Bulletproof AWS Landing Zone CI/CD

Michael Ullrich

2023/11/29

Nuvibit AG, Switzerland
michael.ullrich@nuvibit.com – https://nuvibit.com

Abstract

In today's rapidly evolving cloud landscape, Infrastructure as Code (IaC) stands out as a transformative approach to cloud infrastructure management. This paper focuses on employing IaC, particularly through the use of Terraform, to streamline and secure the IaC CI/CD stack to manage AWS Landing Zone Core Accounts. We dissect and explore how IaC enables businesses to codify and automate the deployment of their cloud-based Landing Zone infrastructure, leading to a more secure, consistent, and scalable management of cloud resources. Emphasis is placed on the utilization of CI/CD pipelines to ensure that the provisioning of the Landing Zone Core resources adheres to the highest standards of efficiency and security.

Introduction

The Cloud Revolution has marked a significant shift in the digital transformation era, with mid-to-large scale enterprises increasingly adopting cloud platforms like Amazon Web Services (AWS) and Microsoft Azure. The move is driven not only by the inherent benefits of cloud computing—scalability, flexibility, and cost-efficiency—but also by the opportunity to harness emerging technologies such as Generative Adversarial Networks (GANs) for innovation. These advanced technologies, often found within cloud platforms, offer businesses a competitive edge.

The Challenge: As businesses embrace these cloud platforms and start to explore their vast capabilities, they often find themselves navigating a complex multi-account landscape. While offering greater control and granularity, this complexity presents significant governance, security, compliance, and cost management challenges.

The Solution: In the following article, I will focus on AWS. An AWS Landing Zone[1) 2)] is an architectural solution that streamlines the creation and operation of a multi-account AWS environment.

It offers centralized capabilities for cloud management, security, governance, and networking. Through this it allows workload teams to focus on business value rather than cloud infrastructure management, reducing cognitive load and driving business growth and transformation.

AWS Landing Zone: A Detailed Examination

AWS Landing Zones provide an architectural solution for creating and managing a multi-account AWS environment. They centralize critical aspects of cloud infrastructure management, including management, security, governance, and networking.[3) 4)]

[1)] See the AWS documentation on building landing zones: `https://docs.aws.amazon.com/de_de/prescriptive-guidance/latest/migration-aws-environment/building-landing-zones.html`

[2)] See the AWS Security Reference Architecture documentation: `https://docs.aws.amazon.com/prescriptive-guidance/latest/security-reference-architecture/architecture.html`

[3)] How BriteCore Improved Security and Scalability by Migrating Insurance Workloads with AWS Landing Zone: `https://tinyurl.com/4eamkzah`

[4)] Swiss Post Innovates on Secure Foundation with AWS: `https://tinyurl.com/yvp9jdem`

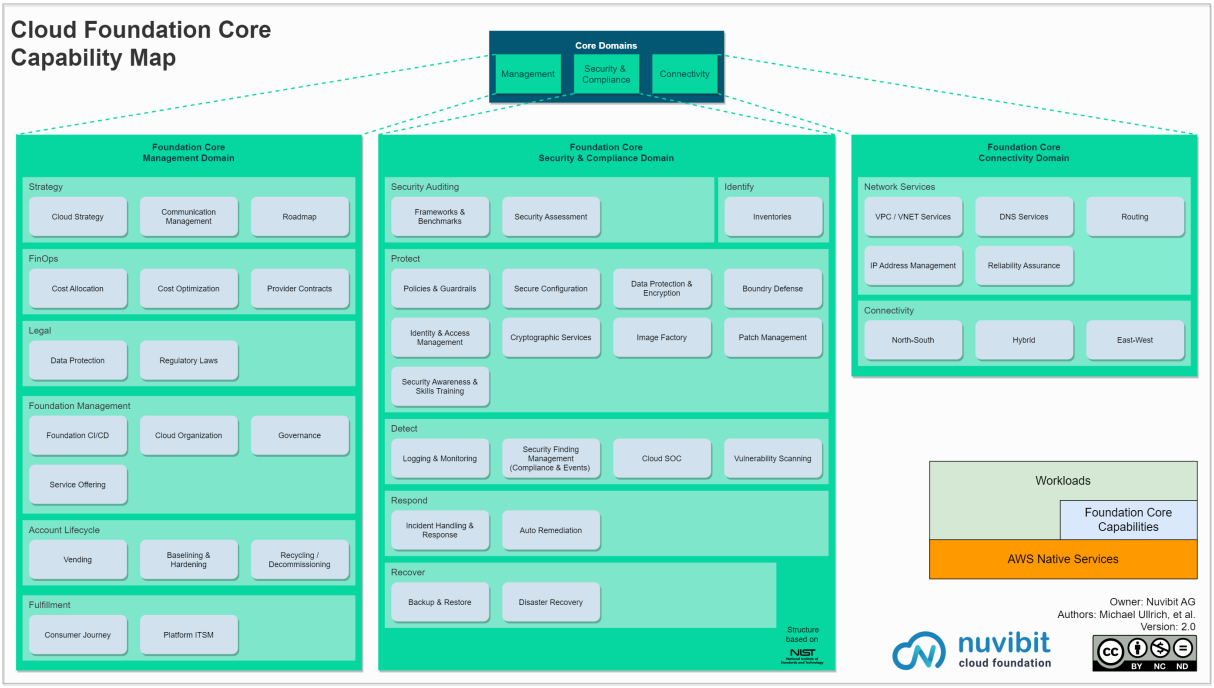

Figure 1: The Nuvibit Cloud Foundation Core Capability Map, showcasing the strategic, security, compliance, and connectivity domains integral to the operational excellence of cloud infrastructure. This map details the various components and processes that form the backbone of the Foundation Core, each contributing to the comprehensive support system for workload management and AWS native services integration.

