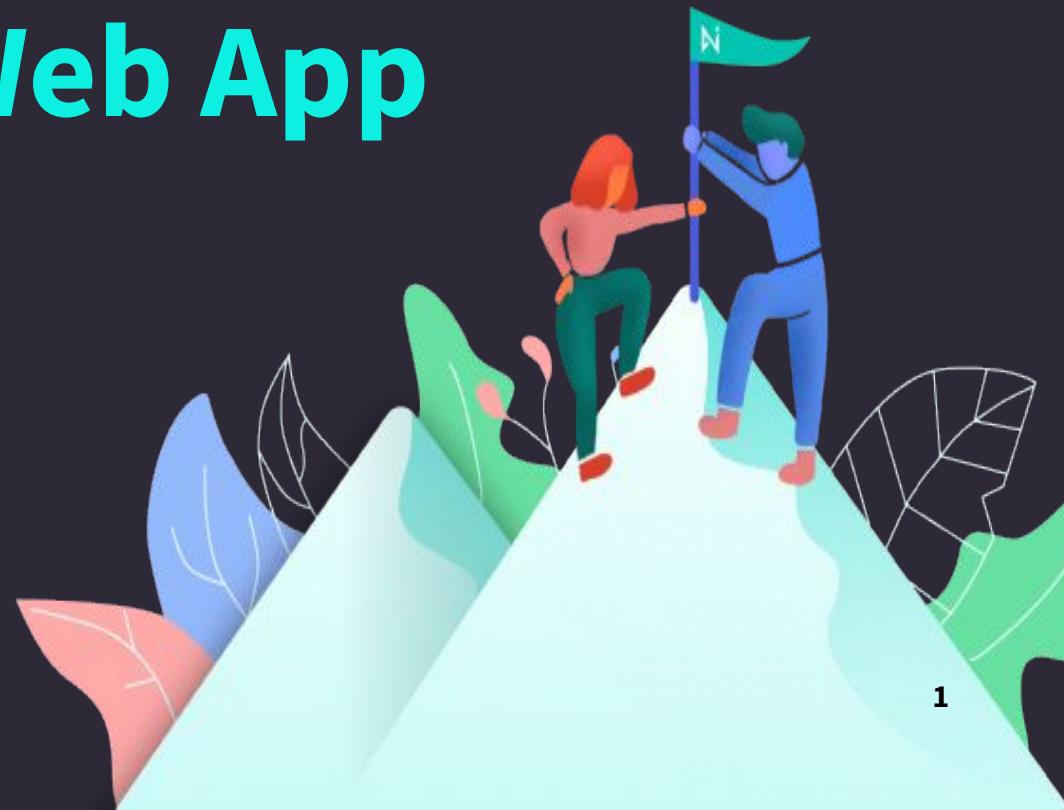




Build and Deploy a Full Stack Web App with AWS

Learn Day 2019
#workshop-aws



Hey, it's Anita & Chris



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



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)

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**

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

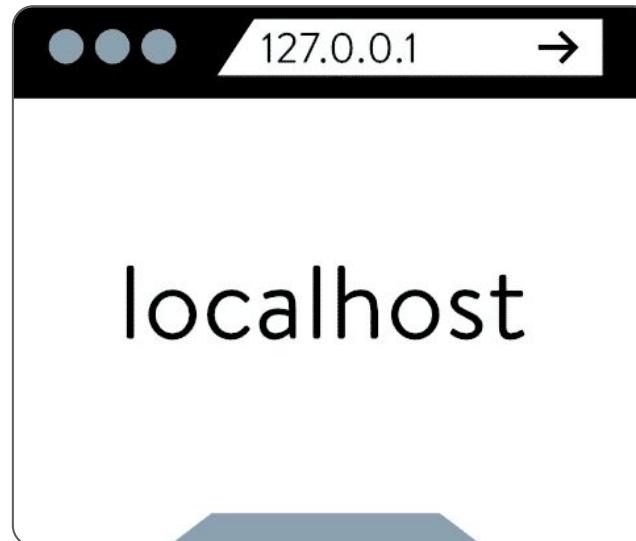
- The “brain” of the application
 - ex. Retrieving information, reading inputs and button presses

3. (Not included) Database

- Where data is stored and organized

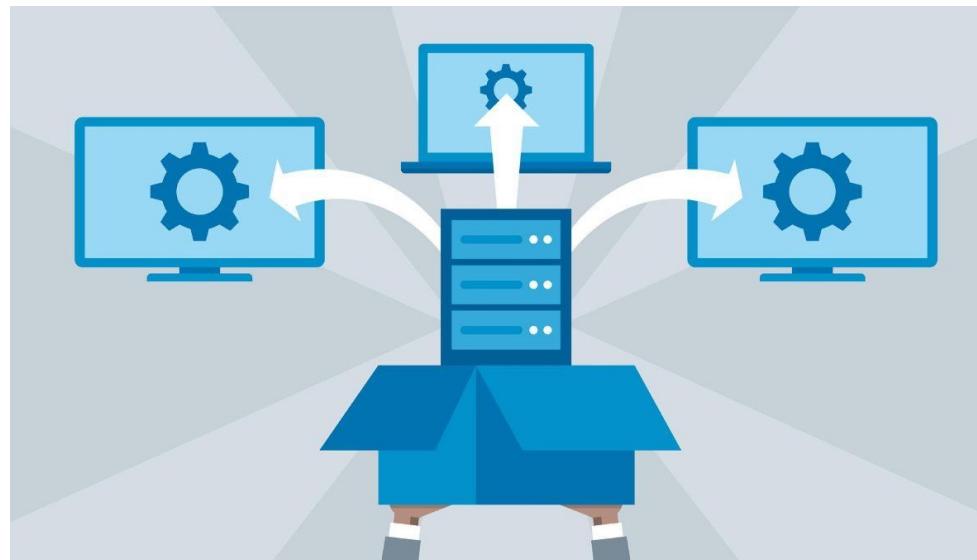
Development vs. Production

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

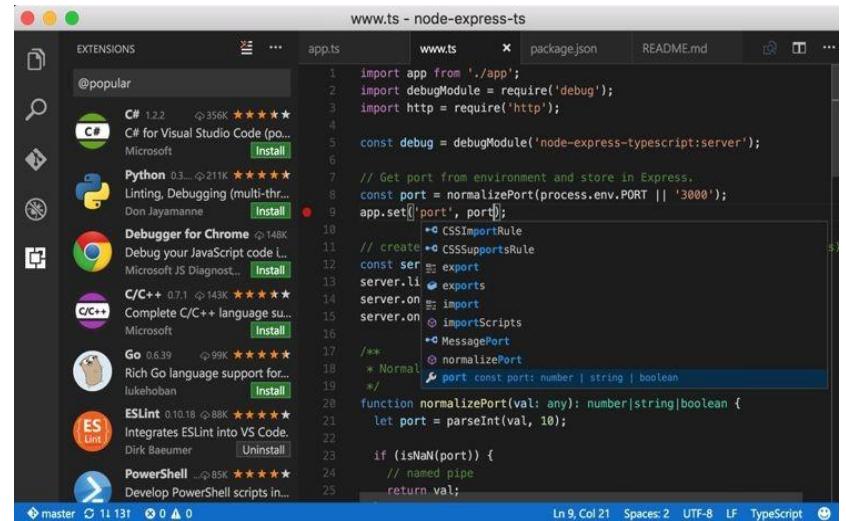
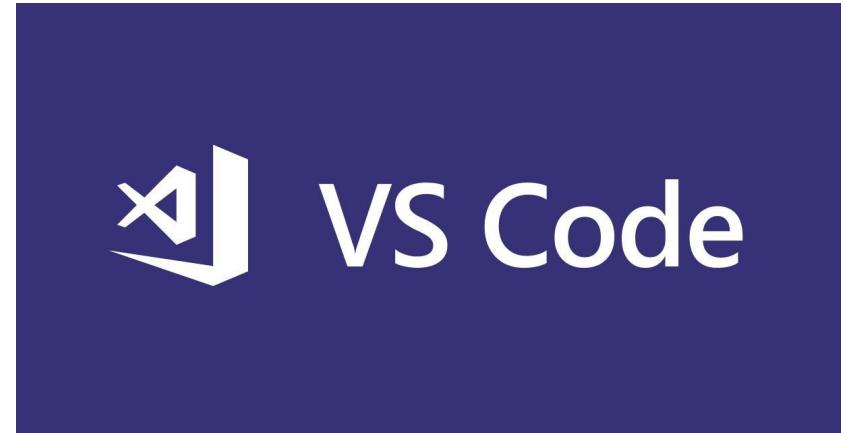
VS Code and NodeJS



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!

