PLATFORM · TEMPLATE

Killing Long-Lived Credentials with Workload Identity and Dynamic Secrets

KAANSYSTEMS.COM/LIBRARY/KILLING-LONG-LIVED-CREDENTIALS · AUGUST 8, 2026

ABOUT THIS TEMPLATE

Replace stored AWS keys, CI secrets, and database passwords with OIDC federation, IRSA, and dynamic secrets: a phased migration plan for regulated SMBs.

THE TEMPLATE

The migration checklist below is phased so that each phase ends with a revocation, not a deployment. A phase is not done while the old credential still works.

```
# No-Long-Lived-Credentials Migration

## Phase 0: Inventory (week 1)
- [ ] Run `aws iam generate-credential-report`; export all access keys with age and last-used
- [ ] Enumerate CI secrets: GitHub org, repo, and environment secrets
- [ ] Grep deploy artifacts for embedded creds: task definitions, Helm values, compose files, .env
- [ ] gitleaks/trufflehog across full git history of all active repos
- [ ] List kubeconfigs with embedded client certs; note expiry (often: none)
- [ ] Build the register: credential | system | owner | blast radius | last used
- [ ] Record the baseline count; this number is the program KPI

## Phase 1: Workloads (weeks 2-4)
- [ ] Register cluster OIDC provider in IAM (IRSA) or enable the EKS Pod Identity agent
- [ ] One IAM role per service; trust pinned to namespace + service account
- [ ] Migrate services off node-role and env-var creds; verify AssumeRoleWithWebIdentity in CloudT
- [ ] Delete AWS credentials from Kubernetes Secrets and task-definition env vars

## Phase 2: CI/CD (weeks 3-5)
- [ ] Create the token.actions.githubusercontent.com OIDC provider in IAM
- [ ] Deploy role trust pinned to repo + branch (JSON below); separate plan-only role for PRs
- [ ] Cut workflows over to aws-actions/configure-aws-credentials with role-to-assume
- [ ] Deactivate, then delete, the CI IAM user and its stored repo secrets

## Phase 3: Humans (weeks 4-6)
- [ ] IAM Identity Center connected to the IdP; permission sets replace per-user policies
- [ ] CLI access via `aws sso login`; session duration 12h or less
- [ ] Create 2 break-glass IAM users: hardware MFA, keys stored offline, CloudTrail alarm on use
- [ ] Flip all remaining IAM user keys Inactive for 7 days, then delete the users

## Phase 4: Data stores (weeks 5-8)
- [ ] RDS IAM auth for low-churn services (Postgres/MySQL/Aurora)
- [ ] Secrets Manager rotation (managed Lambda, alternating users) for the rest
- [ ] Apps fetch DB credentials at connect time from Secrets Manager, never from env vars
- [ ] Rotate every DB password once through the new path to invalidate leaked history

## Phase 5: Third parties (ongoing)
- [ ] Per vendor key: minimum scope, Secrets Manager entry, named owner, 90-day rotation
- [ ] Usage alerting per key (API audit log or billing anomaly)
- [ ] Fold key-rotation posture into vendor risk reviews

## Steady state
- [ ] Static-credential count published weekly (target: break-glass plus vendor keys, nothing else
- [ ] CI check blocking AKIA* and other known key formats in commits
- [ ] SCP denying iam:CreateUser and iam:CreateAccessKey outside the break-glass path (SCPs never
```

The GitHub Actions trust policy, with the conditions that matter:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/token.actions.githubusercontent.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "token.actions.githubusercontent.com:aud": "sts.amazonaws.com",
          "token.actions.githubusercontent.com:sub": "repo:your-org/your-repo:ref:refs/heads/main"
        }
      }
    }
  ]
}
```

Never widen the `sub` condition to `repo:your-org/*` on the deploy role. If several repos genuinely need deploy access, give each its own statement or its own role.

HOW TO USE IT

Revocation is the real test

If revoking a particular credential feels frightening, you have located a system whose ownership and issuance path nobody understands. That fear is the finding; write it down before you fix it. The parallel-run pattern exists to convert fear into boredom: the new path serves traffic for a week while the old credential sits deactivated but recoverable. Apply the same discipline to break-glass. Twice a year, actually open the safe, actually use the offline key, and confirm the alarm pages someone. A break-glass procedure that has never been exercised is a decorative one.

The audit trail arrives for free

Every `AssumeRole` and `AssumeRoleWithWebIdentity` call is a CloudTrail event binding an action to an identity, a session, and a timestamp. Once credentials are issued rather than stored, your rotation control is satisfied structurally: the auditor question "how do you rotate credentials?" gets the answer "they expire hourly, and issuance is logged." That is a materially stronger position for SOC 2 CC6.1 than a spreadsheet of quarterly rotations, and it feeds directly into evidence pipelines, because the proof is a queryable log rather than a screenshot someone remembers to take.

Don't chase the last three

The static-credential count follows an asymptote. It falls fast through the dead keys and CI secrets, slower through data stores, and then lands on a small stubborn number: break-glass plus the vendors that will not do OIDC. Accept the asymptote and guard the perimeter instead. An SCP denying new access-key creation means the count cannot silently climb back up, and a CI secret-format check means the next leaked key is caught at commit time. The durable win is cultural: engineers who joined after the migration have never configured a static AWS key and would not know where to put one. At that point the migration has stopped being a project and become the default, which is the only finish line that holds.