nwPlus

aws educate

Build and Deploy a Full Stack Web App with AWS

Learn Day 2019
#workshop-aws

Hey, it’s Anita & Chris 👋

nwPlus

What will you learn today?

1. How to create a web app with Express and NodeJS
    - Modern web development practices
    - Integrating an external API into your web app
    - Considerations for building user interfaces
2. How to deploy a web app with AWS
    - Introduction to AWS
    - Deploying with the Elastic Beanstalk console

nwPlus

What we won't be covering
(but are still important concepts!)

- Writing Tests
- Databases (MySQL, DBMSs, ...)
- JavaScript Frameworks (React, Vue, ...)
- JavaScript transpilers (Babel, ES6, ...)
- CSS preprocessors (SASS, SCSS, ...)
- Markup and Rendering (HTML)

nwPlus

Agenda

1. **Introduction**
2. **Environment Setup**
3. **Web App Development**
    a. Developing on the web
    b. Adding new functionality
    c. Using an external API
    d. Building user interfaces
4. **Web App Deployment**

nwPlus

What is Full stack? 🤔

A full stack application is a complete application with all of the components:

1. Frontend
    - Layout, buttons, images, user interactions etc.
2. Backend
    - The “brain” of the application
      ex. Retrieving information, reading inputs and button presses
3. (Not included) Database
    - Where data is stored and organized

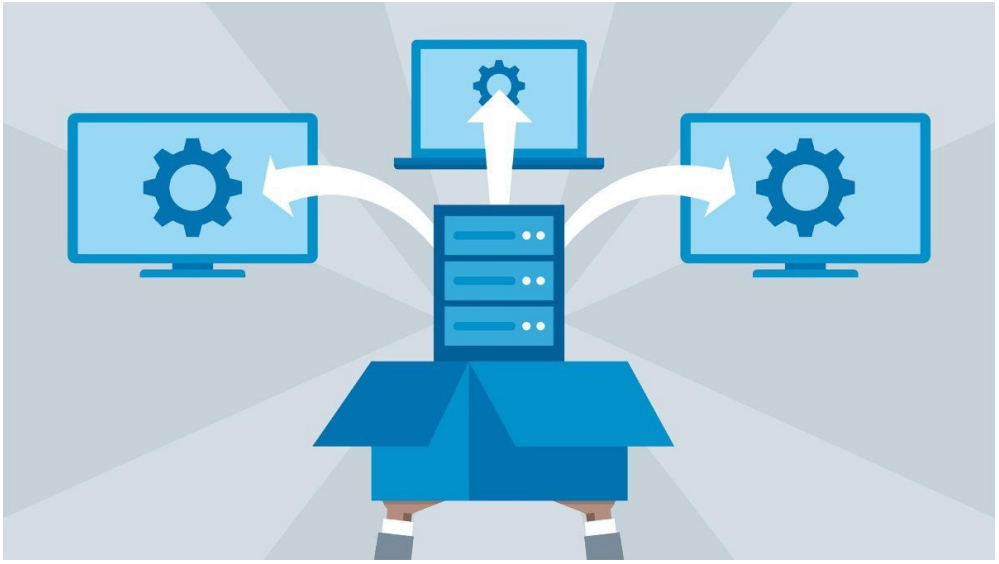
nwPlus

Development vs. Production

When you are writing code, your progress can only be seen on your **local machine** (your own computer) because it is currently in **Development.**

Development vs. Production

When you want your changes to go public, you need to **deploy** it into **Production** (the internet!) so others can see your app!



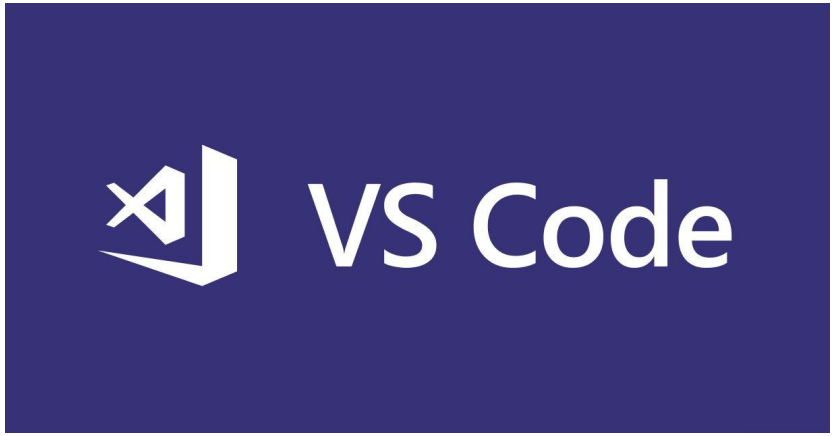
Environment Setup

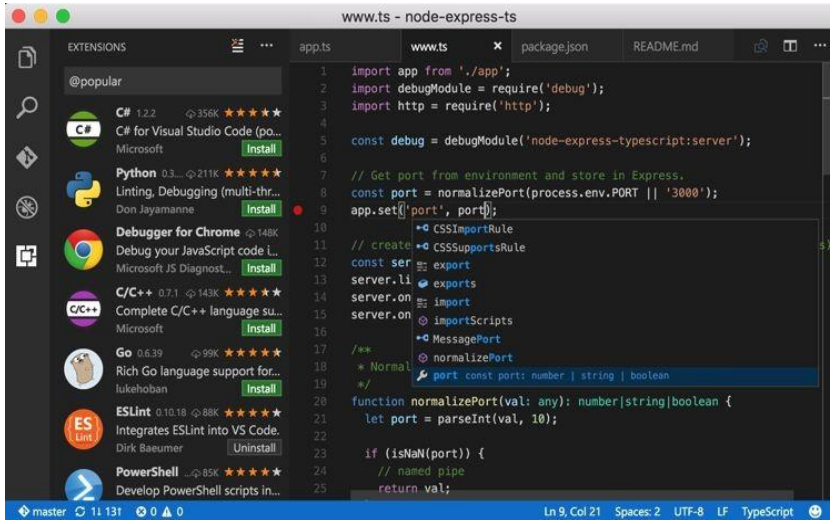
VS Code and NodeJS

Visual Studio Code

VS Code is a simple code editor- think of it like a Microsoft Word for code!

It offers **syntax highlighting** and **code completion**, which is like spelling and grammar check for code!

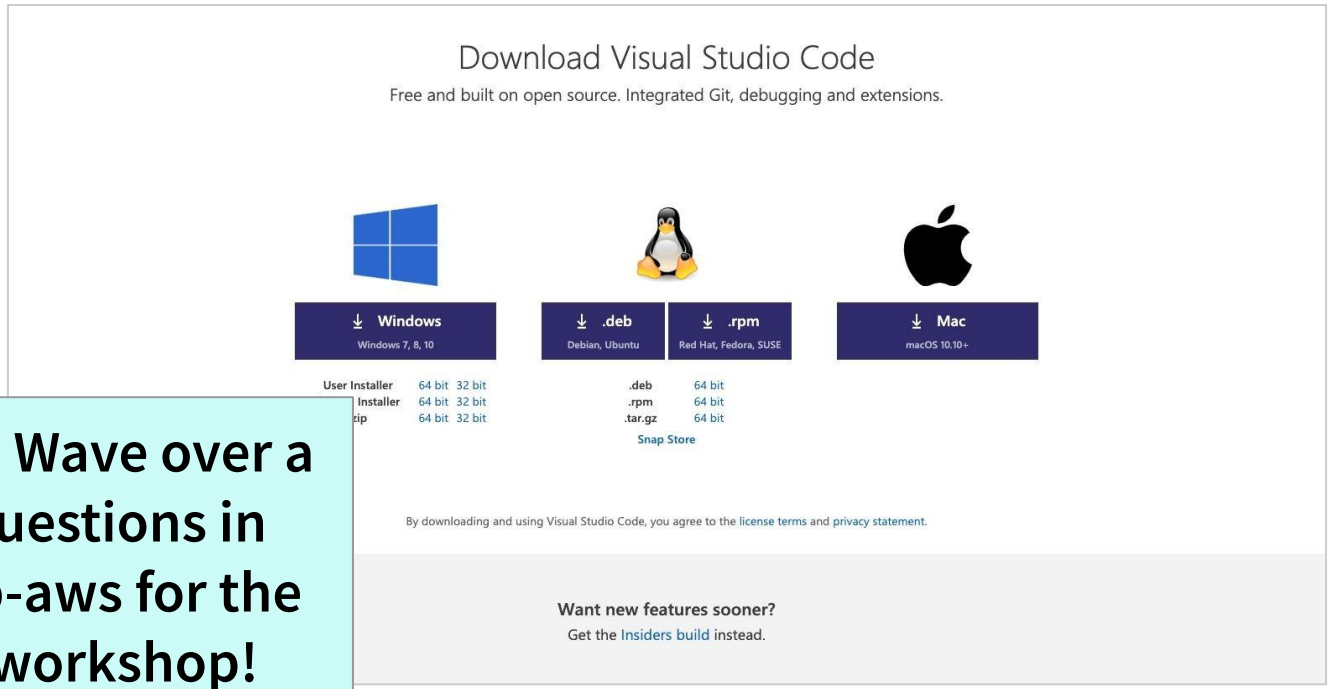




Install Visual Studio Code

Go to this link and download VS Code for your machine:

code.visualstudio.com/download

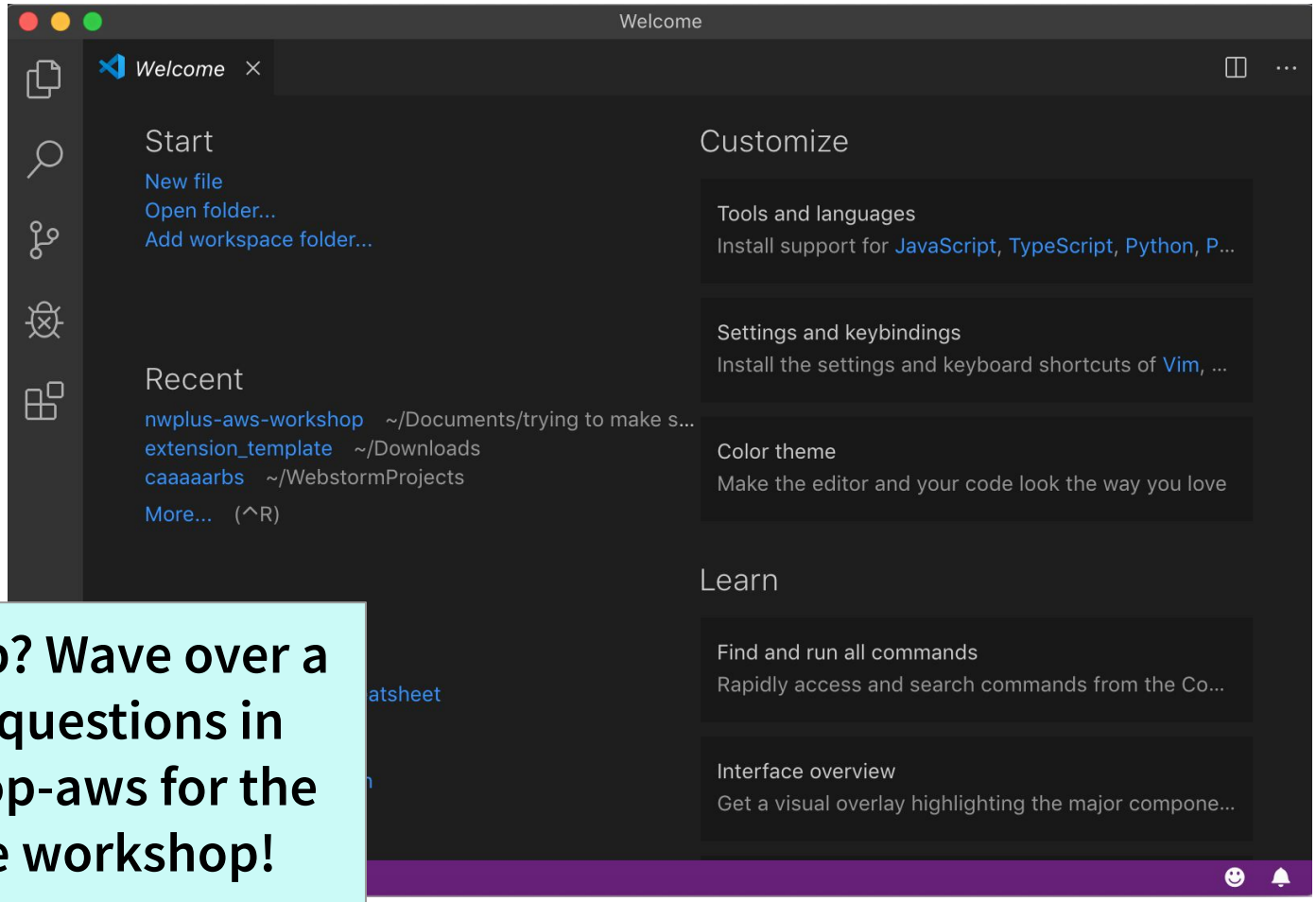


Need help? Wave over a TA or ask questions in #workshop-aws for the rest of the workshop!

nwPlus

Install Visual Studio Code

The program should look like this when opened:



Need help? Wave over a TA or ask questions in #workshop-aws for the rest of the workshop!

