MakerPortal & Database Setup Guide

This guide explains how to quickly setup the MakerPortal and its associated infrastructure with no coding required. Estimated Time to Complete: 25m

Important Note: If you wish to create your own server configuration, simply clone this GitHub repository and proceed as you wish. The README file contains additional information about building the application and has recommendations for server configurations. For those with little experience configuring servers we recommend using our AWS CloudFormation to deploy with no coding required.

# Prerequisites

* [AWS Account](https://aws.amazon.com/free/?trk=7541ebd3-552d-4f98-9357-b542436aa66c&sc_channel=ps&ef_id=Cj0KCQjw2ZfABhDBARIsAHFTxGxivSgDphUWF8vx5aGTd5fjy6obd-xFdH9L3jV3CzatDrxzD8B4DykaAqUoEALw_wcB:G:s&s_kwcid=AL!4422!3!651751058790!e!!g!!aws%20account!19852662149!145019243897&gbraid=0AAAAADjHtp9GJl1OUwl6kWoOQA2WeCbht&gclid=Cj0KCQjw2ZfABhDBARIsAHFTxGxivSgDphUWF8vx5aGTd5fjy6obd-xFdH9L3jV3CzatDrxzD8B4DykaAqUoEALw_wcB&all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=*all&awsf.Free%20Tier%20Categories=*all): Used to login to AWS Console and use CloudFormation.
* [Google OAuth Client ID](https://developers.google.com/identity/protocols/oauth2): Used for authentication with the MakerPortal.

