

DynamoZoo Image Portal

Congratulations, you're now an AWS consultant at tecRacer Consulting GmbH and you already have your first project on hand! Your customer is "NeuTech Innovations GmbH", and they want to create their "DynamoZoo Image Portal" in AWS. You will work to develop a cloud-native, serverless web application that showcases images of animals.

The customer already created an architecture (Figure 1), the user flow and some requirements. You're asked to implement the solution with Infrastructure as Code (Terraform) and create the Lambda function code and the HTML & JavaScript website.

Background

Company Name: NeuTech Innovations GmbH

Background:

Founded in 2016 in Munich, NeuTech Innovations GmbH has positioned itself as a pioneer in digital interactive experiences. Their primary offering has been a range of engaging educational platforms and tools, with a special focus on nature and environmental awareness. Capitalizing on the digital age, NeuTech's platforms often merge intuitive design with rich multimedia content, aiming to educate and captivate users across Europe with the wonders of the natural world.

Seeking to elevate direct consumer engagement, NeuTech aims to establish an interactive online portal, underscoring their products' essence and benefits. Recognizing the importance of a scalable and resource-efficient digital platform, they're venturing into a serverless architecture with the "DynamoZoo Image Portal". This prototype, leveraging AWS services, represents NeuTech's initial strides in its broader digital transformation strategy.

Task:

You will construct a serverless web application on AWS, where users select an animal type from a dropdown to view a corresponding image sourced from an S3 bucket. The solution integrates various AWS services, including Route53, CloudFront, Lambda, DynamoDB, and a CI/CD pipeline, to ensure smooth deployment and user interaction.

Details

Below you can see the architecture from your client (Figure 1) and a user-flow that describes the interaction with the application. The numbers of the user-flow correspond to the numbers mentioned in the architecture.

User-Flow:

1. A user can access a domain (e.g., dynamozoo.neu-tech.de) via a web browser.
 - The transmission should be encrypted using an SSL certificate.
2. The domain delivers a simple static webpage (HTML, JS & CSS) that is only accessible via CloudFront CDN.

3. The webpage contains a dropdown with multiple options (e.g., Dog, Cat, Bird) and a simple button, which points to a URL.
4. This URL points to an AWS API Gateway.
 - The value of the dropdown sent to the API should not be an S3 URL, but just the name of the selected animal (e.g., “dog”).
5. The API Gateway triggers a Lambda function which should receive the selected value of the dropdown (e.g., Dog, Cat, or Bird) in the Lambda event.
 - It searches in a DynamoDB using the key (Dog, Cat, or Bird) and accordingly returns an S3 URL to an image of the respective animal.
6. The Lambda function creates a pre-signed URL for the image in S3.
7. The pre-signed URL will be returned to the webpage via the API Gateway response.
 - The user sees an image of the selected animal in his webbrowser

Additional:

8. A developer can push the website code to an AWS CodeCommit repository via Git.
9. AWS CodeCommit / CodePipeline responds to the changes in the CodeCommit repository and triggers further services.
10. AWS CodeDeploy (or a similar service) updates the website in the S3 bucket and the Lambda function.

