

## MINI-PROJECT-1

NAME :-

Upload files into a S3 Bucket using a REST API via API Gateway.

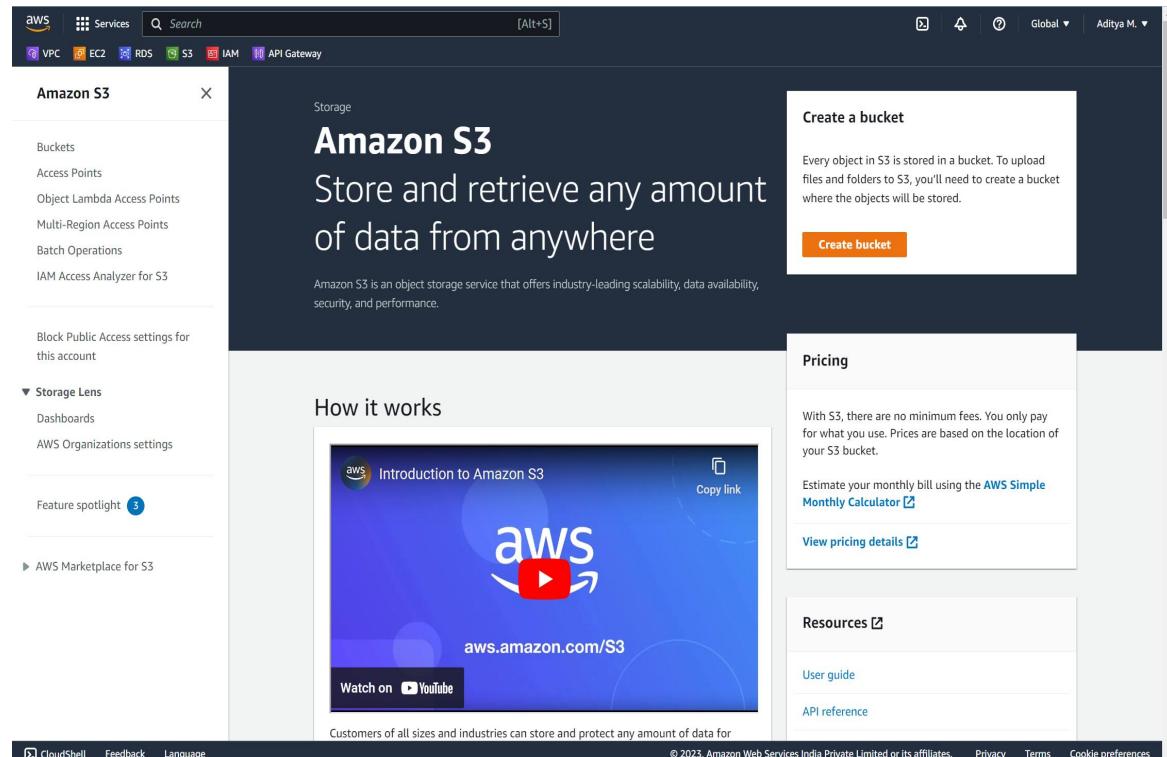
USE CASE :-

There are times when we need to accept file uploads on our websites. For this we setup the following rest API to upload files in S3 bucket.

- 1) S3 Bucket :- Create a S3 bucket to hold our uploaded files.
- 2) IAM :- Create an IAM role and attach policy to it.
- 3) API Gateway :- Create a REST API layer to accept PUT requests for file uploads.

STEPS TO FOLLOW:

- 1) Open S3 service. Click on create bucket. Name the bucket and keep the rest of the settings as it is. Only changes you have to make is enable versioning and create bucket.  
(Note:- Bucket name must be globally unique.)



# ADITYA MAWALKAR

Create bucket [Info](#)  
Buckets are containers for data stored in S3. [Learn more](#)

**General configuration**

Bucket name  
aditya-mini-project-1  
Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

AWS Region  
US East (N. Virginia) us-east-1

Copy settings from existing bucket - *optional*  
Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

**Object Ownership** [Info](#)  
Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

**ACLs disabled (recommended)**  
All objects in this bucket are owned by this account.  
Access to this bucket and its objects is specified using only policies.