A screenshot of the Visual Studio Code interface. The left sidebar shows the 'EXTENSIONS' view with popular extensions like C#, Python, and ESLint listed. The main editor area shows a TypeScript file named 'app.ts' with code for a node-express application. The status bar at the bottom indicates the file is 'www.ts - node-express-ts', the line is 'Ln 9, Col 21', and the file type is 'TypeScript'.

```
1 import app from './app';
2 import debugModule = require('debug');
3 import http = require('http');
4
5 const debug = debugModule('node-express-typescript:server');
6
7 // Get port from environment and store in Express.
8 const port = normalizePort(process.env.PORT || '3000');
9 app.set('port', port);
10
11 // Create CSSImportRule
12 const ser = export
13 server.on import
14 server.on importScripts
15
16 /**
17 * Normal
18 */
19
20 function normalizePort(val: any): number|string|boolean {
21   let port = parseInt(val, 10);
22
23   if (isNaN(port)) {
24     // Named pipe
25     return val;
26   }
27
28   return port;
29 }
```

Install Visual Studio Code

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

code.visualstudio.com/download

Download Visual Studio Code
Free and built on open source. Integrated Git, debugging and extensions.



[!\[\]\(9eba544be39bf7f9cda8b3ecca70ec7d_img.jpg\) Windows](#)
Windows 7, 8, 10

[!\[\]\(1e81a34f224bff3b9263f85ff328a404_img.jpg\) .deb](#)
Debian, Ubuntu

[!\[\]\(5d54376ac18574f003f3b136ec217d16_img.jpg\) .rpm](#)
Red Hat, Fedora, SUSE

[!\[\]\(8cac9fd4e0667387138a70388fb31604_img.jpg\) Mac](#)
macOS 10.10+

User Installer	64 bit	32 bit
zip	64 bit	32 bit
Installer	64 bit	32 bit
	64 bit	32 bit

[!\[\]\(be24e1e01a1162d62bfa8542d88738d4_img.jpg\) .deb](#)
64 bit

[!\[\]\(b2d80df4e0956835de73688fb1277071_img.jpg\) .rpm](#)
64 bit

[!\[\]\(d3fbc9fcb3287b6ec3186b1b1e7e21dc_img.jpg\) .tar.gz](#)
64 bit

[Snap Store](#)

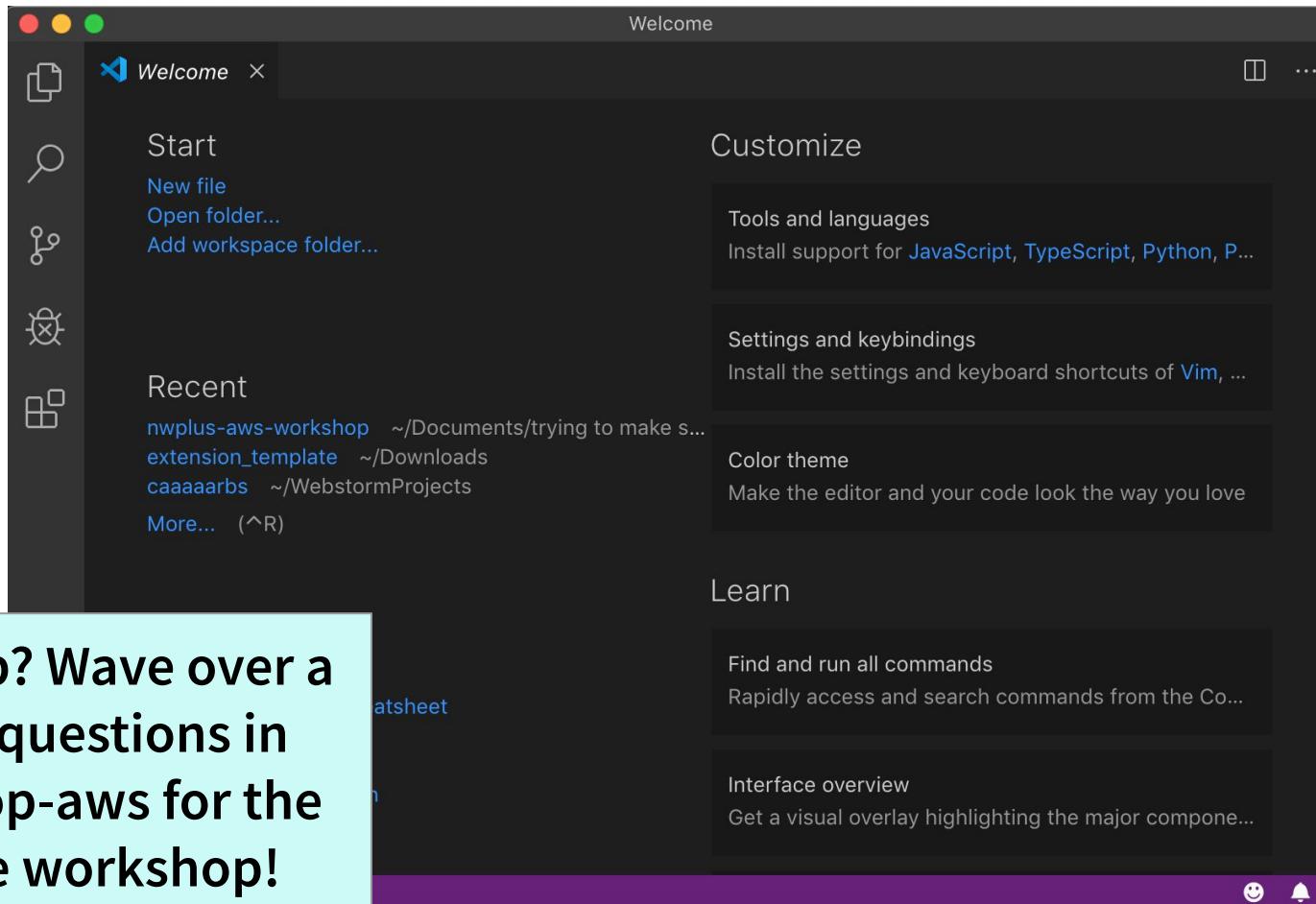
By downloading and using Visual Studio Code, you agree to the [license terms](#) and [privacy statement](#).

Want new features sooner?
Get the [Insiders build](#) instead.

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

Install Visual Studio Code

The program should look like this when opened:



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.

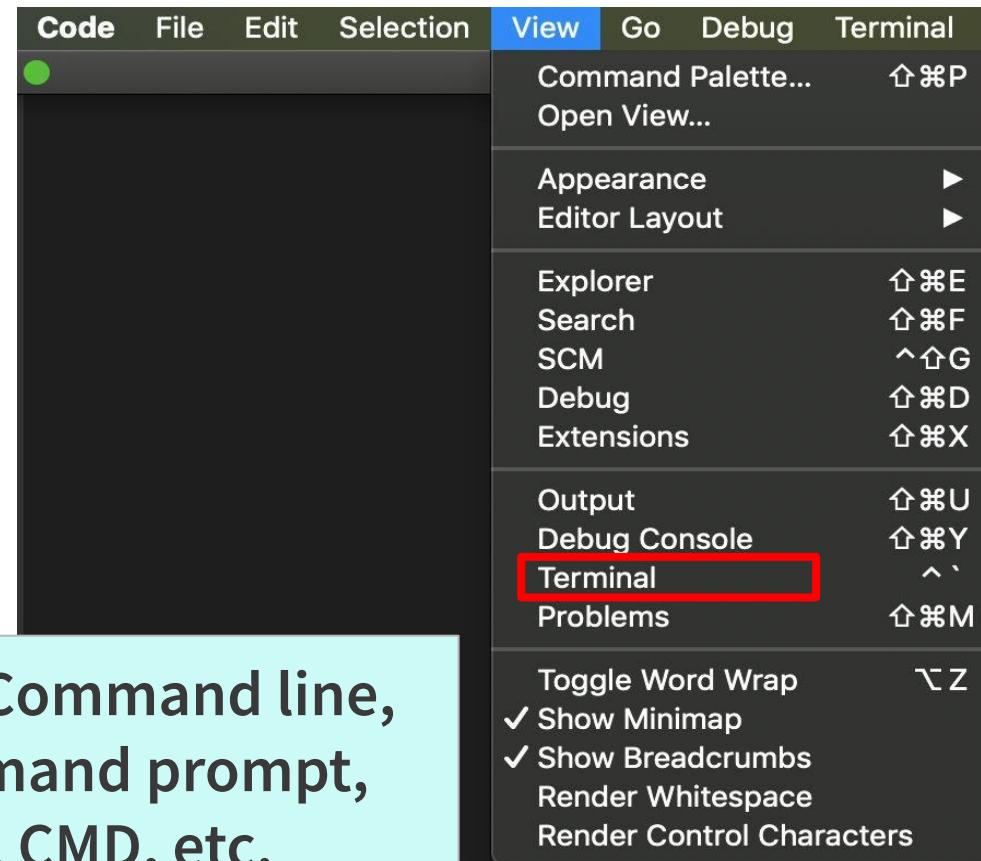
Check for Node.js- Introducing the terminal

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:

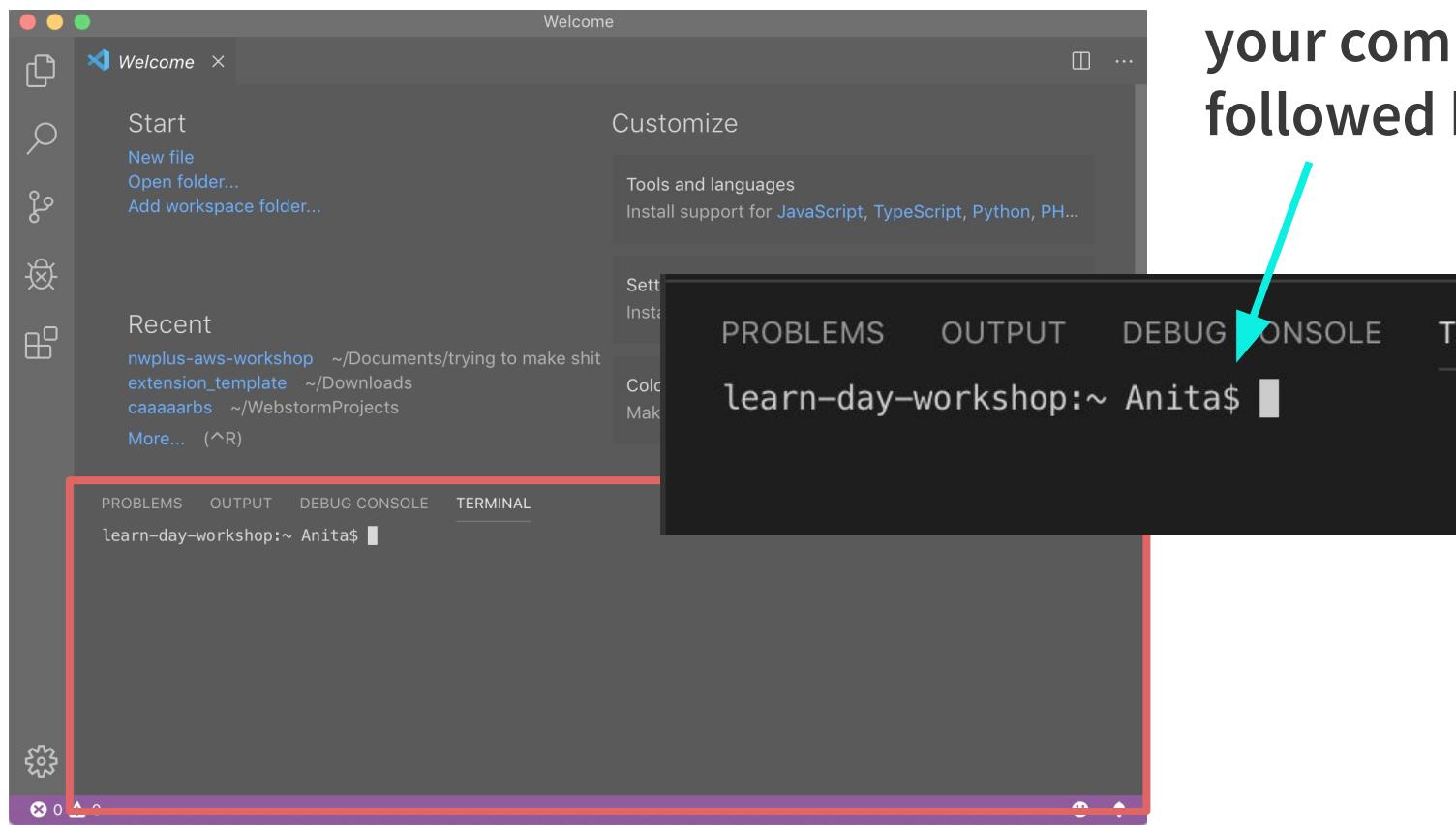
View > Terminal

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!

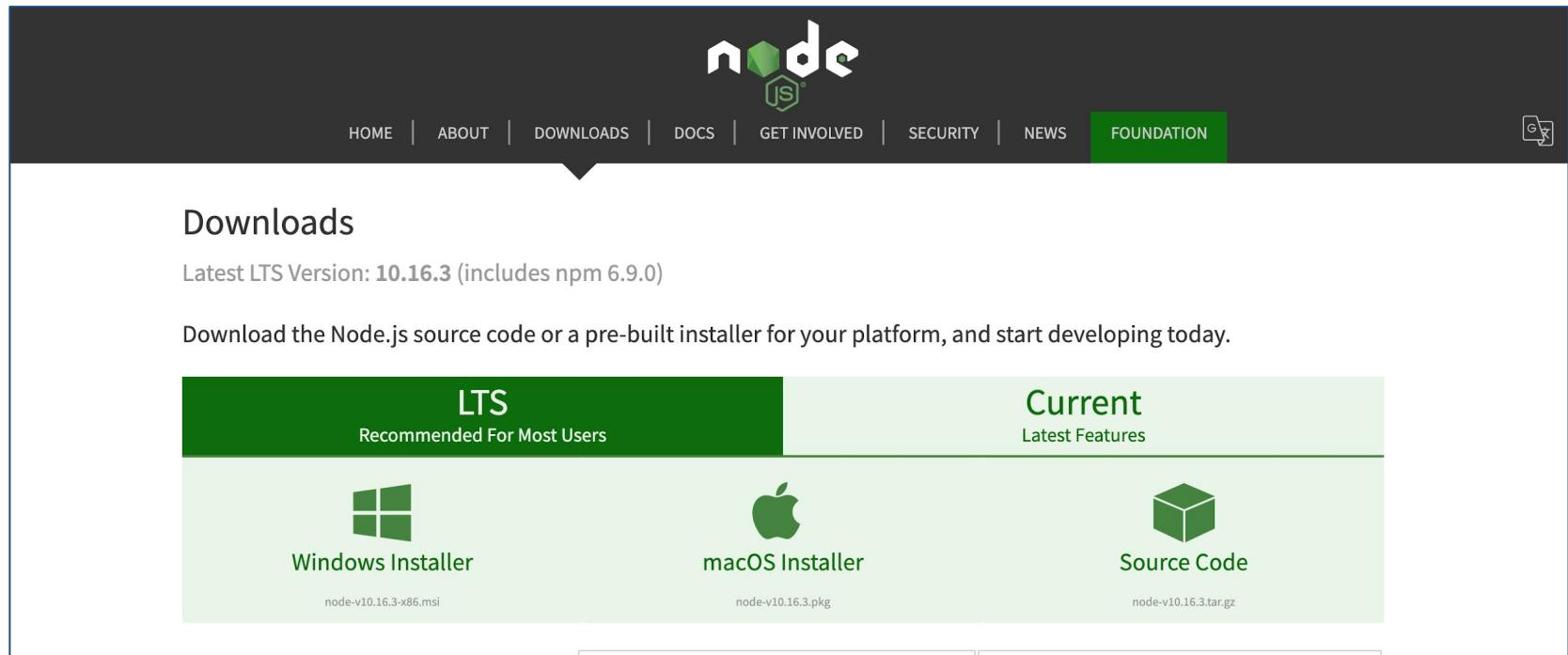
```
learn-day-workshop:~ Anita$ node --version  
v8.12.0
```

Installing Node.js

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

nodejs.org/en/download

version 8+



The screenshot shows the official Node.js website with a dark header. The header includes the Node.js logo, navigation links for HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, NEWS, and FOUNDATION, and a search icon. A green button labeled 'version 8+' is positioned to the right of the download link. Below the header, a section titled 'Downloads' is shown. It highlights the 'Latest LTS Version: 10.16.3 (includes npm 6.9.0)'. It then provides links for 'Windows Installer' (MSI file), 'macOS Installer' (PKG file), and 'Source Code' (tar.gz file).

nodejs.org/en/download

version 8+

Downloads

Latest LTS Version: 10.16.3 (includes npm 6.9.0)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS
Recommended For Most Users

Current
Latest Features

Windows Installer

macOS Installer

Source Code

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!

