

REST architecture and Watson APIs

Unit objectives

- Describe the architecture of Representational State Transfer (REST).
- List best practices for using REST in your applications.
- Describe the representation format of data in REST.
- Explain the advantages of the JavaScript Object Notation (JSON) data format.
- List the IBM Watson services on IBM Cloud.
- Provide examples of Watson services REST APIs.

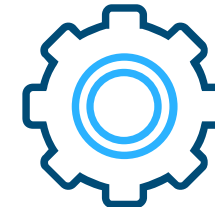
REST concepts, architecture, and characteristics

Topics

- ▶ REST concepts, architecture, and characteristics
 - Applying REST to server-side applications
 - Resources representation and JSON
 - Example: Using REST APIs with IBM Watson

What is REST

- REST is an architecture style for accessing and manipulating resources on the web.
- HTML documents, images, and script files are considered resources.
- REST can retrieve, update, or delete a resource.
- REST uses a Uniform Resource Identifier (URI) to describe the network location of the resource.



REST characteristics

- REST is an architecture style for designing distributed systems. REST is an architecture, not a product.
- REST provides a simple approach for building services for client/server interactions that are based on web resources.
- REST services follow standard protocols, such as HTTP.
- REST services tend to use lightweight data models, such as JSON. XML is also supported.
- REST services are a popular way for applications to interact with server-side applications.



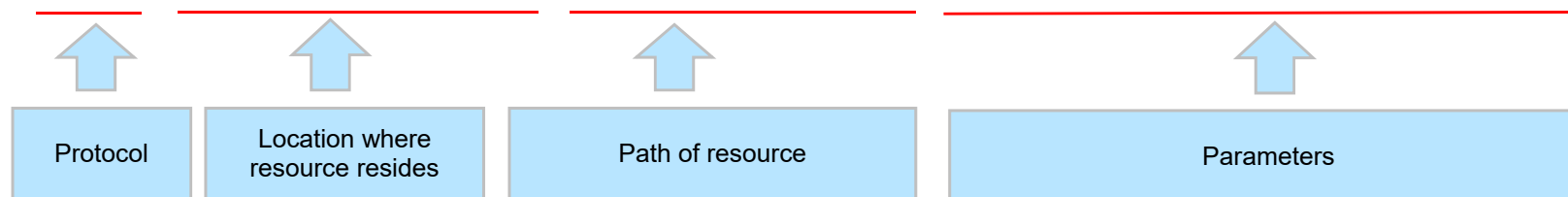
REST architecture constraints

- Uses a client/server model.
 - The client initiates a request by using HTTP. The server processes the request and returns the responses. Separation of concerns is the key principle, for example separating user interface from data storage layer.
- Stateless:
 - Each request from a client must contain all information to understand the request.
- Resource:
 - Any available information is a resource. Each resource must have a unique identifier.
- Uniform interface:
 - REST can retrieve, create, update, or delete a resource by using the following HTTP methods:
 - GET
 - POST
 - PUT
 - DELETE
 - Uses URIs to expose resources:
 - A URI identifies the resource to retrieve or update. The URI describes the network location of the resource and allowed actions.

Resource Identifier and URIs

- In the REST world, URIs are the resources identifiers and used to create, update, retrieve, search for, and delete resources.
- The design of URIs is important because it defines how consistent your model is in accessing resources.
- A URI includes the following parts:
 - Base URL (protocol and the location where the resource is)
 - Context-Root (can be /)
 - Path of resource
 - Operation and parameters (optional)
- URI, URL, URN, and location of resource and parameters example:

<http://example.com/personDetail?firstName=Ahmed&age=28>



URI: <http://example.com/personDetail>

URL: <http://example.com/>

URN: urn:example:persondetail:ahmed:28

Best practices to model URIs in REST

- URI should be simple, intuitive, easy to read, and consistent.
- Avoid having your API start the root domain. Consider using a URI like <http://example.com/api> instead of <http://api.example.com>.
- Resource endpoints should be a plural and not a singular name. For example: <http://example.com/api/products>
- Access a specific instance of a resource to use as an identifier in HTTP GET call. For example: <http://example.com/api/products/prod123>
- Do not use privileged user information as unencrypted parameters. For example: <http://example.com/api/user?ssn=039456337-87>
- To make the REST service more portable and consumable, allow multiple results representations such as JSON and XML. For example:
 - <http://example.com/api/products?format=json>
 - <http://example.com/api/products?format=pdf>
- HTTP POST should be used to create a resource
- Searches should not return a data overload. If required, always schedule a paging mechanism. For example:
<http://example.com/api/products?pageSize=10&pageNo=4>

