ИТОГОВЫЙ ПРОЕКТ

Serverless Telegram Bot on AWS Lambda

Жувака Юлия, EPAM University Programm

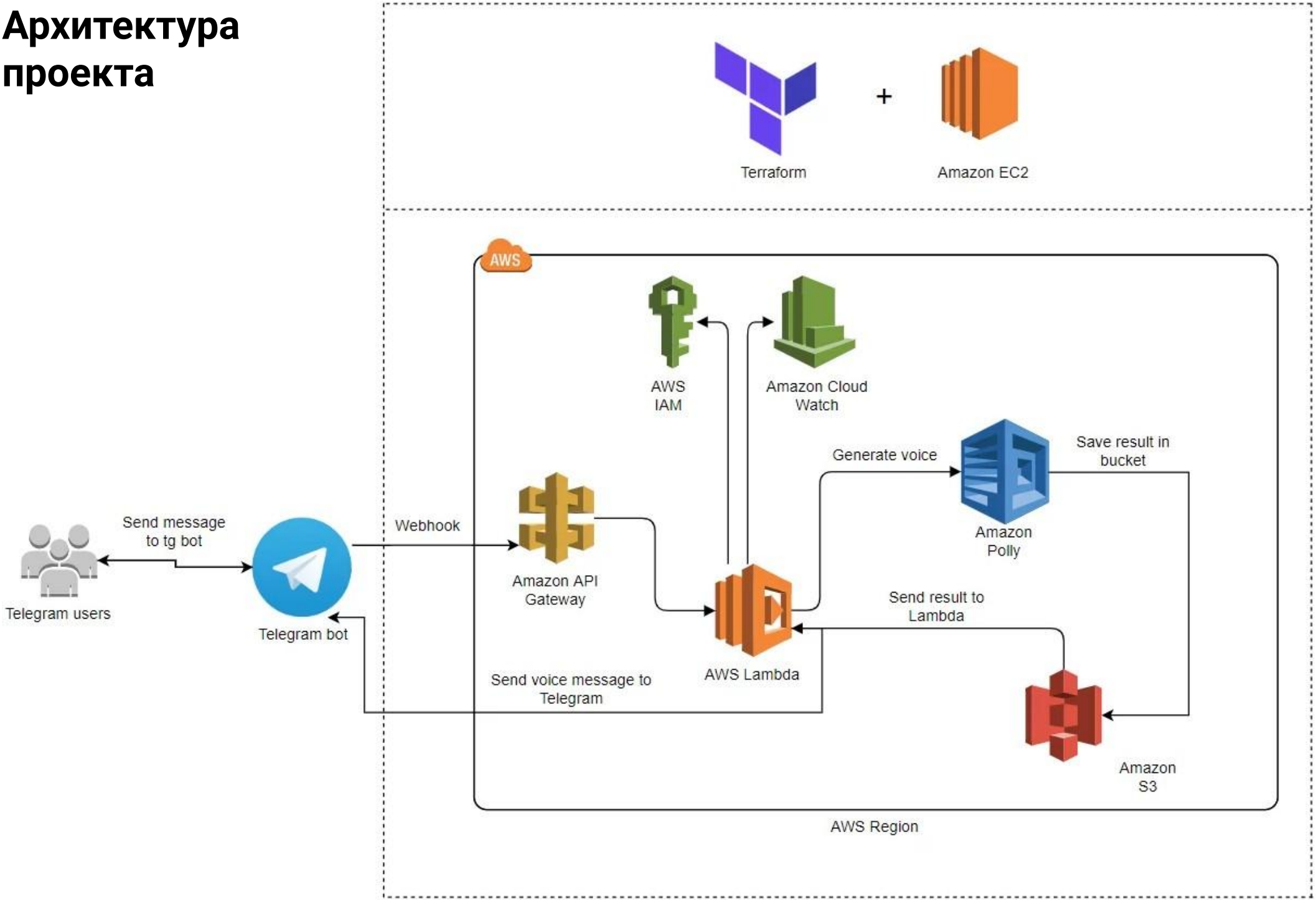
Технологии и инструменты, которые были использованы:

- Telegram
- Webhook
- API Gateway
- Lambda Function
  - Python
  - Environment variables
  - IAM
  - CloudWatch
- Polly
- S3

- EC2
- Terraform

@zhuvaka_projectbot

Архитектура проекта



Шаг 1.

