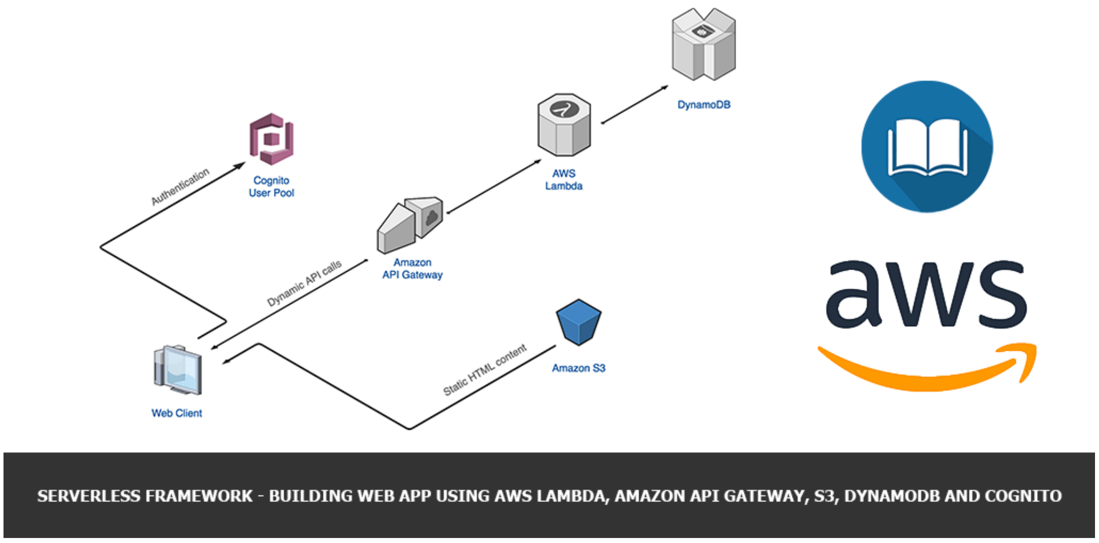
DEV-OPS-NOTES.COM

Dev & Ops tech blog

SERVERLESS FRAMEWORK – BUILDING WEB APP USING AWS LAMBDA, AMAZON API GATEWAY, S3, DYNAMODB AND COGNITO – PART 2

Voiced by Amazon Polly (https://aws.amazon.com/polly/)



(https://i1.wp.com/dev-ops-notes.com/wp-content/uploads/sites/2/2018/09/Serverless-framework-Building-Web-App-using-AWS-Lambda-Amazon-API-Gateway-S3-DynamoDB-and-Cognito.png?ssl=1)

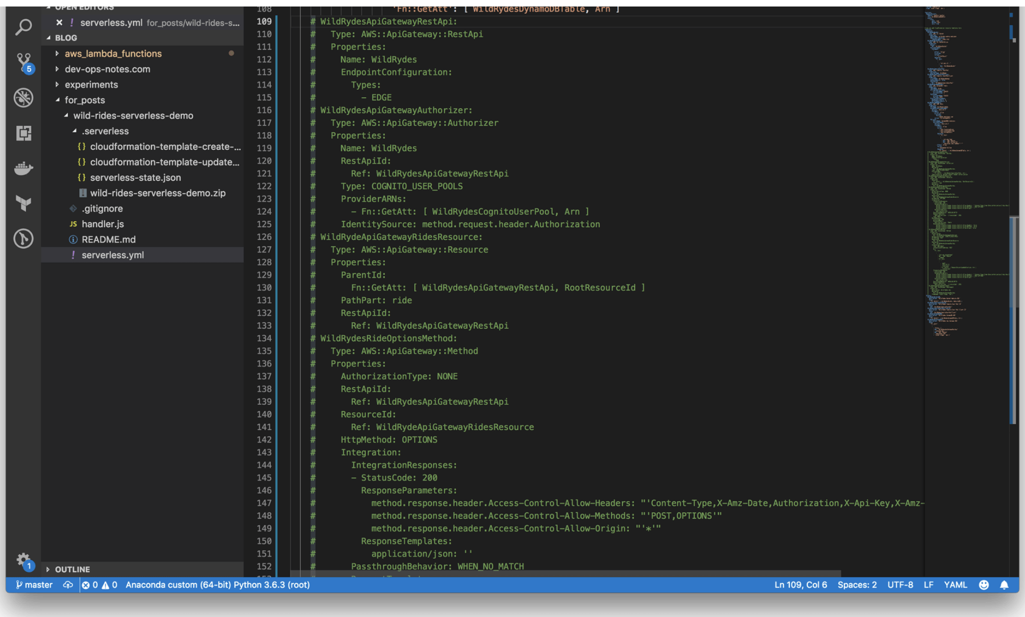
In previous article we've created and deployed a simple web application using which architecture consists of AWS Lambda, Amazon API Gateway, S3, DynamoDB and Cognito using Serverless framework (https://dev-ops-notes.com/cloud/serverless-framework-building-web-app-using-aws-lambda-amazon-api-gateway-s3-dynamodb-and-cognito/). That and this articles are based on original AWS hands-on tutorial (https://aws.amazon.com/getting-started/projects/build-serverless-web-app-lambda-apigateway-s3-dynamodb-cognito/), which we slightly automated.

I did not like the result we've got in first article. And decided to make it more simpler and clear. How? We can replace API Gateway resources with the `events:` which are available on Serverless framework (https://serverless.com/).

