EcoShower - Automated Installation Script

To reinstall or replicate this project on a new machine, follow these steps:

1. **Prerequisites**:
   - `aws-cli` configured with Administrator credentials.
   - `node` (v18+) and `npm` installed.
   - `python3.11` and `zip` installed.
2. **Save & Run**:
   - Save the code below as `setup.sh` in the project root.
   - Run `chmod +x setup.sh`.
   - Run `./setup.sh`.

```
#!/bin/bash
set -e

# ==========================================
# EcoShower - One-Click Installer
# ==========================================

echo "============================================="
echo "   EcoShower Project - Installation Started  "
echo "============================================="

# 1. Configuration
export AWS_REGION=eu-north-1
export ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output
text)
export BUCKET_NAME="ecoshower-frontend-$ACCOUNT_ID-$(date +%s)"

echo "Region: $AWS_REGION"
echo "Account ID: $ACCOUNT_ID"

# 2. DynamoDB Tables
echo "[1/8] Creating DynamoDB Tables..."

create_table() {
    aws dynamodb create-table --table-name $1 --attribute-definitions $3 -
-key-schema $2 --billing-mode PAY_PER_REQUEST --region $AWS_REGION
>/dev/null 2>&1 || echo "Table $1 exists."
}

create_table "EcoShower-Users" "AttributeName=user_id,KeyType=HASH"
"AttributeName=user_id,AttributeType=S"
create_table "EcoShower-Devices" "AttributeName=device_id,KeyType=HASH"
"AttributeName=device_id,AttributeType=S"
create_table "EcoShower-Sessions" "AttributeName=session_id,KeyType=HASH"
"AttributeName=session_id,AttributeType=S"
aws dynamodb create-table --table-name EcoShower-Telemetry --attribute-
```

```
definitions AttributeName=device_id,AttributeType=S
AttributeName=timestamp,AttributeType=S --key-schema
AttributeName=device_id,KeyType=HASH AttributeName=timestamp,KeyType=RANGE
--billing-mode PAY_PER_REQUEST --region $AWS_REGION >/dev/null 2>&1 ||
echo "Table EcoShower-Telemetry exists."

# 3. Cognito
echo "[2/8] Setting up Cognito..."
USER_POOL_ID=$(aws cognito-idp create-user-pool --pool-name EcoShower-
Users --auto-verified-attributes email --region $AWS_REGION --query
'UserPool.Id' --output text)
CLIENT_ID=$(aws cognito-idp create-user-pool-client --user-pool-id
$USER_POOL_ID --client-name EcoShower-WebApp --explicit-auth-flows
ALLOW_USER_PASSWORD_AUTH ALLOW_REFRESH_TOKEN_AUTH ALLOW_USER_SRP_AUTH --
region $AWS_REGION --query 'UserPoolClient.ClientId' --output text)

echo "UserPool: $USER_POOL_ID, Client: $CLIENT_ID"

# Create Admin Group & User
aws cognito-idp create-group --user-pool-id $USER_POOL_ID --group-name
admins --region $AWS_REGION >/dev/null 2>&1 || true
aws cognito-idp admin-create-user --user-pool-id $USER_POOL_ID --username
admin@ecoshower.com --user-attributes Name=email,Value=admin@ecoshower.com
Name=name,Value="System Admin" Name=custom:role,Value=admin --temporary-
password "TempPass123!" --region $AWS_REGION >/dev/null 2>&1 || echo
"Admin user exists."
aws cognito-idp admin-add-user-to-group --user-pool-id $USER_POOL_ID --
username admin@ecoshower.com --group-name admins --region $AWS_REGION
>/dev/null 2>&1 || true

# 4. IAM Role
echo "[3/8] Creating IAM Roles..."
cat > role_policy.json << EOF
{
  "Version": "2012-10-17",
  "Statement": [{"Effect": "Allow", "Principal": {"Service":
"lambda.amazonaws.com"}, "Action": "sts:AssumeRole"}]
}
EOF
aws iam create-role --role-name EcoShower-LambdaRole --assume-role-policy-
document file://role_policy.json >/dev/null 2>&1 || true
rm role_policy.json
sleep 5 # Wait for role propagation
aws iam attach-role-policy --role-name EcoShower-LambdaRole --policy-arn
arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
aws iam attach-role-policy --role-name EcoShower-LambdaRole --policy-arn
arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess
aws iam attach-role-policy --role-name EcoShower-LambdaRole --policy-arn
arn:aws:iam::aws:policy/AWSIoTDataAccess
ROLE_ARN=$(aws iam get-role --role-name EcoShower-LambdaRole --query
'Role.Arn' --output text)

# 5. Lambda
echo "[4/8] Deploying Lambdas..."