nwPlus

Node.js

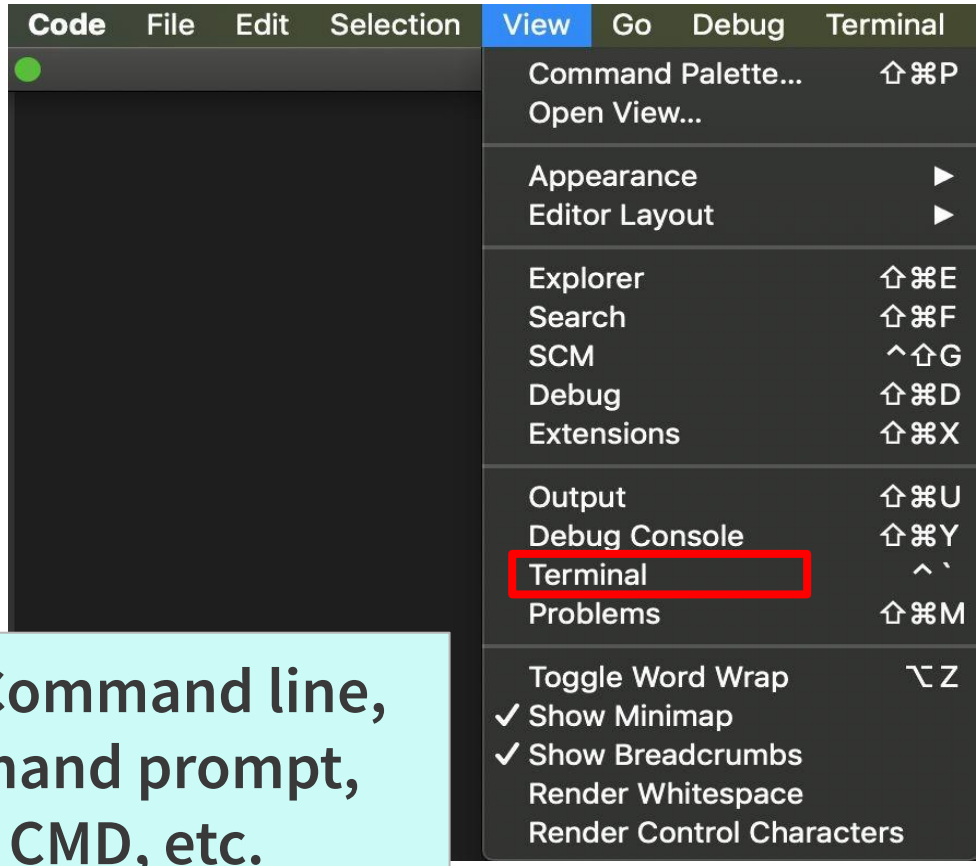
Node.js is a popular environment used to run Javascript and easily build web applications!

It prevents the need to use multiple languages and frameworks when developing an app.

Check for Node.js- Introducing the terminal

A terminal is an interface where you can type out **commands** for your computer to run

To open the terminal from VS Code go to:
View > **Terminal**

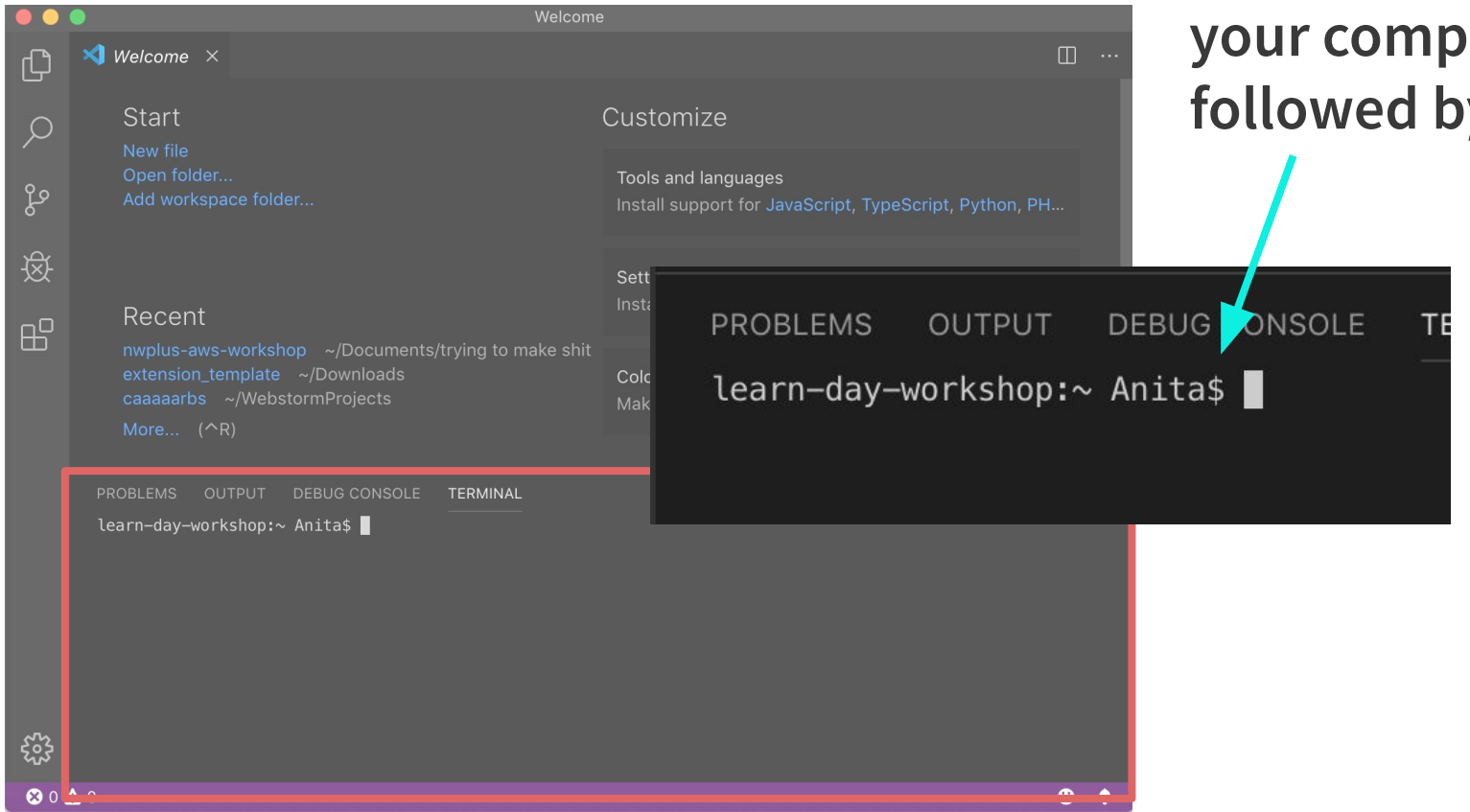


AKA Command line, command prompt, shell, CMD, etc.

Check for Node.js- Introducing the terminal

You've found the terminal!

Some text related to your computer name followed by a '$'



Check for Node.js- Introducing the terminal

Check if you have Node.js by entering the command:

```
$  node --version
```

When we give commands like this, type out the text after the '**$**' and press **ENTER**

If you get a version, you have it installed!

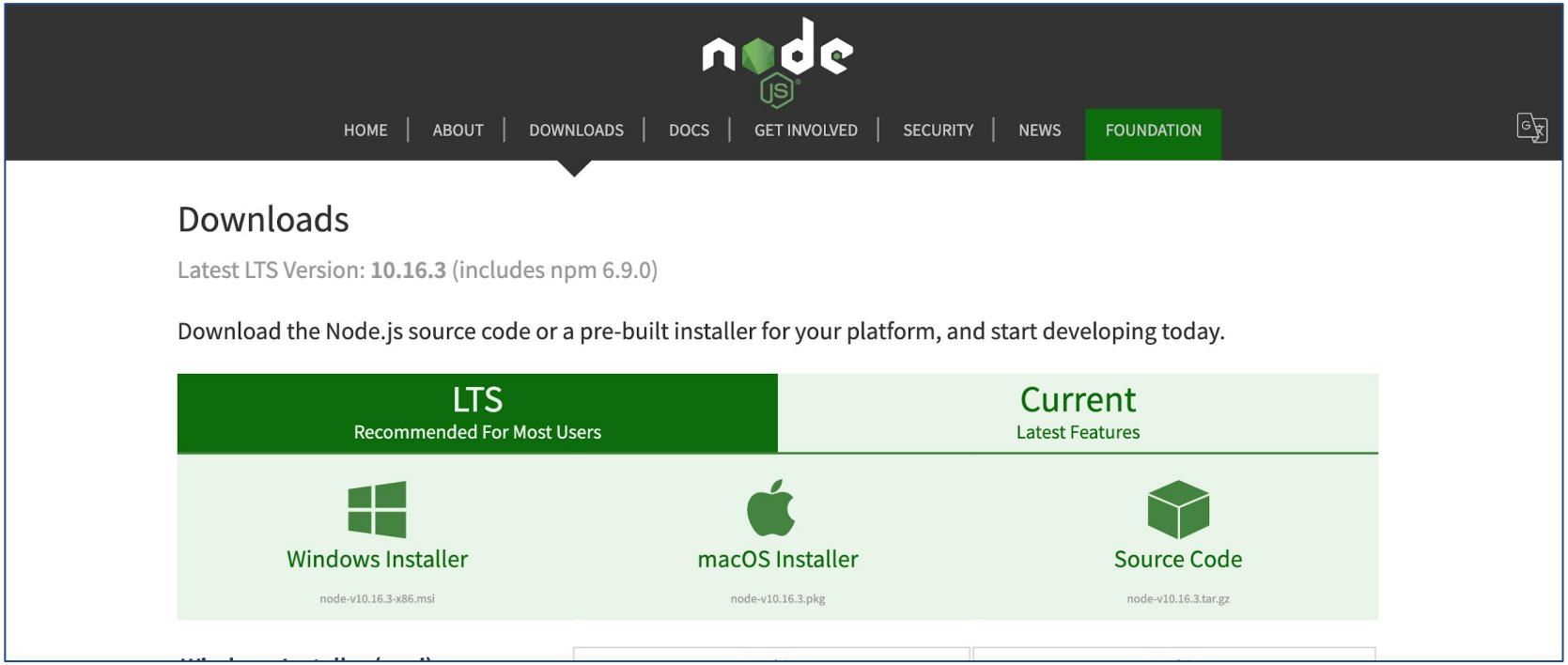
```
learn-day-workshop:~ Anita$ node --version
v8.12.0
```

Installing Node.js

If you don't have Node.js, install it for your machine here:

nodejs.org/en/download

version 8+



Check for npm

npm is a package manager used to help developers install programs they need to build their software. npm should come with your node.js installation.

To check for npm:


```
$  npm --version
```

If you get an error → ask a TA for help!

nwPlus

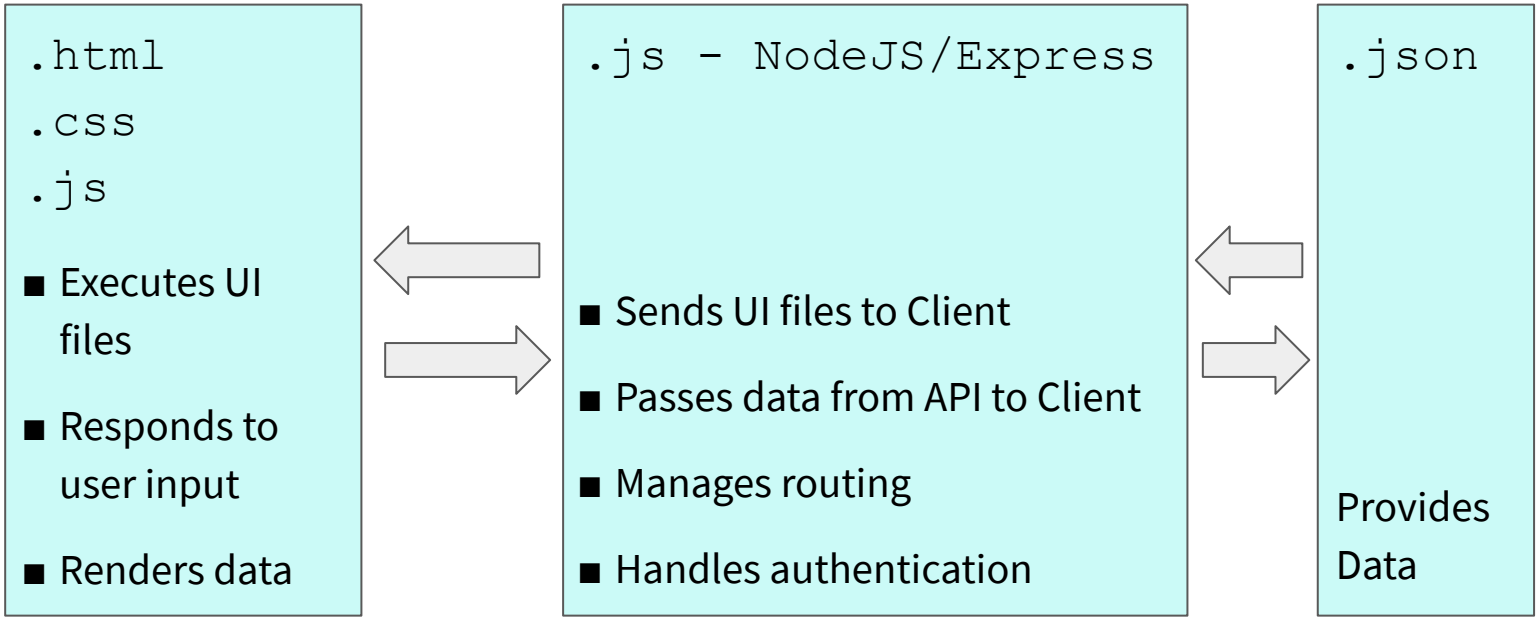
Web App Development

With NodeJS and Express

What we'll be building

See example here

What we'll be building



Our device
"Client"
"Frontend"

