OIDC: Auth GitHub Actions to Push Images to AWS ECR

understand OIDC

You're using GitHub Actions (a tool for automating tasks like building and deploying code) and want it to securely interact with AWS (e.g., to push a Docker image to Amazon ECR). Instead of using long-term AWS credentials (which can be risky), you're using something called OpenID Connect (OIDC) to get temporary access to AWS resources.

Key Parts:

1. **OIDC Provider URL**: https://token.actions.githubusercontent.com
    - This is the URL that GitHub uses to issue a special token when your GitHub Actions workflow runs.
2. **Audience (sts.amazonaws.com)**:
    - This is a label inside the token that tells AWS, "Hey, this token is meant for you."

How It Works:

1. **Workflow Runs**:
    - When your GitHub Actions workflow starts, it needs to do something with AWS (like push a Docker image).
2. **Requesting a Token**:
    - The workflow asks GitHub for a special token from the OIDC provider (the URL mentioned above).
3. **Token Issuance**:
    - GitHub gives the workflow a token. This token has information like:
        - Who is running the workflow (which repository, branch, etc.).
        - The audience (sts.amazonaws.com), telling AWS that the token is for them.
4. **AWS Role Assumption**:
    - The workflow sends this token to AWS and says, "Please let me assume this role so I can push my Docker image."
    - AWS checks:
        - Is this token from GitHub? (It checks the OIDC provider URL).
        - Is the token meant for AWS? (It checks the audience, sts.amazonaws.com).
5. **Temporary Access Granted**:

- If everything checks out, AWS gives the workflow temporary access credentials.
- The workflow can now push the Docker image to Amazon ECR or do whatever it needs to do.

6. **Security**:
    - These credentials are temporary and only valid for a short time, making it much safer than using long-term access keys.

In Summary:

- Your **GitHub Actions Workflow** asks the **GitHub OIDC Provider** for a special token
- The **GitHub OIDC Provider** issues the token, which is then passed to **AWS IAM Role** via **AWS STS**.
- gets temporary access to do its job after **AWS STS** validates the OIDC token

note: all without needing to store or manage long-term AWS credentials. This process is secure and specific to each workflow run.

Configuration details

1. Create an IAM Role in AWS

- Step 1: Go to IAM: Sign in to the AWS Management Console, and navigate to the **IAM** service
- Choose **Identity providers** from the left-hand menu → **add provider**
- Follow github documentation to add **GitHub OIDC Provider**

https://docs.github.com/en/actions/security-for-github-actions/security-hardening-your-deployments/configuring-openid-connect-in-amazon-web-services

- assign **role** and add the needed **policy** (for example allow githubActions to push images to AWS ECR)

2. configure githubAction workflow

- add permissions for requesting the JWT and to checkout the code (details are also in the same documentation link above)
- Role-to-assume: **arn** of the created aws role