

Salesforce.org EZ Datalake

Handover Document

Post-PoC Recommendations and Path to Production

Prepared for Salesforce by the Envision Engineering Team
May 2022



EnvisionEngineering

"© 2022 Amazon Web Services, Inc. or its affiliates. All Rights Reserved.
This AWS Content is provided subject to the terms of the AWS Customer Agreement available at <http://aws.amazon.com/agreement> or other written agreement between Customer and either Amazon Web Services, Inc. or Amazon Web Services EMEA SARL or both."

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

© 2022 Amazon Web Services, Inc. or its affiliates. All rights reserved.

Contents

1	Introduction	4
2	The Given Challenge	4
3	Success Criteria	4
4	Solution Overview	5
4.1	Component Architecture	6
5	Path to Production	10
5.1	Post PoC Recommendations.....	10
5.2	Skills Required	23
5.3	Software Development Languages and Frameworks	23
5.4	AWS Services	23
6	Deployment Guide.....	24
7	User Guide.....	24
8	Lessons Learned.....	25

1 Introduction

This document provides instructions for building and deploying the Salesforce.org PoC code in an AWS account.

2 The Given Challenge

NPOs have a relatively large amount of data but lack the skillset, the resources and mindset that could enable the data for actionable intelligence. Actionable intelligence will help NPOs measure impact, fundraising and drive marketing efforts. For NPOs that are at different stages in their Cloud adoption strategy, or are evaluating how cloud can drive their mission, ingesting data into AWS Cloud and integrating with Salesforce and Tableau requires multiple steps to be done by the user which is a blocker to cloud adoption and realizing the benefits. An easier method with low-tech options to setup AWS resources and integrate with Salesforce for data ingestion into the cloud and make it ready for analytics is desired. This will help NPOs overcome the hurdles associated with big data and analysis and accelerate their digital transformation journey.

3 Success Criteria

The table below lists the Success Criteria for this Proof of Concept and the outcome of each one.

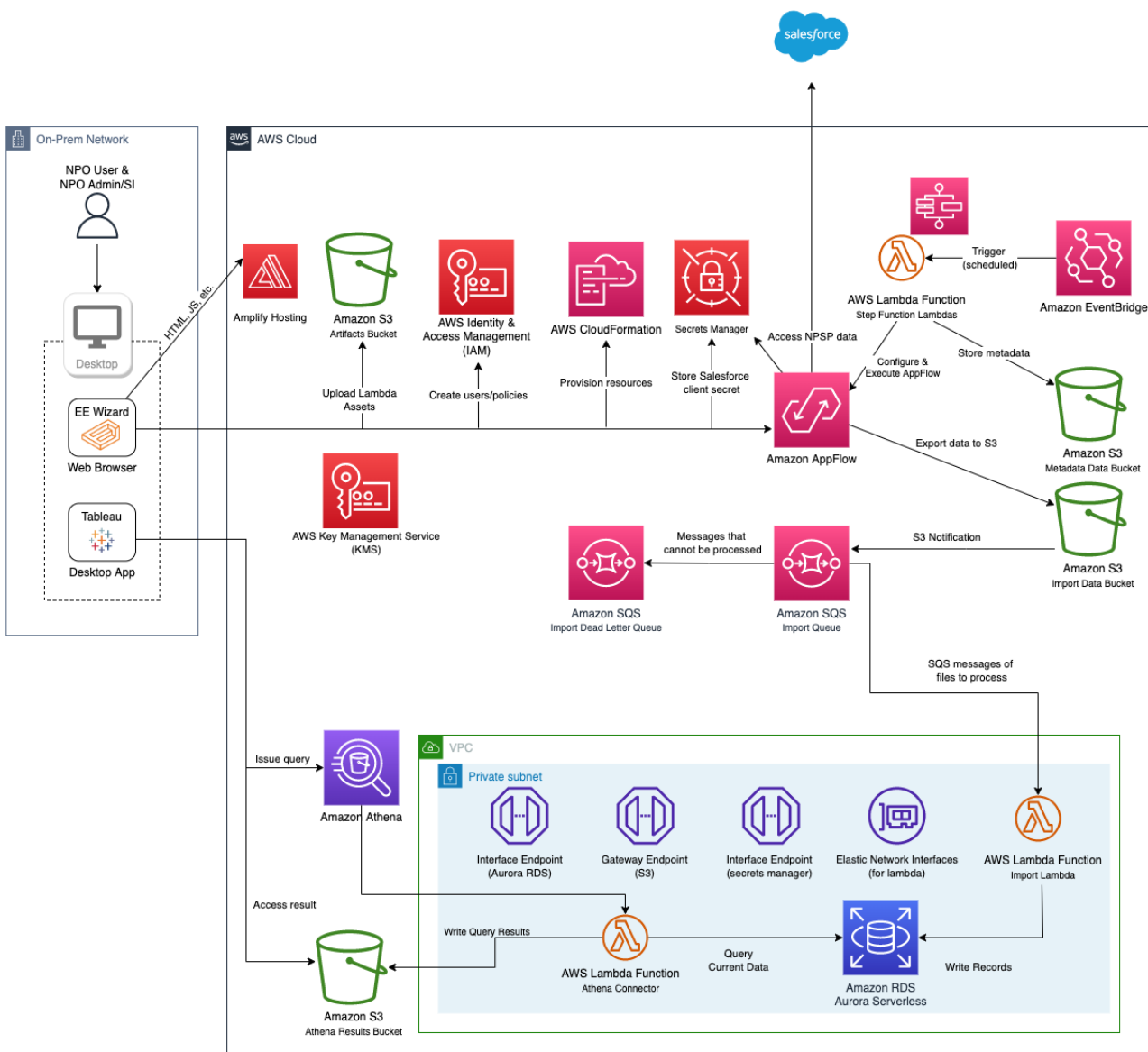
Criteria	Description	Type	Status
SC1	Solution must have a wizard-like UI with default options to deploy required AWS resources and integrate Salesforce. Reduce current steps of 24 to 6-8 click throughs	Must have	Done
SC2	Solution must have flexible options to allow for changing default parameters and allow for manual steps to be carried out for a custom integration with appropriate warnings/notifications to users when deviating from default options	Must have	Done
SC3	Solution must be able to connect to and import data from NPSP schema on initial setup as well as periodic sync operations	Must have	Done

