Infrastructure-Components

Aug 21, 2019

Contents

CHAPTER 1

This is a complete Serverless React App!

Create, Start, and Deploy React Apps easily!:

```
import * as React from 'react';

import { Route, SinglePageApp } from "infrastructure-components";

export default (
    <SinglePageApp
        stackName = "example"
        buildPath = 'build'
        region='eu-west-1' >

        <Route
            path='/'
            name='Infrastructure-Components'
            render={() => <div>Hello from a React Web App!</div>}/>

    </SinglePageApp>
);
```

CHAPTER 2

Infrastructure-Components do all the technical configuration for you

Compile and Pack

Infrastructure-Components transpile your Typescript-based React components and bundle them into ready-to-use packages–without any further configuration required.

- Webpack
- Babel + Loaders
- Typescript

Application

Use state-of-the-art libraries to speed up app development.

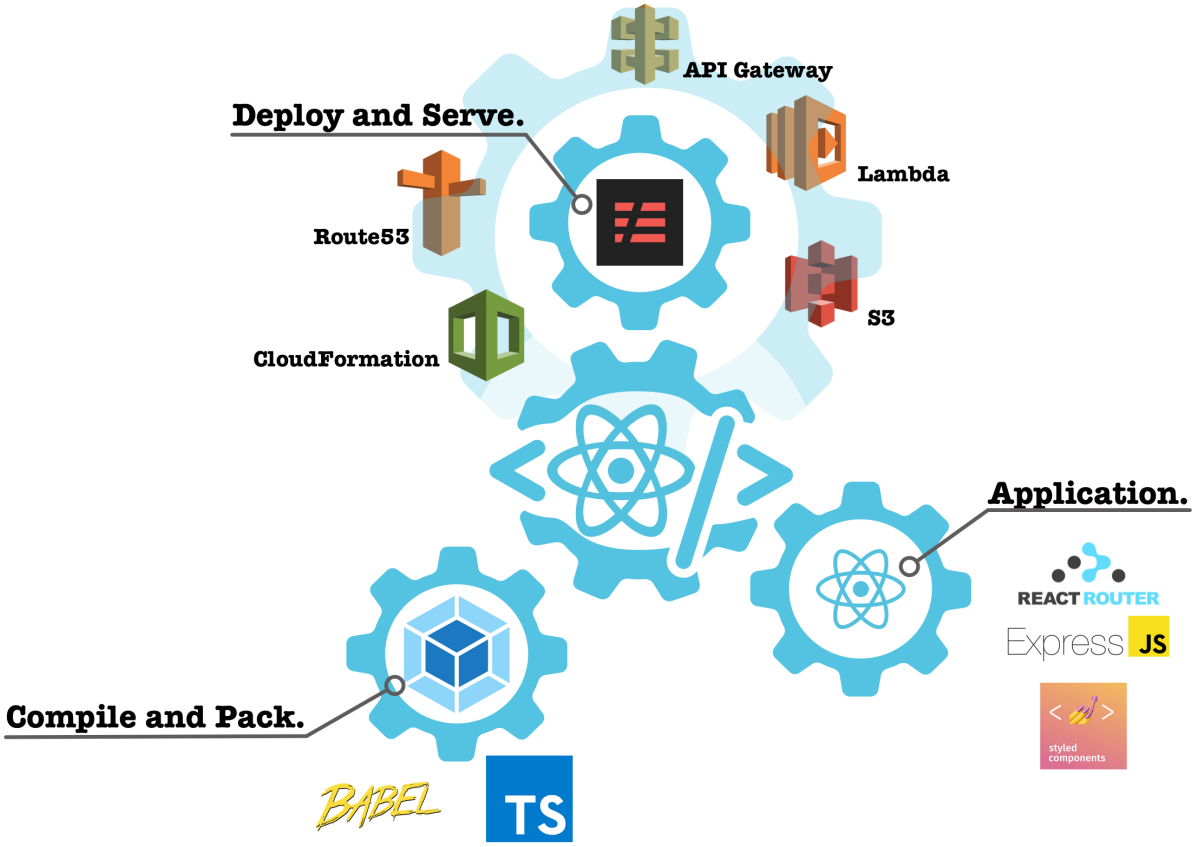
- React Router
- ExpressJs
- Styled Components
- React Helmet
- GraphQL
- ...