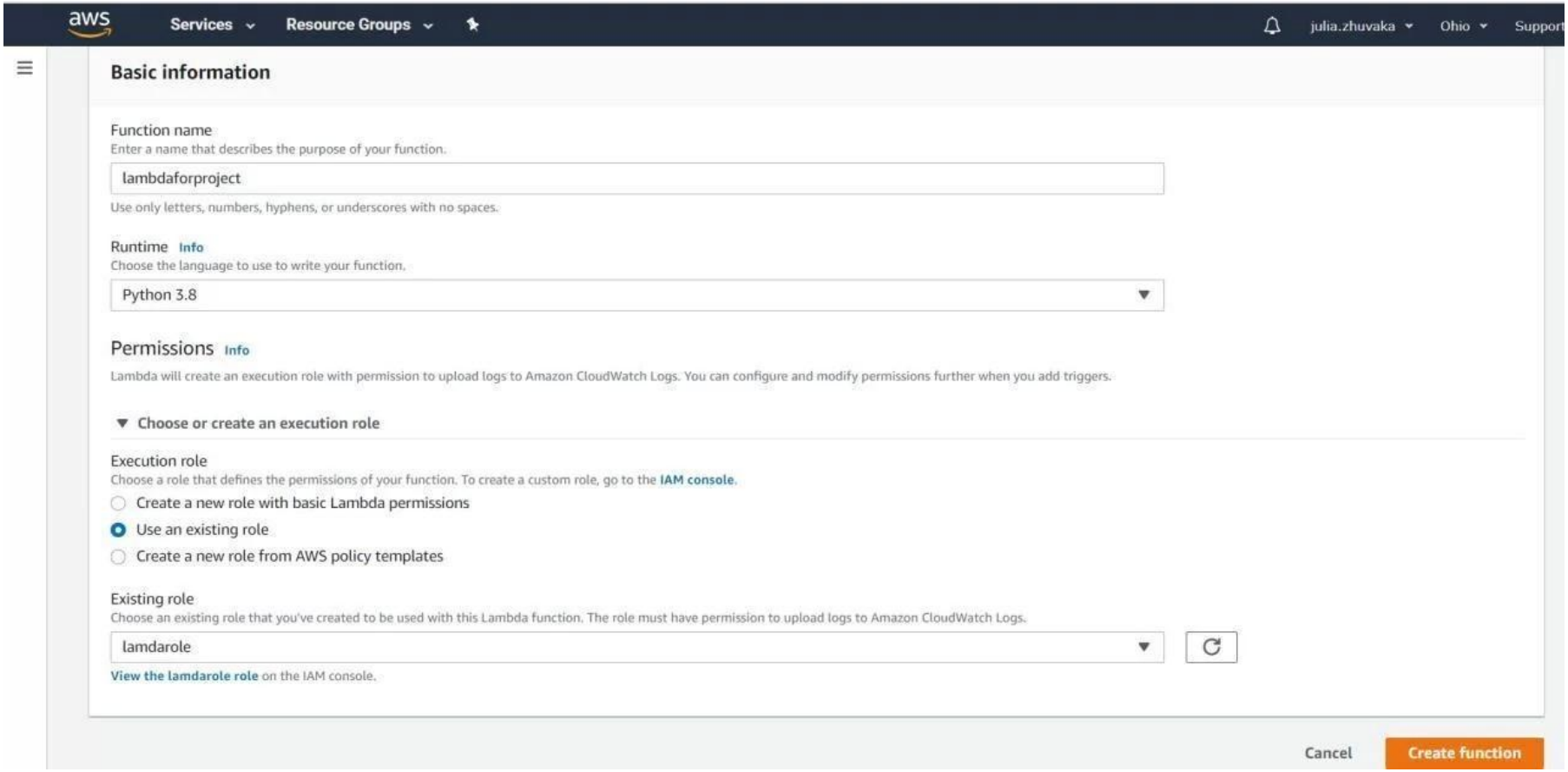
Создаем телеграм-бота

Шаг 2.

