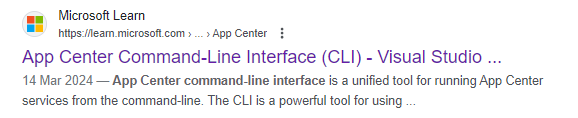
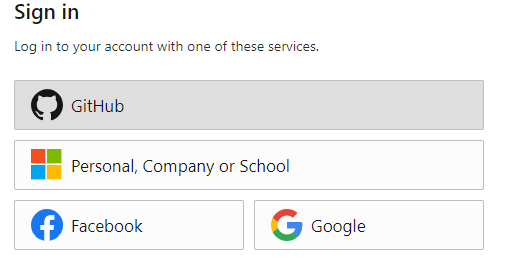
**React Native Code Push**

**STEP-1 [App Center Account Creation]**

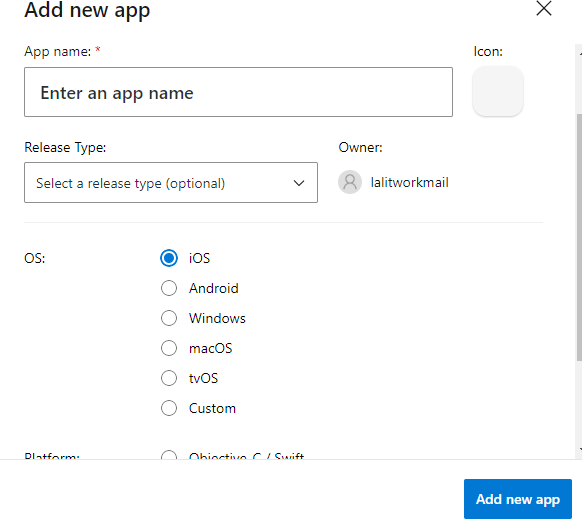
1. **Open “**[**https://appcenter.ms/**](https://appcenter.ms/) **” this link**



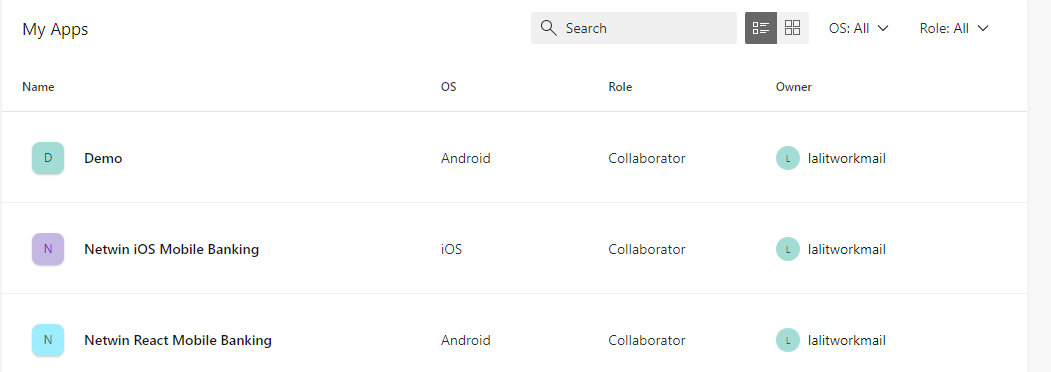
1. **Log on into it using any of the below option, we used Github for that**



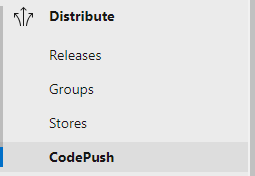
1. **After successful login we get Dashboard from there we have to select Add new Option**
2. **Fill Required Details**
3. **Create Separate App for Both Android & IOS**



1. **Now you can see your Created App listed in dashboard**



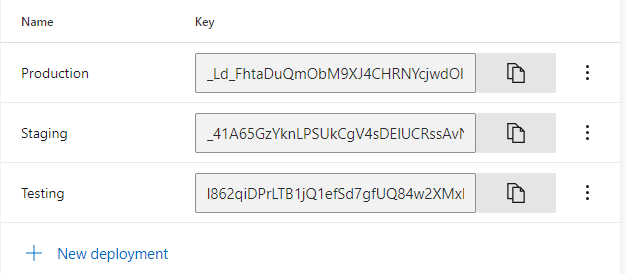
1. **To Generate separate Code Push Key navigate to CodePush Tab of AppCenter**

