# Team :- ThinkNow

## Video Presentation link:- https://youtu.be/aImAQgYyBUE

## Bot Email :- test@mymailbot.tk

You can check the model by sending mail to this mail. We have trained our model with very few examples. We have trained the model for four queries.

### Two for Requests :-

**Software request :-** request regarding user manual of a software for adding ,removing  a user to software.
**Hardware request:-** Request regarding the Hardware networking components available in the company.

### Two for Issues:-

**Software crash issue:-** If software is crashing every time whenever the user is opening the software.

**Power failure issue:-** If a switch/router shutdown or any power failure issue.

**Note:-** All above examples are trained with only 10 example each.

**Function to read mail from S3 bucket. Mailparser is used to parse the mail from MIME format to get from, Subject, Body of the mail. Then it generate the unique token. The subject and body of mail is converted to user language. Then all data is saved to dynmoDB database.**

```
/********************************* Code Block ******************************/
var AWS = require('aws-sdk');
var MailParser = require("mailparser").MailParser;              // to parse the mail from MIME
format
var parser = new MailParser();
var s3 = new AWS.S3();                                  // for reading mail from s3 bucket
var dynamodb=new AWS.DynamoDB();                                 // to store and read data from
dynmodb noSQL database
var comprehend=new AWS.Comprehend();                             // to detect the language of user
var translate = new AWS.Translate();                        // to translate the content to different
languages
 var body,sub,from;

      exports.handler = (event, context, callback) => {          // handler called when any event is
occured
```

```
    s3.getObject({Bucket: "mailinput",Key:event.Records[0].s3.object.key},function(err,data){
        if(!err)
        {
            /** parsing mail header **/
            parser.on('headers', headers => {
                sub=headers.get('subject');
                var temp=headers.get('from');
                from=temp.value[0].address;
            //console.log(sub);
            });

            /** parsing mail body **/
            parser.on('data', data => {
            if (data.type === 'text') {
                body=data.text;
            getlang(sub+"\n"+body);                    // calling function to detect language

              }
            });

            /** input stream of mail to parser in MIME format **/
            parser.write(data.Body.toString());
            parser.end();
        }
        else
        {
            console.log("Error:"+err);
        }
    });


};


/**  function to store the data to dynmodb   ***/
function dbwrite(email,sub,body,ulang,csub,cbody)
{

    var uid=uid_gen(email);
    var params={ TableName: "UserInfo",
    Item: {"uid": {S: uid},
        "email": {S: email},
        "sub": {S:sub},
        "body": {S: body},
        "ulang": {S: ulang},
        "csub": {S: csub},
        "cbody":{S: cbody}
    }};

     dynamodb.putItem(params,function(err,data){                        //saving data to dynmodb
         if(!err)
         {
                console.log("dynodb write success");
              }
              else
              {
                 console.log("dynodb write failed "+err);
              }
          });

      }



      /** function to gen the token **/
      function uid_gen(email)
      {
         var num=Math.floor(Math.random()*100000);
         return email+num;
      }

      /** function for lang detection **/
      function getlang(content)
      {
         var params = {
         Text: content
         };
         comprehend.detectDominantLanguage(params, langcall);              // detecting language and
call langcall function
      }



      /** function to convert user language to english and save it to table **/
      function langcall(err,data)
      {
         if(!err)
         {
            var ulang=data.Languages[0].LanguageCode;
            console.log(ulang);  // language code print
            if(ulang=="en")                                        // if user language is english
            {
              dbwrite(from,sub,body,ulang,sub,body);                    // saving problem statement to
database
              console.log("data saved to dynamodb as same langugage");
            }
            else                                           // if user language is other than english
            {   var csub,cbody;
                var params = {
                SourceLanguageCode: ulang,
                TargetLanguageCode: 'en',
                Text: sub};
                translate.translateText(params, function(err, data) {          // translating subject of mail
to english language
```

```
                    if (err) console.log(err, err.stack); // an error occurred
                    else
                    {
                        csub=data.TranslatedText;
                        console.log("csub: "+data.TranslatedText);
                        var params = {
                        SourceLanguageCode: ulang,
                        TargetLanguageCode: 'en',
                        Text: body};
                        translate.translateText(params, function(err, data) {       // translating body of mail to
english language
                        if (err) console.log(err, err.stack); // an error occurred
                        else
                            {
                                cbody=data.TranslatedText;
                                console.log("cbody: "+data.TranslatedText);
                                dbwrite(from,sub,body,ulang,csub,cbody);               // store the translated
problem statement
                                console.log("data saved to dynamodb after translation");

                            }
                        });
                    }
                });
                }
            }
            else
            {
                console.log(err);
            }
        }

/****************************** End of Function ****************************/
```

**Function to classify the mail content by assigning a particular tag/token to it. Then reply to user with solution in user language and if solution not found then send the problem to resolving group in English language and send email to user in user language about assignment of problem. The mail is sent using mail server on shared hosting machine on cloud by sending a post request to our hosting address. Where a script is written to send an email.**

```
/******************************** Code Block ****************************/
const AWS = require('aws-sdk');
const https = require('https');                         // to send post request to our mail server to send mail
const MonkeyLearn = require('monkeylearn')              // to call the classifier model to classify the
problem statement
const dynamodb = new AWS.DynamoDB();                   // to store and read data from dynmodb
noSQL database
```

```
var translate = new AWS.Translate();                    // to translate the text
let dbClient = new AWS.DynamoDB.DocumentClient();       // to read data from dynmodb

var uid,cbody,csub,email,ulang;
exports.handler = (event, context) => {
   event.Records.forEach((record) => {                  // reading the dynamodb content when triger
      uid=record.dynamodb.Keys.uid.S;
      cbody=record.dynamodb.NewImage.cbody.S;
      csub=record.dynamodb.NewImage.csub.S;
      email=record.dynamodb.NewImage.email.S;
      ulang=record.dynamodb.NewImage.ulang.S;

      //console.log('DynamoDB Record: %j',record.dynamodb );
   });

   /** to classify the problem using ML model on monkeylearn cloud services **/
   const ml = new MonkeyLearn('1534fc2f8f4dd41ee1af8e9bd69c0e9a50dba597');
   let model_id = 'cl_pCXF9YoP';
   let data = [csub+" "+cbody];
   ml.classifiers.classify(model_id, data).then(res => {
   var cat= res.body[0].classifications[0].tag_name;
   var conf=res.body[0].classifications[0].confidence;
   console.log("Tag:"+cat);
   console.log("Confidence:"+conf);

   /** if confidence is less then send the problem statement to expert group to reply  **/
   if(conf<0.4)
   {
      var params = {TableName: "assign",
      Key: {"type": cat}
      };
      console.log("progress 28");
      dbClient.get(params,function(err,data){
         console.log("entered in mail block");
         if(!err)
         {
            var amail=data.Item.amail;
            var name=data.Item.name;
            console.log("email "+email);
            console.log("name "+name);

            var tagdata="";
            var params = {
            TableName: "assign"
            };
            dynamodb.scan(params, function(err, data) {         // reading all tags to send to expert for
classification
            if (err) console.log(err, err.stack); // an error occurred
            else
               {
               //console.log(data);           // successful response
               for(var i=0;i<data.Count;i++)
```

```
                    {
                        tagdata+="{ "+data.Items[i].type.S+":"+data.Items[i].name.S+"}<br>";
                        console.log("tagdata: "+tagdata);
                    }
                     sendmail(amail,uid,"<b>Subject: </b>"+csub+"<br>"+cbody+"<br><br>"+tagdata);
                    }
                });



                /** to notify user that their query is assigned to a resolving group **/
                var sub="In response to your query"
                var body="Your query is assigned to our "+name+" .Your problem will be solved within 
24 hours. Our team will reply your soon";
                //langcall(sub,body);
                langcall(sub,body);
            }
            else
            {
                console.log(err);
            }
        });
    }
    else                // send the solution from knowledgebase
    {
     var params = {TableName: "solution",
        Key: {"irname": cat}
        };
        console.log("progress 28");
        dbClient.get(params,function(err,data){
            console.log("entered in mail block");
            if(!err)
            {
                var sub=data.Item.sub;
                var body=data.Item.body;
                console.log("sub "+sub);
                console.log("Body "+body);
                langcall(sub,body);
            }
            else
            {
                console.log(err);
            }
        });
    }
    });
};



/** funciton to reply the user according to his/her input language **/
 function langcall(sub,body)
```

```
        {

               var tsub,tbody;
                if(ulang=="en")
                {
                   sendmail(email,sub,body);                              // send mail if user language is english
                   console.log("replied in english as same language");
                }
                else
                {
                    var params = {
                    SourceLanguageCode: 'en',
                    TargetLanguageCode: ulang,
                    Text: sub};
                    translate.translateText(params, function(err, data) {       //converting sub to user
language
                    if (err) console.log(err, err.stack); // an error occurred
                    else
                    {
                        tsub=data.TranslatedText;
                        console.log("csub: "+data.TranslatedText);
                        var params = {
                        SourceLanguageCode: 'en',
                        TargetLanguageCode: ulang,
                        Text: body};
                        translate.translateText(params, function(err, data) {       // converting body to user
language
                        if (err) console.log(err, err.stack); // an error occurred
                        else
                            {
                                tbody=data.TranslatedText;
                                console.log("cbody: "+data.TranslatedText);
                                sendmail(email,tsub,tbody);                      // sending the mail user in his/her
language
                                console.log("data sent agter translation");

                            }
                        });
                    }
                });
                }
        }


        /** function to send mail by sending post request to our server **/
        function sendmail(email,sub,body)
        {
            console.log("email is "+email);
            var data='too='+email+'&subb='+sub+'&bodyy='+body;
            console.log("post data: "+data);
            const options = {
            hostname: 'udgamtrust.com',                      //domain where our mail server is running
            port: 443,
            path: '/test_ret.php',                    // script on server to send mail
            method: 'POST',
             headers: {
             'Content-Type': 'application/x-www-form-urlencoded',
                'Content-Length': Buffer.byteLength(data, 'utf8')
                }
            };

            const req = https.request(options, (res) => {            // sending post request to server
            console.log(`statusCode: ${res.statusCode}`);
            console.log(`statusCode: ${JSON.stringify(res.headers)}`);
            res.setEncoding('utf8');

            res.on('data', (d) => {
            console.log("response: "+d);
            });

            req.on('end', (error) => {
            console.error("error: "+error);
            });

            req.on('error', (error) => {
            console.error("error: "+error);
            });
            });
            req.write(data);
            req.end();
        }


  /****************************** End of Function ************************/
```

