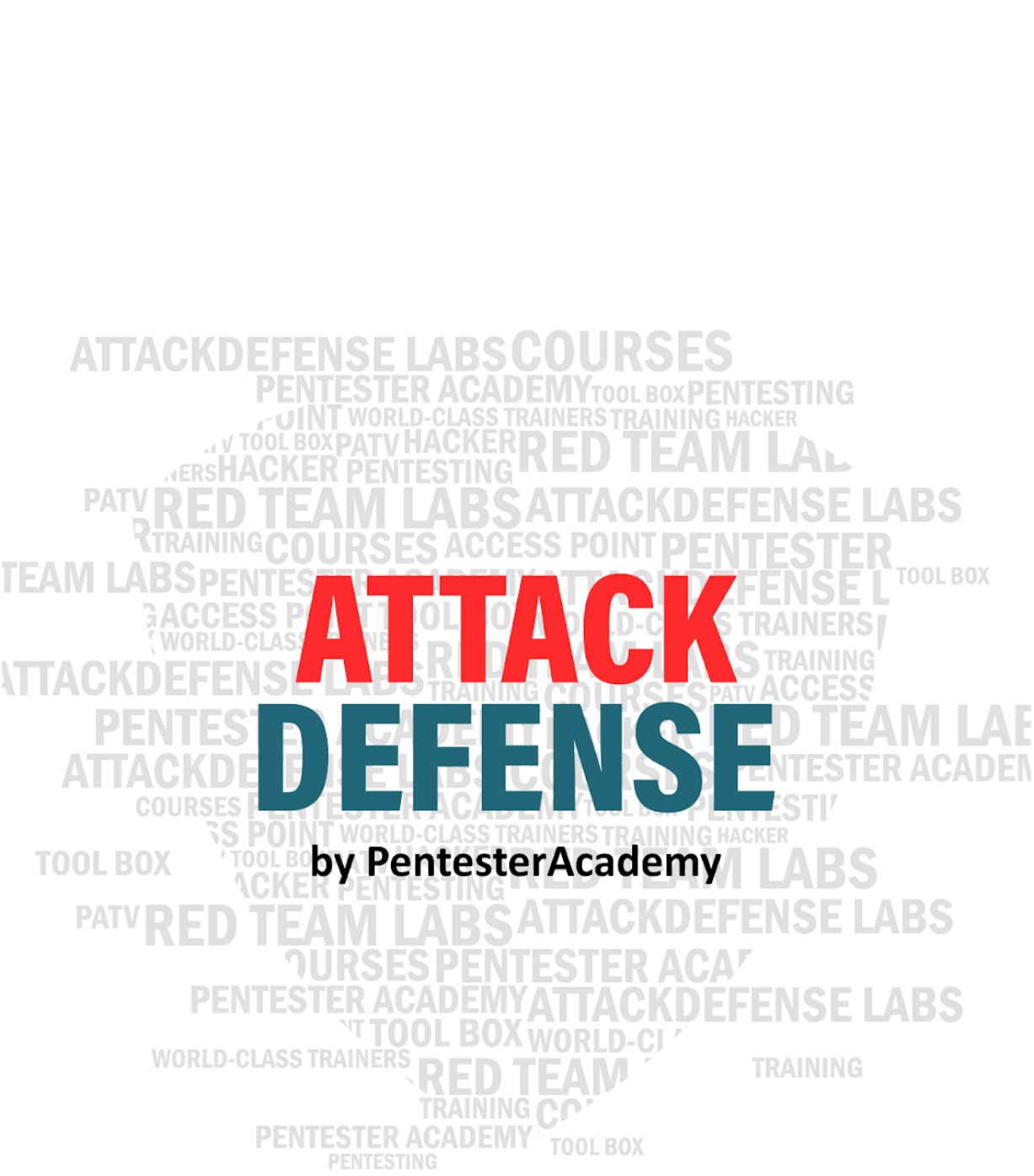
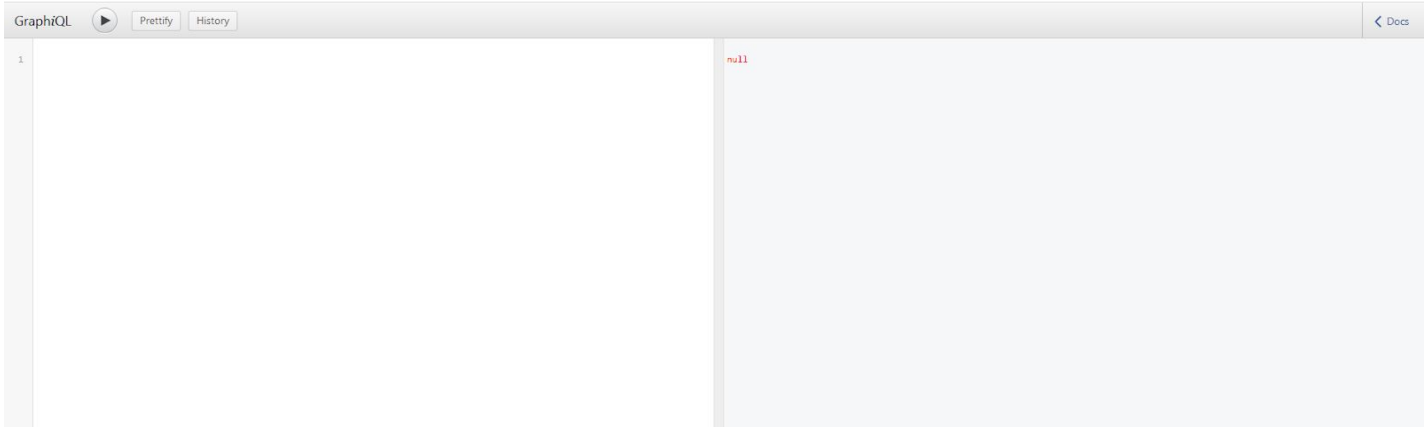