****

1. **Then click on Setting Icon On CodePush Dashboard**



1. **From there you can create new Deployment Key**



**STEP-2 [Android Native Changes]**

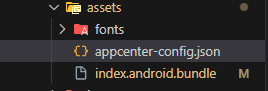
**After that Overview Tab will open & there we get Azure Key , & Further process Documentation**

1. Install below dependency in React app

npm install react-native-code-push –-f

npm install appcenter appcenter-analytics appcenter-crashes --save-exact

1. Create a new file with the filename appcenter-config.json in android/app/src/main/assets/ with the following content:



{

    "app\_secret": "6803654f-102a-4b2e-a98f-4c2519aa6c6a"

}

1. Modify the app's **res/values/strings.xml** to include the following lines:

 <string moduleConfig="true" name="CodePushDeploymentKey">DeploymentKey</string>

    <string name="appCenterCrashes\_whenToSendCrashes" moduleConfig="true" translatable="false">DO\_NOT\_ASK\_JAVASCRIPT</string>

    <string name="appCenterAnalytics\_whenToEnableAnalytics" moduleConfig="true" translatable="false">ALWAYS\_SEND</string>

1. Modify the app's **res/Java/packagename/MainApplication.kt** to include the following lines:

import com.microsoft.codepush.react.CodePush

override fun getJSBundleFile(): String { return CodePush.getJSBundleFile()}

1. Add below lines in  **android/settings.gradle**

include ':app', ':react-native-code-push'

project(':react-native-code-push').projectDir = new File(rootProject.projectDir, '../node\_modules/react-native-code-push/android/app')

1. To Create Separate CodePush for Different ENVs like(we can add our own or rename though) Development , Testing , Debug need to add below lines in **android/app/build.gradle File** .

buildTypes {

        debug {

            signingConfig signingConfigs.debug

            proguardFiles getDefaultProguardFile("proguard-android.txt"), "proguard-rules.pro"

        }

        release {

            signingConfig signingConfigs.debug

            minifyEnabled enableProguardInReleaseBuilds

            proguardFiles getDefaultProguardFile("proguard-android.txt"), "proguard-rules.pro"

            resValue "string", "CodePushDeploymentKey", '\_Ld\_FhtaDuQmObM9XJ4CHRNYcjwdOlOx7N8Kq' // key for particular env

        }

        releaseDevelopment {

            matchingFallbacks = ['release']

            signingConfig signingConfigs.debug

            minifyEnabled enableProguardInReleaseBuilds

            proguardFiles getDefaultProguardFile("proguard-android.txt"), "proguard-rules.pro"

            resValue "string", "CodePushDeploymentKey", '\_41A65GzYknLPSUkCgV4sDElUCRssAvNhytLS'

        }

        releaseTesting {

            matchingFallbacks = ['release']

            signingConfig signingConfigs.debug

            minifyEnabled enableProguardInReleaseBuilds

            proguardFiles getDefaultProguardFile("proguard-android.txt"), "proguard-rules.pro"

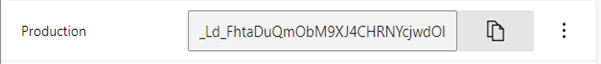
            resValue "string", "CodePushDeploymentKey", 'l862qiDPrLTB1jQ1efSd7gfUQ84w2XMxHKp1\_'

        }

    }

apply from: "../../node\_modules/react-native-code-push/android/codepush.gradle"

1. Key Used in this Generated from AppCenter Portal



**STEP-3** **[Generation of Private/Public.pem]**

1. To Generate of .Pem file need have installed “OpenSSl ” in your System [Click here to download](https://slproweb.com/download/Win64OpenSSL-3_2_1.exe)
2. Then Open bin folder of that Installed software in CMD as **System Administration**

****

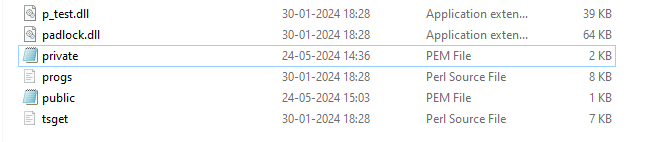
1. Then execute this command to generate Private .pem File “openssl genrsa -out private.pem”

