Cloud Workload Identity Compared: AWS IRSA vs GCP Workload Identity Federation vs Azure Managed Identity vs SPIFFE/SPIRE

Machine Identity / Last updated 2026-06-11 / https://www.scrambleid.com/learn/cloud-workload-identity-compared

In one sentence: AWS IAM Roles for Service Accounts (IRSA), GCP Workload Identity Federation, Azure Managed Identity, and SPIFFE/SPIRE all solve the same fundamental problem (eliminating long-lived shared secrets for workloads) with different surfaces and operational characteristics; the right choice for any given workload is shaped by where the workload runs, how cross-cloud the architecture is, and how much standardization across environments matters.

TL;DR (canonical)

- **Same fundamental architecture, different surfaces.** Each mechanism attests workload identity at the runtime level (cluster, service account, projected token) and issues short-lived credentials.
- **AWS IRSA** for EKS-based workloads in AWS. The default choice for in-AWS workloads. IAM Roles Anywhere for off-cloud workloads needing AWS access.
- **GCP Workload Identity Federation** for general external-to-GCP federation. Workload Identity for GKE for the in-GKE specifics.
- **Azure Managed Identity** for Azure resources. System-assigned (lifecycle tied to resource) or user-assigned (independent lifecycle).
- **SPIFFE/SPIRE** for cross-cloud and heterogeneous environments. Reference implementation of the SPIFFE specification.
- **Choose by workload location, cross-cloud needs, and operational maturity.** Single-cloud → cloud-native. Multi-cloud → SPIFFE/SPIRE plus cloud-native per cloud.
- **Compose with sender-constrained tokens for cross-boundary paths.** Cloud-native handles the workload-to-cloud-IAM hop; sender-constrained tokens (mTLS or DPoP) handle the workload-to-other-service-or-tenant hop.

What cloud workload identity solves

Long-lived service-account credentials in cloud environments are particularly leak-prone:

- Container images bake them in.
- Helm charts and Terraform configs reference them.
- CI/CD pipelines load them into runners.
- Developers debug with them locally.
- Diagnostic dumps include them.
- A single misconfigured S3 bucket exposes them.

Cloud workload identity removes the long-lived credential entirely. The workload's runtime context (the cluster it runs in, the Kubernetes service account it operates under, the OIDC token issued by the cluster's identity provider) becomes the basis for short-lived credential issuance. There is nothing to bake in, copy, or leak.

AWS: IAM Roles for Service Accounts (IRSA) and Roles Anywhere

IRSA for EKS workloads

The flow:

1. EKS clusters have an associated OIDC identity provider configured in IAM.
2. Pods are configured with a Kubernetes service account that is annotated with an IAM role ARN.
3. The pod receives a projected service-account token (a JWT signed by the cluster's OIDC provider).
4. The AWS SDK in the pod calls STS `AssumeRoleWithWebIdentity`, presenting the projected token.
5. STS validates the token (issuer matches the cluster's OIDC provider, audience matches `sts.amazonaws.com`, subject matches the configured pattern), and issues short-lived AWS credentials.
6. The pod uses the credentials to call AWS services. Credentials expire and are automatically refreshed.

What you get: workload-bound, short-lived AWS credentials with no long-lived secret stored anywhere. Each pod has its own scoped credentials.

What it covers: EKS pods accessing AWS services (S3, DynamoDB, KMS, etc.).

What it doesn't cover: workloads outside AWS that need AWS access (use IAM Roles Anywhere), or AWS workloads that need access to non-AWS services (use OIDC federation or sender-constrained tokens).

IAM Roles Anywhere for off-cloud workloads

For workloads outside AWS (on-prem, other clouds) that need AWS access, IAM Roles Anywhere uses an X.509 certificate as the workload's identity. The certificate is issued by a trusted CA

configured as a trust anchor in AWS. The workload signs a request with its certificate; AWS validates against the trust anchor and issues short-lived credentials.

What you get: AWS access from non-AWS workloads without storing long-lived AWS keys.

What it covers: any workload with a TPM/HSM-issued certificate (or a certificate from a CA you trust).

What it doesn't cover: workloads without a PKI (use OIDC federation if your environment supports OIDC tokens).

STS AssumeRole with web identity for federated workloads

For external OIDC-issuing identity providers (GitHub Actions, GitLab CI, third-party IDPs), the same `AssumeRoleWithWebIdentity` STS call works. The trust relationship is configured between the AWS account and the external OIDC provider's issuer URL.

Use case: GitHub Actions workflows accessing AWS without long-lived AWS keys.

