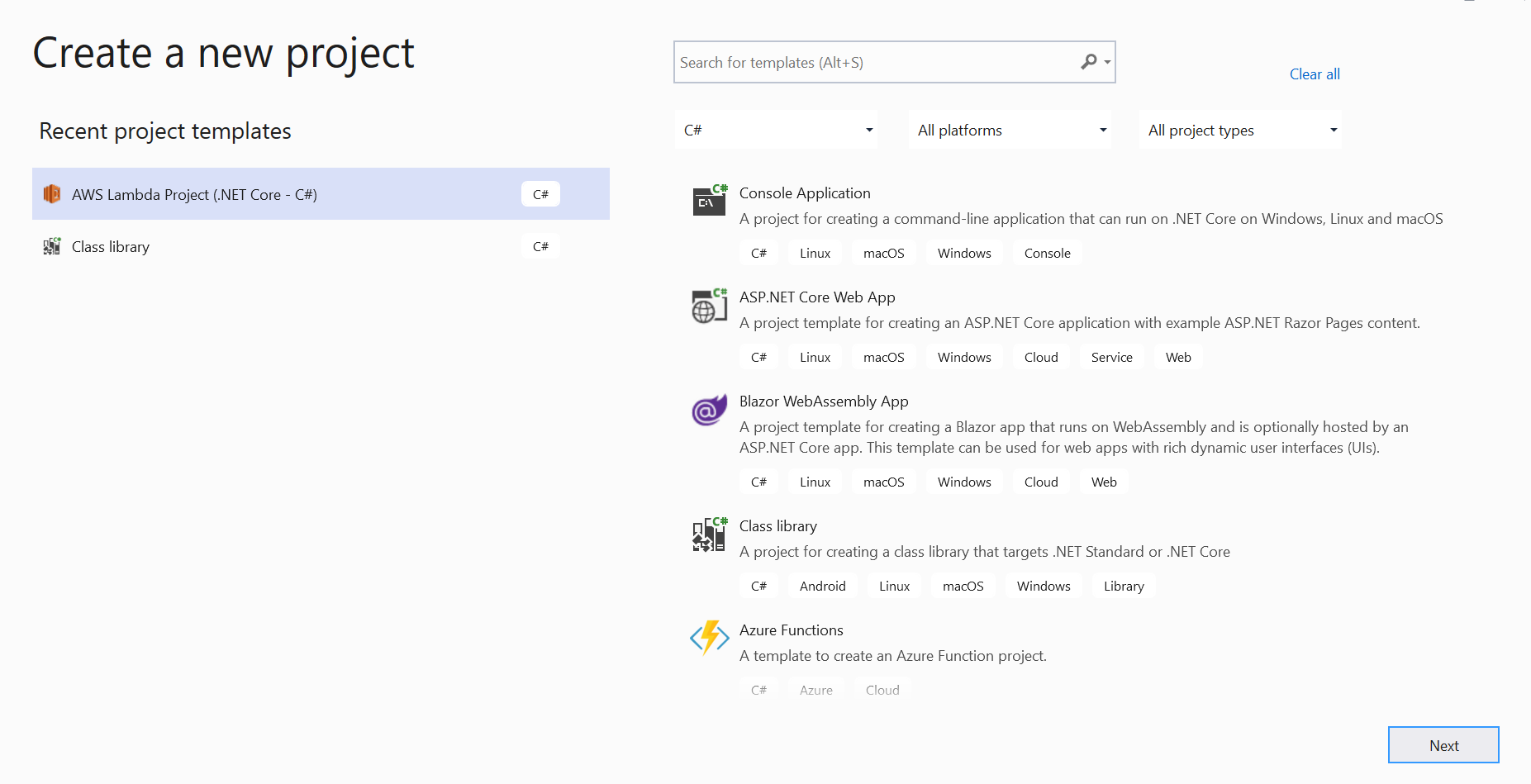
**AWS Lambda deployment Process**

* **This Document gives detailed overview of Creating a serverless lambda application and procedure followed to host the lambda project to AWS through terraform, also gives insights on AWS Step Functions.**

1. **Creating and setting up lambda project**

Choose AWS Lambda Project (.NET Core – C# ) from the templates

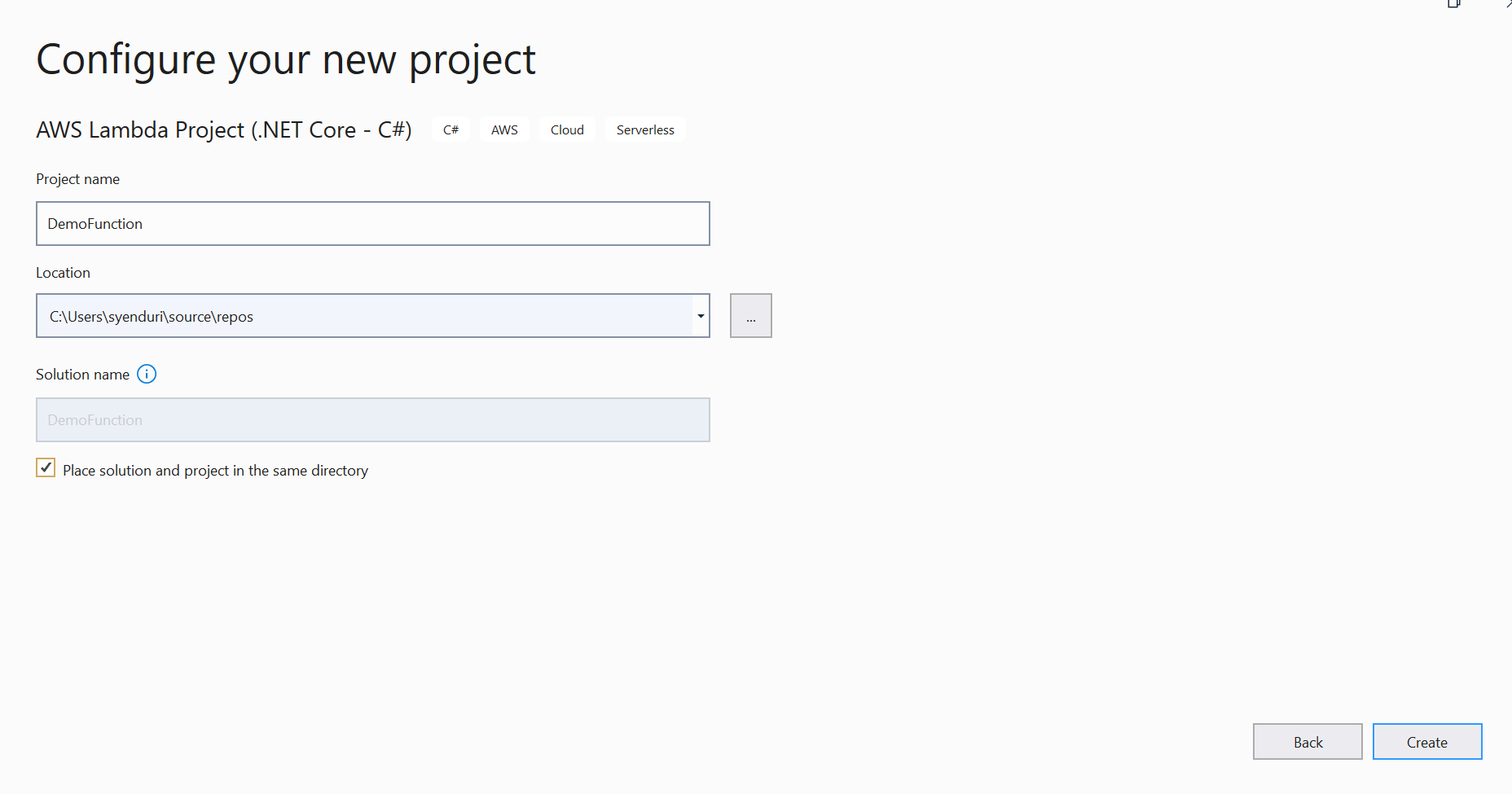


Define solution name and project name

* If the checkbox is not checked, the solution file (.sln) will stay inside solution folder and project folder level.



* If the checkbox is checked, the solution file (.sln) will stay inside project folder level.



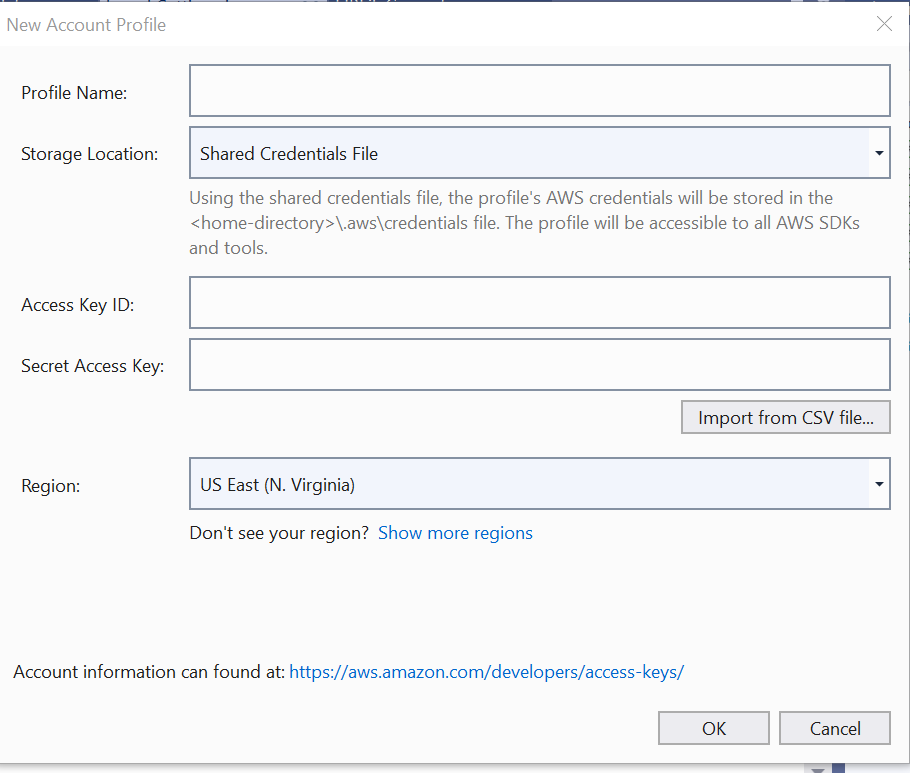
* After creating lambda project , we can see the files ( launchSettings.json, aws-lambda-tools-defaults.json)
* The **launchSettings. json** file is used to store the configuration information, which describes how to start the ASP.NET Core application, using Visual Studio. The file is used only during the development of the application using Visual Studio. It contains only those settings that required to run the application. We can define our **environment variables** inside this file.
* 
* **aws-lambda-tools-defaults.json** file provides default values for the deployment wizard inside Visual Studio and the AWS Lambda commands added to the .NET Core CLI.

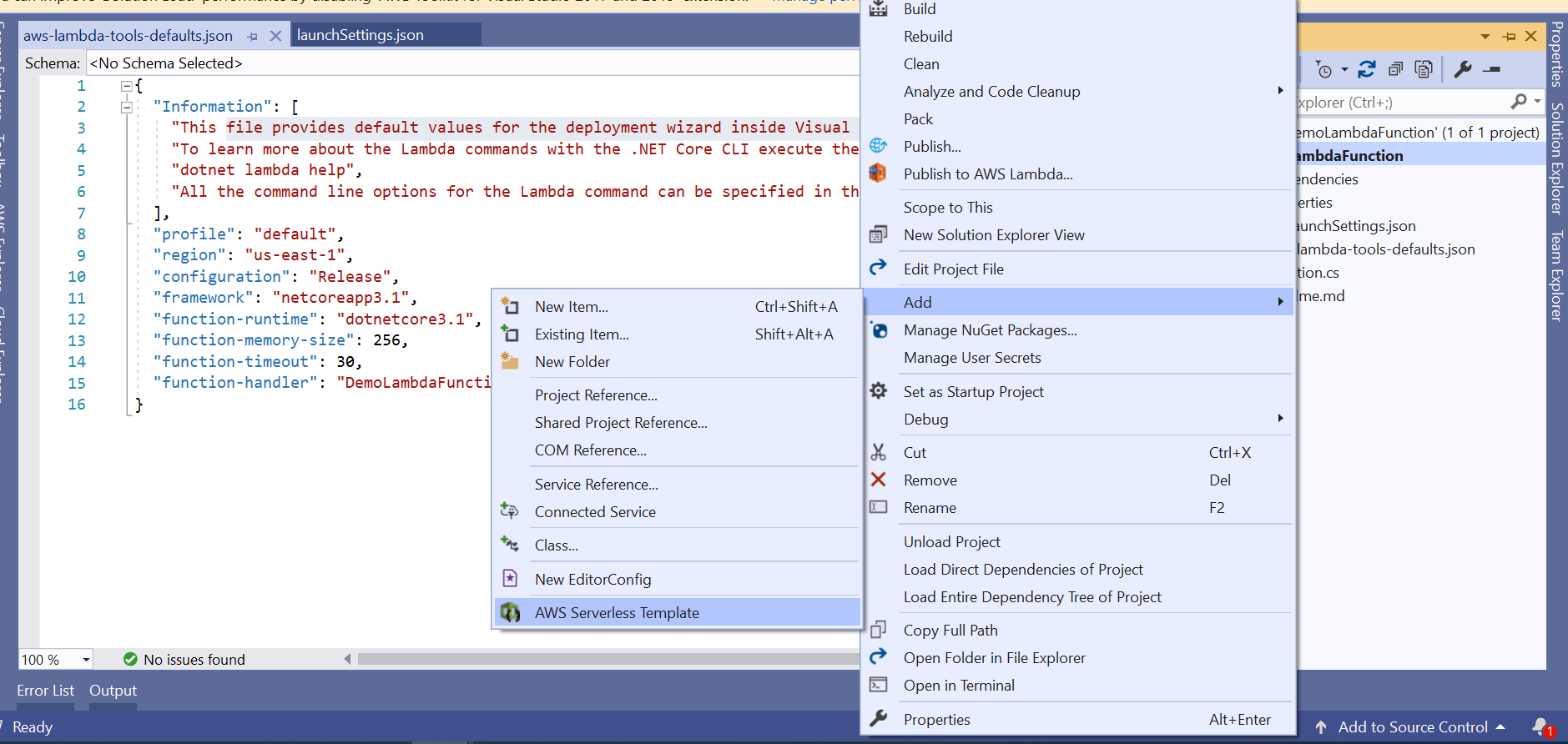
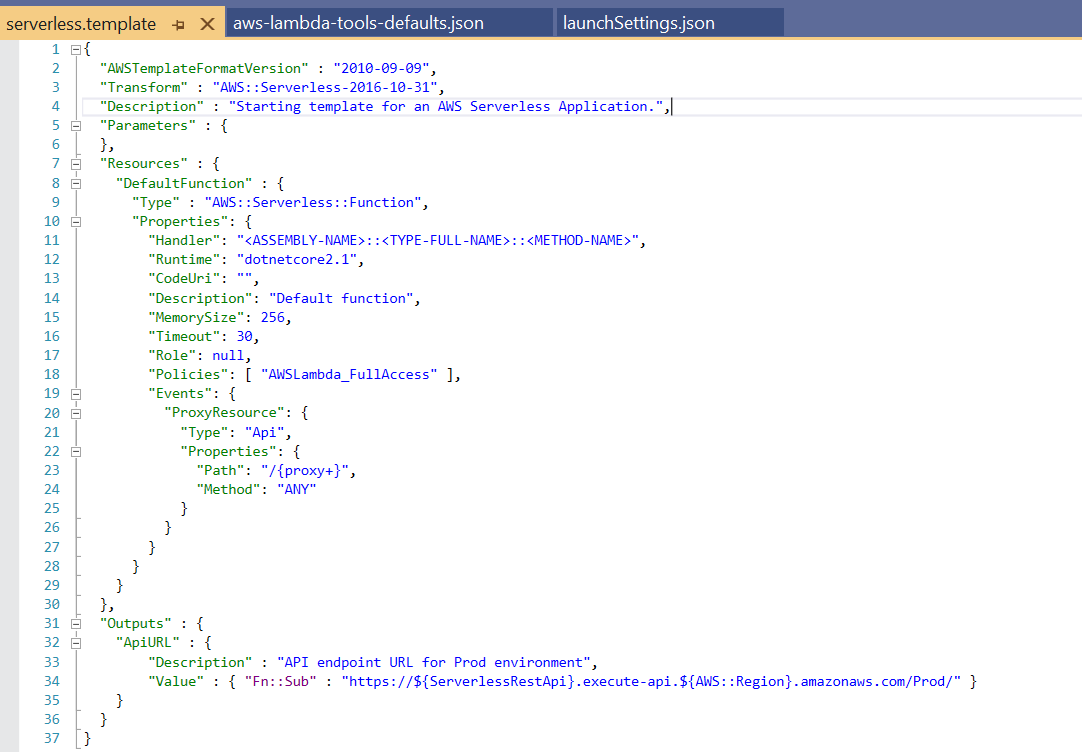
**After Creating lambda and implementing our functionality, we need to publish it to AWS.**

