SER421 Fall 2021 Lab 6 — Submit a file lab6.zip

Objectives:

1. Deploy a "serverless" function and API endpoint on the cloud
2. Distinguish between variants of responsive design websites
3. (EC) Understand key exchange and encryption over https

Activity 1: 50 points

Part 1 (20 pts): Deploy a given example

At the end of this document is a step-by-step guide for deploying a simple lambda echo function as a REST API. Please follow the guide and deploy your own echo example. You will submit the URL to your AWS REST API endpoint, and a screenshot of Postman or cURL posting to the endpoint and receiving a response. You submission should include a your URL endpoint on AWS (which you can put in a README.md|txt, and your modified client code named as lab6act1.zip (include html and js files).

Part 2 (30 pts): Deploy your own custom REST endpoint

Starting with the jokes fetch example in the class repo, do the following:

1. Create an AWS lambda function that randomly returns a joke from a list of jokes. Specifically, look at the bottom of simple_fetch_example.html and the body of function *newJoke*. You can see that what it does is return a joke from the array of jokes. Make a lambda function that provides equivalent functionality.
2. As you did in Part 1, create a REST API endpoint for this new lambda function. This time your API should respond to GET requests with a randomly selected joke (meaning, it should invoke your lambda method).
3. Modify the simple_fetch_example.html *newJoke* function to make an AJAX or fetch invocation to your new API and display the resulting joke in the page (as it does now locally).
4. You may have trouble accessing your API via AJAX/fetch due to CORS. Enable CORS for your API under the Actions menu on the API Resources page. This page shows you how.

Activity 2: 50 points

For this activity you will identify responsive, adaptive, and/or fluid web design websites. You will use your browser dev tools in Responsive Design Mode to assist you with this identification

Part 1 (30 pts): In a Word document file (or OpenOffice) saved as lab6act2.doc[x][|.odt] please indicate whether the following websites are *responsive* and/or *adaptive* and/or *fluid* or *none of these*:

1. phoenix.craiglist.org
2. discord.com
3. winstockfestival.com/
4. homedepot.com
5. lowes.com

For each of these websites, justify your answers with a sentence or two.

Part 2: (20 pts) *Battle of the big-market hardware stores!*

For the Home Depot and Lowe's websites, use your browser's responsive web dev tools to render the site on two devices: an iPad, and a Pixel 2 XL Android device. For each of these 2 sites and 2 devices, capture screenshots in portrait and landscape modes. This means you will have 2*2*2=8 screenshots. Then, explain (sentence or two) how this testing mode corroborates (agrees with) your answer and explanation for to the responsive/adaptive/fluid/none question above.

Firefox page on using RWD tools: https://developer.mozilla.org/en-US/docs/Tools/Responsive_Design_Mode
Google Chrome device page: https://developer.chrome.com/docs/devtools/device-mode/

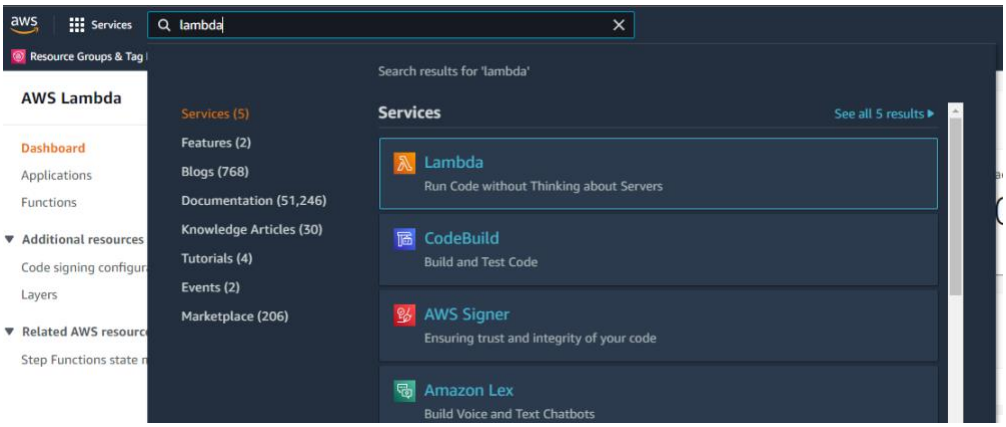
Extra Credit (30 points): *Completed in its entirety, no partial credit, and only submit if you complete both Activity 1 and Activity 2 above successfully. Submit a word doc named lab6ec.docx*

