SETUP GUIDE

Cloud Account and Key Management Hardening Guide for Web3 Teams

Root accounts, IAM, secrets, and KMS for production and signing

Prepared for	Web3 team members	Classification	Internal
Prepared by	Oak Security	Date	2026-06-17

Oak Security GmbH

1. What This Guide Is For

This guide is for the people who hold the keys to a Web3 team's cloud: infrastructure, cloud, and KMS administrators. Cloud accounts run production, and they often hold or gate the signing material that moves funds. When that boundary blurs, a cloud compromise stops being an infrastructure problem and becomes the whole-company compromise.

The threat model is concrete. An attacker who lands a long-lived access key from a leaked `.env`, a poisoned CI job, or a phished console session does not need a zero-day. They enumerate IAM, assume a role with more privilege than anyone intended, create a quiet backdoor user, and either exfiltrate secrets or use a KMS key to sign on your behalf. Most cloud incidents are credential and privilege problems, not exotic exploits.

> **NOTE**
>
> **Baseline rule:** lock the root/owner account, require hardware MFA, grant least privilege, use no long-lived static keys, log everything, and keep a tested break-glass path.

The aim is not to make the cloud unusable. The aim is to make sure that the predictable failures (a leaked key, a compromised laptop, a malicious dependency in CI) cannot be escalated into root access, silent key use, or an unrecoverable account.

For the most sensitive functions, such as production administration, treasury signing, and key-policy changes, a dedicated account and dual control are still the better option. Section 8 covers that profile.

2. Account and Identity Model

A well-partitioned account model contains a breach by default, instead of relying on every administrator to never make a mistake.

Organization and Account Separation

Use an organization (AWS Organizations, GCP organization with folders, Azure management groups) with a dedicated management account at the top. The management account exists to govern, not to run workloads. Do not deploy applications, store secrets, or hold signing keys in it.

Separate accounts or projects by purpose, not by team convenience. The blast radius of a compromised account should stop at that account's boundary.

Account / Project	Holds	Boundary Intent
Management / organization	SCPs, org-wide config, consolidated billing, audit log aggregation.	No workloads. Highly restricted human access.
Production	Live infrastructure, production data.	No experimentation. Change-controlled.
Staging	Pre-production testing.	Mirrors production controls but isolated from production credentials.
Treasury / signing	KMS/HSM signing keys, signer infrastructure.	Smallest possible footprint. Dual control. Separate from general production.
Log archive	Immutable audit logs from all accounts.	Write-mostly. No deletes by operators.

Centralized Identity

Run one identity source. Use centralized SSO (AWS IAM Identity Center, Google Cloud Identity, Azure Entra ID, or an external IdP federated into all of them) so that joiners, leavers,

and MFA enrollment are managed in one place. Federated, short-lived sessions replace per-account local users.

Root vs Federated Identities

The most privileged identity is not an everyday login. It is a recovery and bootstrap credential, and day-to-day administration runs through federated SSO roles with scoped permissions. The equivalents are not identical across providers: AWS has a true root user per account; GCP has no root user, and the closest high-risk identities are the Cloud Identity / Workspace super-admin and the Organization Administrator role (distinct things, one an account, one an IAM role); Azure's analog is the Entra ID emergency-access (break-glass) account plus elevation to User Access Administrator at the root management-group scope, not the everyday Global Administrator role. Treat each provider's highest-privilege identities accordingly.

Never share root. A shared root credential has no accountability, cannot be cleanly rotated when one holder leaves, and turns any single phished or coerced holder into full account compromise. Enforce two-person control as a process (a sealed credential, a documented procedure, alarms on use), not as a hard cryptographic split where one person holds the password and a different person holds the only MFA token. That split is a recoverability hazard: AWS root requires the password and an MFA challenge in the same session, so a hard split means neither person can act alone and both must coordinate in real time during an incident. Register multiple root MFA devices instead (AWS supports up to eight on the root user) and store them in separate sealed locations.

3. Quick Setup Checklist

These are the controls every cloud account should have before it runs anything that matters.

