



Deployment and CI/CD Automation

Skills Bootcamp in Front-End Web Development

Lesson 14.3



The background is a dark charcoal gray with a series of parallel diagonal lines running from the top-left to the bottom-right. Overlaid on this are several teal-colored geometric shapes: a large central triangle pointing right, a smaller triangle to its left, and a square to its right. Scattered around these shapes are various white line-art symbols, including a plus sign, a minus sign, a circle with a dot, a circle with a horizontal line, a circle with a vertical line, a circle with a diagonal line, a circle with a cross, a circle with a dot, a circle with a horizontal line, a circle with a vertical line, a circle with a diagonal line, a circle with a cross, a circle with a dot, a circle with a horizontal line, a circle with a vertical line, a circle with a diagonal line, and a circle with a cross.

WELCOME

Learning Objectives

By the end of class, you will be able to:



Create a CD pipeline between a GitHub Repo and Netlify.



Optimize a CD pipeline between GitHub and Netlify by creating YAML scripts that trigger and run unit tests as well as deploy their code.

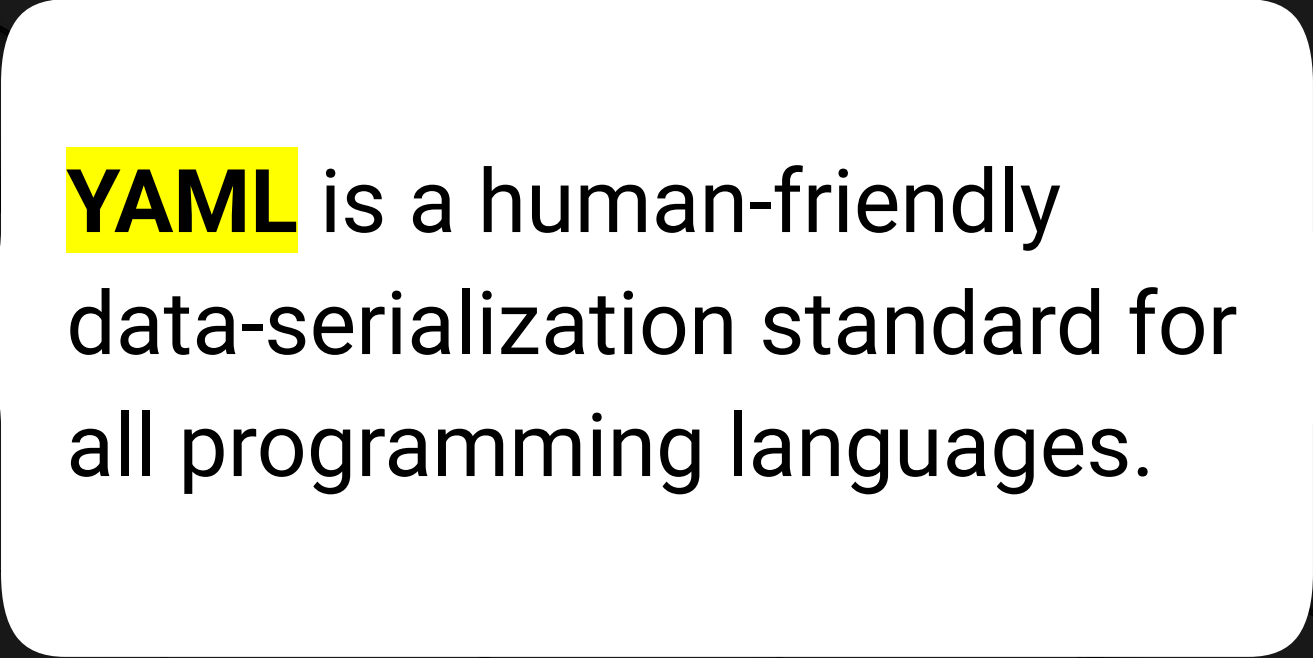


Explain and discuss what YAML is and how it is utilized in a CI/CD pipeline.





What is YAML?