```

```
cd src/lambda
zip -q api.zip lambda_function.py
zip -q telemetry.zip process_telemetry.py

# API Lambda
aws lambda create-function --function-name EcoShower-API --runtime
python3.11 --role $ROLE_ARN --handler lambda_function.lambda_handler --
zip-file fileb://api.zip --timeout 30 --environment "Variables=
{USER_POOL_ID=$USER_POOL_ID,DEVICES_TABLE=EcoShower-
Devices,SESSIONS_TABLE=EcoShower-Sessions,USERS_TABLE=EcoShower-
Users,TELEMETRY_TABLE=EcoShower-Telemetry}" --region $AWS_REGION
>/dev/null 2>&1 || aws lambda update-function-code --function-name
EcoShower-API --zip-file fileb://api.zip --region $AWS_REGION >/dev/null

# Telemetry Lambda
aws lambda create-function --function-name EcoShower-ProcessTelemetry --
runtime python3.11 --role $ROLE_ARN --handler
process_telemetry.lambda_handler --zip-file fileb://telemetry.zip --
timeout 30 --environment "Variables={DEVICES_TABLE=EcoShower-
Devices,SESSIONS_TABLE=EcoShower-Sessions,USERS_TABLE=EcoShower-
Users,TELEMETRY_TABLE=EcoShower-Telemetry}" --region $AWS_REGION
>/dev/null 2>&1 || aws lambda update-function-code --function-name
EcoShower-ProcessTelemetry --zip-file fileb://telemetry.zip --region
$AWS_REGION >/dev/null

TELEMETRY_ARN=$(aws lambda get-function --function-name EcoShower-
ProcessTelemetry --query 'Configuration.FunctionArn' --output text --
region $AWS_REGION)
rm api.zip telemetry.zip
cd ../..

# 6. IoT
echo "[5/8] Configuring IoT..."
aws lambda add-permission --function-name EcoShower-ProcessTelemetry --
statement-id iot-invoke-$(date +%s) --action lambda:InvokeFunction --
principal iot.amazonaws.com --region $AWS_REGION >/dev/null 2>&1 || true
aws iot create-topic-rule --rule-name EcoShower_ProcessTelemetry --topic-
rule-payload "{\"sql\": \"SELECT * FROM 'ecoshower/+/telemetry'\",
\"actions\": [{\"lambda\": {\"functionArn\": \"$TELEMETRY_ARN\"}}],
\"ruleDisabled\": false, \"awsIotSqlVersion\": \"2016-03-23\"}" --region
$AWS_REGION >/dev/null 2>&1 || true

# 7. API Gateway
echo "[6/8] Configuring API Gateway..."
