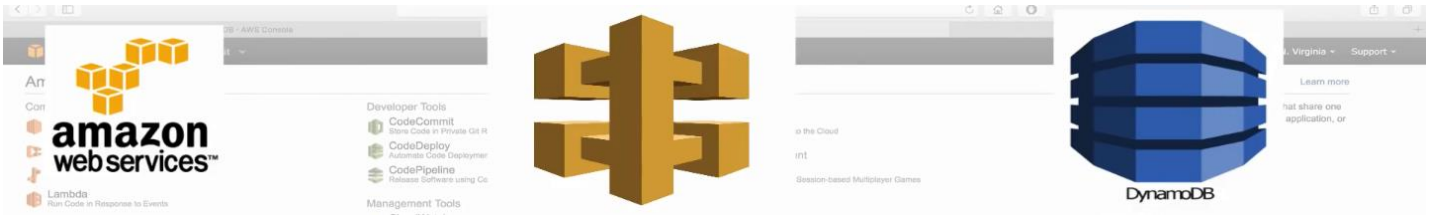


In this video I go over a live example of a very simple DynamoDB table designed to store user comments, and then through the detailed steps required to expose update/query operations on that table over a RestFUL API using Amazon API Gateway. DynamoDB is Amazon's cloud hosted NoSQL database.

API Gateway is used here to define and deploy a fulling functional, externally accessible HTTP/rest API to delegate some instructions to the DynamoDB backend. DynamoDB already exposes web services to directly manipulate the data but they are all POST services. There are several advantages to wrapping AWS services (such as Lambda functions as well) in API Gateway services. One of them here is that you can create a GET service to retrieve data from Dynamo which wraps the POST based services available directly against the DynamoDB.

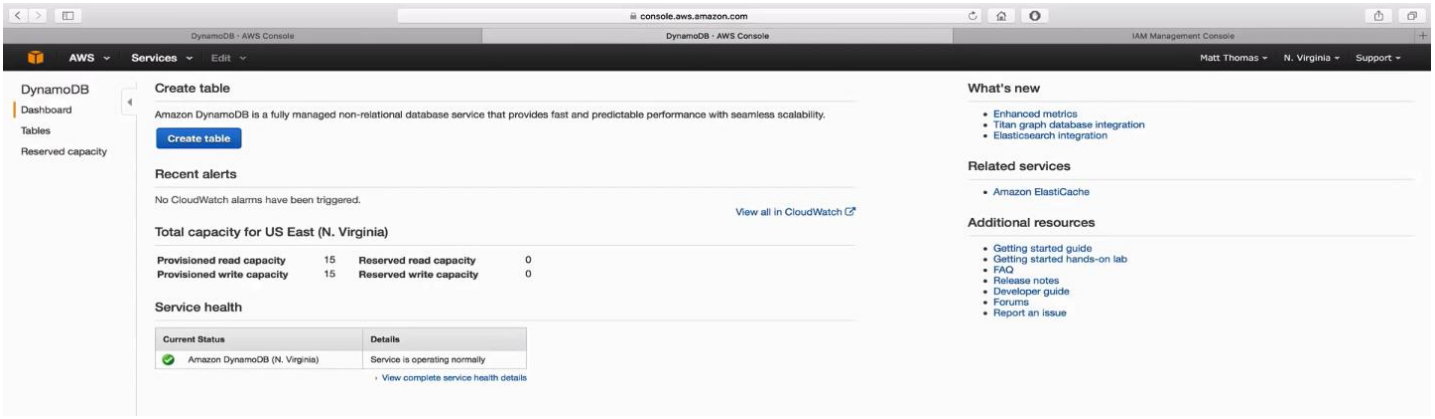
IAM in this example is used to configure the permissions for users/roles/groups to perform operations on AWS resources. In this example we ***create policies to update and insert into DynamoDB table*** and then inform our API Gateway to run the API invocation with that role.

This is an excellent tutorial but it doesn't include all the details you need on setting up the IAM role in particular the trust relationship and DynamoDB policies. Here is the Trust Relationship I used for the role I created for this example - just copy paste it into the JSON editor for the Trust Relationship

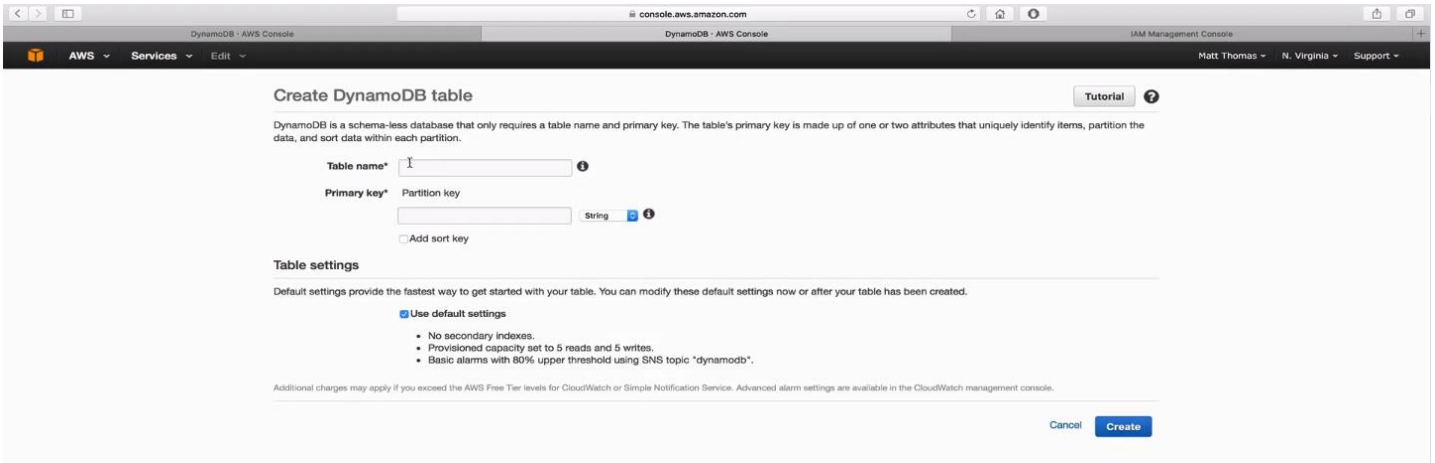
```
{
  "Version": "2012-10-17",
  "Statement": [
      {
    "Sid": "",
    "Effect": "Allow",
    "Principal": {
      "Service": "apigateway.amazonaws.com"
    },
    "Action": "sts:AssumeRole"
  }
 ]
}
```

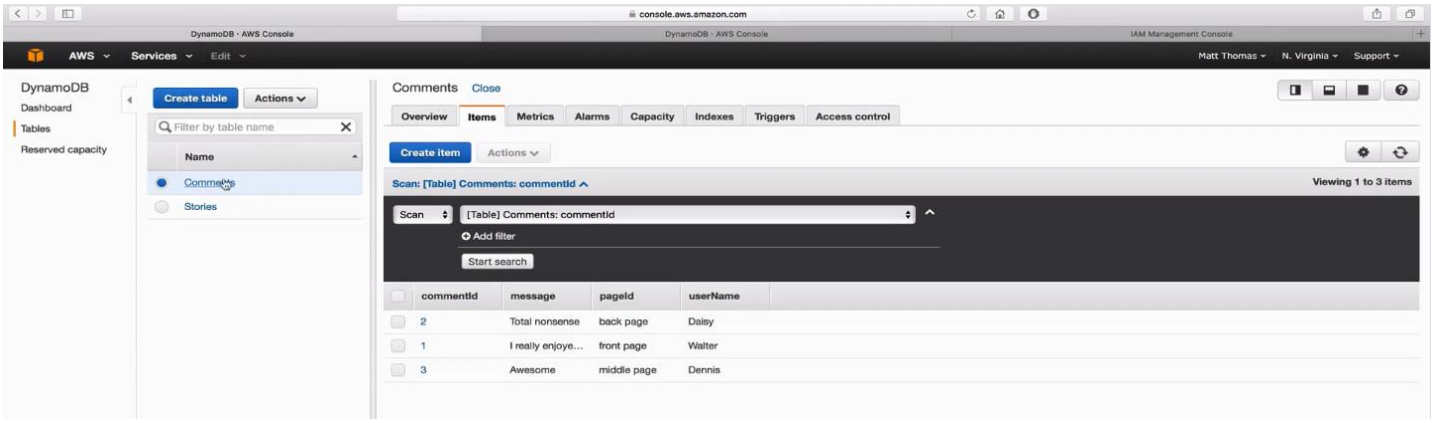
Here is the policy I created to be able to add to and query my Comments Dynamo DB table. YOU WILL HAVE TO CHANGE IT FOR YOUR OWN ACCOUNT - these are referring the to the database resources belonging to my account. The wildcard entry ensures that all sub objects of the main table (such as indexes) are also permission

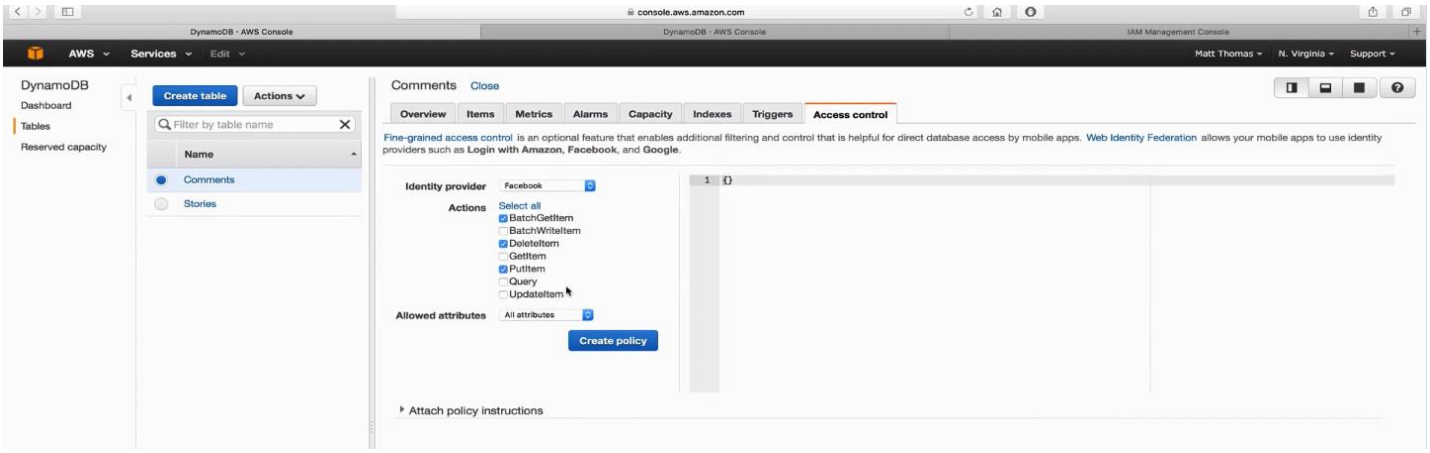
```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "dynamodb:PutItem",
            "dynamodb:Query"
         ],
         "Resource": [
            "arn:aws:dynamodb:us-east-1:ACCOUNT:table/Comments/*",
            "arn:aws:dynamodb:us-east-1:ACCOUNT:table/Comments"
         ]
      }
   ]
}
```


console.aws.amazon.com
DynamoDB · AWS Console
DynamoDB · AWS Console
IAM Management Console
AWS
Services
Edit
Matt Thomas
N. Virginia
Support
DynamoDB
Dashboard
Tables
Reserved capacity
Create table
Amazon DynamoDB is a fully managed non-relational database service that provides fast and predictable performance with seamless scalability.
Create table
Recent alerts
No CloudWatch alarms have been triggered.
View all in CloudWatch
Total capacity for US East (N. Virginia)
Provisioned read capacity 15
Provisioned write capacity 15
Reserved read capacity 0
Reserved write capacity 0
Service health
Current Status
Details
Amazon DynamoDB (N. Virginia)
Service is operating normally
View complete service health details
What's new
Enhanced metrics
Titan graph database integration
Elasticsearch integration
Related services
Amazon ElastiCache
Additional resources
Getting started guide
Getting started hands-on lab
FAQ
Release notes
Developer guide
Forums
Report an issue

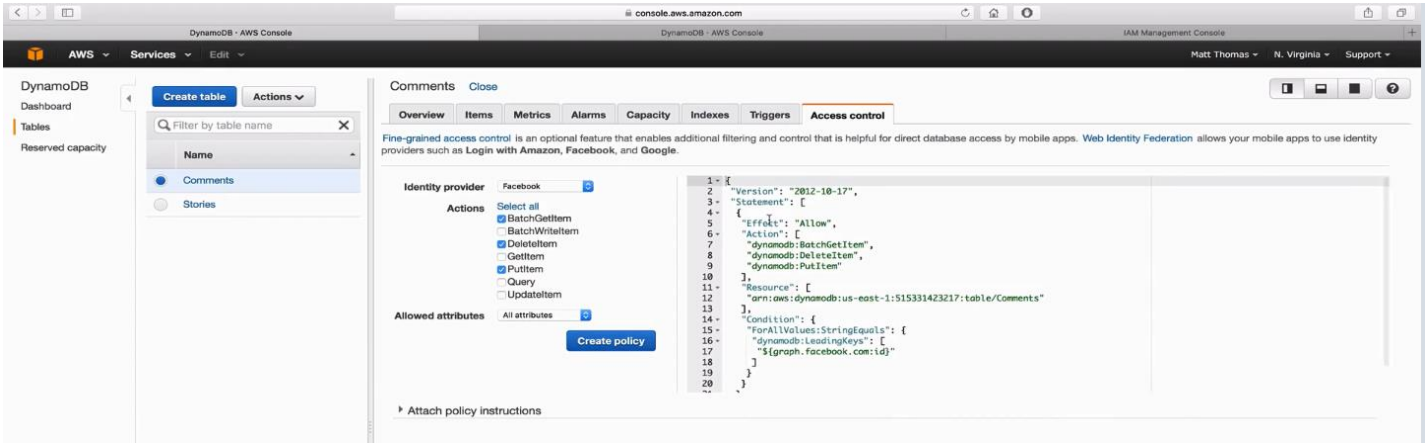

console.aws.amazon.com
DynamoDB · AWS Console
DynamoDB · AWS Console
IAM Management Console
AWS
Services
Edit
Matt Thomas
N. Virginia
Support
DynamoDB
Dashboard
Tables
Reserved capacity
Create table
Actions
Filter by table name
Viewing 2 of 2 Tables
Name
Status
Partition key
Sort key
Indexes
Total read capacity
Total write capacity
Comments
Active
commentId (String)
-
1
10
10
Stories
Active
storyId (String)
-
0
5
5


console.aws.amazon.com
DynamoDB · AWS Console
DynamoDB · AWS Console
IAM Management Console
AWS
Services
Edit
Matt Thomas
N. Virginia
Support
Create DynamoDB table
Tutorial
DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.
Table name*
Primary key*
Partition key
String
Add sort key
Table settings
Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.
Use default settings
No secondary indexes.
Provisioned capacity set to 5 reads and 5 writes.
Basic alarms with 80% upper threshold using SNS topic "dynamodb".
Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.
Cancel
Create

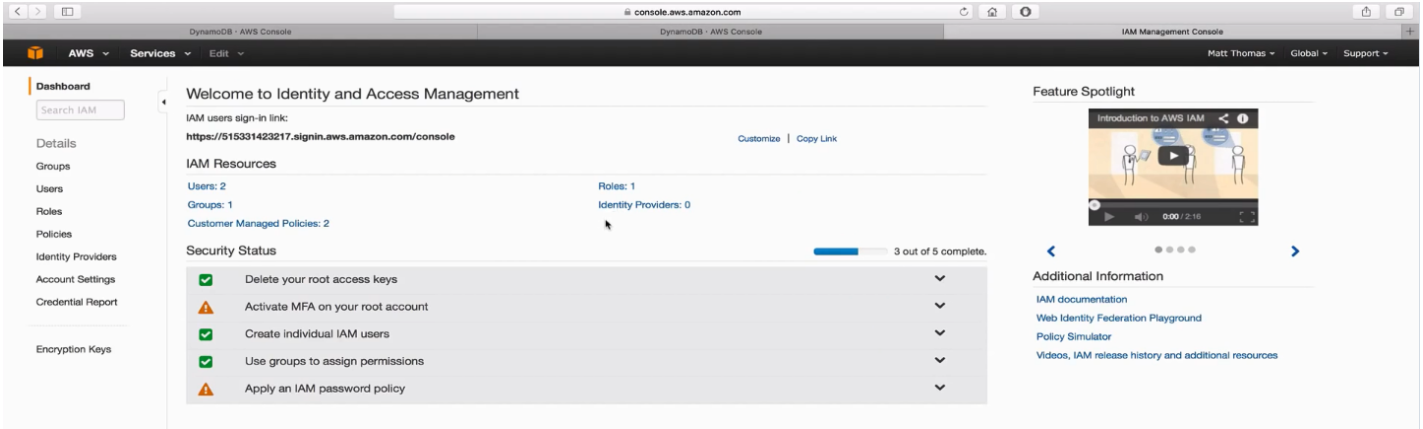

console.aws.amazon.com
DynamoDB · AWS Console
DynamoDB · AWS Console
IAM Management Console
AWS
Services
Edit
Matt Thomas
N. Virginia
Support
DynamoDB
Dashboard
Tables
Reserved capacity
Create table
Actions
Filter by table name
Name
Comments
Stories
Comments
Close
Overview
Items
Metrics
Alarms
Capacity
Indexes
Triggers
Access control
Create item
Actions
Scan: [Table] Comments: commentId
Viewing 1 to 3 items
Scan
[Table] Comments: commentId
Add filter
Start search
commentId
message
pageId
userName
2
Total nonsense
back page
Daisy
1
I really enjoye...
front page
Walter
3
Awesome
middle page
Dennis



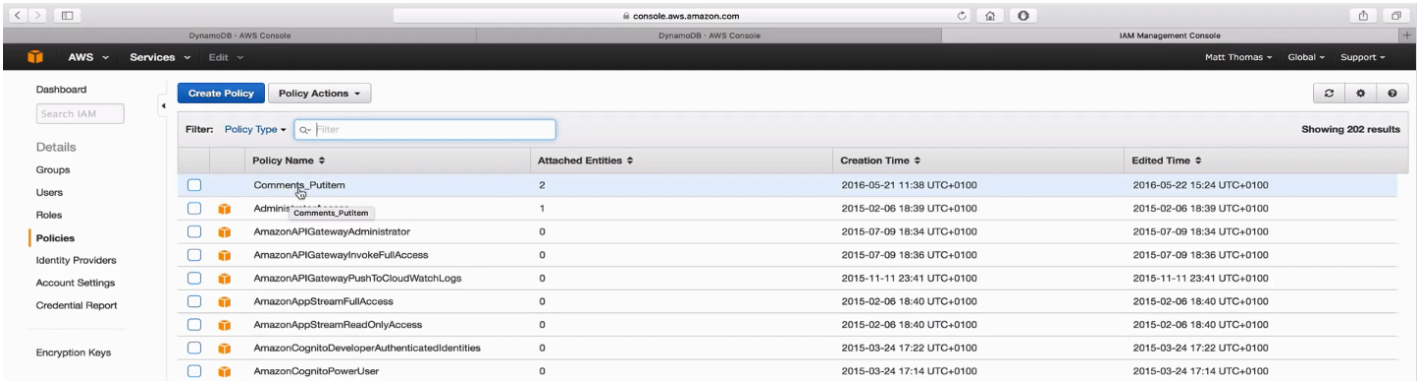
You can also specify the things that can be done on this table and generate a JSON permissions representation as below



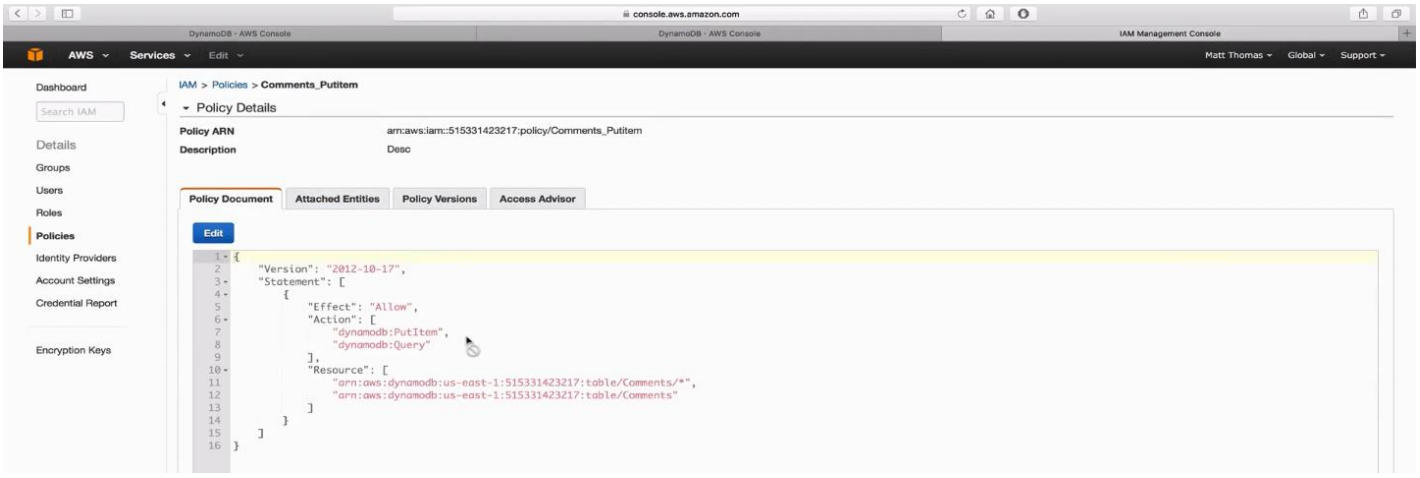
It also creates this JSON document for you that you can then paste into the policy document for this table. This will be used to assign permissions on this table to certain Roles for the users.



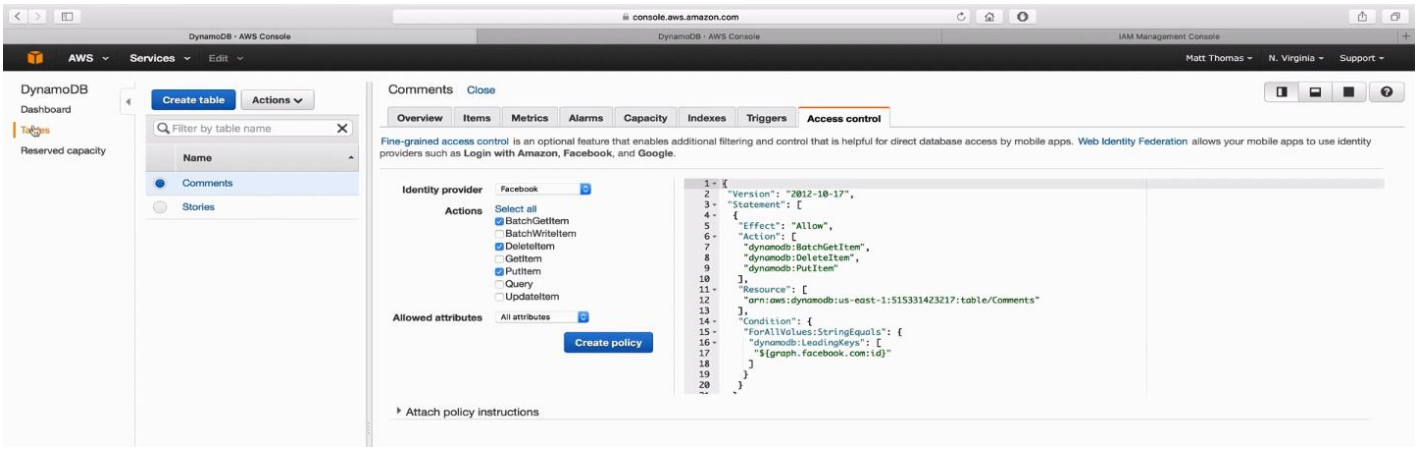
IAM service helps to manage all your resources permissions using Roles, Groups, Policies, etc.

