INDYJS

# VUE-GRAPHQL-APPSYNC-IONIC

**Brad Pillow**
**Chief Software Architect @ WorkHere**

Email: bpillow@pillowsoft.com, brad.pillow@workhere.com
Twitter: @BradPillow
Github: pixelmeister

# VUE

# GETTING STARTED

To install the new package, use one of those commands:

```sh
npm install -g @vue/cli
# OR
yarn global add @vue/cli
```

After installation, you will have access to the `vue` binary in your command line. You can verify that it is properly installed by simply running `vue`, which should present you with a help message listing all available commands.

You can check you have the right version (3.x) with this command:

```sh
vue --version
```

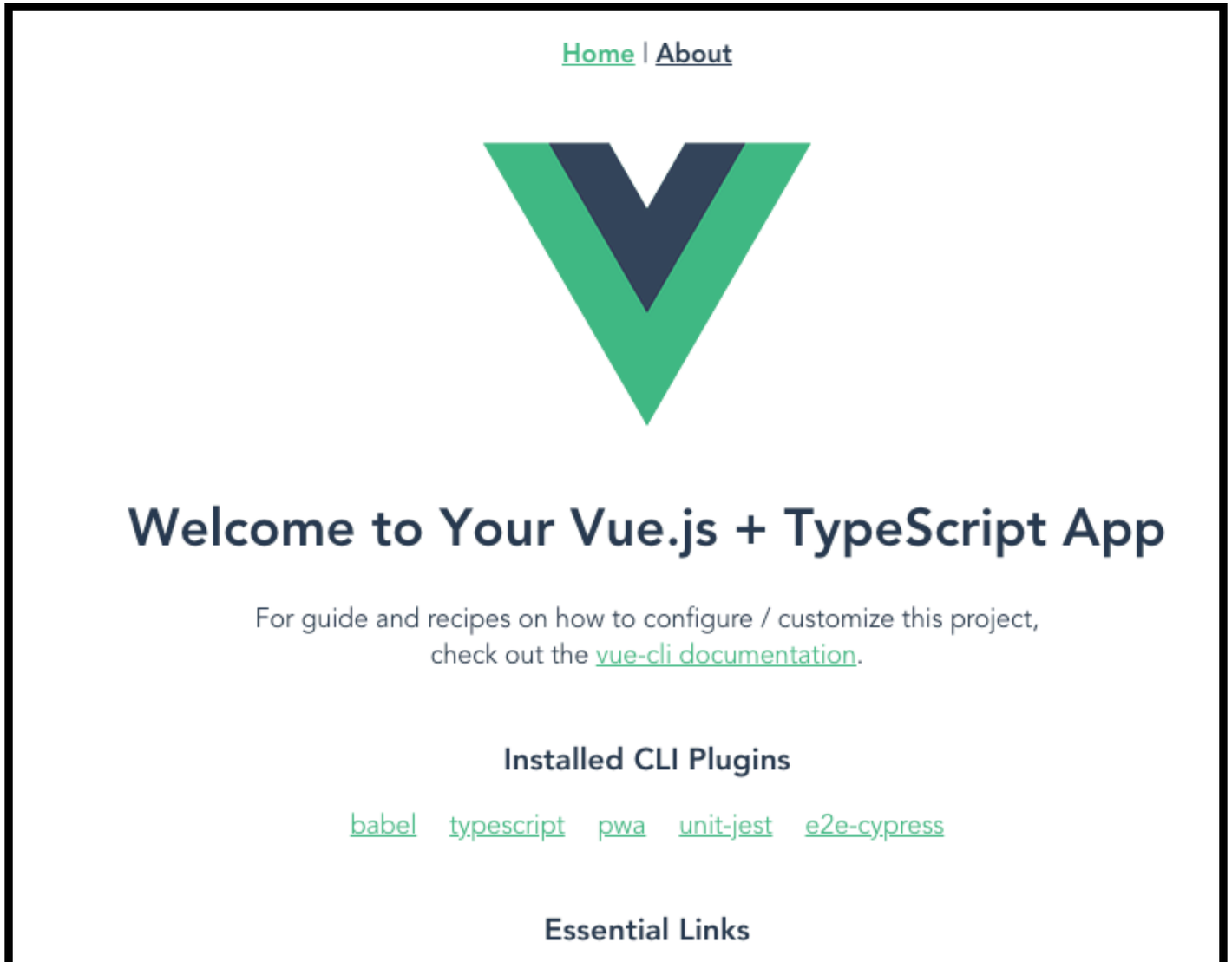
# BUILD & RUN

```
Vue CLI v3.8.0
? Please pick a preset: (Use arrow keys)
❯ tsvue (vue-router, vuex, sass, babel, typescript, pwa, unit-jest, e2e-cypress)
  AllChoices (vue-router, vuex, sass, babel, typescript, pwa, unit-jest, e2e-cypress)
  default (babel, eslint)
  Manually select features
```

```
[3/4] 🔗  Linking dependencies...
[4/4] 🚚  Building fresh packages...
success Saved lockfile.
✨  Done in 11.10s.
⚓  Running completion hooks...

📄  Generating README.md...

🎉  Successfully created project indyjs-vue-graphql-appsync-ionic.
👉  Get started with the following commands:

 $ cd indyjs-vue-graphql-appsync-ionic
 $ yarn serve
```

```
 DONE  Compiled successfully in 3029ms

No type errors found
Version: typescript 3.0.1
Time: 2861ms

  App running at:
  - Local:   http://localhost:8080/
  - Network: http://192.168.86.213:8080/

  Note that the development build is not optimized.
  To create a production build, run yarn build.
```

# RESULT


Home | About
Welcome to Your Vue.js + TypeScript App
For guide and recipes on how to configure / customize this project,
check out the vue-cli documentation.
Installed CLI Plugins
babel typescript pwa unit-jest e2e-cypress
Essential Links

# WHAT DO WE HAVE

- A PWA capable web app
- A redux-like state store call Vuex
- A build in router call “vue-router”, similar to react-router
- E2E testing setup
- Babel & Typescript, with modifications allowed without “ejecting”
- Virtual DOM, one way data flow, similar to typical React goodies
- Several way to create components from templates to JSX

# WHY USE VUE?

- You don't like peanut butter in your chocolate and you don't like HTML in your JS code
- The "paradox of choice"....there are too many options in React and you don't know where to start
- You prefer to use community supported as opposed to company supported open source projects
- You're a fan of templating
- You want a "batteries include" solution

# WHY USE VUE?

- Pre-configured webpack features such as hot module replacement, code-splitting, tree-shaking, efficient long term caching, error overlays, etc.
- ES2017 transpilation (plus common proposals like object rest spread and dynamic import) and usage-based polyfills injection via Babel 7 + preset-env
- Support for PostCSS (with autoprefixer enabled by default) and all major CSS pre-processors
- Auto-generated HTML with hashed asset links and preload/prefetch resource hints
- Modes and cascading environment variables via .env files
- Modern mode: ship native ES2017+ bundle and legacy bundle in parallel
- Build targets: build Vue Single-File Components into a library or native web components

# WHY USE VUE?

- TypeScript/Babel
- PWA
- Vue Router & Vuex
- ESLint / TSLint / Prettier
- Unit Testing via Jest or Mocha
- E2E Testing via Cypress or Nightwatch

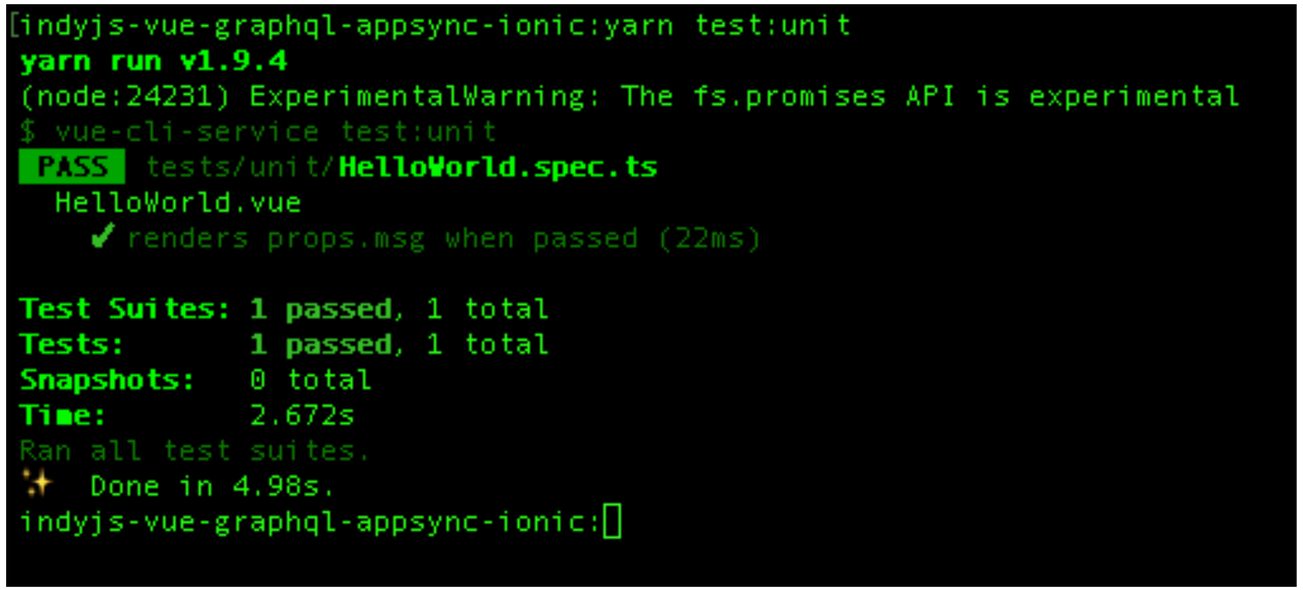
# MORE GOODIES

- Excellent graphql support
- Default framework Vuetify great for web sites, but for hybrid mobile web, Framework7, Ionic and OnsenUI have Vue support, too

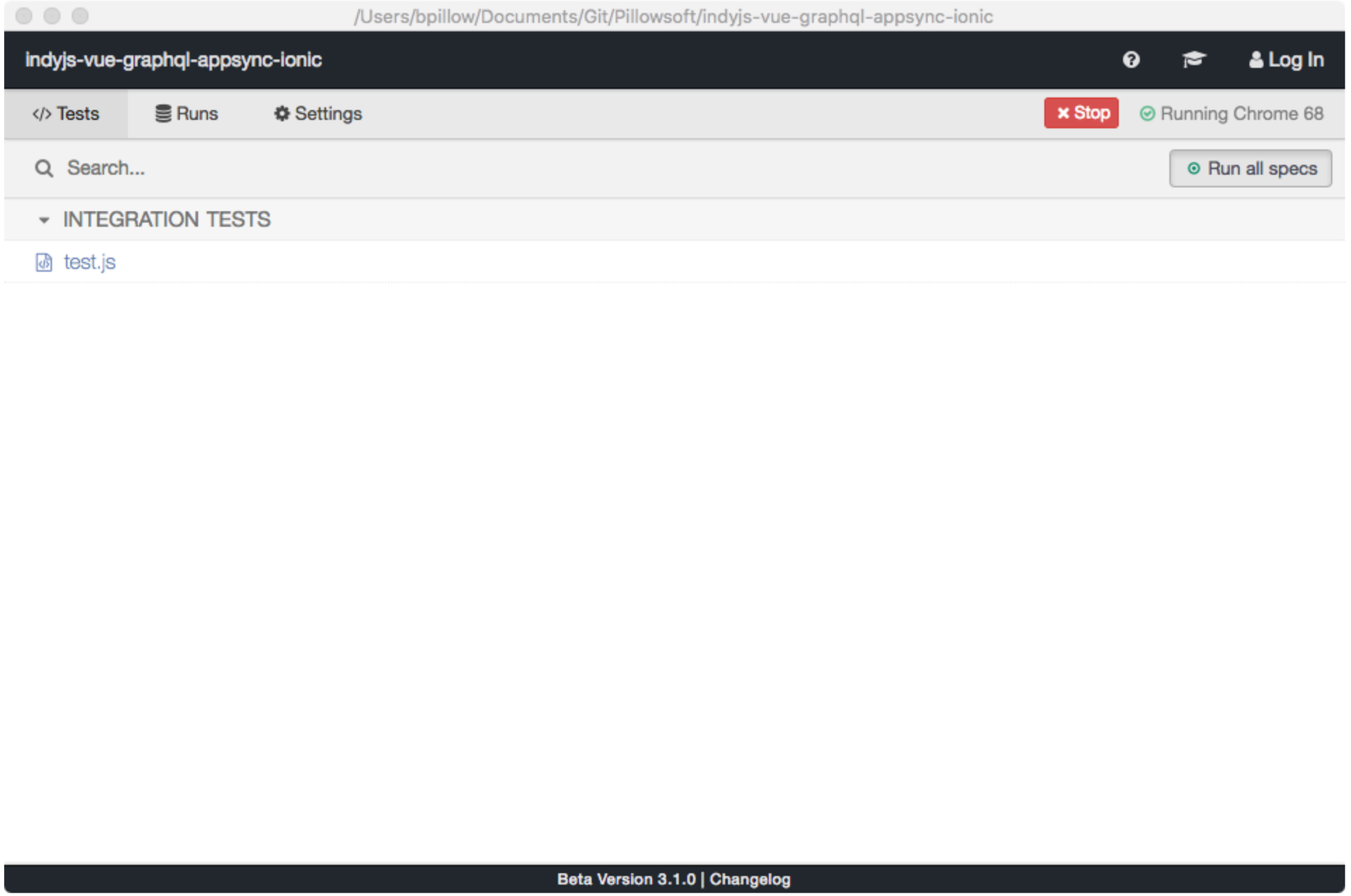
# WHY SHOULDN'T YOU USE VUE?

- Facebook has a lot of smart people working on React which may make for greater advances
- No React-Native…but there is Native-Script for Vue
- You already know React, React Router, JSX, etc. If so, then stick with your knitting.

