Architecture_for_ThoughtMachine_Vault_sandbox_environment_at_GFT_v2

Last edited by **czerniga-gft** 8 months ago

Architecture for ThoughtMachine Vault sandbox environment at GFT v2

URL	Purpose	Security
https://argui.tm.blx-demo.com/	UI for ArgoCD	Public internet through GFT VPN IP
https://core-api.tm.blx-demo.com	TM Core HTTP API	Public internet through GFT VPN IP TODO: for integration with other services we might need to add custom egress IP here (NAT Gateway)
https://ops.tm.blx-demo.com	OPS dashboard for TM	Public internet through GFT VPN IP
https://saml-idp.blx-demo.com	SAML integration endpoint for SSO integration	Public internet through GFT VPN IP TODO: how this should work??
https://documentation.tm.blx-demo.com	Documentation for tm	Public internet through GFT VPN IP
https://tm-grafana.observability.tm.blx-demo.com	Grafana dashboard for TM	Public internet through GFT VPN IP
https://tm-alertmanager.observability.tm.blx-demo.com	Alert manager for TM	Public internet through GFT VPN IP
https://tm-metrics.observability.tm.blx-demo.com	Prometheus (Thanos?) endpoint for TM	Public internet through GFT VPN IP
kafka		b-2.tmkafka.pih9xm.c2.kafka.eu-central-1.amazonaws.com:9096,b-3.tmkafka.pih9xm.c2.kafka.eu-central-1.amazonaws.com:9096,b-1.tmkafka.pih9xm.c2.kafka.eu-central-1.amazonaws.com:9096
workflows-api.tm.blx-demo.com	Workflows API	
data-loader-api.tm.blx-demo.com		
audit-api.tm.blx-demo.com		
access-control-api.tm.blx-demo.com		

TM Service accounts

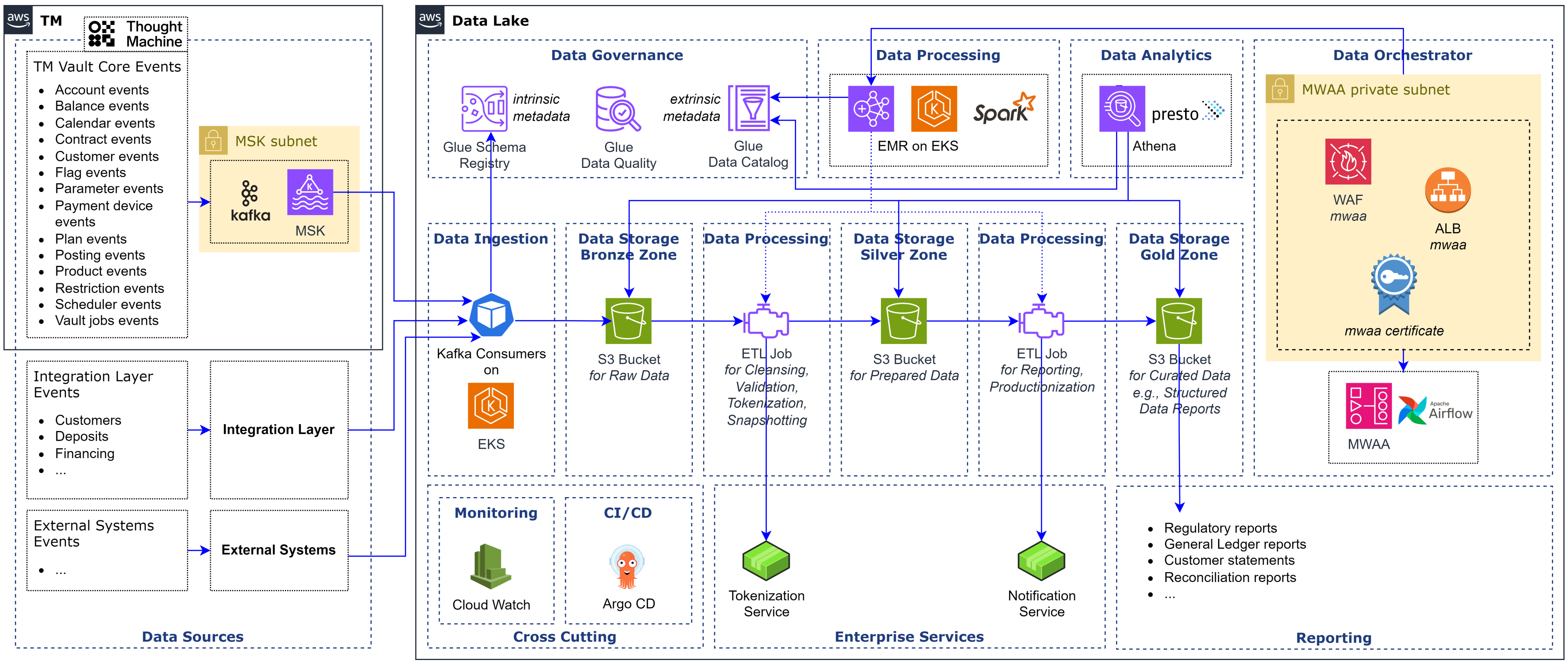
Pattern - please keep (country)-(context-name)-(optional subcontext name)

