**SLACK NOTIFICATION – MLR**

**Use Case:** The user has to get notified in his/her slack channel when he/she has been added as an additional approver.

**For Example:** If Tom is adding David as an additional reviewer, David should be getting a notification on his slack channel about the task details. Nearing the Due Date (say 5 days prior to due date) , If David has not completed the review, David will be getting a reminder about the pending review. Tom will be getting the notification on the incompleted review of David (say 2 day prior to Due Date) to handle the aging review.

A screenshot of a cell phone

Description generated with very high confidence

**XML Schema Creation:**

* Create a XML Schema **Slack XML Schema.**
* Add a schema fragment **TriggerSlackNotification.**

A close up of a logo

Description generated with very high confidence

* A screenshot of a cell phone

  Description generated with very high confidenceIn Slack channel, user will be viewing the task details. (Job Title, Priority, Due Date, Name, Country , Link to direct access the application). Create the above as elements in the schema fragment TriggerSlackNotification.

**HTTP Service:**

* Create new connection SLACK-REST-SERVICES.

<connection id="SLACK-REST-SERVICES">

<url>https://xn087p6pk6.execute-api.us-east-1.amazonaws.com</url>

<check-certificate>false</check-certificate>

<timeout>30000</timeout>

</connection>

A screenshot of a social media post

Description generated with very high confidence

* Create HTTP Service TriggerSlackNotification and Add Input XML Schema Fragment as TriggerSlackNotification

**Implementation:**

<implementation xmlns="http://httpconnector.opentext.com/1.0/implementation" xmlns:cws="http://schemas.cordys.com/cws/1.0" xmlns:c="http://schemas.cordys.com/cws/1.0" xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/" type="HTTP">

<connection-id>SLACK-REST-SERVICES</connection-id>

<uri>/dev/taskdetails</uri>

<http-method>POST</http-method>

<request-handler class="com.opentext.applicationconnector.httpconnector.impl.RestRequestHandler">

<req-headers>

<header name="Content-Type">application/json</header>

<header name="x-api-key">mS3EVwl43R3O4l7292cem4e5GDtfs9M799K83ple</header>

</req-headers>

</request-handler>

<response-handler class="com.opentext.applicationconnector.httpconnector.impl.RestResponseHandler" />

<valid-response-code>200</valid-response-code>

<ignore-cookies>True</ignore-cookies>

</implementation>

**TriggerSlackNotification Request:**

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">

<SOAP:Body>

<TriggerSlackNotification xmlns="http://schemas/acheron/MCMLifeSciences/TriggerNotifications/bpm/1.0">

