit, and push `locat` to it.

Publishing the crate privately

Normally we'd be able to add a dependency on the crate simply by doing something like:

```toml
# in `catscii/Cargo.toml`

[dependencies]
locat = { path = "../locat" }
```

Or even:

```toml
# in `catscii/Cargo.toml`

[dependencies]
locat = { git = "https://github.com/fasterthanlime/locat", rev = "some_commit_hash" }
```

But that's not what I want to show off here.

I also don't want to publish it to crates.io, since it's probably too specific and I don't want to assume the maintenance burden of it.

Instead, let's publish it to Shipyard, a private crate registry that popped up recently.

They have a free tier, and we're only going to publish a free crate there, so, let's try it!

Signing up for Shipyard should give you a crate registry of your own, and the next step is to add it to the Cargo config for `locat`:

```toml
# in `catscii/.cargo/config.toml`

[registries.catscii]
index = "https://git.shipyard.rs/catscii/crate-index.git"
```

(My registry is named `catscii`, yours will probably be named something else).

Next we amend the Cargo manifest to indicate that the crate should *only* be published to our private registry:

```toml
# in `catscii/Cargo.toml`

[package]
name = "locat"
version = "0.1.0"
edition = "2021"
# 👇 only publish there!
```

```
publish = ["catscii"]

[dependencies]
```

Now that we have all that set up, let's try publishing the crate with `cargo publish`:

```
$ cargo publish
note: Found `catscii` as only allowed registry. Publishing to it automatically.
    Updating `catscii` index
error: failed to update registry `catscii`

Caused by:
  failed to fetch `https://git.shipyard.rs/catscii/crate-index.git`

Caused by:
  failed to authenticate when downloading repository

  * attempted to find username/password via git's `credential.helper` support, but failed

  if the git CLI succeeds then `net.git-fetch-with-cli` may help here
  https://doc.rust-lang.org/cargo/reference/config.html#netgit-fetch-with-cli

Caused by:
  failed to acquire username/password from local configuration
```

That didn't work! We need to authenticate in some way.

At the time of this writing, the *proper* authentication model isn't done cooking yet, so I'm hesitant to show it here. Instead, we'll use the hack that works on Rust stable.

First we need to generate a token over at https://shipyard.rs/tokens, and then, just so it's shared across all our repositories, we can put it in the directory that contains both `catscii` and `locat`. In my VM, that's just the home directory, `/home/amos`, or `~` for short:

```
# in `~/.cargo/config.toml`

[http]
user-agent = "shipyard YOUR_TOKEN_HERE"

# this makes troubleshooting much easier: it forces using the `git` cli rather
# than `libgit2`. The latter is famously finicky with authentication
[net]
git-fetch-with-cli = true
```

This is a pretty bad workaround, since that'll get sent with every request cargo makes, let's make a note to try it the Rust nightly way after that.

Also: this is only *half* the authentication story. This takes care of making API requests to the crate registry, which is required when publishing a crate, but it doesn't help with cloning the index, which is just a git repository.

Shipyard, like most private crate registries, provides SSH-based authentication, but this can be annoying to set up in CI. It also provides HTTPS authentication, which is annoying for other reasons.

HTTPS authentication

Because we set our registry URL as `https://git.shipyard.rs/REGISTRY/crate-index.git`, it's expecting HTTPs authentication, so let's try that first.

In `~/.gitconfig`, we'll add:

```
[credential]
helper = store
```

And in `~/.git-credentials`, we'll add our Gitea username (find it here) and password, URL-encoded. Unless you changed it, this is the same password you signed up for Shipyard with, see their docs for details.

```
https://uriencodedusername:uriencodedpassword@git.shipyard.rs
```

Cool Bear's hot tip

To get URI-encoding right, you can use `encodeURIComponent("blah")` in some browser dev tools.

For me, this managed to clone the repository, but it failed with:

```
$ cargo publish
note: Found `catscii` as only allowed registry. Publishing to it automatically.
    Updating `catscii` index
error: no upload token found, please run `cargo login` or pass `--token`
```

Cool Bear's hot tip

You may also encounter:

```
error: unknown SSH host key
  The SSH host key for `ssh.shipyard.rs` is not known and cannot be validated.
```

You can run `ssh git@ssh.shipyard.rs` and accept the host key to fix it.

