Flask + Serverless — API in AWS Lambda the easy way

Michal Haták Follow
Jan 1 · 4 min read

If you either need to quickly deploy small API or just decided to migrate your codebase to leverage advantages of AWS Lambda, you can use a powerful combo of Flask and Serverless framework. In fact, any WSGI application such as Django or so.

There are of course other solutions how to do that, either packing and uploading your app by yourself or using Zappa. The process gets a bit more tricky when you realize some dependencies are not compatible (because of Linux on lambda) and you to handle that too. We gonna take a look on basic setup of Serverless framework together with two amazing plugins to make it breeze.

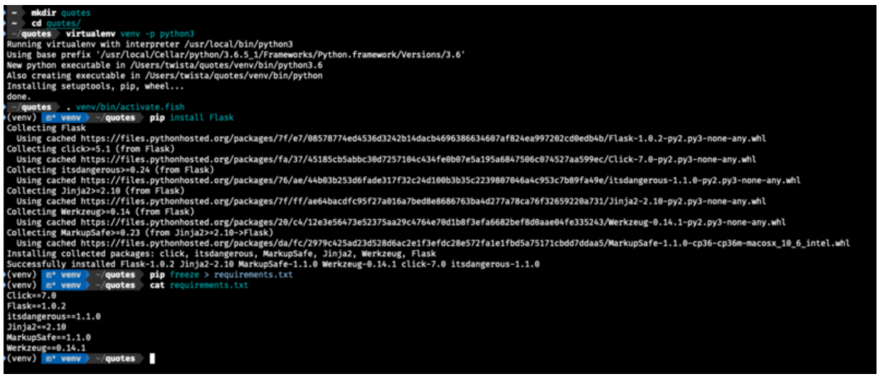
Let's say you already have AWS account. The second thing you need is Serverless framework installed—its darn easy and you can follow the two-step guide here.

Now is time to get hands dirty and make it happen. But, we will need to have Flask app first. Let's build a simple random quote API. Its going to be a very simple example, but the process is very same for larger apps. Also, I will be using virtualenv, but it works with Pipenv too :)

```
# create new dir and jump in
$ mkdir quotes
$ cd quotes/
```

```
# create virtualenv, activate it
$ virtualenv venv -p python3
$ . venv/bin/activate.fish

# install Flask and freeze requirements
$ (venv) pip install Flask
$ (venv) pip freeze > requirements.txt
```

And that's it, we have the foundation now.



```
~ mkdir quotes
~ cd quotes/
~/quotes virtualenv venv -p python3
Running virtualenv with interpreter /usr/local/bin/python3
Using base prefix '/usr/local/Cellar/python/3.6.5_1/Frameworks/Python.framework/Versions/3.6'
New python executable in /Users/twista/quotes/venv/bin/python3.6
Also creating executable in /Users/twista/quotes/venv/bin/python
Installing setuptools, pip, wheel...
done.
~/quotes . venv/bin/activate.fish
(venv) venv ~/quotes pip install Flask
Collecting Flask
  Using cached https://files.pythonhosted.org/packages/7f/e7/08578774ed4536d3242b14dacb4696386634607af824ea997202cd0edb4b/Flask-1.0.2-py2.py3-none-any.whl
Collecting click>=5.1 (from Flask)
  Using cached https://files.pythonhosted.org/packages/fa/37/45185cb5abbc30d7257104c434fe0b07e5a195a6847506c074527aa599ec/Click-7.0-py2.py3-none-any.whl
Collecting itsdangerous>=0.24 (from Flask)
  Using cached https://files.pythonhosted.org/packages/76/ae/44b03b253d6fade317f32c24d100b3b35c2239807046a4c953c7b89fa49e/itsdangerous-1.1.0-py2.py3-none-any.whl
Collecting Jinja2>=2.10 (from Flask)
  Using cached https://files.pythonhosted.org/packages/7f/ff/ae64bacdfc95f27a016a7bed8e8686763ba4d277a78ca76f32659220a731/Jinja2-2.10-py2.py3-none-any.whl
Collecting Werkzeug>=0.14 (from Flask)
  Using cached https://files.pythonhosted.org/packages/20/c4/12e3e56473e52375aa29c4764e70d1b8f3efa6682bef8d0aae04fe335243/Werkzeug-0.14.1-py2.py3-none-any.whl
Collecting MarkupSafe>=0.23 (from Jinja2>=2.10->Flask)
  Using cached https://files.pythonhosted.org/packages/da/fc/2979c425ad23d528d6ac2e1f3efdc28e572fa1e1fbd5a75171cbdd7ddaa5/MarkupSafe-1.1.0-cp36-cp36m-macosx_10_6_intel.whl
Installing collected packages: click, itsdangerous, MarkupSafe, Jinja2, Werkzeug, Flask
Successfully installed Flask-1.0.2 Jinja2-2.10 MarkupSafe-1.1.0 Werkzeug-0.14.1 click-7.0 itsdangerous-1.1.0
(venv) venv ~/quotes pip freeze > requirements.txt
(venv) venv ~/quotes cat requirements.txt
Click==7.0
Flask==1.0.2
itsdangerous==1.1.0
Jinja2==2.10
MarkupSafe==1.1.0
Werkzeug==0.14.1
(venv) venv ~/quotes
```