Создаем Lambda-функцию.

Язык программирования — Python,

роль и разрешения должны позволять функции взаимодействовать с другими сервисами Amazon



✦ ✦ ✦

Services Resource Groups

Gateway APIs > projectapi (ibltqpef8i) > Resources > / (0jtemzn3c3)

Resources Actions / Methods

Names

/

RESOURCE ACTIONS
- Create Method
- Create Resource
- Enable CORS
- Edit Resource Documentation

API ACTIONS
- Deploy API
- Import API
- Edit API Documentation
- Delete API

onses

cy

es

ojectapi (ibltqpef8i) > Resources > / (0jtemzn3c3) > ANY Show all hints

Actions / - ANY - Setup

Choose the integration point for your new method.

Integration type: Lambda Function, HTTP, Mock, AWS Service, VPC Link

Use Lambda Proxy integration

Lambda Region: us-east-2

Lambda Function: lambdaforproject

Use Default Timeout

ojectapi (ibltqpef8i) > Resources > / (0jtemzn3c3) > ANY Show all hints

Actions / - ANY - Method Execution

TEST

Client

Method Request
Auth: NONE
ARN: arn:aws:execute-api:us-east-2:497545553828:ibltqpef8i/*/*/

Integration Request
Type: LAMBDA
Region: us-east-2

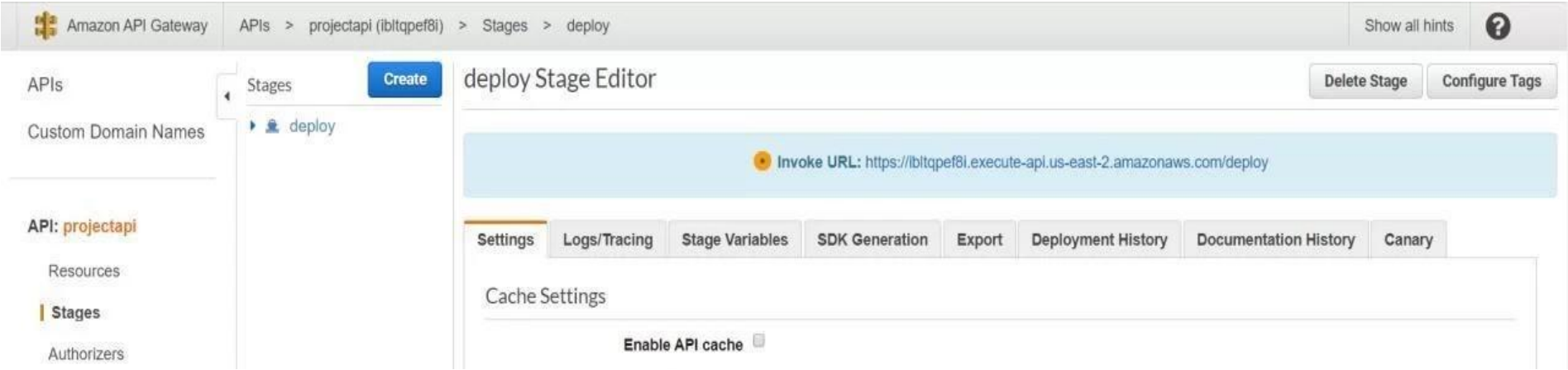
Method Response
HTTP Status: 200
Models: application/json => Empty

Integration Response
HTTP status pattern: -
Output passthrough: Yes

Шаг 3.

Создаем API, связываем с Lambda-функцией.

