Guide Complet : Sécurisation Applications et Stockage AWS pour SaaS

Version: 1.0
Date: Novembre 2025
Destiné à: Architectes Applications et Équipes Backend

Table des Matières

Sécurité Amazon S3

1. Chiffrement et Protection des Données

1.1 Chiffrement Par Défaut Activé

Depuis 2023: Tous les nouveaux buckets S3 ont le chiffrement activé par défaut (SSE-S3). Cependant, pour une sécurité renforcée, utilisez SSE-KMS.

```
# Activer le chiffrement par défaut avec KMS
aws s3api put-bucket-encryption \
    --bucket my-saas-bucket \
    --server-side-encryption-configuration '{
      "Rules": [{
        "ApplyServerSideEncryptionByDefault": {
          "SSEAlgorithm": "aws:kms",
```

```
          "KMSMasterKeyID": "arn:aws:kms:us-east-1:123456789012:key/xxxxx"
        },
        "BucketKeyEnabled": true
      }]
    }'
```

Avantages SSE-KMS:
- ✅ **Audit via CloudTrail** : Qui accède à quelles données
- ✅ **Contrôle granulaire** : Politiques KMS par tenant/application
- ✅ **Rotation automatique** : Rotation annuelle des clés
- ✅ **Compliance** : Requis pour PCI-DSS, HIPAA

1.2 Forcer HTTPS (TLS) Uniquement

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyInsecureConnections",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::my-saas-bucket",
        "arn:aws:s3:::my-saas-bucket/*"
      ],
      "Condition": {
        "Bool": {
          "aws:SecureTransport": "false"
        }
      }
    }
  ]
}
```

```
# Appliquer la politique
aws s3api put-bucket-policy \
    --bucket my-saas-bucket \
    --policy file://enforce-https-policy.json
```

1.3 Versioning et MFA Delete

```
# Activer le versioning
aws s3api put-bucket-versioning \
    --bucket my-saas-bucket \
    --versioning-configuration Status=Enabled

# Activer MFA Delete (nécessite le compte root)
aws s3api put-bucket-versioning \
```

```
    --bucket my-saas-bucket \
    --versioning-configuration Status=Enabled,MFADelete=Enabled \
    --mfa "arn:aws:iam::123456789012:mfa/root-account-mfa-device 123456"
```

MFA Delete nécessite une authentification MFA pour:
- ❌ Supprimer une version d'objet
- ❌ Désactiver le versioning

2. Contrôle d'Accès et Isolation

2.1 Block Public Access (OBLIGATOIRE)

```
# Activer Block Public Access au niveau du compte
aws s3control put-public-access-block \
    --account-id 123456789012 \
    --public-access-block-configuration \
        BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBucket

# Activer au niveau du bucket
aws s3api put-public-access-block \
    --bucket my-saas-bucket \
    --public-access-block-configuration \
        BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBucket
```

Statistique: Les buckets S3 mal configurés restent l'une des principales causes d'incidents de sécurité cloud en 2025.

2.2 Politique Bucket Basée sur le Moindre Privilège

❌ Dangereux:

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:*",
    "Resource": "arn:aws:s3:::my-bucket/*"
  }]
}
```

✅ Sécurisé - Multi-Tenant avec Préfixes:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Sid": "AllowTenantAccess",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/TenantApplicationRole"
      },
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": "arn:aws:s3:::my-saas-bucket/${aws:PrincipalTag/TenantID}/*"
    }
  ]
}
```

Cette politique permet aux utilisateurs d'accéder uniquement aux objets sous leur préfixe tenant.

2.3 S3 Access Points pour Multi-Tenant

