Setting up the AWS Account

1. Create an AWS Account.
2. Create an IAM User with programmatic access and take a note of the **Access Key ID** and **Secret Access Key** of the created IAM User.
3. Create an IAM User Group with the required privileges. Attach AdministratorAccess Policy to the group.
4. Add the IAM User to the newly created group.
5. Configure the AWS Credentials using the Serverless CLI using the following command:
   a. sls config credentials --provider aws --key **aws_access_key_id** --secret **aws_secret_access_key** --profile **profile_name**

Installing Serverless Framework

1. First, install Node.JS 8 or above. We are using NodeJS 10.x runtime for this application.
2. Install the Serverless Framework
   a. **npm install -g serverless**

Deploying the Serverless Application

1. Use this command to deploy the Application:
   a. sls deploy --verbose --profile profile_name
2. This will deploy the application services and output all the required service IDs.
3. Run this command to add **"builders"** and **"property"** data to the table. The application works based on these static data from the DB.
   a. aws dynamodb batch-write-item --request-items file://data.json --region deployed_region
   b. The **"org_id"** field for the Items of type **"property"**, should match one of the **"builder"** item's Id in the **data.json**. By default we have given the Id of the **"InApp Builders"** as **"org_id".**

c. Properties are fetched from the front end using the logged in user's home builder Id, that is selected at the time of Sign Up.
d. Replace the **"deployed_region"** with the exact deployed region.
e. If you want to change the builder or property details, you can change the **data.json** found in the serverless code directory.
f. If you have deployed serverless app in production stage, then change the table name **"dev-entities"** to **"prod_entities".**

Configuring React Application before Deployment

1. Open the "**ConfigAWS.js"** located in the "**src"** folder of the front end application directory.
2. Replace the API Gateway Region and Cognito Region with the deployed region.
3. Replace API Gateway URL, User Pool Id, App Client Id and Identity Pool Id values using the **"ServiceEndpoint"**, **"UserPoolId", "UserPoolClientId", "IdentityPoolId"** from the stack output variables respectively, in the **"ConfigAWS"** JSON.
    a. You can also use the below command to list the stack output on the console.
        i. **sls info -v**

Deploying the React Application

1. To deploy the front end application, we need the S3 Bucket Name that we created for the front end deployments and the CloudFront Distribution ID that we created in the previous step. Take note of both.
2. Go to the "**package.json**" file of the React Application.
3. Change the "**deploy**" script under the "**scripts**" object to:
    a. aws s3 sync build/ **bucket_name** --delete --profile **aws_profile_name**
    b. **bucket_name** can be obtained from **"FrontEndBucketName"** stack output.

4. Change the "**postdeploy**" script under the "**scripts**" object to:
    a. aws cloudfront create-invalidation --distribution-id **CloudFront_Distribution_Id** --paths '/*' --profile **aws_profile_name**
    b. **CloudFront_Distribution_Id** can be obtained from **"HyphenCRMDistributionId"** stack output.
5. From the terminal, run **npm run deploy.**
6. After deployment, use the **"HyphenCRMDistributionOutput",** which is the CloudFront URL for the application, to get started.
7. We have disabled the User Sign Up link for now as per Danny's requirement. But you can use the URL below to access the Sign Up page to register a user.
    a. <HyphenCRMDistributionOutput>/signup