Root and Account Lockdown

Setting	What To Do	Example
Root MFA	Enable hardware MFA on the root/owner account. Seal the credential.	AWS: hardware MFA on root user. GCP/Azure: hardware MFA on the super-admin.
Root access keys	Delete all root access keys. Root should have no programmatic credentials.	AWS: remove root access keys entirely.
Org guardrails	Apply org-wide policies that no account can override.	AWS SCPs, GCP org policies, Azure Policy.
Billing alerts	Set spend and anomaly alarms. Sudden spend often means compromise (crypto mining, mass key use).	AWS Budgets + Cost Anomaly Detection, GCP Budget alerts.

Identity and Access

Setting	What To Do	Example
SSO	Route all human access through centralized SSO with short-lived sessions.	AWS IAM Identity Center, Google Cloud Identity, Entra ID.
MFA everywhere	Require MFA for every human principal. Hardware keys for admins.	WebAuthn / FIDO2 security keys.
Least privilege	Grant scoped roles, not broad admin. Remove wildcard policies.	Avoid `Action: "*"` on `Resource: "*"`.

No static keys	Eliminate long-lived access keys. Use OIDC federation for CI and workloads.	GitHub Actions OIDC to AWS/GCP/Azure.

Logging

Setting	What To Do	Example
Audit logging	Enable account-wide audit logging in every region and aggregate it.	AWS CloudTrail (all regions), GCP Cloud Audit Logs, Azure Activity Log.
Immutability	Send logs to a separate, write-protected account or bucket.	S3 Object Lock, GCS retention/bucket lock.
KMS logging	Log every key-use event (encrypt, decrypt, sign).	CloudTrail data events for KMS, Cloud KMS audit logs.

4. Root and Privileged Access

Root is the credential an attacker wants most and an administrator should touch least.

Root and Owner Lockdown

- Enable hardware MFA on the root/owner account. A phishable TOTP secret on root is a weak point; use a FIDO2/WebAuthn hardware key.
- Remove all programmatic credentials from root. On AWS, root should have zero access keys.
- Seal the root password and MFA recovery material. Store them so that no single person can use root alone, and so that recovery does not depend on one individual's device or memory.
- Alarm on root usage. Any root login or root API call should generate an alert that a human reviews.

Break-Glass Procedure

You need a documented, tested path to regain control when SSO is down or an account is locked. Untested break-glass is a liability, not a control.

- Define when break-glass is allowed (IdP outage, suspected SSO compromise, account lockout).
- Store the credential sealed, and register multiple root MFA devices held in separate sealed locations so the loss of one token does not force you into slow vendor account-recovery. Do not make access depend on one irreplaceable MFA device.
- Require two people to invoke it as a process control (two-person rule, alarmed), not as a password-here / only-MFA-there split that can lock everyone out.
- Treat the root account's recovery email and phone number as break-glass dependencies in their own right: control them, monitor them, and do not let them point at a personal mailbox or a portable phone number.
- Log and alert on every use.

- Test it on a schedule. A recovery path that has never been exercised tends to fail when you need it.

Administer via Roles, Not Static Keys

Day-to-day administration runs through federated SSO roles that issue short-lived credentials. Do not create static IAM admin users with long-lived access keys. A leaked static admin key is durable attacker access; a leaked short-lived session token expires.

Permission Boundaries and SCPs

- Use service control policies (AWS SCPs) or equivalent org policies (GCP org policies, Azure Policy) to set hard ceilings no account or role can exceed. Use them to deny disabling of audit logging, deny use outside approved regions, and protect log and security configuration.
- Use IAM permission boundaries to cap what a role can delegate, so that a role granting permissions cannot create a more privileged role. This blocks the classic privilege-escalation path where a mid-tier role rewrites its own or another role's policy.

Just-in-Time Elevation

High privilege should be temporary. Grant elevated roles on request, time-boxed, with a reason and an approver, and have them expire automatically. Standing admin access is standing attack surface. Just-in-time elevation replaces permanent admin rights with a time-limited, logged grant.

5. Secrets and Credential Management

Static secrets are the most common entry point in cloud breaches. The goal is to have as few long-lived secrets as possible and to make the ones that remain short-lived, scoped, and observable.

