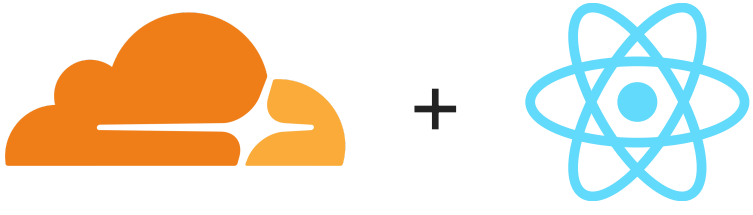
DEPLOY AND TEST FULL-STACK REACT APPS ON CLOUDFLARE



INSTRUCTORS

- Christian Sparks
- Dario Piotrowicz

CHRISTIAN SPARKS

- Builds and Automation team @ Cloudflare
- github.com/cmsparks
- linkedin.com/in/cmsparks

DARIO PIOTROWICZ

- Frameworks team @ Cloudflare
- github.com/dario-piotrowicz

WORKSHOP

Is about creating a simple full stack React application and deploy it to the Cloudflare platform

Alongside comprehensive testing 🧪

AGENDA

- Introduction ~10m
- UI Implementation ~30m
- Project Deployment ~15m

- Break ~5m

- Backend Logic ~1h

- Break ~5m

- UI Integration ~20m
- E2E Test ~20m
- End of workshop ~15m

REQUIREMENTS

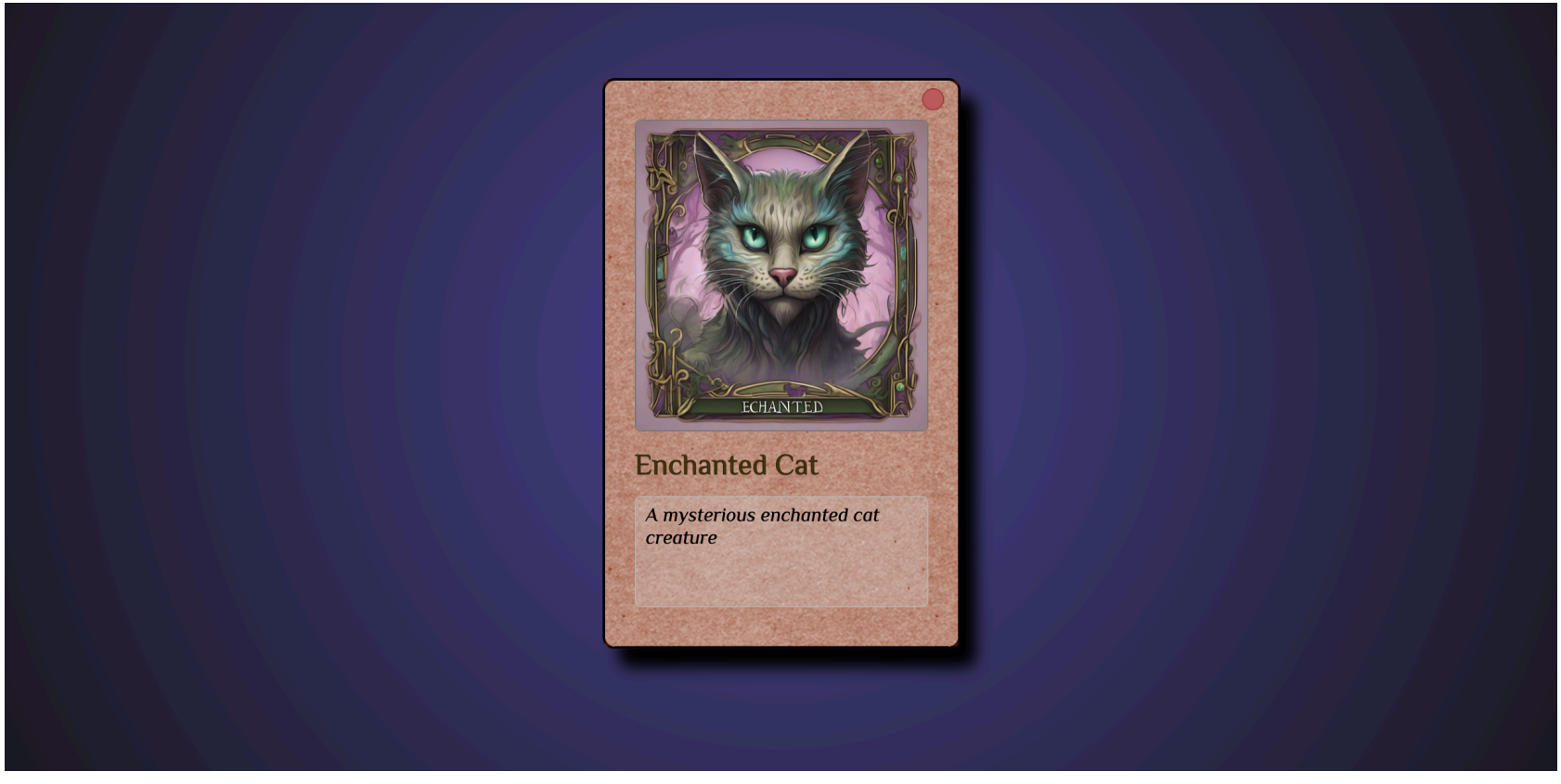
- Your favorite IDE
- `npm` (or your preferred package manager)
- `git` so that you can clone and interact with the workshop's git repository
- A Cloudflare account
- Enthusiasm! 🚀