<jobTitle>PARAMETER</jobTitle>

<reviewPriority>PARAMETER</reviewPriority>

<expiryDate>PARAMETER</expiryDate>

<businessSegment>PARAMETER</businessSegment>

<country>PARAMETER</country>

<appLink>PARAMETER</appLink>

<approver>PARAMETER</approver>

<titleDescription>PARAMETER</titleDescription>

<requestor>PARAMETER</requestor>

<approverMail>PARAMETER</approverMail>

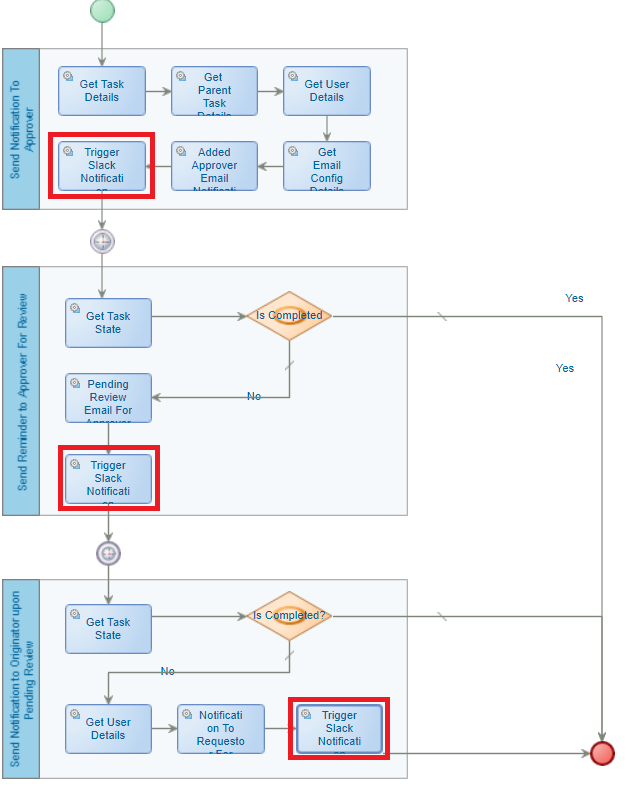
</TriggerSlackNotification>

</SOAP:Body>

</SOAP:Envelope>

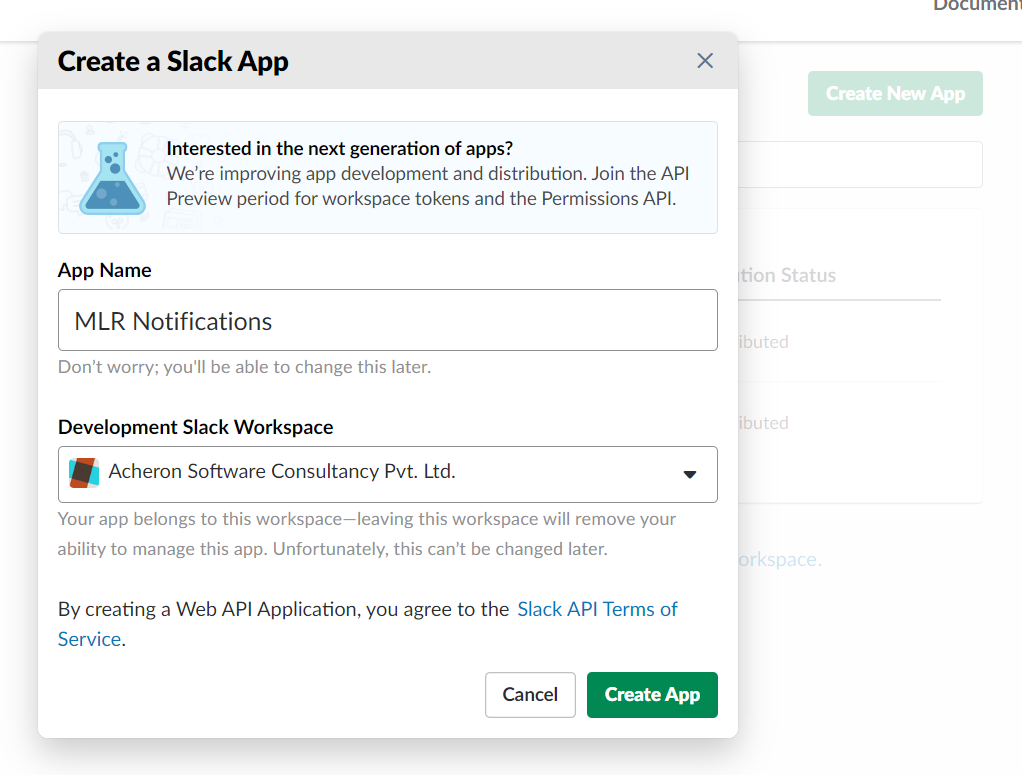
**BPM Integration:**

TriggerSlackNotification Web service is integrated with the TriggerNotificationForAdditionalApprover