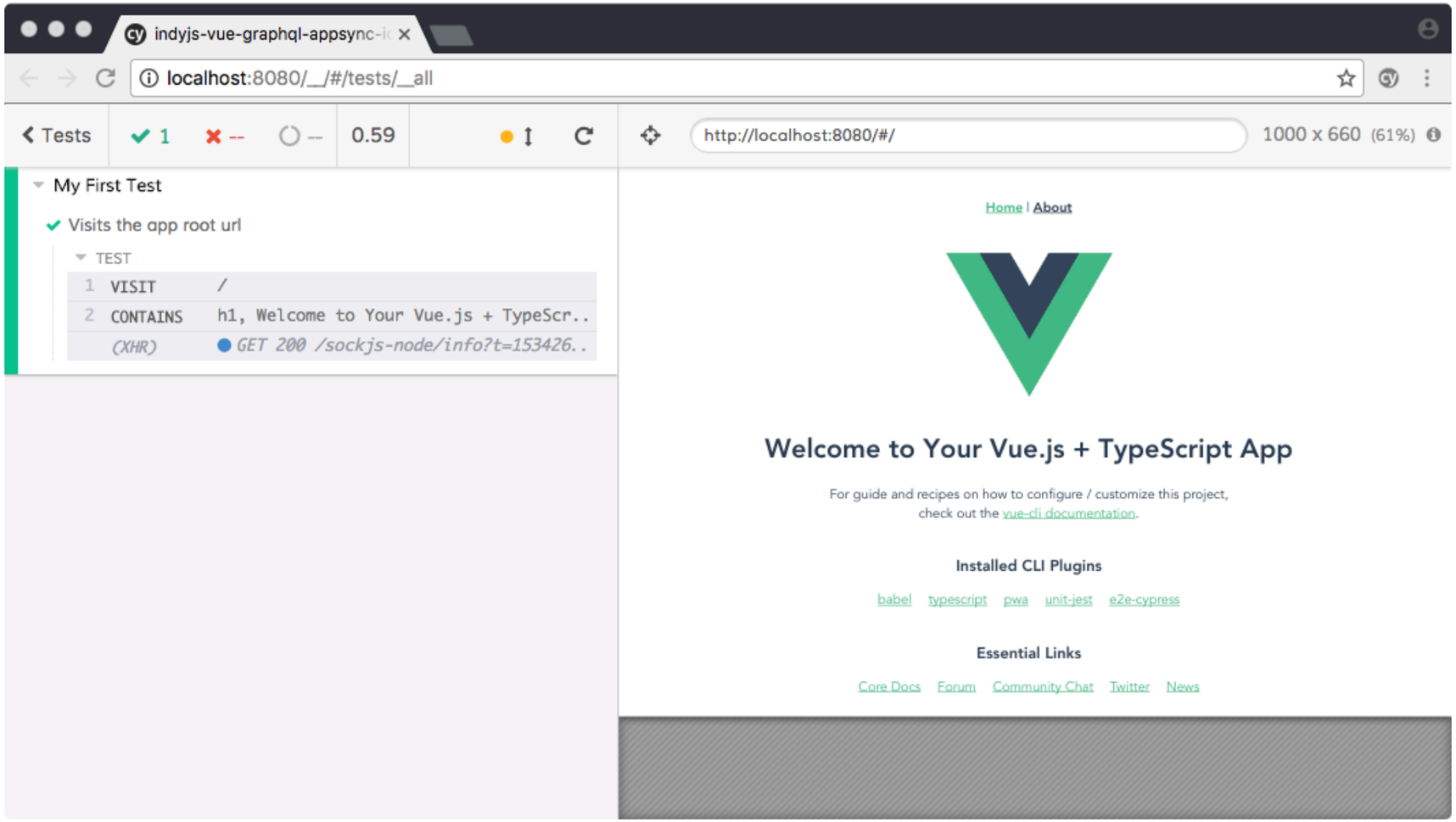
# TESTING


[indyjs-vue-graphql-appsync-ionic:yarn test:unit
yarn run v1.9.4
(node:24231) ExperimentalWarning: The fs.promises API is experimental
$ vue-cli-service test:unit
PASS tests/unit/HelloWorld.spec.ts
HelloWorld.vue
✓ renders props.msg when passed (22ms)
Test Suites: 1 passed, 1 total
Tests: 1 passed, 1 total
Snapshots: 0 total
Time: 2.672s
Ran all test suites.
✨ Done in 4.98s.
indyjs-vue-graphql-appsync-ionic:

# E2E TESTING

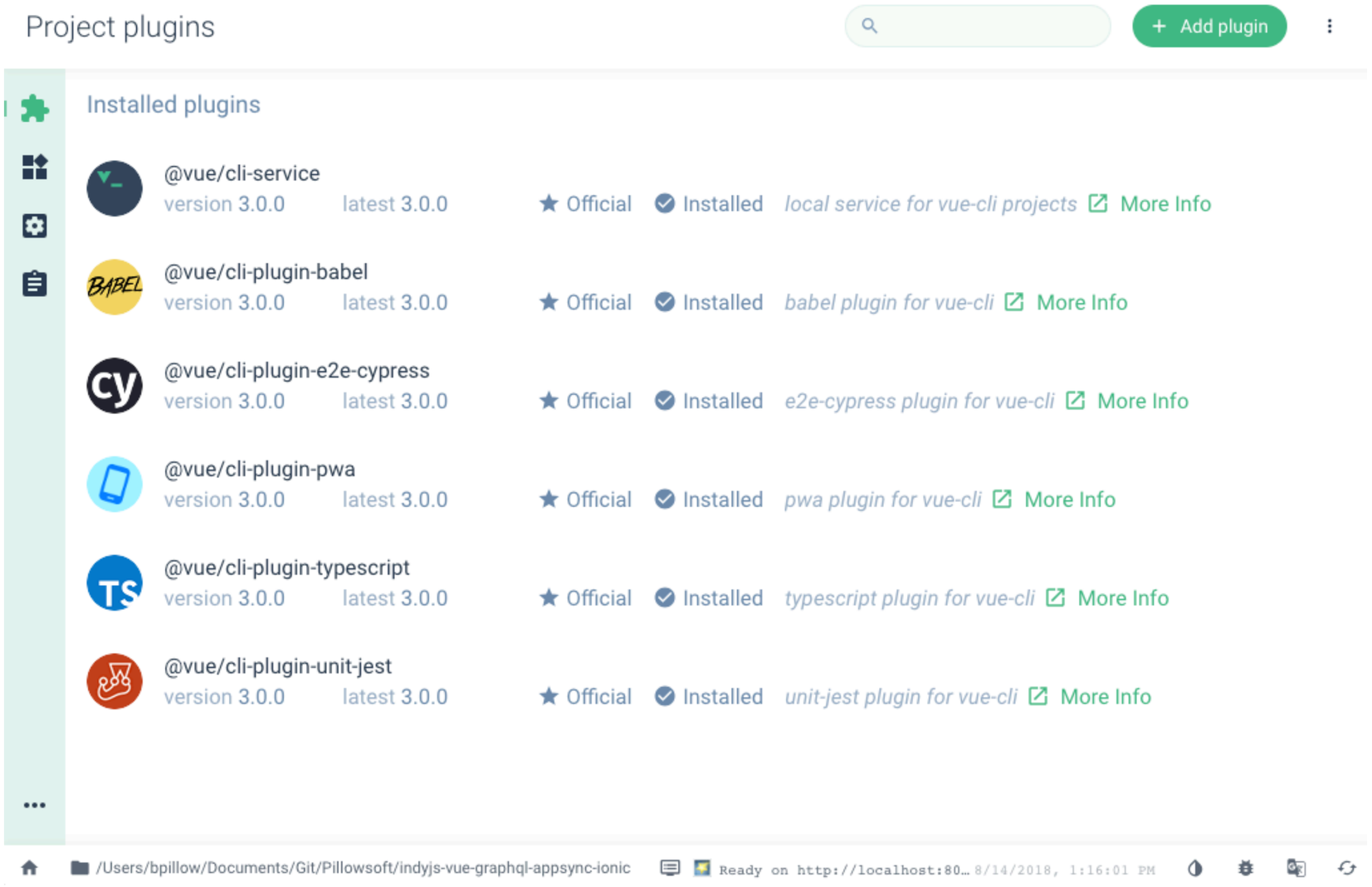

/Users/bpillow/Documents/Git/Pillowsoft/indyjs-vue-graphql-appsync-ionic
indyjs-vue-graphql-appsync-ionic
Log In
Tests
Runs
Settings
Stop
Running Chrome 68
Search...
Run all specs
INTEGRATION TESTS
test.js
Beta Version 3.1.0 | Changelog

# E2E TESTING


indyjs-vue-graphql-appsync-ic
localhost:8080/__/#/tests/__all
Tests
1
--
--
0.59
http://localhost:8080/#/
1000 x 660 (61%)
My First Test
Visits the app root url
TEST
1 VISIT /
2 CONTAINS h1, Welcome to Your Vue.js + TypeScr..
(XHR) GET 200 /sockjs-node/info?t=153426..
Home | About
Welcome to Your Vue.js + TypeScript App
For guide and recipes on how to configure / customize this project,
check out the vue-cli documentation.
Installed CLI Plugins
babel typescript pwa unit-jest e2e-cypress
Essential Links
Core Docs Forum Community Chat Twitter News

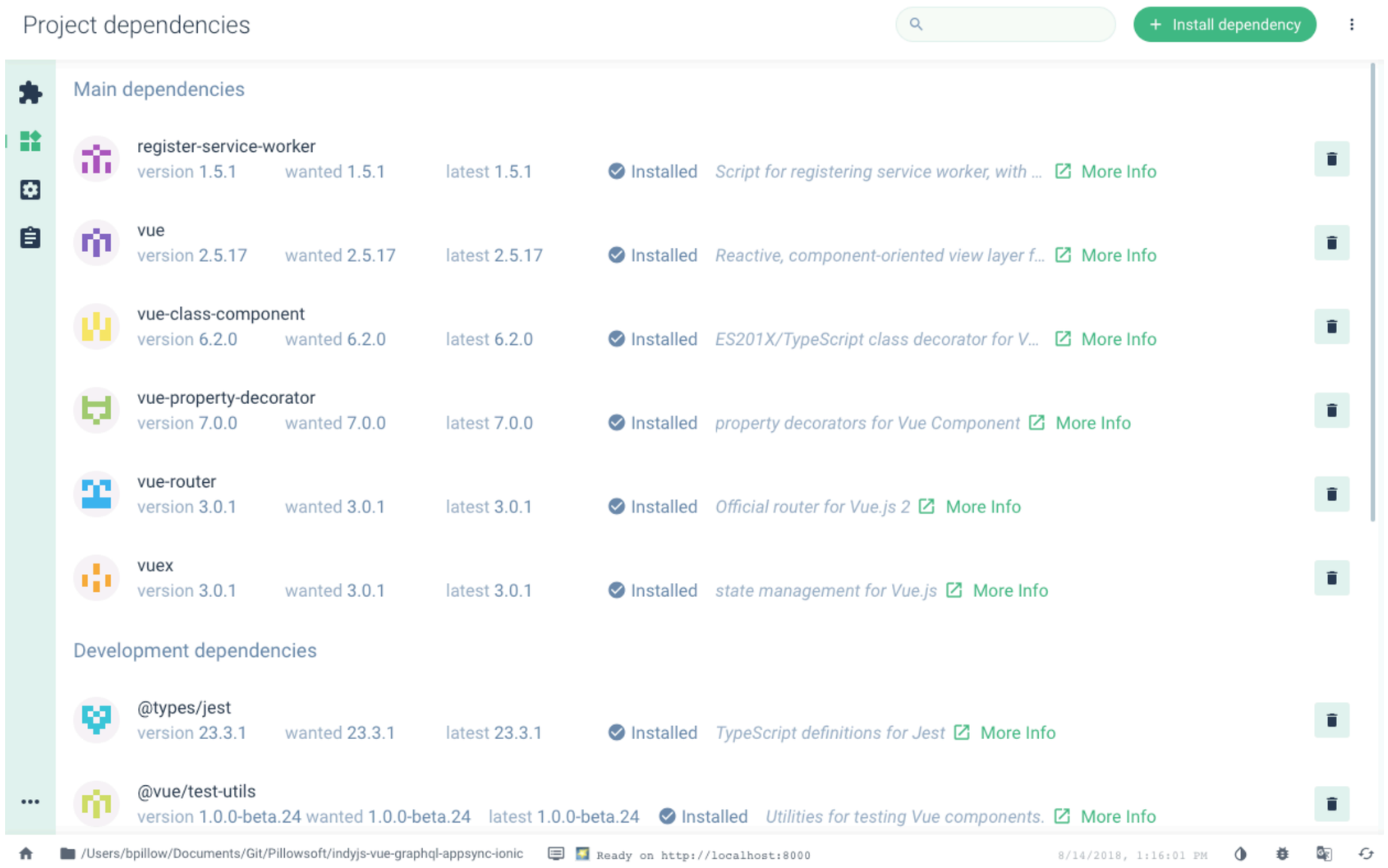
# VUE UI

```
$ vue ui
```


Project plugins
Add plugin
Installed plugins
@vue/cli-service
version 3.0.0 latest 3.0.0 Official Installed local service for vue-cli projects More Info
@vue/cli-plugin-babel
version 3.0.0 latest 3.0.0 Official Installed babel plugin for vue-cli More Info
@vue/cli-plugin-e2e-cypress
version 3.0.0 latest 3.0.0 Official Installed e2e-cypress plugin for vue-cli More Info
@vue/cli-plugin-pwa
version 3.0.0 latest 3.0.0 Official Installed pwa plugin for vue-cli More Info
@vue/cli-plugin-typescript
version 3.0.0 latest 3.0.0 Official Installed typescript plugin for vue-cli More Info
@vue/cli-plugin-unit-jest
version 3.0.0 latest 3.0.0 Official Installed unit-jest plugin for vue-cli More Info
/Users/bpillow/Documents/Git/Pillowsoft/indyjs-vue-graphql-appsync-ionic
Ready on http://localhost:80… 8/14/2018, 1:16:01 PM