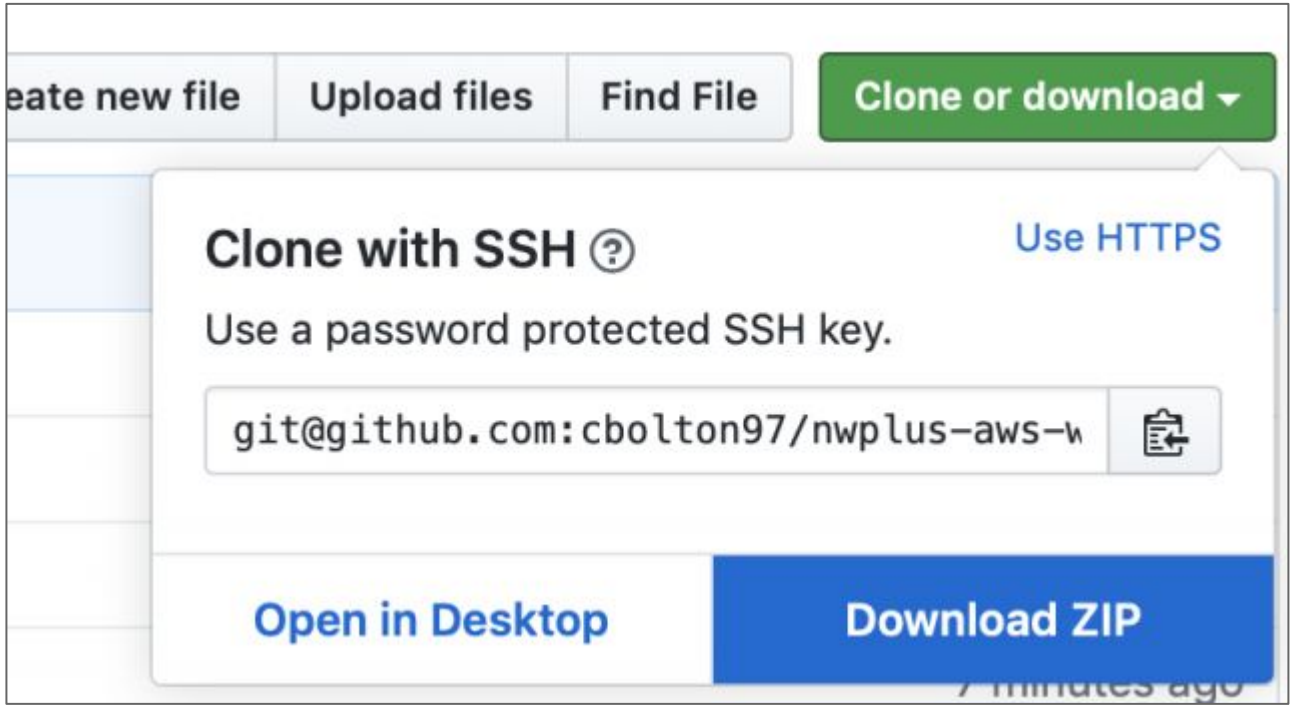
Our server
"Server"
"Backend"

External data
"API"

nwPlus

Download the starter code

http://tiny.cc/lhdaws0

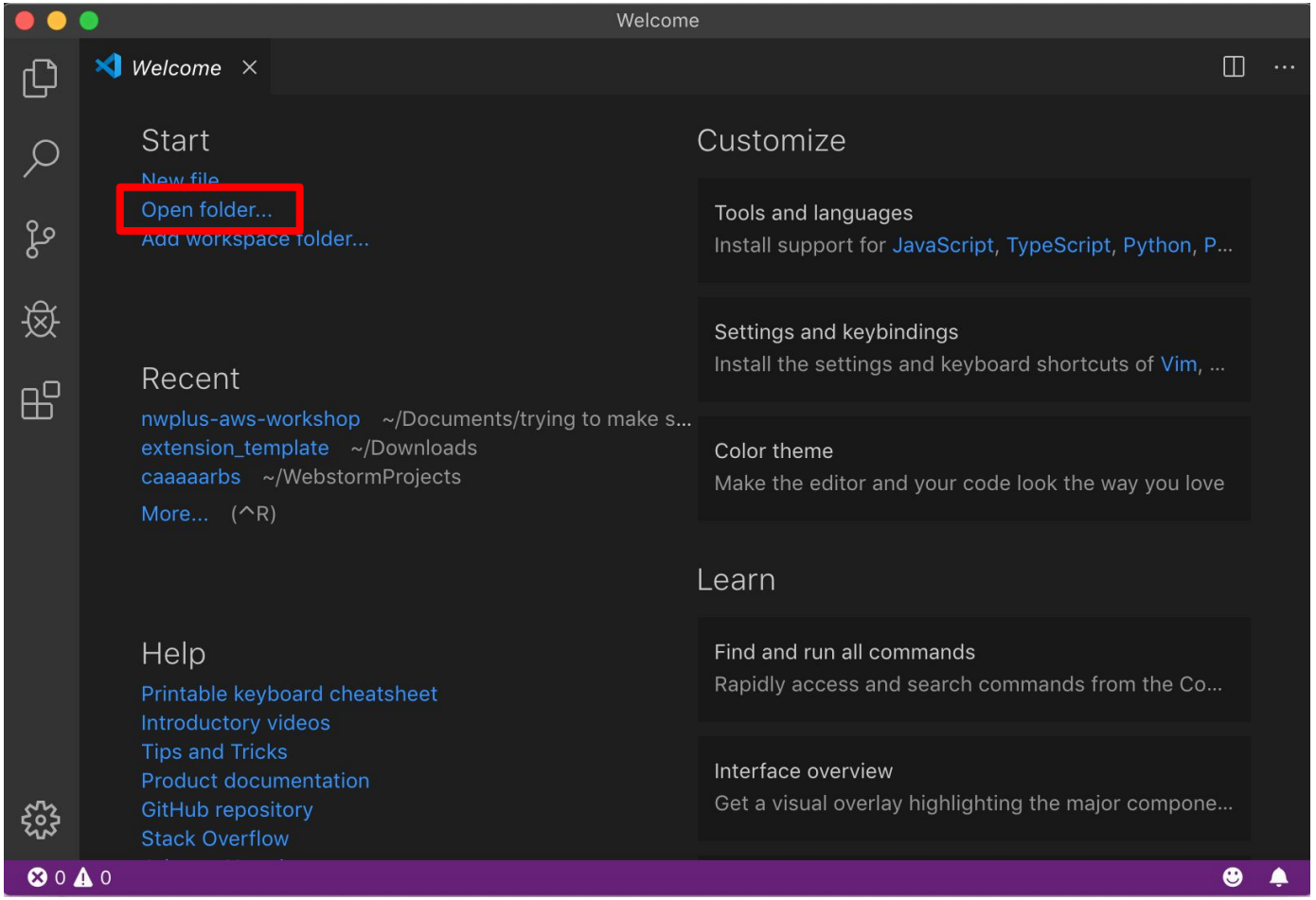


A note on tech choices

- We're using NodeJS and Express for our server, but this is just one "stack" out of many
- For example, [AWS supports the following:]

Opening our project

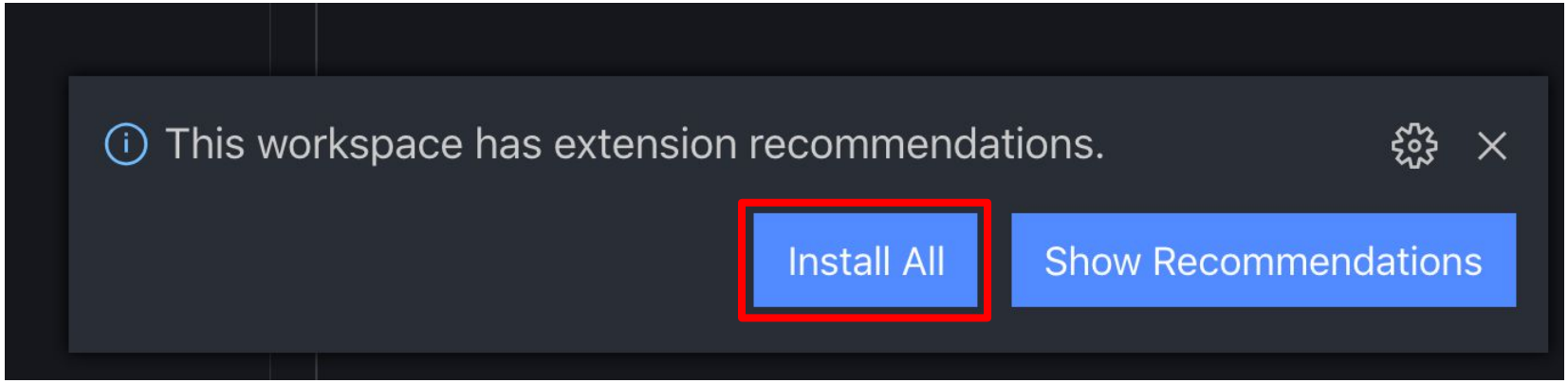
Use the “Open Folder” button to open the starter code



nwPlus

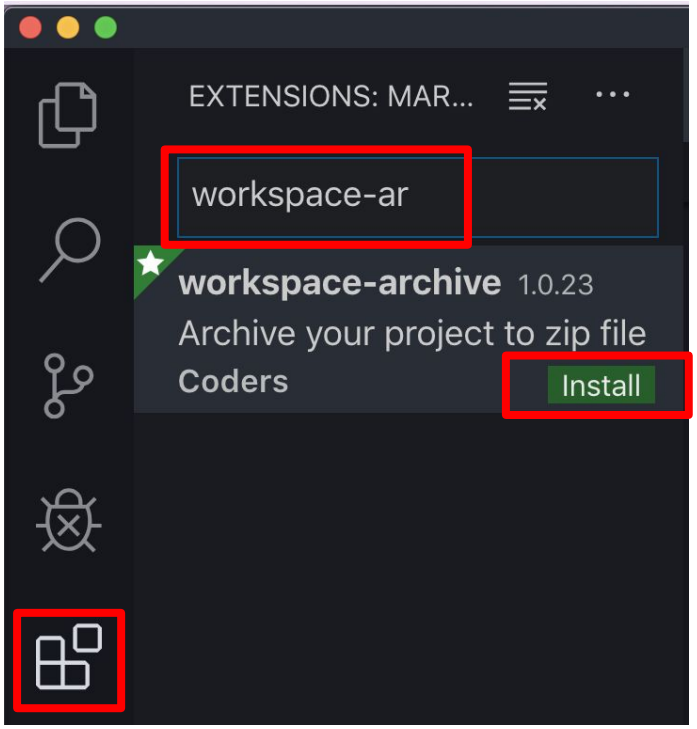
Install the Workspace Archive extension

Click “Install All” when the popup appears in the bottom right corner:



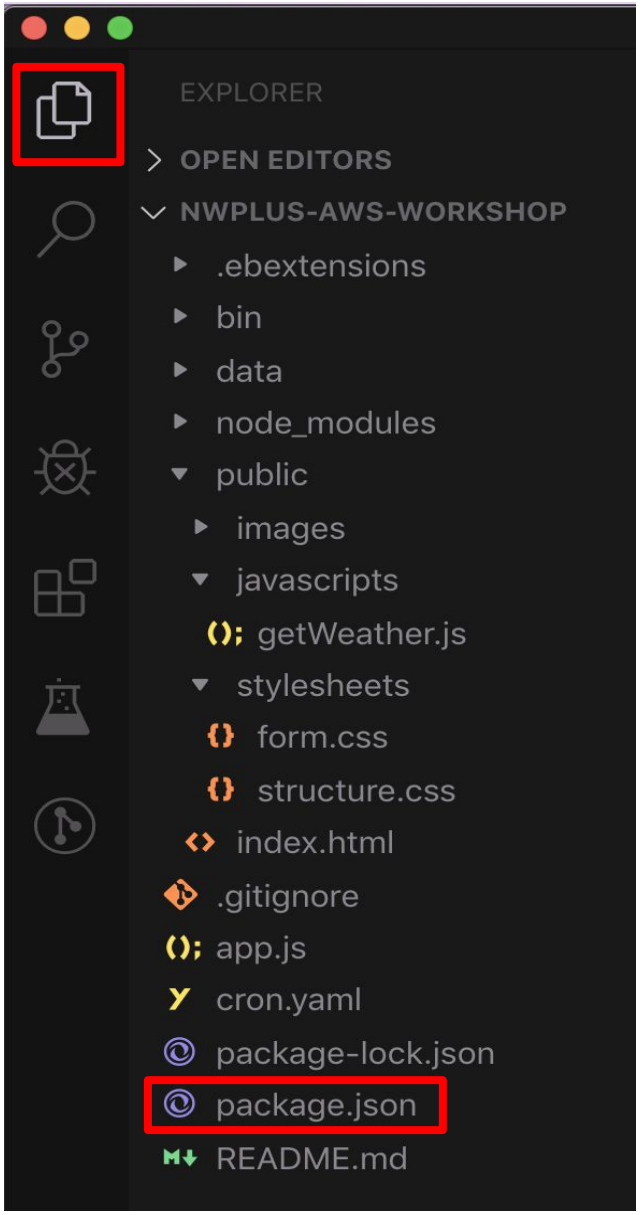
If you don’t get prompted to install the extension:

Go to “Extensions”, search for “workplace-archive” and then install it:



Package.json

Using the sidebar, let's look at `package.json`



Package.json

- This is the configuration file for NodeJS
- Scripts are commands we can use in the terminal
- “dependencies” and “devDependencies” are the software packages we’ll be leveraging

```
"dependencies": {
  "cookie-parser": "~1.4.4",
  "debug": "~2.6.9",
  "express": "~4.16.1",
  "morgan": "~1.9.1",
  "node-fetch": "^2.6.0",
  "request": "^2.88.0"
},
"devDependencies": {
  "nodemon": "^1.19.3"
}
```

```
"scripts": {
  "start": "node ./bin/www",
  "devstart": "nodemon ./bin/www"
},
```

nwPlus

Software Packages

- In NodeJS, we use the Node Package Manager (NPM) to install external software packages
- These packages give us certain functionality without the need to reinvent the wheel
- Example: *Express*, one of these packages, gives us a foundation for creating a reliable web app
- NodeJS-based projects are famous for using thousands of software packages - our project will leverage ~6000 packages in total!

nwPlus

Starting our project

In our Terminal, we'll need to first install the software packages we've defined in package.json:

```
$   npm install
```

Then, we'll want to start our server *locally*:

```
$   npm run devstart
```

nwPlus

Starting our project

After running devstart, you should see the following in your terminal:

```
Server running at http://127.0.0.1:8081/
```

If we open this link (cmd/ctrl + click), we can view the site in our browser:

nwPlus

Some issues with the Starter Files

1. Doesn’t show the temperature
2. Uses fake sample data
3. No loading state

Let’s improve our app by tackling:

1. TODO: Show the temperature
2. TODO: Use a real API for getting the weather
3. TODO: Add a loading state

Show the temperature

How do we know data contains weather information? Or a weather description? Let's look at the sample weather data by opening `data/weather.json`

This data is in the same "shape" as data from the real API source. As you can see, it contains weather information and associated weather descriptions for each location in the sample.

Digging deeper, we can see there is also a temperature attribute!

nwPlus

Show the temperature

Open `public/javascripts/getWeather.js` **and focus on** `handleFormResponse` **on lines 15-24**

This function is doing a few things:

- **Checking to see if our data contains weather information (line 16)**
  - **If it does, we output the description of the weather (line 19)**
  - **If it doesn't, we output an error message (line 22)**

nwPlus

Show the temperature

Now that we know our data contains a city's temperature, we can access it in our code:

```
var temperature = data.main.temp;
```

Then, we can combine it with the weather description:

```
var weatherOutput = `${temperature}°C ${description}`;
```

(Add this part)

nwPlus

Show the temperature

Going back to our site, we can see the following:

To remove the extra decimal, we can add this to our code:

```
var temperature = data.main.temp.toFixed(1);
```

(Add this part)

Show the temperature

In review:

- We can figure out what data we have available by looking at the response our server gives us
  - Alternatively: we could have looked at the API documentation
- We can use Template Literals in JavaScript to combine values into one string
- JavaScript has a number of built in utilities (like `.toFixed()`) that allow us to manipulate data

nwPlus

Let’s improve our app by tackling:

✓ DONE: Show the temperature

2. TODO: Use a real API for getting the weather
3. TODO: Add a loading state

Checkpoint 1 code:

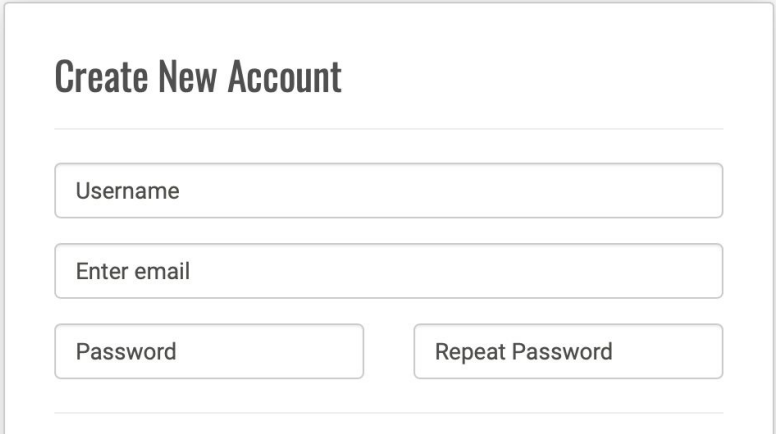
http://tiny.cc/lhdaws1

Using Real Data

We can use OpenWeatherMap's API to get real-time weather reports (this is where our sample came from).

First, you'll need an account in order to get an API key. Register here:

http://tiny.cc/WeatherAPI



nwPlus

Using Real Data

Once you register, you'll get an email with your API Key. In `app.js`, we'll want to add that on line 10:

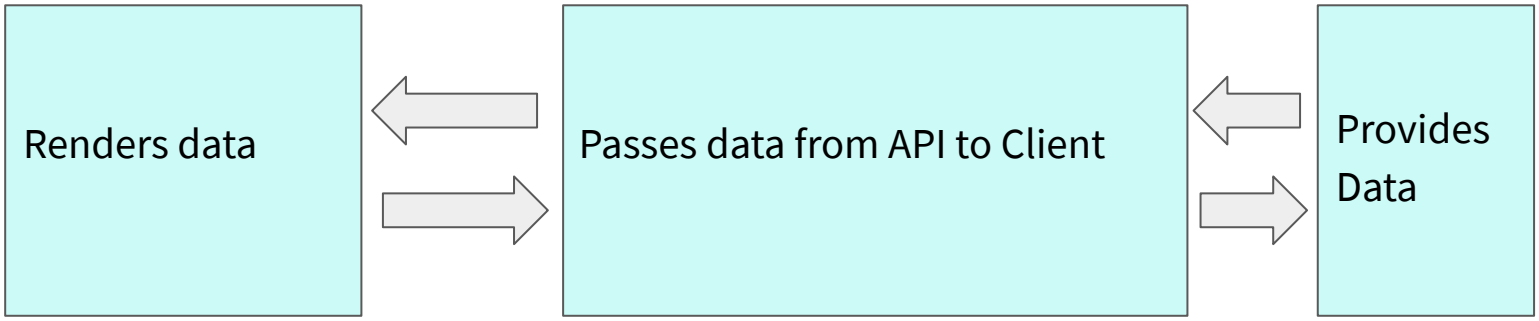
```
var API_KEY = "Enter your API Key here";
```

Replace this with your key, but keep the quotes

Note that in general, you shouldn't store API keys in code since it can be easily stolen. In this case, our key isn't accessing sensitive information, usage doesn't cost money, and our key can easily be revoked, so we're okay.

nwPlus

Using Real Data



In `app.js`, line 26 is where our server passes data onto the client. Right now, this is sample data.

Using the info given to us in the TODO comment, we should be able to replace the sample data with real API data.

Using Real Data

In review:

- External APIs can be used to give us access to vast amounts of data
- API keys should usually be stored securely as environment variables
- Often, your server will act as a “pass through” to external APIs. During this process, you can change the data to fit your use case

nwPlus

Let’s improve our app by tackling:

✓ DONE: Show the temperature

✓ DONE: Use a real API for getting the weather

3. TODO: Add a loading state

Checkpoint 2 code:

http://tiny.cc/lhdaws2

Adding a loading state

Now that our app is pulling data from the internet, we run into issues:

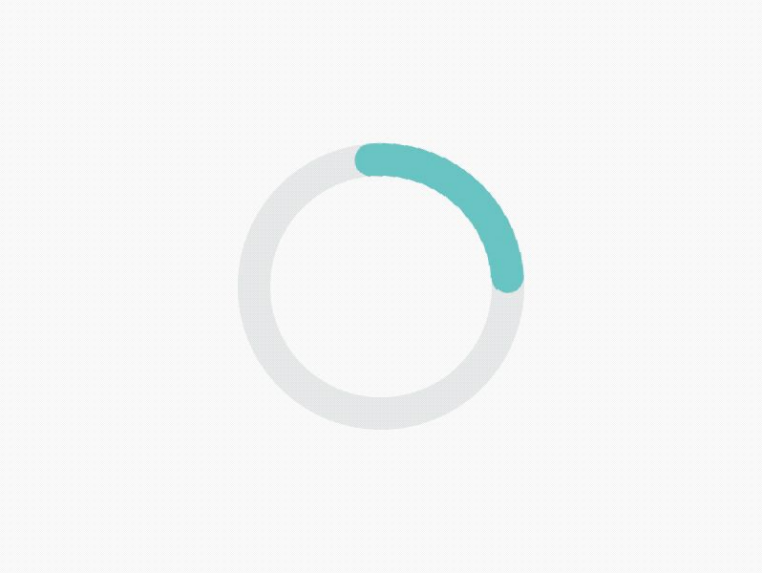
- What happens if that data source disappears?
- What if a user's internet connection is slow?

In general, we don't want to assume anything about the reliability of external APIs.

How can we still provide a good user experience with this in mind?

Adding a loading state

Loading states are a way to give *feedback* to users that their input has been received. Because feedback should be immediate, it is usually handled on the frontend.



Adding a loading state

Let's start by going back to `getWeather.js` and looking at the `handleFormSubmit` function.

`fetch()` is an asynchronous function, meaning the `.then()` portion only runs after the request is complete.

We can add a loading state by adding an output before the `fetch()` is run.

nwPlus

Adding a loading state

Now that the loading state is in, we should add some styling. Open `public/stylesheets/structure.css` **and go to line 45.**

We can use the `'loading'` **class we passed to add an** `opacity` **of** `0.1` **to our loading state.**

Adding a loading state

In review:

- Sending data across the internet can take a variable amount of time
- Giving immediate feedback to users after they interact with your web app is critical
- CSS is used to style our web app, while HTML is used to define our app structure

Let’s improve our app by tackling:

- ✓ DONE: Show the temperature
- ✓ DONE: Use a real API for getting the weather
- ✓ DONE: Add a loading state

Checkpoint 3 code:

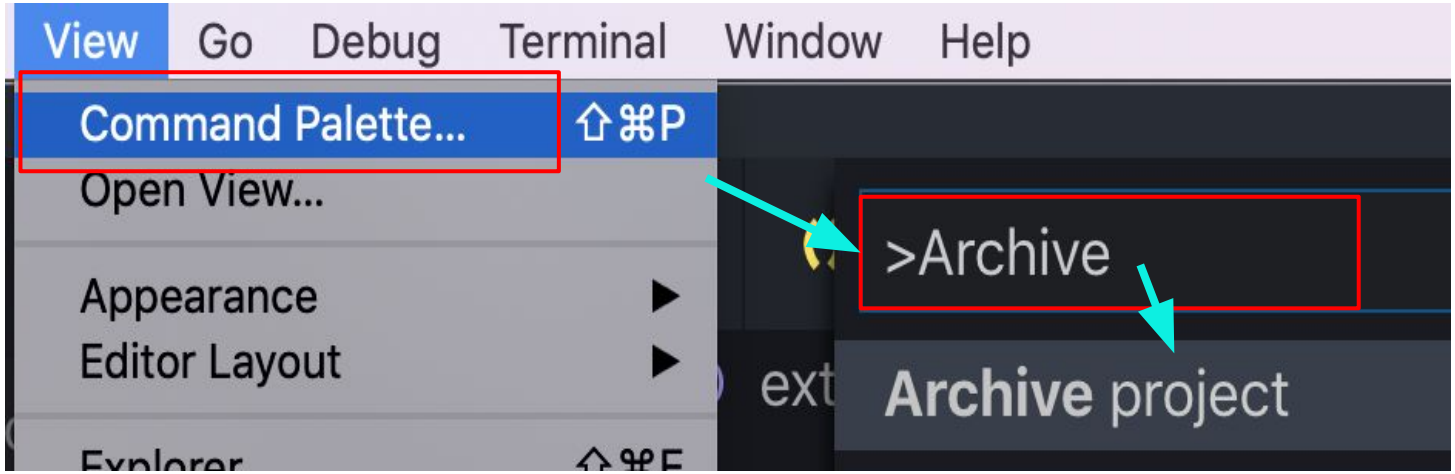
http://tiny.cc/lhdaws3

Stretch Goals

- Style error messages differently from the weather
- Use an animated loading indicator
- Use the web's Geolocation API to find a user's city automatically
- Cache API results to a database
- Use OpenWeatherMap's forecast API to return the weather forecast with a new endpoint

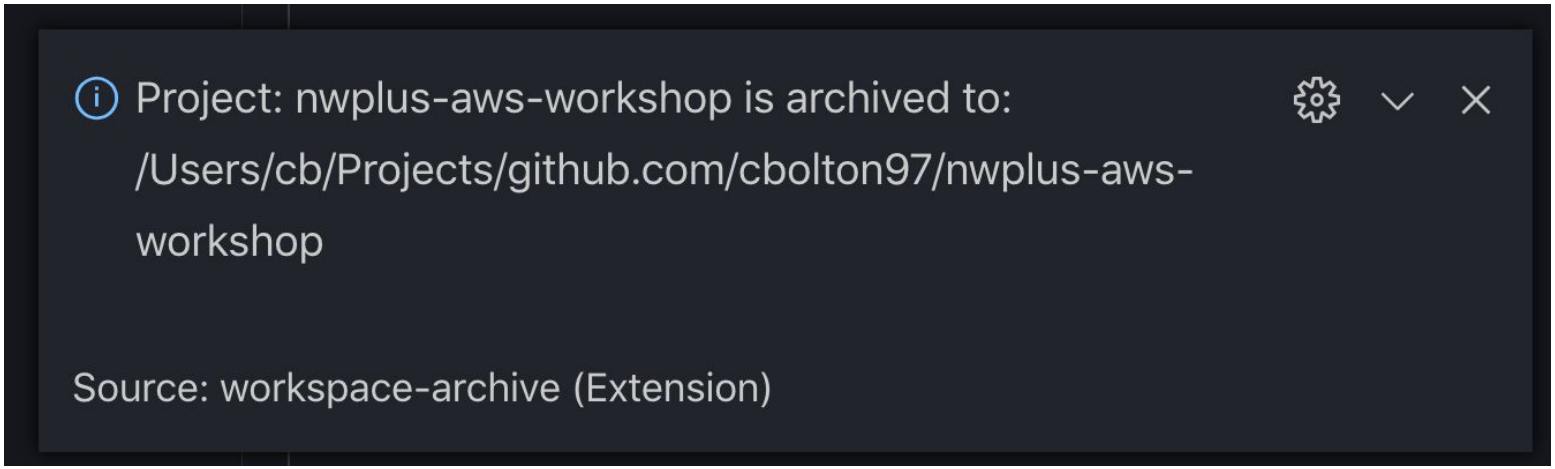
Now we're ready to compress our code for deployment

Open the Command Palette, type "Archive" and select the "Archive project" action:



Now we're ready to compress our code for deployment

The `.zip` file will be created in same folder where our code is:



nwPlus

Deployment with AWS

AWS Elastic Beanstalk

What is Amazon Web Services (AWS)?

