Spacelift Secure Configuration Guide (FedRAMP SCG)

Version: 1.0

Effective Date: March 1, 2026

Applicable Environment: Spacelift FedRAMP Moderate (hosted on Knox Systems)

Disclaimer: The secure configuration of the Spacelift platform is the customer's responsibility. While Spacelift provides the features, controls, and secure defaults described in this guide, it is the responsibility of each agency customer to implement, configure, and maintain these settings in accordance with their organization's security policies, risk appetite, and applicable federal requirements.

1. Introduction

1.1 Purpose

This Secure Configuration Guide (SCG) provides federal agency customers and government contractors with clear, actionable instructions for securely configuring, operating, and hardening their Spacelift environment. It is published in compliance with the FedRAMP Rev 5 Balance Improvement Release requirements for Secure Configuration Guides (effective March 1, 2026).

1.2 Audience

This guide is intended for IT administrators, security engineers, and DevOps/platform engineers responsible for configuring and managing Spacelift within a FedRAMP-authorized environment.

2. Account Model and Administrative Accounts

2.1 Top-Level Administrative Account (Root Space Admin)

In Spacelift, the **top-level administrative account** is the **Root Space Admin** (also referred to as the Account Owner). This is the highest-privilege role within a Spacelift organization and has the following capabilities:

- Full administrative control over all Spaces, stacks, policies, contexts, integrations, and worker pools across the entire account.
- Ability to configure Single Sign-On (SSO) and identity provider settings.
- Ability to configure and manage VCS (Version Control System) integrations.
- Ability to manage the audit trail configuration, including webhook endpoints.
- Ability to enforce organization-wide Multi-Factor Authentication (MFA).
- Ability to create, modify, and delete API keys.
- Ability to manage billing and account-level settings.

How this role is named in Spacelift: The role appears as "Root Space Admin" in the RBAC system, and the initial account creator is automatically assigned this role. Users with the "Admin" designation in the Identity Management section also hold top-level administrative privileges.

Important: When using GitHub as the default identity provider, GitHub organization admins and private account owners always have admin access to their Spacelift account, regardless of login policies. When using SSO (SAML/OIDC), the Root Space Admin role must be explicitly assigned.

2.2 Privileged Accounts

Below the Root Space Admin, Spacelift supports Space Admins, Space Writers, and custom roles that serve as privileged accounts. Privileged accounts operate with elevated access that inherently includes write permissions and extends further to administrative capabilities such as managing policies, integrations, worker pools, and child Space resources — scoped to their assigned Space boundary.

- **Space Admin role:** Administrative control scoped to a specific Space and its child Spaces. Can manage stacks, policies, worker pools, contexts, and integrations within their Space boundary.
- **Space Writer role:** Provides write access to a specific Space and its child Spaces. Can create and modify resources within their Space boundary but without full administrative capabilities.
- **Custom Roles:** Organization-defined roles with granular permissions (e.g., "Infrastructure Developer," "Security Auditor"). Created via Organization Settings → Access Control Center → Roles.

All of the above roles can be attached to users, API keys, and stacks, providing a consistent and flexible RBAC model across all actor types within Spacelift.

3. Securely Provisioning Top-Level Administrative Accounts

3.1 Initial Account Setup

1. **Contact Spacelift** through your Account Executive to initiate FedRAMP environment provisioning.
2. **Complete identity verification** as required by the onboarding process. Spacelift and their vendors only store verification results, never the ID documents themselves.
3. **Receive your FedRAMP tenant URL** (format: `<your-org>.app.fedramp.spacelift.io`).
4. **Create the initial account** by signing in with a supported identity provider. The account creator becomes the initial **Root Space Admin**.

3.2 Configuring the Identity Provider

Recommended (Secure Default): Configure SSO using OIDC with your organization's enterprise identity provider (e.g., Okta, Microsoft Entra ID). This is strongly recommended over using a social identity provider (GitHub, Google, etc.) for FedRAMP environments.