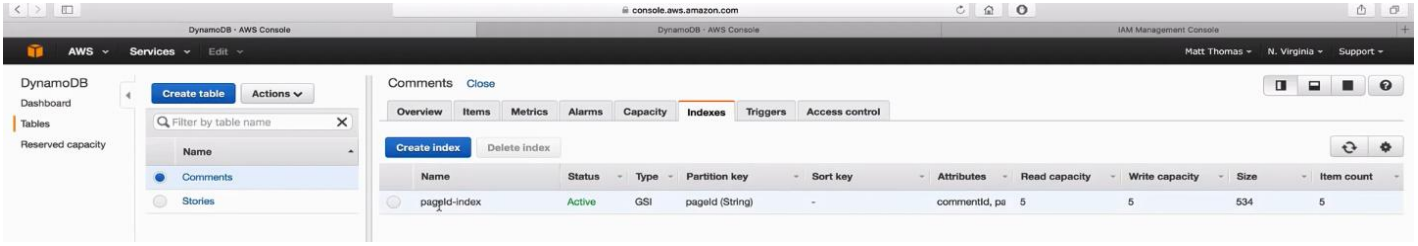


These are all the default policies that AWS has created for use. We have used the ***Create Policy*** button to create our own ***Comments_Putitem*** policy above.

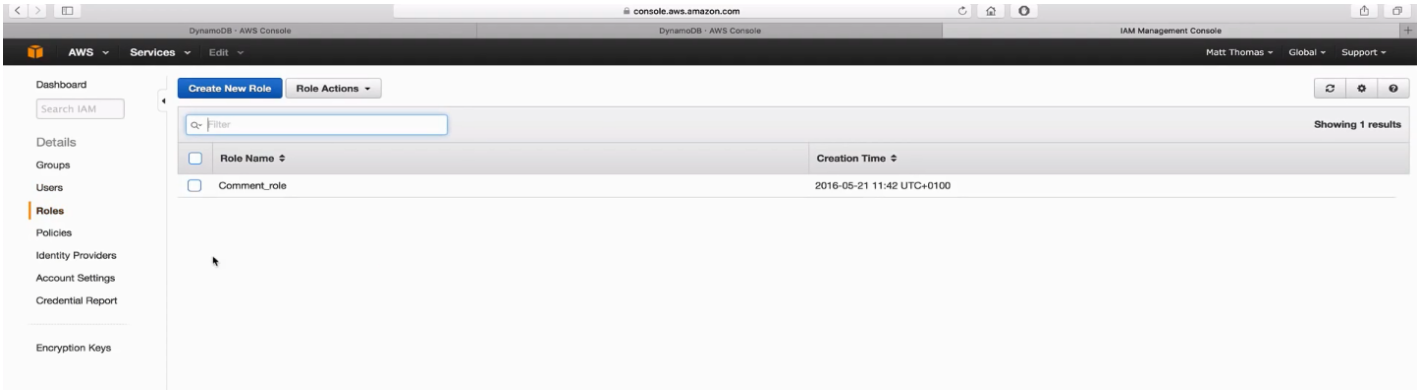


This is the policy, it is the JSON syntax that we created much earlier. It describes what people having this policy can do to the Comments table, the ***PutItem*** and ***Query*** operations can be performed on the table resources listed.

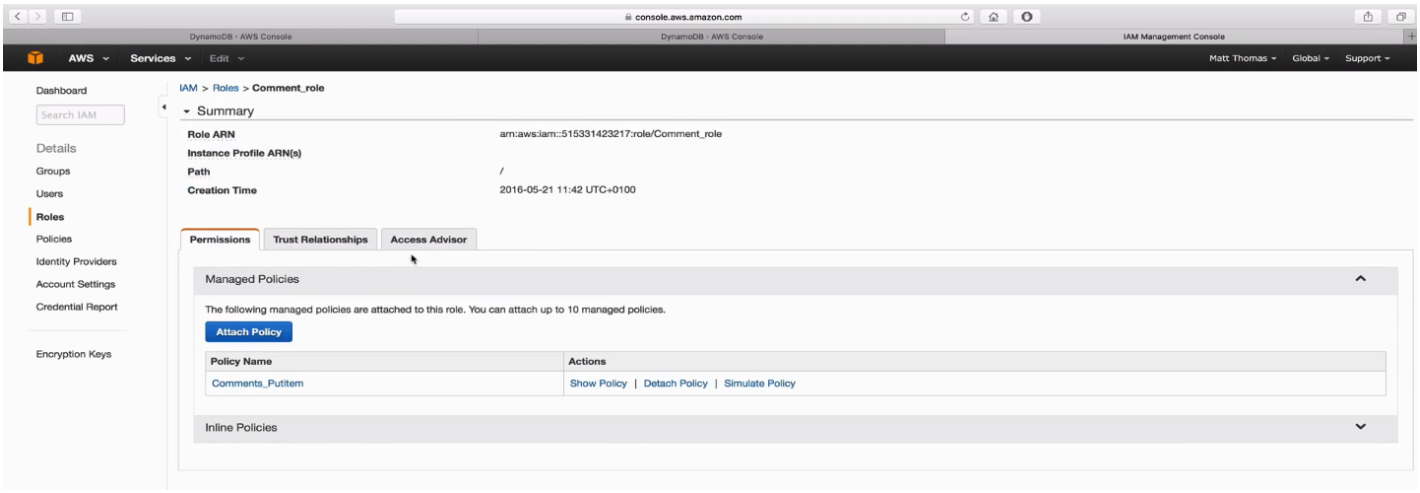




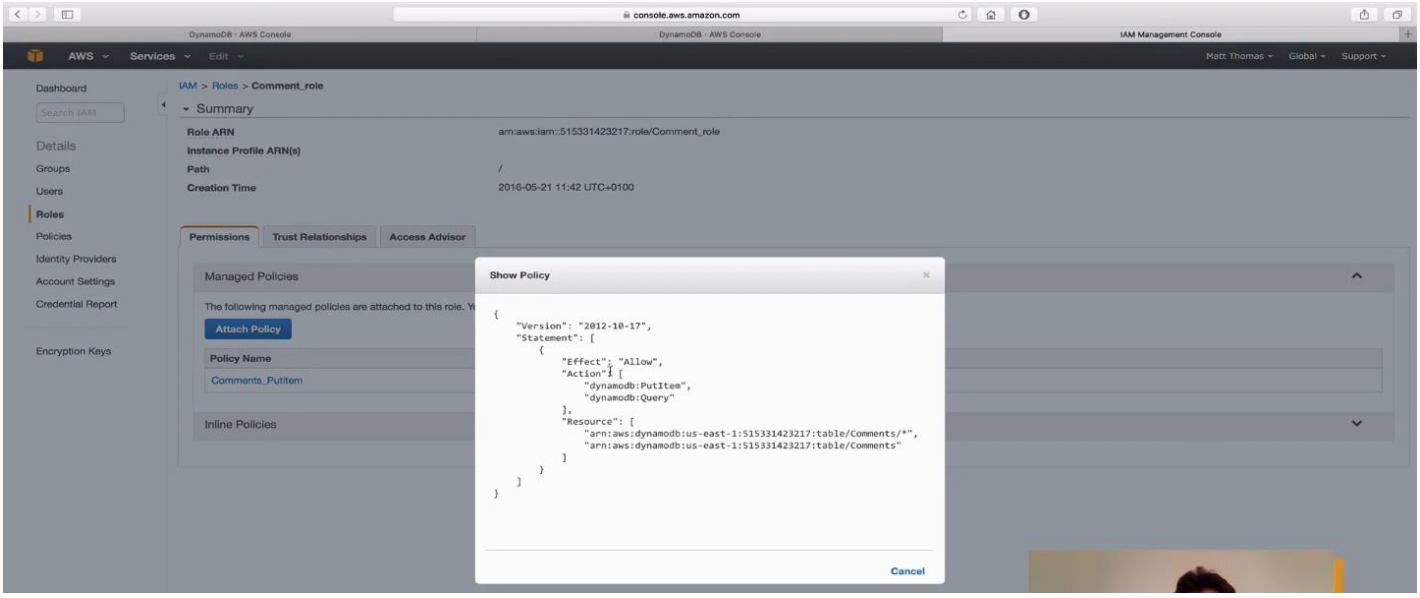
There is an index called ***pageId-index*** created on the Comments table that enables us to query this table, we needed to also assign permission for Query to be able to query the …Comments/* table.



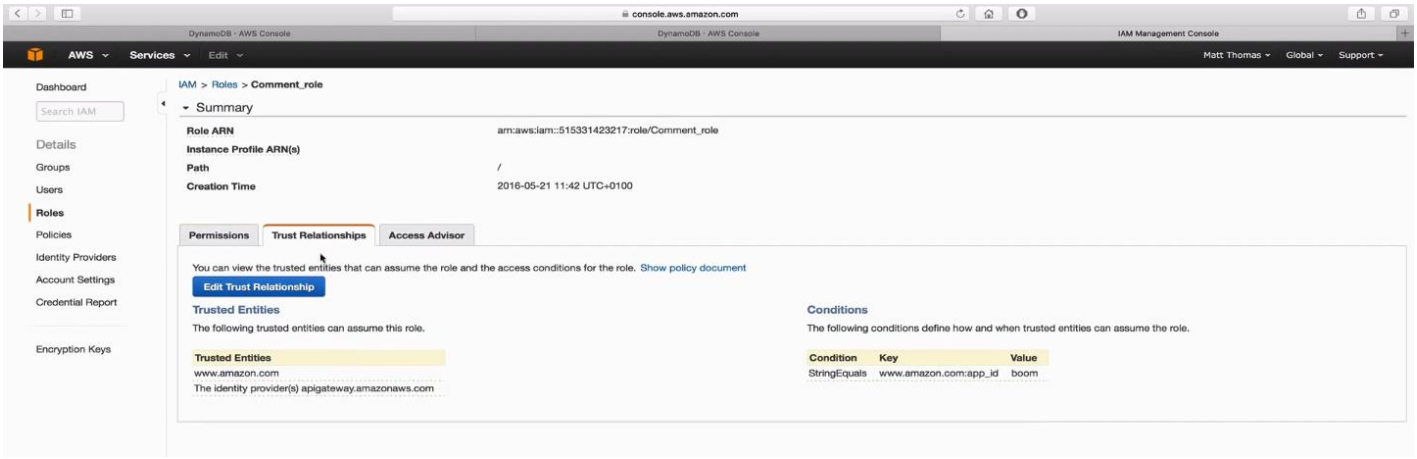
A Policy has to be assigned to a user or to a Role,



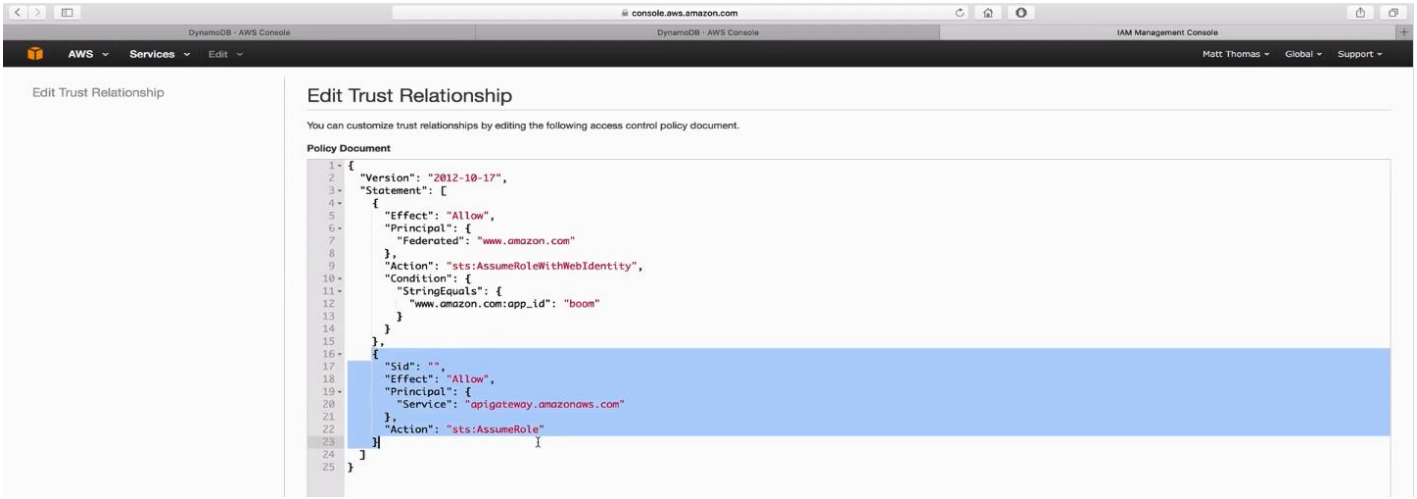
We have attached the ***Comments_Putitem*** policy to this ***Comment_role*** Role,



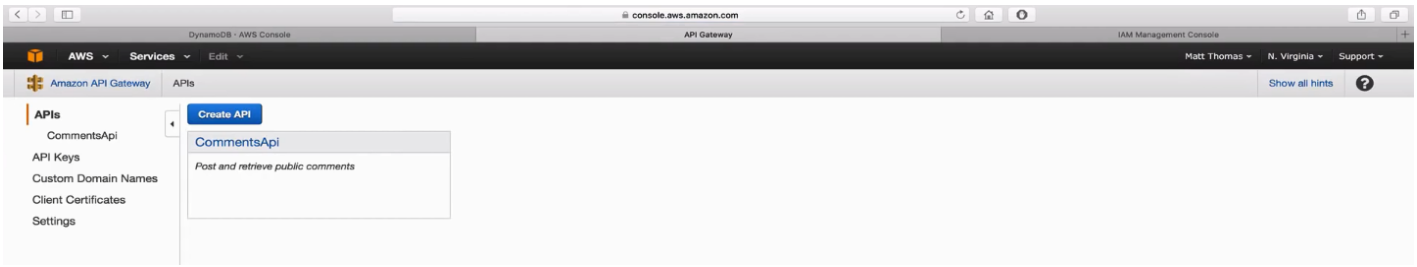
This is the policy used for this role.



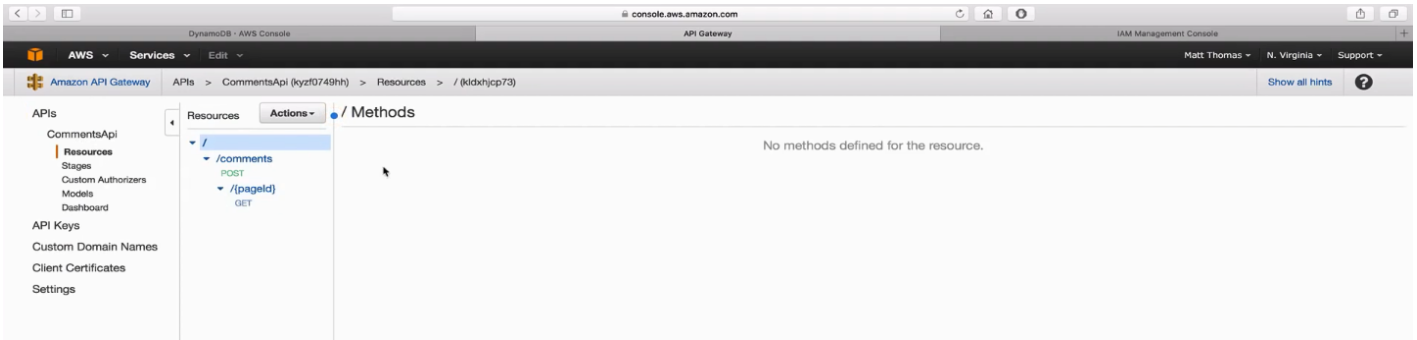
You also have to define a trust relationship for this role,

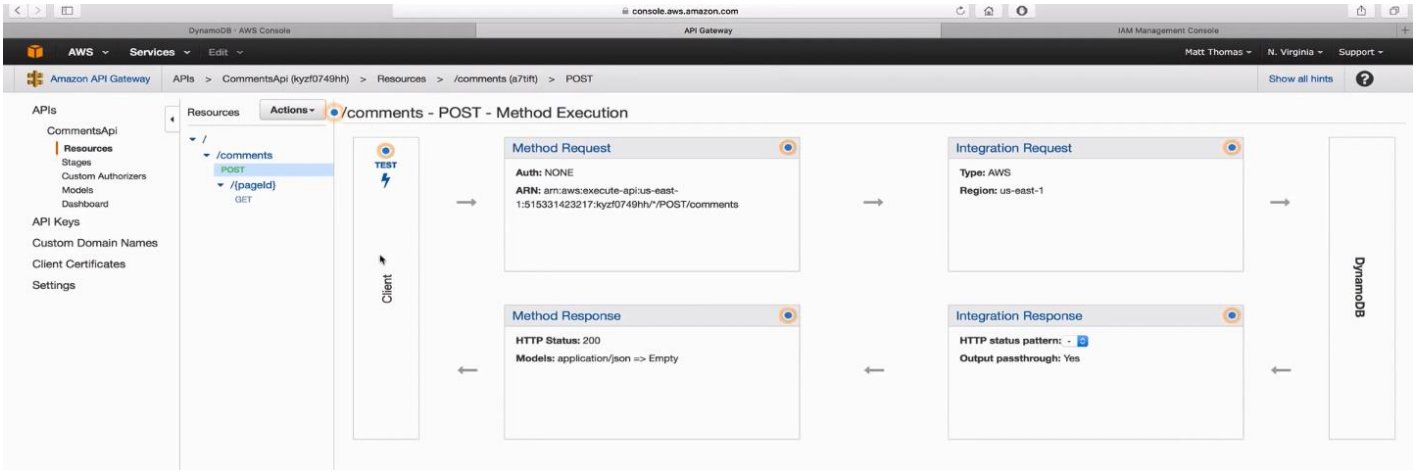


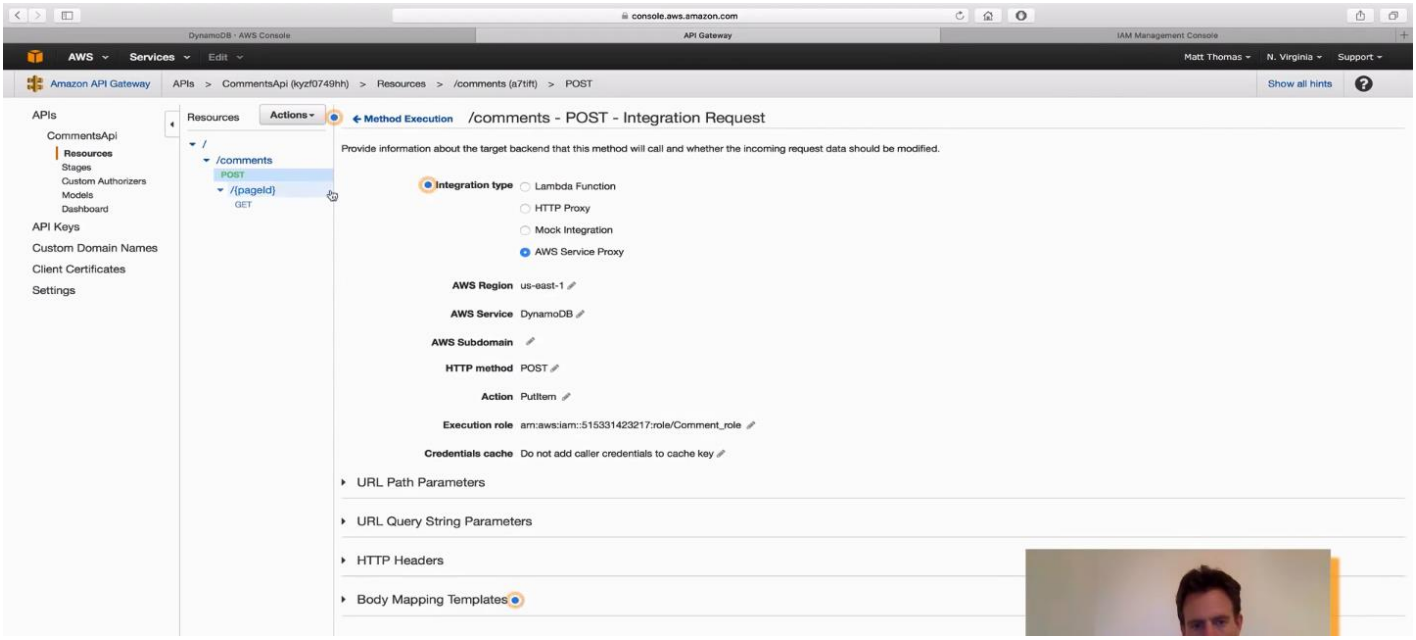
You have to put this entry here that basically says that you are allowing the API Gateway from amazon.com to assume this role and be able to do the things it needs executed

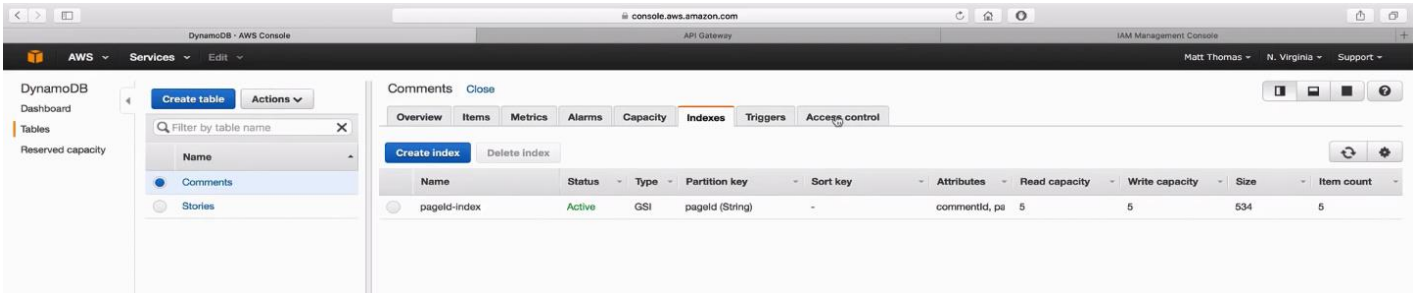


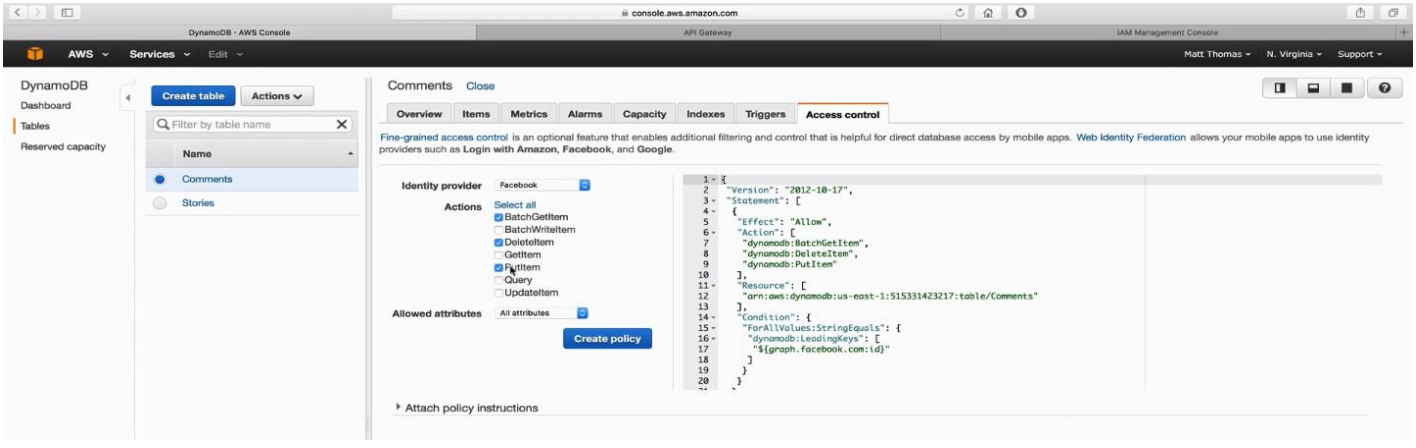
This is the CommentsApi that we created in the API Gateway service.