You may find the final result, which we got at the end of the previous post at my GitHub repository (https://github.com/andreivmaksimov/serverless-framework-aws-lambda-amazon-api-gateway-s3-dynamodb-and-cognito). Please, use tag v1.0 as a starting point. Final result is available at tag v2.0.

REPLACING API GATEWAY RESOURCES

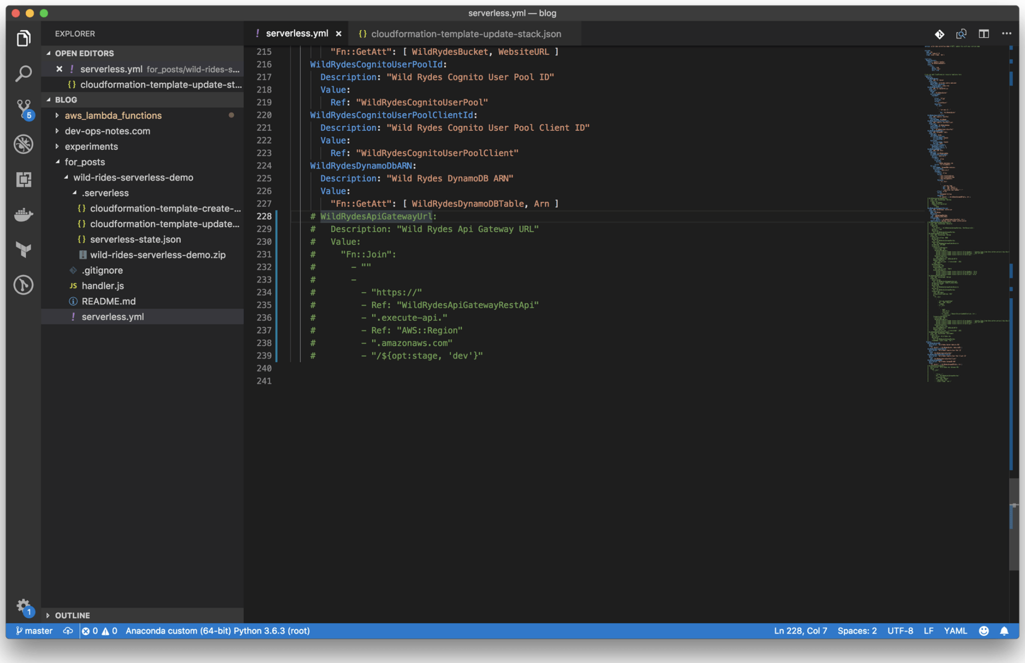
First thing we need to do is to comment all resources, which has `Type: AWS::ApiGateway::*` in



```yaml
              'Fn::GetAtt': [ WildRydesDynamoDBTable, Arn ]
#  WildRydesApiGatewayRestApi:
#    Type: AWS::ApiGateway::RestApi
#    Properties:
#      Name: WildRydes
#      EndpointConfiguration:
#        Types:
#          - EDGE
#  WildRydesApiGatewayAuthorizer:
#    Type: AWS::ApiGateway::Authorizer
#    Properties:
#      Name: WildRydes
#      RestApiId:
#        Ref: WildRydesApiGatewayRestApi
#      Type: COGNITO_USER_POOLS
#      ProviderARNs:
#        - Fn::GetAtt: [ WildRydesCognitoUserPool, Arn ]
#      IdentitySource: method.request.header.Authorization
#  WildRydeApiGatewayRidesResource:
#    Type: AWS::ApiGateway::Resource
#    Properties:
#      ParentId:
#        Fn::GetAtt: [ WildRydesApiGatewayRestApi, RootResourceId ]
#      PathPart: ride
#      RestApiId:
#        Ref: WildRydesApiGatewayRestApi
#  WildRydesRideOptionsMethod:
#    Type: AWS::ApiGateway::Method
#    Properties:
#      AuthorizationType: NONE
#      RestApiId:
#        Ref: WildRydesApiGatewayRestApi
#      ResourceId:
#        Ref: WildRydeApiGatewayRidesResource
#      HttpMethod: OPTIONS
#      Integration:
#        IntegrationResponses:
#        - StatusCode: 200
#          ResponseParameters:
#            method.response.header.Access-Control-Allow-Headers: "'Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-
#            method.response.header.Access-Control-Allow-Methods: "'POST,OPTIONS'"
#            method.response.header.Access-Control-Allow-Origin: "'*'"
#          ResponseTemplates:
#            application/json: ''
#        PassthroughBehavior: WHEN_NO_MATCH
```

(https://i0.wp.com/dev-ops-notes.com/wp-content/uploads/sites/2/2018/09/Serverless-Framework-Commenting-API-Gateway-Resources.png?ssl=1)