**ACLs enabled**  
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership  
Bucket owner enforced

© 2023, Amazon Web Services India Private Limited or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

**Block all public access**  
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

**Block public access to buckets and objects granted through new access control lists (ACLS)**  
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

**Block public access to buckets and objects granted through any access control lists (ACLS)**  
S3 will ignore all ACLs that grant public access to buckets and objects.

**Block public access to buckets and objects granted through new public bucket or access point policies**  
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

**Block public and cross-account access to buckets and objects through any public bucket or access point policies**  
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

**Bucket Versioning**  
Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

Bucket Versioning  
 Disable  Enable

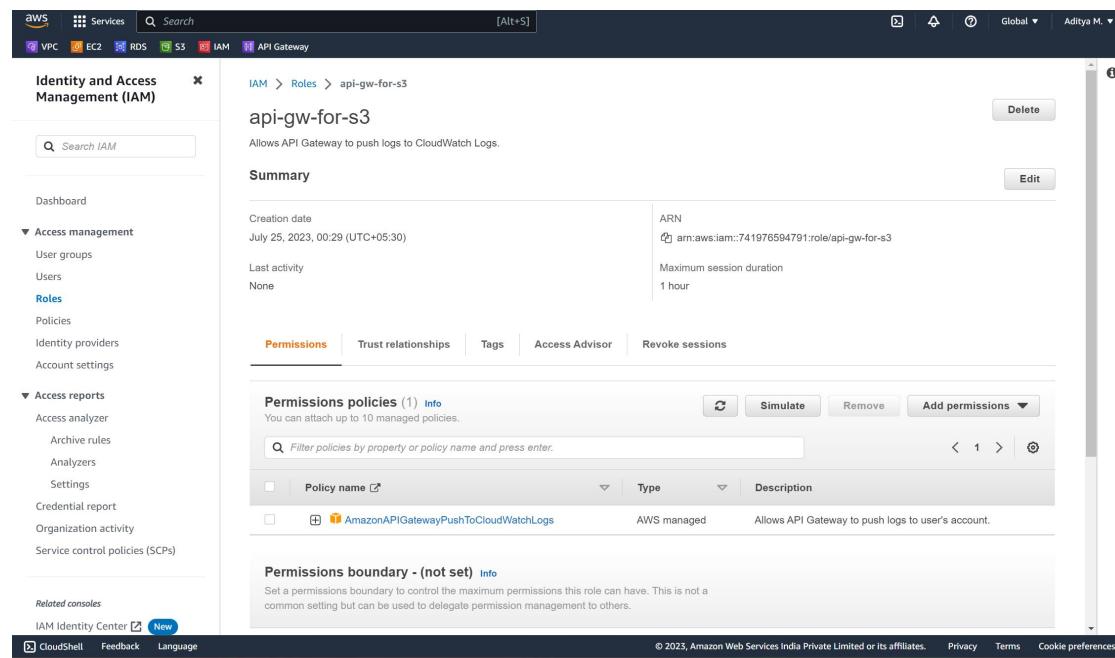
**Tags (0) - optional**  
You can use bucket tags to track storage costs and organize buckets. [Learn more](#)

No tags associated with this bucket.

© 2023, Amazon Web Services India Private Limited or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

- 2) In the next step, we have to create an IAM role and attach a policy to it.
- 3) Open the IAM service. Click on the create roles. Then select the API Gateway service from "use cases for other AWS services" and click next. Give a name and create the role.

- 4) Now role is created. Now we have to attach a polity to the role.
- 5) In the policy tab on the left hand side of IAM service click on create policy. Then choose the service S3.
- 6) Then select an action "PUTOBJECT" and in resources add "ARN".
- 7) In ARN tab put your bucket name and click on any in object name. Give a name to your policy and click on create.
- 8) After that attach the policy to the role which we have created previously.
- 9) In roles, select your created role and click on add permission and then select the policy which we have created recently and save. Now your IAM ROLE is created and working perfectly.  
(Note:- Copy the ARN from the roles. We need that ARN to setup API Gateway.)