Eliminate Long-Lived Access Keys

Long-lived access keys (`AKIA...` and equivalents) are the single most leaked credential class: they end up in code, CI logs, container images, and laptops. Treat any long-lived key as a liability to be removed, not rotated forever.

- Inventory every static key. Identify the owner and purpose of each.
- Replace human static keys with SSO sessions.
- Replace workload and CI static keys with workload identity / OIDC federation.
- Delete unused keys. Set maximum-age alarms on any that survive.

Workload Identity and OIDC Federation

CI systems and workloads should not hold static cloud keys. Use OIDC federation so the workload presents a short-lived, signed identity token and assumes a scoped role.

Source	Mechanism	Result
GitHub Actions	OIDC trust to AWS IAM role / GCP Workload Identity Federation / Azure federated credential.	No cloud secret stored in the CI system.
Kubernetes / containers	IRSA (AWS), Workload Identity (GKE), Workload Identity (AKS).	Pods assume scoped roles without static keys.
EC2 / GCE / VM workloads	Instance roles / attached service accounts with short-lived credentials.	No keys baked into images.

When configuring an OIDC trust, pin the subject claim tightly (specific repository, branch, or environment). A loose trust condition (for example, trusting any repo in an organization, or omitting the branch/environment) lets a fork or an unrelated workflow assume your production role.

Use a Secrets Manager

Application secrets belong in a secrets manager (AWS Secrets Manager, GCP Secret Manager, Azure Key Vault, HashiCorp Vault), not in environment variables baked into images, config files in the repo, or chat.

- Retrieve secrets at runtime via the workload's identity, not via a static key that itself unlocks the manager.
- Scope each secret so a workload can read only what it needs.
- Enable automatic rotation where the manager supports it.
- Audit access. Every read of a sensitive secret should be logged.

Rotation and Scoping

Tight scope limits damage; rotation limits duration. A read-only, single-resource credential that rotates daily is a far smaller prize than a broad, never-expiring one. Scope to the minimum action set and the minimum resource, and rotate on a schedule and immediately on suspicion.

Detect Leaked Keys

- Do not rely on it as a primary control, but know that providers act automatically on keys found in public repositories: AWS detects exposed access keys and applies a quarantine policy and opens a support case with no action needed on your part. This is a backstop, not your detection strategy.
- Run secret scanning on every repository and in pre-commit and CI (push protection, gitleaks, trufflehog).
- Alarm on anomalous credential use: a key used from a new region, a new ASN, or at an unusual time.

Separate Application Secrets from Signing Material

Do not store private keys, mnemonics, or signer secrets next to ordinary application secrets. A database password and a treasury signing key have different blast radii and belong under different controls. Signing material goes in KMS/HSM (Section 6), not in a general-purpose secrets manager as plaintext.

6. KMS, HSM, and Signing Material

For Web3 teams, this is where cloud security meets fund safety. Treat signing keys as material that must never leave a hardware boundary and whose every use is logged.

Keys in KMS/HSM, Never in Files

Generate and hold signing and encryption keys in a managed KMS or HSM (AWS KMS, AWS CloudHSM, Google Cloud KMS, Azure Key Vault / Managed HSM). The private key stays inside the boundary; callers ask the service to sign or decrypt, and the key material is never returned.

Envelope Encryption

For data encryption, use envelope encryption: a KMS-held key encryption key (KEK) wraps a data encryption key (DEK), and only the DEK touches the data. The high-value key never leaves KMS, and you can revoke access by controlling the KEK.

Key Policies and Grants

- Write explicit key policies. Define who can administer a key and, separately, who can use it. Default to deny.
- Use grants for temporary, narrowly scoped delegation rather than broadening the key policy.
- Scope key use to specific principals and operations. A signer should be able to sign with one key, not administer any key.

Separation of Duties

Split key administration from key use. The principal that can change a key policy, schedule deletion, or disable a key must not be the same principal that signs with it. Otherwise a single compromised credential can both grant itself signing rights and remove the evidence.

Role	Can Do	Cannot Do

Key administrator	Create keys, set policies, manage rotation, schedule (delayed) deletion.	Sign or decrypt with the key.
Key user (signer service)	Invoke sign / decrypt within scope.	Change key policy, delete, or disable the key.
Auditor	Read key metadata and usage logs.	Administer or use keys.