YAML is a human-friendly
data-serialization standard for
all programming languages.



is a recursive acronym for

YAML **A**in't **M**arkup **L**anguage

YAML...



Is commonly used for configuration files and in applications where data are being stored or transmitted.



Targets many of the same communications applications as Extensible Markup Language (XML) but has a minimal syntax that intentionally differs from SGML.



Uses both Python-style indentation to indicate nesting and a more compact format that uses [...] for lists and {...} for maps so that JSON files are valid YAML 1.2.

YAML

In this project, we are using YAML to configure our GitHub and Netlify CI/CD pipeline to:



**Install our
dependencies**

**Autonomously
run our unit test**

**Deploy our
codebase to
Netlify**





What is a pipeline?

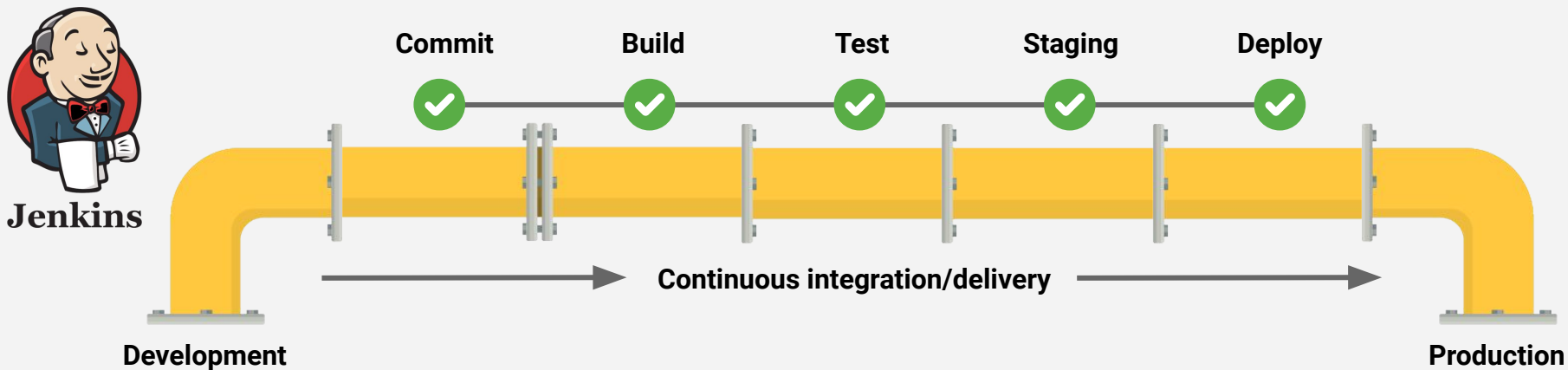
In software development, a **pipeline** is the series of stages that a product goes through on its way to being finished.

It can be useful to think of pipelines as literal pipes—designers and developers push a product down the pipes as they work, refining it as they go.



What Is a Jenkins Pipeline?

A continuous delivery (CD) pipeline is an automated expression of your process for getting software from version control right through to your users and customers.



Every change to your software (committed in source control) goes through a complex process on its way to being released. This process involves building the software in a reliable and repeatable manner, as well as progressing the built software (called a “build”) through multiple stages of testing and deployment.



What is CI/CD?



Continuous integration/ continuous deployment (CI/CD)

is the concept of automatically updating machines on your network whenever your IaC files change.

Advance Testing Concepts and CI/CD Automation

Whenever you change a machine's configuration file:



Continuous integration (CI)