In this article the centralized capabilities of an AWS Landing Zone are collectively referred to as the **Foundation Core**. The Foundation Core capabilities enable workload teams to concentrate on delivering business value rather than being bogged down by the complexities of cloud infrastructure management. By reducing the cognitive load on these teams, AWS Landing Zones facilitate business growth and transformation. For more information on the Foundation Core capabilities check the Nuvibit blog post on the Cloud Foundation Map.[5)]

The technical aspects of the Foundation Core capabilities are composed in dedicated Foundation Core AWS accounts. Below is an overview of Foundation Core Accounts typically established in an AWS Landing Zone environment - see Figure 2 for a visual representation or the Nuvibit blog post on the Reference architecture for AWS Multi-Account Customers.[6)]

- **Organization Management Account:** Oversees and manages the overall AWS Organization, including account structures, billing, and policies (e.g. SCPs) across the entire cloud environment. Also service delegation is managed here.
- **Core Vending Account:** Automates the creation and management of AWS accounts with predefined templates for consistency and policy adherence.
- **Core Security Account:** Serves as the hub for security management, handling identity management, threat detection, and compliance monitoring.
- **Core Log Archive Account:** Aggregates and stores logs securely for auditing and analysis, supporting compliance and investigations.
- **Core Networking Account:** Manages network infrastructure, including connectivity, DNS management, and on-premises to AWS connections.

The Foundation Core resources within an AWS Landing Zone are not limited to Core Accounts but extend across the entire AWS Organization, ensuring a cohesive and secure cloud environment. So the full set of Foundation Core resources can be categorized into two main groups:

[5)] Nuvibit Cloud Foundation Map: `https://nuvibit.com/blog/cloud-foundation-map/`
[6)] Reference architecture for AWS Multi-Account Customers: `https://tinyurl.com/4vc9u6eb`

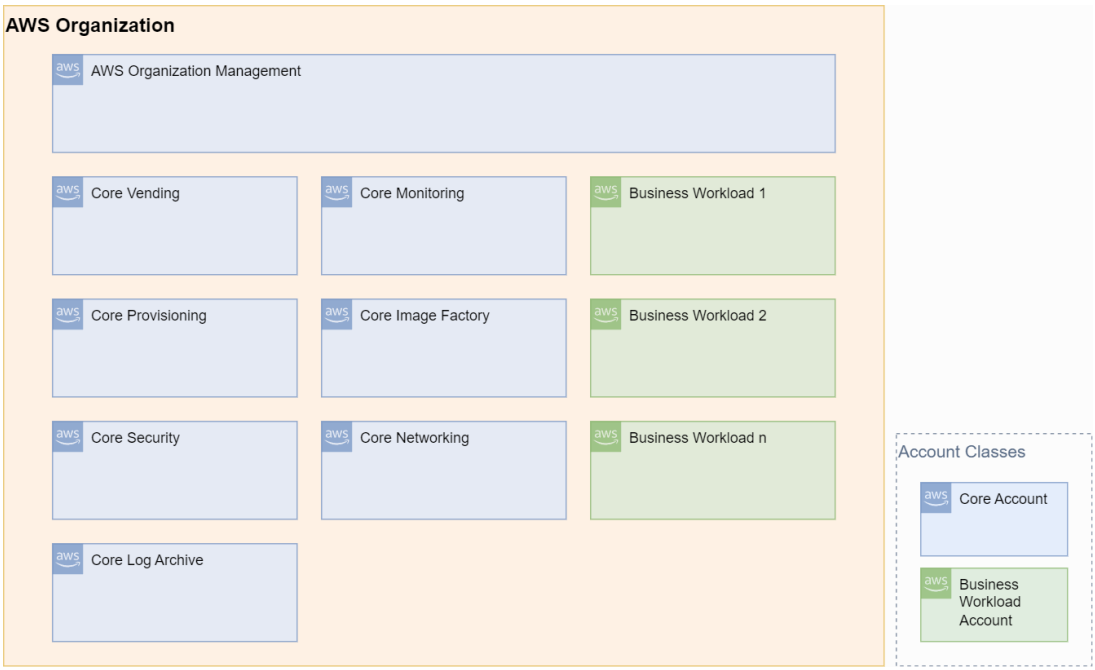

Figure 2: Sample of Foundation Core Accounts providing capabilities to the Business Workload Accounts.

- **Core Account Resources**: These consist of the essential infrastructure services housed within the Core Accounts. They are the backbone of the AWS Landing Zone, providing the necessary operational and security services for the entire cloud environment. These services include centralized logging, security monitoring, identity management, and shared network infrastructure.

- **Account Baselining Resources**: A comprehensive set of resources uniformly deployed across all AWS accounts in the Landing Zone. These resources go beyond security measures to also include standardized governance and networking configurations.

The layout and distribution of Foundation Core resources are visually represented in Figure 3. Here, an ”L”-shape demarcation symbolizes the encompassing nature of these resources, highlighting their presence and integration in every account within the AWS Organization. This visual aide serves to emphasize the comprehensive coverage and systematic approach of the Foundation Core within an AWS Landing Zone setup.