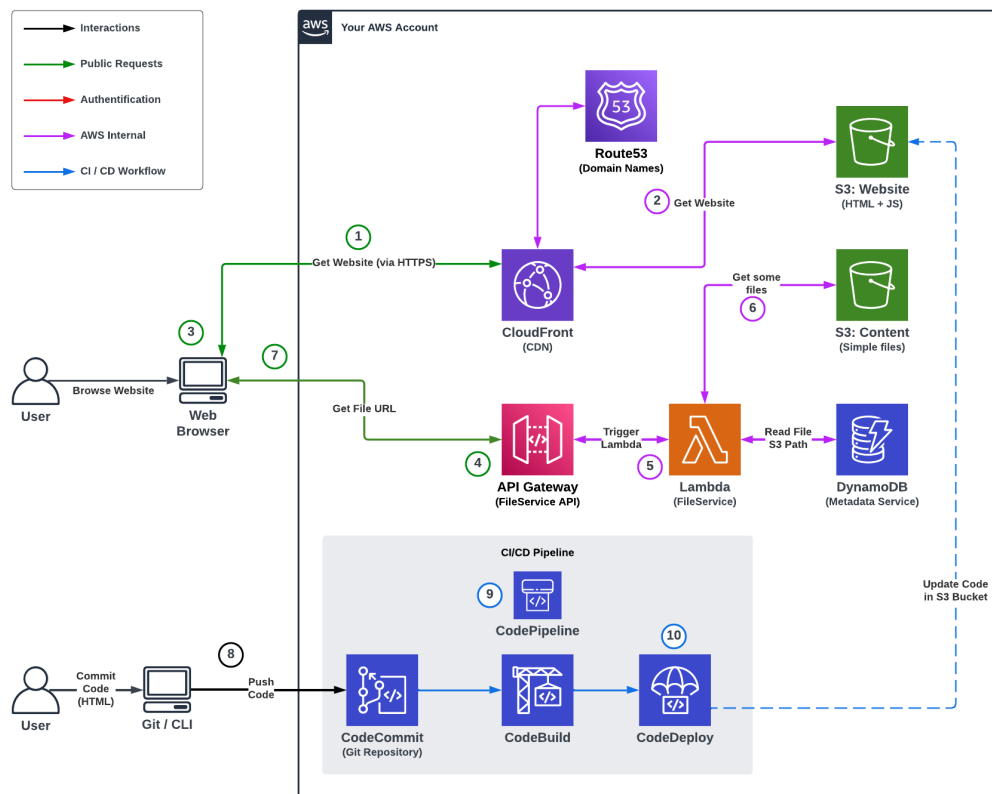


Figure 1 Serverless architecture

More Details

Preview

Below (Figure 2) you can see a preview of how the HTML website can look.

There is a DropDown with the animal names, a button to request the image and some space to display the image of the animal from our DynamoZoo.

Choose image name to request

Image name:




Figure 2 DynamoZoo Preview

Be aware to add all of the following functions:

1. S3 Bucket Website ("Website")
 - Website hosting enabled.
 - Not publicly accessible, only through CloudFront OAI.
2. S3 Bucket for Files ("Content")
 - Not publicly accessible.
 - Downloadable only via pre-signed URL.
3. CloudFront
 - Uses a public certificate from AWS ACM.
 - Domain name set by Route53.
 - Simple distribution for the website bucket with HTTP to HTTPS redirect.
4. API Gateway
 - No authorization, etc. needed.
 - Simple API endpoint that points to AWS Lambda through proxy integration.
 - (Optional: domain name for api, e.g.: "api.dynamozoo.neu-tech.de").
5. AWS Lambda
 - Simple code that performs a DynamoDB query.
 - Creates a pre-signed URL with GET access to the S3 Image.
 - Returns the pre-signed URL.
6. DynamoDB
 - Partition Key: "animal_name"

- Attribute: "s3_url"
 - On-demand mode
 - Encrypted with AWS Managed KMS Key
7. CI/CD Pipeline:
- Git repository + deployment.
 - Simple commit of the website code to CodeCommit and deployment to the S3 bucket with website hosting.

To note:

1. Work in your own, tecRacer AWS account.
2. When considering permissions, adhere to the "least privilege" principle.
3. Use Terraform to create the infrastructure:
 - Organize the code sensibly across multiple files.
 - Try to follow best practices for naming of files, resources, and variables.
 - If you use third-party modules, investigate them, and try to understand what they're doing.
 - Try to avoid static references to other resources.
 - Try to make the code reusable.
4. Ask your colleagues, me, Slack, or the internet for help.
 - But always try to understand what you did and why.
5. Feel free to "start small".
 - This is a great project for an iterative approach. You may start with building single services manually in the AWS console, rebuild them with Terraform, test things out and ask questions during this process.
 - In the end, stitch everything together so you can create and destroy the architecture seamlessly with Terraform.
6. Request as many meetings with "your customer" as you like, to ask questions, demo your current status or else.