ACCOUNT VENDING AS A PLATFORM PRODUCT: CONTROL TOWER, AFT, TERRAFORM, AND GOLDEN AWS LANDING ZONES

Naga Krishna Reddy Muppidi

Independent Researcher Email: nagamuppidi1015@gmail.com

Abstract
AWS account vending is often implemented as an automation task, but mature platform organizations need to treat it as a product with users, service levels, controls, feedback loops, and evidence. This paper presents a reference model for account vending using AWS Control Tower, Account Factory for Terraform, AWS Organizations, IAM Identity Center, security baselines, cost metadata, and developer portal integration. The model frames an AWS account as the primary unit of cloud governance and developer self-service. It defines account archetypes, guardrail categories, product metrics, and an evaluation method for comparing manual account creation with platform-product vending. The central claim is that golden AWS landing zones succeed when governance is packaged as a developer experience rather than imposed as a ticket queue.

Keywords AWS Control Tower, AFT, Terraform, landing zone, platform engineering, internal developer platform, governance

I. Introduction

Account vending is the hidden platform product behind successful AWS adoption. Developers may experience the platform through a portal, a Terraform module, or a CI/CD template, but the trust boundary is the AWS account. A well-vended account already contains identity federation, logging, network attachment, security services, cost tags, backup controls, and organization-level guardrails. A poorly vended account is a future migration project.

This paper argues that AWS account vending should be designed as a product, not as a provisioning script. The argument is timely because platform engineering became a central theme in the DORA research agenda , NIST CSF 2.0 elevated governance as a first-class cybersecurity function , and AWS Control Tower Account Factory for Terraform (AFT) offered a practical path for Terraform-centered account provisioning and customization . The paper contributes a reference architecture, product model, control taxonomy, and evaluation rubric for golden AWS landing zones.

II. From Account Creation to Account Product

Traditional account creation optimizes for initial speed: create account, attach to an OU, invite users, and hand it to a team. Platform-product account vending optimizes for the full lifecycle. The product includes intake, approval, account creation, guardrail attachment, network registration, identity assignment, cost allocation, security baselining, drift remediation, decommissioning, and evidence export.

The distinction matters because cloud operating models scale by repeatability. Control Tower supplies governance scaffolding ; Organizations and SCPs define hierarchical boundaries ; IAM Identity Center provides workforce access patterns ; and AFT turns account creation and customization into a Terraform-backed workflow . The platform team then wraps those primitives with a product interface: account archetypes, service-level objectives, ownership metadata, runbooks, and a roadmap.

L0.28Y Archetype & Baseline controls and customization scope

Application account & Standard VPC attachment, logging, security services, cost tags, CI/CD role, and app-team admin role with boundaries.

Data account & Stronger KMS, Lake Formation or data perimeter hooks, tighter egress, and data-classification tags.

Sandbox account & Budget caps, restricted regions, ephemeral lifecycle, and reduced production connectivity.

Shared services account & Centralized networking, artifact repositories, identity integrations, and highly reviewed cross-account access.
Regulated account & Additional Config conformance packs, evidence export, backup policy, and stricter change approval.

III. Reference Architecture

The architecture begins with a developer portal or request repository. A requester selects an archetype, business unit, data classification, environment, connectivity model, budget owner, and technical owner. The request generates an AFT account request file. AFT creates the account, moves it into the correct OU, runs global customizations, and then applies account-specific customizations. Service Catalog, CloudFormation StackSets, or Terraform modules can apply shared resources .
Account vending product flow for AWS landing zones.
The account baseline should include centralized CloudTrail, Config, Security Hub, GuardDuty, identity assignments, mandatory tags, budgets, backup policy, region restrictions, baseline KMS policy, and log archive destinations . Landing Zone Accelerator can complement Control Tower when the organization needs a more opinionated multi-account baseline for regulated environments . The platform team should make these baselines visible in a catalog, not hidden in a private repository.