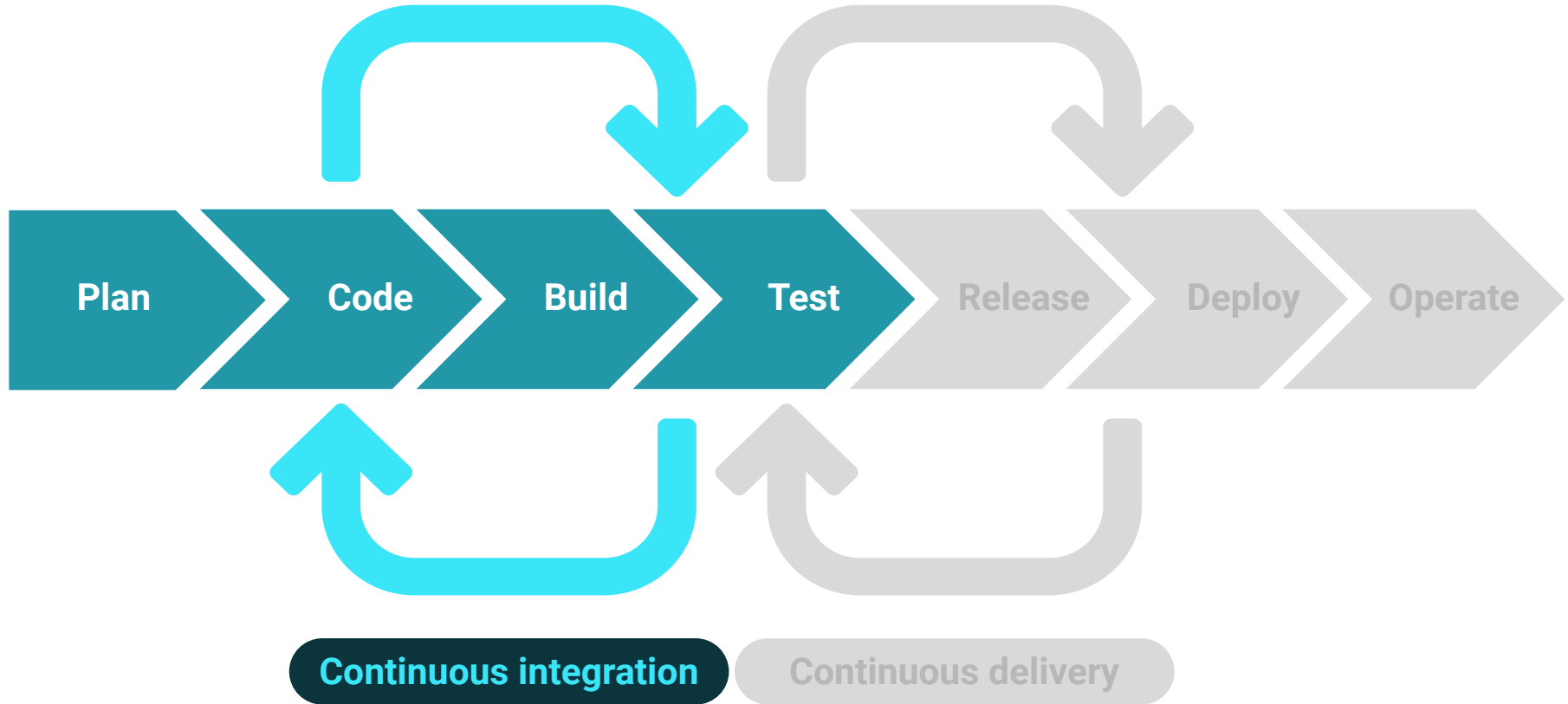
ensures that a new version of that machine is built immediately.

Continuous deployment (CD)

