GraphQL

Marius Landwehr

iOS Developer @grandcentrix

Twitter @mariusLAN

mrs-.github.io

GraphQL

GraphQL

Facebook

What?
How?
Should?

GraphQL

What?

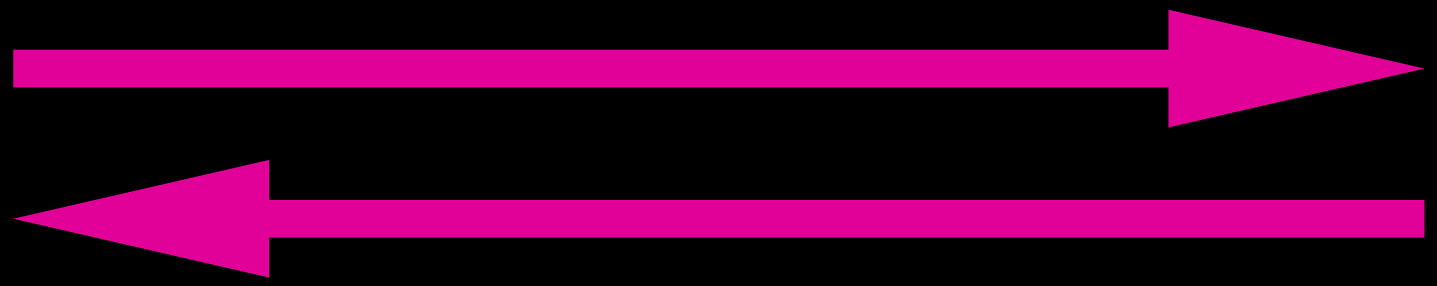
GraphQL

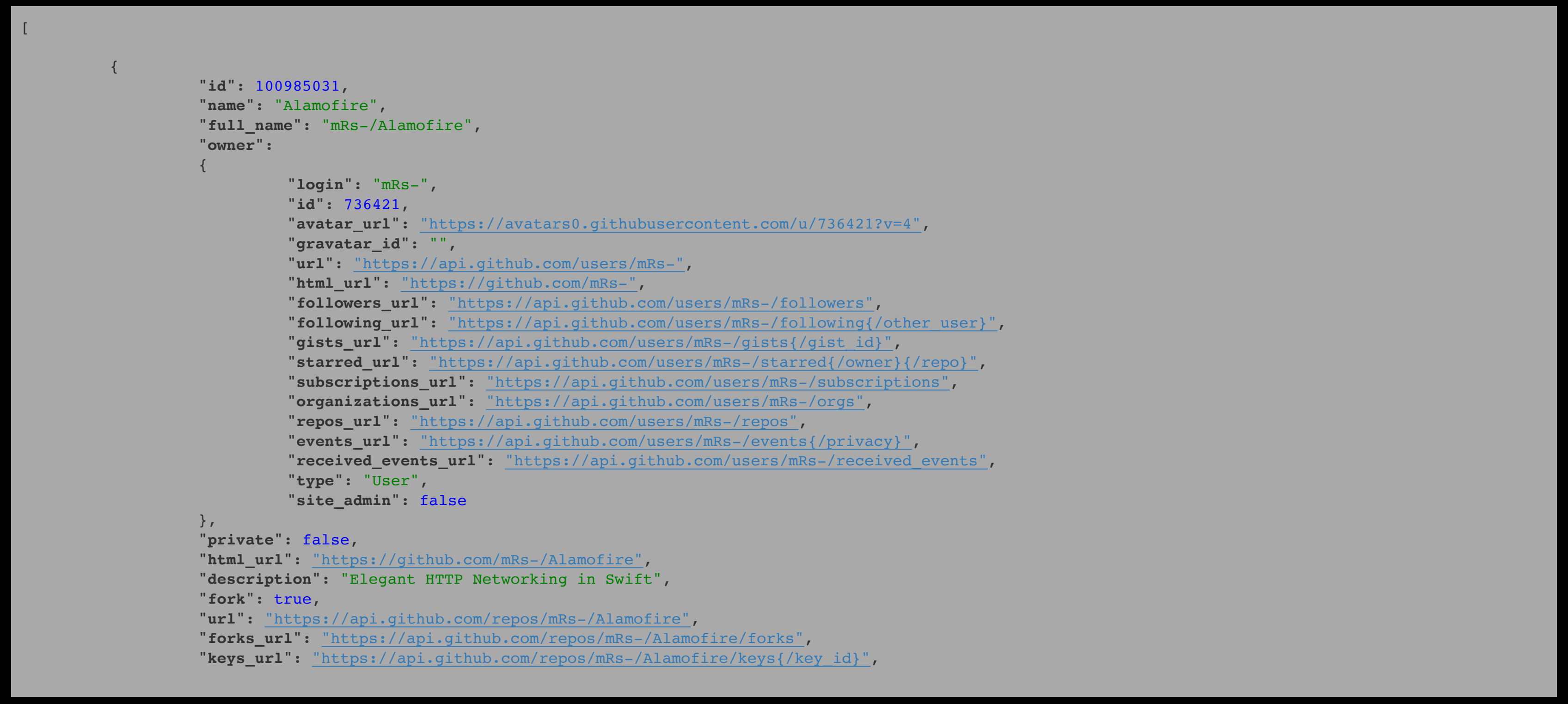
What?