ATTACK
DEFENSE
by PentesterAcademy

Name	GraphQL Basics I
URL	https://attackdefense.com/challengedetails?cid=1991
Type	REST: GraphQL

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

When the lab is launched, the GraphiQL console is provided.



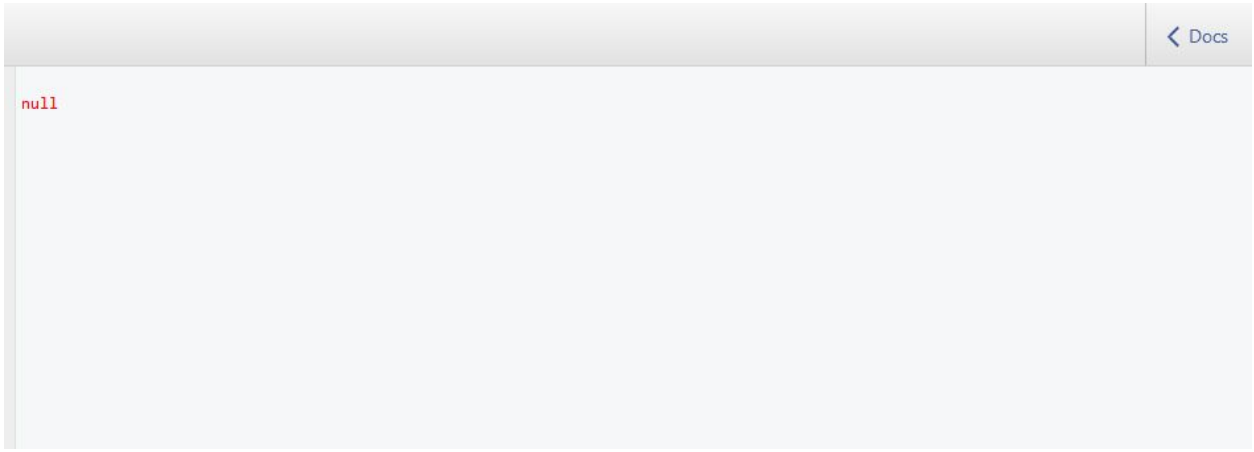
This console could be used to write the GraphQL queries.

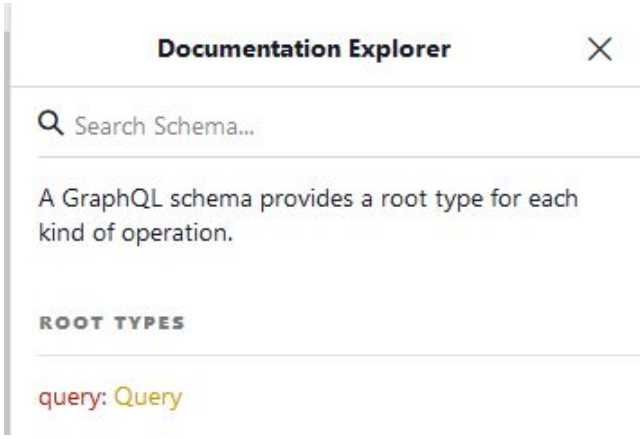
Step 1: Exploring the GraphQL schema.

As it is mentioned in the challenge description that the introspection queries are not disabled. That means that the GraphQL schema could be accessed from the console and auto-completion would also work.

1. Exploring the GraphQL Schema:

Click on the Docs link on the top right of the console window.