Running `cargo login` talks about `crates.io`, so that's certainly wrong. Running `cargo login --registry catscii` does a little better but… you can tell third-party crate registries wasn't on top of mind…

```
$ cargo login --registry catscii
```

```
please paste the API Token found on https://crates.shipyard.rs/me below

(it's now waiting for input)
```

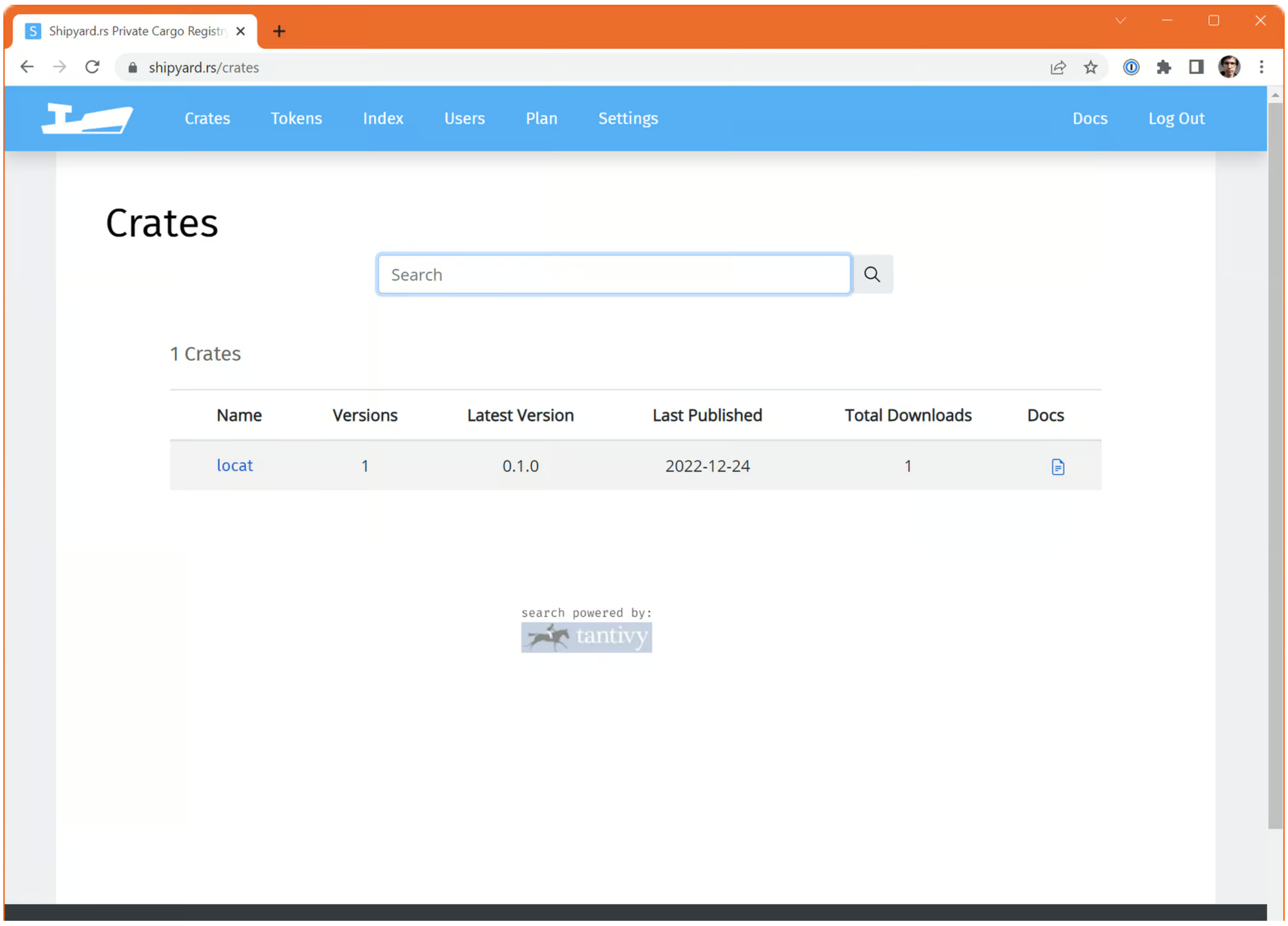
...because the URL it generates is `crates.io` -specific. Anyway, we have a token we generated earlier (it's in `~/.cargo/config.toml`), we can just paste it here.

```
(continued)
REDACTED=
       Login token for `catscii` saved
```

