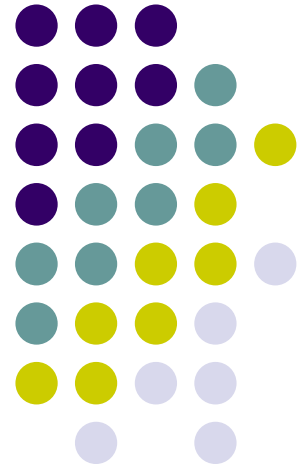


# Big Data Application Architecture

November 28, 2020

---

Mike Spertus  
[mike@spertus.com](mailto:mike@spertus.com)





# REMOTE DEBUGGING AND DEPLOYMENT OF WEB APPLICATIONS



# Remote Development

- Our Kafka cluster is not externally visible, so we cannot locally develop and test the parts of our web app that depend on Kafka
- While this is annoying, it is also very typical that Big Data clusters avoid exposing too many services publicly
- So it is probably good to deal with



# Single Webserver

- Our solution to this is to have a development webserver that you can ssh into
  - Note that it would be bad to allow ssh access to our load-balanced production webserver, which should only be managed by a Continuous Deployment tool like CodeDeploy



# Using a second port

- You can only run one app on a given port at a given time
- We have a quota of 150 listeners on our web balancer, so each student can have two ports of their own
  - You can use the second port to run and compare both your project and the flight\_and\_weather app at the same time
  - Alternatively, you can use the second port for working on enhancements to your project while leaving the existing one deployed
- If you need a third port, please let me know

# Editing in IntelliJ, Running in Single Webserver

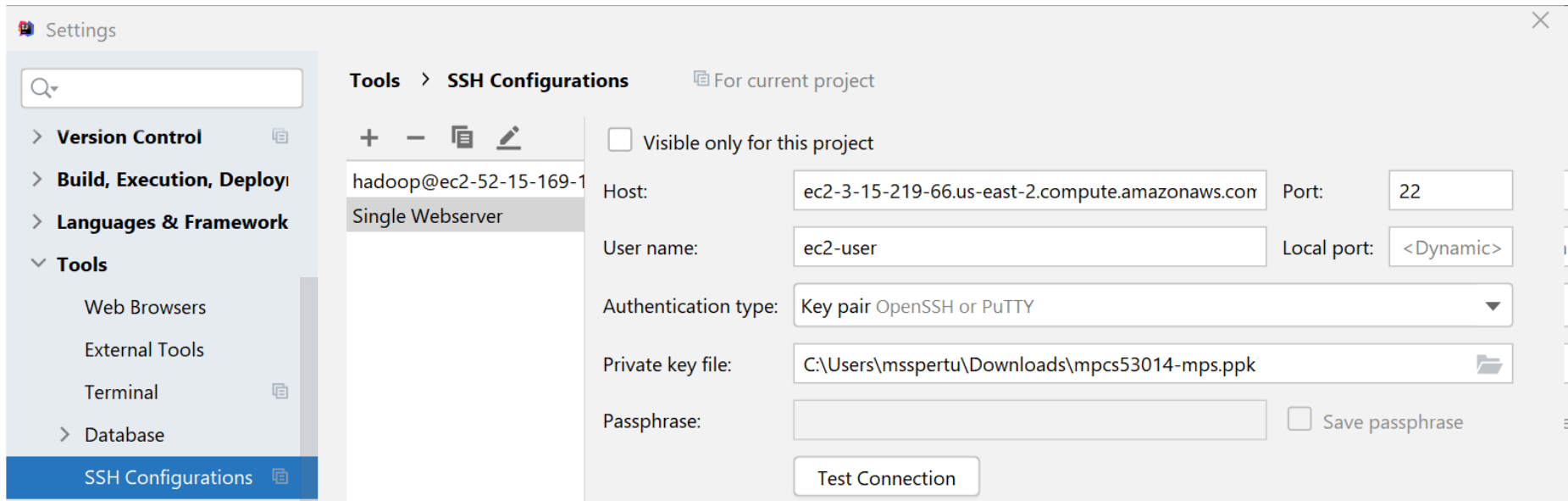


- A good way to develop is to edit your node project in Hive, and then deploy it to Single Webserver
- This has a few differences from our past experiences with IntelliJ deployment, so let's walk through this step-by-step