# ADITYA MAWALKAR

The screenshot shows the AWS IAM Policies page. The left sidebar includes sections for Identity and Access Management (IAM), Access management, Access reports, and Related consoles. The main content area displays a table of 1105 policies, with columns for Policy name, Type, Used ..., and Description. Some policies listed include AdministratorAccess, PowerUserAccess, ReadOnlyAccess, AWSCloudFormationReadOnlyAccess, CloudFrontFullAccess, AWSCloudHSMFullAccess, AWSCloudHSMReadOnlyAccess, ResourceGroupsandTagEditorFullAccess, ResourceGroupsandTagEditorReadOnlyAccess, CloudFrontReadOnlyAccess, CloudSearchFullAccess, CloudSearchReadOnlyAccess, and CloudWatchFullAccess.

The screenshot shows the 'Create policy' wizard at Step 1: Specify permissions. It features a 'Policy editor' interface with tabs for Visual, JSON, and Actions. Under the S3 service, the 'Allow' actions section is expanded, showing options like PutObject, PutObjectLegalHold, PutObjectRetention, PutObjectAcl, PutObjectVersionAcl, PutObjectTagging, and PutObjectVersionTagging. The 'Resources' section is also visible. A note at the bottom states: '⚠ Specified object resource ARN for the PutObjectRetention and 32 more actions. Add Arn to restrict access.' The bottom navigation bar includes CloudShell, Feedback, Language, Privacy, Terms, and Cookie preferences.

# ADITYA MAWALKAR

The screenshot shows the AWS IAM Policy Editor interface for creating a new policy. The top navigation bar includes 'Services' (with 'VPC', 'EC2', 'RDS', 'S3', 'IAM', 'API Gateway'), 'Search' (with 'Alt+S'), and user information ('Global', 'Aditya M.', 'Aditya M.'). The main area is titled 'Specify permissions' with an 'Info' link. It says 'Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.' Below this, 'Step 1 Specify permissions' and 'Step 2 Review and create' are shown. The 'Policy editor' section for 'S3' has an 'Allow' button and 2 Actions: 'DeleteObject' (checked) and 'DeleteObjectVersion'. Under 'Actions allowed', there are sections for 'Write' (DeleteObject, DeleteObjectVersion), 'Tagging' (DeleteObjectTagging, DeleteObjectVersionTagging), and 'Resources' (specific object ARN). A note says 'Specified object resource ARN for the PutObjectRetention and 32 more actions.' An 'Add Arn' link is provided. The bottom right shows 'Switch to deny permissions' and 'Any'. At the bottom, there are links for 'CloudShell', 'Feedback', 'Language', and copyright information ('© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences').

This screenshot shows the 'Specify ARNs' dialog box overlaid on the main IAM policy editor. The dialog has tabs for 'Visual' (selected) and 'Text'. It contains sections for 'Resource bucket name' (checkbox for 'Any bucket name', value 'aditya-project-1'), 'Resource object name' (checkbox for 'Any object name', value '\*'), and 'ARN' (checkbox for 'Any resource', value 'arn:aws:s3:::aditya-project-1/\*'). There is also an 'Add ARNs' button. Below the dialog, the main policy editor shows a note about specified object resource ARNs and an 'Add Arn' link. At the bottom, there are buttons for 'Cancel' and 'Next', and status indicators ('Security: 0', 'Errors: 0', 'Warnings: 0', 'Suggestions: 2'). The footer links are identical to the first screenshot.

# ADITYA MAWALKAR

The screenshot shows the AWS IAM Roles page. The left sidebar is collapsed. The main content area shows the details for the 'api-gw-for-s3' role. The 'Summary' tab is selected, displaying creation date (July 25, 2023, 00:29 UTC+05:30), last activity (None), ARN (arn:aws:iam::741976594791:role/api-gw-for-s3), and maximum session duration (1 hour). Below this is the 'Permissions' tab, which lists one managed policy: 'AmazonAPIGatewayPushToCloudWatchLogs'. This policy allows API Gateway to push logs to CloudWatch Logs. The 'Permissions policies' section shows this policy attached. The 'Permissions boundary - (not set)' section is shown but empty.