Web App Development

With NodeJS and Express



What we'll be building

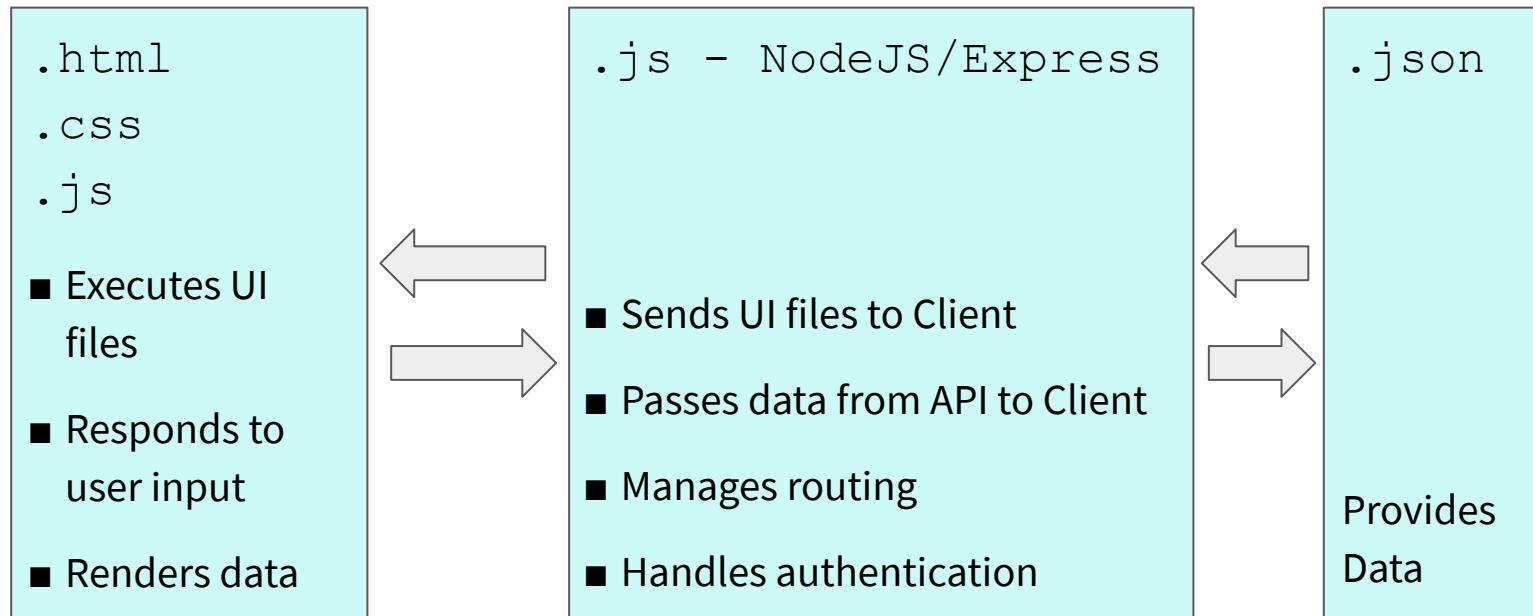
What's the weather in Toronto ?

Clear Sky

[See example here](#)



What we'll be building



Our device
“Client”
“Frontend”

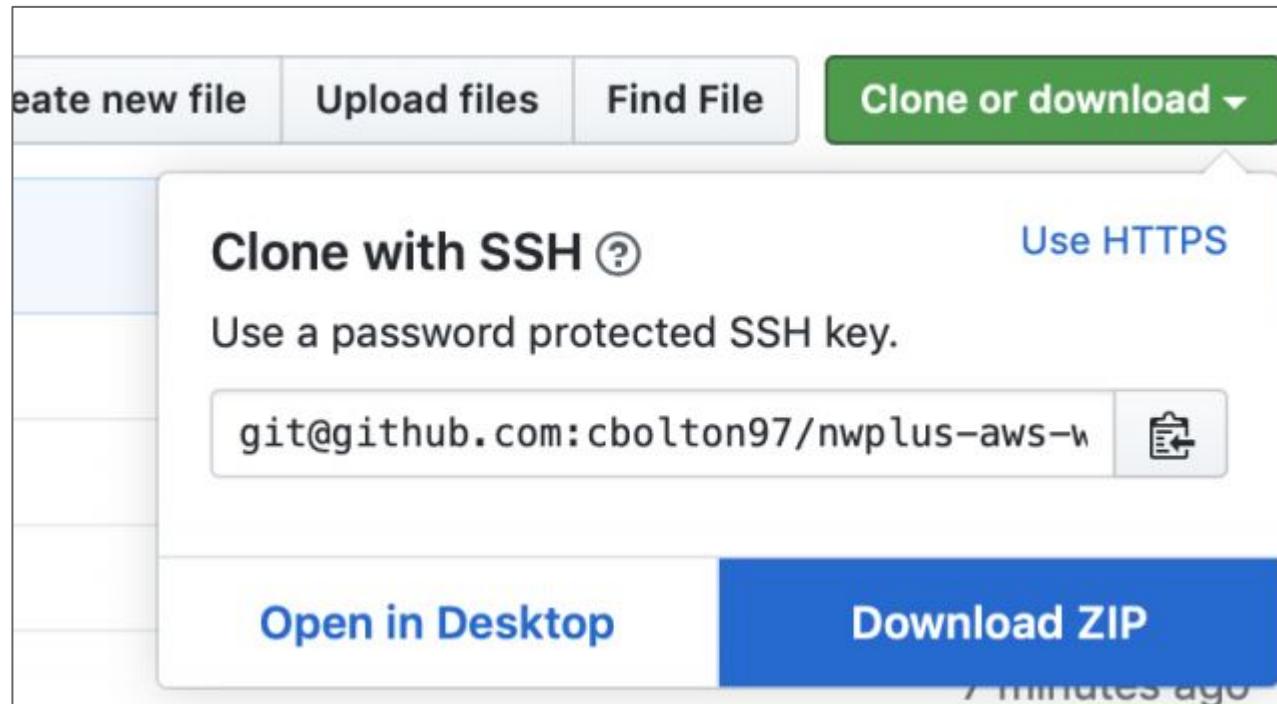
Our server
“Server”
“Backend”

External data
“API”