AWS is a Cloud computing platform created by Amazon!

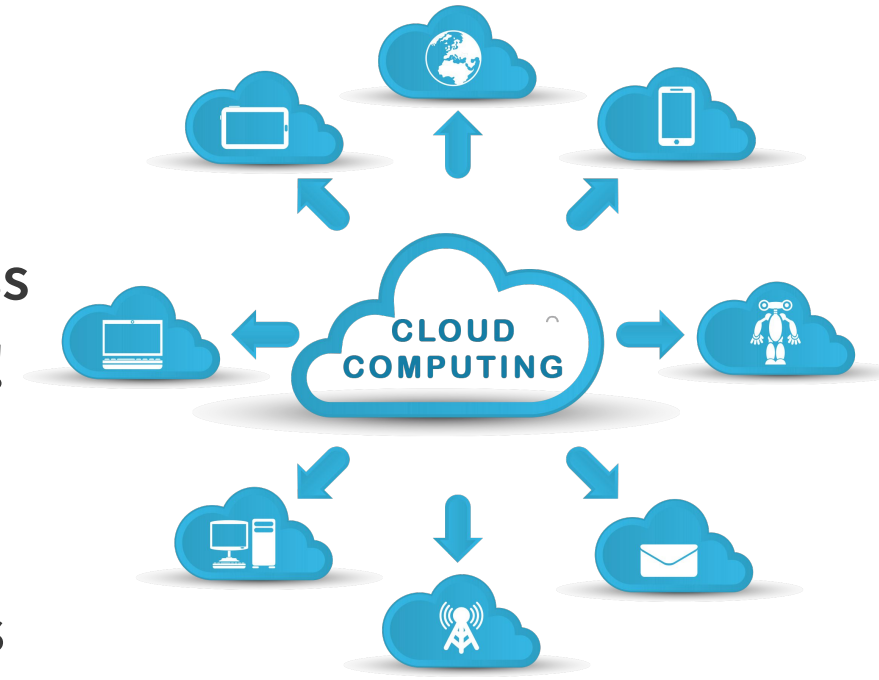
Fun fact: **34% of the world's cloud** is hosted on AWS

nwPlus

What and why cloud?

Cloud services is all of the computing services (servers, databases, storage drives, etc.) you would need to run a business or website... but on the internet!

It is a fast, scalable, and affordable option for developers



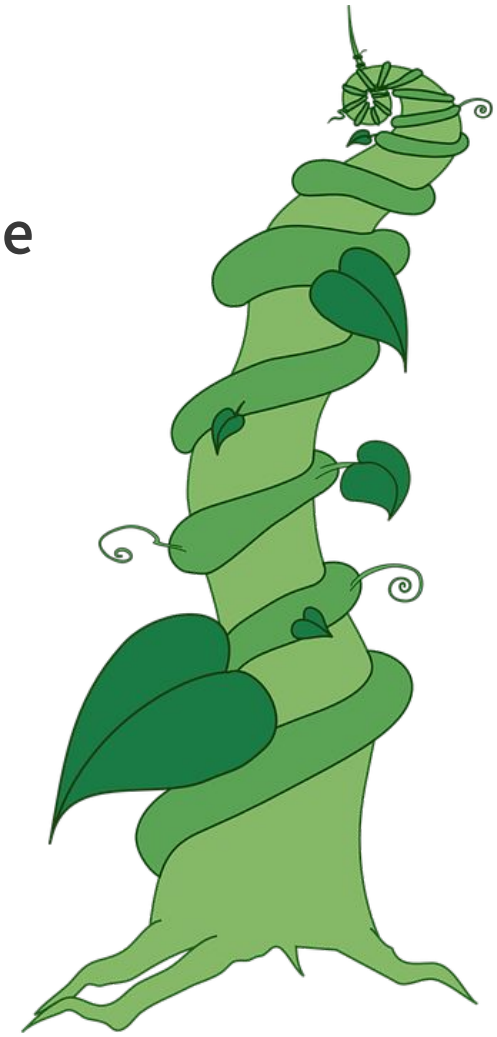
nwPlus

Elastic Beanstalk

Elastic Beanstalk (EB) lets you use all the computing services you need to deploy your project without needing to think about which ones you're using!

It **grows** and **scales** with the **click of a button** 🌱

We will be using Elastic beanstalk to deploy your project.



nwPlus

If you haven't done so already ... Sign up for an AWS account!

If you're a student you can sign up for **AWS Educate**. AWS Educate is an Amazon program that provides students comprehensive resources for building skills in cloud technology. Join AWS Educate and you will receive **$100** in AWS Promotional Credits to use in this workshop!

mlhlocal.host/aws-student-signup

If you're a non-student you can still sign up for an **AWS** account with 12 months of free tier access. However, a Credit Card is required to complete your registration.

mlhlocal.host/aws-signup

nwPlus

If you haven't done so already ... Sign up for an AWS account!

If you're a student you can sign up for **AWS**

TIP: Use your @alumni.ubc.ca email to get quick verification

mlhlocal.host/aws-student-signup

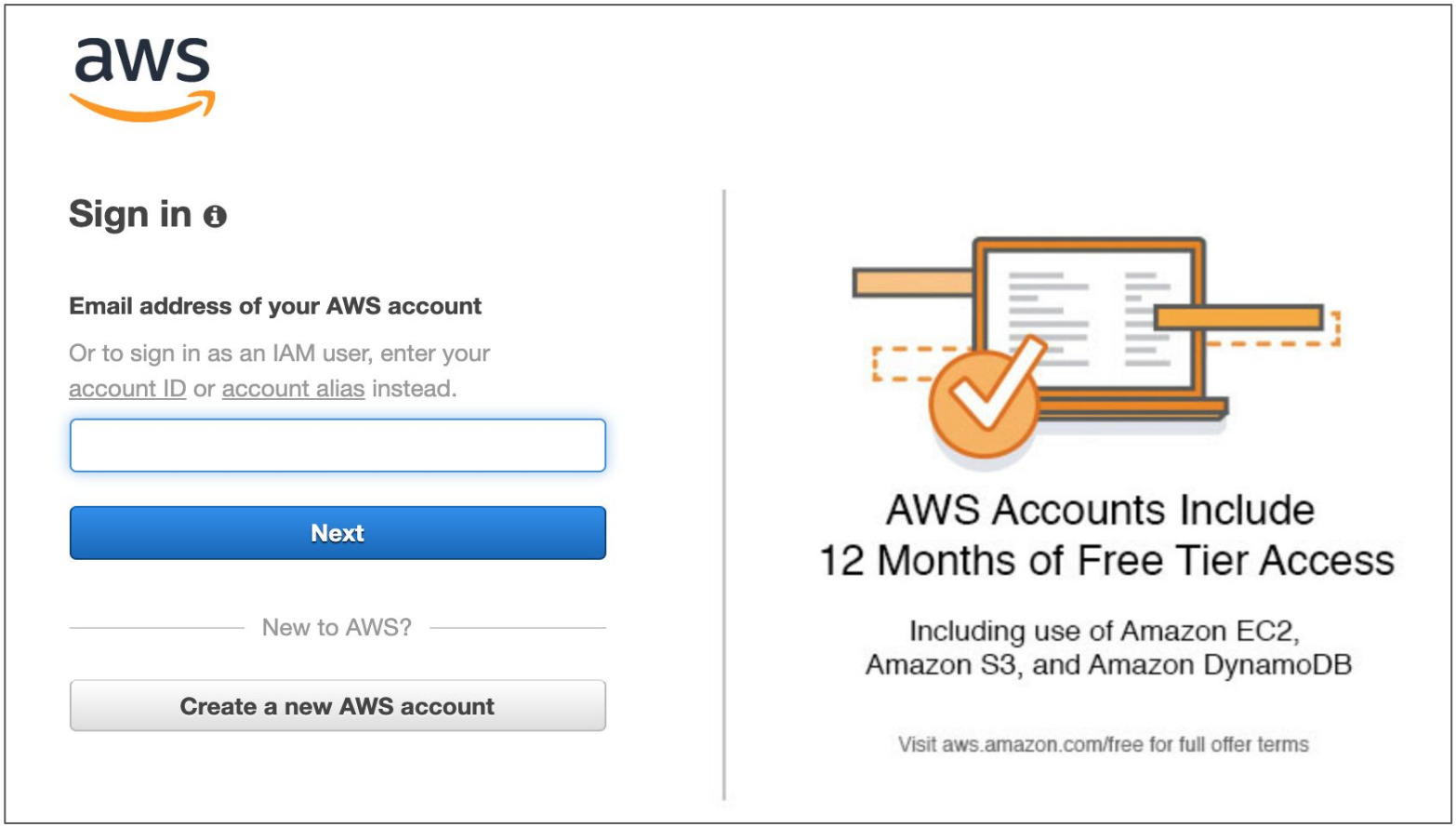
If you're a non-student you can still sign up for an **AWS** account with 12 months of free tier access. However, a Credit Card is required to complete your registration.