****

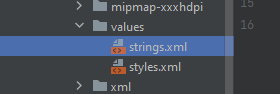
1. Then execute this command to generate Public .PemFile “openssl rsa -pubout -in private.pem -out public.pem”

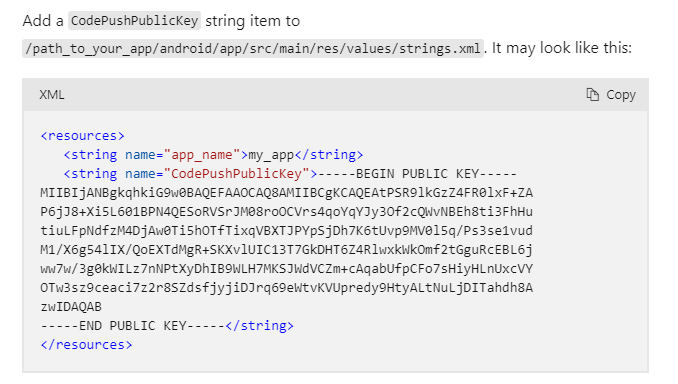
****

1. Here you can see two file were created

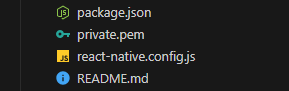


1. Now Add Public Key to Apps Strings.xml **file [ANDROID]**

****

****

1. Add Private.pem file to Root Of Your React Native Project

****

**STEP-4** **[React Native File Changes]**

**In App.js File**

import React, { Component } from 'react';

import { View, ActivityIndicator } from 'react-native';

import Navigation from './src/Navigation';

import CodePush from 'react-native-code-push';

import {\_getAsyncStorage, \_saveAsyncStorage} from './src/common/util';

import { connect, mapStateToProps, mapDispatchToProps } from './src/App';

var updateDialogOptions = {

  updateTitle: 'Latest App Update Available. Do you want to Install?',

  optionalIgnoreButtonLabel: 'No',

  optionalInstallButtonLabel: 'Yes',

};

const codePushOptions = {

  installMode: CodePush.InstallMode.IMMEDIATE,

  deploymentKey: '\_41A65GzYknLPSUkCgV4sDElUCRssAvNhytLS',

  checkFrequency: CodePush.CheckFrequency.ON\_APP\_START,

  updateDialog: updateDialogOptions,

};

class App\_ extends Component {

  constructor(props) {

    super(props);

    this.state = {

      isPackageDownloading: false,

      isUpdateReady: false,

      appUptoDate:false,

    };

  }

  componentDidMount() {

    this.checkForUpdate();

  }

  checkForUpdate = async () => {

    this.setState({ isPackageDownloading: true });

    try {

      await CodePush.checkForUpdate(codePushOptions?.deploymentKey).then((update)=>{

        if (!update) {

          console.log("The app is up to date!");

          this.setState({ isPackageDownloading: false, isUpdateReady: false,appUptoDate: true});

      } else {

        CodePush.sync(

          {},

          status => {

            switch (status) {

              case CodePush.SyncStatus.UP\_TO\_DATE:

                this.setState({ appUptoDate: true });

                break;

              case CodePush.SyncStatus.DOWNLOADING\_PACKAGE:

                this.setState({ isPackageDownloading: true });

                break;

              case CodePush.SyncStatus.INSTALLING\_UPDATE:

                break;

              case CodePush.SyncStatus.UPDATE\_INSTALLED:

                this.setState({ isPackageDownloading: false, isUpdateReady: true });

                break;

            }

          },

          // Progress callback

          progress => {

            // Update your loading indicator based on progress.percent

          }

        );

      }

      })

    } catch (error) {

      // Handle deployment not found error

      this.setState({ isPackageDownloading: false });

      this.setState({ deploymentNotFound: true });

    }

  };

  render() {

    const { isPackageDownloading, isUpdateReady,deploymentNotFound,appUptoDate } = this.state;

    if (isPackageDownloading) {

      // Show a loading indicator while the package is downloading

      return (

        <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>

          <ActivityIndicator size="large" color="#0000ff" />

        </View>

      );