Google Cloud: Workload Identity Federation and GKE Workload Identity

Workload Identity Federation (general)

GCP's Workload Identity Federation lets external workloads (other clouds, on-prem, OIDC-issuing identity providers) authenticate to GCP without storing long-lived service-account keys. The flow:

1. A Workload Identity Pool is configured in GCP, with provider entries for the external identity sources (AWS via OIDC, GitHub Actions, generic OIDC, SAML).
2. The external workload authenticates to its native identity (an AWS IAM role's STS token, a GitHub Actions OIDC token, etc.).
3. The workload calls Google's STS endpoint (`sts.googleapis.com`) to exchange the external token for a federated GCP credential.
4. The federated credential can directly access GCP resources, or be used to impersonate a GCP service account for additional capabilities.
5. Credentials are short-lived (≤ 1 hour by default).

What you get: GCP access from any workload that can present a valid OIDC or AWS-STS token; no long-lived service-account JSON keys.

What it covers: external-to-GCP federation, multi-cloud workloads needing GCP access.

Workload Identity for GKE

For workloads running in GKE specifically, Workload Identity for GKE is the in-cluster mechanism. Each Kubernetes service account is mapped to a GCP service account; pods receive credentials based on this mapping.

What you get: GKE pods accessing GCP services without service-account JSON keys mounted into containers.

What it covers: GKE pods accessing GCP services.

Microsoft Azure: Managed Identity and Workload Identity

Managed Identity for Azure resources

Azure Managed Identity is Azure's first-party identity for Azure resources. Two variants:

- **System-assigned managed identity:** lifecycle tied to the Azure resource (when the VM, App Service, Function, etc. is deleted, the identity is deleted with it).
- **User-assigned managed identity:** independent lifecycle, can be attached to multiple resources, can outlive any specific resource.

The Azure resource's runtime presents its managed identity to Azure AD (Microsoft Entra) via an internal mechanism (IMDS endpoint on VMs, environment-injected token on App Service); Entra issues a short-lived access token for the requested Azure resource.

What you get: Azure resources accessing Azure services without storing long-lived credentials.

What it covers: Azure-native resources accessing Azure-native services.

Microsoft Entra Workload Identity for AKS

For AKS-specific workloads, Microsoft Entra Workload Identity is the in-AKS mechanism. Each Kubernetes service account is mapped to an Entra application; pods receive Entra tokens via a projected service-account token.

What you get: AKS pods accessing Azure resources without storing client secrets.

Federated credentials for external OIDC

Entra also supports federated credentials, where an external OIDC issuer (GitHub Actions, GitLab CI, generic OIDC) is trusted to issue tokens that can be exchanged for Entra access tokens.

What you get: external workloads accessing Azure without storing Azure credentials.

SPIFFE/SPIRE: cross-cloud and heterogeneous environments

SPIFFE (Secure Production Identity Framework for Everyone) is an open specification for workload identity. SPIRE is the reference implementation. Together they provide workload identity across heterogeneous environments.

The key concept is the SPIFFE Verifiable Identity Document (SVID), which can be a JWT or X.509 certificate. SVIDs are issued by SPIRE based on attestable properties of the workload's runtime (which node it's on, which container, which Kubernetes service account, etc.).

What you get: unified workload identity across multiple clouds, on-prem, and edge environments. Workloads in AWS, GCP, Azure, and on-prem all receive SVIDs from the same logical SPIRE deployment.

What it covers: cross-cloud service-to-service, hybrid environments, edge-to-cloud, environments where cloud-native mechanisms would create separate parallel identity systems.

Operational complexity: SPIRE deployment, attestation policies, federation between SPIRE servers across environments. Materially more operational overhead than the single-cloud-native mechanisms.

Side-by-side comparison

Property	AWS IRSA	GCP WIF	Azure MI	SPIFFE/SPIRE
Primary scope	AWS workloads accessing AWS	External-to-GCP federation; GCP access	Azure-native resources accessing Azure	Cross-cloud, heterogeneous
Standards basis	OIDC (cluster-issued tokens), AWS-proprietary STS	OIDC, AWS-STS, SAML federation	Azure-AD-proprietary, OIDC for federated	SPIFFE specification (open)
Workload runtime support	EKS native; Roles Anywhere for off-cloud	GKE native; WIF for general federation	Azure-managed resources; AKS via Workload Identity	Any (Kubernetes, VMs, on-prem, edge)
Credential lifetime	Configurable, typically 1 hour	Configurable, typically 1 hour	Short-lived, automatic	SVID-configurable, typically minutes
Cross-cloud	Via Roles Anywhere or OIDC federation	Native (WIF is the federation mechanism)	Via federated credentials	Native (the design point)
Operational maturity	High; widespread production use	High; widespread production use	High; first-party in Azure	Growing; CNCF graduated; production at scale at hyperscale customers
Operational overhead	Low (managed by AWS)	Low (managed by GCP)	Low (managed by Azure)	Higher (deploy and operate SPIRE)
Ecosystem and tooling	Extensive AWS-native tooling	Extensive GCP-native tooling	Extensive Azure-native tooling	Open-source ecosystem; vendor-neutral
Best fit	AWS-native architectures	GCP-native and multi-cloud-with-GCP	Azure-native architectures	True multi-cloud with strong unification needs