SC4	Solution must display required endpoints for Athena to connect Tableau	Must have	Done
SC5	Solution must authenticate user into SF to import data, create appropriate IAM roles needed for deployment and, allow sys admin to create users with limited privileges: to view data analytics and run SQL queries	Must have	Done
SC6	Solution must ensure that data is consistent by validating imported data by running queries after wizard is run and setting triggers to run data sync when underlying data catalog changes	Must have	Done
SC7	When data is deleted in Salesforce, data can become out of sync and invalidate the analytics. The solution must check for data consistency between S3 and Salesforce before data is brought in. If inconsistent, the user must be notified to work with SI to take necessary action in the back end.	Must have	Done
ST1	Support for Private Connect	Stretch	Not Done
ST2	Filter options	Stretch	Not Done
ST3	Solution shows in-app guidance for Tableau Desktop installation and connectivity (Added for improved UX)	Stretch	Done
ST4	Solution shows a way for NPO Users to monitor the data lake resources in a single view	Stretch	Done
ST5	Solution shows how to reconnect or resume data lake hydration	Stretch	Done

4 Solution Overview

This section provides an overview of the components and services we used to develop the Proof of Concept.

4.1 Component Architecture



4.1.1 Component Summary

The zip file deployed to Amplify contains the UI code as well as zip files and YAML files to be hosted as assets by Amplify. This means of the two major components, the front-end and the back-end, the front-end is exploded inside of the zip file whereas the back-end components are all mostly contained within zip files inside of the Amplify zip file (therefore they are compressed within the zip file). When the UI sets up the infrastructure, the back-end setup process will fetch files from the Amplify host and move them into a

system accessible location to be used with the setup process. In all, the high-level components of the system are thus: UI, CloudFormation templates and code asset zip files.

Component Group	Component	Description
Front-end	UI	The static web UI which the users use that interacts with AWS APIs to set up the system.
Back-end	AWS Step Functions	The step function handles orchestration of the daily/weekly/monthly synchronization between Salesforce and AWS. It is the step function's job to perform any necessary setup or initialization logic, execute the import process as well as any cleanup logic to expose the data to the user.
Back-end	Import Queue	The step function never handles the actual data to be imported, only controlling the processes which operate on it. Once data extraction from Salesforce has begun it will move to the Import Queue to be written to RDS.
Back-end	Data Storage & Query Compute	Storage and querying of the data are handled by RDS and Athena. Athena has access to the VPC and RDS cluster through an AWS lambda which is responsible for performing the query and writing results to the output bucket.
Back-end	Support Infrastructure	The remaining miscellaneous resources are categorized into the support infrastructure component because they don't have a natural place otherwise. They contain the things like buckets or the VPC & networking setup which can't be clearly assigned to any one component.

4.1.2 Code: Solution Directory Structure

The system is divided into two parts, app is the front-end web UI and infra are all the back-end components. The docs folder contains documentation written in markdown which is relevant to development of the system. They are structured as follows:

Folder	Description
app	front-end web UI
infra	back-end components

docs	documentation written in markdown which is relevant to development of the system
------	--

4.1.2.1 Front-end Web UI

The front-end web UI is a single page application (SPA) and it is based on CRA (create-react-app) with the typescript template. The web application uses [Chakra UI](#) as its main UI component library and uses [MobX State Tree \(MST\)](#) as the underlaying library for representing the data models for the UI. The web application is sub-divided into 3 parts:

Folder	Description
public	The public folder is where the root of the static files is located and contain files which need no build process in order to produce the final asset.
scripts	The scripts folder contains developer helper scripts, described below, for working with the system.
src	Last, the src directory contains all the source Typescript files for the web UI.

The following scripts are available to run:

- `npm run delete-stack` — Running this command will trigger a process which will prompt you to answer some questions and then according to the answers will delete the stack you requested. This command is useful because it will ensure that resources produced by the stack but are not part of the stack get deleted. Some of these things may include CloudWatch logs generated by Lambda or files inside an S3 bucket that needs to be deleted before the stack can be deleted.
- `npm run delete-datalake` — The most commonly used command to clean up a data lake which is no longer needed. This command will find all the stacks for the given installation ID and use the above delete stack command to clean them all up. This process ensures no left-over resources are left in the account.
- `npm run copy-cf` and `npm run copy-assets` — These commands are run automatically during the build process. They pull files from the (assumed) already built infra folder and place them into the public folder so that they may be used by the system from the Amplify URL. Also, in order to bootstrap the system, the CloudFormation templates are copied into the project as .ts files so that the front-end may create the first buckets stack.