For the API itself, let's say we have mighty service like that, no rocket science.

```python
from collections import namedtuple
from random import choice

from flask import Flask, jsonify

Quote = namedtuple("Quote", ("text", "author"))

quotes = [
    Quote("Talk is cheap. Show me the code.", "Linus T
    Quote("Programs must be written for people to read
    Quote("Always code as if the guy who ends up maint
          "John Woods"),
    Quote("Give a man a program, frustrate him for a d
    Quote("Progress is possible only if we train ourse
          "Edsger W. Dijkstra")
]
```

Now its time to wire it with the Serverless framework (SLS). Because we have created our API first, it would be hard to use `sls create` command. On the other hand, that could be the case when we have code we want to just deploy or migrate. For having an SLS service we need to create serverless.yml file (in our root) manually. The file will look like this:

```
service: quotes

provider:
  name: aws
  runtime: python3.6
  stage: dev
  region: us-east-1
  memorySize: 128
```

That's almost minimum you need to specify for having a declared service, although we don't have any link to our API (function) yet. But before we will do that, let's install two SLS plugins. The first thing we need is to make Lambda understand WSGI (protocol Flask/Django is using), second is to make SLS pack our python requirements into our deployment package.

```
$ sls plugin install -n serverless-wsgi
$ sls plugin install -n serverless-python-requirements
```

Those two commands make the job and we can see that SLS registered those two in our serverless.yml file.