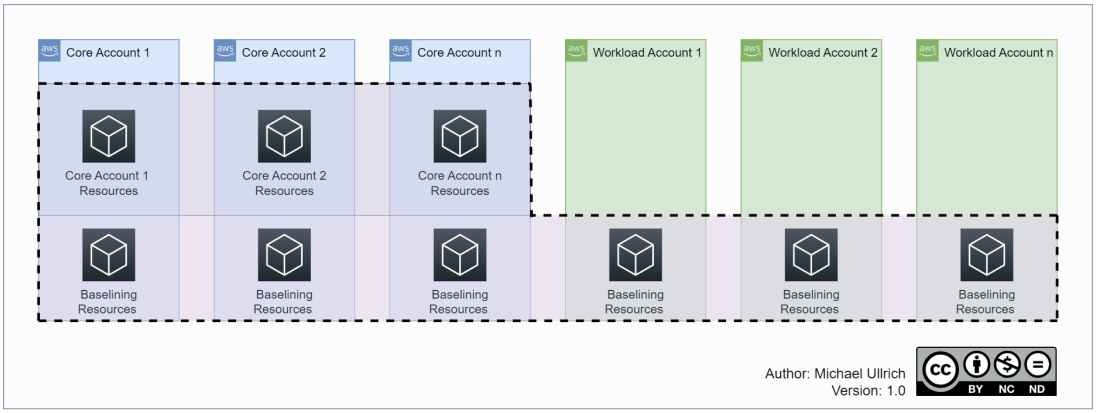

Figure 3: The ”L” visually highlights the extent of the Foundation Core resources within the AWS Landing Zone, encapsulating the critical infrastructure and account baselining resources necessary for a secure, standardized, and interconnected cloud environment.

Infrastructure as Code (IaC) in AWS Landing Zones

Best practices suggest using Infrastructure as Code (IaC) tools, like Terraform, for provisioning and managing cloud services. Benefits include:

- **Speed and Efficiency:** IaC automates the provisioning process, dramatically reducing setup and deployment time.
- **Consistency and Repeatability**: IaC ensures precise and consistent infrastructure deployment, reducing the potential for human error.
- **Agility and Scalability**: Facilitates quick provisioning, modification, or scaling of AWS resources.
- **Security and Compliance**: IaC integrates security directly into deployment workflows, enabling automated, policy-based compliance checks prior to provisioning. This pre-emptive approach ensures adherence to standards, safeguarding the infrastructure from compliance violations.
- **Collaboration and Governance**: Enhances teamwork with version control and provides comprehensive governance over infrastructure changes.

Enhancing IaC Deployment Through CI/CD Automation

The automation of Infrastructure as Code (IaC) deployment through Continuous Integration and Continuous Deployment (CI/CD) pipelines significantly streamlines the integration and delivery of infrastructure changes. By utilizing CI/CD tools for automatic updates, we achieve more consistent and error-free deployments, enhancing both the efficiency and security of our infrastructure operations.

The subsequent chapters will explore the essential features and capabilities of an enterprise-grade IaC CI/CD solution for the Foundation Core resources.

Principal Use Cases for Core Provisioning

Infrastructure as Code CI/CD Pipelines will orchestrate the deployment of the Foundation Core features. These features integrate various AWS Services and may extend across several Core Accounts. The deployment process encompasses the principal use cases shown in Figure 4:

- **Use Case 1**: A Core Pipeline deploys AWS resources to a single Core Account.
- **Use Case 2**: Multiple Core Pipelines are utilized to deploy AWS resources to a single Core Account.
- **Use Case 3**: One Core Pipeline manages the deployment of AWS resources to various Core Accounts.
- **Use Case 4**: A Baseline Pipeline deploys AWS resources consistently across all AWS accounts.

Composition of IaC Provisioning Pipelines

An effective IaC provisioning pipeline is characterized by its collaborative and modular structure, as illustrated in Figure 5. Key aspects include:

- Integration of features developed by disparate teams into a unified Core Pipeline, enhancing collective oversight and facilitating centralized updates.
- Composition of the Core Pipeline from multiple, independently maintained components, each housed in its own code repository for granular control and update management.

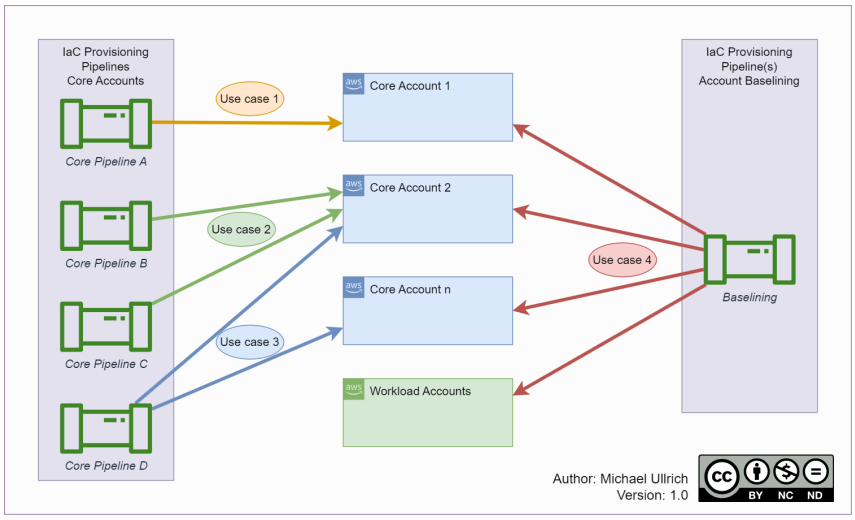

Figure 4: Foundation Core Provisioning Use-Cases: This diagram illustrates the various use cases for provisioning Foundation Core resources. Use case 1 demonstrates provisioning to a single Core Account, use case 2 shows multiple Core Pipelines to deploy to a single Core Account, use case 3 shows one Core Pipeline provisioning to various Core Accounts, while use case 4 represents the account baselining across all AWS accounts.

