PROJ-611-001 Adoption of Cloud Computing and Blockchain Technology in the Industry

REVISION FINAL

Flash - Local Express

Adoption of Cloud Computing in the Industry

Instructor:

Mayra Samaniego MSc. Ph.D. (c)

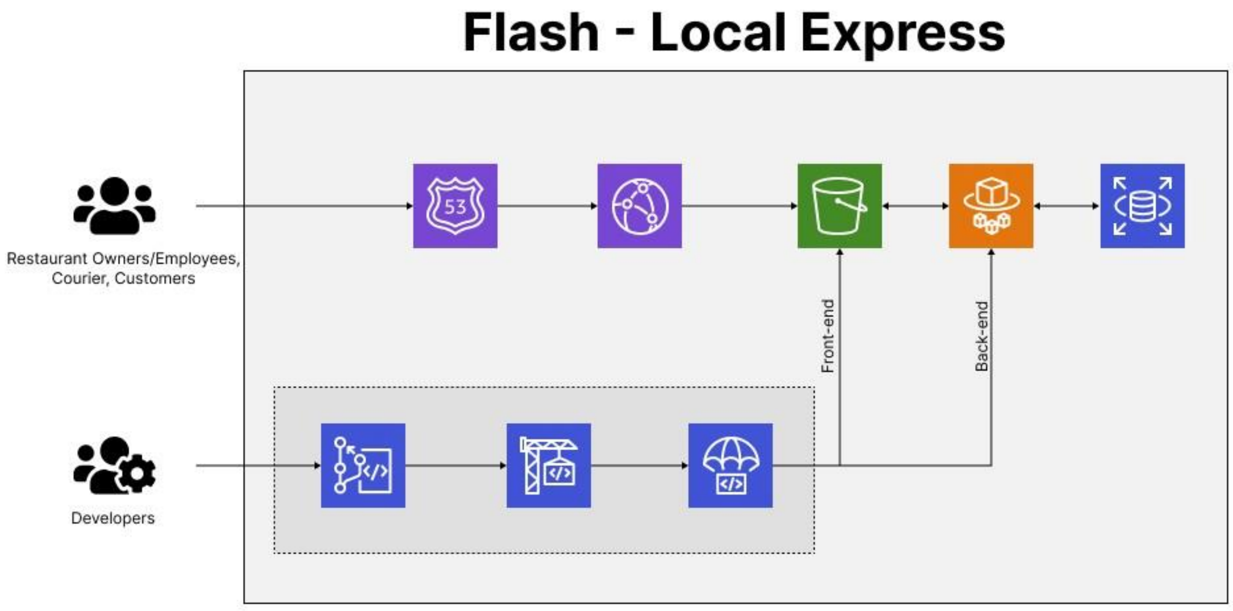
Members:

Hai Nam Nguyen – 000520322 – nguyen0465@saskpolytech.ca

Cong Chi Tai Nguyen - 000516006 - nguyen6169@saskpolytech.ca

Xuan Hieu Nguyen – 000518043 – nguyen8191@saskpolytech.ca

1. **Architecture**



- Route53 is a scalable Domain Name System (DNS) to register DNS of our domain name.
- CloudFront is a content delivery network (CDN) service for fast and secure content delivery globally.
- S3 for scalable and secure frontend hosting, providing scalable compute capacity based on demand and will be used for secure and efficient storage such as images.
- RDS as a managed relational database service, will handle the storage and retrieval of structured data related to delivery information.
- ECS will be utilized for containerized backend deployment.
- VPC allowing for a private and isolated network environment.
- IAM will be implemented to manage user access securely, controlling permissions and roles within the application.
- CodeCommit will be used as a fully managed source control service to securely store and manage application source code.
- CodePipeline will be implemented to automate the build, test, and deployment phases of the application development process.

2. **Frontend:**

Prerequisites:

- React documentation: https://react.dev/learn
- NextJS documentation: https://nextjs.org/docs
- Tailwind documentation: https://v2.tailwindcss.com/docs

a. Setup NextJS project
    a. Install NextJS: npx create-next-app@latest, follow documentation at https://nextjs.org/docs/getting-started/installation

  b. Running NexJS locally: npm run dev
  c. Build & Export static folder: npm run build

b. Setup AWS CodeCommit
  a. Getting AWS CodeCommit Credentials: https://docs.aws.amazon.com/codecommit/latest/userguide/setting-up-gc.html?icmpid=docs_acc_console_connect_np
  b. Connect via AWS CLI: https://aws.amazon.com/cli/
  c. Add, Commit, and Push code Git: https://www.atlassian.com/git/tutorials/syncing

c. Setup AWS CodeBuild
  a. Create a new project
  b. Setup environment variables
  c. Setup command lines to build

d. Setup AWS CodeDeploy
  a. Create a new deployment
  b. Deploy folder `out` to S3

e. Setup AWS S3
  a. Setup a new bucket
  b. Open to public for testing purposes
  c. Set static web hosting to index.html
  d. Setup bucket policy for the AWS Route53

f. Setup AWS CloudFront
  a. Create distribution
  b. Pointing to AWS S3 bucket

g. Setup AWS Route53
  a. Create CNAME record
  b. Route traffic to AWS CloudFront

3. Backend:

Prerequisites:

- AWS account: personal and learner lab
- AWS CLI installed on local machine
- Docker installed on local machine

a. Setup Django project

- Create a Django project locally.
- Install necessary dependencies.
- Develop REST API app and business logic.
- Test the application locally.

b. Setup AWS Services

  i. RDS PostgreSQL
  - Create a new RDS instance with PostgreSQL as the engine.
  - Use the endpoint, username, and password for connecting to the database.

  ii. CodeCommit
  - Create a new repository in CodeCommit.

- Clone the repository to local machine.
- Copy the Django project into this repository.
- Push code to the CodeCommit repository.

iii. CodeBuild

- Create a new build project.
- Configure the build settings, including the source (CodeCommit), environment (Docker), and build commands (Docker build and push to ECR).

iv. CodeDeploy

- Create a new application and deployment group.
- Configure the deployment settings, including the deployment type (ECS), service role, and deployment configuration.
- Create a new deployment using the Docker image from ECR.

v. ECR (Elastic Container Registry)

- Create a new repository to store backend Docker images.

vi. ECS (Elastic Container Service)

- Create a new cluster.
- Configure the cluster settings, including networking and instance types.
- Create a new task definition, specifying backend Docker image from ECR.
- Create a new service using the task definition.