



THE NEED FOR SPEED



THE NEED FOR SPEED

Practical tips for optimising your CI/CD pipelines
(when your tests take all day)

Hi, I'm Zan 🙌

CircleCI Developer Advocate

I enable, educate, and inspire developer communities in CircleCI, DevOps, and CI/CD topics across EMEA.



twitter.com/zmarkan

github.com/zmarkan

@zmarkan



THE NEED FOR SPEED

Practical tips for optimising your CI/CD pipelines
(when your tests take all day)

A close-up shot of Chris Pratt as Peter Quill in a blue flight suit with shoulder harnesses. He has a concerned expression and is looking slightly to his left. The background is the interior of a futuristic spacecraft with metallic walls and glowing lights.

GROOT! PUT YOUR SEAT BELT ON!

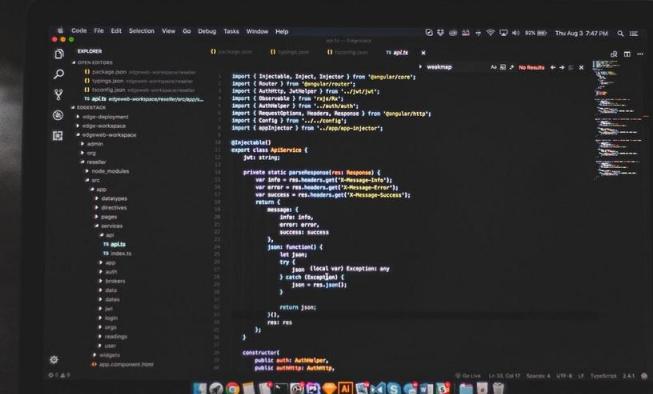
Story about an All Day Build





 circleci

@zmarkan



A screenshot of a laptop screen displaying a code editor. The editor shows a file named 'http.service.ts' with the following code:

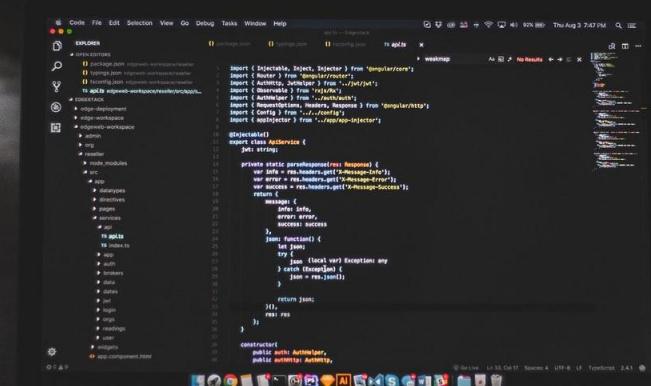
```
import { Injectable, Inject, Injector } from '@angular/core';
import { Router } from '@angular/router';
import { Observable } from 'rxjs/Observable';
import { Headers, Response } from 'angular/http';
import { AppInjector } from './app-injector';
import { Observable } from 'rxjs';

@Injectable()
export class HttpService {
  private string: string;
  private static parseHeaders(res: Response) {
    var pdfs = res.headers.get('X-PDF-Links');
    var success = res.headers.get('X-Message-Success');
    return {
      message: success,
      error: pdfs,
      errorType: 'error',
      errorMessages: [],
      success: success
    };
  }
  constructor(private injector: Injector) {
    setPromise();
  }
  private setPromise() {
    let json: (local var) <any>;
    catch (exception) {
      json = exception;
    }
    return json;
  }
  passRes(res) {
  }
}
public static parseHeaders(res: Response): any {
  return this.parseHeaders(res);
}
```









```
import { Injectable, Inject, Injector } from '@angular/core';
import { Router } from '@angular/router';
import { Observable } from 'rxjs/Observable';
import { Headers, Response } from 'angular/http';
import { AppInjector } from './app-injector';
import { RouterEvent } from '@angular/router';

@Injectable()
export class ApiService {
  static parseResponse(res: Response) {
    var fail = res.headers.get('X-Message-Fail');
    var success = res.headers.get('X-Message-Success');
    return {
      message: fail ? fail : success ? success : '',
      error: fail ? true : false,
      status: res.status
    };
  }
}

constructor(private injector: AppInjector) { }

private static parseResponse(res: Response) {
  var fail = res.headers.get('X-Message-Fail');
  var success = res.headers.get('X-Message-Success');
  return {
    message: fail ? fail : success ? success : '',
    error: fail ? true : false,
    status: res.status
  };
}

  constructor(private injector: AppInjector) { }

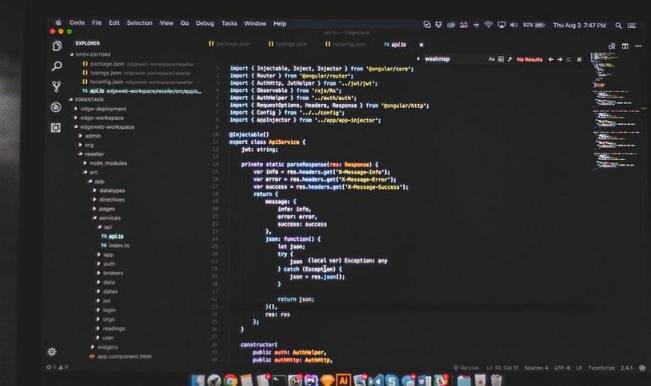
  static parseResponse(res: Response) {
    var fail = res.headers.get('X-Message-Fail');
    var success = res.headers.get('X-Message-Success');
    return {
      message: fail ? fail : success ? success : '',
      error: fail ? true : false,
      status: res.status
    };
  }
}
```





