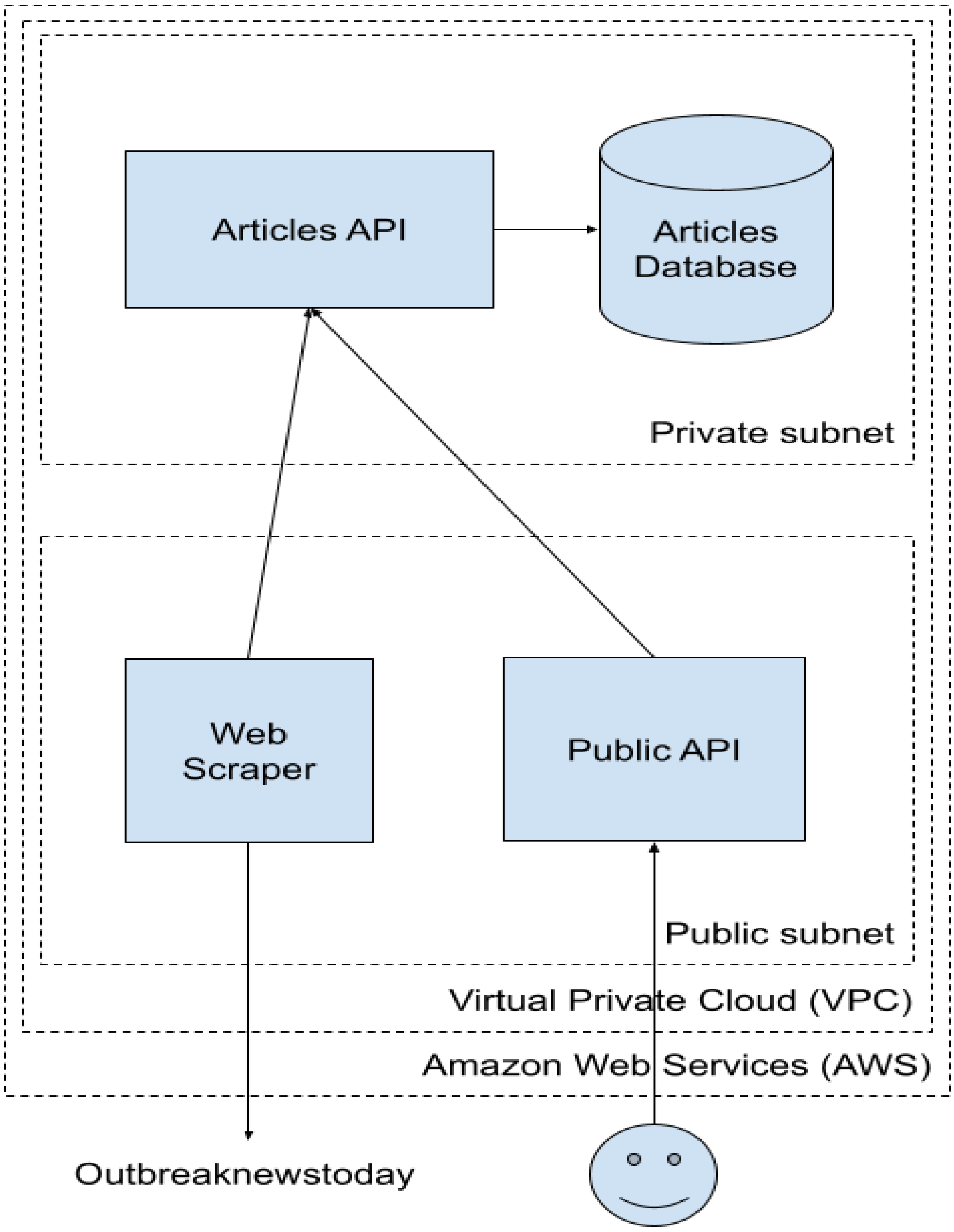
Design Details

Development of API Module and Deployment

We intend to develop the web service module using Flask-RESTPlus and deploy it to an AWS instance.



Sample HTTP Calls

GET Request:

Query Parameter	**Format**	**Required**	**Notes**
location	String	yes	
start_date	yyyy-MM-ddTHH:mm:ss	yes	
end_date	yyyy-MM-ddTHH:mm:ss	yes	
keyterms	Comma separated list of strings e.g. “Anthrax,Zika”	no	non-case sensitive

200 Response:

```
{
        articles: [<object::article>]
}
```

Article Object:

```
{
        url: <string>,
        date_of_publication: <string::date>,
        headline: <string>,
        main_text: <string>,
        reports: [<object::report>]
}
```

Report Object:

```
{
        diseases: [<string::disease>],
        syndromes: [<string::syndrome>],
        event_date: <string::date>,
        locations: [<object::location>]
}
```

Location Object:

```
{
        country: <string>,
        location: <string>
}
```

204 Response:
No Content

400 Response:
Bad Request due to malformed input

Development Stack

Language

We will be using Python as the implementation language. Principally because the team is familiar with the language, but also because there are widely supported libraries such as Scrapy and Flask-RESTPlus for web scraping and backend development. Python also provides good frameworks for data visualisation and data manipulation such as Spacy and NLTK for natural language processing.

Python also has good scalability and has good unit testing frameworks which simplify performing code and testing simultaneously by adopting the test-driven development (TDD) approach.

Libraries

For backend, we will mainly use Flask-RESTPlus which provides more simplicity, flexibility and fine-grained control. For web scraping, we will use Scrapy which is an effective tool specifically created for downloading, cleaning and saving data from the web. We use JSON library to store data in JSON format.

We'll be using Swagger for the API documentation and PyTest for unit testing. That way, the development will be in a single language reducing the language curve.

Deployment Environment

AWS supports a wide range of operating systems builtin as virtual machines which allows customers to focus on their application. For the database, we will use DynamoDB for better compatibility with AWS cloud. AWS Comprehend can also be used for natural language processing.

Design considerations

The "Outbreak News Today" website has articles under several categories including "US News", "Asia" and "Headlines". Scraping individual region specific categories would result in poor geographical coverage or duplication and a large number of irrelevant articles. "Headlines" on the other hand appears to contain mostly outbreak reports though with some missing. It may also be possible to search for the desired keywords through the website's own search function.

We have previously used REST APIs and MongoDB. However, within the AWS ecosystem, there are frameworks which allow for easier API creation and deployment.

For example, AWS AppSync makes creating GraphQL APIs from AWS DynamoDB queries simple. Furthermore, AWS CloudSearch makes searching through unstructured text easier.

Lambda functions may also be useful for the web scraper in particular. This would allow us to programmatically create a timed scraping task using CloudWatch events rather than going through the operating system. Otherwise, it may be possible to run the web scraper only once. For the cost consideration, the frequency of the request for making extracting articles from the website to update the database will be limited to 1M per month.