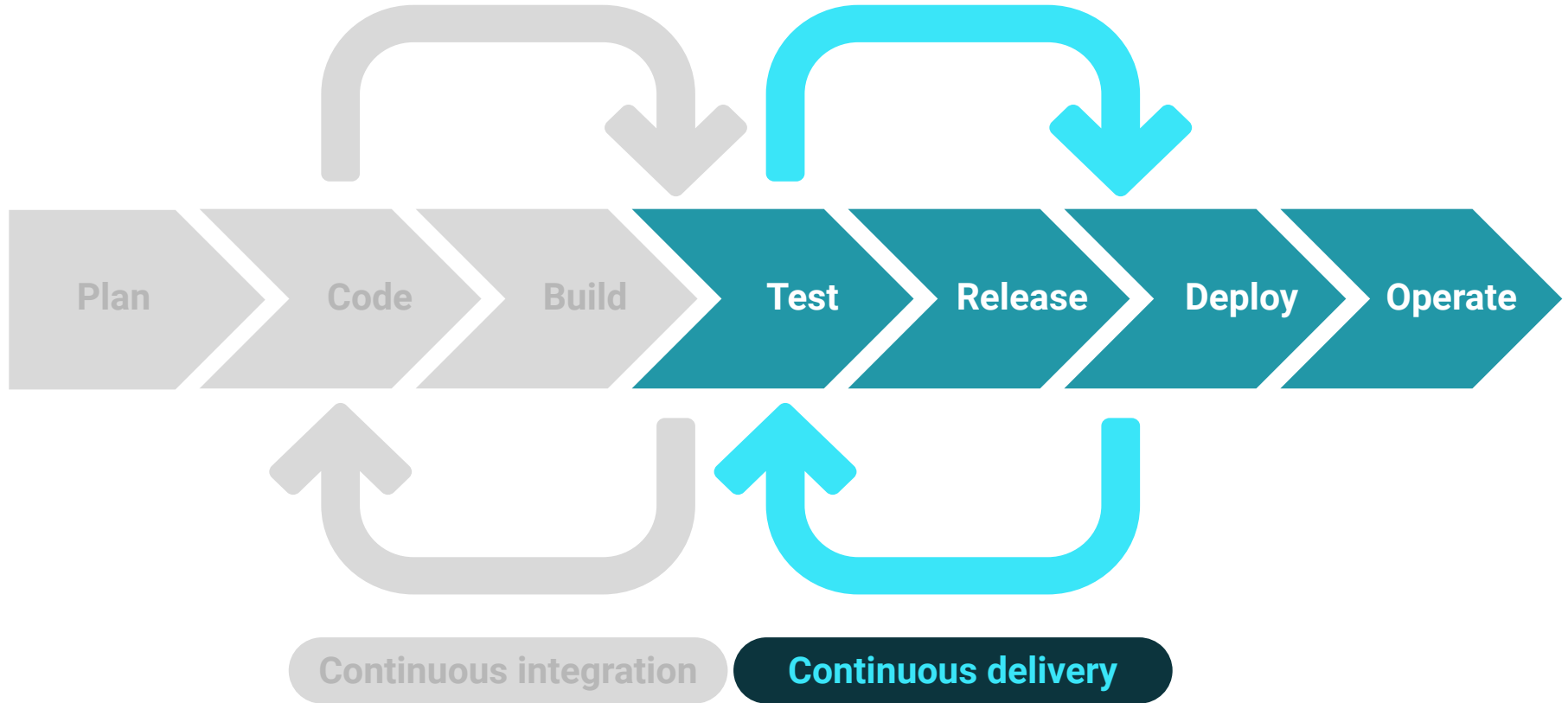
ensures that this new version is automatically deployed to your live environment.

The primary advantage to CI/CD is that it allows you to manage your entire network simply by updating IaC text files.

Continuous Integration

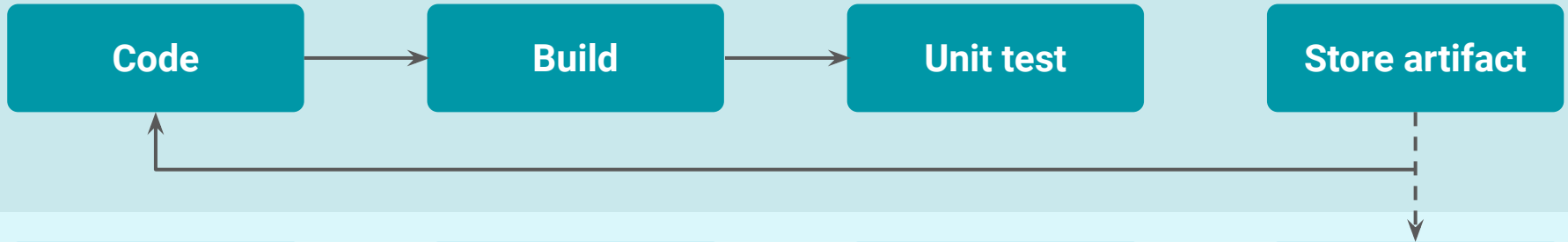


Continuous Delivery



CI/CD Pipeline Workflow

Continuous integration



Continuous delivery

Continuous deployment



Time to Code

Game of Thrones Character Gallery: Deployment

Suggested Time:

15 Minutes

Game of Thrones Character Gallery: Deployment

Instructions:



Create a [Netlify](#) account with [GitHub](#) credentials.



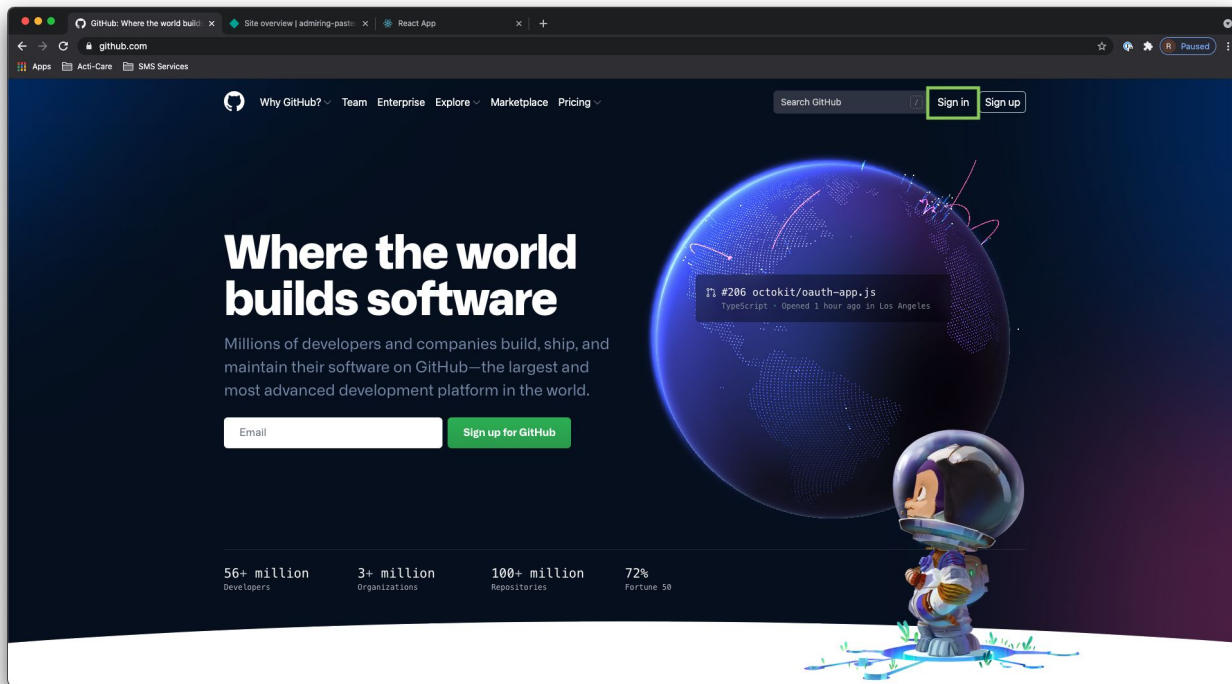
Create a repo called `got-character-gallery` on `GitHub`.



Create a continuous development pipeline between `Netlify` and the `got-character-gallery` repo housed on `GitHub`.