Property	AWS IRSA	GCP WIF	Azure MI	SPIFFE/SPIRE
Common gap	Cross-org integrations beyond AWS	Cross-org integrations beyond GCP	Cross-cloud beyond Entra federation	Operational complexity for small deployments

How to choose

A practical decision flow:

1. **Where do my workloads run?**
   - Single cloud (AWS, GCP, or Azure): use that cloud's native mechanism. Done.
   - Multi-cloud with one dominant cloud: use each cloud's native mechanism for its workloads, plus federation for the cross-cloud paths.
   - True multi-cloud with no dominant: SPIFFE/SPIRE plus cloud-native per cloud.
2. **What cross-cloud or cross-org integration do I have?**
   - None: the cloud-native mechanism is sufficient.
   - Some, with one cloud dominant: federation from the secondary clouds to the dominant cloud's IAM.
   - Significant, with no dominant: SPIFFE/SPIRE as the unification layer.
3. **What's my operational maturity for managing identity infrastructure?**
   - Lean platform team: cloud-native mechanisms only. Avoid SPIRE operational overhead.
   - Mature platform team with multi-cloud reality: SPIRE is workable and worth the unification.
4. **What's my standards posture?**
   - Standards-first: SPIFFE/SPIRE is more standards-aligned (open specification, vendor-neutral).
   - Cloud-native-first: each cloud's native mechanism is the simplest path.
5. **What's my AI/agent identity story?**
   - Agents using cloud workload identity for primary: cloud-native is fine.
   - Agents needing chain-aware delegation, MCP server identity, or cross-tenant: layer ScrambleID's M2M control plane on top.

How ScrambleID composes

ScrambleID does not replace cloud workload identity; it composes with it.

The typical pattern in a cloud-native architecture:

1. Workloads use the cloud-native mechanism (IRSA, WIF, Managed Identity) for their primary identity to cloud resources.
2. For service-to-service paths that cross cloud boundaries, cross-tenant boundaries, or extend to AI agents and MCP servers, workloads exchange their cloud-native credentials for ScrambleID-issued tokens via RFC 7523 JWT client assertion.
3. The ScrambleID tokens are sender-constrained (mTLS or DPoP), short-lived (≤ 300 seconds), and audience-bound; for multi-hop scenarios the architecture follows the RFC 8693 Token Exchange pattern.
4. Cloud-native identity stays the workload-to-cloud-IAM control plane. ScrambleID is the cross-cloud, cross-tenant, agent-aware control plane on top.

For deeper coverage of the M2M control plane, see M2M Authentication Without Secrets and Sender-Constrained Tokens (mTLS, DPoP).

Configuration sketches

Concrete examples for each cloud, simplified for clarity. For production deployments, follow the cloud's full documentation.

AWS IRSA for EKS

```
# Kubernetes service account
apiVersion: v1
kind: ServiceAccount
metadata:
  name: my-app-sa
  namespace: my-namespace
  annotations:
    eks.amazonaws.com/role-arn: arn:aws:iam::123456789012:role/MyAppRole
```

The role's trust policy:

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {"Federated": "arn:aws:iam::123456789012:oidc-provider/oidc.eks.us-east-
1.amazonaws.com/id/EXAMPLED539D4633E53DE1B716D3041E"},
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {
      "StringEquals": {
        "oidc.eks.us-east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B716D3041E:sub":
"system:serviceaccount:my-namespace:my-app-sa",
        "oidc.eks.us-east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B716D3041E:aud":
"sts.amazonaws.com"
      }
    }
  }]
}
```

The pod runs with this service account; the AWS SDK auto-detects the projected token and assumes the role.

GCP Workload Identity Federation (from AWS)

```
# Create a workload identity pool
gcloud iam workload-identity-pools create my-pool \
  --location global