Client

Server

GET /users/mRs-/repos





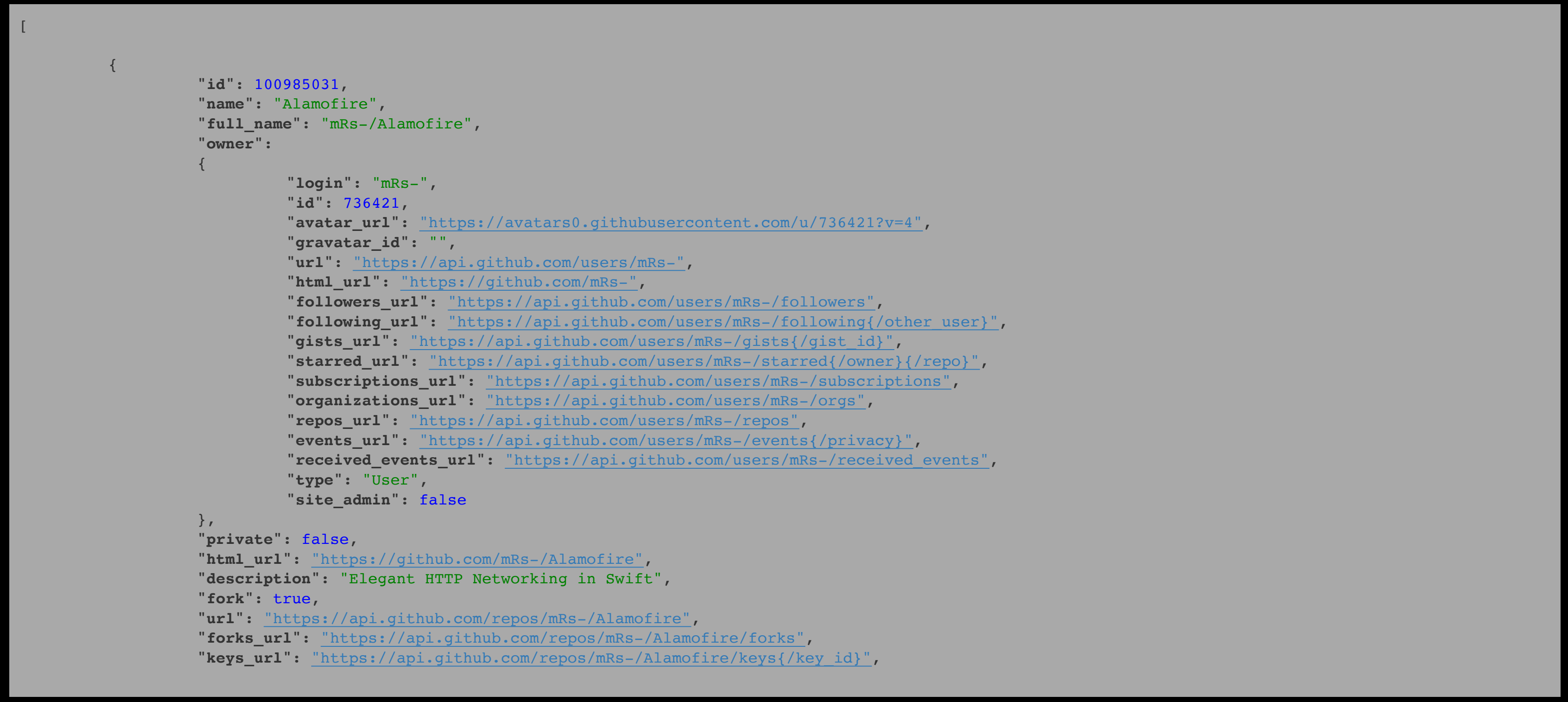
```
[
        {
                "id": 100985031,
                "name": "Alamofire",
                "full_name": "mRs-/Alamofire",
                "owner":
                {
                        "login": "mRs-",
                        "id": 736421,
                        "avatar_url": "https://avatars0.githubusercontent.com/u/736421?v=4",
                        "gravatar_id": "",
                        "url": "https://api.github.com/users/mRs-",
                        "html_url": "https://github.com/mRs-",
                        "followers_url": "https://api.github.com/users/mRs-/followers",
                        "following_url": "https://api.github.com/users/mRs-/following{/other_user}",
                        "gists_url": "https://api.github.com/users/mRs-/gists{/gist_id}",
                        "starred_url": "https://api.github.com/users/mRs-/starred{/owner}{/repo}",
                        "subscriptions_url": "https://api.github.com/users/mRs-/subscriptions",
                        "organizations_url": "https://api.github.com/users/mRs-/orgs",
                        "repos_url": "https://api.github.com/users/mRs-/repos",
                        "events_url": "https://api.github.com/users/mRs-/events{/privacy}",
                        "received_events_url": "https://api.github.com/users/mRs-/received_events",
                        "type": "User",
                        "site_admin": false
                },
                "private": false,
                "html_url": "https://github.com/mRs-/Alamofire",
                "description": "Elegant HTTP Networking in Swift",
                "fork": true,
                "url": "https://api.github.com/repos/mRs-/Alamofire",
                "forks_url": "https://api.github.com/repos/mRs-/Alamofire/forks",
                "keys_url": "https://api.github.com/repos/mRs-/Alamofire/keys{/key_id}",
```