Applying REST to server-side applications

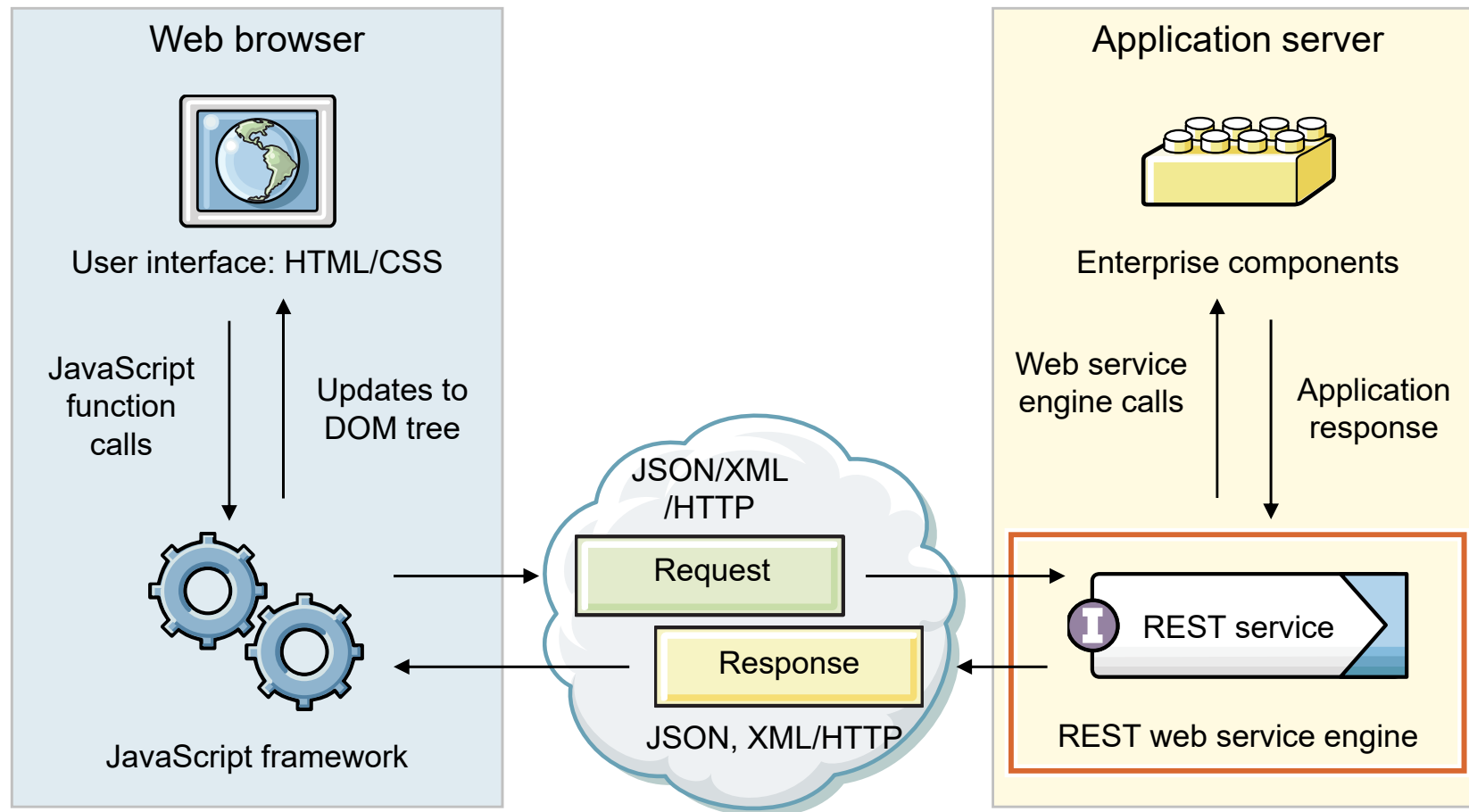
Topics

- REST concepts, architecture, and characteristics
- ▶ Applying REST to server-side applications
- Resources representation and JSON
- Example: Using REST APIs with IBM Watson

Applying REST to server-side applications

- In a more general sense, web resources represent a source of information:
 - HTML documents define the structure of a web page.
 - CSS documents define the presentation of a web page.
 - Image files provide a visual representation of information.
- With REST services, you make your server applications available as a web resource:
 - A REST service is an entry point to an application on the server.
 - HTTP method verbs are used to call a REST service.
 - A URI specifies the REST service to call. The URI describes the network location of the server application resource.