1. **Deployment process**

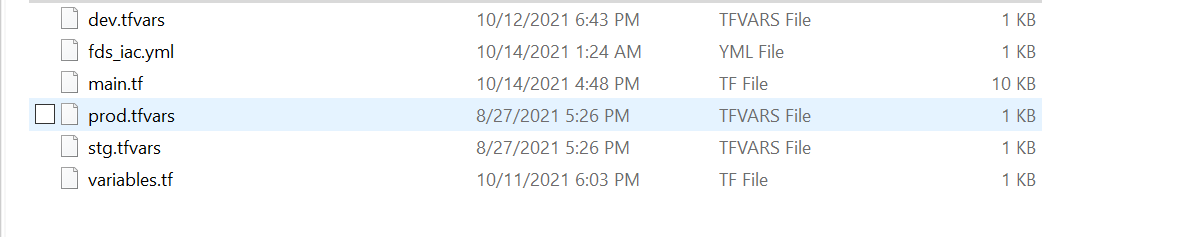
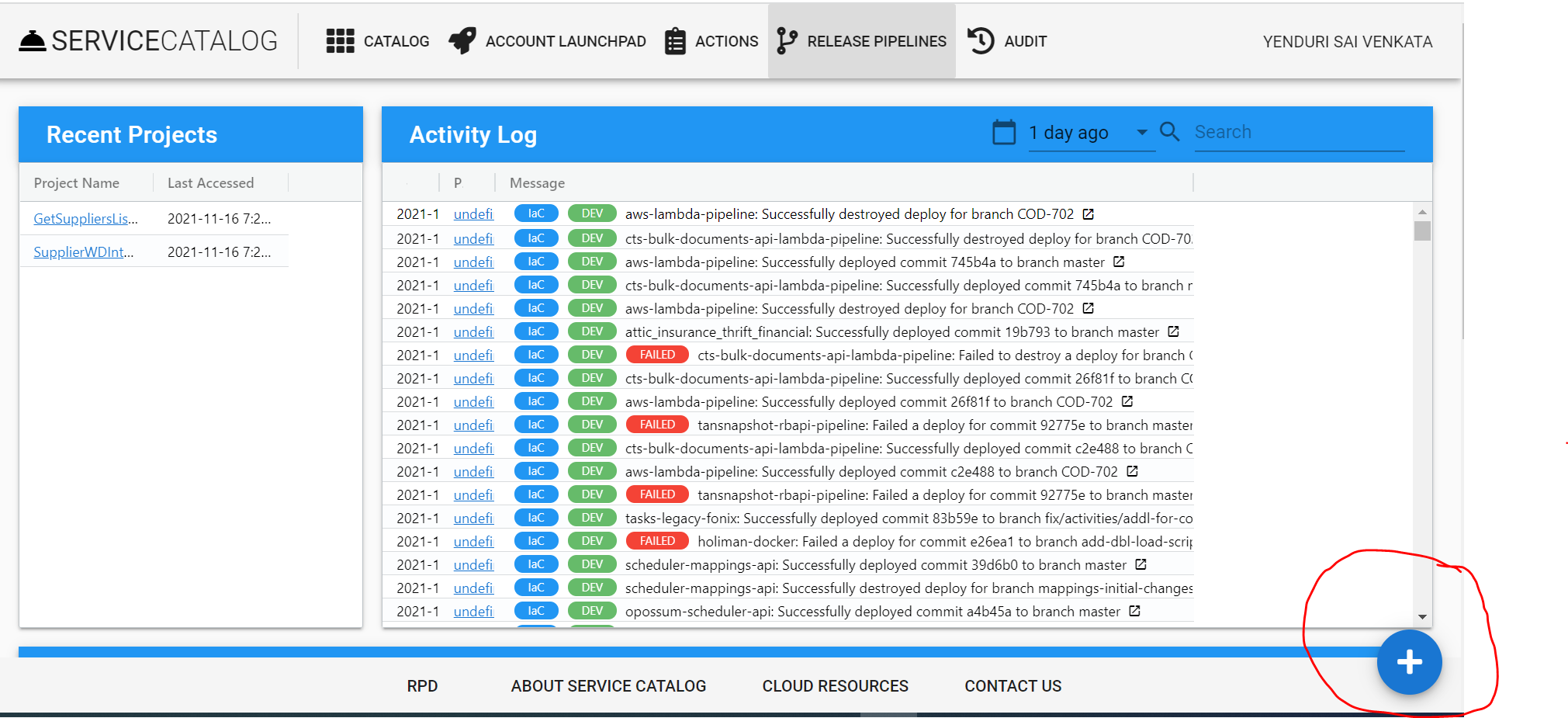
This can be done in 2 ways.

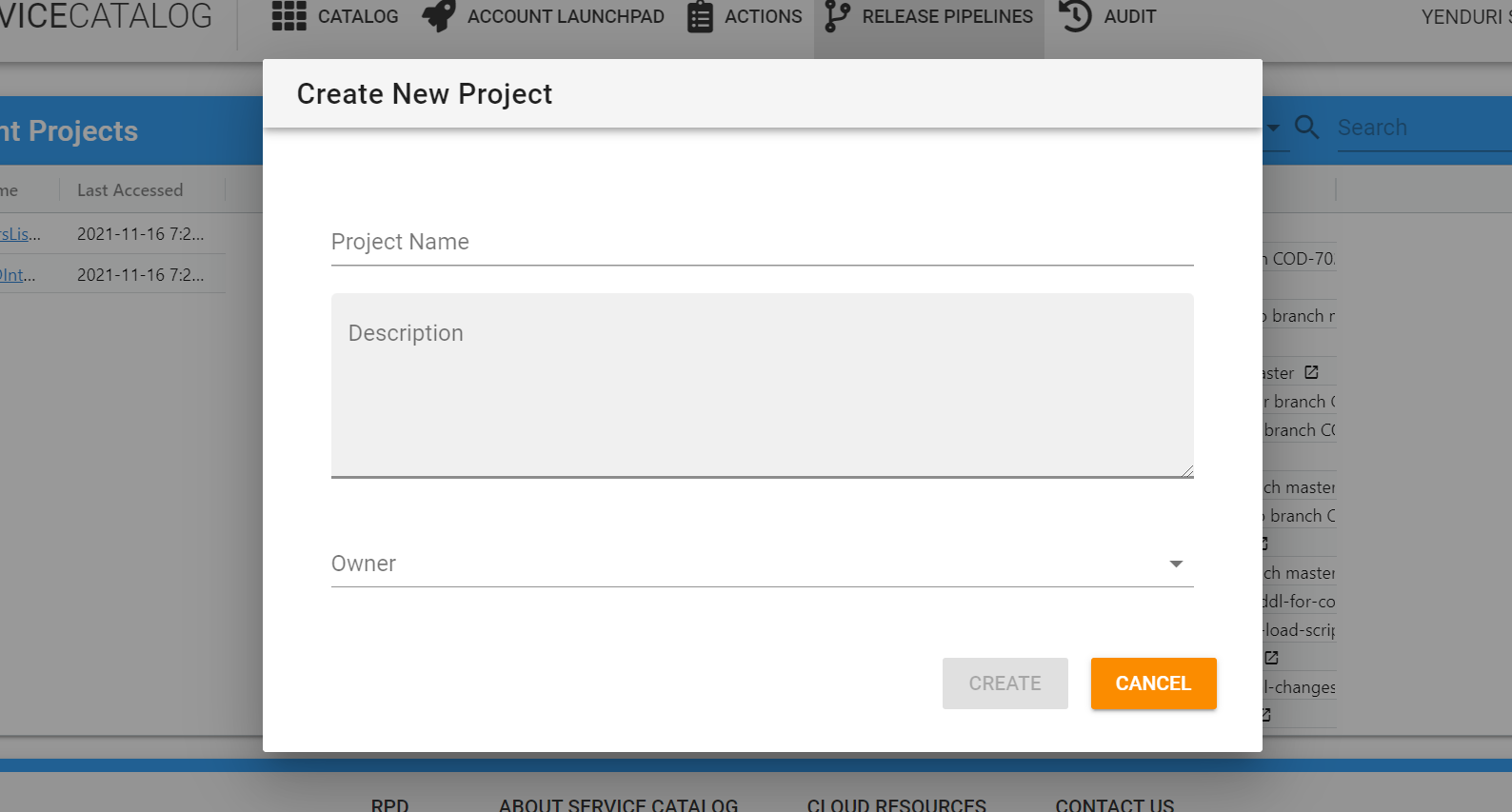
1. **Publishing directing from AWS Explorer**
2. **Deploying the resources through terraform**
3. **In the first way**, we need to connect our Visual Studio to AWS Explorer which will be visible after installing AWS SDK. We need to create an AWS profile using Credentials(Access and Secret keys)

