pwc

From over-privileged to Just-in-Time— Revolutionizing AWS CI/CD access control

The challenges of traditional static IAM roles for CI/CD pipelines

Most organizations still rely on shared or common IAM roles across their CI/CD pipelines for automating their cloud deployments, often granting broad permissions to multiple users who trigger these pipelines. Over time, these roles accumulate excessive privileges, creating hidden risks and violating the principle of least privilege and also access tracking becomes difficult.

Without Just-in-Time access or automated guardrails, pipelines continue to operate with static roles and elevated rights that are rarely reviewed or updated. This lack of fine-grained, temporary time-bound access can expose organizations to potential misuse, credential leakage, and compliance gaps proving that static IAM roles can't keep pace with today's dynamic delivery environments.

How JIT access solves IAM challenges

In most cloud environments, users are the ones who initiate or indirectly trigger CI/CD pipelines, whether manually or through GitOps workflows. This means that each user carries specific responsibility and entitlement tied to the actions they perform in the particular AWS environment. To confirm secure and accountable operations, access control should be centered around these users rather than static pipeline permissions.

The solution builds on this principle by introducing Just-in-Time (JIT) access for deployments, allowing pipelines to dynamically assume IAM roles having custom permissions that the triggering users are entitled to use. This approach ensures that permission remains tightly aligned with individual responsibilities, reducing risks from over-provisioned access. Combined with persona-based access and centralized identity through AWS IAM Identity Center, it establishes a scalable and auditable framework for managing access across AWS environments.

This solution combines three core design pillars:

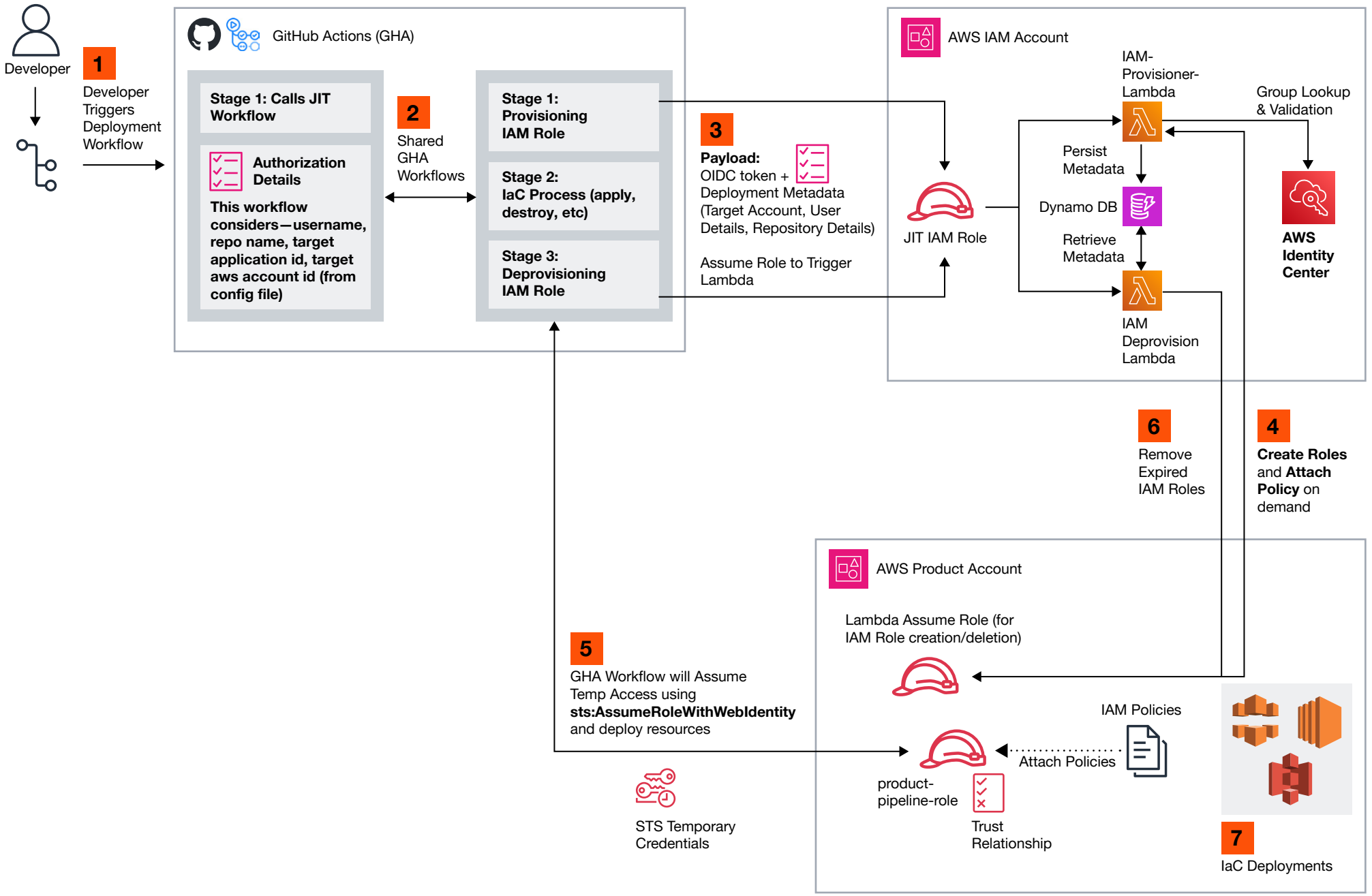
Persona-based access— Identify and group users into logical personas that align with the company's operating model and functional responsibilities.

