# Deployment Script Documentation

Aim: ■■ ■■■■■■■■■■ ■■ ■■■■■■■■■ AWS ■■ ■■ Full Stack ■■■■■■■■■■ ■■ CloudFormation ■■ ■■■■■■■ ■■ ■■■■■■■■ ■■■■■, ■■■■■■■■■■ ■■ S3/CloudFront ■■ ■■■■ ■■■■■, ■■■■■■■ Docker ■■■■■ ■■ ECR ■■ ■■■■ ■■■■■ ■■ ■■■■■■■ health-checks ■■■■ ■■■■■■■■ ■■■■■■■■ ■■■■■ ■■■■

## Theory:

Theory: ■■■■■■■■■■ CloudFormation ■■ ■■■■■■ ■■ ■■■■■■■■■ ■■ ■■■■■■■■/■■■■■■ ■■■■■ ■■■■
■■ AWS CLI, Docker, npm, ■■ CloudFront/S3 ■■ ■■■■ ■■■■■■■■■ ■■■■■ ■■■■ ■■■■■■■■■■ ■■■■ ■■■■■■ ■■■■■■/■■■■■■ ■■■■■, ■■■■■■■ ■■■■■■■■, ■■■■■■■■■■ ■■■■■■ ■■ S3 ■■■■■, CloudFront invalidation, Docker ■■■■■■/■■■ ■■ ■■■■■■■ health-check ■■■■■■ ■■■■■