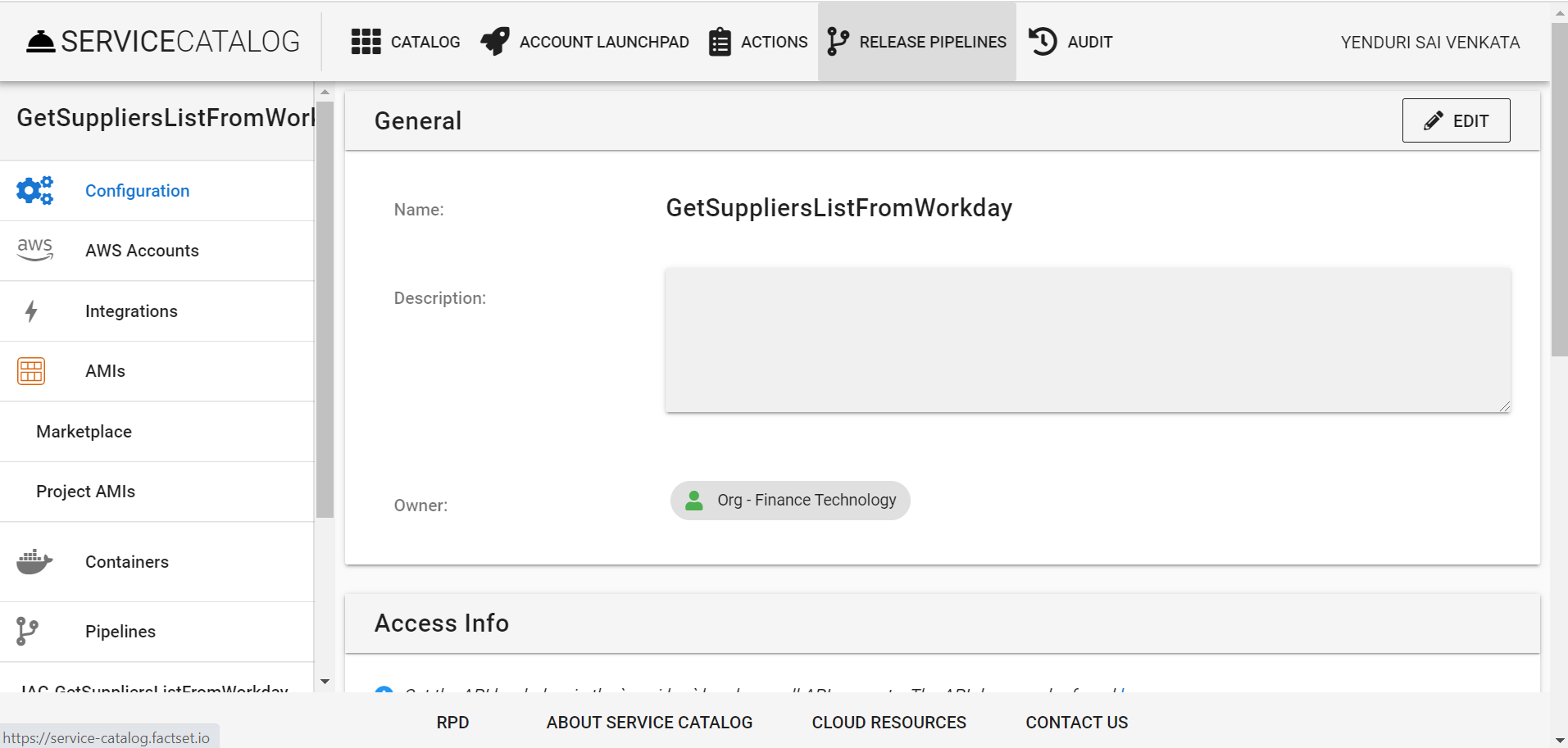
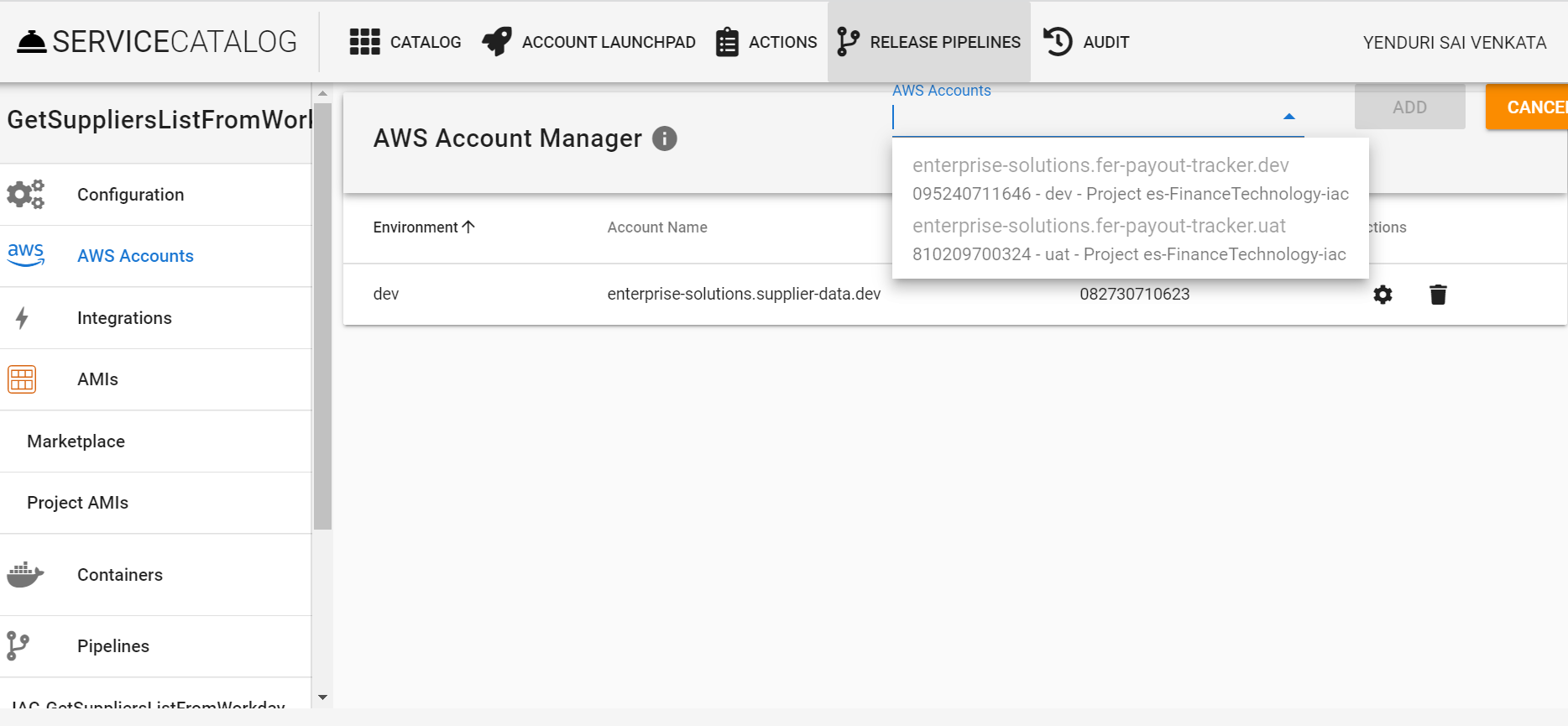
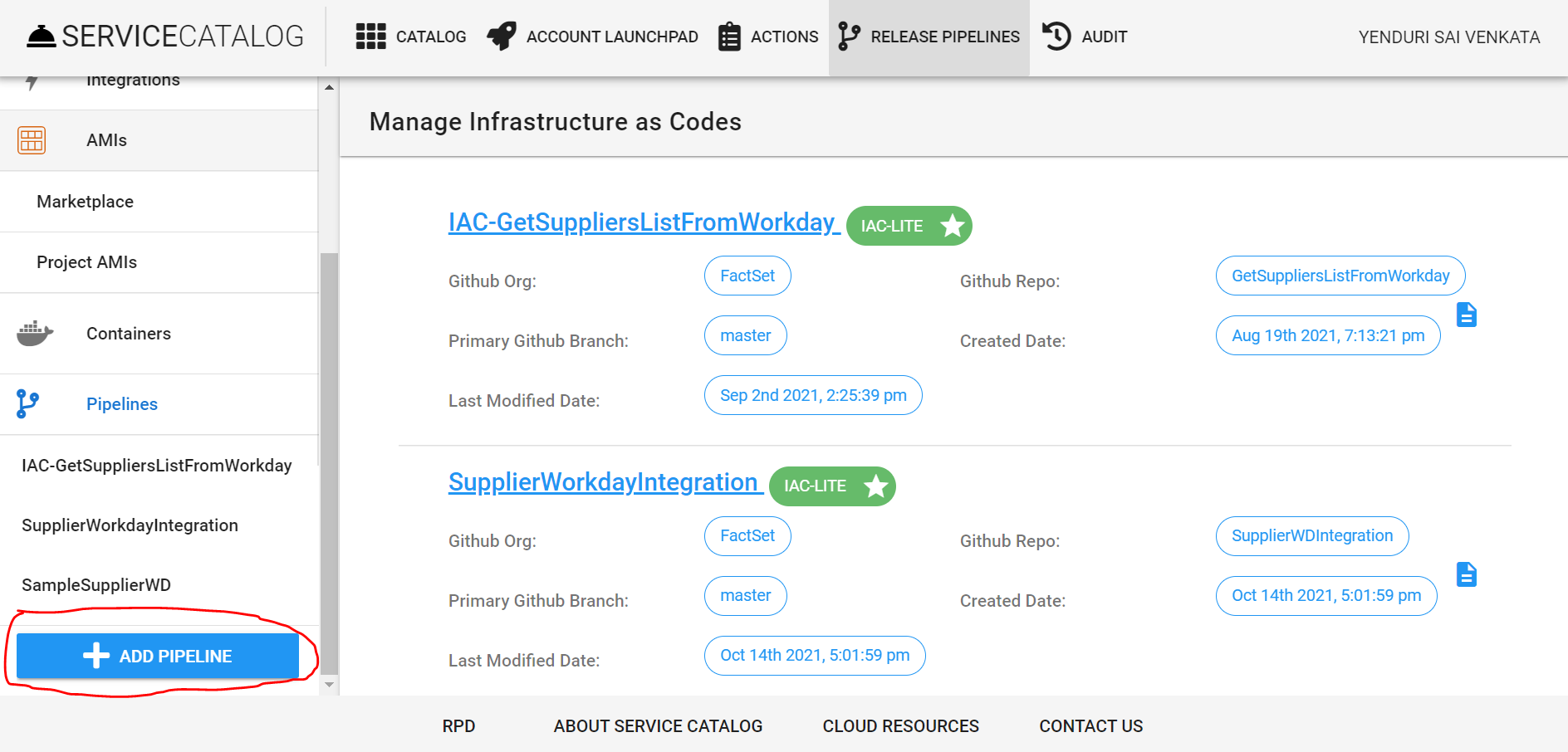
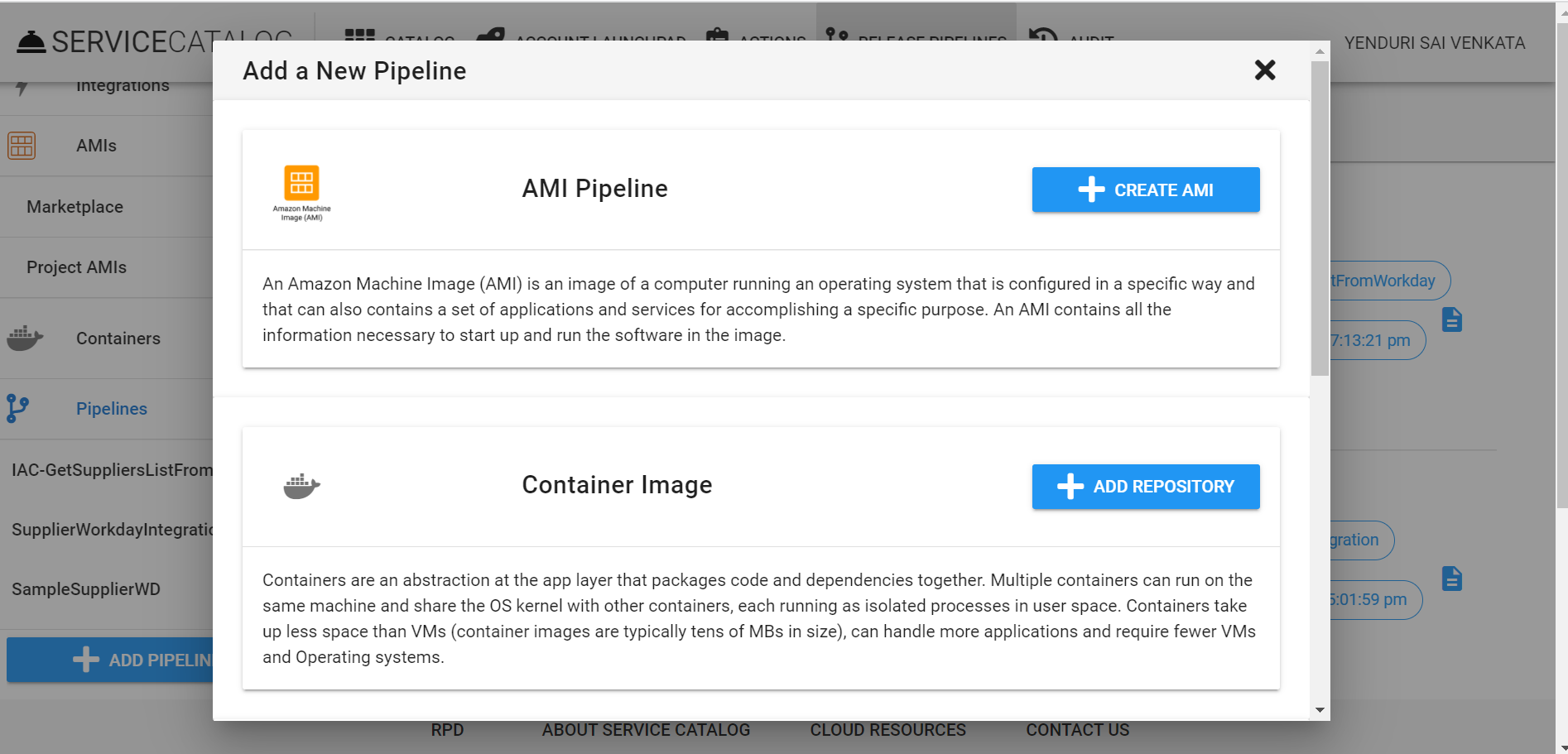
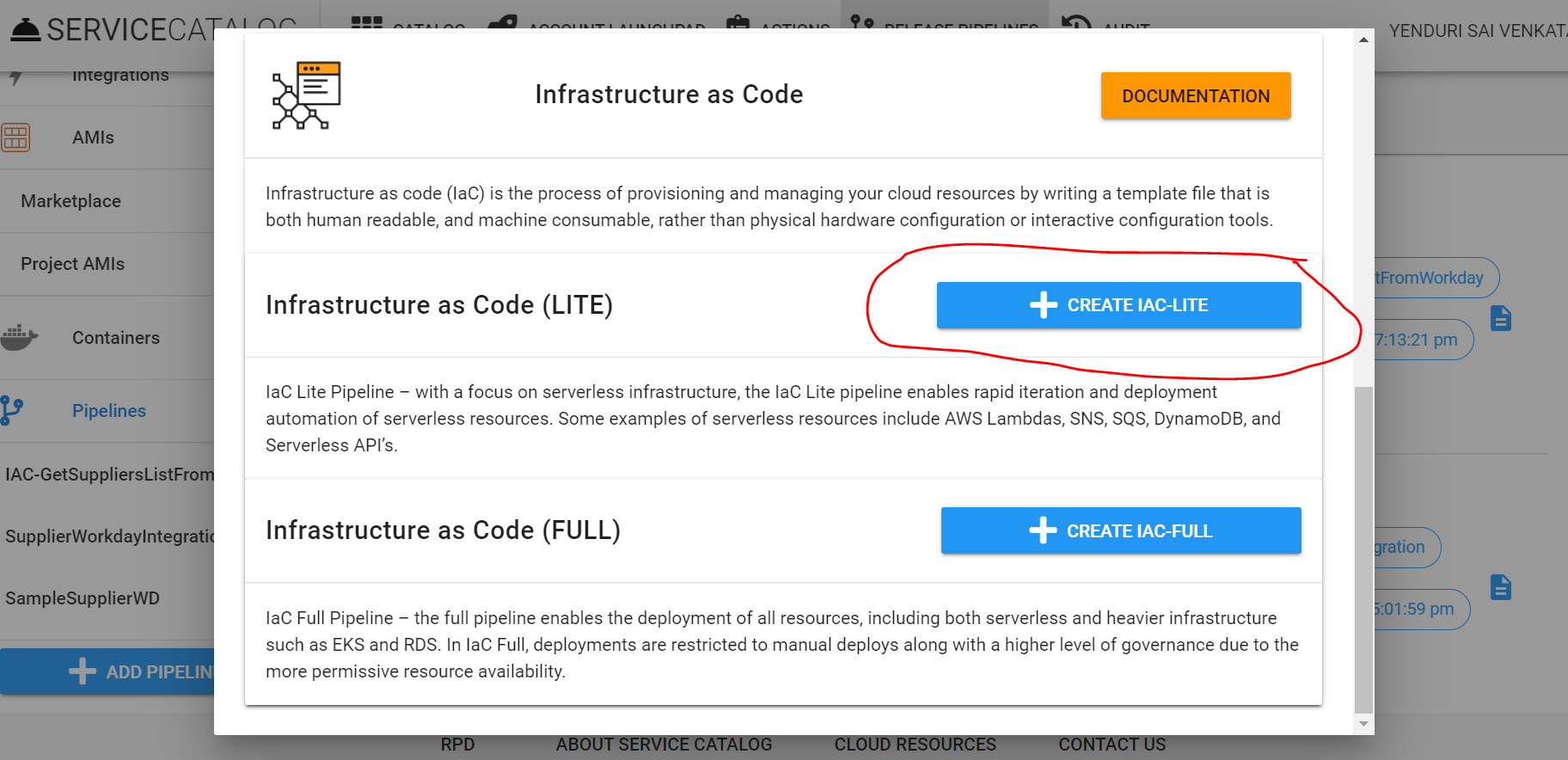
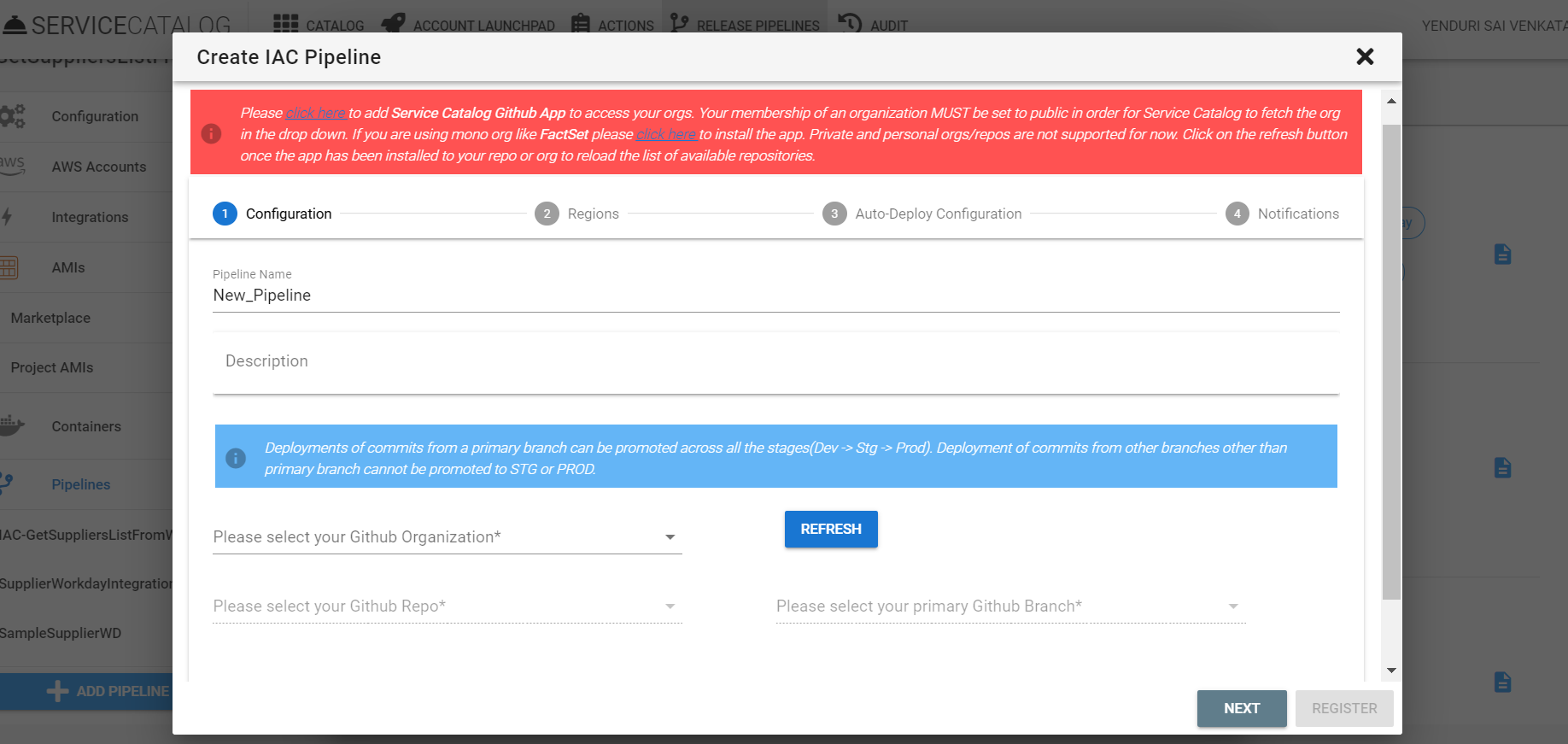
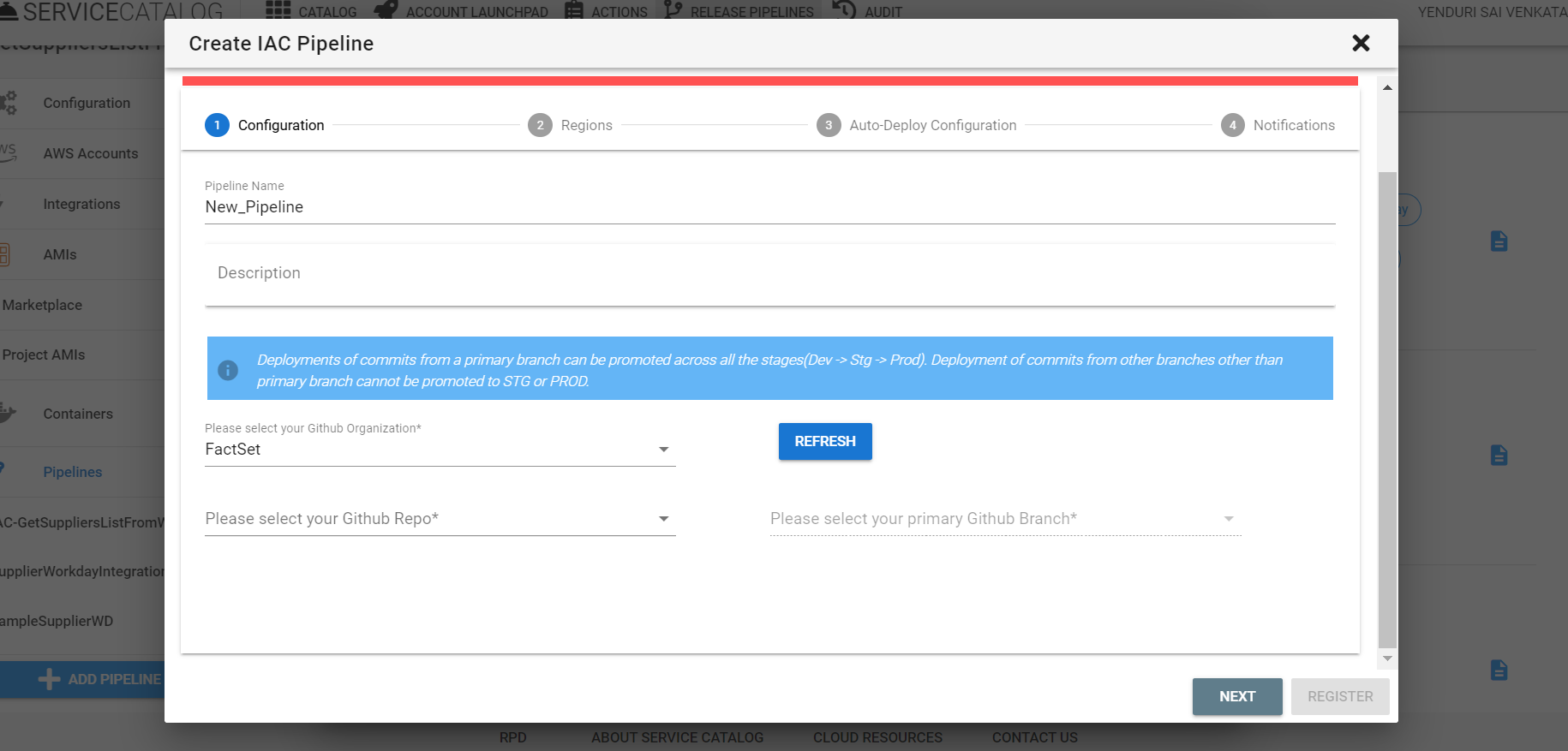