To configure Identity Provider follow the steps:

1. Navigate to **Organization Settings → Single Sign-On**.
2. Configure your OIDC identity provider following the appropriate setup guide:
   - Okta OIDC Setup Guide
   - Microsoft Entra ID OIDC Setup Guide
   - GitLab OIDC Setup Guide
   - OneLogin OIDC Setup Guide
3. **Configure group claims** in your IdP to pass user group membership to Spacelift. This enables IdP Group Mapping for role-based access control.
4. **Set up backup credentials** before switching to SSO to prevent lockout scenarios. Navigate to Organization Settings → API Keys and delete them after successful configuration.

Security Implication: Using a social identity provider (GitHub, Google, etc.) without enterprise controls such as enforced MFA and conditional access policies on the IdP side exposes the Spacelift account to risks from compromised social accounts. Enterprise SSO with OIDC provides centralized session management and revocation without the risk of expiring certification instead of SAML.

Reference: https://docs.spacelift.io/integrations/single-sign-on#setting-up-oidc

3.3 Configuring Multi-Factor Authentication (MFA)

Spacelift provides **IdP-independent MFA** using FIDO2 security keys. This adds a second authentication factor managed directly within Spacelift, on top of whatever authentication your identity provider requires.

Recommended Configuration: Enforce MFA for all users in the organization and use FIPS-validated hardware security keys.

Steps to enforce MFA organization-wide:

1. Navigate to **Organization Settings → Authentication → Multi-Factor Authentication**.
2. Click **Enforce MFA**.
3. Once enforced, every active user must register at least one FIDO2 security key. Existing sessions (except the current one) will be invalidated.

Steps for individual users to register a security key:

1. Go to Personal Settings → Multi-Factor Authentication.
2. Click Enable and follow the prompt to register your FIDO2 security key.
3. Name the key for easy identification.
4. Register at least one backup security key.

Security Implication: Without MFA enforcement, a compromised IdP session can grant full access to the Spacelift account. With MFA enforced, even if an attacker obtains IdP credentials, they cannot authenticate to Spacelift without the physical FIDO2 security key.

4. Authorization and Access Control

4.1 Spaces and Hierarchical Access Control

Setting: Spaces

Location: Organization Settings → Spaces

Secure Default: All resources are placed in a single Root Space

Recommended Configuration: Create a hierarchical Space structure that aligns with your organizational boundaries, environments, and compliance requirements.

Security Implications: Spaces provide logical isolation boundaries for stacks, policies, worker pools, contexts, and integrations. Without proper Space segmentation, all users with access to any part of Spacelift can potentially see or modify all resources. Properly structured Spaces enable micro-segmentation, restricting lateral movement and minimizing the blast radius of a compromised account.

Best practices:

- Separate Spaces by environment (e.g., `dev`, `staging`, `prod`).
- Separate Spaces by team or project if different groups require different access levels.
- Use Space inheritance for shared policies and contexts, but restrict inheritance for sensitive environments.
- FedRAMP and commercial workloads must be fully isolated — use separate Spacelift accounts or strictly separated Spaces with independent worker pools and encryption keys.

Reference: https://docs.spacelift.io/concepts/spaces

4.2 Role-Based Access Control (RBAC)

Setting: Custom Roles and Role Bindings

Location: Organization Settings → Access Control Center → Roles

Secure Default: Built-in roles (Space Admin, Space Writer, Space Reader)

Recommended Configuration: Define custom roles with the minimum permissions needed for each function. Follow the principle of least privilege.

Security Implications: Overly broad role assignments (e.g., giving all users Space Admin) increase the risk of accidental or malicious changes to infrastructure. Custom roles allow decomposing broad permissions into specific, composable actions. For example, a "Deployment Operator" role might only include `run:trigger`, `run:confirm`, and `stack:read` - without permission to create or modify stacks.

Instructions:

1. Navigate to Organization Settings → Access Control Center → Roles.
2. Click **Create Role**.
3. Provide a descriptive name and select only the actions needed.
4. Assign roles to users, API keys, Stacks, or IdP groups scoped to specific Spaces.

Key principle: The `space:read` action is required to view any subjects within a Space. Without it, users cannot see resources even if they have other permissions.