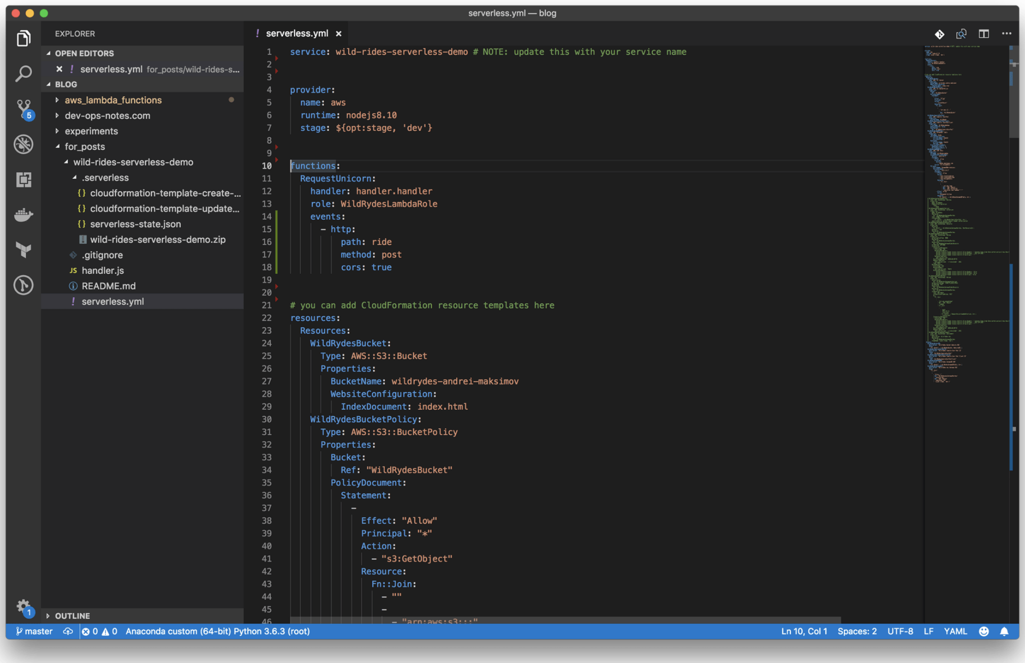
Also, you'll need to comment `WildRydesApiGatewayUrl` in the `Outputs:` section, because we're removed API Gateway declaration:



```yaml
      "Fn::GetAtt": [ WildRydesBucket, WebsiteURL ]
  WildRydesCognitoUserPoolId:
    Description: "Wild Rydes Cognito User Pool ID"
    Value:
      Ref: "WildRydesCognitoUserPool"
  WildRydesCognitoUserPoolClientId:
    Description: "Wild Rydes Cognito User Pool Client ID"
    Value:
      Ref: "WildRydesCognitoUserPoolClient"
  WildRydesDynamoDbARN:
    Description: "Wild Rydes DynamoDB ARN"
    Value:
      "Fn::GetAtt": [ WildRydesDynamoDBTable, Arn ]
  # WildRydesApiGatewayUrl:
  #   Description: "Wild Rydes Api Gateway URL"
  #   Value:
  #     "Fn::Join":
  #       - ""
  #       -
  #         - "https://"
  #         - Ref: "WildRydesApiGatewayRestApi"
  #         - ".execute-api."
  #         - Ref: "AWS::Region"
  #         - ".amazonaws.com"
  #         - "/${opt:stage, 'dev'}"
```

(https://i0.wp.com/dev-ops-notes.com/wp-content/uploads/sites/2/2018/09/Serverless-Framework-Commenting-API-Gateway-Resources-Outputs.png?ssl=1)Now we can start adding the same configuration by using `events:` declaration in `functions:` section. Let's publish our existing function `RequestUnicorn` using

```
  RequestUnicorn:
    handler: handler.handler
    role: WildRydesLambdaRole
    events:
      - http:
          path: ride
          method: post
          cors: true
```



```
service: wild-rides-serverless-demo # NOTE: update this with your service name


provider:
  name: aws
  runtime: nodejs8.10
  stage: ${opt:stage, 'dev'}


functions:
  RequestUnicorn:
    handler: handler.handler
    role: WildRydesLambdaRole
    events:
      - http:
          path: ride
          method: post
          cors: true


# you can add CloudFormation resource templates here
resources:
  Resources:
    WildRydesBucket:
      Type: AWS::S3::Bucket
      Properties:
        BucketName: wildrydes-andrei-maksimov
        WebsiteConfiguration:
          IndexDocument: index.html
    WildRydesBucketPolicy:
      Type: AWS::S3::BucketPolicy
      Properties:
        Bucket:
          Ref: "WildRydesBucket"
        PolicyDocument:
          Statement:
            -
              Effect: "Allow"
              Principal: "*"
              Action:
                - "s3:GetObject"
              Resource:
                Fn::Join:
                  - ""
                  -
                    - "arn:aws:s3:::"
```

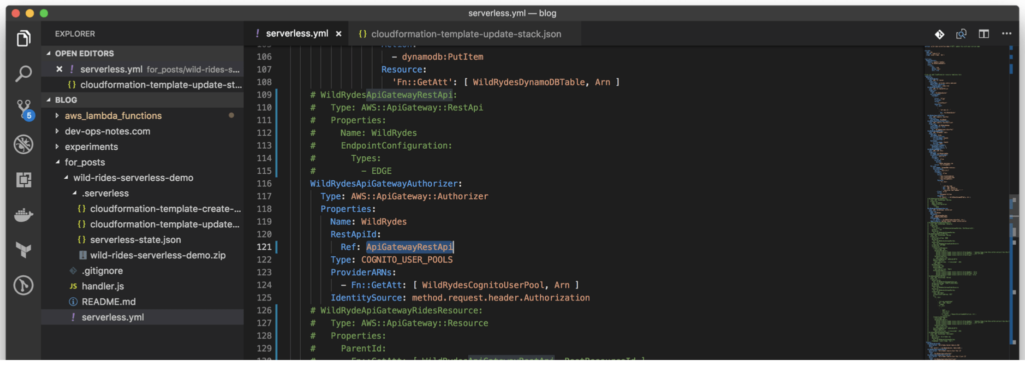
(https://i0.wp.com/dev-ops-notes.com/wp-content/uploads/sites/2/2018/09/Serverless-Framework-Publish-API-Gateway-Resources-Events.png?ssl=1)

I removed all not necessary comments from the file to make the file more readable.