What?

Client

Server

GET /users/mRs-/repos



```
[
        {
                "id": 100985031,
                "name": "Alamofire",
                "full_name": "mRs-/Alamofire",
                "owner":
                {
                        "login": "mRs-",
                        "id": 736421,
                        "avatar_url": "https://avatars0.githubusercontent.com/u/736421?v=4",
                        "gravatar_id": "",
                        "url": "https://api.github.com/users/mRs-",
                        "html_url": "https://github.com/mRs-",
                        "followers_url": "https://api.github.com/users/mRs-/followers",
                        "following_url": "https://api.github.com/users/mRs-/following{/other_user}",
                        "gists_url": "https://api.github.com/users/mRs-/gists{/gist_id}",
                        "starred_url": "https://api.github.com/users/mRs-/starred{/owner}{/repo}",
                        "subscriptions_url": "https://api.github.com/users/mRs-/subscriptions",
                        "organizations_url": "https://api.github.com/users/mRs-/orgs",
                        "repos_url": "https://api.github.com/users/mRs-/repos",
                        "events_url": "https://api.github.com/users/mRs-/events{/privacy}",
                        "received_events_url": "https://api.github.com/users/mRs-/received_events",
                        "type": "User",
                        "site_admin": false
                },
                "private": false,
                "html_url": "https://github.com/mRs-/Alamofire",
                "description": "Elegant HTTP Networking in Swift",
                "fork": true,
                "url": "https://api.github.com/repos/mRs-/Alamofire",
                "forks_url": "https://api.github.com/repos/mRs-/Alamofire/forks",
                "keys_url": "https://api.github.com/repos/mRs-/Alamofire/keys{/key_id}",
```