## Cleaned Code (comments removed):

```
set -e
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
NC='\033[0;0m'
STACK_NAME="fullstack-app"
TEMPLATE_FILE="cloudformation-template.yaml"
ENVIRONMENT="production"
REGION="us-east-1"
print_info() {
    echo -e "${GREEN}[INFO]${NC} $1"
}
print_error() {
    echo -e "${RED}[ERROR]${NC} $1"
}
print_warning() {
    echo -e "${YELLOW}[WARNING]${NC} $1"
}
check_aws_cli() {
    if ! command -v aws &> /dev/null; then
        print_error "AWS CLI is not installed. Please install it first."
        exit 1
    fi
    print_info "AWS CLI found"
}
validate_template() {
    print_info "Validating CloudFormation template..."
    aws cloudformation validate-template \
        --template-body file://${TEMPLATE_FILE} \
        --region ${REGION} > /dev/null
    print_info "Template validation successful"
}
deploy_stack() {
    print_info "Deploying CloudFormation stack: ${STACK_NAME}"
    if aws cloudformation describe-stacks \
        --stack-name ${STACK_NAME} \
        --region ${REGION} &> /dev/null; then
        print_warning "Stack exists. Updating..."
        aws cloudformation update-stack \
            --stack-name ${STACK_NAME} \
            --template-body file://${TEMPLATE_FILE} \
            --parameters \
                ParameterKey=EnvironmentName,ParameterValue=${ENVIRONMENT} \
                ParameterKey=KeyName,ParameterValue=${EC2_KEY_NAME} \
                ParameterKey=DBUsername,ParameterValue=${DB_USERNAME} \
                ParameterKey=DBPassword,ParameterValue=${DB_PASSWORD} \
            --capabilities CAPABILITY_NAMED_IAM \
```

```
            --region ${REGION}
        print_info "Waiting for stack update to complete..."
        aws cloudformation wait stack-update-complete \
            --stack-name ${STACK_NAME} \
            --region ${REGION}
    else
        print_info "Creating new stack..."
        aws cloudformation create-stack \
            --stack-name ${STACK_NAME} \
            --template-body file://${TEMPLATE_FILE} \
            --parameters \
                ParameterKey=EnvironmentName,ParameterValue=${ENVIRONMENT} \
                ParameterKey=KeyName,ParameterValue=${EC2_KEY_NAME} \
                ParameterKey=DBUsername,ParameterValue=${DB_USERNAME} \
                ParameterKey=DBPassword,ParameterValue=${DB_PASSWORD} \
            --capabilities CAPABILITY_NAMED_IAM \
            --region ${REGION}
        print_info "Waiting for stack creation to complete..."
        aws cloudformation wait stack-create-complete \
            --stack-name ${STACK_NAME} \
            --region ${REGION}
    fi
    print_info "Stack deployment completed successfully"
}
get_outputs() {
    print_info "Retrieving stack outputs..."
    ALB_DNS=$(aws cloudformation describe-stacks \
        --stack-name ${STACK_NAME} \
        --query 'Stacks[0].Outputs[?OutputKey==`LoadBalancerDNS`].OutputValue' \
        --output text \
        --region ${REGION})
    CLOUDFRONT_URL=$(aws cloudformation describe-stacks \
        --stack-name ${STACK_NAME} \
        --query 'Stacks[0].Outputs[?OutputKey==`CloudFrontURL`].OutputValue' \
        --output text \
        --region ${REGION})
    DB_ENDPOINT=$(aws cloudformation describe-stacks \
        --stack-name ${STACK_NAME} \
        --query 'Stacks[0].Outputs[?OutputKey==`DatabaseEndpoint`].OutputValue' \
        --output text \
        --region ${REGION})
    BUCKET_NAME=$(aws cloudformation describe-stacks \
        --stack-name ${STACK_NAME} \
        --query 'Stacks[0].Outputs[?OutputKey==`FrontendBucketName`].OutputValue' \
        --output text \
        --region ${REGION})
    echo ""
    print_info "=== Deployment Information ==="
    echo "Load Balancer DNS: ${ALB_DNS}"
    echo "CloudFront URL: https://${CLOUDFRONT_URL}"
    echo "Database Endpoint: ${DB_ENDPOINT}"
    echo "Frontend Bucket: ${BUCKET_NAME}"
    echo ""
}
deploy_frontend() {
    print_info "Building frontend application..."
    cd frontend
    npm install
    npm run build
    print_info "Deploying frontend to S3..."
    aws s3 sync dist/ s3://${BUCKET_NAME}/ \
        --delete \
        --cache-control "public, max-age=31536000" \
        --region ${REGION}
    print_info "Invalidating CloudFront cache..."
    DISTRIBUTION_ID=$(aws cloudfront list-distributions \
        --query "DistributionList.Items[?Origins.Items[0].DomainName=='${BUCKET_NAME}.s3.amazonaws.com'].Id" \
        --output text)
    aws cloudfront create-invalidation \
        --distribution-id ${DISTRIBUTION_ID} \
        --paths "/*" \
```

```
        --region ${REGION}
    cd ..
    print_info "Frontend deployment completed"
}
deploy_backend() {
    print_info "Building backend Docker image..."
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    ECR_REPO="${ACCOUNT_ID}.dkr.ecr.${REGION}.amazonaws.com/${STACK_NAME}-backend"
    aws ecr get-login-password --region ${REGION} | \
        docker login --username AWS --password-stdin ${ECR_REPO}
    cd backend
    docker build -t ${STACK_NAME}-backend:latest .
    docker tag ${STACK_NAME}-backend:latest ${ECR_REPO}:latest
    docker push ${ECR_REPO}:latest
    cd ..
    print_info "Backend deployment completed"
}
run_migrations() {
    print_info "Running database migrations..."
    print_warning "Database migrations should be run manually or via CI/CD pipeline"
}
health_check() {
    print_info "Running health checks..."
    MAX_ATTEMPTS=30
    ATTEMPT=0
    while [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
        HTTP_CODE=$(curl -s -o /dev/null -w "%{http_code}" http://${ALB_DNS}/health)
        if [ "$HTTP_CODE" == "200" ]; then
            print_info "Health check passed!"
            return 0
        fi
        ATTEMPT=$((ATTEMPT + 1))
        print_warning "Health check attempt ${ATTEMPT}/${MAX_ATTEMPTS} failed. Retrying..."
        sleep 10
    done
    print_error "Health check failed after ${MAX_ATTEMPTS} attempts"
    return 1
}
rollback() {
    print_warning "Initiating rollback..."
    aws cloudformation cancel-update-stack \
        --stack-name ${STACK_NAME} \
        --region ${REGION}
    print_info "Rollback initiated"
}
cleanup() {
    print_info "Cleaning up resources..."
    aws s3 rm s.      ----make a pdf out of it with aim , theory , code and output and remove all the comments
```

## Sample Output:

```
[INFO] AWS CLI found
[INFO] Validating CloudFormation template...
[INFO] Template validation successful
[INFO] Deploying CloudFormation stack: fullstack-app
[INFO] Creating new stack...
[INFO] Waiting for stack creation to complete...
[INFO] Stack deployment completed successfully
[INFO] Retrieving stack outputs...
Load Balancer DNS: alb-example-123456.us-east-1.elb.amazonaws.com
CloudFront URL: https://d111111abcdef8.cloudfront.net
Database Endpoint: mydb.cluster-abcdef.us-east-1.rds.amazonaws.com
Frontend Bucket: fullstack-app-frontend-bucket
[INFO] Building frontend application...
... (npm install and build logs) ...
[INFO] Deploying frontend to S3...
[INFO] Invalidating CloudFront cache...
[INFO] Frontend deployment completed
[INFO] Building backend Docker image...
... (docker build and push logs) ...
[INFO] Backend deployment completed
[INFO] Running health checks...
[INFO] Health check passed!
```