The src folder is organized based upon how the different components are used and has the following subfolders:

Folder	Description
--------	-------------

api	contains all functions which interact with the AWS SDK and by extension the AWS API
components	contains all the reusable .tsx React components which are used throughout the web UI
data	contains the Typescript formatted version of the CloudFormation templates
data/cf	
helpers	contains various reusable helper files for the UI
images	contains all the .png files needed for the UI
models	contain the processes and logic required to execute each step of the system, including the setup of all the infrastructure using CloudFormation
models/operations	
models/steps	
routes	contains the different pages of the system, the top level .tsx file for each page
themes	defines the styling for each page's color changes so that the pages and components files only need to identify what color they are using but not define the details of how that affects each element

4.1.2.2 Back-end Infrastructure

The back-end infra folder is basically divided into 2 parts: the lambda code and the CloudFormation templates. However, the lambda code actually takes up 2 folders: custom_resources and lambdas. Additionally, there exists an assets folder as below:

Folder	Description
assets	hold the built assets from the custom_resources and lambda folders which they write their final output zip file to
cf	contains all the .yaml files which are CloudFormation templates
custom_resources	contains CloudFormation custom resource lambdas needed for the system to be set up properly
lambdas	The custom_resources and lambda projects are very simple projects with just a src folder containing source files and not much else in terms of folder organization.

The custom_resources and lambda folders are NPM projects and have their own build process each, however they are the same command (npm install). There also exists one other custom resource lambda that is not in this project. It exists within the cf/buckets.yaml and is inlined into the template using JavaScript. The reason for this division is because a bucket with the zip file is needed before executing the CloudFormation which needs the custom resource. This lambda is basically responsible for

bootstrapping all the assets needed by CloudFormation and putting them into a bucket to be referenced. Therefore, it cannot be inside the asset it is bootstrapping.

5 Path to Production

5.1 Post PoC Recommendations

Envision Engineering have built a Proof of Concept (PoC) for Salesforce.org. There are gaps that we recommend you address before you take the Proof of Concept to production. We have produced a non-exhaustive list of gaps that we believe exist between the proof of concept and a production-ready solution.

5.1.1 Github

EZ Datalake app uses AWS Amplify to host the web app in the NPO User Account. A [manual deployment](#) is used to publish the app without connecting to a Git provider. For EE's testing and demonstration, we upload assets to host the site on AWS Amplify as described [here](#).

All apps > Manual deploy

Manual deploy

Manually upload objects to deploy your app. You can choose to drag and drop the artifacts directly, pull a zip from an existing S3 bucket or any other URL.

Start a manual deployment

App name
Give this app a name or we will generate a default for you

Environment name
Give this resource a meaningful environment name, like dev, test, or prod, or we will generate a default for you

Method

☒ Drag and drop

☐ Amazon S3

☐ Any URL

sforg-04-19-2022.zip

×

Cancel Previous **Save and deploy**

5.1.1.1 Post-PoC recommendation

As a post-PoC step, the required asset files should be hosted on github for the user to download and upload to AWS Amplify or the user can specify a public URL to the location where the asset files are stored (see screenshot below). Build the zip file by running `./clean.sh && ./build.sh` from the root of the project. This will create a zip file under `app/amplify.zip` which can be pushed to a github release. This is a **required post-PoC step**.

The screenshot shows the 'Manual deploy' page in AWS Amplify. The 'Start a manual deployment' section includes fields for 'App name' (filled with 'ez-datalake') and 'Environment name'. Under the 'Method' section, three options are shown: 'Drag and drop', 'Amazon S3', and 'Any URL' (which is selected). Below the methods, the 'Resource URL' field is highlighted with a red circle. It contains the text 'Ex: https://myhostedurl.com/packagezipfile.zip'. A note below the field states: 'Note: Only zip the contents of your build directory, not the entire directory itself.' At the bottom right, there are buttons for 'Cancel', 'Previous', and 'Save and deploy'.

To make the file available to Amplify, once built it can be stored in Github using the Releases feature. See how to [create a release once the artifact is built here](#). This manual mechanism is recommended for stable releases. For automatic releases of the main branch, see [Github's Automated Releases](#) documentation for how to set that up.

5.1.2 Documentation on Github to create AWS Account

AWS Amplify requires the creation of an AWS account as a step prior to hosting the EZ Datalake App. Github Readme should have documentation to help the NPO User to create an AWS account and follow best practices to secure the account. Few suggestions are:

- AWS Account Creation:

<https://aws.amazon.com/premiumsupport/knowledge-center/create-and-activate-aws-account/>

- Enable MFA for AWS account root user:

https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa.html

- https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_enable_virtual.html#enable-virt-mfa-for-root

5.1.3 In-app Guidance/Documentation Review

EZ Datalake App has instructions in all steps including how to create IAM User, Salesforce account, AppFlow connections, connecting to Tableau, etc. We recommend you review all guidance including verbiage and add more documentation as necessary based on user testing and feedback.

5.1.4 In-app support for alternate regions

In Step 1 of the 6 step process in EZ Datalake, we will ask the NPO user to select the region where the resources should be provisioned.

AWS Region

AWS has many data centers grouped by geographical regions. Select a region that is closest to your location. We will keep your data in the region you select. If you are not sure, you can select the US East (Ohio) Region

US East (Northern Virginia) Region

← Previous

Next →

AWS Region

AWS has many data centers grouped by geographical regions. Select a region that is closest to your location. We will keep your data in the region you select. If you are not sure, you can select the US East (Ohio) Region.