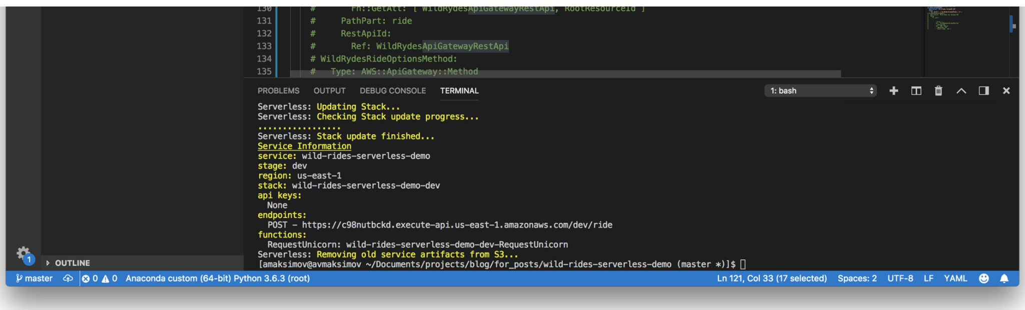
Let's deploy our infrastructure using the following command:

```
sls deploy
```

Now we need to implement API Gateway Authorizer. I think, we can uncomment one of the previously commented resources and modify it's reference to the API Gateway.



```
                - dynamodb:PutItem
              Resource:
                'Fn::GetAtt': [ WildRydesDynamoDBTable, Arn ]
    # WildRydesApiGatewayRestApi:
    #   Type: AWS::ApiGateway::RestApi
    #   Properties:
    #     Name: WildRydes
    #     EndpointConfiguration:
    #       Types:
    #         - EDGE
    WildRydesApiGatewayAuthorizer:
      Type: AWS::ApiGateway::Authorizer
      Properties:
        Name: WildRydes
        RestApiId:
          Ref: ApiGatewayRestApi
        Type: COGNITO_USER_POOLS
        ProviderARNs:
          - Fn::GetAtt: [ WildRydesCognitoUserPool, Arn ]
        IdentitySource: method.request.header.Authorization
    # WildRydeApiGatewayRidesResource:
    #   Type: AWS::ApiGateway::Resource
    #   Properties:
    #     ParentId:
```



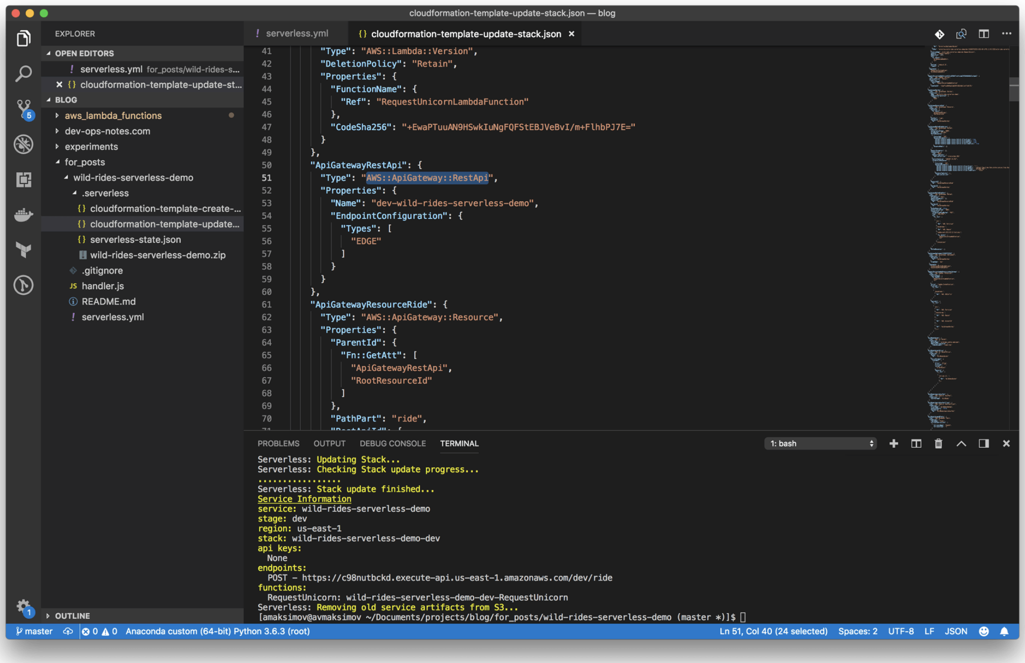
```
#       Fn::GetAtt: [ WildRydesApiGatewayRestApi, RootResourceId ]
#     PathPart: ride
#     RestApiId:
#       Ref: WildRydesApiGatewayRestApi
# WildRydesRideOptionsMethod:
#   Type: AWS::ApiGateway::Method
```

```
Serverless: Updating Stack...
Serverless: Checking Stack update progress...
.................
Serverless: Stack update finished...
Service Information
service: wild-rides-serverless-demo
stage: dev
region: us-east-1
stack: wild-rides-serverless-demo-dev
api keys:
  None
endpoints:
  POST - https://c98nutbckd.execute-api.us-east-1.amazonaws.com/dev/ride
functions:
  RequestUnicorn: wild-rides-serverless-demo-dev-RequestUnicorn
Serverless: Removing old service artifacts from S3...
[amaksimov@avmaksimov ~/Documents/projects/blog/for_posts/wild-rides-serverless-demo (master *)]$
```

(https://i1.wp.com/dev-ops-notes.com/wp-content/uploads/sites/2/2018/09/Serverless-Framework-Commenting-API-Gateway-Midified-Authorizer.png?ssl=1)

You may be interested, where I got `ApiGatewayRestApi` as a reference to the API Gateway, which we never declared. The reason is the Serverless framework which converts `serverless.yaml` file to the CloudFormation template which we deploying each time we're calling `sls deploy` command. You may find it's content in `.serverless/cloudformation-template-update-stack.json` file inside our project structure after the first deploy.

