Guide de Déploiement AWS - Calculateur de Coûts AWS

Table des Matières

Vue d'ensemble

Ce guide détaille le déploiement d'une plateforme de calcul de coûts AWS construite avec Next.js, Firebase Authentication, et Firestore. La plateforme permet aux utilisateurs de calculer et d'estimer les coûts d'infrastructure AWS en temps réel.

Fonctionnalités

- Authentification utilisateur avec Firebase Auth
- Calculateur de coûts pour EC2, EBS, RDS
- Sauvegarde des configurations utilisateur
- Export en CSV avec colonnes personnalisées (ENV, QUOI)
- Interface responsive et moderne

Prérequis

Outils Requis

- **AWS CLI** configuré avec permissions appropriées
- **Node.js** 18+ et **npm/pnpm**
- **Compte Firebase** avec projet configuré
- **Git** pour le contrôle de version
- **Certificat SSL** (via AWS Certificate Manager)

Permissions AWS Minimales

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:*",
        "cloudfront:*",
        "route53:*",
        "certificatemanager:*",
        "iam:*",
```

```
        "cloudformation:*"
      ],
      "Resource": "*"
    }
  ]
}
```

Architecture Recommandée

Architecture de Production

```
Internet → CloudFront → S3 (Next.js Static) → Firebase
    ↓
Route 53 (DNS) → ACM (SSL)
    ↓
CloudWatch (Monitoring)
```

Composants AWS

1. **S3 Bucket** - Hébergement statique
2. **CloudFront** - CDN pour performance globale
3. **Route 53** - Gestion DNS
4. **Certificate Manager** - Certificats SSL
5. **CloudWatch** - Monitoring et logs
6. **IAM** - Gestion des accès

Configuration Firebase

1. Création du Projet Firebase

```
# Installer Firebase CLI
npm install -g firebase-tools

# Se connecter à Firebase
firebase login

# Initialiser le projet
firebase init hosting
```

2. Configuration Firestore

```
// Règles de sécurité Firestore
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /saved-sessions/{sessionId} {
      allow read, write: if request.auth != null &&
        request.auth.uid == resource.data.userId;
      allow create: if request.auth != null &&
```

```
        request.auth.uid == request.resource.data.userId;
    }
  }
}
```

3. Variables d'Environnement

```
# .env.production
NEXT_PUBLIC_FIREBASE_API_KEY=votre_cle_api
NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN=votre-projet.firebaseapp.com
NEXT_PUBLIC_FIREBASE_PROJECT_ID=votre-projet-id
NEXT_PUBLIC_FIREBASE_STORAGE_BUCKET=votre-projet.appspot.com
NEXT_PUBLIC_FIREBASE_MESSAGING_SENDER_ID=votre_sender_id
NEXT_PUBLIC_FIREBASE_APP_ID=votre_app_id
```

Déploiement sur AWS

Étape 1: Préparation du Build

```
# Cloner le repository
git clone <votre-repo>
cd awsdashboard-main

# Installer les dépendances
npm install

# Configurer les variables d'environnement
cp .env.local.example .env.local
# Modifier .env.local avec vos vraies valeurs Firebase

# Build de production
npm run build
npm run export
```

Étape 2: Création du Bucket S3

```
# Créer le bucket S3
aws s3 mb s3://votre-calculateur-aws-costs --region eu-west-1

# Configurer pour hébergement web statique
aws s3 website s3://votre-calculateur-aws-costs \
  --index-document index.html \
  --error-document error.html
```

Étape 3: Upload des Fichiers

```
# Synchroniser les fichiers
aws s3 sync ./out s3://votre-calculateur-aws-costs \
  --delete \
  --cache-control max-age=86400 \
  --metadata-directive REPLACE

# Configurer les permissions
aws s3api put-bucket-policy \
  --bucket votre-calculateur-aws-costs \
  --policy '{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "PublicReadGetObject",
        "Effect": "Allow",
        "Principal": "*",
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::votre-calculateur-aws-costs/*"
      }
    ]
  }'
```

Étape 4: Configuration CloudFront

```
// cloudfront-config.json
{
  "CallerReference": "aws-cost-calculator-2024",
  "Comment": "CDN pour Calculateur de Coûts AWS",
  "DefaultCacheBehavior": {
    "TargetOriginId": "S3-votre-calculateur-aws-costs",
    "ViewerProtocolPolicy": "redirect-to-https",
    "Compress": true,
    "ForwardedValues": {
      "QueryString": false,
      "Cookies": {"Forward": "none"}
    },
    "TrustedSigners": {
      "Enabled": false,
      "Quantity": 0
    },
    "MinTTL": 0,
    "DefaultTTL": 86400,
    "MaxTTL": 31536000
  },
  "Origins": {
    "Quantity": 1,
    "Items": [
      {
        "Id": "S3-votre-calculateur-aws-costs",
        "DomainName": "votre-calculateur-aws-costs.s3.amazonaws.com",
```