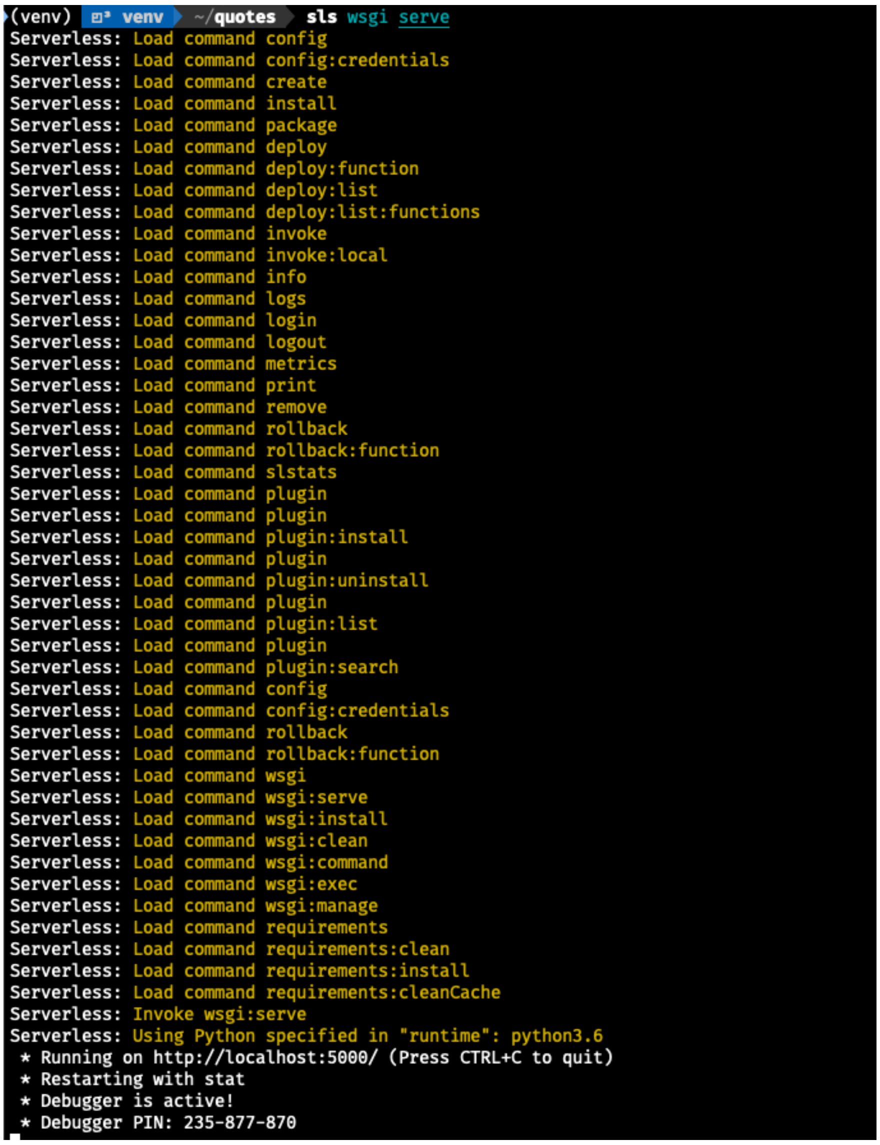
Note: Serverless-wsgi plugin should be able to pack the requirements too, but I have found that the second plugin is more configurable and can eventually pack those in docker (if they need compilation and you are running non-linux system). Also, the plugin supports Pipenv which we are using on one of our projects.

```
service: quotes

provider:
  name: aws
  runtime: python3.6
  stage: dev
  region: us-east-1
  memorySize: 128

plugins:
  - serverless-wsgi
  - serverless-python-requirements

custom:
  wsgi:
    app: app.app
    packRequirements: false

functions:
  app:
    handler: wsgi.handler
    events:
      - http: ANY /
      - http: 'ANY {proxy+}'
```

You can see a new section (custom) which holds configuration for those plugins. The wsgi part is saying where your app is and turning off packing of requirements. The last part (functions) declare what our service contains. We can have more function within one service and also require specific permissions. In this case, we are just saying that all requests will be served by through WSGI handler, which is provided by our installed plugin.

Local development

Before we deploy our API, we can verify everything works locally. As the serverless-wsgi plugin is smart, we can simply run `sls wsgi serve` to have local env. up and running :)