IV. Governance as Developer Experience

The core product challenge is making governance feel like a paved road rather than a gate. NIST CSF 2.0 emphasizes governance outcomes, but those outcomes must be translated into developer-facing controls . For example, "asset inventory" becomes account metadata and tags; "protective technology" becomes SCPs, Config rules, and identity boundaries; "response" becomes Security Hub and incident-routing integration.
A good account-vending product offers transparent constraints. Developers should know which regions are enabled, which services are denied, how break-glass works, what cost center will be charged, how to request a new connectivity pattern, and which evidence is automatically collected. The platform should reject unsafe requests early, explain the rejection, and provide a safe alternative.

V. Control Taxonomy

The model uses five control categories. Preventive controls include SCPs, permission boundaries, region restrictions, and network guardrails. Detective controls include Config, CloudTrail, Security Hub, GuardDuty, and drift detection. Corrective controls include automated pull requests, account quarantine, baseline re-application, and exception expiration. Product controls include ownership metadata, portal documentation, service-level objectives, and feedback loops. Evidence controls include immutable logs, ticket links, baseline version, and compliance exports.
L0.34Y Signal & Interpretation
Provisioning lead time & Time from approved request to usable account. Measures product speed.
Baseline drift rate & Percent of accounts differing from their assigned archetype. Measures control durability.
Exception debt & Count and age of open exceptions. Measures governance health.
Developer rework & Number of rejected or revised requests. Measures clarity of the product interface.
Evidence completeness & Percent of accounts with inventory, logging, control, and owner evidence attached.

VI. Evaluation Design

A platform team can evaluate account vending using a before-and-after migration or an A/B comparison between manually created accounts and vended accounts. Primary outcomes are provisioning lead time, baseline drift, number of post-provisioning tickets, control coverage, and audit evidence completeness. Secondary outcomes are developer satisfaction, number of platform exceptions, and security-incident findings per account.
The evaluation normalizes by account archetype. Sandbox accounts provision faster and accept more automatic expiration. Regulated accounts carry more controls and longer lead time. The product is successful when the tradeoff is explicit, measurable, and accepted by users rather than negotiated ad hoc.

VII. Discussion

Account vending can fail in two opposite ways. It can become too centralized, requiring a platform team to approve every decision. It can also become too flexible, letting teams bypass the landing zone through custom Terraform. The product model balances these risks by offering opinionated archetypes with controlled extension points. Extension points should be versioned, documented, and reviewed like APIs.

The strongest organizational pattern is a platform team that owns the vending product and security teams that own control objectives, with application teams owning workload-specific risk. Team Topologies calls for reducing cognitive load and clarifying team interaction modes . In account vending, that means developers should not need to know every Control Tower, SCP, or Config detail to receive a safe AWS account. They should understand the product contract.

VIII. Implementation Notes for AWS Platform Teams

The most important implementation decision is where the platform API lives. Some organizations expose account vending through a portal such as Backstage; others use a GitOps request repository. The portal pattern improves discoverability and can integrate approval, documentation, and ownership data. The GitOps pattern improves auditability and lets platform engineers reuse existing code-review controls. A mature platform can support both by making the portal generate the same versioned request object that a Git workflow would accept.

Account metadata deserves the same rigor as infrastructure code. The request should include business owner, technical owner, cost center, data classification, application identifier, environment, expected lifetime, connectivity pattern, and exception references. That metadata should propagate into tags, inventory, budget configuration, security findings, and evidence exports. Without this metadata, Control Tower and AFT can create accounts quickly, but downstream governance will still depend on spreadsheets and tribal knowledge.

A second implementation decision is how to handle extensions. A golden landing zone cannot predict every account requirement. The platform should define supported extension points such as account-specific Terraform modules, StackSet attachments, VPC endpoint bundles, security exception objects, or IAM Identity Center permission-set assignments. Extensions should be reviewed through policy-as-code and should not be allowed to mutate the global baseline directly. This keeps the vending product flexible without letting every account become a fork.

Operationally, the platform team should publish a service catalog page for every account archetype. Each page should state provisioning lead time, included controls, excluded use cases, cost implications, break-glass process, decommissioning process, and escalation path. This documentation is part of the product, not a supplement. When developers understand the account contract, the platform team receives fewer ambiguous requests and security teams receive fewer late-stage exceptions.

