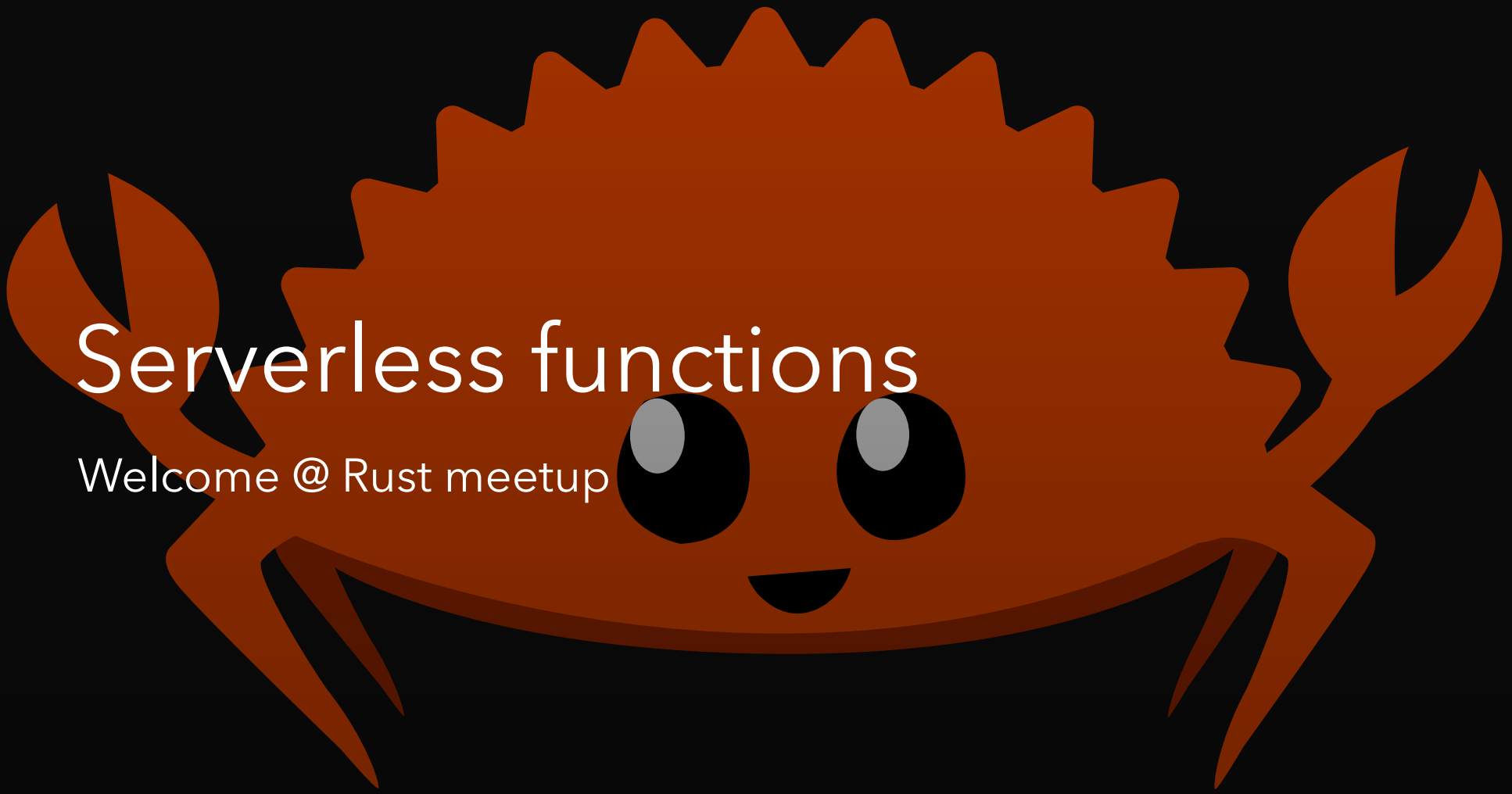


Serverless functions

Welcome @ Rust meetup

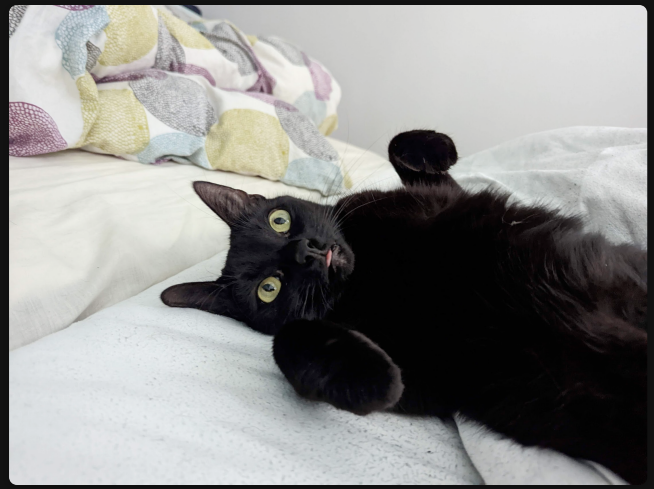
About me

Name: Patrick Lamprecht

Nickname: Lampe

Hobbies:

- Via Ferrata fan
- Loves cats



- Likes low level languages like Rust
- Pizza lover / Hobby pizza maker

- Working at

- Age at *George*: ~4 years

What is serverless?

