👋 hi!

- fp enthusiast and low-level hacker for half my life
- the smaller half of the Lustre core team

Installing Erlang

Gleam compiles to Erlang code, so Erlang needs to be installed to run Gleam code. Some of the above package managers (e.g. Homebrew) will install Erlang alongside Gleam automatically.

Precompiled builds for many popular operating systems can be downloaded from the Erlang solutions website.

Once Erlang has been installed you can check it is working by typing `erl -version` in your computer's terminal. You will see version information like this if all is well:

```
erl -version
Erlang (SMP,ASYNC_THREADS) (BEAM) emulator version 12.1.5
```

Linux

Using Homebrew

Homebrew will install Erlang alongside Gleam automatically, though it can be manually installed by running the following:

```
brew update
brew install erlang
```

Alpine Linux (Community repository)

```
apk add erlang
```

Arch Linux (Community repository)

```
pacman -S erlang
```

Gentoo Linux (already installed after installing gleam through emerge)

```
emerge --ask dev-lang/erlang
```

Fedora

```
dnf install elixir erlang
```

Debian, Ubuntu

Add Erlang Solutions repo key:

```
wget -qO- https://binaries2.erlang-solutions.com/GPG-KEY-pmanager.asc | sudo tee
/etc/apt/keyrings/erlang.asc
```

We are about to add APT source line in form of:

```
deb [signed-by=/etc/apt/keyrings/erlang.asc] http://binaries2.erlang-solutions.com/ubuntu/
$DIST contrib
```

We need to determine `$DIST` first, based on our distro codename (found via `lsb_release -c`). Go to binaries2.erlang-solutions.com, then click to "debian" or "ubuntu", then "dist". We will see directories like "stretch-esl-erlang-27", "noble-esl-erlang-27". Pick one corresponding to our Debian/Ubuntu version and replace `$DIST` with it.

```
# Add as APT source
echo 'deb [signed-by=/etc/apt/keyrings/erlang.asc] http://binaries2.erlang-
solutions.com/ubuntu/ $DIST contrib' | sudo tee /etc/apt/sources.list.d/erlang.list

# Update apt and install esl-erlang
sudo apt update
sudo apt install esl-erlang
```

```
curl -L “$GITHUB_URL" | tar xz
```

Could this all be really simple?

Statically compiling Erlang

```
./configure

    –disable-pie \
    --enable-builtin-zlib \
    --disable-dynamic-ssl-lib \
    --with-ssl \
    —-enable-static-nifs \
    --enable-static-drivers \
```

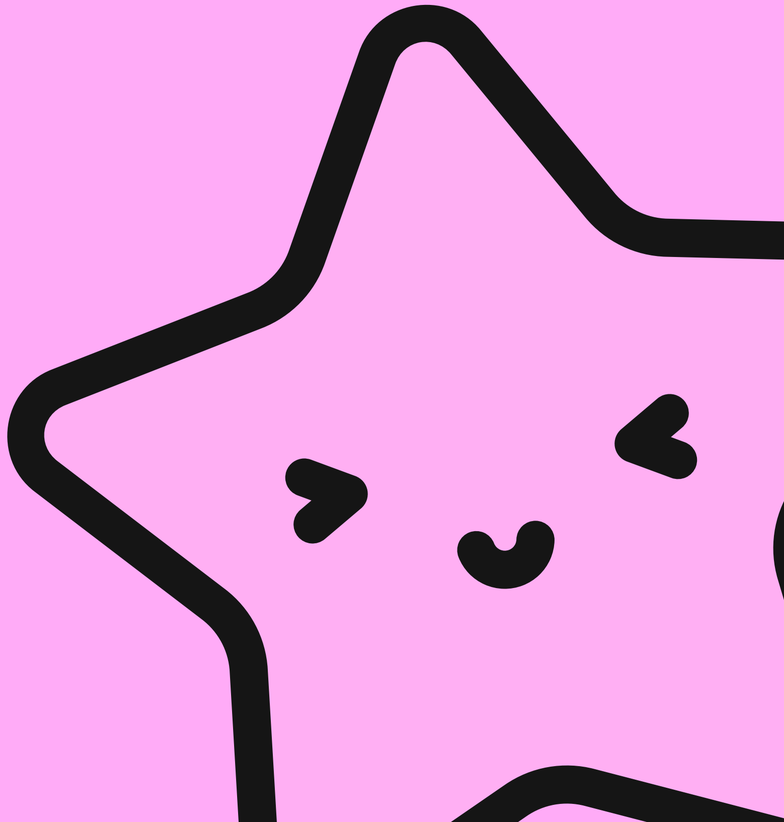
Statically compiling Erlang

```
./configure
    LIBS="-lncursesw -ltinfo -lcrypto -lssl -lstdc++" \
    LDFLAGS="-static -static-libgcc -static-libstdc++" \
    –disable-pie \
    --enable-builtin-zlib \
    --disable-dynamic-ssl-lib \
    --with-ssl \
    –-enable-static-nifs \
    --enable-static-drivers \
```