name	requestor - accountable	description	expected expiry
blx-upload-clu	Szczepankiewicz, Jarosław	For internal purpose of testing in / ex connectivity to TM vault instance	not determined. We expect this to exist as long as there is TM GFT instance required
pl-mb	Starobrat, Piotr		

High Level Architecture

Figure #1: TM

Figure #2: Data Lake



There is drawio file with the source of these both diagram...

Architecture Description

Overview

This architecture blueprint delineates the deployment topology for Thought Machine (TM) Vault Core on AWS, showcasing a strategic configuration of resources that bolsters operational efficacy, network organization, and service intercommunication. The design is predicated on the principles of high availability, distributed across multiple AWS availability zones, with an overarching emphasis on fortifying security, enabling comprehensive monitoring, and facilitating an automated deployment pipeline. Each component is meticulously selected and positioned to fulfill critical roles within the system's infrastructure, thus reinforcing the system's integrity, performance, and scalability.

Roles for Access and Core Operations

1. DevOps (Persona): Entrusted with the operational stewardship of the system, the DevOps Engineer is the sentinel of infrastructure integrity, orchestrating cloud resources, container management, and deployment automatization with precision. Leveraging tools such as AWS CloudFormation for infrastructure as code, this role is pivotal in maintaining system health, managing access, and executing security protocols to ensure uninterrupted performance and reliability.
2. Site Reliability Engineer - SRE (Persona): Anchored in the ethos of resilience, the SRE architects a robust infrastructure designed to scale gracefully under the flux of digital demand. Utilizing automation, such as AWS CloudWatch for system metrics and AWS Auto Scaling for resource elasticity, the SRE is the architect of disaster recovery strategies and the guarantor of high system availability.
3. TM Setup Engineer - TMSE (Persona): The TMSE orchestrates the foundational deployment and nuanced configuration of the TM Vault Core. With an acute focus on the initial setup and adaptive configurations, the TMSE ensures that the Vault Core is seamlessly integrated within the AWS ecosystem, leveraging services like Amazon RDS for database management, and sustaining the system's readiness for future expansion.
4. Smart Contract Developer (Persona): the Smart Contract Developer, is tasked with the formulation and validation of smart contracts.
5. Microservices (Component): Constituting a network of independent yet harmoniously interacting services, the Microservices component is a testament to modularity, communicating via HTTP and Streaming APIs. Developed, deployed, and managed individually, these services embody the agility of the system, supported by AWS Lambda and Amazon API Gateway for efficient operations and seamless service integration.

