# ALL THINGS API

#APITALKS

## MEET THE SPEAKERS
### & Cloud Elements Intro

**Ramana Lashkar**
Principal Engineer

**Tyler Toth**
Software Engineer

**Paris Roman**
UI/UX Engineer

cloud elements

#APITALKS

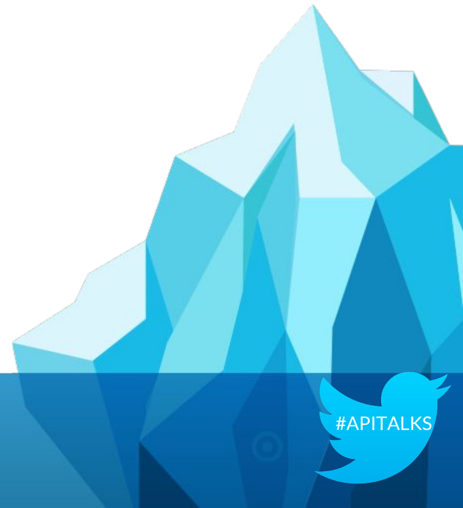
## PURPOSE & AGENDA

### PURPOSE

All Thing API meetup is for API practitioners, implementers and developers to come together to share their experience and learnings on defining, designing, building, launching, managing and consuming APIs based on REST/SOAP/GraphQL/JSON/XML technologies.

### AGENDA

- What is a REST API
- CRUD Operations
- Common Patterns & Best practices for defining the best REST API
- Authentication
- Consuming a REST API
- Live Examples of consuming a REST API from NodeJS and through UI



## SAMPLE PERSONAS

**Dave, The Developer**
Your raw, unedited API is targeted directly at Dave the Developer.  He discovers new APIs through developer communities and builds his own integrations.

**Izzy, IT Manager**
Izzy knows the in's and out's of her company's apps, as each system that is purchased, Izzy works with to keep connected, sharing data seamlessly. Izzy is the technical integrator.

**Pete, Product Manager**
As a product manager, Pete is responsible for connecting business needs with market demands, managing priorities, and meeting with customers.

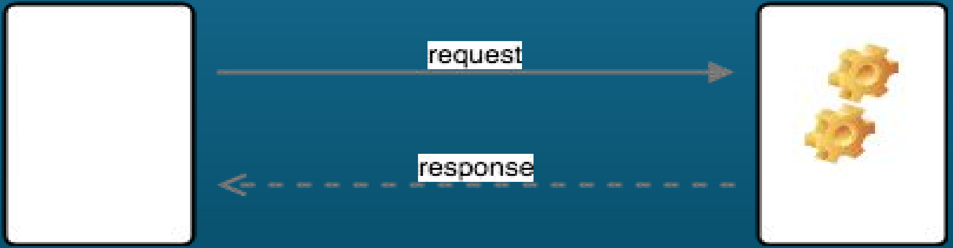
## What is an API ?

An API stands for an Application Programming Interface, which  is a software intermediary that enables two or more applications to communicate with one another. APIs enable organizations to extract and share data in an accessible manner

Provides a predictable way to communicate between systems

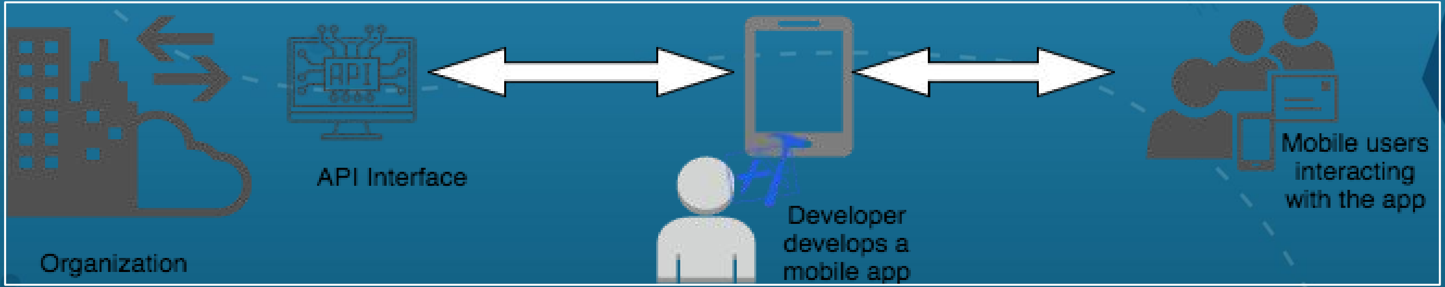
## How does an API work?