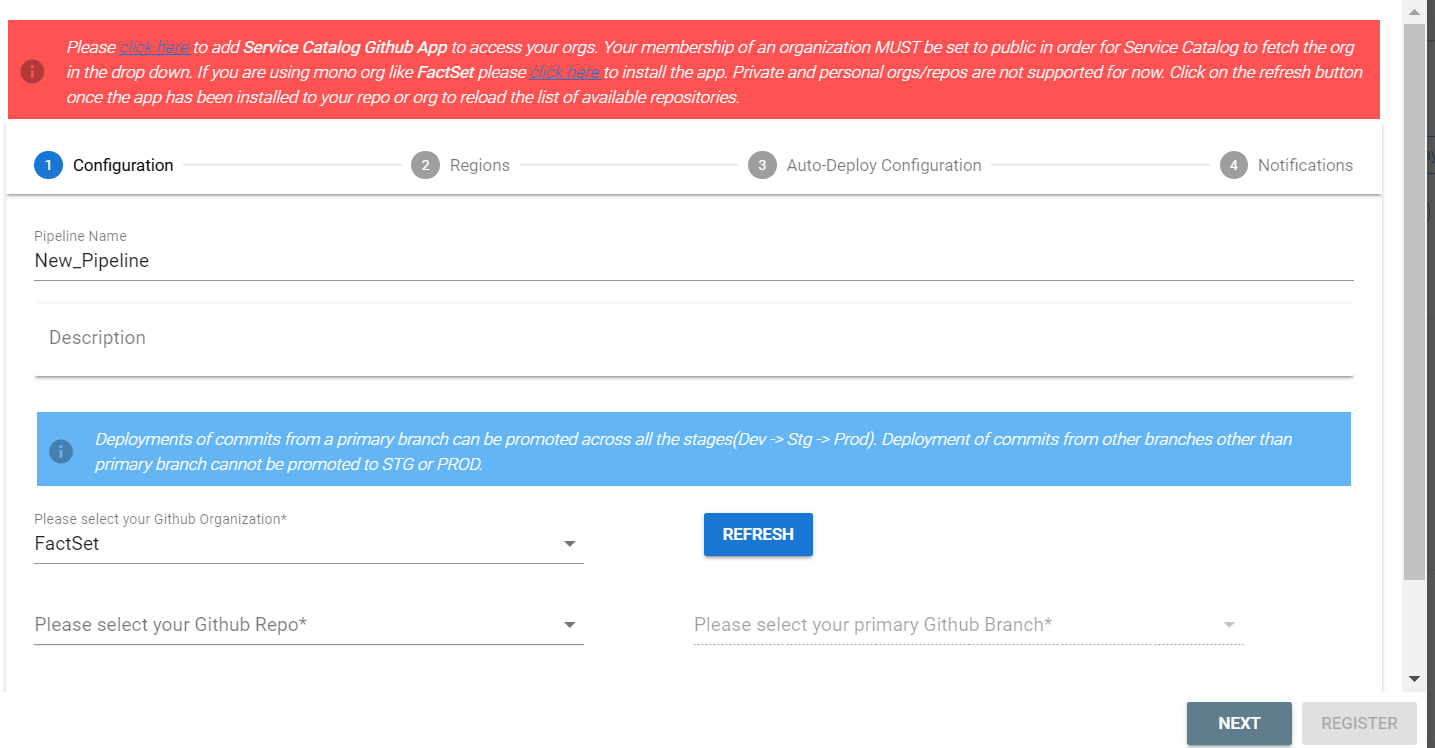
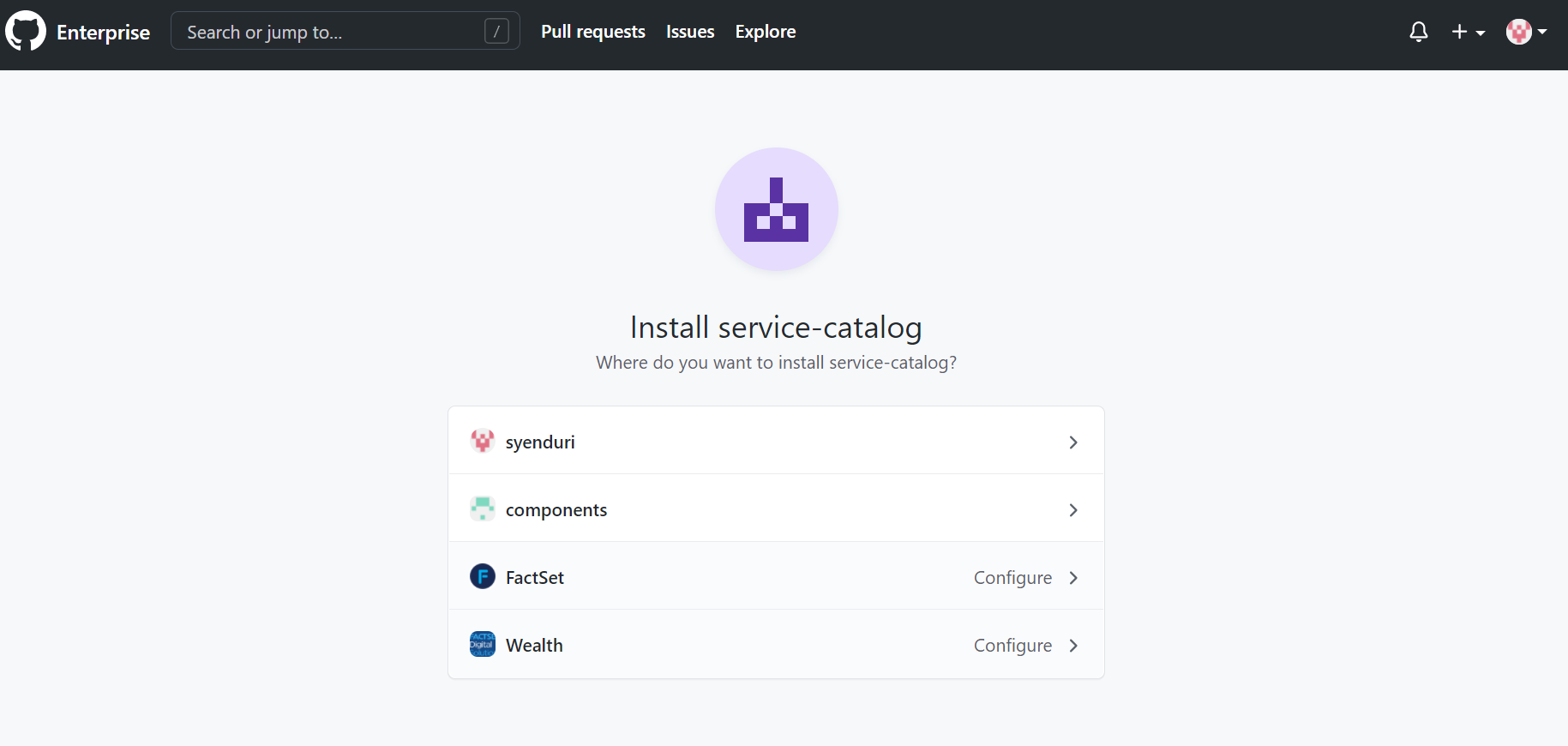
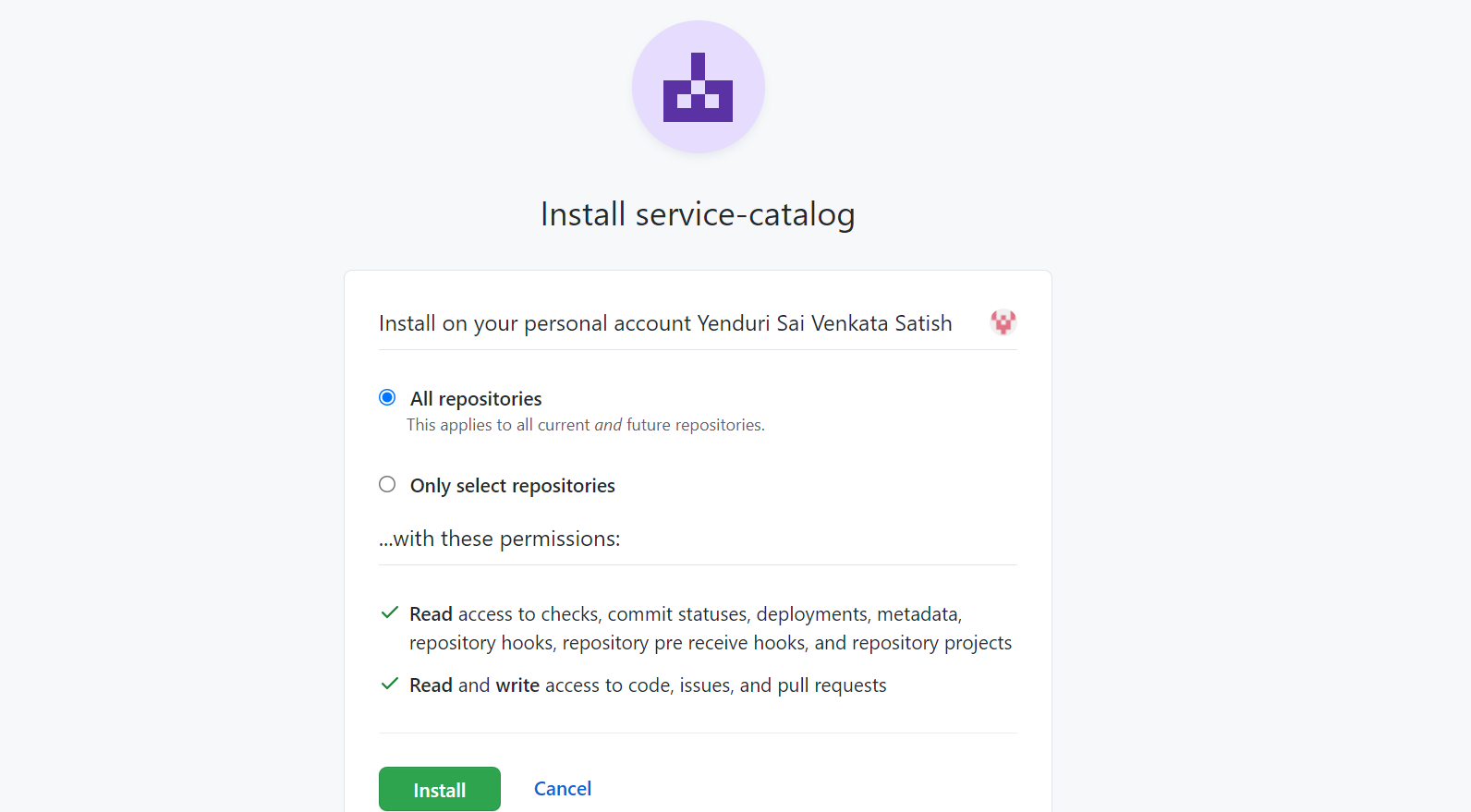
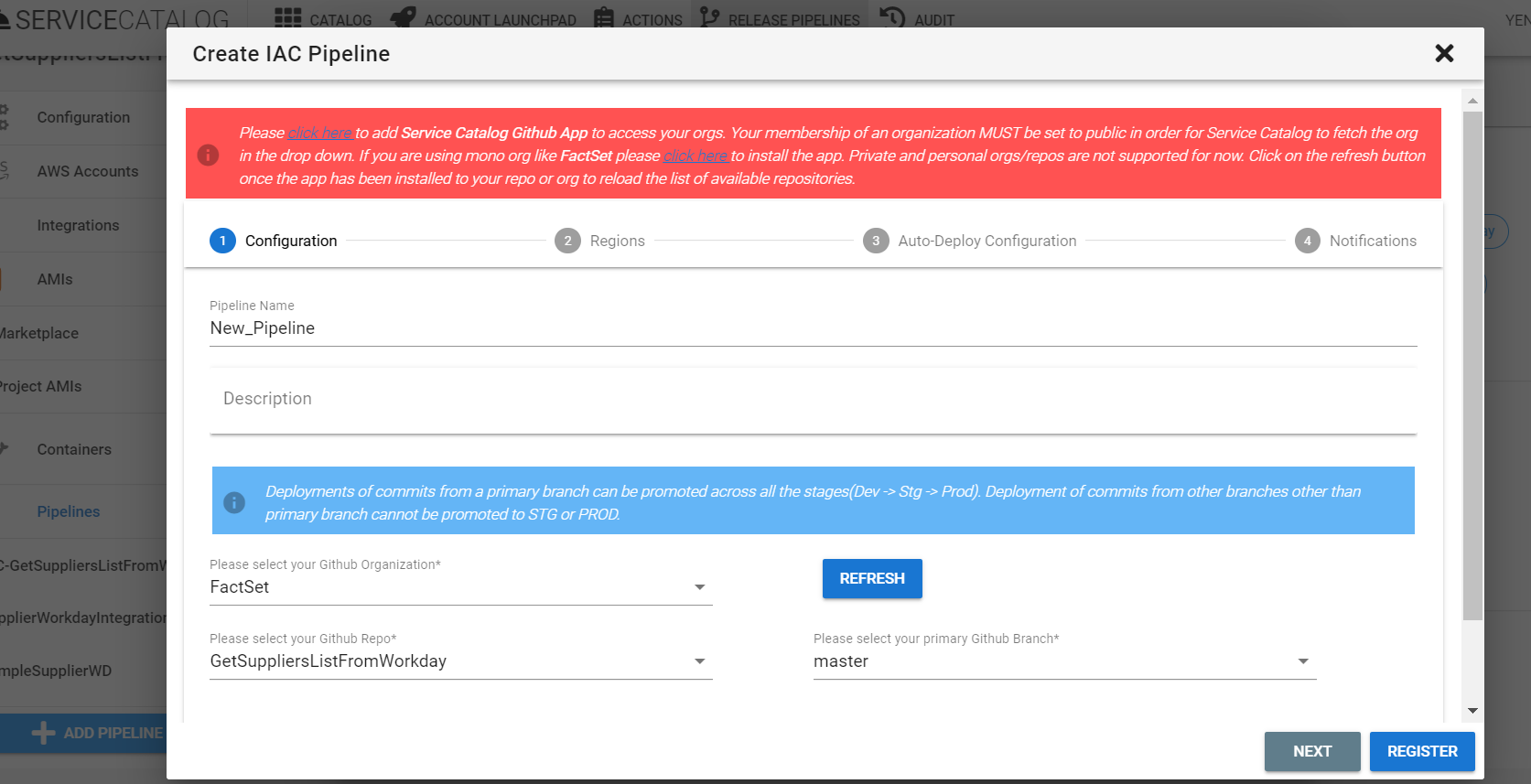
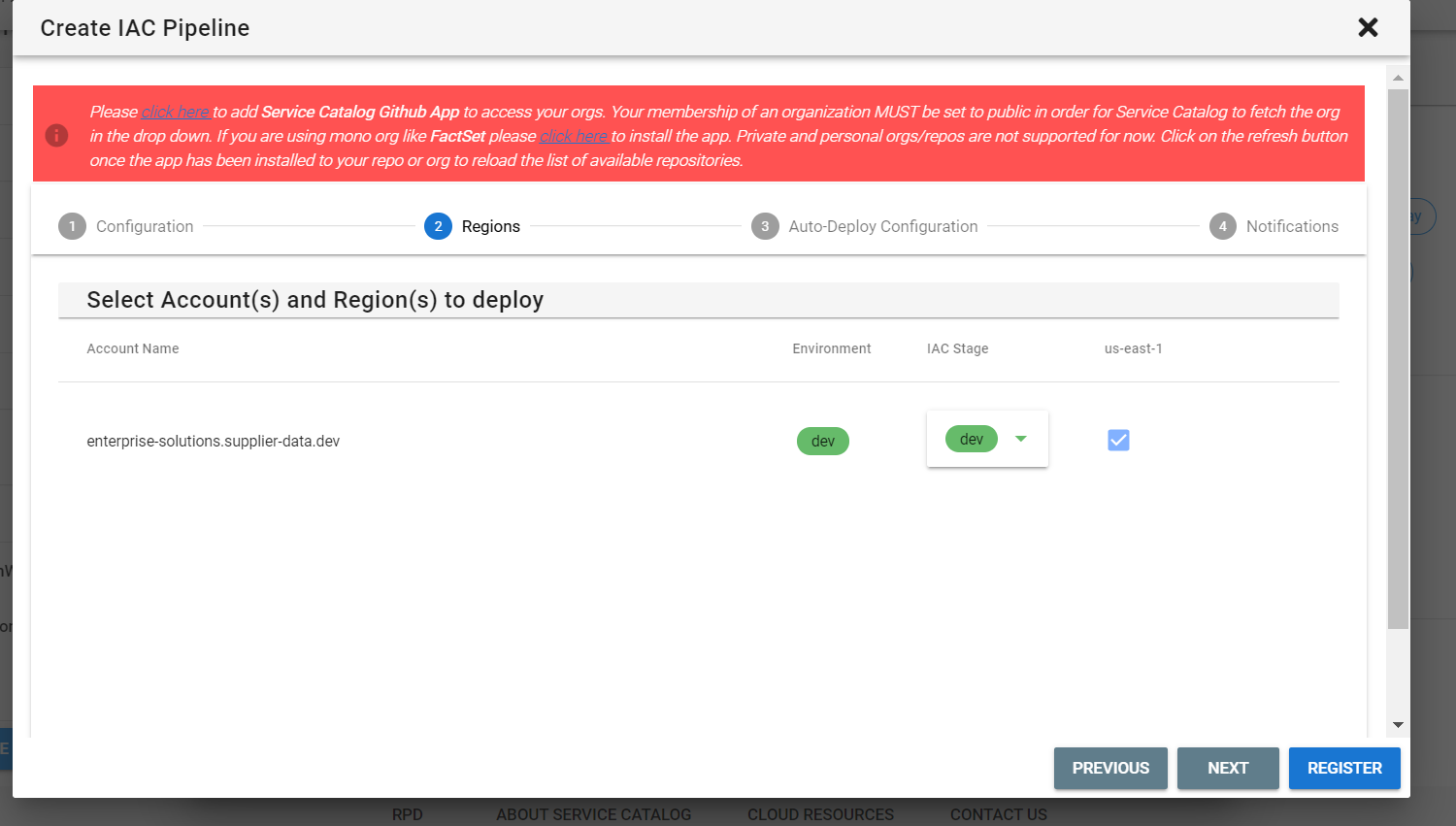
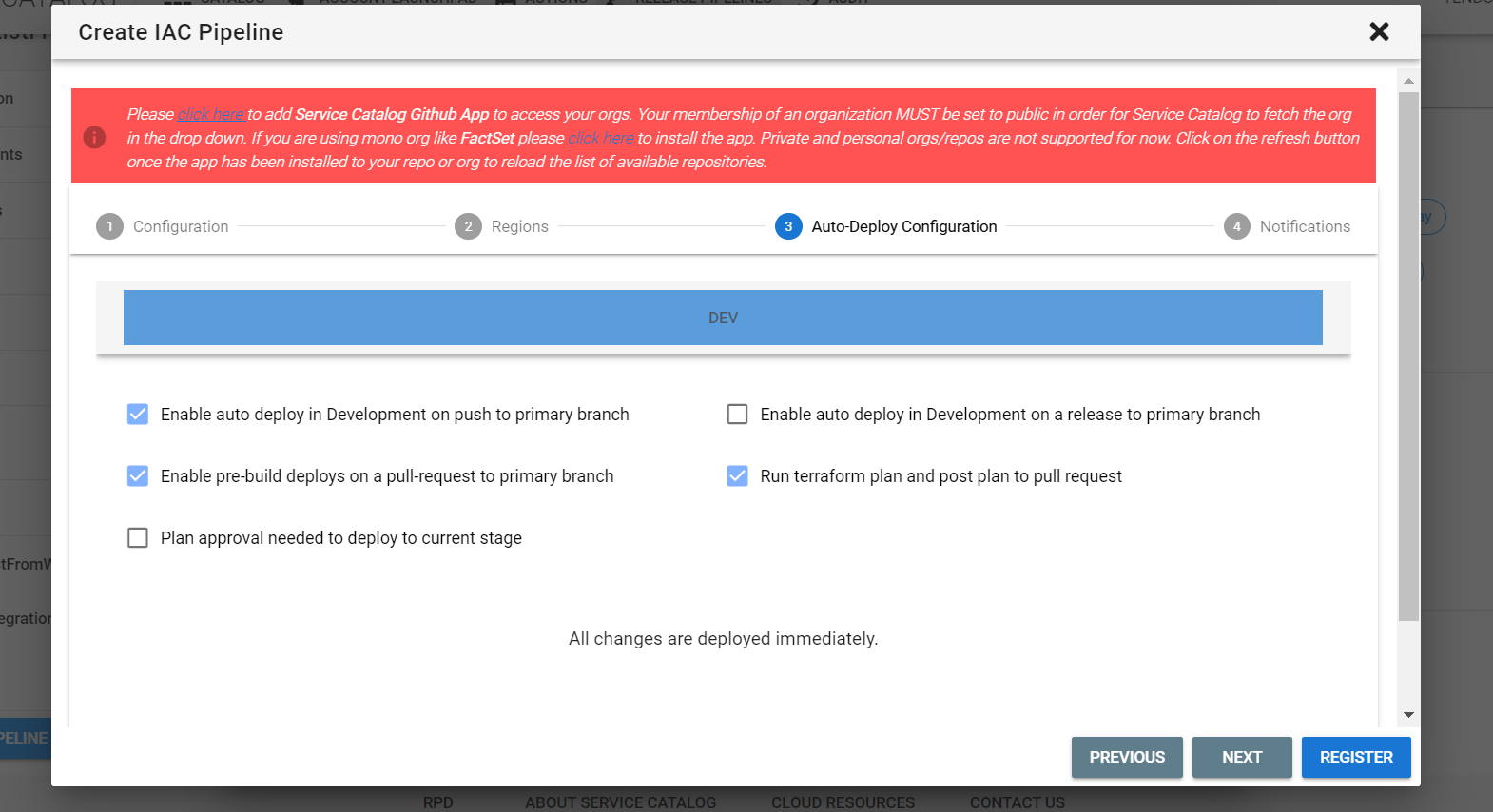