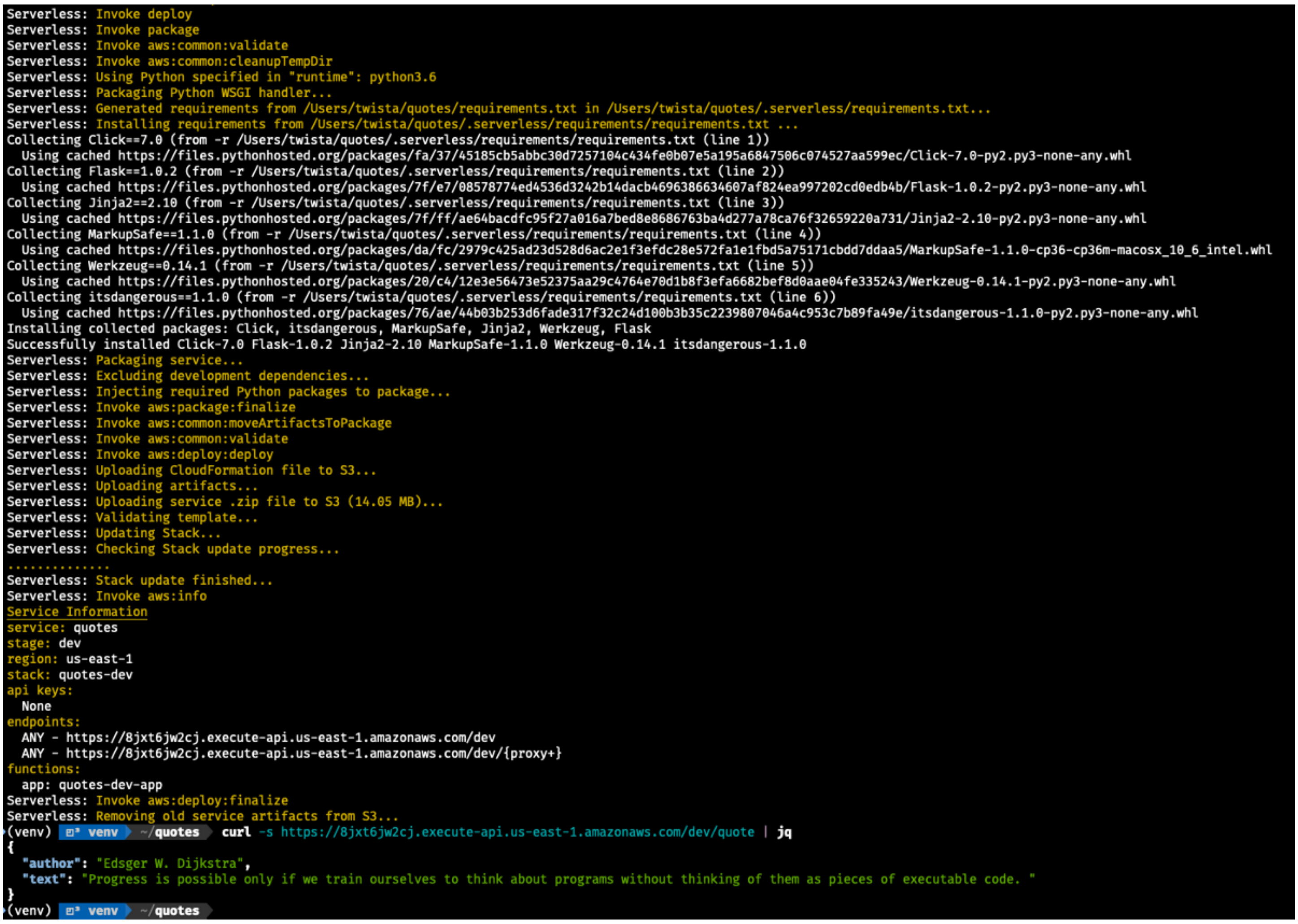


```
(venv)  venv  ~/quotes  sls wsgi serve
Serverless: Load command config
Serverless: Load command config:credentials
Serverless: Load command create
Serverless: Load command install
Serverless: Load command package
Serverless: Load command deploy
Serverless: Load command deploy:function
Serverless: Load command deploy:list
Serverless: Load command deploy:list:functions
Serverless: Load command invoke
Serverless: Load command invoke:local
Serverless: Load command info
Serverless: Load command logs
Serverless: Load command login
Serverless: Load command logout
Serverless: Load command metrics
Serverless: Load command print
Serverless: Load command remove
Serverless: Load command rollback
Serverless: Load command rollback:function
Serverless: Load command slstats
Serverless: Load command plugin
Serverless: Load command plugin
Serverless: Load command plugin:install
Serverless: Load command plugin
Serverless: Load command plugin:uninstall
Serverless: Load command plugin
Serverless: Load command plugin:list
Serverless: Load command plugin
Serverless: Load command plugin:search
Serverless: Load command config
Serverless: Load command config:credentials
Serverless: Load command rollback
Serverless: Load command rollback:function
Serverless: Load command wsgi
Serverless: Load command wsgi:serve
Serverless: Load command wsgi:install
Serverless: Load command wsgi:clean
Serverless: Load command wsgi:command
Serverless: Load command wsgi:exec
Serverless: Load command wsgi:manage
Serverless: Load command requirements
Serverless: Load command requirements:clean
Serverless: Load command requirements:install
Serverless: Load command requirements:cleanCache
Serverless: Invoke wsgi:serve
Serverless: Using Python specified in "runtime": python3.6
 * Running on http://localhost:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 235-877-870
```

Local env. included

Deploy time!

Is as easy as type one command. Run `sls deploy` in your terminal and SLS will do the job. The first deployment will take a bit more, due to the initial setup, but any other one is fast enough.