 circleci

@zmarkan



```
import { Injectable, Inject, Injector } from '@angular/core';
import { Router } from '@angular/router';
import { Observable } from 'rxjs';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { IHttpResponse } from './ihttpresponse';
import { AppInjector } from './appInjector';
import { HttpClientService } from './http-client.service';

@Injectable()
export class ApiService {
  private static parseResponse(res: Response) {
    var safe = res.headers.get('X-Message-Safe');
    var success = res.headers.get('X-Message-Success');
    return safe ? true : success ? false : false;
  }

  message(res: Response): boolean {
    var safe, error, errorMessage;
    if (res.ok) {
      safe = true;
      error = false;
      errorMessage = '';
    } else {
      safe = false;
      error = true;
      errorMessage = res.statusText;
    }
    return { safe, error, errorMessage };
  }

  constructor(private injector: Injector) {
    setHeaders();
  }

  setHeaders() {
    var json = (localVar) => {
      try {
        const localVar = JSON.parse(localVar);
        if (localVar === null) {
          return localVar;
        }
        return JSON.stringify(localVar);
      } catch (exception) {
        return localVar;
      }
    };
    return json;
  }

  parseResponse(res: Response): IHttpResponse {
    const json = this.setHeaders();
    const parsedRes = json(res);
    if (!parsedRes.safe) {
      return {
        status: res.status,
        message: res.statusText,
        data: null,
        error: true
      };
    }
    const data = parsedRes.data;
    if (data === null) {
      return {
        status: res.status,
        message: res.statusText,
        data: null,
        error: false
      };
    }
    return {
      status: res.status,
      message: res.statusText,
      data: data,
      error: false
    };
  }
}

constructor(private injector: Injector) {
  setHeaders();
}
```



Hindsight is
20/20



THE NEED FOR SPEED

Practical tips for optimising your CI/CD pipelines
(when your tests take all day)