An API is a messenger that takes requests and tells the system what you want to do, then returns the response back to you.



#APITALKS

# More Understanding of an API


API
API Interface
Organization
Developer develops a mobile app
Mobile users interacting with the app

#APITALKS

# A simple classifications of APIs

Web service APIs
- SOAP
- RPC (XML, JSON)
- REST

WebSocket APIs

Library-based APIs

Client Based APIs
- Java API
- Android API

OS Functions and routines
- Access to file system
- Access to user interface

Object remoting APIs
- CORBA
- .NET Remoting

Hardware APIs
- drivers
- PCI buses

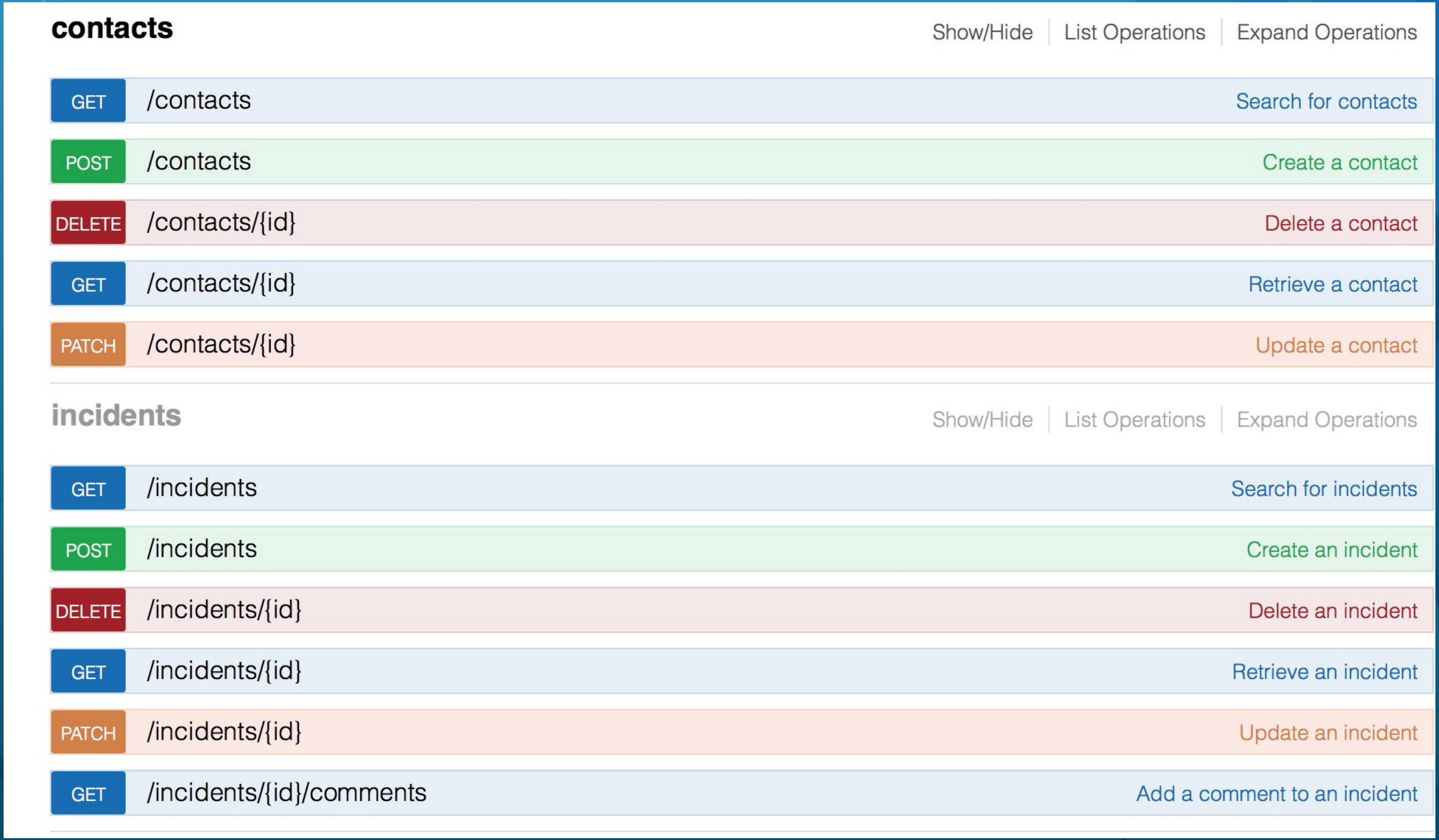
#APITALKS

## Different Types of Web services APIs

- RPC
  - RPCs are a form of interprocess communication (IPC), in that different processes have different address spaces
- SOAP
  - SOAP interfaces are **method-based**. Interface design is set of supported methods and data structures of each method
- REST
  - REST interfaces are resource-based. The most important aspect of the design is the URI structure that allows a consumer to navigate the object graph embodied by the API
- GraphQL
  - GraphQL is an open sourced API specification from Facebook that is touted as the next evolution from RESTful APIs.

#APITALKS

# Representational State Transfer (REST)



**contacts**

Show/Hide | List Operations | Expand Operations

| | | |
|---|---|---|
| GET | /contacts | Search for contacts |
| POST | /contacts | Create a contact |
| DELETE | /contacts/{id} | Delete a contact |
| GET | /contacts/{id} | Retrieve a contact |
| PATCH | /contacts/{id} | Update a contact |

**incidents**

Show/Hide | List Operations | Expand Operations

| | | |
|---|---|---|
| GET | /incidents | Search for incidents |
| POST | /incidents | Create an incident |
| DELETE | /incidents/{id} | Delete an incident |
| GET | /incidents/{id} | Retrieve an incident |
| PATCH | /incidents/{id} | Update an incident |
| GET | /incidents/{id}/comments | Add a comment to an incident |

#APITALKS

# Representational State Transfer (REST)

- REST - REpresentational State Transfer
- REST interfaces are Resource-based.
- Representational
- Six Constraints
  - Uniform Interface
  - Stateless
  - Cacheable
  - Client-Server
  - Layered System