mlhlocal.host/aws-signup

nwPlus

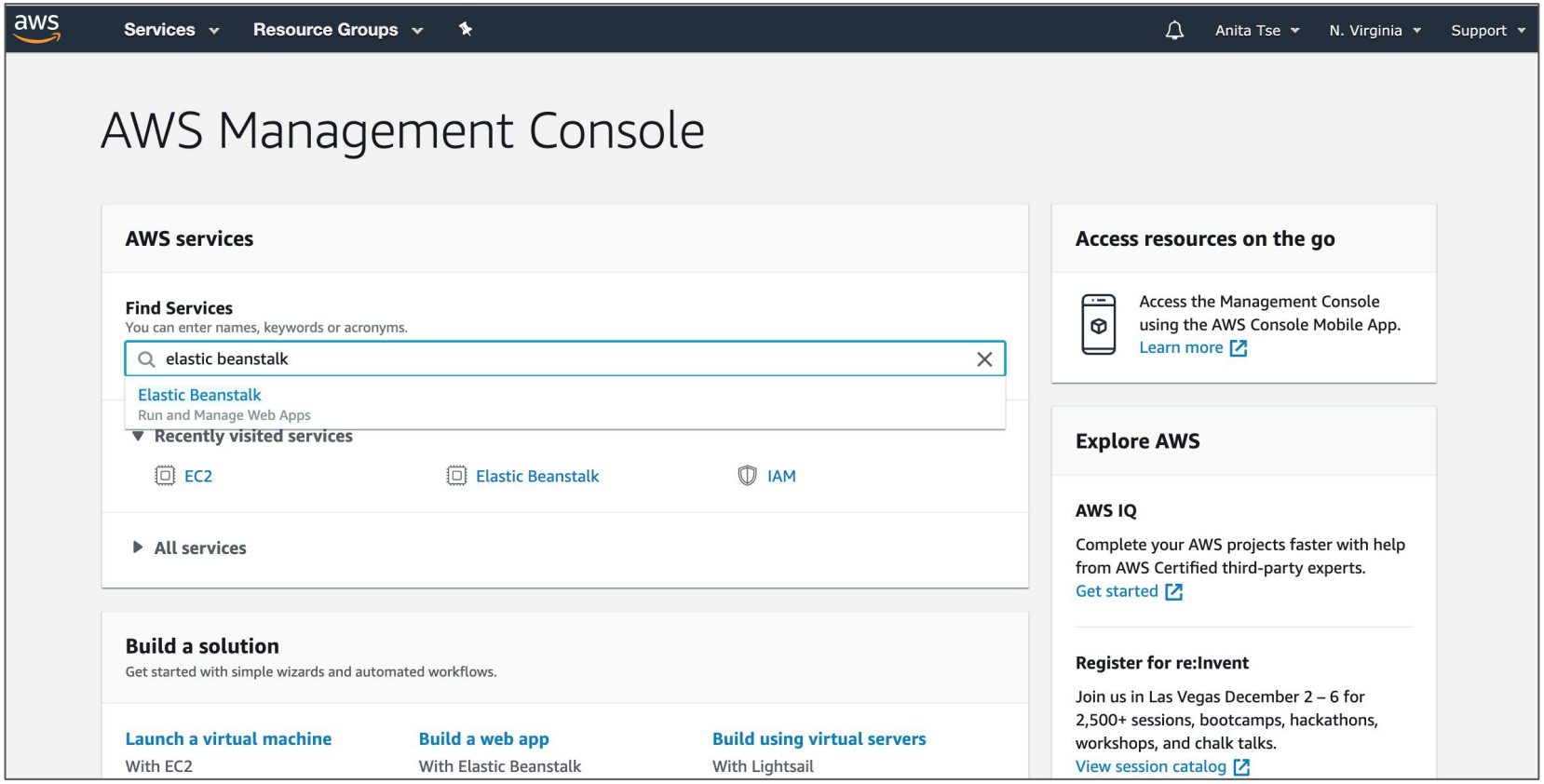
Log into the AWS Console

console.aws.amazon.com

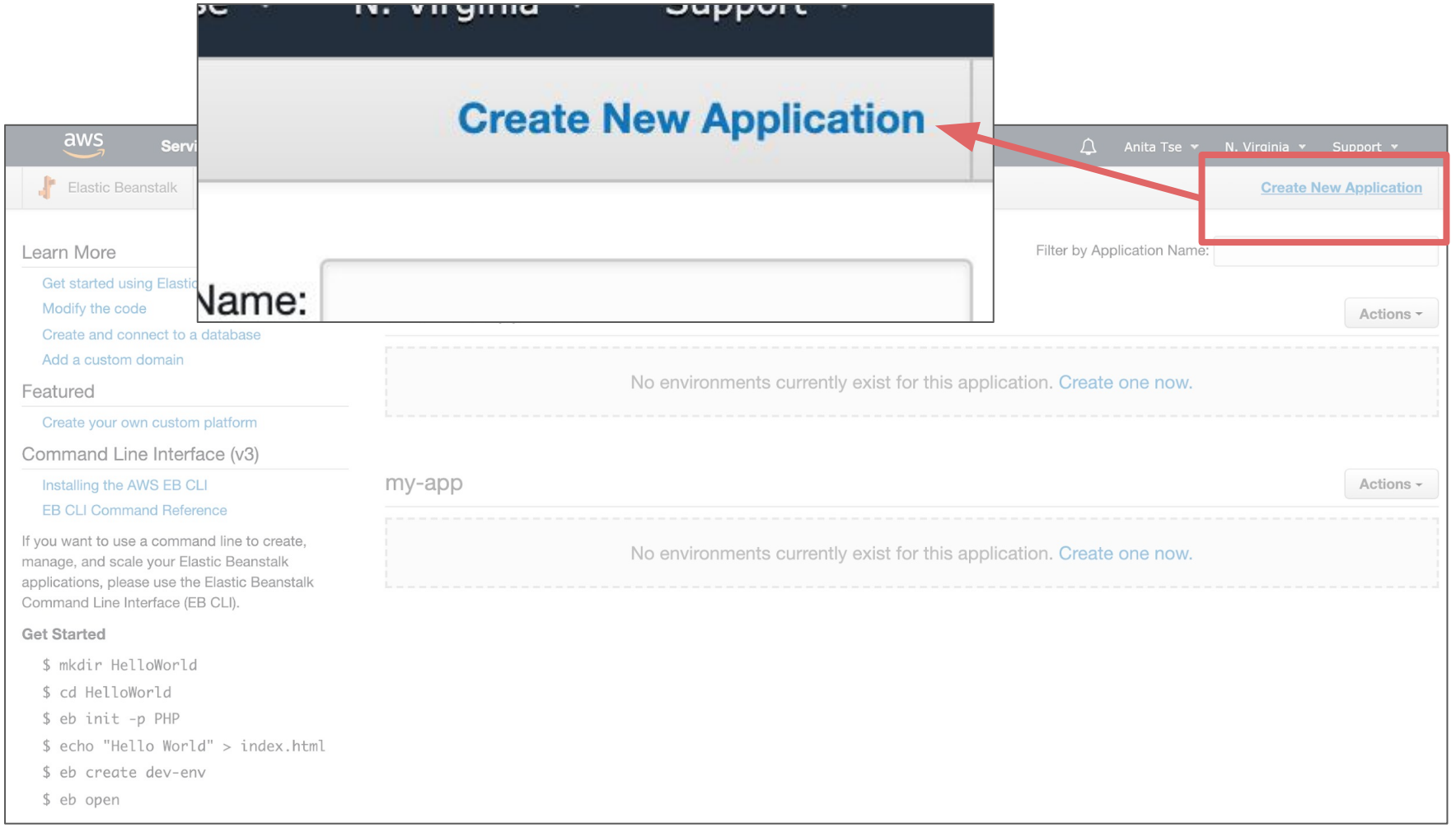


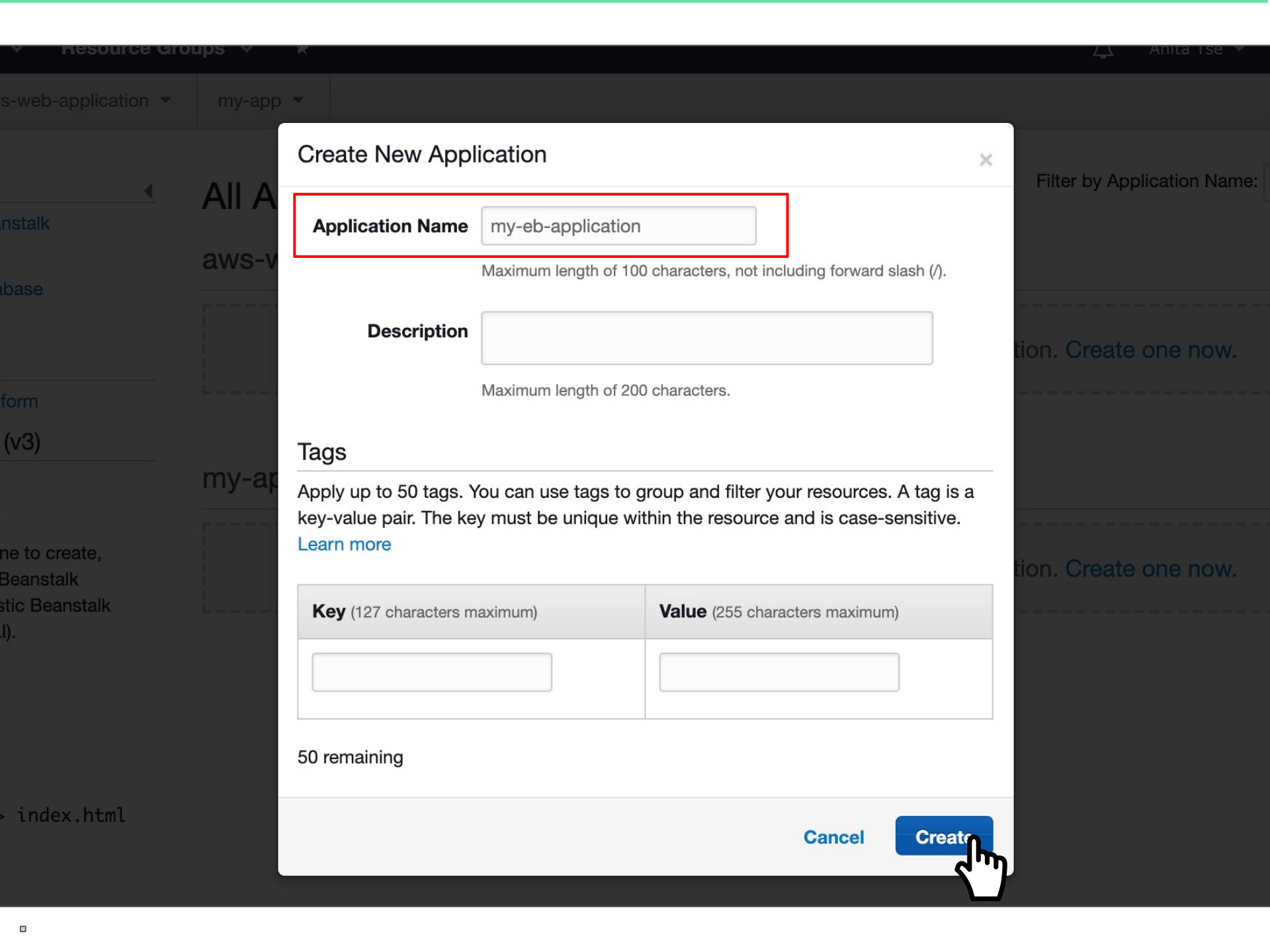
Navigate to the EB Console

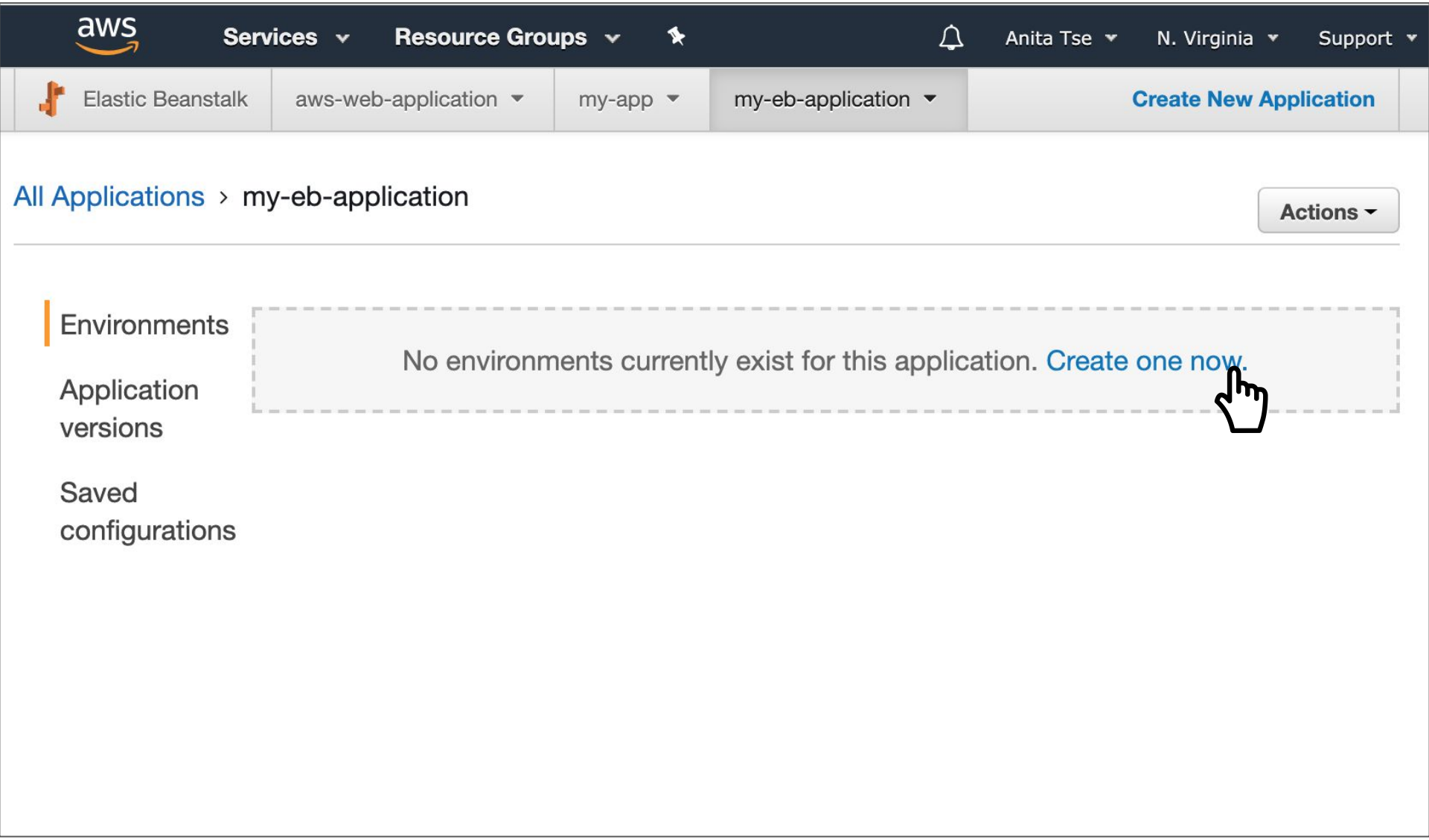
Type out “elastic beanstalk” in the search bar.

