AWS IAM Risk Analyzer — Development Documentation (Console + Code)

1. Motivation & Design Decisions

Why this project exists: AWS does not provide a single, interactive dashboard for IAM risk analysis that combines high-risk detection, least-privilege recommendations, unused identity cleanup, and visual charts.

- Serverless First: Chose Lambda + API Gateway for fully managed, cost-efficient infrastructure with zero server maintenance.
- Principle of Least Privilege: Lambda uses only IAMReadOnlyAccess to prevent accidental modifications.

2. AWS Console Steps

- Lambda Setup
- Created Lambda function IAMLambdaAnalyzer.
- Selected Python 3.14 runtime.
- Chose existing role IAMAnalyzerLambdaRole with:
- IAMReadOnlyAccess — to read IAM users, roles, and policies.
- CloudWatchLogsFullAccess — to monitor Lambda execution.
- Deployed Lambda code that:
- Lists users, roles, and attached policies.
- Flags high-risk policies (* or service:*).
- Adds least-privilege recommendations.
- Identifies unused users/roles.
- API Gateway Setup
- Created HTTP API LambdaAPIGateway.
- Added route GET /iam-data integrated with the Lambda function.
- Enabled CORS:
- Access-Control-Allow-Origin: * (so browser dashboard could fetch API data).
- Allowed methods GET, OPTIONS.
- Deployed stage prod with Auto-Deploy enabled.
- Verified the API endpoint via curl to ensure data was returned correctly.
- Testing & Debugging
- Used CloudWatch Logs to confirm Lambda executed successfully and returned expected JSON.
- Tested API in browser and terminal to verify CORS and response format.

3**. Lambda Function Logic**

- Data Collection:
- Fetch all IAM users and roles.
- Fetch attached policies for users and roles.
- High-Risk Detection:

- Policies with * or service:* actions are flagged HIGH.
- Least-Privilege Recommendations:
- Suggest restricting wildcard actions and resources.
- Unused Identities:
- Users or roles with no attached policies are marked as unused.

4. Frontend / Dashboard

- Built a static HTML + JS dashboard to display:
- Users, Roles, Policies, High-Risk flags, Recommendations.
- Cleanup summary for unused accounts.
- Interactive filters (high-risk only, users only, roles only).
- Live search by username, role, or policy name.
- Added Chart.js doughnut chart showing High-Risk vs Low-Risk policies.
- Configured config.js for API endpoint to safely publish code without exposing secrets.
- Served HTML through local Python server to avoid browser file:// fetch restrictions.

5. Unique Implementation Notes

- AWS does not offer a ready-made, interactive IAM risk dashboard with least-privilege recommendations.
- This project flags AWS-managed roles correctly, which many tools skip.
- Combines serverless backend, security logic, and modern frontend UX.

7. Security Considerations

- No AWS credentials stored in frontend.
- Lambda uses least-privilege role.
- API Gateway returns read-only IAM data only.
- CORS configured explicitly to allow only frontend access.