Finally, publishing works:

```
$ cargo publish
note: Found `catscii` as only allowed registry. Publishing to it automatically.
    Updating `catscii` index
warning: manifest has no description, license, license-file, documentation, homepage or repositor
See https://doc.rust-lang.org/cargo/reference/manifest.html#package-metadata for more info.
   Packaging locat v0.1.0 (/home/amos/locat)
   Verifying locat v0.1.0 (/home/amos/locat)
   Compiling locat v0.1.0 (/home/amos/locat/target/package/locat-0.1.0)
    Finished dev [unoptimized + debuginfo] target(s) in 0.90s
   Uploading locat v0.1.0 (/home/amos/locat)
    Updating `catscii` index
```

The crate publishes successfully, and we can see it from the Shipyard UI:



Crates

1 Crates

Name	Versions	Latest Version	Last Published	Total Downloads	Docs
locat	1	0.1.0	2022-12-24	1	🗎

However, this didn't spark joy. Let's try SSH authentication, too.

SSH authentication

To reset the state a little, first let's bump the version of our `locat` crate:

```
# in `locat/Cargo.toml`

[package]
name = "locat"
# 👇 bumped!
version = "0.2.0"
edition = "2021"
publish = ["catscii"]

[dependencies]
```

Then we'll straight up:

- remove `~/.git-credentials`
- remove the `[credential]` section from `~/.gitconfig`
- delete `~/.cargo/config.toml` (the `~` one, not the locat one)

And also, we'll need to wipe the cloned crate index, to make sure it can do it again.

```
$ ls ~/.cargo/registry/index
github.com-1ecc6299db9ec823 git.shipyard.rs-somehash
$ rm -rf ~/.cargo/registry/index/git.shipyard.rs-*
```

In locat's cargo config, we now need to use an ssh URL:

```
# in `locat/.cargo/config.toml`

[registries.catscii]
index = "ssh://git@ssh.shipyard.rs/catscii/crate-index.git"
```

Then we need to generate an SSH key from within the VM:

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/amos/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/amos/.ssh/id_rsa
Your public key has been saved in /home/amos/.ssh/id_rsa.pub
The key fingerprint is:
```

```
SHA256:AS0UXGvbfbTa2cVgJ2aOK/X5Ttek2HcpYptAO+aJN6c amos@miles
The key's randomart image is:
+---[RSA 3072]----+
|     o++.        |
|      o...       |
|       .+     B .|
|       . + . B * |
|        S.. + + +|
|        . .. O *+|
|         =.o+.B.B|
|        +o=o+ .+o|
|       ..E+o   .o|
+----[SHA256]-----+
```

Then we need to add it to Shipyard, through the web UI.

Cool Bear's hot tip

Make sure to back up that SSH key for later, when we set up everything again from scratch!

And now, let’s try everything again!

```
$ cargo publish
note: Found `catscii` as only allowed registry. Publishing to it automatically.
    Updating `catscii` index
error: failed to update registry `catscii`

Caused by:
  failed to fetch `ssh://git@ssh.shipyard.rs/catscii/crate-index.git`

Caused by:
  failed to authenticate when downloading repository

  * attempted ssh-agent authentication, but no usernames succeeded: `git`

  if the git CLI succeeds then `net.git-fetch-with-cli` may help here
  https://doc.rust-lang.org/cargo/reference/config.html#netgit-fetch-with-cli