```
# Créer un Access Point par tenant
aws s3control create-access-point \
    --account-id 123456789012 \
    --name tenant-a-access-point \
    --bucket my-saas-bucket \
    --policy '{
      "Version": "2012-10-17",
      "Statement": [{
        "Effect": "Allow",
        "Principal": {"AWS": "arn:aws:iam::123456789012:role/TenantARole"},
        "Action": ["s3:GetObject", "s3:PutObject"],
        "Resource": "arn:aws:s3:us-east-1:123456789012:accesspoint/tenant-a-access-point/objec
      }]
    }'
```

Avantages S3 Access Points:
- ✅ Politique d'accès dédiée par tenant
- ✅ Isolation simplifiée
- ✅ Aucune modification de la politique bucket principale

3. Journalisation et Surveillance

3.1 Activer S3 Server Access Logs

```
# Activer les logs d'accès
aws s3api put-bucket-logging \
    --bucket my-saas-bucket \
    --bucket-logging-status '{
      "LoggingEnabled": {
```

```
        "TargetBucket": "my-s3-access-logs",
        "TargetPrefix": "logs/my-saas-bucket/"
      }
    }'
```

3.2 S3 Event Notifications

```
# Configurer des notifications pour les modifications d'objets
aws s3api put-bucket-notification-configuration \
    --bucket my-saas-bucket \
    --notification-configuration '{
      "LambdaFunctionConfigurations": [{
        "Id": "ObjectCreatedAlert",
        "LambdaFunctionArn": "arn:aws:lambda:us-east-1:123456789012:function:S3SecurityAudit",
        "Events": ["s3:ObjectCreated:*", "s3:ObjectRemoved:*"]
      }]
    }'
```

3.3 CloudTrail pour Audit S3

```
# Activer les événements de données S3 dans CloudTrail
aws cloudtrail put-event-selectors \
    --trail-name MyTrail \
    --event-selectors '[{
      "ReadWriteType": "All",
      "IncludeManagementEvents": true,
      "DataResources": [{
        "Type": "AWS::S3::Object",
        "Values": ["arn:aws:s3:::my-saas-bucket/*"]
      }]
    }]'
```

4. Lifecycle et Gouvernance

4.1 Politique de Lifecycle

```
{
  "Rules": [
    {
      "Id": "TransitionToInfrequentAccess",
      "Status": "Enabled",
      "Filter": {
        "Prefix": "archives/"
      },
      "Transitions": [
        {
          "Days": 30,
          "StorageClass": "STANDARD_IA"
        },
```

```
        {
          "Days": 90,
          "StorageClass": "GLACIER"
        }
      ]
    },
    {
      "Id": "DeleteOldVersions",
      "Status": "Enabled",
      "NoncurrentVersionExpiration": {
        "NoncurrentDays": 90
      }
    }
  ]
}
```

```
aws s3api put-bucket-lifecycle-configuration \
    --bucket my-saas-bucket \
    --lifecycle-configuration file://lifecycle.json
```

Sécurité Amazon RDS

1. Chiffrement

1.1 Chiffrement au Repos

```
# Créer une instance RDS avec chiffrement
aws rds create-db-instance \
    --db-instance-identifier prod-database \
    --db-instance-class db.r6g.xlarge \
    --engine postgres \
    --master-username admin \
    --master-user-password $(aws secretsmanager get-secret-value --secret-id prod/db/password
    --allocated-storage 100 \
    --storage-encrypted \
    --kms-key-id arn:aws:kms:us-east-1:123456789012:key/xxxxx \
    --vpc-security-group-ids sg-xxxxx \
    --db-subnet-group-name private-db-subnet-group \
    --multi-az \
    --backup-retention-period 30 \
    --preferred-backup-window "03:00-04:00" \
    --preferred-maintenance-window "mon:04:00-mon:05:00"
```

Important: Le chiffrement ne peut être activé **qu'à la création**. Pour chiffrer une base existante :

1. Créer un snapshot

2. Copier le snapshot avec chiffrement
3. Restaurer une nouvelle instance depuis le snapshot chiffré

1.2 Chiffrement en Transit (SSL/TLS)

```
-- PostgreSQL: Forcer SSL
ALTER SYSTEM SET ssl = on;
ALTER SYSTEM SET ssl_min_protocol_version = 'TLSv1.2';