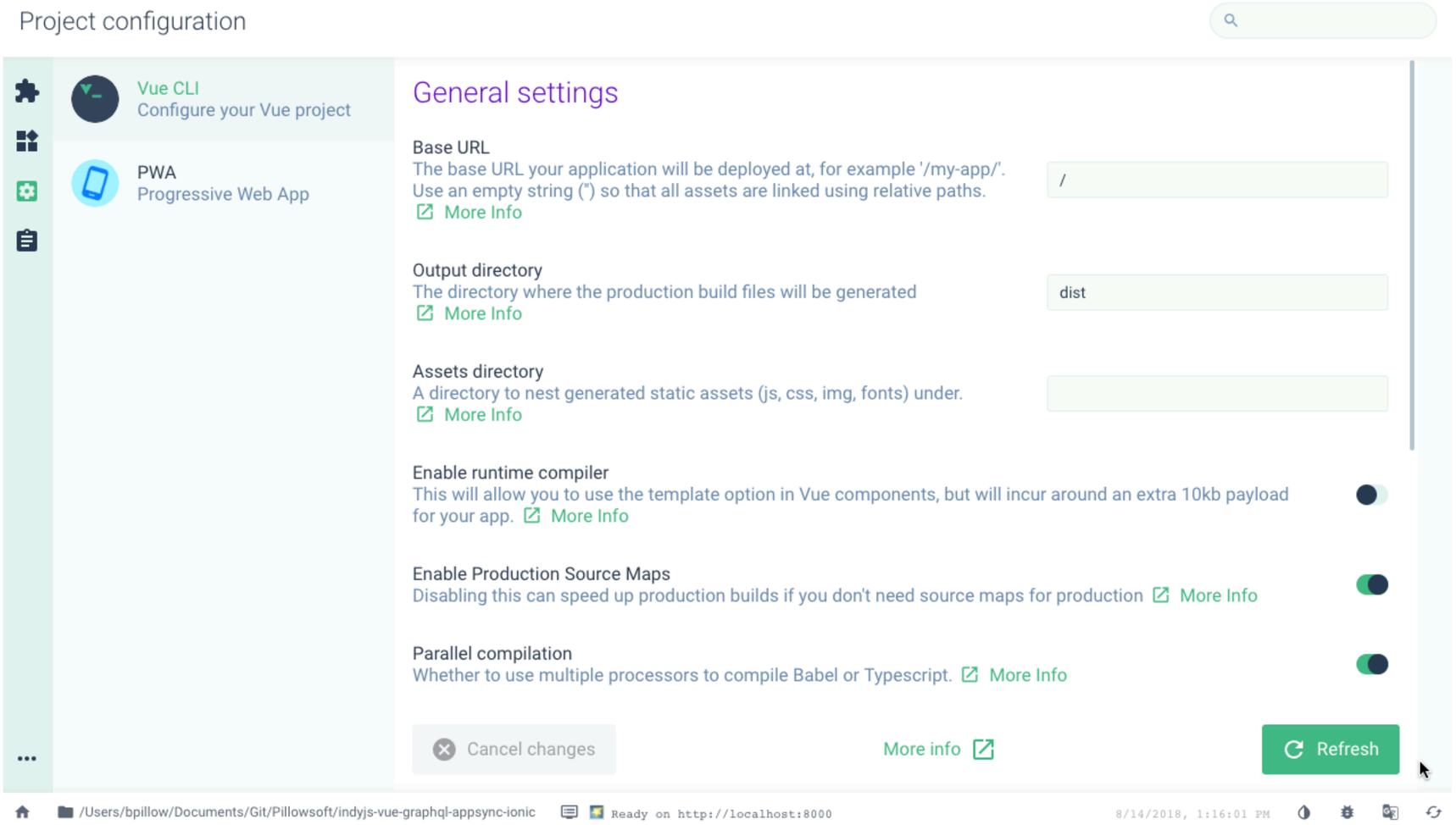
# VUE UI

```
$ vue ui
```


Project dependencies
Install dependency
Main dependencies
register-service-worker
version 1.5.1 wanted 1.5.1 latest 1.5.1 Installed Script for registering service worker, with ... More Info
vue
version 2.5.17 wanted 2.5.17 latest 2.5.17 Installed Reactive, component-oriented view layer f... More Info
vue-class-component
version 6.2.0 wanted 6.2.0 latest 6.2.0 Installed ES201X/TypeScript class decorator for V... More Info
vue-property-decorator
version 7.0.0 wanted 7.0.0 latest 7.0.0 Installed property decorators for Vue Component More Info
vue-router
version 3.0.1 wanted 3.0.1 latest 3.0.1 Installed Official router for Vue.js 2 More Info
vuex
version 3.0.1 wanted 3.0.1 latest 3.0.1 Installed state management for Vue.js More Info
Development dependencies
@types/jest
version 23.3.1 wanted 23.3.1 latest 23.3.1 Installed TypeScript definitions for Jest More Info
@vue/test-utils
version 1.0.0-beta.24 wanted 1.0.0-beta.24 latest 1.0.0-beta.24 Installed Utilities for testing Vue components. More Info
/Users/bpillow/Documents/Git/Pillowsoft/indyjs-vue-graphql-appsync-ionic
Ready on http://localhost:8000
8/14/2018, 1:16:01 PM

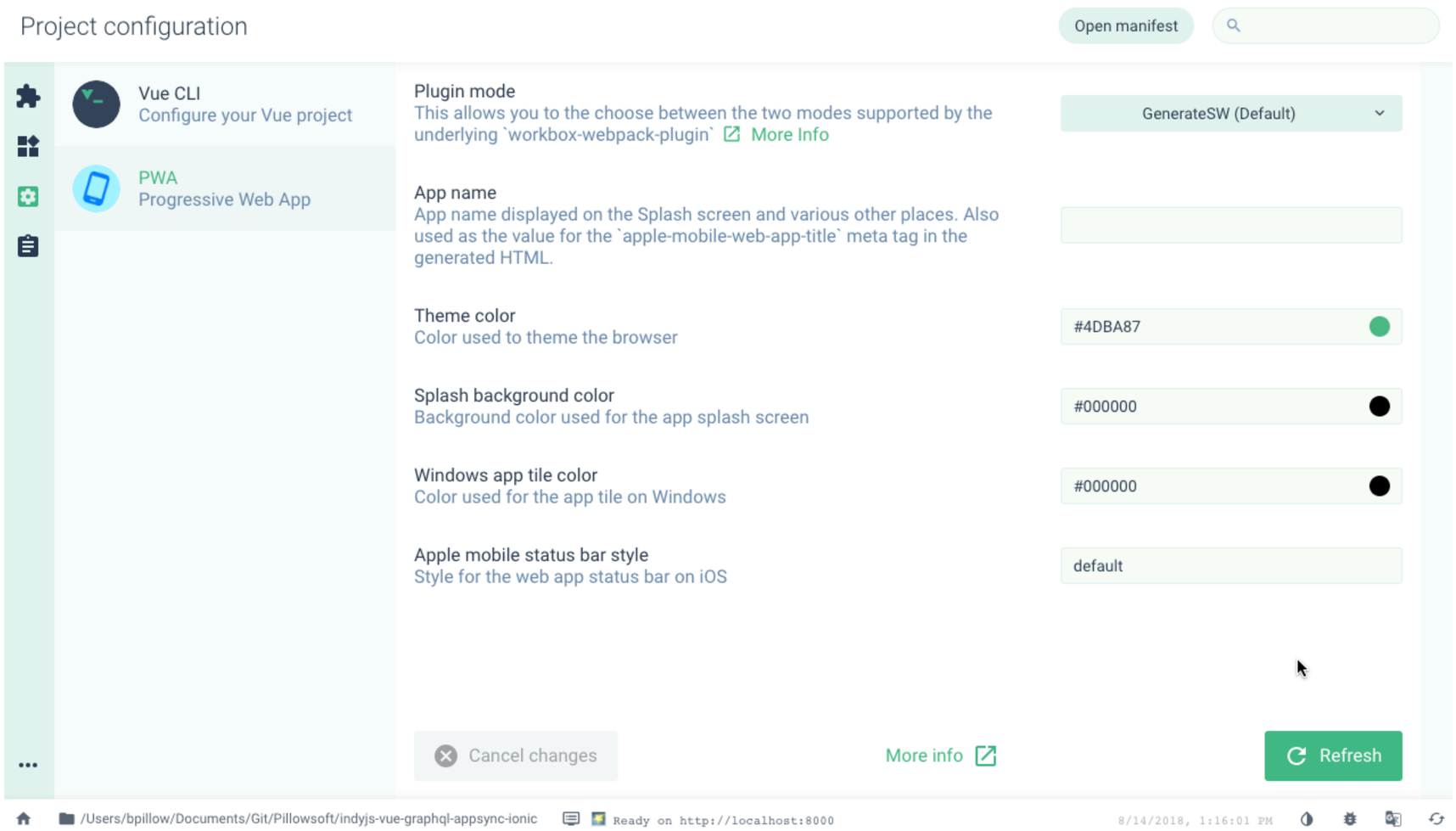
# VUE UI

```
$ vue ui
```


Project configuration
Vue CLI
Configure your Vue project
PWA
Progressive Web App
General settings
Base URL
The base URL your application will be deployed at, for example '/my-app/'.
Use an empty string (') so that all assets are linked using relative paths.
More Info
/
Output directory
The directory where the production build files will be generated
More Info
dist
Assets directory
A directory to nest generated static assets (js, css, img, fonts) under.
More Info
Enable runtime compiler
This will allow you to use the template option in Vue components, but will incur around an extra 10kb payload for your app. More Info
Enable Production Source Maps
Disabling this can speed up production builds if you don't need source maps for production More Info
Parallel compilation
Whether to use multiple processors to compile Babel or Typescript. More Info
Cancel changes
More info
Refresh
/Users/bpillow/Documents/Git/Pillowsoft/indyjs-vue-graphql-appsync-ionic
Ready on http://localhost:8000
8/14/2018, 1:16:01 PM

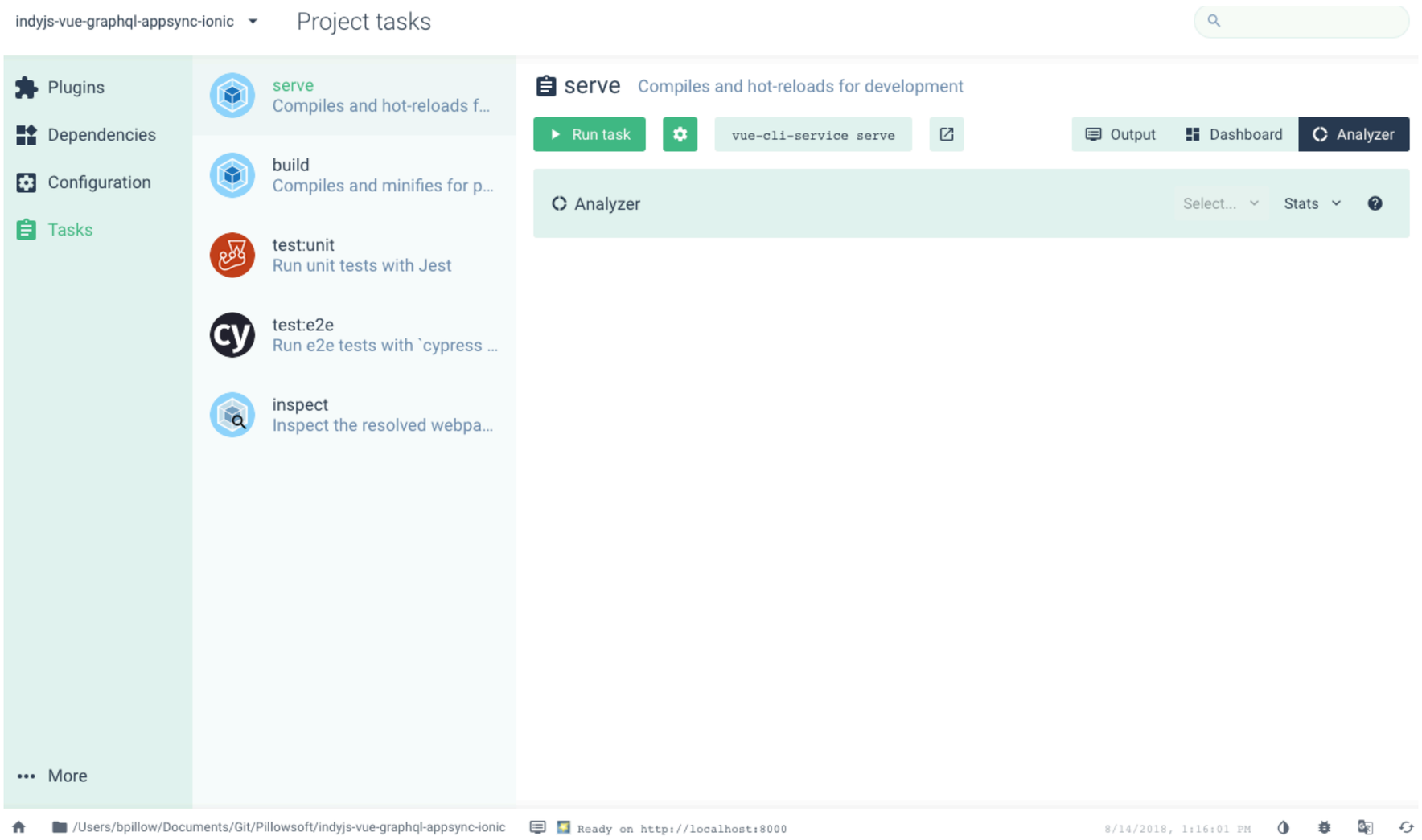
# VUE UI

```
$ vue ui
```


Project configuration
Open manifest
Vue CLI
Configure your Vue project
PWA
Progressive Web App
Plugin mode
This allows you to the choose between the two modes supported by the underlying `workbox-webpack-plugin` More Info
GenerateSW (Default)
App name
App name displayed on the Splash screen and various other places. Also used as the value for the `apple-mobile-web-app-title` meta tag in the generated HTML.
Theme color
Color used to theme the browser
#4DBA87
Splash background color
Background color used for the app splash screen
#000000
Windows app tile color
Color used for the app tile on Windows
#000000
Apple mobile status bar style
Style for the web app status bar on iOS
default
Cancel changes
More info
Refresh
/Users/bpillow/Documents/Git/Pillowsoft/indyjs-vue-graphql-appsync-ionic
Ready on http://localhost:8000
8/14/2018, 1:16:01 PM