GET /repos/mRs-/Alamofire/contributors

[{ "login": "mRs-", "id": 736421, "avatar_url": "https://avatars0.githubusercontent.com/u/736421?v=4",
"gravatar_id": "", "url": "https://api.github.com/users/mRs-", "html_url": "https://github.com/mRs-",
"followers_url": "https://api.github.com/users/mRs-/followers", "following_url": "http

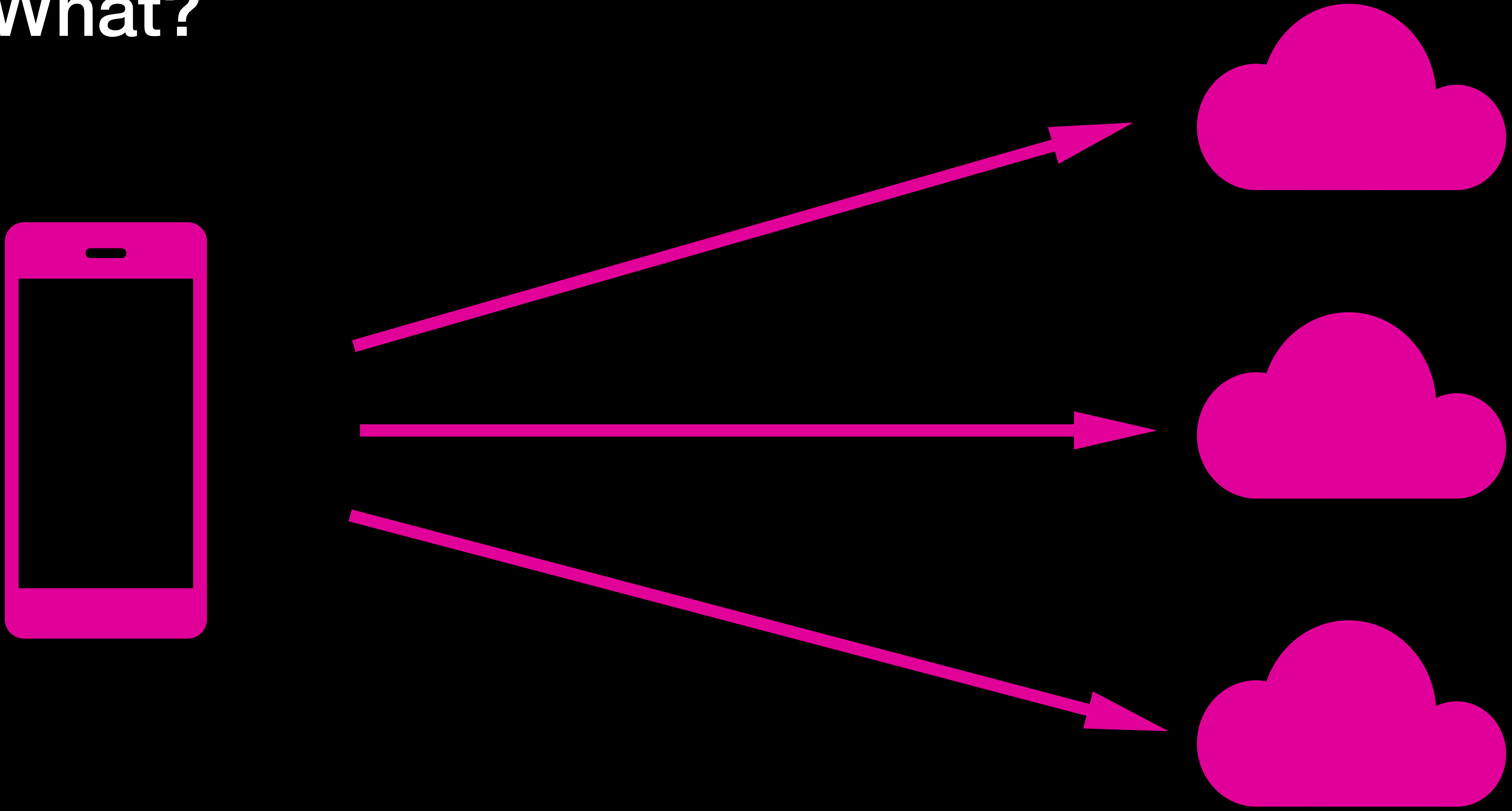
GET /repos/mRs-/Black-Friday-Deals/contributors