**Function to send the solution sent by the resolving group to user in the user language.**

```
/**************************** Code Block ****************************/

var AWS = require('aws-sdk');
var MailParser = require("mailparser").MailParser;             // to parse the mail from MIME
format
var parser = new MailParser();
var s3 = new AWS.S3();                                 // for reading mail from s3 bucket
const https = require('https');                        // to send post request to our mail server to send
mail
var dynamodb=new AWS.DynamoDB();                            // to store and read data from
dynmodb noSQL database
var comprehend=new AWS.Comprehend();                        // to detect the language of user
var translate = new AWS.Translate();                        // to translate the content to different
languages
let dbClient = new AWS.DynamoDB.DocumentClient();                   // to read data from dynmodb
 var body,ulang,uid,email;

        exports.handler = (event, context, callback) => {
            s3.getObject({Bucket: "treply",Key:event.Records[0].s3.object.key},function(err,data){
                if(!err)
                {
                    parser.on('headers', headers => {
                        uid=headers.get('subject');

                    //console.log(sub);
                    });

                    parser.on('data', data => {
                    if (data.type === 'text') {
                        body=data.text;
                        var ptype,info,solution;
                        /** index of problem code block **/
                        var i=body.indexOf("{");
                        var j=body.indexOf("}");
                        if(j-i>1)
                        {
                            var k=body.indexOf(":",i);
                            /** getting problem code **/
                            if(k!=-1)
                                ptype=body.slice(i+1,k);
                            else
                                k=i;
                            /** getting about problem code if exits **/
                            info=body.slice(k+1,j);
                        }
                        /** index of solution block **/
                        j=body.indexOf("{",j);
                        i=body.indexOf("}",j);
                        solution=body.slice(j+1,i);

                    /** removing Re from subject of mail to get the token **/
                    uid=uid.replace("Re:"," ");
                    uid=uid.trim();
                    console.log("uid: "+uid);
                    var params = {TableName: "UserInfo",
                    Key: {"uid": uid}
                    };
                    dbClient.get(params,function(err,dat){
                    if(!err && dat.Item)
                    {
                        console.log(dat);
                        email=dat.Item.email;
                        ulang=dat.Item.ulang;
                        console.log("email "+email);
                        console.log("ulang "+ulang);

                        console.log("body: "+body);
                        var sub="Solution to your query";
                        langcall(sub,solution);
                    }
                    else
                    {
                        console.log(err);
                    }
                });
                      }
                    });

                    /** writing stream of mail in MIME format **/
                    parser.write(data.Body.toString());
                    parser.end();
                }
                else
                {
                    console.log("unable to read data from s3 "+err);
                }
            });


        };





        /**  function to store the data to dynmodb   ***/
        function dbwrite(email,sub,body,ulang,csub,cbody)
        {

            //var uid=uid_gen(email);
            var params={ TableName: "UserInfo",
            Item: {"uid": {S: uid},
                "email": {S: email},
                "sub": {S:sub},
                "body": {S: body},
                "ulang": {S: ulang},
                "csub": {S: csub},
                "cbody":{S: cbody}
            }};

             dynamodb.putItem(params,function(err,data){
                 if(!err)
                 {
                    console.log("dynodb write success");
                 }
                 else
                 {
                     console.log("dynodb write failed");
                 }
             });

        }







        /** function to send post data to server to send the mail **/
        function sendmail(email,sub,body)
        {
            console.log("email is "+email);

            var data='too='+email+'&subb='+sub+'&bodyy='+body+"<br>";
            console.log("post data: "+data);
            const options = {
            hostname: 'udgamtrust.com',
            port: 443,
            path: '/test_ret.php',
            method: 'POST',
             headers: {
             'Content-Type': 'application/x-www-form-urlencoded',
                'Content-Length': Buffer.byteLength(data, 'utf8')
                }
            };
            console.log("size: "+Buffer.byteLength(data, 'utf8'));

            const req = https.request(options, (res) => {                    // sending post request
            console.log(`statusCode: ${res.statusCode}`);
            console.log(`statusCode: ${JSON.stringify(res.headers)}`);
            res.setEncoding('utf8');

            res.on('data', (d) => {
            console.log("response: "+d);
            });

            req.on('end', (error) => {
            console.error("error: "+error);
            });


            req.on('error', (error) => {
            console.error("error: "+error);
            });
            });
            req.write(data);
            req.end();
        }
```

```
       /** function to send email by converting to user language **/
       function langcall(sub,body)
        {
               var tsub,tbody;
                if(ulang=="en")
                {
                   sendmail(email,sub,body);                                   // sending mail to user in english
language
                   console.log("replied in english as same language");
                }
                else
                {
                    var params = {
                    SourceLanguageCode: 'en',
                    TargetLanguageCode: ulang,
                    Text: sub};
                    translate.translateText(params, function(err, data) {
                    if (err) console.log(err, err.stack); // an error occurred
                    else
                    {
                        tsub=data.TranslatedText;
                        console.log("csub: "+data.TranslatedText);
                        var params = {
                        SourceLanguageCode: 'en',
                        TargetLanguageCode: ulang,
                        Text: body};
                        translate.translateText(params, function(err, data) {
                        if (err) console.log(err, err.stack); // an error occurred
                        else
                            {
                                tbody=data.TranslatedText;
                                console.log("cbody: "+data.TranslatedText);
                                sendmail(email,tsub,tbody);                    //sending mail after translating to user
language
                                console.log("data sent agter translation");

                            }
                        });
                    }
                });
                }
        }



/************************** End of Function *************************/
```

**Function to read the reply from the resolving group and decode it. Then it will add the solution to dynmoDB database. If the tag is new this function will add new tag to ML model and train the model by adding new problem statement to model. If the tag is already present then train the problem statement with existing tag.**

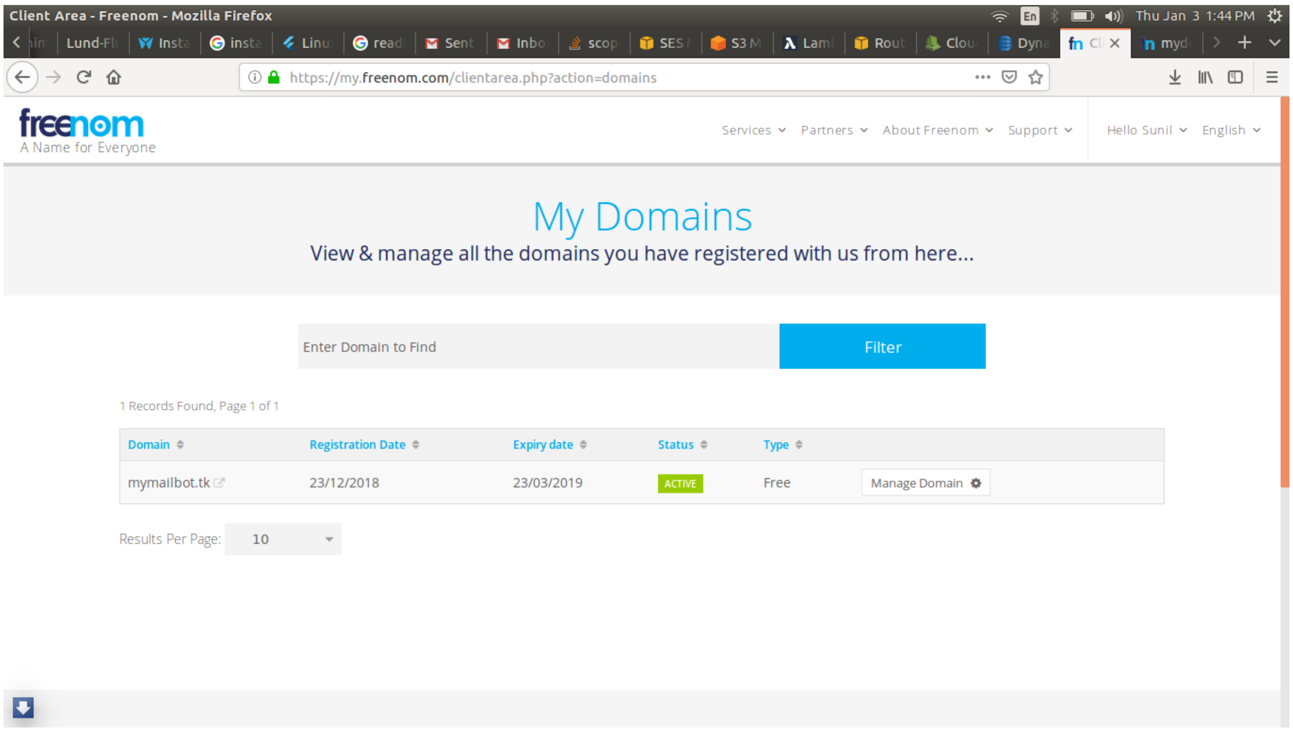
```
/****************************** Code Block  ******************************/

var AWS = require('aws-sdk');
var MailParser = require("mailparser").MailParser;             // to parse the mail from MIME format
var parser = new MailParser();
var s3 = new AWS.S3();                                          // for reading mail from s3 bucket
const MonkeyLearn = require('monkeylearn');                    // to call the classifier model to train the
model
var dynamodb=new AWS.DynamoDB();                               // to store and read data from dynmodb
noSQL database
let dbClient = new AWS.DynamoDB.DocumentClient();              // to read data from dynmodb
 var body,sub,uid;

        exports.handler = (event, context, callback) => {

            var key=event.Records[0].s3.object.key;
            s3.getObject({Bucket: "treply",Key:key},function(err,data){
                if(!err)
                {
                    /** reading header of mail i.e. subject **/
                    parser.on('headers', headers => {
                        uid=headers.get('subject');
                    });

                    /** to parse the body of the mail **/
                    parser.on('data', data => {
                    if (data.type === 'text') {
                        body=data.text;

                    /** to remove the RE: from the subject to get the token**/
                    uid=uid.replace("Re:"," ");
                    uid=uid.trim();
                    console.log("uid: "+uid);
                    var params = {TableName: "UserInfo",
                    Key: {"uid": uid}
                    };
                    dbClient.get(params,function(err,dat){          // to read data from dynmodb
                    if(!err)
                    {
                        var csub=dat.Item.csub;                    // user problem subject in english language
                        var cbody=dat.Item.cbody;                  // user problem body in english language

                        var ptype,info,solution;
                        /** index of problem code block **/
                        var i=body.indexOf("{");
                        var j=body.indexOf("}");
```

```
if(j-i>1)
{
    var k=body.indexOf(":",i);
    /** getting problem code **/
    if(k!=-1)
        ptype=body.slice(i+1,k);
    else
        k=i;
    /** getting about problem code if exits **/
    info=body.slice(k+1,j);
}
/** index of solution block **/
j=body.indexOf("{",j);
i=body.indexOf("}",j);
solution=body.slice(j+1,i);

console.log("Pcode: "+ptype);
console.log("Info: "+info);
console.log("Solution: "+solution);

if(ptype!=null)
{
    ptype=ptype.trim();
    ptype=ptype.toUpperCase();
var params = {TableName: "assign",
Key: {"type": ptype}
};
dbClient.get(params,function(err,data){                //reading data from dynmodb
    /** if flag is present then send the problem statement for training */
    if(!err && data.Item)
    {
        console.log("data found: "+data.Item.name);
        upload_data(ptype,csub+" "+cbody);
    }
    else if(!err)
    {
        /** if tag is new then add tag to ML model and send the problem statement for
training **/
        if(info)
        {
            sub="Solution for your Issue";
            tagupdate(ptype,info);
            addsol(ptype,sub,solution);
            addtag(ptype,csub+" "+cbody);
        }
    }
    else
    {
        console.log("data not found "+err);
    }

});
```

```
                }

            }
            else
            {
                console.log(err);
            }
        });
              }
            });
            parser.write(data.Body.toString());
            parser.end();
        }
        else
        {
            console.log("unable to read data from s3");
        }
    });
};





/**  function to store the tag and taginfo to dynmodb   ***/
function tagupdate(tag,info)
{
    //var uid=uid_gen(email);
    var params={ TableName: "assign",
    Item: {"type": {S: tag},
        "name": {S: info},
        "amail": {S: "bishnoi.sunil62@gmail.com"}
    }};

     dynamodb.putItem(params,function(err,data){
         if(!err)
         {
            console.log("dynodb write success");
         }
         else
         {
             console.log("dynodb write failed");
         }
     });
}


/** function to add solution to database **/
function addsol(tag,sub,body)
{
    //var uid=uid_gen(email);
    var params={ TableName: "solution",
         Item: {"irname": {S: tag},
            "body": {S: body},
            "sub": {S: sub}
         }};

          dynamodb.putItem(params,function(err,data){
             if(!err)
             {
               console.log("dynodb write success");
             }
             else
             {
                console.log("dynodb write failed");
             }
          });
      }




   /** function to add tag in ML model **/
   function addtag(tag,text)
   {
      const ml = new MonkeyLearn('1534fc2f8f4dd41ee1af8e9bd69c0e9a50dba597');
      let model_id = 'cl_pCXF9YoP';
      let data = {"name":tag};
      ml.classifiers.tags.create(model_id, data).then(res => {
         console.log(res.body);
         upload_data(tag,text);
      })
   }



   /** function to add data to ML model for training**/
   function upload_data(tag,text)
   {
      const ml = new MonkeyLearn('1534fc2f8f4dd41ee1af8e9bd69c0e9a50dba597');
      let model_id = 'cl_pCXF9YoP';
      let data = [{text:text,tags:[tag]}];
      ml.classifiers.upload_data(model_id, data).then(res => {
         console.log(res.body);
      })
   }
```