# Create an ssh configuration for Single Webserver



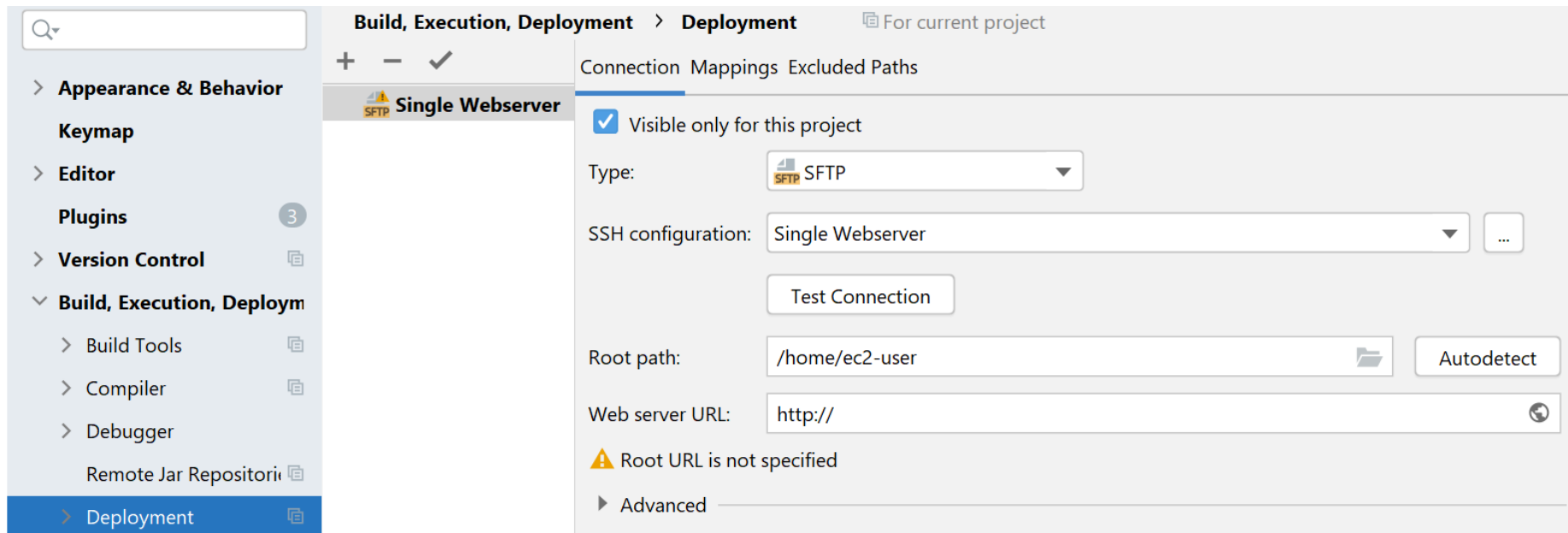
- File/Settings/Tools/SSH Configurations
- This is similar to the one you created for our name node, only with a different host and username



# Create a Deployment for your web application



- With your Node project open in IntelliJ, choose File/Settings/Build, Execution, Deployment/Deployment
- Create a new Connection using the SSH Configuration you just created





# Setup the mappings for the deployment



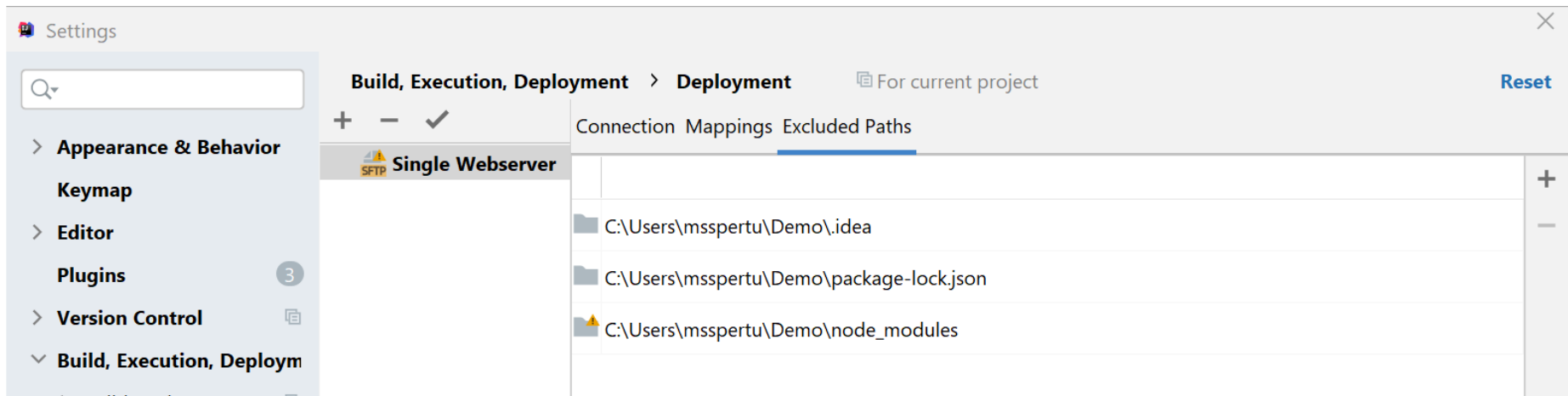
- Since CodeDeploy gave `/home/ec2-user/cnetid` to root, use `dev/cnetid/project` in the Deployment's Mapping tab