Centralized identity through AWS IAM Identity Center— Point of trust for managing user groups (based on user personas) for validating deployments across the AWS organization.

Just-in-Time (JIT) access for infrastructure deployments — Temporary IAM roles created based on validated user personas and made available for assumption to CI/CD pipelines.

Together, they create a scalable, auditable, and secure access framework for AWS deployments via CI/CD pipelines.

Architecture:



The numbered steps shown in the diagram are explained in detail below:

1	User commits the IaC code to GitHub repository
2	GitHub Repo integration with GitHub actions triggers the Developer workflow
3	Central (Reusable) JIT Access Workflow is triggered by the Developer workflow passing authorization details (target account, repository name, user details, etc.)
4	A. JIT Workflow assumes JIT Lambda Role B. JIT Lambda Role invokes the IAM Provisioning Lambda passing authorization details (target account, repository name, user details, etc.)
5	IAM Provisioning Lambda performs RBAC validation ensuring that the user is a member of the Product Deployers Group
6	IAM Provisioning Lambda creates product-pipeline-role and creates temporary session credentials
7	Temporary session credentials are passed back to the JIT Workflow by the IAM Provisioning Lambda
8	A. JIT Workflow authenticates using temporary session credentials and assumes the newly created product-pipeline-role B. IaC is deployed on behalf of the developer using the temporary product-pipeline-role
9	A. IAM Deprovisioning Lambda is invoked after IaC deployment is complete B. IAM Deprovisioning Lambda deletes the temporary product-pipeline-role

Persona-based access model

Access is structured around pre-defined personas that reflect job functions such as Cloud Admin, Security, Billing, Product, etc. This model enforces least-privilege principles by mapping permissions directly to responsibilities, ensuring consistency across AWS accounts and simplifying ongoing access management.

Centralized access with AWS IAM Identity Center

Once personas are defined, the next step is centralizing access. AWS IAM Identity Center (formerly AWS SSO) becomes the core management layer for managing user groups and access across AWS environments. Its primary purpose is to organize users into logical groups based on their personas or job functions, enabling consistent, scalable permission management. By integrating with external identity providers such as Microsoft Entra ID, it allows seamless single sign-on (SSO) and unified authentication across multiple AWS accounts and applications.

Beyond access management, IAM Identity Center also acts as a validation layer, ensuring that only approved groups and users can assume designated roles or deploy through pipelines. This centralized approach not only simplifies governance but also strengthens compliance by continuously validating that access aligns with defined personas and least-privilege principles.

Just-in-Time (JIT) access for IaC deployments

For all automated deployments across AWS environments, **JIT access** must be enabled. In JIT process, pipeline roles are dynamically provisioned based on the user's personas and time-bound access requirements, enforcing least-privilege principles. This approach not only strengthens security but also ensures that all deployments pass through controlled, auditable, and security-scanned workflows.

Overview of group creation in AWS Identity Center

Group creation in AWS Identity Center is a fundamental process for managing user access and permissions across AWS accounts. Identity Center must be housed in a central identity account where administrators can create groups to logically organize users with similar access needs, enabling efficient assignment of permissions and policies.

There are multiple approaches that we can take to create and manage logical groups in the AWS Identity Center. In this blog, we are using the default Identity Center directory for managing groups and users.

Note: AWS Identity Center also supports external identity sources like Microsoft Entra ID, Okta, etc. You can manage user and group creation externally and sync it with Identity Center via SCIM.

Group Creation Logic

For each AWS account x persona, create a dedicated Identity Center group for housing valid users:
Naming: Company_Org_AccountId_Persona
(e.g. PwC_GP_1234532121_Product_Admin).