Получаем ссылку от API Gateway, которая вызывает Лямбду

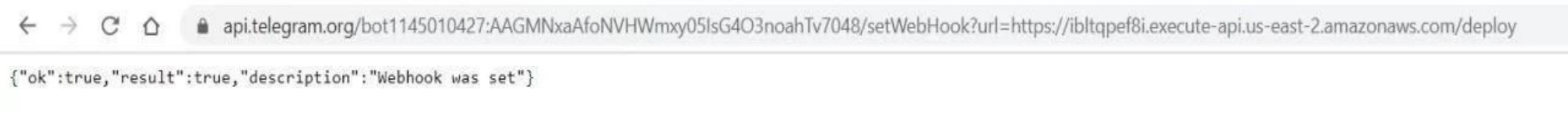


✦ ✦ ✦

Шаг 4. Подключаем Webhook

Webhook имеет следующий вид:

https://api.telegram.org/bot{TOKEN}/setWebHook?url={ССЫЛКА, КОТОРУЮ ПОЛУЧИЛИ ОТ API - GATEWAY ПОСЛЕ ДЕПЛОЯ}



Шаг 5. Код на Python для выполнения функции

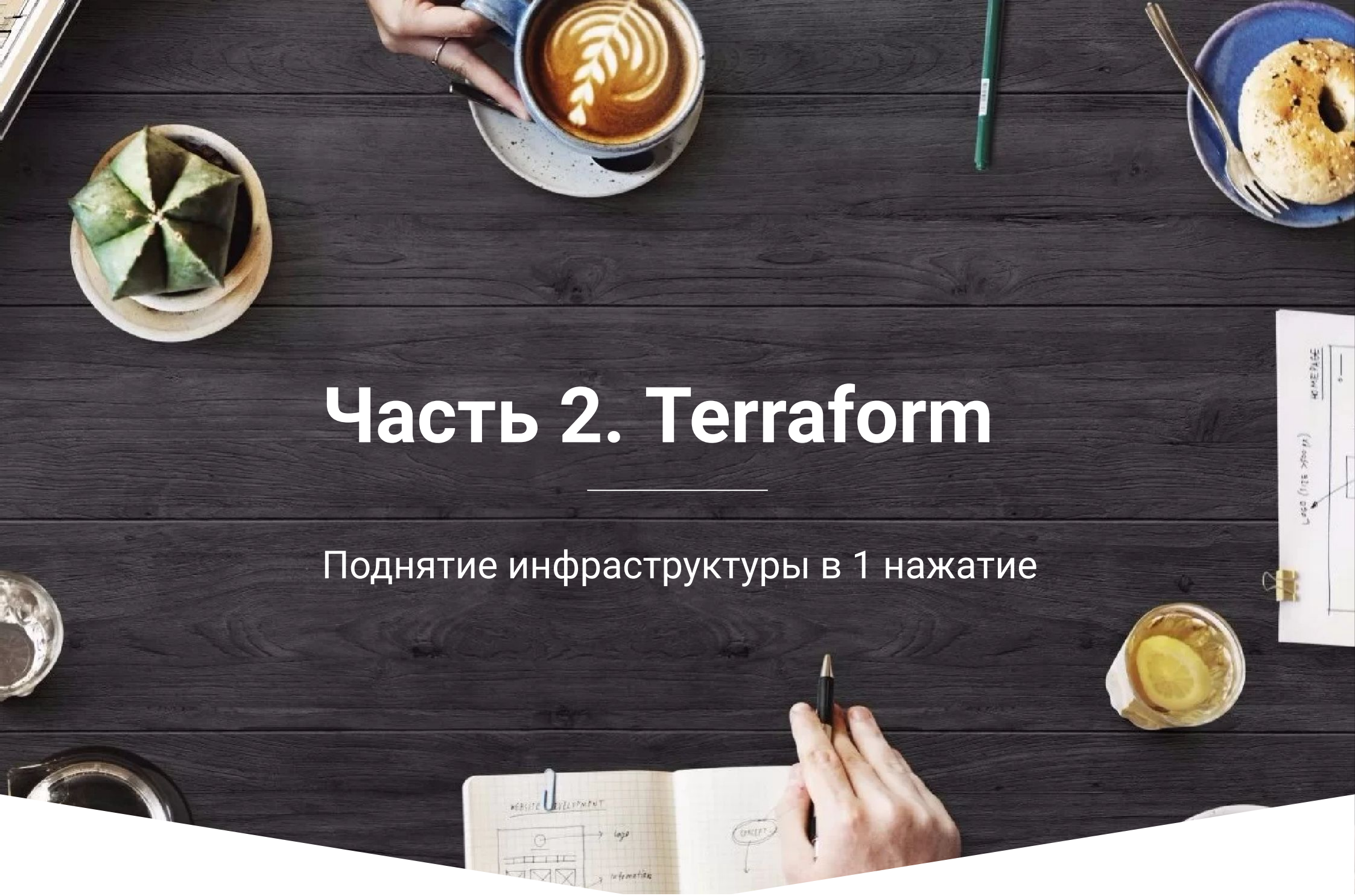
```
def lambda_handler(event, context):
        message = json.loads(event['body'])['message']
        chat_id = message['chat']['id']
        reply = message['text']
        command = reply.split()[0][1:]
        filename = '_'.join(reply.split()[:3]) + ".ogg"
        audio_link = https://{}.s3.{}.amazonaws.com/{}".format(os.environ['S3_BUCKET_NAME'],
        os.environ['REGION'], filename)
        if command=="start":
            send_message(chat_id)
        else:
            generate_voice(reply, filename)
            send_audio(chat_id, audio_link, filename)
        return {
            'statusCode': 200
        }
```

```
def generate_voice(text, filename):
    polly_client = boto3.client('polly')
    response = polly_client.synthesize_speech(VoiceId=os.environ['VOICE_NAME']
                        OutputFormat='ogg_vorbis',
                        Text = text)

    with closing(response["AudioStream"]) as stream:
                output = os.path.join("/tmp/", filename)
                with open(output, "wb") as file:  file.write(stream.read())
s3.upload_file('/tmp/' + filename,  os.environ['S3_BUCKET_NAME']
  filename)
```

```
def send_audio(chat_id, link, filename):
    url = URL + "sendVoice?voice={}&chat_id={}".format(link, chat_id)
    requests.get(url)

    s3.delete_object(
        Bucket=os.environ['S3_BUCKET_NAME'],
        Key=filename
        )
```