PROJECT OVERVIEW

In this workshop we'll be creating a trading card generator app built using:

- the Remix React full-stack framework
- Cloudflare Workers
- Cloudflare Resources

COMPLETED APPLICATION

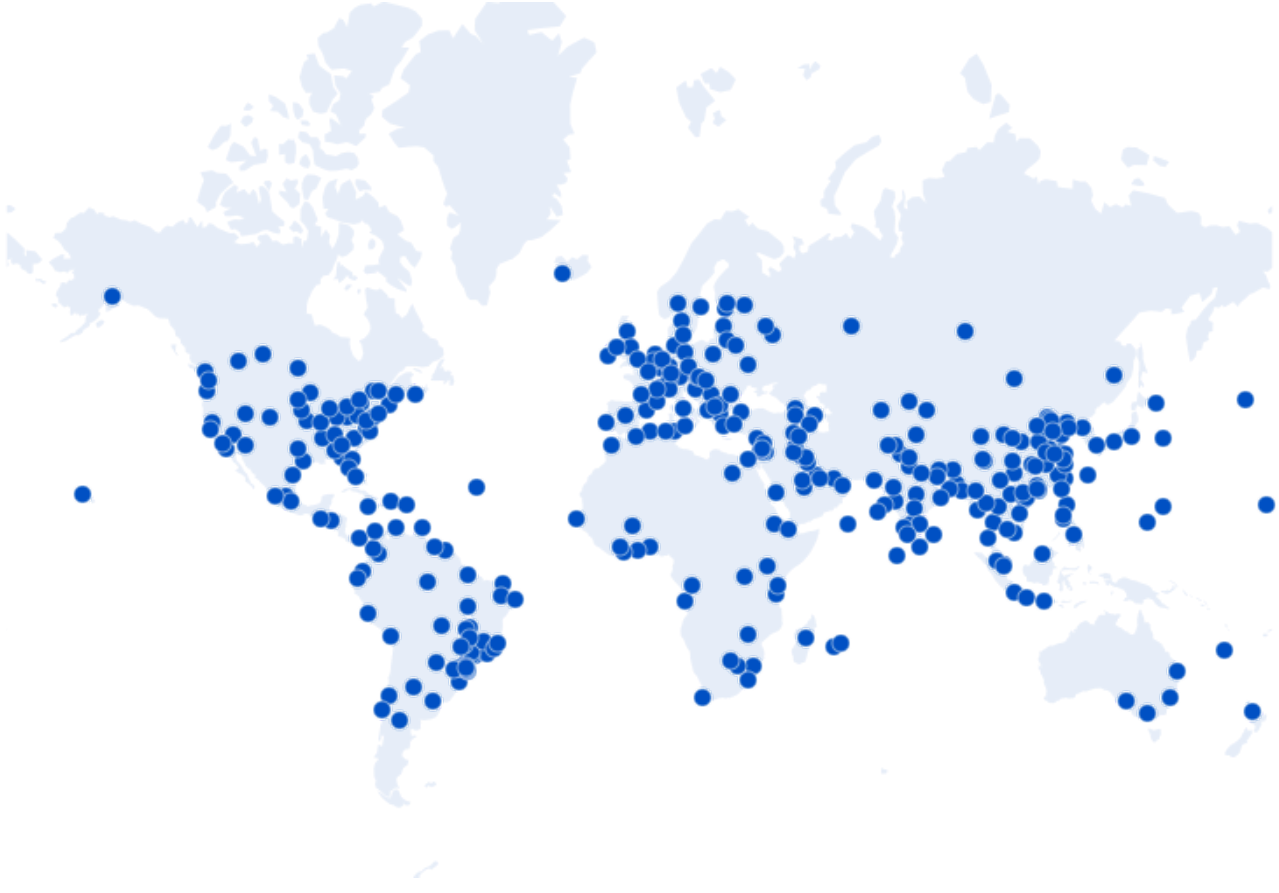


CLOUDFLARE WORKERS

Cloudflare's Javascript Runtime/Platform

Learn more: Cloudflare Workers , How workers work

LOW LATENCY



Learn more: Cloudflare network

V8 BASED

- highly performant
- always up to date with Chrome and Node.js
- V8 isolates => (almost) zero cold starts