  - Code on Demand (optional)

#APITALKS

## RESOURCE BASED

- Things (resources) VS actions
- Nouns vs Verbs
- Identified by URIs
  - **Example: GET** *https://obscura.zendesk.com/api/v2/tickets*
  - **POST** *https://obscura.zendesk.com/api/v2/tickets*
- Separate from their representation(s)

## Representations

- Resource state transferred between client and server
- Typically JSON or XML
- **Example :**
  - Resource: person (ex: Ramana)
  - Service: contact information (GET)
  - Representation:
    - Name, address, phone
    - JSON or XML or YAML format
    - *{"name" : "Ramana", "address" : "Tollway Plaza","phone": "234-234-2345"}*
    - *<?xml version="1.0" encoding="UTF-8" ?><name>Ramana</name><address>Tollway Plaza</address><phone>234-234-2345</phone>*

#APITALKS

## Uniform Interface

- Interface between clients and servers
- It simplifies and decouples the architecture
- Fundamental to the RESTful design
- For us, this means...
    - HTTP Verbs (GET, POST, PATCH, PUT, DELETE)
    - URIs (resource names)
    - Http Response (status, body)

## Stateless

- Server contains no client state
- Each request contains enough context to process the message
- If any session state, it's held on the client

#APITALKS

## Client-Server

- Always assume a disconnected system between client/server
- Separation of concerns
- Uniform interface is the link between the two

## Cacheable

- Server responses (representations) are cacheable
- Implicitly
- Explicitly (like max age, expires..etc)
- Negotiated

#APITALKS

## Layered System

- Client can’t assume direct connection to the server
- Software or Hardware intermediaries between client and servers
- This improves scalability
- Negotiated

## Code on Demand (Optional)

- Servers can temporarily extend client
- Transfer logic to client
- Example
    - JavaScript, code snippet is sent to the client to execute

#APITALKS

# HTTP VERBS

Most applications you have ever seen on the internet are CRUD (Create, Read, Update, Delete) applications and that the HTTP verbs match to these actions 1 to 1
http://tools.ietf.org/html/rfc2616

POST
    Create

GET
    Read

PUT
    Update or Create

PATCH
    Update / Partially updating a resource
    http://tools.ietf.org/html/rfc5789