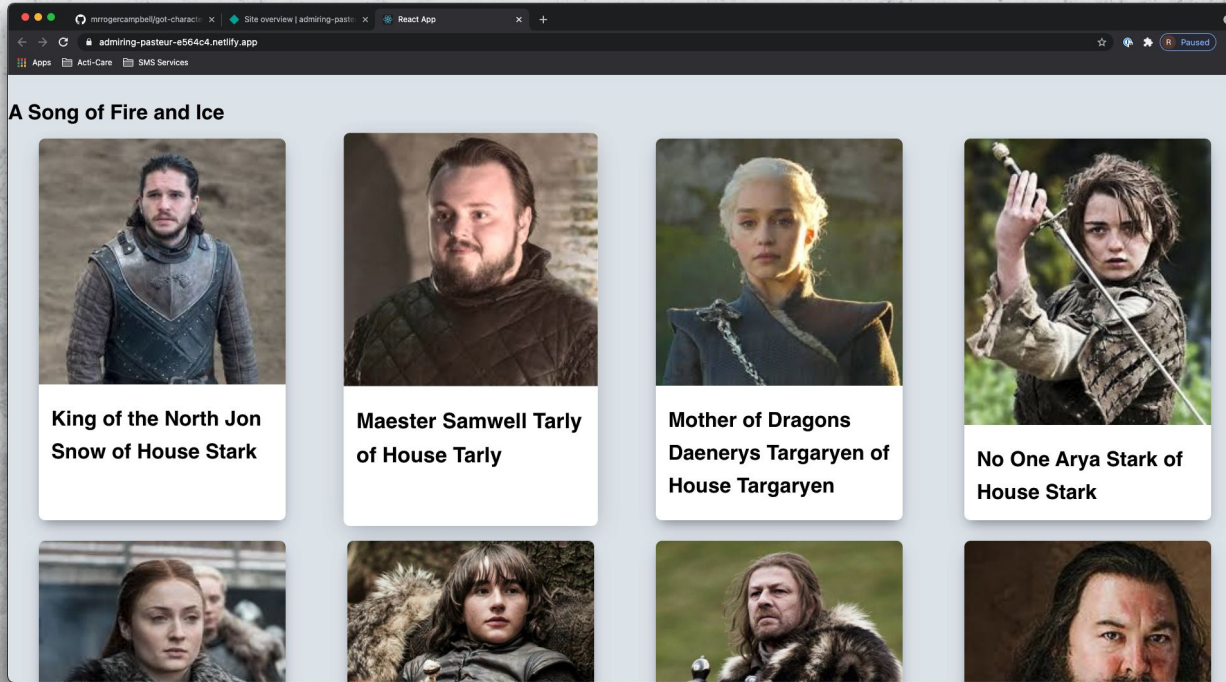
Game of Thrones Character Gallery: Deployment

Go to [GitHub](https://github.com) and sign in:



Game of Thrones Character Gallery: Deployment

Example: <https://stupefied-joliot-0bb3db.netlify.app>





Pre-Written Tests

Tests

Solution: App.test.js

```
// App.test.js

// importing the App Component
import App from '../App'
// importing the shallow render method from the enzyme package
import { shallow } from 'enzyme'

// Created a custom setup method, when invoked it returns a shallowly rendered instance of the App component
const setup = () => shallow(<App />)

// Created a custom setup method which accepts two arguments:
// 1. wrapper - the wrapper which you want to search; ie the component
// 2. val - the data-test value you would like to locate
const findByTestAttr = (wrapper, val) => wrapper.find(`[data-test='${val}']`)

// testing to see if
it('App Component Renders Without Error', () => {
  const wrapper = setup()
  const appComponent = findByTestAttr(wrapper, "component-app")

  expect(appComponent.length).toBe(1)
})
```



We are testing to see if the **App** component renders without errors.

Tests

Solution: CharacterGallery.test.js

```
// CharacterGallery.test.js
import CharacterGallery from '../components/CharacterGallery'
import { shallow } from 'enzyme'

const setup = () => shallow(<CharacterGallery />)

const findByTestAttr = (wrapper, val) => wrapper.find(`[data-test='${val}']`)

it('CharacterGallery Component Renders Without Error', () => {
  const wrapper = setup()

  const charComponent = findByTestAttr(wrapper, "component-char-gallery")

  expect(charComponent.length).toBe(1)
})
```



We are testing to see if the
CharacterGallery component
renders without errors.



