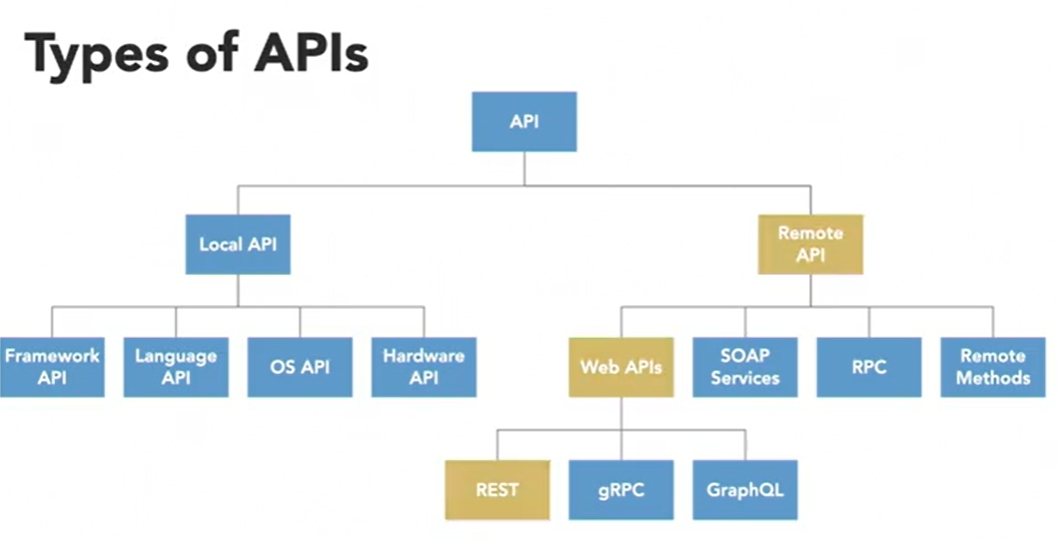
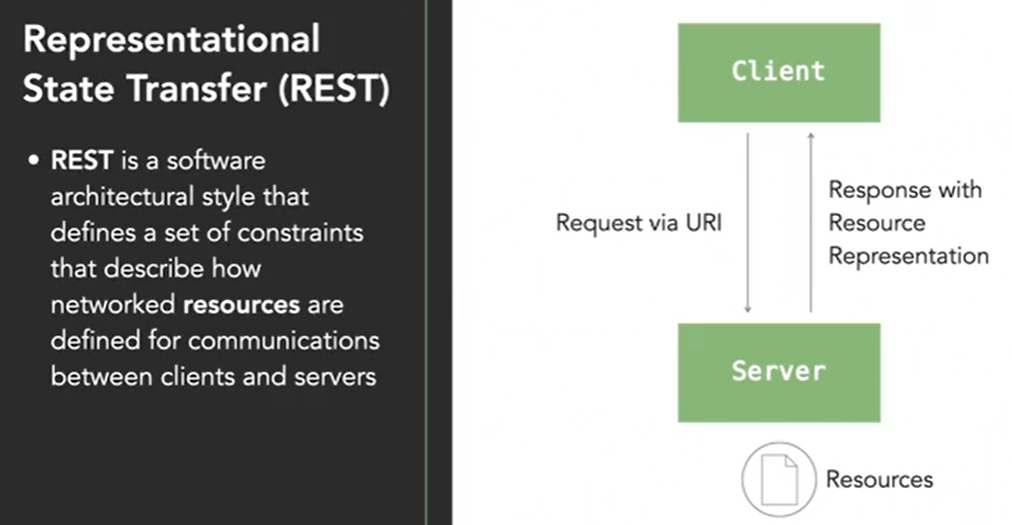
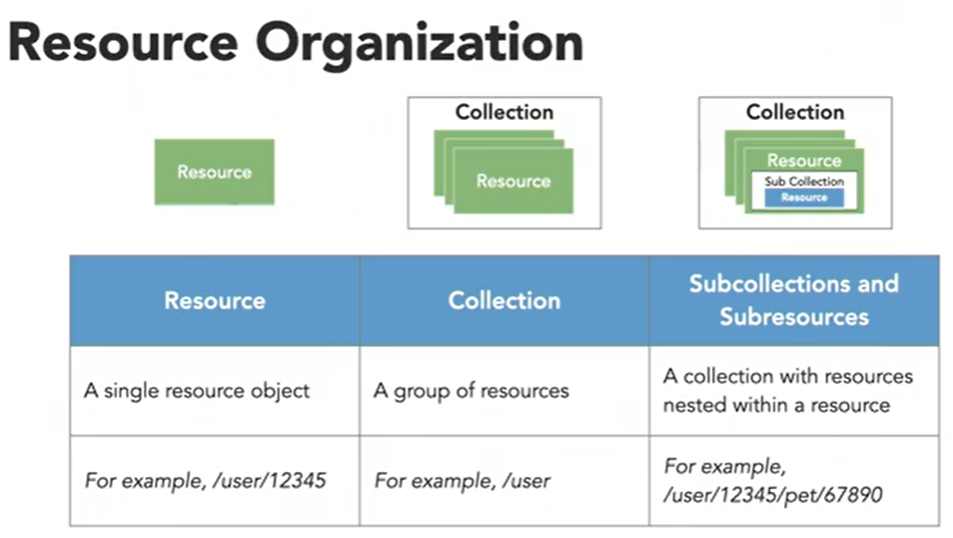


