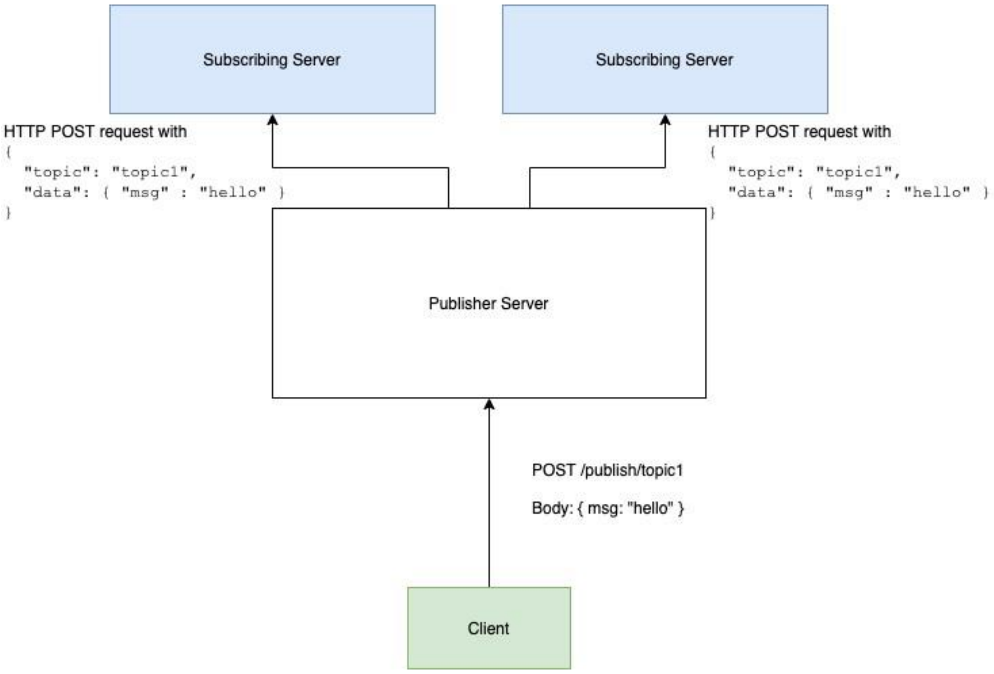
⍰

Pangaea Take-home assignment

For this challenge we'll be creating a HTTP notification system. A server (or set of servers) will keep track of `topics -> Subscribers` where a topic is a string and a subscriber is an HTTP endpoint. When a message is published on a topic, it should be forwarded to all subscriber endpoints.

The backend teams at Pangaea currently use PHP ⍰Laravel) and NodeJS and Java. Your take home must use Java. The tools and frameworks within these languages are up to you.



Publisher Server Endpoints

Create a subscription

`POST /subscribe/{topic}`

Expected Body

```
{
  url: string
}
```

Success Response

```
{
  url: string,
  topic: string
}
```

Example Request / Response

```
POST /subscribe/topic1
// body
{
  url: "http://mysubscriber.test"
}
```

Response:

```
{
  url: "http://mysubscriber.test",
  topic: "topic1"
}
```

Publish message to topic

`POST /publish/{topic}`

POSTing to this endpoint should send HTTP requests to any current subscribers for the specified `{topic}`. It is valid to publish to topics with no subscribers. If there are multiple subscribers they should all be notified.

Expected Body

```
// must be a javascript object {}, it can contain any keys and have nested data
{
  [key: string]: any
}
```

Expected Response

- Should give a meaningful HTTP response code based on whether the publish was successful or not

Payload sent to subscribers

```
{
  topic: string
  data: object // whatever data was sent in the publish body
}
```

Full Examples:

```
./start-server.sh
curl -X POST -H "Content-Type: application/json" -d '{ "url": "http://localhost:9000/test1"}' http://localhost:8000/subscribe/topic1 curl -
X POST -H "Content-Type: application/json" -d '{ "url": "http://localhost:9000/test2"}' http://localhost:8000/subscribe/topic1 curl -X POST
-H "Content-Type: application/json" -d '{"message": "hello"}' http://localhost:8000/publish/topic1
```

The above example assumes that the `start-server.sh` script starts the publisher server on port 8000 and another server is running on port 9000 (subscriber). The subscriber will be getting data forwarded to it when its corresponding topic is published, which it will then receive and print the data to verify everything is working at the test1 and test2 endpoints.