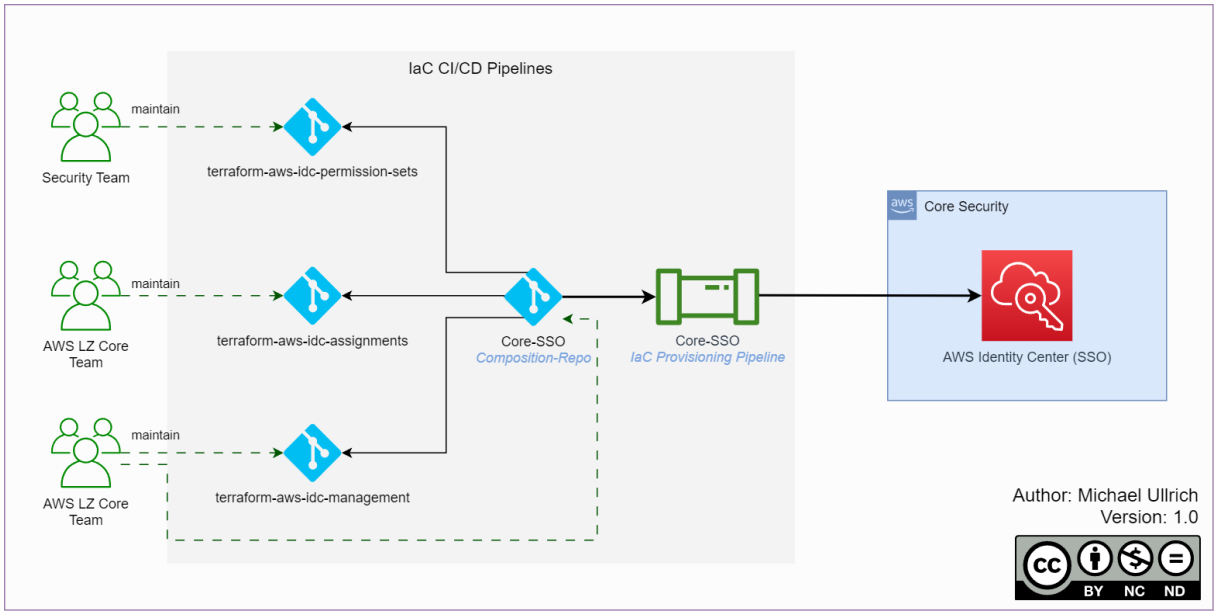

Figure 5: Illustration of collaborative IaC pipeline architecture, highlighting the convergence of multiple repositories managed by different stakeholder teams into a single, cohesive AWS Landing Zone feature deployment.

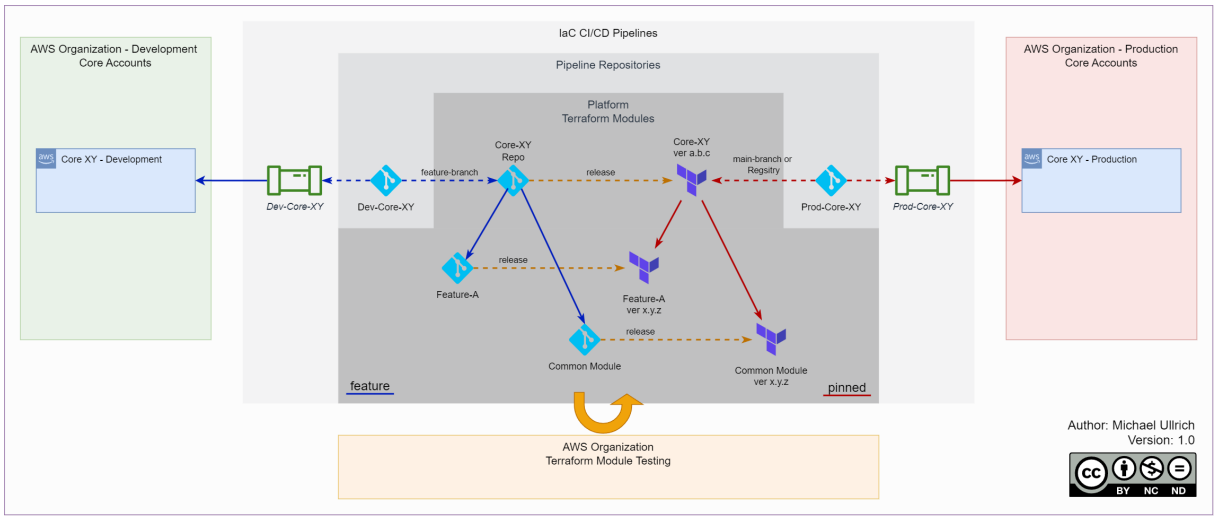

Figure 6: The cycle from Development to Production leveraging IaC CI/CD Pipelines.

Foundation Core IaC CI/CD Quality

To maintain the high standards of operational excellence within the Foundation Core environment, it is highly recommended to manage at least two separate AWS Organizations.[7)]

A strategy must be devised to enable the development and testing of new features within a dedicated development AWS Organization, with a subsequent and controlled transition of validated and approved features into the live production AWS Organization, as shown in Figure 6.

The deployment practices we adhere to for the Foundation Core Features are as follows:

- Deployment for Foundation Core Development:
  - Carried out directly from feature branches within the repositories.
  - Any merges into the main/release branch trigger the creation of a new module version within the Terraform registry, thereby generating an artifact.
- Deployment for Foundation Core Production:
  - Conducted exclusively using Terraform modules that have been officially released through a private Terraform registry and thoroughly tested.
  - Mandates a manual approval step that occurs between the execution of the Terraform plan and the actual application of the Terraform changes. For highly critical Core Accounts this step can be enhanced with the request of a one time password, as far as technically supported.