Learn more: Fine-Grained Sandboxing with V8 Isolates

CHEAP

- Free plan -> 100,000 requests per day
- $5 per month ->10 million included per month
  + $0.30 per additional million

Learn more: Workers pricing

CLOUDFLARE PAGES

- Workers + Static Assets Hosting
- Ideal for web pages/applications

Learn more: Cloudflare Pages

REMIX

- Fullstack React framework
- Modern UX & DX
- Web standards based
- Soon to be React Router v7
- Alternatives: Next.js , tanstack, waku, etc...

Learn more: Remix

REMIX + CLOUDFLARE

- Out of the box cloudflare support
- Polished APIs for accessing Cloudflare resources
- Starter template included in C3

GITHUB REPOSITORY

https://github.com/cmsparks/react-summit-cloudflare-fullstack-workshop-nov2024

EXERCISE 01 - UI IMPLEMENTATION

Let's implement our application's UI form:
exercises/01-ui-card-form

Git Tags:

exr01_start	*start of the exercise*
exr01_step_1	*solution till step 1*
exr01_step_2	*solution till step 2*
exr01_step_3	*solution till step 3*
exr01_done	*complete solution*

WHAT'S NEXT?

We still need to add stateful backend functionality to our application so we can:

- Generate card artwork
- Save and share our cards

To do that, we can use some useful Cloudflare resources, called Bindings

WHAT ARE BINDINGS?

Workers Bindings do two things:

- They grant capabilities/permissions (i.e. your worker has access to an R2 Bucket)
- They inject the service's API into your Worker's environment

Learn more: Bindings

USING BINDINGS

To generate and save cards, we'll be using the following Bindings:

- Workers KV
- R2
- Workers AI

Learn more: Workers KV , R2 , Workers AI

BENEFITS OF BINDINGS

There are multiple benefits to this model:

- Lower security risks
- No boilerplate needed
- Clear visibility on resources usage

Learn more: Why Workers environment variables contain live objects

EXERCISE 02 - BINDINGS & DEPLOYING

Let's setup our bindings and deploy our worker:
exercises/02-bindings.md

Git Tags:

exr02_start	*start of the exercise*
exr02_done	*exercise solution*

EXERCISE 03 - CARDMANAGER CLASS IMPLEMENTATION

Let's implement our Workers code using the CardManager class:
exercises/03-business-logic

Git Tags:

exr03_start	*start of the exercise*
exr03_step_1	*solution till step 1*
exr03_step_2	*solution till step 2*
exr03_step_3	*solution till step 3*
exr03_done	*exercise solution*

EXERCISE 04 - UI INTEGRATION

Let's now integrate our new backend logic with our frontend:
exercises/04-ui-integration

Git Tags:

exr04_start	*start of the exercise*
exr04_step_1	*solution till step 1*
exr04_step_2	*solution till step 2*
exr04_done	*exercise solution*

END-TO-END (E2E) TESTING

- Aims to test a system's overall behavior (against user expectations)
- More realistic than unit/integration testing
- More expensive and time consuming

PLAYWRIGHT

- E2E Testing framework from Microsoft
- Many features such as automatic waiting and retries, trace viewer, codegen, etc...
- Multi browsers and multi languages support
- Alternatives: cypress, puppeteer, etc...

Learn more: Playwright

EXERCISE 05 - E2ES IMPLEMENTATION

Let's implement some e2e tests:
exercises/05-e2e

Git Tags:

exr05_start	*start of the exercise*
exr05_step_1	*solution till step 1*
exr05_step_2	*solution till step 2*
exr05_step_3	*solution till step 3*
exr05_step_4	*solution till step 4*
exr05_step_5	*solution till step 5*
exr05_done	*complete exercise solution*

EXTRA EXERCISES

https://github.com/cmsparks/react-summit-cloudflare-fullstack-workshop-nov2024/tree/main/exercises/extra

Q & A

FEEDBACK

We'd appreciate you giving us any feedback you have on our workshop at this linked Google Form