Sardar Kulveer Singh

Salesforce-azureDevops cicD

**Table of Contents**

**Purpose2**

**Scope2**

**Prerequisite3**

**Creation of salesforce account3**

**Setting Up SFDX Project5**

**Push and retrieve code from salesforce without using Version Control7**

**Deploy Salesforce Application Using Azure DevOps8**

**Certificate & Key8**

**Configuration of connected App (Salesforce)9**

**Azure DevOps Setup of salesforce project10**

**Sonar Cloud set-up 11**

**Implementation of CI pipeline for salesforce application 12**

**Implementation of CD pipeline for salesforce application 13**

## Purpose:

8 the power of Azure DevOps and streamline your Salesforce application deployment process.

## Scope:

The scope for deploying Salesforce SFDX applications to Salesforce using Azure DevOps includes the following aspects:

* **Setup and Configuration:**
  + **Configuring Azure DevOps**: This involves creating Azure DevOps projects, setting up repositories, and configuring pipelines.
  + **Setting up Salesforce SFDX**: This includes installing Salesforce CLI and configuring authentication to Salesforce orgs, creating connected apps for service connections.
* **Packaging:**
* **Defining Build Pipelines**: Creating pipelines to convert the Salesforce SFDX application, retrieve metadata from source control, and validate the application's structure, creating the actual deployable folder.
* **Sonar Analysis**: setting up SonarQube for the code analysis and sending the reports to the sonar.
* **Continuous Integration (CI):** Configuring pipelines to trigger on every commit, ensuring that changes are continuously built and validated.
* **Deployment Automation:**
  + **Release Pipelines:** Setting up release pipelines to manage the deployment process from development to production environments.
  + **Artifact Management:** Managing the artifacts generated during the build process and storing them for deployment.
  + **Deploying to Salesforce Orgs:** Automating the deployment of Salesforce SFDX applications to different environments (e.g., sandbox, production).
  + **Environment Configuration:** Handling environment-specific configurations during the deployment process.
* **Testing and Validation:**
  + **Test Automation:** Integrating automated testing into the deployment pipelines, including unit tests.
  + **Validation and Quality Checks:** Implementing code quality checks, static code analysis, and code coverage checks to ensure the application meets quality standards, and publishing test results and test coverage to azure pipelines.

## Prerequisites:

* Azure DevOps Account
* Salesforce Account

# Creation of Salesforce Account

To create a Salesforce account, follow these steps:

* Go to the Salesforce website: Visit the Salesforce website at www.salesforce.com.
* Click on "Try for Free" or "Start Free Trial": Look for a button or link on the website that allows you to start a free trial or create a new account. The exact wording may vary.

A screenshot of a computer

Description automatically generated with medium confidence

* Choose the edition: Salesforce offers different editions of its platform, tailored for various business needs. Select the edition that best suits your requirements. Options may include Salesforce Essentials, Professional, Enterprise, or Unlimited.
* Fill out the registration form: Provide the necessary information in the registration form. This typically includes your name, email address, job title, company name, and contact information. You may also be asked to provide some additional details about your business.

A screenshot of a computer

Description automatically generated with low confidence

* Verify your email address: After submitting the registration form, Salesforce may send a verification email to the email address you provided during registration. Follow the instructions in the email to verify your email address.

A screenshot of a computer

Description automatically generated with medium confidence

* Set up your password: Once your email address is verified, you will be prompted to set up a password for your Salesforce account. Choose a strong password that meets the specified requirements.
* Complete the setup process: Salesforce may guide you through a setup wizard or provide a series of steps to configure your account. This may include selecting your preferred language, time zone, and other account settings.
* Explore the Salesforce environment: Once the initial setup is complete, you will have access to your Salesforce account. Take some time to familiarize yourself with the Salesforce interface, navigation, and features.

# Setting Up SDFX Project:

* Install Salesforce Cli
* To connect the VsCode with Salesforce org we need SalesforceDX CLI. Download the Salesforce CLI

## Install Salesforce Extension Pack For Visual Studio Code

A screenshot of a computer

Description automatically generated

* With Salesforce Extension pack all below extension will install automatically.