API_ID=$(aws apigateway create-rest-api --name EcoShower-API --region
$AWS_REGION --query 'id' --output text)
ROOT_ID=$(aws apigateway get-resources --rest-api-id $API_ID --query
'items[?path==`/`].id' --output text --region $AWS_REGION)
AUTH_ID=$(aws apigateway create-authorizer --rest-api-id $API_ID --name
CognitoAuth --type COGNITO_USER_POOLS --provider-arns "arn:aws:cognito-
idp:$AWS_REGION:$ACCOUNT_ID:userpool/$USER_POOL_ID" --identity-source
'method.request.header.Authorization' --query 'id' --output text --region
$AWS_REGION)
```

```
# Setup Proxy
aws apigateway put-method --rest-api-id $API_ID --resource-id $ROOT_ID --
http-method ANY --authorization-type COGNITO_USER_POOLS --authorizer-id
$AUTH_ID --region $AWS_REGION >/dev/null
aws apigateway put-integration --rest-api-id $API_ID --resource-id
$ROOT_ID --http-method ANY --type AWS_PROXY --integration-http-method POST
--uri "arn:aws:apigateway:$AWS_REGION:lambda:path/2015-03-
31/functions/arn:aws:lambda:$AWS_REGION:$ACCOUNT_ID:function:EcoShower-
API/invocations" --region $AWS_REGION >/dev/null

PROXY_ID=$(aws apigateway create-resource --rest-api-id $API_ID --parent-
id $ROOT_ID --path-part "{proxy+}" --query 'id' --output text --region
$AWS_REGION)
aws apigateway put-method --rest-api-id $API_ID --resource-id $PROXY_ID --
http-method ANY --authorization-type COGNITO_USER_POOLS --authorizer-id
$AUTH_ID --region $AWS_REGION >/dev/null
aws apigateway put-integration --rest-api-id $API_ID --resource-id
$PROXY_ID --http-method ANY --type AWS_PROXY --integration-http-method
POST --uri "arn:aws:apigateway:$AWS_REGION:lambda:path/2015-03-
31/functions/arn:aws:lambda:$AWS_REGION:$ACCOUNT_ID:function:EcoShower-
API/invocations" --region $AWS_REGION >/dev/null

aws lambda add-permission --function-name EcoShower-API --statement-id
apigw-$(date +%s) --action lambda:InvokeFunction --principal
apigateway.amazonaws.com --source-arn "arn:aws:execute-
api:$AWS_REGION:$ACCOUNT_ID:$API_ID/*/*/*" --region $AWS_REGION >/dev/null
2>&1 || true
aws apigateway create-deployment --rest-api-id $API_ID --stage-name prod -
-region $AWS_REGION >/dev/null
API_URL="https://$API_ID.execute-api.$AWS_REGION.amazonaws.com/prod"

# 8. Frontend
echo "[7/8] Deploying Frontend..."
aws s3 mb s3://$BUCKET_NAME --region $AWS_REGION >/dev/null

# Inject Config
cat > src/frontend/src/config.js << EOF
export const config = {
  API_URL: '$API_URL',
  COGNITO_USER_POOL_ID: '$USER_POOL_ID',
  COGNITO_CLIENT_ID: '$CLIENT_ID',
  AWS_REGION: '$AWS_REGION',
  WATER_COST_PER_LITER: 0.008,
  DEFAULT_TARGET_TEMP: 38,
  MIN_TEMP: 30,
  MAX_TEMP: 45,
  TELEMETRY_REFRESH_INTERVAL: 2000,
  DASHBOARD_REFRESH_INTERVAL: 30000,
};
export default config;
EOF

cd src/frontend
npm install && npm run build
```

```
aws s3 sync dist/ s3://$BUCKET_NAME --region $AWS_REGION

# CloudFront
cat > cf.json << EOF
{
    "CallerReference": "ecoshower-$(date +%s)",
    "Origins": {
        "Quantity": 1,
        "Items": [{"Id": "S3", "DomainName":
"$BUCKET_NAME.s3.amazonaws.com", "S3OriginConfig":
{"OriginAccessIdentity": ""}}]
    },
    "DefaultCacheBehavior": {
        "TargetOriginId": "S3",
        "ViewerProtocolPolicy": "redirect-to-https",
        "AllowedMethods": {"Quantity": 2, "Items": ["GET", "HEAD"]},
        "ForwardedValues": {"QueryString": false, "Cookies": {"Forward":
"none"}},
        "MinTTL": 0
    },
    "Enabled": true,
    "Comment": "EcoShower"
}
EOF
CF_DOMAIN=$(aws cloudfront create-distribution --distribution-config
file://cf.json --query 'Distribution.DomainName' --output text)
rm cf.json
cd ../..

echo "=============================================="
echo "   INSTALLATION COMPLETE!                     "
echo "=============================================="
echo "Frontend: https://$CF_DOMAIN"
echo "API: $API_URL"
echo "Admin: admin@ecoshower.com / TempPass123!"
echo "=============================================="
```