[{ "login": "mRs-", "id": 736421, "avatar_url": "https://avatars0.githubusercontent.com/u/7364
"gravatar_id": "", "url": "https://api.github.com/users/mRs-", "html_url": "https://github.com/m
"followers_url": "https://api.github.com/users/mRs-/followers", "following_url": "http

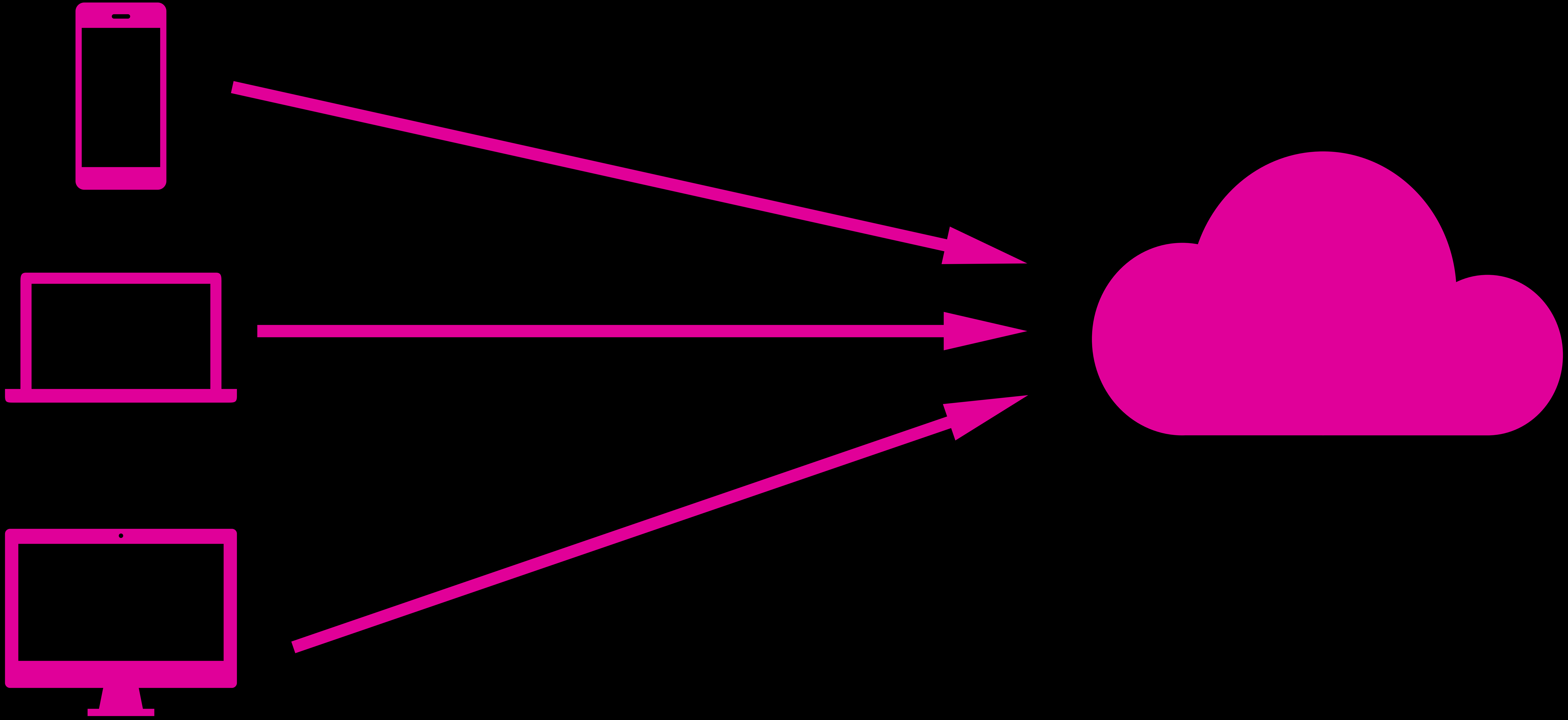
GET /repos/mRs-/Blaupause/contributors

[{ "login": "mRs-", "id": 736421, "avatar_url": "https://avatars0.githubusercontent.com/u/736421?v
"gravatar_id": "", "url": "https://api.github.com/users/mRs-", "html_url": "https://github.com/mRs-
"followers_url": "https://api.github.com/users/mRs-/followers", "following_url": "http