The screenshot shows the AWS IAM Policies page. A new policy is being created with the following details:

- Policy name:** s3-bucket-policy
- Description - optional:** Add a short explanation for this policy.
- Permissions defined in this policy:** Allow (1 of 382 services) - S3, Limited: Write, Resource: BucketName | string like | aditya-project-1, ObjectPath | string like | None
- Add tags - optional:** No tags associated with the resource.

The bottom of the screen shows the AWS navigation bar with CloudShell, Feedback, Language, and other links.

# ADITYA MAWALKAR

The screenshot shows the AWS IAM Roles page. The left sidebar is collapsed. The main content area shows the 'api-gw-for-s3' role under 'Roles'. The 'Permissions' tab is selected. It displays one managed policy, 'AmazonAPIGatewayPushToCloudWatchLogs', which allows API Gateway to push logs to CloudWatch Logs. The ARN of the role is listed as 'arn:aws:iam::741976594791:role/api-gw-for-s3'. The 'Summary' section shows the creation date as July 25, 2023, 00:29 (UTC+05:30) and the last activity as 'None'. The maximum session duration is set to 1 hour.

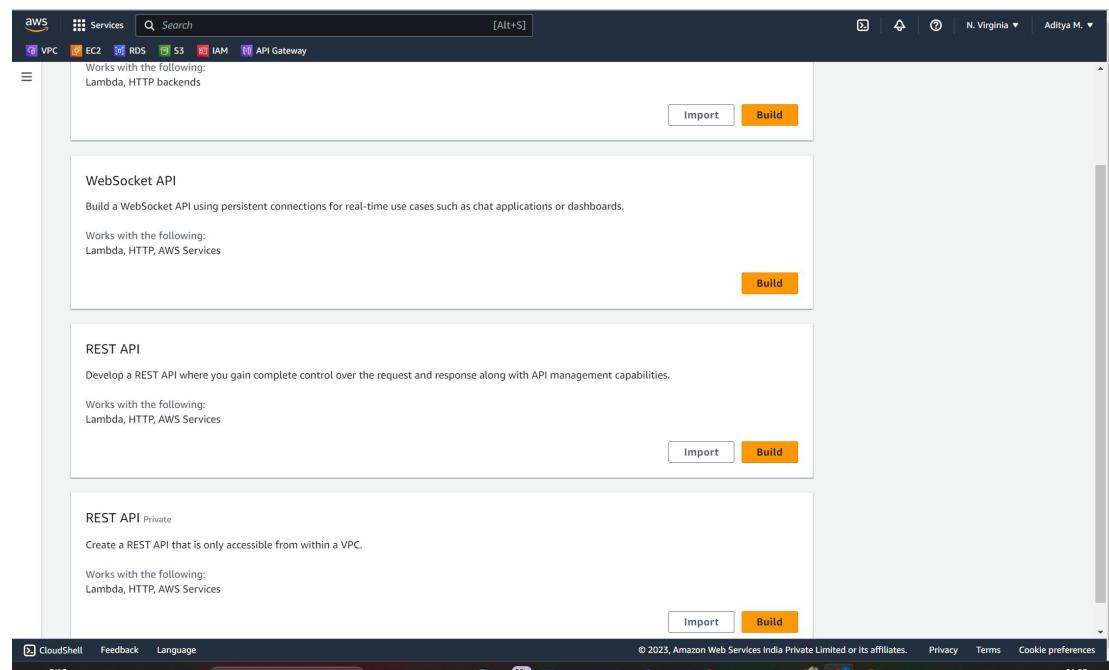
The screenshot shows the 'Add permissions' dialog for the 'api-gw-for-s3' role. Under 'Current permissions policies (1)', the 's3-bucket-policy' is selected. The 'Other permissions policies (Selected 1/861)' section lists various AWS managed policies, including 'AdministratorAccess', 'PowerUserAccess', 'ReadOnlyAccess', 'AWSCloudFormationReadOnlyAccess', 'CloudFrontFullAccess', 'AWSCloudHSMFullAccess', 'AWSCloudHSMReadOnlyAccess', 'ResourceGroupsandTagEditorFullAccess', and 'ResourceGroupsandTagEditorReadOnlyAccess'. The 's3-bucket-policy' is highlighted with a blue selection bar.