Example: Application Model Architecture for REST services



What is a RESTful web service

A *RESTful web service* or *REST service* is a web service that follows the principles of REST.

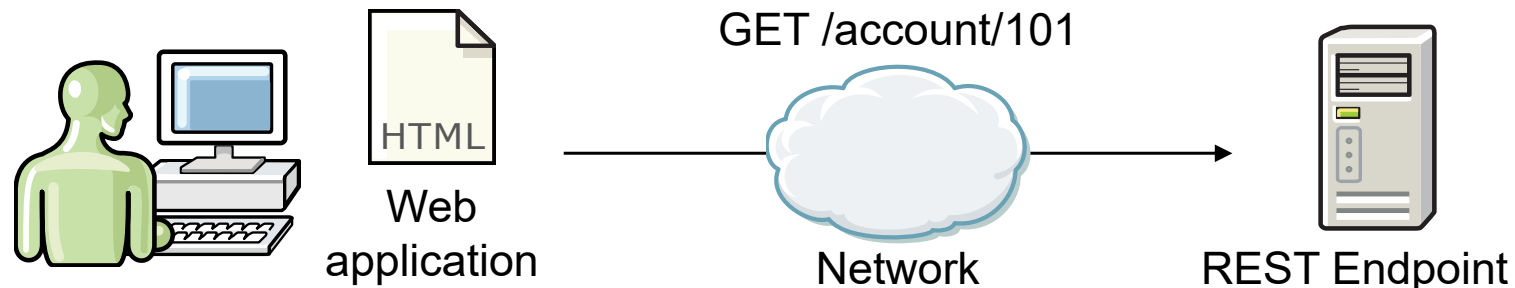
A web server hosts web resources, which are applications and sources of information. For example, the IBM stock service contains information about the current stock price:

- *Identifiers* uniquely reference web resources. The resource path `/stock/IBM/` represents the IBM stock resource on the server.
- The client uses HTTP methods as a uniform interface to interact with resources. To retrieve the current IBM stock price, send a `GET` operation to `/stock/IBM`.

Best practices for RESTful web services

- The implementation of a REST web service follows some basic design principles:
 - Use HTTP methods explicitly:
 - To retrieve a resource, use `GET`.
 - To create a resource on the server, use `POST`.
 - To change the state of a resource or to update it, use `PUT`.
 - To remove or delete a resource, use `DELETE`.
 - Be stateless.
 - Expose directory structure-like URIs.
 - Expose a resource in some data format, such as XML, JSON, or both.

Example: Sending an HTTP GET request to a REST service



```
GET /account/101 HTTP/1.1
Host: example.com
Accept: application/json,
        application/xml
```

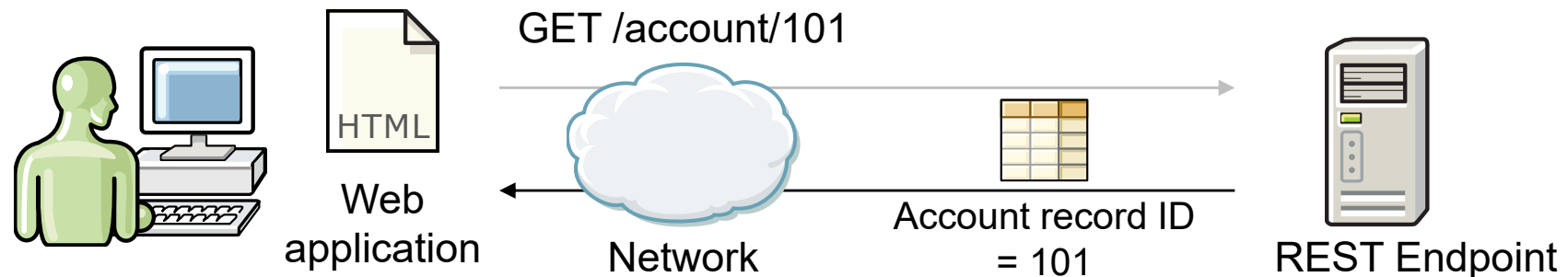
1

To retrieve a web resource from the server, the client web application sends an HTTP GET request.

2

The name of the server resource is `/account/101`. This resource path represents an account record with an ID value of 101.