User-group-based personas

Access to AWS environments will be based on personas that tie back to high-level job responsibilities of the users. Below are the key personas that will be leveraged to grant permissions to AWS accounts. These permission mappings will be maintained as a metadata in the Lambda functions which creates JIT IAM roles across AWS accounts.

Note: We can segregate these permissions based on environment (i.e. production environment will have fine-grained access compared to non-prod environments).

For this blog we are considering below user personas:

	AWS Managed Policy	Custom Policy
Network Persona	• AWSNetworkManagerFullAccess • AWSShieldDRTAccessPolicy • AWSSupportAccess • CloudFrontFullAccess • NetworkAdministrator • SecretsManagerReadWrite	• Customized policy statements as per organizations' requirements
Product Admin Persona	• PowerUserAccess	• Customized policy statements as per organizations' requirements
Container Admin Persona	• AmazonEKSClusterPolicy • AmazonVPCReadOnlyAccess • IAMReadOnlyAccess • AWSKeyManagementServicePowerUser • AWSCloudTrail_ReadOnlyAccess • SecretsManagerReadWrite	• Customized policy statements as per organizations' requirements

Overview of Lambda functions

Provisioning Lambda

IAM provisioning Lambda creates product-pipeline-role and creates temporary session credentials.

1. Temporary session credentials are passed back to the JIT workflow by the IAM provisioning Lambda.
2. JIT workflow authenticates using temporary session credentials and assumes the newly created product-pipeline-role.
3. IaC is deployed on behalf of the developer using the temporary pipeline-role.

Core functionalities of Lambda

Input validation » User authentication » Authorization check » Temporary role creation » Success or failure tracking

Key inputs

- email: User's email address for Identity Center lookup
- branch: Git branch or deployment context
- target_aws_account: Target AWS account name
- target_aws_account_id: Target AWS account ID
- group_type: Permission group type (default: "Product-Deployers")

Key outputs

- transaction_id: Unique ID for tracking this provisioning request
- iam_role_arn: ARN of created temporary role
- role_name: Name of created role
- role_ttl_hours: Role expiration time (default: 24 hours)
- status: "provisioned" when successful

Input validation and processing

- Receives user email, branch, target account, and group type
- Generates unique transaction ID for tracking
- Validates all required fields are present

User authentication and authorization

- **Identity center integration:** Looks up user by email in AWS SSO
- **Group membership check:** Verifies user belongs to required deployment group
- **Account-specific authorization:** Ensures user can deploy to the requested AWS account

Transaction tracking user

- **DynamoDB storage:** Records all provisioning requests with status tracking
- **Audit trail:** Maintains complete history of who requested what access when
- **Status management:** Updates transaction status (processing » provisioned/failed)

Cross-account role creation

- **Temporary IAM roles:** Creates time-limited roles (default 24 hours TTL)
- **Policy attachment:** Applies appropriate permissions based on group type
- **Trust relationships:** Configures roles to be assumable only by authorized pipeline roles

Deprovisioning Lambda

This Lambda function removes temporary IAM roles created by the provisioning Lambda, completing the JIT access lifecycle.

Core-components flow

Input validation » Transaction lookup » Cross-account role assumption » Policy detachment » Role deletion » Deprovisioning record creation

Key inputs (consumes from provisioning Lambda)

- transaction_id: Original provisioning transaction ID
- target_aws_account: Account where role was created
- role_arn: Full ARN of the temporary role to delete

Key outputs

- deprovisioning_transaction_id: New transaction ID for cleanup record
- status: “deprovisioned” when successful
- original_transaction_id: Reference back to provisioning request

Transaction retrieval

- Fetches original provisioning transaction from DynamoDB using transaction ID
- Extracts role details and target account information

IAM role cleanup

- Assumes cross-account role in target AWS account
- Detaches all managed and inline policies from the temporary role
- Deletes the IAM role completely

Deprovisioning record creation

- Creates new deprovisioning transaction in DynamoDB
- Preserves all original transaction data for audit continuity
- Links back to original provisioning transaction ID

Conclusion

Combination of JIT, user personas, and centralized IAM Identity Center play a pivotal role in granting dynamic access to CI/CD pipelines as it ensures least privilege, user entitlements and auditable IAM access. This solution can provide a modern serverless-based architecture enabling scalable and resilient IAM activities supporting large enterprises at scale.

As the technology world started to move towards Zero Trust architecture, it is high time organizations move out of static credentials for their CI/CD and users accessing AWS environments and embrace dynamic verified IAM access with least privilege and short-term access.

References

- Implementing just-in-time privileged access to AWS with Microsoft Entra and AWS IAM Identity Center
- Aligning IAM policies to user personas for AWS Security Hub