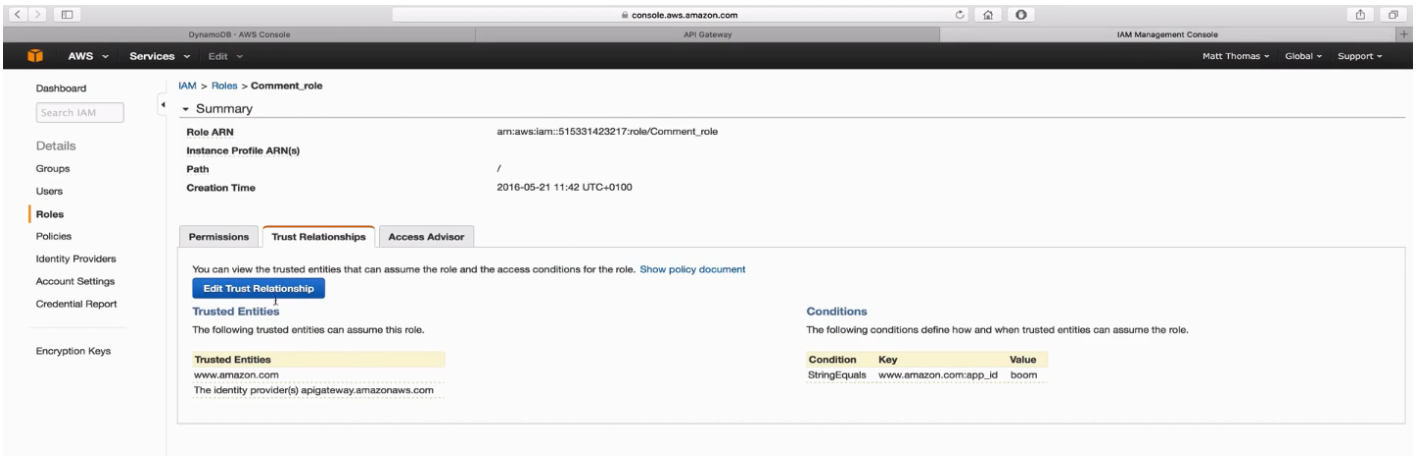


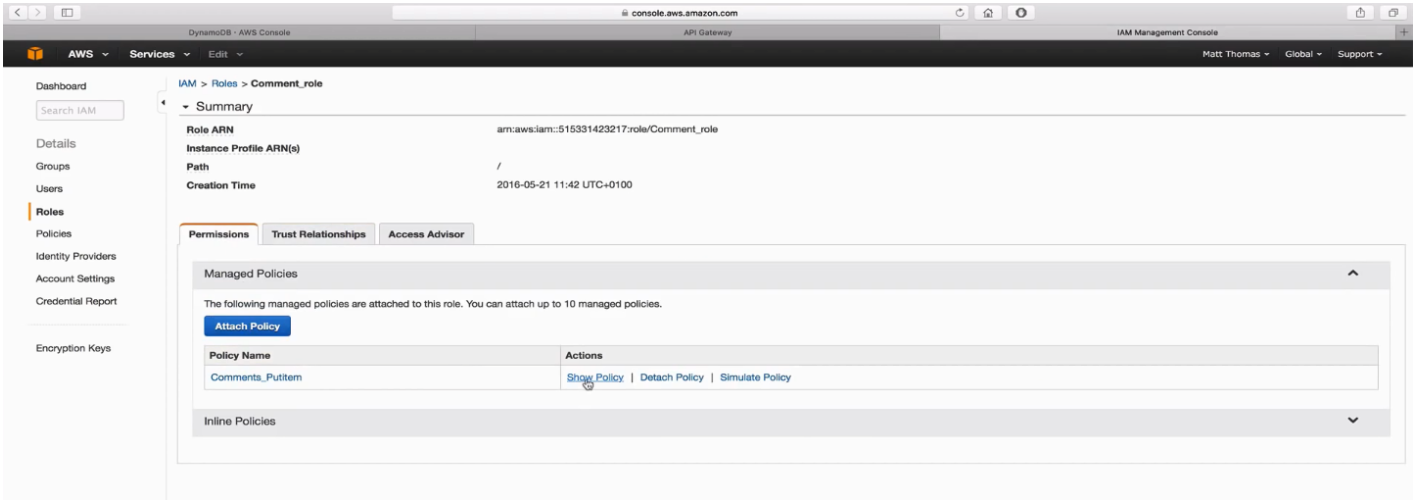

/comments - POST - Method Execution
Method Request
Auth: NONE
ARN: arn:aws:execute-api:us-east-1:515331423217:kyzf0749hh/*/POST/comments
Integration Request
Type: AWS
Region: us-east-1
Method Response
HTTP Status: 200
Models: application/json => Empty
Integration Response
HTTP status pattern: -
Output passthrough: Yes
Client
DynamoDB

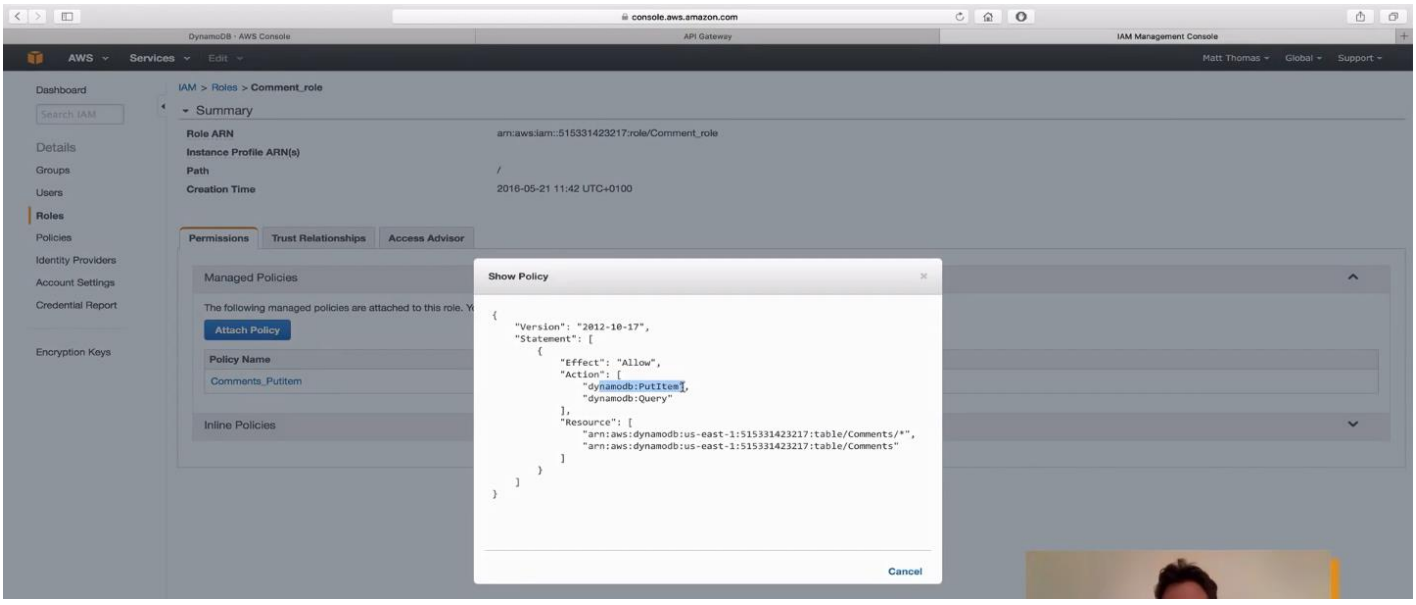

/comments - POST - Integration Request
Provide information about the target backend that this method will call and whether the incoming request data should be modified.
Integration type
Lambda Function
HTTP Proxy
Mock Integration
AWS Service Proxy
AWS Region us-east-1
AWS Service DynamoDB
AWS Subdomain
HTTP method POST
Action PutItem
Execution role arn:aws:iam::515331423217:role/Comment_role
Credentials cache Do not add caller credentials to cache key
URL Path Parameters
URL Query String Parameters
HTTP Headers
Body Mapping Templates

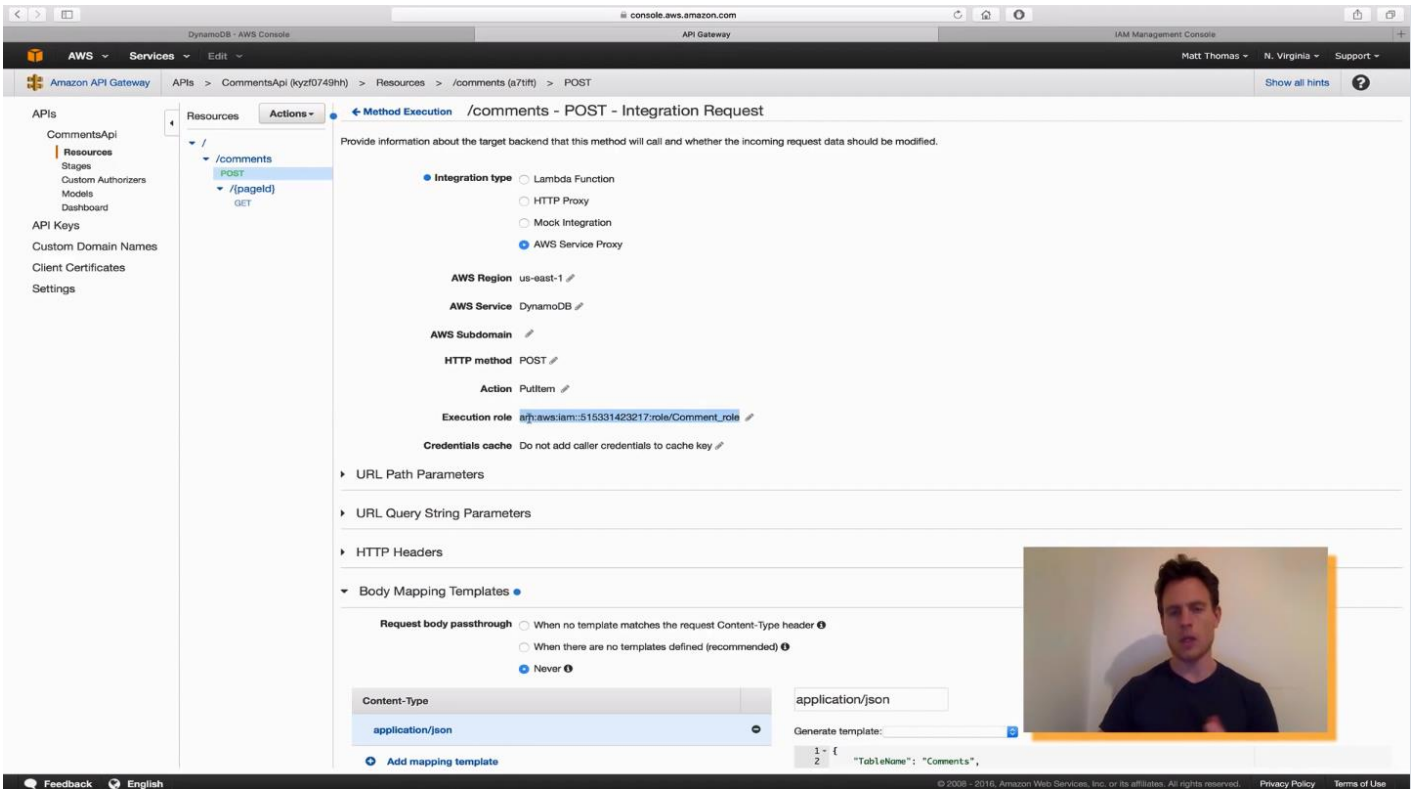

DynamoDB
Create table
Actions
Comments
Stories
Comments Close
Overview
Items
Metrics
Alarms
Capacity
Indexes
Triggers
Access control
Create index
Delete index
Name
Status
Type
Partition key
Sort key
Attributes
Read capacity
Write capacity
Size
Item count
pageId-index
Active
GSI
pageId (String)
5
5
534
5


Access control
Fine-grained access control is an optional feature that enables additional filtering and control that is helpful for direct database access by mobile apps. Web Identity Federation allows your mobile apps to use identity providers such as Login with Amazon, Facebook, and Google.
Identity provider Facebook
Actions Select all
BatchGetItem
BatchWriteItem
DeleteItem
GetItem
PutItem
Query
UpdateItem
Allowed attributes All attributes
Create policy
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": [
"dynamodb:BatchGetItem",
"dynamodb:DeleteItem",
"dynamodb:PutItem"
],
"Resource": [
"arn:aws:dynamodb:us-east-1:515331423217:table/Comments"
],
"Condition": {
"ForAllValues:StringEquals": {
"dynamodb:LeadingKeys": [
"${graph.facebook.com:id}"
]
}
}
Attach policy instructions

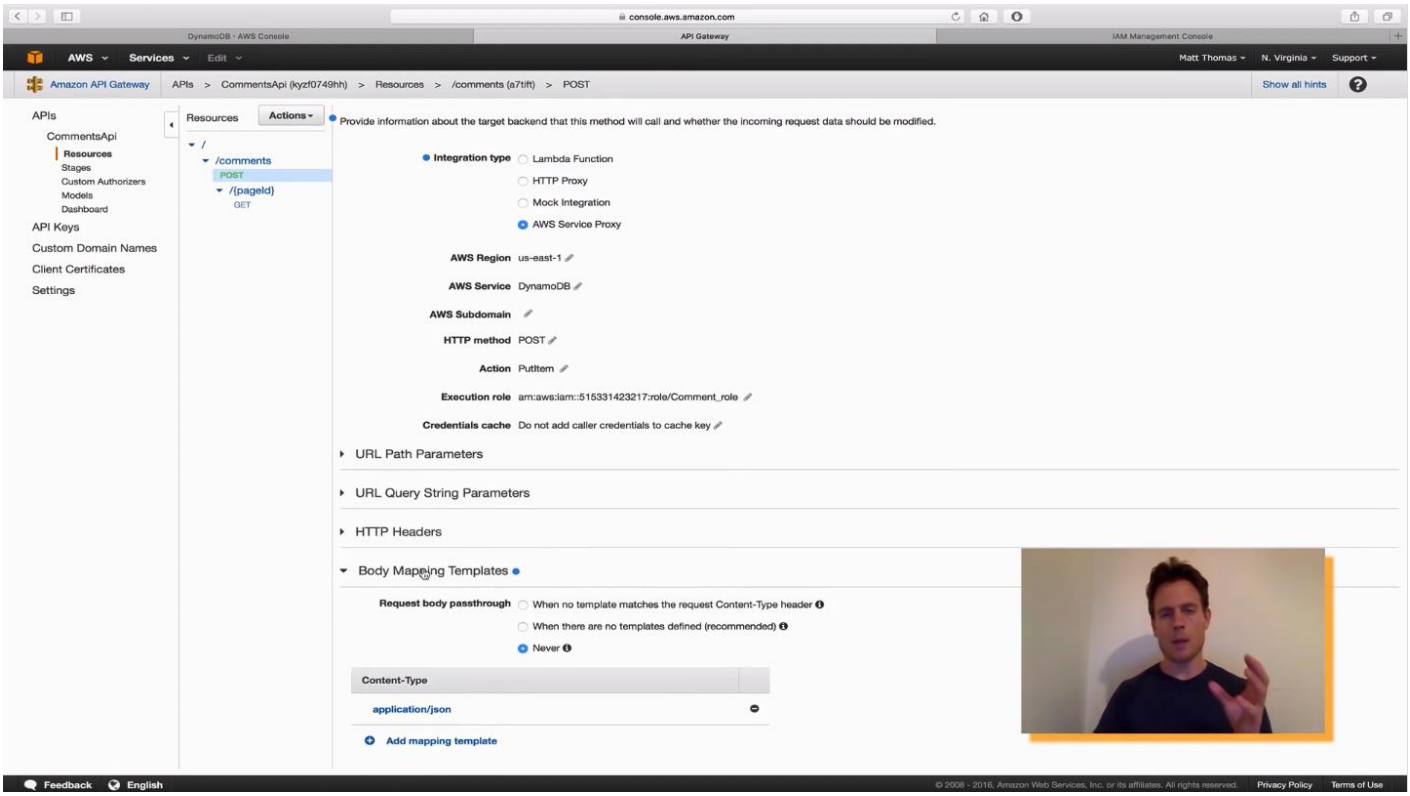

console.aws.amazon.com
DynamoDB - AWS Console
API Gateway
IAM Management Console
AWS
Services
Edit
Matt Thomas
Global
Support
Dashboard
Search IAM
Details
Groups
Users
Roles
Policies
Identity Providers
Account Settings
Credential Report
Encryption Keys
IAM > Roles > Comment_role
Summary
Role ARN
arn:aws:iam::515331423217:role/Comment_role
Instance Profile ARN(s)
Path
/
Creation Time
2016-05-21 11:42 UTC+0100
Permissions
Trust Relationships
Access Advisor
You can view the trusted entities that can assume the role and the access conditions for the role. Show policy document
Edit Trust Relationship
Trusted Entities
The following trusted entities can assume this role.
Trusted Entities
www.amazon.com
The identity provider(s) apigateway.amazonaws.com
Conditions
The following conditions define how and when trusted entities can assume the role.
Condition
Key
Value
StringEquals
www.amazon.com:app_id
boom


console.aws.amazon.com
DynamoDB - AWS Console
API Gateway
IAM Management Console
AWS
Services
Edit
Matt Thomas
Global
Support
Dashboard
Search IAM
Details
Groups
Users
Roles
Policies
Identity Providers
Account Settings
Credential Report
Encryption Keys
IAM > Roles > Comment_role
Summary
Role ARN
arn:aws:iam::515331423217:role/Comment_role
Instance Profile ARN(s)
Path
/
Creation Time
2016-05-21 11:42 UTC+0100
Permissions
Trust Relationships
Access Advisor
Managed Policies
The following managed policies are attached to this role. You can attach up to 10 managed policies.
Attach Policy
Policy Name
Actions
Comments_PutItem
Show Policy | Detach Policy | Simulate Policy
Inline Policies

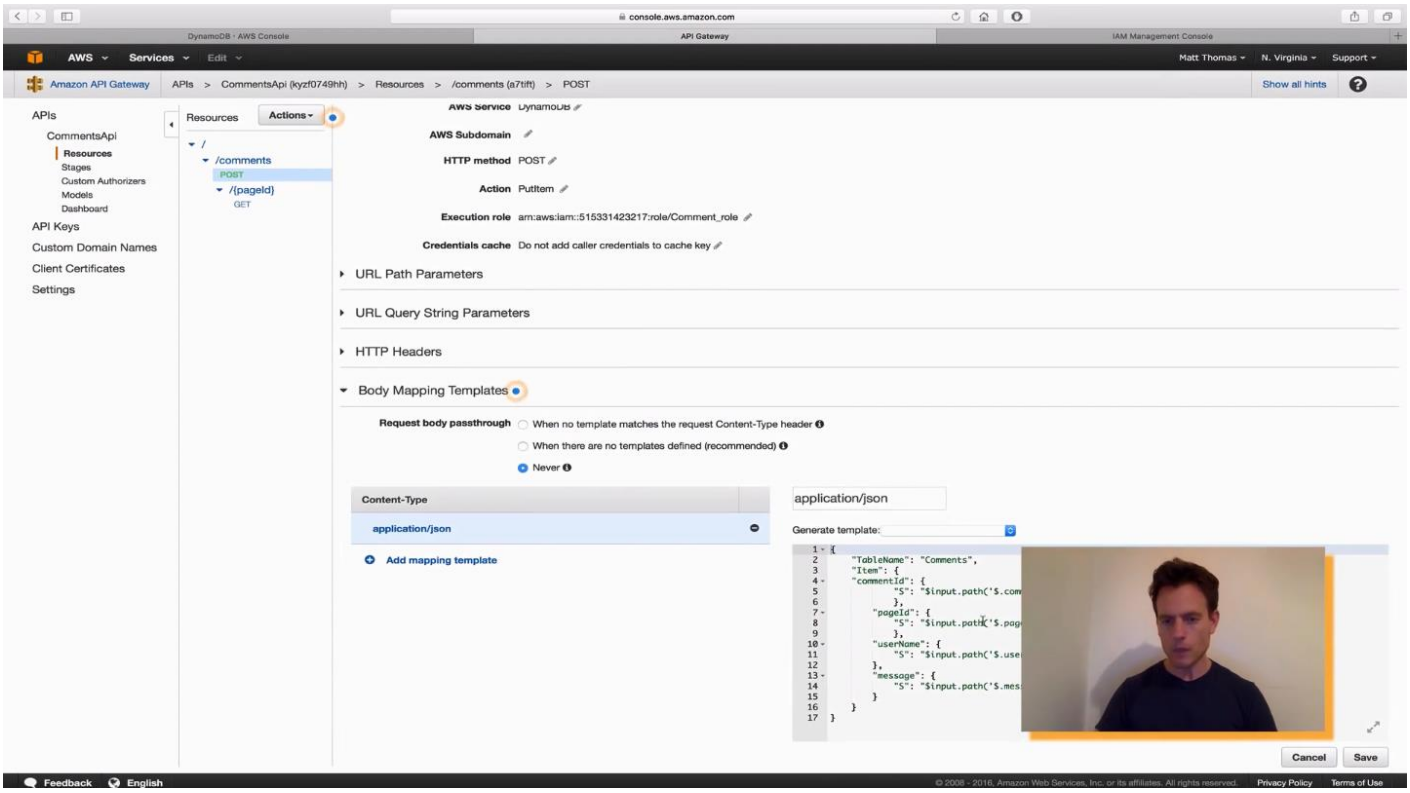

console.aws.amazon.com
DynamoDB - AWS Console
API Gateway
IAM Management Console
AWS
Services
Edit
Matt Thomas
Global
Support
Dashboard
Search IAM
Details
Groups
Users
Roles
Policies
Identity Providers
Account Settings
Credential Report
Encryption Keys
IAM > Roles > Comment_role
Summary
Role ARN
arn:aws:iam::515331423217:role/Comment_role
Instance Profile ARN(s)
Path
/
Creation Time
2016-05-21 11:42 UTC+0100
Permissions
Trust Relationships
Access Advisor
Managed Policies
Attach Policy
Policy Name
Comments_PutItem
Inline Policies
Show Policy
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": [
"dynamodb:PutItem",
"dynamodb:Query"
],
"Resource": [
"arn:aws:dynamodb:us-east-1:515331423217:table/Comments/*",
"arn:aws:dynamodb:us-east-1:515331423217:table/Comments"
]
}
]
}
Cancel