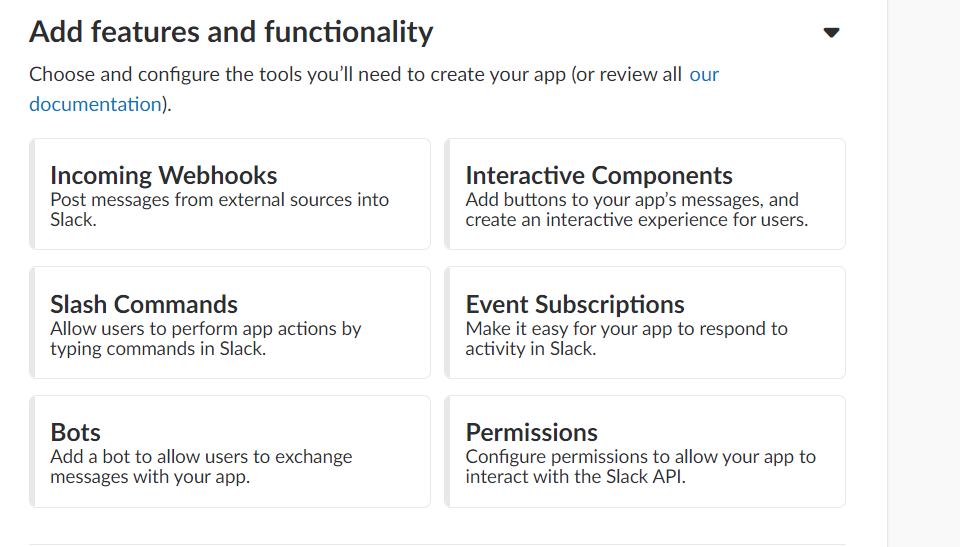


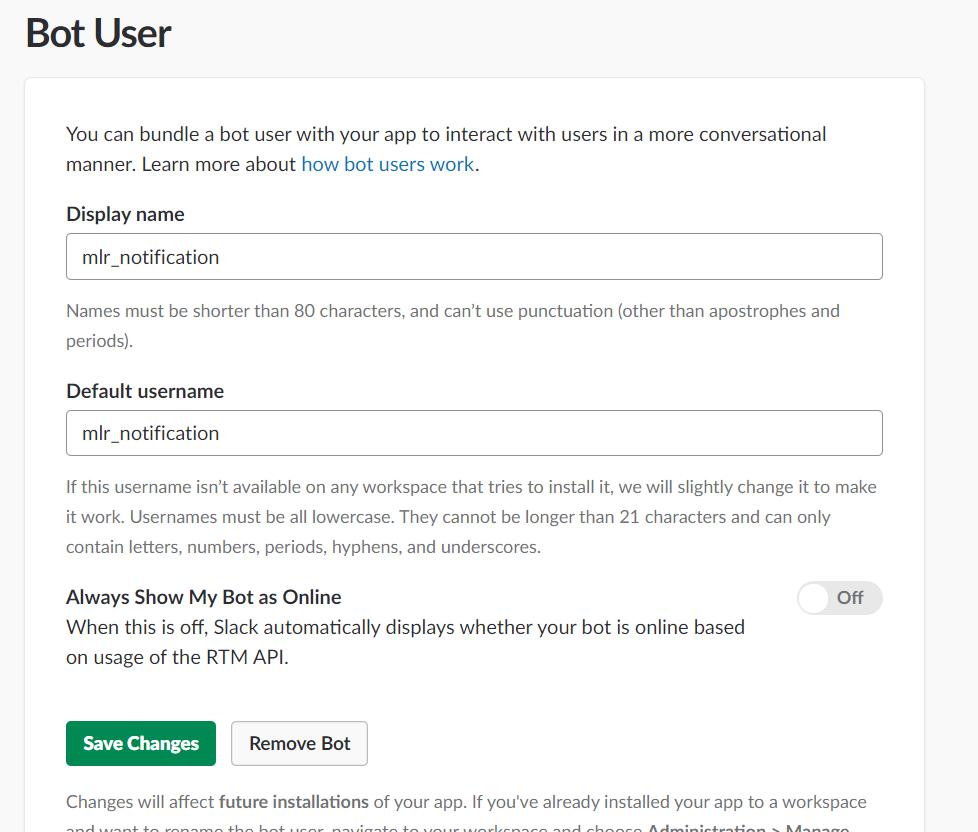
SLACK Configurations

1. Go to <https://api.slack.com/apps> to create Create a Slack App,



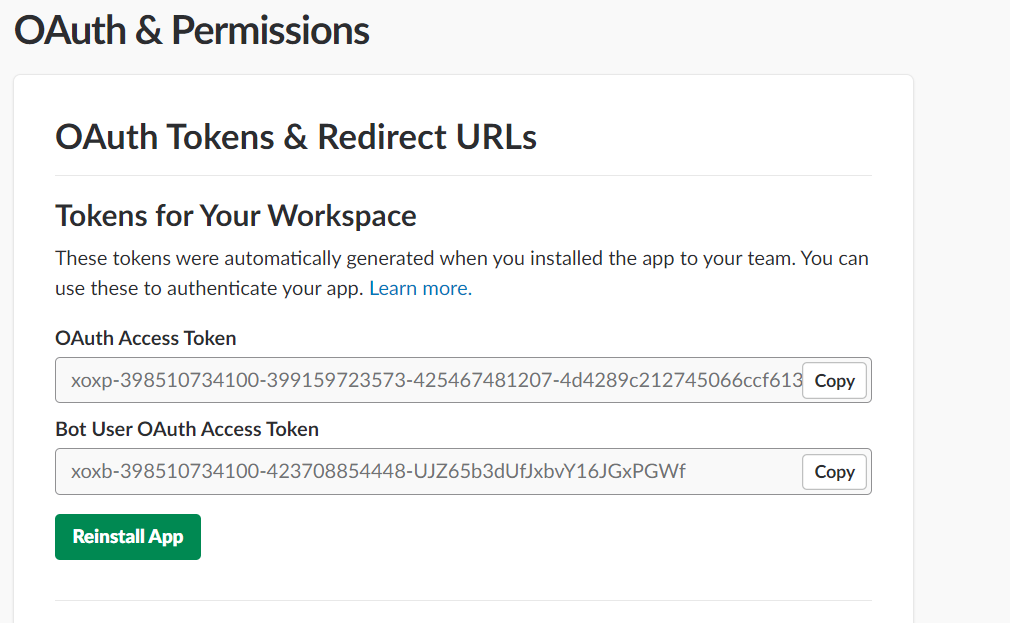
1. In the leading page, Add a Bot user





This will enable the basic permissions the Bot requires for the application.

1. OAuth and Bot Token will be generated as below, for our case, Bot token is sufficient



1. The Bot is a node application, deployed in any serverless platform. We can go with AWS Lambda and API gateway. For the authentication part to the app, the Bot token generated will be used.
2. We are done with slack setup and next we code the Bot.

**BOT Application**

1. Create a Lambda function using serverless framework and include the required libraries in the below code snippet

const serverless = require('serverless-http');

const bodyParser = require('body-parser');

const express = require('express');

const SlackBot = require('slackbots');

const app = express();

app.use(bodyParser.json({

strict: false

}));

//configure the bot

const bot = new SlackBot({

token: process.env.SLACK\_BOT\_AUTH\_TOKEN,

name: process.env.BOT\_NAME

});

//task details endpoint

app.post('/taskdetails', (req, res) => {

var taskdetails = req.body;

if (taskdetails) {

var message = {

"text": taskdetails.titleDescription,

"attachments": [{

"fallback": "Approval Task details",

"color": "#0576b9",

"title": "Click for direct access to the request",

"title\_link": taskdetails.appLink,

"fields": [{

"title": "Job Title",