# AWS CloudFormation MakerPortal

## CloudFormation Architecture

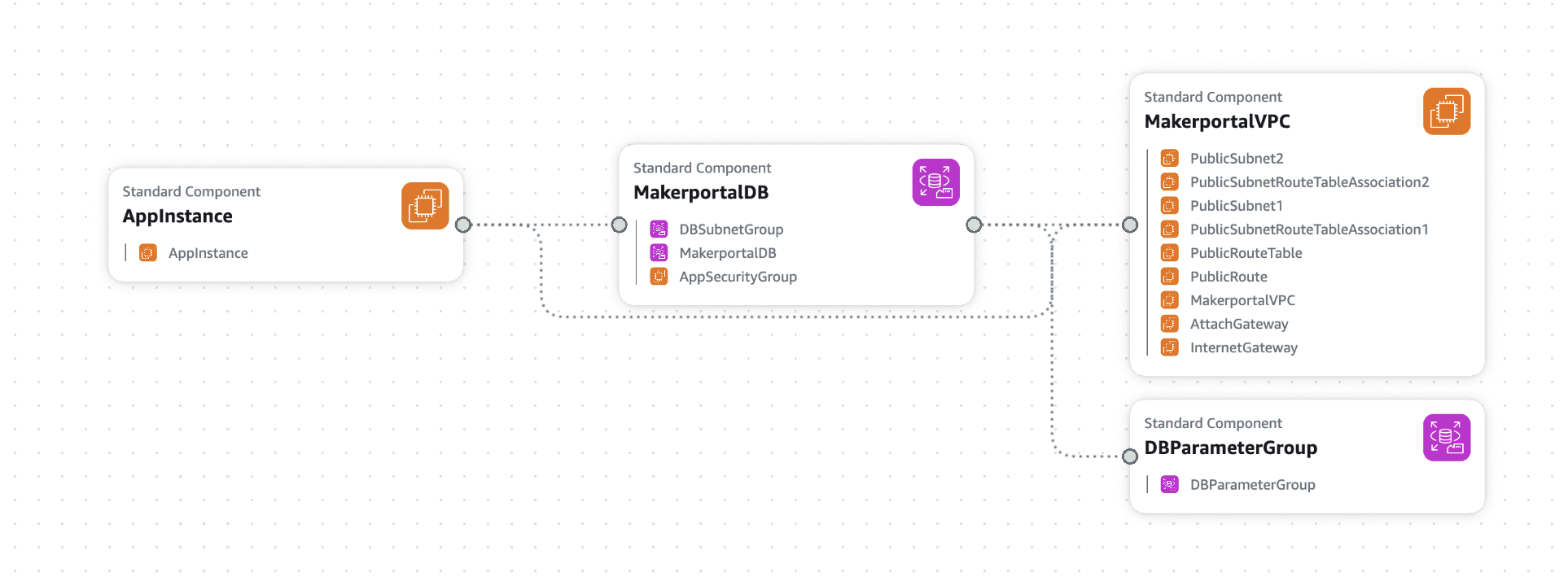


Figure 1: CloudFormation Stack Diagram

The MakerPortal system consists of a MySQL database which feeds data to the Portal Box and MakerPortal through an API. The MakerPortal and API are hosted in an AWS EC2 instance, while the database is hosted on AWS RDS. We’ve created a CloudFormation template which allows you to set up your own MakerPortal with no coding needed. The database lives in a subnet group which talks to the virtual private cloud hosting the web application. The following section will walk you through the process of building the system.

## Prerequisites for CloudFormation

* [AWS Account](https://aws.amazon.com/free/?trk=7541ebd3-552d-4f98-9357-b542436aa66c&sc_channel=ps&ef_id=Cj0KCQjw2ZfABhDBARIsAHFTxGxivSgDphUWF8vx5aGTd5fjy6obd-xFdH9L3jV3CzatDrxzD8B4DykaAqUoEALw_wcB:G:s&s_kwcid=AL!4422!3!651751058790!e!!g!!aws%20account!19852662149!145019243897&gbraid=0AAAAADjHtp9GJl1OUwl6kWoOQA2WeCbht&gclid=Cj0KCQjw2ZfABhDBARIsAHFTxGxivSgDphUWF8vx5aGTd5fjy6obd-xFdH9L3jV3CzatDrxzD8B4DykaAqUoEALw_wcB&all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=*all&awsf.Free%20Tier%20Categories=*all): Used to login to AWS Console and use CloudFormation.
  + Task Duration: ~5 minutes
  + Notes: Requires credit card even for free tier usage. MakerPortal instances are unlikely to incur charges over $3-5 per month, and depending on usage will remain in the free tier.
* [Google OAuth Client ID](https://developers.google.com/identity/protocols/oauth2): Obtain an OAuth Client ID to enable the authentication for the web application.
  + Task Duration: ~3 minutes
  + Follow these steps to create the Client ID needed for the application:

1. Create or login to your Google account.
2. Visit the [Google Console](https://console.developers.google.com).
3. APIs & Services → Credentials → Create Credentials
4. Choose “Web Application” for the type.
5. Click “Create” and copy down your Client ID.
6. Enter a name for your credential set. This is all for now, but we will revisit this at the end to finish setting up the configuration.

## CloudFormation Deployment

1. Login to the AWS Console.
2. Make sure your region is set to us-east-2.
3. Search for “CloudFormation” and visit the service.
4. Click “Create Stack”
5. Download the makerportal-template.yaml file (full code in Appendix if unavailable - copy into a file and save).
6. “Choose an existing template” → “Upload a template file” → choose the file you just downloaded.
7. Fill in the parameters:
   1. **Stack name:** choose something descriptive for your deployment, will not impact anything externally.
   2. **AMIId:** automatically set to a public Amazon Machine Image for the server and app configuration.
   3. **AdminEmail:** your email address.
   4. **AdminName:** your name or nickname.
   5. **DBName:** what you want your database to be labeled as. Defaults to “makerportal”.
   6. **DBPassword:** password for the database, write it down somewhere in case you want to make manual modifications down the line.
   7. **DBUser:** name of administrator for the database. Defaults to “admin”.
   8. **KeyName:** name of keypair used to SSH into the server. You may need to create a keypair by visiting EC2 → Network & Security → Key Pairs → Create key pair
      1. Enter whatever name you like, choose RSA and .pem
   9. **OAuthClientID:** the Client ID from the credentials you made in the Google Console.