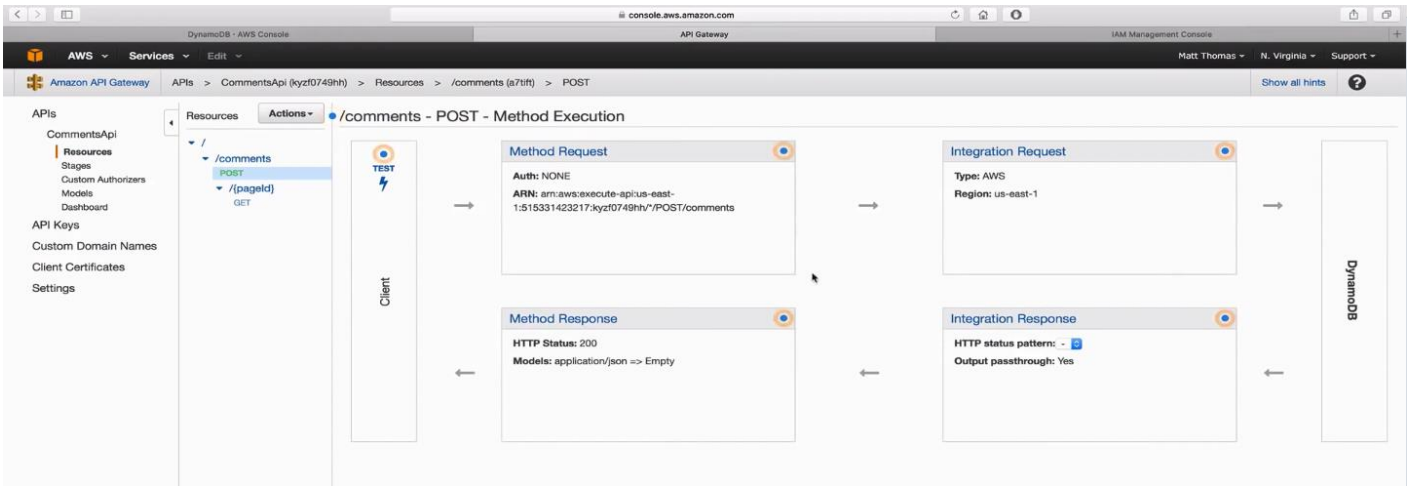
We also MUST specify the Role that can invoke this particular endpoint as the Comments Role above



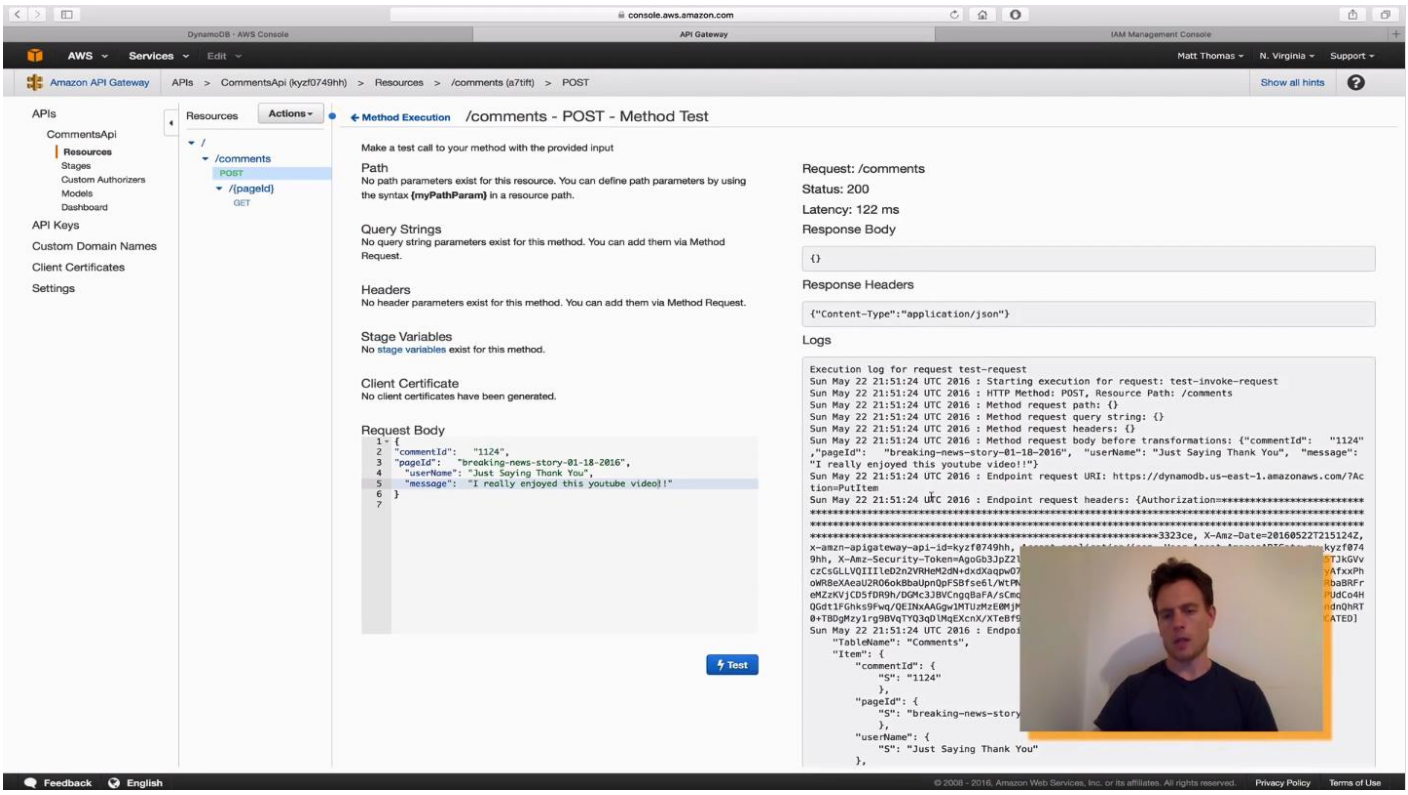
We are going to intercept the POST request from our API Gateway and then apply a mapping template to the body of the request, the Content-Type we are expecting is JSON

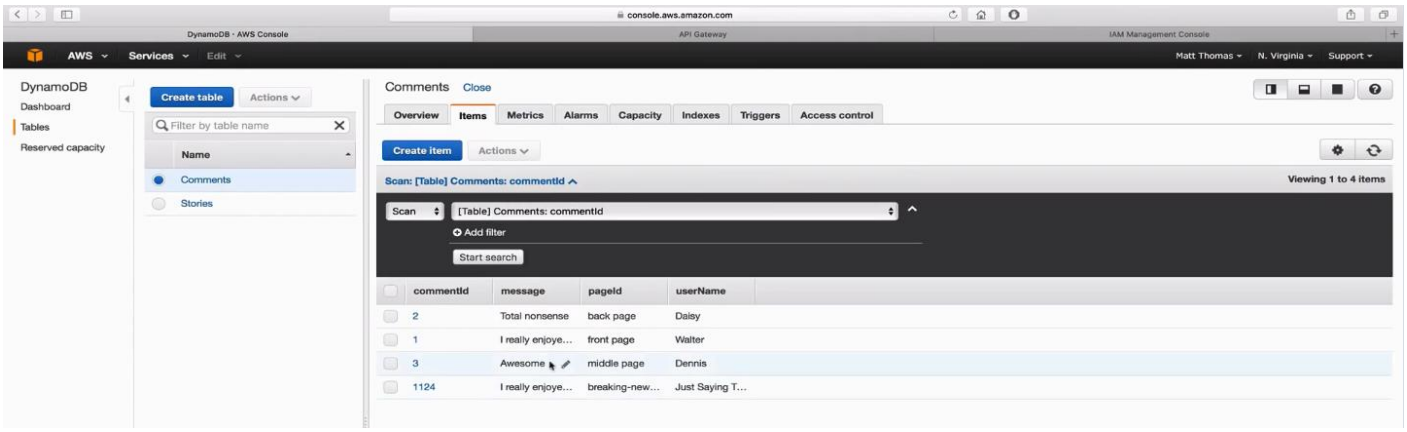


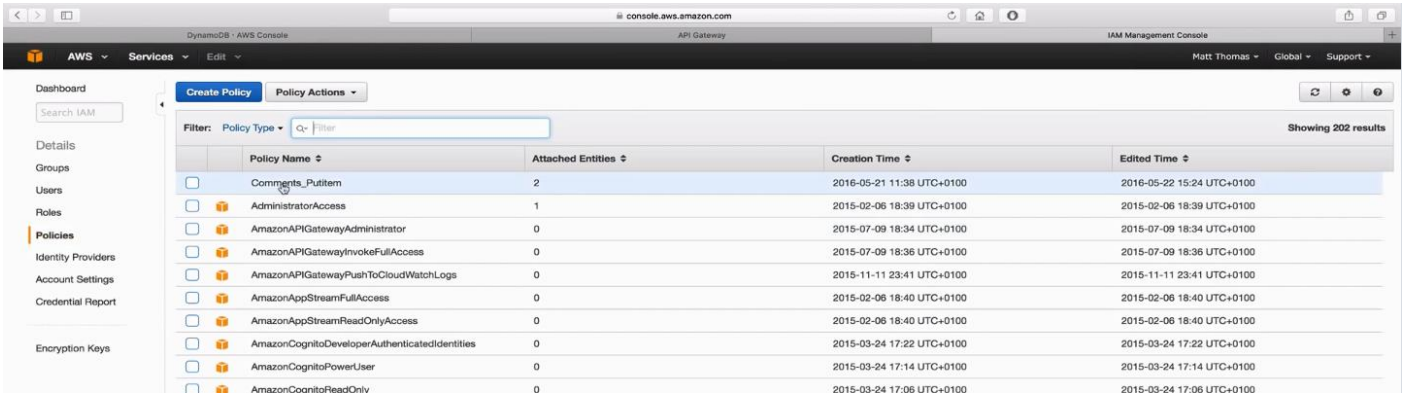
We will then take the expected JSON and transform it into a form that DynamoDB service's PutItem endpoint understands using the mapping template shown above. Note how we are hardcoding the TableName to be Comments so that this is the only table that this endpoint can interact with.

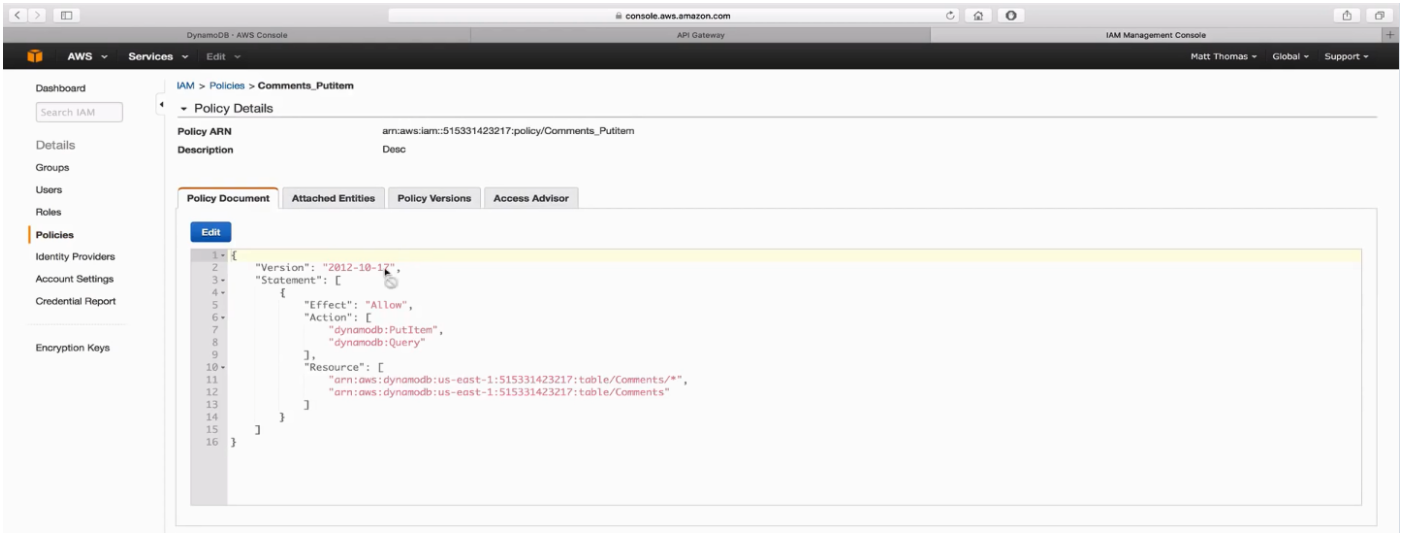


We can test this

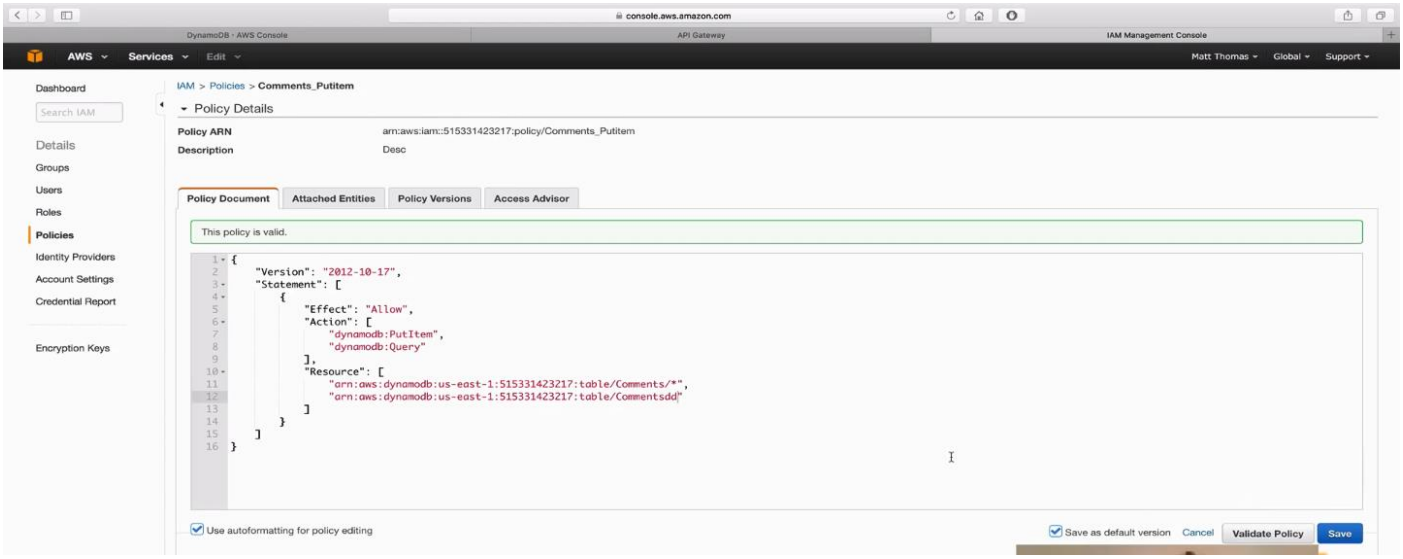

console.aws.amazon.com
DynamoDB - AWS Console
API Gateway
IAM Management Console
AWS
Services
Edit
Matt Thomas
N. Virginia
Support
Amazon API Gateway
APIs > CommentsApi (kyzf0749hh) > Resources > /comments (a7tift) > POST
Show all hints
APIs
CommentsApi
Resources
Stages
Custom Authorizers
Models
Dashboard
API Keys
Custom Domain Names
Client Certificates
Settings
Resources
Actions
/
/comments
POST
/{pageId}
GET
Method Execution
/comments - POST - Method Test
Make a test call to your method with the provided input
Path
No path parameters exist for this resource. You can define path parameters by using the syntax {myPathParam} in a resource path.
Query Strings
No query string parameters exist for this method. You can add them via Method Request.
Headers
No header parameters exist for this method. You can add them via Method Request.
Stage Variables
No stage variables exist for this method.
Client Certificate
No client certificates have been generated.
Request Body
{
"commentId": "1124",
"pageId": "breaking-news-story-01-18-2016",
"userName": "Just Saying Thank You",
"message": "I really enjoyed this youtube video!!"
}
Test
Request: /comments
Status: 200
Latency: 122 ms
Response Body
{}
Response Headers
{"Content-Type":"application/json"}
Logs
Execution log for request test-request
Sun May 22 21:51:24 UTC 2016 : Starting execution for request: test-invoke-request
Sun May 22 21:51:24 UTC 2016 : HTTP Method: POST, Resource Path: /comments
Sun May 22 21:51:24 UTC 2016 : Method request path: {}
Sun May 22 21:51:24 UTC 2016 : Method request query string: {}
Sun May 22 21:51:24 UTC 2016 : Method request headers: {}
Sun May 22 21:51:24 UTC 2016 : Method request body before transformations: {"commentId": "1124" ,"pageId": "breaking-news-story-01-18-2016", "userName": "Just Saying Thank You", "message": "I really enjoyed this youtube video!!"}
Sun May 22 21:51:24 UTC 2016 : Endpoint request URI: https://dynamodb.us-east-1.amazonaws.com/?Action=PutItem
Sun May 22 21:51:24 UTC 2016 : Endpoint request headers: {Authorization=
"TableName": "Comments",
"Item": {
"commentId": {
"S": "1124"
},
"pageId": {
"S": "breaking-news-story
},
"userName": {
"S": "Just Saying Thank You"
},
Feedback
English
© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

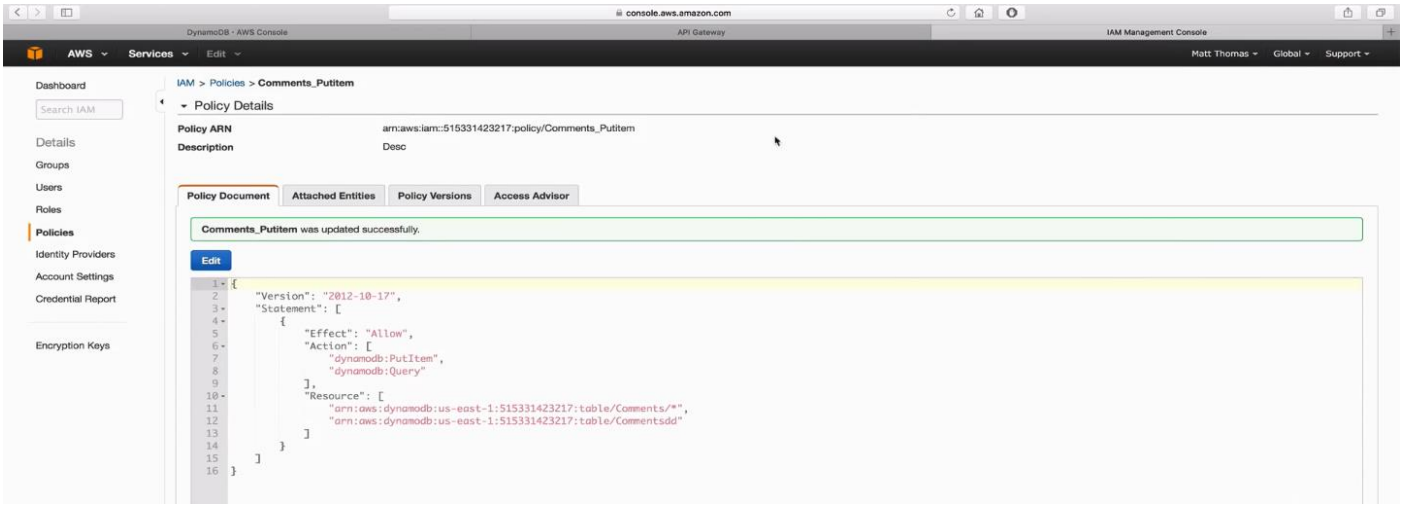

console.aws.amazon.com
DynamoDB - AWS Console
API Gateway
IAM Management Console
AWS
Services
Edit
Matt Thomas
N. Virginia
Support
DynamoDB
Dashboard
Tables
Reserved capacity
Create table
Actions
Filter by table name
Name
Comments
Stories
Comments
Close
Overview
Items
Metrics
Alarms
Capacity
Indexes
Triggers
Access control
Create item
Actions
Scan: [Table] Comments: commentId
Viewing 1 to 4 items
Scan
[Table] Comments: commentId
Add filter
Start search
commentId
message
pageId
userName
2
Total nonsense
back page
Daisy
1
I really enjoye...
front page
Walter
3
Awesome
middle page
Dennis
1124
I really enjoye...
breaking-new...
Just Saying T...


console.aws.amazon.com
DynamoDB - AWS Console
API Gateway
IAM Management Console
AWS
Services
Edit
Matt Thomas
Global
Support
Dashboard
Search IAM
Details
Groups
Users
Roles
Policies
Identity Providers
Account Settings
Credential Report
Encryption Keys
Create Policy
Policy Actions
Filter:
Policy Type
Filter
Showing 202 results
Policy Name
Attached Entities
Creation Time
Edited Time
Comments_Putitem
2
2016-05-21 11:38 UTC+0100
2016-05-22 15:24 UTC+0100
AdministratorAccess
1
2015-02-06 18:39 UTC+0100
2015-02-06 18:39 UTC+0100
AmazonAPIGatewayAdministrator
0
2015-07-09 18:34 UTC+0100
2015-07-09 18:34 UTC+0100
AmazonAPIGatewayInvokeFullAccess
0
2015-07-09 18:36 UTC+0100
2015-07-09 18:36 UTC+0100
AmazonAPIGatewayPushToCloudWatchLogs
0
2015-11-11 23:41 UTC+0100
2015-11-11 23:41 UTC+0100
AmazonAppStreamFullAccess
0
2015-02-06 18:40 UTC+0100
2015-02-06 18:40 UTC+0100
AmazonAppStreamReadOnlyAccess
0
2015-02-06 18:40 UTC+0100
2015-02-06 18:40 UTC+0100
AmazonCognitoDeveloperAuthenticatedIdentities
0
2015-03-24 17:22 UTC+0100
2015-03-24 17:22 UTC+0100
AmazonCognitoPowerUser
0
2015-03-24 17:14 UTC+0100
2015-03-24 17:14 UTC+0100
AmazonCognitoReadOnly
0
2015-03-24 17:06 UTC+0100
2015-03-24 17:06 UTC+0100