In `Character.test.js` we are
performing more in-depth testing.

Questions?





Activity: Develop Character Gallery

In this activity, you will reinforce and build upon the testing and React skills that you have learned so far by designing and developing test based on feature requirements inside the activity's [README](#).

Suggested Time:

20 Minutes



Time's Up! Let's Review.

Review: Develop Character Gallery

Solution: App.js

```
// App.js
import './App.css';
import React from 'react';
import CharacterGallery from './components/CharacterGallery'

function App() {

  return (
    <div data-test="component-app">
      <CharacterGallery />
    </div>
  );
}

export default App;
```



Here we are just importing and rendering the `CharacterGallery` within the `App` component.

Review: Develop Character Gallery

Solution: CharacterGallery.js

```
// CharacterGallery.js
import React from 'react';
// Be sure to copy the json file from the activity directory into your project
import characterData from '../data/characterData.json'
import Character from './Character'

const CharacterGallery = () => {
  const listOfChars = characterData.map((char, i) => <Character {...char} key={i} />)
  return (
    <div data-test="component-char-gallery">
      {listOfChars}
    </div>
  );
};

export default CharacterGallery;
```

Review: Develop Character Gallery

In this component, we are:

01

Importing all the character data from `characterData.json` and storing them in a variable called `characterData`.

02

Iterating over the `characterData` variable with a `map method` and returning a new array that contains an instance of the `Character` component for each character dataset in the `characterData` array.

- We are storing this new array within the `listOfChars` variable.

03

Rendering the `listOfChars` variable within the `CharacterGallery` component.

Review: Develop Character Gallery

Solution: Character.js

```
// Character.js
import React from 'react';

const Character = ({ name, imgUrl, birth, death, race, realm, spouse }) => {
  return (
    <div data-test='component-character'>
      <h1 data-test='char-name'>{name}</h1>

      <img data-test='char-img' src={imgUrl} alt={name} />

      <ul data-test='char-list'>
        <li data-test='char-birth'>
          Date of Birth: {birth}
        </li>
        <li data-test='char-death'>
          Date of Death: {death}
        </li>
        <li data-test='char-race'>
          Race: {race}
        </li>
        <li data-test='char-realm'>
          Realm: {realm}
        </li>
        <li data-test='char-spouse'>
          Spouse: {spouse}
        </li>
      </ul>
    </div>
  );
};

export default Character;
```

Review: Develop Character Gallery

Here, we are:

01

Destructuring the props being passed to the Character component

- For more on destructuring props in React, see this [article](#).

02

Rendering each destructured prop into its corresponding element.

Questions?



A close-up, high-angle shot of a computer keyboard. The central focus is a large, white, rectangular key with rounded corners. On this key, there is a dark blue icon of a coffee cup with three wavy lines above it representing steam. Below the icon, the word "Break" is printed in a dark blue, serif font. The key is set against a light-colored, textured keyboard surface. Surrounding the main key are other keys: to the left, a key with double quotation marks; above, a key with a right bracket; and to the right, a key with a left bracket. The lighting is soft and even, highlighting the texture of the keys and the surface.

Break



Activity: Repo Setup and Deployment

In this activity, you will reinforce and build upon the development skills that you have learned so far by creating a **remote repo** on GitHub and connecting it to your **local repo** and deploying to Netlify

Suggested Time:

15 Minutes



Time's Up! Let's Review.

Questions?