For the extra credit you will need to download *Wireshark*, a network packet sniffing utility. Steps:

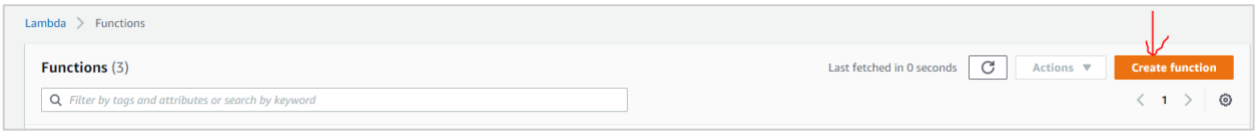
1. Bring up wireshark, and start a capture on the loopback interface (lo0). In the filter line, specify the filter string: tcp.port==8443 and tcp and not tcp.len==0
2. Locally, run the https_echo.js server in the NodeHttp directory on the class public repo
3. First, open Google Chrome, and ensure you have only one tab open. Access http://localhost:8443. You will get a one-line textbox; type in anything and hit return
4. In Wireshark, you should have seen some packet traffic. With that traffic, do the following:
   a. Provide screen captures of each of the asymmetric key exchange handshake packets
   b. Identify the set of cipher suites (screenshot) available in the browser, and the one selected by the server.
   c. Identify the place (packet) in the conversation where the client and server start using symmetric keys and explain why you think this is the place.
5. Now repeat steps 1-4 using a Firefox browser, answering the same questions, and performing the same captures.
6. Answer the following by comparing the 2 browsers' traffic:
   a. Is there a packet in one that does not appear in the traffic of the other? Identify it and explain what it is
   b. Compare the cipher suites of the 2 browsers and identify the differences, if any.
   c. Was there any difference in the cipher selected by the server?
7. Clear your wireshark traffic, go back and select your main network interface (probably your wireless interface, unless you are wired somewhere). Enter a filter of "ssl and tcp and not tcp.len==0" and start a capture. Now hit our sample app at https://swent1linux.asu.edu/simplemvcex using either Firefox or Chrome (both are not required). Review the Client Hello, Server Hello and key exchange protocols. Did the browser and server end up using the same protocol you used for your local server? Explain the outcome you see.