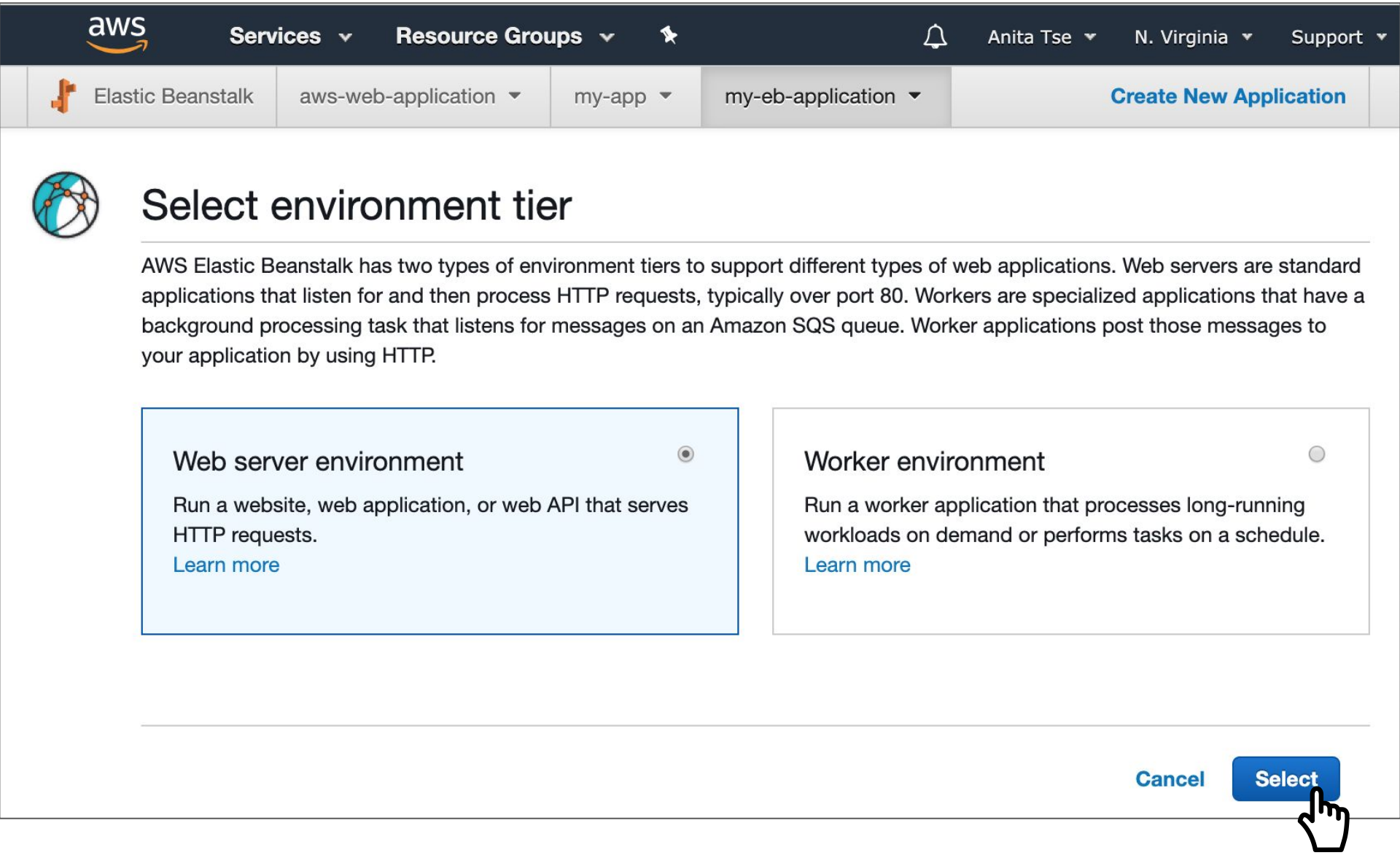


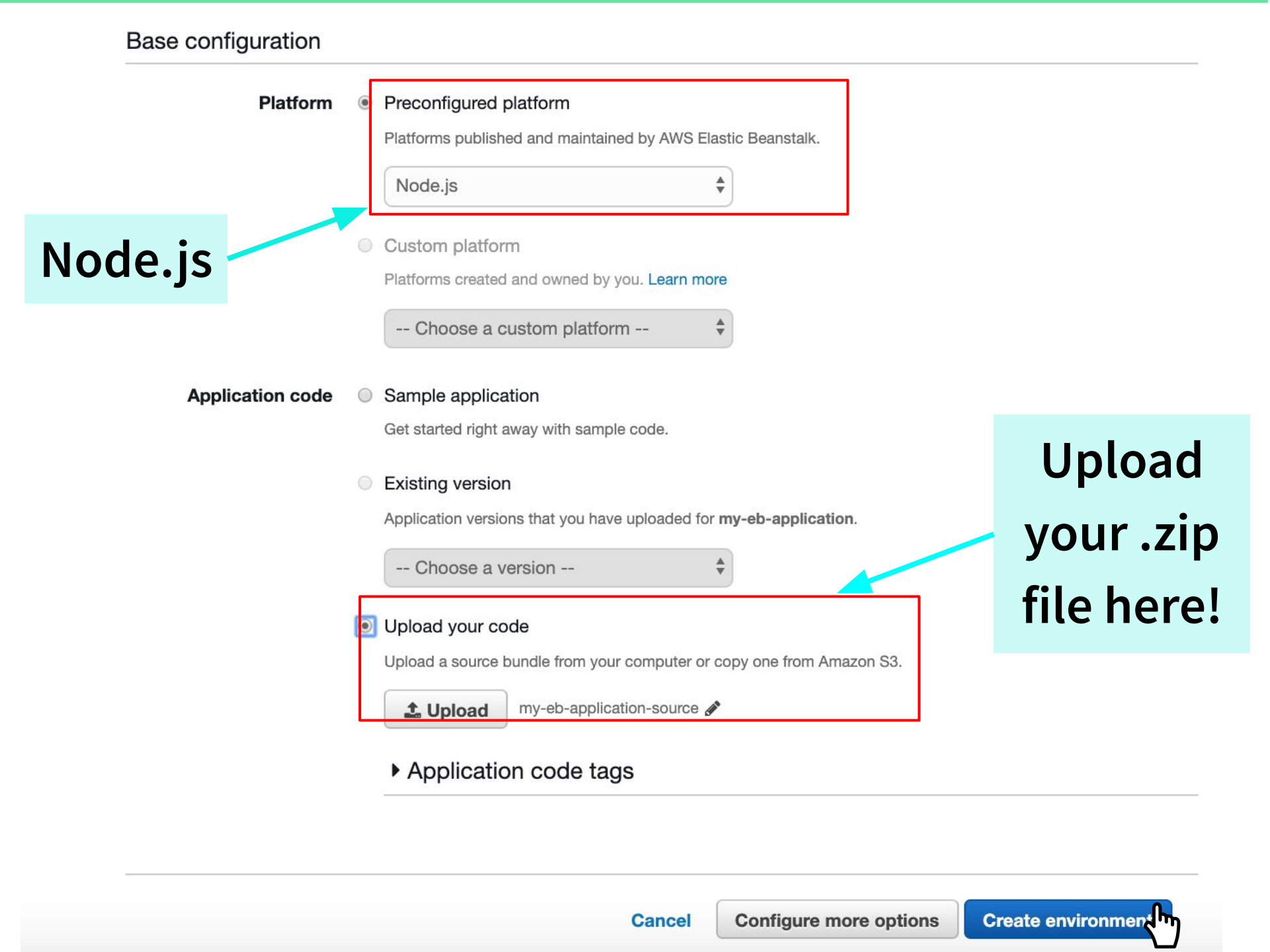
Create a new application



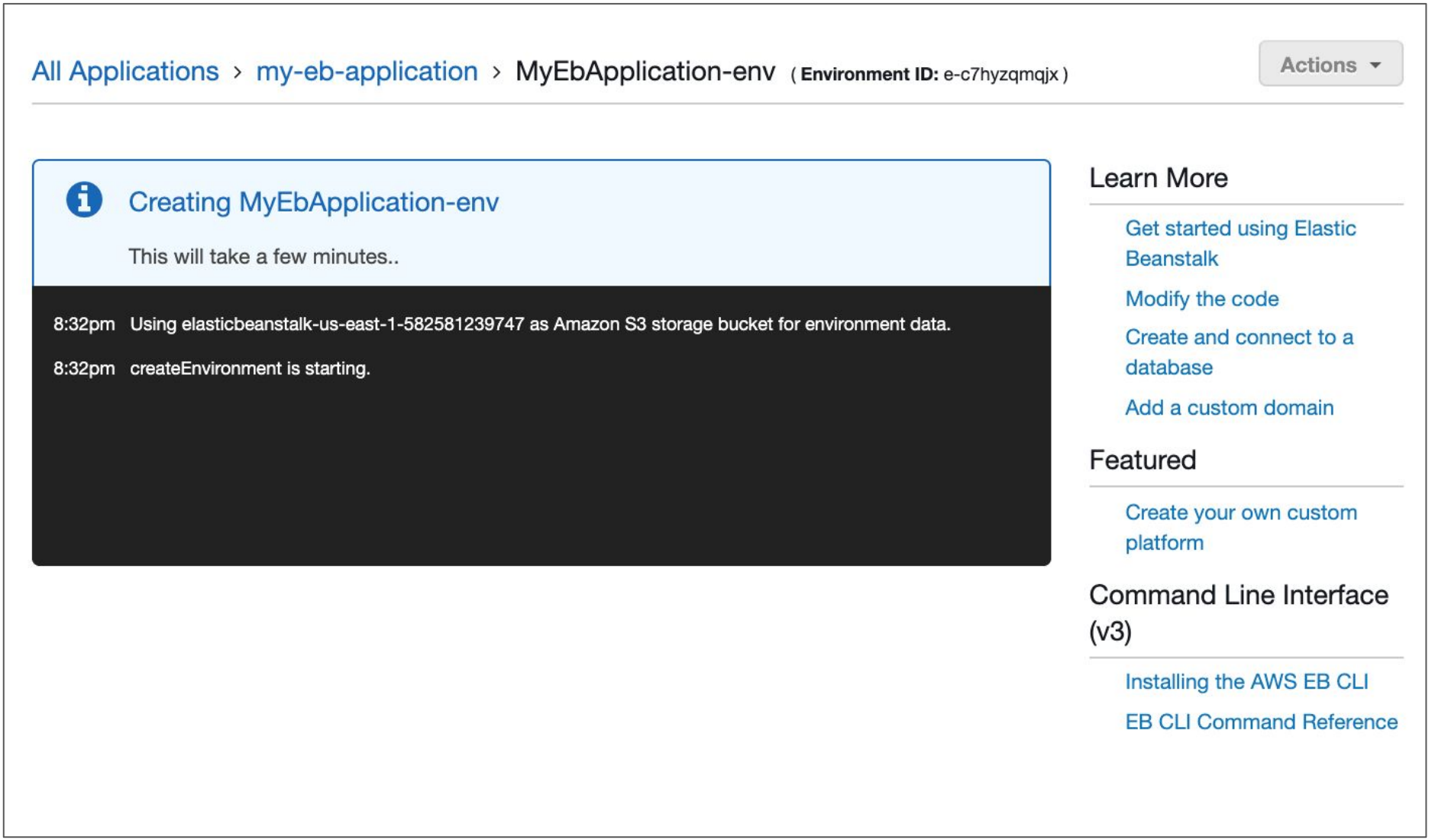

Create New Application
Application Name
my-eb-application
Maximum length of 100 characters, not including forward slash (/).
Description
Maximum length of 200 characters.
Tags
Apply up to 50 tags. You can use tags to group and filter your resources. A tag is a key-value pair. The key must be unique within the resource and is case-sensitive.
Learn more
Key (127 characters maximum)
Value (255 characters maximum)
50 remaining
Cancel
Create
Filter by Application Name:
index.html


aws
Services
Resource Groups
Anita Tse
N. Virginia
Support
Elastic Beanstalk
aws-web-application
my-app
my-eb-application
Create New Application
All Applications
my-eb-application
Actions
Environments
Application versions
Saved configurations
No environments currently exist for this application. Create one now.