Example: Receiving an HTTP GET response from a REST service



```
HTTP/1.1 200 OK
Content-Length: 81
Content-Type: application/json
Accept: application/json
Date: Wed, 19 Jun 2013 13:19:23 GMT
{
  "name": "John",
  "accountId": "101",
  "type": "savings",
  "balance": 50.00
}
```

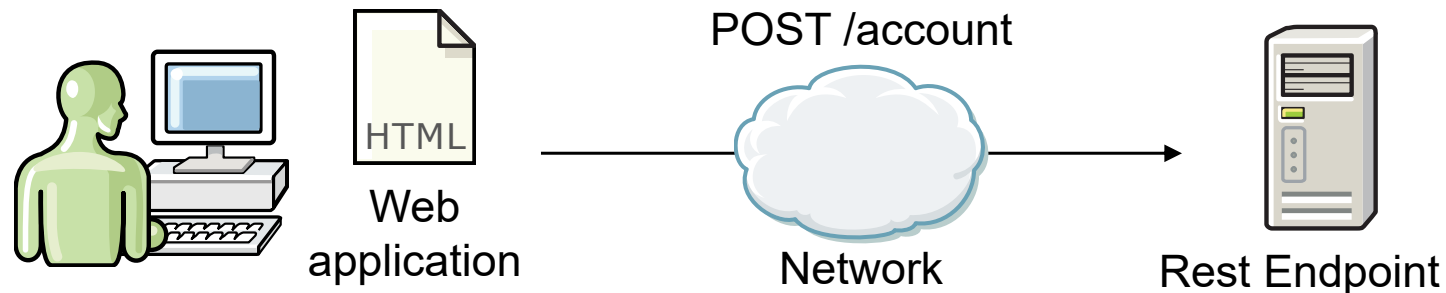
3

The HTTP status code of 200 indicates that the REST service operation completed successfully.

4

The body of the HTTP response message contains information about the account in a JSON data format.

Example: Sending an HTTP POST request to a REST service



```
POST /account/ HTTP/1.1
Host: example.com
Content-Type: application/json
Accept: application/json,
        application/xml

{
  "name": "John",
  "type": "savings",
  "ssn": "12345678"
}
```

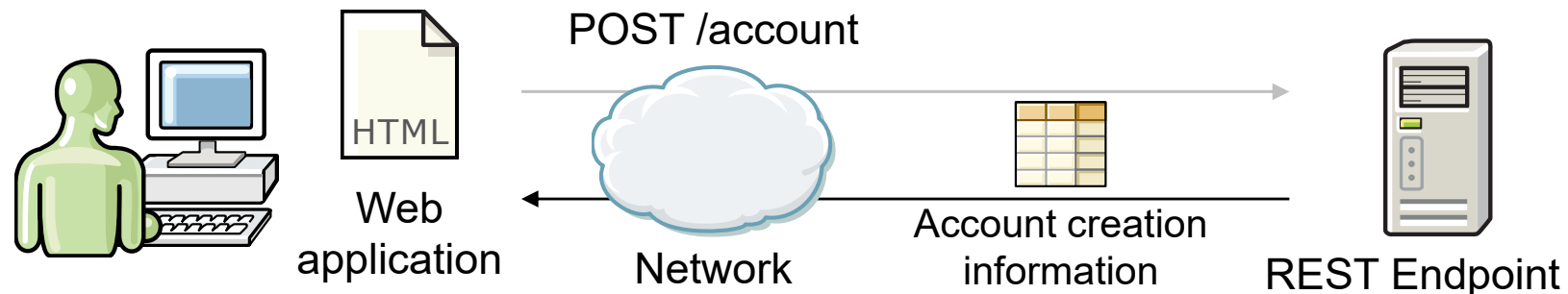
1

To create a web resource from the server, the client web application sends an HTTP POST request.

2

The name of the server resource is /account. The user completes the HTTP form body with required values to create an account.

Example: Receiving an HTTP POST response from a REST service



```
HTTP/1.1 200 OK
Content-Length: 81
Content-Type: application/json
Date: Wed, 19 Jun 2019 13:19:23 GMT
{
  "name": "John",
  "accountId": "101",
  "type": "savings",
  "balance": 0.00,
  "activeDate" : "2019-04-19"
}
```

3 The HTTP status code of 200 indicates that the REST service operation completed successfully.

4 The body of the HTTP response message can contain some information about the create account process in a JSON data format.