## Adding URIs to Google Console (< 2 minutes)

1. Return to Google Console → APIs & Services → Credentials → Your Client ID
2. Add your public EC2 endpoint as an Authorized JavaScript Origin. It should look like the following example:
   1. <http://ec2-instance-ip.region.compute.amazonaws.com>
3. Add three variants of your public EC2 endpoint as Authorized Redirect URIs. They should look like the following examples:
   1. <http://ec2-instance-ip.region.compute.amazonaws.com>
   2. <http://ec2-instance-ip.region.compute.amazonaws.com/>
   3. <http://ec2-instance-ip.region.compute.amazonaws.com/api/login.php>

# Appendix

## makerportal-template.yaml Full Code

AWSTemplateFormatVersion: '2010-09-09'

Description: "MakerPortal Turnkey Deployment"

Parameters:

DBName:

Type: String

Default: "makerportal"

DBUser:

Type: String

Default: "admin"

AllowedPattern: "[a-zA-Z][a-zA-Z0-9]\*"

DBPassword:

Type: String

NoEcho: true

OAuthClientID:

Type: String

AdminName:

Type: String

Description: "Admin user's full name"

AdminEmail:

Type: String

Description: "Admin user's email"

AMIId:

Type: AWS::EC2::Image::Id

Default: ami-03267e5440a28b3ce

KeyName:

Type: AWS::EC2::KeyPair::KeyName

Resources:

# VPC and Networking Resources

MakerportalVPC:

Type: AWS::EC2::VPC

Properties:

CidrBlock: 10.0.0.0/16

EnableDnsSupport: true

EnableDnsHostnames: true

Tags:

- Key: Name

Value: MakerportalVPC

InternetGateway:

Type: AWS::EC2::InternetGateway

Properties:

Tags:

- Key: Name

Value: MakerportalIGW

AttachGateway:

Type: AWS::EC2::VPCGatewayAttachment

Properties:

VpcId: !Ref MakerportalVPC

InternetGatewayId: !Ref InternetGateway

# Create subnets in two AZs for RDS requirements

PublicSubnet1:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref MakerportalVPC

CidrBlock: 10.0.1.0/24

AvailabilityZone: !Select [0, !GetAZs '']

MapPublicIpOnLaunch: true

Tags:

- Key: Name

Value: MakerportalPublicSubnet1

PublicSubnet2:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref MakerportalVPC

CidrBlock: 10.0.2.0/24

AvailabilityZone: !Select [1, !GetAZs '']

MapPublicIpOnLaunch: true

Tags:

- Key: Name

Value: MakerportalPublicSubnet2

PublicRouteTable:

Type: AWS::EC2::RouteTable

Properties:

VpcId: !Ref MakerportalVPC

Tags:

- Key: Name

Value: MakerportalPublicRouteTable

PublicRoute:

Type: AWS::EC2::Route

DependsOn: AttachGateway

Properties:

RouteTableId: !Ref PublicRouteTable

DestinationCidrBlock: 0.0.0.0/0

GatewayId: !Ref InternetGateway

PublicSubnetRouteTableAssociation1:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

SubnetId: !Ref PublicSubnet1

RouteTableId: !Ref PublicRouteTable

PublicSubnetRouteTableAssociation2:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

SubnetId: !Ref PublicSubnet2

RouteTableId: !Ref PublicRouteTable

# Security Groups

AppSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: "Allow SSH, HTTP, HTTPS and MySQL"

VpcId: !Ref MakerportalVPC

SecurityGroupIngress:

- IpProtocol: tcp

FromPort: 22

ToPort: 22

CidrIp: 0.0.0.0/0

- IpProtocol: tcp

FromPort: 80

ToPort: 80

CidrIp: 0.0.0.0/0

- IpProtocol: tcp

FromPort: 443

ToPort: 443

CidrIp: 0.0.0.0/0

- IpProtocol: tcp

FromPort: 3306

ToPort: 3306

CidrIp: 0.0.0.0/0

# Database Resources

DBParameterGroup:

Type: AWS::RDS::DBParameterGroup

Properties:

Description: "Parameter group for MakerPortal database"