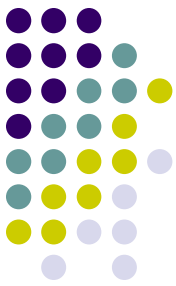
The screenshot shows the IntelliJ IDEA Settings dialog, specifically the 'Build, Execution, Deployment' section, with the 'Deployment' sub-tab selected. The 'Mappings' sub-tab is active, showing a table with one entry: 'Single Webserver' (SFTP). The table has three columns: 'Local path', 'Deployment path', and 'Web path'. The 'Local path' is 'C:\Users\msspertu\Demo', the 'Deployment path' is 'dev/spertus/Demo', and the 'Web path' is '/'. Below the table, there is a note: 'Local path is absolute. Deployment path is relative to the server root path (/home/ec2-user). Web path is relative to the web server URL (http://).'. There is an 'Add New Mapping' button at the bottom. The left sidebar shows the 'Build, Execution, Deployment' section expanded, with 'Build Tools' and 'Compiler' visible. The top right of the dialog has a 'Reset' button and a 'For current project' checkbox.



# Excluded Paths

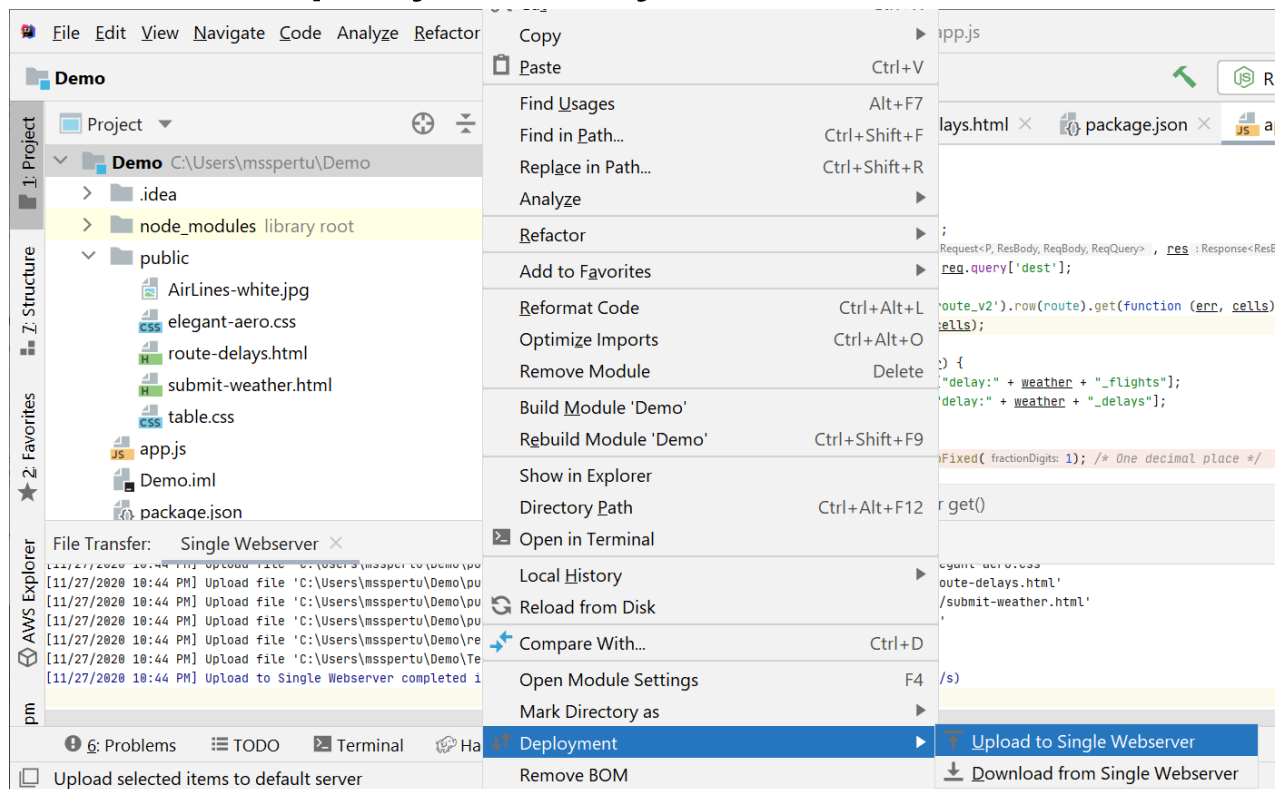
- We will not want to upload everything to Single Webserver
  - E.g., not only will your node\_modules directory be huge, it may need to be compiled differently for Single Webserver than your laptop
- In the Deployment's Excluded Paths tab, add Local Paths as below





# Deploy to Single Webserver

- When you are ready to test out your application, right click on the root folder in IntelliJ and Upload to the Deployment you created