Часть 2. Terraform

Поднятие инфраструктуры в 1 нажатие

Для поднятия инфраструктуры нам понадобятся

01

API Gateway

Файл, в котором описано создание нового API,, присваивается метод, связь с Lambda-функцией, деплой, настройка веб-хука

02

Lambda function

В данном документе создаем роль и policy для взаимодействия функции с другими сервисами AWS, указываем где лежит код для лямбды, создаем саму функцию, добавляем Environment variables и добавляем триггер на срабатывание - API Gateway

03

S3 bucket

Создаем корзину в S3 для загрузки аудио файлов, а также добавляем policy, которая делает все объекты в корзине публичными

API Gateway

```
resource "aws_api_gateway_rest_api" "terraform_api" {
  name= "terraform_api_tg_bot"
  description = "Terraform Serverless Application"
}


resource "aws_api_gateway_method" "proxy" {

  rest_api_id     = aws_api_gateway_rest_api.terraform_api.id
  resource_id     = aws_api_gateway_resource.proxy.id
  http_method     = "ANY"

   authorization = "NONE"
 }

 resource "aws_api_gateway_integration" "lambda" {
   rest_api_id = aws_api_gateway_rest_api.terraform_api.id
   resource_id = aws_api_gateway_method.proxy.resource_id
   http_method = aws_api_gateway_method.proxy.http_method

   integration_http_method = "POST"
   type= "AWS_PROXY"
   uri = aws_lambda_function.terraform_lambda.invoke_arn
   }
```

```
resource "aws_api_gateway_deployment" "terraform_api" {
  depends_on = [
    aws_api_gateway_integration.lambda,
    aws_api_gateway_integration.lambda_root,
  ]

  rest_api_id = aws_api_gateway_rest_api.terraform_api.id
  stage_name = "test"
}

 data "http" "webhook" {
   url =
   "https://api.telegram.org/bot${var.bot_token}/setWebHook?url=${aws_api_gateway_deployment.terraform_api
 }
```