STEPS TO MAKE AN AWS LAMBDA FUNCTION AND A REST ENDPOINT

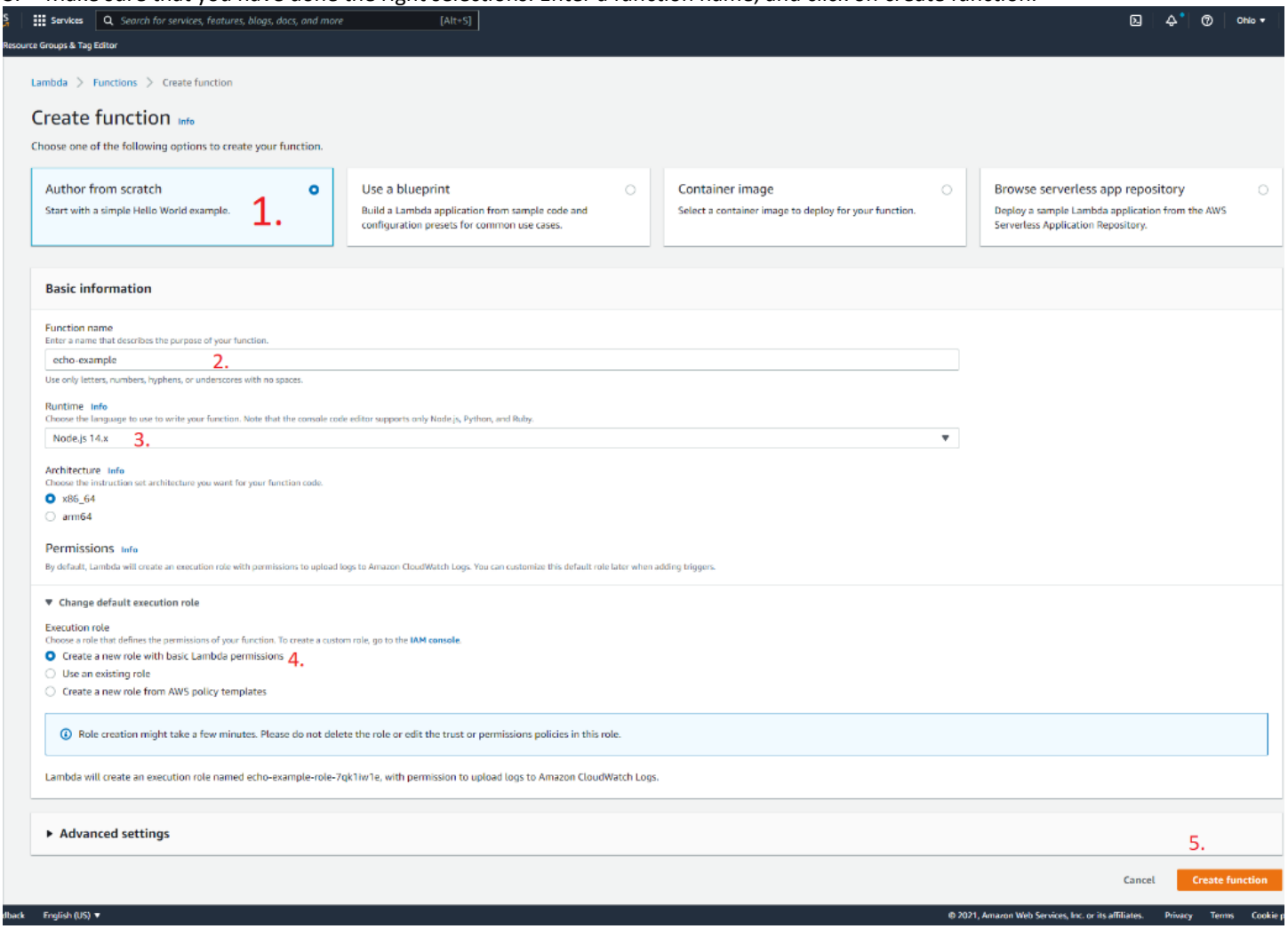
1. Log in to your AWS account, search lambda and click on Lambda under services.



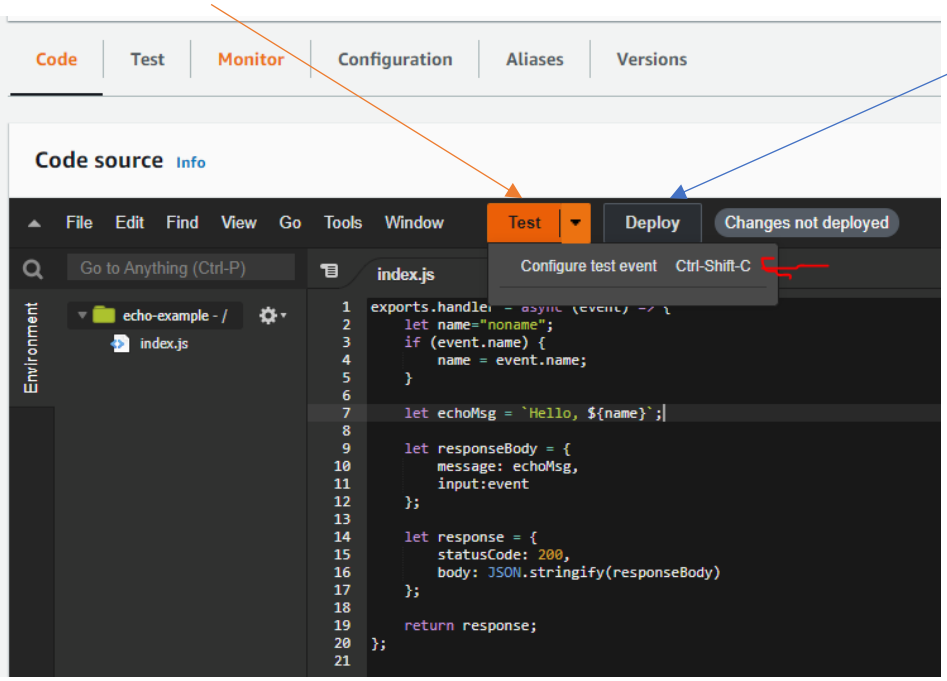
2. Click on create function.