10) Now the next step is, go to the API Gateway service. create the "REST API".

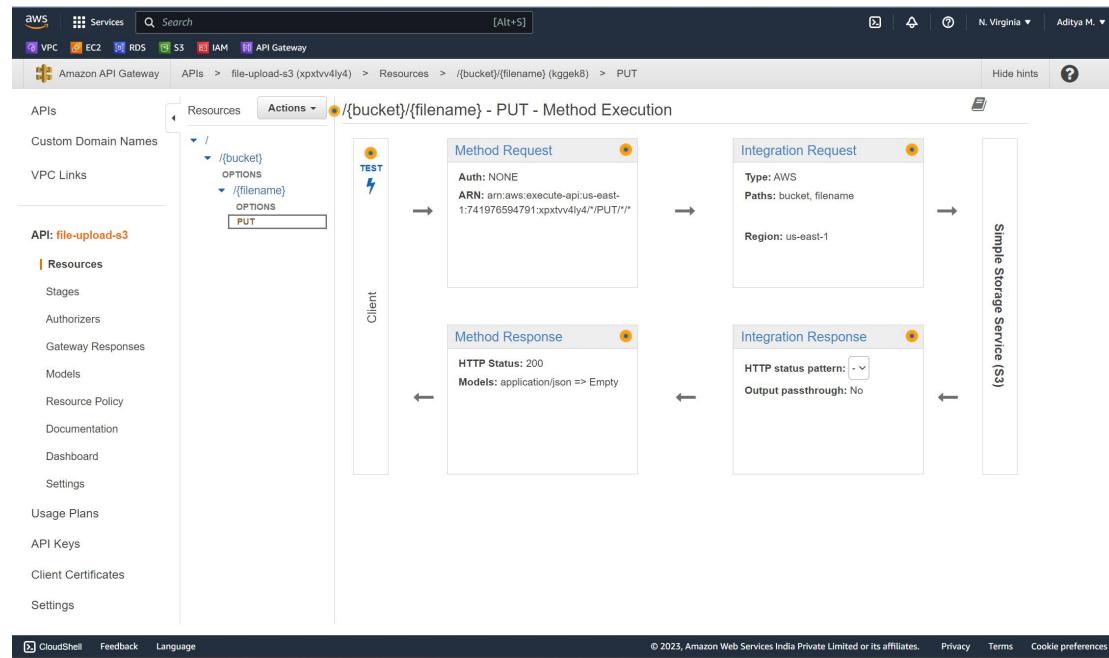
11) In this Create a new API.

12) After that click on actions and create resources with the name of bucket and in the resource path "{bucket}" enable CORS and create the resource.

- 13) Then you have to create another resource under that with the name of "filename" and in the resource path "{filename}" enable CORS and create the resource.
- 14) After this step, let's create PUT method under the "filename" resource.
- 15) Go to the action click "Create Method" and select "PUT". In this select "AWS Services" and then select a preferred AWS region, AWS service is "S3", HTTP method is "PUT", action type "use path override", then in path override put a path "{bucket}/{filename}", and then put the ARN which we copied from roles in the execution role and click "Save".
- 16) Next thing is, in PUT method go to the "Integration Method" and add path in the "URL Path Parameters".
- 17) Then go to the settings and add "binary media types" and save.
- 18) Now our API is successfully created. Next is go and deploy the API from the action drop-down menu. Name the deployment stage as "[New Stage]" and deploy.
- 19) Copy the API endpoint and test it in the POSTMAN.



# ADITYA MAWALKAR



The screenshot shows the configuration for an integration request. The left sidebar lists various API resources and settings.