**/****************************** End of Function ****************************/**

**Free domain name registered and name server are configured according to AWS SES.**



https://my.freenom.com/clientarea.php?action=domains

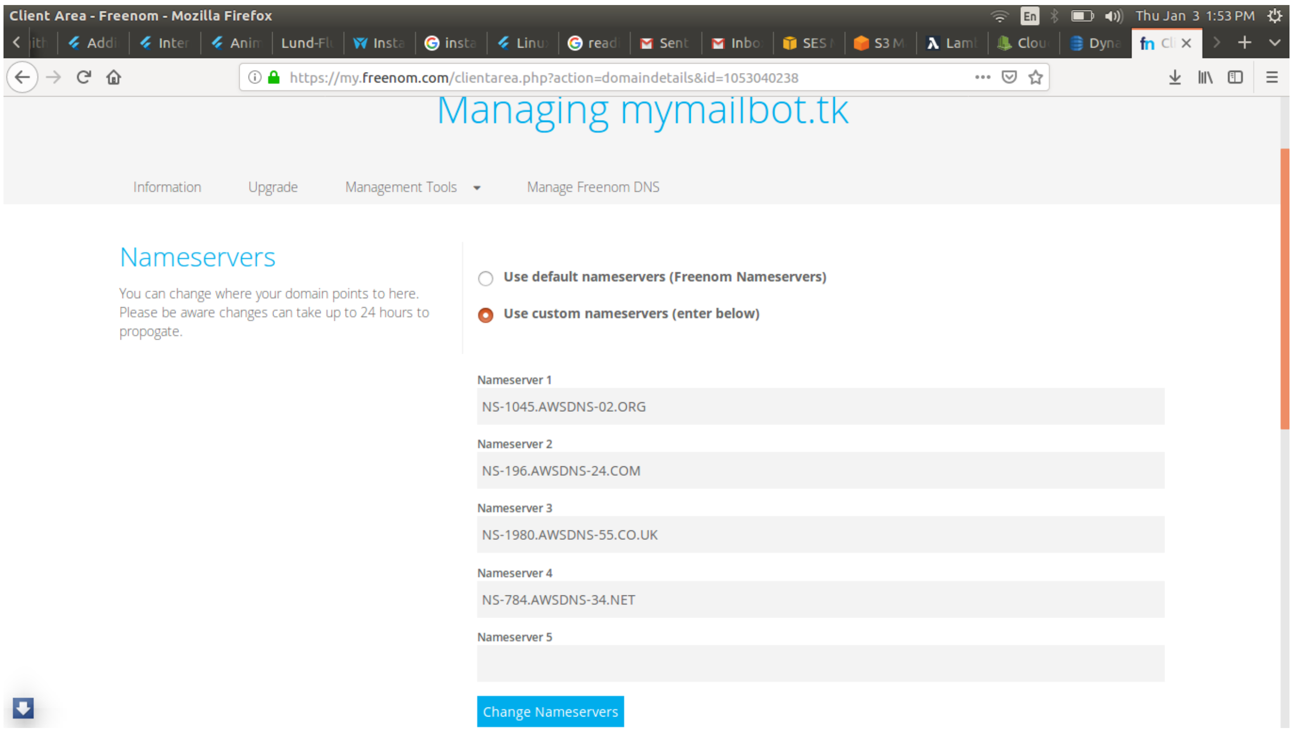
My Domains

View & manage all the domains you have registered with us from here...

1 Records Found, Page 1 of 1

| Domain | Registration Date | Expiry date | Status | Type | |
|---|---|---|---|---|---|
| mymailbot.tk | 23/12/2018 | 23/03/2019 | ACTIVE | Free | Manage Domain |

Results Per Page: 10



https://my.freenom.com/clientarea.php?action=domaindetails&id=1053040238

Managing mymailbot.tk

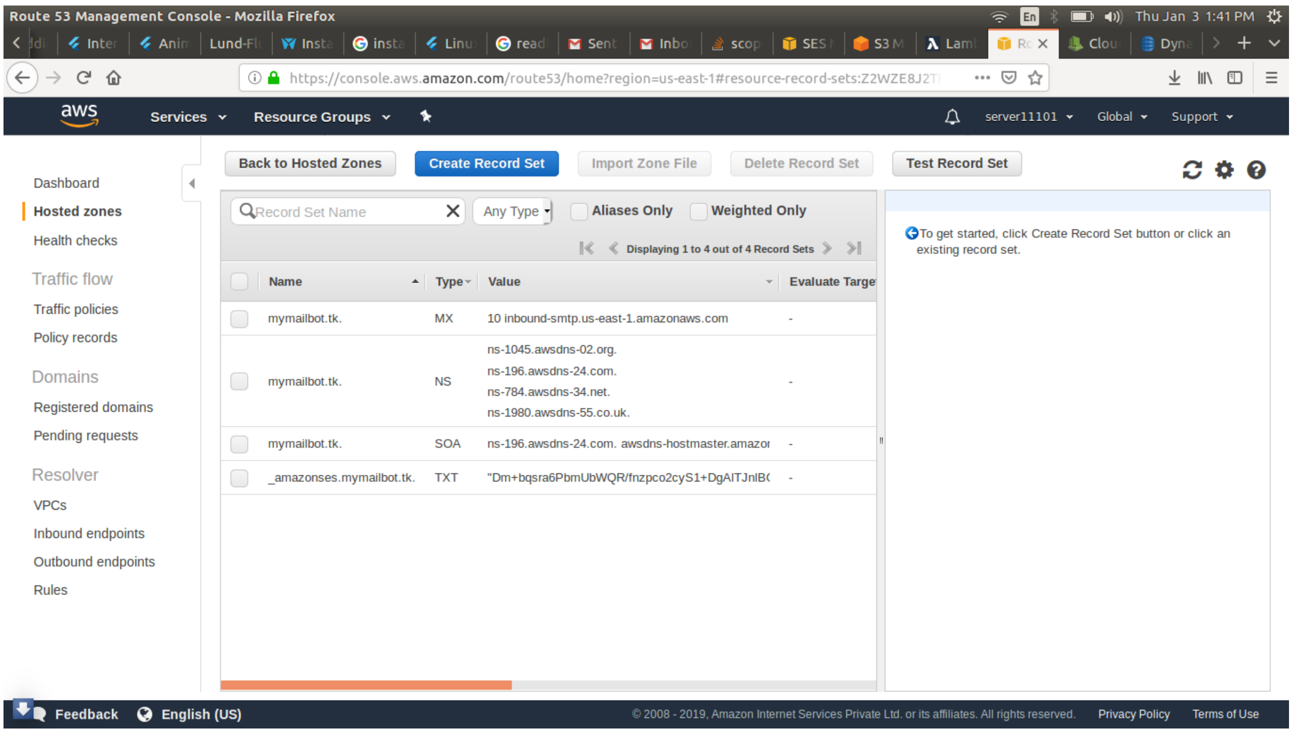
Nameservers

You can change where your domain points to here. Please be aware changes can take up to 24 hours to propogate.

Use default nameservers (Freenom Nameservers) [✓ Use custom nameservers (enter below)]

| | |
|---|---|
| Nameserver 1 | NS-1045.AWSDNS-02.ORG |
| Nameserver 2 | NS-196.AWSDNS-24.COM |
| Nameserver 3 | NS-1980.AWSDNS-55.CO.UK |
| Nameserver 4 | NS-784.AWSDNS-34.NET |
| Nameserver 5 | |

