IDENTITY & ACCESS MANAGEMENT / AWS

AWS TEAM: Temporary Elevated Access Management with IAM Identity Center

Just-in-time admin access closes one of the most common — and most avoidable — gaps in AWS security. Here's what TEAM is, and a full walkthrough of implementing it.

Published July 2026·~16 min read·Implementation guide

Executive Summary

Most AWS environments accumulate standing (always-on) administrative access over time — IAM users and roles with broad, permanent permissions that are used rarely but remain active every day. This is consistently one of the highest-impact findings in security reviews: it expands blast radius, complicates audit, and creates risk that is disproportionate to the actual business need, since genuine admin activity is usually infrequent and time-bound.

AWS TEAM (Temporary Elevated Access Management) is an open-source solution, built by AWS, that replaces standing privileged access with a request-approve-time-bound-revoke workflow, layered directly on top of **IAM Identity Center** — the AWS-native service most organizations already use for workforce access. Engineers request elevated access to a specific account and permission set, for a specific reason, for a limited time. An approver signs off. Access is granted only for the approved window, then automatically revoked. Every step is logged.

This isn't a new access-control model bolted on top of AWS — it's a workflow layer that uses IAM Identity Center's existing account assignment mechanism, meaning it fits into environments that already use Identity Center for SSO without requiring a parallel identity or permissions system. It's self-hosted (deployed into your own AWS account via CloudFormation/Amplify), free to use, and built on serverless components, so the ongoing cost is typically low.

WHAT IT SOLVES
Standing privileged access — replaces permanent admin roles with time-bound, approved, audited access
BUILT ON
IAM Identity Center account assignments, Amplify, AppSync, Cognito, DynamoDB, Lambda, CloudTrail Lake
DEPLOYMENT MODEL
Self-hosted in a dedicated delegated administrator account — open source, no licensing cost
TYPICAL EFFORT
Half a day to a day for a first working deployment; longer to fully integrate policies and onboard teams
KEY OUTCOME
Elevated access becomes the exception, not the default — with a full audit trail via CloudTrail Lake

01 The Problem: Why Standing Access Is a Risk

Every permanent administrative credential is a standing liability, whether or not it's ever misused. It's a target for compromise, a variable in every access review, and a permanent expansion of blast radius that has to be accounted for in every security control you build afterwards — MFA policy, session monitoring, conditional access, the lot.

The uncomfortable truth in most environments is that genuine need for elevated access is *infrequent*: a production incident, a one-off migration task, a quarterly access review. Yet the access itself is almost always provisioned as permanent, because time-bound access has historically been operationally painful to implement — someone has to remember to grant it, remember to revoke it, and keep a record of why it was needed.

AWS TEAM exists specifically to remove that operational friction, so that the default posture can shift from "always on, just in case" to "granted on request, for a reason, for a limited time."

---

02 What Is AWS TEAM?

TEAM is an open-source reference solution published by AWS that layers a request-and-approval workflow on top of IAM Identity Center's native account assignment capability. Rather than provisioning a permanent permission set assignment, TEAM provisions it **temporarily** — for exactly as long as an approved request is active — and removes it automatically when the window closes or the requester ends the session early.

The solution's full documentation — architecture diagrams, configuration reference, and FAQs — is maintained at aws-samples.github.io/iam-identity-center-team, and is worth keeping bookmarked alongside this guide as the authoritative source when the solution changes.

It's designed around four personas, each with a distinct view of the same system:

	ROLE IN THE WORKFLOW
Requester	Requests elevated access to a specific account and permission set, with a business justification and a requested duration
Approver	Reviews pending requests against eligibility policy and approves or rejects them
Auditor	Has read-only visibility into all requests, approvals, and session activity for compliance and review purposes
Admin	Configures eligibility policies, approval policies, and manages the TEAM deployment itself

Key capabilities:

- **Time-bound access** — every grant has a defined start and end time, enforced automatically via IAM Identity Center account assignment creation and removal.
- **Configurable approval workflows** — eligibility policies define who can request access to what, and whether a request needs approval, self-approval, or is auto-approved.
- **Native AWS access portal integration** — once granted, access shows up in the same AWS access portal (SSO start page) that users already use, with no separate tooling to invoke the session.
- **Full audit trail** — every request, approval, activation, and revocation is logged, and session-level API activity is captured via a dedicated CloudTrail Lake event data store.
- **Self-service end** — requesters can end an active session early once the task is complete, rather than leaving access open for the full approved window.

---

03 Architecture Overview

TEAM deploys as a standalone web application (via AWS Amplify) into a **dedicated delegated administrator account** for IAM Identity Center — deliberately separate from your IAM Identity Center management account, so the TEAM application itself never has standing access to your identity source. It authenticates users via SAML against IAM Identity Center, and drives temporary account assignments through the IAM Identity Center APIs.