Family: mysql8.0

Parameters:

log\_bin\_trust\_function\_creators: 1

max\_allowed\_packet: 16777216

character\_set\_server: utf8mb4

collation\_server: utf8mb4\_unicode\_ci

max\_connections: 100

wait\_timeout: 28800

interactive\_timeout: 28800

# Create DB Subnet Group with both subnets

DBSubnetGroup:

Type: AWS::RDS::DBSubnetGroup

Properties:

DBSubnetGroupDescription: "Subnet group for MakerPortal database"

SubnetIds:

- !Ref PublicSubnet1

- !Ref PublicSubnet2

MakerportalDB:

Type: AWS::RDS::DBInstance

Properties:

DBName: !Ref DBName

Engine: mysql

EngineVersion: 8.0

MasterUsername: !Ref DBUser

MasterUserPassword: !Ref DBPassword

AllocatedStorage: 20

DBInstanceClass: db.t3.micro

PubliclyAccessible: true

DBParameterGroupName: !Ref DBParameterGroup

DBSubnetGroupName: !Ref DBSubnetGroup

VPCSecurityGroups:

- !GetAtt AppSecurityGroup.GroupId

# EC2 Instance

AppInstance:

Type: AWS::EC2::Instance

DependsOn: MakerportalDB

Properties:

InstanceType: t3.micro

ImageId: !Ref AMIId

KeyName: !Ref KeyName

SubnetId: !Ref PublicSubnet1

SecurityGroupIds:

- !Ref AppSecurityGroup

UserData:

Fn::Base64:

!Join ["", [

"#!/bin/bash -xe\n",

"\n",

"# Set up logging\n",

"exec > >(tee /var/log/user-data.log|logger -t user-data -s 2>/dev/console) 2>&1\n",

"\n",

"echo \"Starting MakerPortal setup script\"\n",

"\n",

"CONFIG\_PATH=\"/var/www/html/portalbox/config/config.ini\"\n",

"DB\_HOST=\"", !GetAtt MakerportalDB.Endpoint.Address, "\"\n",

"DB\_NAME=\"", !Ref DBName, "\"\n",

"DB\_USER=\"", !Ref DBUser, "\"\n",

"DB\_PASS=\"", !Ref DBPassword, "\"\n",

"\n",

"echo \"Setting up config.ini\"\n",

"mkdir -p /var/www/html/portalbox/config\n",

"cat > $CONFIG\_PATH <<EOF\n",

"[database]\n",

"driver=mysql\n",

"host=$DB\_HOST\n",

"database=$DB\_NAME\n",

"username=$DB\_USER\n",

"password=$DB\_PASS\n",

"\n",

"[oauth]\n",

"google\_oauth\_client\_id=", !Ref OAuthClientID, "\n",

"EOF\n",

"\n",

"echo \"Config created at $CONFIG\_PATH\"\n",

"\n",

"# Wait for DB to be ready with better error handling\n",

"echo \"Waiting for DB to be ready at $DB\_HOST...\"\n",

"MAX\_RETRIES=60\n",

"COUNT=0\n",

"DB\_READY=false\n",

"\n",

"while [ $COUNT -lt $MAX\_RETRIES ]; do\n",

" echo \"Attempt $COUNT to connect to database...\"\n",

" \n",

" if mysql -h $DB\_HOST -u $DB\_USER -p$DB\_PASS -e \"SELECT 1;\" $DB\_NAME; then\n",

" echo \"Successfully connected to database!\"\n",

" DB\_READY=true\n",

" break\n",

" else\n",

" echo \"Database not ready yet, waiting...\"\n",

" sleep 10\n",

" COUNT=$((COUNT+1))\n",

" fi\n",

"done\n",

"\n",

"if [ \"$DB\_READY\" = false ]; then\n",

" echo \"ERROR: Could not connect to database after $MAX\_RETRIES attempts. Exiting.\"\n",