**Integration type:** AWS Service (selected)

**AWS Region:** us-east-1

**AWS Service:** Simple Storage Service (S3)

**AWS Subdomain:** `(bucket).s3.us-east-1.amazonaws.com`

**HTTP method:** PUT

**Path override:** `{bucket}/{filename}`

**Execution role:** arn:aws:iam::741976594791:role/api-gw-for-s3

**Credentials cache:** Do not add caller credentials to cache key

**Content Handling:** Passthrough

**Use Default Timeout:**

**URL Path Parameters:**

| Name     | Mapped from                  | Caching                  |
|----------|------------------------------|--------------------------|
| bucket   | method.request.path.bucket   | <input type="checkbox"/> |
| filename | method.request.path.filename | <input type="checkbox"/> |

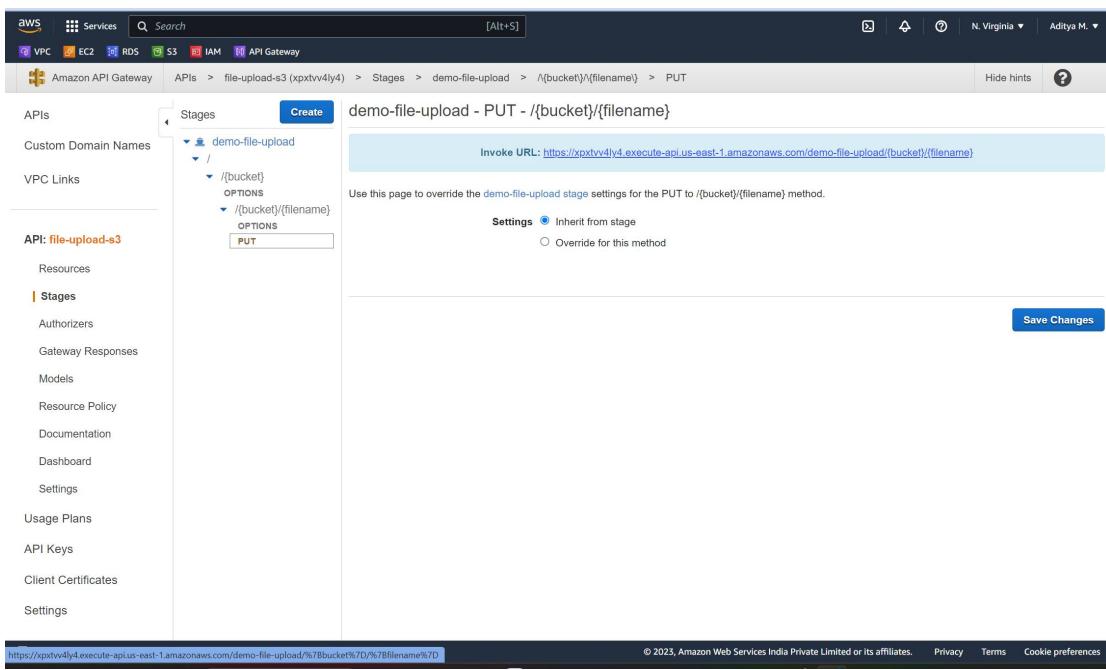
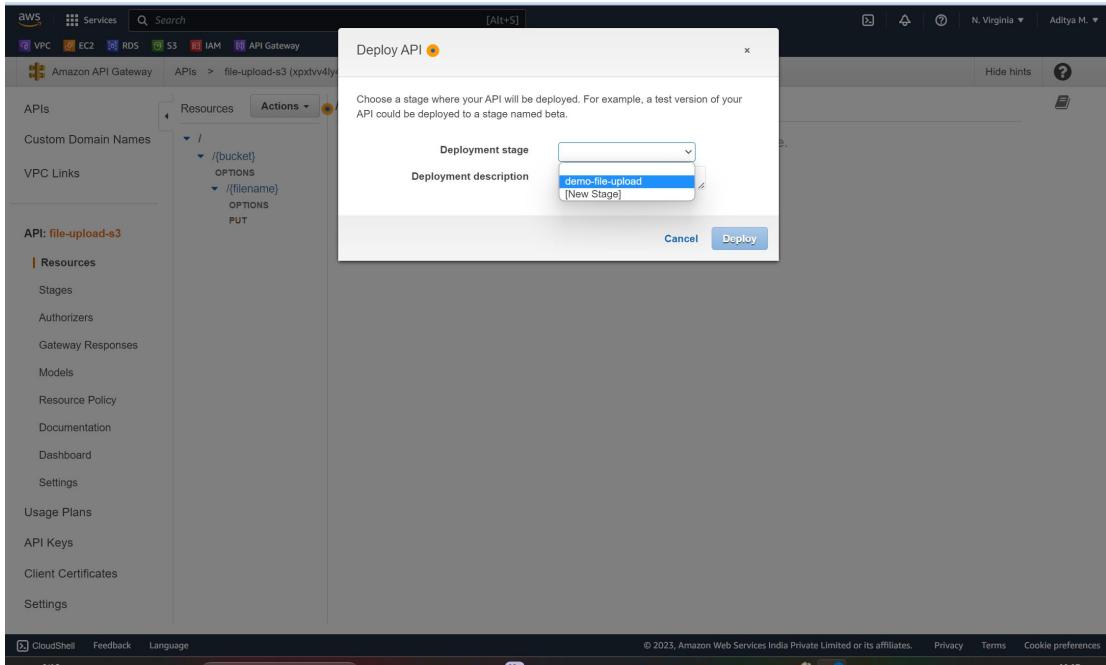
**Add path**