"value": taskdetails.jobTitle,

"short": true

},

{

"title": "Review Priority",

"value": taskdetails.reviewPriority,

"short": true

},

{

"title": "Name",

"value": taskdetails.businessSegment,

"short": true

},

{

"title": "Country",

"value": taskdetails.country,

"short": true

},

{

"title": "Due date",

"value": taskdetails.expiryDate,

"short": true

}

],

"footer": "Acheron MLR",

"footer\_icon": "https://platform.slack-edge.com/img/default\_application\_icon.png"

}]

};

//get the username by email

bot.getUserByEmail(taskdetails.approverMail).then(function(userDetails) {

if (userDetails.name != null || userDetails.name != '' || userDetails.name != undefined) {

//post message to the user

bot.postMessageToUser(userDetails.name, '', message).then(function(response) {

res.send({"success" : "Message has been sent to " + userDetails.name });

res.status(200).end();

}).fail(function(error) {

res.send(JSON.stringify(error));

res.status(400).end();

});

}else{

res.send({"error":"Error with getting user details"});

res.status(400).end();

}

}).fail(function(error){

res.send({"error":"Given user email did not match with users of the slack workspace"});

res.status(400).end();

});

} else {

res.status(400).end();

res.send({"error" : "No task details were found"});

}

});

module.exports.handler = serverless(app);

1. Create a serverless.yml as below

service: mlr-slack-app

provider:

name: aws

runtime: nodejs6.10

stage: dev

region: us-east-1

functions:

app:

handler: index.handler

events:

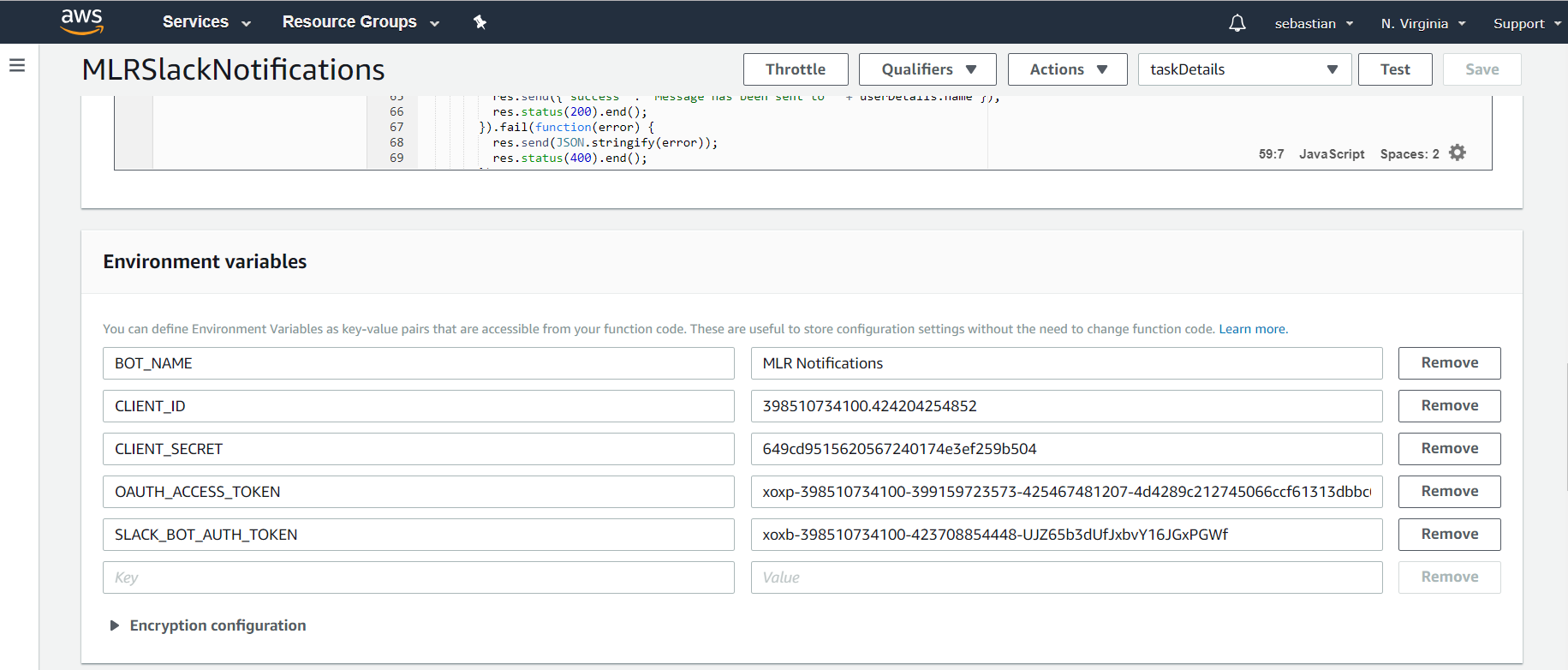
- http: ANY /

- http: 'ANY {proxy+}'