US East (Ohio) Region
✓ US East (Northern Virginia) Region
US West (Northern California) Region
US West (Oregon) Region

← Previous
Next →

Below is the list of the available regions and of the required services are available as of April 2022. This list needs to be updated and maintained as services are rolled out across the regions. Refer to the handover document to see the list of services required for the EZ Datalake to be deployed successfully in any region.

Core services, such as S3, EC2, Lambda, IAM, KMS, CloudWatch, CloudFormation, SNS and SQS are not listed here as they are required for a region to go live and so are assumed to be present in every region the user would use. The documentation to look at to determine if a service is located within a given region is [is documented on the AWS Regional Services page](#).

Region	Name	AWS Aurora Serverless	AWS AppFlow	AWS Amplify	AWS Step Functions	AWS Secrets Manager	AWS Athena
us-east-1	US East (N. Virginia)	✓	✓	✓	✓	✓	✓
us-east-2	US East (Ohio)	✓	✓	✓	✓	✓	✓
us-west-1	US West (N. California)	✓	✓	✓	✓	✓	✓
us-west-2	US West (Oregon)	✓	✓	✓	✓	✓	✓
ca-central-1	Canada (Central)	✓	✓	✓	✓	✓	✓
ap-northeast-1	Asia Pacific (Tokyo)	✓	✓	✓	✓	✓	✓
ap-northeast-2	Asia Pacific (Seoul)	✓	✓	✓	✓	✓	✓
ap-south-1	Asia Pacific (Mumbai)	✓	✓	✓	✓	✓	✓

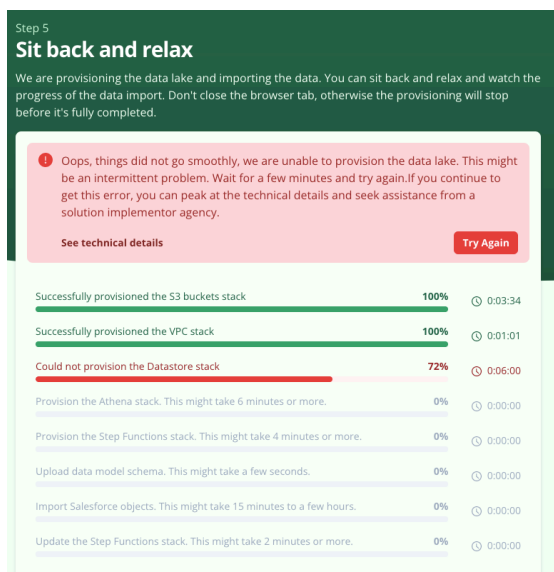
ap-southeast-1	Asia Pacific (Singapore)	✓	✓	✓	✓	✓	✓
ap-southeast-2	Asia Pacific (Sydney)	✓	✓	✓	✓	✓	✓
eu-west-1	Europe (Ireland)	✓	✓	✓	✓	✓	✓
eu-west-2	Europe (London)	✓	✓	✓	✓	✓	✓
eu-west-3	Europe (Paris)	✓	✓	✓	✓	✓	✓
eu-central-1	Europe (Frankfurt)	✓	✓	✓	✓	✓	✓
eu-north-1	Europe (Stockholm)	✗	✗	✓	✓	✓	✓
me-south-1	Middle East (Bahrain)	✗	✗	✓	✓	✓	✓
ap-southeast-3	Asia Pacific (Jakarta)	✗	✗	✗	✓	✓	✗
ap-northeast-3	Asia Pacific (Osaka)	✗	✗	✗	✓	✓	✓
eu-south-1	Europe (Milan)	✗	✗	✓	✓	✓	✓
af-south-1	Africa (Cape Town)	✗	✓	✗	✓	✓	✓
sa-east-1	South America (São Paulo)	✗	✓	✓	✓	✓	✓
ap-east-1	Asia Pacific (Hong Kong)	✗	✗	✓	✓	✓	✓

5.1.5 Review instructions for installing Tableau Desktop and connecting to Athena

In Step 6, of the 6 step process in EZ Datalake, we offer instructions for NPO users to install Tableau and use the information provided by EZ Datalake setup to connect to AWS Athena. These instructions are based on EE's research across Tableau's documentation, community forums and observed behavior (when documentation was unavailable) through our testing with Tableau Desktop. We recommend review and verification of these instructions for Tableau Desktop installation for Mac and Windows and location of the athena.properties file with the Tableau Tech team to validate and correct as needed.

5.1.6 Implement Error-handling Processes

The easy datalake app shows errors on the UI if they occur during deployment of AWS resources. An example screenshot is shown below. Users can click on provided links to understand error conditions. However, error handling in the application, is not exhaustive. We recommend implementing more robust error handling to help customers understand if there was an error deploying the AWS resources for this solution and a documented process for users to get the needed help from SIs.



5.1.7 Evaluate Aurora Serverless v2

At the time of this writing, Aurora Serverless v2 is available in regions where Aurora Serverless is available. Aurora Serverless v2 will provide the user with significantly faster cold start time, quicker scaling time for large workloads, separate scaling for writers or readers, multiple availability zone support, global databases and read replicas. These benefits may be attractive to some customers.

