|  |
| --- |
| Université de Reims Champagne Ardenne |
| Rapport sur la Mise en place d'une solution de vote basée sur une application serverless et une solution d'intégration |
| RT0907 (DAS0902) - Programmation Cloud |

|  |
| --- |
| Idrissa Sow  25/09/2024 |

Table des matières

[1. Introduction 2](#_Toc178147239)

[2. Comparaison des plateformes serverless 2](#_Toc178147240)

[3. Architecture de la solution 2](#_Toc178147241)

[4. Détails techniques des composants 2](#_Toc178147242)

[5. Intégration des données 3](#_Toc178147243)

[6. Sécurité et gestion des accès 3](#_Toc178147244)

[7. Interface utilisateur (Page Web) 3](#_Toc178147245)

[8. Scalabilité et disponibilité 3](#_Toc178147246)

[9. Échanges et communication 3](#_Toc178147247)

[10. Conclusion 3](#_Toc178147248)

## 1. Introduction

* Présentation du projet : objectif et enjeux d'un système de vote serverless.
* Choix de la plateforme : justification du choix d’AWS pour le projet.

## 2. Comparaison des plateformes serverless

* **AWS Lambda** vs autres services serverless (Google Cloud Functions, Azure Functions).
* Avantages d'AWS : scalabilité, coût, intégration des services, sécurité.
* Justification du choix d'AWS pour ce projet.

## 3. Architecture de la solution

* **Vue d'ensemble** de l'architecture serverless.
* **Services AWS utilisés** :
  + **AWS Lambda** pour la liste des candidats et la collecte des votes.
  + **Amazon API Gateway** pour l'exposition des fonctions Lambda en tant qu’API RESTful.
  + **Amazon DynamoDB** ou **RDS** pour le stockage des candidats et des votes.
  + **AWS Step Functions** pour orchestrer les appels entre les fonctions Lambda.
  + **Amazon S3** et **CloudFront** pour héberger la page web du front-end.
* **Schéma d’architecture** (optionnel, avec diagramme).

## 4. Détails techniques des composants

* **Liste des candidats** :
  + Fonction Lambda interrogeant DynamoDB/RDS via API Gateway pour retourner la liste.
* **Collecte des votes** :
  + Fonction Lambda enregistrant les votes dans DynamoDB/RDS via API Gateway.
* **Orchestration et workflow** :
  + Utilisation de AWS Step Functions pour synchroniser les processus.

## 5. Intégration des données

* Explication du choix de la base de données : DynamoDB (NoSQL) ou RDS (SQL).
* Comment la base de données interagit avec les fonctions Lambda.
* Traitement des votes et mise à jour de l'état du vote via des transactions dans la base.

## 6. Sécurité et gestion des accès

* Utilisation d’AWS IAM pour la gestion des permissions.
* Mesures de sécurité pour protéger les données sensibles (chiffrement, accès restreint).
* Protection des endpoints API avec **Amazon Cognito** (gestion des utilisateurs).

## 7. Interface utilisateur (Page Web)

* Hébergement de la page web sur Amazon S3.
* Utilisation d’Amazon CloudFront pour la distribution à faible latence.
* Fonctionnement de l'interface web : affichage des candidats et envoi des votes.

## 8. Scalabilité et disponibilité

* Explication de la scalabilité automatique de Lambda.
* Haute disponibilité garantie par DynamoDB et les services AWS.
* Gestion de la montée en charge (augmentation des utilisateurs).

## 9. Échanges et communication

* Communication entre les services : Lambda, API Gateway, DynamoDB/RDS.
* Gestion des requêtes synchrones (API Gateway) et asynchrones (SQS, SNS).

## 10. Conclusion

* Récapitulatif des avantages de la solution AWS.
* Résultats attendus : efficacité, sécurité, coût optimisé.
* Perspectives d’amélioration et d’évolution de la solution.