**Document Title**: Junior Engineer Setup

**Project Title**: Hatching Sparrow CRM

**Date**: 6/12/2024

**Author(s)**: Ryan O’Connor, Full Stack Software Engineer

**Confidentiality:** N/A

**Comprehensive Guide for Junior Engineers: Development and Deployment**

**Table of Contents**

1. Joining the AWS Organization
2. Setting Up Your AWS Account
3. Installing Necessary Tools
   * Node.js and npm
   * AWS CLI
4. Development
   * Configuring Your Local Environment
   * Learning About AWS Services
   * Local Development Setup for AWS Lambda
5. Deployment
   * Assuming the IAM Role
   * Packaging the Lambda Function
   * Deploying the Lambda Function
   * Invoking the Lambda Function
6. Summary
7. Additional Resources

**1. Joining the AWS Organization**

**Steps to Join the AWS Organization**

1. **Receive an Invitation:**
   * You will receive an email invitation from the AWS organization admin to join the organization.
2. **Accept the Invitation:**
   * Follow the instructions in the email to accept the invitation. This will involve creating your own AWS root account if you don't have one already.
3. **Log in to Your AWS Account:**
   * After accepting the invitation, log in to your AWS root account to confirm that you are now part of the organization.

**2. Setting Up Your AWS Account**

**Steps to Set Up Your AWS Account**

1. **Create Your AWS Root Account:**
   * Visit the AWS sign-up page and create your root account.
2. **Accept the Organization Invitation:**
   * Log in to your newly created AWS root account.
   * Accept the invitation to join the AWS organization from the email you received.

**3. Installing Necessary Tools**

**Node.js and npm**

1. **Install Node.js and npm:**
   * **macOS (using Homebrew):**

brew install node

* + **Windows (using Chocolatey):**

choco install nodejs

**AWS CLI**

1. **Install AWS CLI:**
   * **macOS (using Homebrew):**

brew install awscli

* + **Windows (using Chocolatey):**

choco install awscli

**Git Bash**

1. **Install Git Bash (Windows):**
   * Download and install Git Bash from the [Git for Windows](https://gitforwindows.org/) website.

**4. Development**

**Configuring Your Local Environment**

1. **Configure AWS CLI:**
   * Open Git Bash or your preferred terminal.
   * Run the following command to configure the AWS CLI with your root account access keys:

bash

Copy code

aws configure

* + Enter your AWS Access Key ID, Secret Access Key, default region, and default output format when prompted.

**Learning About AWS Services**

1. **AWS Lambda**:
   * [AWS Lambda Developer Guide](https://docs.aws.amazon.com/lambda/latest/dg/welcome.html)
   * [What is AWS Lambda?](https://aws.amazon.com/lambda/)
2. **DynamoDB**:
   * [Amazon DynamoDB Documentation](https://docs.aws.amazon.com/dynamodb/index.html)
   * [What is Amazon DynamoDB?](https://aws.amazon.com/dynamodb/)
3. **API Gateway**:
   * [Amazon API Gateway Documentation](https://docs.aws.amazon.com/apigateway/index.html)
   * [What is Amazon API Gateway?](https://aws.amazon.com/api-gateway/)
4. **S3**:
   * [Amazon S3 Documentation](https://docs.aws.amazon.com/s3/index.html)
   * [What is Amazon S3?](https://aws.amazon.com/s3/)

**Local Development Setup for AWS Lambda**

1. **Initialize a New Node.js Project:**
   * Create a new directory for your project and navigate to it:

mkdir my-lambda-function

cd my-lambda-function

* + Initialize a new Node.js project:

npm init -y

1. **Install AWS SDK:**
   * Install the AWS SDK for JavaScript:

npm install aws-sdk

1. **Create Lambda Function Code:**
   * Create a new file named index.js and add the following sample code:

const AWS = require('aws-sdk');

exports.handler = async (event) => {

return {

statusCode: 200,

body: JSON.stringify('Hello from Lambda!'),

};

};

**5. Deployment**

**Assuming the IAM Role**

1. **Assume the IAM Role:**
   * Use the aws sts assume-role command to assume the role provided by the organization admin:

aws sts assume-role --role-arn arn:aws:iam::<management-account-id>:role/<role-name> --role-session-name <session-name> --duration-seconds 43200 > assume-role-output.json

* + Extract the credentials from the output:

export AWS\_ACCESS\_KEY\_ID=$(jq -r '.Credentials.AccessKeyId' assume-role-output.json)

export AWS\_SECRET\_ACCESS\_KEY=$(jq -r '.Credentials.SecretAccessKey' assume-role-output.json)

export AWS\_SESSION\_TOKEN=$(jq -r '.Credentials.SessionToken' assume-role-output.json)

**Packaging the Lambda Function**

1. **Create a Deployment Package:**
   * Zip your project files:

zip -r function.zip .

**Deploying the Lambda Function**

1. **Deploy the Lambda Function:**
   * Use the AWS CLI to create a new Lambda function:

aws lambda create-function --function-name myLambdaFunction --runtime nodejs14.x --role arn:aws:iam::<your-role-arn> --handler index.handler --zip-file fileb://function.zip

**Invoking the Lambda Function**

1. **Invoke the Lambda Function:**
   * Test your Lambda function by invoking it:

aws lambda invoke --function-name myLambdaFunction output.txt

1. **Check the Output:**
   * Open output.txt to see the result of your Lambda function invocation.

**6. Summary**

By following this guide, you will be able to join the AWS organization, assume roles for accessing AWS resources, and set up your local development environment for AWS Lambda using Node.js. This setup ensures secure access to AWS resources and provides a streamlined development workflow.

**7. Additional Resources**

* [AWS IAM User Guide](https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html)
* [AWS Security Token Service (STS) Documentation](https://docs.aws.amazon.com/STS/latest/APIReference/Welcome.html)
* [Best Practices for IAM](https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html)
* [AWS Lambda Developer Guide](https://docs.aws.amazon.com/lambda/latest/dg/welcome.html)