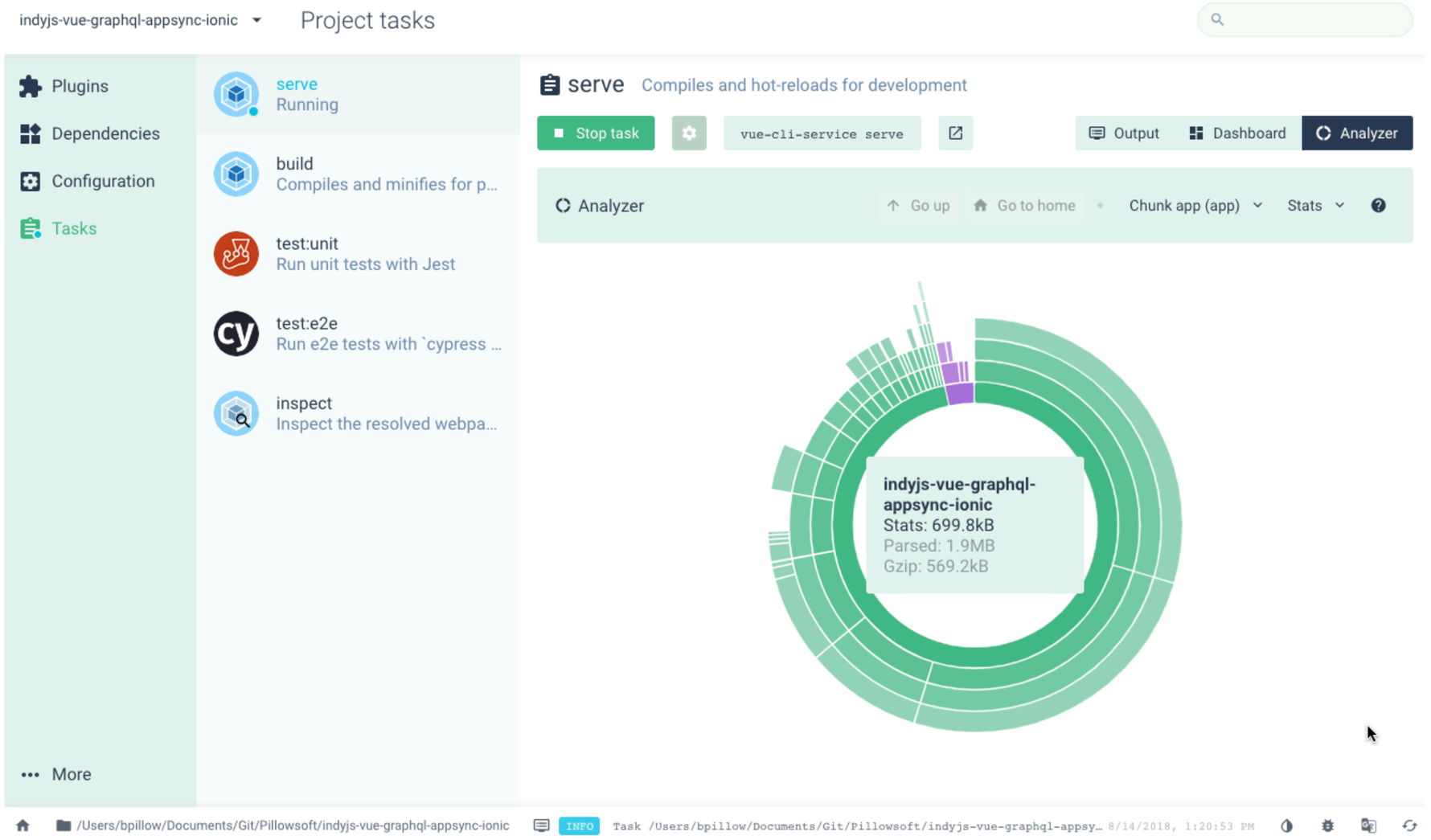
# VUE UI

```
$ vue ui
```

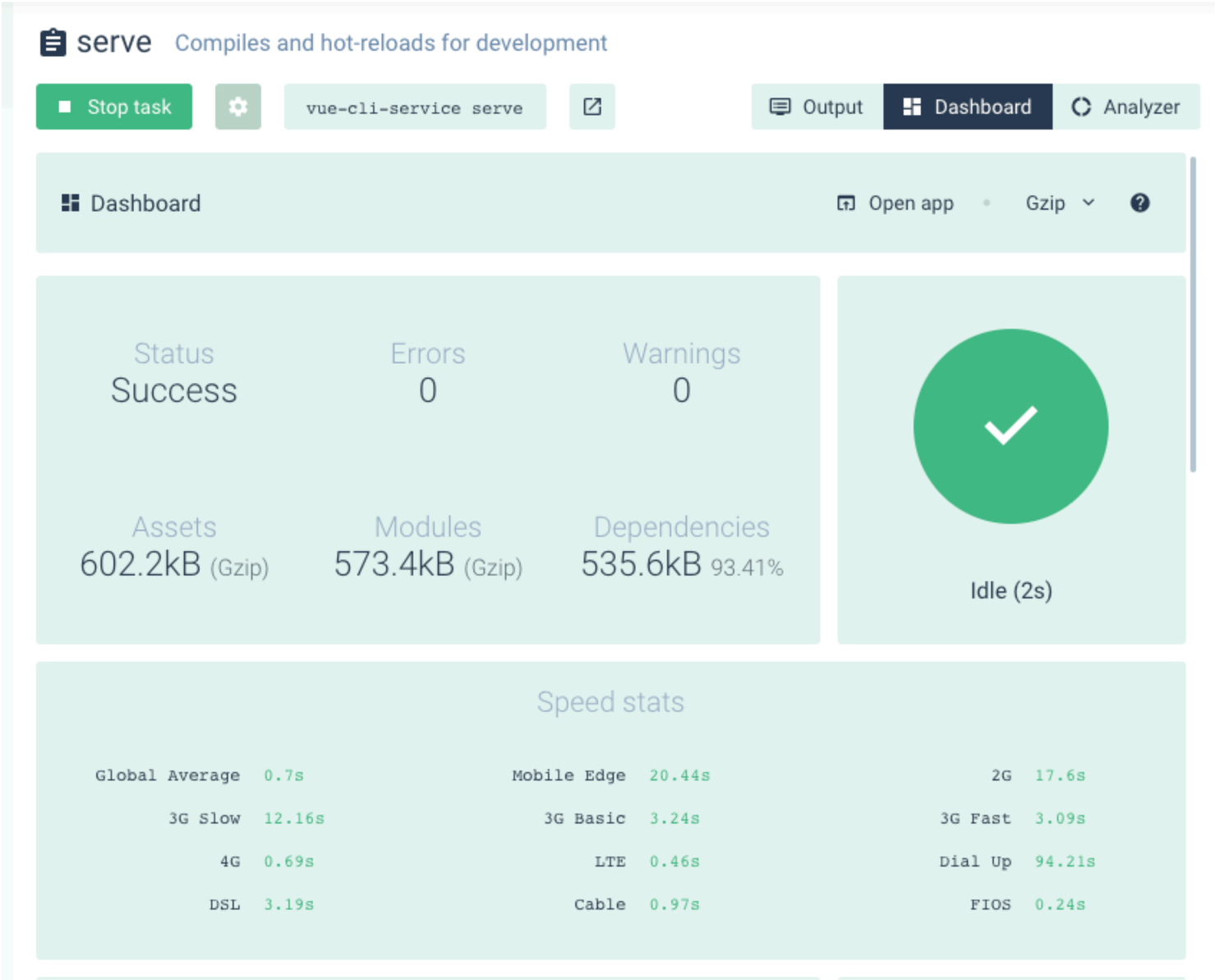

indyjs-vue-graphql-appsync-ionic
Project tasks
Plugins
Dependencies
Configuration
Tasks
More
serve
Compiles and hot-reloads f...
build
Compiles and minifies for p...
test:unit
Run unit tests with Jest
test:e2e
Run e2e tests with `cypress ...
inspect
Inspect the resolved webpa...
serve
Compiles and hot-reloads for development
Run task
vue-cli-service serve
Output
Dashboard
Analyzer
Analyzer
Select...
Stats
/Users/bpillow/Documents/Git/Pillowsoft/indyjs-vue-graphql-appsync-ionic
Ready on http://localhost:8000
8/14/2018, 1:16:01 PM

# VUE UI

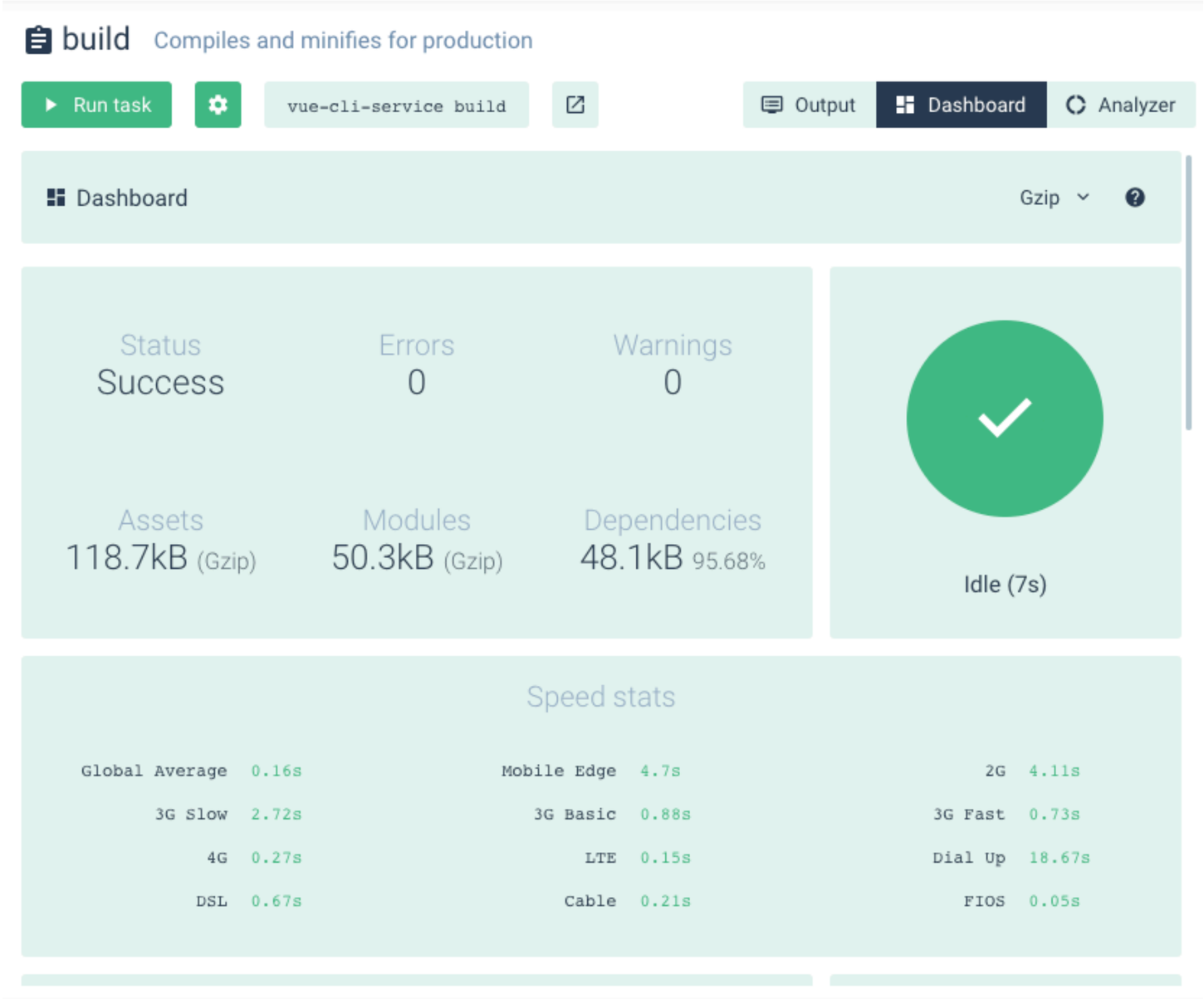
```
$ vue ui
```


indyjs-vue-graphql-appsync-ionic
Project tasks
Plugins
Dependencies
Configuration
Tasks
More
serve
Running
build
Compiles and minifies for p...
test:unit
Run unit tests with Jest
test:e2e
Run e2e tests with `cypress ...
inspect
Inspect the resolved webpa...
serve
Compiles and hot-reloads for development
Stop task
vue-cli-service serve
Output
Dashboard
Analyzer
Analyzer
Go up
Go to home
Chunk app (app)
Stats
indyjs-vue-graphql-appsync-ionic
Stats: 699.8kB
Parsed: 1.9MB
Gzip: 569.2kB
/Users/bpillow/Documents/Git/Pillowsoft/indyjs-vue-graphql-appsync-ionic
INFO
Task /Users/bpillow/Documents/Git/Pillowsoft/indyjs-vue-graphql-appsy... 8/14/2018, 1:20:53 PM

