Ralph Klos
PROJECTS & CONSULTANCY

GUIDE

Working with the IAM Builder

Least-privilege roles, trust policies and permissions for AWS IAM

FOR
Developers, cloud and platform teams

FROM
Ralph Klos Projects & Consultancy

VERSION
IAM Builder 0.1

DATE
6 October 2026

Contents

1. In short . . . 3

2. Opening it . . . 3

3. Region and accounts . . . 3

4. A role in three questions . . . 4

5. From code or from logs . . . 5

6. Only a policy . . . 6

7. The result . . . 6

8. Rolling it out . . . 6

9. Saving projects . . . 7

10. Safety and limits . . . 7

1. In short

The IAM Builder makes IAM roles with only the permissions an application really needs. For each role you answer three questions: who uses it, what may it do and what starts it. You get the trust policy, the permissions policy and, across accounts, the policy on the other side too, as JSON or Terraform.

- **The minimum that works: read** or **read and write** picks the actions that are needed together, not every read or write action of the service. You tick more yourself.
- **Any AWS service:** with the actions, resource types and ARNs from the AWS Service Authorization Reference.
- **From your code or from the logs:** paste code (boto3, JavaScript SDK, AWS CLI) or CloudTrail events, and the actions that are really used are added.
- **Fully in your browser:** the page does not call AWS, asks for no keys and sends nothing out, not even pasted code and logs. The page fixes that itself in its Content-Security-Policy.
- **Dutch and English:** switch with **NL / EN** at the top. Your choices stay stored in the browser.

> Every statement in the result says why it is there. A reviewer can read the policy without knowing the builder.

2. Opening it

- Online: **rkpc.be/tools/iam-builder/**. Nothing is installed and nothing is stored on the website.
- Or locally: open `index.html` of the `rkpc-iam` folder in a recent browser (Edge, Chrome, Firefox, Safari). Also without internet.
- The browser remembers the last state. **New** clears everything, after a confirmation in the page.

3. Region and accounts

Fill in the region and the accounts the roles and resources live in, each with a name and a 12-digit account ID. As long as an ID is missing, the AWS documentation example 111122223333 is in the policies and the page warns. With more than one account you pick the account per role and per resource; that is how the builder also makes access across accounts.

4. A role in three questions

Click **+ New role**. Each role is a card with three questions. The name fills itself in from who uses the role; you can change it yourself.

4.1 1. Who uses this role?

This becomes the trust policy. Pick from one list:

CHOICE	WHAT THE TRUST POLICY GETS
EC2 server	`ec2.amazonaws.com`, an instance profile, and optionally Session Manager and the CloudWatch agent
ECS task	A task role and an execution role, with `aws:SourceAccount` and `aws:SourceArn`; ECR, logs and ECS Exec
Lambda function	`lambda.amazonaws.com`, logs only to its own log group, optionally VPC and X-Ray
Another AWS service	Over 30 services (Backup, Glue, Step Functions, CodeBuild, EKS Pod Identity …) or your own service principal, with `aws:SourceAccount` and optionally `aws:SourceArn`
GitHub Actions	OIDC with `aud` and a `sub` for your organization, repository and branch or environment; never a bare *
Another account	One role or the account, optionally with an external ID and MFA
External identity provider	OIDC or SAML, with the conditions on audience and subject

> `aws:SourceAccount` protects against the confused deputy problem: the service may assume the role only when it works for your account. The example at `aws:SourceArn` follows the service you pick.

4.2 2. What may the role do?

This becomes the permissions policy. Click **+ AWS service** and pick from the list: the ready-made resources at the top, all AWS services below.

KIND	HOW IT WORKS
Ready-made	S3, DynamoDB, SQS, SNS, Secrets Manager, Parameter Store, ECR and Lambda. You fill in the name and pick the level (e.g. read, write, read and write). The row shows the minimum and the extra actions as checkboxes, each with why you would need it. Optionally the KMS key, only through that service.
Any other service	Pick one or more resource types and fill in the parts of the ARN. Then click **Minimum: read** or **read and write**, or tick the actions yourself. The actions of the picked resource types are at the top.

The minimum holds the actions that are needed together: an update does not work when the code cannot find or read what it updates. For Backup with a backup plan that is `GetBackupPlan`, `ListBackupPlans`, `ListBackupPlanVersions` and `UpdateBackupPlan`. For about 50 common services the minimum comes from a curated list; for the others it is an estimate based on the resource types. The page says which.