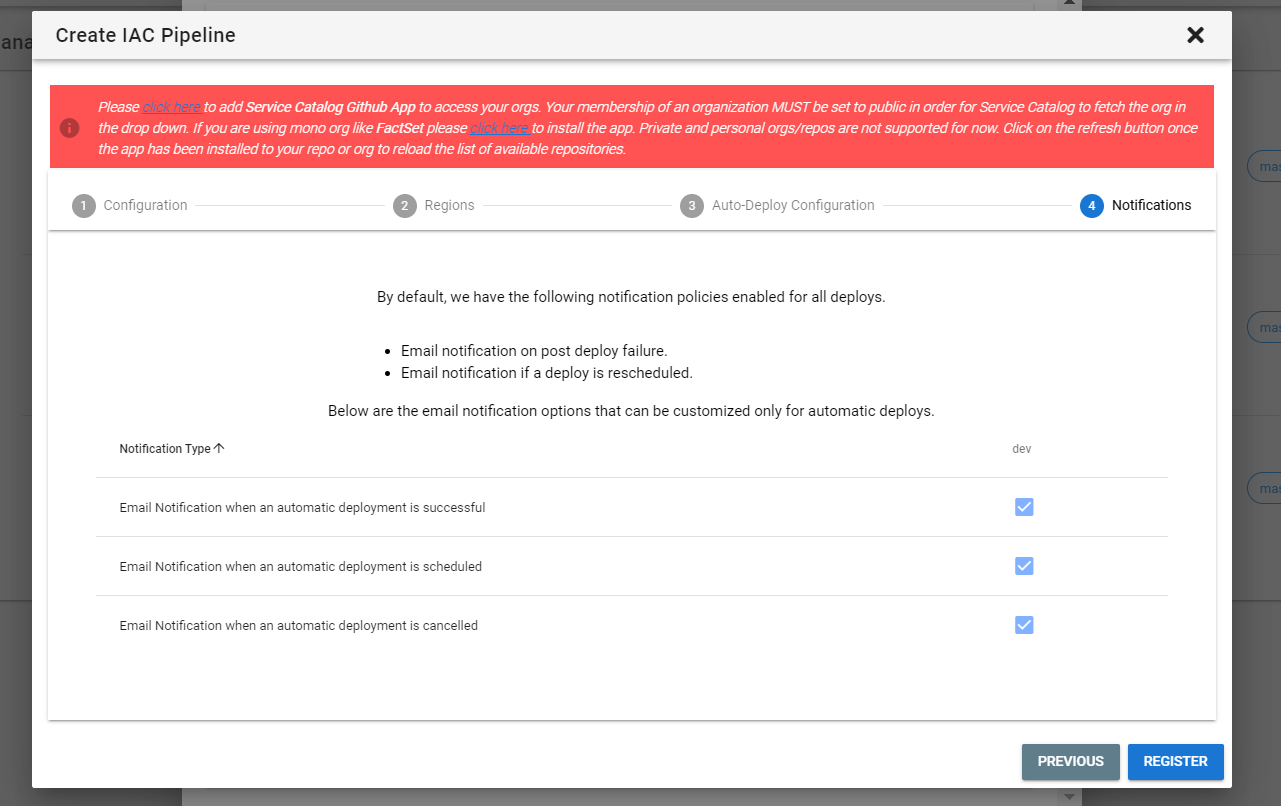
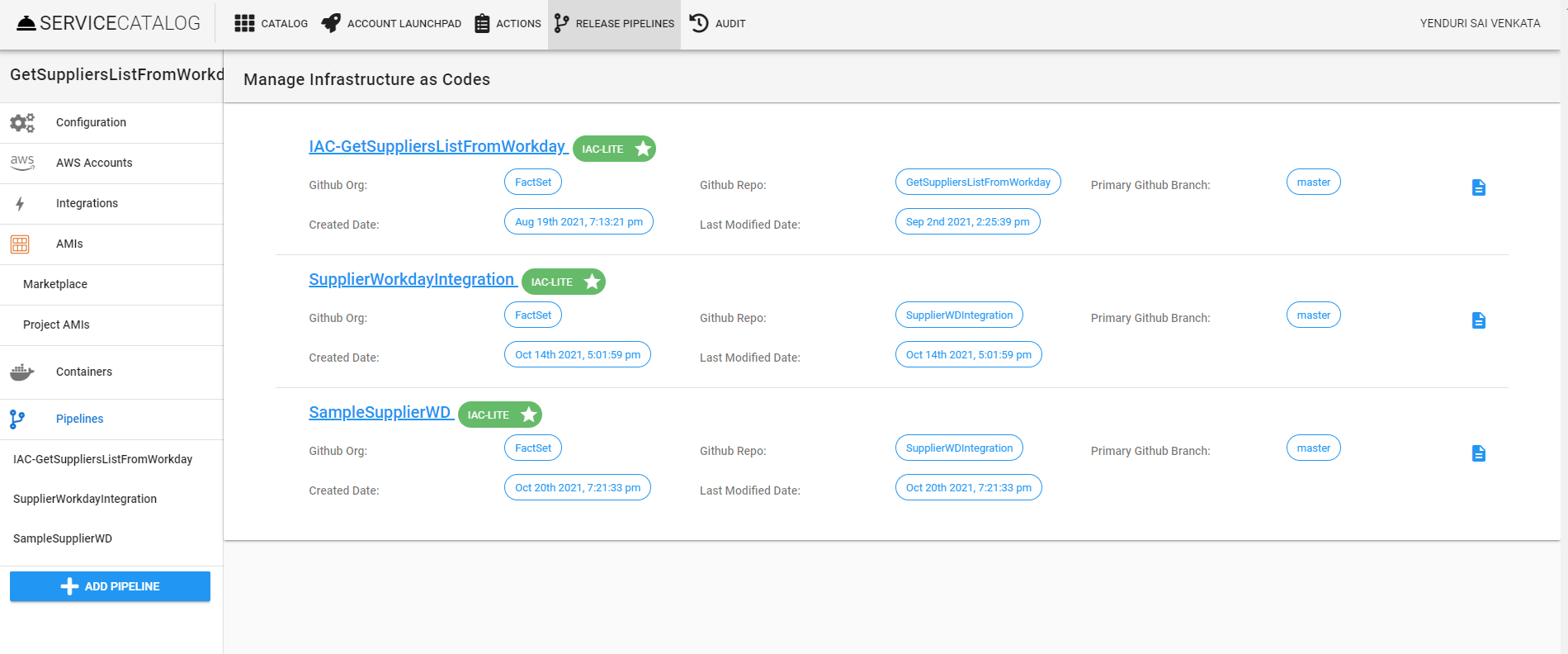
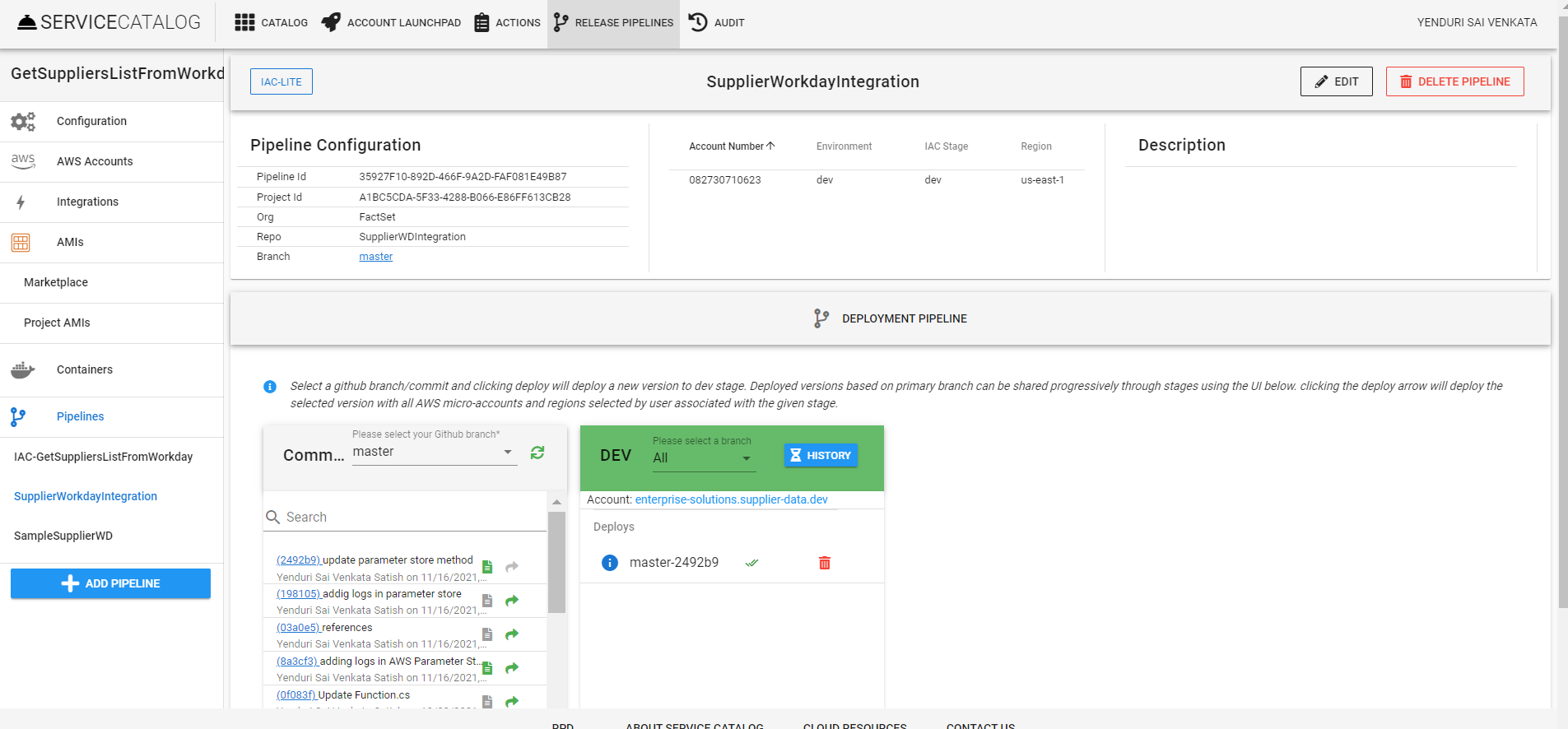
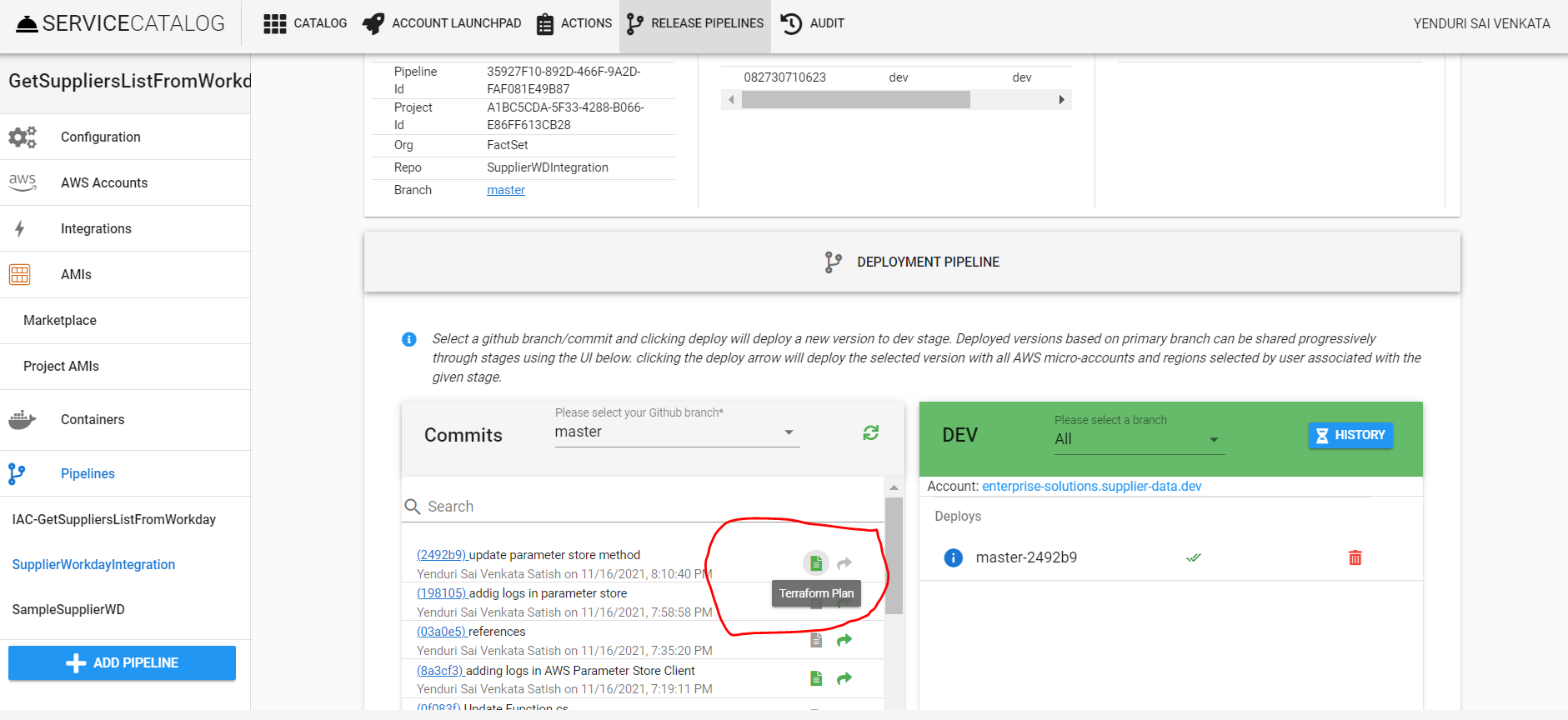
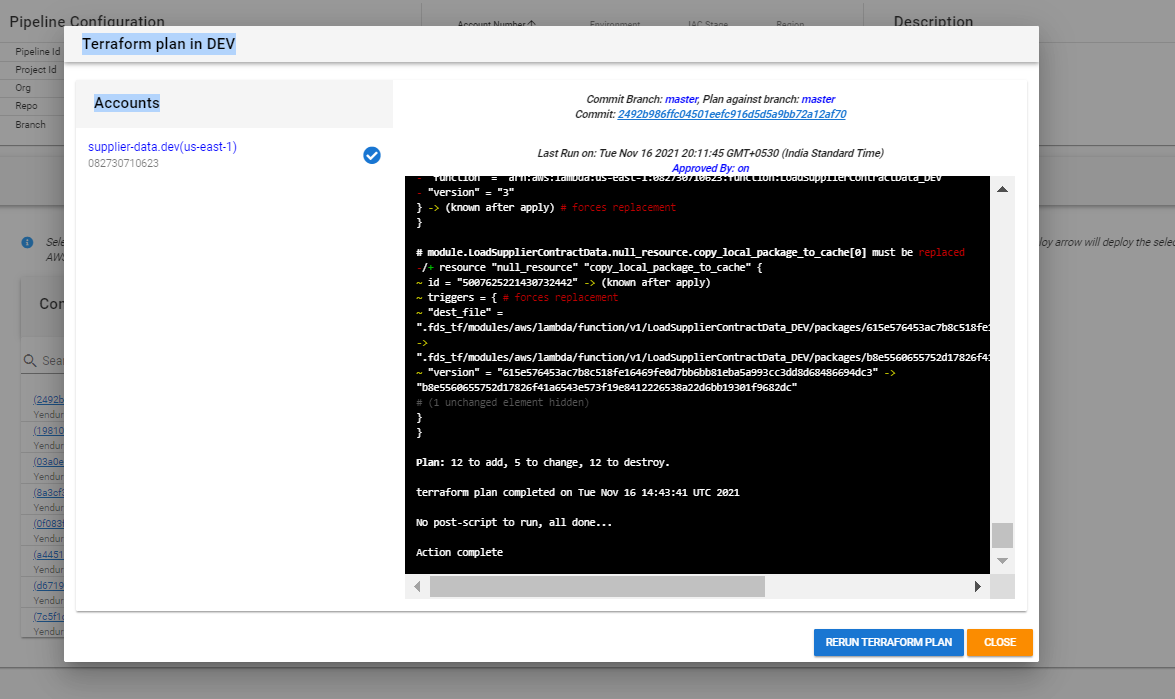
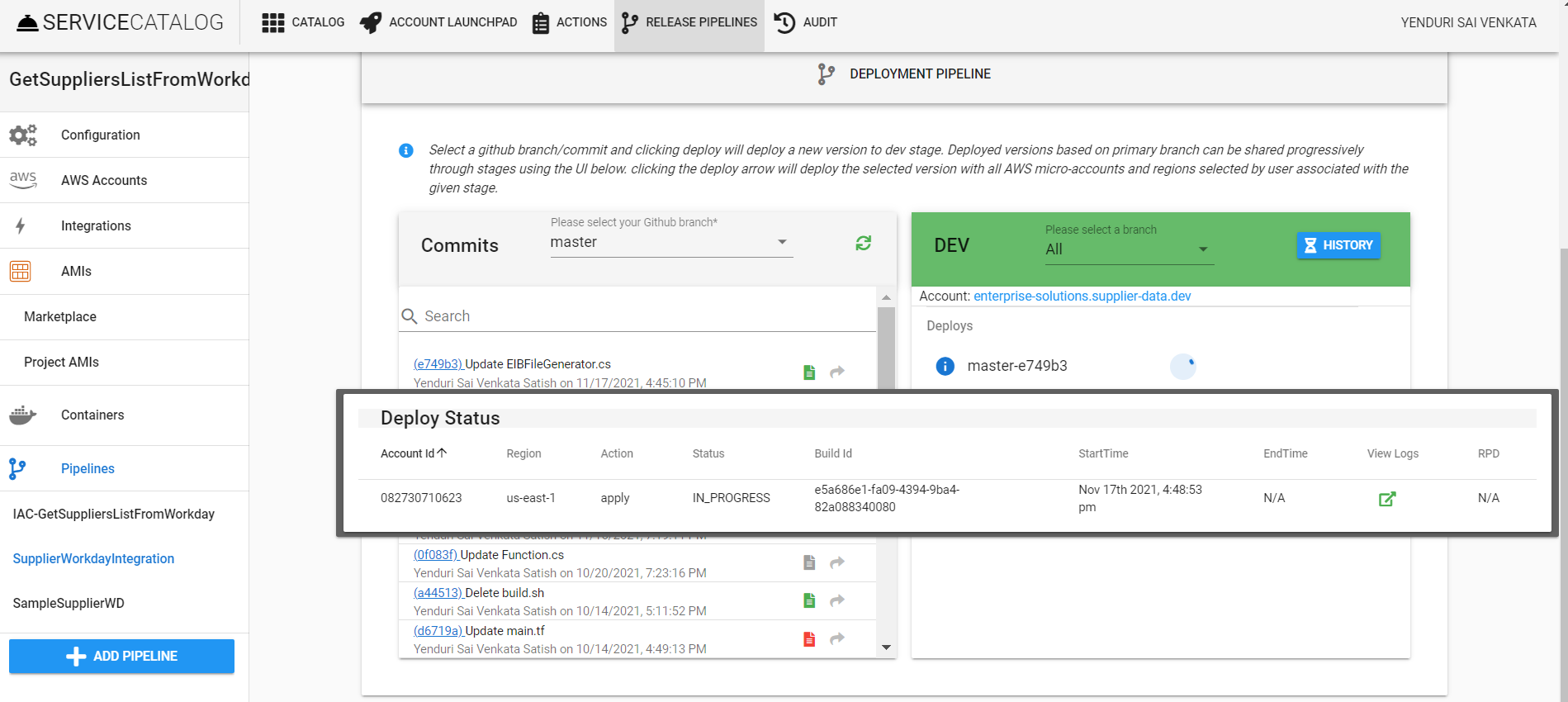
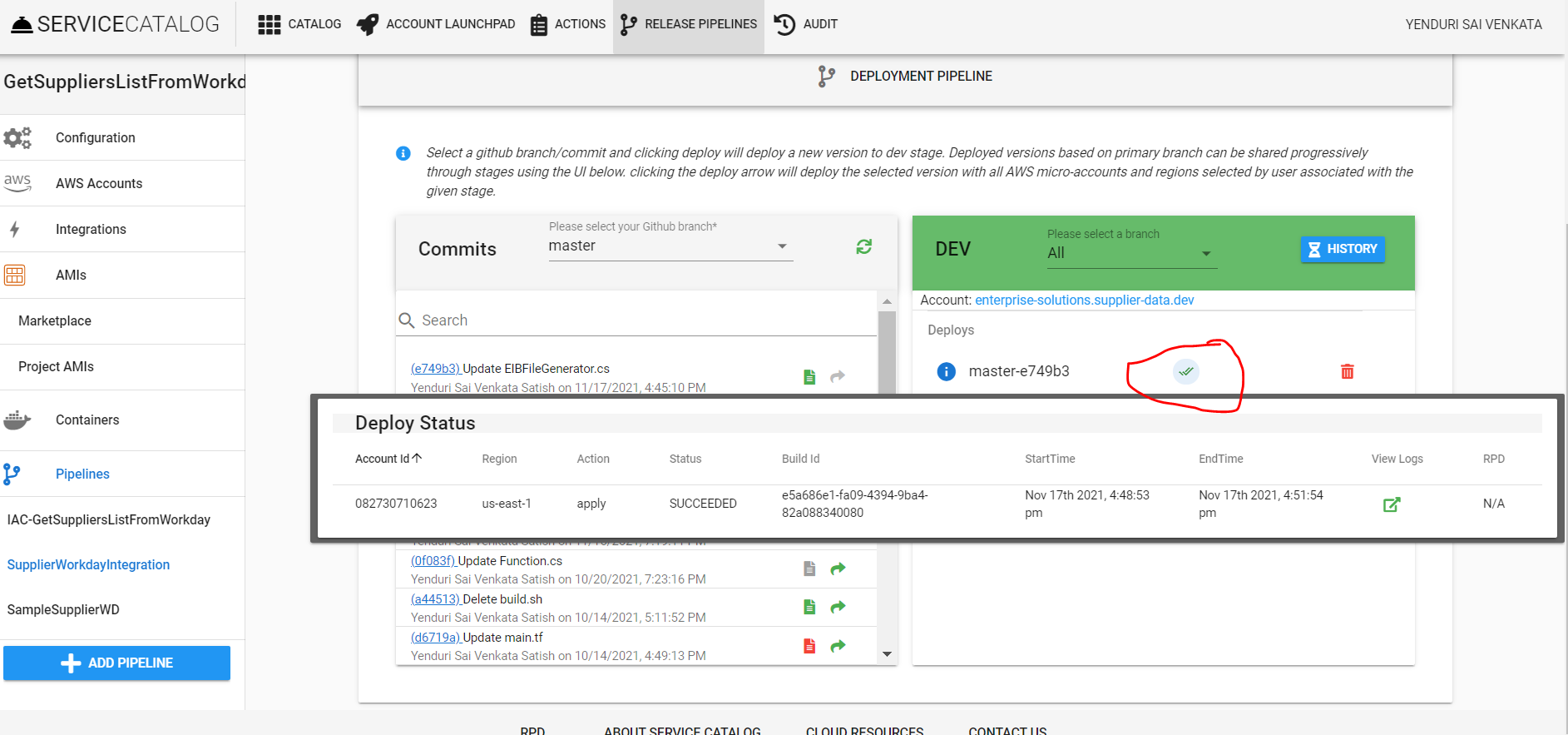