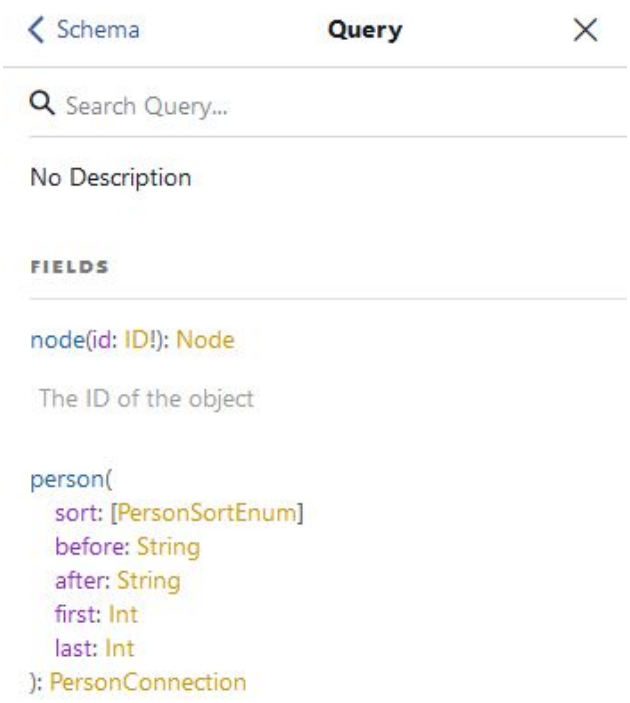




The Documentation Explorer panel could be used to view the GraphQL schema.

Click on the “Query” keyword (orange link).

That leads to the following page:

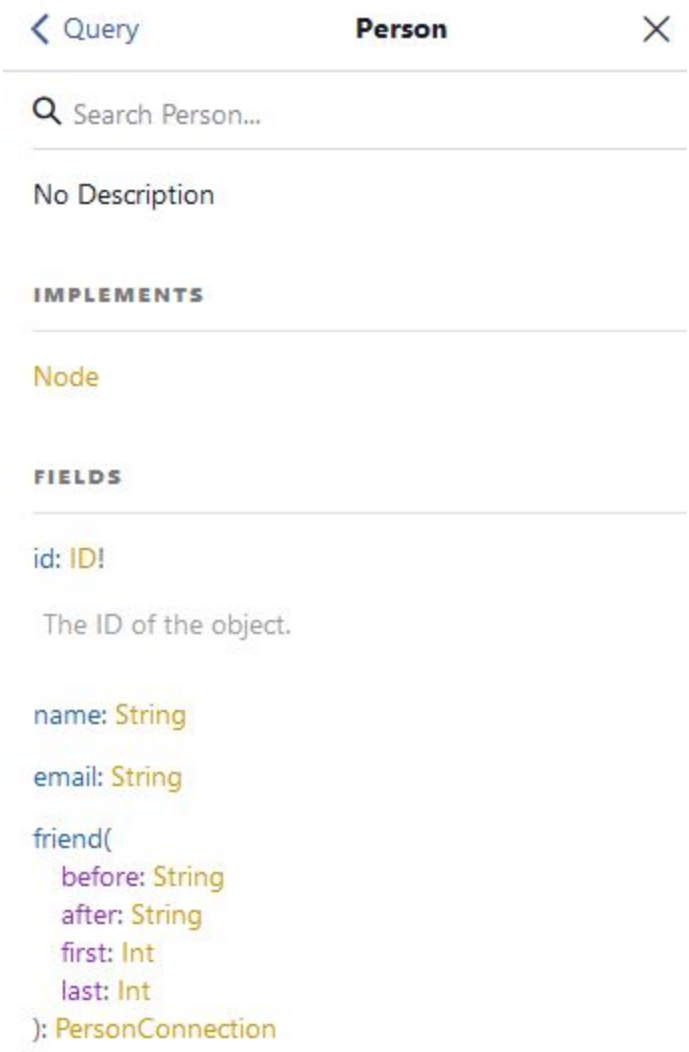


The above page shows that there is a node and a person field.

We can also make searches using the provided search bar.

Searching for “person” keyword in the search bar:

Select the last result: Person



That shows up the Person object. It contains the following fields:

id, name, email, friend (which is itself an object).

Step 2: Making queries to the backend database.