Reference: https://docs.spacelift.io/concepts/authorization/rbac-system

4.3 User Management Strategy

Setting: User Management method

Location: Organization Settings → Identity Management

Secure Default: User Management UI

Recommended Configuration: Choose **one** of the two mutually exclusive approaches below and apply it consistently across your organization. Do not mix methods, as this can lead to unpredictable access behavior.

Option A — User Management UI (recommended for most organizations)

Admins manually assign roles to individual users, API keys, or IdP groups via the UI or Terraform provider. Role assignments take effect immediately and are easy to audit visually.

- **Best for:** organizations with a small-to-medium user base and straightforward access requirements.
- **Drawback:** requires manual updates when team membership changes; risk of stale access if offboarding processes are not followed.

Option B — Login Policies (recommended for large or complex organizations)

Roles are programmatically assigned at login time using OPA/Rego policies evaluated against IdP claims (e.g., group membership, email domain). Access is automatically revoked when a user leaves an IdP group. Without a restrictive login policy, any user who can authenticate through your IdP may gain access to Spacelift, so policies should explicitly grant least-privilege access based on IdP group membership and deny access to non-members.

- **Best for:** organizations with dynamic team membership, complex access rules, or a need for fully auditable, policy-as-code access control.
- **Drawback:** requires Rego expertise to author and maintain policies.

To configure, navigate to Policies, click Create Policy, select Login as the policy type, write a Rego policy that explicitly grants access to authorized groups and denies all others, then attach the policy to your account.

Best practices for Login Policies:

- Grant admin access only to a small, named set of users or groups.
- Use `deny` rules to explicitly block access for users outside your team.
- Prefer group-based access over individual user allowlisting — when an employee leaves the IdP group, they automatically lose Spacelift access.
- Use the roles rule to assign RBAC roles for specific Spaces rather than granting blanket admin access.

Important: These two methods are mutually exclusive per account. Enabling Login Policies disables the User Management UI for role assignment. Choose your approach during initial setup and apply it consistently.

Regardless of strategy, ensure role assignments are regularly reviewed and offboarded users are promptly removed from your IdP groups or explicit role bindings.

References:

- https://docs.spacelift.io/concepts/policy/login-policy
- https://docs.spacelift.io/concepts/user-management

4.4 API Key Management

Setting: API Keys (Secret-Based and OIDC-Based)

Location: Organization Settings → API Keys

Secure Default: No API keys created by default

Recommended Configuration: Prefer OIDC-based API keys to eliminate static credentials. If secret-based keys are necessary, scope them to the minimum required Spaces and roles.

Security Implications: Secret-based API keys are static credentials. If compromised, they provide persistent access until rotated or deleted. OIDC-based API keys are more secure because they use short-lived tokens issued by an external identity provider, eliminating the risk of long-lived credential exposure.

Best practices:

- Use OIDC-based API keys wherever possible.
- For secret-based keys, assign the narrowest possible role (avoid admin unless absolutely necessary).
- Store API key secrets in an approved secrets manager — Spacelift does not store or allow retrieval of the secret after creation.
- Rotate API keys on a regular schedule.
- Monitor API key usage through the audit trail.
- API keys are billed as users; each key used during a billing cycle counts against your user total.

Reference: https://docs.spacelift.io/integrations/api

5. Worker Pool

5.1 Private Worker Pools

Setting: Worker Pool configuration

Location: Organization Settings → Worker Pools

Secure Default: In FedRAMP, public workers are not available. Private worker pools must be provisioned by the customer using dedicated FIPS-compliant image, and customer-managed encryption keys (BYOK) are required to ensure the run state and sensitive data remain under your exclusive control.

Recommended Configuration: Deploy private workers within your own AWS infrastructure with network isolation, scoped IAM roles, and encrypted communication.

Security Implications: Private workers execute IaC runs within your infrastructure, meaning code, state, and credentials remain under your control. The temporary run state is encrypted so only your workers can decrypt it.

Instructions for initial setup (FedRAMP):

Since public workers are not available in the FedRAMP environment, you cannot use Spacelift itself to bootstrap your first worker pool. Instead:

1. Run your initial Terraform/OpenTofu configuration through an external CI/CD pipeline (e.g., GitHub Actions) to provision your first private worker pool, including its BYOK encryption key configuration.
2. Once the first worker pool is operational, use Spacelift stacks to manage subsequent worker pools.