Download the starter code

<http://tiny.cc/lhdaws0>



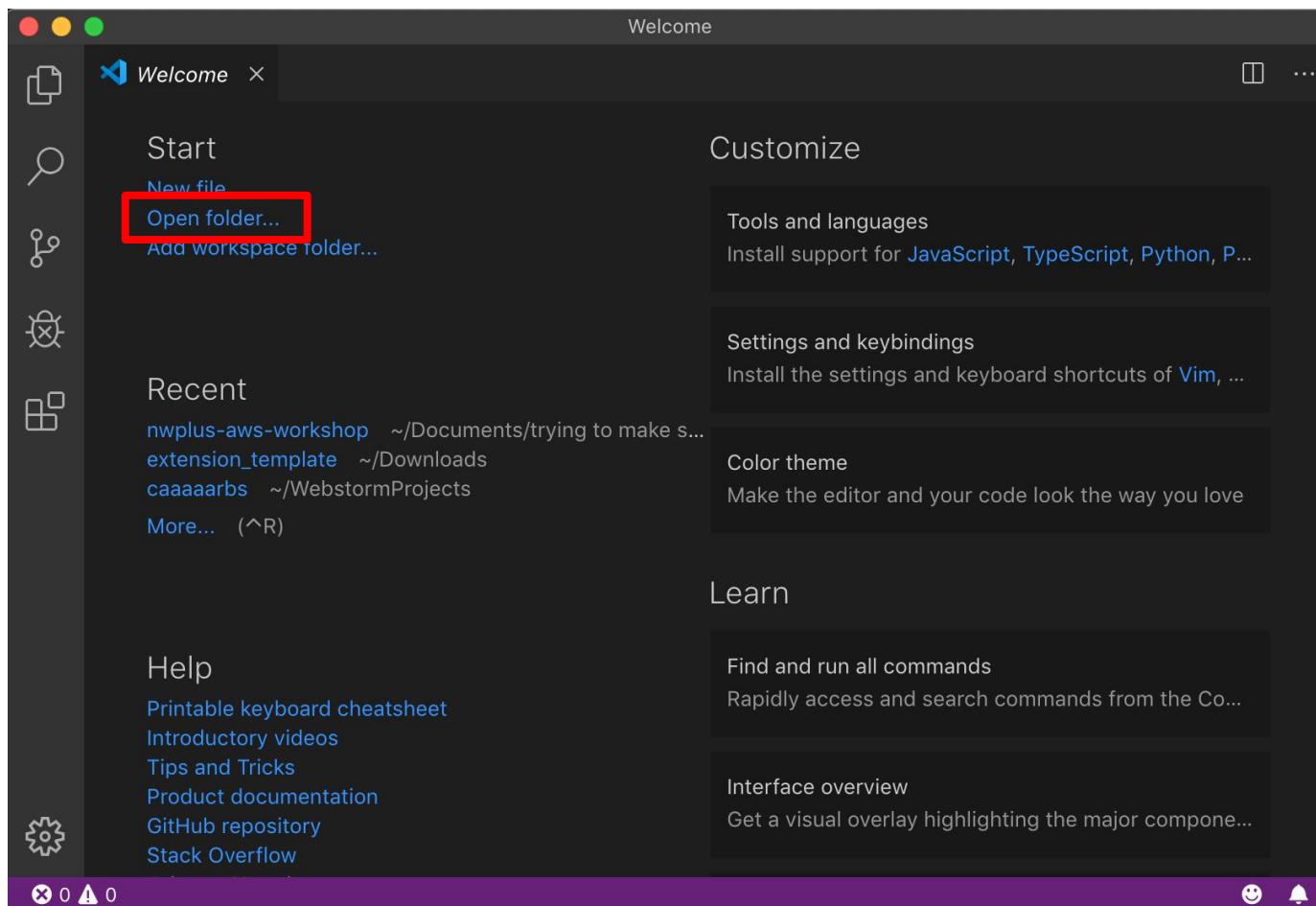
A note on tech choices

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



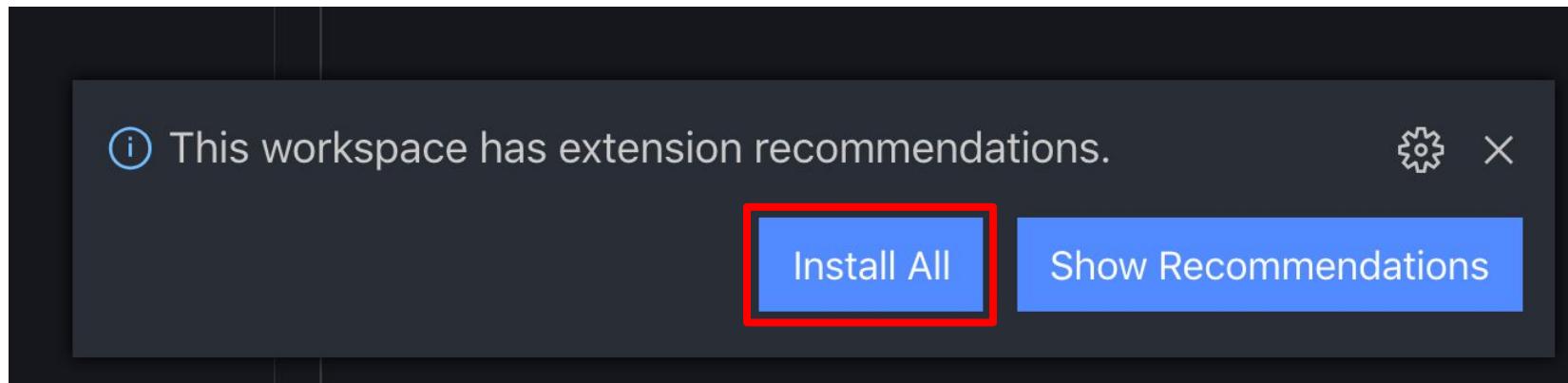
Opening our project

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



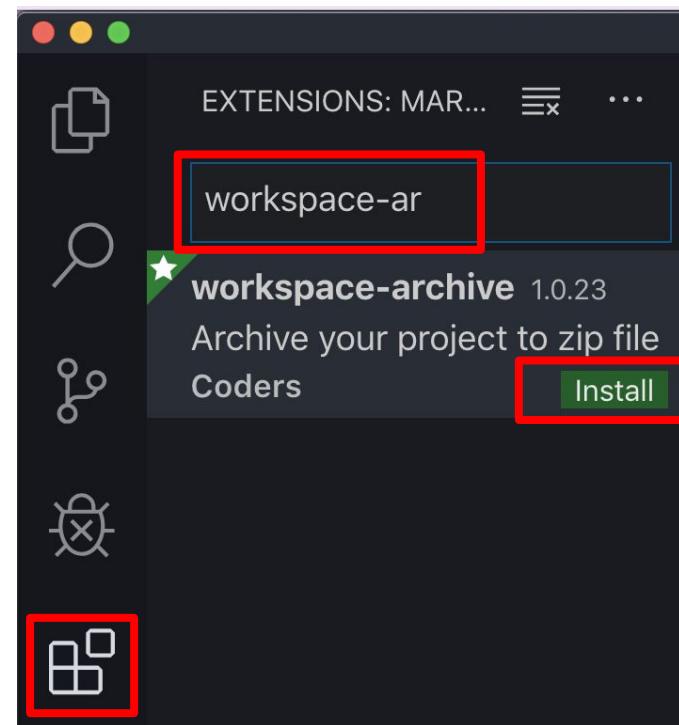
Install the Workspace Archive extension

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



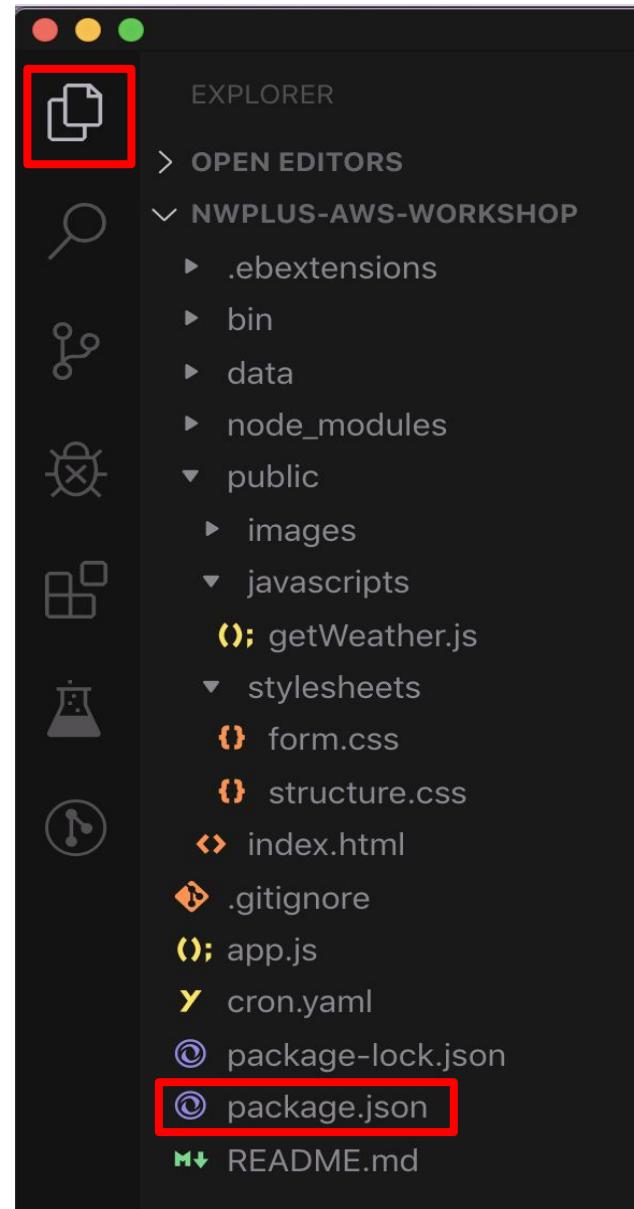
If you don't get prompted to install the extension:

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"  
},
```



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!

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
```

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:

What's the weather in city name... [Go ?](#)



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!



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)

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)

Show the temperature

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

What's the weather in [Vancouver](#) ?

36.33°C Super Sunny

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

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>

Create New Account

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:

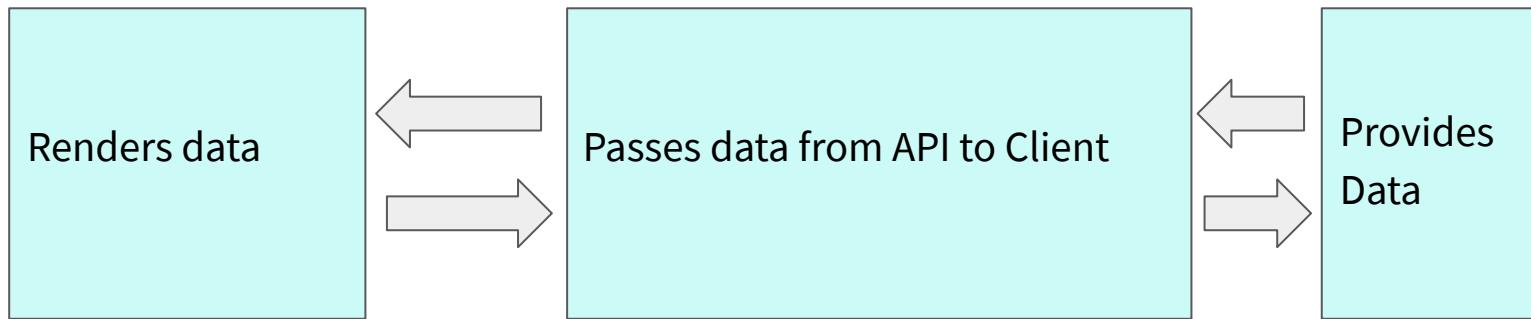
```
var API_KEY = "Enter your API Key here";
```

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.



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



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?

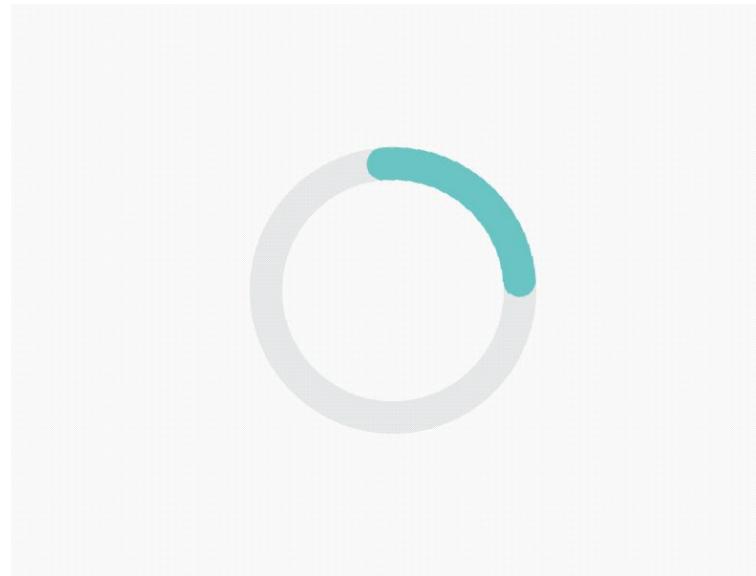
In general, we don't want to assume anything about the reliability of external APIs.