Resources representation and JSON

Topics

- REST concepts, architecture, and characteristics
- Applying REST to server-side applications
- ▶ Resources representation and JSON
- Example: Using REST APIs with IBM Watson

Resource representation in REST

- A resource representation typically reflects the state of a resource and its attributes when a client application requests it. Resource representations in this sense are mere snapshots in time.
- The resource representation might be as simple as a representation of a record in a database that consists of a mapping between column names and a format.
- If the system has a data model, then according to this definition a resource representation is a snapshot of the attributes of one of the things in your system's data model.
- The more common data representations in REST are:
 - JSON
 - XML
 - HTML
 - Text/Plan
- JSON is one of most popular formats to represent resource data.

Introduction to JSON

- JSON is a text format for structured data:
 - Its syntax is derived from the object literals of JavaScript, according to *ECMA-262 ECMAScript language Standard, Third Edition*:
<http://www.ecma-international.org/publications/standards/Ecma-262.htm>
 - JSON is a platform-neutral and language-neutral data format
- The main design goal of JSON is to provide a minimal and portable textual data interchange format.
- JSON is not a markup language. Unlike XML, it does not use descriptive tags to encapsulate the data <title></title>.
- JSON is built on two structures:
 - Name-value pairs
 - List of values

JSON data types

`"Hello world!\n"`

A **string** is a sequence of zero or more Unicode characters.

`-1.4719e7`

A **number** includes an integer part that can be prefixed with a sign and followed by a fraction or an exponent.

`{"name": "John"}`

An **object** is an unordered collection of zero or more name-value pairs.

`["a", "b", "c"]`

An **array** is an ordered sequence of zero or more values.

`true`

A **Boolean** is a literal value of either **true** or **false**.

`null`

The keyword **null** represents a null value.

JSON data type: Objects

An unordered collection of key/value pairs:

- Curly brackets ({ }) hold object declarations.
- Colons separate object keys and values.
- Commas separate each key-value pair.
- Keys are strings.
- Values can be any JSON data type.
- Objects can be nested.

```
{
  "name": {
    "first": "John",
    "last": "Smith"
  },
  "id": 101,
  "email": "john.smith@example.com"
}
```

JSON data type: Arrays

An ordered sequence of values with these characteristics:

- Arrays must begin and end with square brackets ([]).
- Commas separate array values.
- Arrays can be nested.

```
[ "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",  
  "Friday", "Saturday"]  
  
[  
  [0, -1, 0],  
  [{"one":1}, 0, "hello"],  
  [0, , 1]  
]
```

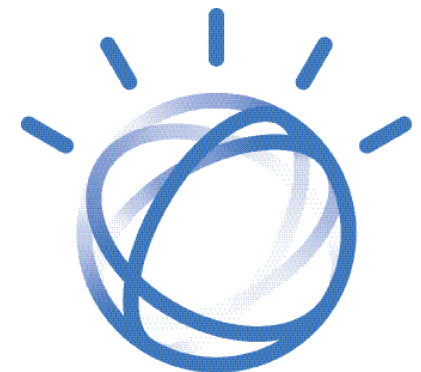
Example: Using REST APIs with IBM Watson

Topics

- REST concepts, architecture, and characteristics
- Applying REST to server-side applications
- Resources representation and JSON
- ▶ Example: Using REST APIs with IBM Watson

IBM Watson

- IBM Watson is the artificial intelligence (AI) offering from IBM.
- Watson uses natural language processing (NLP), computer vision, and machine learning technologies to reveal insights from large amounts of unstructured data.
- You do not need to know the details of every associated AI subfield.
- You must have a high-level understanding of each subfield.
- You must know how to apply the correct AI technology to the problem by using AI application programming interfaces (APIs) or a ready-to-use AI framework.