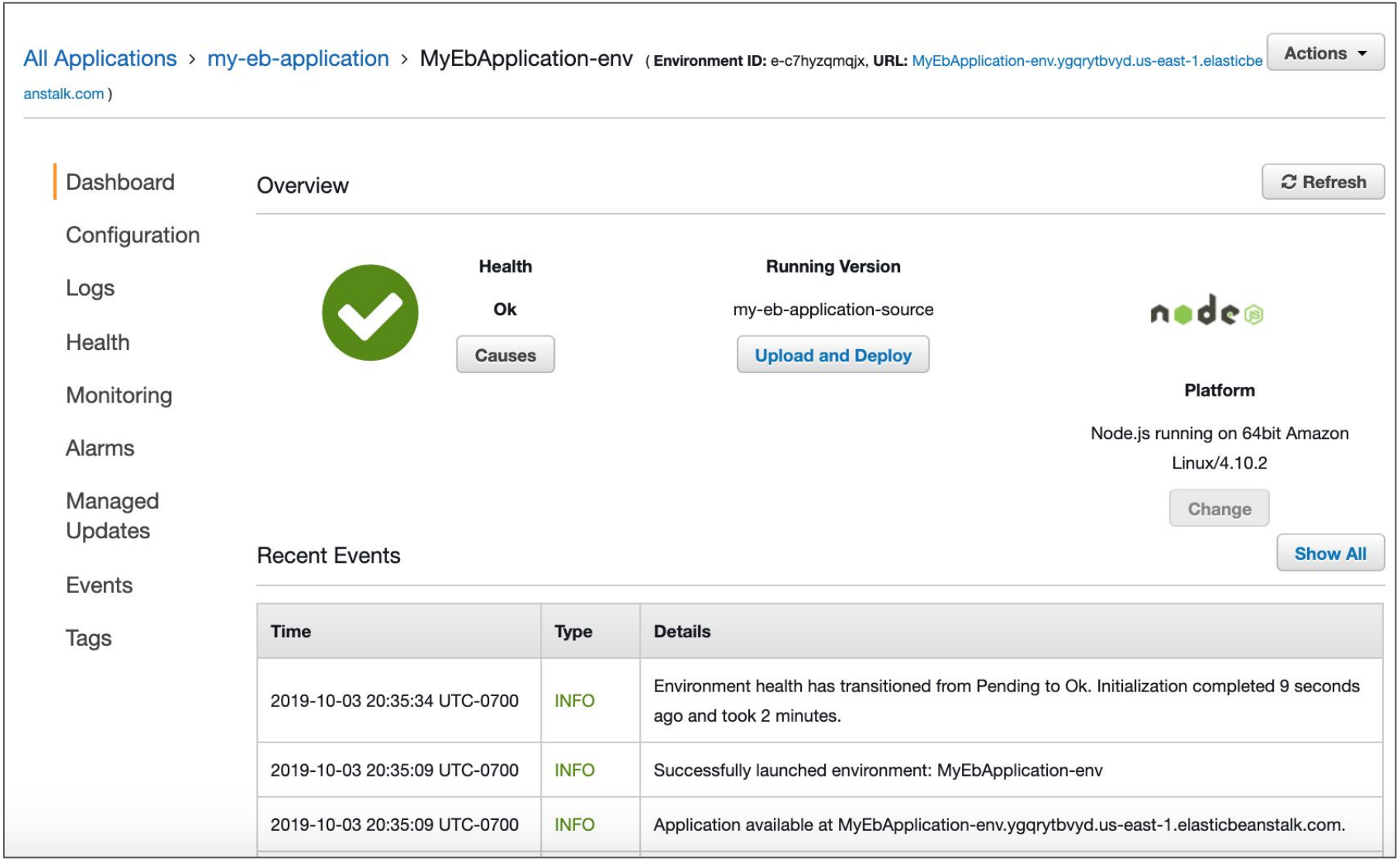

aws
Services
Resource Groups
Anita Tse
N. Virginia
Support
Elastic Beanstalk
aws-web-application
my-app
my-eb-application
Create New Application
Select environment tier
AWS Elastic Beanstalk has two types of environment tiers to support different types of web applications. Web servers are standard applications that listen for and then process HTTP requests, typically over port 80. Workers are specialized applications that have a background processing task that listens for messages on an Amazon SQS queue. Worker applications post those messages to your application by using HTTP.
Web server environment
Run a website, web application, or web API that serves HTTP requests.
Learn more
Worker environment
Run a worker application that processes long-running workloads on demand or performs tasks on a schedule.
Learn more
Cancel
Select


Base configuration
Platform
Preconfigured platform
Platforms published and maintained by AWS Elastic Beanstalk.
Node.js
Node.js
Custom platform
Platforms created and owned by you. Learn more
-- Choose a custom platform --
Application code
Sample application
Get started right away with sample code.
Existing version
Application versions that you have uploaded for my-eb-application.
-- Choose a version --
Upload your code
Upload a source bundle from your computer or copy one from Amazon S3.
Upload
my-eb-application-source
Upload your .zip file here!
Application code tags
Cancel
Configure more options
Create environment

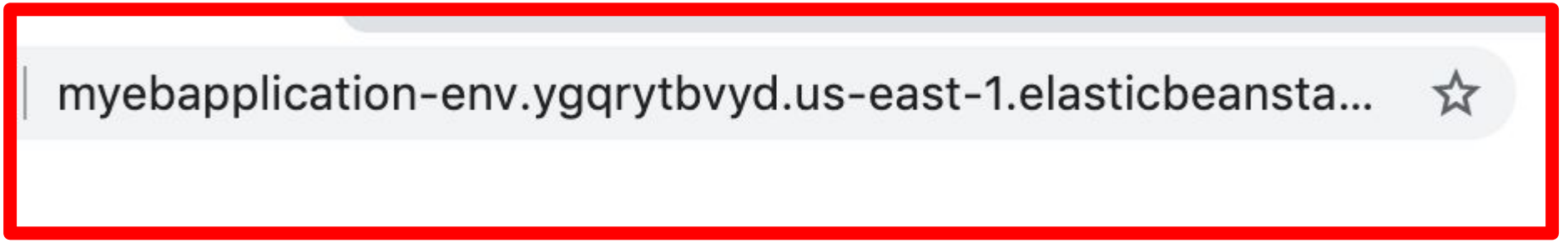
Watch your project deploy!



Click on the link to see your project live!



Notice that the address is no longer from your local machine



Try sharing the link to your friends and see if they can view your app!

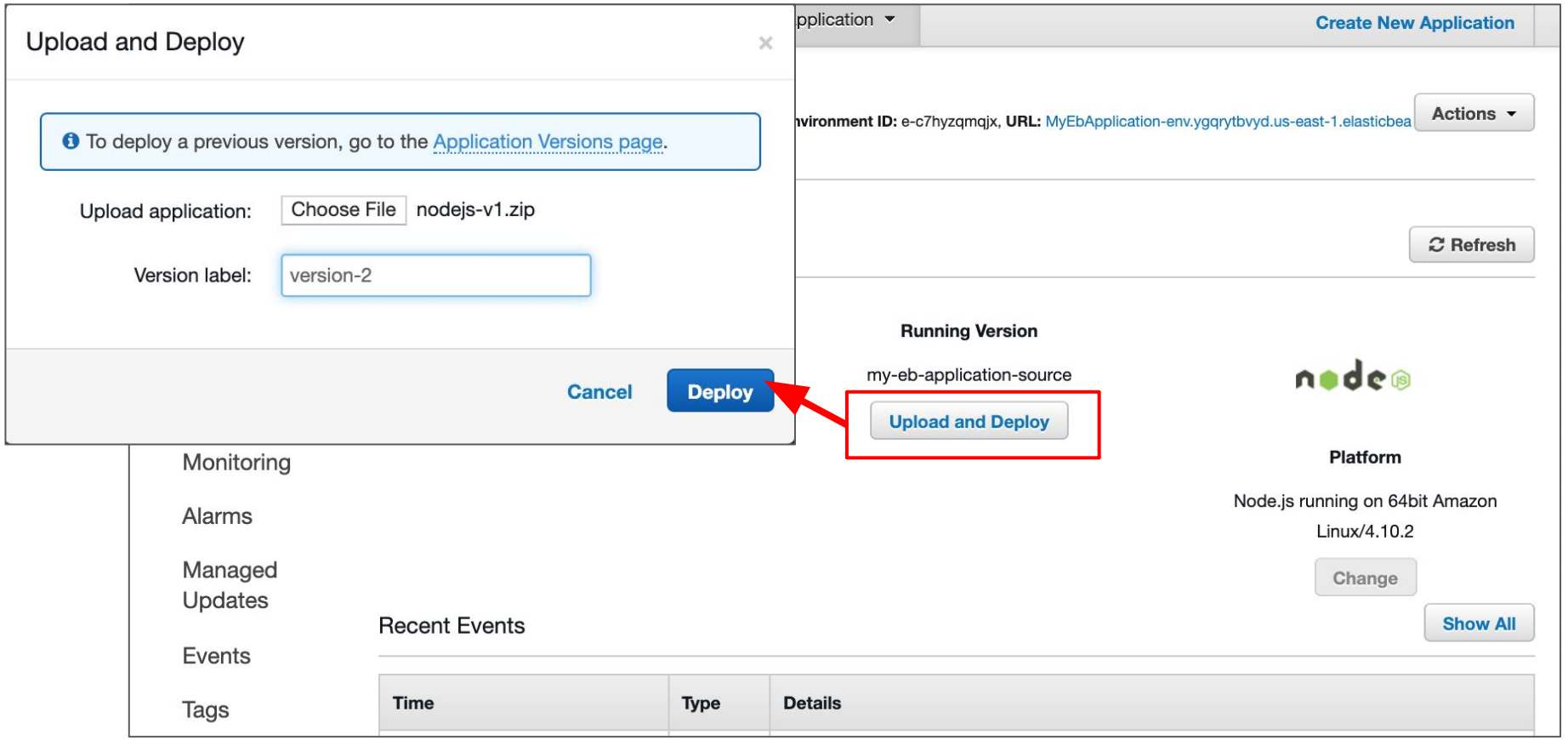
Redeploying in the future

If you make new changes to your code, you will need to **redeploy** your changes so they will reflect on your live link (Production server).

Just zip up your updated code and upload it to the Elastic Beanstalk console again!

Redeploying in the future

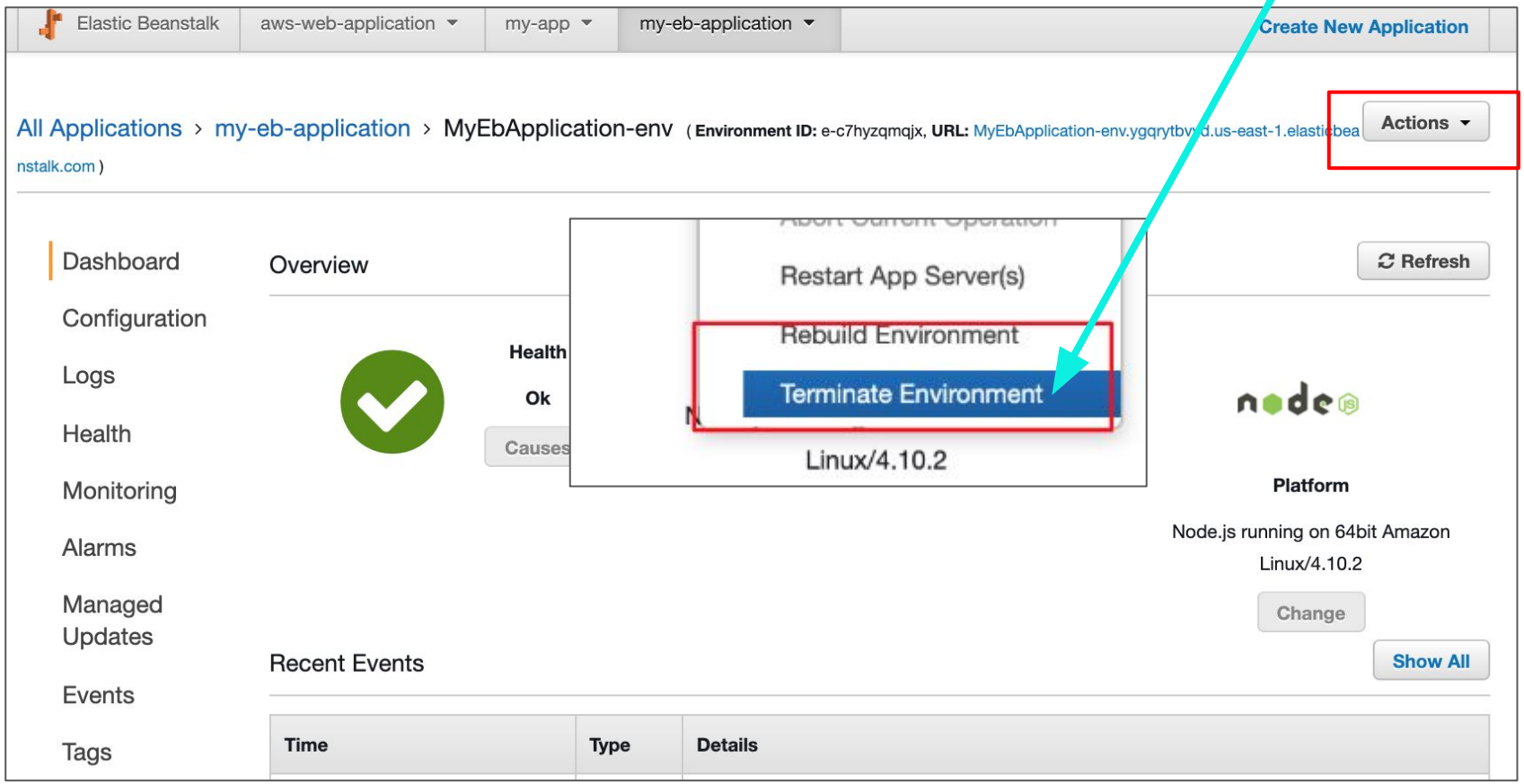
Upload your code and give your version a name → the changes will be seen shortly!



Cleaning up

To save costs, be sure to Terminate your environment when you are finished

Actions → Terminate Environment



In the future: EB CLI (Command Line Interface)

A Command Line Interface allows you to set up and perform Elastic Beanstalk operations **directly on your Terminal** for a faster and more streamlined deployment!

```
$  eb create
```

```
$  eb deploy
```

You did it! You learned:

1. How to create a web app with Express and NodeJS
    - Modern web development practices
    - Integrating an external API into your web app
    - Considerations for building user interfaces
2. How to deploy a web app with AWS
    - Introduction to AWS
    - Deploying with the Elastic Beanstalk console

nwPlus

Next steps

1. Take a look at the stretch goals (slide 51)
2. Review the concepts we didn’t cover (slide 4)
3. Check out the EB CLI (Slide 73)
4. Come to Build Day!

nwPlus

nwPlus

aws educate

Build and Deploy a Full Stack Web App with AWS

Learn Day 2019
#workshop-aws