Securing the Foundation Core IaC CI/CD Stack

The Foundation Core IaC CI/CD stack, pivotal for provisioning and managing services within the AWS Landing Zone, is deemed a Tier-0 asset of the highest security criticality. As the central mechanism for orchestrating the deployment and ongoing management of Foundation Core resources, the stack's security is of utmost importance. Any compromise could potentially jeopardize the entire AWS Landing Zone, endangering associated workloads and data. Therefore, fortifying the Foundation Core IaC CI/CD stack is crucial to safeguard the integrity and confidentiality of the AWS infrastructure.

Designing a Robust Foundation Core IaC CI/CD Stack

This chapter details the foundational design principles for a resilient Foundation Core IaC CI/CD Stack. The intended architecture is depicted in Figure 7. On the left, the diagram presents the CI/CD components located outside of AWS, showcasing pipeline repositories that assemble feature modules and initiate the pipeline

[7)] You're probably missing an AWS environment...: `https://nuvibit.com/blog/aws-foundation-testing/`

process. The CI/CD pipeline gains access to the AWS environment by using Level-0 AWS Principals, which act as intermediaries to the Target Principals within the Core Accounts. It is within these Core Accounts that the core features are deployed.

Design Principals

- Level-0 AWS Principals should avoid of persistent credentials for the CI/CD pipeline.
- Level-0 AWS Principals must be centralized within a single account for stringent oversight and collective management.
- The Core CI/CD Account requires robust preventative controls (such as Service Control Policies) to facilitate headless operations.
- Core Provisioning Use Cases 1 through 3 must be adequately supported.
- There must be a facility for integrating multiple repositories, managed by disparate teams, into a singular Foundation Core feature deployable via the CI/CD Pipeline.
- The quality assurance workflow, as delineated in the previous section, should be seamlessly implementable.
- All Core CI/CD resources—including repositories, pipelines, and AWS resources—must be managed exclusively through IaC.
- Target Principals should be granted privileges on an individual basis to adhere to the principle of least privilege.
- As and when AWS provides the feature, Level-0 AWS Principals should be equipped with Multi-Factor Authentication (MFA), necessitating a one-time password (OTP) for each production pipeline execution.
- The Terraform state for each pipeline should be securely stored within the Core CI/CD Account, accessible only to the Level-0 Principal associated with that pipeline.

Reference Implementation

This chapter will outline the reference implementation using Azure DevOps as the CI/CD platform. The overview of the provisioned resources per Pipeline are shown in Figure 9.

Preparing the AWS Environment

As shown in Figure 8, the following initial preparatory steps are required:

- Vend and promote the Core CI/CD Account as one of the five delegated administrators for AWS CloudFormation.[8)]
- Configure an Amazon Elastic Container Service (ECS) within the new Core CI/CD Account to serve as the Azure DevOps Pipeline Agent.
- Create the IAM Role *core-cicd-provisioner-role*, establishing a trust relationship with the ECS task execution role *ado-agent-cicd-role*.
- Provision the IAM Role *core-cicd-stacksets-admin-role* to act as the execution role for Self-Managed CloudFormation StackSets.
- (Optional) If provisioning to the Organization Management Account is desired, provision the IAM Role *core-cicd-stacksets-execution-role*, with a trust to *core-cicd-stacksets-admin-role*, via CloudFormation. This is necessary as StackSets cannot deploy to the Organization Management Account as of November 2023.

[8)] CloudFormation StackSets delegated administration: `https://aws.amazon.com/blogs/mt/cloudformation-stacksets-delegated-administration/`