All we need to do is to find `AWS::ApiGateway::RestApi` (API Gateway itself) resource declaration and take it's name as a reference.



cloudformation-template-update-stack.json — blog

```
      "Type": "AWS::Lambda::Version",
      "DeletionPolicy": "Retain",
      "Properties": {
        "FunctionName": {
          "Ref": "RequestUnicornLambdaFunction"
        },
        "CodeSha256": "+EwaPTuuAN9HSwkIuNgFQFStEBJVeBvI/m+FlhbPJ7E="
      }
    },
    "ApiGatewayRestApi": {
      "Type": "AWS::ApiGateway::RestApi",
      "Properties": {
        "Name": "dev-wild-rides-serverless-demo",
        "EndpointConfiguration": {
          "Types": [
            "EDGE"
          ]
        }
      }
    },
    "ApiGatewayResourceRide": {
      "Type": "AWS::ApiGateway::Resource",
      "Properties": {
        "ParentId": {
          "Fn::GetAtt": [
            "ApiGatewayRestApi",
            "RootResourceId"
          ]
        },
        "PathPart": "ride",
```

```
Serverless: Updating Stack...
Serverless: Checking Stack update progress...
.................
Serverless: Stack update finished...
Service Information
service: wild-rides-serverless-demo
stage: dev
region: us-east-1
stack: wild-rides-serverless-demo-dev
api keys:
  None
endpoints:
  POST - https://c98nutbckd.execute-api.us-east-1.amazonaws.com/dev/ride
functions:
  RequestUnicorn: wild-rides-serverless-demo-dev-RequestUnicorn
Serverless: Removing old service artifacts from S3...
[amaksimov@avmaksimov ~/Documents/projects/blog/for_posts/wild-rides-serverless-demo (master *)]$
```

(https://i1.wp.com/dev-ops-notes.com/wp-content/uploads/sites/2/2018/09/Serverless-Framework-Generated-CloudFormation-Template.png?ssl=1)Let's redeploy our stack to make sure everything's working:

again. Getting web application sources:

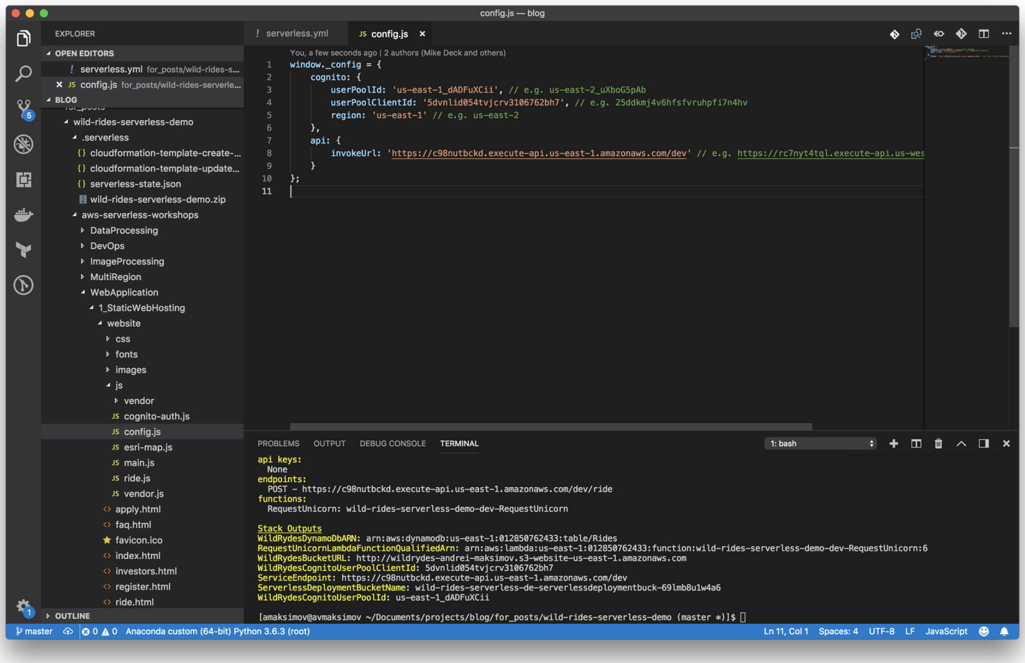
```
git clone https://github.com/awslabs/aws-serverless-workshops/
```

As you remember, we already described all needed `Outputs:` in `resources:` section of our `serverless.yaml` file. So, all we need to do is to execute the following command to get it:

```
sls info --verbose
```

