Lecture 8 - Serverless - Lambda, DynamoDB, API Gateway (1h)

- Q&A about the previous lesson (3m)

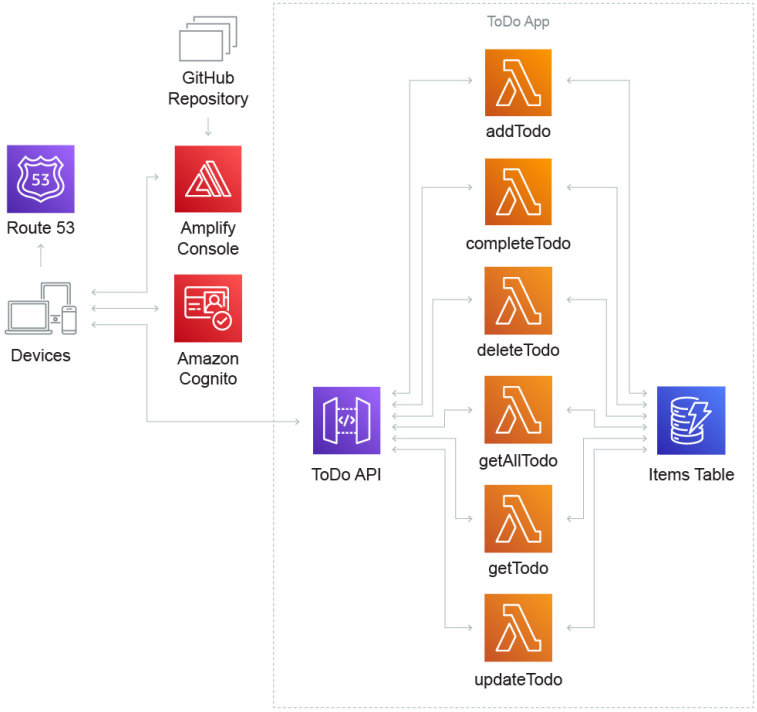
Lambda

https://aws.amazon.com/lambda/faqs/

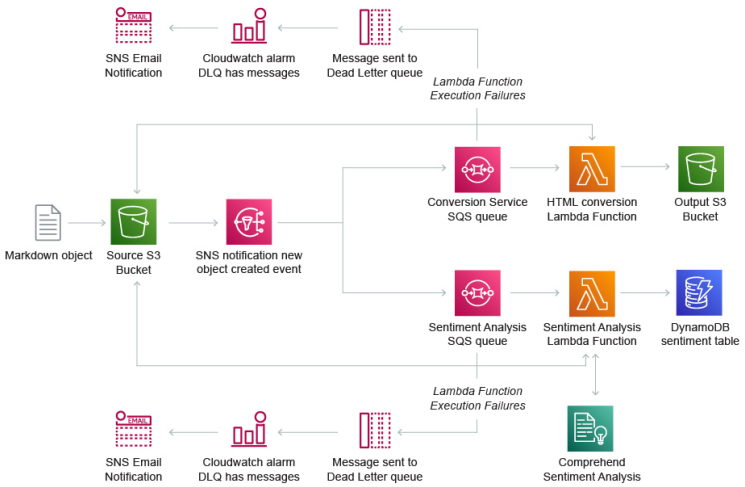
- Serverless computing → https://aws.amazon.com/serverless/
  - no infrastructure management → only manage your code
  - fastest time to market, awesome for MVPs and PoCs
  - pay per request and execution time
  - cost-efficient, no more over-provisioning and capacity-planning
  - infinitely scalable and elastic
  - awesome for event-driven architectures
- Serverless in AWS
  - Lambda
  - DynamoDB
  - Cognito
  - API Gateway
  - S3
  - SQS
  - SNS
  - Kinesis (Firehose, Data Streams)
  - Aurora Serverless
  - Step Functions

- Fargate (Serverless containers)