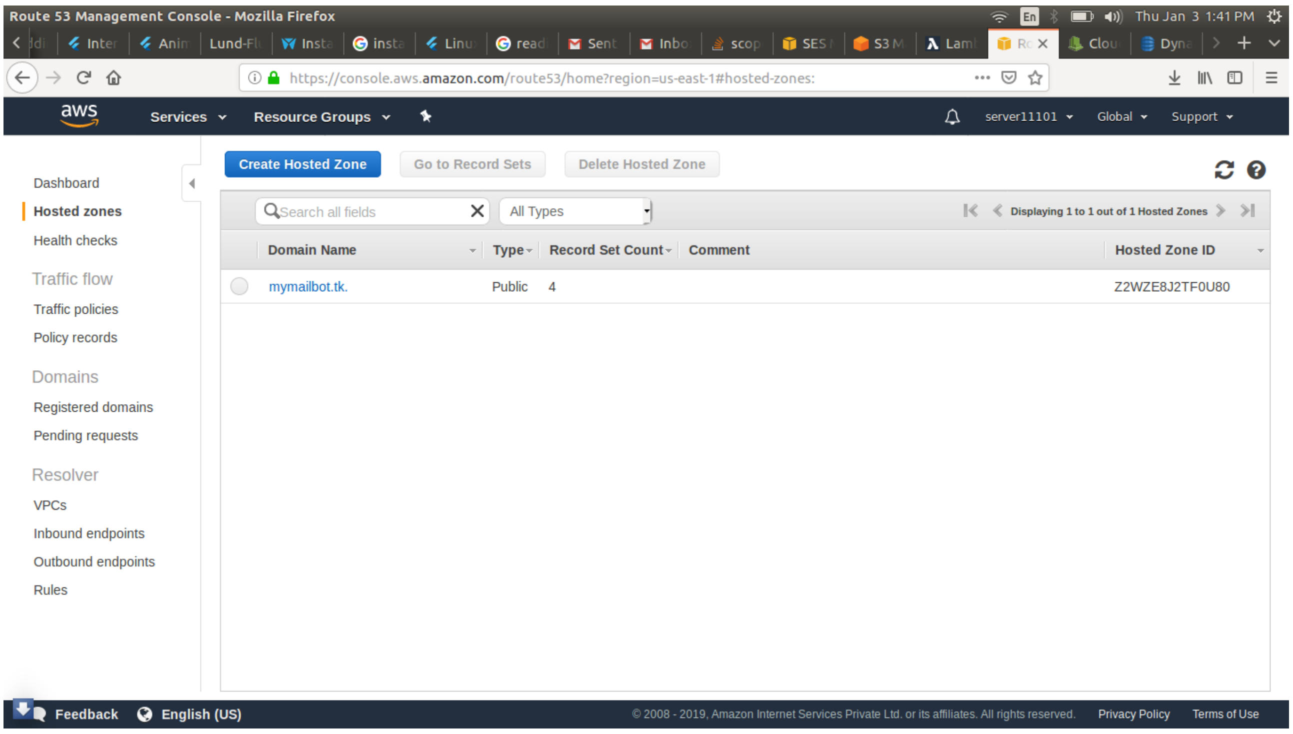
## Domain configuration on AWS Route 53



| Name | Type | Value | Evaluate Targe |
|---|---|---|---|
| mymailbot.tk. | MX | 10 inbound-smtp.us-east-1.amazonaws.com | - |
| mymailbot.tk. | NS | ns-1045.awsdns-02.org.<br>ns-196.awsdns-24.com.<br>ns-784.awsdns-34.net.<br>ns-1980.awsdns-55.co.uk. | - |
| mymailbot.tk. | SOA | ns-196.awsdns-24.com. awsdns-hostmaster.amazoı | - |
| _amazonses.mymailbot.tk. | TXT | "Dm+bqsra6PbmUbWQR/fnzpco2cyS1+DgAITJnIB( | - |

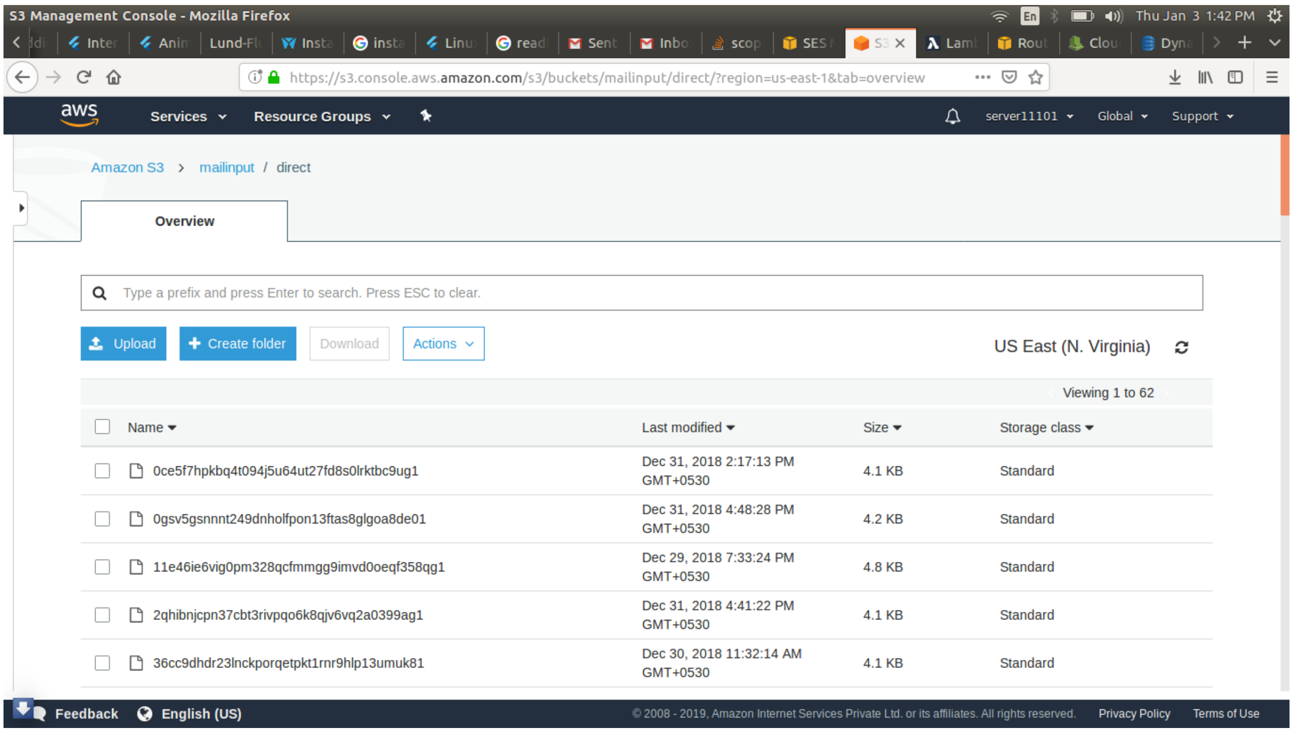
To get started, click Create Record Set button or click an existing record set.

## Domain verified on SES



| Domain Name | Type | Record Set Count | Comment | Hosted Zone ID |
|---|---|---|---|---|
| mymailbot.tk. | Public | 4 | | Z2WZE8J2TF0U80 |

## Email MIME files in the S3 Bucket


S3 Management Console - Mozilla Firefox
https://s3.console.aws.amazon.com/s3/buckets/mailinput/direct/?region=us-east-1&tab=overview
Services
Resource Groups
server11101
Global
Support
Overview
Upload
Create folder
Download
Actions
Feedback
English (US)
© 2008 - 2019, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

Amazon S3 › mailinput / direct

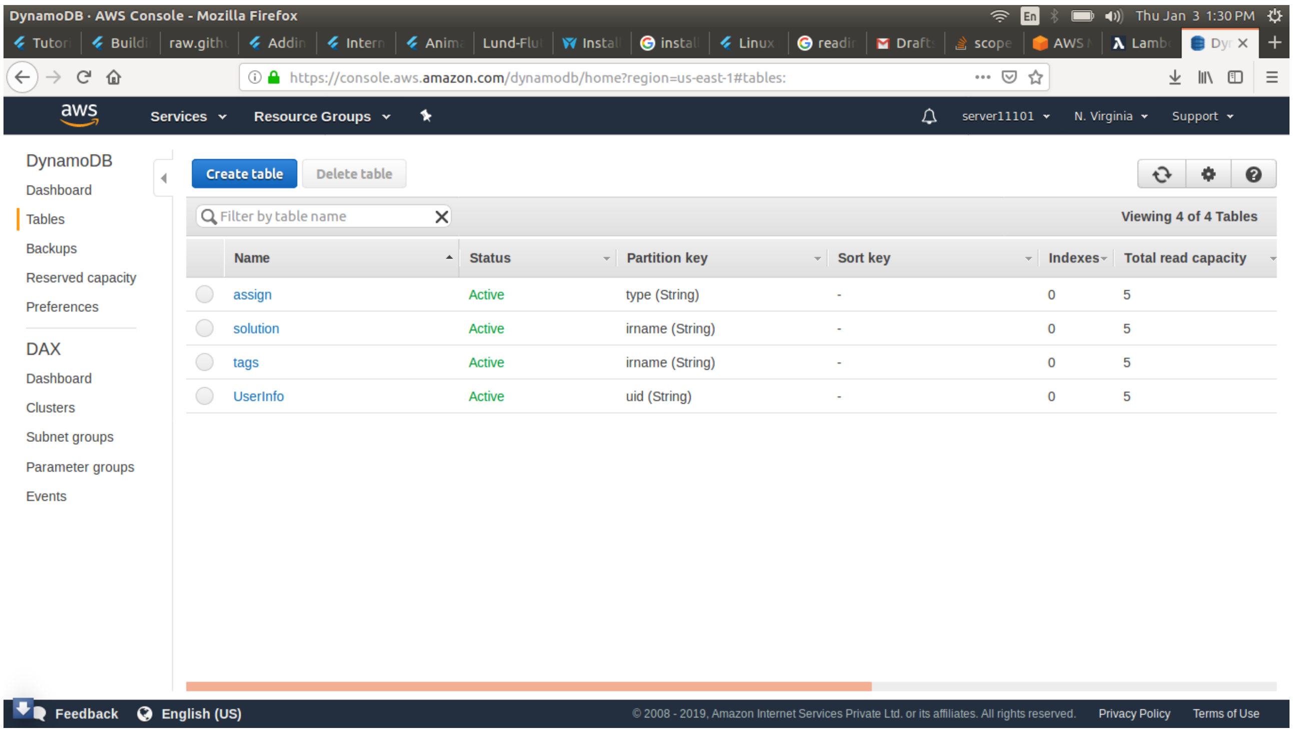
Type a prefix and press Enter to search. Press ESC to clear.

US East (N. Virginia)

Viewing 1 to 62

| Name | Last modified | Size | Storage class |
|---|---|---|---|
| 0ce5f7hpkbq4t094j5u64ut27fd8s0lrktbc9ug1 | Dec 31, 2018 2:17:13 PM GMT+0530 | 4.1 KB | Standard |
| 0gsv5gsnnnt249dnholfpon13ftas8glgoa8de01 | Dec 31, 2018 4:48:28 PM GMT+0530 | 4.2 KB | Standard |
| 11e46ie6vig0pm328qcfmmgg9imvd0oeqf358qg1 | Dec 29, 2018 7:33:24 PM GMT+0530 | 4.8 KB | Standard |
| 2qhibnjcpn37cbt3rivpqo6k8qjv6vq2a0399ag1 | Dec 31, 2018 4:41:22 PM GMT+0530 | 4.1 KB | Standard |
| 36cc9dhdr23lnckporqetpkt1rnr9hlp13umuk81 | Dec 30, 2018 11:32:14 AM GMT+0530 | 4.1 KB | Standard |