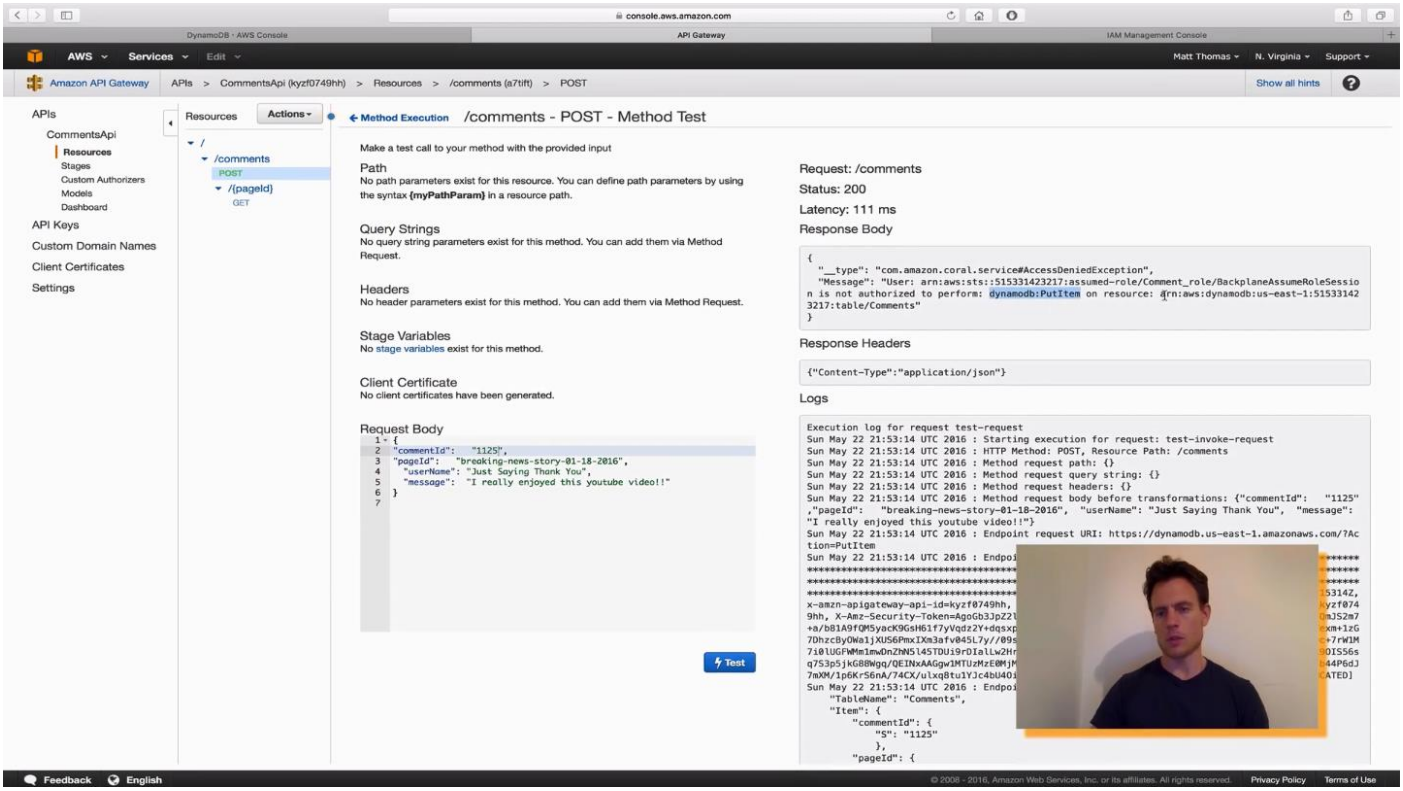


Let us change the above

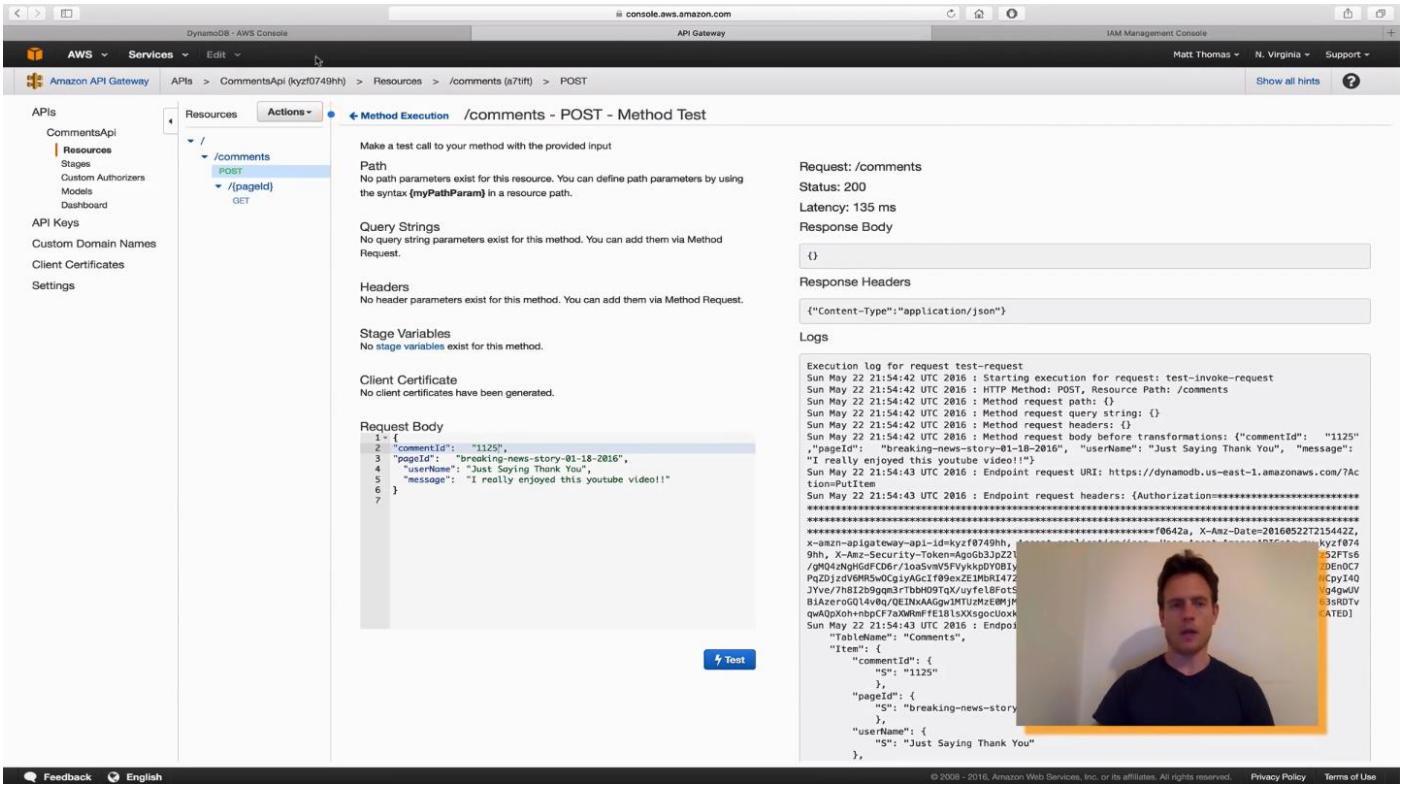




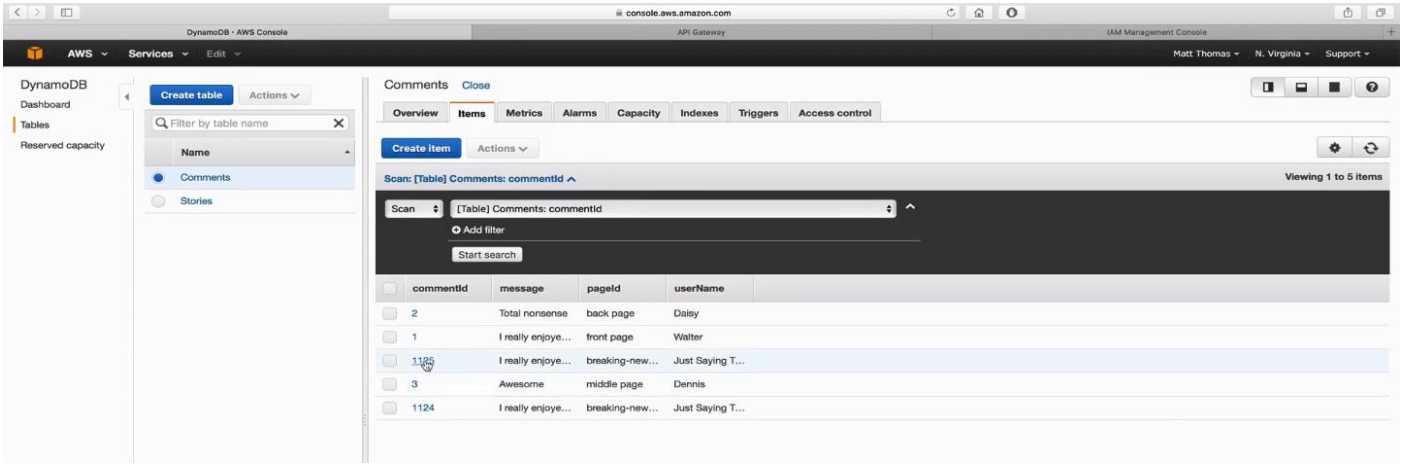
Now we have removed this users' access to the Comments table for PutItem and Query. Let us now see what we will get when we make the POST call again as below



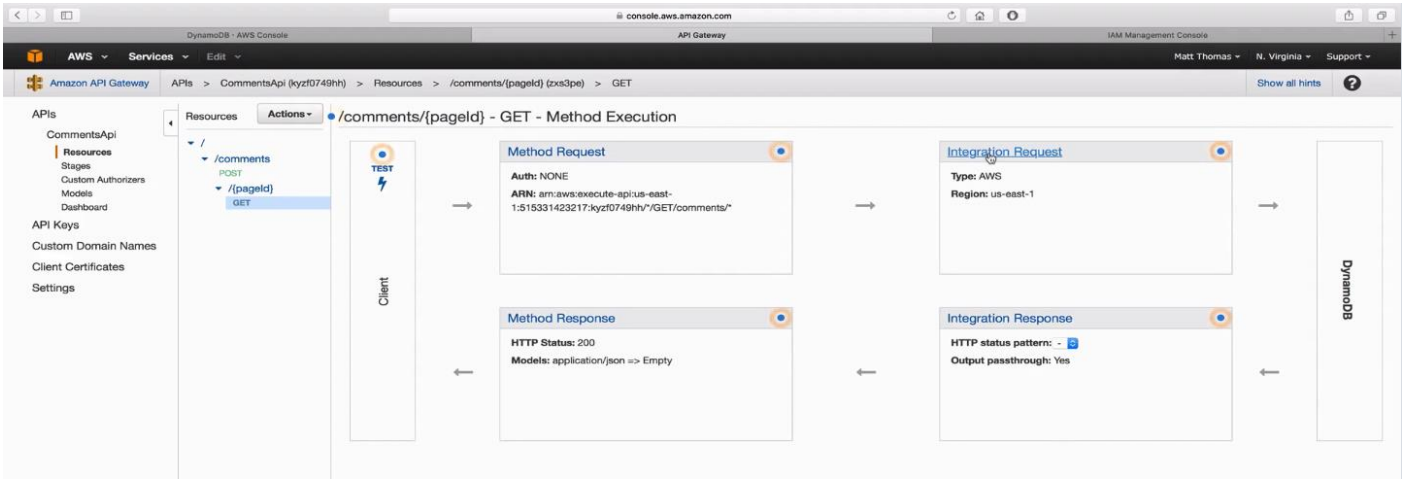
We now see a message saying the Role we are using does not have access to work on the Comments DynamoDB table



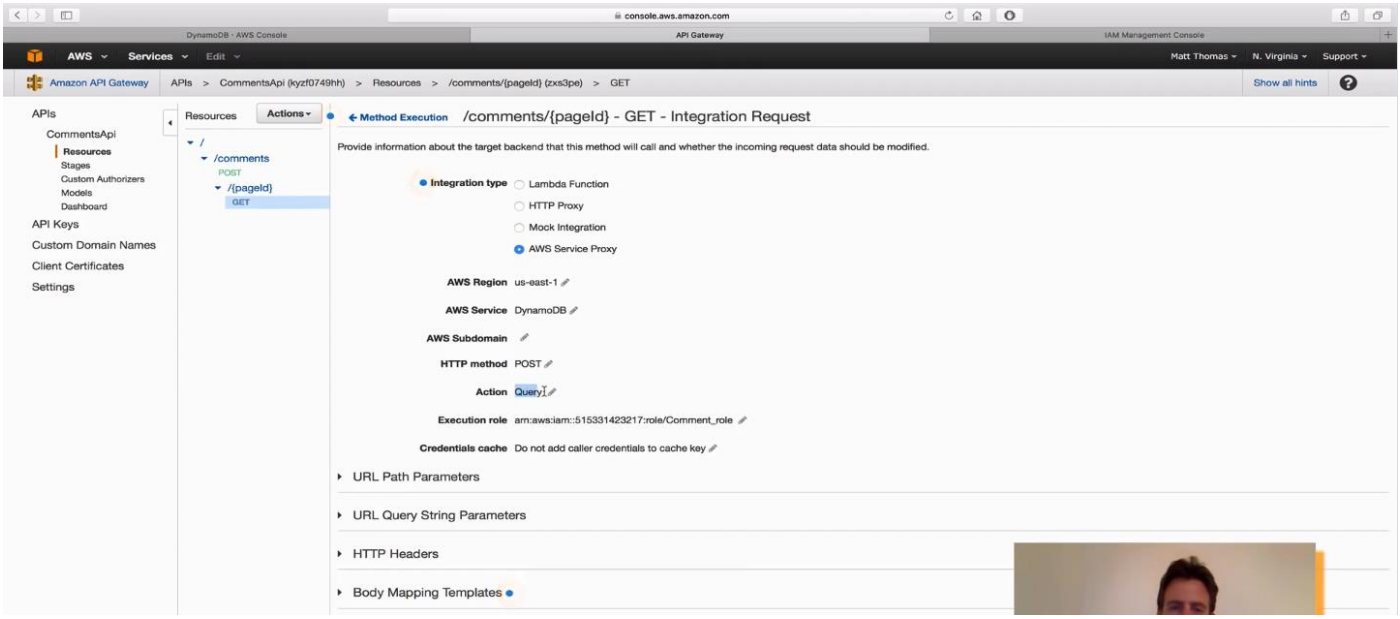
We can reverse the policy and test again



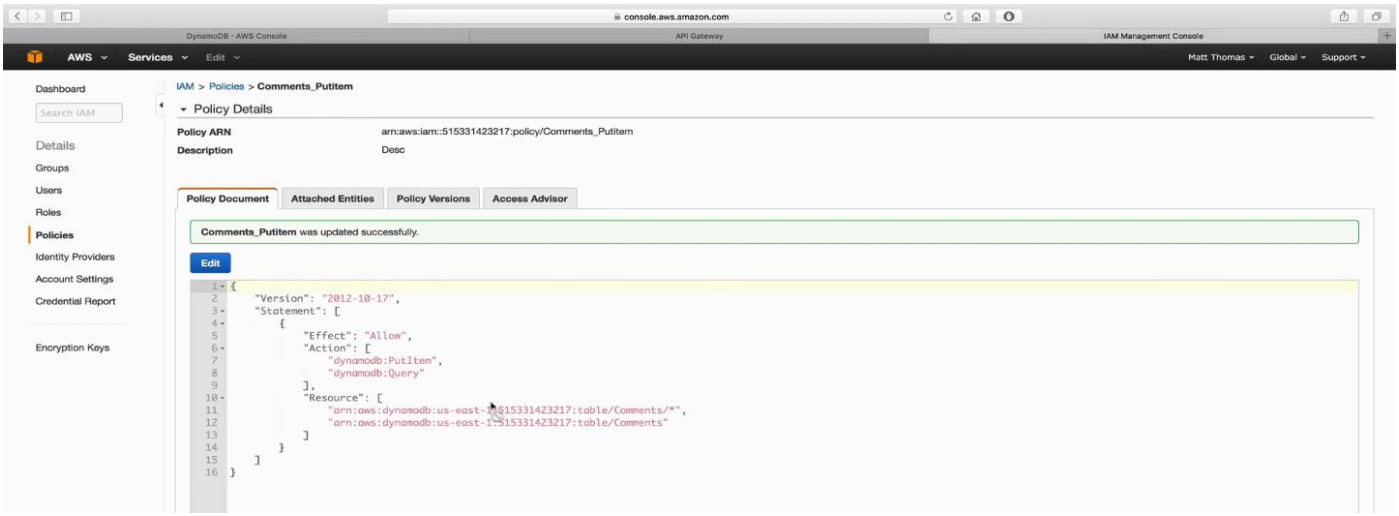
We can now see the new record showing up in DynamoDB.

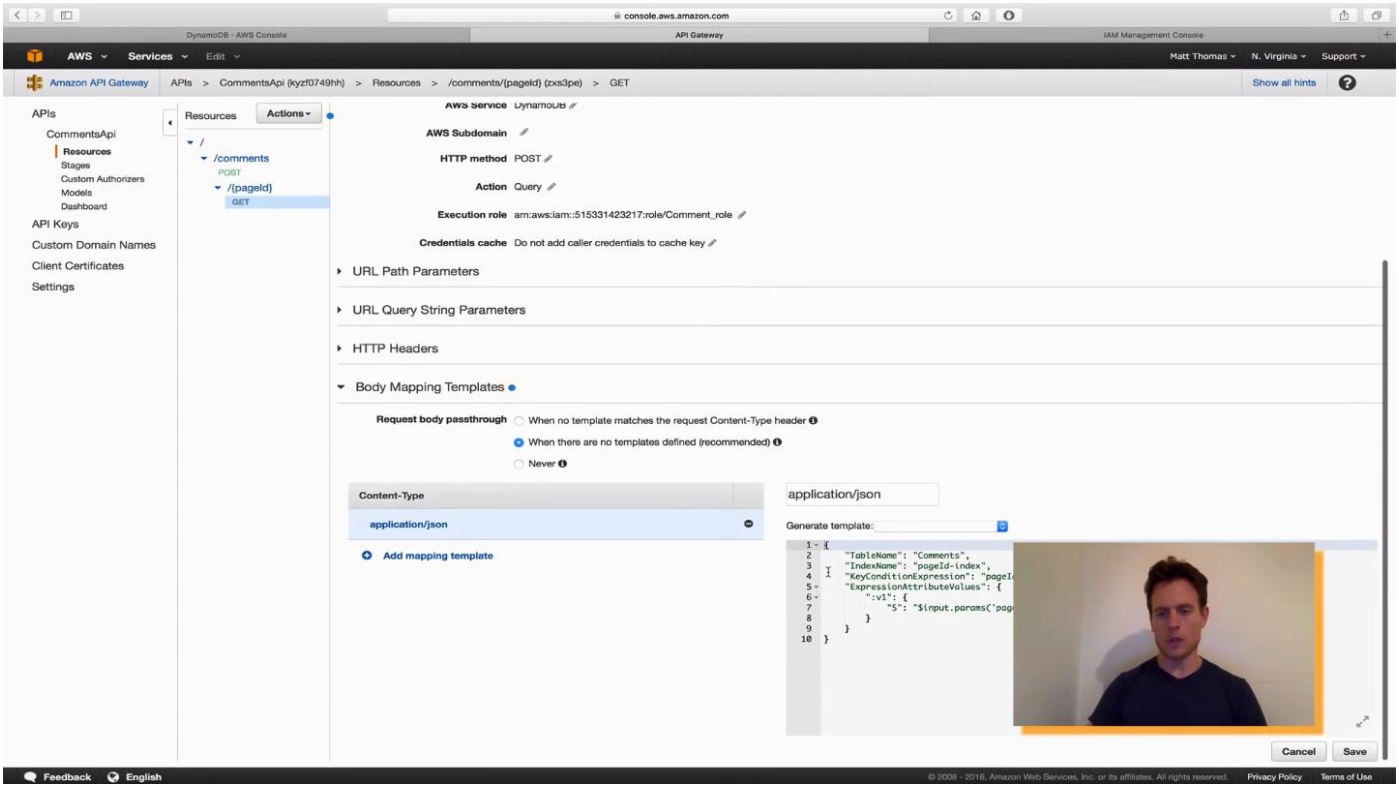


Let us now test the GET endpoint with the ***pageid*** as a route parameter.

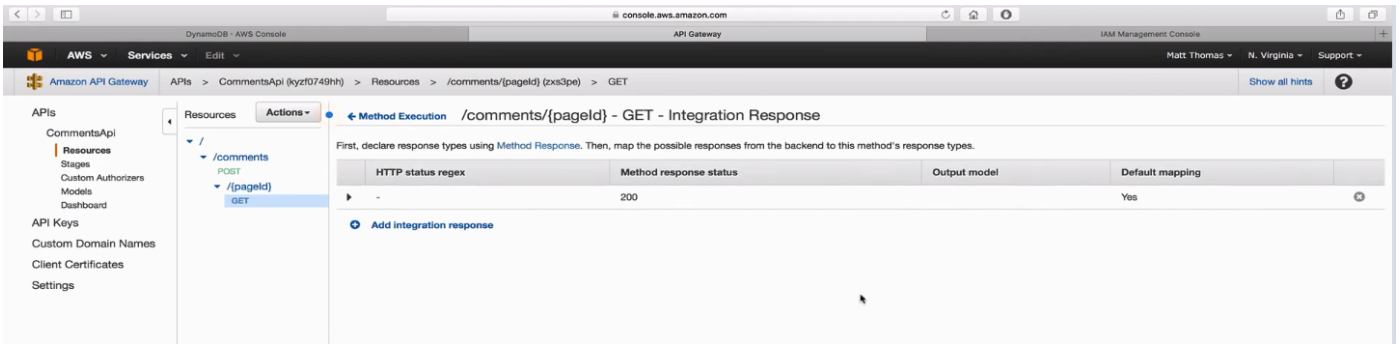


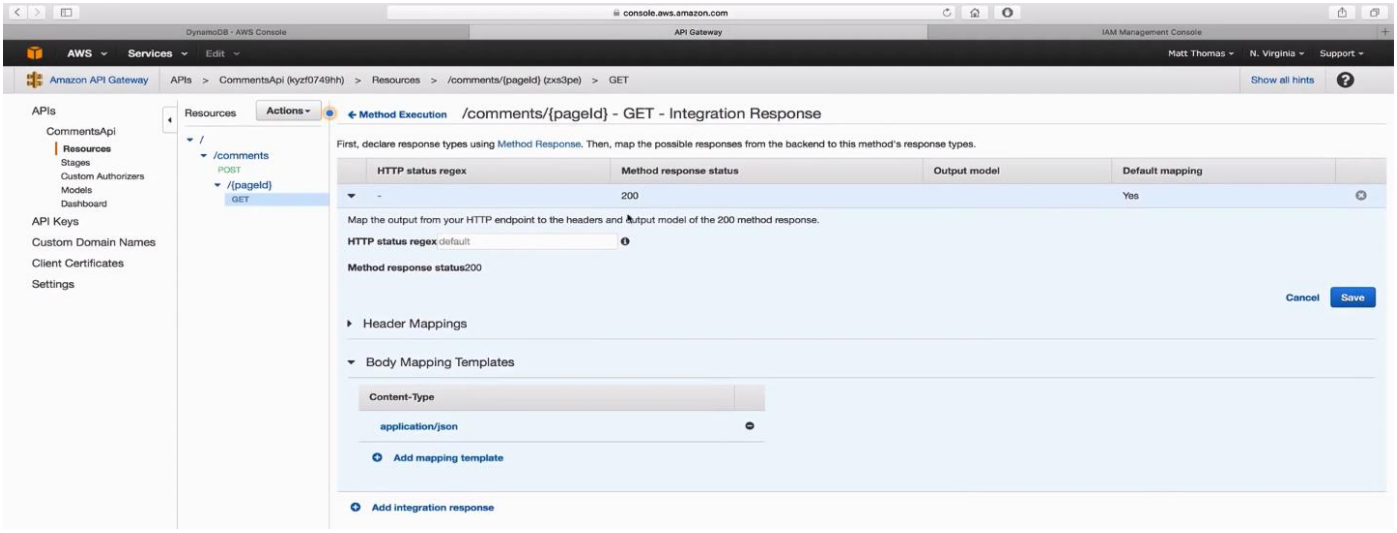
Note that if you are calling DynamoDB directly, you have to make a POST call even if you are actually doing a GET call from the browser. The difference is that when making a GET call, you will have to make a POST call but with the Query action instead of the ***PutItem*** action. We are using the same execution Role that was used for the POST request earlier