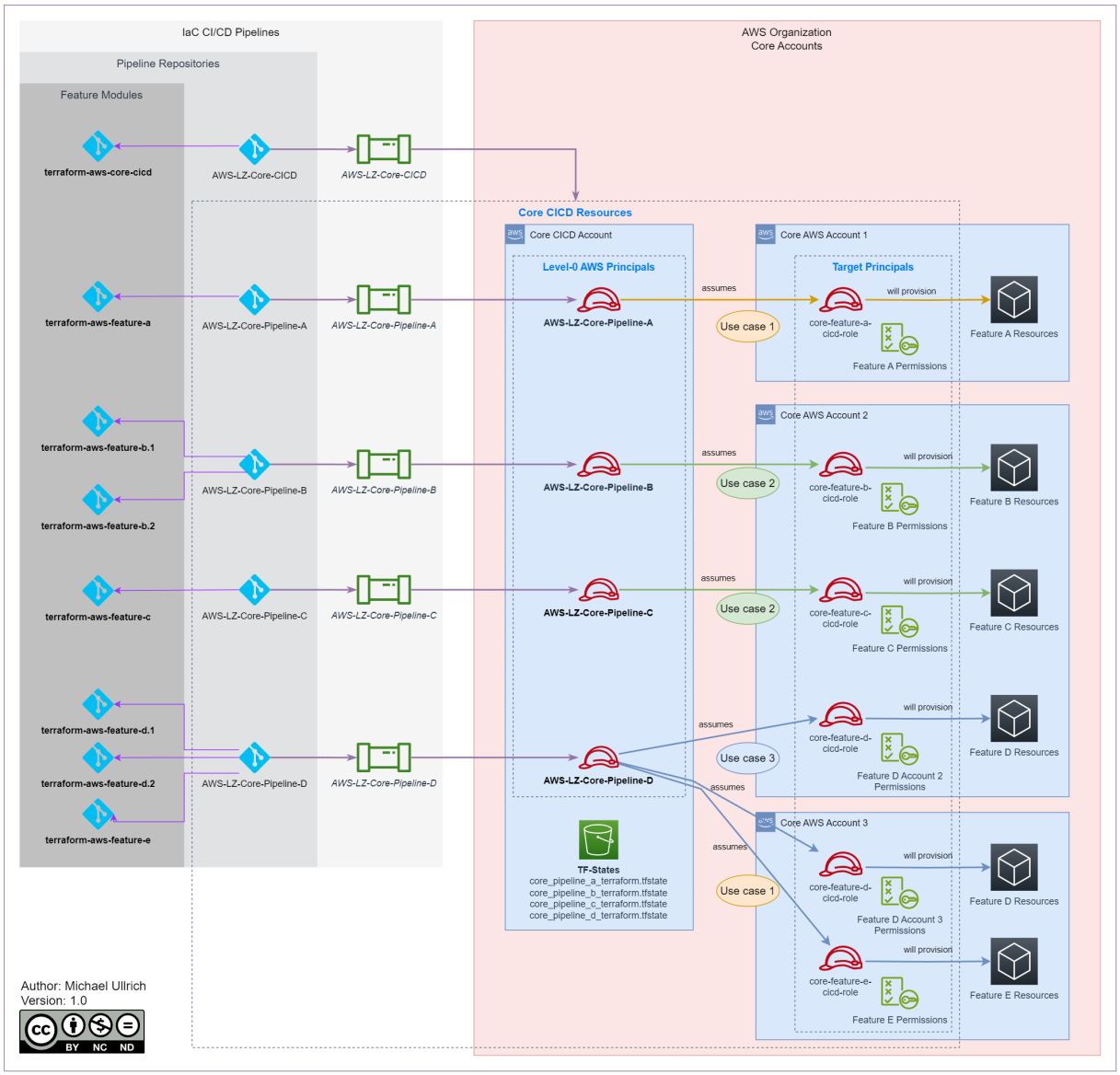

Figure 7: Target Architecture of the Foundation Core IaC CI/CD Workflow within the AWS Landing Zone: This diagram depicts the interplay between pipeline repositories, CI/CD pipelines, and the AWS Organization's Core Accounts. It illustrates how various Terraform modules within the repository are linked to their corresponding CI/CD pipelines, which in turn are connected to the core AWS accounts, demonstrating the flow for use cases 1 through 3 and the associated AWS Resource provisioning.

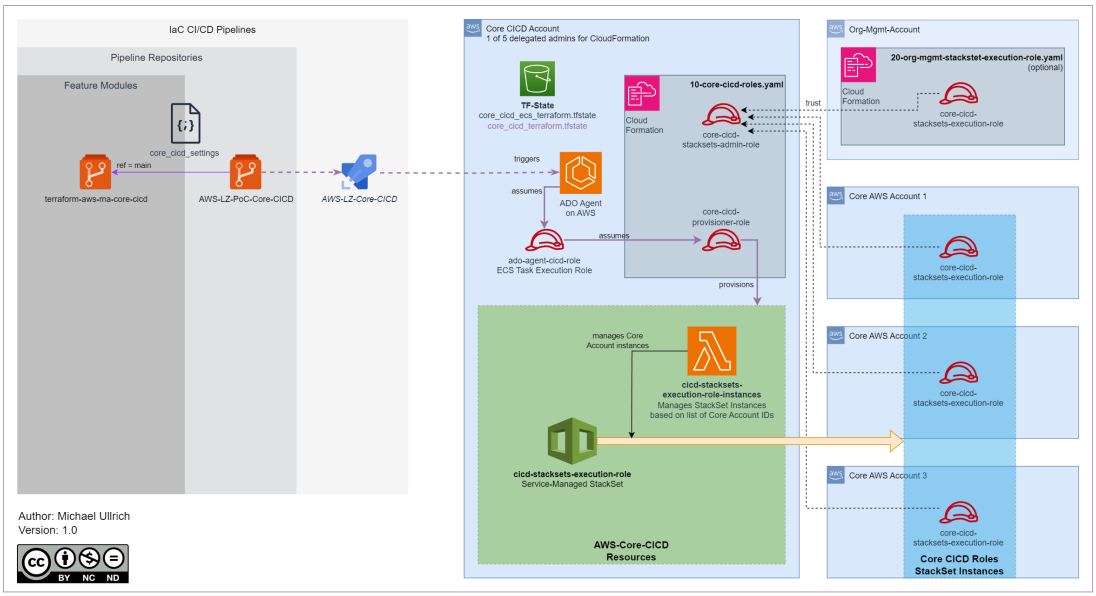

Figure 8: AWS Resource Prerequisites for the Proposed Solution

- Establish the CI/CD pipeline *AWS-LZ-Core-CICD* in Azure DevOps.
- Utilize this pipeline to provision the Service-Managed StackSet *cicd-stacksets-execution-role*, which in turn will create the *core-cicd-stacksets-execution-role* in the other Core Accounts.

IaC to Provision the Foundation Core IaC CI/CD Stack

The AWS provisioning process leverages Elastic Container Service (ECS) as the Azure DevOps Agent for pipelines, enabling the utilization of transient credentials to bolster security. The setup is directed by two Terraform HCL maps, elaborated upon in Section CI/CD Landscape Specification. The solution is realized through two main stages: AWS-Side and Azure DevOps-Side.

Setting up the AWS Environment

This phase involves configuring the IAM Roles that serve as Principals for the Azure DevOps Pipelines, enabling smooth CI/CD workflows. The AWS-side execution comprises the following procedures:

- The ECS Task's initial execution role (*ado-agent-cicd-role*) will assume the IAM Role *core-cicd-provisioner-role* to initiate operations.
- A Level-0 AWS Principal, as defined in the Terraform HCL map *core_cicd_aws_settings*, will be established for each Azure Devops Pipeline. This principal will gain access to the Terraform state storage.
- Each Target Principal within the Core Accounts will be allocated a corresponding Self-Managed StackSet that defines its permissions and associated trustee. Terraform will manage these StackSet instances in accordance with the configurations outlined in the HCL map *core_cicd_ado_settings*.

