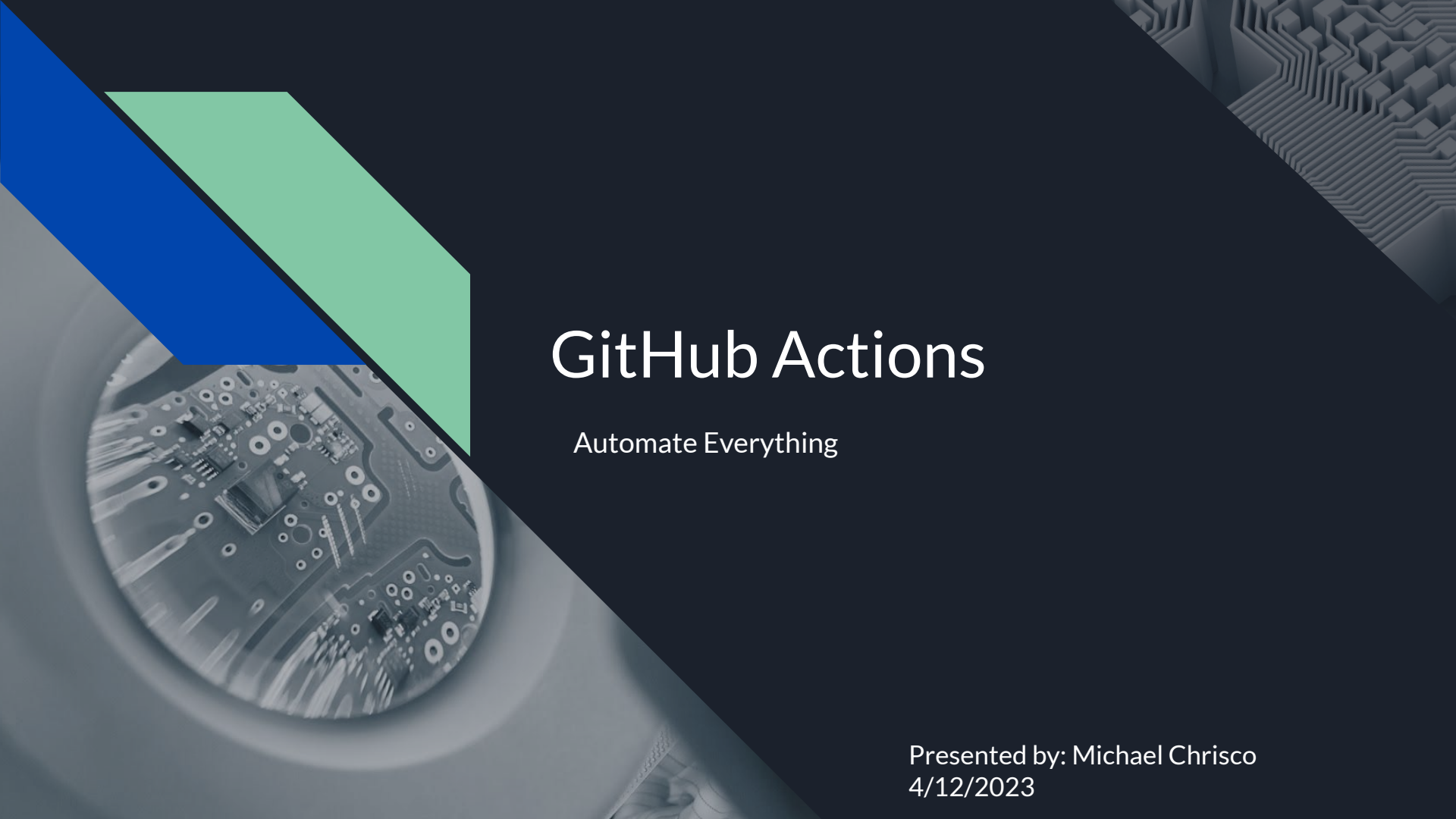

GitHub Actions

Automate Everything

Presented by: Michael Chrisco
4/12/2023

What is GitHub Actions?

Continuous Integration (CI) allows teams to establish a consistent and automated way to build, package, and test applications. With integration processes (AKA workflows and script pipelines) in place, teams are more likely to commit code changes more frequently, which leads to better collaboration and software quality. In a nutshell, CI allows teams to build and test the application before it gets deployed.

Continuous Delivery (CD) is the ability to deploy changes of all types including new features, configuration changes, bug fixes and experiments, into production.

GH Actions facilitates CI/CD by providing a platform that can automatically build, test, and deploy with a wide variety of software stacks. In my talk, I am going to be showcasing how projects have used GitHub Actions, some examples of GH Action specific features, and how to get started with your CI/CD journey to automate everything as much as feasibly possible.

Why should I care?

Automated testing enables **continuous delivery**, which ensures software quality, security and increases the profitability of code in production.

CI/CD pipelines enable a much shorter time to market for new product features, creating happier customers and lowering strain on development.

The great increase in overall speed of delivery enabled by CI/CD pipelines improves an organization’s competitive edge.

Organizations with a successful CI/CD pipeline can attract great talent. By moving away from traditional waterfall methods, engineers and developers are no longer bogged down with repetitive activities that are often highly dependent on the completion of other tasks.