# SSH into Single Webserver

- Log into Single Webserver
  - Either through putty/ssh like with our name node
  - Or by Connecting through the Connect button in the Single Webserver Instance page in the EC2 Console
- Go to the deployment directory and install your dependencies with  
`npm install`

```
ec2-user@ip-172-31-1-18:~/dev/spertus/Demo
Using username "ec2-user".
Authenticating with public key "mpcs53014-mps"
Last login: Sat Nov 28 18:48:58 2020 from ec2-3-16-146-0.us-east-2.compute.amaz
naws.com

  _ |  _ | _ )
 _ | ( _ | /   Amazon Linux 2 AMI
 _ | \ _ | _ |

https://aws.amazon.com/amazon-linux-2/
12 package(s) needed for security, out of 17 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-1-18 ~]$ cd dev/spertus/Demo/
[ec2-user@ip-172-31-1-18 Demo]$ ls
app.js  Demo.iml  package.json  public  result.mustache  Test.iml
[ec2-user@ip-172-31-1-18 Demo]$ npm install
```

# Stop any other application running on the same port



- If you have another application running on the same port, you will need to stop it
- If you deployed it w/CodeDeploy, it will be managed by `forever`, so identify and stop as follows

```
ec2-user@ip-172-31-1-18:~/dev/spertus/Demo
[ec2-user@ip-172-31-1-18 Demo]$ # Check for apps by me on port 3001
[ec2-user@ip-172-31-1-18 Demo]$ forever list | grep mspertus | grep 3001
(node:17601) Warning: Accessing non-existent property 'padLevels' of module exports inside circular dependency
(Use `node --trace-warnings ...` to show where the warning was created)
(node:17601) Warning: Accessing non-existent property 'padLevels' of module exports inside circular dependency
data: [74] gGhU /usr/bin/node app.js 3001 ip-172-31-11-144.us-east-2.compute.internal 8070 b-2.mpcs53014-kafka.fwx2ly.c4.kafka.us-east-2.amazonaws.com:9092,b-1.mpcs53014-kafka.fwx2ly.c4.kafka.us-east-2.amazonaws.com:9092 16171 16178 mspertus_flight_and_weather /home/ec2-user/.forever/gGhU.log 0:18:7:26.2149999999996508
[ec2-user@ip-172-31-1-18 Demo]$ forever stop mspertus_flight_and_weather
(node:17621) Warning: Accessing non-existent property 'padLevels' of module exports inside circular dependency
(Use `node --trace-warnings ...` to show where the warning was created)
(node:17621) Warning: Accessing non-existent property 'padLevels' of module exports inside circular dependency
info: Forever stopped process:
      uid  command      script
d  id      logfile      uptime      forever pi
[0] gGhU /usr/bin/node app.js 3001 ip-172-31-11-144.us-east-2.compute.internal 8070 b-2.mpcs53014-kafka.fwx2ly.c4.kafka.us-east-2.amazonaws.com:9092,b-1.mpcs53014-kafka.fwx2ly.c4.kafka.us-east-2.amazonaws.com:9092 16171 16178 mspertus_flight_and_weather /home/ec2-user/.forever/gGhU.log 0:18:7:48.9080000000003085
[ec2-user@ip-172-31-1-18 Demo]$ # Confirm it's gone
[ec2-user@ip-172-31-1-18 Demo]$ forever list | grep mspertus | grep 3001
(node:17643) Warning: Accessing non-existent property 'padLevels' of module exports inside circular dependency
(Use `node --trace-warnings ...` to show where the warning was created)
(node:17643) Warning: Accessing non-existent property 'padLevels' of module exports inside circular dependency
[ec2-user@ip-172-31-1-18 Demo]$
```