IX. Threats to Validity

This architecture may vary across AWS organizations. Highly regulated enterprises may require additional approvals, network segmentation, data residency controls, or evidence formats that change the product design. Small organizations may find AFT and Control Tower more complex than their account count justifies. Finally, the term "golden" can hide risk: a golden baseline that is not maintained becomes an organization-wide stale dependency. The metrics expose that decay.

X. Conclusion

For AWS platform engineers, account vending is not a back-office activity. It is the foundation of developer self-service, security governance, cost allocation, and operational evidence. Control Tower, AFT, DORA platform-engineering research, NIST CSF 2.0, and maturing internal developer platforms make it reasonable to treat account vending as a formal product. The architecture gives teams a path to deliver golden AWS landing zones that are fast enough for developers and governed enough for regulated enterprises.

The views and opinions expressed in this article are solely those of the author and do not represent or reflect the views, positions, policies, or opinions of the author’s employer or any affiliated organization. The content is provided for informational purposes only. The employer makes no representations or warranties regarding

the accuracy or completeness of the content and does not endorse or accept responsibility for any statements, conclusions, or materials contained herein.

References

1. Amazon Web Services, "Overview of AWS Control Tower Account Factory for Terraform," AWS Control Tower User Guide, accessed 2024.
2. Amazon Web Services, "AWS Control Tower Account Factory for Terraform," GitHub repository, accessed 2024.
3. HashiCorp, "Manage AWS accounts using Control Tower Account Factory for Terraform," HashiCorp Developer documentation, accessed 2024.
4. Amazon Web Services, "AWS Control Tower user guide," AWS Documentation, accessed 2024.
5. Amazon Web Services, "AWS Organizations user guide," AWS Documentation, accessed 2024.
6. Amazon Web Services, "Service control policies," AWS Organizations User Guide, accessed 2024.
7. Amazon Web Services, "AWS IAM Identity Center user guide," AWS Documentation, accessed 2024.
8. Amazon Web Services, "AWS Security Reference Architecture," AWS Prescriptive Guidance, accessed 2024.
9. Amazon Web Services, "AWS Well-Architected Framework," AWS Documentation, accessed 2024.
10. Amazon Web Services, "Landing Zone Accelerator on AWS implementation guide," AWS Documentation, accessed 2024.
11. Amazon Web Services, "AWS Service Catalog administrator guide," AWS Documentation, accessed 2024.
12. Amazon Web Services, "AWS CloudFormation StackSets," AWS Documentation, accessed 2024.
13. Amazon Web Services, "AWS Config developer guide," AWS Documentation, accessed 2024.
14. Amazon Web Services, "AWS Security Hub user guide," AWS Documentation, accessed 2024.
15. Amazon Web Services, "Amazon GuardDuty user guide," AWS Documentation, accessed 2024.
16. DORA and Google Cloud, "2024 Accelerate State of DevOps Report," 2024.
17. NIST, "The NIST Cybersecurity Framework (CSF) 2.0," NIST CSWP 29, 2024.
18. M. Souppaya, K. Scarfone, and D. Dodson, "Secure Software Development Framework (SSDF) Version 1.1," NIST SP 800-218, 2022.
19. Joint Task Force, "Security and privacy controls for information systems and organizations," NIST SP 800-53 Rev. 5, 2020.
20. U.S. Department of Commerce, NTIA, "The Minimum Elements for a Software Bill of Materials," 2021.
21. Open Source Security Foundation, "SLSA v1.0: Supply-chain Levels for Software Artifacts," 2023.
22. HashiCorp, "Terraform language documentation," accessed 2024.
23. Backstage Authors, "Backstage documentation," accessed 2024.
24. Cloud Native Computing Foundation, "Platform Engineering Maturity Model," 2023.
25. M. Skelton and M. Pais, Team Topologies. IT Revolution Press, 2019.
26. N. Forsgren, J. Humble, and G. Kim, Accelerate: The Science of Lean Software and DevOps. IT Revolution Press, 2018.
27. L. Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles, "A survey of DevOps concepts and challenges," ACM Computing Surveys, vol. 52, no. 6, 2019.
28. B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, Eds., Site Reliability Engineering. OReilly Media, 2016.
29. International Organization for Standardization, "ISO/IEC 27001:2022 Information security management systems," 2022.
30. OWASP Foundation, "Software Assurance Maturity Model," accessed 2024.