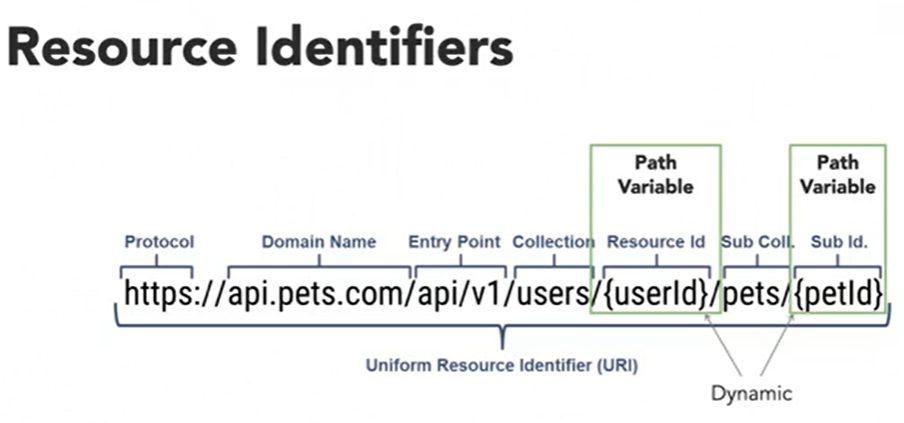
* An API is an interface exposed by a piece of software to allow other software to communicate with it programmatically. The API is typically divided in two parts, the **interface** and the **implementation**. The interface exposes operations that the API can perform. The software that communicates with or calls the API to invoke these operations is known as the **client**.
* There are two major categories for APIs: **local** and **remote**. A local API call takes place on a single computer, while a remote API call allows a client on one computer to execute an operation on another computer.
* **RESTful APIs** are part of a class of web APIs that sit on top of the HTTP protocol.



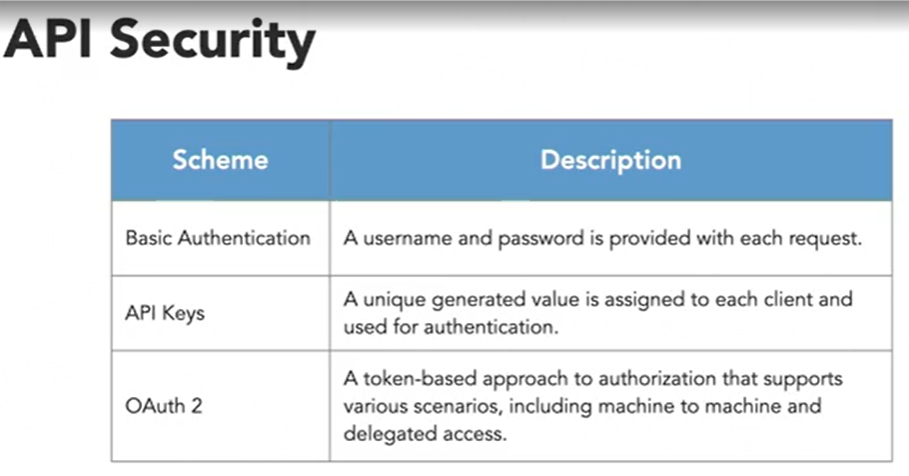
* The whole point of REST is to describe an informal set of rules that influence how resources are exposed over the web and how clients access them.