# Now run your app with Node

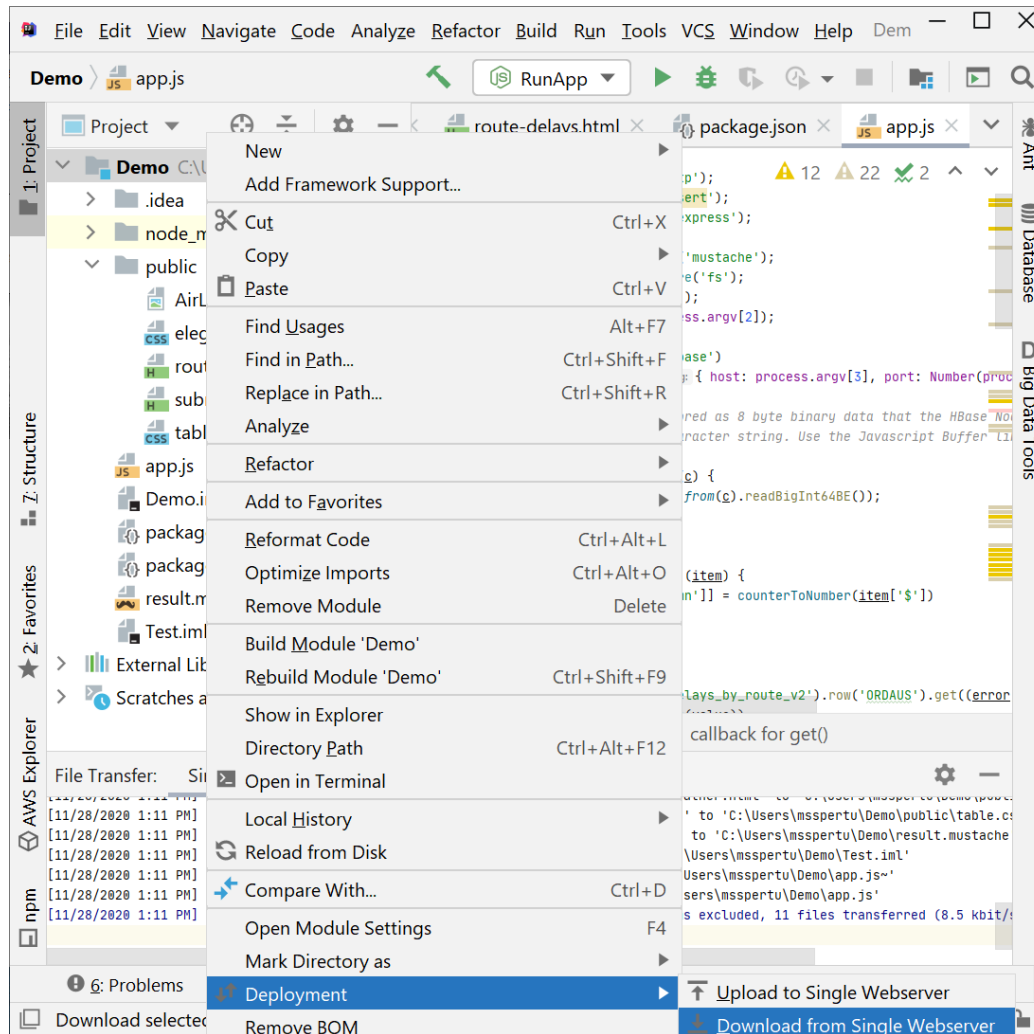
- Run it with node for easy debugging
  - As opposed to a “service” mode like forever

```
[ec2-user@ip-172-31-1-18 Demo]$  
[ec2-user@ip-172-31-1-18 Demo]$  
[ec2-user@ip-172-31-1-18 Demo]$ node app.js 3001 ip-172-31-11-144.us-east-2.compute.internal 8070 b-2.mpcs53014-kafka.fwx2ly.c4.kafka.us-east-2.amazonaws.com:9092,b-1.mpcs53014-kafka.fwx2ly.c4.kafka.us-east-2.amazonaws.com:9092
```

- And test with your app

The screenshot shows a web browser window with the address bar displaying `3.15.219.66:3001/submit-weather.html`. The browser's bookmark bar includes links for 'Getting Started', 'Misc', 'Most Visited', 'Amazon', 'ProxyIt!', 'Admin', 'Homotopy Type Theo...', 'CS', and 'Japanese font