Time to Code

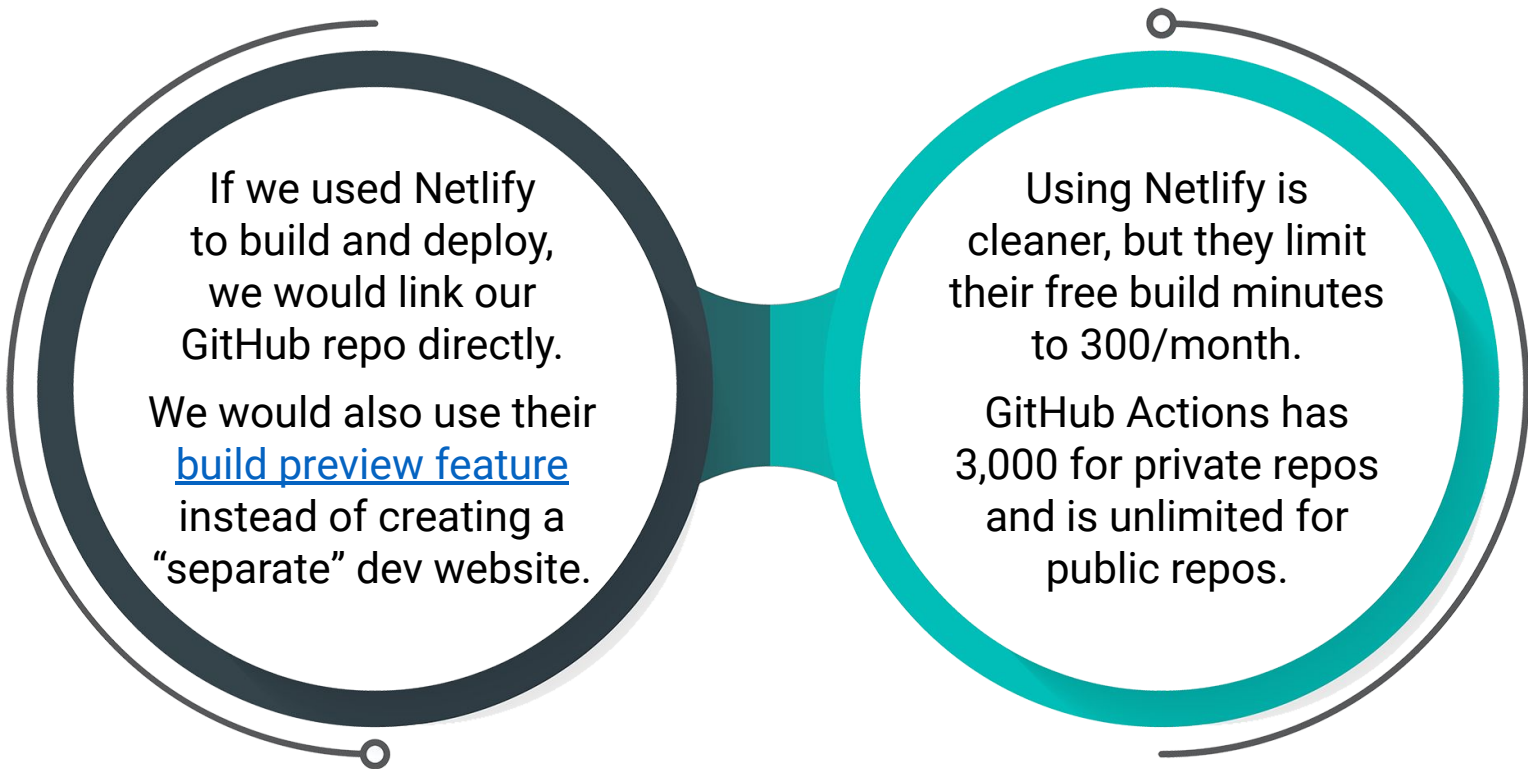


Configure GitHub and Netlify

Suggested Time:

15 Minutes

Configure GitHub and Netlify





Time to Code

Set Up Workflow Actions and YAML

Suggested Time:

15 Minutes

Solution: Set Up Workflow Actions and YAML

01

In the **root of your project**, create a **.github** directory.

- `mkdir .github`

02

Inside of **.github**, create a directory called **workflows**.

- `mkdir ../github/workflows`

03

Inside of **workflows/push.yml**, copy the code from:

[06-We_SetupWorkflowActionsAndYaml](#)

Questions?





Time to Code



Push It!

Suggested Time:

15 Minutes

Questions?





Today's Challenge:

Updating Your React Portfolio

(Instructions sent via Slack)

Suggested Time:

4 hours

Questions?



*The
End*