CloudHSM and Single-Tenant Options

If you require single-tenant, FIPS-validated Level 3 hardware and control of the HSM cluster, use AWS CloudHSM or Azure Managed HSM rather than the multi-tenant default. The exact validation (FIPS 140-2 vs 140-3 Level 3) depends on the specific product and hardware generation and changes over time, so confirm the current NIST CMVP certificate for your chosen service rather than relying on a fixed claim if you have a compliance requirement. The tradeoff is operational overhead and your responsibility for HSM user management and backup.

Threshold and MPC Signing

For high-value signing, consider threshold signatures or MPC so that no single complete key exists in one place. The benefit is real only when the shares are held in genuinely separate trust domains with independent compromise paths. Running every share in the same cloud account, under the same operator, or behind the same orchestration layer gives you the operational complexity of MPC with the blast radius of a single key. Note also that MPC protects key confidentiality and availability, not authorization: an attacker who controls the quorum of signer services still produces valid signatures, so transaction-authorization controls (policy, approvals, the multisig discipline in the multisig guide) remain necessary. This is an option to weigh against operational complexity, not a universal requirement.

Audit Every Key-Use Event

Log every sign, decrypt, and administrative action on every key (CloudTrail data events for KMS, Cloud KMS audit logs, Key Vault logging). Send these logs to the immutable archive.

Alarm on unexpected signing volume, signing from new principals, and any key-policy or key-state change.

Keep Keys Non-Exportable (Provider-Specific)

The principle is that key material must never leave the boundary. How you enforce it differs by provider, and getting this wrong gives false confidence.

- **AWS KMS.** Standard KMS keys are non-exportable by design. There is no API to extract key material, and there is no "export" action to deny. The control that matters here is the key `Origin`: if your policy requires HSM-native generation, deny creation of keys with imported (external) material, and keep `kms:GetPublicKey` available (the public key is meant to be readable).
- **GCP Cloud KMS.** Software and HSM-protection-level keys are non-extractable; there is nothing to toggle. Do not create keys you intend to import unless your model requires it.
- **Azure Key Vault / Managed HSM.** Keys have an `exportable` attribute. Set it false, and do not attach a key-release (secure key release) policy, or the key can leave the boundary by design.

Do not write a generic "deny export" key-policy statement on AWS and assume it does anything: the action it targets does not exist. Enforce the real, provider-specific control above.

7. The Recommended Setup Flow

Step 1: Lock Root and Add Hardware MFA

Enable a FIDO2/WebAuthn hardware key on the root/owner account. Delete all root access keys. Seal the root password and MFA recovery material so no single person can use root alone. Set an alarm on any root usage.

This is the single highest-value step. A locked root with hardware MFA defeats the most damaging escalation path: an attacker taking over the account itself.

Step 2: Enable Org Guardrails

Turn on the organization layer (AWS Organizations, GCP org with folders, Azure management groups). Apply SCPs or org policies that no account can override: deny disabling audit logging, deny use outside approved regions, protect the log archive, and deny key export.

Step 3: Set Up SSO and Least-Privilege Roles

Stand up centralized SSO (IAM Identity Center, Cloud Identity, Entra ID) and route all human access through short-lived, scoped roles. Remove standing local admin users. Define roles by job function with least privilege, and add just-in-time elevation for high-privilege work.

Step 4: Eliminate Static Keys in Favor of OIDC

Inventory every long-lived access key. Replace CI and workload keys with OIDC federation and workload identity, pinning trust conditions to specific repositories, branches, or environments. Replace human keys with SSO sessions. Delete what remains and alarm on any new static key.

Step 5: Deploy a Secrets Manager with Rotation

Move application secrets into a secrets manager. Have workloads read secrets at runtime via their own identity. Scope each secret to the minimum, and enable automatic rotation. Turn on secret scanning and push protection across all repositories.

Step 6: Move Signing Keys into KMS with Strict Key Policies

Generate signing and encryption keys in KMS/HSM as non-exportable. Write explicit key policies that separate administration from use. Scope signer services to sign-only on their specific key. For high-value signing, evaluate threshold/MPC.

