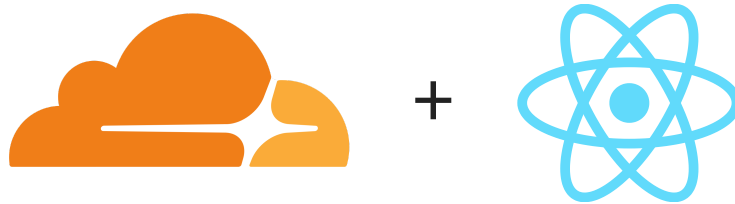


DEPLOY AND TEST FULL-STACK REACT APPS ON CLOUDFLARE





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

- Create Project (10m)
- UI implementation (30m)
- Deployment & Worker setup (10m)
- Break 1 (5m)
- Setup Workers code (1h)
- Break 2 (5m)
- UI Integration Tests (20m)
- E2E Test (20m)
- End of workshop (15m)

Total Time: ~3h

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 Services, including
 - Workers KV & R2 to store the card's information
 - and Workers AI to generate the card's image

COMPLETED APPLICATION





The Cloudflare's Javascript Runtime

(How Workers works)

LOW LATENCY



(Cloudflare network)

V8 BASED



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

(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


(Workers pricing)



CLOUDFLARE PAGES

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



- Fullstack React framework
- Modern UX & DX
- Web standards based
- Soon to be React Router v7 
- Alternatives: [Next.js](#), [tanstack](#), [waku](#), [@lazarv/react-server](#), etc...

REMIX + CLOUDFLARE

- Out of the box cloudflare support
- Polished APIs for accessing Cloudflare resources
- Started template included in [C3](#)



GITHUB REPOSITORY

EXERCISE 01 - UI IMPLEMENTATION

Let's implement our application's UI form:

[exercises/01-ui-card-form](#)

Start commit: `exr01 start`

(commits list)

WHAT'S NEXT?

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

- Generate card artwork
- Save and share that card artwork

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

USING WORKERS BINDINGS

To generate and save trading cards in our fullstack app, we'll be using Cloudflare Workers with the following Workers bindings:

- Workers KV - A lightweight key value store, used to store our card metadata
- R2 - S3 Compatible object storage, used for our card image data
- Workers AI - API for inference related tasks, for image generation

BENEFITS OF BINDINGS

There are multiple benefits to this model:

- Lower security risk because you can't accidentally leak an API token
- No boilerplate needed to initialize your API client
- Easier to understand what resources are being used

EXERCISE 02 - BINDINGS & DEPLOYING

Let's setup our bindings and deploy our worker:

[exercises/02-bindings.md](#)

Start commit: *exr01 done - exr02 start*

(commits list)

EXERCISE 03 - CARDMANAGER CLASS IMPLEMENTATION

Let's implement our Workers code using the
CardManager class:

[exercises/03-business-logic](#)

Start commit: *exr02 done - exr03 start*

(commits list)

EXERCISE 04 - UI INTEGRATION

Let's now integrate our new backend/workers logic
with our frontend/UI:

[exercises/04-ui-integration](#)

Start commit: *exr03 done - exr04 start*

(commits list)

END-TO-END (E2E) TESTING

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



PLAYWRIGHT

- E2E Testing framework from Microsoft
- Fast and intuitive
- Many features such as auto retries, trace viewers, codegen, etc...
- Multi browser and multi languages support
- Alternatives: cypress, puppeteer, etc...

EXERCISE 05 - E2ES IMPLEMENTATION

Let's implement an e2e test that tests our card generation logic:

[exercises/05-e2e](#)

Start commit: *exr04 done - exr05 start*

(commits list)

Q & A

FEEDBACK

We'd appreciate you giving us any feedback you have on our workshop at this linked [Google Form](#)