However, there is no support for using Aurora Serverless v2 with CloudFormation at this time. In addition, Aurora Serverless v2 does not have an auto-pause feature like v1 has. This may lead to potential cost increase for users who do not use the system frequently.

5.1.8 Review App-level Logging

We recommend you review the current items that are logged by the backend code to ensure they are both sufficient and necessary for audit purposes.

5.1.9 Configurable CloudWatch Log Duration

Currently, CloudWatch logs are kept indefinitely. We recommend you make the duration of CloudWatch logs configurable so that you won't incur an unnecessary cost in storage, but at the same time have the logs available long enough to do an audit.

5.1.10 Step Function

The step function in the system currently doesn't handle some failure scenarios. It is recommended **not** to add an error handler however, because this error handler masks the error and won't set off the alarm appropriately since the exit is successful. Additionally, there is no logic to perform on failure since it needs a developer's attention and there's no resources to recover. However, if some systems external to the step function fail, the step function does not handle it properly. If, for example, an AppFlow fails while extracting data from Salesforce, the step function waits indefinitely for "Success" which will never occur. A failure check to this choice in the step function should be added which shuts down the whole system (go to a Fail type).

5.1.11 Github Governance

- **Credential Security:** Future changes to the product must be handled with care. Future changes to the product **must** be well governed. Without this needed governance, AWS credentials of customers will be at risk. Future changes to the code must be monitored and handled with care. **The github project should be setup with very clear code review policies and practices and code releases must be handled by trusted individuals.**
- Project defragmentation risk: we recommend governance policies to specify and control how users can clone, fork the repo, etc
- Support to user: we recommend policies to determine how users will submit support tickets, etc
- Showing costs to use: we recommend adding guidance on educating user on how to explore costs. We recommend documentation for users to learn AWS console to see cost

5.1.12 How to edit objects and fields - post deployment

The system writes .schema.json files to a bucket in S3. In order to change the objects and/or fields the system should import, changing these files is required. This task is a bit technical and as so may be difficult for some users to accomplish. Improper formatting or syntax errors will result in complete system failure. As a feature as part of the system, the UI could do these tasks from the resume screen to make this feature more widely available.

5.1.12.1 Adding/Removing Fields in an Existing Object

For a Salesforce Object which is already being imported to the system, to add or remove a field you must download the corresponding file in S3. Visit the AWS S3 console, and in the sf-metadata-<installationId> bucket, under the schemas folder will be all the files. Find the one that starts with the Salesforce Object name you intend to edit and download the file. In an editor of your choice, move fields within the JSON file to/from the excluded and properties top level fields. Fields in the excluded will not be imported, and fields in the properties will be imported. If you simply remove the field, it will get repopulated into the properties field so this does not work and so must be copied into the exclude section. Once complete, upload the file back to S3 with the same name. See the section below on how to do that. The next time the step function runs to sync the data, your changes will be applied. To do this manually, see the section below on running the sync manually.

5.1.12.2 Adding/Removing Salesforce Objects

To add a Salesforce Object you need to write a new .schema.json file and upload it to the sf-metadata-<installationId> bucket under the schemas folder. It is important that it is named <SalesforceObjectName>.<installationId>.schema.json with SalesforceObjectName being the name of the object (such as Account, or Opportunity, etc.) and installationId being the installation ID for the system. Other files will be similarly named. Removing an Object is done by deleting the file from the S3 bucket. To add a file, fill out the JSON file with the following structure:

```
{
  "name": "NameOfObjectInSalesforce",
  "installationId": "abcd1234",
  "type": "object",
  "label": "Label of Object in Salesforce",
  "exclude": {
    "FieldName": {
      "type": "string",
      "label": "Field Name",
      "$comment": "Field Name"
    },
    // ... Any other fields you do not want
  },
  "properties": {
    "ImportedField": {
      "type": "string",
      "label": "Imported Field",
      "$comment": "Imported Field"
    },
    // ... All remaining fields you do want imported
  }
}
```

It is important to note that any fields missing from the properties section which are not explicitly part of the exclude will be automatically added to the properties section. Therefore, to add all fields except the ones in the exclude section, just leave the properties section blank like: "properties": {} and the system will fill in the missing properties. To upload the file, see the section below on uploading schema files.

The next time the step function runs to sync the data, your changes will be applied. To do this manually, see the section below on running the sync manually.

5.1.12.3 Uploading the Schema File

Visit the AWS S3 console and find the sf-metadata-<installationId> bucket and click on it. Go into the schemas folder and click Upload. Click Add files to add the file you wish to upload. Below, expand the Properties section by clicking on it. You will have to specify to override to use the encryption key for that bucket, search for the key named metadata for your installation ID (see screenshot below). Click Upload and wait for the upload to be completed.

Server-side encryption settings
Server-side encryption protects data at rest. [Learn more](#)

Server-side encryption

- ☐ Do not specify an encryption key
- ☒ Specify an encryption key

Encryption settings

- ☐ Use default encryption bucket settings
- ☒ Override default encryption bucket settings

Encryption key type
To upload an object with a customer-provided encryption key (SSE-C), use the AWS CLI, AWS SDK, or Amazon S3 REST API.