Best practices:

- Use dedicated FIPS-compliant worker image
- Deploy workers in dedicated VPCs with restricted security groups.
- Scope IAM roles assumed by workers to the minimum permissions required.
- Use separate worker pools for different environments (dev, staging, prod) or compliance boundaries.
- Regularly update worker images to incorporate the latest security patches.

Reference: https://docs.spacelift.io/concepts/worker-pools

6. Secret Management

6.1 Environment Variables and Secrets

Setting: Stack and Context environment variables

Location: Stack → Environment tab; Contexts

Recommended Configuration: Minimize the storage of secrets within Spacelift. Use OIDC-based cloud integrations and external secret managers wherever possible.

Security Implications: While Spacelift provides strong encryption for stored secrets, the most secure approach is to eliminate static secrets entirely:

- **Preferred (Highest security):** Use OIDC-based cloud integrations to assume short-lived roles with no static credentials.
- **Recommended:** Store high-risk secrets in an external secret manager (e.g., AWS Secrets Manager, HashiCorp Vault) and retrieve them at runtime using pre-initialization hooks on private workers.

- **Acceptable for low-risk credentials:** Use Spacelift's built-in secret storage with secret visibility for values like monitoring system tokens.

Variable visibility settings:

Visibility	Behavior	Recommendation
`plaintext`	Value visible in UI and logs	Use only for non-sensitive configuration
`secret`	Write-only; masked in logs; extra-encrypted at rest	Use for sensitive values that must be stored in Spacelift

Reference: https://docs.spacelift.io/concepts/configuration/environment

6.2 Cloud Integration Credentials

Setting: Cloud provider integration method

Location: Organization Settings → Cloud Integrations; or Stack → Integration settings

Secure Default: No cloud integration configured by default

Recommended Configuration: Use OIDC federation for all cloud provider integrations to eliminate static access keys.

Security Implications: Static AWS access keys, Azure service principal secrets, or GCP service account keys are long-lived credentials that pose a significant risk if compromised. OIDC-based integrations generate short-lived, dynamically-created credentials that expire automatically (default: 1 hour for AWS). Spacelift never stores these dynamic credentials and masks them in run logs.

Best practices:

- Use OIDC federation with AWS, Azure, and GCP.
- Scope IAM roles to the minimum permissions required for each stack.
- Use ExternalId in AWS trust policies to prevent confused-deputy attacks.
- Enable session tagging on AWS integrations to include run and stack metadata in CloudTrail logs for auditing.
- Use separate cloud integration roles for read-only (proposed runs) vs. read-write (tracked runs/deployments).

Reference: https://docs.spacelift.io/integrations/cloud-providers/oidc

7. Policy-Driven Governance

Setting: Spacelift Policies (OPA/Rego)

Location: Enforce Guardrails → Policies

Secure Default: No policies attached by default. Note that while no push policy is attached by default, Spacelift does have a built-in default push behavior: pushes to the **tracked branch** trigger a **tracked run** (deployment), and pushes to **any other branch** trigger a **proposed run** (plan/preview). This default behavior can be fully customized by attaching a push policy. See the Default Git Push Policy documentation for the equivalent policy definition.

Recommended Configuration: Define and attach policies across all relevant stacks to enforce separation rules, access guardrails, deployment controls, and run authorization. Manage policies as code using the Spacelift Terraform provider for full auditability.

Security Implications: Policies are a key isolation and governance mechanism in FedRAMP environments. Without policies, Spacelift relies solely on RBAC for access control, with no enforcement of deployment workflows, change approval, or run authorization. Policies allow you to express compliance rules as code, version-control them, and apply them consistently across all stacks.