Step 7: Enable Logging, Alerting, and Billing Alarms

Turn on account-wide audit logging in every region and aggregate it into an immutable, separate archive. Enable KMS key-use logging. Set alarms on root usage, new IAM users/roles/keys, key-policy changes, anomalous credential use, and billing anomalies. A sudden spend spike is often the first visible sign of compromise.

8. Extra Rules for Critical-Access Users

Production and treasury infrastructure raise the stakes: a single mistake or compromise here can move funds or take down the company. If you administer these systems, use the stricter profile below.

Control	Requirement
Dedicated accounts	Run treasury/signing and critical production in dedicated accounts, isolated from general workloads and from each other.
Dual control	Require two-person approval for key-policy changes, IAM/SCP changes, and signer configuration changes. No unilateral changes to who can sign.
Hardware-backed MFA	All access uses FIDO2/WebAuthn hardware keys. No SMS or TOTP-only on critical principals.
Break-glass	Keep a sealed break-glass credential with multiple registered MFA devices in separate sealed locations, two-person process control, and a controlled recovery email/phone. Tested on a schedule. Avoid a hard password/MFA split that can lock everyone out.
No single all-powerful role	No role can both administer and use signing keys, and no role can disable logging. Separate duties enforced by policy, not convention.
Immutable logging	Critical-account logs go to a centralized, write-protected archive that operators cannot delete.
Non-exportable keys	Signing keys are non-exportable, enforced with the provider-specific control from Section 6 (not a generic "deny export" statement, which is a no-op on AWS KMS).
Just-in-time elevation	No standing admin on production or treasury. Elevation is time-boxed, reasoned, and approved.

Least-privilege signing	Signer services can sign with one key only. They hold no other cloud privilege.

9. Things To Avoid

These are the patterns behind most cloud and key-management incidents.

- Using the root/owner account for daily work instead of sealing it.
- Long-lived access keys committed to code, baked into images, or stored in CI secrets when OIDC federation is available.
- Wildcard IAM policies (`Action: "*"` on `Resource: "*"`) and over-broad managed admin policies handed out as a default.
- Loose OIDC trust conditions that let any repository, fork, or branch assume a production role.
- Public storage buckets holding anything that is not deliberately public. Default to private and block public access at the account level.
- Disabling, scoping down, or failing to aggregate audit logging. An attacker who can turn off logging operates blind to you.
- Exportable KMS keys, or storing private keys and mnemonics as plaintext in a secrets manager or environment variables.
- Shared admin credentials and shared root. No accountability, no clean rotation, no clean offboarding.
- A single role that both administers and uses signing keys.
- No billing or anomaly alerts, so a key-abuse or mining spree runs until the invoice arrives.
- Storing the break-glass credential beside the account it recovers, or never testing it.

10. If Something Suspicious Happens

For a suspected cloud key or account compromise, move fast and in order: cut off access, preserve the trail, then rebuild. Do not ″clean up″ first and lose the evidence of what the attacker did.

Immediate Actions

1. Revoke active sessions and disable the suspected credentials. For a suspected key compromise, deactivate the access key or revoke the role session immediately.
2. If a human account is suspected, force sign-out across SSO and disable the principal.
3. Notify the security owner from a separate, trusted channel. Do not assume your normal channel is uncompromised.
4. Do not delete resources or logs yet. You need them for both forensics and to understand scope.

Contain

- Restrict the blast radius: tighten or deny SCPs/org policies on the affected account, lock down the affected roles, and cut network paths if a workload is implicated.
- For suspected signing-key compromise, disable use of the affected KMS key (disable the key or deny the use grant) and stop the signer service. Do not schedule key deletion yet. Disabling is instantly reversible and preserves the key and its evidence; scheduled deletion only takes effect after a mandatory waiting window (7 to 30 days on AWS KMS) and is reversible during that window, but it is the wrong first move while you are still establishing scope.
- Snapshot for forensics: capture the audit trail, IAM and key-policy state, and any affected instances/volumes before changing them.

Review the Audit Trail

Work the logs (CloudTrail, Cloud Audit Logs, Activity Log, plus KMS key-use logs) for the indicators an attacker leaves behind.