Lambda Function

```
resource "aws_iam_role" "iam_for_lambda" {
  name = "iam_for_lambda"
  assume_role_policy = <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "sts:AssumeRole",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Effect": "Allow",
      "Sid": ""
    }
  ]
}
EOF
}


resource "aws_iam_policy" "policy" {
  name        = "policy"
  policy      = <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "*",
            "Resource": "*"
```

```
resource "aws_iam_policy_attachment" "test-attach" {

 name       = "test-attachment"
 roles      = ["${aws_iam_role.iam_for_lambda.name}"]
 policy_arn = "${aws_iam_policy.policy.arn}"
}

      environment {
        variables = {
          BOT_ID = "bot token"
         VOICE_NAME = "Maxim"
         S3_BUCKET_NAME = "zhuvakatelegram"
         START_MESSAGE = "some text”
         REGION = "us-east-2"
        }
      }

resource "aws_lambda_function" "terraform_lambda" {
  s3_bucket = "zhuvakabot"
  s3_key    = "lambda.zip"
  function_name = "terraform_lambda_tg_bot"
  role= "${aws_iam_role.iam_for_lambda.arn}"  handler
  = "lambda_function.lambda_handler"
  runtime = "python3.8"

}

resource "aws_lambda_permission" "apigw" {
   statement_id  =  "AllowAPIGatewayInvoke"
   action = "lambda:InvokeFunction"
   function_name = aws_lambda_function.terraform_lambda.function_name
   principal = "apigateway.amazonaws.com"

   source_arn = "${aws_api_gateway_rest_api.terraform_api.execution_arn}/*/*"
 }
```

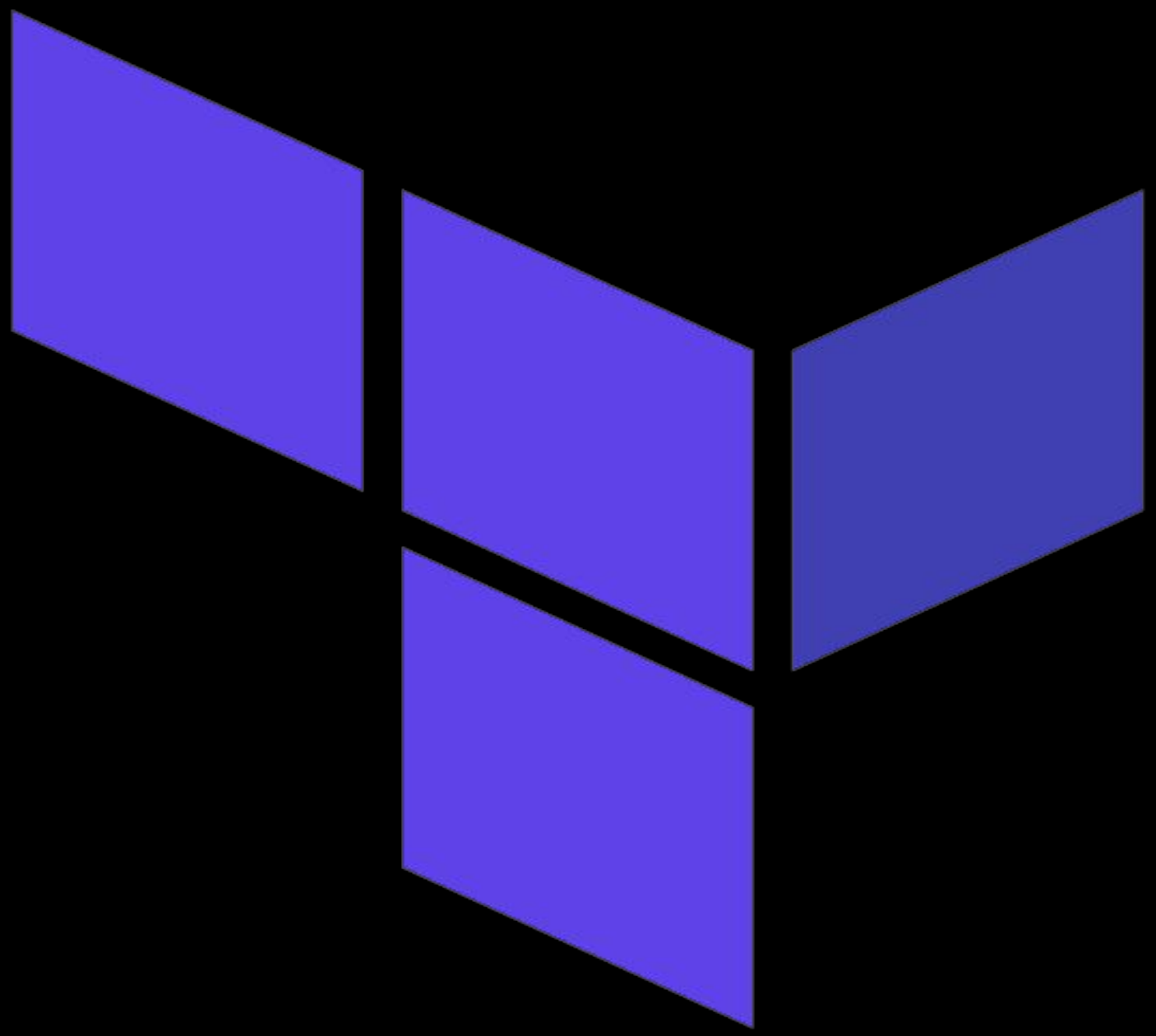
S3 Bucket

```
resource "aws_s3_bucket" "zhuvakayuliiatg" {
  bucket = "zhuvakatelegram"
}

resource "aws_s3_bucket_policy" "zhuvakayuliiatg" {
  bucket = "${aws_s3_bucket.zhuvakayuliiatg.id}"

  policy = <<POLICY
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Action": "s3:GetObject",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::zhuvakatelegram/*",
      "Principal": "*"
    }
  ]
}
POLICY
}
```