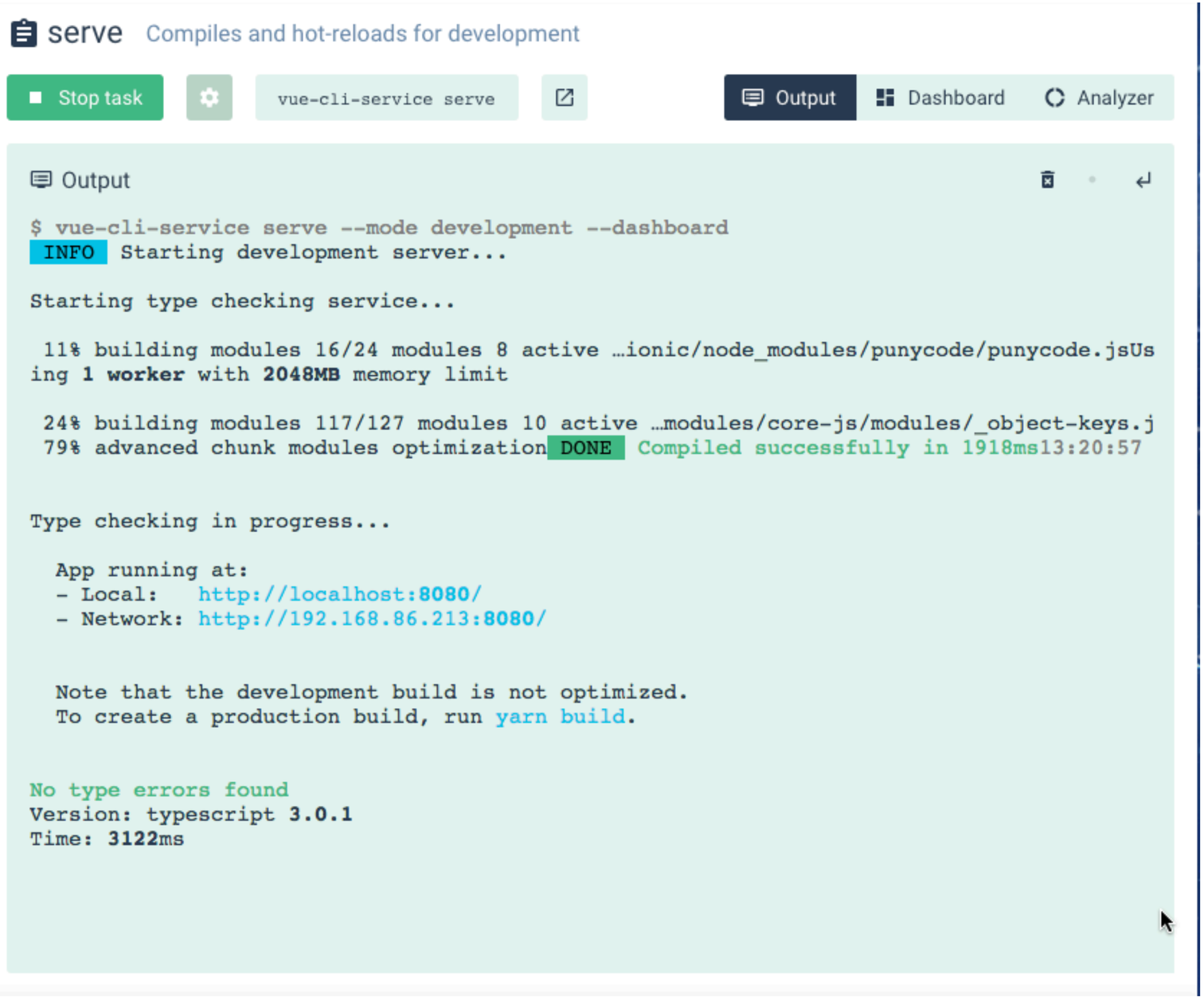
# VUE UI


serve
Compiles and hot-reloads for development
Stop task
vue-cli-service serve
Output
Dashboard
Analyzer
Dashboard
Open app
Gzip
Status
Success
Errors
0
Warnings
0
Assets
602.2kB (Gzip)
Modules
573.4kB (Gzip)
Dependencies
535.6kB 93.41%
Idle (2s)
Speed stats
Global Average 0.7s
3G Slow 12.16s
4G 0.69s
DSL 3.19s
Mobile Edge 20.44s
3G Basic 3.24s
LTE 0.46s
Cable 0.97s
2G 17.6s
3G Fast 3.09s
Dial Up 94.21s
FIOS 0.24s

# VUE UI


build
Compiles and minifies for production
Run task
vue-cli-service build
Output
Dashboard
Analyzer
Dashboard
Gzip
Status
Success
Errors
0
Warnings
0
Assets
118.7kB (Gzip)
Modules
50.3kB (Gzip)
Dependencies
48.1kB 95.68%
Idle (7s)
Speed stats
Global Average 0.16s
3G Slow 2.72s
4G 0.27s
DSL 0.67s
Mobile Edge 4.7s
3G Basic 0.88s
LTE 0.15s
Cable 0.21s
2G 4.11s
3G Fast 0.73s
Dial Up 18.67s
FIOS 0.05s

# VUE UI

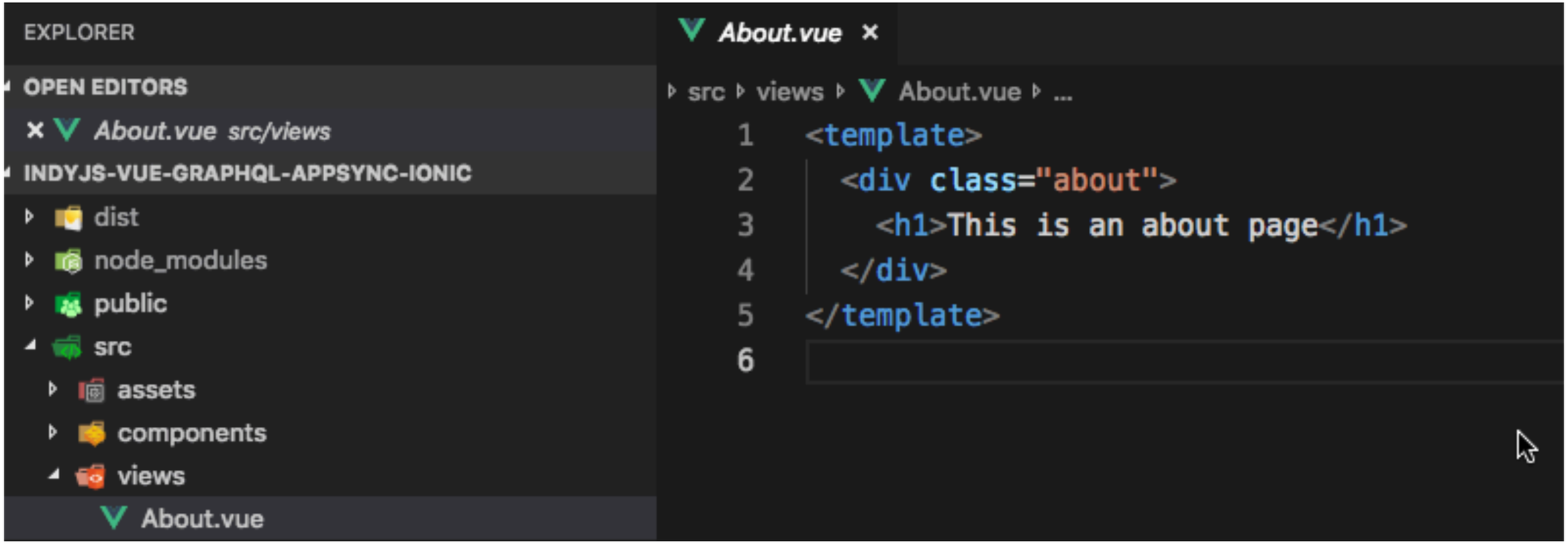

serve
Compiles and hot-reloads for development
Stop task
vue-cli-service serve
Output
Dashboard
Analyzer
Output
$ vue-cli-service serve --mode development --dashboard
INFO Starting development server...
Starting type checking service...
11% building modules 16/24 modules 8 active …ionic/node_modules/punycode/punycode.jsUs
ing 1 worker with 2048MB memory limit
24% building modules 117/127 modules 10 active …modules/core-js/modules/_object-keys.j
79% advanced chunk modules optimization DONE Compiled successfully in 1918ms13:20:57
Type checking in progress...
App running at:
- Local: http://localhost:8080/
- Network: http://192.168.86.213:8080/
Note that the development build is not optimized.
To create a production build, run yarn build.
No type errors found
Version: typescript 3.0.1
Time: 3122ms

# QUICK INTRO TO VUE

# FEATURES

- Reactive Interfaces
- Declarative Rendering
- Data Binding
- Directives
- Template Logic
- Components
- Event Handling
- Computed Properties
- CSS Transitions and Animations
- Filters

# SIMPLE COMPONENT – "ABOUT" FROM PROJECT



# SIMPLE SINGLE FILE COMPONENT

Hello.vue

```vue
<template>
  <p>{{ greeting }} World!</p>
</template>

<script>
module.exports = {
  data: function () {
    return {
      greeting: 'Hello'
    }
  }
}
</script>

<style scoped>
p {
  font-size: 2em;
  text-align: center;
}
</style>
```

Line 21, Column 1 Spaces: 2 Vue Component

Hello.vue

```vue
<template lang="jade">
div
  p {{ greeting }} World!
  other-component
</template>

<script>
import OtherComponent from './OtherComponent.vue'

export default {
  data () {
    return {
      greeting: 'Hello'
    }
  },
  components: {
    OtherComponent
  }
}
</script>

<style lang="stylus" scoped>
p
  font-size 2em
  text-align center
</style>
```

Line 27, Column 1 Spaces: 2 Vue Component

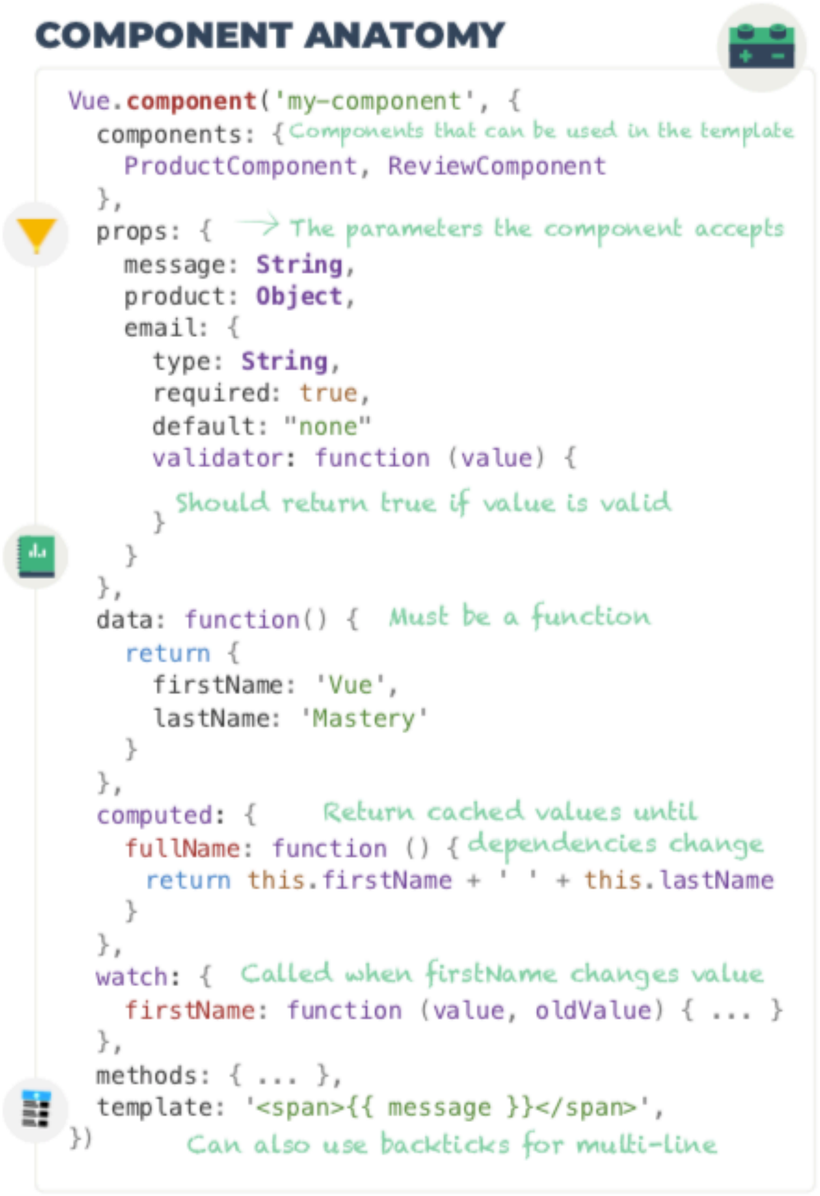
# TS SINGLE FILE COMPONENT – "HOME" FROM PROJECT

Home.vue

src › views › Home.vue › ...

```vue
<template>
  <div class="home">
    <img alt="Vue logo" src="../assets/logo.png">
    <HelloWorld msg="Welcome to Your Vue.js + TypeScript App"/>
  </div>
</template>

<script lang="ts">
import { Component, Vue } from 'vue-property-decorator';
import HelloWorld from '@/components/HelloWorld.vue'; // @ is an alias to /src

@Component({
  components: {
    HelloWorld,
  },
})
export default class Home extends Vue {}
</script>
```

## COMPONENT ANATOMY



```
Vue.component('my-component', {
  components: {Components that can be used in the template
    ProductComponent, ReviewComponent
  },
  props: {  → The parameters the component accepts
    message: String,
    product: Object,
    email: {
      type: String,
      required: true,
      default: "none"
      validator: function (value) {
        Should return true if value is valid
      }
    }
  },
  data: function() {  Must be a function
    return {
      firstName: 'Vue',
      lastName: 'Mastery'
    }
  },
  computed: {    Return cached values until
    fullName: function () { dependencies change
      return this.firstName + ' ' + this.lastName
    }
  },
  watch: {  Called when firstName changes value
    firstName: function (value, oldValue) { ... }
  },
  methods: { ... },
  template: '<span>{{ message }}</span>',
})      Can also use backticks for multi-line
```

Next several snippets also from Vue Master Cheat Sheet, a must have!

https://www.vuemastery.com/vue-cheat-sheet/

More cheat sheets:

https://gist.github.com/LeCoupa/da7d86e80d3b61a61b0dd251b390744f

https://github.com/dekadentno/vue-cheat-sheet

# BASICS

## EXPRESSIONS

```
<div id="app">
  <p>I have a {{ product }}</p>
  <p>{{ product + 's' }}</p>
  <p>{{ isWorking ? 'YES' : 'NO' }}</p>
  <p>{{ product.getSalePrice() }}</p>
</div>
```

## BINDING

```
<a v-bind:href="url">...</a>
```

shorthand →

```
<a :href="url">...</a>
```

True or false will add or remove attribute:

```
<button :disabled="isButtonDisabled">...
```

## DIRECTIVES

Element inserted/removed based on truthiness:

```
<p v-if="inStock">{{ product }}</p>
```

```
<p v-else-if="onSale">...</p>
<p v-else>...</p>
```

Toggles the display: none CSS property:

```
<p v-show="showProductDetails">...</p>
```

Two-way data binding:

```
<input v-model="firstName" >
```

| | |
|---|---|
| `v-model.lazy="..."` | Syncs input after change event |
| `v-model.number="..."` | Always returns a number |
| `v-model.trim="..."` | Strips whitespace |

## LIST RENDERING

```
<li v-for="item in items" :key="item.id">
  {{ item }}
</li>
```

key always recommended

To access the position in the array:

```
<li v-for="(item, index) in items">...
```

To iterate through objects:

```
<li v-for="(value, key) in object">...
```

Using v-for with a component:

```
<cart-product v-for="item in products"
              :product="item" :key="item.id">
```