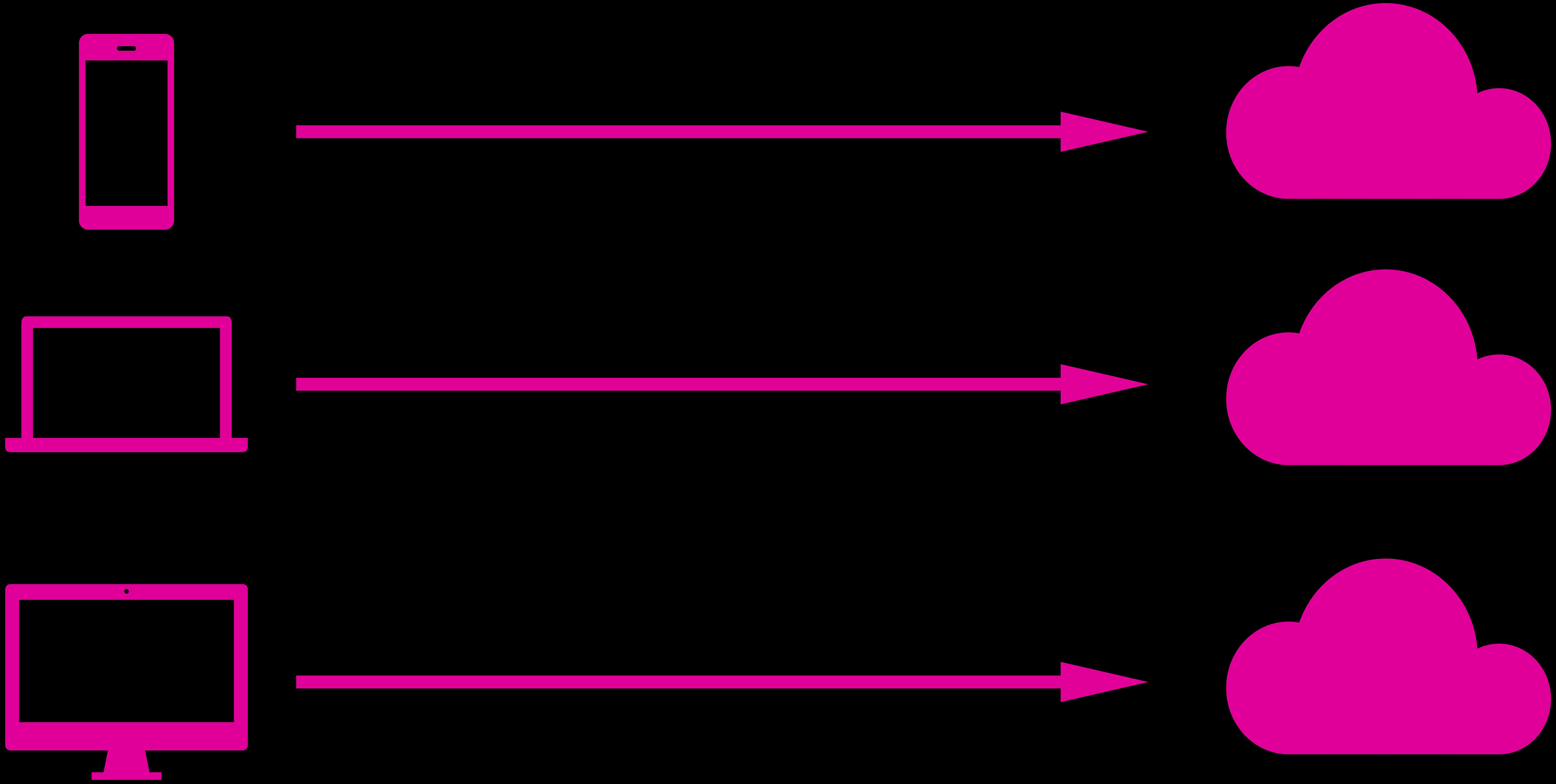
What?