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.



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:

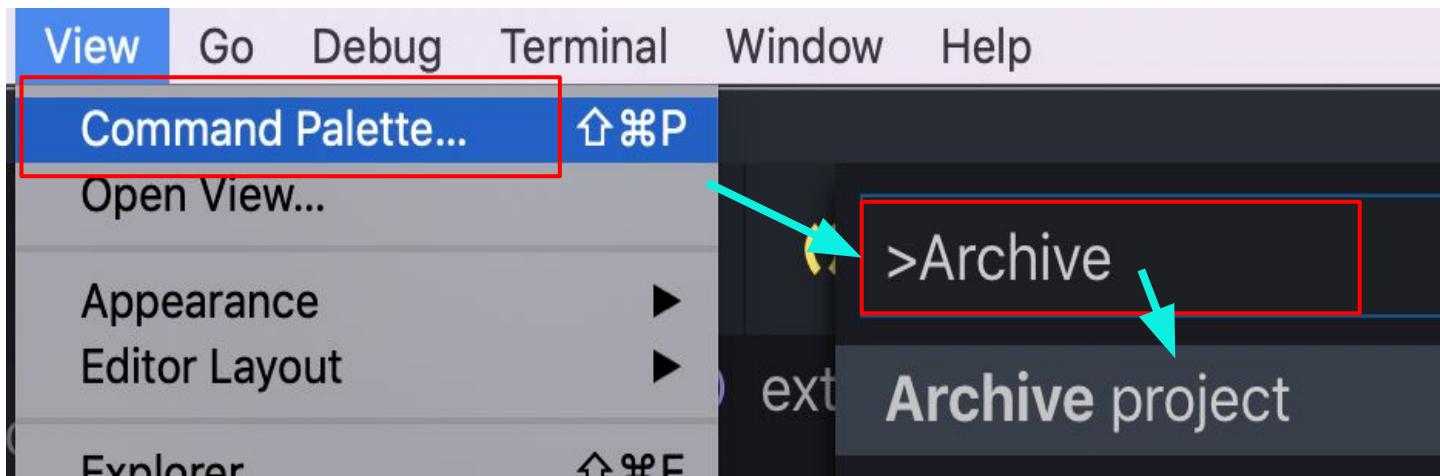
<http://tiny.cc/lhdaws3>

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

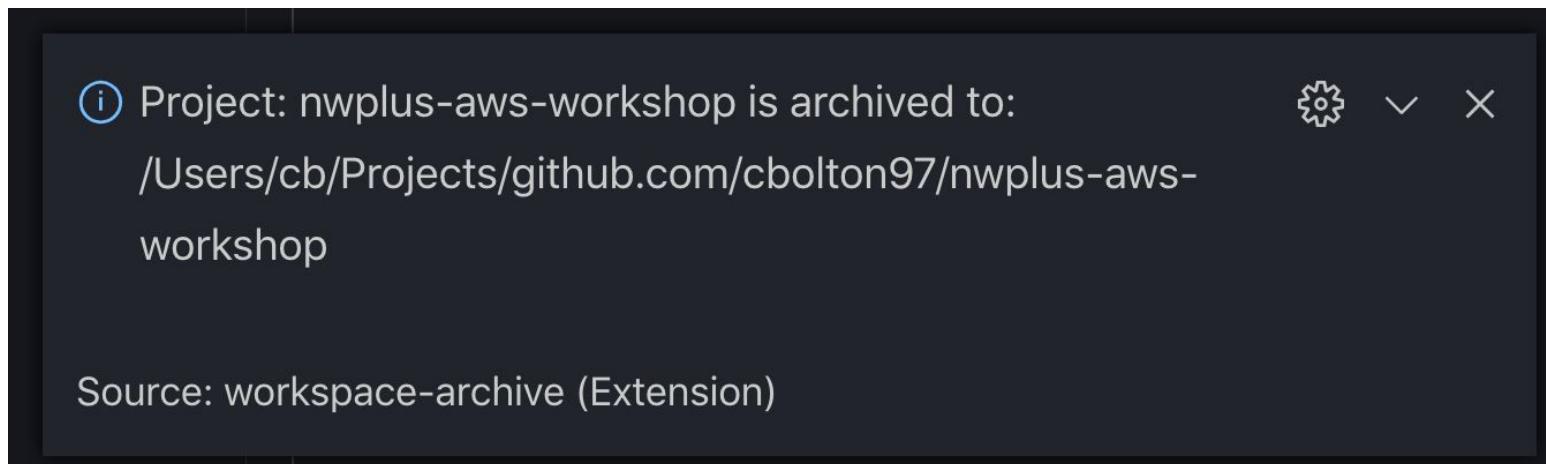
Now we're ready to compress our code for deployment

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:



Deployment with AWS

AWS Elastic Beanstalk



What is Amazon Web Services (AWS)?

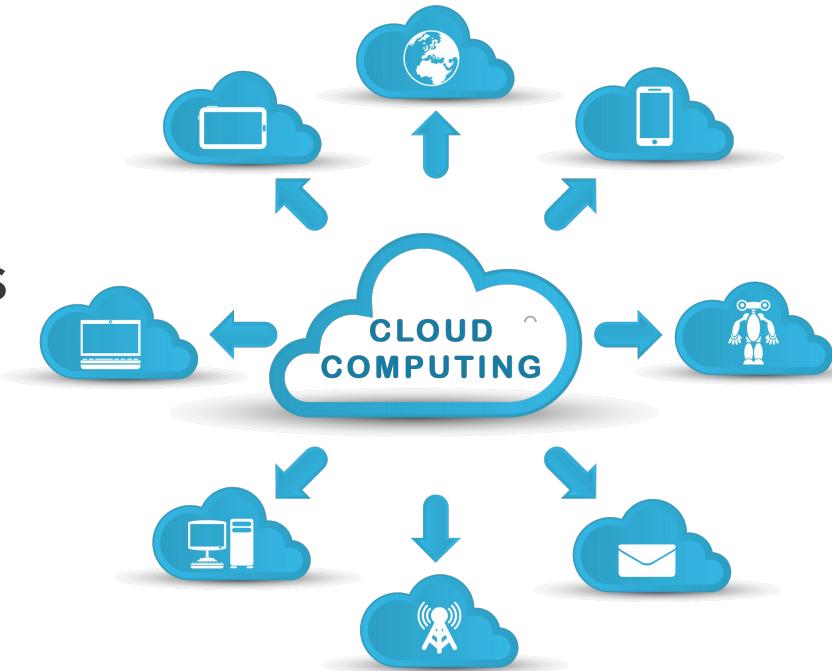
AWS is a Cloud computing platform created by Amazon!

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



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!



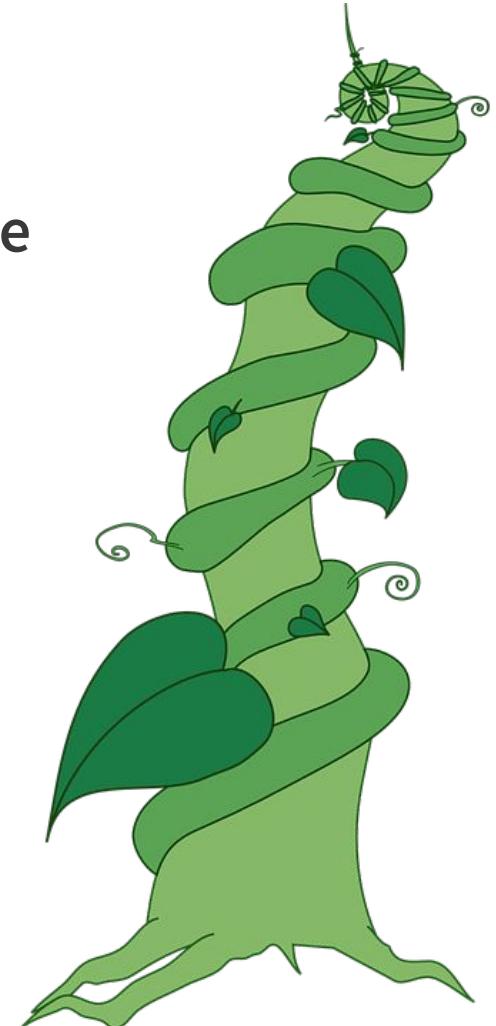
It is a fast, scalable, and affordable option for developers

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.



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

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



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-student-signup

mlhlocal.host/aws-signup



Log into the AWS Console

console.aws.amazon.com

aws

Sign in i

Email address of your AWS account
Or to sign in as an IAM user, enter your [account ID](#) or [account alias](#) instead.

Next

New to AWS?