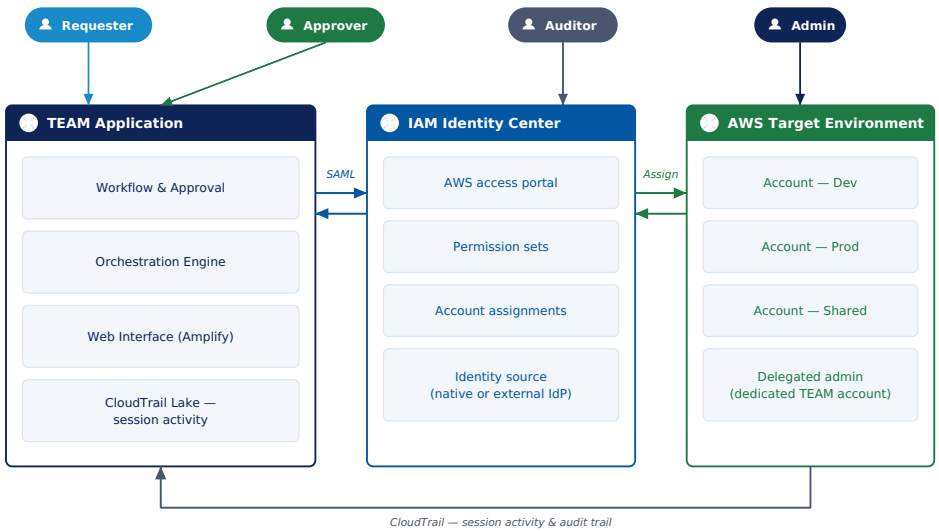

Figure 1 — TEAM sits between your workforce and IAM Identity Center, provisioning temporary account assignments on approval and logging every session via CloudTrail Lake.

At a high level, the day-to-day workflow looks like this:

1. A **Requester** signs in to TEAM (via the same IAM Identity Center identity used elsewhere) and submits a request: target account, permission set, justification, duration.
2. TEAM evaluates the request against configured **eligibility policy** — if it requires approval, it's routed to an **Approver**; if auto-approved, it proceeds immediately.
3. On approval, TEAM calls the IAM Identity Center APIs to create a **temporary account assignment** for the requester, scoped to the requested account and permission set.
4. The requester accesses the account exactly as they normally would — via the AWS access portal — for the duration of the approved window.
5. At expiry (or when the requester ends the session early), TEAM removes the account assignment, revoking access automatically.
6. All activity — the request, approval, session start/end, and the API calls made during the session — is recorded, with session activity captured via CloudTrail Lake for later review by an Auditor.

---

04 Prerequisites

Before starting deployment, make sure the following are in place:

- **AWS Organizations with IAM Identity Center enabled**, using either the native identity store or an external IdP (e.g. Okta, Entra ID) as the identity source.
- **A dedicated AWS account** to host TEAM, registered as a **delegated administrator** for IAM Identity Center. Don't deploy TEAM into your management account or an existing shared services account — isolating it limits the blast radius of the TEAM application itself.
- **Permission sets already created** in IAM Identity Center for the access levels you intend to make available via TEAM (e.g. `AdministratorAccess`, `PowerUserAccess`, or custom

permission sets).

- **An organization-level CloudTrail Lake event data store** configured to capture management and data events across the accounts TEAM will grant access to — this is what powers session-level audit visibility.
- **Two IAM Identity Center groups** for TEAM's own access control: a **TEAM administrators** group and a **TEAM auditors** group, with the relevant individuals added.
- **A local or CloudShell development environment** with the AWS CLI v2, Git, `git-remote-codecommit`, Node.js, and `jq` installed, plus an AWS CLI named profile with administrator access to the delegated admin account.

> **Worth noting.** TEAM is an AWS open-source sample solution (published under the `aws-samples` GitHub organization), not an official AWS service with SLAs or support. Treat it the way you would any self-hosted open-source security tooling: review the code, test thoroughly in a non-production organization first, and assign clear ownership for keeping it patched and maintained. The official documentation site is the best place to confirm current prerequisites and deployment steps before you start, since this guide reflects a point-in-time snapshot.

---

05 Step-by-Step Implementation

The following walks through a first deployment end-to-end, based on the current AWS TEAM deployment process.

1

Set up your delegated administrator account

In your IAM Identity Center management account, register the dedicated account you've set aside as a **delegated administrator** for IAM Identity Center. This lets TEAM manage account assignments without needing management-account-level access itself.

```
# From the management account
aws organizations register-delegated-administrator \
  --account-id <TEAM_ACCOUNT_ID> \
  --service-principal sso.amazonaws.com
```

2

Clone the TEAM repository

From your development environment, authenticated against the delegated admin account:

```
git clone https://github.com/aws-samples/iam-identity-center-team.git
cd iam-identity-center-team
```

3

Configure deployment parameters

Edit `parameters.sh` to set the values for your environment — at minimum, your AWS CLI profile name, target region, IAM Identity Center instance ARN, identity store ID, and the TEAM administrator/auditor group IDs created during prerequisites.