How Watson works

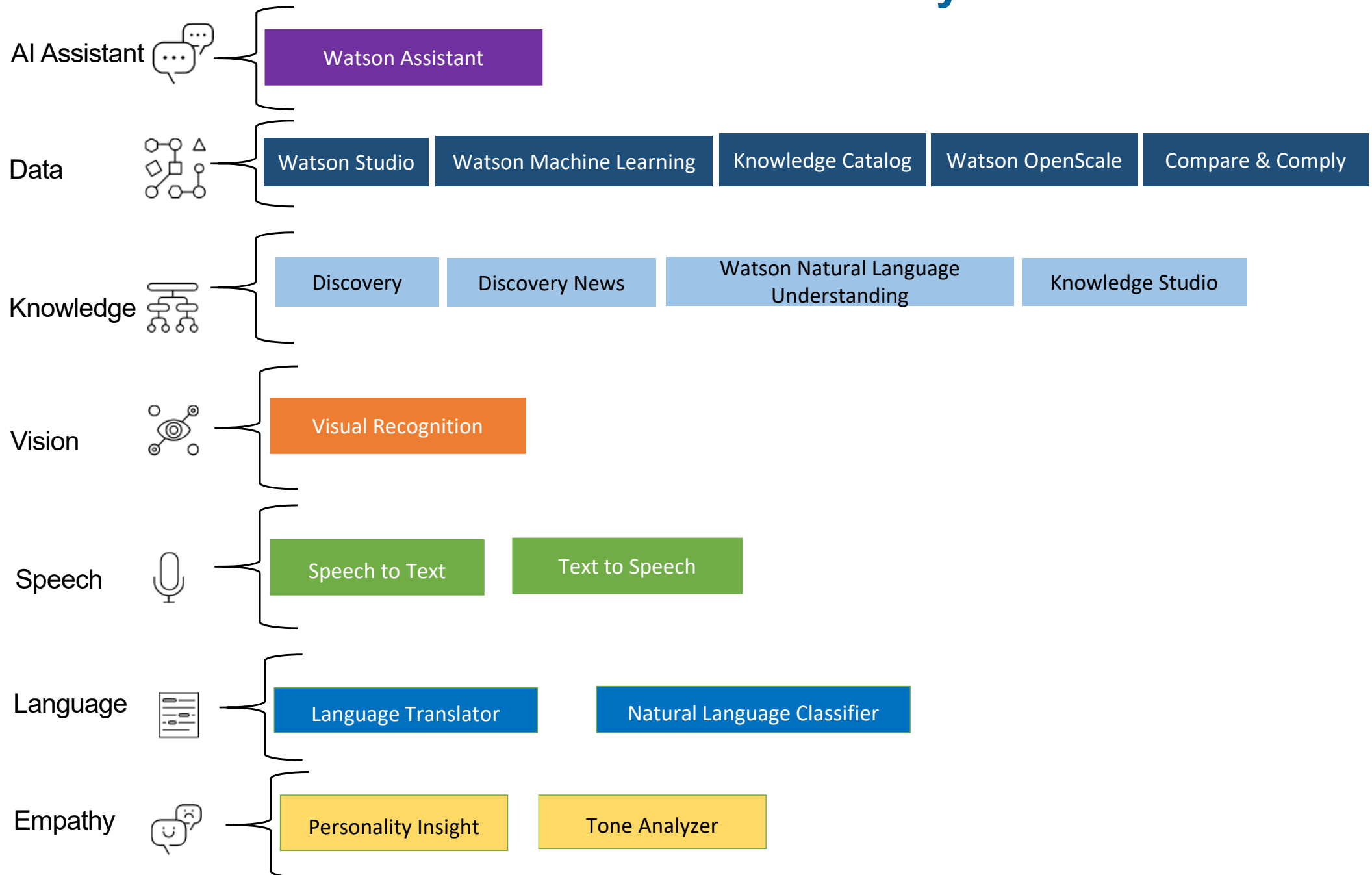
What is Watson, and how is it helping businesses across the globe to build a smarter future?

Watch the following video to learn about Watson technology or discover the different parts that make up Watson:

<https://www.youtube.com/watch?v=r7E1TJ1HtM0>



Watson services on IBM Cloud - Summary



AI services on IBM Cloud catalog

Catalog

Q Search the catalog...

Filter


All Categories

Compute
Containers
Networking
Storage
AI >
Analytics
Databases
Developer Tools
Integration
Internet of Things
Security and Identity
Starter Kits
Web and Mobile
Web and Application


AI

 **Watson Assistant**
Lite • IBM • IAM-enabled


Watson Assistant lets you build conversational interfaces into any application, device, or channel.

 **Compare and Comply**
Lite • IBM • IAM-enabled


Process governing documents to convert, identify, classify, and compare important elements

 **Discovery**
Lite • IBM • IAM-enabled


Add a cognitive search and content analytics engine to applications.

 **Knowledge Catalog**
Lite • IBM • IAM-enabled


Discover, catalog, and securely share enterprise data.

 **Knowledge Studio**
Lite • IBM • IAM-enabled


Teach Watson the language of your domain.

 **Language Translator**
Lite • IBM • IAM-enabled


Translate text, documents, and websites from one language to another. Create industry or region-specific translations via

 **Machine Learning**
Lite • IBM • IAM-enabled


IBM Watson Machine Learning - make smarter decisions, solve tough problems, and improve user outcomes.