-- Vérifier les connexions SSL
SELECT datname, usename, client_addr, ssl, cipher
FROM pg_stat_ssl
JOIN pg_stat_activity ON pg_stat_ssl.pid = pg_stat_activity.pid;
```

```
# Télécharger le certificat RDS
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem

# Connexion PostgreSQL avec SSL
psql "host=mydb.xxxxx.us-east-1.rds.amazonaws.com port=5432 dbname=mydb user=admin sslmode=ver
```

2. Isolation Réseau

2.1 Sous-Réseaux Privés UNIQUEMENT

```
# Terraform - DB Subnet Group dans sous-réseaux privés
resource "aws_db_subnet_group" "private_db_subnet" {
  name       = "private-db-subnet-group"
  subnet_ids = [
    aws_subnet.private_db_1.id,
    aws_subnet.private_db_2.id
  ]

  tags = {
    Name = "Private DB Subnet Group"
  }
}

# Security Group - Accès uniquement depuis l'application tier
resource "aws_security_group" "rds_sg" {
  name        = "rds-security-group"
  description = "Allow access from application tier only"
  vpc_id      = aws_vpc.main.id

  ingress {
    description     = "PostgreSQL from App Tier"
    from_port       = 5432
    to_port         = 5432
    protocol        = "tcp"
    security_groups = [aws_security_group.app_tier.id]
```

```
  }

  egress {
    description = "No outbound required"
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

3. Sauvegardes et Récupération

3.1 Sauvegardes Automatiques

```
# Modifier la période de rétention des sauvegardes
aws rds modify-db-instance \
    --db-instance-identifier prod-database \
    --backup-retention-period 30 \
    --preferred-backup-window "03:00-04:00" \
    --apply-immediately
```

3.2 Snapshots Manuels

```
# Créer un snapshot manuel
aws rds create-db-snapshot \
    --db-instance-identifier prod-database \
    --db-snapshot-identifier prod-db-snapshot-$(date +%Y%m%d-%H%M%S)

# Copier vers une autre région (DR)
aws rds copy-db-snapshot \
    --source-db-snapshot-identifier arn:aws:rds:us-east-1:123456789012:snapshot:prod-db-snapsh
    --target-db-snapshot-identifier prod-db-snapshot-20251115-dr \
    --source-region us-east-1 \
    --region eu-west-1 \
    --kms-key-id arn:aws:kms:eu-west-1:123456789012:key/yyyyy
```

3.3 Point-in-Time Recovery

```
# Restaurer à un point dans le temps spécifique
aws rds restore-db-instance-to-point-in-time \
    --source-db-instance-identifier prod-database \
    --target-db-instance-identifier prod-database-restored \
    --restore-time 2025-11-15T10:30:00Z \
    --db-subnet-group-name private-db-subnet-group \
    --vpc-security-group-ids sg-xxxxx
```

4. Gestion des Secrets avec AWS Secrets Manager

```
import boto3
import json
import psycopg2

def get_db_connection():
    """Obtenir une connexion DB en récupérant les credentials depuis Secrets Manager"""
    client = boto3.client('secretsmanager')

    try:
        response = client.get_secret_value(SecretId='prod/database/credentials')
        secret = json.loads(response['SecretString'])

        # Connexion PostgreSQL
        conn = psycopg2.connect(
            host=secret['host'],
            port=secret['port'],
            database=secret['dbname'],
            user=secret['username'],
            password=secret['password'],
            sslmode='verify-full',
            sslrootcert='/path/to/global-bundle.pem'
        )

        return conn
    except Exception as e:
        print(f"Error connecting to database: {e}")
        raise
```

4.1 Rotation Automatique des Mots de Passe

```
# Activer la rotation automatique (tous les 30 jours)
aws secretsmanager rotate-secret \
    --secret-id prod/database/credentials \
    --rotation-lambda-arn arn:aws:lambda:us-east-1:123456789012:function:RDSPasswordRotation \
    --rotation-rules AutomaticallyAfterDays=30
```

5. Monitoring et Audit

5.1 Enhanced Monitoring

```
# Activer Enhanced Monitoring
aws rds modify-db-instance \
    --db-instance-identifier prod-database \
    --monitoring-interval 60 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/rds-monitoring-role
```