Deploying Azure DevOps Components

Subsequent to configuring AWS principals, the Azure DevOps Core Pipelines setup will proceed. The following steps will be processed in detail:

- Azure DevOps Repositories will be instantiated as delineated by the Terraform HCL map *core_cicd_ado_settings*, with one repository per pipeline.
- These Repositories will be preconfigured with the *azure_pipelines.yaml* files and aligned Terraform providers, targeting the specified AWS Core Accounts.

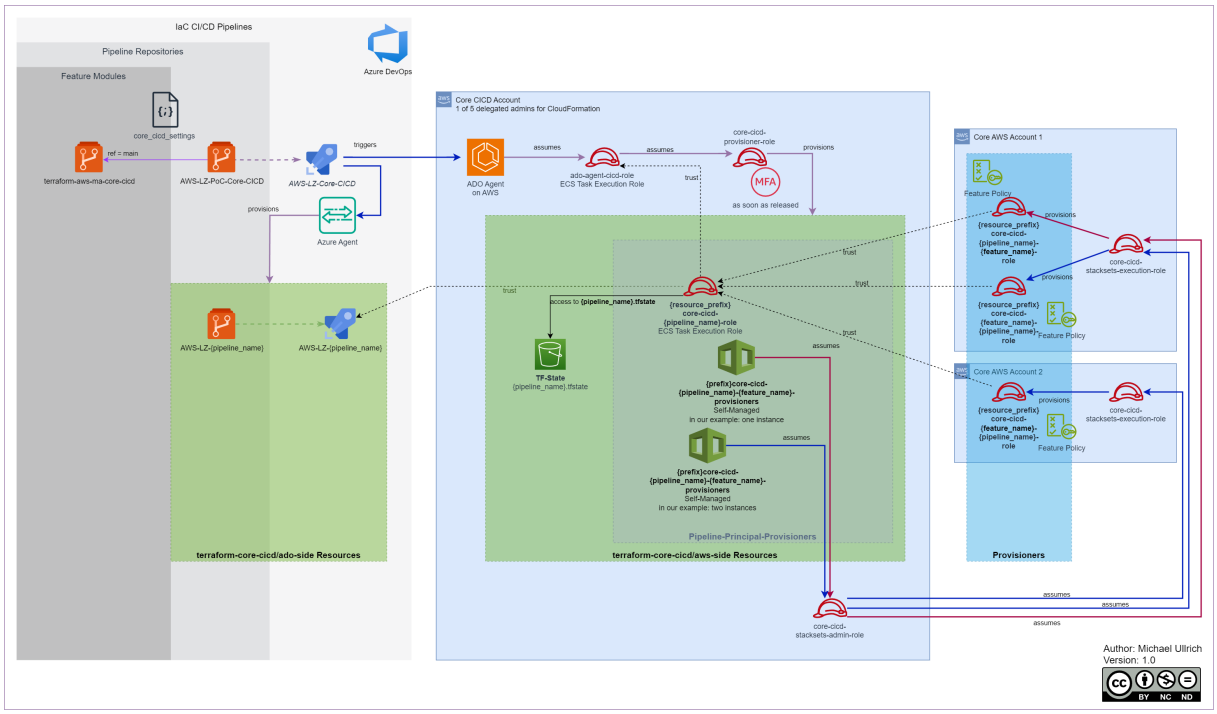

Figure 9: Configuration of AWS and Azure DevOps resources to implement the Foundation Core IaC CI/CD framework.

- Azure DevOps Pipelines will then be established, linked to their respective Repositories, and integrated with the AWS ECS-based Azure DevOps Agent to process build tasks.

- To meet the requirement to provision to two or more AWS Organizations (see Foundation Core IaC CI/CD Quality) multiple release Reposiotry branches and corresponding Pipelines can be specified.

Foundation Core IaC CI/CD Specification

Outlined below are the Terraform HCL maps that specify the AWS and Azure DevOps environments of the Foundation Core IaC CI/CD Stacks.

Core CICD Settings AWS-Side

```
core_cicd_aws_settings = {
  org_mgmt_account_id = "428460870630"
  pipelines = [
    {
      pipeline_name : "Core-Security-Controls"
      aws_targets : [
        {
          feature_name : "ScpManagement"
          cf_template_name : "scp_management.yaml.tftpl"
          accounts : ["core_security_controls"]
        },
        {
          feature_name : "SsoManagement"
          cf_template_name : "sso_management.yaml.tftpl"
          accounts : ["core_security_controls"]
        },
        {
          feature_name : "OrgCloudTrail"
          cf_template_name : "org_ct_management.yaml.tftpl"
          accounts : ["core_security_controls", "core_log_archive"]
        }
      ]
    },
    {
      pipeline_name : "Core-Security-Management"
      aws_targets : [
        {
          feature_name : "FullAccess"
          cf_template_name : "admin_access.yaml.tftpl"
          accounts : ["core_security_management", "core_monitoring
                "]
        },
        {
          feature_name : "SemperCoreSecurity"
          cf_template_name : "semper_core_security.yaml.tftpl"
          accounts : ["core_security_management"]
        },
        {
          feature_name : "SemperCoreLogging"
          cf_template_name : "semper_core_log_archive.yaml.tftpl"
          accounts : ["core_log_archive"]
        },
      ]
    }
  ]
  account_id_lookup : {
    "core_security_controls" : "870806180162"
    "core_security_management" : "806996015295"
    "core_log_archive" : "390555411271"
    "core_monitoring" : "971661981486"
    "core_provisioning" : "319689038059"
  }
}
```