" exit 1\n",

"fi\n",

"\n",

"# Find schema file (search multiple locations)\n",

"SCHEMA\_PATH=\"\"\n",

"POSSIBLE\_LOCATIONS=(\n",

" \"/var/www/html/portalbox/config/schema.sql\"\n",

" \"/var/www/html/portalbox/schema.sql\"\n",

" \"/var/www/portalbox/config/schema.sql\"\n",

" \"/var/www/portalbox/schema.sql\"\n",

" \"/var/www/html/config/schema.sql\"\n",

" \"/var/www/schema.sql\"\n",

")\n",

"\n",

"for location in \"${POSSIBLE\_LOCATIONS[@]}\"; do\n",

" if [ -f \"$location\" ]; then\n",

" SCHEMA\_PATH=\"$location\"\n",

" echo \"Found schema file at $SCHEMA\_PATH\"\n",

" break\n",

" fi\n",

"done\n",

"\n",

"if [ -z \"$SCHEMA\_PATH\" ]; then\n",

" echo \"Searching for schema.sql in the entire /var/www directory...\"\n",

" FOUND\_SCHEMA=$(find /var/www -name \"schema.sql\" -type f | head -1)\n",

" \n",

" if [ -n \"$FOUND\_SCHEMA\" ]; then\n",

" SCHEMA\_PATH=\"$FOUND\_SCHEMA\"\n",

" echo \"Found schema file at $SCHEMA\_PATH\"\n",

" else\n",

" echo \"ERROR: No schema.sql file found in /var/www. Trying to download instead.\"\n",

" SCHEMA\_PATH=\"/tmp/schema.sql\"\n",

" wget -O $SCHEMA\_PATH https://raw.githubusercontent.com/Bucknell-ECE/PortalBox-database/master/schema/schema.sql\n",

" fi\n",

"fi\n",

"\n",

"if [ ! -f \"$SCHEMA\_PATH\" ]; then\n",

" echo \"ERROR: Could not find or download schema file. Exiting.\"\n",

" exit 1\n",

"fi\n",

"\n",

"# Apply schema with extensive error checking\n",

"echo \"Applying schema to database from $SCHEMA\_PATH...\"\n",

"\n",

"# Modify the schema to include SQL\_SECURITY INVOKER for functions\n",

"echo \"Adding SQL\_SECURITY INVOKER to functions in schema...\"\n",

"sed -i 's/CREATE FUNCTION get\_user\_balance\_for\_card/CREATE FUNCTION get\_user\_balance\_for\_card/g' $SCHEMA\_PATH\n",

"sed -i 's/READS SQL DATA/READS SQL DATA\\n SQL SECURITY INVOKER/g' $SCHEMA\_PATH\n",

"\n",

"# Apply schema with verbose output\n",

"mysql -h $DB\_HOST -u $DB\_USER -p$DB\_PASS $DB\_NAME < $SCHEMA\_PATH\n",

"SCHEMA\_RESULT=$?\n",

"\n",

"if [ $SCHEMA\_RESULT -ne 0 ]; then\n",

" echo \"ERROR: Failed to apply schema to database (exit code: $SCHEMA\_RESULT)\"\n",

" echo \"Trying to apply schema in smaller chunks...\"\n",

" \n",

" # Split the schema file and apply it in chunks\n",

" csplit -f \"/tmp/schema\_part\_\" $SCHEMA\_PATH \"/DROP PROCEDURE/\" \"/DROP FUNCTION/\" \"/CREATE TABLE/\" \"/INSERT INTO/\" \"/DELIMITER/\" \"/CREATE PROCEDURE/\" \"/CREATE FUNCTION/\" \"{\*}\"\n",

" \n",

" for part in /tmp/schema\_part\_\*; do\n",

" echo \"Applying schema part: $part\"\n",

" mysql -h $DB\_HOST -u $DB\_USER -p$DB\_PASS $DB\_NAME < $part\n",

" if [ $? -ne 0 ]; then\n",

" echo \"WARNING: Error applying schema part $part\"\n",

" fi\n",

" done\n",

"fi\n",

"\n",

"# Verify schema was applied by checking for users table\n",

"echo \"Verifying schema was applied...\"\n",

"if ! mysql -h $DB\_HOST -u $DB\_USER -p$DB\_PASS -e \"SHOW TABLES;\" $DB\_NAME | grep -q \"users\"; then\n",