# ADITYA MAWALKAR

The screenshot shows the AWS API Gateway settings page. On the left sidebar, under the 'Settings' section, the 'Default Endpoint' tab is selected. It displays a note about the default execute-api endpoint and a radio button for 'Enabled'. Below this is the 'API Key Source' section, which is set to 'HEADER'. The 'Content Encoding' section is also visible. At the bottom right, there is a 'Save Changes' button.

The screenshot shows the AWS API Gateway resources page. Under the 'Actions' dropdown for a specific resource path, the 'Delete API' option is highlighted. The URL at the bottom of the browser window is <https://us-east-1.console.aws.amazon.com/api-gateway/home?region=us-east-1#>.

## ADITYA MAWALKAR



20) Go to the POSTMAN and create a new workspace and in that create a "PUT" request.

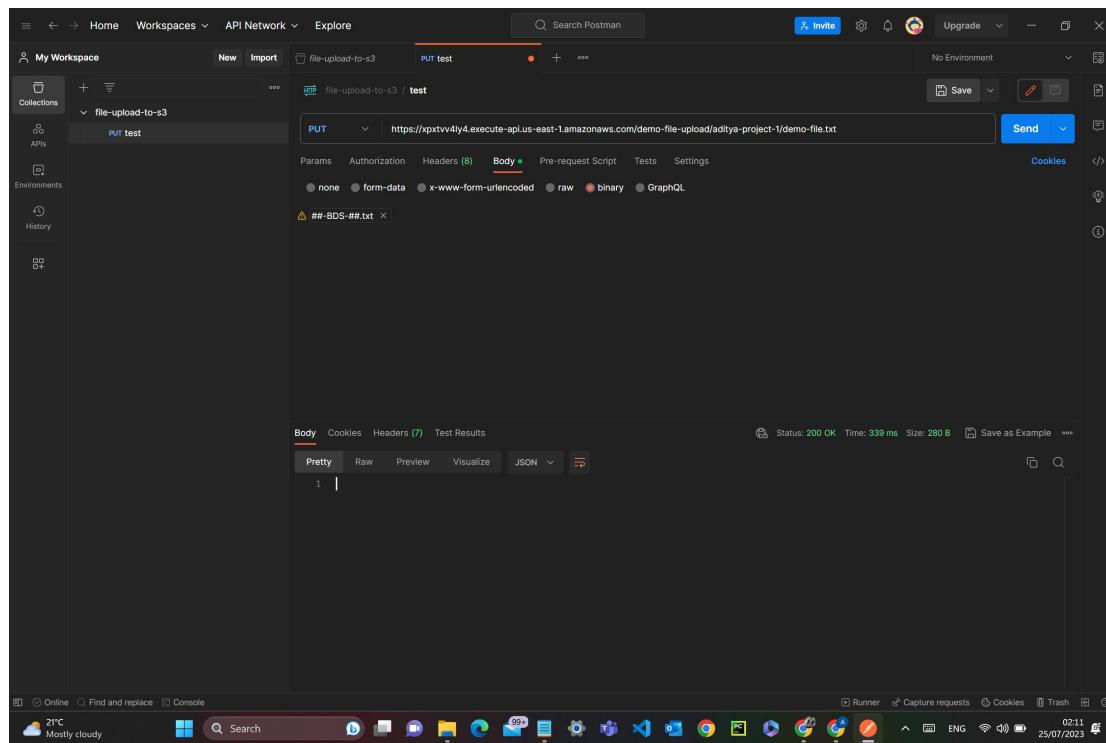
21) Paste the API endpoint and click on body -> binary and select the file and send.