Tagging is always in the minimum

Tag and Untag of the resource itself are always in the minimum, at every level: tag policies and scanning tools check that a resource's tags are kept up to date. The builder takes the service's own actions (`sqs:TagQueue`, `ssm:AddTagsToResource`, `iam:TagRole`, `backup:TagResource` …), only on that one resource and only in the role's policy. If a service cannot tag the picked type (a Glue table), the page says so.

> The builder never leaves an action that can be limited to a resource on `*`: it asks for the resource type. Actions that can change permissions (privilege escalation) and `iam:PassRole` get a warning or an error.

4.3 3. What starts it? (optional)

For ECS, Lambda and EC2: an Application Load Balancer, an EventBridge rule or Scheduler, an S3 event or an SQS queue. The builder then makes the permission that goes with it: a `lambda:InvokeFunction` permission with `SourceArn` and `SourceAccount`, a role for the Scheduler or the rights to read the queue.

5. From code or from logs

5.1 + From code

For a new system this is the exact route. Paste the code that calls AWS; the actions it uses are added, with their resource type. You fill in the names yourself.

- Python with boto3 (clients, resources and paginators)
- JavaScript SDK v3 (commands) and v2
- AWS CLI, also `aws s3 cp` and `sync` with the buckets they name

The code can also hold calls that never run. So review the result.

5.2 + From logs

For a system that already runs. Paste CloudTrail events of the role (a log file with `Records`, the output of `aws cloudtrail lookup-events` or JSON lines), or a policy IAM Access Analyzer generated. The actions are added and the names are filled in from the ARNs.

```
aws cloudtrail lookup-events --max-results 50 \
  --lookup-attributes AttributeKey=Username,AttributeValue=<role-session> > events.json
```

> Logs only show what ran in that period. A monthly job or an error path is then missing. So combine the logs with the code.

6. Only a policy

Need permissions without a new role, e.g. for a user, a group, an existing role or an IAM Identity Center permission set? Click **+ Only a policy (no role)**. You then only get question 2, and the result is a customer managed policy (limit 6,144 characters).

7. The result

On the right you see per role what you get. The bar shows how full a document is against the IAM limit; errors and warnings are above it.

DOCUMENT	WHAT FOR
Trust policy	Who may assume the role (limit 2,048 characters)
Permissions	What the role may do, as an inline policy (limit 10,240 characters)
Other side	Across accounts: the bucket policy, queue policy, key policy or resource policy that lets the role in
Lambda permission	Who may start the function (ALB, EventBridge, S3)
Identity provider	The OIDC or SAML provider, once per account

Switch between **JSON** and **Terraform**. **Copy** and **Download** work per document; **All (.tf)** gives one Terraform file with all roles and policies.

8. Rolling it out

Roll out in a test account first and let the application run a full cycle. Validate the policies with IAM Access Analyzer.

```
aws accessanalyzer validate-policy --policy-type IDENTITY_POLICY \
  --policy-document file://my-role-permissions.json
aws iam create-role --role-name my-role \
  --assume-role-policy-document file://my-role-trust.json
aws iam put-role-policy --role-name my-role --policy-name my-role \
  --policy-document file://my-role-permissions.json
```

Or with Terraform:

```
terraform init
terraform plan
terraform apply
```

If the application gets an `AccessDenied`, the missing action is in the error message and in CloudTrail. Paste those events through **+ From logs** and the action is added.

9. Saving projects

Name the project and click **Save**. You get a readable JSON file with every choice. Put it in Git next to the Terraform code. Later you click **Open...**, pick the file and everything is back as it was. A file that does not come from the builder is refused; unknown or wrong values are ignored.

> The project file holds account IDs, role and resource names. Keep it where the code is, not in a public repository.

10. Safety and limits

- **No AWS access:** the builder asks for no keys, calls no AWS API and sends nothing out.
- **Public sources:** the actions, resource types and ARNs come from the AWS Service Authorization Reference; the conditions follow the IAM user guide and each service's own guide.
- **The minimum is a starting point:** the curated lists cover about 50 services; elsewhere it is an estimate. Code and logs give the exact picture.
- **A starting point, not a guarantee:** review the permissions, test in a test account and validate with IAM Access Analyzer before you roll out.

Questions? Email ralph@rkpc.be or visit rkpc.be.