	PURPOSE
PROFILE	Named AWS CLI profile for the delegated admin account
REGION	AWS Region for deployment
IDC_INSTANCE_ARN	ARN of your IAM Identity Center instance
IDENTITY_STORE_ID	Identity store ID backing IAM Identity Center
ADMIN_GROUP_ID / AUDITOR_GROUP_ID	Group IDs for the TEAM administrator and auditor groups

4

Run the initialization script

This provisions the foundational resources TEAM needs before the main deployment — including a CodeCommit repository that the Amplify app will build from.

```
chmod +x init.sh deploy.sh
./init.sh
```

5

Run the deployment script

This is the main deployment step — it provisions the Amplify application, AppSync API, Cognito user pool, DynamoDB tables, and the Lambda functions that drive the workflow engine. Expect this to take around 20 minutes.

```
./deploy.sh
```

Once complete, the script outputs the Amplify app URL — this is the web interface your users will access TEAM through.

6

Integrate with IAM Identity Center (SAML)

TEAM authenticates users via a SAML 2.0 connection to IAM Identity Center, configured as a custom SAML application. Run the integration helper script to retrieve the values you'll need:

```
./integration.sh
```

This outputs the **Application Start URL**, **ACS URL**, and **SAML Audience**. In the IAM Identity Center console, create a new custom SAML 2.0 application, enter these values, configure attribute mapping (mapping the subject to each user's email or username), download the IdC-provided SAML metadata, and assign the users or groups who should have access to TEAM.

7

Configure the Cognito user pool

Back in your deployment environment, populate `details.json` with the SAML metadata URL from the IAM Identity Center application you just created, then run:

```
./cognito.sh
```

This wires the Cognito user pool (which fronts the Amplify app) to trust SAML assertions from IAM Identity Center, completing the authentication chain: user → IAM Identity Center → SAML → Cognito → TEAM application.

8

Define eligibility and approval policies

Sign in to the TEAM application as an Admin (using an account in your TEAM administrators group) and configure:

- **Eligibility policies** — which users or groups can request access to which accounts and permission sets, and the maximum duration they can request.
- **Approval policies** — whether a given eligibility requires approval, self-approval, or is auto-approved, and who the eligible approvers are.

Start narrow — a small number of well-understood eligibilities (e.g. a specific ops team, a specific account, a capped duration) — and expand coverage once the workflow is proven.

9

Run an end-to-end test

Before rolling out broadly, walk through the full cycle yourself: submit a request as a test Requester, approve it as a test Approver, confirm the account assignment appears in the AWS access portal, use it briefly, end the session, and confirm the assignment is removed and the activity appears in CloudTrail Lake.

06 Security & Operational Considerations

Isolate the TEAM account. Keep it as a dedicated delegated administrator account, separate from other workloads — it holds a sensitive amount of trust over account assignments across your organization.

Enforce MFA for TEAM sign-in. Since TEAM can grant elevated access, authentication into TEAM itself should be held to at least the same standard as the access it grants.

Keep the management account out of scope. TEAM should never be configured to grant access to the IAM Identity Center management account itself — that account should stay under tighter, separately governed controls.

Plan for break-glass. TEAM is a dependency for accessing other accounts — make sure you retain a documented, separately secured emergency access path in case TEAM itself is unavailable.

Review session durations regularly. Default to the shortest duration that's realistic for the task, and revisit eligibility policies periodically rather than treating initial configuration as permanent.

Treat it as open-source software you own. Track the upstream repository for updates and security fixes, and assign clear internal ownership for patching the deployment over time.

07 Cost Considerations

TEAM is free and open source — there's no licensing cost — but it provisions billable AWS resources. Since almost everything is serverless (Amplify, AppSync, Lambda, DynamoDB, Cognito), cost scales with usage and is typically modest for most organizations. The main cost variable to plan for is the **CloudTrail Lake event data store**, since ingestion and storage costs scale with event volume across the accounts in scope — it's worth right-sizing retention and event selectors deliberately rather than capturing everything by default. Run the components through the AWS Pricing Calculator against your expected usage before rolling out broadly.

08 Summary

Standing privileged access is a persistent, quietly compounding risk in most AWS environments — not because anyone decided it should be that way, but because time-bound access has historically been more operationally painful than leaving things permanently open. AWS TEAM removes that friction by layering a proper request-approve-expire workflow

directly on IAM Identity Center, so temporary access becomes the easy default rather than the exception that never gets built.

It's not a large lift to deploy — a working first version can be up in well under a day — but the eligibility and approval policy design deserves real thought, since that's what determines whether the workflow actually earns trust from the teams using it. Start with a narrow, well-understood use case, prove the end-to-end flow, and expand from there.

Reference guide based on the AWS Security Blog post "Temporary elevated access management with IAM Identity Center" and the official TEAM documentation, as of July 2026. Verify current deployment steps against the upstream documentation before implementation, as the solution may have changed since this was written.