CircleCI

Build pipeline defined in
.circleci/config.yml

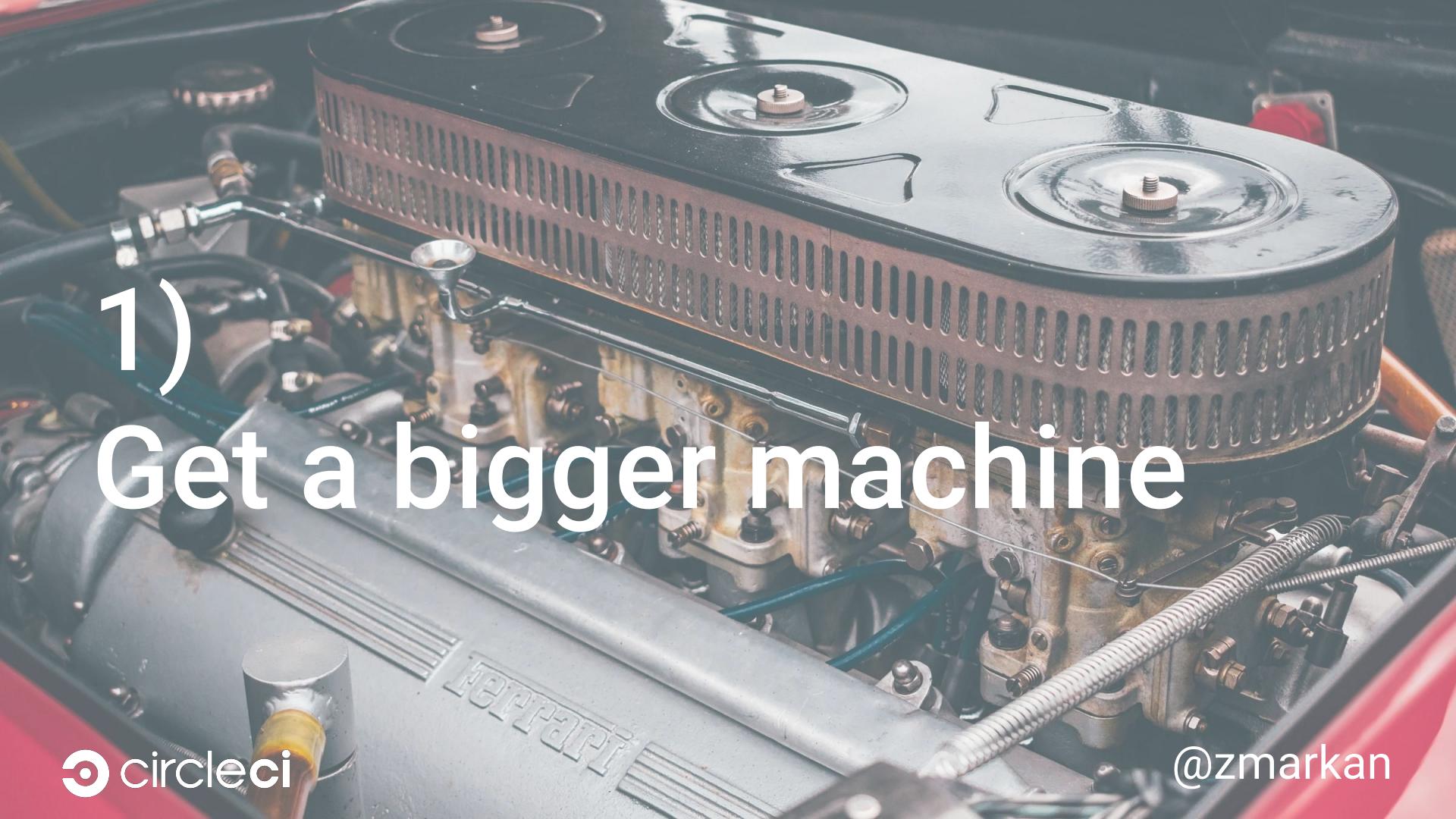
Workflows, Jobs, &
Commands

Execution environment

```
version: 2.1

# Define the jobs we want to run for this project
jobs:
  build:
    docker:
      - image: circleci/<language>:<version TAG>
        auth:
          username: mydockerhub-user
          password: $DOCKERHUB_PASSWORD # context / project UI env-var reference
    steps:
      - checkout
      - run: echo "this is the build job"
  test:
    docker:
      - image: circleci/<language>:<version TAG>
        auth:
          username: mydockerhub-user
          password: $DOCKERHUB_PASSWORD # context / project UI env-var reference
    steps:
      - checkout
      - run: echo "this is the test job"

# Orchestrate our job run sequence
workflows:
  build_and_test:
    jobs:
      - build
      - test
```