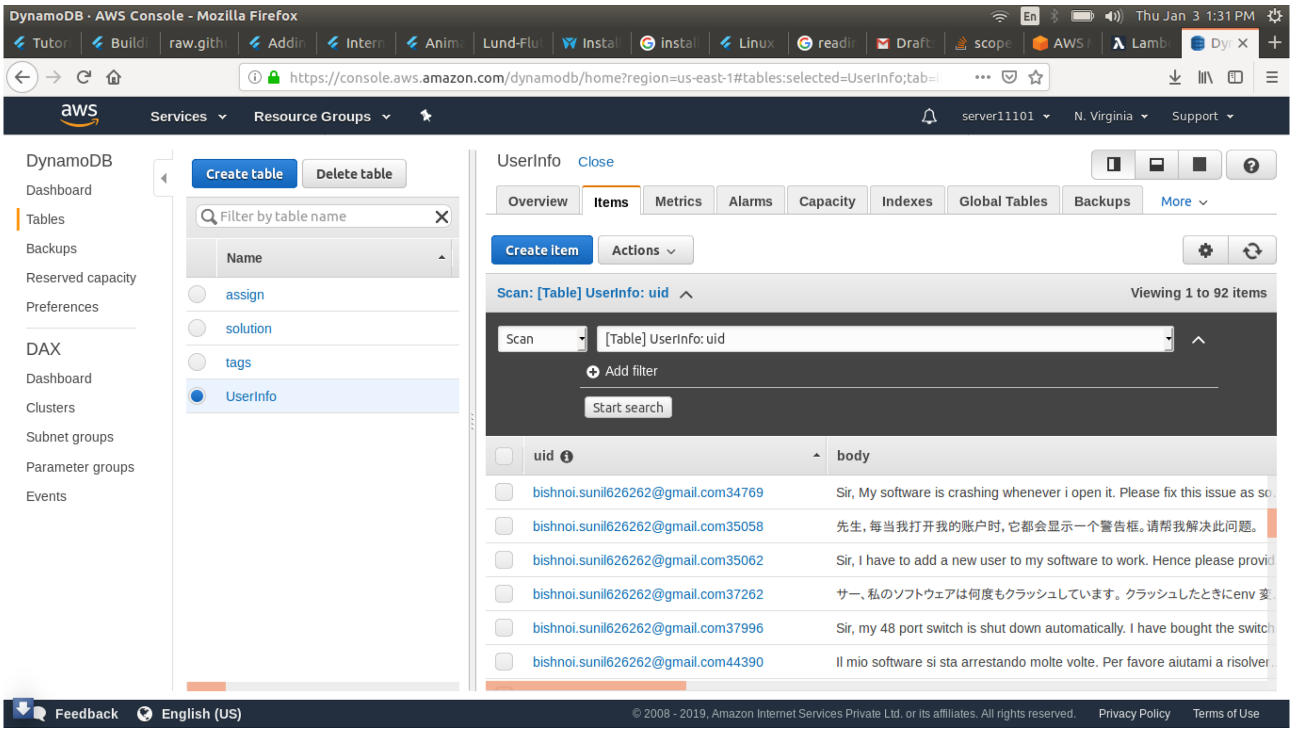
## Tables in dynmoDB



Viewing 4 of 4 Tables

| Name | Status | Partition key | Sort key | Indexes | Total read capacity |
|---|---|---|---|---|---|
| assign | Active | type (String) | - | 0 | 5 |
| solution | Active | irname (String) | - | 0 | 5 |
| tags | Active | irname (String) | - | 0 | 5 |
| UserInfo | Active | uid (String) | - | 0 | 5 |

## Table to store info about a query requested by a user



| uid | body |
|---|---|
| bishnoi.sunil626262@gmail.com34769 | Sir, My software is crashing whenever i open it. Please fix this issue as so |
| bishnoi.sunil626262@gmail.com35058 | 先生，每当我打开我的账户时，它都会显示一个警告框。请帮我解决此问题。 |
| bishnoi.sunil626262@gmail.com35062 | Sir, I have to add a new user to my software to work. Hence please provid |
| bishnoi.sunil626262@gmail.com37262 | サー、私のソフトウェアは何度もクラッシュしています。クラッシュしたときにenv 変 |
| bishnoi.sunil626262@gmail.com37996 | Sir, my 48 port switch is shut down automatically. I have bought the switch |
| bishnoi.sunil626262@gmail.com44390 | Il mio software si sta arrestando molte volte. Per favore aiutami a risolver |

## Information about solution table in dynmoDB

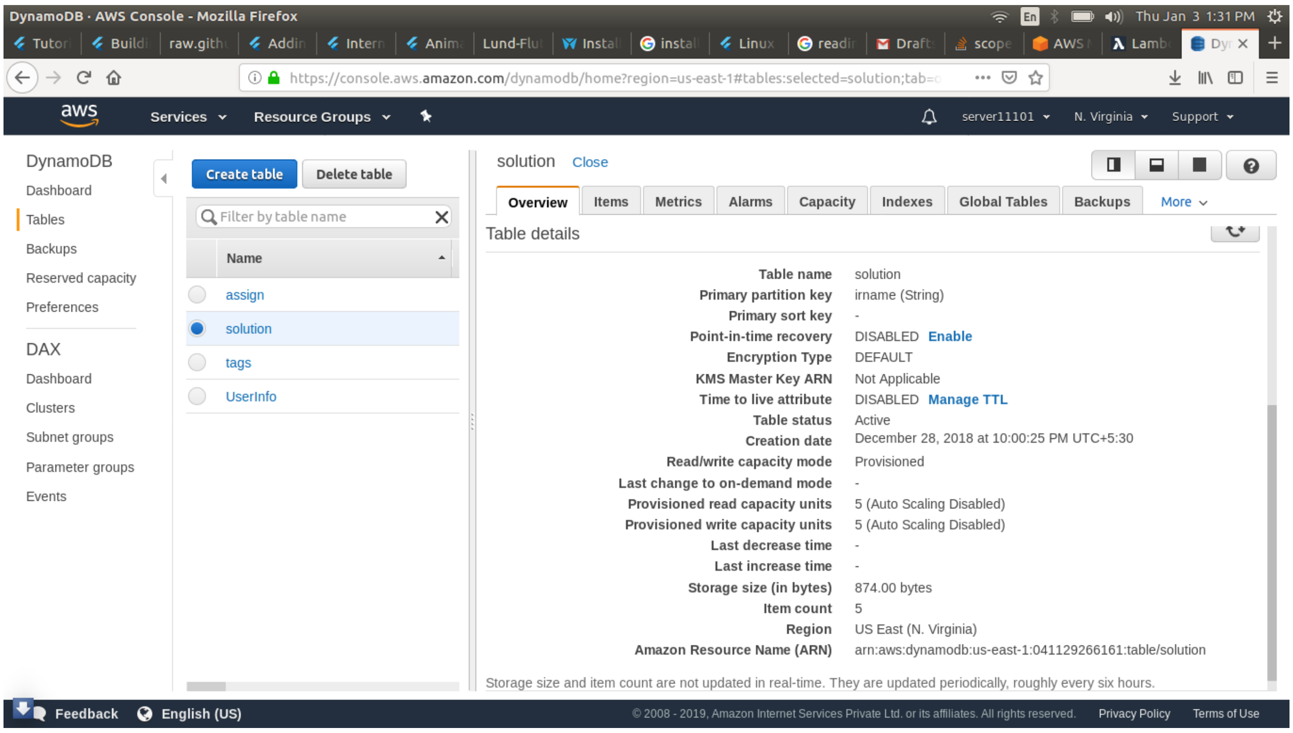
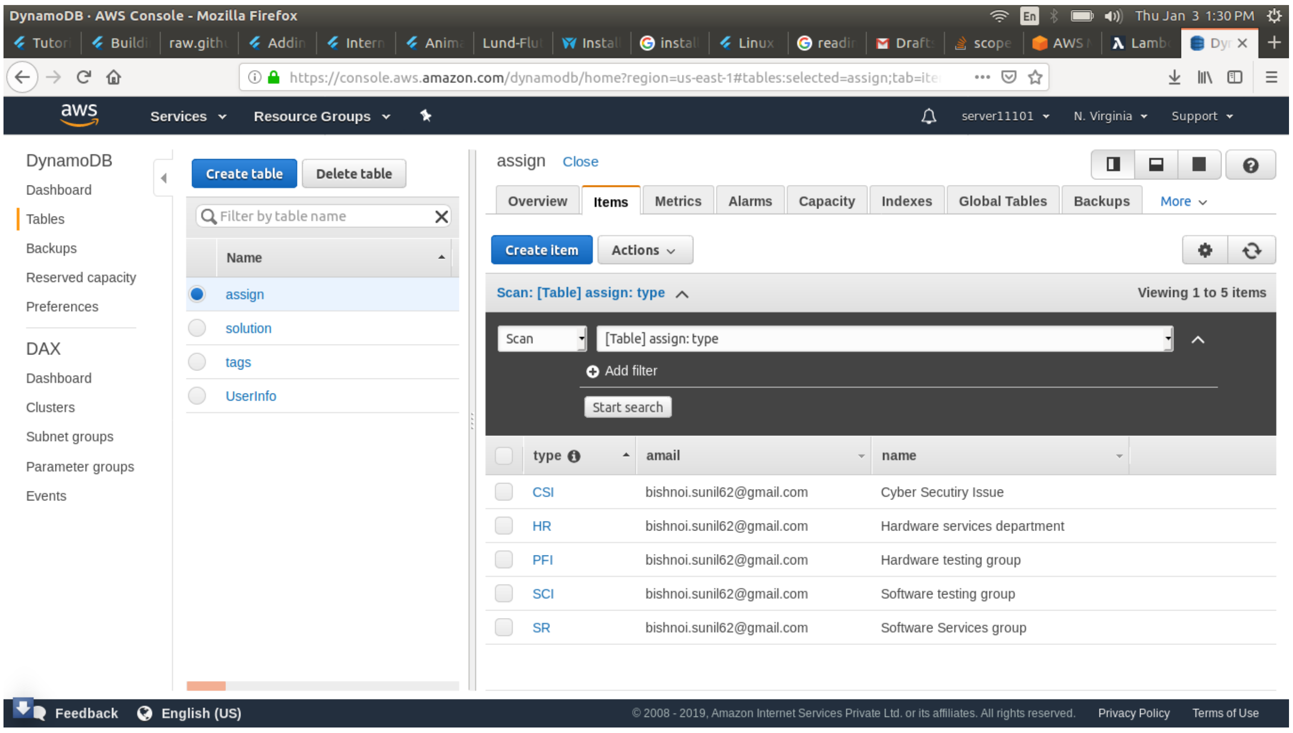


Table details

| | |
|---|---|
| Table name | solution |
| Primary partition key | irname (String) |
| Primary sort key | - |
| Point-in-time recovery | DISABLED Enable |
| Encryption Type | DEFAULT |
| KMS Master Key ARN | Not Applicable |
| Time to live attribute | DISABLED Manage TTL |
| Table status | Active |
| Creation date | December 28, 2018 at 10:00:25 PM UTC+5:30 |
| Read/write capacity mode | Provisioned |
| Last change to on-demand mode | - |
| Provisioned read capacity units | 5 (Auto Scaling Disabled) |
| Provisioned write capacity units | 5 (Auto Scaling Disabled) |
| Last decrease time | - |
| Last increase time | - |
| Storage size (in bytes) | 874.00 bytes |
| Item count | 5 |
| Region | US East (N. Virginia) |
| Amazon Resource Name (ARN) | arn:aws:dynamodb:us-east-1:041129266161:table/solution |