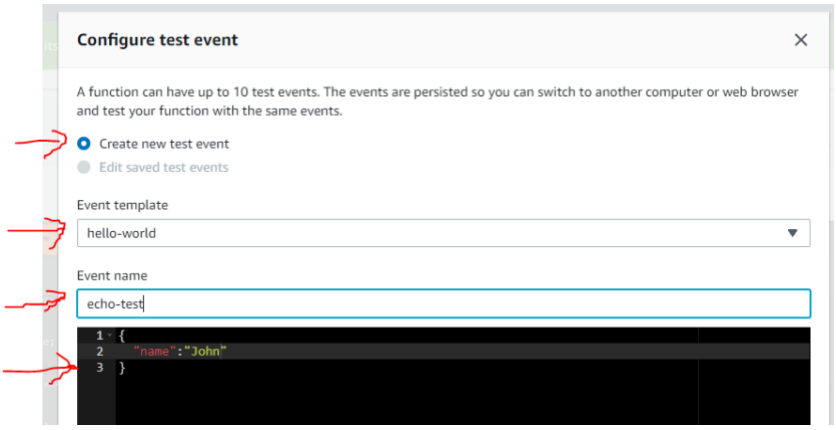
3. Make sure that you have done the right selections. Enter a function name, and click on create function.



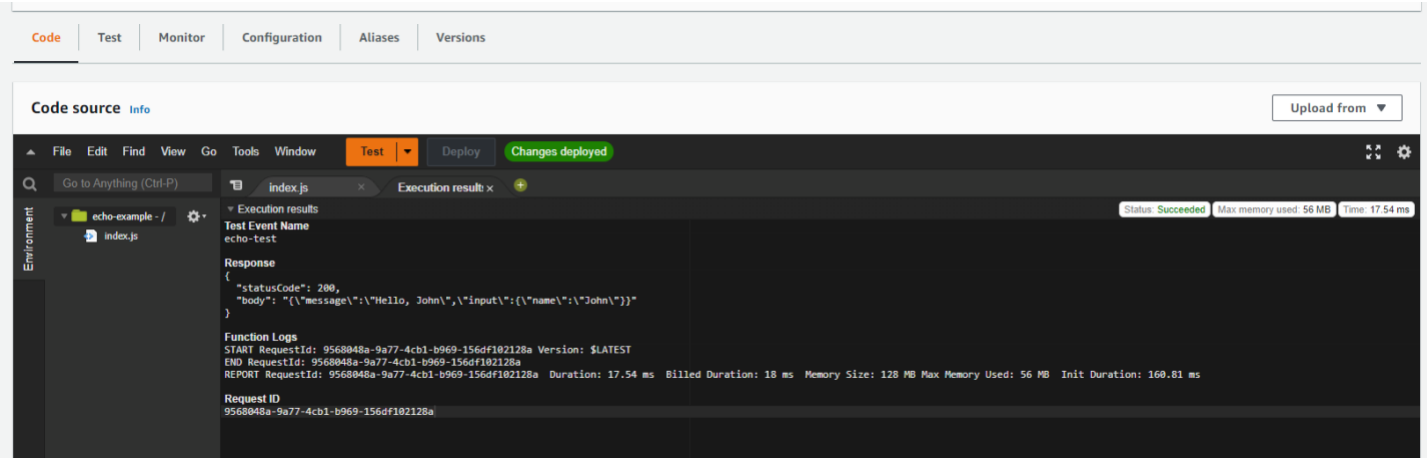
4. You'll be redirected to a page with a source code editor. Type in the code in the image and then click on Deploy to deploy it. Then click on Test. Note in line 7 that these are backticks ` not single quotes '