Spacelift uses **Open Policy Agent (OPA)** and its rule language **Rego** to evaluate policies at various decision points. The following policy types are relevant for FedRAMP environments:

- **Login Policy** — Controls who can log in to Spacelift and with what level of access. Use to explicitly grant access based on IdP group membership and deny all others. This is the primary enforcement point for access control when using Login Policies instead of the User Management UI.
- **Push Policy** — Determines how Git push events are interpreted. Use to restrict which branches can trigger tracked (deployment) runs, enforce branch protection, and pass signed tokens as run metadata for signed run workflows.
- **Initialization Policy** — Evaluated by the private worker pool launcher before a run starts. Use to verify signed run tokens, block unauthorized runs, and enforce pre-execution checks on private workers.
- **Plan Policy** — Evaluated after the planning phase. Use to enforce organizational rules on infrastructure changes, block high-risk resource modifications, and require human approval for sensitive changes.
- **Approval Policy** — Controls who can approve or reject a run. Use to enforce multi-party approval for production deployments or changes to sensitive environments.
- **Trigger Policy** — Selects stacks for which to trigger a tracked run when a blocking run terminates. Use to manage stack dependency chains and automate controlled deployment pipelines.
- **Notification Policy** — Routes and filters notifications. Use to alert security teams on high-risk policy events or run failures.

Best practices:

- Manage all policies as code using the Spacelift Terraform provider for version control and auditability.
- Unit test all policies using `opa` test before deploying them to production.
- Use the Policy Workbench to simulate and validate policy behavior against real inputs.
- Apply the principle of least privilege: use `deny` rules to block by default and only explicitly allow authorized actions.
- Use Policy Flags to pass contextual data (e.g., affected file paths, team ownership) between policy types for complex approval workflows.

Reference: https://docs.spacelift.io/concepts/policy

8. Audit, Logging, and Monitoring

8.1 Audit Trail

Setting: Built-in Audit Trail

Location: Organization Settings → Audit Trail

Recommended Configuration: Enabled; logs retained for 30 days. Enable audit trail webhooks to export all audit events to an external SIEM or log storage system for long-term retention and analysis.

Security Implications: The audit trail records all operations that change Spacelift resources, including the actor identity, source of identity, roles attached, contextual metadata, and action-specific payload. Sensitive arguments (like secrets) are sanitized in audit records. Without external forwarding, audit data is only retained for 30 days and is only accessible within Spacelift — which may not satisfy federal log retention requirements.

Instructions to enable audit webhooks:

1. Navigate to Organization Settings → Audit Trail → Configuration tab.
2. Click **Set up**.
3. Provide a webhook endpoint URL and a verification secret.
4. Optionally specify custom headers and enable the **Include runs** option to capture run state change events.
5. Spacelift provides a reference implementation for forwarding audit events to an AWS S3 bucket.

Important audit events to monitor:

- Audit trail being disabled or deleted (treat as a high-priority security alert).
- Login failures and denied access attempts.
- Changes to SSO/MFA configuration.
- API key creation, modification, or deletion.
- Role assignment changes.
- Policy creation, modification, or deletion.
- Stack creation or deletion.

Reference: https://docs.spacelift.io/integrations/audit-trail

8.2 Run Logs

Setting: Run log storage and redaction

Location: Stack Settings → Auto-redact secrets in logs

Recommended Configuration: Enable auto-redaction of secrets in logs. Export run logs to external storage for long-term retention and compliance.

Security Implications: Run logs may inadvertently contain sensitive values. Auto-redaction automatically masks known secret patterns (e.g., AWS credentials) from log output. For additional protection, any value explicitly marked as `secret` in the environment is redacted in all log output.

8.3 Observability Integrations

Setting: Datadog and Prometheus integrations

Location: Organization Settings → Integrations

Recommended Configuration: Enable observability integrations to monitor stack health, run durations, and failure rates.

Reference: https://docs.spacelift.io/integrations/observability

9. Source Code and VCS Security

9.1 VCS Agent Pools

Setting: VCS Agent Pools for private/on-premise VCS

Location: Organization Settings → VCS Agent Pools

Recommended Configuration: For on-premise or private VCS systems, deploy VCS agents within your infrastructure.

Security Implications: VCS agents enable Spacelift to communicate with on-premise version control systems without exposing them to the public internet. VCS agents use gRPC over HTTP/2 for secure connectivity. Without VCS agents, Spacelift requires direct network access to your VCS, which may necessitate opening firewall rules.