```
        "S3OriginConfig": {
          "OriginAccessIdentity": ""
        }
      }
    ]
  },
  "Enabled": true,
  "PriceClass": "PriceClass_100"
}
```

Étape 5: Certificat SSL

```
# Demander un certificat SSL (région us-east-1 obligatoire pour CloudFront)
aws acm request-certificate \
  --domain-name calculateur-aws.votre-domaine.com \
  --validation-method DNS \
  --region us-east-1
```

Étape 6: Configuration Route 53

```
# Créer la zone hébergée
aws route53 create-hosted-zone \
  --name votre-domaine.com \
  --caller-reference $(date +%s)

# Ajouter les enregistrements A et AAAA pour CloudFront
aws route53 change-resource-record-sets \
  --hosted-zone-id VOTRE_ZONE_ID \
  --change-batch '{
    "Changes": [
      {
        "Action": "CREATE",
        "ResourceRecordSet": {
          "Name": "calculateur-aws.votre-domaine.com",
          "Type": "A",
          "AliasTarget": {
            "DNSName": "votre-distribution-cloudfront.cloudfront.net",
            "EvaluateTargetHealth": false,
            "HostedZoneId": "Z2FDTNDATAQYW2"
          }
        }
      }
    ]
  }'
```

Optimisations

1. Performance Next.js

```
// next.config.js
/** @type {import('next').NextConfig} */
const nextConfig = {
  output: 'export',
  trailingSlash: true,
  images: {
    unoptimized: true
  },
  experimental: {
    optimizeCss: true
  },
  compiler: {
    removeConsole: process.env.NODE_ENV === 'production'
  }
}

module.exports = nextConfig
```

2. Optimisation Bundle

```
{
  "scripts": {
    "build:optimized": "NODE_ENV=production next build && next export",
    "analyze": "ANALYZE=true npm run build"
  }
}
```

3. Cache Headers S3

```
# Optimiser les headers de cache
aws s3 cp ./out s3://votre-calculateur-aws-costs \
  --recursive \
  --cache-control "public, max-age=31536000, immutable" \
  --exclude "*.html" \
  --exclude "sw.js"

# Headers spéciaux pour HTML
aws s3 cp ./out s3://votre-calculateur-aws-costs \
  --recursive \
  --cache-control "public, max-age=0, must-revalidate" \
  --include "*.html"
```

4. Compression CloudFront

```
// Configuration avancée CloudFront
{
  "CacheBehaviors": [
    {
```

```
      "PathPattern": "*.js",
      "Compress": true,
      "DefaultTTL": 31536000,
      "ViewerProtocolPolicy": "redirect-to-https"
    },
    {
      "PathPattern": "*.css",
      "Compress": true,
      "DefaultTTL": 31536000,
      "ViewerProtocolPolicy": "redirect-to-https"
    }
  ]
}
```

Sécurité

1. Politique de Sécurité Contenu (CSP)

```
<!-- À ajouter dans les headers -->
<meta http-equiv="Content-Security-Policy"
      content="default-src 'self';
               script-src 'self' 'unsafe-inline' *.googleapis.com *.gstatic.com;
               style-src 'self' 'unsafe-inline' *.googleapis.com;
               connect-src 'self' *.googleapis.com *.firebaseio.com
*.cloudfunctions.net;">
```

2. Configuration WAF (optionnel)

```
{
  "Rules": [
    {
      "Name": "AWSManagedRulesCommonRuleSet",
      "Priority": 1,
      "Statement": {
        "ManagedRuleGroupStatement": {
          "VendorName": "AWS",
          "Name": "AWSManagedRulesCommonRuleSet"
        }
      }
    }
  ]
}
```

3. Restrictions Géographiques

```
// CloudFront Geo Restriction
{
  "GeoRestriction": {
```

```
    "RestrictionType": "whitelist",
    "Locations": ["FR", "BE", "CH", "CA"]
  }
}
```

Surveillance et Monitoring

1. CloudWatch Alarms

```
# Alarme pour erreurs 4xx
aws cloudwatch put-metric-alarm \
  --alarm-name "CalculateurAWS-4xxErrors" \
  --alarm-description "Erreurs 4xx élevées" \
  --metric-name "4xxErrorRate" \
  --namespace "AWS/CloudFront" \
  --statistic "Sum" \
  --period 300 \
  --threshold 10 \
  --comparison-operator "GreaterThanThreshold"

# Alarme pour latence
aws cloudwatch put-metric-alarm \
  --alarm-name "CalculateurAWS-HighLatency" \
  --alarm-description "Latence élevée" \
  --metric-name "OriginLatency" \
  --namespace "AWS/CloudFront" \
  --statistic "Average" \
  --period 300 \
  --threshold 1000 \
  --comparison-operator "GreaterThanThreshold"
```

2. Logs Analytics