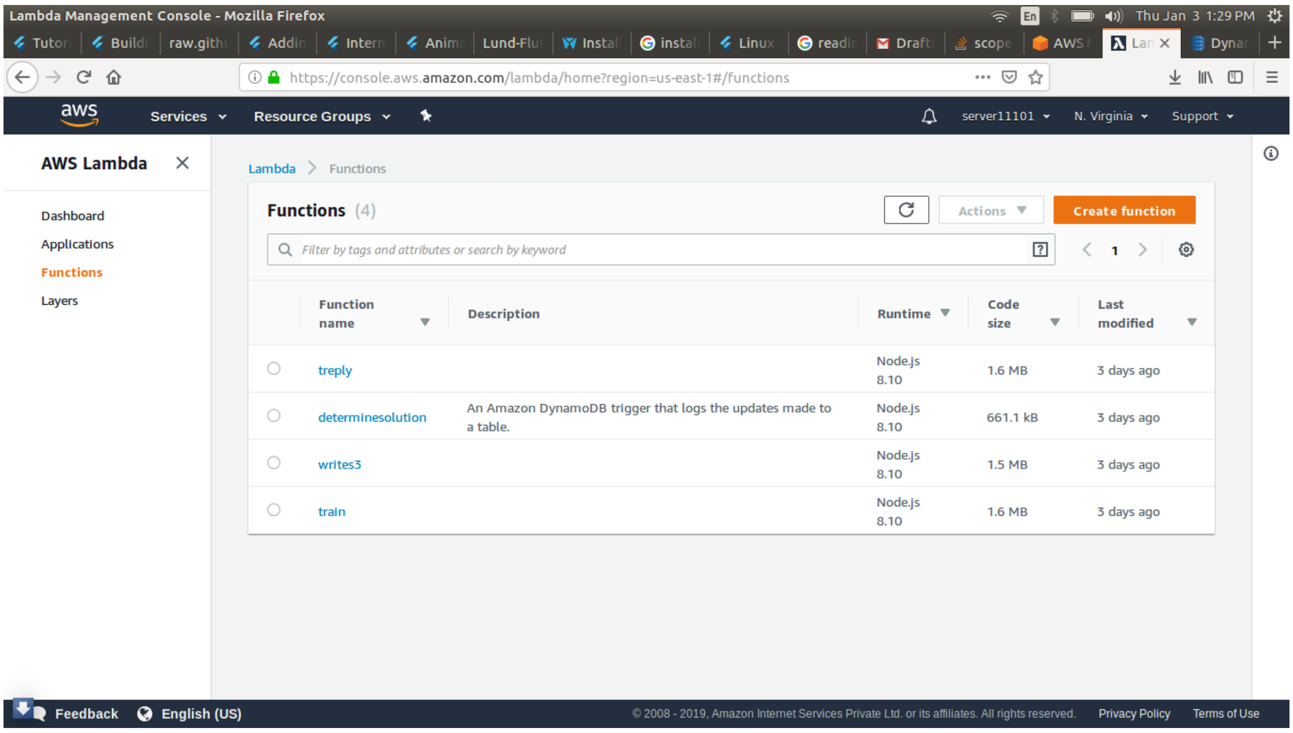
Storage size and item count are not updated in real-time. They are updated periodically, roughly every six hours.

## Assign table content in dynmoDB



| type | amail | name |
|---|---|---|
| CSI | bishnoi.sunil62@gmail.com | Cyber Secutiry Issue |
| HR | bishnoi.sunil62@gmail.com | Hardware services department |
| PFI | bishnoi.sunil62@gmail.com | Hardware testing group |
| SCI | bishnoi.sunil62@gmail.com | Software testing group |
| SR | bishnoi.sunil62@gmail.com | Software Services group |

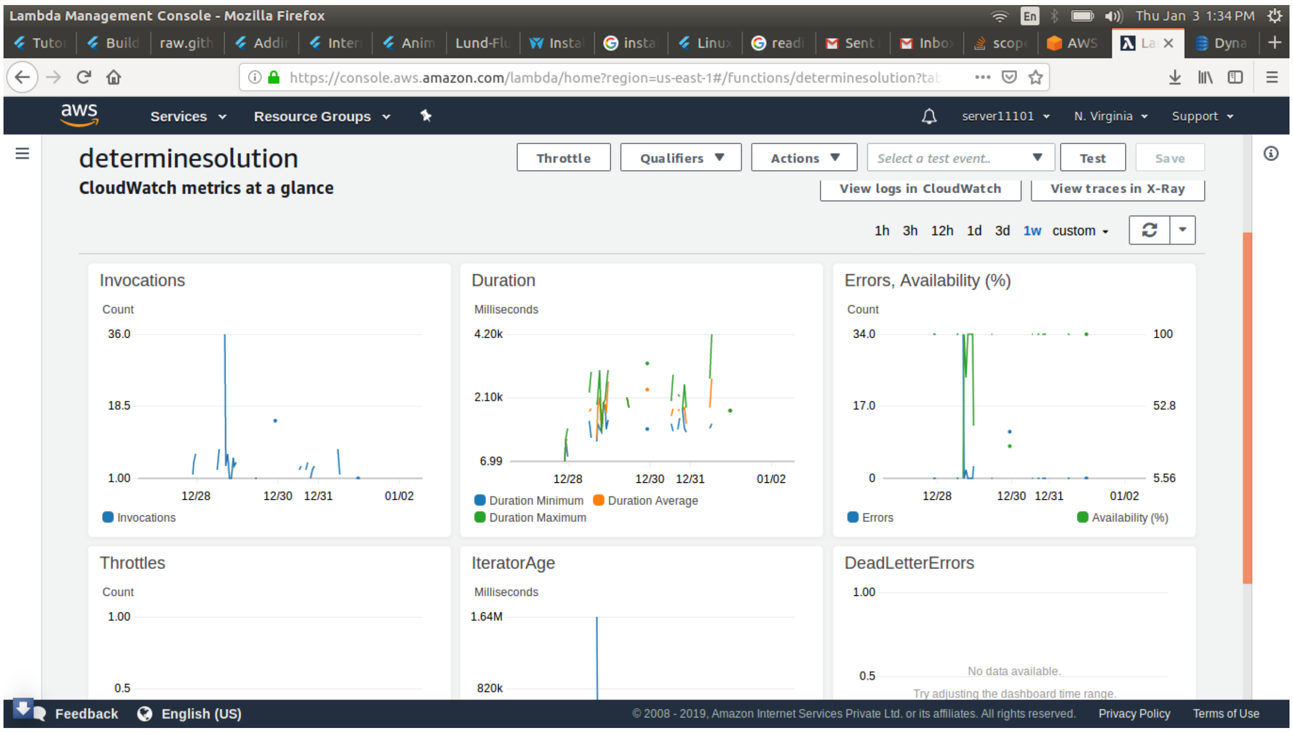
## Lambda functions


Lambda Management Console - Mozilla Firefox
Thu Jan 3 1:29 PM
https://console.aws.amazon.com/lambda/home?region=us-east-1#/functions
aws
Services
Resource Groups
server11101
N. Virginia
Support
AWS Lambda
Dashboard
Applications
Functions
Layers
Lambda > Functions
Actions
Create function
Filter by tags and attributes or search by keyword
Feedback
English (US)
© 2008 - 2019, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

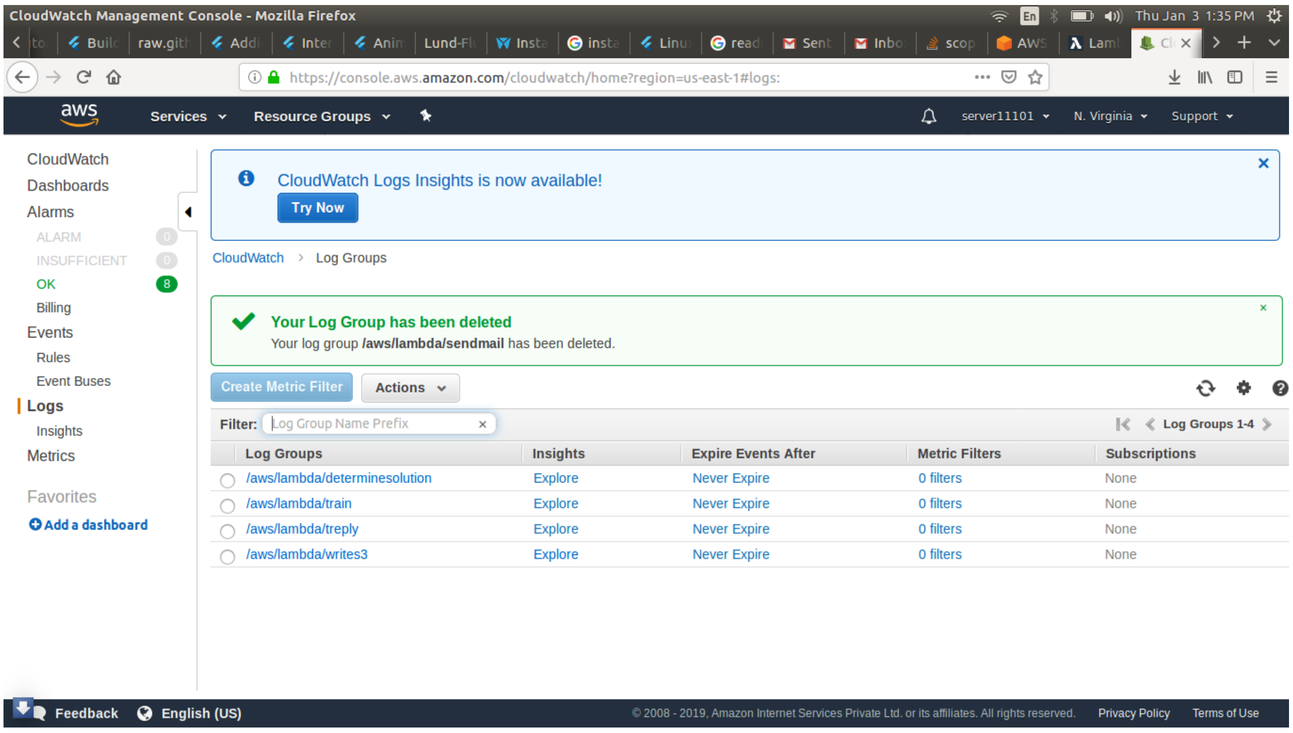
Functions (4)

| Function name | Description | Runtime | Code size | Last modified |
|---|---|---|---|---|
| treply | | Node.js 8.10 | 1.6 MB | 3 days ago |
| determinesolution | An Amazon DynamoDB trigger that logs the updates made to a table. | Node.js 8.10 | 661.1 kB | 3 days ago |
| writes3 | | Node.js 8.10 | 1.5 MB | 3 days ago |
| train | | Node.js 8.10 | 1.6 MB | 3 days ago |