" echo \"ERROR: Schema verification failed - users table does not exist\"\n",

" \n",

" # Last resort - apply minimal schema to create basic tables\n",

" echo \"Attempting to create minimal schema...\"\n",

" mysql -h $DB\_HOST -u $DB\_USER -p$DB\_PASS $DB\_NAME <<EOSQL\n",

"CREATE TABLE IF NOT EXISTS roles (\n",

" id INT UNSIGNED AUTO\_INCREMENT NOT NULL,\n",

" name TEXT NOT NULL,\n",

" is\_system\_role INT(1) UNSIGNED NOT NULL DEFAULT 0,\n",

" description TEXT,\n",

" PRIMARY KEY (id)\n",

");\n",

"\n",

"INSERT IGNORE INTO roles(id, name, is\_system\_role, description) VALUES\n",

" (1, 'unauthenticated', 1, 'Role of users who have not authenticated'),\n",

" (2, 'user', 1, 'Role of authenticated users who have not be granted additional permissions.'),\n",

" (3, 'admin', 1, 'Role for users who administer the system.');\n",

"\n",

"CREATE TABLE IF NOT EXISTS users (\n",

" id INT UNSIGNED AUTO\_INCREMENT NOT NULL,\n",

" name TEXT NOT NULL,\n",

" email VARCHAR(512) NOT NULL,\n",

" comment TEXT,\n",

" role\_id INT UNSIGNED NOT NULL,\n",

" is\_active INT(1) UNSIGNED NOT NULL,\n",

" PRIMARY KEY (id)\n",

");\n",

"\n",

"CREATE TABLE IF NOT EXISTS api\_keys (\n",

" id INT UNSIGNED AUTO\_INCREMENT NOT NULL,\n",

" name TEXT NOT NULL,\n",

" token CHAR(32) NOT NULL,\n",

" PRIMARY KEY(id)\n",

");\n",

"EOSQL\n",

"else\n",

" echo \"Schema verification successful - users table exists\"\n",

"fi\n",

"\n",

"# Insert admin user\n",

"echo \"Creating admin user: ", !Ref AdminEmail, "\"\n",

"mysql -h $DB\_HOST -u $DB\_USER -p$DB\_PASS $DB\_NAME <<EOSQL\n",

"INSERT INTO users (name, email, role\_id, is\_active)\n",

"VALUES ('", !Ref AdminName, "', '", !Ref AdminEmail, "', 3, 1);\n",

"EOSQL\n",

"\n",

"# Generate and insert API key\n",

"echo \"Generating API key...\"\n",

"API\_KEY=$(cat /proc/sys/kernel/random/uuid | tr -d '-')\n",

"mysql -h $DB\_HOST -u $DB\_USER -p$DB\_PASS $DB\_NAME <<EOSQL\n",

"INSERT INTO api\_keys (name, token)\n",

"VALUES ('Makerportal', '$API\_KEY');\n",

"EOSQL\n",

"\n",

"echo \"API key generated and stored: $API\_KEY\"\n",

"echo \"API key: $API\_KEY\" > /home/ec2-user/api\_key.txt\n",

"\n",

"# Update file permissions\n",

"echo \"Setting permissions on MakerPortal files...\"\n",

"find /var/www -type d -exec chmod 755 {} \\;\n",

"find /var/www -type f -exec chmod 644 {} \\;\n",

"chown -R apache:apache /var/www/html\n",

"\n",

"# Restart Apache\n",

"echo \"Restarting Apache...\"\n",

"systemctl restart httpd\n",

"\n",

"echo \"MakerPortal setup complete!\"\n"

]]

Outputs:

WebsiteURL:

Description: "URL for the MakerPortal website"

Value: !Sub "http://${AppInstance.PublicDnsName}"

DbEndpoint:

Description: "Endpoint for the RDS Database"

Value: !GetAtt MakerportalDB.Endpoint.Address

VPCID:

Description: "VPC ID"

Value: !Ref MakerportalVPC

SubnetID:

Description: "Public Subnet ID"

Value: !Ref PublicSubnet1