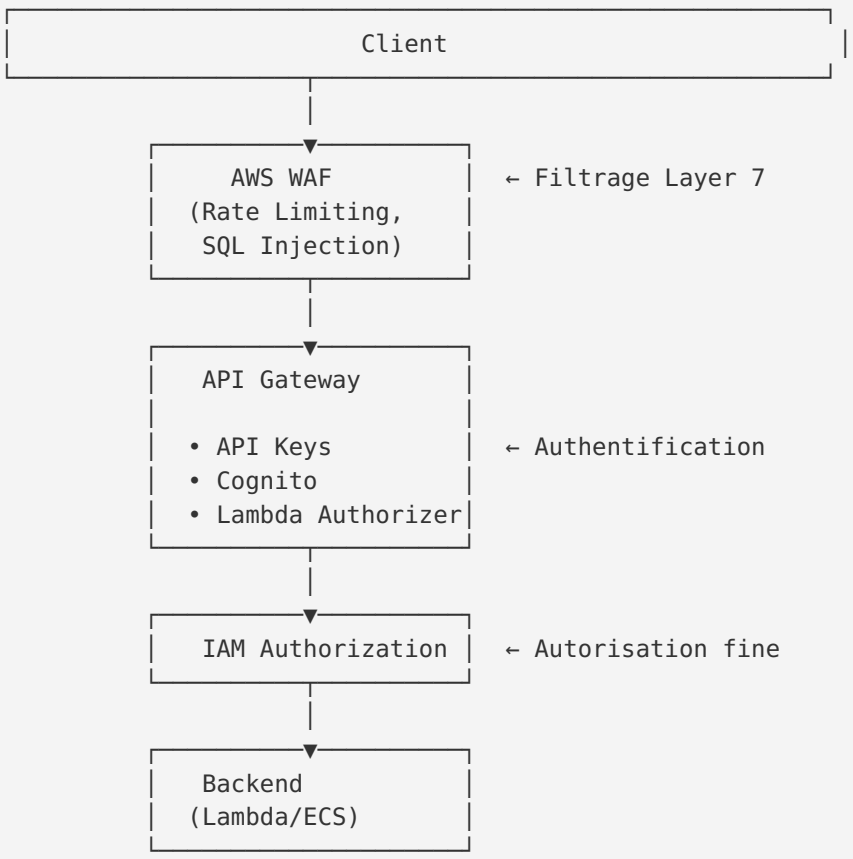
5.2 Performance Insights

```
# Activer Performance Insights
aws rds modify-db-instance \
    --db-instance-identifier prod-database \
    --enable-performance-insights \
    --performance-insights-retention-period 7
```

Sécurité API Gateway

1. Authentification et Autorisation

1.1 Architecture de Sécurité Multi-Couches



1.2 Cognito User Pools

```
# CloudFormation - API Gateway avec Cognito
Resources:
  MyApi:
    Type: AWS::ApiGatewayV2::Api
```

```
    Properties:
      Name: SecureAPI
      ProtocolType: HTTP

  CognitoAuthorizer:
    Type: AWS::ApiGatewayV2::Authorizer
    Properties:
      Name: CognitoAuthorizer
      ApiId: !Ref MyApi
      AuthorizerType: JWT
      IdentitySource:
        - $request.header.Authorization
      JwtConfiguration:
        Audience:
          - !Ref UserPoolClient
        Issuer: !Sub https://cognito-idp.${AWS::Region}.amazonaws.com/${UserPool}

  SecureRoute:
    Type: AWS::ApiGatewayV2::Route
    Properties:
      ApiId: !Ref MyApi
      RouteKey: "GET /secure"
      AuthorizationType: JWT
      AuthorizerId: !Ref CognitoAuthorizer
      Target: !Sub integrations/${MyIntegration}
```

