Courser apigee:

https://www.coursera.org/learn/api-development-apigee-gcp

security:

<https://www.coursera.org/learn/api-security-apigee-gcp>

node js mongo db:

<https://www.coursera.org/lecture/server-side-nodejs/rest-api-with-express-mongodb-and-mongoose-g8Dbb>

specialization:

https://www.coursera.org/specializations/apigee-api-gcp

[**https://www.youtube.com/watch?v=OhPFgqHz68o**](https://www.youtube.com/watch?v=OhPFgqHz68o) **(mean appli)**

[**https://www.youtube.com/watch?v=a-dn6k6Frq4**](https://www.youtube.com/watch?v=a-dn6k6Frq4) **(mean js generator)**

**Preparation of Apigee certification:**

[**https://apigee.com/api-management/#/certification?tab=study-guide**](https://apigee.com/api-management/#/certification?tab=study-guide)

[**https://community.apigee.com/articles/56382/apigee-api-platform-learning-guide.html**](https://community.apigee.com/articles/56382/apigee-api-platform-learning-guide.html) **(all topics)**

[**https://community.apigee.com/questions/11216/i-am-planning-to-take-apigee-certification-can-som.html**](https://community.apigee.com/questions/11216/i-am-planning-to-take-apigee-certification-can-som.html) **(more topics)**

Grammarly

Vendor@royalcyber.com  
Cyber2013

udemy

id vendor@royalcyber.com

pass Cyber@2017

[**https://developer.ibm.com/apiconnect/**](https://developer.ibm.com/apiconnect/) **customers**

Hi , we are IBM partner, Royal Cyber Offers API Connect services and resources. Is your org using IBM APIC? I would be happy to take you through our services. Let’s chat

**Data modeling** is a process used to define and analyze **data** requirements needed to support the business processes within the scope of corresponding information systems in organizations.

API BaaS models the data for your apps as application-specific collections of data entities managed within an organization. The following is an overview of the component hierarchy that makes up the data model, and is intended to help you understand how data is stored, managed and accessed

Understanding APIs and API proxies

<https://docs.apigee.com/api-platform/fundamentals/understanding-apis-and-api-proxies>

Api design:

<https://docs.apigee.com/api-platform/fundamentals/best-practices-api-proxy-design-and-development>

request response:

https://docs.apigee.com/api-platform/fundamentals/understanding-handling-request-response-data

<https://docs.apigee.com/api-platform/fundamentals/introduction-flow-variables>

scoping access control:

<https://www.google.com.pk/search?rlz=1C1CHBF_enPK791PK793&ei=PXcOXICdJteF1fAPjZam4AY&q=Scoping+access+controls+apigee&oq=Scoping+access+controls+apigee&gs_l=psy-ab.3..33i21.3693.5289..5492...0.0..0.257.1341.2-6......0....1..gws-wiz.......0i71j33i22i29i30.aTlrqRKYhh4>

https://community.apigee.com/questions/36860/question-limiting-developer-app-access-using-scope.html

<https://www.youtube.com/watch?v=WLtHTaU85ZU>

<https://www.youtube.com/watch?v=wlCFA8maquk>

## **What is an API?**

An API is an interface that makes it easy for one application to 'consume' capabilities or data from another application. By defining stable, simplified entry points to application logic and data, APIs enable developers to easily access and reuse application logic built by other developers. In the case of 'Web APIs', that logic and data is exposed over the network.

## **What is an API proxy?**

You expose APIs on Apigee Edge by implementing API proxies. API proxies decouple the app-facing API from your backend services, shielding those apps from backend code changes. As you make backend changes to your services, apps continue to call the same API without any interruption.

In an API proxy configuration, there are two types of endpoints:

* **ProxyEndpoint:** Defines the way client apps consume your APIs. You configure the ProxyEndpoint to define the URL of your API proxy. The proxy endpoint also determines whether apps access the API proxy over HTTP or HTTPS. You usually attach policies to the ProxyEndpoint to enforce security, quota checks, and other types of access control and rate-limiting.
* **TargetEndpoint:** Defines the way the API proxy interacts with your backend services. You configure the TargetEndpoint to forward requests to the proper backend service, including defining any security settings, HTTP or HTTPS protocol, and other connection information. You can attach policies to the TargetEndpoint to ensure that response messages are properly formatted for the app that made the initial request.