* When we try to publish, it takes aws-lambda-tools-defaults.json as a reference for creating lambda function in AWS.
* We can also add our customized template for deploying lambda to AWS i.e., serverless.template file. It can be added as follows
* 
* **Sample Serverless.template file**
* 
* We can define our parameters for deploying the lambda like lambda name, triggers, environment variables etc.

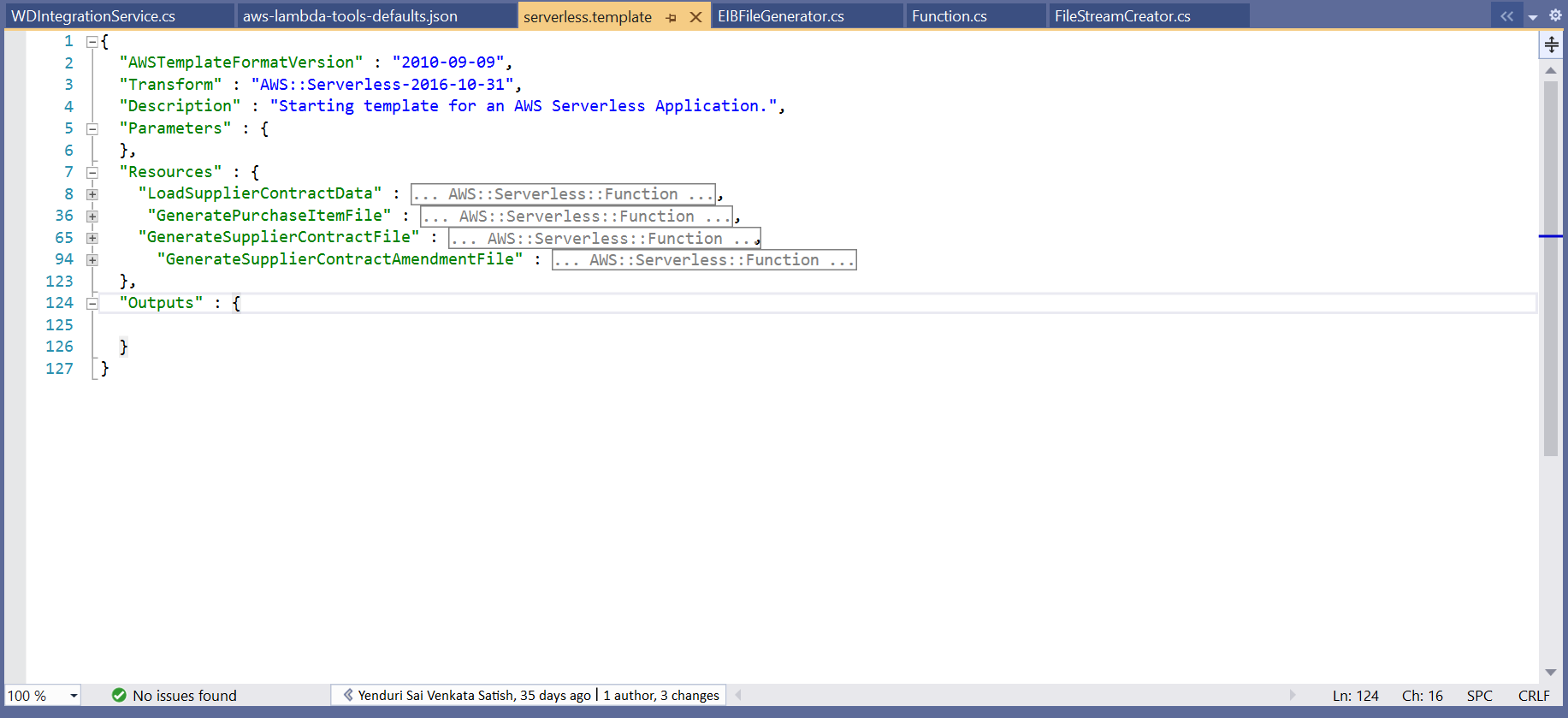
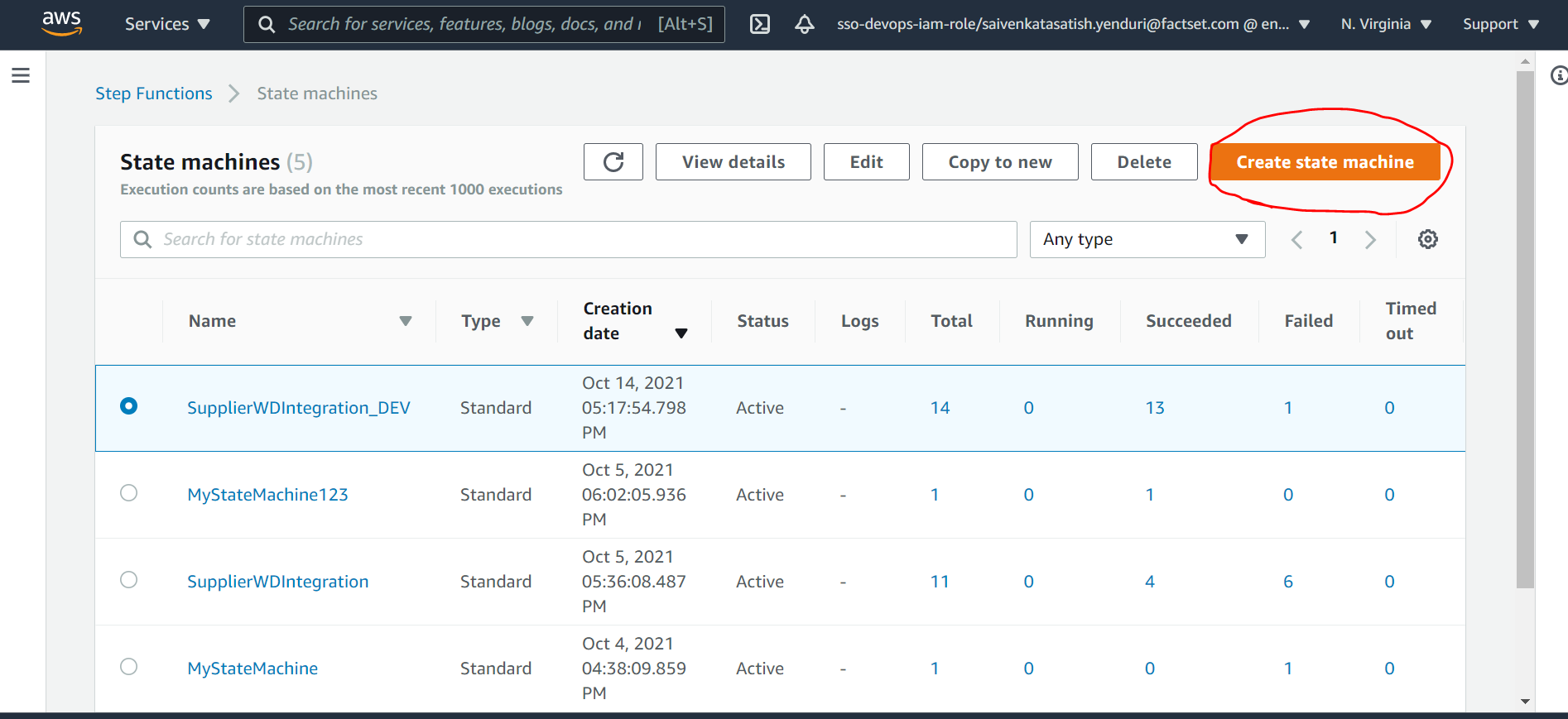
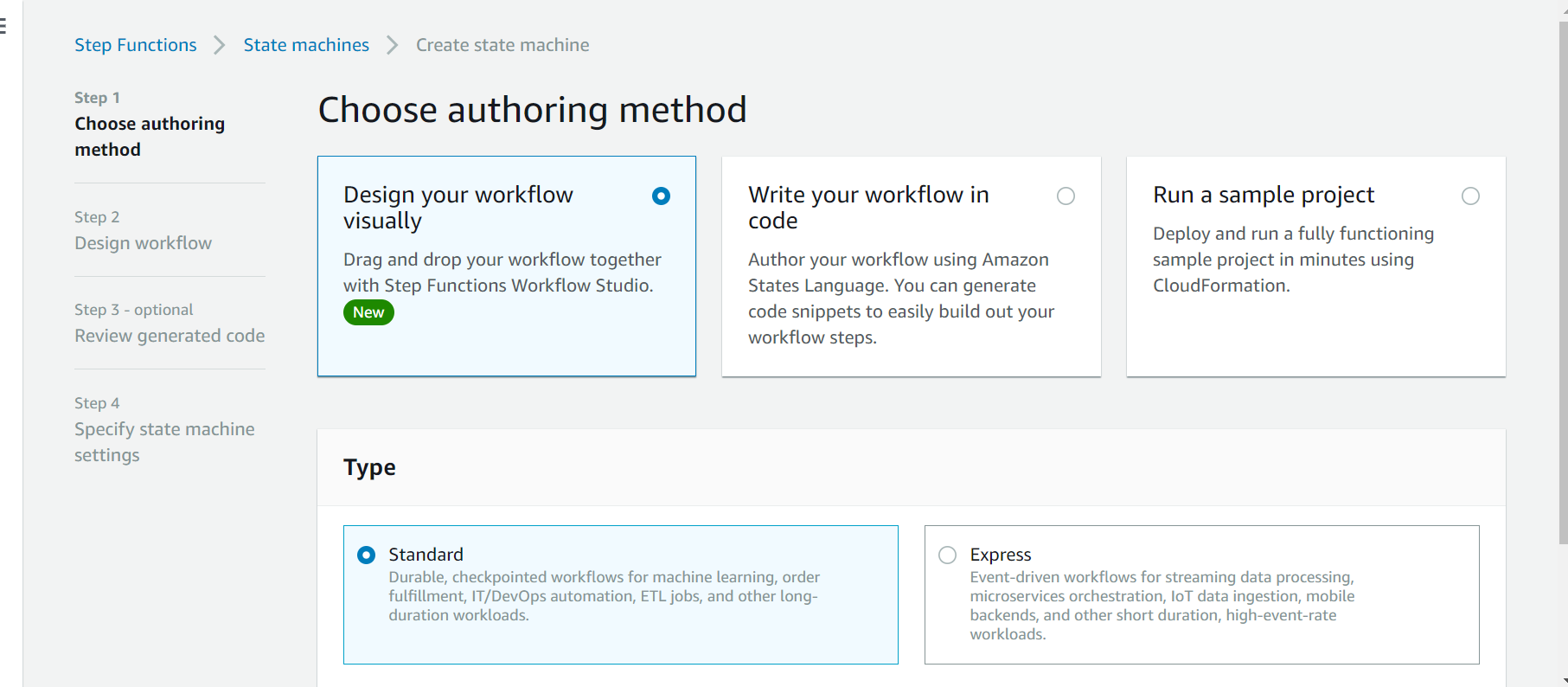
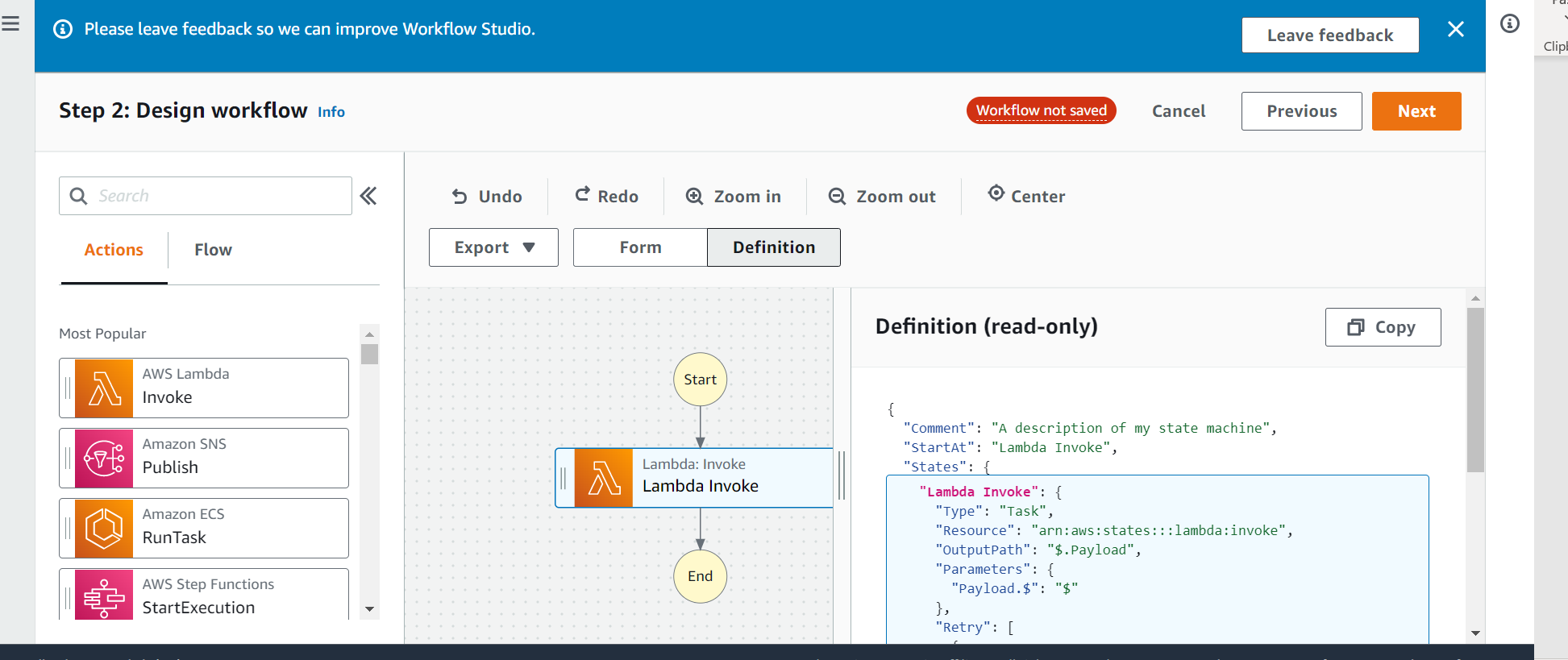
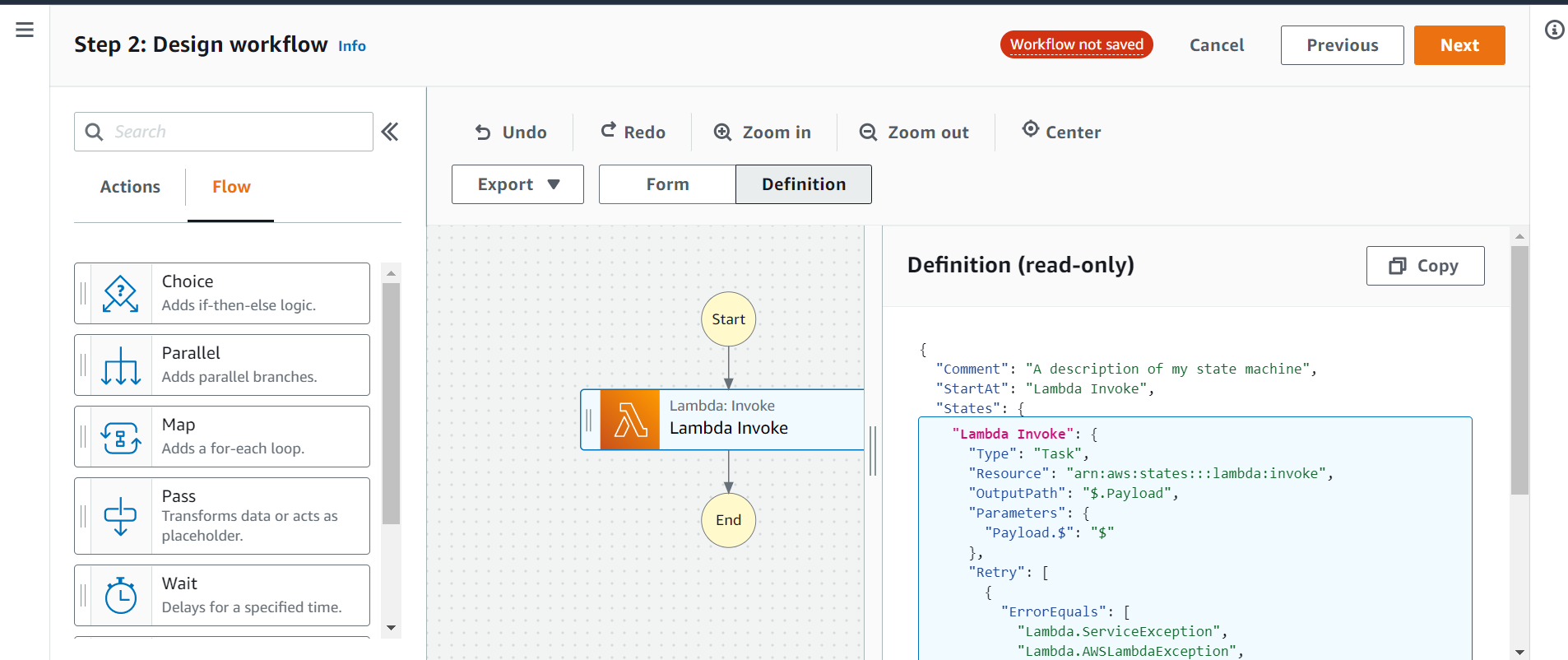
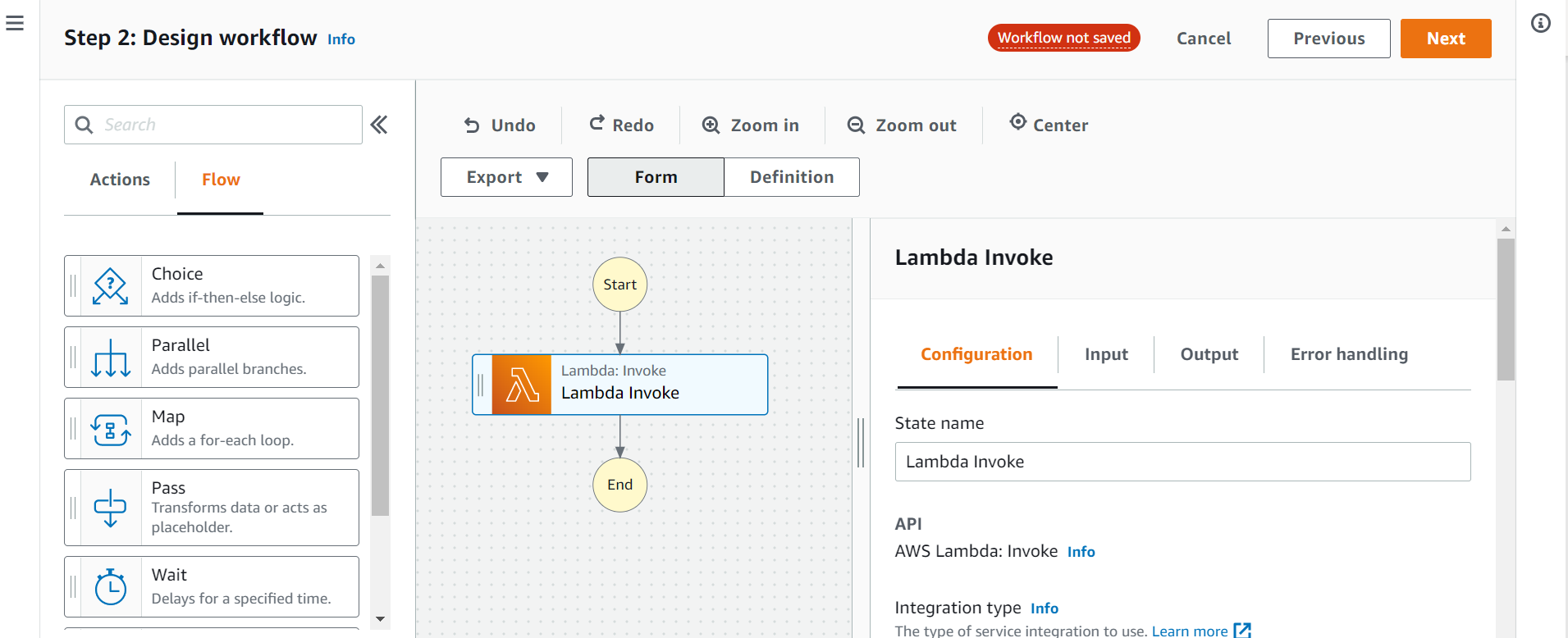
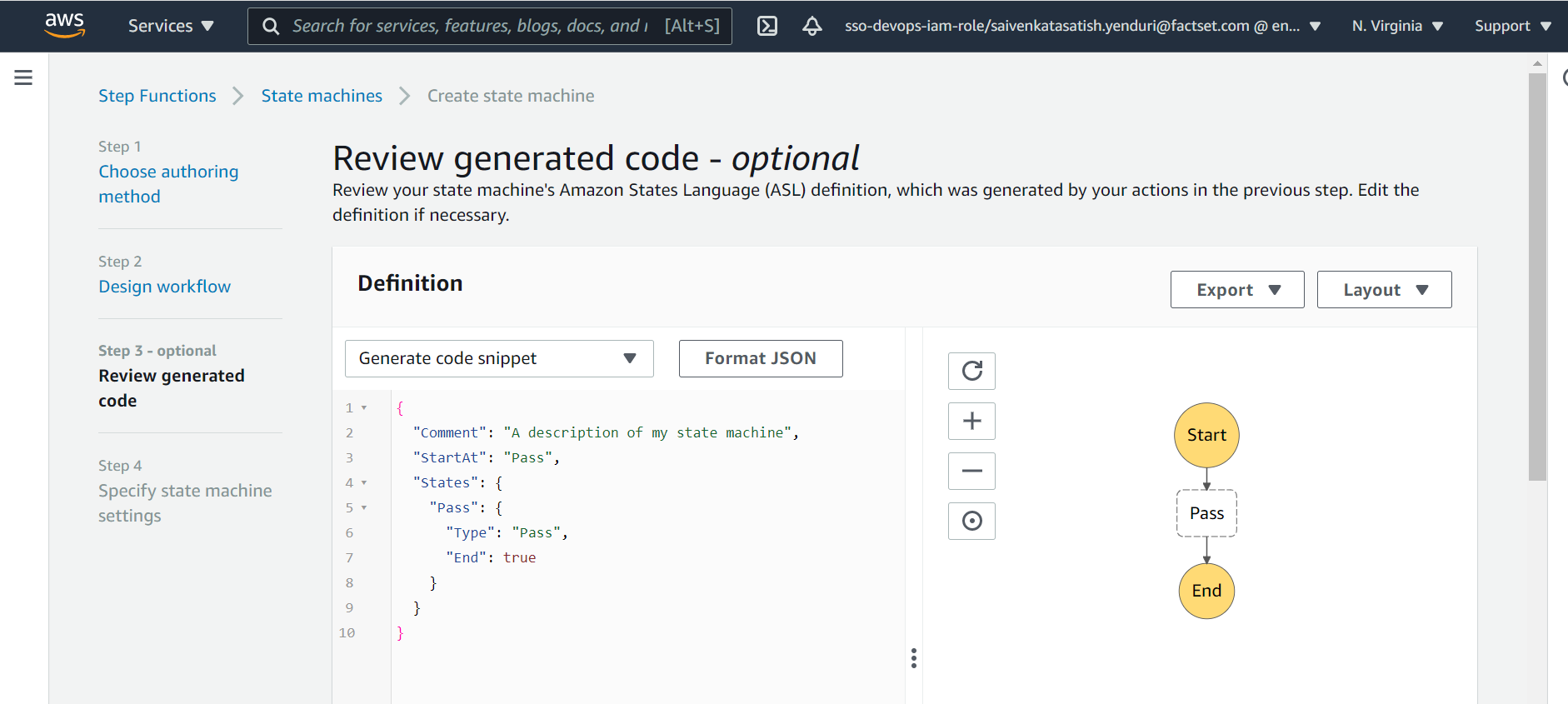
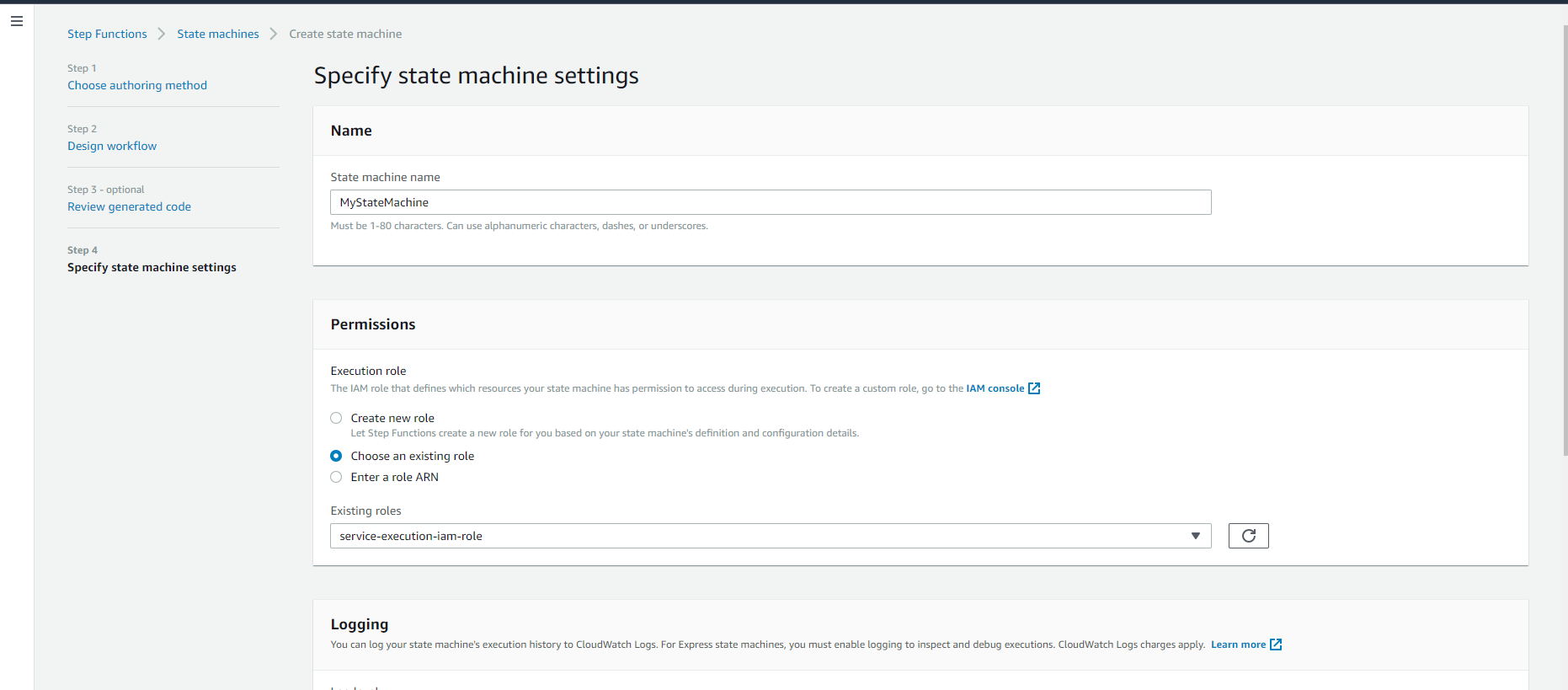
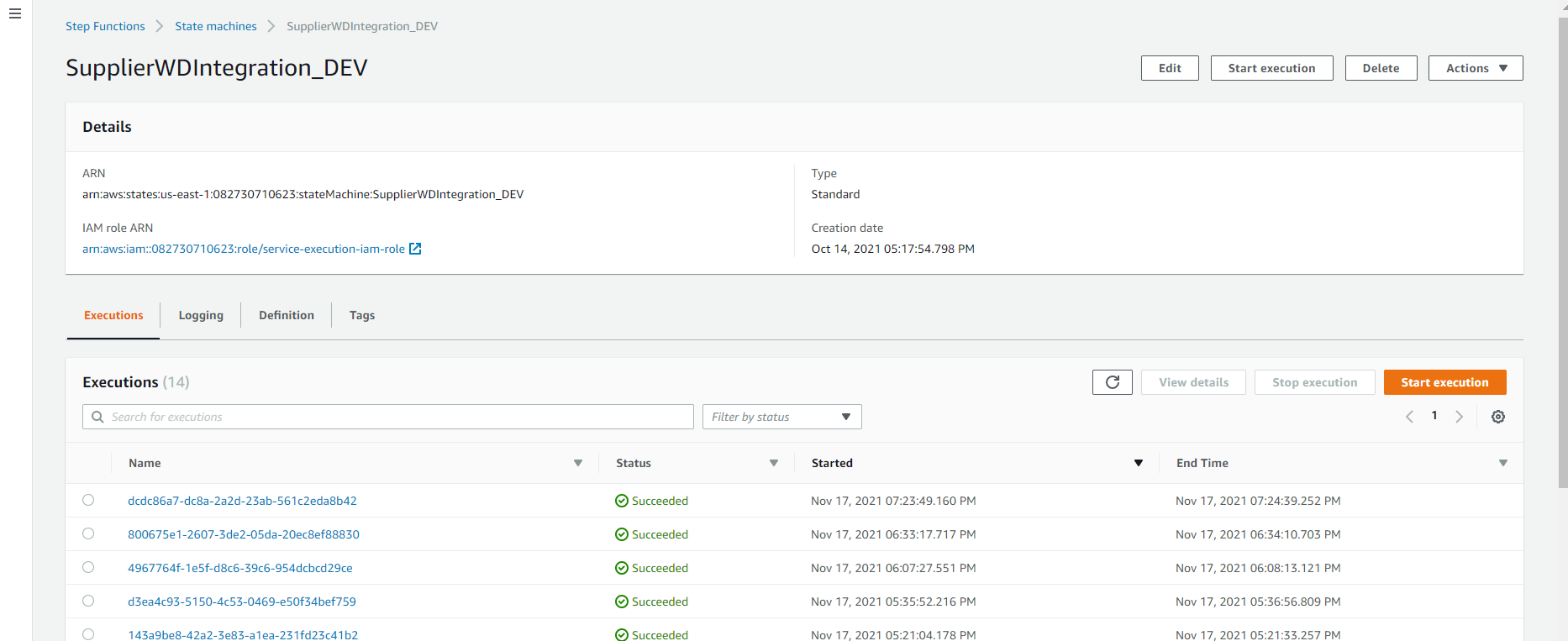
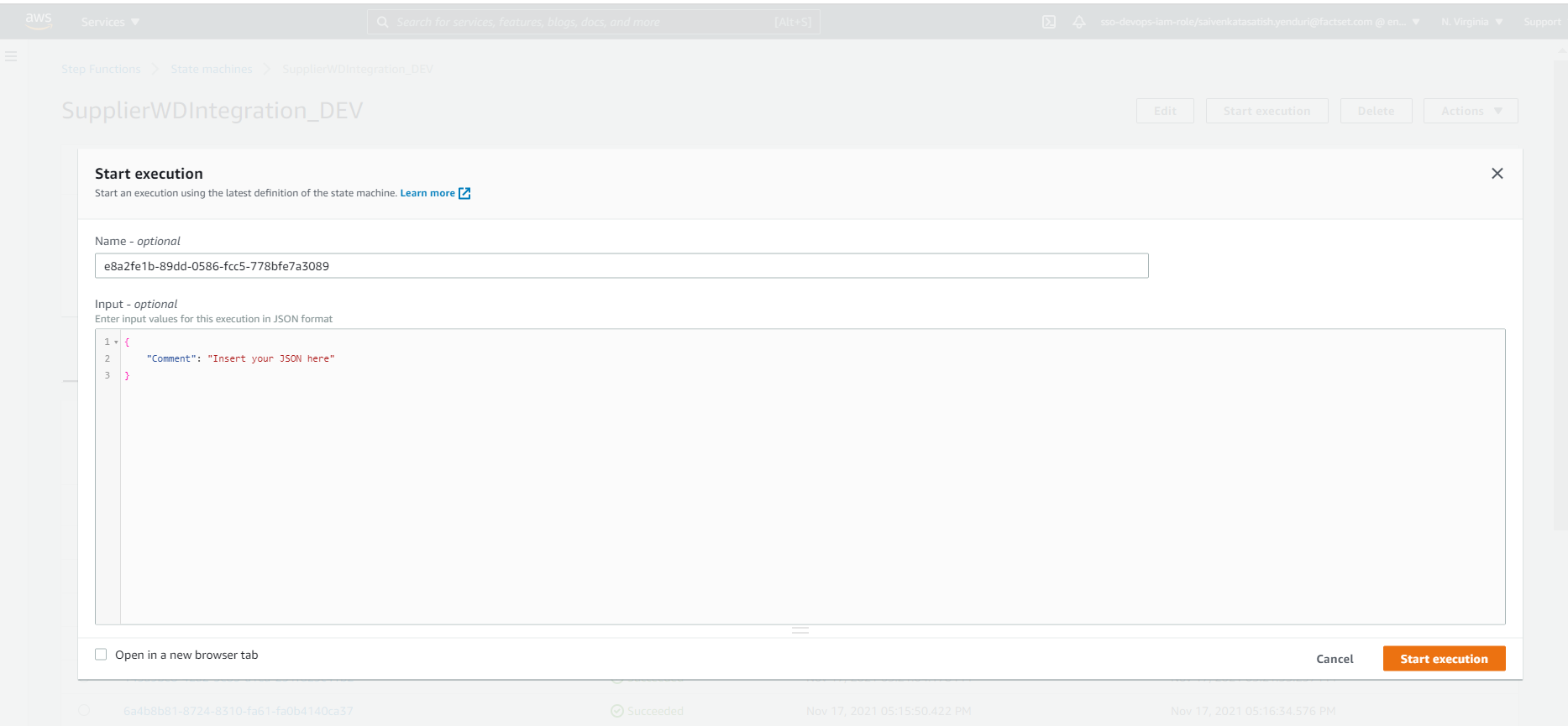
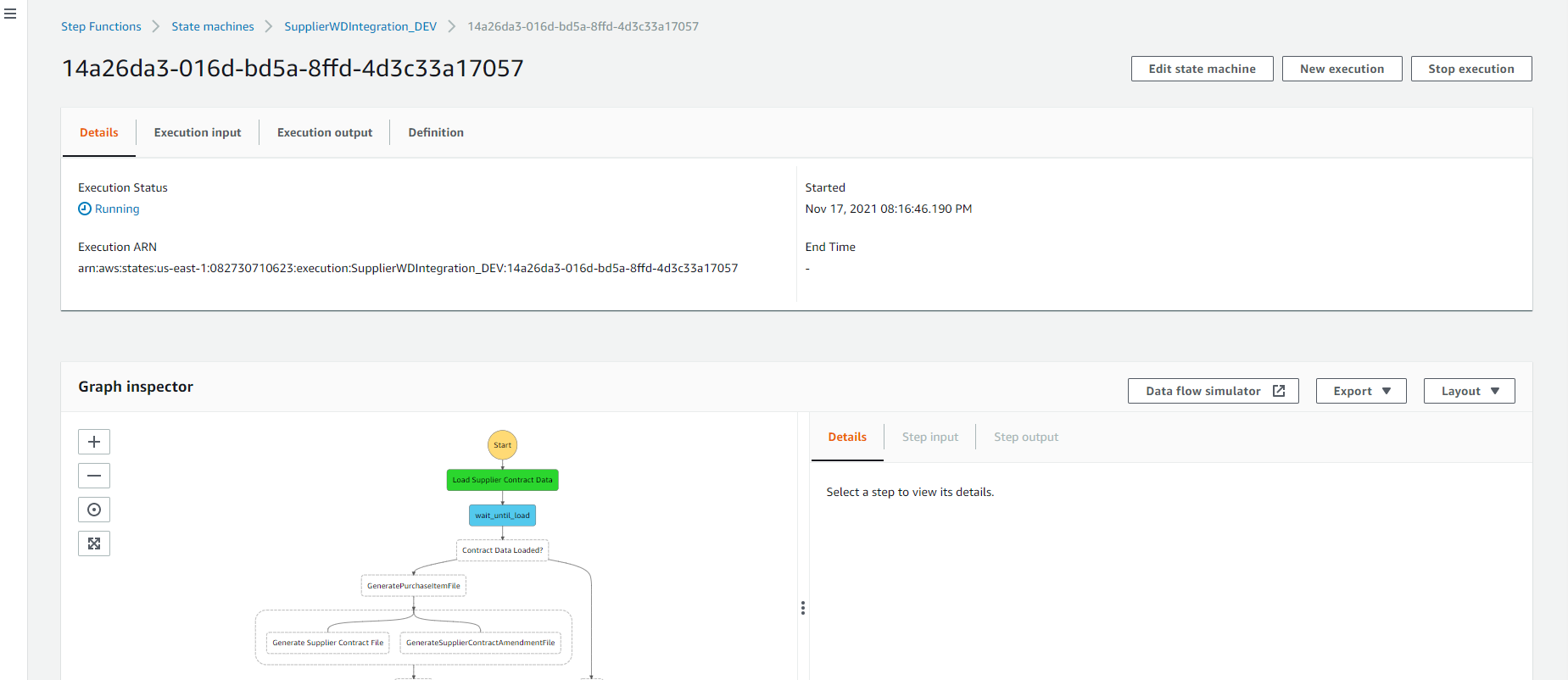
1. **The second way** is to deploy the entire project resources through terraform.