1.3 Lambda Authorizer Personnalisé

```
import json

def lambda_handler(event, context):
    """Lambda Authorizer avec validation de token JWT custom"""
    token = event['authorizationToken']

    # Valider le token (ex: JWT, API key custom, etc.)
    if validate_token(token):
        principal_id = get_principal_id(token)
        tenant_id = get_tenant_id(token)

        # Générer la politique IAM
        policy = generate_policy(principal_id, 'Allow', event['methodArn'], tenant_id)

        # Ajouter le context (disponible dans le backend)
        policy['context'] = {
            'tenantId': tenant_id,
            'userId': principal_id,
            'permissions': get_user_permissions(principal_id)
        }

        return policy
    else:
        # Token invalide - refuser
        return generate_policy('user', 'Deny', event['methodArn'], None)
```

```
def generate_policy(principal_id, effect, resource, tenant_id):
    """Générer une politique IAM"""
    auth_response = {
        'principalId': principal_id
    }

    if effect and resource:
        policy_document = {
            'Version': '2012-10-17',
            'Statement': [{
                'Action': 'execute-api:Invoke',
                'Effect': effect,
                'Resource': resource
            }]
        }
        auth_response['policyDocument'] = policy_document

    return auth_response

def validate_token(token):
    """Valider le token (implémenter votre logique)"""
    # Ex: Vérifier signature JWT, expiration, issuer, etc.
    return True  # Placeholder

def get_principal_id(token):
    """Extraire l'ID utilisateur du token"""
    return "user-123"  # Placeholder

def get_tenant_id(token):
    """Extraire l'ID tenant du token"""
    return "tenant-a"  # Placeholder

def get_user_permissions(user_id):
    """Récupérer les permissions utilisateur"""
    return "read,write"  # Placeholder
```

2. AWS WAF pour API Gateway

2.1 Configuration WAF

```
# Créer un Web ACL
aws wafv2 create-web-acl \
    --name api-gateway-waf \
    --scope REGIONAL \
    --default-action Allow={} \
    --rules file://waf-rules.json \
    --visibility-config SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=A

# Associer au API Gateway
aws wafv2 associate-web-acl \
```

```
    --web-acl-arn arn:aws:wafv2:us-east-1:123456789012:regional/webacl/api-gateway-waf/xxxxx \
    --resource-arn arn:aws:apigateway:us-east-1::/restapis/xxxxx/stages/prod
```

2.2 Règles WAF Essentielles

```
{
  "Name": "RateLimitRule",
  "Priority": 1,
  "Statement": {
    "RateBasedStatement": {
      "Limit": 2000,
      "AggregateKeyType": "IP"
    }
  },
  "Action": {
    "Block": {}
  },
  "VisibilityConfig": {
    "SampledRequestsEnabled": true,
    "CloudWatchMetricsEnabled": true,
    "MetricName": "RateLimitRule"
  }
}
```

3. Throttling et Quotas

```
# Créer un Usage Plan
aws apigateway create-usage-plan \
    --name "Standard-Plan" \
    --throttle burstLimit=500,rateLimit=1000 \
    --quota limit=100000,period=DAY \
    --api-stages apiId=xxxxx,stage=prod

# Créer une API Key
aws apigateway create-api-key \
    --name "TenantA-API-Key" \
    --enabled

# Associer l'API Key au Usage Plan
aws apigateway create-usage-plan-key \
    --usage-plan-id xxxxx \
    --key-id yyyyy \
    --key-type API_KEY
```

4. Logs et Monitoring

4.1 Activer CloudWatch Logs

```
# Activer les logs d'exécution
aws apigateway update-stage \
    --rest-api-id xxxxx \
    --stage-name prod \
    --patch-operations \
        op=replace,path=/*/logging/dataTrace,value=true \
        op=replace,path=/*/logging/loglevel,value=INFO
```