Request and response variables

When you make a request to an API proxy, you can pass any or all of the following information, depending on the way the API proxy is configured:

* Request headers
* Query params
* Form data
* XML or JSON payloads
* Resource URIs

https://docs.apigee.com/api-platform/fundamentals/understanding-handling-request-response-data

### **Common policies to access flow variables**

Edge defines several policies that you can use to process the request and response data. These policies include:

* [AssignMessage policy](https://docs.apigee.com/api-platform/reference/policies/assign-message-policy): Creates or modifies HTTP request or response messages during an API proxy flow. Also creates and populates new flow variables.
* [ExtractVariables policy](https://docs.apigee.com/api-platform/reference/policies/extract-variables-policy): Extract content from messages, including headers, URI paths, payloads, and query parameters, for use in a condition statement. The policy then applies a text pattern to message content and upon finding a match sets a designated variable.
* [JSONtoXML policy](https://docs.apigee.com/api-platform/reference/policies/json-xml-policy) and [XMLtoJSON policy](https://docs.apigee.com/api-platform/reference/policies/xml-json-policy): Converts messages from JavaScript Object Notation (JSON) to the extensible markup language (XML) format, or vice versa.
* [JavaCallout policy](https://docs.apigee.com/api-platform/reference/policies/java-callout-policy), [JavaScript policy](https://docs.apigee.com/api-platform/reference/policies/javascript-policy), [PythonScript policy](https://docs.apigee.com/api-platform/reference/policies/python-script-policy), [RegularExpressionProtection policy](https://docs.apigee.com/api-platform/reference/policies/regular-expression-protection): These policies let you write a script to access flow variables containing request and response data.

### **https://docs.apigee.com/api-platform/reference/policies/access-control-policy**

### **Samples**

The mask values in the following IPv4 samples identify which of the four octets (8, 16, 24, 32 bits) the match rule considers when allowing or denying access. The default value is 32. See the [mask](https://docs.apigee.com/api-platform/reference/policies/access-control-policy#sourceaddress) attribute in the Element reference for more information.

[DENY 10.10.10.10](https://docs.apigee.com/api-platform/reference/policies/access-control-policy#deny-10101010)[DENY 10.10.10.\*](https://docs.apigee.com/api-platform/reference/policies/access-control-policy#deny-101010)[DENY 10.10.\*.\*](https://docs.apigee.com/api-platform/reference/policies/access-control-policy#deny-1010)[DENY 10.10.10.\*, ALLOW 10.10.10.20](https://docs.apigee.com/api-platform/reference/policies/access-control-policy#deny-101010-allow-10101020)[MORE](https://docs.apigee.com/api-platform/reference/policies/access-control-policy)

<AccessControl name="ACL">  
  <IPRules noRuleMatchAction = "ALLOW">  
    <MatchRule action = "DENY">  
      <SourceAddress mask="32">10.10.10.10</SourceAddress>  
    </MatchRule>  
  </IPRules>  
</AccessControl>

Deny all requests from client address: 10.10.10.10

Allow requests from any other client address.

When you send API requests you are trying to get a service to 'do something.' The service will then tell you what it did or give you what it asked for. The payload is the part of that response that is communicating directly to you. In REST APIs this is usually some JSON formatted data.

Let's say you want to see a random picture of a cat (because you are on the internet). You go to The Cat API (yes it exists - this is the internet we are talking about) and you send an API request. You get back an JSON object with a link to a cat picture along with a few other pieces of information. That JSON is the payload.

Payload isn't just something you get from the server though. You can also send payload to the server as well. This is known as the body of your request and is once again usually a JSON object that you send to the server. It is used to make more complex requests or when you are trying to create or modify an object on the service and you need to give it information about the object you are creating

Request & Response are payloads in Web Service.

Typically, it is

1. **XML**for **SOAP**based Web Services
2. **JSON**for **REST**based Web Services

* Choosing between path, payload, headers, and query parameters see
* Methods for calling back-end services (service call out and othe r see)
* Fault handling and error conditions