Create a new AWS account



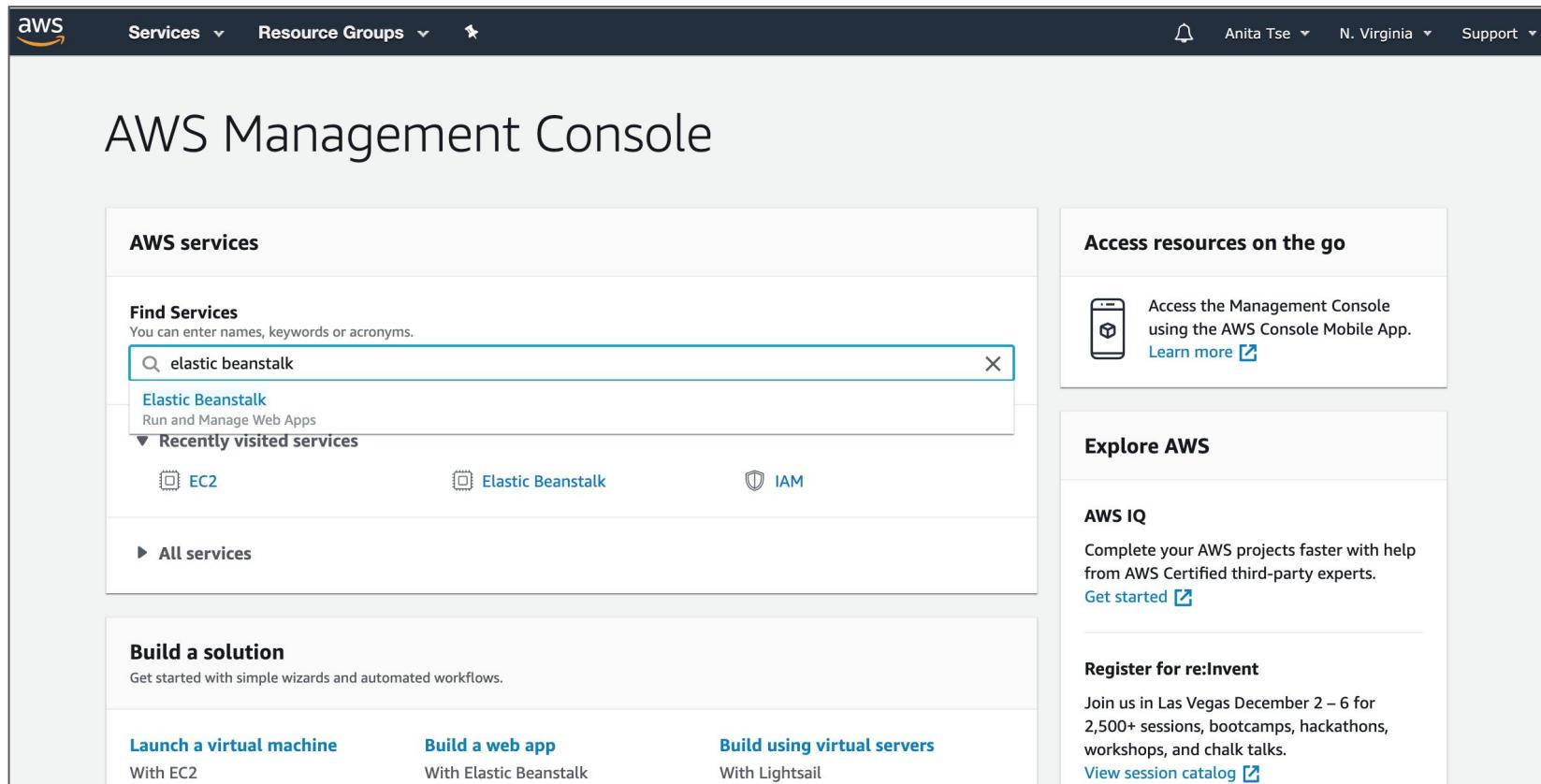
**AWS Accounts Include
12 Months of Free Tier Access**

Including use of Amazon EC2,
Amazon S3, and Amazon DynamoDB

Visit aws.amazon.com/free for full offer terms

Navigate to the EB Console

Type out “elastic beanstalk” in the search bar.



The screenshot shows the AWS Management Console homepage. At the top, there is a search bar with the text "elastic beanstalk" entered. Below the search bar, the "AWS services" section is visible, showing a list of services including "Elastic Beanstalk" under "Find Services". To the right, there are two sidebar boxes: "Access resources on the go" and "Explore AWS". The "Access resources on the go" box contains a link to the AWS Console Mobile App. The "Explore AWS" box contains a link to "AWS IQ". At the bottom right of the page, there is a logo for "nwPlus" with a stylized 'N' icon.

AWS Management Console

AWS services

Find Services
You can enter names, keywords or acronyms.

elastic beanstalk

Elastic Beanstalk
Run and Manage Web Apps

Recently visited services

EC2 Elastic Beanstalk IAM

All services

Build a solution
Get started with simple wizards and automated workflows.

Launch a virtual machine With EC2 **Build a web app** With Elastic Beanstalk **Build using virtual servers** With Lightsail

Access resources on the go
Access the Management Console using the AWS Console Mobile App. [Learn more](#)

Explore AWS

AWS IQ
Complete your AWS projects faster with help from AWS Certified third-party experts. [Get started](#)

Register for re:Invent
Join us in Las Vegas December 2 – 6 for 2,500+ sessions, bootcamps, hackathons, workshops, and chalk talks. [View session catalog](#)

nwPlus

Create a new application

The screenshot shows the AWS Elastic Beanstalk 'Create New Application' interface. A red arrow points from the 'Create New Application' button in the top navigation bar to the 'Create New Application' button in the main content area. The main content area shows a 'Name:' input field and a message: 'No environments currently exist for this application. Create one now.' Below this is a section for the application 'my-app' with the same message.

Create New Application

Name:

No environments currently exist for this application. [Create one now.](#)

my-app

No environments currently exist for this application. [Create one now.](#)

aws Services

Elastic Beanstalk

Learn More

- Get started using Elastic Beanstalk
- Modify the code
- Create and connect to a database
- Add a custom domain

Featured

- Create your own custom platform

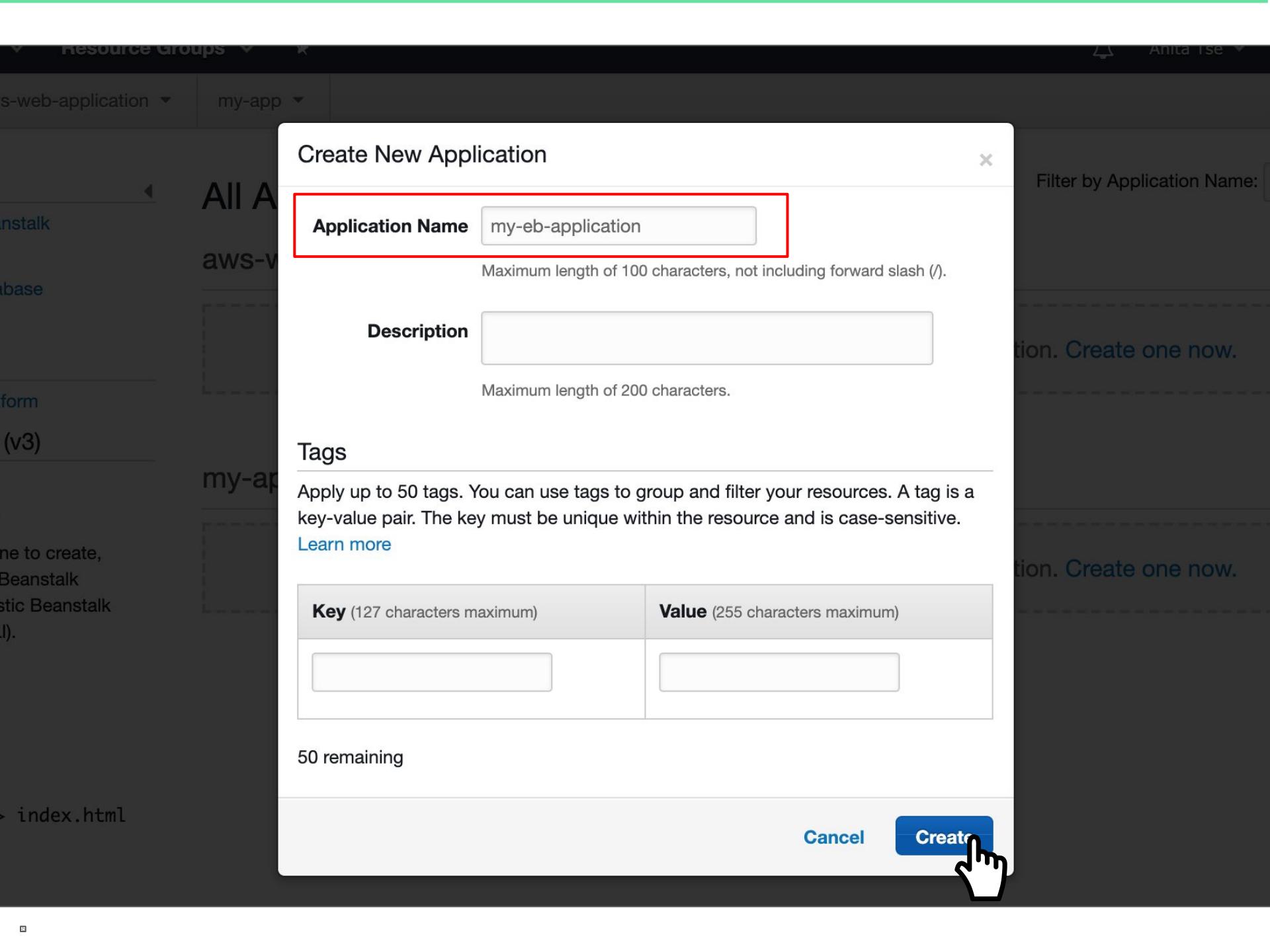
Command Line Interface (v3)

- Installing the AWS EB CLI
- EB CLI Command Reference

If you want to use a command line to create, manage, and scale your Elastic Beanstalk applications, please use the Elastic Beanstalk Command Line Interface (EB CLI).

