PUSHING THE AWS LIMITS - REPORT

QUESTION

- Can you deploy packages on AWS lambda that are bigger than the **50 MB** hard limit?

THE PROCESS

- In order to get around this hard limit for deployment packages, the **Zappa** tool claims 'infinite scaling' for python web apps. It works by automatically packaging up your application and local virtual environment into a zip file, replacing any dependencies with versions precompiled for Lambda, setting up the function handler and necessary WSGI Middleware, uploading the zip archive to S3, creating and managing the necessary Amazon IAM policies and roles, registering it as a new Lambda function, creating a new API Gateway resource, creating WSGI-compatible routes for it, linking it to the new Lambda function, and finally deleting the archive from your S3 bucket.
- Following this guide, I started with a small flask app and scaled up from there. The following are most of the bugs I encountered and how to fix them.

BUGS [SOLVED]

- Make sure zappa is installed in your virtual environment!! And make sure your virtual environment is activated when you deploy!

- Using EC2 Instance vs. local
    - Since zappa uses many of the precompiled lambda libraries this takes away some of the hassel of making sure your libraries are compiled on the right OS (ie, Amazon Linux). You don't have to worry about python modules, pip install on any OS will do the trick. You do have to worry about any additional libraries needed for your app (ie, ones cloned from github)
    - EC2 can be tricky because you need the right permissions. If you launch an EC2 instance is needs:
        - to be large enough for zappa to deploy
        - Create an IAM role with the appropriate permissions (ie, full APIGateway access, can invoke lambda function, api gateway trust relationship)

- When using a Amazon EC2 instance, it will probably be **missing pip/lxml dependencies**
    - **Solution:**

        - https://stackoverflow.com/questions/37592172/installing-lxml-in-python34-on-amazon-ec2
        - `sudo yum install`
            - `gcc`
            - `libxml2-devel`
            - `libxslt-devel`
            - `python-devel`

- When using the **python newspaper module** you need to modify the `settings.py` file. When `newspaper.Article` is called it builds a web scraper. This is stored in lambda's `/tmp` directory not the working directory.
    - **Solution:**
        - https://gist.github.com/JamesChevalier/d212c1998360520dd8a0e67cf6f2fd9c
        - change `[venv]/lib/python2.7/site-packages/newspaper/settings.py` **`DATA_DIRECTORY`** variable from `'.newspaper_scraper'` to `'/tmp/.newspaper_scraper'`

- When **installing zappa** in your virtual environment, you may come across a Memory Error.
    - **Solution:**
        - https://chirale.org/2017/01/15/memory-error-on-pip-install-solved/
        - `pip install zappa --no-cache-dir`

- `A client error (AccessDenied) occurred when calling the PutObject operation: Access Denied`
    - There are no regions in S3, so all bucket names are shared globally, so every person must have their own unique bucket names (i.e., two people cannot have the same bucket name even if they're using different regions or different accounts)
    - You are getting Access Denied error because the bucket name already exists, but you don't have permission to edit it, so Access denied
    - **Solution:**
        - Create the bucket first in the S3 console. You will be notified if the bucket name already exists

- Note: Proxy integrations cannot be configured to transform responses -- this needs to be hardcoded in the app function or mappings can be added in the API Gateway
    - Example: how to return HTML from lambda function
        - https://kennbrodhagen.net/2016/01/31/how-to-return-html-from-aws-api-gateway-lambda/
    - Example: how to pass querystring to lambda (ie. PUT request)
        - https://stackoverflow.com/questions/31329958/how-to-pass-a-querystring-or-route-parameter-to-aws-lambda-from-amazon-api-gatew

- **MemoryError on zappa deploy/update while zipping project**
    - GitHub Issue thread
        - The line `json.dumps(serialized_body).encode(self.DEFAULT_ENCODING)` will crash on a large JSON encoding with a memory error. It's a known issue with doing json.dumps on low-end machines:
            - http://stackoverflow.com/questions/24239613/memoryerror-using-json-dumps
    - **Solution:**
        - You need a bigger EC2 instance! (I had to upgrade from t2.micro to t2.small)

- `DistutilsOptionError: must supply either home or prefix/exec-prefix -- not both`
    - I received this error when I switched from `slim_handler: false` to `slim_handler: true`.
    - **Solution:**
        - https://stackoverflow.com/questions/24257803/distutilsoptionerror-must-supply-either-home-or-prefix-exec-prefix-not-both
        - Switched from python2.7 to python3 and this somehow solved the problem.

- `InvalidParameterValueException: An error occurred (InvalidParameterValueException) when calling the CreateFunction operation: The role defined for the function cannot be assumed by Lambda.`
    - Sometimes there's a **race condition** when you first call deploy. This can be caused by zappa creating an IAM role and then assigning it to something before it's had time to propagate at the AWS end.

    - **Solution:**
        - https://github.com/Miserlou/Zappa/issues/249
        - wait a minute and then run deploy again it works

- `invalid ELF header`
    - The was the case when I cloned and built a shared library that wasn't a python module (ie, MITIE). At this point I was working locally, so I compiled with my MacOS. This was problematic when running lambda code on Amazon Linux.
    - **Solution:**
        - https://stackoverflow.com/questions/29994411/invalid-elf-header-when-using-the-nodejs-ref-module-on-aws-lambda
        - libmitie.so needs to be compiled on EC2 instance

- `'NoneType' object is not callable`
    - Unfortunately, this is a really hard one to debug because there are many possibilities as to where it's going wrong and sometimes they cannot be reproduced
    - Possible solutions
        - You need to jsonify return data if you're using Flask
            - https://stackoverflow.com/questions/34057851/python-flask-typeerror-dict-object-is-not-callable
        - You're calling an object that "doesn't exist" or it's in the wrong directory. Remember, Lambda puts files in the `/tmp` directory
        - You ran out of space in `/tmp` and are indeed missing files. This was my downfall.

BUGS [UNSOLVED]

- ~500MB of non-persistent disk space in /tmp directory
    - when slim handler is true, zappa executes as follows:
        1. Creates two Zip files for S3. The first is a small handler-only zip (apx 5M) that contains the Zappa handler.py file and zappa's dependencies. The second is the application with all of its dependencies
        2. Copies the versioned application zip to proj_current_project.zip
        3. Adds a ZIP_PATH=proj_current_project.zip line to the zappa_settings.py module that is loaded by the handler
        4. When the handler sees the ZIP_PATH= line, it downloads that zip from S3 into /tmp, unzips it, and adds it to path.

    - so the big problem is the size of /tmp which has a hard limit ephemeral disk capacity ("/tmp" space) 512 MB. So if your unzipped project is bigger than that, no bueno. the mitie model alone is 329 MB, but the total size of the project zip file (with everything else) is ~480 MB + 40 MB for the size of the handler, so 520 MB total. SO CLOSE

CONCLUSION

- At end of all this the lambda hard limit claims to be 50MB, but the true hard limit is 512 MB with the zappa work around using the /tmp directory