- ☐ Amazon S3-managed keys (SSE-S3)
An encryption key that Amazon S3 creates, manages, and uses for you. [Learn more](#)
- ☒ AWS Key Management Service key (SSE-KMS)
An encryption key protected by AWS Key Management Service (AWS KMS). [Learn more](#)

AWS KMS key

- ☐ AWS managed key (aws/s3)

Search: meta

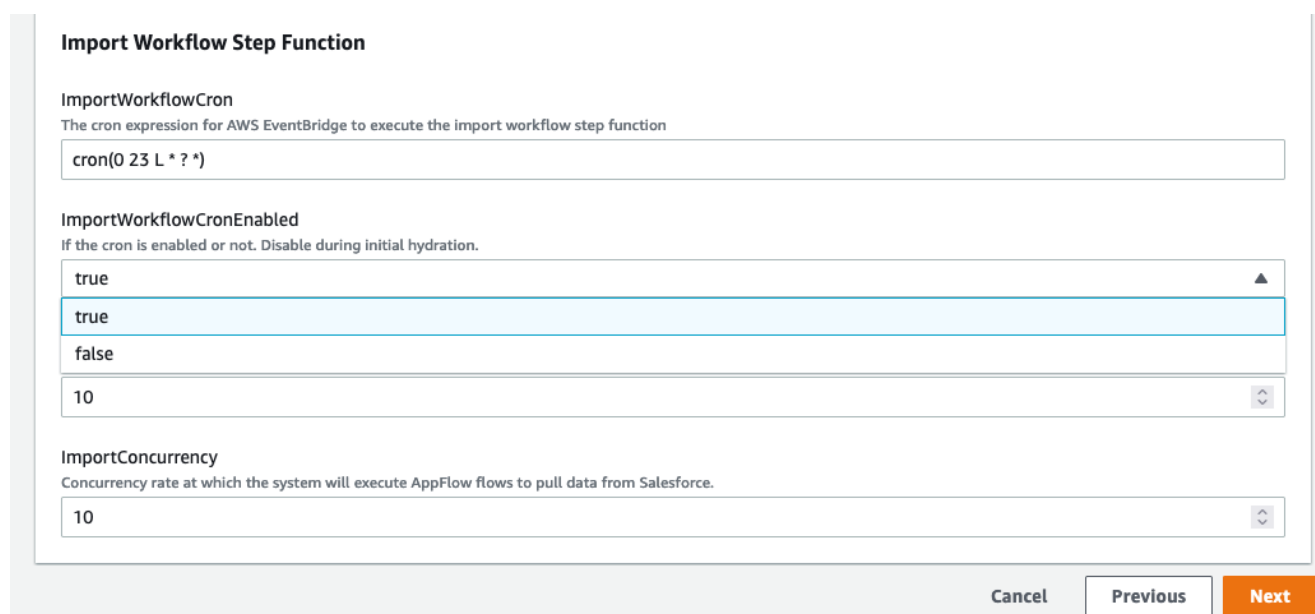
arn:aws:kms:us-west-1:699108794281:key/8e5a9355-544f-471b-b591-b113ea53f5df:sf-meta-data-dhlumlofqd3c

[Choose AWS KMS key](#) [Create key](#)

5.1.12.4 Manually Applying the Changes

First, the automated step function execution must be disabled so that two step functions are not running simultaneously. Visit the AWS CloudFormation console and click on the stack named `sforg-stepfn-<installationId>`. Next click the Update button. Click Next to use the current template.

At the bottom, find the parameter named `ImportWorkflowCronEnabled` and change the value in the dropdown to false (see screenshot below). Click Next two times, check the box that says "I acknowledge ..." and click Update stack. Once completed and the stack status is `UPDATE_COMPLETE`, move to the next step.



Import Workflow Step Function

ImportWorkflowCron
The cron expression for AWS EventBridge to execute the import workflow step function

`cron(0 23 L * ? *)`

ImportWorkflowCronEnabled
If the cron is enabled or not. Disable during initial hydration.

true
true
false
10

ImportConcurrency
Concurrency rate at which the system will execute AppFlow flows to pull data from Salesforce.

10

Cancel Previous Next

With the step function disabled, visit the AWS Step Functions console and click on the step function named `sf_import-<installationId>`. Ensure the list of "Executions" does not have a status of Running anywhere on the page. Next, click the Start execution button and when the dialog appears click Start execution again (no input is necessary).

When the step function completes, the sync is done. Remember to enable the automated step function execution by going back to CloudFormation and reversing the above steps to change `ImportWorkflowCronEnabled` back to true so that the step function will still run on its regularly scheduled time.

5.1.13 Testing

This section describes the testing description, the testing that has been done and recommendations of further testing needed for the app.

Salesforce limits during Testing

During testing, certain limits may affect the EZ Datalake app. One such limit is the refresh token limit per connected app. This results in AppFlow connectors to Salesforce needing to be refreshed/reconnected which may affect periodic data imports. In real use cases, this does not affect the NPO User since they are not expected to use more than 5 connectors. However, this limit could be exceeded during testing. More details on this can be read here: <https://developer.salesforce.com/forums/?id=9060G000000UUjlQAG>

This section describes the testing description, the testing that has been done and recommendations of further testing needed for the app.