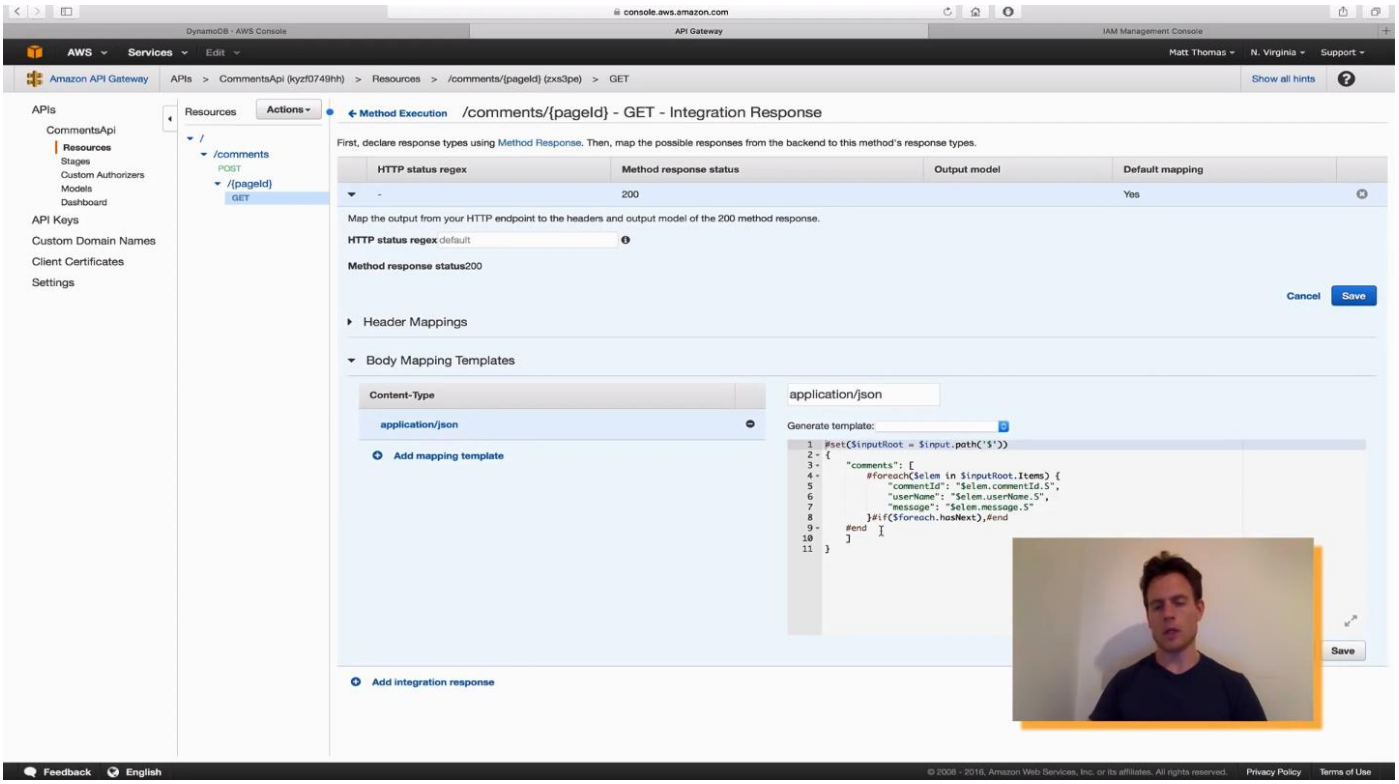


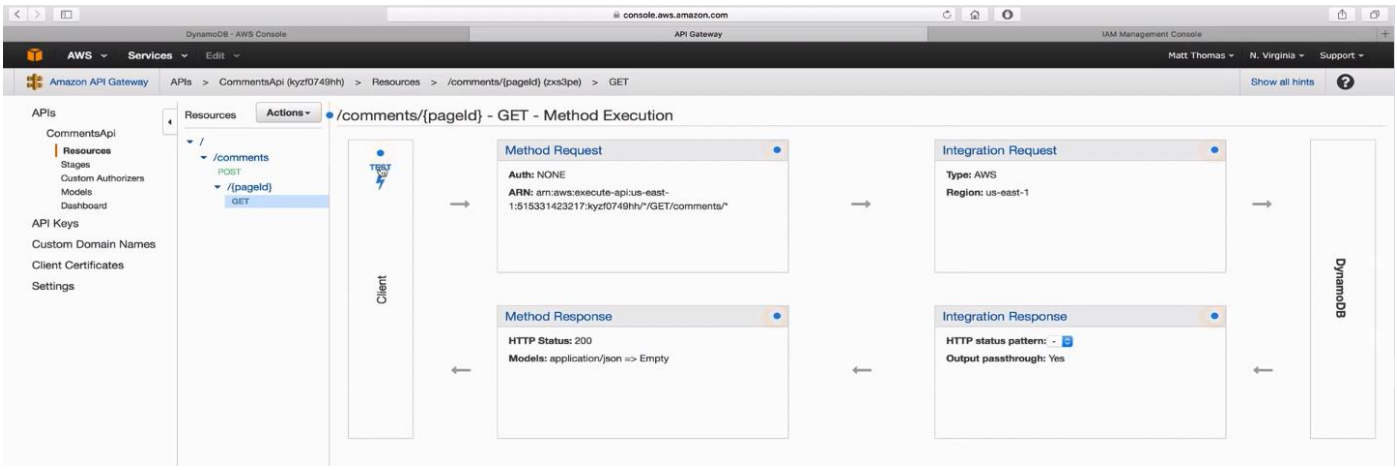
We are going to be pulling the page-id parameter from the URL request parameters and then put it in the ***params*** object that we will send in the request to DynamoDB



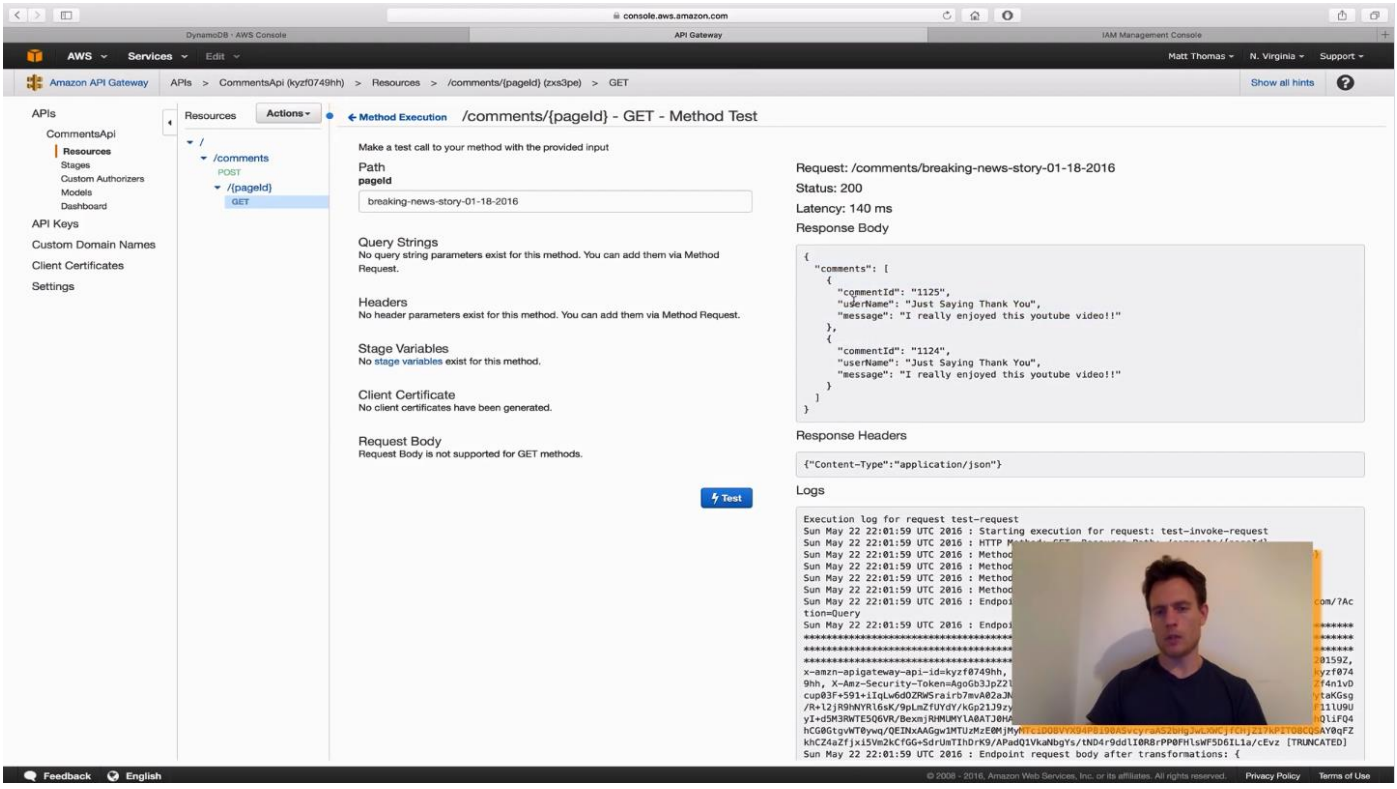


For a 200 response, we can then map back the DynamoDB response before sending it to the browser

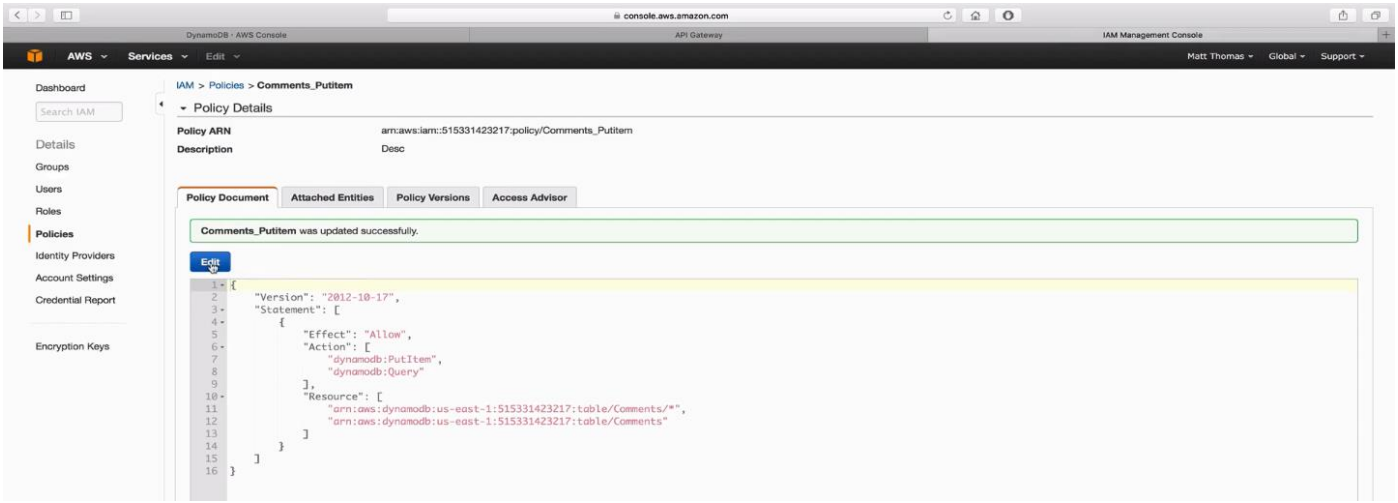


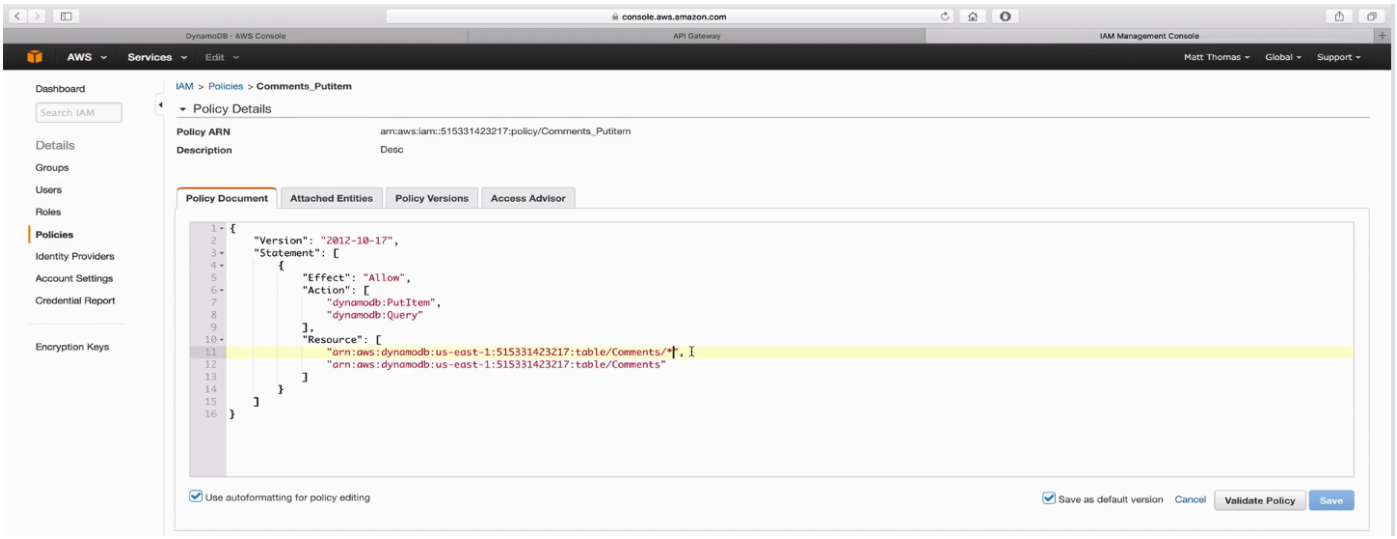


Let us test this endpoint now

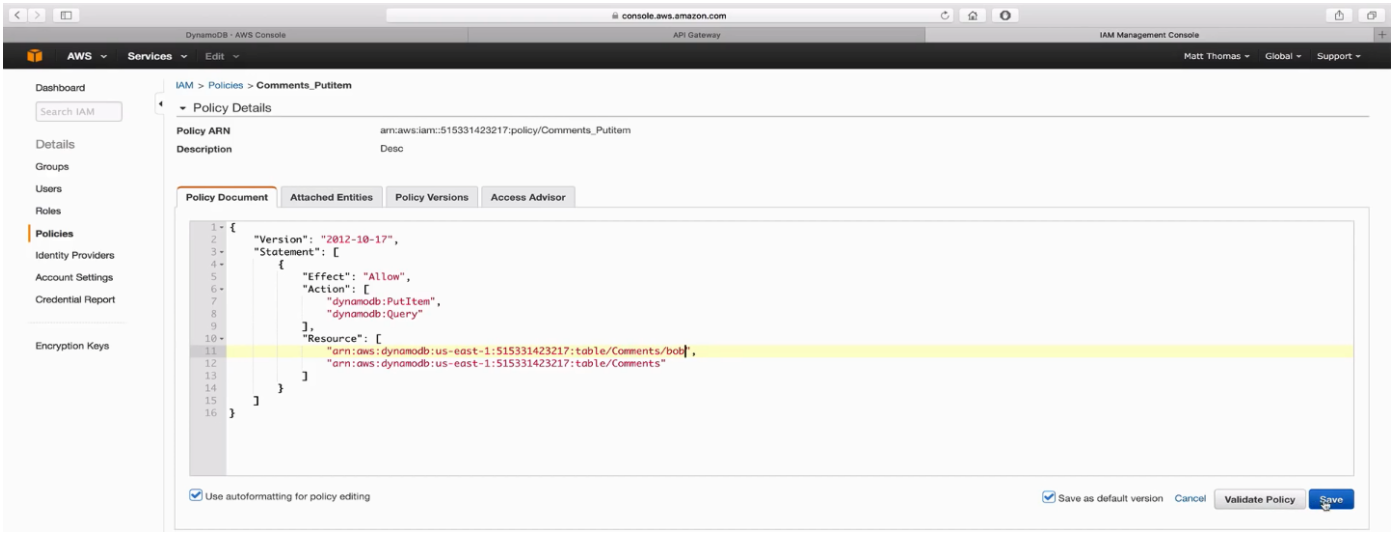


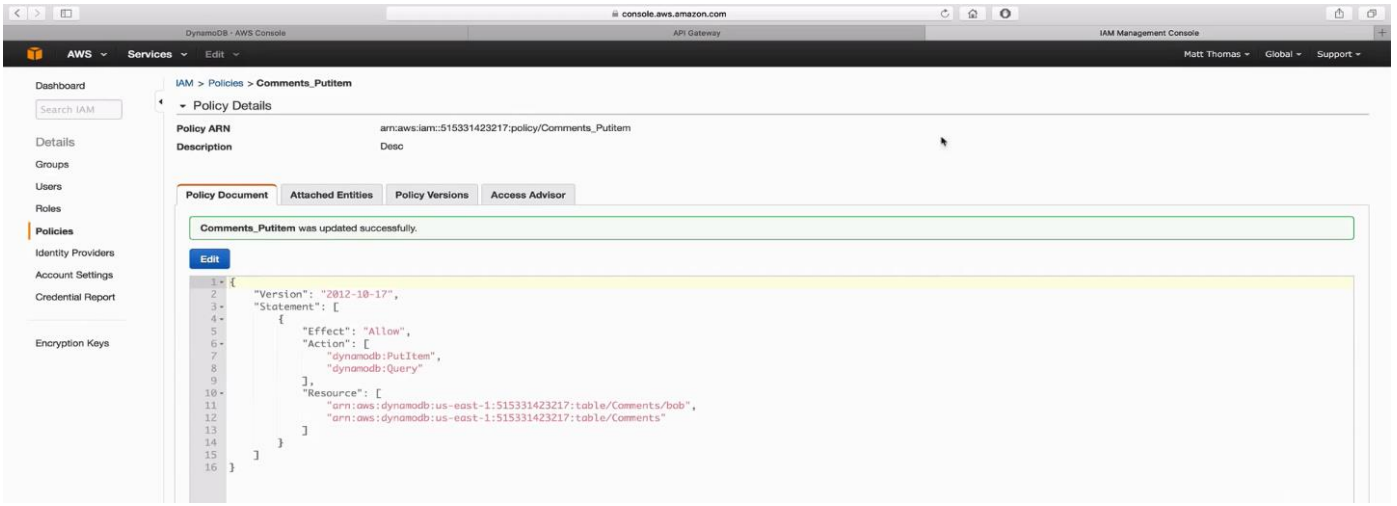
We then get the correct response with the objects as above