(Note:- Put you bucket name in "{bucket}" and any random filename in "{filename}").

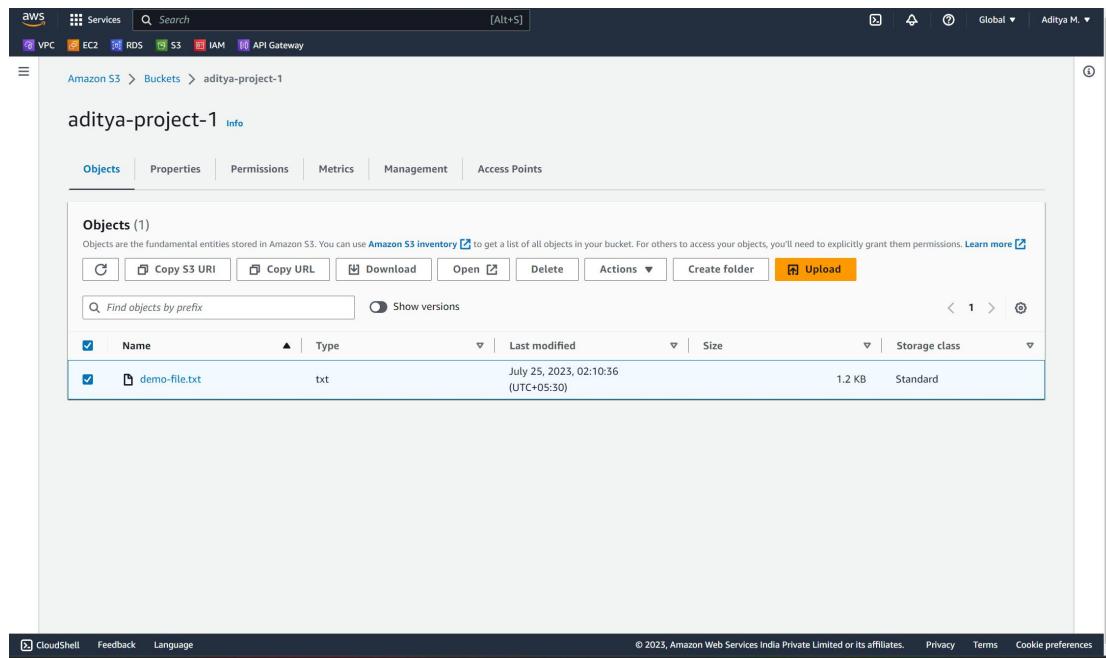
22) If status is 200 Ok and in body "1" then the file is upload to the S3 bucket.

The screenshot shows the Postman application interface. A collection named 'file-upload-to-s3' contains a test script. The test step is a PUT request to the URL `https://xpxtvv4ly4.execute-api.us-east-1.amazonaws.com/demo-file-upload/{bucket}/{filename}`. The Body tab is selected, showing the response body as '1'. The status bar at the bottom indicates a 200 OK status with a time of 537 ms and a size of 280 B.

This screenshot is identical to the one above, showing the Postman interface with a successful PUT request to the AWS Lambda endpoint. The response body is '1', and the status bar indicates a 200 OK status with a time of 537 ms and a size of 280 B.



23) To check the output go to the S3 service and open your bucket you can see the file uploaded in the bucket.



END :-

So that's how you set a file upload via API Gateway into S3 bucket.