Caused by:
  Gitea: Unauthorized
  ; class=Ssh (23); code=Eof (-20)
```

This ended up failing for me because VS Code forwarded my host SSH agent (which is 1Password), so it only tried using the SSH keys I had *on the host*, as opposed to the key I just generated on the VM.

That’s fine, I can just add one of my host’s SSH keys to Shipyard instead.

I ended up having to mess with this a little more, and add back:

```
[net]
```

```
git-fetch-with-cli = true
```

…otherwise nothing would work. Things are complicated by the fact that I have other SSH keys associated to another shipyard account, so I think I'm hitting an edge case you hopefully will not hit yourselves.

After that, publishing `catscii@0.2.0` worked:

```
$ cargo publish
note: Found `catscii` as only allowed registry. Publishing to it automatically.
    Updating `catscii` index
warning: manifest has no description, license, license-file, documentation, homepage or repositor
See https://doc.rust-lang.org/cargo/reference/manifest.html#package-metadata for more info.
   Packaging locat v0.2.0 (/home/amos/locat)
   Verifying locat v0.2.0 (/home/amos/locat)
   Compiling locat v0.2.0 (/home/amos/locat/target/package/locat-0.2.0)
    Finished dev [unoptimized + debuginfo] target(s) in 1.65s
   Uploading locat v0.2.0 (/home/amos/locat)
    Updating `catscii` index
```

But keep in mind that it's re-using the `cargo login` step from the "HTTP authentication" section: our token is still stored in here:

```
$ cat ~/.cargo/credentials
[registries.catscii]
token = "REDACTED="
```

Depending on `locat` from `catscii`

Now we can add a dependency on `locat`! First we need to set the registry URL:

```
# in `locat/.cargo/config.toml`

[registries.catscii]
index = "ssh://git@ssh.shipyard.rs/catscii/crate-index.git"

[net]
git-fetch-with-cli = true
```

And then, add the dependency! I wonder if `cargo add` will work there.

```
$ cargo add locat --registry catscii
    Updating `catscii` index
      Adding locat v0.2.0 to dependencies
```

Yes it will! Here's the resulting `Cargo.toml` bit:

```
# in `locat/Cargo.toml`
```

```
[dependencies]
locat = { version = "0.2.0", registry = "catscii" }
```

Wonderful. Let's do a quick `cargo check` just to see...

```
$ cargo check
    Updating `catscii` index
error: failed to download from `https://crates.shipyard.rs/api/v1/crates/locat/0.2.0/download`

Caused by:
  failed to get successful HTTP response from `https://crates.shipyard.rs/api/v1/crates/locat/0.2
  body:
  {"errors":[{"detail":"Unlike Crates.io, Shipyard.rs requires authentication for all API request
```

Ah. So there's actually *three* halves to cargo authentication:

- Cloning the registry index (over Git: HTTPS or SSH)
- The publish API (token)
- The download API (token)

I said we'd try the nightly thing, so let's try the nightly thing.

Switching to Rust nightly

We only need to do that in the `catscii` directory. Let's pin to a recent-ish nightly:

```
# in `catscii/rust-toolchain.toml`

[toolchain]
channel = "nightly-2022-12-24"
components = ["rustfmt", "clippy", "rust-src"]
```

And set up our `catscii` registry:

```
# in `catscii/.cargo/config.toml`

[registries.catscii]
index = "ssh://git@ssh.shipyard.rs/catscii/crate-index.git"
```

Our next `cargo check` invocation downloads that nightly version:

```
cargo check
info: syncing channel updates for 'nightly-2022-12-24-x86_64-unknown-linux-gnu'
info: latest update on 2022-12-24, rust version 1.68.0-nightly (af3e06f1b 2022-12-23)
info: downloading component 'cargo'
(cut)
info: installing component 'cargo'
(cut)
error: failed to download `locat v0.2.0 (registry `catscii`)`
```

```
Caused by:
  unable to get packages from source

Caused by:
  authenticated registries require `-Z registry-auth`
```

And complains that we haven't set the right experimental flags.

Let's fix that:

```
# in `catscii/.cargo/config.toml`

[registries.catscii]
index = "ssh://git@ssh.shipyard.rs/catscii/crate-index.git"