- Not `No servers`
- > Serverless computing is a method of providing backend services on an as-used basis. Servers are still used, but a company that gets backend services from a serverless vendor is charged based on usage, not a fixed amount of bandwidth or number of servers. - Cloudflare
- Summary: Pay what you use and don’t care about servers

Free stuff

- Services that are freely available
- Good for hobby projects to experiment
- (some) can also be used in production later


Do we get to mess with him now?
No. We wait until the worst possible moment.
CI/CD
Demo

Live demos

Knative

https://knative.dev/docs/install/yaml-install/serving/install-serving-with-yaml/

```
docker build -t dev.local/knative-serving .
```

```
k apply -f service.yaml
```

```
k get ksvc
```

AWS Lambda

```
brew tap cargo-lambda/cargo-lambda
brew install cargo-lambda
```

```
cargo lambda new aws-lambda
# Is this function an HTTP function? Yes
# Which service is this function receiving events from? AWS Lambda function URLs
```

```
cargo add reqwest --no-default-features --features rustls-tls
```

```
use lambda_http::{run, service_fn, Body, Error, Request, RequestExt, Response};

async fn function_handler(event: Request) -> Result<Response<Body>, Error> {
    let body = reqwest::get("https://www.rust-lang.org")
        .await
        .map_err(|_| "failed to get")?
        .text()
        .await
        .map_err(|_| "failed to convert to text")?;
    let resp = Response::builder()
        .status(200)
        .header("content-type", "text/html")
        .body(body.into())
        .map_err(Box::new)?;
    Ok(resp)
}
```

Fermyon Spin/Cloud

https://cloud.fermyon.com/

```
spin new http-rust
```

```
use anyhow::Result;
use spin_sdk::outbound_http::send_request;
use spin_sdk::{
    http::{Request, Response},
    http_component,
};

#[http_component]
fn handle_fermyon_spin(req: Request) -> Result<Response> {
    let request = http::Request::builder()
        .uri("https://www.rust-lang.org")
        .body(None)?;

    let response = send_request(request)?;
    Ok(http::Response::builder()
        .status(200)
        .body(response.into_body())?)
}
```

`spin.toml`

Cloudflare Worker

https://github.com/cloudflare/workers-sdk

```
brew install cloudflare-wrangler2
```

Check that you have version 2 of `wrangler`

```
wrangler generate cloudflare-worker cloudflare/workers-sdk/templates/experimental/worker-rust
```

Templates can be found here: https://github.com/cloudflare/workers-sdk/tree/main/templates

```
cargo add reqwest
```

```rust
#[event(fetch)]
async fn main(req: Request, env: Env, ctx: Context) -> Result<Response> {
    let body = reqwest::get("https://www.rust-lang.org")
        .await
        .map_err(|_| "failed to get")?
        .text()
        .await
        .map_err(|_| "failed to convert to text")?;
    Response::ok(body)
}
```

Just the tip of the ice berg

- Vercel
- Netlify
- Google Cloud Functions
- and many more…
- For more inspiration see http://freestuff.dev

Summary

Knative

Benefits

- Runs on any Kubernetes
- Only one small kubernetes yaml
- Just needs a docker image (can run anything)
- Shuts down pods when not used

Drawbacks

- Needs warmup (>1 second)
- Long startup with function calling another
- Most complex setup compared to others
- Dockerfile

AWS Lambda

Benefits

- ARM instances -> cost reduction
- Fast startup (about half a second, with latency)
- Easy deployment
- Function URLS (no api gateway needed anymore)
- No Dockerfile
- Whole AWS ecosystem (monitoring, logging,…)

Drawbacks

- IAM problems (can’t use admin token for cargo-lambda, still valid?)

Fermyon spin/cloud

Benefits

- Interesting technology (WASI)
- Fast startup
- Easy setup and deployment
- No Dockerfile

Drawbacks

- WASI is relatively young
- Many crates miss WASI support
- WASI has no standard for sockets till now (wasi-sockets)
- One thread (wasi-threads)

Cloudflare Worker

Benefits

- Uses WASM
- WASM support on crates.io is good (eg reqwest)
- Offer key/value store and pub/sub
- Very easy setup with prompt to login in browser
- URL per function

Drawbacks

- Sending mails is problematic (openssl has no wasm/wasi support)

Serverless != no server

You just don’t care about the servers anymore ;)

Thank you very much for listening to me
when I talk about my hobbies!

END