Deploy and Serve

Deploy your application with a single command! Infrastructure-Components create the whole infrastructure stack for you.

- Lambda-Functions
- API-Gateway
- S3
- DynamoDB
- CloudFront

- Route53
- CloudFormation
- ...



2.1 Getting Started

2.1.1 Start with an Example

Our GitHub-Repository contains exemplary projects of each supported architecture topology:

- Single-Page-App
- Isomorphic App

Fork or clone any of these repositories and run `npm install`.

2.1.2 Install manually

You can install infrastructure-components easily:

```
npm install --save infrastructure-components
```

infrastructure-scripts provide all the scripts required to *build*, *start*, and *deploy*. This lib contains many libraries that you only need during development/deployment. Thus, install this library as devDependency:

```
npm install --save-dev infrastructure-scripts
```

Infrastructure-components use the Serverless framework that you need to install globally:

```
npm install -g serverless
```

Finally, apps (e.g. single-page-app, isomorphic-app) and components (environment, webapp) can have further dependencies. Have a look at them in this documentation.

2.2 Apps

Apps represent the top-level-components of an **infrastructure-components**-based project. In your entry-point source file, e.g. `src/index.tsx` (yes, you can use typescript with jsx-extension in this file – out of the box), you need to export an app-component as default, like this:

```
import * as React from 'react';

import {
    SinglePageApp
} from "infrastructure-components";

export default (
    <SinglePageApp
        stackName = "spa-example"
        buildPath = 'build'
        region='us-east-1' />
);
```

The app-component determines the architecture of your project at runtime. Each architecture has advantages and may be suited for certain use-cases.

While a change of the architecture is a breaking change in a traditional project setup, **infrastructure-compponents** support this out of the box! If you want to change the architecture of your application, just replace the top-level-component and you're done!

Each app-component supports running it offline (on your development machine) and deploying it to the Amazon Web Services (AWS) cloud with a single command!

2.2.1 SinglePageApp

A Single-Page-App (SPA) is an interactive web application that rewrites the current page rather than loading new pages from a server. In fact, a SPA consists of a very basic html that simply loads the app's Javascript-code. Once loaded, this code creates a user experience that avoids interruption between successive pages and behaves more like a desktop application than a traditional website.

apps/spa provides further details on Infrastructure-Component's `SinglePageApp`.

2.2.2 ServiceOrientedApp

A Service-Oriented-App (SOA) is an interactive web application just like a Single-Page-App. Additionally, it supports services. These services run as AWS Lambda function on the server side.

apps/soa provides further details on Infrastructure-Component's `ServiceOrientedApp`.

2.2.3 IsomorphicApp

An Ismorphic-App (aka universal app) is an interactive web application that complements the advantages of a single-page-app with the ability of server-side-rendering. In an isomorphic setting, the server renders the whole Javascript-code and returns a full html-file to the browser. As a result, the browser can display the html without any further processing.

An Isomorphic-App downloads the Javascript-code to the browser, too. This enables a dynamic user experience.

apps/isomorphic provides further details on Infrastructure-Component's `IsomorphicApp`.

2.3 Components

Components complement the top-level-apps of an **infrastructure-components**-based project. Components are children (direct or indirect) of the app, like:

```
<SinglePageApp
    stackName = "example"
    buildPath = 'build'
    region='us-east-1' >

    <Route
        path='/'
        name='Infrastructure-Components'
        render={() => <div>Hello from a React Web App!</div>}/>

</SinglePageApp>
```

Note: Which components you can use and may depend on the top-level-app.