* For this, the solution should be in github and should have these files
* 
* .tfvar files hold the values of variables of respective environments
* Variables.tf has the definition of the variables
* Main.tf is the file which is executed and have the details of all resources that needs to be created in the deployment process.
* .yml file is the file where our deployment starts, It contains post and pre scripts to build our project( it contains the path for build.sh shell script file)
* After setting up these files in github , go to **service catalog** (<https://service-catalog.factset.io/catalog> ), navigate to **Release Pipelines.** Here we need to add new project by clicking the plus symbol as follows.
* 



* After creating new project, we can see this project page with some menu items,
* 
* Go to AWS accounts and link an existing micro account to our project, **Only one project can be mapped to a micro account**
* 
* We can see the list of micro accounts that can be mapped.
* After linking the micro account click on add pipeline shown on left side menu
* 
* Choose a pipeline type which suits best for your project based on resources you are using.
* 
* For our project, we are choosing IAC-LITE pipeline ( Infrastructure as Code LITE)
* 
* Give a name to your pipeline and select the github repository in which the project code is residing.
* 
* Select the organization as **FactSet,** then we can see our repositories in the below list.
* 
* If our repository is not present in the list please follow the steps to grant Service Catalog to access the FactSet repositories.
* 
* 
* 
* After that, we will see our repository in the dropdown list. Select the repository and the branch that is to be deployed.
* 
* Confirm the region to deploy
* 
* Choose deployment settings
* 
* Enable the email notifications if necessary
* 
* Click on register. After registering, our pipeline will be visible in the list.
* 
* On selecting our pipeline, we can see the details as follows
* 
* We can see all the code commits of the respective branch. To run the terraform plan, click on the icon present on commit.
* 
* When we run the terraform plan, we can see the logs as shown below.
* 
* After terraform plan executed without any issues, click on the arrow icon to deploy in dev branch, to check the deployment logs hover on icon present on the target environment. It will show the view logs icon. By clicking on view logs we can see the deployment logs.
* 
* If there are no issues with the deployment, we can see the status by the icon present in the environment component.
* 