5. You should see following screen. This is the payload to test the lambda function against. Click on Create.

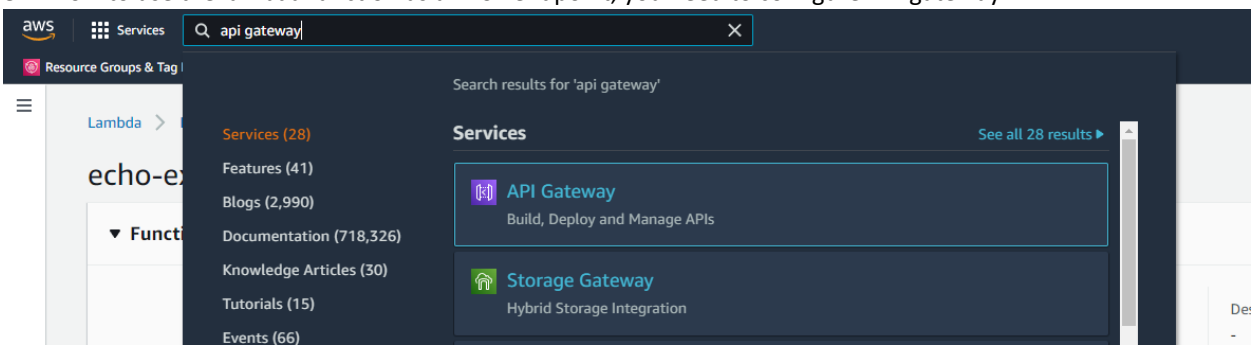


6. Click on Test. You should see the following output.

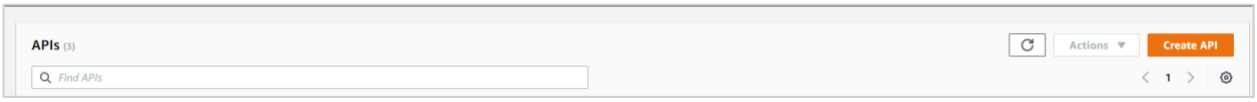


7. Once you are satisfied with your test, you can Deploy (right beside the 'Test' button). When you are deployed you will see the green "Changes deployed" as you see in the image (you can Test and Deploy in any order, but of course it is always better to Test first, then Deploy!)

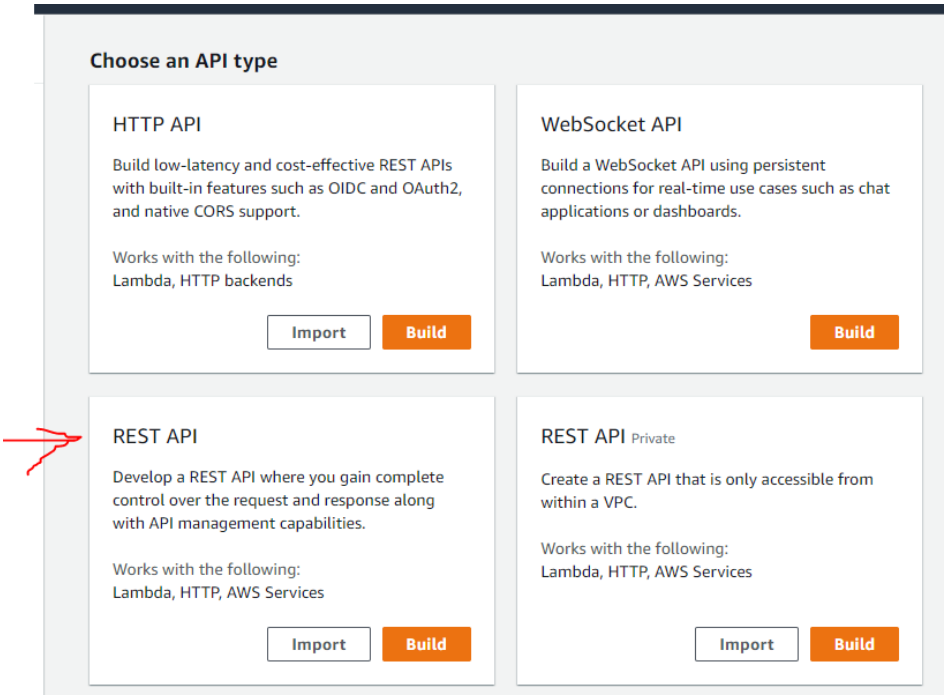
8. Now to use the lambda function as a REST endpoint, you need to configure API gateway.



