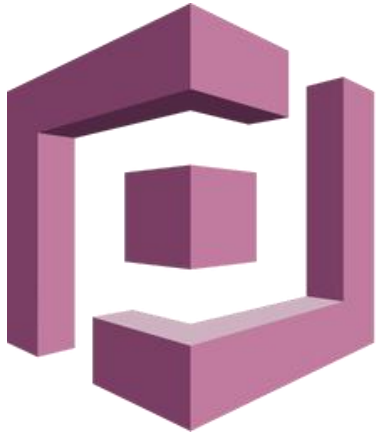
Leveraging AWS Cognito Identity Pool



About me

- Currently Pursing B.Tech Final Year in College.
- Work at *AntStack* as Serverless Developer.
- AWS Community Builder
- AWS Certified Developer Associate.

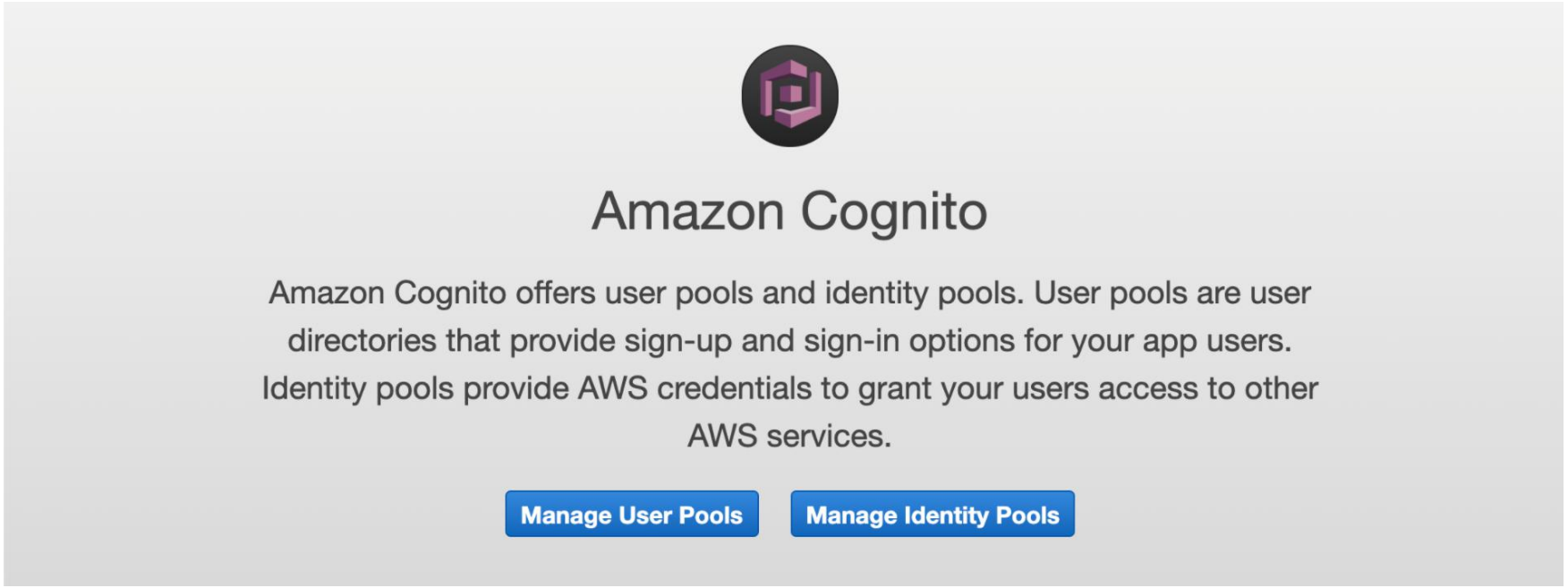
@DebarshiMonda20

Debarshi Mondal

Content

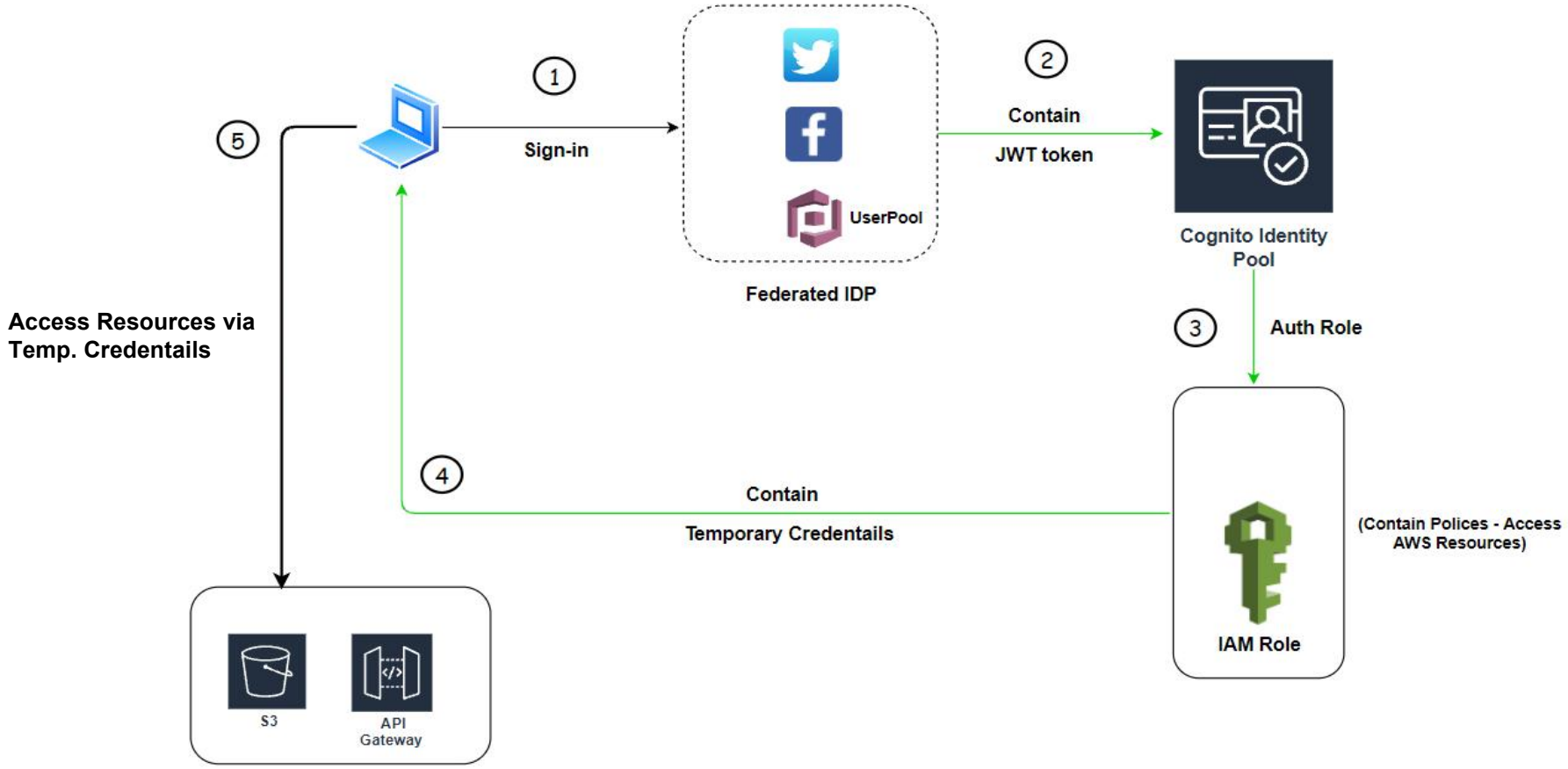
- Cognito Identity Pool
- Architecture Flow
- Understanding with Example
- Integrate Auth & Un-auth Features with Reactjs.
- Conclusion

Cognito Identity Pool:

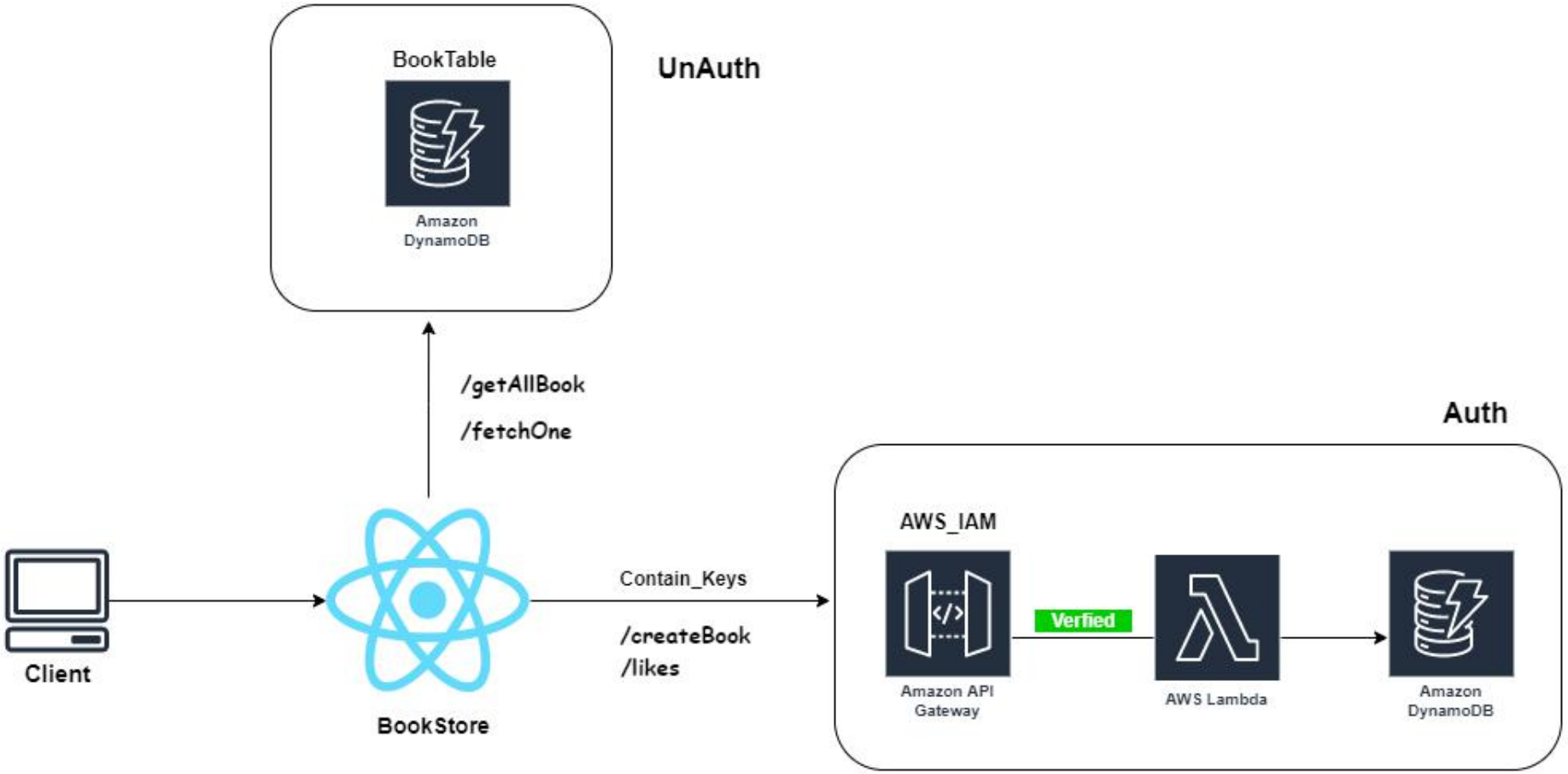


Architecture Flow

- Cognito Identity Pool



Bookstore Flow (Example)



Auth Feature - 01:

- Configuring Cognito **Identity Provider** (Google & Twitter) in ReactJs

```
const allowAccess = async (res, data, twitter_data) => {   // after successful Social Login....
  if ((res && data) || twitter_data) {
    var Keys = new AWS.CognitoIdentityCredentials({
      IdentityPoolId: 'ap-south-1:ba440c5b-f76d-4b05-9127-f51863f46694',  //replace with your Identity Pool
      Logins: {
        'api.twitter.com': twitter_data ? `${twitter_data.oauth_token};${twitter_data.oauth_token_secret}` : null,
        'accounts.google.com': (res && data) ? res.tokenObj.id_token : null,
      }
    });
    Keys.get(async function () {
      var tokens = {
        accessKey: Keys.accessKeyId,
        secretAccessKey: Keys.secretAccessKey,
        sessionToken: Keys.sessionToken
      };
      localStorage.setItem('tokens', JSON.stringify(tokens));
      localStorage.setItem('user', data ? data.email : twitter_data.screen_name);
      credentials.clearCachedId();
      window.location.href = '/';
    });
  }
}
```

Auth Feature - 02:

- Configuring **tokens & headers** in ReactJs.

```
import { AwsClient } from "aws4fetch";

AWS: Add Debug Configuration | AWS: Edit Debug Configuration
export const sendRequest = async (data, path, method) => {
  const values = localStorage.getItem('tokens');
  const { accessKey, sessionToken, secretAccessKey } = JSON.parse(values) || {};

  const aws = new AwsClient({
    service: 'execute-api',
    region: 'ap-south-1',
    accessKeyId: accessKey,
    secretAccessKey,
    sessionToken
  });


  var url = `https://id.execute-api.ap-south-1.amazonaws.com/dev/${path}`;
  const request = await aws.sign(url, {
    method: method || 'PUT',
    headers: {
      'content-type': "application/json"
    },
    body: JSON.stringify(data)
  });
  return await (await fetch(request)).json();
}
```

UnAuth Feature:

- Configuring Unauth Feature & **Calling DynamoDB** within ReactJs.
- The Unauth role must have **sufficient** permissions to call DynamoDB.

```
const credentials = new AWS.CognitoIdentityCredentials({
  IdentityPoolId: 'ap-south-1:ba440c5b-f76d-4b05-9127-f51863f46694'
})
const DynamoDB = new AWS.DynamoDB.DocumentClient({
  credentials,
});

const fetch_one = async (id) => {
  DynamoDB.get({
    TableName: "Book_Table",
    Key: {
      book_id: id * 1,
    },
  }).promise().then((data) => {
    if (data.Item && data.Item.likes > 0) {
      setLikes(data.Item && data.Item.likes)
    } else setLikes(0);
  });
}
```

Conclusion:

- Cognito Identity Pool
- Auth & Unauth Features.
- 3-layer restriction - Auth Feature.
- Access AWS resources **directly** from Reactjs(Frontend) - Unauth Feature.
- Also Integrated with **SAM**(Serverless Application Model).
- GitHub repo: **github.com/LENO-DEV/Cognito-Identity-Pool**