4.2 Métriques CloudWatch Personnalisées

```
import boto3
import time

cloudwatch = boto3.client('cloudwatch')

def lambda_handler(event, context):
    start_time = time.time()

    try:
        # Traiter la requête
        result = process_request(event)

        # Métrique de succès
        cloudwatch.put_metric_data(
            Namespace='MyAPI/Requests',
            MetricData=[{
                'MetricName': 'SuccessfulRequests',
                'Value': 1,
                'Unit': 'Count',
                'Dimensions': [
                    {'Name': 'Endpoint', 'Value': event['resource']},
                    {'Name': 'TenantId', 'Value': event['requestContext']['authorizer']['tenan
                ]
            }]
        )

        return result
    except Exception as e:
        # Métrique d'erreur
        cloudwatch.put_metric_data(
            Namespace='MyAPI/Requests',
            MetricData=[{
                'MetricName': 'FailedRequests',
                'Value': 1,
                'Unit': 'Count'
            }]
        )
        raise
```

```
    finally:
        # Métrique de latence
        duration = (time.time() - start_time) * 1000
        cloudwatch.put_metric_data(
            Namespace='MyAPI/Performance',
            MetricData=[{
                'MetricName': 'RequestDuration',
                'Value': duration,
                'Unit': 'Milliseconds'
            }]
        )
```

Sécurité DynamoDB

1. Chiffrement

1.1 Chiffrement au Repos

```
# Créer une table avec chiffrement KMS
aws dynamodb create-table \
    --table-name UserData \
    --attribute-definitions \
        AttributeName=UserId,AttributeType=S \
        AttributeName=TenantId,AttributeType=S \
    --key-schema \
        AttributeName=TenantId,KeyType=HASH \
        AttributeName=UserId,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST \
    --sse-specification \
        Enabled=true,SSEType=KMS,KMSMasterKeyId=arn:aws:kms:us-east-1:123456789012:key/xxxxx
```

2. Point-in-Time Recovery (PITR)

```
# Activer PITR
aws dynamodb update-continuous-backups \
    --table-name UserData \
    --point-in-time-recovery-specification PointInTimeRecoveryEnabled=true

# Restaurer à un point dans le temps
aws dynamodb restore-table-to-point-in-time \
    --source-table-name UserData \
    --target-table-name UserData-Restored \
    --restore-date-time 2025-11-15T10:00:00Z
```

3. Contrôle d'Accès Fine-Grained

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:PutItem",
        "dynamodb:Query",
        "dynamodb:UpdateItem",
        "dynamodb:DeleteItem"
      ],
      "Resource": "arn:aws:dynamodb:us-east-1:123456789012:table/UserData",
      "Condition": {
        "ForAllValues:StringEquals": {
          "dynamodb:LeadingKeys": ["${aws:PrincipalTag/TenantID}"]
        }
      }
    }
  ]
}
```

Cette politique permet aux utilisateurs d'accéder uniquement aux items dont la partition key correspond à leur TenantID.

4. DynamoDB Streams pour Audit

```
import boto3
import json

def lambda_handler(event, context):
    """Auditeur DynamoDB Streams"""
    for record in event['Records']:
        if record['eventName'] == 'REMOVE':
            # Enregistrer la suppression pour audit
            log_deletion(record)
        elif record['eventName'] in ['INSERT', 'MODIFY']:
            # Détecter les modifications sensibles
            detect_sensitive_changes(record)

def log_deletion(record):
    """Logger les suppressions pour audit"""
    print(f"Item deleted: {record['dynamodb']['Keys']}")
    # Envoyer à CloudWatch Logs, S3, etc.

def detect_sensitive_changes(record):
    """Détecter les modifications de données sensibles"""
    if 'NewImage' in record['dynamodb']:
        new_image = record['dynamodb']['NewImage']
```

```
        if 'email' in new_image or 'phone' in new_image:
            print(f"Sensitive data modified: {record['dynamodb']['Keys']}")
            # Alerter l'équipe sécurité
```

Bonnes Pratiques Multi-Services

1. Defense-in-Depth