```
// Configuration logs CloudFront
{
  "Logging": {
    "Enabled": true,
    "IncludeCookies": false,
    "Bucket": "logs-calculateur-aws.s3.amazonaws.com",
    "Prefix": "access-logs/"
  }
}
```

3. Dashboard CloudWatch

```
{
  "widgets": [
    {
```

```
      "type": "metric",
      "properties": {
        "metrics": [
          ["AWS/CloudFront", "Requests", "DistributionId", "VOTRE_DISTRIBUTION_ID"],
          [".", "BytesDownloaded", ".", "."],
          [".", "4xxErrorRate", ".", "."],
          [".", "5xxErrorRate", ".", "."]
        ],
        "period": 300,
        "stat": "Sum",
        "region": "us-east-1",
        "title": "Métriques Calculateur AWS"
      }
    }
  ]
}
```

Scripts d'Automatisation

1. Script de Déploiement

```
#!/bin/bash
# deploy.sh

set -e

echo "🚀 Déploiement du Calculateur de Coûts AWS"

# Variables
BUCKET_NAME="votre-calculateur-aws-costs"
DISTRIBUTION_ID="VOTRE_DISTRIBUTION_ID"

# Build
echo "📦 Construction de l'application..."
npm run build
npm run export

# Upload vers S3
echo "☁️ Upload vers S3..."
aws s3 sync ./out s3://$BUCKET_NAME \
  --delete \
  --cache-control max-age=86400

# Invalidation CloudFront
echo "🔄 Invalidation du cache CloudFront..."
aws cloudfront create-invalidation \
  --distribution-id $DISTRIBUTION_ID \
  --paths "/*"
```

```
echo "✅ Déploiement terminé!"
echo "🌐 URL: https://calculateur-aws.votre-domaine.com"
```

2. Script de Surveillance

```
#!/bin/bash
# monitor.sh

# Vérification de santé
HEALTH_URL="https://calculateur-aws.votre-domaine.com"
STATUS=$(curl -s -o /dev/null -w "%{http_code}" $HEALTH_URL)

if [ $STATUS -eq 200 ]; then
  echo "✅ Site accessible - Status: $STATUS"
else
  echo "❌ Site inaccessible - Status: $STATUS"
  # Envoyer notification (SNS, Slack, etc.)
fi

# Vérification des métriques CloudWatch
aws cloudwatch get-metric-statistics \
  --namespace AWS/CloudFront \
  --metric-name Requests \
  --dimensions Name=DistributionId,Value=$DISTRIBUTION_ID \
  --start-time $(date -u -d '1 hour ago' +%Y-%m-%dT%H:%M:%S) \
  --end-time $(date -u +%Y-%m-%dT%H:%M:%S) \
  --period 3600 \
  --statistics Sum
```

Maintenance

1. Mises à Jour Régulières

```
# Mise à jour des dépendances
npm audit
npm update

# Vérification de sécurité
npm audit fix

# Test avant déploiement
npm run test
npm run build
```

2. Sauvegarde Firebase

```
# Export Firestore
gcloud firestore export gs://votre-bucket-backup/$(date +%Y-%m-%d)
```

```
# Sauvegarde des règles
firebase firestore:rules > firestore-rules-$(date +%Y-%m-%d).rules
```

3. Rotation des Certificats

```
# Vérifier l'expiration du certificat
aws acm describe-certificate \
  --certificate-arn $CERT_ARN \
  --query 'Certificate.NotAfter'
```

Coûts Estimés

Estimation Mensuelle (Trafic Moyen)

- **S3 (20 GB)**: ~0.50€
- **CloudFront (100 GB)**: ~8.50€
- **Route 53 (1 zone)**: ~0.50€
- **Certificate Manager**: Gratuit
- **CloudWatch (logs basiques)**: ~2.00€
- **Firebase (utilisateurs actifs < 50k)**: Gratuit

Total estimé: ~11.50€/mois

Optimisation des Coûts

1. **Utiliser S3 Intelligent Tiering**
2. **Configurer des politiques de lifecycle S3**
3. **Optimiser les classes de prix CloudFront**
4. **Utiliser des reserved instances pour trafic prévisible**

Conclusion

Ce guide fournit une architecture robuste et optimisée pour déployer votre calculateur de coûts AWS. La solution est:

- **Scalable**: Supporte la croissance du trafic
- **Sécurisée**: Bonnes pratiques de sécurité appliquées
- **Performante**: CDN global et optimisations
- **Économique**: Coûts maîtrisés et prévisibles
- **Maintenable**: Scripts d'automatisation et monitoring

Pour toute question ou assistance, consultez la documentation AWS ou contactez votre équipe DevOps.

Guide créé pour le déploiement du Calculateur de Coûts AWS - Version 1.0