Text Extraction Service

Themes

Create a service to extract text from images. Rationale:

1) To learn/demonstrate knowledge/skills of Python, Flask, AWS, REST
2) Serve part of a wider service to infer data from images, which also demonstrates other technical knowledge/skills.

Problem space

1) Require Service extract text from images; very low volume
2) Privacy is a core need (images may contain sensitive data)
3) Learn/demonstrate knowledge of Python, WebApps, AWS (Textract, S3, others), REST
4) Incur minimal costs
5) Need to balance complexity wrt effort/duration to develop and the value of demonstrating this skill.
6) In order to demo, us github for code repository, and public cloud to host application
7)

Solution Space

Source requirement	High Level Solution Decisions
Require Service extract text from images	Use AWS Text Extract (has low cost after few months)
Privacy is a core need (images may contain sensitive data)	Server must be protected from unauthorised client access. Minimum: Server must validate client (using PKI). No credentials etc in source code/github etc. Ideal: User must authenticate to server and server must enforce user authorisation
Learn/demonstrate Python,WebApps, AWS, REST	Use Python & relevant packages, Flask as WebApps, PyCharm and Pytest to drive development, AWS SDK to manage cloud instances and deployment.
Incur minimal costs	Use community/free offerings of PythonCharm, Github, AWS
Need to balance complexity wrt effort/duration to develop and the value of demonstrating this skill.	Focus on production quality code, but compromise on detail and less important error paths. Develop automated tests in key areas. **Scaling is not important.**

Decisions

Prototype:

1) Python text extract programme. Discovered that the AWS Textract service cannot accurately identify the lines of a credit card statement, instead returning a set of lines for ‘cells’. Need to introduce a simple algorithm to join up these cells, but due to page angle etc its not 100% effective, leaving a small number of lines incomplete. Decide to accept this rather explore other approaches to building lines.
2) Flask REST application. Initial development on local dev machine. Move to AWS when prototype working
3) Automated testing: in module tests combined with API level automated tests using Pytest/Urllib.
4) Cloud hosting: build AWS resource and deployment pipeline

Low level decisions/considerations : 1

1) **AWS access control:**Access to AWS resource to be controlled by AWS IAM identitied and permissions.
2) **URI structure and mapping:** text extracted is not an independant resource but a subresource of image. Mappings (controlled by Flask route/function mappings) are:

   /image/<image-file> POST(file): return key, GET(key): return image-file

   /image/<image-file>/textExtracted/<text-file> POST(): return key (creates textExtracted resource); GET(key): return text-file

   /image/<image-file> DELETE(key): (delete image and text extracted0

3) **Service design:** Separate REST and true Model
4) **Configuration;** use Python config model; but model on environments and separate confidentiality config that can be controlled during deployment.