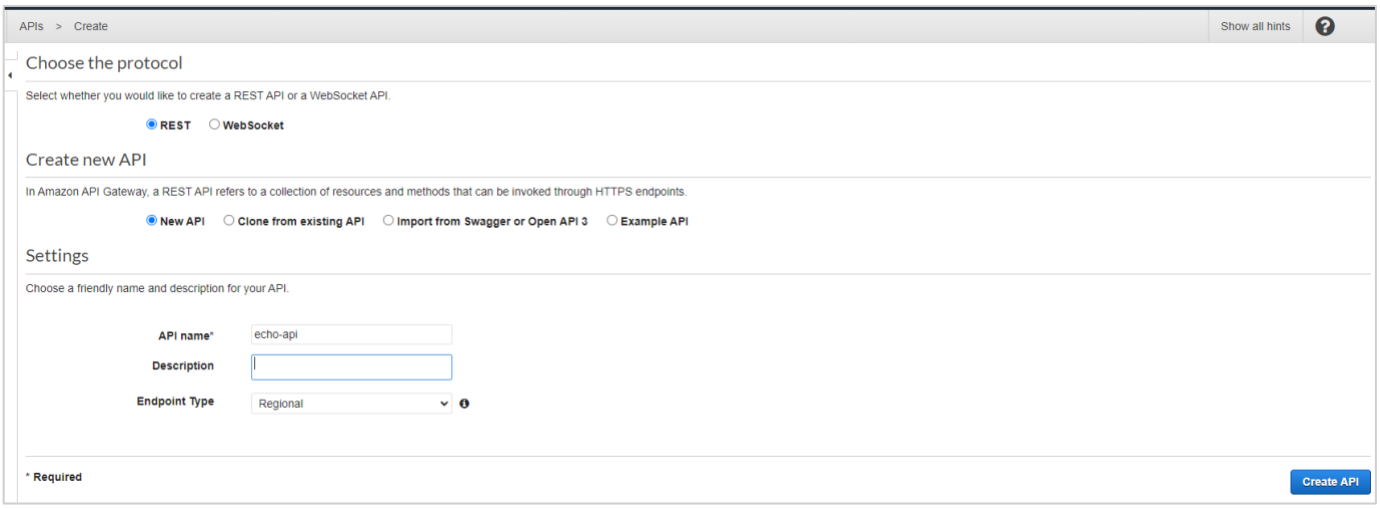
9. Click on Create API.



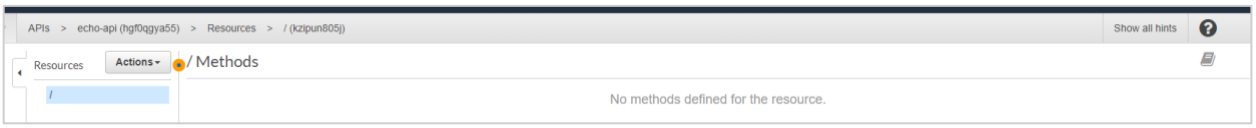
10. Click on Build - REST API (pointed in image with the red arrow)



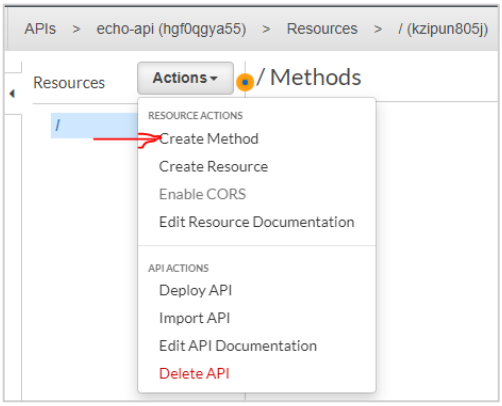
11. Type in the API name and click on Create API



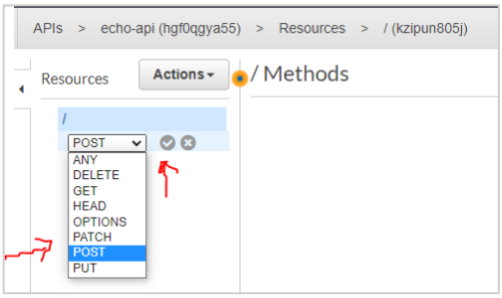
12. You should see the following screen.



