|  |  |
| --- | --- |
| *A close up of a logo  Description automatically generated* | *DEPARTMENT OF INFORMATION TECHNOLOGY* |

|  |  |
| --- | --- |
| Semester | T.E. Semester V – Information Technology |
| Subject | Advance DevOps Lab |
| Subject Professor In-charge | Prof. Indu Anoop |
| Laboratory | (Leave blank for now) |

|  |  |  |
| --- | --- | --- |
| Student Name | Shreyash Kakde | |
| Roll Number | 20101A0049 | |
| Grade and Subject Teacher’s Signature |  |  |

|  |  |  |
| --- | --- | --- |
| Experiment | 11-12 | |
| Problem Statement | AWS lambda | |
| Resources / Apparatus Required | Hardware: Computer System | Software: Ubuntu/AWS |
| Details | **What is AWS Lambda?**  AWS lambda is a serverless compute service that lets you run your code without worrying about provisioning or managing any server. You can run your application or backend service using AWS scale the infrastructure with high availability.  The code which you run on AWS Lambda is called a lambda function. Currently, it supports the following programming languages.   * Java * Python * C# * Node.js * Go * PowerShell * Ruby   What is serverless?  Serverless most often refers to serverless applications. Serverless applications are ones that don’t require you to provision or manage any servers. You can focus on your core product and business logic instead of responsibilities like operating system (OS) access control OS patching, provisioning, right-sizing, scaling and availability. By building your application on a serverless platform, the platform manages these responsibilities for you. For service or platform to be considered serverless, it should provide the following capabilities.  No server management – You don’t have to provision or maintain any servers. There is no software or runtime to install, maintain or administer.  Flexile scaling – You can scale your application automatically or y adjusting its capacity through toggling the units of consumption rather than units of individual servers.  High availability – Serverless applications have built-in availability and fault tolerance. You don’t need to architect for these capabilities because the services running the applications provide them by default.  No idle capacity – You don’t have to pay for idle capacity. There is no need to pre-provision or over-provision capacity for things like compute and storage. There is no charge when your code isn’t running.  **AWS Lambda features:**   1. AWS Lambda easily scales the infrastructure without any additional configuration. It reduces the operational work involved. 2. It offers multiple options like AWS $3, CloudWatch, DynamoDB, API Gateway, Kinesis, Code Commit, and many more to trigger an event. 3. You don't need to invest upfront. You pay only for the memory used by the lambda function and minimal cost on the number of requests hence cost-efficient. 4. AWS Lambda is secure. It uses AWS IAM to define all the roles and security policies. 5. It offers fault tolerance for both services running the code and the function. You do not have to worry about the application down. | |
| OUTPUT | **Step 1:** Create IAM Role: Create role with following attached permissions:  • AWSLambdaFullAccess  • AmazonS3FullAccess  • CloudWatchFullAccess  **Step 2**: Create a bucket in AWS S3 to upload image. Ensure region of bucket is same as that of lambda function.  **Step 3:** Create a Lambda Function using node.js Blueprint Template for uploading image to s3 Bucket.  **Step 4:** Create a trigger to invoke creation of logs in Cloud Watch when an image is uploaded in the specified bucket. Test the code and view the logs in Cloud Watch.  Configure Event name in Test configure button to mention your s3 bucket name and image name to be uploaded     1. **S3 Bucket:** Creation of bucket and later on Image upload to trigger lambda function.      1. **Lambda Function:**       **3) Cloud Watch Logs:** | |
| Conclusion | In this exepriment we learned its workflow,various functions and created first lambda function using Python/java/**node.js** | |