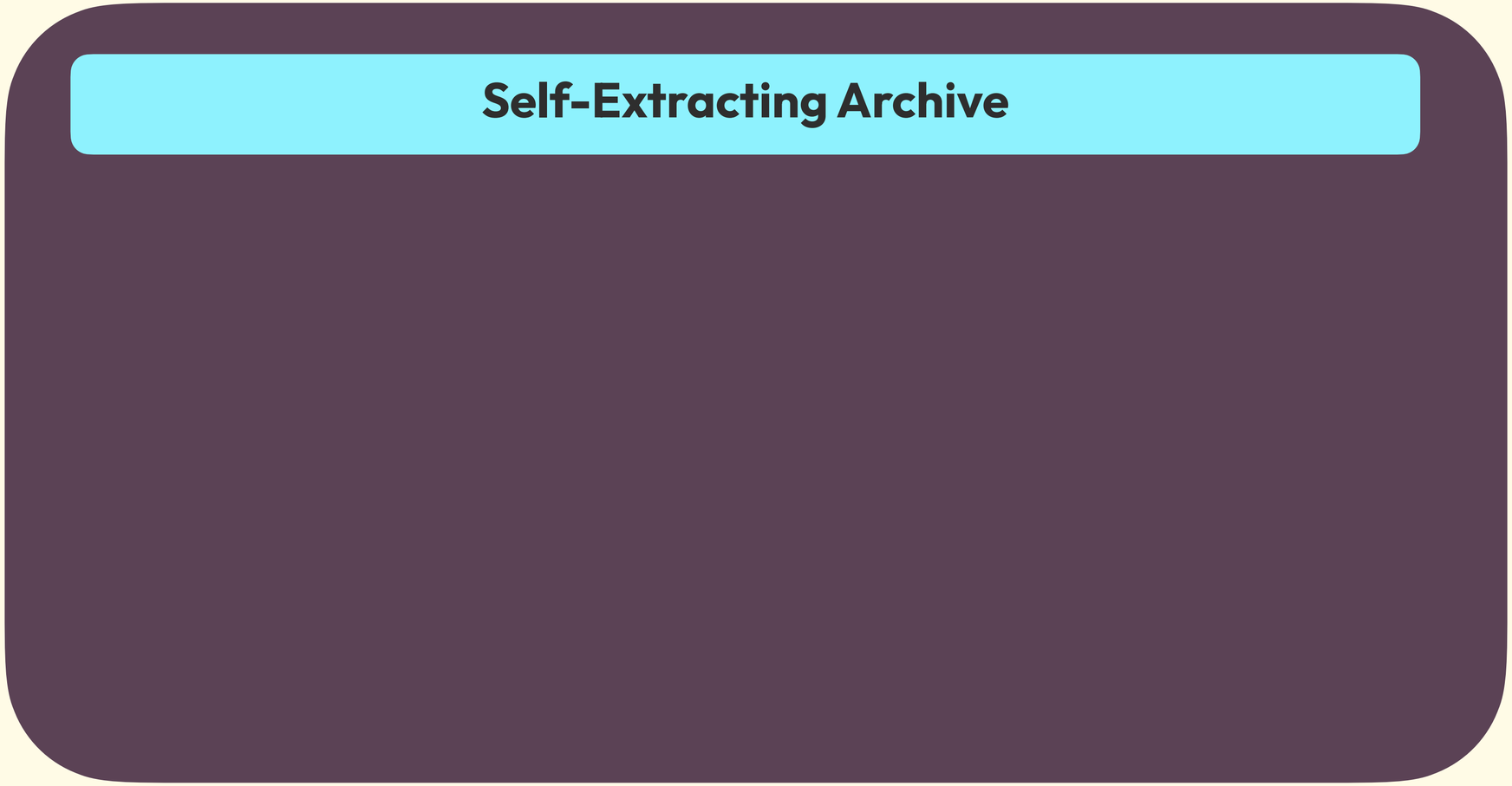
Statically compiling Erlang

```
./configure CC=clang CXX=clang \
    LIBS="-lncursesw -ltinfo -lcrypto -lssl -lstdc++" \
    LDFLAGS="-static -static-libgcc -static-libstdc++" \
    –disable-pie \
    --enable-builtin-zlib \
    --disable-dynamic-ssl-lib \
    --with-ssl \
    —-enable-static-nifs \
    --enable-static-drivers \
```