browser'. The main content area features a form titled 'Submit weather report'. The form includes a 'Station:' label followed by a text input field containing the value '0'. Below this, there is a row of weather conditions with checkboxes: 'fog', 'rain', 'snow', 'hail', 'thunder', and 'tornado'. The 'thunder' checkbox is currently checked. At the bottom of the form is a 'Submit' button.

# If you make changes, you can download back into IntelliJ





# When you're ready to deploy

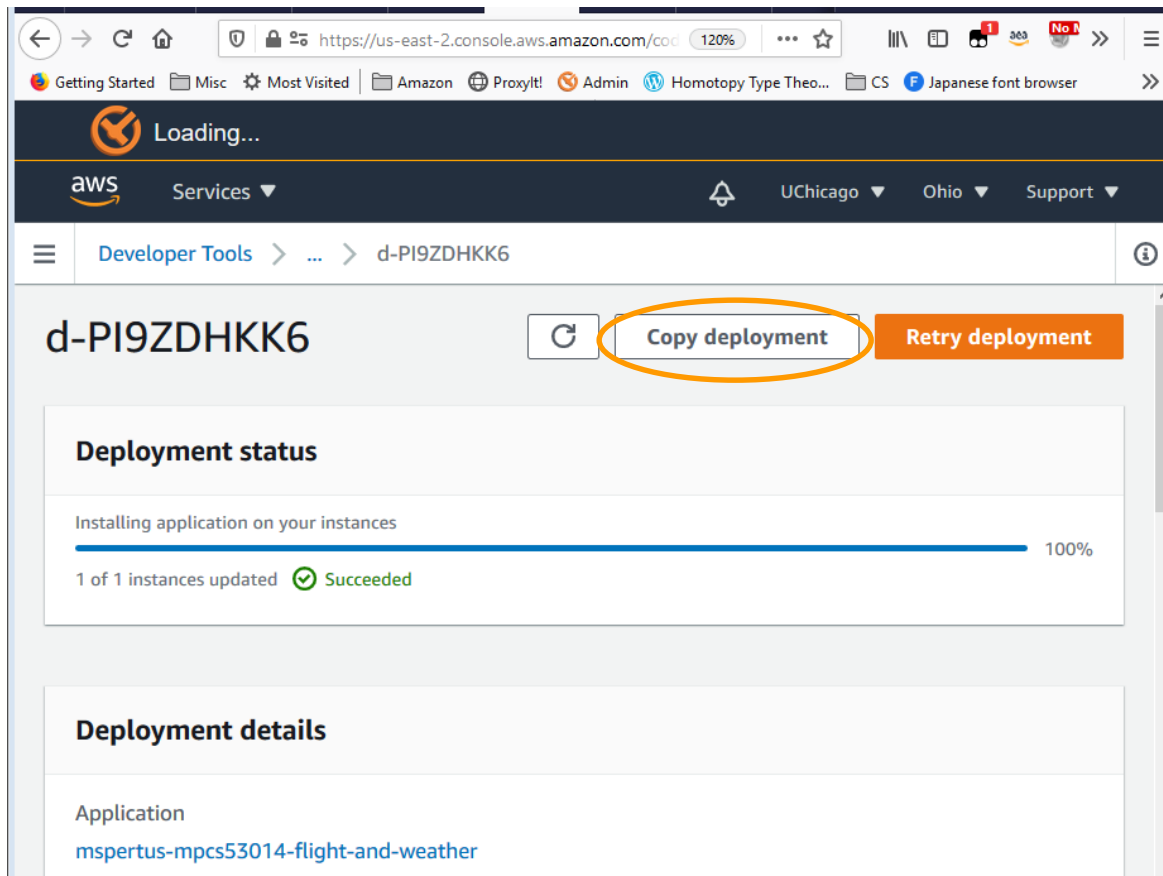
- You will want to test and debug your deployment on Single Webserver
- Stop your node process (ctrl-C) if it's still running in your terminal
- Create your deployment archive and upload to S3 as usual
- Now use CodeDeploy
  - but not quite as usual