Видео Terraform




Q&A

Код бота

```
import boto3
import os
import json
import requests
from contextlib import closing


TELE_TOKEN=os.environ['BOT_ID']
URL = "https://api.telegram.org/bot{}/".format(TELE_TOKEN)

s3 = boto3.client('s3')

def generate_voice(text, filename):
    polly_client = boto3.client('polly')
    response =
polly_client.synthesize_speech(VoiceId=os.environ['VOICE_N
AME'],
                    OutputFormat='ogg_vorbis',
                    Text = text)

    with closing(response["AudioStream"]) as stream:
                output = os.path.join("/tmp/", filename)
                with open(output, "wb") as file:
                    file.write(stream.read())


    s3.upload_file('/tmp/' + filename,
      os.environ['S3_BUCKET_NAME'],
      filename)


def send_audio(chat_id, link, filename):
    url = URL +
"sendVoice?voice={}&chat_id={}".format(link, chat_id)
    requests.get(url)

    s3.delete_object(
        Bucket=os.environ['S3_BUCKET_NAME'],
        Key=filename
        )
```

```
def send_message(chat_id):
    final_text = os.environ['START_MESSAGE']
    url = URL +
"sendMessage?text={}&chat_id={}".format(final_text,
chat_id)
    requests.get(url)

def lambda_handler(event, context):
        message = json.loads(event['body'])['message']
        chat_id = message['chat']['id']
        reply = message['text']
        command = reply.split()[0][1:]
        filename = '_'.join(reply.split()[:3]) + ".ogg"
        audio_link =
"https://{}.s3.{}.amazonaws.com/{}".format(os.environ['S
3_BUCKET_NAME'], os.environ['REGION'], filename)
        if command=="start":
            send_message(chat_id)
        else:
            generate_voice(reply, filename)
            send_audio(chat_id, audio_link, filename)
        return {
            'statusCode': 200
        }
```