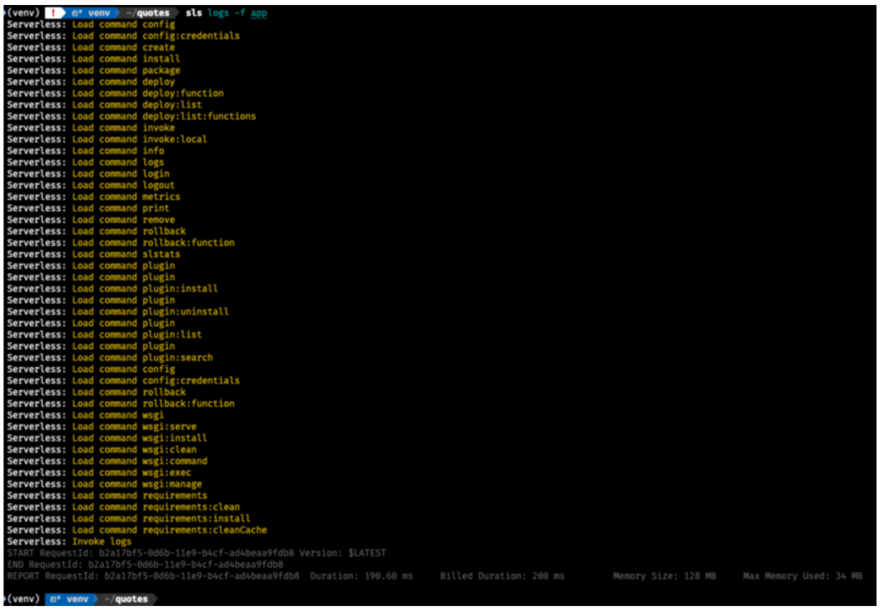
```
Serverless: Invoke deploy
Serverless: Invoke package
Serverless: Invoke aws:common:validate
Serverless: Invoke aws:common:cleanupTempDir
Serverless: Using Python specified in "runtime": python3.6
Serverless: Packaging Python WSGI handler...
Serverless: Generated requirements from /Users/twista/quotes/requirements.txt in /Users/twista/quotes/.serverless/requirements.txt...
Serverless: Installing requirements from /Users/twista/quotes/.serverless/requirements/requirements.txt ...
Collecting Click==7.0 (from -r /Users/twista/quotes/.serverless/requirements/requirements.txt (line 1))
  Using cached https://files.pythonhosted.org/packages/fa/37/45185cb5abbc30d7257104c434fe0b07e5a195a6847506c074527aa599ec/Click-7.0-py2.py3-none-any.whl
Collecting Flask==1.0.2 (from -r /Users/twista/quotes/.serverless/requirements/requirements.txt (line 2))
  Using cached https://files.pythonhosted.org/packages/7f/e7/08578774ed4536d3242b14dacb4696386634607af824ea997202cd0edb4b/Flask-1.0.2-py2.py3-none-any.whl
Collecting Jinja2==2.10 (from -r /Users/twista/quotes/.serverless/requirements/requirements.txt (line 3))
  Using cached https://files.pythonhosted.org/packages/7f/ff/ae64bacdfc95f27a016a7bed8e8686763ba4d277a78ca76f32659220a731/Jinja2-2.10-py2.py3-none-any.whl
Collecting MarkupSafe==1.1.0 (from -r /Users/twista/quotes/.serverless/requirements/requirements.txt (line 4))
  Using cached https://files.pythonhosted.org/packages/da/fc/2979c425ad23d528d6ac2e1f3efdc28e572fa1e1fbd5a75171cbdd7ddaa5/MarkupSafe-1.1.0-cp36-cp36m-macosx_10_6_intel.whl
Collecting Werkzeug==0.14.1 (from -r /Users/twista/quotes/.serverless/requirements/requirements.txt (line 5))
  Using cached https://files.pythonhosted.org/packages/20/c4/12e3e56473e52375aa29c4764e70d1b8f3efa6682bef8d0aae04fe335243/Werkzeug-0.14.1-py2.py3-none-any.whl
Collecting itsdangerous==1.1.0 (from -r /Users/twista/quotes/.serverless/requirements/requirements.txt (line 6))
  Using cached https://files.pythonhosted.org/packages/76/ae/44b03b253d6fade317f32c24d100b3b35c2239807046a4c953c7b89fa49e/itsdangerous-1.1.0-py2.py3-none-any.whl
Installing collected packages: Click, itsdangerous, MarkupSafe, Jinja2, Werkzeug, Flask
Successfully installed Click-7.0 Flask-1.0.2 Jinja2-2.10 MarkupSafe-1.1.0 Werkzeug-0.14.1 itsdangerous-1.1.0
Serverless: Packaging service...
Serverless: Excluding development dependencies...
Serverless: Injecting required Python packages to package...
Serverless: Invoke aws:package:finalize
Serverless: Invoke aws:common:moveArtifactsToPackage
Serverless: Invoke aws:common:validate
Serverless: Invoke aws:deploy:deploy
Serverless: Uploading CloudFormation file to S3...
Serverless: Uploading artifacts...
Serverless: Uploading service .zip file to S3 (14.05 MB)...
Serverless: Validating template...
Serverless: Updating Stack...
Serverless: Checking Stack update progress...
..............
Serverless: Stack update finished...
Serverless: Invoke aws:info
Service Information
service: quotes
stage: dev
region: us-east-1
stack: quotes-dev
api keys:
  None
endpoints:
  ANY - https://8jxt6jw2cj.execute-api.us-east-1.amazonaws.com/dev
  ANY - https://8jxt6jw2cj.execute-api.us-east-1.amazonaws.com/dev/{proxy+}
functions:
  app: quotes-dev-app
Serverless: Invoke aws:deploy:finalize
Serverless: Removing old service artifacts from S3...
(venv)  venv  ~/quotes  curl -s https://8jxt6jw2cj.execute-api.us-east-1.amazonaws.com/dev/quote | jq
{
  "author": "Edsger W. Dijkstra",
  "text": "Progress is possible only if we train ourselves to think about programs without thinking of them as pieces of executable code. "
}
(venv)  venv  ~/quotes
```