## Invocation frequency and error frequency of determinesolution function



## Log streams for lambda functions



CloudWatch Logs Insights is now available! Try Now

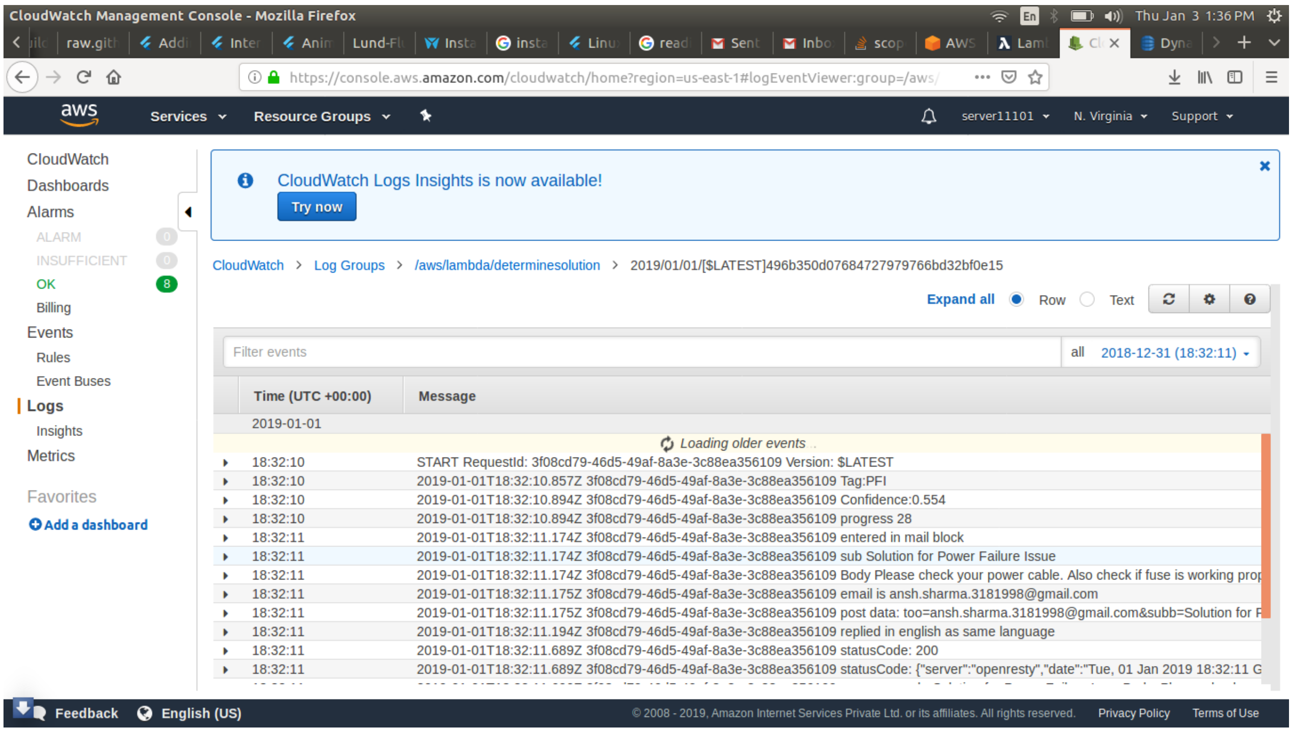
CloudWatch > Log Groups

**Your Log Group has been deleted**
Your log group **/aws/lambda/sendmail** has been deleted.

Filter: Log Group Name Prefix — Log Groups 1-4

| Log Groups | Insights | Expire Events After | Metric Filters | Subscriptions |
|---|---|---|---|---|
| /aws/lambda/determinesolution | Explore | Never Expire | 0 filters | None |
| /aws/lambda/train | Explore | Never Expire | 0 filters | None |
| /aws/lambda/treply | Explore | Never Expire | 0 filters | None |
| /aws/lambda/writes3 | Explore | Never Expire | 0 filters | None |

## Logs of a lambda function

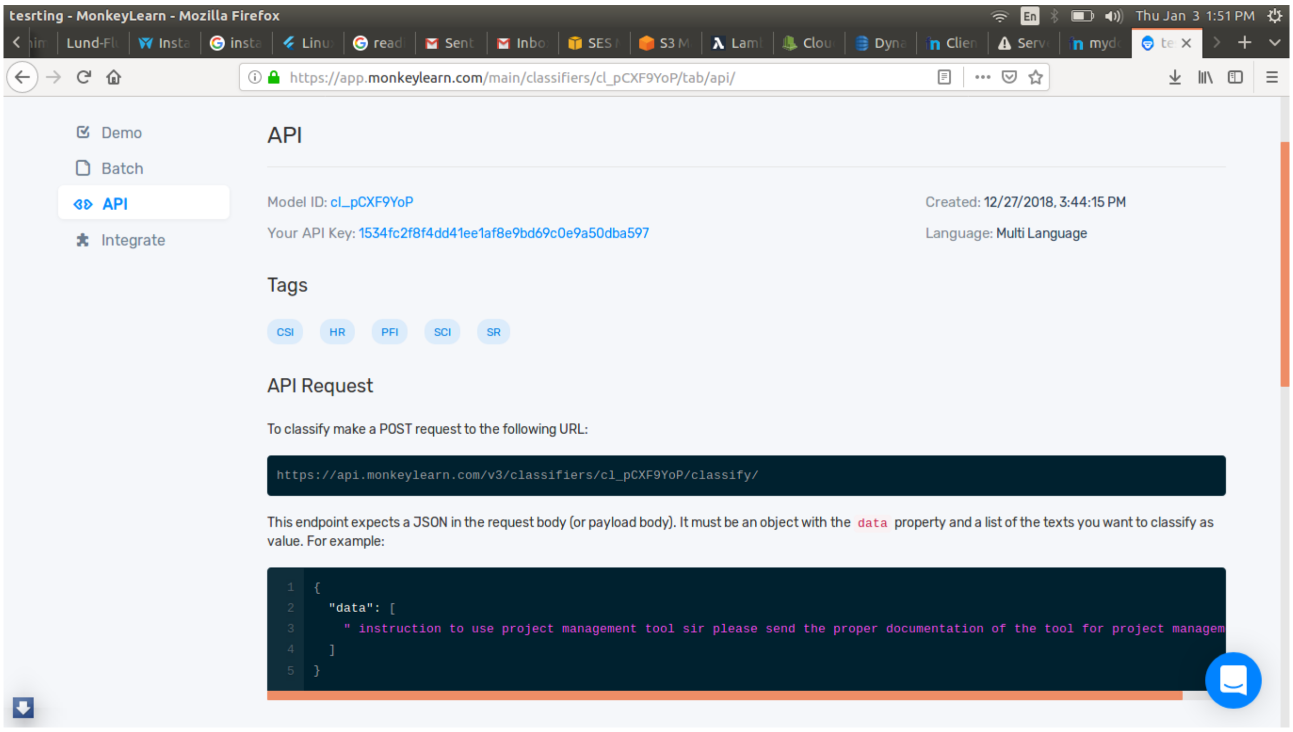


CloudWatch > Log Groups > /aws/lambda/determinesolution > 2019/01/01/[$LATEST]496b350d07684727979766bd32bf0e15

```
Time (UTC +00:00)   Message
2019-01-01
                    Loading older events ..
18:32:10   START RequestId: 3f08cd79-46d5-49af-8a3e-3c88ea356109 Version: $LATEST
18:32:10   2019-01-01T18:32:10.857Z 3f08cd79-46d5-49af-8a3e-3c88ea356109 Tag:PFI
18:32:10   2019-01-01T18:32:10.894Z 3f08cd79-46d5-49af-8a3e-3c88ea356109 Confidence:0.554
18:32:10   2019-01-01T18:32:10.894Z 3f08cd79-46d5-49af-8a3e-3c88ea356109 progress 28
18:32:11   2019-01-01T18:32:11.174Z 3f08cd79-46d5-49af-8a3e-3c88ea356109 entered in mail block
18:32:11   2019-01-01T18:32:11.174Z 3f08cd79-46d5-49af-8a3e-3c88ea356109 sub Solution for Power Failure Issue
18:32:11   2019-01-01T18:32:11.174Z 3f08cd79-46d5-49af-8a3e-3c88ea356109 Body Please check your power cable. Also check if fuse is working prop
18:32:11   2019-01-01T18:32:11.175Z 3f08cd79-46d5-49af-8a3e-3c88ea356109 email is ansh.sharma.3181998@gmail.com
18:32:11   2019-01-01T18:32:11.175Z 3f08cd79-46d5-49af-8a3e-3c88ea356109 post data: too=ansh.sharma.3181998@gmail.com&subb=Solution for F
18:32:11   2019-01-01T18:32:11.194Z 3f08cd79-46d5-49af-8a3e-3c88ea356109 replied in english as same language
18:32:11   2019-01-01T18:32:11.689Z 3f08cd79-46d5-49af-8a3e-3c88ea356109 statusCode: 200
18:32:11   2019-01-01T18:32:11.689Z 3f08cd79-46d5-49af-8a3e-3c88ea356109 statusCode: {"server":"openresty","date":"Tue, 01 Jan 2019 18:32:11 G
```

**Classifier model on monkeylearn cloud platform**

We have used monkeylearn cloud platform for classification ML model because it has limited free usage access and options for different models with different algorithms. We have used SVM(Support Vector Machine) algorithm in our classification model.



tesrting - MonkeyLearn - Mozilla Firefox
https://app.monkeylearn.com/main/classifiers/cl_pCXF9YoP/tab/api/
Demo
Batch
API
Integrate