## ACTIONS / EVENTS

Calls addToCart method on component:

```
<button v-on:click="addToCart">...
```

shorthand

```
<button @click="addToCart">...
```

Arguments can be passed:

```
<button @click="addToCart(product)">...
```

To prevent default behavior (e.g. page reload):

```
<form @submit.prevent="addProduct">...
```

Only trigger once:

```
<img @mouseover.once="showImage">...
```

| | |
|---|---|
| `.stop` | Stop all event propagation |
| `.self` | Only trigger if event.target is element itself |

Keyboard entry example:

```
<input @keyup.enter="submit">
```

Call onCopy when control-c is pressed:

```
<input @keyup.ctrl.c="onCopy">
```

## LIFECYCLE HOOKS

| | |
|---|---|
| `beforeCreate` | `beforeUpdate` |
| `created` | `updated` |
| `beforeMount` | `beforeDestroy` |
| `mounted` | `destroyed` |

## USING A SINGLE SLOT

Component template:

```
<div>
  <h2>I'm a title</h2>
    <slot>
      Only displayed if no slot content
    </slot>
</div>
```

Use of component with data for slot:

```
<my-component>
  <p>This will go in the slot</p>
</my-component>
```

## MULTIPLE SLOTS

Component template:

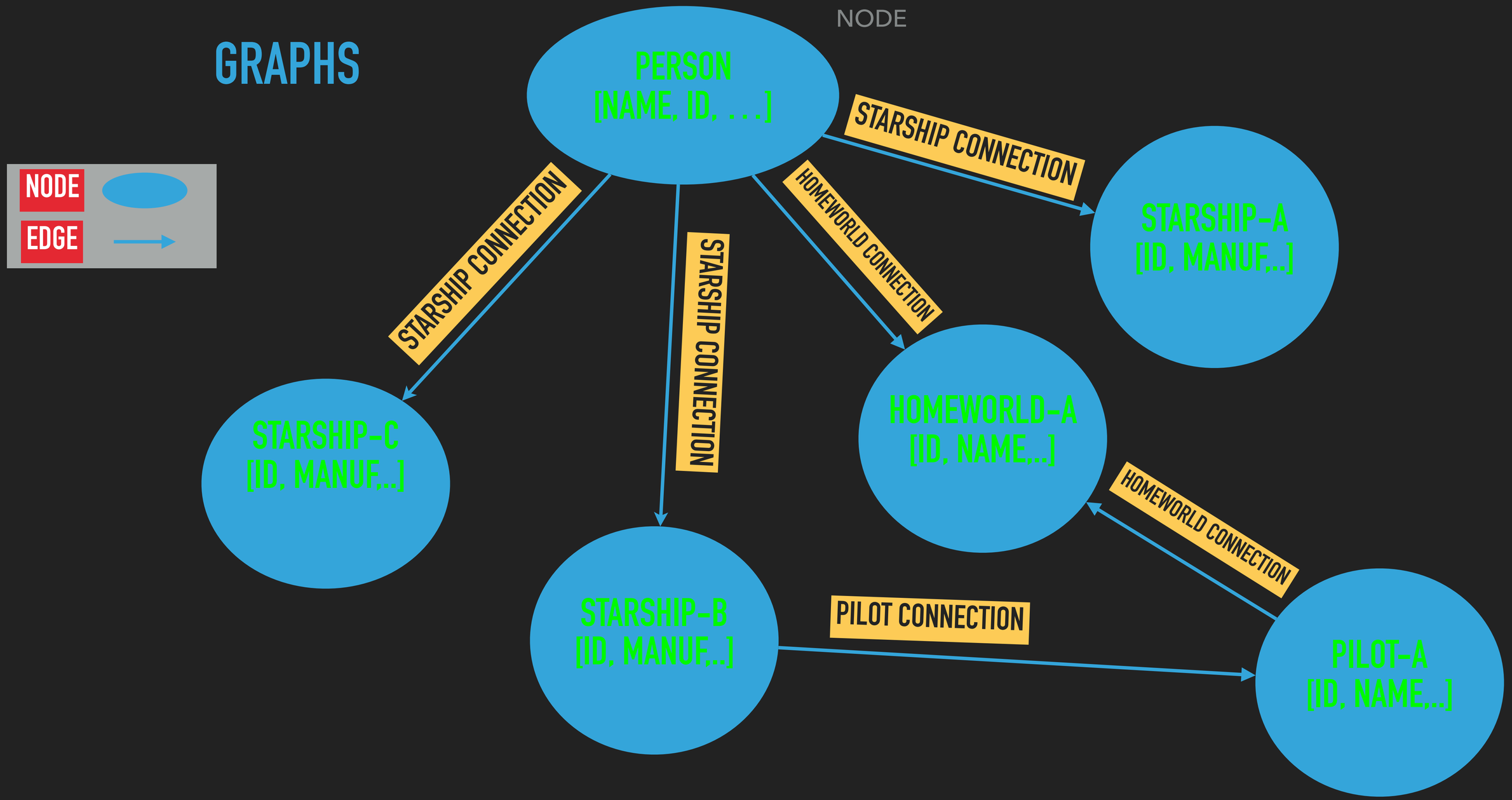
```
<div class="container">
  <header>
    <slot name="header"></slot>
  </header>
  <main>
    <slot>Default content</slot>
  </main>
  <footer>
    <slot name="footer"></slot>
  </footer>
</div>
```

# QUICK INTRO TO GRAPHQL

# VALUE PROPS

- Fetch only what you need
- Your data is a big graph and can pull from multiple sources
- Use GraphQL for local-state and obviate the need for Vuex, via apollo-link-state

# GRAPHS


NODE
EDGE
NODE
PERSON
[NAME, ID, . . .]
STARSHIP CONNECTION
STARSHIP CONNECTION
STARSHIP CONNECTION
HOMEWORLD CONNECTION
STARSHIP-A
[ID, MANUF,..]
STARSHIP-C
[ID, MANUF,..]
HOMEWORLD-A
[ID, NAME,..]
HOMEWORLD CONNECTION
STARSHIP-B
[ID, MANUF,..]
PILOT CONNECTION
PILOT-A
[ID, NAME,..]

# BASIC QUERY

```
{
  person(personID: 4) {
    name
  }
}
```

```
{
  "data": {
    "person": {
      "name": "Darth Vader"
    }
  }
}
```

# NESTED FIELDS

```
{
  person(personID: 4) {
    name
    gender
    homeworld {
      name
    }
  }
}
```

```
{
  "data": {
    "person": {
      "name": "Darth Vader",
      "gender": "male",
      "homeworld": {
        "name": "Tatooine"
      }
    }
  }
}
```

# ARGUMENTS

```
{
  allStarships(first: 7) {
    edges {
      node {
        id
        name
        model
        costInCredits
        pilotConnection {
          edges {
            node {
              name
              homeworld {
                name
              }
            }
          }
        }
      }
    }
  }
}
```


```
{
  "data": {
    "allStarships": {
      "edges": [
        {
          "node": {
            "id": "c3RhcnNoaXBzOjI=",
            "name": "CR90 corvette",
            "model": "CR90 corvette",
            "costInCredits": 3500000,
            "pilotConnection": {
              "edges": []
            }
          }
        },
        {
          "node": {
            "id": "c3RhcnNoaXBzOjM=",
            "name": "Star Destroyer",
            "model": "Imperial I-class Star Destroyer",
            "costInCredits": 150000000,
            "pilotConnection": {
              "edges": []
            }
          }
        },
        {
          "node": {
```

# Schema

```
input CreatePictureInput {
  id: ID!
  name: String
  owner: String
  createdAt: String
}

input DeletePictureInput {
  id: ID!
}

type Dog {
  name: String
  breed: String
}

type Mutation {
  addPicture(name: String, visibility: Visibility, file: S3ObjectInput): Picture
  createPicture(input: CreatePictureInput!): Picture
  updatePicture(input: UpdatePictureInput!): Picture
  deletePicture(input: DeletePictureInput!): Picture
}

type Picture {
  id: ID!
  name: String
  visibility: Visibility
  owner: String
  file: S3Object
  createdAt: String
}

type PictureConnection {
  items: [Picture]
  nextToken: String
}
```

# Schema

```
type Query {
  empty: Boolean
  getPicture(id: ID!): Picture
  listPictures(first: Int, after: String): PictureConnection
  queryPicturesByOwnerIndex(owner: String!, first: Int, after: String): PictureConnection
  yo: String
}

type S3Object {
  bucket: String!
  region: String!
  key: String!
}

input S3ObjectInput {
  bucket: String!
  region: String!
  localUri: String
  visibility: Visibility
  key: String
  mimeType: String
}
```

# Schema

```
type Subscription {
  onCreatePicture(
    id: ID,
    name: String,
    owner: String,
    createdAt: String
  ): Picture
    @aws_subscribe(mutations: ["createPicture"])
  onUpdatePicture(
    id: ID,
    name: String,
    owner: String,
    createdAt: String
  ): Picture
    @aws_subscribe(mutations: ["updatePicture"])
  onDeletePicture(
    id: ID,
    name: String,
    owner: String,
    createdAt: String
  ): Picture
    @aws_subscribe(mutations: ["deletePicture"])
}

input UpdatePictureInput {
  id: ID!
  name: String
  owner: String
  createdAt: String
}

enum Visibility {
  public
  private
}

schema {
  query: Query
  mutation: Mutation
}
```

# VUE-GRAPHQL

# INSTALL

- https://akryum.github.io/vue-apollo/

```
npm install --save vue-apollo graphql apollo-client apollo-link apollo-link-http apollo-cache-inmemory graphql-tag
```

# ADD CLIENT INITIALIZATION

## Apollo client

In your app, create an `ApolloClient` instance and install the `VueApollo` plugin:

```js
import Vue from 'vue'
import { ApolloClient } from 'apollo-client'
import { HttpLink } from 'apollo-link-http'
import { InMemoryCache } from 'apollo-cache-inmemory'
import VueApollo from 'vue-apollo'

const httpLink = new HttpLink({
  // You should use an absolute URL here
  uri: 'http://localhost:3020/graphql',
})

// Create the apollo client
const apolloClient = new ApolloClient({
  link: httpLink,
  cache: new InMemoryCache(),
  connectToDevTools: true,
})

// Install the vue plugin
Vue.use(VueApollo)
```

# ADD PROVIDER INITIALIZATION

## Apollo provider

The provider holds the apollo client instances that can then be used by all the child components. Inject it into your components with `provide` :

```js
const apolloProvider = new VueApollo({
  defaultClient: apolloClient,
})

new Vue({
  el: '#app',
  provide: apolloProvider.provide(),
  render: h => h(App),
})
```

# DATA RESULTS

## Simple query

Use `gql` to write your GraphQL queries:

```js
import gql from 'graphql-tag'
```

Put the gql query directly as the value:

```js
apollo: {
  // Simple query that will update the 'hello' vue property
  hello: gql`{hello}`,
},
```

You can then access the query with `this.$apollo.queries.<name>`.

You can initialize the property in your vue component's `data` hook:

```js
data () {
  return {
    // Initialize your apollo data
    hello: '',
  },
},
```

You can then use your property as usual in your vue component:

```vue
<template>
  <div class="apollo">
    <h3>Hello</h3>
    <p>
      {{hello}}
    </p>
  </div>
</template>
```

For example, you could add the `fetchPolicy` apollo option like this:

```js
apollo: {
  // Query with parameters
  ping: {
    query: gql`query PingMessage($message: String!) {
      ping(message: $message)
    }`,
    variables: {
      message: 'Meow'
    },
    // Additional options here
    fetchPolicy: 'cache-and-network',
  },
},
```

# IONIC-VUE

# IONIC FEATURES


Ref: https://www.angularminds.com/blog/article/whats-new-features-introduced-in-ionic-4.html

# IONIC 4 – WEB STANDARDS

For those new to Web Components, the term refers to a collection of Web APIs with broad support on modern mobile and desktop browsers, such as Custom Elements and Shadow DOM. While Web Components have been hyped for a few years now, we feel browser and developer support has finally hit a critical mass, making them ready for primetime. But we're not the only ones – many traditional frameworks and UI libraries have started adopting them as well (ex: Angular Elements).

Ref: https://beta.ionicframework.com/docs/

# IONIC FEATURES – WEB/NATIVE

- **Shadow DOM:** There's multiple wins here, but one of this is that by embracing native browser APIs and web-standards, Ionic is able to reduce the amount of client-side code required to ship to all of your users. Additionally, shadow dom helps consuming Ionic components from any web app even easier by encapsulating its styles.

- **CSS Variables:** CSS Variables are at the core of how Ionic's theming works. You can modify the overall look and feel of your app by just changing a few variables, all without build tools. Expect a lot more content soon covering "why" CSS Variables have proven to be awesome for Ionic and developers.

- **Ionicons 4.0:** Now available and distributed as web components with drastically reduced sizes, and brand new icon forms reflecting the latest iOS and Material Design styles. Learn more

- **Native API:** Ionic Native 5.0 Beta has been upgraded to also be framework independent! You can now use the wrappers outside of Angular as simple classes while still offering Angular providers that work with dependency injection. Check out the new Native API Docs

Ref: https://beta.ionicframework.com/docs/

# IONIC FEATURES - CAPACITOR

Capacitor is a cross-platform app runtime that makes it easy to build web apps that run natively on iOS, Android, Electron, *and* the web. We call these apps "Native Progressive Web Apps" and they represent the next evolution beyond Hybrid apps.

Capacitor provides a consistent, web-focused set of APIs that enable an app to stay as close to web-standards as possible, while accessing rich native device features on platforms that support them. Adding native functionality is easy with a simple Plugin API for Swift on iOS, Java on Android, and JavaScript for the web.

Capacitor is a spiritual successor to Apache Cordova and Adobe PhoneGap, with inspiration from other popular cross-platform tools like React Native and Turbolinks, but focused entirely on enabling modern web apps to run on all major platforms with ease. Capacitor has backwards-compatible support for many existing Cordova plugins.

# IONIC VUE

### Cross Platform

Build web apps that run equally well on iOS, Android, Electron, and as Progressive Web Apps

### Native Access

Access the full Native SDK on each platform, and easily deploy to App Stores (and the web!)

### Use with Ionic

Capacitor provides native functionality for web apps, and is optimized for Ionic Framework

### Web Native

Build apps with standardized web technologies that will work for decades, and easily reach users on the app stores *and* the mobile web.