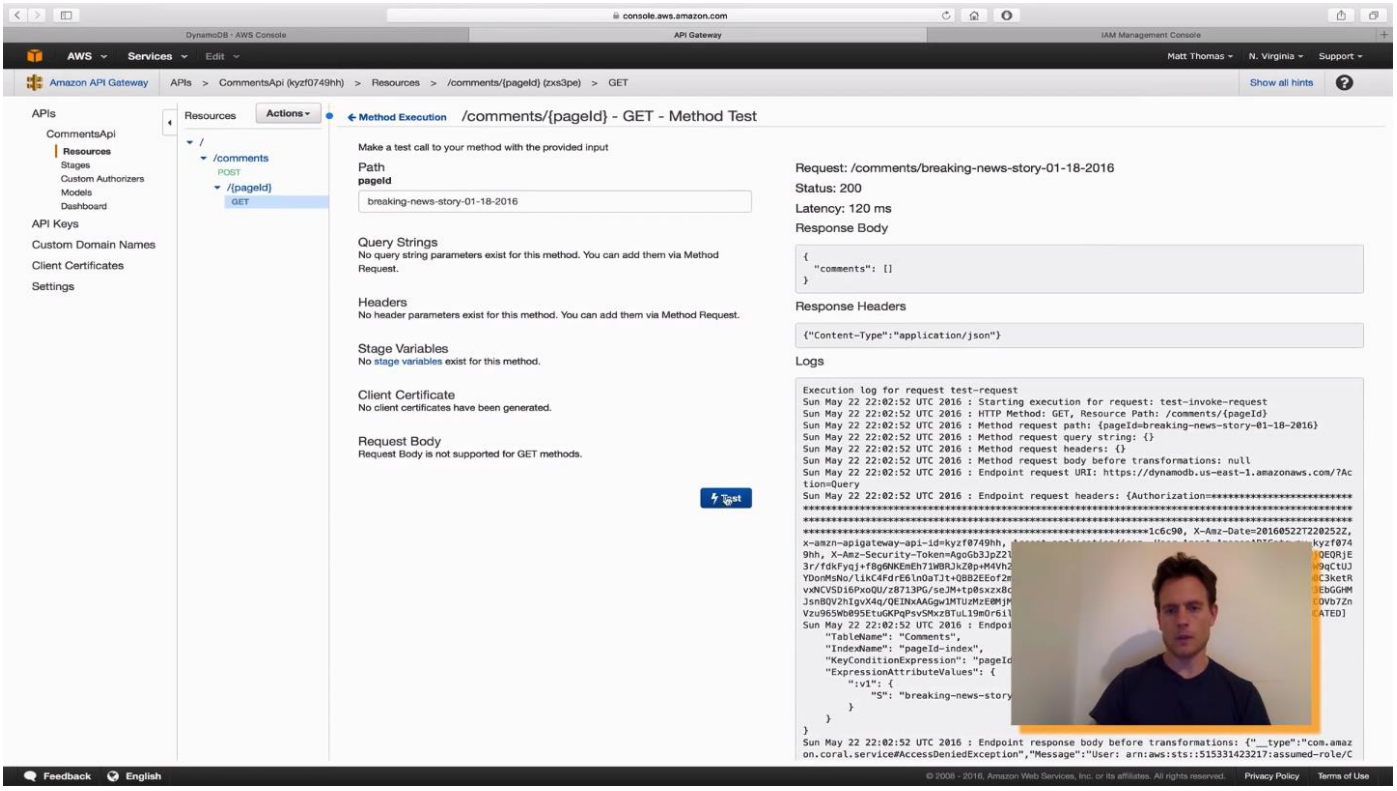


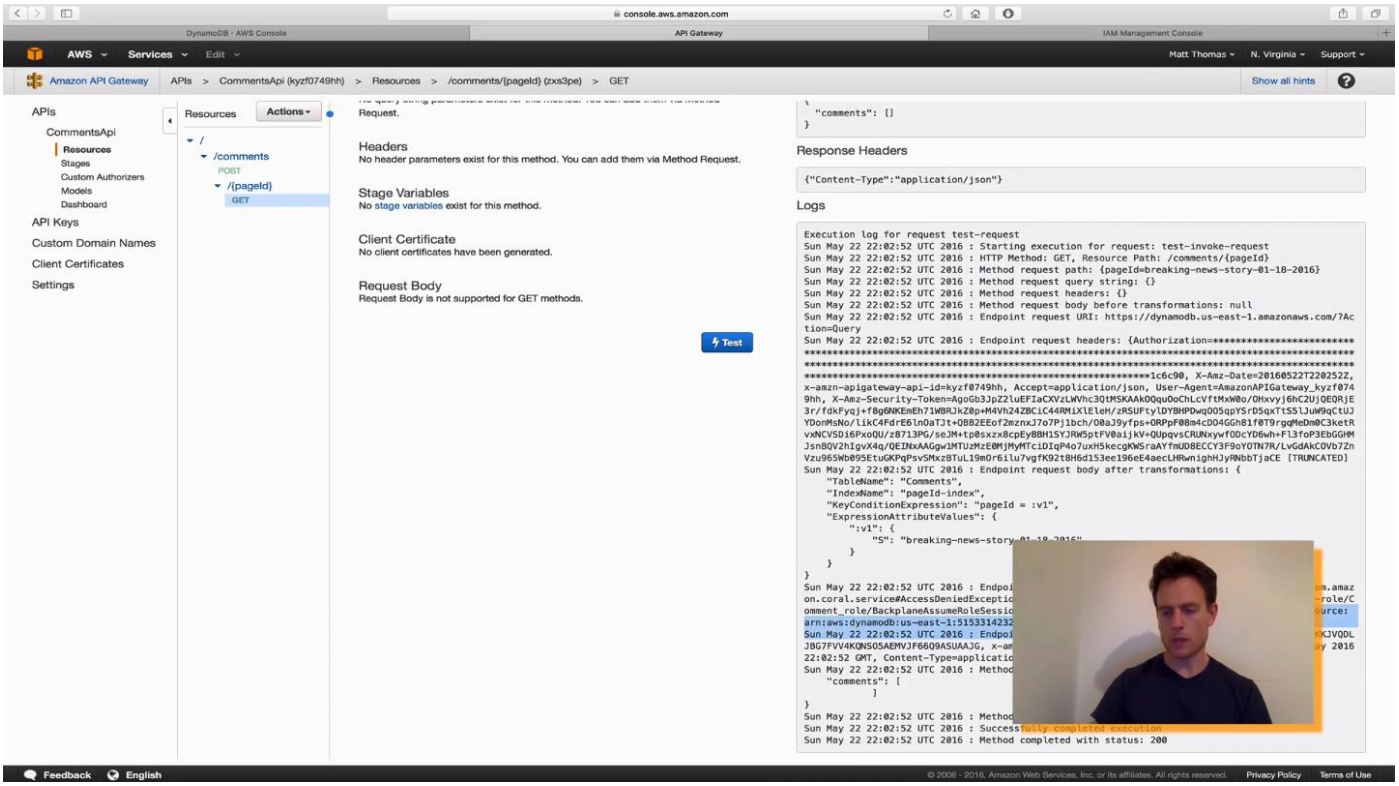


Let's edit again

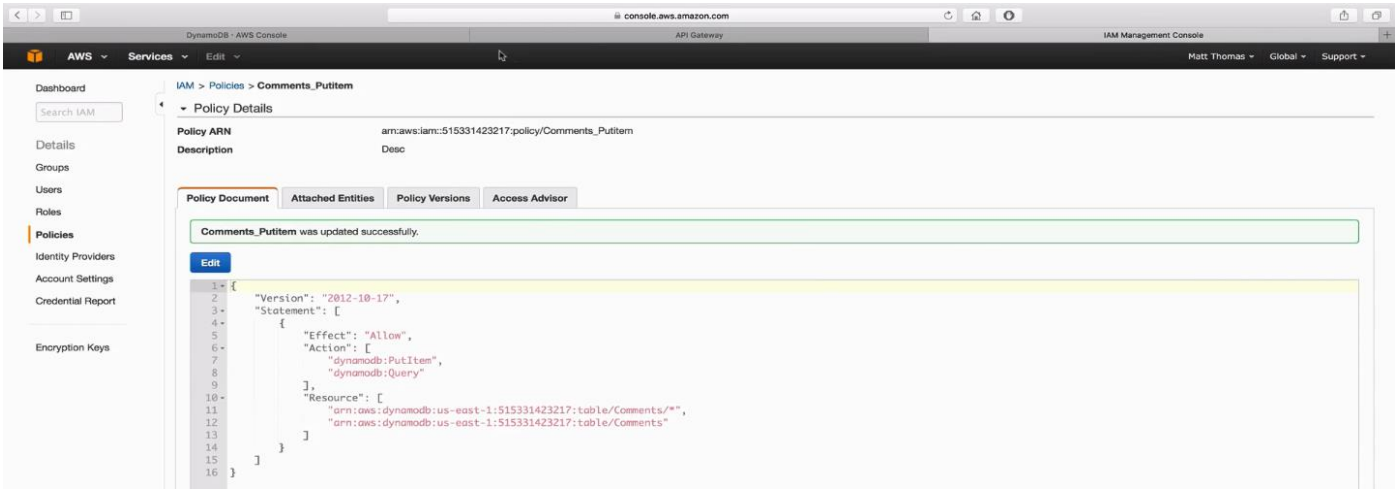




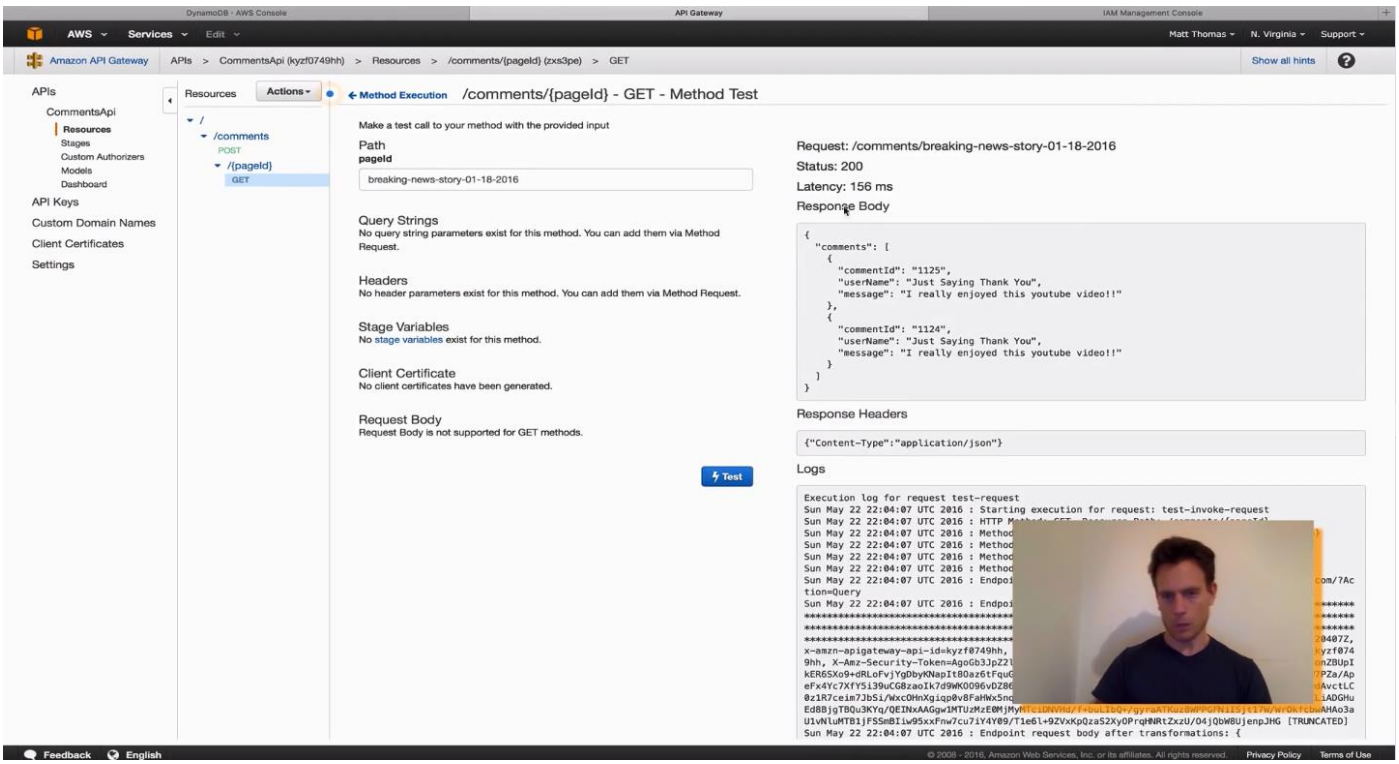




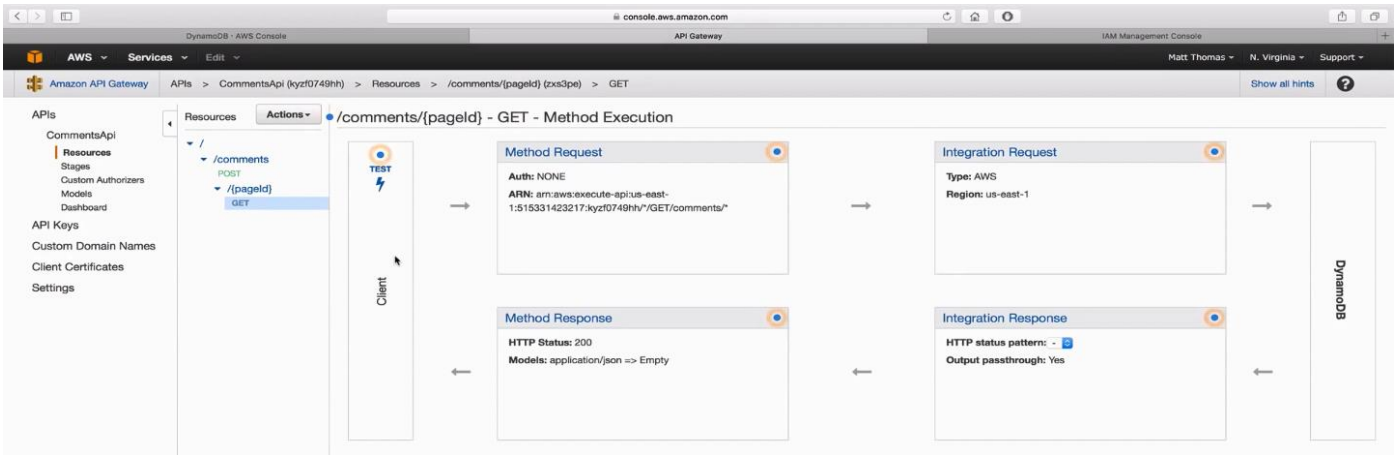
We again get an empty array returned along with a message that says that the role being used is not authorized to perform the Query on the Comments table. This shows how important it is to grant the correct permisiions on our database resources.



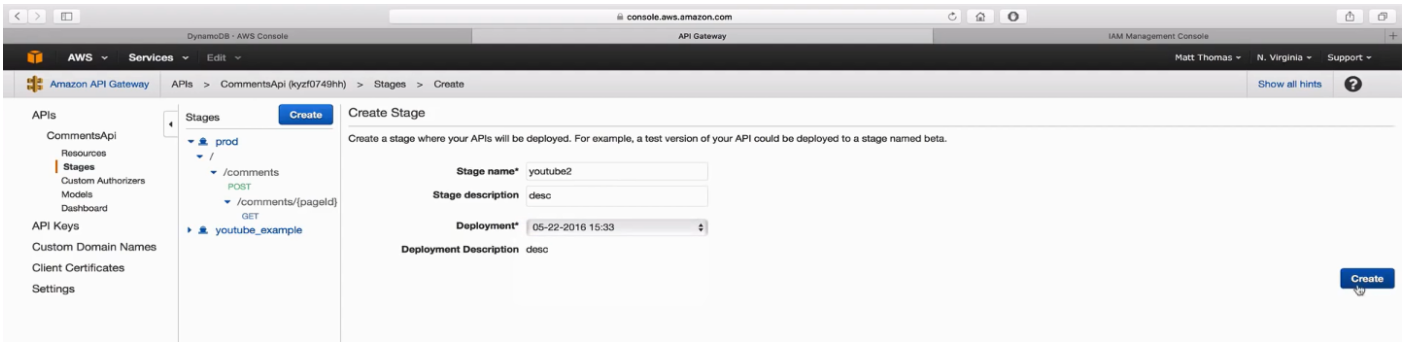
Correct the edit again



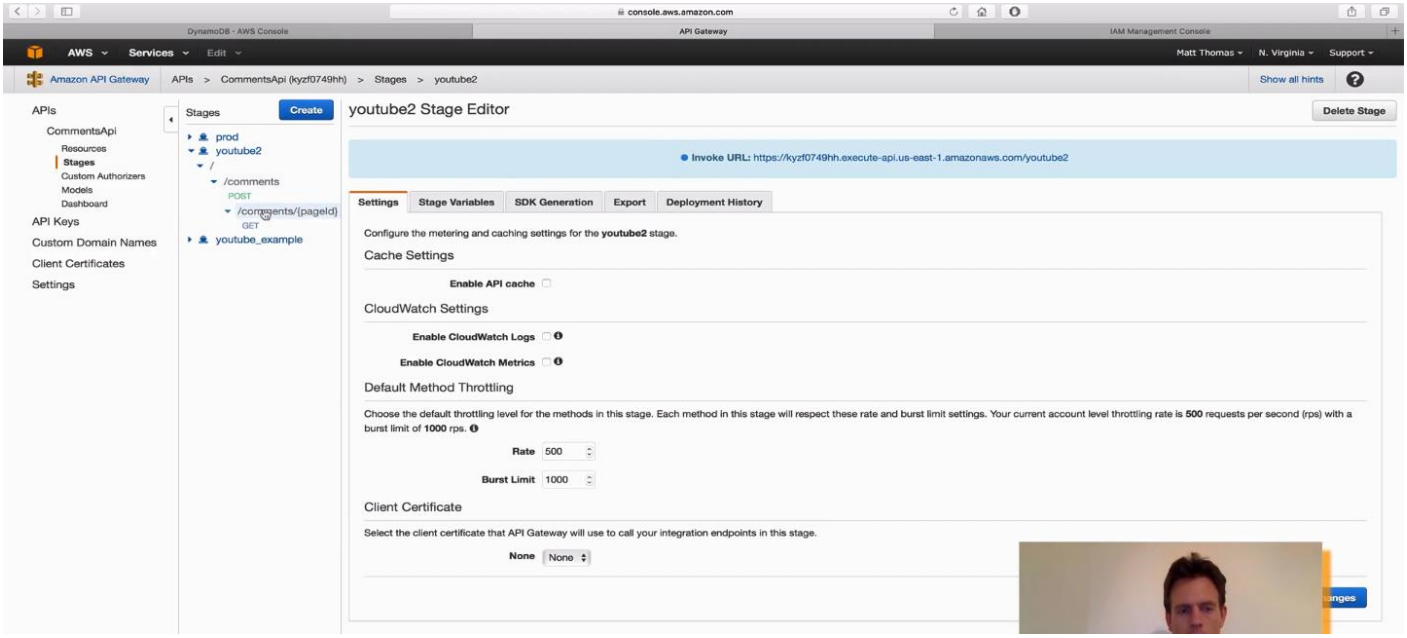
We are now able to query again with the correct permission.



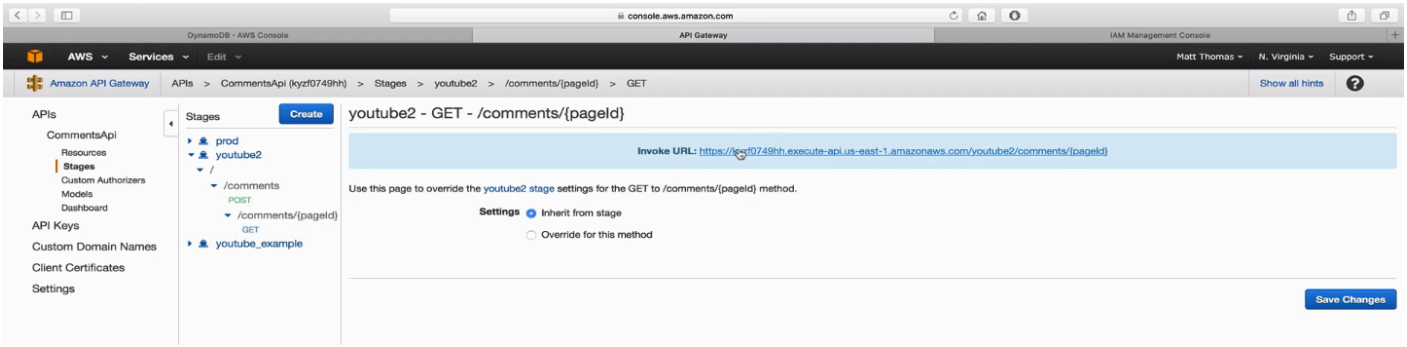
Let us now present our API to the world in a proper manner, we can use Stages as below



Stages allow us to create different versions of our APIs for things like dev, prod, staging, test, etc.

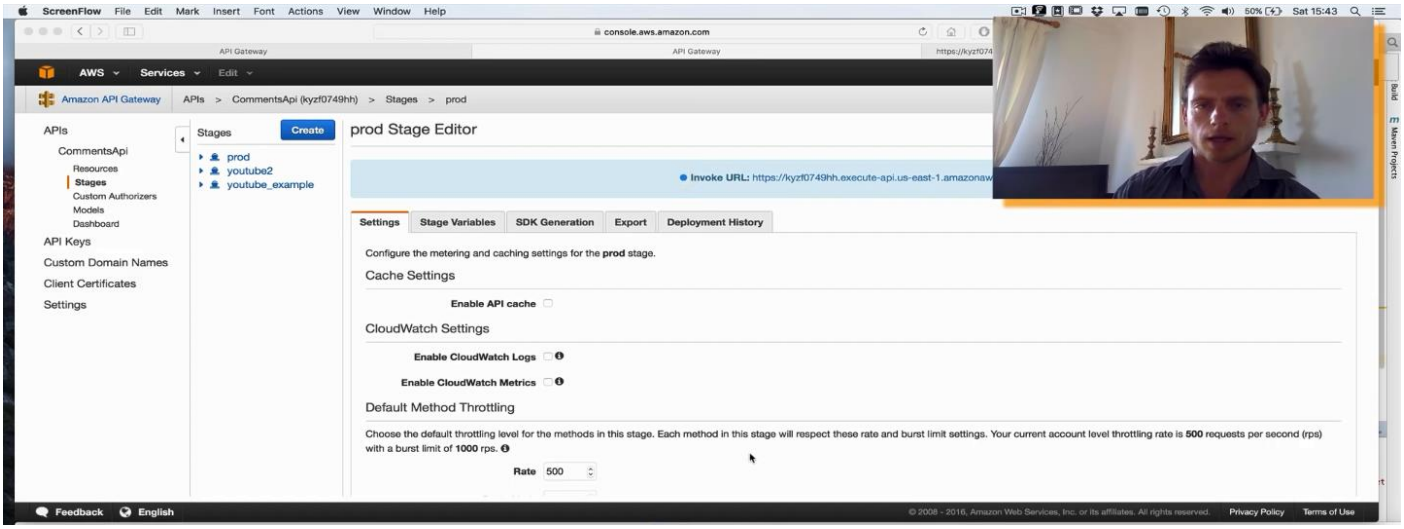


The youtube2 stage now has an actual copy of the former stage we worked in but with the ***youtube2*** stage name. we also have the public URL for this API.

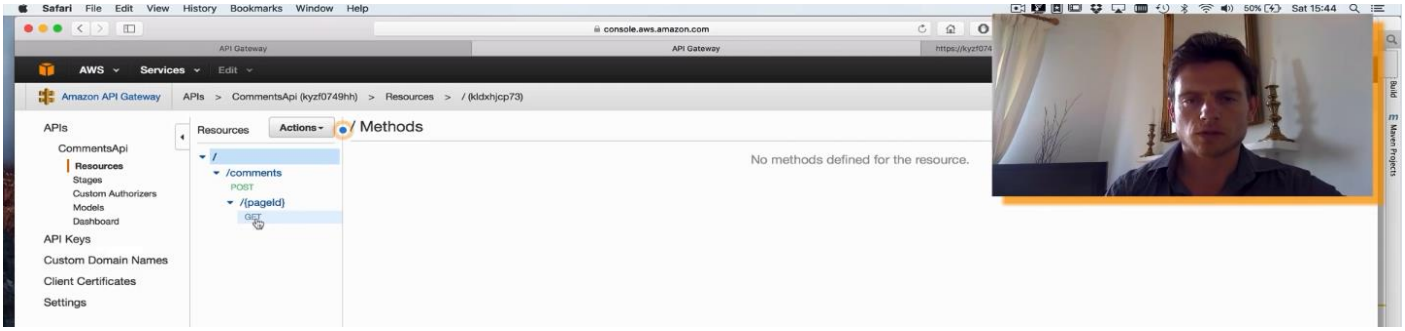


We can query this GET call from our browser and get response as above

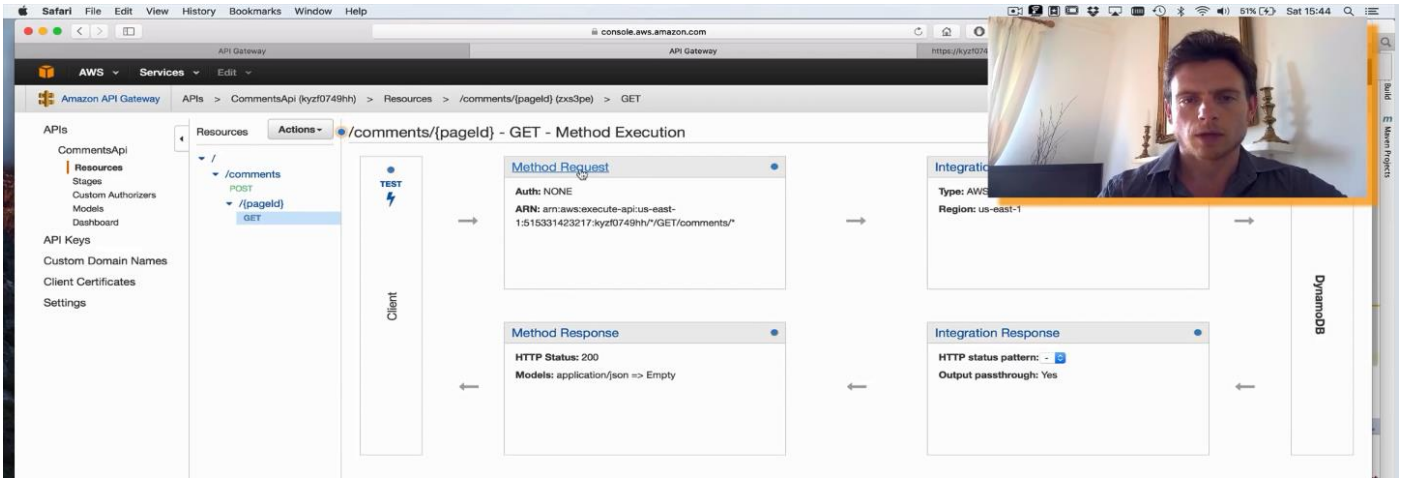
Securing Your Public API Gateway Endpoints with API Key

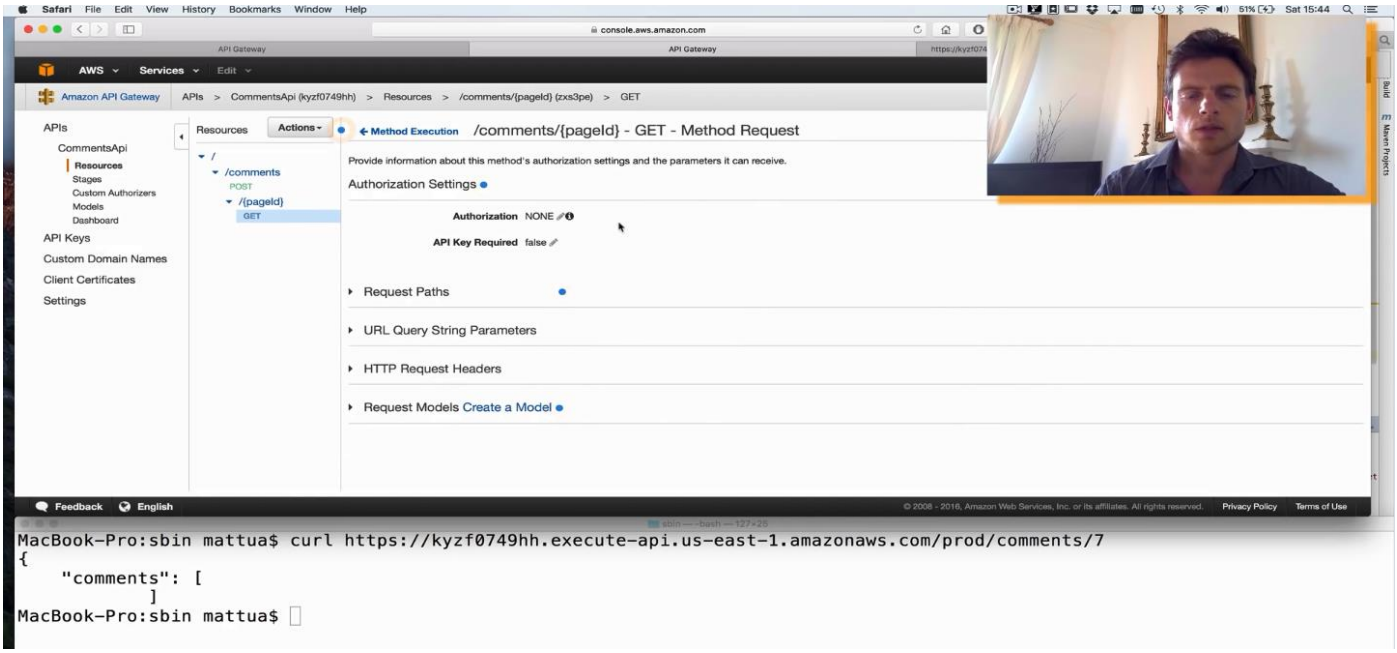


There are 2 ways to protect your public API Gateway endpoints by restricting who can call them, ***use a secret API Key*** or ***use a dedicated IAM Role***.