Type	Subtype	Description	Details	Test Coverage	Execution Method	EE Recommendations
Functional Testing	End to end testing	End-to-end testing is a Software testing methodology to test EZ Datalake from start to end, going through all 6 steps	Regions	US East (Ohio) Region US East (Northern Virginia) Region US West (Northern California) Region US West (Oregon) Region	Manual	We recommend testing in the regions being desired by the customer for the beta rollout and continued testing in any new regions added to the EZ Datalake app
			NPO Organization structure and data size	Demo Organization - Sandbox Clone and Production	Manual	Testing has been done only in the demo organization that was setup for Envision Engineering. We recommend creation of demo organizations with different structures and sizes of data that are representative of various NPOs to test out the solution
			Private Connect	Not covered		EE has not tested EZ Datalake setup with private connect. This is a feature that was dropped based on discussion with Salesforce on and was not factored into design
			Tableau Connection	Tableau Desktop Mac OS Tableau Desktop Windows	- - Manual	We recommend testing of Tableau connection on all desired Operating Systems (OS). We also recommend testing of non-desktop versions of Tableau. Note that instructions to the user on connecting to Athena may need to be updated based on above testing on various OS and Tableau products

			Sync Period Options	Daily & Weekly sync	Manual	EE has only done limited testing of the monthly sync option due to the nature of the short-lived engagement. We recommend testing all sync period options: monthly, weekly and daily
			Deletion Scenarios	Daily & Weekly sync (Monthly sync have not been tested – as timelines were out of PoC timeline)	Manual	Salesforce has listed out all deletion test scenarios here . We recommend continued testing of all these scenarios to be carried out especially for monthly tests. EE has shared test results in an accompanying document as: EZDatalake_Testing_Deletion_Scenarios.pdf
			Combinations of inputs	Not covered		EE has not tested all combinations of user inputs/options and/or features provided by the EZ Datalake app. We recommend automated testing tools as suggested in UI Testing to get to desired test coverage here
	UI Testing	To validate whether the visual elements (buttons, links, text fields) are displayed correctly and work as intended on different browsers and OS		Firefox Safari Chrome	Manual	We recommend testing on all desired browsers and OSs. A suggested set is: Chrome, Firefox and IE. We also recommend UI test automation with tools such as: https://www.cypress.io/ or https://playwright.dev/
	Unit Testing			lambdas folder used for step functions and import are covered. The cloud formation custom resource lambdas not covered are not covered. The UI also has no coverage	Manual	We recommend that unit tests be reviewed and enhanced to have more cases checked to increase coverage. We also recommend unit tests for UI to be added through automated tools like https://www.cypress.io/ or https://playwright.dev/
	Integration Testing	Testing where software modules are integrated logically and		Not covered		None for API. We recommend UI end-to-end integration testing to be automated with tools such

		tested as a group. The purpose of integration testing is to expose defects in the interaction between software modules when they are integrated				as https://www.cypress.io/ or https://playwright.dev/
	System Testing			CFN Nag + npm_audit	Automated	
Performance Testing			Data Size	Not covered		EE recommend testing with NPO demo organizations with millions of records for performance load testing. The only testing done was loading a single Salesforce Object which had > 1M records to ensure it worked.
Security Testing		To maintain security of the app (proof-of-concept), identify threats, uncover vulnerabilities and risks	Internal	Prototype security testing		EE goes through internal /proof-of-concept/prototype security approval process. We recommend going through additional Salesforce's testing process prior to production release

5.1.14 Miscellaneous

Perspective	Category	Recommendation
Security	Data Protection	Review existing and utilize additional appropriate safeguards to protect data in transit and at rest. Safeguards include fine-grained access controls to objects, creating and controlling the encryption keys used to encrypt your data
Security	Incident Response	Review current incident response processes and determine if and how automated response and recovery will become operational and managed for AWS assets
Security	Incident Response	Define and document relevant break-glass procedures
Platform	Automate release management	Introduce application lifecycle and route to production by using several environments
People	Training to SI partners	People training to support the NPO Users and evolve the solution on the infrastructure and application level
Operations	Monitoring	Customer should include the components deployed by the proof of concept in their own monitoring tools
Operations	Logging & Alerting	The proof of concept has been configured to output all logs to CloudWatch. This log data should be integrated into any existing logging systems

Operations	Logging	Ensure app-level logging is sufficient for audit purposes and augment as needed
Operations	Release and Change Management processes	Create capability to manage, plan, and schedule changes to the solution, if needed

5.2 Skills Required

In order to take the Envision Engineering Proof of Concept technical architecture to production, Salesforce.org and/or their selected partners will need to be proficient in each of the following technologies to support the solution:

- [Typescript](#) - Both front-end and back-end
- [React](#) and [MobX](#) are used on the front-end
- AWS CloudFormation - YAML templates are used
- AWS Lambda & AWS Step Functions understanding are core to the back-end

5.3 Software Development Languages and Frameworks

- [Typescript](#) - Both front-end and back-end
- [React](#) and [MobX](#) are used on the front-end
- AWS CloudFormation

5.4 AWS Services

- AWS KMS - KMS is used to encrypt traffic in flight and data at rest. From S3 buckets to RDS storage, AppFlow in flight data and everything else in the system, a customer managed KMS key is created to encrypt and decrypt the data.
- AWS S3 - S3 is used for two purposes: 1) System configuration & asset storage, which uses very little space (costing less than \$0.01 per month) and is permanent. 2) Temporary data storage during imports and for the results of querying from Athena, which is large in size (size of Salesforce data for 24 hours after import + size of all query output for the last 30 days) and is automatically cleaned up after a short period of time.
- AWS IAM - IAM controls access to all resources such that all parts of the system adhere to the principle of least privilege.
- AWS RDS - Aurora Serverless Postgres database. Data is stored here and executes queries initiated by AWS Athena.