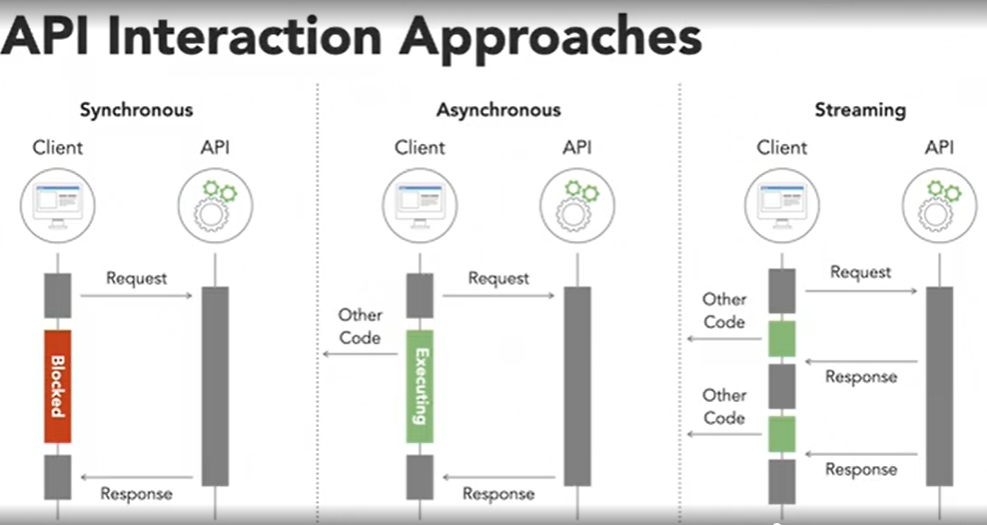




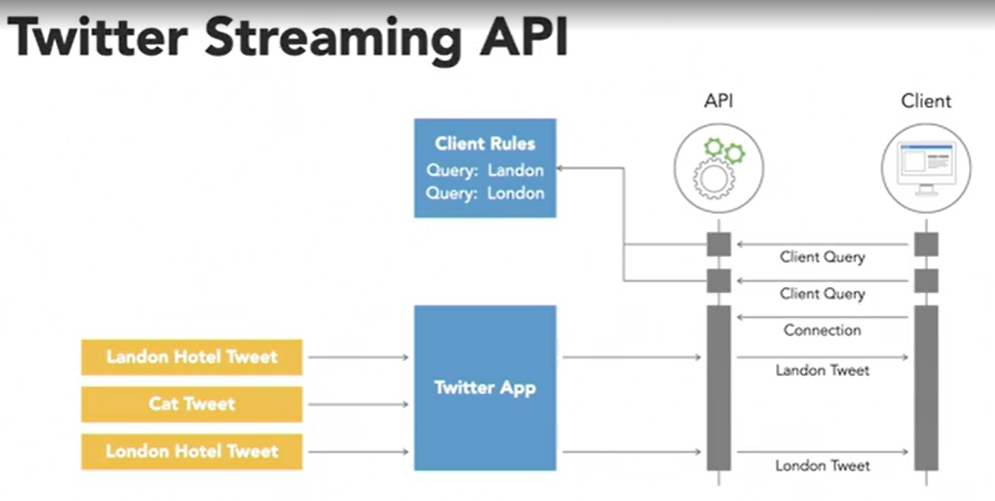
* A **Resource** can be thought of as a variation of some stored entity. The resource depicts that entity’s data elements but it doesn’t need to be exactly the same as the entity. The resource is then interacted with using a representation, which depicts the resource in a specific format like JSON or XML.
* When performing actions on a REST API, like creating or retrieving a resource, clients work with representations instead of directly handling the resource itself.



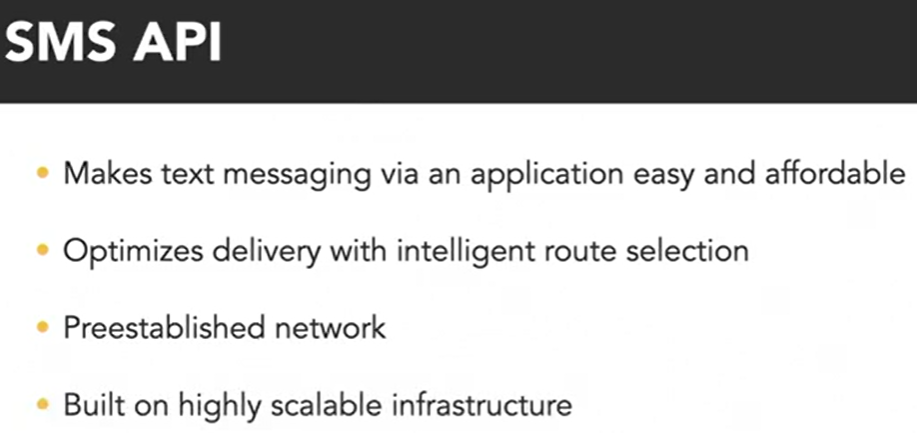
* To make a call using the HTTP Client we need to do four things:
  + **Create the client**
  + **Create the request**
  + **Send the request and receive response**
  + **Work with the response**
* When requests are sent synchronously, the API call blocks the execution of our code. It waits for the response to come back from the API. If we make the calls asynchronously, it will allow our code to continue executing while we wait on the response from the API.
* Add timeouts when calling to APIs.



* Twitter API 2.0 includes a **Filtered Stream Endpoint** that pushes tweets with certain content to a developer’s client application. The interaction starts by the client registering up to 25 filters that specify queries that will be used to match tweets. These filters are established by sending a POST to the Rules Endpoint on the Twitter API. The rules are specified in the JSON request body and can be as simple as a keyword or can use a complex set of operators found in the documentation. After a client registers their filters, they make a connection to the API’s Stream Endpoint. When a user makes a tweet, Twitter determines if the tweet matches the client’s rule. If it does, it’s passed on to the client via their stream. This will continue until the connection between the API and the client is severed.



* Use the *WebClient* class when writing API calls with Spring. *RestTemplate* is in maintenance mode.



* **Feign** builds interfaces that define API calls, instead of building API requests with the Builder. It uses an encoder to determine how to serialize the request body content. It can also use an interceptor to send the credentials required for Basic Authentication. Feign uses annotations and the interface method signature to derive the HTTP request. *@RequestLine* is the annotation used to determine the HTTP method. *@Param* is used to designate values for a template expression.