### Extensible

Easily add custom native functionality with a simple Plugin API, or use existing Cordova plugins with our compatibility layer.

### Open Source

Capacitor is completely open source (MIT) and maintained by **Ionic** and its community.

Home
Introduction
Installation
Building
Components
Layout
Theming
Publishing
FAQ
Dev Resources

API Reference
CLI Reference
Native APIs

v4 Migration Guide

## Card

API

Card components are a great way to display important pieces of content, and are quickly emerging as a core design pattern for apps. Cards can contain images, buttons, text and more. Cards can also be used standalone, in a list or in a grid.

MARKUP PREVIEW

HTML

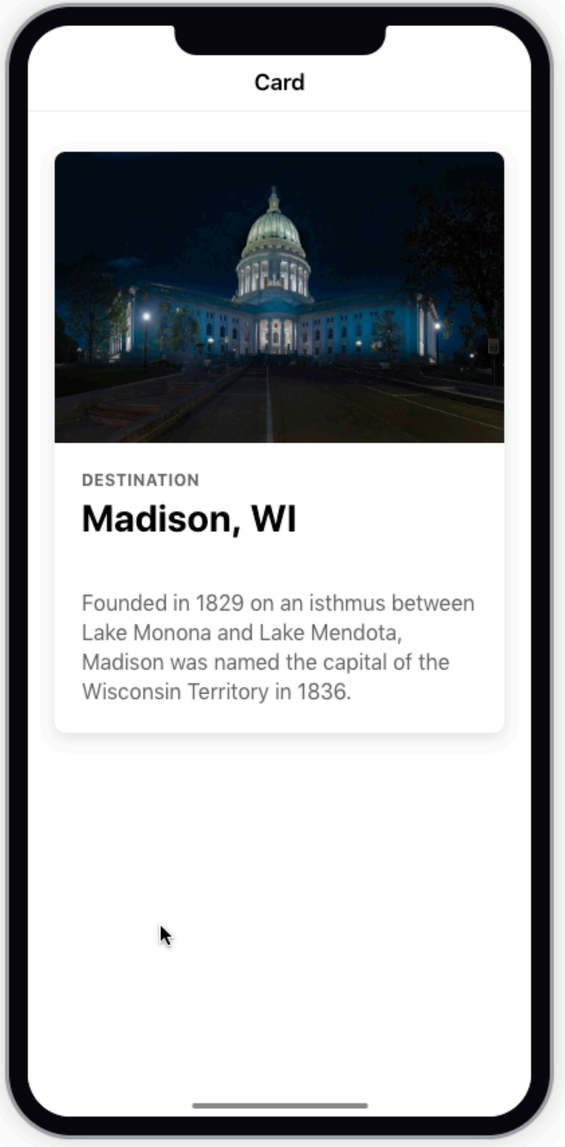
```
<ion-card>
  <ion-img src="/assets/myImg.png"></ion-img>

  <ion-card-content>
    <ion-card-title>Hello World</ion-card-title>

    <p>The content for this card</p>
  </ion-card-content>
</ion-card>
```

## Checkbox

API

A checkbox can be used to let the user know they need to make a binary decision. It provides a clear visual of either a true or false choice. ion-checkbox should always be used with ion-item. If you prefer the look of radio inputs, you can also use `ion-radio` or if you prefer toggles you can use `ion-toggle` .

MARKUP PREVIEW



# USEFUL

- https://blog.ionicframework.com/adding-aws-amplify-to-an-ionic-4-app/
- https://beta.ionicframework.com/docs/components
- Use the Ionic Router (smooth native-is page transitions) via: https://github.com/ModusCreateOrg/ionic-vue

# AWS APPSYNC
# &
# VUE

# BENEFITS

- GraphQL as a service
- Fully supports GraphQL subscriptions for real-time updates
- Server-less, so you pay based on requests/data/storage
- Automatically manages offline data/persistence and conflict resolution
- Authorization/Authentication vis Amazon Congnito, or role your own
- Multiple data resolver types supported: DynamoDB, Lambda, Elastic Search and “Local” (for general pub-sub)
- Supports Vue, React, React Native and vanilla JS
- GraphQL operations can be simple lookups, complex queries & mappings, full text searches, fuzzy/keyword searches or geo lookups

# RECENT NEW FEATURES

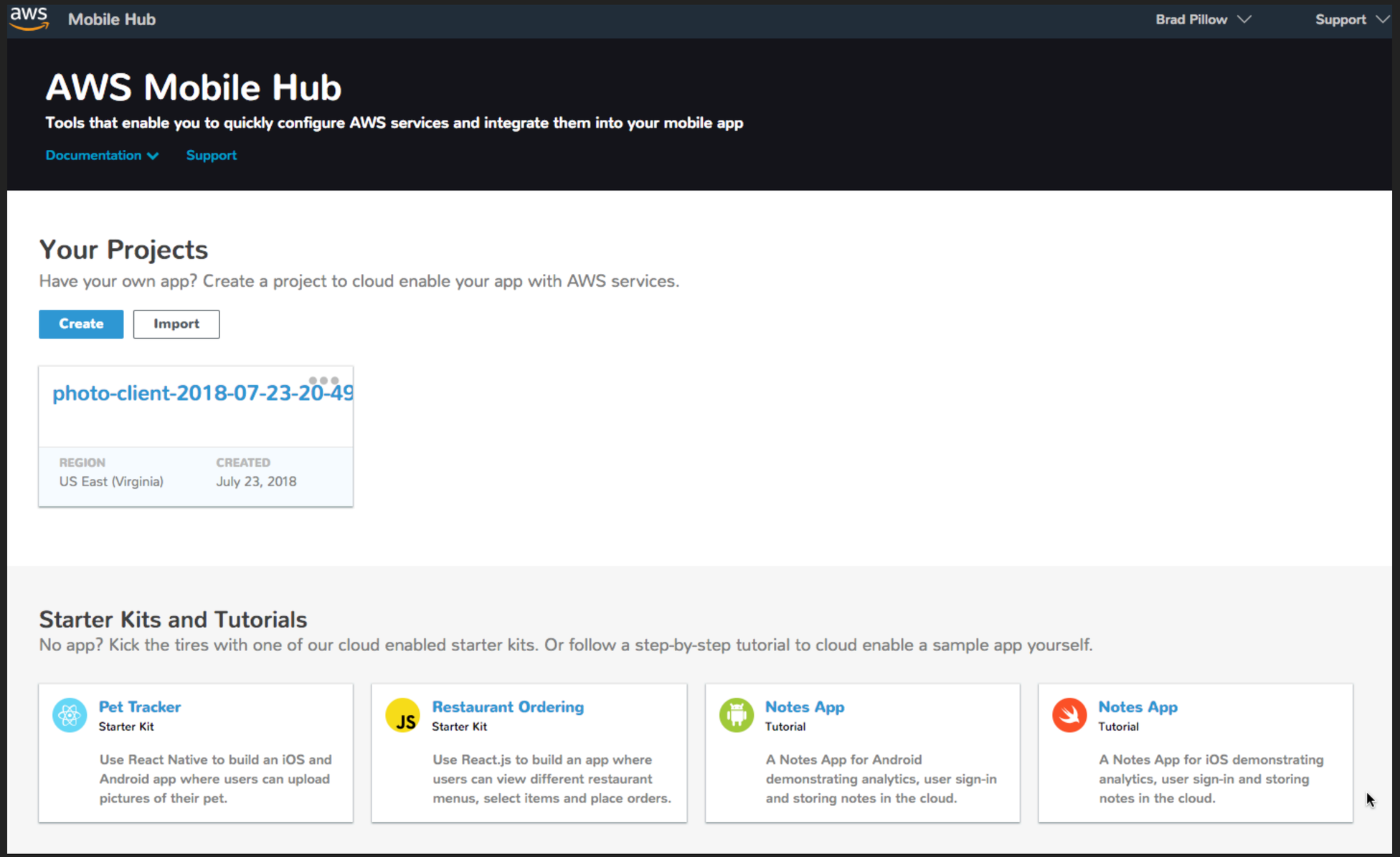
https://aws.amazon.com/blogs/mobile/new-aws-appsync-features-and-whitelist-removal/

- Ability to autogenerate a GraphQL schema and resolvers from an existing Amazon DynamoDB table
- Android support for offline queries and mutations
- Interface and union support in GraphQL schemas
- A “local” resolver to perform data transformations or publish actions using subscriptions
- Ability to access request headers, including custom headers, within a GraphQL resolver
- Resolver helper functions for common tasks
- AWS CloudTrail support, i.e. logging

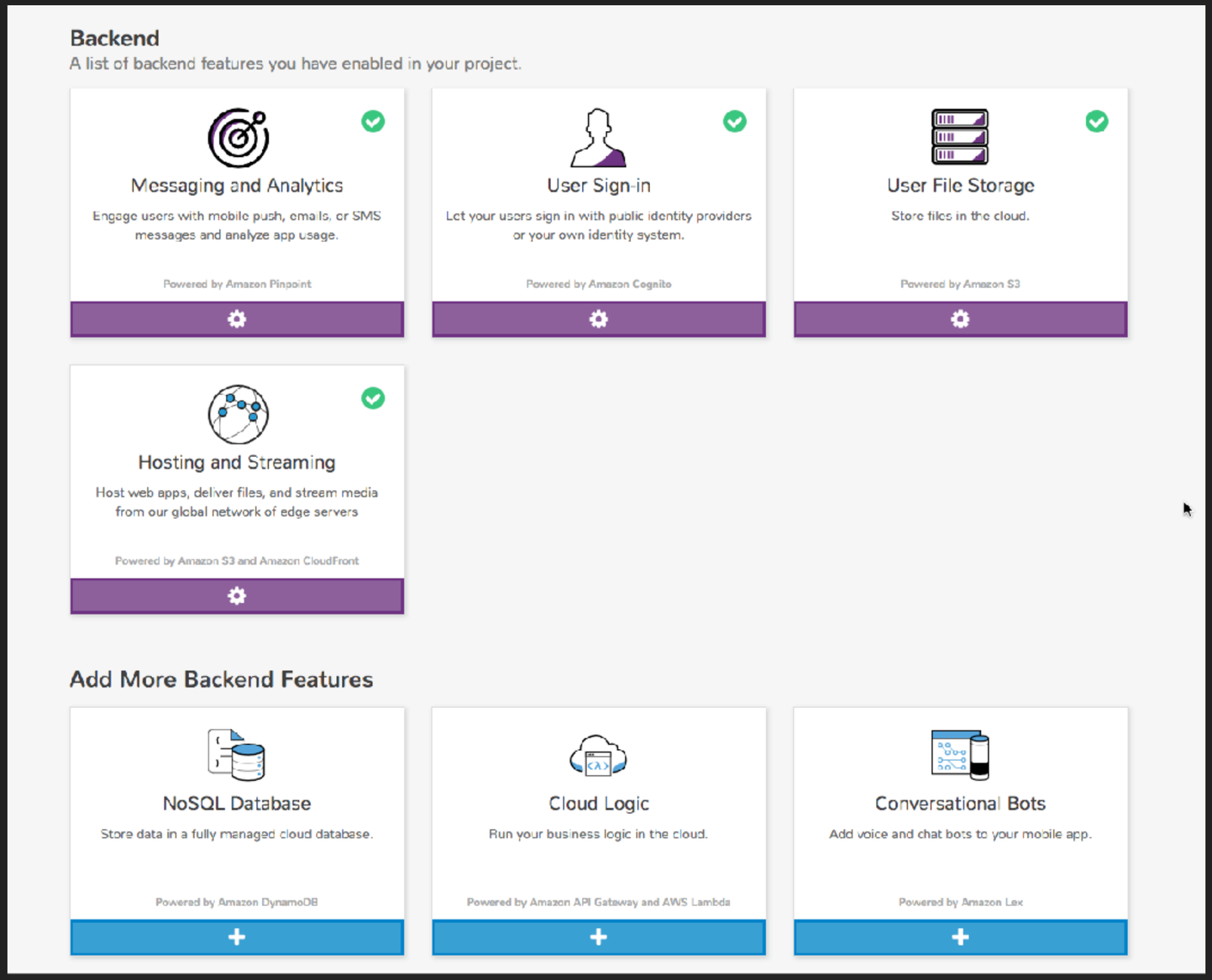
# MORE...