Look For	Why It Matters
New IAM users, roles, or access keys	The most common persistence mechanism. Any unexplained principal is suspect.
Changed or attached policies	Privilege escalation: a role that gained new permissions or a wildcard policy.
Key-policy or grant changes	An attacker granting themselves signing/decrypt rights, or disabling a key.
Unexpected sign/decrypt events	Key use from new principals, new regions, or at unusual volume.
Logging changes	Any attempt to stop, scope down, or redirect audit logging.
New resources or regions	Mining instances, unexpected resources, or activity in regions you do not use.
Federation/trust changes	New OIDC trusts or loosened conditions enabling external assumption of roles.

Rotate

Rotate everything plausibly exposed, most sensitive first:

- The compromised credential and any credential it could reach.
- All access keys and tokens in the affected account.
- Secrets the attacker could have read from the secrets manager.
- OIDC trust conditions, if a loose trust was the entry point.
- Signing keys: if a signing key was exposed (not merely used), rotate to a new key and, for on-chain signers, rotate the on-chain authority if your protocol allows it. A KMS signing key cannot be assumed clean once its use principal is compromised.

Decide Whether to Rebuild

If compromise of an account cannot be ruled out, rebuild from a known-good state rather than trusting a cleaned account: redeploy infrastructure from version control into a fresh

account, restore data from trusted backups, recreate roles and trusts from scratch, and reconnect only after old credentials are revoked. Do not resume treasury, signing, or production administration until the security owner agrees the environment is clean.

11. Monthly Self-Check

Once a month, run this short review across the accounts you administer.

Check	Done
Root/owner accounts have hardware MFA and no access keys.	
Root usage alarm is active and shows no unexplained root activity.	
Break-glass credential is sealed, split, and was tested recently.	
All human access runs through SSO with MFA; no standing local admin users.	
No wildcard IAM policies or unexplained broad admin grants.	
Long-lived access keys are inventoried; CI and workloads use OIDC.	
Any surviving static keys are within max-age and have owners.	
Secrets are in the secrets manager with rotation; none in repos, images, or env files.	
Secret scanning and push protection are active on all repositories.	
KMS signing keys are non-exportable; admin and use duties are separated.	
Key-use logging is on, and key-policy changes are alarmed.	
Audit logging is on in all regions and aggregated to the immutable archive.	
Billing and anomaly alarms are active.	
No public storage buckets hold non-public data.	

Stale principals, roles, keys, and OIDC trusts have been reviewed.	

12. References

- AWS Security Best Practices (root user): https://docs.aws.amazon.com/IAM/latest/UserGuide/root-user-best-practices.html
- AWS IAM Security Best Practices: https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html
- AWS Well-Architected Security Pillar: https://docs.aws.amazon.com/wellarchitected/latest/security-pillar/welcome.html
- Google Cloud Architecture Framework (Security): https://cloud.google.com/architecture/framework/security
- Google Cloud Enterprise foundations blueprint: https://cloud.google.com/architecture/security-foundations
- CIS Benchmarks (AWS, GCP, Azure foundations): https://www.cisecurity.org/cis-benchmarks
- AWS KMS Developer Guide: https://docs.aws.amazon.com/kms/latest/developerguide/overview.html
- AWS KMS best practices: https://docs.aws.amazon.com/kms/latest/developerguide/best-practices.html
- AWS CloudHSM: https://docs.aws.amazon.com/cloudhsm/latest/userguide/introduction.html
- Google Cloud KMS documentation: https://cloud.google.com/kms/docs
- Azure Key Vault security: https://learn.microsoft.com/en-us/azure/key-vault/general/security-features
- Configuring OpenID Connect in AWS (GitHub docs): https://docs.github.com/en/actions/security-for-github-actions/security-hardening-your-deployments/configuring-openid-connect-in-amazon-web-services
- GitHub Actions OIDC overview: https://docs.github.com/en/actions/security-for-github-actions/security-hardening-your-deployments/about-security-hardening-with-openid-connect

- GCP Workload Identity Federation: https://cloud.google.com/iam/docs/workload-identity-federation
- AWS CloudTrail User Guide: https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-user-guide.html
- Google Cloud Audit Logs: https://cloud.google.com/logging/docs/audit