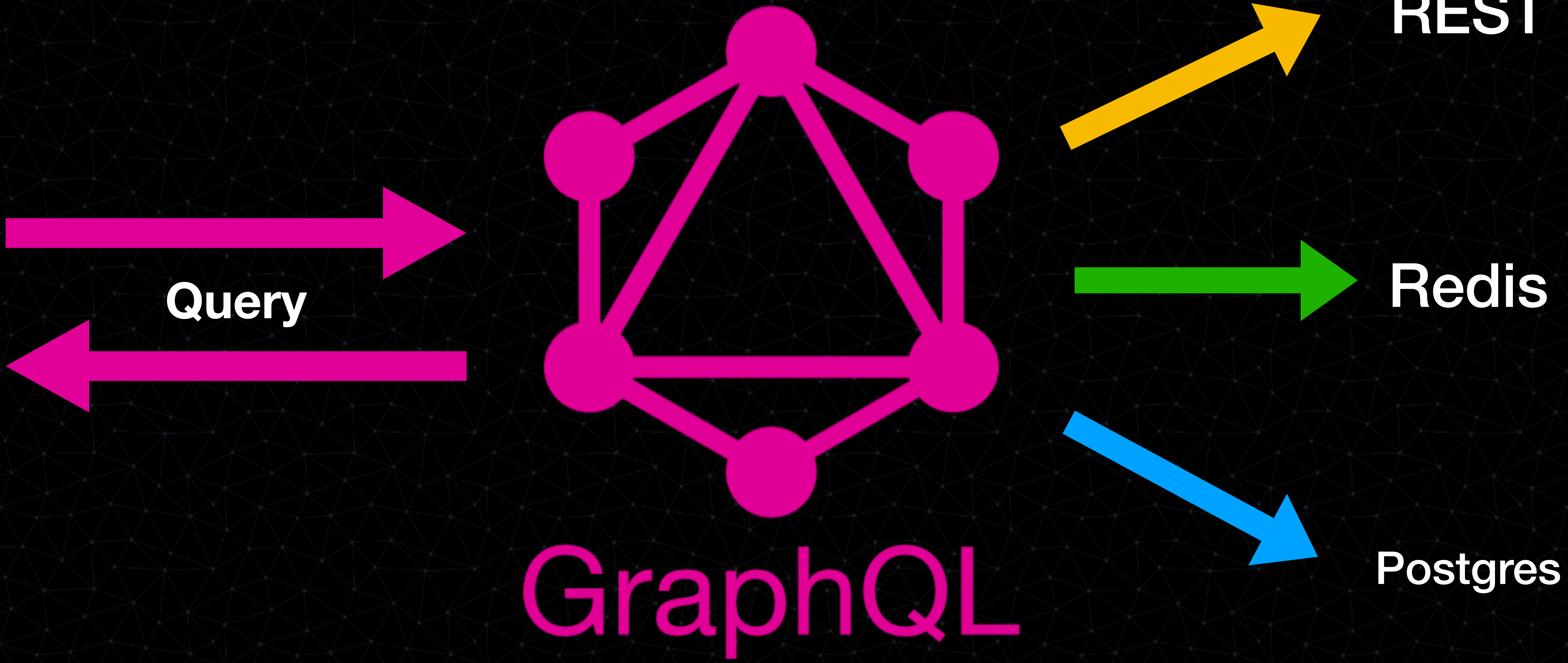
What?



What?



What?



How?

GraphQL

How?

Query

How?

Query

Fields

How?

Query

Arguments

How?

Query

Aliases

How?

Query

Fragments

How?

Query

Variables

How?

Query

Directives

How?

Mutation

Change something on the Servers

Demo!

GraphQL

http://graphql.org/learn/queries/

How?