Now we're ready to edit the `config.js` file:

```
window._config = {
    cognito: {
        userPoolId: 'us-east-1_dADFuXCii', // e.g. us-east-2_uXboG5pAb
        userPoolClientId: '5dvnlid054tvjcrv3106762bh7', // e.g.
25ddkmj4v6hfsfvruhpfi7n4hv
        region: 'us-east-1' // e.g. us-east-2
    },
    api: {
        invokeUrl: 'https://c98nutbckd.execute-api.us-east-1.amazonaws.com/dev' //
e.g. https://rc7nyt4tql.execute-api.us-west-2.amazonaws.com/prod',
    }
};
```



```
window._config = {
    cognito: {
        userPoolId: 'us-east-1_dADFuXCii', // e.g. us-east-2_uXboG5pAb
        userPoolClientId: '5dvnlid054tvjcrv3106762bh7', // e.g. 25ddkmj4v6hfsfvruhpfi7n4hv
        region: 'us-east-1' // e.g. us-east-2
    },
    api: {
        invokeUrl: 'https://c98nutbckd.execute-api.us-east-1.amazonaws.com/dev' // e.g. https://rc7nyt4tql.execute-api.us-wes
    }
};
```

```
api keys:
  None
endpoints:
  POST - https://c98nutbckd.execute-api.us-east-1.amazonaws.com/dev/ride
functions:
  RequestUnicorn: wild-rides-serverless-demo-dev-RequestUnicorn

Stack Outputs
WildRydesDynamoDbARN: arn:aws:dynamodb:us-east-1:012850762433:table/Rides
RequestUnicornLambdaFunctionQualifiedArn: arn:aws:lambda:us-east-1:012850762433:function:wild-rides-serverless-demo-dev-RequestUnicorn:6
WildRydesBucketURL: http://wildrydes-andrei-maksimov.s3-website-us-east-1.amazonaws.com
WildRydesCognitoUserPoolClientId: 5dvnlid054tvjcrv3106762bh7
ServiceEndpoint: https://c98nutbckd.execute-api.us-east-1.amazonaws.com/dev
ServerlessDeploymentBucketName: wild-rides-serverless-de-serverlessdeploymentbuck-69lmb8u1w4a6
WildRydesCognitoUserPoolId: us-east-1_dADFuXCii

[amaksimov@avmaksimov ~/Documents/projects/blog/for_posts/wild-rides-serverless-demo (master *)]$
```

[(https://i0.wp.com/dev-ops-notes.com/wp-content/uploads/sites/2/2018/09/Serverless-Framework-Static-Web-Application-Configuration.png?ssl=1)](https://i0.wp.com/dev-ops-notes.com/wp-content/uploads/sites/2/2018/09/Serverless-Framework-Static-Web-Application-Configuration.png?ssl=1)

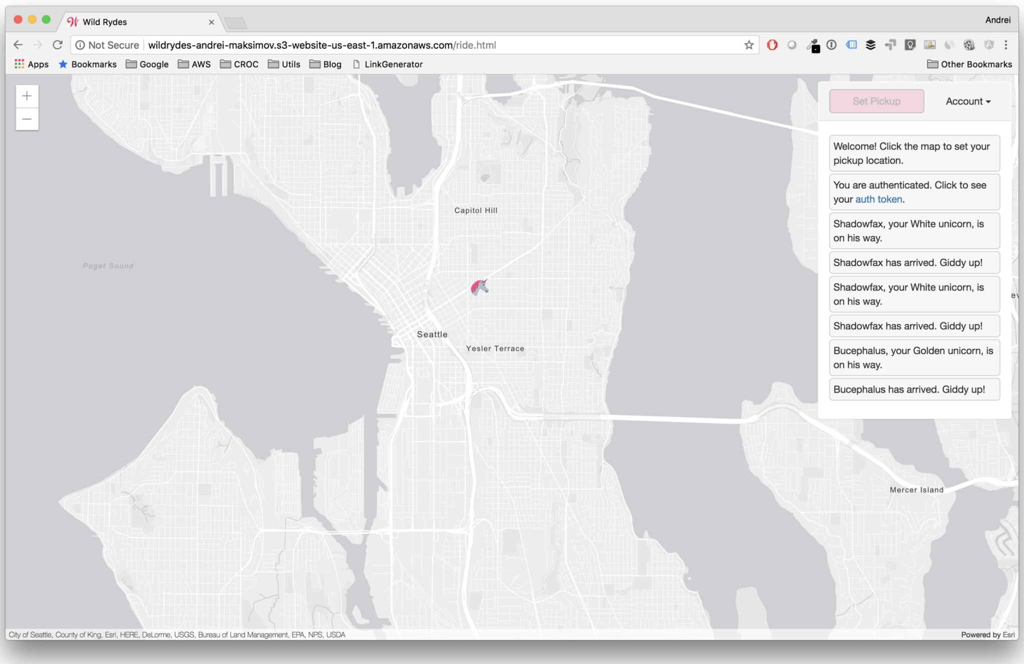
```
s3://wildrydes-firstname-lastname
rm -rf ./aws-serverless-workshops
```

Redeploy the stack, if you did not do it earlier:

```
sls deploy
```

TESTING

Now our application is up and running. All we need to do is to verify its functionality by opening the `WildRydesBucketURL`, registering new user using `/register.html` URL, verifying user manually using Cognito web interface and logging in using `/ride.html` URL. The whole testing process is described in my first post (https://dev-ops-notes.com/cloud/serverless-framework-building-web-app-using-aws-lambda-amazon-api-gateway-s3-dynamodb-and-cognito/) and original AWS (https://aws.amazon.com/getting-started/projects/build-serverless-web-app-lambda-apigateway-s3-dynamodb-cognito/) tutorial.