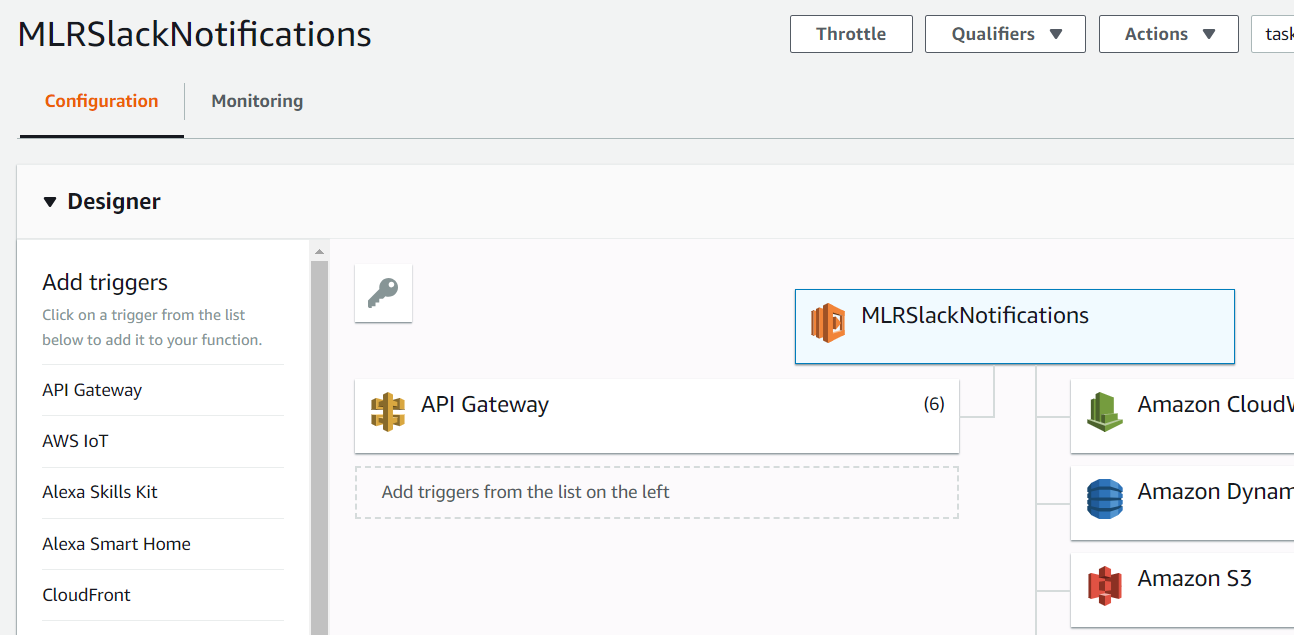
1. Package the application along with node modules and serverless.yml

**AWS Configurations**

1. We need API Gateway to server as endpoint and Lambda to run the Bot code.
2. Deploy the packaged code into AWS Lambda
3. Set up the slack configurations in the environment variables of lambda