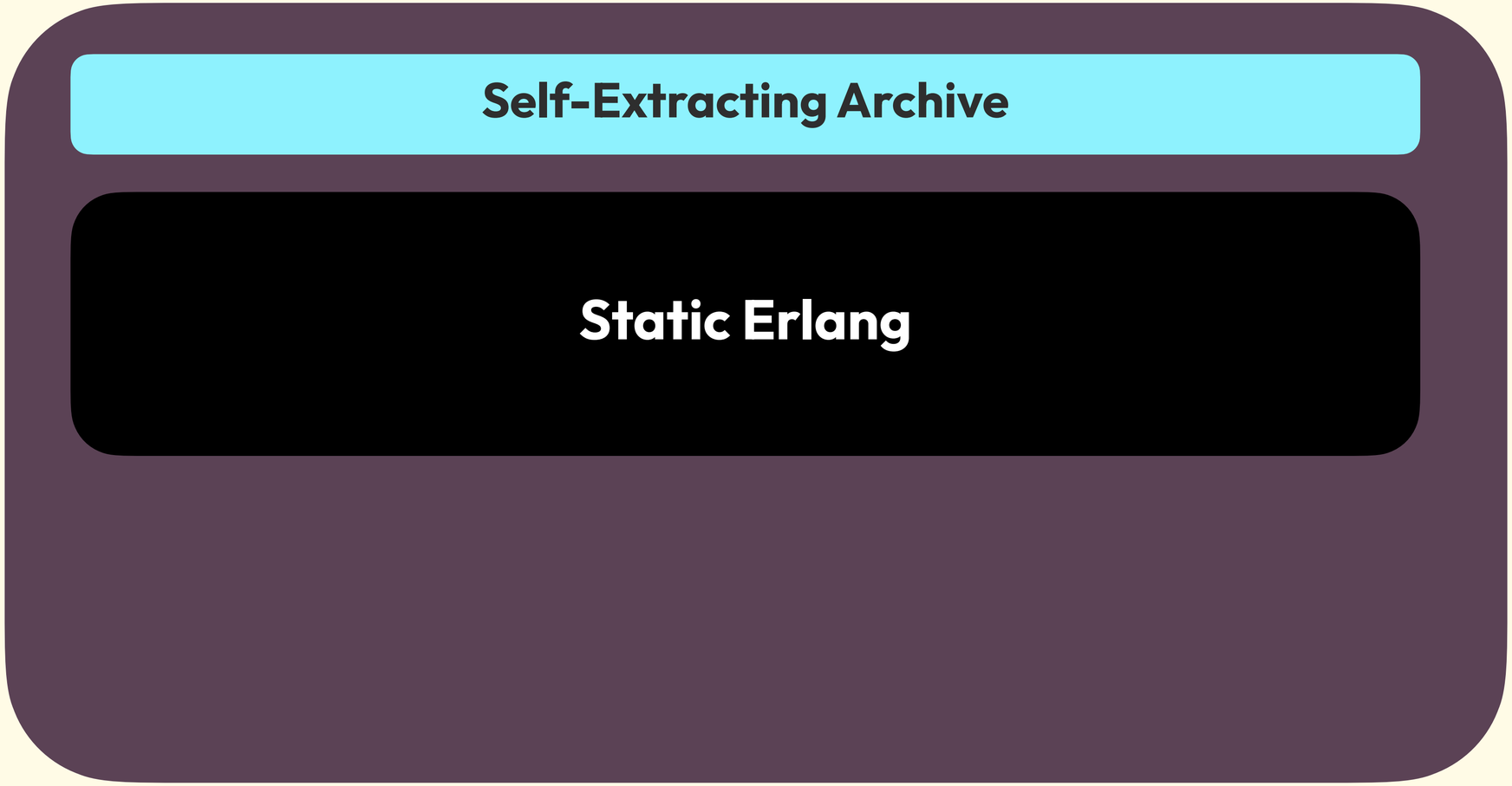
Statically compiled Erlang!