- Authorization, authentication, logging, analytics, messaging via AWS Amplify lib: https://aws-amplify.github.io/amplify-js/

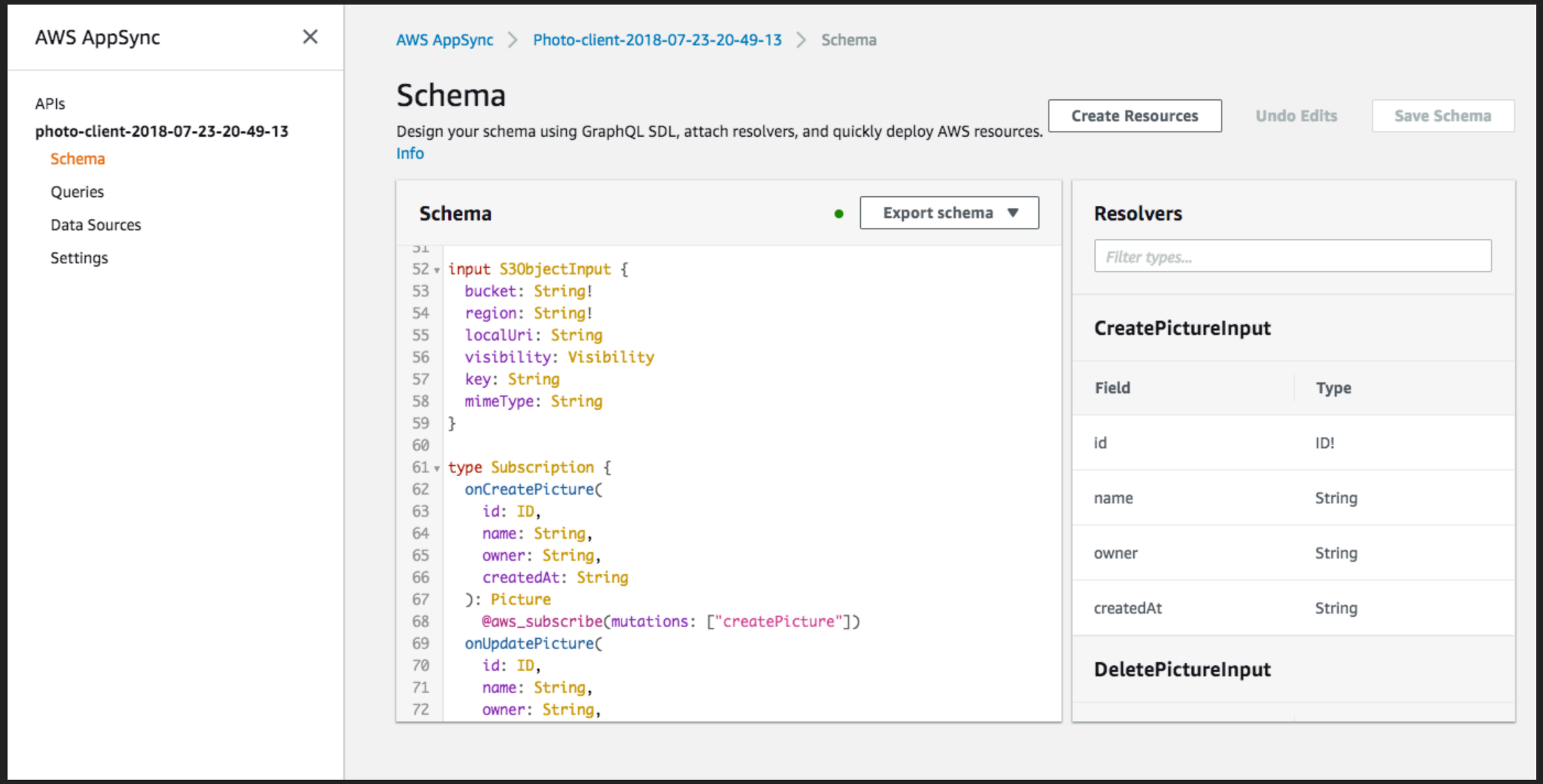
# AWS MOBILE HUB - DASHBOARD / CREATE APP



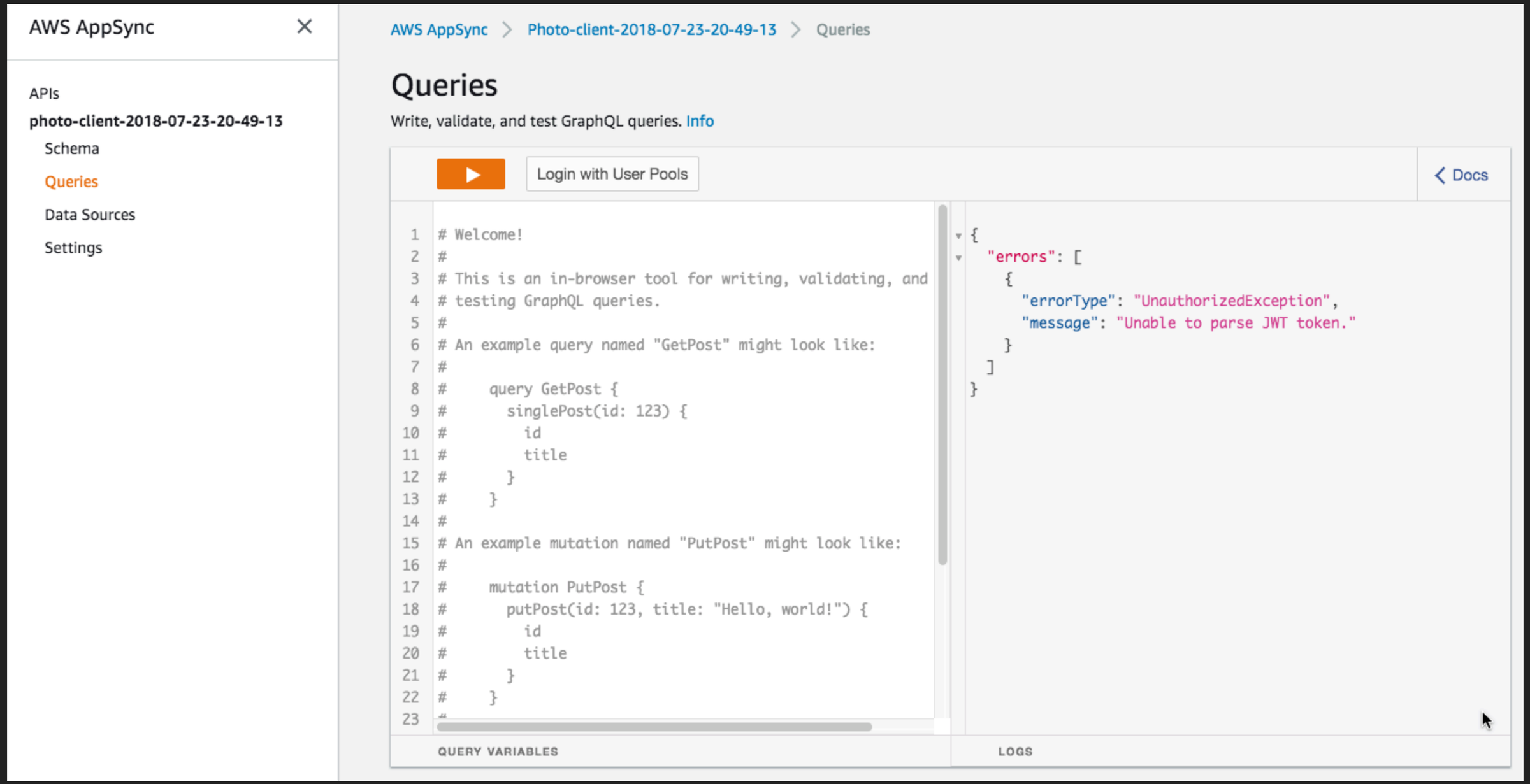
# AWS MOBILE HUB - SETTINGS / ADD FEATURE TO YOUR APP



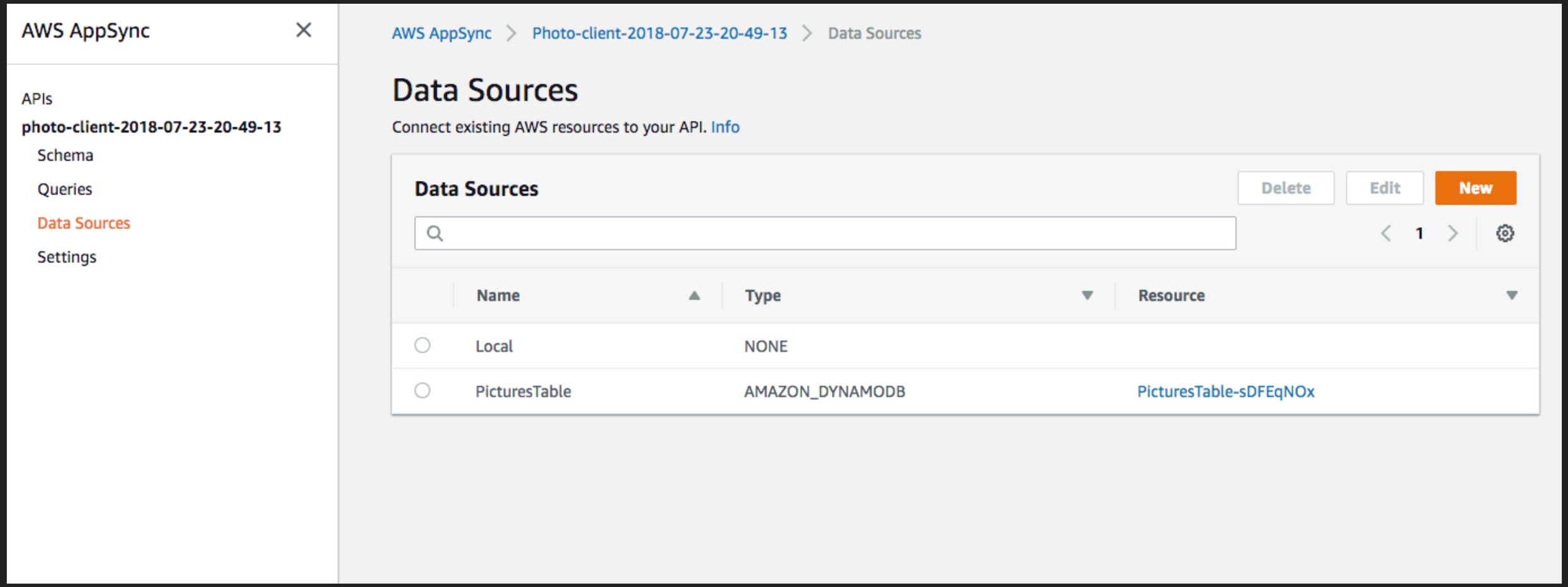
# APPSYNC - SCHEMA ENTRY



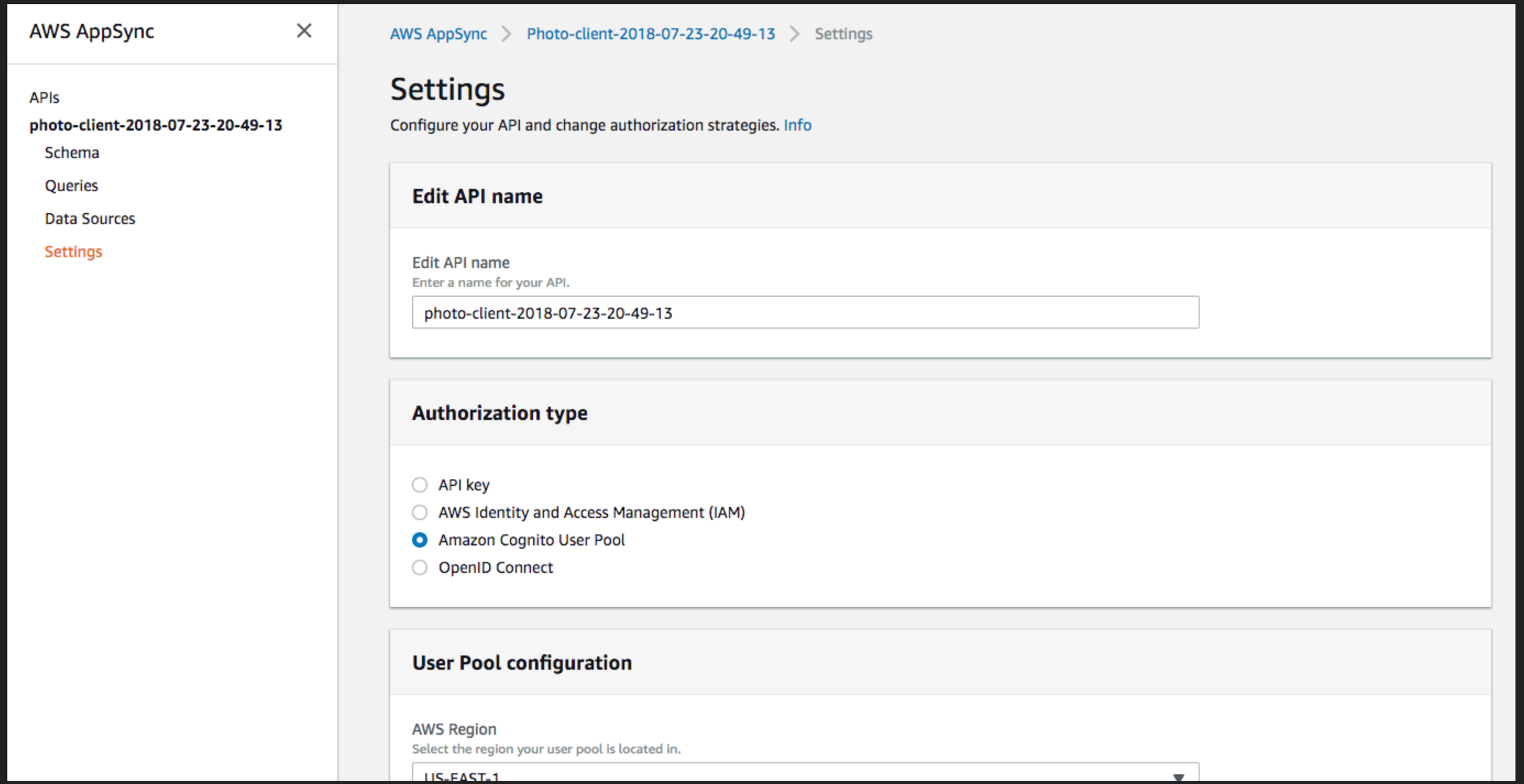
# APPSYNC - QUERY TESTING



# APPSYNC - DATA SOURCES



# APPSYNC - SETTINGS



# APPSYNC - RESOLVERS

AWS AppSync

APIs

**photo-client-2018-07-23-20-49-13**

- Schema
- Queries
- Data Sources
- Settings

## Resolver for Mutation.addPicture

Data source name

Select the data source to resolve.

PicturesTable

## Configure the request mapping template.

Translate a GraphQL query into a format specific to your data source. Info

Select a sample template

```
{
     "version" : "2017-02-28",
     "operation" : "PutItem",
     "key" : {
         "id" : { "S" : "${util.autoId()}" }
     },

    #set( $attribs = $util.dynamodb.toMapValues($ctx.args) )
    #set( $attribs.owner = $util.dynamodb.toDynamoDB($util.defaultIfNullOrBlank($context.identity.claims.username, "ANON")))
    #set( $attribs.file = $util.dynamodb.toS3Object($ctx.args.file.key, $ctx.args.file.bucket, $ctx.args.file.region, $ctx.
    #set( $attribs.createdAt = $util.dynamodb.toDynamoDB($util.time.nowISO8601()))
    "attributeValues" : $util.toJson($attribs)
}
```

## Configure the response mapping template.

Translate the results back to GraphQL. Info

Select a sample template

```
## Pass back the result object as is. **
$util.toJson($ctx.result)
```

# THANKS!

CODE IS HERE: HTTPS://GITHUB.COM/PIXELMEISTER/INDYJS-AUGUST-2018-TALK

# WORKHERE IS HIRING!

INTERESTED? CONTACT: rick@workhere.com

# DEMO/CODE!

https://github.com/pixelmeister/indyjs-august-2018-talk

# LOOK HERE FOR MORE...

- https://cli.vuejs.org/
- https://medium.com/the-vue-point/vue-cli-3-0-is-here-c42bebe28fbb