# Add an AWS provider
gcloud iam workload-identity-pools providers create-aws aws-provider \
  --workload-identity-pool my-pool \
  --location global \
  --account-id 123456789012

# Bind a service account to the pool
gcloud iam service-accounts add-iam-policy-binding \
  my-gcp-sa@my-project.iam.gserviceaccount.com \
  --role roles/iam.workloadIdentityUser \
  --member
"principalSet://iam.googleapis.com/projects/.../locations/global/workloadIdentityPools/my-
pool/attribute.aws_role/arn:aws:sts::123456789012:assumed-role/MyAwsRole"
```

The AWS workload then exchanges its STS token for a GCP credential via Google's STS endpoint.

Azure Managed Identity for App Service

```
# Enable system-assigned managed identity on an App Service
az webapp identity assign \
  --resource-group my-rg \
  --name my-app

# Grant the managed identity access to a Key Vault
az role assignment create \
  --assignee <managed-identity-object-id> \
  --role "Key Vault Secrets User" \
  --scope /subscriptions/.../resourceGroups/my-rg/providers/Microsoft.KeyVault/vaults/my-kv
```

The App Service runtime presents its managed identity to Entra; Entra issues an access token for Key Vault.

SPIFFE/SPIRE workload registration

```
# Register a workload selector
spire-server entry create \
  -spiffeID spiffe://example.org/ns/my-namespace/sa/my-app \
  -parentID spiffe://example.org/spire/agent/k8s_psat/cluster-a/<node-id> \
  -selector k8s:ns:my-namespace \
  -selector k8s:sa:my-app
```

The workload requests an SVID from the SPIRE agent; the agent attests the workload (Kubernetes pod selectors) and issues the SVID.

Common failure modes

1. **Trust-relationship misconfiguration.** The most common failure: AWS IRSA trust policy doesn't match the actual subject claim, GCP Workload Identity Pool provider conditions are too restrictive or too permissive, Azure federated credentials misconfigured. Test trust relationships during deployment, not after.
2. **Audience-claim mismatches.** Workloads expect `sts.amazonaws.com`, the cluster issues a different audience. Validate the `aud` claim on tokens.
3. **Trust-anchor management for IAM Roles Anywhere.** The CA certificates used as trust anchors must be managed (rotation, revocation). A compromised CA is a compromised identity.
4. **GKE Workload Identity not bound correctly.** The mapping between Kubernetes service account and GCP service account requires both sides to be configured; a one-sided configuration leaves the workload broken.
5. **Cross-region token caching.** Workloads in one region using cached tokens issued for another region can fail authentication or audit.
6. **SPIRE deployment complexity.** Multi-cluster SPIRE with federation requires careful design. Many SPIRE failures are operational (server downtime, attestation policy gaps) rather than fundamental.

Standards alignment

Standard	Relevance
OAuth 2.0 / OIDC	Underlying protocols for federation
RFC 7523	JWT client assertion (used in some federation flows)
SPIFFE specification	Cross-environment workload identity model
SPIFFE Federation API	Trust relationships across SPIRE deployments

Standard	Relevance
NIST SP 800-207	Zero Trust applied to non-human identity
FedRAMP IA controls	Authentication for federal-facing CSPs

Key Takeaway

Cloud workload identity is a class of mechanisms where a workload running in a cloud environment authenticates to the cloud's IAM (and to external services) without holding any persistent shared secret. The dominant mechanisms: AWS IAM Roles for Service Accounts (IRSA) for EKS workloads; AWS IAM Roles Anywhere for off-cloud workloads needing AWS access; GCP Workload Identity Federation for general external-to-GCP federation and Workload Identity for GKE for in-cluster workloads; Azure Managed Identity (system-assigned and user-assigned) for Azure resources, plus Microsoft Entra Workload Identity for AKS and federated credentials for external OIDC issuers; SPIFFE/SPIRE for cross-cloud and heterogeneous environments. The decision flow: single cloud → cloud-native; multi-cloud with one dominant → cloud-native per cloud plus federation; true multi-cloud → SPIFFE/SPIRE plus cloud-native. ScrambleID composes with cloud workload identity rather than replacing it: cloud-native mechanisms handle workload-to-cloud-IAM authentication; ScrambleID handles cross-cloud, cross-tenant, agent-aware, and chain-aware delegation paths via RFC 7523 JWT client assertions and sender-constrained tokens.

FAQ

What is cloud workload identity?