 **Natural Language Classifier**
IBM • IAM-enabled


Natural Language Classifier uses advanced natural language processing and machine learning techniques to create custom

 **Natural Language Understanding**
Lite • IBM • IAM-enabled


Analyze text to extract meta-data from content such as concepts, entities, emotion, relations, sentiment and more.

 **Personality Insights**
Lite • IBM • IAM-enabled


The Watson Personality Insights derives insights from transactional and social media data to identify psychological traits.

 **Speech to Text**
Lite • IBM • IAM-enabled


Low-latency, streaming transcription

 **Text to Speech**
Lite • IBM • IAM-enabled


Synthesizes natural-sounding speech from text.

 **Tone Analyzer**
Lite • IBM • IAM-enabled


Tone Analyzer uses linguistic analysis to detect three types of tones from communications: emotion, social, and

 **Visual Recognition**
Lite • IBM • IAM-enabled


Find meaning in visual content! Analyze images for scenes, objects, faces, and other content. Choose a default model off the

 **Watson OpenScale**
Lite • IBM • IAM-enabled

IBM Watson OpenScale is an enterprise-grade environment for AI infused applications that provides enterprises with

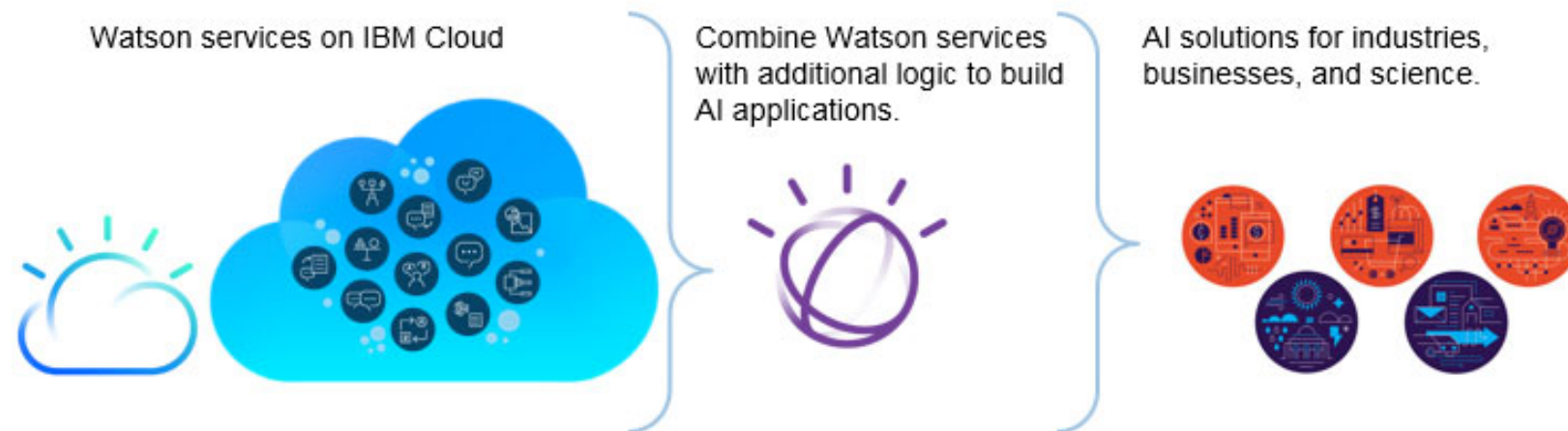
 **Watson Studio**
Lite • IBM • IAM-enabled

Embed AI and machine learning into your business. Create custom models using your own data.

 **PowerAI**
Third Party • IAM-enabled

The accelerated deep learning platform for enterprise. Built on the IBM PowerAI platform, powered by Nimbix.

Building AI solutions with IBM Watson services on IBM Cloud



Getting started with Watson services on IBM Cloud

You must complete these steps to use the Watson services on IBM Cloud:

1. Set up your IBM Cloud account.
2. Find and select a Watson service in the IBM Cloud catalog.
3. Create the service instance.
4. Get the service credentials to authenticate to your service from your application.
5. Start coding your application and calling the Watson APIs to infuse AI capabilities into your app.