1. **Creating Step Functions**

* We can create more than 1 lambdas in a single solution, here we are creating 4 lambdas and defining those as resources in serverless.template file. So that when we publish it will create total 4 lambdas in AWS.
* 
* For our process, we need to execute these lambdas in a particular order. To achieve that we have used AWS Step Functions service.
* Navigate to Step Functions service in AWS and in that, we would find State Machines tab.
* 
* 
* Click on **Create State Machine,** now we need to choose the authoring method at the first step. (choose workflow and type).
* 
* Next step is to **design workflow** where we define the states and their execution workflow. We can do this using drag and drop or writing workflow code.
* 
* 
* We can add Action states that are available or Flow states like choice, parallel etc., On right side we can view as definition of our workflow or as input form.
* 
* In the next step, we need to review our State Machine code. It is shown both in diagrammatic form and as a code snippet.
* 
* In the final step, define the state machine name and define an execution rule for the state machine.
* 
* After creating state machine, we can navigate to our state machine. It contains details like executions, logging, definition, role etc.,
* 
* We can start new execution by clicking on the button, a dialog box opens with an optional input that our state machine takes.
* 
* We can view the status of our states while our state machine is executing.
* 
* We can also watch logs of each step in the cloud watch logs.

Git hub code URL : <https://github.factset.com/FactSet/SupplierWDIntegration>