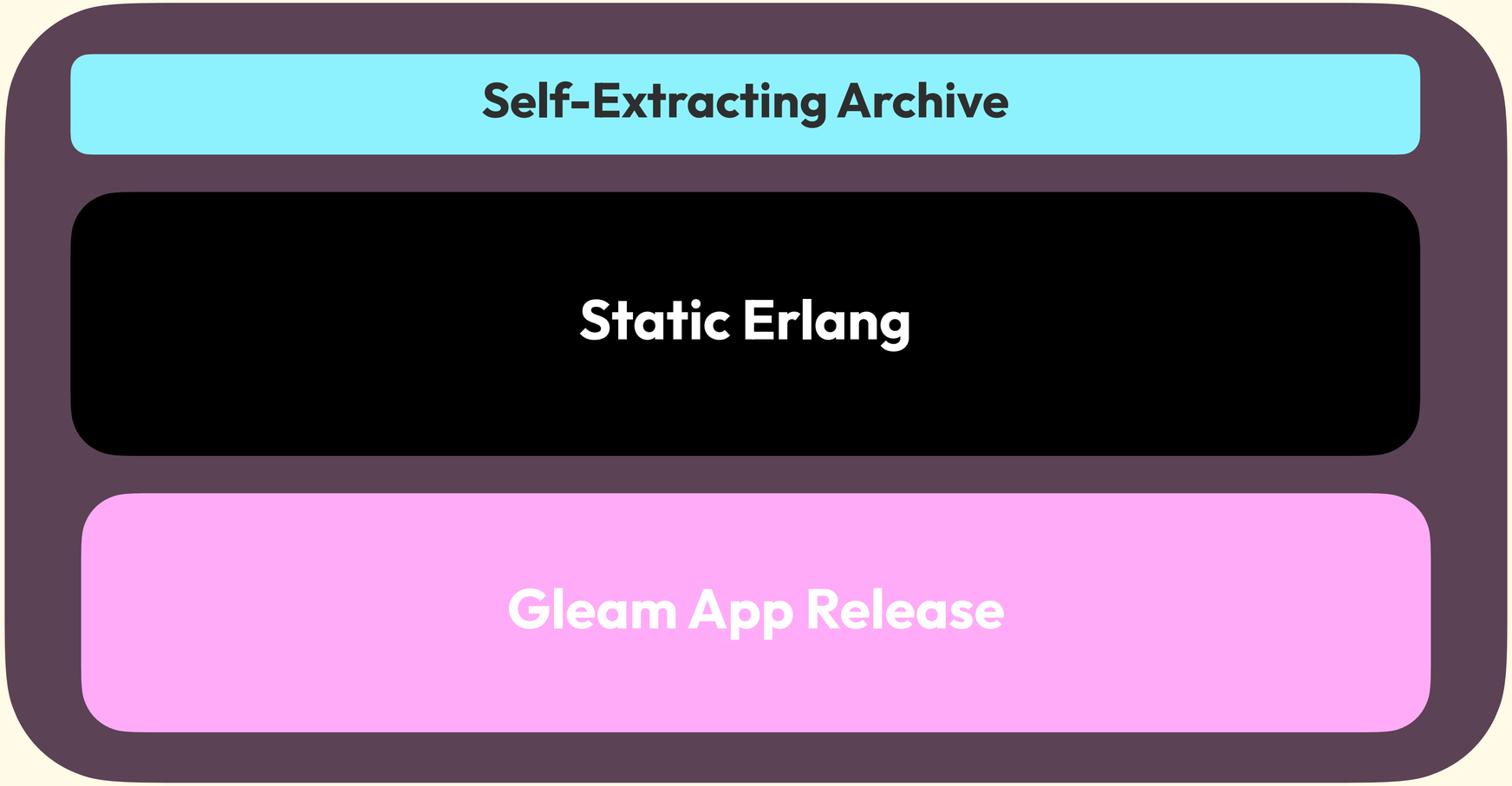
gleepack?



gleepack?



gleepack?



Burrito?

How Escripts get run

- Self-extracting archive

How Escripts get run

- Self-extracting archive
- escript

How Escripts get run

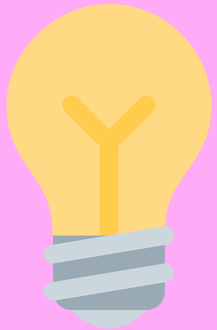
- Self-extracting archive
- escript
- erl

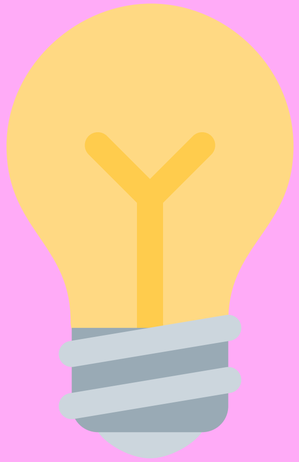
How Escripts get run

- Self-extracting archive
- escript
- erl
- erlexec

How Escripts get run

- Self-extracting archive
- escript
- erl
- erlexec
- beam.smp





Gleepack!

- Statically link with the BEAM

Gleepack!

- Statically link with the BEAM
- Extract an OTP Release into memory

Gleepack!

- Statically link with the BEAM
- Extract an OTP Release into memory
- Do some linker magic to inject ourselves into the function loading files

Gleepack!

- Statically link with the BEAM
- Extract an OTP Release into memory
- Do some linker magic to inject ourselves into the function loading files
- …

Gleepack!

- Statically link with the BEAM
- Extract an OTP Release into memory
- Do some linker magic to inject ourselves into the function loading files
- …

Gleepack!

- Statically link with the BEAM
- Extract an OTP Release into memory
- Do some linker magic to inject ourselves into the function loading files
- …