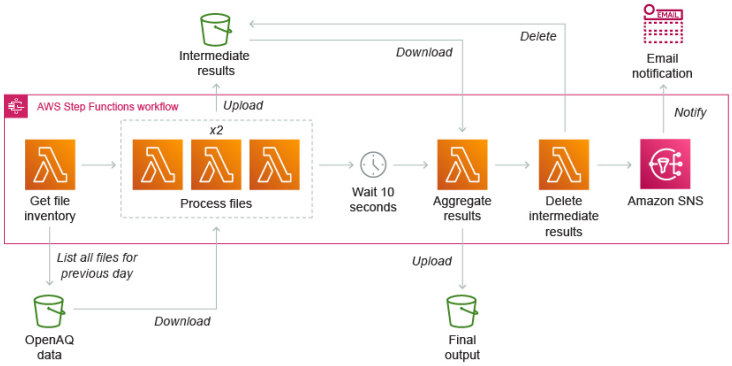
- General use cases
  - web-apps → https://github.com/aws-samples/lambda-refarch-webapp



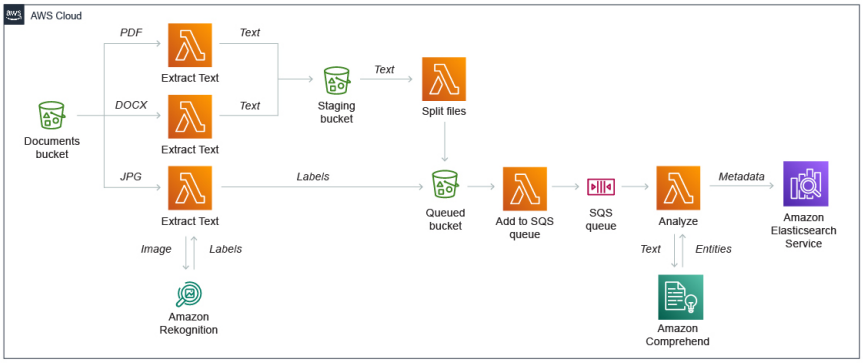
  - data-processing → https://github.com/aws-samples/lambda-refarch-fileprocessing



  - batch processing → https://github.com/aws-samples/aws-lambda-etl-ref-architecture



- event ingestions → Sample Code



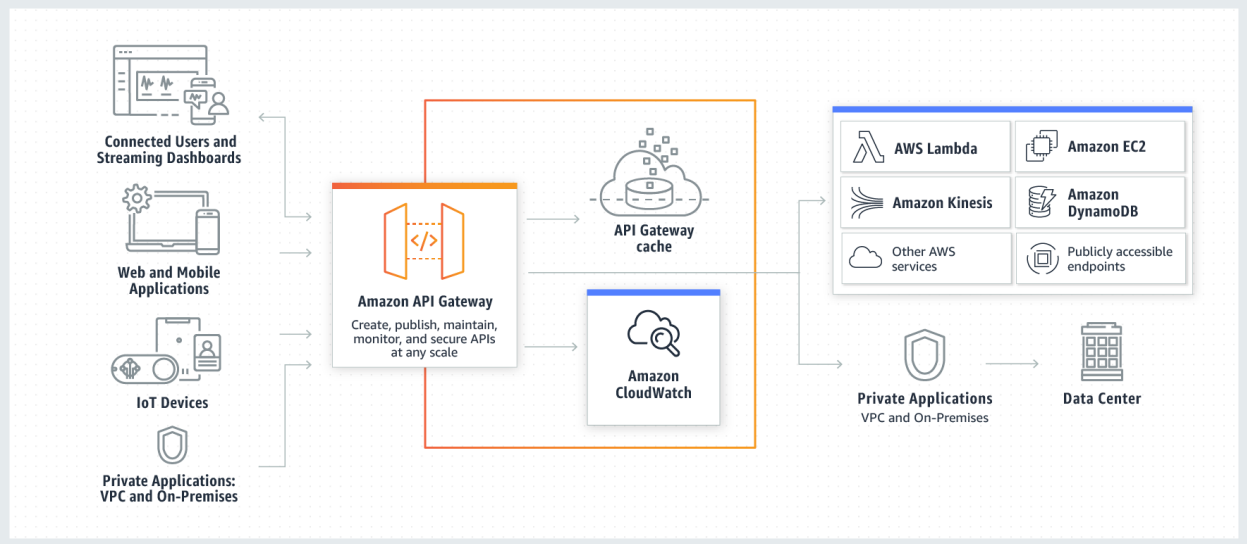
- lambda@edge → CloudFront distributions requests modifications https://aws.amazon.com/lambda/edge/
- snowball edge → https://aws.amazon.com/snowball/
- infrastructure automation
  - scheduled actions (cleanups, snapshots, credentials rotations etc)
  - event-driven actions (bucket object upload, CloudWatch event, CloudTrail event etc)
  - security and access control (credentials invalidations, firewall rules modifications etc)
  - CloudFormation custom resources
  - everything that is not present in AWS