Wild Rydes
Andrei
Not Secure wildrydes-andrei-maksimov.s3-website-us-east-1.amazonaws.com/ride.html
Apps
Bookmarks
Google
AWS
CROC
Utils
Blog
LinkGenerator
Other Bookmarks
Set Pickup
Account
Capitol Hill
Puget Sound
Seattle
Yesler Terrace
Mercer Island
City of Seattle, County of King, Esri, HERE, DeLorme, USGS, Bureau of Land Management, EPA, NPS, USDA
Powered by Esri

Welcome! Click the map to set your pickup location.

You are authenticated. Click to see your auth token.

Shadowfax, your White unicorn, is on his way.

Shadowfax has arrived. Giddy up!

Shadowfax, your White unicorn, is on his way.

Shadowfax has arrived. Giddy up!

Bucephalus, your Golden unicorn, is on his way.

Bucephalus has arrived. Giddy up!

(https://i1.wp.com/dev-ops-notes.com/wp-content/uploads/sites/2/2018/09/Serverless-Framework-Deployed-Web-Application-End-Result.png?ssl=1)

RESULT

Let remove all commented sections and take a look under the final result:

```
  runtime: nodejs8.10
  stage: ${opt:stage, 'dev'}

functions:
  RequestUnicorn:
    handler: handler.handler
    role: WildRydesLambdaRole
    events:
      - http:
          path: ride
          method: post
          cors: true
          authorizer:
            type: COGNITO_USER_POOLS
            authorizerId:
              Ref: WildRydesApiGatewayAuthorizer

# you can add CloudFormation resource templates here
resources:
  Resources:
    WildRydesBucket:
      Type: AWS::S3::Bucket
      Properties:
        BucketName: wildrydes-andrei-maksimov
        WebsiteConfiguration:
          IndexDocument: index.html
    WildRydesBucketPolicy:
      Type: AWS::S3::BucketPolicy
      Properties:
        Bucket:
          Ref: "WildRydesBucket"
        PolicyDocument:
          Statement:
            -
              Effect: "Allow"
              Principal: "*"
              Action:
                - "s3:GetObject"
              Resource:
                Fn::Join:
                  - ""
                  -
                    - "arn:aws:s3:::"
                    -
                      Ref: "WildRydesBucket"
                    - "/*"
    WildRydesCognitoUserPool:
      Type: AWS::Cognito::UserPool
      Properties:
        UserPoolName: WildRydes
        GenerateSecret: false
        UserPoolId:
          Ref: "WildRydesCognitoUserPool"
    WildRydesDynamoDBTable:
      Type: AWS::DynamoDB::Table
      Properties:
        TableName: Rides
        AttributeDefinitions:
          - AttributeName: RideId
            AttributeType: S
        KeySchema:
          - AttributeName: RideId
            KeyType: HASH
        ProvisionedThroughput:
          ReadCapacityUnits: 5
          WriteCapacityUnits: 5
    WildRydesLambdaRole:
      Type: AWS::IAM::Role
      Properties:
        RoleName: WildRydesLambda
        AssumeRolePolicyDocument:
          Version: '2012-10-17'
          Statement:
            - Effect: Allow
              Principal:
                Service:
                  - lambda.amazonaws.com
              Action: sts:AssumeRole
        Policies:
          - PolicyName: DynamoDBWriteAccess
            PolicyDocument:
              Version: '2012-10-17'
              Statement:
                - Effect: Allow
                  Action:
                    - logs:CreateLogGroup
                    - logs:CreateLogStream
                    - logs:PutLogEvents
                  Resource:
                    - 'Fn::Join':
                      - ':'
                      -
                        - 'arn:aws:logs'
                        - Ref: 'AWS::Region'
                        - Ref: 'AWS::AccountId'
                        - 'log-group:/aws/lambda/*:*:*'
                - Effect: Allow
                  Action:
                    - dynamodb:PutItem
                  Resource:
                    'Fn::GetAtt': [ WildRydesDynamoDBTable, Arn ]
        RestApiId:
          Ref: ApiGatewayRestApi
        Type: COGNITO_USER_POOLS
        ProviderARNs:
          - Fn::GetAtt: [ WildRydesCognitoUserPool, Arn ]
        IdentitySource: method.request.header.Authorization
  Outputs:
    WildRydesBucketURL:
      Description: "Wild Rydes Bucket Website URL"
      Value:
        "Fn::GetAtt": [ WildRydesBucket, WebsiteURL ]
    WildRydesCognitoUserPoolId:
      Description: "Wild Rydes Cognito User Pool ID"
      Value:
        Ref: "WildRydesCognitoUserPool"
    WildRydesCognitoUserPoolClientId:
      Description: "Wild Rydes Cognito User Pool Client ID"
      Value:
        Ref: "WildRydesCognitoUserPoolClient"
    WildRydesDynamoDbARN:
      Description: "Wild Rydes DynamoDB ARN"
      Value:
        "Fn::GetAtt": [ WildRydesDynamoDBTable, Arn ]
```

As you can see, now we have much less code.