Cloud workload identity is a class of mechanisms where a workload running in a cloud environment authenticates to the cloud's IAM (and to external services) without holding any persistent shared secret. The cloud or a federated trust mechanism issues short-lived credentials based on attestable properties of the workload's runtime: which cluster it's in, which Kubernetes service account, which namespace, which container image. The credentials cannot be exfiltrated to other workloads because they are tied to the runtime context.

What's the difference between IRSA, Workload Identity Federation, and Managed Identity?

IRSA (AWS IAM Roles for Service Accounts) is AWS's mechanism for EKS, where a Kubernetes service account is mapped to an IAM role and the workload assumes that role via a projected token. Workload Identity Federation (GCP) is a more general mechanism that lets external (or internal) workloads exchange OIDC tokens for short-lived GCP credentials; Workload Identity for GKE is the Kubernetes-specific variant. Managed Identity (Azure) is Azure's first-party identity for Azure resources, with system-assigned (lifecycle tied to the resource) and user-assigned (independent

lifecycle) variants. The patterns converge architecturally: workload runtime attests identity, cloud IAM issues short-lived credentials.

When do I need SPIFFE/SPIRE?

SPIFFE (Secure Production Identity Framework for Everyone) and its reference implementation SPIRE provide workload identity across heterogeneous environments: multiple clouds, on-prem, edge. Use SPIFFE/SPIRE when workloads span clouds (AWS plus GCP plus on-prem) or when you need a unified identity model across environments where each cloud's native mechanism would create separate parallel identity systems. For single-cloud workloads, the cloud's native mechanism is typically simpler.

Can I use these mechanisms together?

Yes, and many real deployments do. A multi-cloud platform might use IRSA for AWS workloads, Workload Identity Federation for GCP workloads, Managed Identity for Azure workloads, plus SPIFFE/SPIRE as the unifying layer for cross-cloud service-to-service. The trade-off is operational complexity (more identity systems to govern) versus operational fit (each cloud's native mechanism is the natural choice for its own workloads). Most enterprises end up with the cloud-native mechanism per cloud plus a federation or unification layer where cross-cloud is needed.

Are these mechanisms phishing-resistant?

Phishing-resistance is a property of human authentication ceremonies; the term doesn't directly apply to machine-to-machine. The relevant property for machine identity is replay-resistance and sender-constraint. Cloud workload identity mechanisms inherently bind credentials to the workload's runtime context (the cluster, the service account, the projected token). A leaked credential cannot be replayed from a different workload. For cross-cloud paths, layer sender-constrained tokens (mTLS or DPoP) on top of workload identity to extend the same property to in-flight credentials.

How does ScrambleID fit with these mechanisms?

ScrambleID's M2M control plane composes with cloud-native workload identity rather than replacing it. The pattern: workloads use cloud-native mechanisms (IRSA, WIF, Managed Identity) for their primary identity to cloud resources. For service-to-service paths that cross cloud boundaries or extend beyond what the cloud-native IAM covers (cross-tenant integrations, external partner APIs, agent-to-tool calls, AI workloads needing chain-aware delegation), workloads exchange their cloud-native credentials for ScrambleID-issued tokens via JWT client assertion (RFC 7523), bound via mTLS or DPoP. The result is a single unified identity story without re-implementing cloud IAM.

References (public)

- AWS IAM Roles for Service Accounts (IRSA): https://docs.aws.amazon.com/eks/latest/userguide/iam-roles-for-service-accounts.html
- AWS IAM Roles Anywhere: https://docs.aws.amazon.com/rolesanywhere/latest/userguide/introduction.html
- Google Cloud Workload Identity Federation: https://cloud.google.com/iam/docs/workload-identity-federation
- GKE Workload Identity: https://cloud.google.com/kubernetes-engine/docs/concepts/workload-identity
- Azure Managed Identities: https://learn.microsoft.com/en-us/entra/identity/managed-identities-azure-resources/
- Microsoft Entra Workload Identity for AKS: https://learn.microsoft.com/en-us/azure/aks/workload-identity-overview
- SPIFFE / SPIRE: https://spiffe.io/
- RFC 7523 (JWT Profile for OAuth 2.0): https://datatracker.ietf.org/doc/html/rfc7523
- NIST SP 800-207 (Zero Trust): https://csrc.nist.gov/publications/detail/sp/800-207/final

Related reading

- Service Account Replacement
- M2M Authentication Without Secrets
- Sender-Constrained Tokens (mTLS, DPoP)
- GitHub Actions OIDC Federation Across Clouds
- client_secret vs JWT Client Assertion vs mTLS
- What Is Non-Human Identity (NHI)?
- What Is AI Agent Identity?