# 👇 new!
[unstable]
registry-auth = true
```

And then... it just works!

Using `locat` from `catscii`

Normally we'd have to make some code changes to get to the IP address, but since we're deploying to fly.io, we're behind a proxy already, and they set a header for that, so we can just use it.

We'll add a `locat` field in our `ServerState` struct:

```
// in `catscii/src/main.rs`

use std::sync::Arc;
use locat::Locat;

#[derive(Clone)]
struct ServerState {
    client: reqwest::Client,
    locat: Arc<Locat>,
}
```

Initialize it:

```
    let state = ServerState {
        client: Default::default(),
        locat: Arc::new(Locat::new("todo_geoip_path.mmdb", "todo_analytics.db")),
    };
```

And call it from `root_get`:

```
// in `catscii/src/main.rs`
```

Show all 34 lines

```
fn get_client_addr(headers: &HeaderMap) -> Option<IpAddr> {
    let header = headers.get("fly-client-ip")?;
    let header = header.to_str().ok()?;
    let addr = header.parse::<IpAddr>().ok()?;
    Some(addr)
}

async fn root_get(headers: HeaderMap, State(state): State<ServerState>) -> Response<BoxBody> {
    let tracer = global::tracer("");
    let mut span = tracer.start("root_get");
    span.set_attribute(KeyValue::new(
        "user_agent",
        headers
            .get(header::USER_AGENT)
            .map(|h| h.to_str().unwrap_or_default().to_owned())
```

Running our changes locally

Even though we're building the production version of our app using Docker, it should still run with `cargo run`, so let's try that:

```
$ cargo run
(cut)
{"timestamp":"2022-12-24T15:03:56.138667Z","level":"INFO","fields":{"message":"Listening on 0.0.0
```

Trying to visit `localhost:8080` from my host's browser prints... nothing. Because the `fly-client-ip` header isn't set.

We can try it with `curl` (in another terminal) and set an arbitrary IP there:

```
$ curl --output /dev/null --header 'fly-client-ip: 8.8.8.8' localhost:8080
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  105k  100  105k    0     0   165k      0 --:--:-- --:--:-- --:--:--  165k
```

And the server shows:

```
{"timestamp":"2022-12-24T15:05:14.740401Z","level":"WARN","fields":{"message":"Could not determin
```

Which makes perfect sense, since we haven't yet connected the GeoIP database and everything. This is neat, because it lets us experiment with deploying the new code without breaking anything.

Running our changes in production

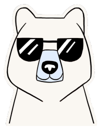
Even though our new code doesn't do anything useful yet, let's try deploying it to production.

We should probably make a `Justfile`, so we don't forget which commands to run.

You can install `just` with `cargo install just` for now, and then:

```
# in `catscii/Justfile`

# just manual: https://github.com/casey/just#readme

_default:
  just --list

deploy:
  DOCKER_BUILDKIT=1 docker build --target app --tag catscii .
  fly deploy --local-only
```

Now tell me cool bear: do you think this'll go smoothly?



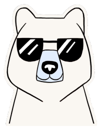
You're asking, so... probably not?

You're absolutely correct. It's time for.. more Docker golfing!

Missing `COPY`

```
$ just deploy
(cut)
 => ERROR [builder 7/7] RUN --mount=type=cache,target=/root/.rustup     --mount=type=cache,target
------
 > [builder 7/7] RUN --mount=type=cache,target=/root/.rustup     --mount=type=cache,target=/root/
#18 0.159 + cargo build --release
#18 0.241 error: failed to parse manifest at `/app/Cargo.toml`
#18 0.241
#18 0.241 Caused by:
#18 0.241   no index found for registry: `catscii`
------
executor failed running [/bin/sh -c set -eux;           cargo build --release;          objcopy -
error: Recipe `deploy` failed on line 7 with exit code 1
```

Contest number one! It says it can't find a registry named `catscii`. And yet, the `.cargo/config.toml` is *right there on disk*.



Yes, but... *looks at heading* it's not in the build context? We need to copy it?

Correct!