DELETE
    As it says, DELETE

#APITALKS

## POST

- POST is for creating a resources
- POST is neither safe nor idempotent. It is therefore recommended for non-idempotent resource requests.
- POST usually should accept a body

Example: *POST /tickets*

```
{
 "ticket": {
   "subject": "My printer is on fire!",
   "comment": {
     "body": "The smoke is very colorful."
   }
 }

}
```

Response

```
{
 "ticket": {
   "id":     35436,
   "subject": "My printer is on fire!",
   ...
 }
}
```

Response Status
Http Status Code: 200 OK or 201 CREATED

#APITALKS

## GET

- GET is for getting the resources
  Example: *GET /tickets/35436*
  Response

```
{
 "ticket": {
   "id":     35436,
   "subject": "My printer is on fire!",
   "status":  "open",
   ...
 },
 "audit": {
   "events": [...],
   ...
 }
}
```

Get tickets *GET /tickets*

```
{
 "tickets": [
  {
    "id":     35436,
    "subject": "Help I need somebody!",
    ...
  },
  {
    "id":     20057623,
    "subject": "Not just anybody!",
    ...
  }
 ]
}
```

Response Status
Http Status Code: 200 OK

#APITALKS

## PUT

- PUT is for creating or replacing a resource
- PUT on an existing resource will replace the resource with the new details
- PUT usually should accepts a body for payload and resource id in the PATH

Example: *PUT /tickets/35436*

```
{
 "ticket": {
   "subject": "My printer is on fire! updated",
   "comment": {
     "body": "The smoke is very colorful...now its all dark"
   }
 }

}
```

Response

```
{
 "ticket": {
   "id":     35436,
   "subject": "My printer is on fire! updated",
   ...
 }
}
```

Response Status
Http Status Code: 200 OK or 201 if  its CREATED

#APITALKS

## PATCH

- PATCH is for partially updating a resource
- PATCH can have only the fields you want to modify and the ones that are not passed in gets unchanged

Example: *PATCH /tickets/35436*

```
{
 "ticket": {
   "subject": "My printer is on fire! Updated for Patch",
 }

}
```

Response

```
{
 "ticket": {
   "id":     35436,
   "subject": "My printer is on fire! Updated for Patch",
   ...
 }
}
```

Response Status
Http Status Code: 200 OK

#APITALKS

## DELETE

- DELETE is for deleting a resource

  Example: *DELETE /tickets/35436*

  Response
  No Content

  Response Status
  Http Status Code: 204 No Content

#APITALKS

# AUTHENTICATION

| | |
|---|---|
| **Basic Auth:**<br>- Widely used protocol for simple username/ password<br>- Provides no confidentiality protection for the transmitted credentials. | **API Key:**<br>- Doesn’t require shared credentials<br>- Requires API to be accessed by a unique key comprised of numbers and letters<br>- Typically goes in the Authorization header in lieu of username and password |
| **SAML**:<br>- XML-based open standard data format<br>- Popular with organizations using single sign-on, corporate applications and in many older legacy applications. | **OAuth & OAuth 2:**<br>- Authorize third-party access to their server resources without sharing their credentials<br>- Credential tokens are long lived, typically a year.<br>- OAuth 2 delegates security to the HTTPS protocol.<br>- OAuth 2 introduced the use of refresh tokens that allow authentications to expire<br>- OAuth 2 sometime requires pre- and post-hooks |

#APITALKS

# Authentication Example

*GET /tickets/35436*

**Request Headers**

Authorization: Basic gErypPlm4dOVgGRvA1ZzMH5MQ3nLo8bo==
Content-Type: “application/json”
Accept: “application/json”

## Response

```
{
 "ticket": {
   "id":     35436,
   "subject": "My printer is on fire!",
   "status":  "open",
   ...
 },
 "audit": {
   "events": [...],
   ...
 }
}
```