Public Hosted Zone Endpoints

- CI/CD APIs block
  - argui.tm.blx-demo.com - A portal to ArgoCD, fortified by security measures restricting public internet access, with connectivity provisioned through GFT VPN IP.
- Observability APIs block
  - tm-grafana.observability.tm.blx-demo.com - A dashboard for Grafana, facilitating real-time system monitoring.
  - tm-alertmanager.observability.tm.blx-demo.com - A centralized alert management interface.
  - tm-metrics.observability.tm.blx-demo.com - A Prometheus (Thanos) metrics collection endpoint.
- Core APIs block
  - saml-dev.tm.blx-demo.com - The SAML identity provider for secure Single Sign-On (SSO) integration.
  - core-api.tm.blx-demo.com - The fulcrum for the TM Vault Core HTTP API.
  - ops.tm.blx-demo.com - The operations dashboard centralizing system management.
  - documentation.tm.blx-demo.com - A repository for hosting comprehensive system documentation.

Deployment Zones

- TM - A holistic representation of the Thought Machine's cloud-based services, emphasizing a secure, multi-tenant environment.
- Shared Services - A suite of foundational services that provide cross-cutting capabilities to support the application environments within the architecture.
- Audit - An aggregation of monitoring tools, including AWS Security Hub and GuardDuty, offering overarching insights and security compliance reporting.
- Apps-[ENV] - The dynamic application environment designated by “[ENV]”, signifying the segregated deployment stages such as development, staging, and production. This zone is fortified with Amazon EKS, facilitating the orchestration of microservices which are integral to the system’s operations.
- Network, VPN, etc. - The network infrastructure, articulated through Amazon VPC and secured VPN access, forming the backbone for secure, scalable, and resilient communication channels.
- Data Lake - A centralized repository, architected on Amazon S3 and other AWS data management managed services, for storing structured and unstructured data at scale, primed for analytics and business intelligence.

Components and Systems

This section provides an overarching narrative of the compute and network framework within the AWS cloud environment, underscoring the strategic allocation of resources and the meticulous design considerations that underpin the deployment of the Thought Machine Vault Core.

Overview

- Public Hosted Zone: Manages DNS through Route 53 with distinct endpoints for Core APIs, Observability APIs, and CI/CD APIs, ensuring proper domain resolution and traffic routing.
- WAF and ALB: Web Application Firewall and Application Load Balancers provide security and load balancing across the DMZs in public subnets.
- Certificates: SSL/TLS certificates provided by ACM for secure communication.
- TM Vault Cluster: The backbone of the deployment, running on EKS with Istio service mesh integrating TM Deployments, Observability, and CI/CD components.
- Observability Tools: Grafana, Alert Manager, Prometheus/Thanos for monitoring and alerts.
- CI/CD Tools: Argo UI and Argo CD manage deployments and cloud infrastructure, along with additional tools for DNS management, load balancing, secret management, and auto-scaling.
- IaC Tools: Infrastructure as Code is managed by Terraform and Spacelift.
- Database and Messaging: Aurora DB and Apache Kafka (MKS) provide database and messaging services, with connections managed through pg-pool and direct links to the TM Vault Cluster.
- Secret Manager: Manages secrets for both the TM Vault Cluster and Aurora DB.
- Jumphost: A secure entry point for the TM Setup Engineer to access the EKS Cluster.
- VPC Configurations: The architecture spans multiple VPCs, segregating public subnets with DMZs, private subnets with the TM Vault Cluster, and separate subnets for Kafka and Aurora DB.

Infrastructure and Network Topology

- Availability Zones and VPC Configuration: The architecture is judiciously spread across three AWS availability zones, ensuring high availability and fault tolerance. Within this multi-AZ deployment, distinct VPCs are architected for discrete functional segments:

  - TM VPC: Hosts the core infrastructure of the Thought Machine, including a Demilitarized Zone (DMZ), or "Public Subnet", in each availability zone for secure exposure of public-facing APIs. Every Public Hosted Zone's APIs are judiciously routed through their dedicated DMZs, each fortified with Web Application Firewalls (WAF), Application Load Balancer (ALB), and dedicated SSL/TLS certificate, ensuring robust security and high availability.