Код API.tf

```
 provider "aws" {
   region = "us-east-2"
 }

resource "aws_api_gateway_rest_api" "terraform_api" {
  name        = "terraform_api_tg_bot"
  description = "Terraform Serverless Application"
}

 resource "aws_api_gateway_resource" "proxy" {
   rest_api_id = aws_api_gateway_rest_api.terraform_api.id
   parent_id   =
aws_api_gateway_rest_api.terraform_api.root_resource_id
   path_part   = "{proxy+}"
}

resource "aws_api_gateway_method" "proxy" {
   rest_api_id   =
aws_api_gateway_rest_api.terraform_api.id
   resource_id   = aws_api_gateway_resource.proxy.id
   http_method   = "ANY"
   authorization = "NONE"
 }

 resource "aws_api_gateway_integration" "lambda" {
   rest_api_id = aws_api_gateway_rest_api.terraform_api.id
   resource_id = aws_api_gateway_method.proxy.resource_id
   http_method = aws_api_gateway_method.proxy.http_method

   integration_http_method = "POST"
   type                    = "AWS_PROXY"
   uri                     =
aws_lambda_function.terraform_lambda.invoke_arn
 }
```

```
  resource "aws_api_gateway_method" "proxy_root" {
   rest_api_id   =
aws_api_gateway_rest_api.terraform_api.id
   resource_id   =
aws_api_gateway_rest_api.terraform_api.root_resource_id
   http_method   = "ANY"
   authorization = "NONE"
 }

 resource "aws_api_gateway_integration" "lambda_root" {
   rest_api_id =
aws_api_gateway_rest_api.terraform_api.id
   resource_id =
aws_api_gateway_method.proxy_root.resource_id
   http_method =
aws_api_gateway_method.proxy_root.http_method

   integration_http_method = "POST"
   type                    = "AWS_PROXY"
   uri                     =
aws_lambda_function.terraform_lambda.invoke_arn
 }


 resource "aws_api_gateway_deployment" "terraform_api" {
   depends_on = [
     aws_api_gateway_integration.lambda,
     aws_api_gateway_integration.lambda_root,
   ]

   rest_api_id =
aws_api_gateway_rest_api.terraform_api.id
   stage_name  = "test"
 }


data "http" "webhook" {
  url =
"https://api.telegram.org/bot${var.bot_token}/setWebHook
?url=${aws_api_gateway_deployment.terraform_api.invoke_u
rl}"
}
```

Код Lambda.tf

```
resource "aws_iam_role" "iam_for_lambda" {
  name = "iam_for_lambda"
  assume_role_policy = <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "sts:AssumeRole",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Effect": "Allow",
      "Sid": ""
    }
  ]
}
EOF
}


resource "aws_iam_policy" "policy" {
  name        = "policy"
  policy      = <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "*",
            "Resource": "*"
        }
    ]
}
EOF
}

resource "aws_iam_policy_attachment" "test-attach" {
  name       = "test-attachment"
  roles      = ["${aws_iam_role.iam_for_lambda.name}"]
  policy_arn = "${aws_iam_policy.policy.arn}"
}
```

```
resource "aws_lambda_function" "terraform_lambda" {
  s3_bucket = "zhuvakabot"
  s3_key    = "lambda.zip"
  function_name = "terraform_lambda_tg_bot"
  role          = "${aws_iam_role.iam_for_lambda.arn}"
  handler = "lambda_function.lambda_handler"
  runtime = "python3.8"

}

  environment {
    variables = {
      BOT_ID = ""
        VOICE_NAME = "Maxim"
        S3_BUCKET_NAME = "zhuvakatelegram"
        START_MESSAGE = "Привет! умею озвучивать
текст, который ты мне пишешь:) Попробуй мне что-то
написать"
        REGION = "us-east-2"
    }
  }

resource "aws_lambda_permission" "apigw" {
   statement_id  = "AllowAPIGatewayInvoke"
   action        = "lambda:InvokeFunction"
   function_name =
aws_lambda_function.terraform_lambda.function_name
   principal     = "apigateway.amazonaws.com"


   source_arn =
"${aws_api_gateway_rest_api.terraform_api.execution_arn}
/*/*"
 }
```