## Response Headers

X-Rate-Limit: 700
X-Rate-Limit-Remaining: 699

## Response Status

Http Status Code: 200 OK

#APITALKS

# Errors

## Standard Response Errors

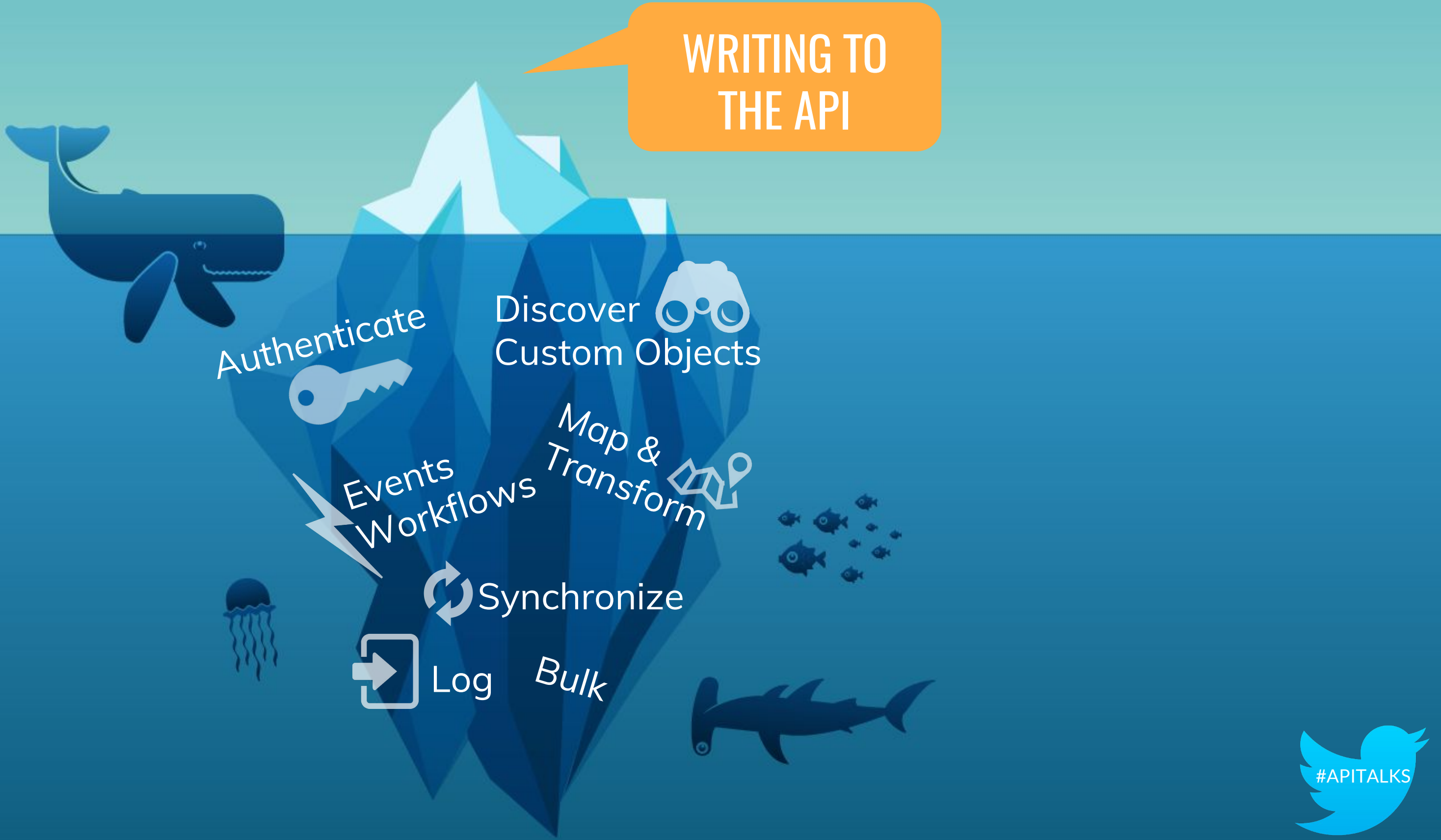
**2xx -** received, understood, and accepted a request.
**3xx -** The caller must take further action in order to complete the request.
**4xx -** An error occurred in handling the request. The most common cause of this error is an invalid parameter.
**5xx -** received and accepted the request, but an error occurred in the service while handling it.