# CodeDeploy to Single Webserver



- Copy one of your app's existing deployments





# Deployment settings

- Confirm all the usual settings (S3 URL, QuickDeploy)
- Since we manually killed the previous deployment earlier, tell it not to worry if it fails to kill the previous deployment
  - This setting often comes in handy if an application crashes as well

## Additional deployment behavior settings

ApplicationStop lifecycle event failure - *optional*

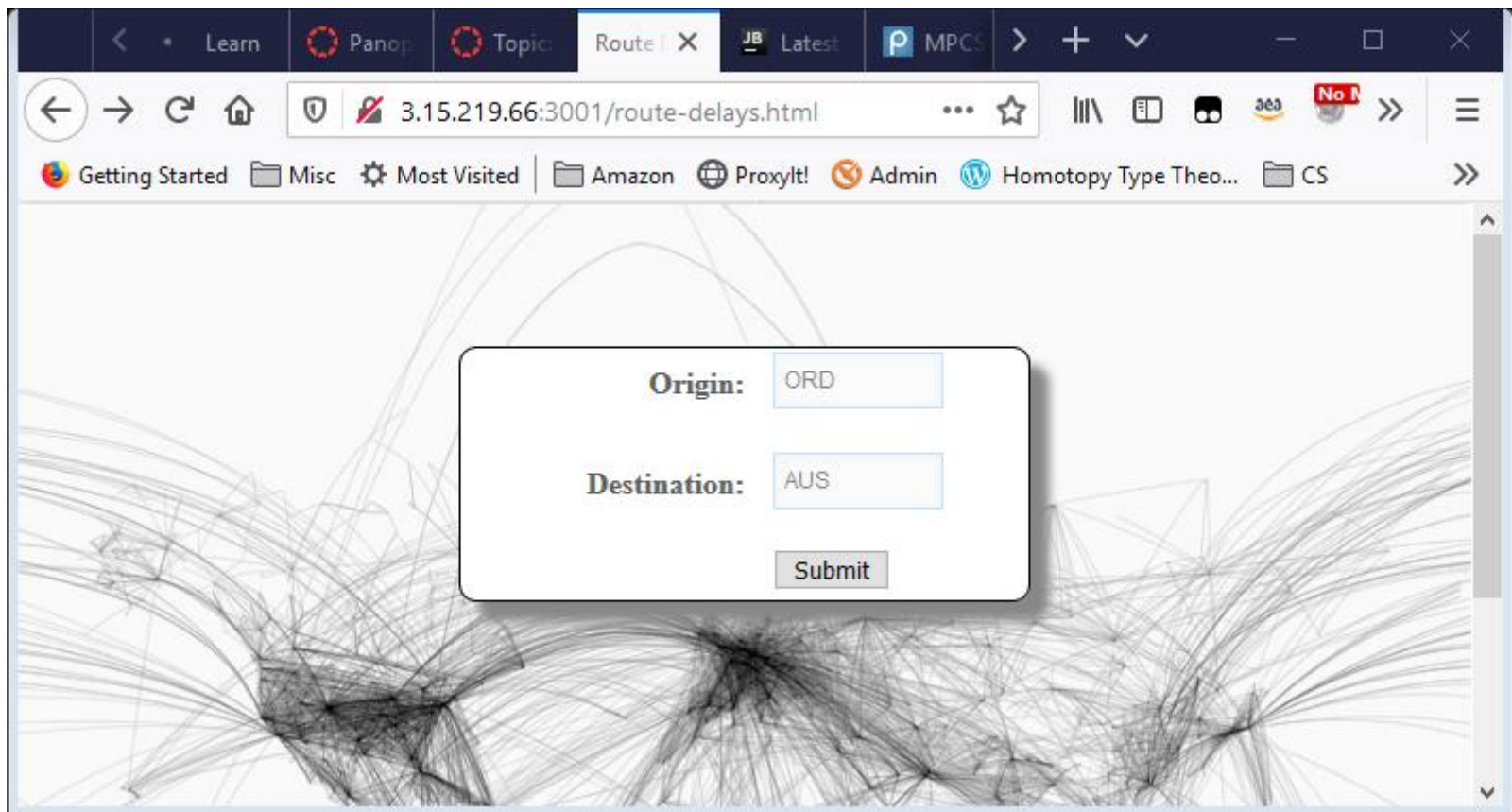
Type a deployment group name

☒ Don't fail the deployment to an instance if this lifecycle event on the instance fails



# Deploy and test

- Hit Create Deployment and check if it works



# When ready, deploy to our production cluster



- Once you are ready to deploy, use CodeDeploy to deploy to our production cluster
- You can do this as before, but the following slides show some possible refinements

# Tip: Deploy even if ApplicationStop fails



- As mentioned above, if there is not a previous app running, you may need to tell CodeDeploy that is OK

## Additional deployment behavior settings

ApplicationStop lifecycle event failure - *optional*

Type a deployment group name

☒ Don't fail the deployment to an instance if this lifecycle event on the instance fails

# Tip: Don't tell CodeDeploy about our load balancer



- While our LoadBalanced deployment group uses a Load Balancer, there's no reason for CodeDeploy to worry about it
  - We leave it running all the time and it's shared
  - This isn't a real production app, so we aren't worrying about letting in process transactions drain
  - Cycling the load balancer can interfere with other students' apps
  - Draining and restarting the load balance can add ten minutes to the deploy process
  - The constant cycling may contribute to some of the erratic Load Balancer behavior we've experienced

# Editing your LoadBalanced deployment group



- Based on the above, it may be helpful to edit your application's LoadBalanced Deployment Group

The screenshot shows the AWS CodeDeploy console interface. On the left is a navigation sidebar with 'Developer Tools' and 'CodeDeploy' sections. The main content area shows the breadcrumb path 'Developer Tools > CodeDeploy > Applications > mspertus-mpcs53014-flight-and-weather > LoadBalanced'. Below the breadcrumb, the title 'LoadBalanced' is followed by three buttons: 'Edit' (circled in orange), 'Delete', and 'Create deployment'. Below this is a section titled 'Deployment group details' containing a table with the following information:

Deployment group name	Application name	Compute platform
LoadBalanced	mspertus-mpcs53014-flight-and-weather	EC2/On-premises
Deployment type	Service role ARN	Deployment configuration
In-place	arn:aws:iam::787308793:role/mpcs53014-CodeDeploy	CodeDeployDefault.ALIATOnce

# Clear “Enable load balancing”



- Note that your deployment group will still be behind a load balancer
  - But we’re keeping that a secret from CodeDeploy!

## Load balancer

Select a load balancer to manage incoming traffic during the deployment process. The load balancer blocks traffic from each instance while it's being deployed to and allows traffic to it again after the deployment succeeds.