1)
Get a bigger machine

Resource Classes

```
jobs:  
  build:  
    docker:  
      - image: buildpack-deps:trusty  
        auth:  
          username: mydockerhub-user  
          password: $DOCKERHUB_PASSWORD # context / project UI env-var reference  
        resource_class: xlarge  
      steps:  
        ... // other config
```

https://circleci.com/docs/2.0/configuration-reference/#resource_class



Diminishing returns



A close-up photograph of a classic Ferrari dashboard. The central feature is a large, round tachometer with the text "GIRI x100" at the top and "ferrari" in the center. Below it is a smaller gauge with scales for "BENZINA" (gasoline) and "ACQUA" (water temperature). To the right is another gauge with scales from 20 to 110. On the left, there's a fuel cap with the Ferrari prancing horse logo. The dashboard is dark with silver accents and has a vintage aesthetic.

2)

Seek Insights



zmarkan
Zan Markan

■ Insights

All Projects

Last 90 Days

📅 Data below refers to the last 90 days. Please note that the data is not real time and there may be up to a 24 hour delay. Option to select the custom date range is coming soon.

■ Insights show usage metrics of active projects during the last 90 days.

X

Workflow runs	Total workflow duration ⓘ	Total credits consumed ⓘ	Overall success rate ⓘ
105	7h 12m 41s	1,648	58%

PROJECT	WORKFLOW	TOTAL CREDITS ⓘ	TOTAL DURATION ⓘ	RUNS	SUCCESS RATE ⓘ
CircleCITest	workflow	29	2m 58s	1	100%
Sunshine-Security	workflow	7	45s	1	0%
Weer9292	workflow	2	15s	1	0%
andfun-kotlin-gdg-finder	workflow	23	2m 21s	1	0%
android-pipelines-sample	workflow	44	5m 13s	3	66%
circleci-ml-pipeline	weekly	438	44m 19s	4	100%

Tuning what's running



Run subset of build...

- on commit,
- on main branch,
- on release tag...

Skip test runs with [skip-ci]

3) Cache is King



Caching

Dependencies

<https://circleci.com/docs/2.0/caching/>

Artifacts

<https://circleci.com/docs/2.0/persist-data/>

Git source

<https://circleci.com/docs/2.0/caching/#source-caching>

Docker layers

<https://circleci.com/docs/2.0/docker-layer-caching/>

```
version: 2
jobs:
  build:
    docker:
      # DLC does nothing here, its caching depends on commonality of the image layers.
      - image: circleci/node:9.8.0-stretch-browsers
        auth:
          username: mydockerhub-user
          password: $DOCKERHUB_PASSWORD # context / project UI env-var reference
    steps:
      - checkout
      - setup_remote_docker:
          docker_layer_caching: true
      # DLC will explicitly cache layers here and try to avoid rebuilding.
      - run: docker build .
```

4) Go Parallel

Test Parallelism

Using CircleCI CLI

Split by time, name, file size

Manually

Using env vars

```
# ~/.circleci/config.yml
version: 2
jobs:
  test:
    docker:
      - image: circleci/<language>:<version TAG>
        auth:
          username: mydockerhub-user
          password: $DOCKERHUB_PASSWORD # context / project UI env-var reference
    parallelism: 4
```

<https://circleci.com/docs/2.0/parallelism-faster-jobs/>



@zmarkan

5)

Go



or go



6)