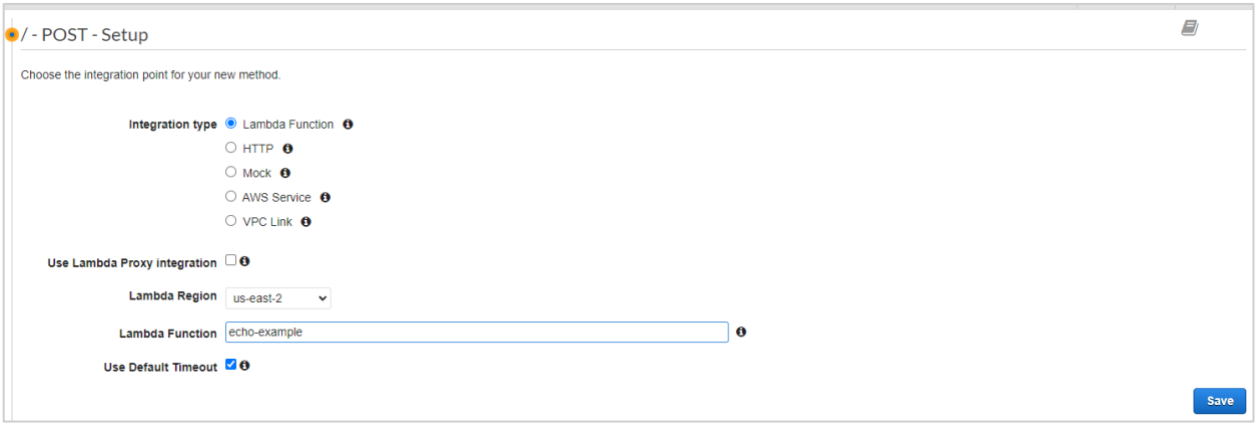
13. Click on Create Method under Actions drop down



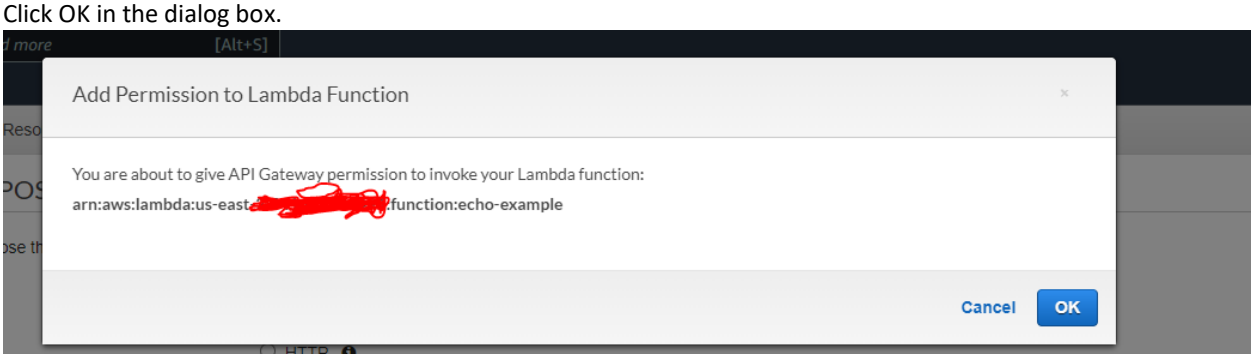
14. Select POST, click on the tick mark.



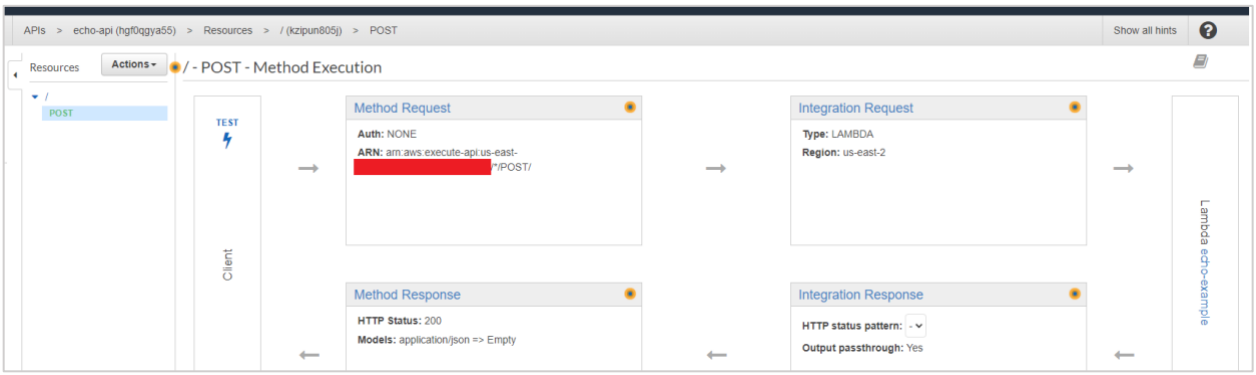
15. Type in the lambda function name (the one we created earlier: echo-example, this should autocomplete as you type in the name) and Click on Save.