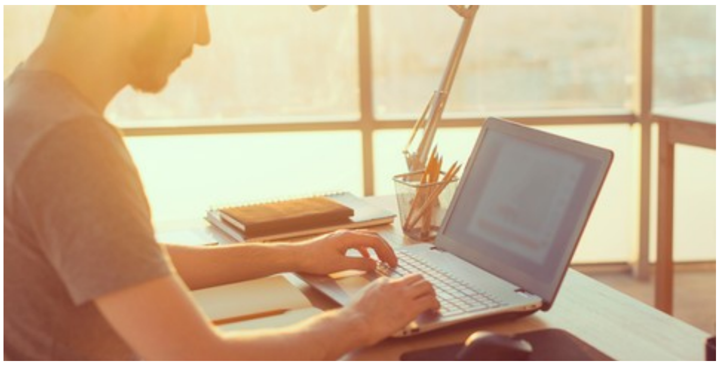
RESOURCE CLEANUP

To cleanup everything you need to call

```
aws s3 rm s3://wildrydes-firstname-lastname --recursive
sls remove
```

FINAL WORDS

Hope, you've found this article helpful. If you have any questions, please, feel free to ask them in comments section. Also, you may find additional example of API Gateway integrations using Serverless framework in it's Events documentation (https://serverless.com/framework/docs/providers/aws/events/apigateway/#share-authorizer).



(https://www.addtoany.com/add_to/facebook?linkurl=https
%3A%2F%2Fdev-ops-notes.com%2Faws%2Fserverless-
framework-building-web-app-using-aws-lambda-amazon-api-
gateway-s3-dynamodb-and-cognito-part-2%2F&
linkname=Serverless%20framework%20%E2%80%93
%20Building%20Web%20App%20using%20AWS%20Lambda
%2C%20Amazon%20API%20Gateway%2C%20S3
%2C%20DynamoDB%20and%20Cognito%20%E2%80%93
%20Part%202)
(https://www.addtoany.com/add_to/twitter?linkurl=https
%3A%2F%2Fdev-ops-notes.com%2Faws%2Fserverless-
framework-building-web-app-using-aws-lambda-amazon-api-
gateway-s3-dynamodb-and-cognito-part-2%2F&
linkname=Serverless%20framework%20%E2%80%93
%20Building%20Web%20App%20using%20AWS%20Lambda
%2C%20Amazon%20API%20Gateway%2C%20S3
%2C%20DynamoDB%20and%20Cognito%20%E2%80%93
%20Part%202)
(https://www.addtoany.com/add_to/linkedin?linkurl=https
%3A%2F%2Fdev-ops-notes.com%2Faws%2Fserverless-
framework-building-web-app-using-aws-lambda-amazon-api-
gateway-s3-dynamodb-and-cognito-part-2%2F&
linkname=Serverless%20framework%20%E2%80%93
%20Building%20Web%20App%20using%20AWS%20Lambda
%2C%20Amazon%20API%20Gateway%2C%20S3
%2C%20DynamoDB%20and%20Cognito%20%E2%80%93
%20Part%202)
(https://www.addtoany.com/add_to/google_plus?linkurl=https
%3A%2F%2Fdev-ops-notes.com%2Faws%2Fserverless-
framework-building-web-app-using-aws-lambda-amazon-api-
gateway-s3-dynamodb-and-cognito-part-2%2F&
linkname=Serverless%20framework%20%E2%80%93
%20Building%20Web%20App%20using%20AWS%20Lambda
%2C%20Amazon%20API%20Gateway%2C%20S3
%2C%20DynamoDB%20and%20Cognito%20%E2%80%93
%20Part%202)
(https://www.addtoany.com/share)

(https://www.addtoany.com/add_to/facebook?linkurl=https%3A%2F%2Fdev-ops-notes.com%2Fcloud%2Fserverless-framework-building-web-app-using-aws-lambda-amazon-api-gateway-s3-dynamodb-and-cognito-part-2%2F&linkname=Serverless%20framework%20%E2%80%93%20Building%20Web%20App%20using%20AWS%20Lambda%2C%20Amazon%20API%20Gateway%2C%20S3%2C%20DynamoDB%20and%20Cognito%20-%20Part%202%20-%20Dev-Ops-Notes.com)
(https://www.addtoany.com/add_to/twitter?linkurl=https%3A%2F%2Fdev-ops-notes.com%2Fcloud%2Fserverless-framework-building-web-app-using-aws-lambda-amazon-api-gateway-s3-dynamodb-and-cognito-part-2%2F&linkname=Serverless%20framework%20%E2%80%93%20Building%20Web%20App%20using%20AWS%20Lambda%2C%20Amazon%20API%20Gateway%2C%20S3%2C%20DynamoDB%20and%20Cognito%20-%20Part%202%20-%20Dev-Ops-Notes.com)
(https://www.addtoany.com/add_to/linkedin?linkurl=https%3A%2F%2Fdev-ops-notes.com%2Fcloud%2Fserverless-framework-building-web-app-using-aws-lambda-amazon-api-gateway-s3-dynamodb-and-cognito-part-2%2F&linkname=Serverless%20framework%20%E2%80%93%20Building%20Web%20App%20using%20AWS%20Lambda%2C%20Amazon%20API%20Gateway%2C%20S3%2C%20DynamoDB%20and%20Cognito%20-%20Part%202%20-%20Dev-Ops-Notes.com)
(https://www.addtoany.com/add_to/google_plus?linkurl=https%3A%2F%2Fdev-ops-notes.com%2Fcloud%2Fserverless-framework-building-web-app-using-aws-lambda-amazon-api-gateway-s3-dynamodb-and-cognito-part-2%2F&linkname=Serverless%20framework%20%E2%80%93%20Building%20Web%20App%20using%20AWS%20Lambda%2C%20Amazon%20API%20Gateway

%2C%20S3%2C%20DynamoDB%20and%20Cognito%20-%20Part%202%20-%20Dev-Ops-Notes.com)
(https://www.addtoany.com/share)