- Where to start

    - builders → https://aws.amazon.com/getting-started/deep-dive-serverless/
    - workshops → https://workshops.aws/card/serverless
    - reference architectures → https://aws.amazon.com/architecture
- How it works
    - isolated lightweight nested vm → https://aws.amazon.com/blogs/aws/firecracker-lightweight-virtualization-for-serverless-computing/
        - no shell
        - no access to files
        - no permissions/users
        - no os updates
        - think of it as a code stored somewhere and executed on-demand
    - readonly filesystem (write to temporary filesystem `/tmp` is possible); NEW - EFS for lambda https://aws.amazon.com/blogs/aws/new-a-shared-file-system-for-your-lambda-functions/
    - processing power is added with memory (more memory allocated - more virtual CPUs available)
    -
- Main features
    - pay per request + execution time → fast and efficient functions means cost-efficient functions (note the max execution time is no more than 15min)
    - lots of integrations with other services
        - API Gateway
        - Kinesis (data transfomations)
        - DynamoDB (trigger lambda on event)
        - S3 (trigger lambda on event)
        - CloudFront (lambda@edge)
        - CloudWatch event bridge (scheduled actions with crontab)
        - CloudWatch (logs streaming to Elastic or Kinesis)

        - SNS, SQS (processing messages)
        - Cognito (user pools and identity pools)
    - runtimes
        - node
        - python
        - java
        - c# (.net core, powershell)
        - go
        - ruby
        - custom runtime
        - containers (on fargate or runtime API)
- Workshop → https://aws.amazon.com/getting-started/hands-on/build-serverless-web-app-lambda-apigateway-s3-dynamodb-cognito/

API Gateway



- Fully managed RESTful and WebSockets API gateway https://aws.amazon.com/api-gateway/faqs/
- Main features

    - integrates well with other AWS services
        - Load Balancers
        - Lambda
        - WAF
        - (and more)
    - API versioning and environments → create api stages and environments for them
    - Supports creating custom API keys
    - Swagger/OpenAPI compatible
    - API requests caching, throttling, requests transformation and validation
    - supports canary deployments
- Use cases
    - exposing any(?) AWS service or backend as an https endpoint
    - invoking lambda by network
- Endpoint types
    - edge-optimized (CloudFront powered delivery)
    - regional
- Auth
    - IAM (from within AWS or with sigv4 headers)
    - Cognito user pools
    - Lambda (custom) authorizer - JWT, OAuth
- For certification (read services faq and related whitepaper)
    - know websocket vs restful API gateway features
    - stages/environments
    - authorization/authentication options for API Gateway and use cases
    - know the most advertised features
    - know use cases
- Workshops → https://webapp.serverlessworkshops.io/

DynamoDB

- Fully managed NoSQL database (all pros and cons of being noSQL apply here) → https://aws.amazon.com/dynamodb/faqs/
  - HA out of the box
  - *Infinitely* scalable performance and storage
  - fast and consistent storage
  - event-driven with DynamoDB streams
  - other benefits of being serverless - no overprovisioning, pay only for what you use etc
- Main features
  - consists of **tables** that have **primary keys**
  - data stored in **rows,** stored items optionally have **attributes**
  - item max size is **400kb**
  - data types supported - scalar, document, set
  - support partition keys + sort keys (which have to be unique) → better explained in Stefane Maarek's video
  - capacity planned with CU (1WCU = 1KB, 1RCU=4KB)
    - provisioned https://aws.amazon.com/dynamodb/pricing/provisioned/
    - on-demand https://aws.amazon.com/dynamodb/pricing/on-demand/
    - there are calculators, just use them 🙂
  - consistency mods
    - strong
    - eventually (default)
  - data stored in partitions
- Other Features or Services
  - DAX (like a proxy or cache for perfomance boost) → https://aws.amazon.com/dynamodb/dax/

    - DynamoDB streams (for event-driven architecture) → https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html
- For certification (read services faq and related whitepaper)
    - DAX
    - DynamoDB streams
    - Global tables
- Workshops → https://amazon-dynamodb-labs.com/

Q&A session

- topic discussion
- sharing useful external resources and links