Deploy & test

And that's it. We have our service up & running with the power of AWS Lambda behind.

Final Notes

- you can easily make python-requirements use docker to pack requirements which require compilation
- if you get an internal error or want to see recent logs. Open CloudWatch in your AWS console or just type `sls logs -f app`



```
(venv) ! ⊡* venv ~/quotes sls logs -f app
Serverless: Load command config
Serverless: Load command config:credentials
Serverless: Load command create
Serverless: Load command install
Serverless: Load command package
Serverless: Load command deploy
Serverless: Load command deploy:function
Serverless: Load command deploy:list
Serverless: Load command deploy:list:functions
Serverless: Load command invoke
Serverless: Load command invoke:local
Serverless: Load command info
Serverless: Load command logs
Serverless: Load command login
Serverless: Load command logout
Serverless: Load command metrics
Serverless: Load command print
Serverless: Load command remove
Serverless: Load command rollback
Serverless: Load command rollback:function
Serverless: Load command slstats
Serverless: Load command plugin
Serverless: Load command plugin
Serverless: Load command plugin:install
Serverless: Load command plugin
Serverless: Load command plugin:uninstall
Serverless: Load command plugin
Serverless: Load command plugin:list
Serverless: Load command plugin
Serverless: Load command plugin:search
Serverless: Load command config
Serverless: Load command config:credentials
Serverless: Load command rollback
Serverless: Load command rollback:function
Serverless: Load command wsgi
Serverless: Load command wsgi:serve
Serverless: Load command wsgi:install
Serverless: Load command wsgi:clean
Serverless: Load command wsgi:command
Serverless: Load command wsgi:exec
Serverless: Load command wsgi:manage
Serverless: Load command requirements
Serverless: Load command requirements:clean
Serverless: Load command requirements:install
Serverless: Load command requirements:cleanCache
Serverless: Invoke logs
START RequestId: b2a17bf5-0d6b-11e9-b4cf-ad4beaa9fdb8 Version: $LATEST
END RequestId: b2a17bf5-0d6b-11e9-b4cf-ad4beaa9fdb8
REPORT RequestId: b2a17bf5-0d6b-11e9-b4cf-ad4beaa9fdb8  Duration: 190.60 ms     Billed Duration: 200 ms        Memory Size: 128 MB     Max Memory Used: 34 MB

(venv) ⊡* venv ~/quotes
```

logs from our API

- in most cases, you can fit into AWS free tier—which is nice especially for prototypes
- you can specify stages to distinguish between staging and production environment

· · ·

I hope it helps you to start with Lambda and if it was clear enough. If you like the article, you can follow me on Twitter.