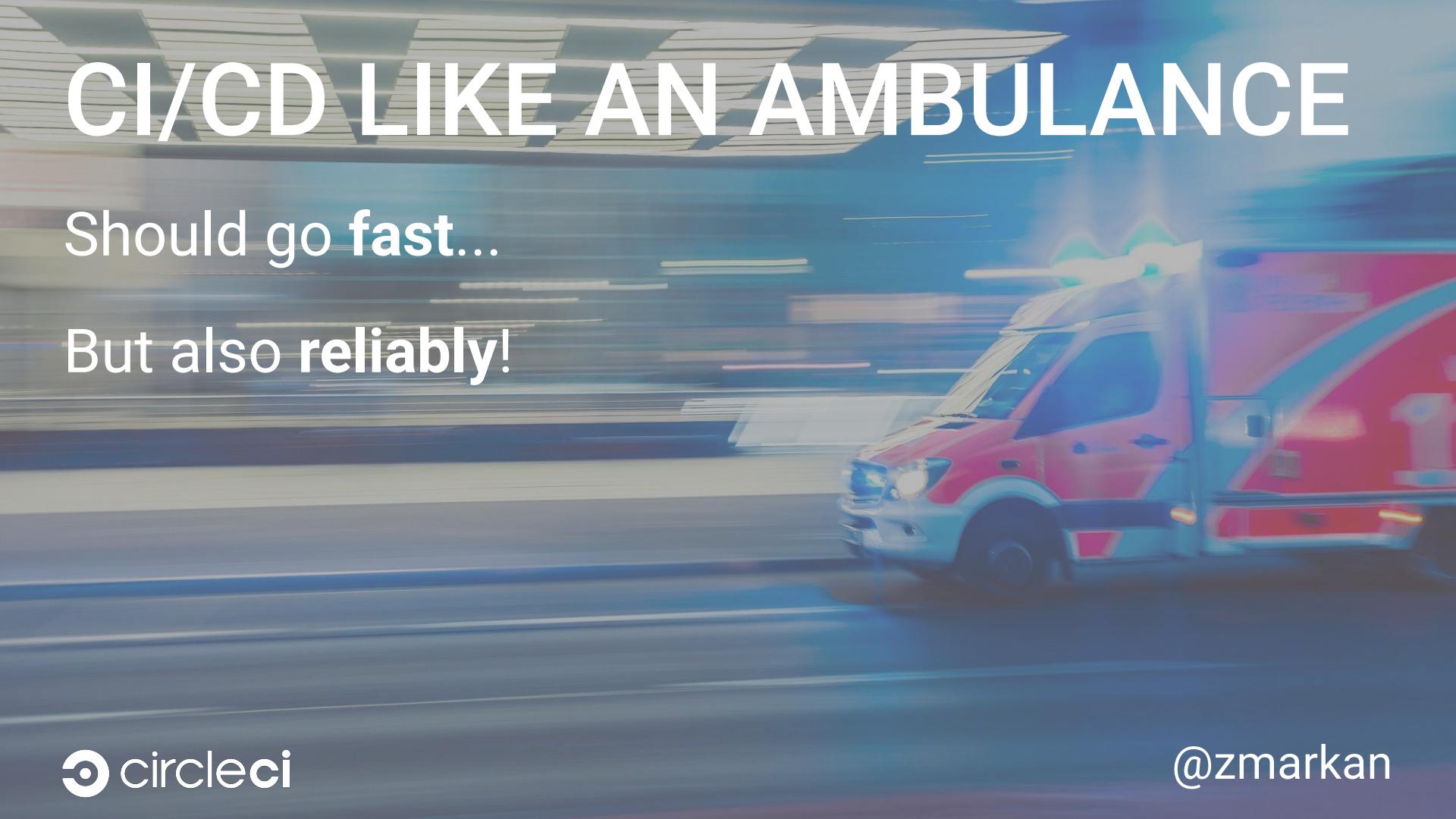
Teamwork





CI/CD IS LIKE AN AMBULANCE

CI/CD LIKE AN AMBULANCE



Should go **fast**...

But also **reliably**!

Numbers to back this up!