Writing queries in the console window to fetch the desired results from the backend database:

1. Use the following query to fetch all the person objects and display their name and email fields.

Query:

```
{
  person {
            edges {
            node {
            name
            email
            }
            }
  }
}
```

Response:

```
{
  "data": {
    "person": {
      "edges": [
        {
          "node": {
            "name": "John Doe",
            "email": "johndoe@example.com"
          }
        },
        {
          "node": {
            "name": "David Smith",
            "email": "davidsmith@example.com"
          }
        },
        {
          "node": {
            "name": "Michael Jones",
            "email": "michaeljones@example.com"
          }
        },
        {
          "node": {
            "name": "Chris Johnson",
            "email": "chrisjohnson@example.com"
          }
        },
```


...

```
        {
          "node": {
            "name": "Paul Williams",
            "email": "paulwilliams@example.com"
          }
        },
        {
          "node": {
            "name": "Daniel Rodriguez",
            "email": "danielrodriguez@example.com"
          }
        },
        {
          "node": {
            "name": "James Garcia",
            "email": "jamesgarcia@example.com"
          }
        },
        {
          "node": {
            "name": "Maria Gonzalez",
            "email": "mariagonzalez@example.com"
          }
        }
      ]
    }
  }
}
```


Note: While typing the queries, the editor would provide suggestions (auto-completion) since the Introspection Queries are not disabled (in default settings).

2. Use the following query to fetch all the person objects and display their name and email fields and all of their friends names and email fields.

Query:

```
{
  person {
        edges {
        node {
        name
        email
        friend {
        edges {
        node {
        name
```

```
            email
            }
            }
            }
            }
            }
  }
}
```

Response:


```
{
  "data": {
    "person": {
      "edges": [
        {
          "node": {
            "name": "John Doe",
            "email": "johndoe@example.com",
            "friend": {
              "edges": [
                {
                  "node": {
                    "name": "David Smith",
                    "email": "davidsmith@example.com"
                  }
                },
                {
                  "node": {
                    "name": "Michael Jones",
                    "email": "michaeljones@example.com"
                  }
                }
              ]
            }
          }
        },
        {
          "node": {
            "name": "David Smith",
            "email": "davidsmith@example.com",
            "friend": {
              "edges": [
                {
                  "node": {
                    "name": "John Doe",
                    "email": "johndoe@example.com"
                  }
                },
```

```
        {
          "node": {
            "name": "Maria Gonzalez",
            "email": "mariagonzalez@example.com",
            "friend": {
              "edges": [
                {
                  "node": {
                    "name": "John Doe",
                    "email": "johndoe@example.com"
                  }
                },
                {
                  "node": {
                    "name": "David Smith",
                    "email": "davidsmith@example.com"
                  }
                },
                {
                  "node": {
                    "name": "Michael Jones",
                    "email": "michaeljones@example.com"
                  }
                },
                {
                  "node": {
                    "name": "Chris Johnson",
                    "email": "chrisjohnson@example.com"
                  }
                },
                {
                  "node": {
                    "name": "Mike Lee",
                    "email": "mikelee@example.com"
                  }
                }
              ]
            }
          }
        }
      ]
    }
  }
}
```


3. Use the following query to fetch all the person objects and display their name and email fields and all of their friends names and email fields and also fetching the records for their friends.

Query:

```
{
  person {
        edges {
        node {
        name
        email
        friend {
        edges {
        node {
        name
        email
        friend {
        edges {
                node {
                name
                email
                }
        }
        }
        }
        }
        }
        }
  }
}
```

Response:


```
{
  "data": {
    "person": {
      "edges": [
        {
          "node": {
            "name": "John Doe",
            "email": "johndoe@example.com",
            "friend": {
              "edges": [
                {
```


...

```
        "node": {
          "name": "David Smith",
          "email": "davidsmith@example.com",
          "friend": {
            "edges": [
              {
                "node": {
                  "name": "John Doe",
                  "email": "johndoe@example.com"
                }
              },
              {
                "node": {
                  "name": "Chris Johnson",
                  "email": "chrisjohnson@example.com"
                }
              },
              {
                "node": {
                  "name": "Paul Williams",
                  "email": "paulwilliams@example.com"
                }
              }
            ]
          }
        }
      },
```


...


```
    {
      "node": {
        "name": "Mike Lee",
        "email": "mikelee@example.com",
        "friend": {
          "edges": [
            {
              "node": {
                "name": "Daniel Rodriguez",
                "email": "danielrodriguez@example.com"
              }
            },
            {
              "node": {
                "name": "Maria Gonzalez",
                "email": "mariagonzalez@example.com"
              }
            }
          ]
        }
      }
    }
  ]
}
}
}
```

Note:

1. The response we get has the same shape as the request. So when using GraphQL, we always know what type of response to expect.
2. GraphQL always returns only the data wwe asked for and nothing more.

Conclusion: In this lab we have explored the basics of GraphQL, mainly making different queries and the documentation to find out more information on any object (provided that the introspection queries are enabled, which is true for default settings).

References:

1. GraphQL (https://graphql.org)