1. Apex
2. Salesforce CLI Integration
3. Apex Interactive Debugger
4. Apex Replay Debugger
5. Visualforce
6. Aura Components
7. Lightning Web Component

## **Create Salesforce DX Project**

Now its time to create your first project in VsCode. To Create the project open **Command Palette** or press **Ctrl + Shit + P.**Then type **SFDX: Create Project with Manifest.**

A screenshot of a computer program

Description automatically generated with medium confidence

* Then provide your project name and select location where you want to save your project.

A screenshot of a computer

Description automatically generated with medium confidence

## **Connect With Your Salesforce Org**: Authorize an Org

* Our project is ready on our local machine, Its time to connect our VsCode with Salesforce. Again, open **Command Palette** or press **Ctrl + Shit + P.** This time we need to type or Select “**SFDX: Authorize an Org**“.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

* In the last step you need to provide the org alias name. Then it will take you to your default browser and ask you for org credentials. If prompted to allow access, click **Allow**A screenshot of a phone

  Description automatically generated with low confidence

# Push and retrieve code from salesforce without using Version Control:

## **Retrieve Component** From Salesforce Org Using VsCode

* Now you want to extract all your Salesforce component in your VsCode? Open **Package.xml**file from **Manifest**folder. Then Modify your Package.xml file to add and remove component. Finally Click on Package.xml and select “**SFDX: Retrieve This Source from Org**” option.

A screenshot of a computer program

Description automatically generated with medium confidence

* It’ll retrieve all the metadata from your org which is specified in your package.xml file

## **Deploy Component**Using Visual Studio Code

# Now do your changes in VSCode and then select the file which you want to deploy in your org. Then Right click on VsCode and select the “**SFDC: Deploy This Source to Org**” option for deployment.

A screenshot of a computer

Description automatically generated

# **Deploy Salesforce Application Using Azure DevOps:**

## **Certificate & Key :**

The first step is to create a self-signed certificate and private key that we need for configuring the DevOps process to authorize with Salesforce org.

# If your operating system is Windows, you have to install OpenSSL before you attempt this command. In linux, you don’t have to install anything. Just execute the commands.

In your terminal/command prompt, type the following command. This creates the private key named ‘**server.key**

* **openssl genpkey -algorithm RSA -pkeyopt rsa\_keygen\_bits:2048 -out server.key**

openssl genpkey -algorithm RSA -pkeyopt rsa\_keygen\_bits:2048 -out server.key

* **openssl req -new -key server.key -out server.csr**

openssl req -new -key server.key -out server.csr

* **openssl req -new -key server.key -out server.csr**

## Configuration of connected App (Salesforce):

In this step, we will create a new connected app for DevOps process to authorize using the certificate that we created earlier.

1. Login in to Salesforce Developer Org
2. Navigate to Setup -> Apps -> App Manager
3. Create a new Connected App with the following details and save it.
   * Connected App Name = “DevOps App”
   * Contact Email = specify your personal email address
   * Enable OAuth Settings = tick mark it to checked state
   * Call-back URL = http://localhost:1717/OauthRedirect
   * Use digital signatures = tick mark it to checked state
     + Browse and select the **server.crt** file from your local machine.
   * Selected OAuth scopes
     + Access and manage your data (api)
     + Access your basic information (id, profile, email, address, phone)
     + Perform requests on your behalf at any time (refresh token, offline access)
     + Provide access to your data via the Web (web)
   * Require Secret for Web Server Flow = tick mark it to checked state
4. Click the “Manage” button the connected app, set the following and save.
   * Permitted Users = Admin approved users are pre-authorized
5. After saving the permitted users, scroll down to “Profiles” related list and click the “Manage Profiles” button. Add the “System Administrator” profile or equivalent profile that your DevOps user is setup with.

**Test if SFDX authorization to SF org is successfull or not.** On executing the force:auth:jwt:grant command, it should say “Successfully authorized xxx@xxx.com with org ID 00D2x000000aBcDEAM”. If this fails, then either the ClientId copied & pasted is not proper or certificate/key file is not generated properly from OpenSSL. Ensure to replace the username, client id with you own values.