Reference: https://docs.spacelift.io/concepts/vcs-agent-pools

9.2 Code Integrity

Setting: Signed Runs (`plugin-signed-runs`)

Recommended Configuration: Use the Spacelift signed runs plugin to ensure that only runs triggered by an authorized CI/CD pipeline (e.g., GitHub Actions) can be executed on your private worker pools.

Security Implications: Without signed runs, there is no cryptographic guarantee that the code being deployed was authorized by your CI/CD pipeline. Signed runs ensure the provenance and integrity of infrastructure code before it is applied by Spacelift, preventing unauthorized or manually triggered runs from executing on your private worker pools.

Prerequisites:

- Code must be stored in GitHub (monorepos are not currently supported).
- A private worker pool must be configured.
- A Push policy must be created using the provided policy body.
- The signing secret must be configured in both GitHub Actions secrets and the Initialization policy.

Reference: https://github.com/spacelift-solutions/plugin-signed-runs

10. State Management

10.1 Terraform/OpenTofu State Storage

Setting: State storage backend

Location: Stack Settings

Recommended Configuration: For maximum control, use external state storage (e.g., an S3 backend in your own AWS account with encryption, versioning, and access logging).

Security Implications: Terraform state files may contain sensitive information such as resource identifiers, IP addresses, and occasionally secrets. When managed by Spacelift, state is encrypted at rest. With external state storage, you retain full control over encryption keys, access policies, and retention.

Reference: https://docs.spacelift.io/vendors/terraform/state-management

11. Account Decommissioning

11.1 Decommissioning Top-Level Administrative Accounts

When an administrator leaves the organization or an account must be retired:

1. **Revoke SSO access:** Remove the user from the relevant IdP groups so they can no longer authenticate to Spacelift.
2. **Delete user's MFA keys:** Navigate to Organization Settings → Multi-Factor Authentication, find the user, and delete their registered security keys.
3. **Revoke explicit role bindings:** Remove any direct user role bindings in Organization Settings → Identity Management.
4. **Rotate API keys:** If the departing admin had knowledge of any API key secrets, rotate or delete those keys immediately.
5. **Update Login Policies:** If using login policies, remove any individual allow rules for the user.
6. **Review audit trail:** Examine recent activity by the departing user to identify any unauthorized changes.

11.2 Decommissioning API Keys

1. Navigate to Organization Settings → API Keys.
2. Locate the key to be decommissioned.
3. Delete the key. This takes effect immediately — active sessions using this key are invalidated.
4. Verify via the audit trail that no further activity is recorded under the deleted key.

11.3 Decommissioning a Spacelift Account

To fully decommission a Spacelift organization:

1. Destroy all managed infrastructure via tracked runs or tasks.
2. Delete all stacks, contexts, policies, and integrations.
3. Delete all worker pools and decommission associated worker infrastructure.
4. Delete all API keys.
5. Remove all users.
6. Mark account for deletion in settings.
7. Contact Spacelift support and the Knox Compliance team to formally close the account and confirm data deletion.

12. API Capability (SCG-ENH-API)

Spacelift provides a comprehensive **GraphQL API** for viewing and adjusting all security settings programmatically. All operations available in the Spacelift UI can also be performed via the API, including:

- Managing users, API keys, and role bindings
- Creating and configuring Spaces, stacks, policies, and contexts
- Managing worker pools and cloud integrations
- Querying audit trail events
- Triggering and confirming runs

Additionally, the **spacectl** CLI tool and the **Spacelift Terraform Provider** enable infrastructure-as-code management of Spacelift's own configuration.

API Endpoint: `https://<your-account>.app.fedramp.spacelift.io/graphql`

Authentication: JWT bearer token obtained by exchanging an API key ID and secret, or via OIDC federation.

Reference: https://docs.spacelift.io/integrations/api

Version	Date	Description	Author	Approved by
1.0	February 23, 2026	Initial policy	Marcin Prokop (Security Engineer)	Wojciech Syrkiewicz-Trepiak (VP of Security)