Why should I care Part 2. The interesting stuff

- Build/Test/Deploy (normal CI/CD)
- Split out workflows to lint, give specific feedback to developers.
- Run your own boxes!
- Github Marketplace has loads of ready-made scripts to use for your project.
- Auto-create releases for your project.
- Label your Pull Requests auto-magically (using committed files)
- Create badges for your project
- Auto-deploy builds/deploy expo to mobile stores: https://docs.expo.dev/eas-update/github-actions/
- Create your own: https://github.com/actions/typescript-action

….And many many more: https://github.com/sdras/awesome-actions

How Continuous Integration Works

Whenever code changes occur, GitHub Actions verifies the code via an automated build (usually a Linux container), allowing team members to detect problems before the changes hit production.

Github will trigger an action that will hook into GitHub Actions, informing the developer if the build passes or fails.

Green Check Mark = Good to go!

Red X = Job failed :(

Continuous Delivery

Continuous Delivery (CD) is the automated process in which new features, configuration changes, bug fixes and experiments, get deployed into production. By adding CD to a workflow, it allows teams to get code out fast and deploy often.

Having a CD pipeline can help streamline complex deploy procedures and ensures the code is always in a deployable state. By automating the deployment process, this procedure reduces risk. It also reduces stress for the team by letting the CI/CD platform do the deployment rather than a local machine, which can be less stable than a reproducible docker container.

If a project is deployable, chances are the project can be scripted out to deploy on a CD delivery model using the same commands. (terraform, chef, bash scripts, powershell, etc.. etc…)

GitHub Actions as a CI/CD solution

- Its officially supported by GitHub!
- Lots of examples in almost every popular stack within their own documentation.
- No magic, just short lived Linux/macOS/Windows boxes.
- Its relatively easy to understand (big plus for maintainability)
- Fully integrated with GitHub
- Keeps ssh keys/environment variables safe for build/deployment purposes.
- Reasonable price range (about the same as CircleCI)

Anatomy of a GitHub Actions Script

A project's continuous integration and deployment process is orchestrated through multiple *.yml/*.yaml files within the `.github/workflows/` folder at the root of the repository.

Components of GitHub Action script:

- **Events**: An event is a specific activity in a repository that triggers a workflow run.
- **Workflows**: Responsible for orchestrating multiple jobs.
- **Jobs**: Responsible for running a series of steps that perform commands.
- **Steps**: Run commands (such as installing dependencies or running tests) and shell scripts to do the work required for your project.

Hello World - Our first GH Action script

We will be building a trivial GH Action workflow and then go into useful versions of the above!

Things to remember:

- All CI/CD is just command line/scripts under the hood. NO MAGIC!
- If you can do it in the command line, you can do it as a script.
- No need to make it do everything you need at once, find out how to build -> test -> deploy

GH Action Script - MVP

```yaml
name: Ruby

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  test:

    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v3
      - name: Set up Ruby
        uses: ruby/setup-ruby@359bebbc29cbe6c87da6bc9ea3bc930432750108
        with:
          ruby-version: '3.1'
      - name: Install dependencies
        run: gem install awesome_bot
      - name: Verify all links are valid
        run: awesome_bot --allow-redirect README.md

```

TatStat - Example

TatStat: https://github.com/Shift3/tatstat-RN/actions

Other Examples!

React Boilerplate with GH Actions:
https://github.com/michaelachrisco/gh-actions-boilerplate-client-react/actions

IFG: https://github.com/Shift3/ifg/actions

New GH Action Features (2023)!

GH Actions has default capability to scan for security vulnerabilities and secrets for different technologies/workflows. Make sure you don't commit something you shouldn’t!

Manage cache workflows from the web interface.

Import from other Ci/CD platforms automatically with Github Action Importer

…..and many more: https://github.blog/changelog/label/actions/

Pros/Cons of the Platform - Pros

Pros:

GH Actions is well supported by GitHub

Easier to understand than most CI/CD competitors.

You can Self-host runners to bring down cost/run specific machines (Mac OS)

Same exact docker containers that other platforms uses (CircleCI ubuntu latest same).

Onboarding is sooo much easier.

Pros/Cons of the Platform - Cons

Cons:

If you use community packages, tend to break (PHP/Laravel experience).

They are moving fast, so you have to keep up with breaking changes. Instructions will often hard-set git commit refs which can be brittle.

Command line tool is not the greatest, broke a few times running working GH Actions. It's getting better but still having issues.

Personal Observations

You can make multiple workflows within different yml files rather than just one universal like CircleCI. Splitting out processes and labeling is much easier than other CI/CD platforms in the same space.

The process of getting onboarded with GH Actions is extremely easy. Actions tab and add a workflow. They allow you to drop in some common stacks (python/ruby/php/javascript(node)).

Their caching is comparable to CircleCI. yarn/npm both supported

Speed is reasonable, taking an average of 2m 30 sec for a non-cached build and an average of 2 mins on a cached workflow.

Personal Observations

Most platforms are supporting GH Actions (like expo, node, Django, etc...). There are official guides on how to integrate with it.

Workflows, jobs, and other such verbiage is the same as CircleCI. The nice part is how much GH specific terms that the workflows can use. This can make it easier for new people to take a look for the first time.

Price is about the same as CircleCI for what we use it for.

Resources

Official Documentation

Quick Start Instructions

Circleci vs GH Actions (by CircleCI)

Migrating from CircleCI to GH Actions

GH Action Changelog

Collection of awesome actions

Thank you!

Questions? Comments?