```

# Copy sources and build them
WORKDIR /app
COPY src src
# 👇 new!
COPY .cargo .cargo
COPY Cargo.toml Cargo.lock ./
```

Git's not here, man

You know what else is a compile-time dependency now?

```
$ just deploy
(cut)
 > [builder 8/8] RUN --mount=type=cache,target=/root/.rustup     --mount=type=cache,target=/root/
#19 0.163 + cargo build --release
#19 0.233     Updating `catscii` index
#19 0.234 error: failed to get `opentelemetry-honeycomb` as a dependency of package `catscii v0.1
#19 0.234
#19 0.234 Caused by:
#19 0.234   failed to load source for dependency `opentelemetry-honeycomb`
#19 0.234
#19 0.234 Caused by:
#19 0.234   Unable to update registry `catscii`
#19 0.234
#19 0.234 Caused by:
#19 0.234   failed to fetch `ssh://git@ssh.shipyard.rs/catscii/crate-index.git`
#19 0.234
#19 0.234 Caused by:
#19 0.234   could not execute process `git fetch --force --update-head-ok 'ssh://git@ssh.shipyard
#19 0.234
#19 0.234 Caused by:
#19 0.234   No such file or directory (os error 2)
------
executor failed running [/bin/sh -c set -eux;           cargo build --release;          objcopy -
error: Recipe `deploy` failed on line 7 with exit code 1
```

That's right! It's ~~the square hole~~ Git!

```
# Install compile-time dependencies
RUN set -eux; \
    apt update; \
    apt install --yes --no-install-recommends \
      # 👇 new!
      git-core curl ca-certificates gcc libc6-dev pkg-config libssl-dev \
      ;
```

Does that mean it needs to re-download and re-install all packages from scratch on the next `docker build` run? You betcha!

It's been two minutes already. I guess that was an unlucky run. How are you doing today? Is it a lovely day for you? Not necessarily the weather, but, you know, your internal weather? Have you been tending to your inner garden? It's important, you know. Sometimes self-care might feel like you're being selfish, but really you're just getting yourself in a position where you can actually help others again, whereas if y- oh would you look at that, it finished.

What's an ssh?

```
#19 0.485 Caused by:
#19 0.485   process didn't exit successfully: `git fetch --force --update-head-ok 'ssh://git@ssh.
```

```
#19 0.485   --- stderr
#19 0.485   error: cannot run ssh: No such file or directory
#19 0.485   fatal: unable to fork
```

Oh for crying out loud.

```
# Install compile-time dependencies
RUN set -eux; \
    apt update; \
    apt install -y --no-install-recommends \
      # 👇 new!
      openssh-client git-core curl ca-certificates gcc libc6-dev pkg-config libssl-dev \
      ;
```

Well I guess we have time for more chit chat. What songs are you listening to right now? I've been trying to find Winter-themed music that isn't offensive to my sensibilities. That mostly means Frank Sinatra, or some other jazz pianists. Teddy Wilson's "As Times Goes By" is surprisingly g- oh, break's over!

Host key verification failed

```
#19 0.965 Caused by:
#19 0.965   process didn't exit successfully: `git fetch --force --update-head-ok 'ssh://git@ssh.
#19 0.965   --- stderr
#19 0.965   Host key verification failed.
#19 0.965   fatal: Could not read from remote repository.
#19 0.965
#19 0.965   Please make sure you have the correct access rights
#19 0.965   and the repository exists
```

Oh. Uh. A little web search pointed to this, let's try it:

```
# Copy sources and build them
WORKDIR /app
COPY src src
COPY .cargo .cargo
COPY Cargo.toml Cargo.lock ./
# 👇
RUN ssh-keyscan ssh.shipyard.rs >> ~/.ssh/known_hosts
RUN --mount=type=cache,target=/root/.rustup \
# (etc.)
```

No, wait, nevermind:

```
 > [builder 8/9] RUN ssh-keyscan ssh.shipyard.rs >> ~/.ssh/known_hosts:
#19 0.159 /bin/sh: 1: cannot create /root/.ssh/known_hosts: Directory nonexistent
```

Okay, sure:

```
RUN mkdir --parents ~/.ssh/ && ssh-keyscan ssh.shipyard.rs >> ~/.ssh/known_hosts
```

I'm fairly sure that defeats the purpose, but oh well.

Permission denied (publickey)

```
#20 1.208 Caused by:
#20 1.208   process didn't exit successfully: `git fetch --force --update-head-ok 'ssh://git@ssh.
#20 1.208   --- stderr
#20 1.208   Warning: Permanently added the ECDSA host key for IP address '15.204.143.93' to the l
#20 1.208   git@ssh.shipyard.rs: Permission denied (publickey).
#20 1.208   fatal: Could not read from remote repository.
#20 1.208
#20 1.208   Please make sure you have the correct access rights
#20 1.208   and the repository exists
```

Ok. I'll save you the trouble. BuildKit also lets us "mount our SSH agent" (to Docker, every problem is mount-shaped), so we can do this:

```
# Copy sources and build them
WORKDIR /app
COPY src src
COPY .cargo .cargo
COPY Cargo.toml Cargo.lock ./
RUN --mount=type=cache,target=/root/.rustup \
    --mount=type=cache,target=/root/.cargo/registry \
    --mount=type=cache,target=/root/.cargo/git \
    --mount=type=cache,target=/app/target \
    # 👇 new!
    --mount=type=ssh \
    set -eux; \
    cargo build --release; \
    objcopy --compress-debug-sections ./target/release/catscii ./catscii
```

Permission denied (publickey), still

The error message is exactly the same, except it includes `--mount=type=ssh`.

This is because, you need to tell `docker build` to share your SSH credentials. Otherwise any `Dockerfile` could just steal them. I still think the security model is kinda wonky there, but, we've written our own `Dockerfile` from start to finish and I trust it somewhat.

Let's adjust our `docker build` invocation in the `Justfile`:

```
deploy:
  #                               👇 new!
  DOCKER_BUILDKIT=1 docker build --ssh default --target app --tag catscii .
  fly deploy --local-only
```

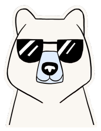
Because, again, I have multiple Shipyard accounts, and my SSH agent offers keys that are associated with the wrong account (an edge case you shouldn't be hitting), I set up another agent quickly:

```
# THIS IS FOR MY WEIRD EDGE CASE ONLY
# (You probably don't need it)
$ sudo apt install keychain
$ keychain

 * keychain 2.8.5 ~ http://www.funtoo.org
 * Starting ssh-agent...

$ source ~/.keychain/miles-sh
$ ssh-add ~/.ssh/id_rsa
$ ssh-add -l
3072 SHA256:AS0UXGvbfbTa2cVgJ2aOK/X5Ttek2HcpYptAO+aJN6c amos@miles (RSA)
```

Failed to get successful HTTP response, got 401

Here's our next error:

```
 > [builder 9/9] RUN --mount=type=cache,target=/root/.rustup     --mount=type=cache,target=/root/
#20 0.136 + cargo build --release
#20 0.184     Updating `catscii` index
#20 1.943  Downloading crates ...
#20 2.566 error: failed to download from `https://crates.shipyard.rs/api/v1/crates/locat/0.2.0/do
#20 2.566
#20 2.566 Caused by:
#20 2.566   failed to get successful HTTP response from `https://crates.shipyard.rs/api/v1/crates
#20 2.566   body:
#20 2.566   {"errors":[{"detail":"Unlike Crates.io, Shipyard.rs requires authentication for all A
```



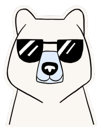
Wait wait wait wait. I know this one. Can I try something?

Sure?