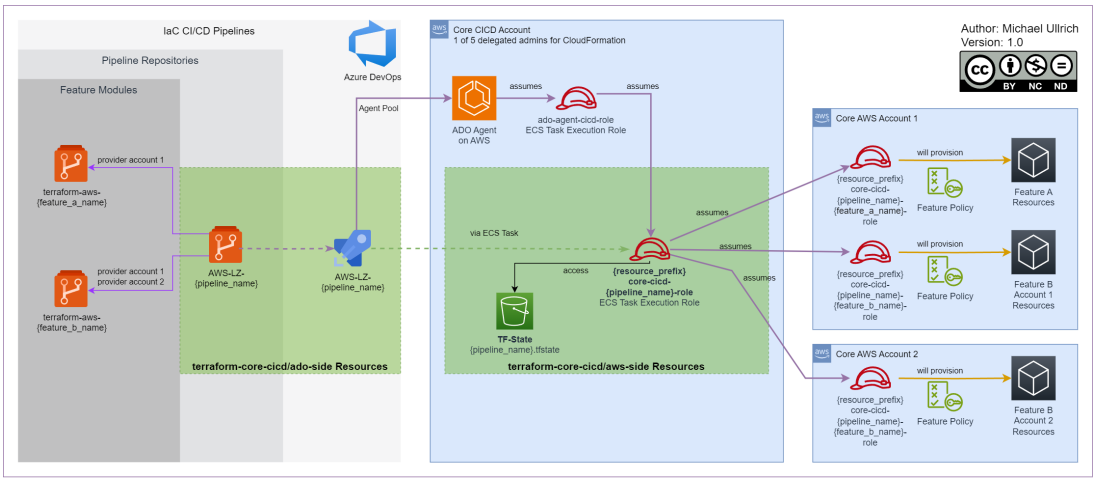

Figure 10: Demonstration of an Foundation Core IaC CI/CD Stack in operation.

Core CICD Settings Azure DevOps-Side

```
core_cicd_ado_settings = {
  project_name = "AWS-LZ"
  repo = {
    name_prefix = "AWS-LZ-"
  }
  pipeline_common = {
    pipeline_prefix     = "AWS-LZ-PoC-"
    aws_agent_pool_name = "CICD-CORE-POOL"
    access_to_variable_groups = [
      "ACCESS_TOKEN"
    ]
  }
  pipelines = [
```

```
    {
      pipeline_name : "Core-Security-Controls"
      tf_version : "= 1.3.9"
      aws_provider_version : "= 5.0.0"
    },
    {
      pipeline_name : "Core-Security-Management"
      tf_version : "= 1.3.9"
      aws_provider_version : "= 5.10.0"
    }
  ]
}
```

Operational View of a Core IaC CI/CD Stack

Figure 10 illustrates an operational Foundation Core IaC CI/CD Stack, endowed with the following functionalities:

- The Pipeline Repository consolidates two distinct Terraform Feature Modules, A and B.
  - Feature Module A is tasked with deploying AWS resources to AWS Core Account 1.
  - Feature Module B is designed to deploy AWS resources to both AWS Core Account 1 and AWS Core Account 2.
- The Pipeline execution occurs within AWS, conducted by the ECS-based Azure DevOps Agent.
- The pipeline will assume the corresponding Level-0 Principal.
- The Level-0 Principal is authorized to assume the least privileged Target Principals in the Core Accounts necessary for:
  - Deploying Feature Module A to AWS Core Account 1.
  - Deploying Feature Module B to AWS Core Account 1 and AWS Core Account 2.

Conclusion

The exploration within this document has provided an in-depth view of the pivotal role IaC plays in managing the CI/CD stack for Foundation Core Accounts. Through the application of IaC principles, utilizing tools such as Terraform, we have established a blueprint for a secure, repeatable, and scalable infrastructure. This framework not only upholds the structural integrity of Core Accounts but also paves the way for organizations to adopt a cloud strategy that is resilient, adaptable, and aligned with the best practices of automation, security, and compliance. As we continue to navigate through the advancements in cloud computing, the strategies detailed here will serve as a guide for organizations aiming to refine their cloud operations and harness the full potential of AWS services.

Glossary of Terms

Term	Description
AWS	Amazon Web Services, offering reliable, scalable, and inexpensive cloud computing services.
AWS Landing Zone	A solution that automates the setup of an AWS cloud environment that's secure, multi-account and based on best practices.
AWS Organization	An administrative service for managing multiple AWS accounts.
Foundation Core	The set of core services and resources that provide the underlying infrastructure for an AWS Landing Zone.
Foundation Core Account	An AWS account within a Landing Zone that houses the core services and resources, also known as Core Account.
Business Workload Account	An AWS account designated for running specific business applications and workloads.
Feature Modules	Reusable blocks of infrastructure as code that encapsulate a specific cloud feature or service.
Pipeline Repository	A version control repository where the code and resources for a CI/CD pipeline are stored.
Core Pipeline	A CI/CD pipeline responsible for deploying and managing the core infrastructure components.
Level-0 AWS Principal	A high-level AWS identity with extensive permissions used to manage infrastructure and services.
Target Principal	An AWS identity that is targeted by a pipeline for the deployment of resources.
Azure DevOps	A Microsoft Azure cloud service offering development tools for collaboration, testing, and releasing software.

Table 1: Glossary of commonly used terms in the context of AWS infrastructure and CI/CD processes.