    } else if (isUpdateReady) {

      // If an update is ready, display a message to restart the app

      return (

        <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>

          <Text>Update is ready, please restart the app.</Text>

        </View>

      );

    } else if (appUptoDate) {

      {console.log('continue to app',appUptoDate)}

      // If no update is available or downloading, render the main app content

      return <Navigation />;

    }

  }

}

export default connect(

  mapStateToProps,

  mapDispatchToProps,

)(CodePush(codePushOptions)(App\_));

**STEP-5** **[STEPS TO FOLLOW WHILE CODE PUSH]** [**Click Here For Refi**](https://learn.microsoft.com/en-us/appcenter/distribution/codepush/cli#code-signing)

1. **Update deployment Key of your Project/Bank**

const codePushOptions = {

  installMode: CodePush.InstallMode.IMMEDIATE,

  deploymentKey: '\_41A65GzYknLPSUkCgV4sDElUCRssAvNhytLS',

  checkFrequency: CodePush.CheckFrequency.ON\_APP\_START,

  updateDialog: updateDialogOptions,

};

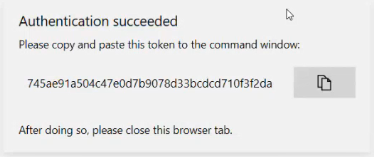
1. **Then Execute Code Push Code**

npx appcenter codepush release-react -a netwinAddon/Netwin-React-Mobile-Banking -d Staging -k private.pem -m --description "DEPOSIT SCREEN CHANGES BY DEVELOPER"

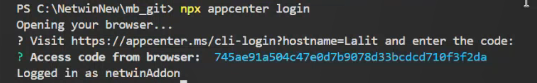
1. **If Above code get Error Then Need to Login App Center Again From VSCode**

npx appcenter login

1. **After executing above code It will redirect to App Center Portal and you get Authentication code**

****

1. **Then Enter the Authentication code**

****

### **Command Syntax BreakDown**

npx appcenter codepush release-react -a <appName> -d <deploymentName> [options]

### **Parameters and Options**

* -a, --app <appName>
  + Description: Specifies the application name in the format ownerName/appName.
  + Example: netwinAddon/Netwin-React-Mobile-Banking
  + Required: Yes
* -d, --deployment-name <deploymentName>
  + Description: Specifies the deployment where the update should be released. Common deployment names are Staging and Production.
  + Example: Staging
  + Required: Yes
* -k, --private-key-path <privateKeyPath>
  + Description: Specifies the path to a private key file used to sign the update. This is optional and used for enhanced security.
  + Example: private.pem
  + Required: No
* -m, --mandatory
  + Description: Marks the release as mandatory. If an update is mandatory, users must install it.
  + Required: No
  + Flag: This is a boolean flag. When included, it sets the release as mandatory.
* --description <description>
  + Description: Provides a description of the release. This is helpful for keeping track of changes in each release.
  + Example: "Testing the codepush update for xyz bugfix"
  + Required: No

### **Full Example Command**

npx appcenter codepush release-react -a netwinAddon/Netwin-React-Mobile-Banking -d Staging -k private.pem -m --description "Hemant banking homescreen recharge slider"

### **Detailed Breakdown**

* **npx appcenter codepush release-react:** This is the base command used to release a React Native update via CodePush.
* **-a netwinAddon/Netwin-React-Mobile-Banking**: Specifies the application name. Here, **netwinAddon** is the owner and **Netwin-React-Mobile-Banking** is the app name.
* **-d Staging**: Specifies that the release should be deployed to the Staging deployment.
* **-k private.pem**: Uses private.pem as the private key file for signing the update.
* **-m**: Marks this release as mandatory.
* **--description "Banking home screen changes for recharge slider"**: Provides a description for this release, detailing the changes or new features included.

### **Get Code push Code Directly From App Center**

**You can get Release code directly from Appcenter/Project/Distribute/CodePush Option**

****

**Refi Link :- “**<https://medium.com/@ritika100898/implementing-codepush-for-over-the-air-updates-in-react-native-535e750ec308>**”**

**PemKey Refi Link : - “**<https://learn.microsoft.com/en-us/appcenter/distribution/codepush/rn-get-started>**”**