2.3.1 Webapp

The WebApp-Component is available only in an apps/isomorphic. In this context, it creates a client-app with a custom html and Javascript code.

See components/webapp for more details.

2.3.2 Service

The Service-Components is available in apps/soa and apps/isomorphic. It specifies a server-side route to one or many components/middleware-components

See components/service for more details.

2.3.3 Middleware

The Middleware-Components is available only in an apps/isomorphic or as child of a components/service. In an Isomorphic App context, it specifies a server-side function that runs whenever a user requests a page from the server.

See components/middleware for more details.

2.3.4 Route

A Route-Component specifies a custom path (at the domain of your app) that gets served by its render-function. This function lets you easily render your own React-components.

See components/route for more details.

2.3.5 Environment

An Environment-Component defines a runtime environment of your app. With environments you can distinguish your development-environments from your production-environment. An environment lets you attach a real domain to it, like www.your-domain.com.

See components/environment for more details.

2.3.6 DataLayer

The DataLayer-component adds a NoSQL-database (DynamoDB) to your app. It takes It takes components/entry and components/service as children. The DataLayer is available in a apps/soa and in an apps/isomorphic.

See components/datalayer for more details.

2.3.7 Entry

The Entry-component describes the type of items in your database. The entry must be a child of a components/datalayer.

See components/entry for more details.

2.4 Scripts

The library infrastructure-scripts provides the scripts command.

Run it with one of the arguments specified below and the relative path to the file that exports the your app-component, e.g. `src/index.tsx`.

Scripts enable you to `build`, `start` (offline), `deploy`, and attach a `domain` to your **infrastructure-components**-based project.

2.4.1 Build

The `build`-script prepares your project for local start or deployment:

```
scripts build src/index.tsx
```

If you prefer using the usual `npm run build` command for building, simply add the following script to your `package.json` file:

```
"scripts": {
  "build": "scripts build src/index.tsx"
}
```

The build process adds further scripts to your `package.json`. These let you start your software stack offline, start hot development, and deploy it.

Which scripts are created depends on your app-component and its `<Environment />`- and `<WebApp />`-components.

Look at the app-components for more details on the created scripts:

- apps/spa
- apps/isomorphic

2.4.2 Run Offline

Run `scripts {your_stackName} src/index.tsx` or `npm run {your_stackName}` to start your `<SinglePageApp />` or your `<WebApp />` within an `<IsomorphicApp />` in hot-development-mode.

Wait until the console says that your app is running and open localhost:3000 in your browser.

Changes to your source code become effective immediately in this mode. Just edit your source code and reload your page in the browser. Note that an `<IsomorphicApp />` does not run with a backend (e.g. middlewares) in this mode!

If you want to stop the app, use "ctrl-c" (or whatever command your console-application uses to interrupt a running script).

2.4.3 Start

The script `npm run start-{your_environment_name}` starts your `<IsomorphicApp />` locally (offline).

Open your the url localhost:3000 in a browser and you can see your application in action. Have a look at the console of your development environment for outputs made on server-side (e.g. middlewares)

Note: Changes at your source code require running `npm run build` before they become effective in this mode!

2.4.4 Deploy

Once you ran the `build` script, your `package.json` will contain a script for each environment your app contains:

```
npm run deploy-{your_environment_name}
```

From here, the scripts create the whole infrastructure stack on your AWS account. You'll get back an URL that now serves your app.

Note: This script may take some time to complete!

2.4.5 Domain

Have a look at our tutorial on how to register and prepare a domain within AWS.

If you specified an `<Environment/>`-component with a ready-to-use-domain and once you deployed your app, you can initialize the domain with the following command:

```
npm run domain-{your_environment_name}
```

Note: You only need to run this command once. But it may take quite some time to complete!

2.5 Help and Support

Infrastructure-Components are under active development. If you find a bug or need support of any kind, please have a look at our Spectrum-Chat.

Further, we frequently publish descriptions and tutorials on new features on Medium.com.