Options to call Watson APIs

- There are basically two options to call APIs:
 - Using any REST API client like cURL or Postman.
By using `curl` or the Postman plug-in in your browser, you can run commands to Watson services endpoints. For example, using cURL:

```
curl -X POST \ -H "Content-Type: application/json" \ -u  
"apikey:{apikey}" \ -d @parameters.json \  
"{url}/v1/analyze?version=2018-11-16"
```
 - Using SDKs
You can use the native language wrappers that call Watson Services. There are several options, such as Node.js, Python, Java, Go, dotnet, Swift, and Unit. For more information, see:
<https://github.com/watson-developer-cloud>
- Data input and output on Watson API calls
 - APIs support JSON for non-binary data input and output, and might also support CSV or XML. Formats default to JSON when not specified.

Watson service example: Watson Natural Language Understanding



- Analyze semantic features of text input, including the following items:
Categories, concepts, emotions, entities, keywords, metadata, relations, semantic roles, and sentiment.
- Categorize content.
- Develop custom annotation models to identify domain-specific entities and relations in unstructured text by using Knowledge Studio.
- Example applications: Categorize news articles and blog posts and sort them based on general concepts, keywords, and entities.

Watson Natural Language Understanding demonstration: Input text

<https://natural-language-understanding-demo.ng.bluemix.net/>

Natural Language Understanding

Natural Language Understanding is a collection of APIs that offer text analysis through natural language processing. This set of APIs can analyze text to help you understand its concepts, entities, keywords, sentiment, and more. Additionally, you can create a custom model for some APIs to get specific results that are tailored to your domain. This system is for demonstration purposes only and is not intended to process Personal Data. No Personal Data is to be entered into this system as it may not have the necessary controls in place to meet the requirements of the General Data Protection Regulation (EU) 2016/679.

 [Get Started](#)

[API Reference](#)

[Documentation](#)

[Fork on GitHub](#)

[Start for free in IBM Cloud](#)

Examine a news article or other content

Text

URL

In the rugged Colorado Desert of California, there lies buried a treasure ship sailed there hundreds of years ago by either Viking or Spanish explorers. Some say this is legend; others insist it is fact. A few have even claimed to have seen the ship, its wooden remains poking through the sand like the skeleton of a prehistoric beast. Among those who say they've come close to the ship is small-town librarian Myrtle Botts. In 1933, she was hiking with her husband in the Anza-Borrego Desert, not far from the border with Mexico. It was early March, so the desert would have been in bloom, its washed-out yellows and grays beaten back by the riotous invasion of wildflowers. Those wildflowers were what brought the Bottses to the desert, and they ended up near a tiny settlement called Agua Caliente. Surrounding place names reflected the strangeness and severity of the land: Moonlight Canyon, Hellhole Canyon, Indian Gorge. To enter the desert is to succumb

Watson Natural Language Understanding demonstration: Analyze results

For results unique to your business needs consider building a [custom model](#).

* This system is for demonstration purposes only and is not intended to process Personal Data. No Personal Data is to be entered into this system as it may not have the necessary controls in place to meet the requirements of the General Data Protection Regulation (EU) 2016/679.

Analyze

Sentiment

Emotion

Keywords

Entities

Categories

Concept

Syntax

Semantic Roles

Review the overall [sentiment](#) and targeted sentiment of the content.

[JSON](#) ^

```
{
  "sentiment": {
    "document": {
      "score": -0.544138,
      "label": "negative"
    }
  }
}
```

Overall Sentiment

Negative  -0.54

Targeted Sentiment



Additional resources for developers

For more information, see the following resources:

- Explore the complete list of Watson APIs: [Watson Products and Services](#)
- Get started on IBM Cloud: [IBM Cloud essentials](#)
- Access developer's resources:
 - [Build with Watson](#)
 - [Documentation and API Reference](#)
 - [Watson SDKs](#)
 - [Building Cognitive Applications with IBM Watson Services](#)
 - [AI articles and tutorials](#)
 - [Watson webinars](#)
 - [Building with Watson: Application Starter Kits for developers](#)
 - [Watson Starter Kits](#)

Documentation and other information sources

- REST APIs:
 - *RESTful web services* - The basics at IBM Developer:
<https://www.ibm.com/developerworks/library/ws-restful/>
 - *ECMA-262 ECMAScript Language Standard, Third Edition*:
<http://www.ecma-international.org/publications/standards/Ecma-262.htm>
- IBM CEO Ginni Rometty describes a new era in technology and business:
<https://www.youtube.com/watch?v=bMLYKhiZCVI>

Unit summary

- Describe the architecture of Representational State Transfer (REST).
- List best practices for using REST in your applications.
- Describe the representation format of data in REST.
- Explain the advantages of the JavaScript Object Notation (JSON) data format.
- List the IBM Watson services on IBM Cloud.
- Provide examples of Watson services REST APIs.