API

Model ID: cl_pCXF9YoP

Your API Key: 1534fc2f8f4dd41ee1af8e9bd69c0e9a50dba597

Created: 12/27/2018, 3:44:15 PM

Language: Multi Language

Tags

CSI HR PFI SCI SR

API Request

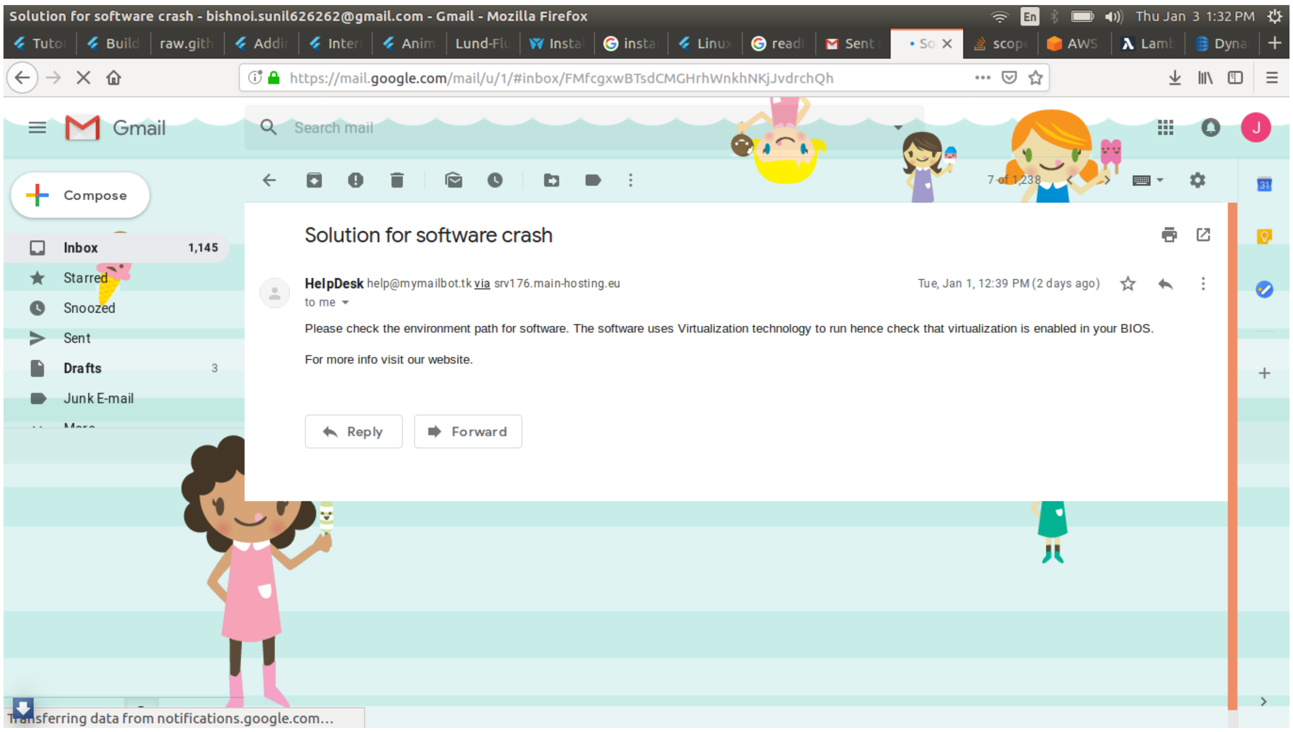
To classify make a POST request to the following URL:

```
https://api.monkeylearn.com/v3/classifiers/cl_pCXF9YoP/classify/
```

This endpoint expects a JSON in the request body (or payload body). It must be an object with the `data` property and a list of the texts you want to classify as value. For example:

```
{
  "data": [
    " instruction to use project management tool sir please send the proper documentation of the tool for project managem
  ]
}
```

**An email from bot using solution from knowledge base**



Solution for software crash - bishnoi.sunil626262@gmail.com - Gmail - Mozilla Firefox
https://mail.google.com/mail/u/1/#inbox/FMfcgxwBTsdCMGHrhWnkhNKjJvdrchQh
Gmail
Search mail
Compose
Inbox 1,145
Starred
Snoozed
Sent
Drafts 3
Junk E-mail
7 of 1,238
Reply
Forward
Transferring data from notifications.google.com...

Solution for software crash

HelpDesk help@mymailbot.tk via srv176.main-hosting.eu

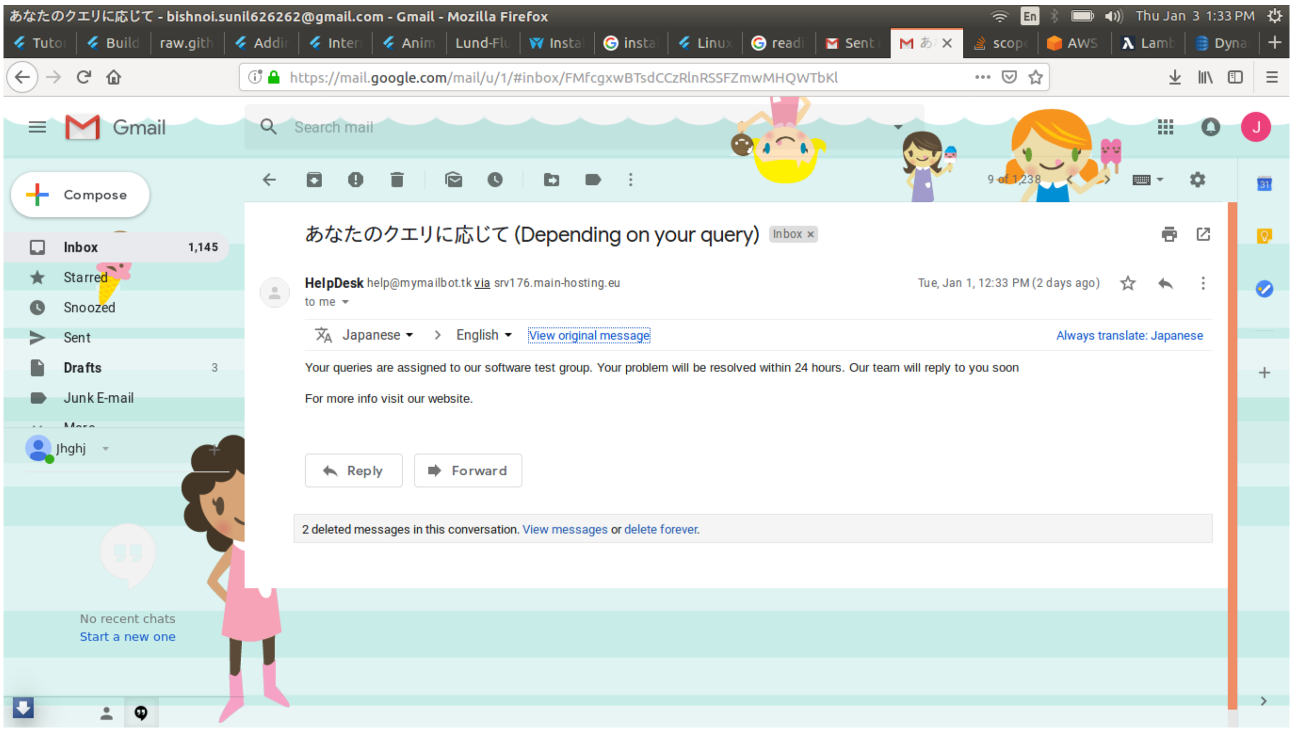
to me

Tue, Jan 1, 12:39 PM (2 days ago)

Please check the environment path for software. The software uses Virtualization technology to run hence check that virtualization is enabled in your BIOS.

For more info visit our website.

## Email from server if query is assigned to a resolving group



あなたのクエリに応じて (Depending on your query) Inbox

HelpDesk help@mymailbot.tk via srv176.main-hosting.eu Tue, Jan 1, 12:33 PM (2 days ago)
to me

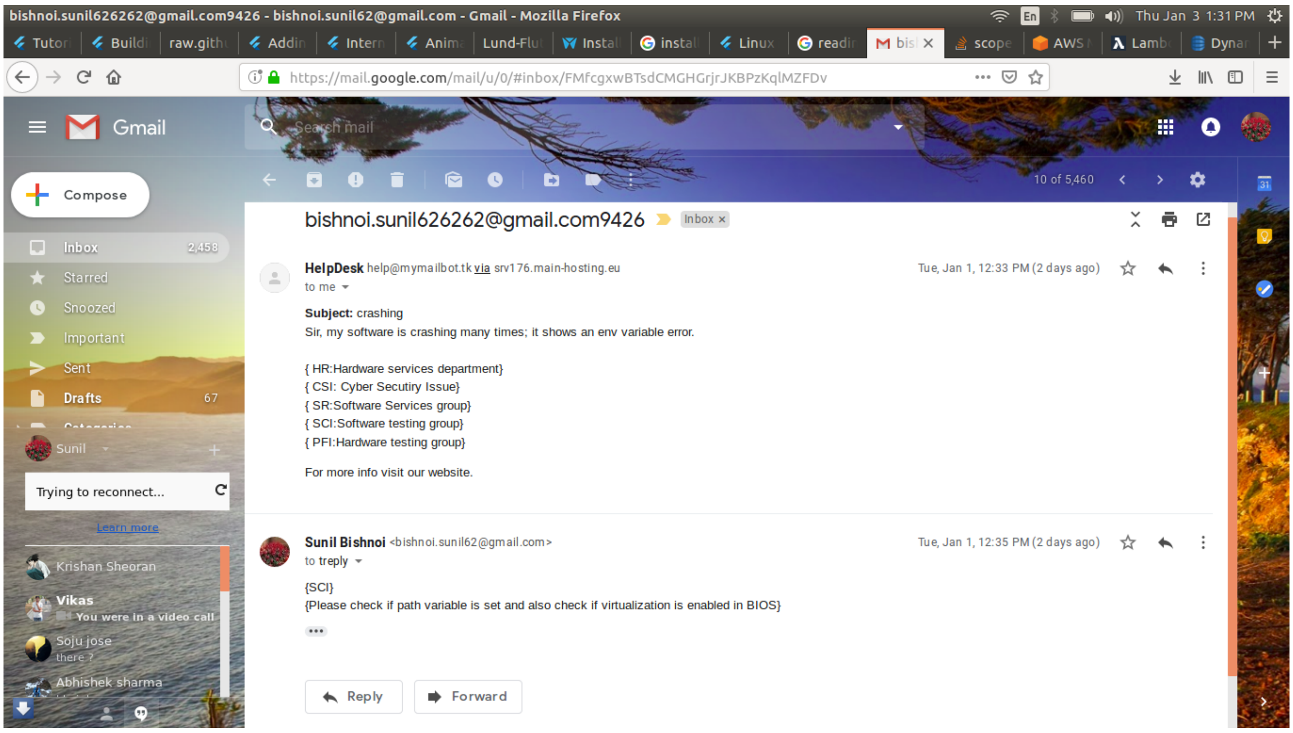
Japanese > English View original message Always translate: Japanese

Your queries are assigned to our software test group. Your problem will be resolved within 24 hours. Our team will reply to you soon

For more info visit our website.

2 deleted messages in this conversation. View messages or delete forever.

## An email from server to resolving group and reply of resolving group in a defined format



bishnoi.sunil626262@gmail.com9426 Inbox

HelpDesk help@mymailbot.tk via srv176.main-hosting.eu Tue, Jan 1, 12:33 PM (2 days ago)
to me

**Subject:** crashing
Sir, my software is crashing many times; it shows an env variable error.

{ HR:Hardware services department}
{ CSI: Cyber Secutiry Issue}
{ SR:Software Services group}
{ SCI:Software testing group}
{ PFI:Hardware testing group}

For more info visit our website.

Sunil Bishnoi <bishnoi.sunil62@gmail.com> Tue, Jan 1, 12:35 PM (2 days ago)
to treply

{SCI}
{Please check if path variable is set and also check if virtualization is enabled in BIOS}

**The resolving group receive mail with all defined tags with definition as well as query of the user in mail. If the query belong to defined group then resolving group send that tag in one braces and solution in other braces. If the query does not belong to any tag defined then user create a new tag with its definition in one braces and solution in other braces. The server receive the mail from the resolving group and process it. If the new tag is given it will add that tag in ML classification Model and train it with new problem statement corresponding to that tag. It also save that tag in database with its solution.**
**If the tag is existing tag then it will train the model with the problem statement. Hence next time it will identify that problem and send the appropriate solution to user rather that assigning the query to a resolving group.**

**Thank You**