# Introduction to REST APIs

- Today I am going to talk about the term which you people heard a lot as Rest APIs. REST states as Representational State Transfer (REST) and is an architectural style that uses simple HTTP calls for inter-machine communication instead of more complex options (RPC, SOAP).

- REST uses message-based communication and relies on the HTTP standard to describe these messages. Using HTTP protocol means REST is a simple Request/Response mechanism and each request returns a subsequent response. The REST architecture uses 6 – key constraints as guidelines.

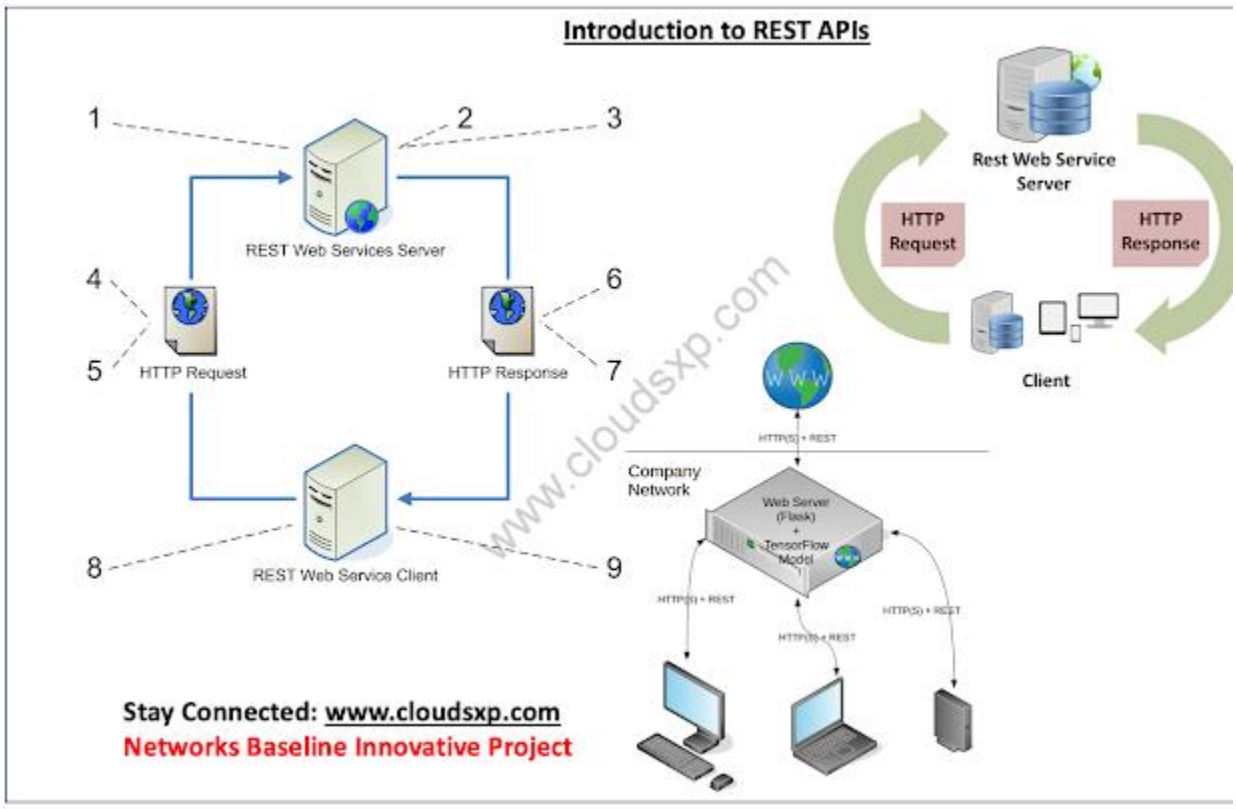


Fig 1.1- REST APIs

**Client-Server**: This model creates a uniform interface which separates clients from Servers.
**Stateless:** protocol where no client context is stored on the server between requests. Each request from any client contains all the information necessary to service the request, and session state is held in the client.
**Cacheable**: Messages are enabled caching within the network path so intermediate servers can cache the data for re-use.
**Layered system**: A client cannot tell whether it is connected directly to the end server. This layered approach improves scalability, load balancing and shared caching.
**Uniform interface**: The uniform interface simplifies and decouples the architecture, which enables each part to evolve independently.
**Code on demand**: Is an optional constraint which allows permits Servers the ability to extend.
Java applets and JavaScript as needed.

REST was built on the principles of HTTP and these Services can return types data such as;

- XML
- JSON - (JavaScript Object Notation)
- HTML
- Plain Text
- Binary/octet (Images, PDF's…)

## What is HATEOAS and why is it important for REST API?

HATEOAS stands for Hypertext as The Engine of Application State. It means that hypertext should be used to find a way through the API. Note the Server responds with a "link" tag and URL needed to complete the specified action in the GET method.

An example:
GET /account/12345 HTTP/1.1

HTTP/1.1 200 OK
<?xml version="1.0"?>
<account>
<account_number>12345</account_number>
<balance currency="usd">-25.00</balance>
<link rel="deposit" href="/account/12345/deposit" /> </account>

## Why is REST popular?

- RESTful Web services are easily leveraged by most tools, including those that are free and inexpensive. RESTful Services are less complex, more efficient (use smaller message format) and provide better performance than SOAP (Simple Operation Access Protocol).

- REST provides a lightweight architecture that promotes scalability and a very loose coupling as it supports billions of users that are unaware of network topology. In short REST is the architecture of the Web as it works today and building Web Applications to use the architecture make a lot of sense.

## Benefits

- Extensibility
- Customizability
- Reusability
- Visibility
- Portability
- Reliability

**Typical REST API calls**:

HTTP is a Request/Response protocol. A client makes a Request to the Server and the Server sends back a Response. The client builds a Request consisting of Headers and Payload and Sends to a URL with an HTTP Method GET, PUT, POST, HEAD, DELETE.

- GET – Retrieves a resource
- POST – Creates a resource
- PUT – Updates a resource
- DELETE – Deletes a resource

The Server builds a Response with Headers & Payload and sends it back to the Client along with a status code validating the "response".