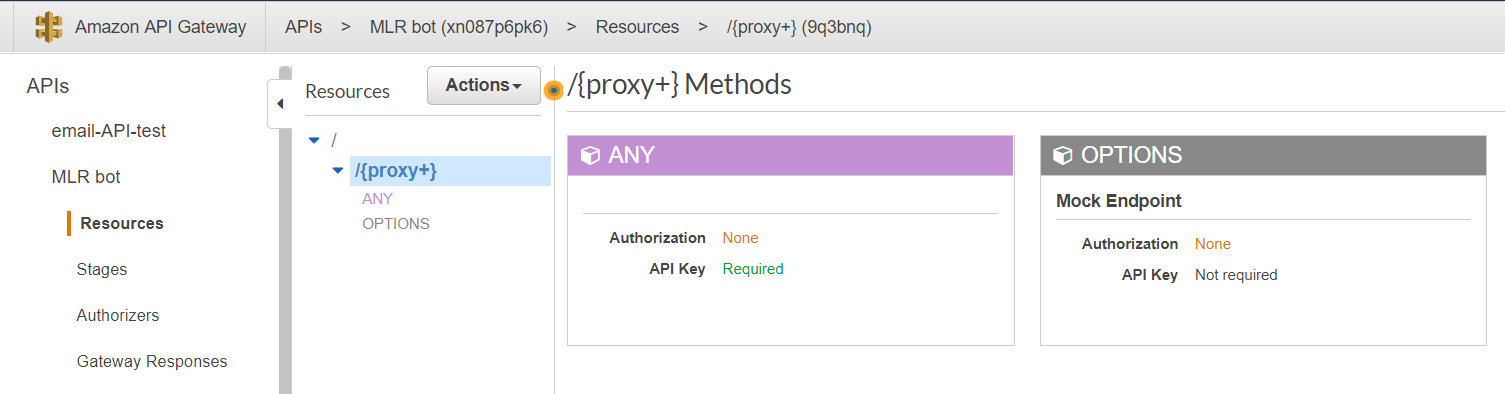


1. Configure the API Gateway as a trigger to this Lambda function

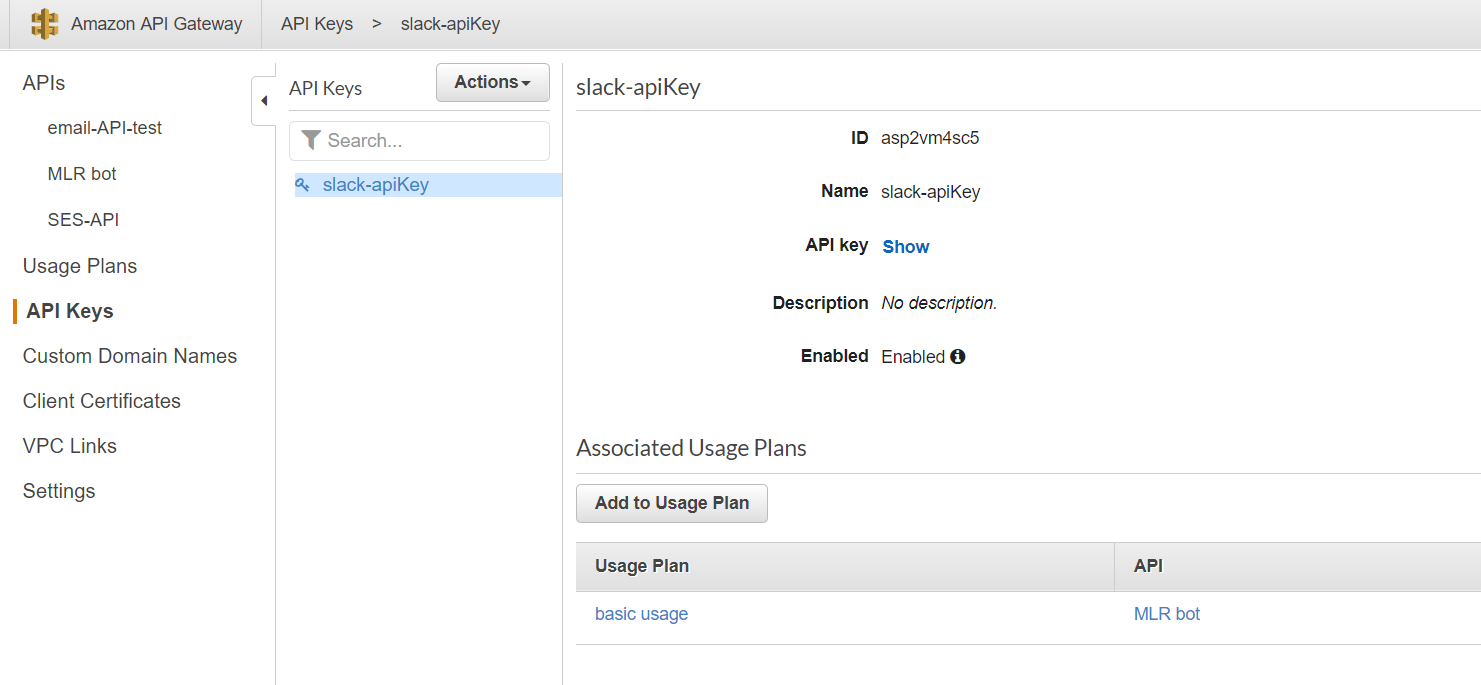


Now Configure API gateway,

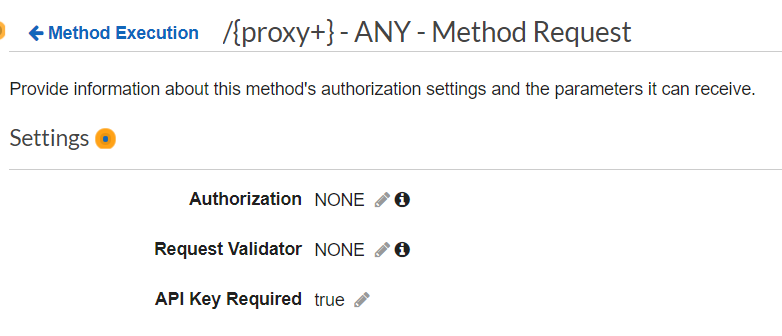
1. Create a proxy gateway Endpoint adding the Lambda function



1. Create an API key for adding a security factor to the API. This API key should be passed in the header of the request.



1. Enable API key in the Configuration,



1. Deploy the API and configure this endpoint for the http service.
2. The endpoint point and contents of the message will be handled by the lambda function.
3. That all is needed to setup Slack and AWS.

**Authors:**

**Process Suite setup :**  Nandhini Selvam

**Slack and AWS setup :**  Sebastian N Joy