Opstream Home Assignment

The purpose of the following challenge is to evaluate your proficiency in the tools we commonly use here at Opstream.

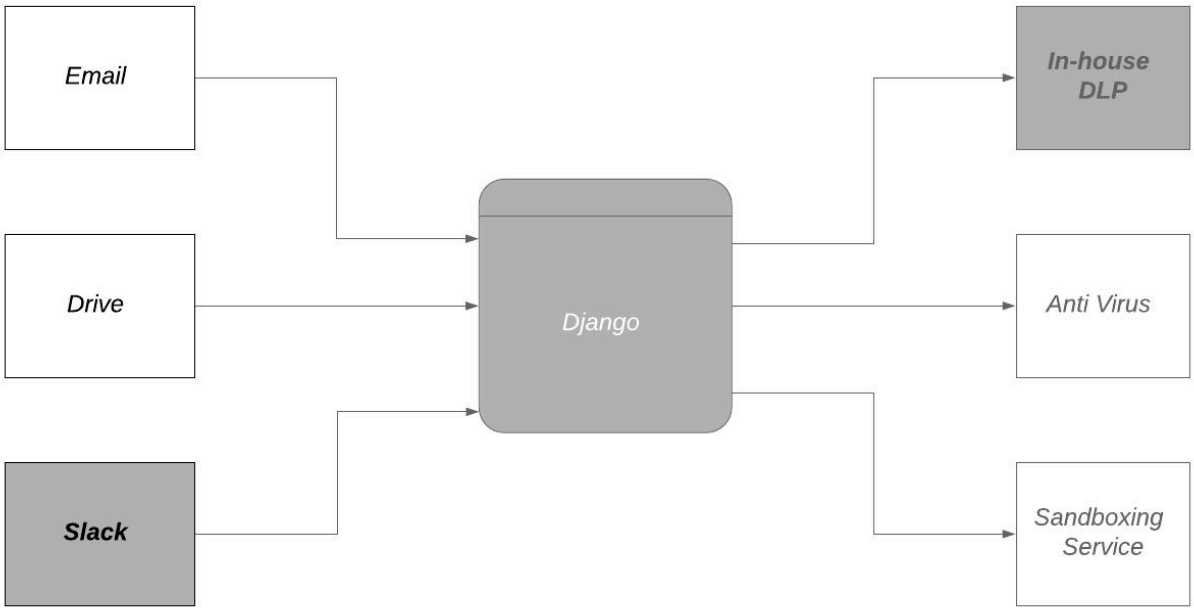
We expect you to push your code to a Git repository and share it with us, and then kindly delete it so we can use this assignment in the future with other candidates as well.

Please feel free to write instructions in a README file describing how to run the project and how to run its tests.

Goal

Create a system that secures an organization's communication channels by listening to files and messages flowing in and scanning them using a set of security tools.

In V1, the system should use Slack channels of an organization as its only input AND a Data Loss Prevention (DLP) tool that will be developed in house. However, we should wrap and use the DLP like another external security tool.



Asynchronous Distributed Tasks

Interacting with third party services does often happen using APIs, using a framework like Celery is not enough when many requests are required since it's blocking. With async code it is possible to start new requests while waiting for the current one.

```
import asyncio
import json



class Manager:

   def __init__(self, queue_name: str, tasks: dict):
       self.loop = asyncio.get_event_loop()
       self.queue = queue_name
       self.tasks = tasks

   async def _get_messages(self):
       """Read and pop messages from SQS queue
       """
       raise NotImplementedError

   async def main(self):
       """For a given task:
       >>> async def say(something):
               pass

       Messages from queue are expected to have the format:
       >>> message = dict(task='say', args=('something',), kwargs={})
       >>> message = dict(task='say', args=(), kwargs={'something': 'something else'})
       """

       while True:
           messages = await self._get_messages()
           for message in messages:
               body = json.loads(message['Body'])

               task_name = body.get('task')
               args = body.get('args', ())
```

```
                kwargs = body.get('kwargs', {})

                task = self.tasks.get(task_name)
                self.loop.create_task(task(*args, **kwargs))
            await asyncio.sleep(1)
```

Task

Follow the next steps:

1. Open a free Slack account https://slack.com/pricing/free
2. Use Slack events api https://api.slack.com/events-api to listen to messages
3. Create a simple Data Loss Prevention tool that given a file, open its content and try to look for patterns
(for example: a credit card number), using a list of regular expressions, make it possible to manage those
patterns using django admin
4. Use django admin to also show messages that were caught by the DLP tool, show the message, its content and
the pattern that caught it
5. Use the class above to create a container with distributed tasks to search for leaks in files and messages
5. Write unit tests where consider appropriate
6. **BONUS**: add an action flow, so when DLP is giving a negative response, saying that the message contains a
leak, the system should switch the message on slack with a message saying that the original message was blocked

Tools

Solve this assignment using the following tools:

* Python 3: https://www.python.org/
* Git: https://git-scm.com/
* Docker: https://www.docker.com/
* Docker Compose: https://docs.docker.com/compose

* Django: https://www.djangoproject.com/
* Postgres: https://www.postgresql.org/
* SQS: https://aws.amazon.com/sqs/