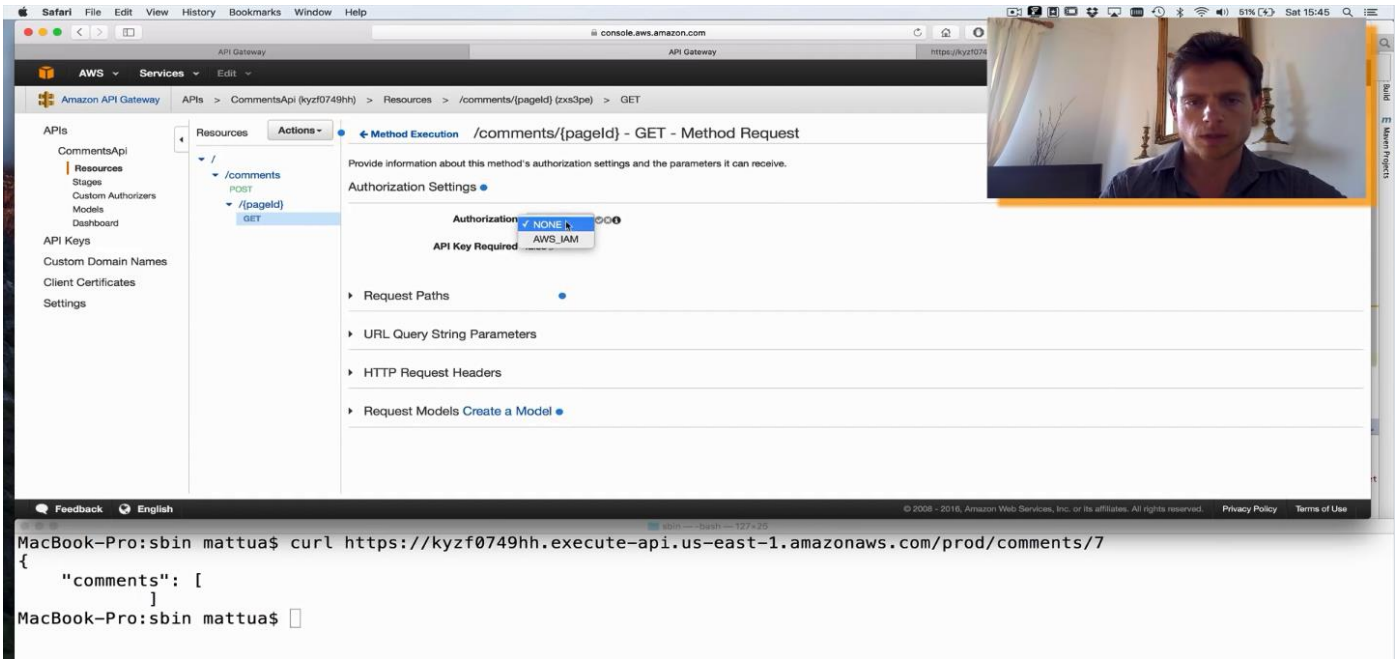


Let us now restrict access to who can make the GET method calls to our endpoint, we will use a secret API Key

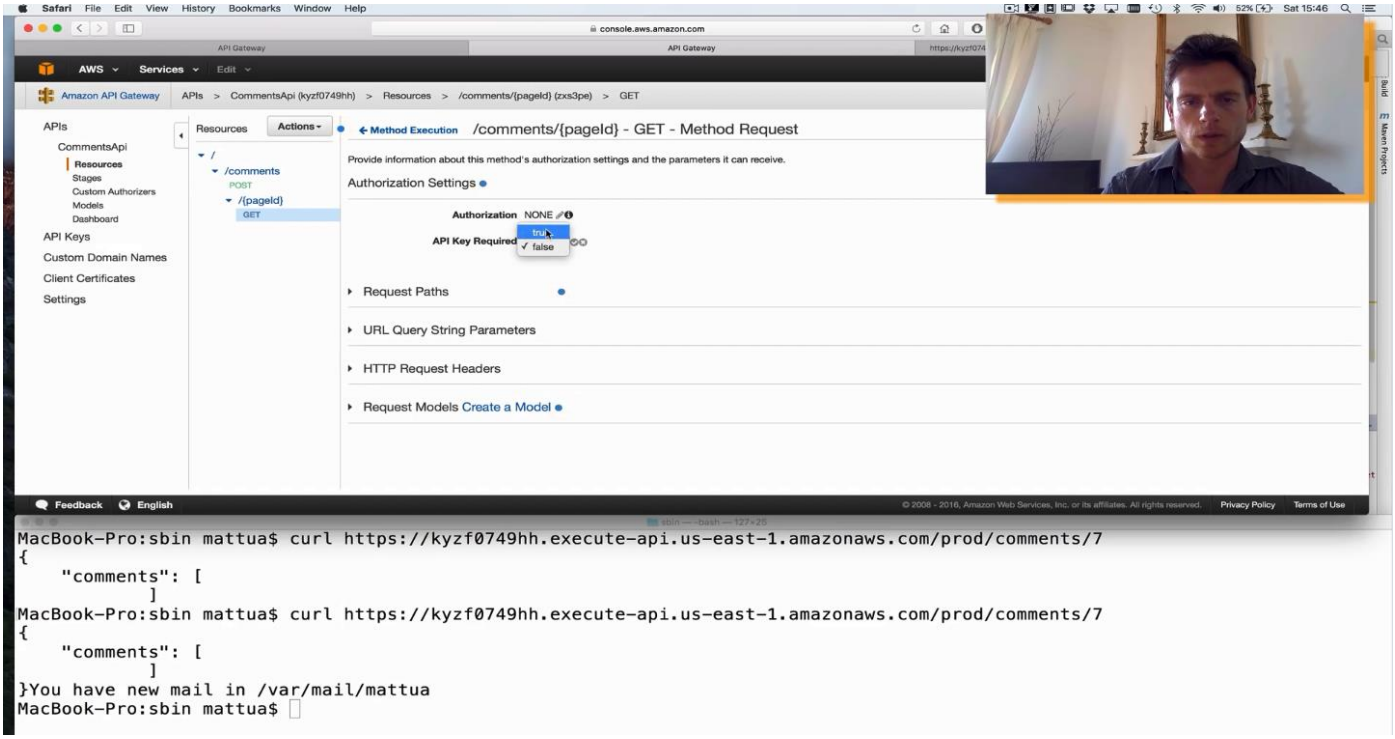




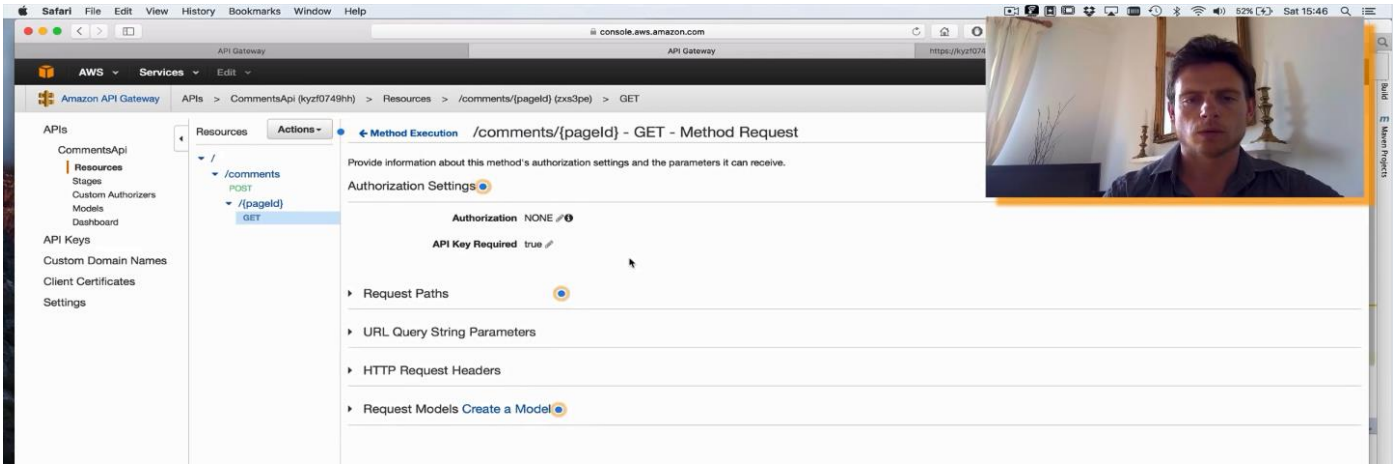
There are 2 Authorization Settings that we can tune, the ***Authorization header setting*** and the ***API Key Required setting***.

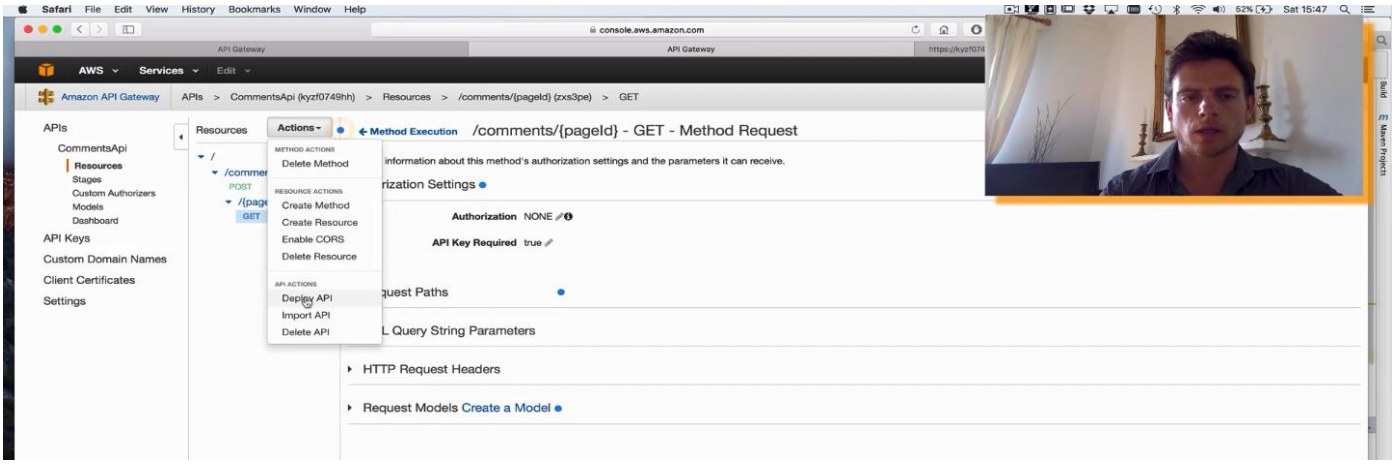


The Authorization header setting can be used with the above options

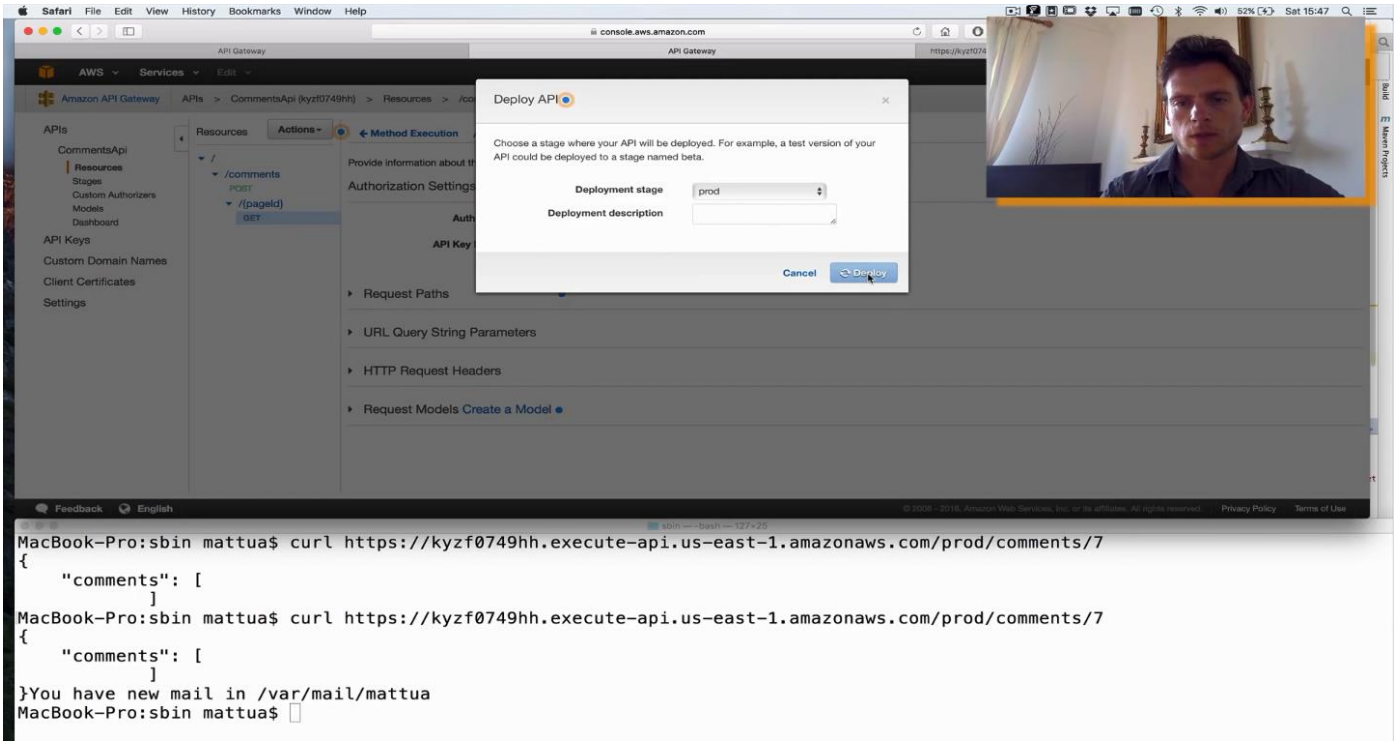


We can now enable the API Key Required setting so that this endpoint should only work for requests with the correct API Key.

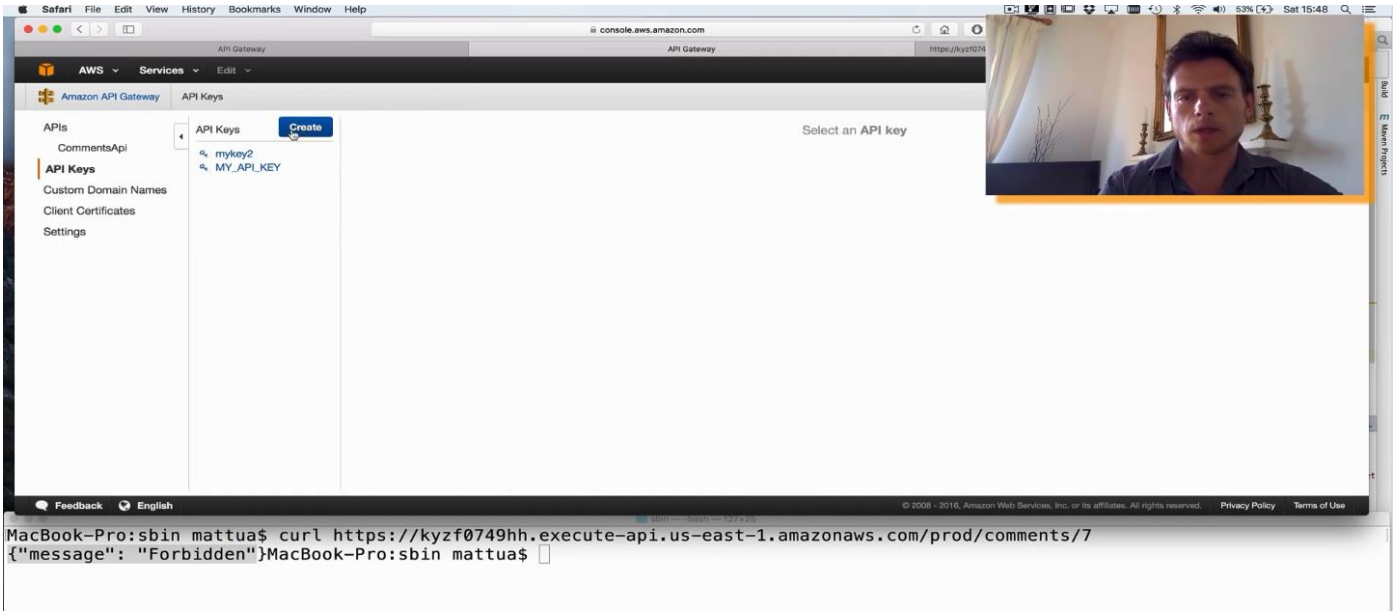




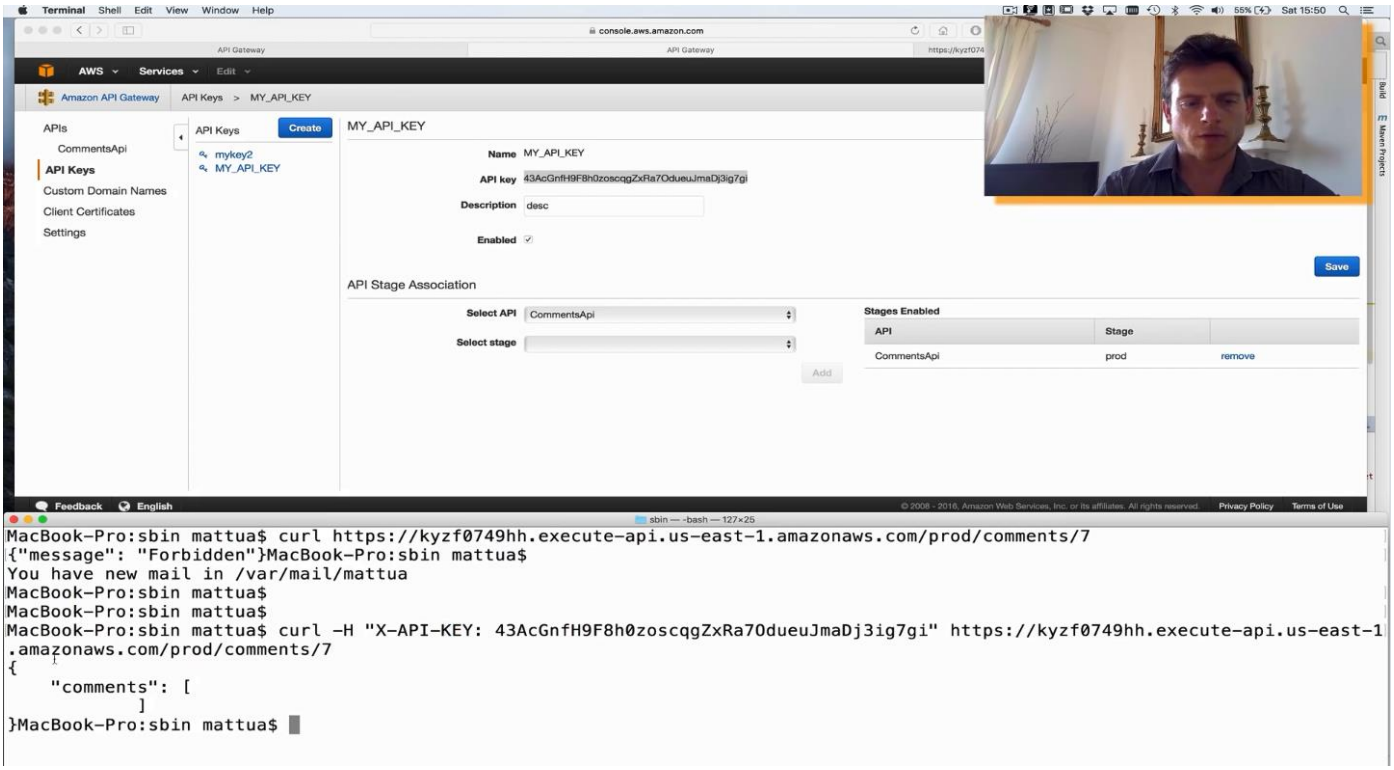
You then need to deploy the new version of your API to make this change effective



When we now try to access this API, we get a Forbidden message as above because we did not pass on an API Key.



You can create an API Key using the API Keys tab on the left, an API Key is just any long string as shown below



The API Key is generated for us by the API Gateway automatically when we click ***Create*** button, we then need to tell it what API this is for and what stage to use this key for. This API Key will have to be specified as a header parameter in the HTTP request. This means that we can not test this from the browser just like a hacker would but we will use the ***curl*** tool and provide the XAPIKEY_HEADER as above.