5. Run Test on BrowserStack Against with the Safari Browser

5.1 Sign Up for a Free BrowserStack Account

The nice thing about the Selenium WebDriver concept is, that it is fairly easy to replace the local selenium server with a remote one. In this course we will use BrowserStack as our provider. Please go ahead and sign up for a free trail at https://www.browserstack.com/users/sign_up to follow along. After you have signed up you can open the "Automate" dashboard at https://automate.browserstack.com/dashboard.

5.2 Get BrowserStack Automate Credentials

Go to https://www.browserstack.com/accounts/settings, look for the section "Automate" and copy your "Username" and "Access Key", we need in the next step.

5.3 Reconfigure CodeceptJS to Run Your Test in BrowserStack Cloud

To run our test against the Safari browser in the BrowserStack cloud we need to reconfigure CodeceptJS a bit. Open the `codecept.conf.js` file and add the following two properties to the `helpers.WebDriver` object:

1. `user: BROWSERSTACK_USERNAME`
2. `key: BROWSERSTACK_ACCESS_KEY`

Also change `browser: chrome` to `browser: safari` and add the following code:

```
desiredCapabilities: {
  os: "osx",
  browser_version: "12",
  resolution: "1920x1080"
},
```

The `browser_version" is needed here since if you do not define it, BrowserStack will randomly choose a Safari Version, this is not what we want.

5.4 Run Tests on BrowserStack

Run `npx codeceptjs run --steps` and open the BrowserStack Automate Dasboard https://automate.browserstack.com/dashboard.

We can find the running test in the left sidebar. If you watch the test run video which BrowserStack will provide you, you can see that the test stops to early but BrowserStack indicates the test as passed - whats wrong here? Just continue reading, we will fix this soon.

5.5 Install CodeceptJS BrowserStack Helper

To improve the test status on BrowserStack we will install the CodeceptJS BrowserStack helper. To do so, run `npm install codeceptjs-bshelper --save-dev` and add the following code to the already existing `helpers` property in the `codecept.conf.js` file:

```
"BrowserstackHelper": {
  "require": "codeceptjs-bshelper",
  "user": "BROWSERSTACK_USERNAME",
  "key": "BROWSERSTACK_ACCESS_KEY"
},
"REST": {}
```

The empty `REST` property is needed to enable the REST helper which is used to communicate with the BrowserStack API.

Re-run your test with `npx codeceptjs run --steps` .

After the test run is finished the CodeceptJS BrowserStack helper outputs the direct link to the job run on BrowserStack, open it and you will see that the test run is marked as failed and the scenario name is used as test name.

PS: The BrowserStack link is public to make it easy to share with all your fellow colleagues (think about adding it to a bug report or something similar).

PPS: Add a `project: "practical-e2e-testing"` key to the `helpers.WebDriver.desiredCapabilities` object to tag the BrowserStack job with a project name.

The full example config now looks like:

```
exports.config = {
  tests: "./*_test.js",
  output: "./output",
  helpers: {
    WebDriver: {
      url: "http://the-internet.herokuapp.com",
      browser: "safari",
      desiredCapabilities: {
        browser_version: "12",
        resolution: "1920x1080",
        project: "practical-e2e-testing"
      },
      user: "BROWSERSTACK_USERNAME",
      key: "BROWSERSTACK_ACCESS_KEY"
    },
    BrowserstackHelper: {
      require: "codeceptjs-bshelper",
      user: "BROWSERSTACK_USERNAME",
      key: "BROWSERSTACK_ACCESS_KEY"
    },
    REST: {}
  },
  include: {
    I: "./steps_file.js"
  },
  bootstrap: null,
  mocha: {},
  name: "my-auto-e2e-tests",
};
```