Building Multiple Types of Websites

Due to the breadth of service options from Cloud Computing, there are many ways to build websites, from static to serverless to virtualized to PaaS. Let's take a look at a few different examples.

The following screencast is a step by step Demo of three websites getting built (AWS Static S3, AWS Lambda in Python, and EC2 Spot Instance).

Watch "Demo: A Tale of Three Websites" in the following screencast.

Video Link: https://www.youtube.com/watch?v=acmuuHhrmSs[110]

Instructions AWS S3 Website

To build a very simple static hosted website on AWS S3, you can follow S3 Hosting Instructions here[111]. The general concept in a static website is the HTML assets generate before viewing. These assets then go into a Content Delivery Network that distributes the assets to edge locations world-wide. This technique is an optimal design for the fastest possible web page viewing time.

Build a simple static S3 Website on AWS in the following screencast.

Video Link: https://www.youtube.com/watch?v=zFO-rcYY3B4[112]

Instructions AWS Lambda Website

AWS Lambda is perhaps the most straightforward way to build a website that delivers HTML quickly. To do this step, use the following directions.

1. Use AWS Cloud9 and "right-click" to a new lambda function.

2. Paste the code below into the editor.
The following example demonstrates the Python code necessary to build a Lambda function that returns HTML.

[107]https://cloud.google.com/appengine/docs/standard/python3/quickstart
[108]https://cloud.google.com/source-repositories/docs/quickstart-triggering-builds-with-source-repositories
[109]https://github.com/noahgift/gcp-hello-ml
[110]https://www.youtube.com/watch?v=acmuuHhrmSs
[111]https://docs.aws.amazon.com/AmazonS3/latest/dev/WebsiteHosting.html
[112]https://www.youtube.com/watch?v=zFO-rcYY3B4

```
def lambda_handler(event, context):
    content = """
    <html>
    <p> Hello website Lambda </p>
    </html>
    """
    response = {
        "statusCode": 200,
        "body": content,
        "headers": {"Content-Type": "text/html",},
    }
    return response
```

3. “Right-click” deploy the lambda function
4. Log in to the AWS console and click on the API Gateway icon in the AWS Lambda section. Verify that it returns “Hello website Lambda.”

You can also follow on how to “Build a simple AWS Lambda Website” in the following screencast.

Video Link: https://www.youtube.com/watch?v=lrr6h7YIcI8[113]

Instructions AWS EC2 Website

An older but still useful way to build a website is to launch a Virtual Machine, install a webserver on it, and then serve out traffic via a language like PHP, Python, or Ruby. To setup, a Linux, Apache, MySQL, PHP (LAMP) configuration, do the following.

Follow the tutorial on setting up LAMP Site here[114] or feel free to improvise and follow the more straightforward guide shown, as shown in the following screencast, “Build a simple AWS EC2 Website”.

Video Link: https://www.youtube.com/watch?v=xrG6UyhZE9Q[115]

Instructions AWS Elastic Beanstalk Website

AWS Elastic Beanstalk is an excellent option for a PaaS target for Flask on the AWS platform. You can refer to this Github project[116] for the sample code to build this demo.

The main steps are below.

A. Install the `eb` tool via these instructions[117].

B. Create a Flask application, as shown.

[113] https://www.youtube.com/watch?v=lrr6h7YIcI8
[114] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-lamp-amazon-linux-2.html
[115] https://www.youtube.com/watch?v=xrG6UyhZE9Q
[116] https://github.com/noahgift/Flask-Elastic-Beanstalk
[117] Installebtool:https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb-cli3-install.html

```
from flask import Flask
from flask import jsonify
app = Flask(__name__)

@app.route('/')
def hello():
    """Return a friendly HTTP greeting."""
    print("I am inside hello world")
    return 'Hello World! CD'

@app.route('/echo/<name>')
def echo(name):
    print(f"This was placed in the url: new-{name}")
    val = {"new-name": name}
    return jsonify(val)


if __name__ == '__main__':
   # Setting debug to True enables debug output. This line should be
   # removed before deploying a production app.
   application.debug = True
   application.run()
```

C. Use the `eb deploy` command as referenced here[118].

Build a simple Flask AWS Elastic Beanstalk Website in the following screencast.

Video Link: https://www.youtube.com/watch?v=51lmjwXvVw8[119]

[118] https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb3-deploy.html
[119] https://www.youtube.com/watch?v=51lmjwXvVw8