- AWS AppFlow - Performs Salesforce data extraction and writes the data to S3 to be imported into RDS.
- AWS EC2 (for VPC and networking) - VPC, Security Groups, Route tables, etc. which are necessary to support RDS.
- AWS Lambda - Application logic for step functions to import the data.
- AWS Step Functions - Primary import logic controller. Executes AWS SDK calls or invokes AWS Lambda to perform the import into RDS.
- AWS Secrets Manager - Safe storage of the root RDS credentials as well as Salesforce access and refresh tokens used by AppFlow. Note that with Appflow, AWS Secrets Manager encrypts with an AWS managed key. More details can be found here: <https://docs.aws.amazon.com/appflow/latest/userguide/data-protection.html#encryption-rest>
- AWS Athena - Set up a catalog to allow querying the RDS instance.
- AWS SNS - Creates a topic which the system can publish messages to. Published messages are sent as emails to specific users of the system to alert them of potential issues and maintenance.
- AWS CloudWatch - Stores logs for AWS Lambda and has system alarms. Additionally, schedules the execution of the AWS Step Function to run when it is configured to run.
- AWS SQS - As data is ready to be imported to RDS, it is queued in SQS. The AWS Lambda responsible for imports will pull from this queue to know what S3 files still need to be imported into RDS.
- AWS CloudFormation - Used to create all the resources and infrastructure the system needs to operate.

AWS offers a range of [free digital training courses](#) as well as [instructor-led classroom training](#) to meet the needs of any skill level. These courses may help Salesforce.org or potential partners to gaining additional knowledge on the AWS platform in general and learn details about specific services relevant for the POC.

See <https://aws.amazon.com/training/> for details.

6 Deployment Guide

You will find the entire codebase in an accompanying ZIP file. Please consult the **README.md** file that is located in the root of the accompanying ZIP file.

7 User Guide

The User Guide shows how to upload the assets and deploy the AWS Amplify app to deploy AWS resources for the data lake and import the data.

8 Lessons Learned

We learned a number of lessons building this Proof of Concept. We have documented them for the team taking the Proof of Concept to Production to consider.

Area	Lesson
Compound Fields	AppFlow and Salesforce don't quite work perfectly together yet. Salesforce does not allow fetching of Compound fields from the bulk API but requires use of the bulk API for large tables (with over 1 million rows). AppFlow accepts new fields that are Compound fields in Salesforce regardless if the bulk API will be used or not. This leads to the potential where a Compound field is requested by AppFlow for an object which requires using the bulk API, leading to an AppFlow failure that will not succeed on retry.
AppFlow & CDC	<p>AppFlow doesn't handle CDC (Change Data Capture) as a single stream on a single source of data. In order to perform CDC on a data set, a manual execution of AppFlow to get all data followed by setting up the corresponding Object's events type in AppFlow to receive deltas on changes to the data. After that the changes must be applied manually to the data set, through a custom Lambda or Glue job, to keep a single source of data for the system. Doing a new manual execution will result in a new source of data and using AppFlow's incremental flow results in rows being duplicated in the data set (each change will create a new row with the same data instead of performing an "upsert" operation).</p> <p>The approach of using an Object's events was not used because it doesn't ensure that the schema and hard deleted records are captured as events may be missed or fail. Secondly, we found no easy way to tie Salesforce Objects to Salesforce Events and we were unsure if all objects are ensured to have an events notification type. This can however be used, combined with the full import in this proof of concept, to create a hybrid system that has very low latency (hour or minute) with guaranteed schema and hard deleted row accuracy within a reasonable amount of time (whatever the full sync period is, weekly for example).</p>
Advanced Schema Changes	Salesforce has rather flexible schema change capabilities. Because of the potential amount of time between AppFlow executions, it is possible one or more schema changes have occurred which AppFlow would need to handle when updating a data set. However, the information given by AppFlow, to a custom AppFlow connector, is insufficient in determining how to perform destructive actions. The information given, name of column and type in the schema, can make things like a column rename (where the data is to be kept) and drop of one column and addition of another of the same type indistinguishable from each other.

<p>Discussed Alternative AWS Services & A True Data Lake</p>	<p>AWS Lake Formation is not yet usable for mutable data stores such as the Salesforce data. This is because it only allows updates at the file level on S3 (of which may contain multiple records inside which may not have changed or may be changing concurrently with another change). In order to perform “upserts” to an object storage system such as S3, tools like Apache Hudi can be used, but this requires some work to get everything setup, configured and connected together so is not quite an off the shelf software system like AWS services.</p> <p>However, due to the nature of Salesforce data being normalized and using Id primary keys with reference fields pointing to other tables rather than a partitioning scheme which groups records in a table by one or more fields, this makes the query performance suffer when scaling to larger data sets. This is because the system has no way to know which file on S3 contains the Id which is requested in the query, so it must check all files (aka a full table scan). Standard databases, such as provided by RDS, allow for indexing tables with binary trees to ensure a certain level of performance when fetching by the Id or joining two or more tables together. As such, the system moved away from a traditional data lake type system and uses a legacy database system for storage and query computation. There is a limit to RDS performance which a traditional data lake would surpass but at that level the data would need to be reorganized into partitions (likely with multiple tables becoming one table with several partitions) and would have to be organized according to how the data was intended to be queried. At this scale, an engineer is required to create this reorganization logic and all supporting infrastructure and is therefore out of scope of this system.</p>
--	---