## Http Status Codes

**200 success**
**201 created**
**202 accepted**
**204 no_content**
**302 redirect**
**304 not_modified**

**400 bad_request**
**401 unauthorized**
**403 forbidden**
**404 not_found**
**405 method_not_allowed**
**409 conflict**
**412 precondition_failed**
**429 too_many_requests**
**500 internal_server_error**
**503 unavailable**

#APITALKS


WRITING TO THE API
Authenticate
Discover Custom Objects
Map & Transform
Events Workflows
Synchronize
Log
Bulk
#APITALKS

## Best Practices

**REST URLs**

**POST** /tickets
**GET** /tickets/{id}
**PUT** /tickets/{id}
**PATCH** /tickets/{id}
**DELETE** /tickets/{id}
**GET** /tickets (With search parameters and pagination)

Sub resources
**GET** /tickets/{id}/comments

Sometimes you might end up having an action like below where it can't be represented as resource
POST /files/{id}/copy

**Response**

- Use JSON than XML
- Consistency
- No Wrapper Objects in the response
- Include metadata in headers instead of response data
- Consistent Error Responses

#APITALKS

## Lets see it LIVE

#APITALKS

## CLOSING THOUGHTS

- **APIs MISSION CRITICAL**
  - APIs are business Products
  - API should be part of your end-to end Business design
  - Consistency is the key factor
  - Version your APIs
- **Some of the most common errors for include:**
  - Authentication credentials change due to changes in passwords
  - API calls have changed and therefore failing
  - The service may be down completely or down for scheduled maintenance

- **Standardize error responses**, so that your support team can understand and react to errors more rapidly and efficiently vs. researching each endpoint.

#APITALKS

# Helpful Links

https://blog.cloud-elements.com

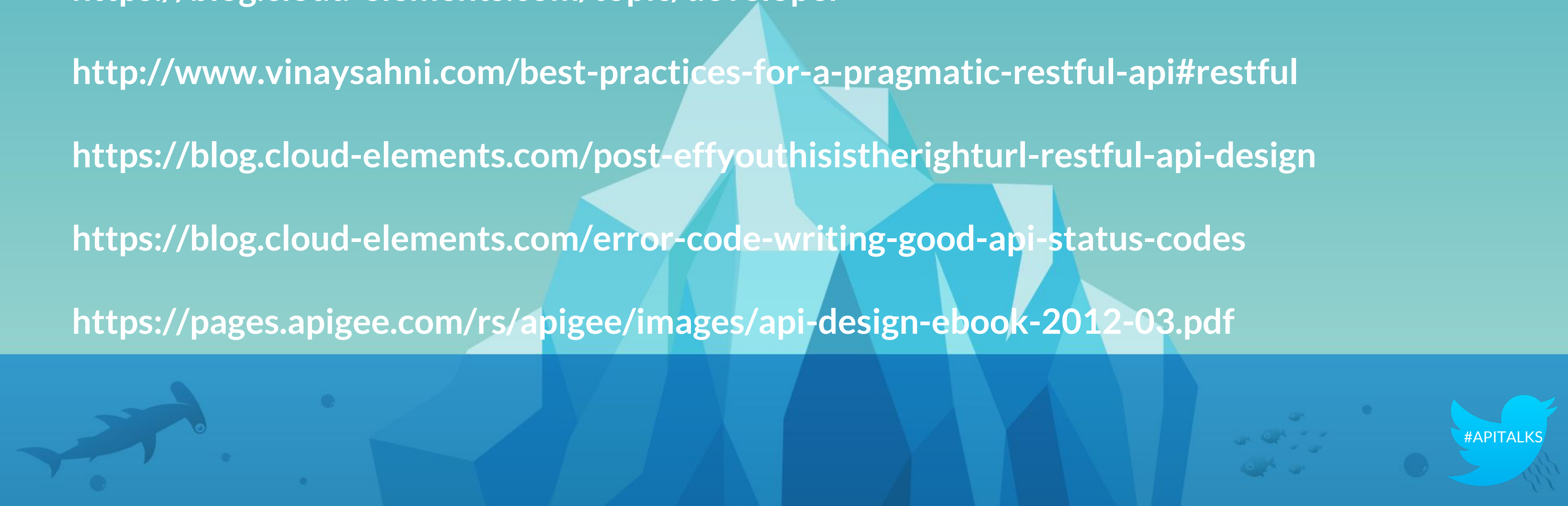
https://blog.cloud-elements.com/topic/developer

http://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api#restful

https://blog.cloud-elements.com/post-effyouthisistherighturl-restful-api-design

https://blog.cloud-elements.com/error-code-writing-good-api-status-codes

https://pages.apigee.com/rs/apigee/images/api-design-ebook-2012-03.pdf



#APITALKS

# Q&A.

#APITALKS