Get Started

```
$ mkdir HelloWorld
$ cd HelloWorld
$ eb init -p PHP
$ echo "Hello World" > index.html
$ eb create dev-env
$ eb open
```



[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.](#)





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

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 

► Application code tags

Node.js

**Upload
your .zip
file here!**

[Cancel](#)

[Configure more options](#)

[Create environment](#)



Watch your project deploy!

All Applications > my-eb-application > MyEbApplication-env (Environment ID: e-c7hyzqmqjx)

Actions ▾

Creating MyEbApplication-env

This will take a few minutes..

8:32pm Using elasticbeanstalk-us-east-1-582581239747 as Amazon S3 storage bucket for environment data.

8:32pm createEnvironment is starting.

Learn More

[Get started using Elastic Beanstalk](#)

[Modify the code](#)

[Create and connect to a database](#)

[Add a custom domain](#)

Featured

[Create your own custom platform](#)

Command Line Interface (v3)

[Installing the AWS EB CLI](#)

[EB CLI Command Reference](#)

Click on the link to see your project live!

All Applications > my-eb-application > MyEbApplication-env (Environment ID: e-c7hyzqmjx, URL: MyEbApplication-env.ygqrytbvyd.us-east-1.elasticbeanstalk.com)

Actions ▾

Dashboard

Overview

Refresh

Configuration

Logs

Health

Monitoring

Alarms

Managed
Updates

Events

Tags



Health

Ok

Causes

Running Version

my-eb-application-source



Platform

Node.js running on 64bit Amazon Linux/4.10.2

Change

Show All

Recent Events

Time	Type	Details
2019-10-03 20:35:34 UTC-0700	INFO	Environment health has transitioned from Pending to Ok. Initialization completed 9 seconds ago and took 2 minutes.
2019-10-03 20:35:09 UTC-0700	INFO	Successfully launched environment: MyEbApplication-env
2019-10-03 20:35:09 UTC-0700	INFO	Application available at MyEbApplication-env.ygqrytbvyd.us-east-1.elasticbeanstalk.com.

Notice that the address is no longer from your local machine

myebapplication-env.ygqrytbvyd.us-east-1.elasticbeansta...



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!

The image shows the AWS Elastic Beanstalk interface. On the left, a modal dialog titled 'Upload and Deploy' is open. It contains fields for 'Upload application:' (Choose File: nodejs-v1.zip) and 'Version label:' (version-2). At the bottom are 'Cancel' and 'Deploy' buttons, with 'Deploy' highlighted by a red arrow and a red box. On the right, the main application details page is shown. It displays the 'Running Version' as 'my-eb-application-source'. Below this is a 'Upload and Deploy' button, also highlighted with a red box and a red arrow. The page also shows the 'Platform' as 'Node.js running on 64bit Amazon Linux/4.10.2' with 'Change' and 'Show All' buttons. A sidebar on the left includes 'Monitoring', 'Alarms', 'Managed Updates', 'Events', and 'Tags'.

Upload and Deploy

To deploy a previous version, go to the [Application Versions page](#).

Upload application: Choose File nodejs-v1.zip

Version label: version-2

Cancel Deploy

Running Version

my-eb-application-source

Upload and Deploy

Platform

Node.js running on 64bit Amazon Linux/4.10.2

Change Show All

Monitoring

Alarms

Managed Updates

Events

Tags

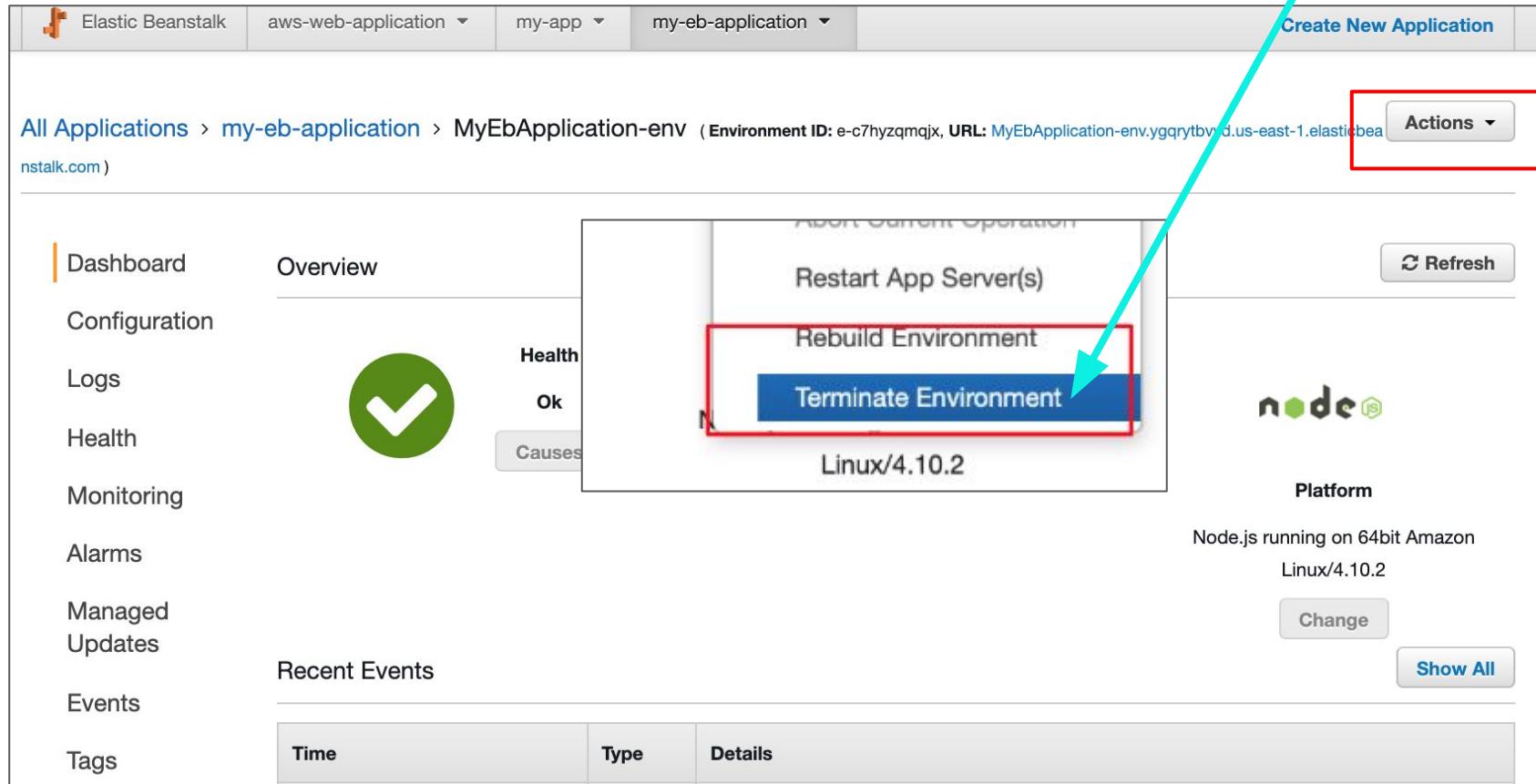
Recent Events

Time Type Details

Cleaning up

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

**Actions →
Terminate
Environment**



The screenshot shows the AWS Elastic Beanstalk console for the 'my-eb-application' environment. The 'Actions' dropdown menu is open, and the 'Terminate Environment' option is highlighted with a red box and a cyan arrow pointing to it. The 'Actions' button itself is also highlighted with a red box. The 'Dashboard' tab is selected, showing a green checkmark icon and the text 'Health: Ok'. The 'Platform' section indicates 'Node.js running on 64bit Amazon Linux/4.10.2'. The 'Recent Events' table is empty.

Elastic Beanstalk

aws-web-application ▾ my-app ▾ my-eb-application ▾

Create New Application

All Applications > my-eb-application > MyEbApplication-env (Environment ID: e-c7hyzqmqjx, URL: MyEbApplication-env.yggrytbv.d.us-east-1.elasticbeanstalk.com)

Actions ▾

Dashboard Overview

Configuration

Logs

Health

Monitoring

Alarms

Managed Updates

Events

Tags

Recent Events

Time Type Details

Restart App Server(s)

Rebuild Environment

Terminate Environment

Linux/4.10.2

node.js

Platform

Node.js running on 64bit Amazon Linux/4.10.2

Change

Show All

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

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!



Build and Deploy a Full Stack Web App with AWS

Learn Day 2019
#workshop-aws