```
┌─────────────────────────────────────────────────────┐
│  Layer 7: Application Logic Validation              │
├─────────────────────────────────────────────────────┤
│  Layer 6: Business Rules Authorization              │
├─────────────────────────────────────────────────────┤
│  Layer 5: IAM Policies (Resource-level)             │
├─────────────────────────────────────────────────────┤
│  Layer 4: API Gateway Authorizers                   │
├─────────────────────────────────────────────────────┤
│  Layer 3: AWS WAF (Rate Limiting, SQL Injection)    │
├─────────────────────────────────────────────────────┤
│  Layer 2: Network Security (Security Groups, NACLs) │
├─────────────────────────────────────────────────────┤
│  Layer 1: Encryption (Data at Rest & in Transit)    │
└─────────────────────────────────────────────────────┘
```

2. Principe du Moindre Privilège

Chaque composant doit avoir uniquement les permissions nécessaires:

- **Lambda** → Accès DynamoDB spécifique (table + leading keys)
- **API Gateway** → Invocation Lambda spécifique
- **Application** → Accès S3 limité à un préfixe tenant

3. Surveillance Continue

```
# AWS Config Rules pour conformité
aws configservice put-config-rule --config-rule '{
  "ConfigRuleName": "s3-bucket-public-read-prohibited",
  "Source": {
    "Owner": "AWS",
    "SourceIdentifier": "S3_BUCKET_PUBLIC_READ_PROHIBITED"
  }
}'
```

Checklist Applications & Stockage

✅ Amazon S3 (Priorité Critique)

- [ ] **Block Public Access activé (compte + buckets)**
- [ ] **Chiffrement par défaut SSE-KMS**
- [ ] **HTTPS obligatoire (politique bucket)**
- [ ] **Versioning activé**
- [ ] **MFA Delete activé sur buckets critiques**
- [ ] **Access Logs activés**
- [ ] **CloudTrail data events activés**
- [ ] **Lifecycle policies configurées**

✅ Amazon RDS (Priorité Critique)

- [ ] **Chiffrement activé (KMS)**
- [ ] **SSL/TLS obligatoire**
- [ ] **Sous-réseaux privés uniquement**
- [ ] **Security Groups restrictifs**
- [ ] **Backup rétention ≥ 30 jours**
- [ ] **Multi-AZ activé (production)**
- [ ] **Enhanced Monitoring activé**
- [ ] **Performance Insights activé**
- [ ] **Secrets Manager pour credentials**
- [ ] **Rotation automatique des mots de passe**

✅ API Gateway (Priorité Critique)

- [ ] **Authentification configurée (Cognito/Lambda Authorizer)**
- [ ] **AWS WAF activé et configuré**
- [ ] **Rate Limiting configuré**
- [ ] **Usage Plans et Quotas**
- [ ] **CloudWatch Logs activés**
- [ ] **X-Ray tracing activé**

- [ ] **CORS configuré correctement**
- [ ] **API Keys pour clients B2B**

✅ DynamoDB (Priorité Importante)

- [ ] **Chiffrement KMS activé**
- [ ] **Point-in-Time Recovery (PITR) activé**
- [ ] **Fine-grained access control (IAM)**
- [ ] **DynamoDB Streams pour audit**
- [ ] **Auto Scaling configuré**
- [ ] **Backups automatiques**
- [ ] **CloudWatch Contributor Insights activé**

Références et Ressources

Documentation Officielle AWS

- S3 Security Best Practices
- RDS Encryption Best Practices
- API Gateway Security
- DynamoDB Security

Conclusion

La sécurisation des applications et du stockage AWS nécessite:

1. **Chiffrement systématique** au repos et en transit
2. **Isolation multi-tenant** avec politiques IAM et préfixes
3. **Authentification forte** avec Cognito ou Lambda Authorizers
4. **Protection WAF** contre les attaques Layer 7
5. **Audit continu** avec CloudTrail et DynamoDB Streams
6. **Sauvegardes automatiques** avec rétention adaptée

Ces pratiques garantissent une infrastructure d'applications et de stockage conforme, sécurisée et résiliente pour vos applications SaaS critiques.

Document préparé pour: [Nom du Client]
Contact support: [Email de l'équipe Backend/Apps]
Dernière mise à jour: Novembre 2025