* sfdx force:auth:jwt:grant --clientid 3MVG97quAmFZJfVyzexU2c1VnTmNIkZ5g1IwJ\_abcd\_menLDWTuYasRhsInZHkA.Jfw.BmI4rbHYmjdzZBeqC --jwtkeyfile server.key --username xxx@xxx.com --instanceurl <https://login.salesforce.com>

## Azure DevOps Setup of salesforce project:

## **Project Setup in Azure**

1. Login to dev.azure.com

# Create a new project named “Dreamhouse-Salesforce” And save it.

* + Visibility should be Private.
  + Version Control should be selected as Azure Repos.
  + Work Item Process can be left with default option (Agile)

1. Open the project.
2. Click the “Repos” tab.
3. Click on “Generate Git Credentials”. Copy the username and password somewhere safely. If you refresh this screen, you will not be able to see the password again. We will require this username & password when executing git commands. (I needed this step while configuring the DevOps in Ubuntu Linux machine. While trying out in Windows machine, it didn’t even need this generated Git credentials.)
4. Push your code to the azure repos

## A screenshot of a computer Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

# Creating the Azure Pipeline for automated build & deploy :

## **Sonar Cloud set-up:**

* If your code is on Azure, go to the [SonarCloud](https://www.sonarsource.com/products/sonarcloud/" \t "__blank) product page and choose **Set up** or **Login**, then select **Azure** from the list of DevOps cloud platforms.

A screen shot of a log in

Description automatically generated with low confidence

* You will be taken to the Microsoft login page. Sign in using your Microsoft credentials.
* Once you have successfully logged in, you will see the SonarCloud welcome screen.
* Select **Import an organization from Azure** to bring your projects into SonarCloud .

A screenshot of a computer

Description automatically generated

* You need to enter the name of your Azure DevOps organization and an Azure-generated Personal Access Token created in that organization.
* To create the token, go to your Azure DevOps organization **User settings** > **Personal access tokens**, then select **+ New token**.

A screenshot of a computer

Description automatically generated with medium confidence

* On the next page, under **Scopes**, make sure that you specify at least the scope **Code** > **Read & write**.
* Then, click **Create** to generate the token.
* When the personal access token is displayed, copy/paste it into the field on the SonarCloud setup page.

## Creating the Azure Pipeline (CI):

1. In the Azure Project page open in the browser, navigate to “Pipelines” tab.
2. Click on “Create Pipeline”
3. Choose “Azure Repos Git” for “Where is your code?”
4. Choose “SF DevOps” project for “Select your repository”.
5. Choose “Starter Pipeline” for “Configure your pipeline”.
6. Overwrite the code shown in “Review your pipeline YAML” and paste the following code. (**Note**: a usual mistake that people do is to copy-paste the pipeline code and mess up the indentation of the yml code. If indentation is not right, you will have a tough time running the pipeline.)

**Click on below notepad to view the yaml pipeline script.**



* Click on the “Variables” button and create the following variables
  + **salesforceDevOrgClientId** = paste the client id from the connected app we created in Step 2
  + **salesforceDevOrgInstanceURL** = https://login.salesforce.com
  + **salesforceDevOrgUserName** = type the username of your developer org
* Click on “Save & Run”. You will see the pipeline starting to run. You can click the build instance and see what the azure pipeline is doing while it is executing the commands.
* After the pipeline is executed, you can open the Salesforce Org in browser and navigate to Setup -> Environments -> Deploy -> Deployment Status. Here, you will see a recent Deployment Validation Success & a Deployment Success.

## Creating the Azure Pipeline (CD):

* Click on Release pipeline and create a new pipeline

A screenshot of a computer

Description automatically generated with medium confidence

* Add two task in the stages (Stages depends upon the environments you want to deploy)

A screenshot of a computer

Description automatically generated

* Add following script in the task wich authorize the salesforce accounts & Deploys the code to your environmanet

A screenshot of a computer

Description automatically generated with medium confidence

* This script use to authorize to your salesforce Account.

A screenshot of a computer

Description automatically generated with medium confidence

* The above script used to deploy the code to the salesforce account

Once the pipline is succefully completed you can see the test results, code coverage & analysis reportds in your build pipeline dashboard.

**Test Results:**

A screenshot of a computer

Description automatically generated with medium confidence

**CodeCoverage:**

A screenshot of a computer

Description automatically generated with low confidence

A screenshot of a graph

Description automatically generated with low confidence

**Sonar Analysis Report:**

**A screenshot of a computer

Description automatically generated with medium confidence**