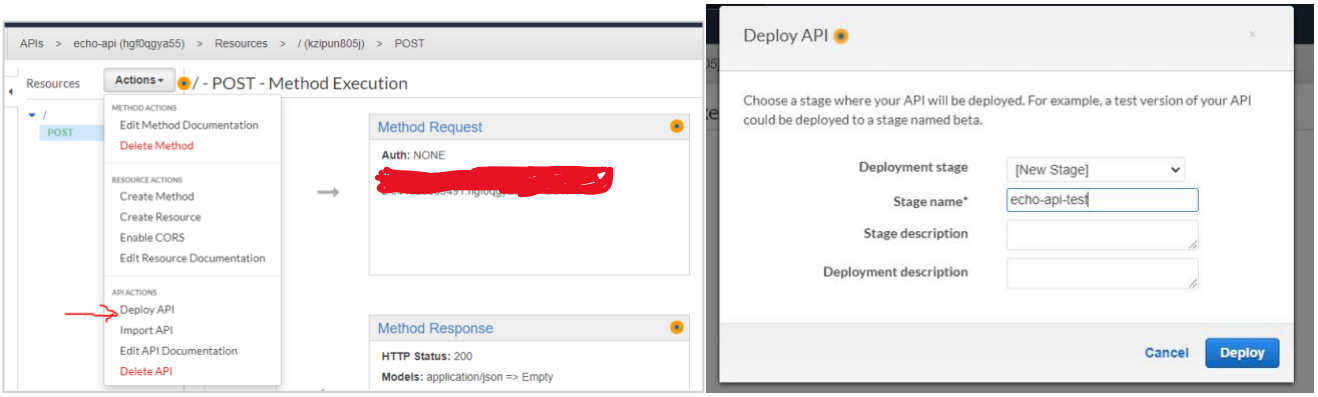
Click OK in the dialog box.



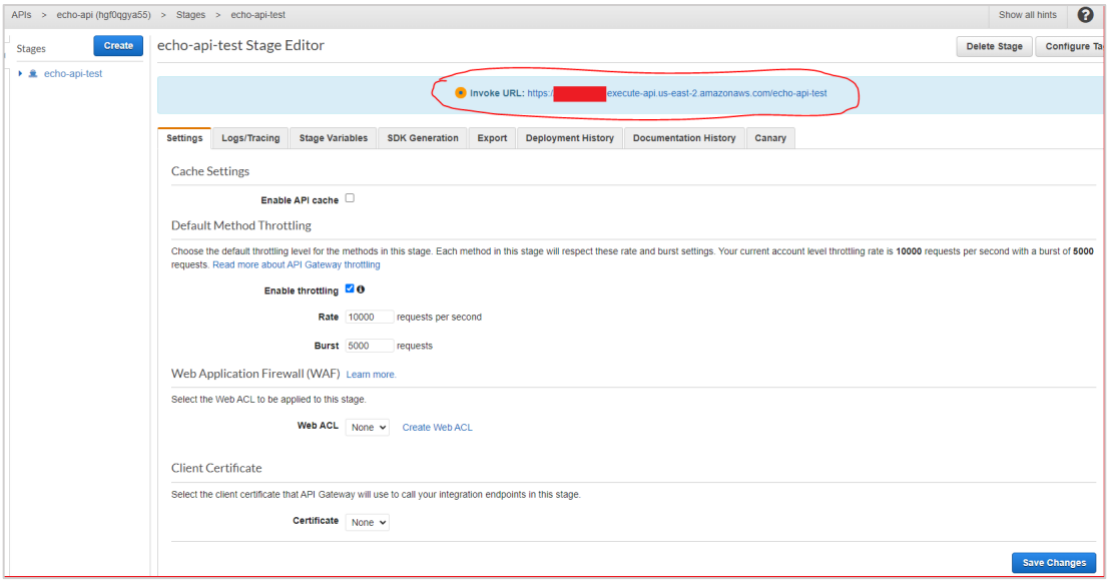
16. You should see the following screen.



17. To deploy the API



18. Write down the URL at the top, it can be used to test the API from cURL/Postman. Click on Save Changes.



19. Go ahead and test your REST API endpoint from Postman/cURL. (Make sure to select POST method, and the JSON payload similar to what is done in step 6)