Schema

Describes your Types, Queries and Mutations

How?

```
schema {
  query: Query
  mutation: Mutation
}
# The query type, represents all of
the entry points into our object graph
type Query {
  hero(episode: Episode): Character
  reviews(episode: Episode!): [Review]
  search(text: String): [SearchResult]
  character(id: ID!): Character
  droid(id: ID!): Droid
  human(id: ID!): Human
  starship(id: ID!): Starship
}
# The mutation type, represents all
updates we can make to our data
type Mutation {
  createReview(episode: Episode,
review: ReviewInput!): Review
}

# The episodes in the Star Wars
trilogy
enum Episode {
  # Star Wars Episode IV: A New Hope,
released in 1977.
  NEWHOPE
  # Star Wars Episode V: The Empire
Strikes Back, released in 1980.
  EMPIRE
  # Star Wars Episode VI: Return of
the Jedi, released in 1983.
  JEDI
}

# A character from the Star Wars
universe
interface Character {
  # The ID of the character
  id: ID!
  # The name of the character
  name: String!
```

```
  # The friends of the character, or
an empty list if they have none
  friends: [Character]
  # The friends of the character
exposed as a connection with edges
  friendsConnection(first: Int, after:
ID): FriendsConnection!
  # The movies this character appears
in
  appearsIn: [Episode]!
}

# Units of height
enum LengthUnit {
  # The standard unit around the world
  METER
  # Primarily used in the United
States
  FOOT
}

# A humanoid creature from the Star
Wars universe
type Human implements Character {
  # The ID of the human
  id: ID!
  # What this human calls themselves
  name: String!
  # Height in the preferred unit,
default is meters
  height(unit: LengthUnit = METER):
Float
  # Mass in kilograms, or null if
unknown
  mass: Float
  # This human's friends, or an empty
list if they have none
  friends: [Character]
  # The friends of the human exposed
as a connection with edges
  friendsConnection(first: Int, after:
ID): FriendsConnection!
  # The movies this human appears in
  appearsIn: [Episode]!
```

```
  # A list of starships this person
has piloted, or an empty list if none
  starships: [Starship]
}
# An autonomous mechanical character
in the Star Wars universe
type Droid implements Character {
  # The ID of the droid
  id: ID!
  # What others call this droid
  name: String!
  # This droid's friends, or an empty
list if they have none
  friends: [Character]
  # The friends of the droid exposed
as a connection with edges
  friendsConnection(first: Int, after:
ID): FriendsConnection!
  # The movies this droid appears in
  appearsIn: [Episode]!
  # This droid's primary function
  primaryFunction: String
}
# A connection object for a
character's friends
type FriendsConnection {
  # The total number of friends
  totalCount: Int
  # The edges for each of the
character's friends.
  edges: [FriendsEdge]
  # A list of the friends, as a
convenience when edges are not needed.
  friends: [Character]
  # Information for paginating this
connection
  pageInfo: PageInfo!
}
# An edge object for a character's
friends
type FriendsEdge {
  # A cursor used for pagination
  cursor: ID!
```

```
  # The character represented by this
friendship edge
  node: Character
}
# Information for paginating this
connection
type PageInfo {
  startCursor: ID
  endCursor: ID
  hasNextPage: Boolean!
}
# Represents a review for a movie
type Review {
  # The number of stars this review
gave, 1-5
  stars: Int!
  # Comment about the movie
  commentary: String
}
# The input object sent when someone
is creating a new review
input ReviewInput {
  # 0-5 stars
  stars: Int!
  # Comment about the movie, optional
  commentary: String
}
type Starship {
  # The ID of the starship
  id: ID!
  # The name of the starship
  name: String!
  # Length of the starship, along the
longest axis
  length(unit: LengthUnit = METER):
Float
}
union SearchResult = Human | Droid |
Starship
```

Demo!

GraphQL

Should?

GraphQL

Questions?

GraphQL

GraphQL

http://graphql.org - General GraphQL Resources

http://apollographql.com - Apollo GraphQL Resources

http://mRs-.github.io - Keynote

https://github.com/mRs-/GraphQL-Example - Example Code