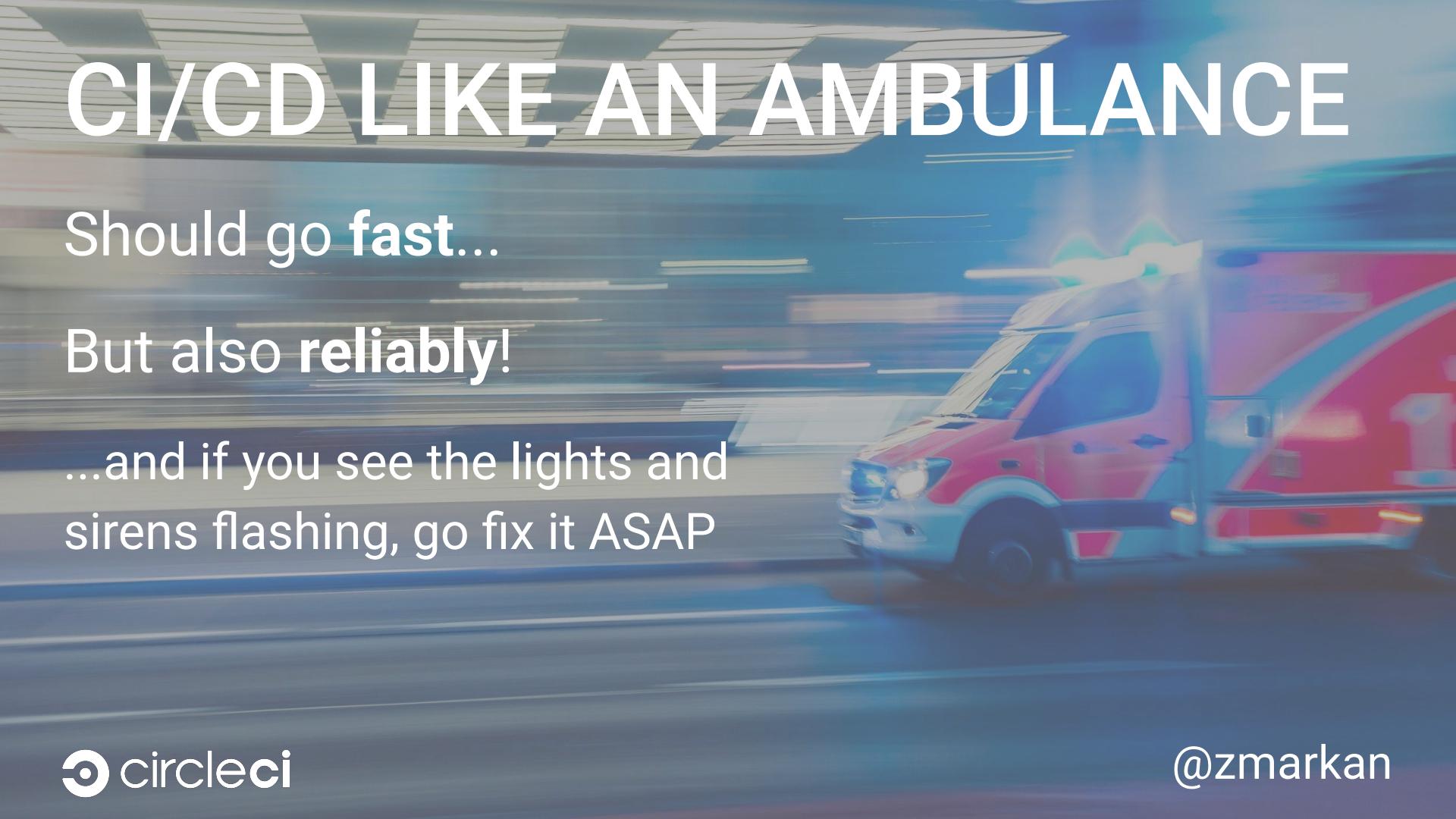
Data from 160k projects

- 5-10 mins for workflow run
- < 1 hr to recovery

<https://circleci.com/resources/2020-state-of-software-delivery/>



CI/CD LIKE AN AMBULANCE



Should go **fast**...

But also **reliably**!

...and if you see the lights and
sirens flashing, go fix it ASAP



THE NEED FOR SPEED

Practical tips for optimising your CI/CD pipelines
(when your tests take all day)

Where to go next?



circleci.com/developer



discuss.circleci.com



circleci.com/blog