```
RUN --mount=type=cache,target=/root/.rustup \
    --mount=type=cache,target=/root/.cargo/registry \
    --mount=type=cache,target=/root/.cargo/git \
    --mount=type=cache,target=/app/target \
    --mount=type=ssh \
    set -eux; \
    # 👇
    rustc --version; \
    exit 251; \
    # (etc.)
```

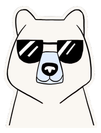
```
$ just deploy
(cut)
#20 0.120 + rustc --version
#20 0.156 rustc 1.66.0 (69f9c33d7 2022-12-12)
```

```
#20 0.156 + exit 251
```



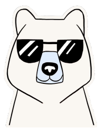
AhAh! It's not using the right version of Rust inside Docker, because...

...because `rust-toolchain.toml` is not part of the build context.



Whoa, these *are* easy to miss, you weren't kidding!

Yeah! But you know what? Some developers actually love technologies that make it easy to make mistakes, because then there's always something to do! Every single one of these commits counts towards *some* metric, and it makes them look very productive: "saving the day" is usually remembering one of a hundred footguns and applying the same fix, over and over.



Huh. Funny how this works out.

Yes yep. Anyway:

```
# now also copying the toolchain file 👇
COPY Cargo.toml Cargo.lock rust-toolchain.toml ./
```

(Also don't forget to remove `exit 251` from the `Dockerfile`)

No token found, again

```
#20 0.281  Downloading crates ...
#20 0.300 error: failed to download `locat v0.2.0 (registry `catscii`)`
#20 0.300
#20 0.300 Caused by:
#20 0.300   unable to get packages from source
#20 0.300
#20 0.300 Caused by:
#20 0.300   no token found for `catscii`, please run `cargo login --registry catscii`
#20 0.300   or use environment variable CARGO_REGISTRIES_CATSCII_TOKEN
```

Ah. So that one only worked locally because we had done `cargo login`. I see.

Well... this is a docker. Every problem is a mount away. Even secrets!

Because we're going to commit that file, let's make sure it's encrypted with `git-crypt` first.

```
# in `.gitattributes`
```

```
.envrc filter=git-crypt diff=git-crypt
# 👇 new!
secrets/* filter=git-crypt diff=git-crypt
```

Then we can store *just the value* of our token in `secrets/shipyard-token`:

```
REDACTED
```

(Make sure that file has no trailing newline!)

Let's make sure `git-crypt` picks it up:

```
$ git-crypt status -e
    encrypted: secrets/shipyard-token
    encrypted: .envrc
```

Now we can mount it as a secret:

```
RUN --mount=type=cache,target=/root/.rustup \
    --mount=type=cache,target=/root/.cargo/registry \
    --mount=type=cache,target=/root/.cargo/git \
    --mount=type=cache,target=/app/target \
    --mount=type=ssh \
    # 👇 new!
    --mount=type=secret,id=shipyard-token \
    set -eux; \
    # 👇 also new!
    export CARGO_REGISTRIES_CATSCII_TOKEN=$(cat /run/secrets/shipyard-token); \
    rustc --version; \
    cargo build --release; \
    objcopy --compress-debug-sections ./target/release/catscii ./catscii
```

And pass it to `docker build`:

```
deploy:
  DOCKER_BUILDKIT=1 docker build \
    --ssh default \
    # 👇 there we go
    --secret id=shipyard-token,src=secrets/shipyard-token \
    --target app \
    --tag catscii \
    .
  fly deploy --local-only
```

And finally, `just deploy` works!

Visiting the site, I can see this in my logs:

```
$ fly logs
(cut)
```

```
2022-12-24T17:39:27Z app[fb691ea1] cdg [info]{"timestamp":"2022-12-24T17:39:27.906817Z","level":"
```

Nice! Now we “just” need to make it do the thing!

You're reading the Building a Rust service with Nix series.

1 2 3 4 5 6 7 8 9 10 11 12

Comment on /r/fasterthanlime

Here's another article just for you:

Frustrated? It's not you, it's Rust

Aug 14, 2020 38 min #rust · #traits · #sized · #lifetimes

Learning Rust is... an experience. An emotional journey. I’ve rarely been more frustrated than in my first few months of trying to learn Rust.
What makes it worse is that it doesn’t matter how much prior experience you have, in Java, C#, C or C++ or otherwise - it’ll still be unnerving.

In fact, *more* experience probably makes it worse! The habits have settled in deeper, and there’s a certain expectation that, by now, you should be able to get that done in a shorter amount of time.

Read more