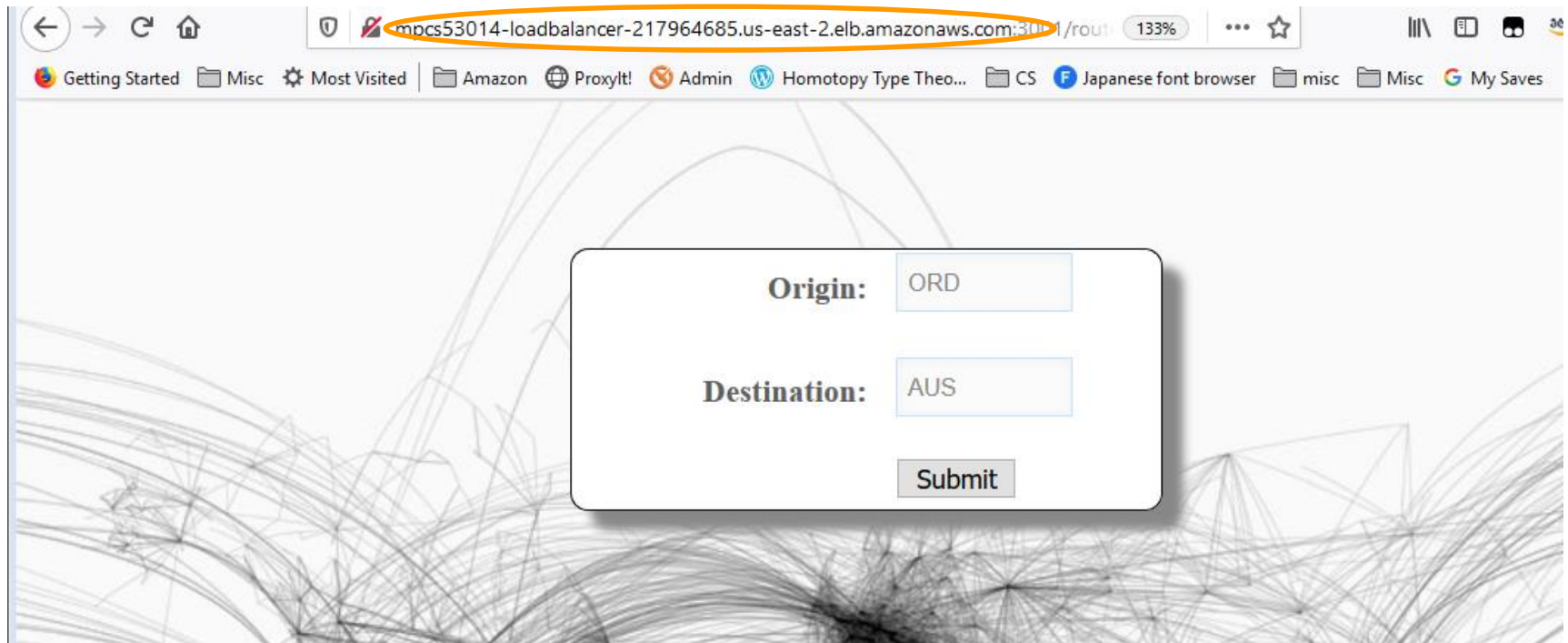
☐ Enable load balancing





# Deploy and test

- Your deployment should run quickly now
- If it succeeds, be sure to test





# **SOME MORE DEPLOYMENT FAQS**

# My deployment succeeded, but it is still running the old app



- If you have multiple applications sharing the same port, it is likely that you will have to stop the old application before the new one can bind to the port
- One workaround is to use a different port for each application
  - Remember you can choose two
- Or you can shutdown the previous app
  - Instructions on next slide

# How do I stop a deployed application?



- If it's just on Single Webserver, you can just do “forever stop” like we did above
- However, if it's deployed to a fleet of LoadBalancers, you will want to use CodeDeploy

# Stopping an application with CodeDeploy



- I create a modified version of my deployment archive named stop\_flight\_and\_weather.zip
- The only change is that the appspec.yml consists only of an ApplicationStop hook

A screenshot of a Notepad++ editor window. The title bar shows the file path: C:\Users\msspertu\msspertu\_stop\_flight\_and\_weather\appspec.yml - Notepad++. The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, Window, and ?. The toolbar contains various icons for file operations. The tab bar shows several open files: app.js, application\_start.sh, write\_to\_hbase\_v2hql, new 1, application\_start.sh, application\_stop.sh, after\_install.sh, and appspec.yml. The main text area displays the following YAML content:

```
1 version: 0.0
2 os: linux
3
4 hooks:
5   ApplicationStop:
6     - location: bin/application_stop.sh
7       timeout: 120
8       runas: ec2-user
9
```

The status bar at the bottom indicates: YAML Ain't Markup Lang length: 131 lines: 9, Ln: 1 Col: 1 Pos: 1, Unix (LF), UTF-8, and INS.

# Stopping an application with CodeDeploy (cont)



- Now deploy the “stop application” archive

A screenshot of the AWS CodeDeploy console. The browser address bar shows the URL: https://us-east-2.console.aws.amazon.com/codesuite/codedeploy/application/mspertus-mpcs53014-flight-and-weathe. The console interface has a dark header with the AWS logo and a 'Loading...' status. Below the header, there's a sidebar with a menu icon. The main content area shows the 'Deployment type' as 'In-place'. Under 'Revision type', there are two radio buttons: 'My application is stored in Amazon S3' (selected) and 'My application is stored in GitHub'. Under 'Revision location', there's a text input field containing 's3://mspertus-mpcs53014/stop\_flight\_and\_weather.zip', which is circled in orange. Below the input field, there's a placeholder text 's3://bucket-name/folder/object.[zip|tar|tgz]'. Under 'Revision file type', there's a dropdown menu with '.zip' selected.

Getting Started Misc Most Visited Amazon ProxyIt! Admin Homotopy Type Theo... CS Japanese font browser misc Misc My Saves

aws Loading...

Services ▼

Deployment type

In-place

Revision type

☒ My application is stored in Amazon S3 ☐ My application is stored in GitHub

Revision location

Copy and paste the Amazon S3 bucket where your revision is stored

s3://bucket-name/folder/object.[zip|tar|tgz]

Revision file type



# Now Deploy your new App

- Deploy the new app as normal, but be sure to tell it not to fail if it can't stop the previous deployment

## Additional deployment behavior settings

ApplicationStop lifecycle event failure - *optional*

Type a deployment group name

☒ Don't fail the deployment to an instance if this lifecycle event on the instance fails