    - for CI/CD APIs block it's WAF "sandbox-tm-argocd", ALB "k8s-argocd-argocdar", certificate "argui.tm.blx-demo.com".
    - for Observability APIs block it's WAF "sandbox-tm-monitoring", ALB "k8s-tmmonito-tmmonito", wildcard certificate "*.observability.tm.blx-demo.com".

        - for Core APIs block it's WAF "sandbox-tm-vault-core", ALB "k8s-istiosys-tmingres", wildcard certificate "*.tm.blx-demo.com".
    - Shared Services VPC: Encapsulates essential services that provide shared capabilities across the ecosystem, featuring a private subnet with a dedicated jump host to facilitate secure, administrative access for the TM Setup Engineer.
    - Apps-[ENV] VPC: Serves as the operational bedrock for the application environments. It is designed to host microservices within a private subnet, isolated from direct internet access, thus reinforcing the security and integrity of the application layer.
- Private Networking and Access Considerations: Within the secure confines of the private subnets, the EKS TM cluster operates alongside pivotal services such as Aurora DB/Postgres for resilient data storage and MSK/Apache Kafka for robust event streaming. This architecture is by design, devoid of direct internet access, to elevate the security posture of the infrastructure.
  A strategically positioned jump host in the EKS TM cluster’s private subnet affords the TM Setup Engineer a highly secure conduit for essential configuration and administrative operations. This dedicated access point is a cornerstone in maintaining the sanctity of the system’s internal mechanisms.
- Service Orchestration and Management: The cluster's intrinsic components, including TM Vault Deployments and ingress functionalities, are adeptly orchestrated by an Istio service mesh. Istio’s service mesh architecture introduces an abstraction layer for traffic management, empowering operators with fine-grained control and observability into the system's microservices without altering their codebase. The benefits of this approach are manifold, encompassing:
    - Enhanced Traffic Control: Istio provides dynamic routing capabilities that facilitate canary releases, A/B testing, and blue-green deployments, allowing for seamless updates and rollbacks.
    - Robust Security Postures: Through strong identity assertions, Istio fortifies inter-service communication, ensuring that all traffic is securely encrypted and authenticated within the mesh.
    - Observability and Monitoring: With rich telemetry data, Istio gives operators a comprehensive view of the service interactions, enabling prompt detection and rectification of issues.
    - Policy Enforcement: Istio allows for the uniform application of governance policies across the board, ensuring consistent adherence to critical operational parameters.
    - This orchestrated symphony of services, guided by Istio’s service mesh, exemplifies a mature and sophisticated approach to microservice management, pivotal for maintaining a resilient, secure, and scalable cloud-native application stack.
- Networking Services and Management: The TM's networking capabilities are augmented by an Elastic IP, labeled as "sandbox-tm-ngw-eip", which provides a static point of internet access within the AWS environment. All networking components are orchestrated under the vigilant management of AWS Route 53, offering a cohesive domain structure within "tm.blx-demo.com".
- Certificates and Secrets Management: Centralized management of SSL/TLS certificates is facilitated through AWS Certificate Manager (ACM), streamlining the deployment and renewal of certificates across services. AWS Secrets Manager (ASM) is employed to centralize the management of secrets, providing a secure repository for sensitive information such as credentials and API keys, thereby ensuring their confidentiality and integrity.
- Security and Compliance Considerations: Our architecture fortifies security and compliance through a trio of AWS services:
    - Inspector: Integrated within our master plane, AWS Inspector systematically evaluates our AWS resources, enhancing our security posture by identifying potential vulnerabilities and ensuring our deployment adheres to established security protocols.
    - Security Hub: As the aggregation point for security data, AWS Security Hub consolidates findings from across the architecture, providing a unified dashboard for security alerts and compliance status. This aids in swift identification and management of potential security issues within our ecosystem.
    - GuardDuty: Employed for its vigilant threat detection capabilities, AWS GuardDuty monitors operational data for anomalous behavior indicative of security threats or unauthorized actions, ensuring continuous monitoring and immediate response.
    - These services collectively ensure our architecture is not only robust against threats but also maintains compliance with evolving security standards.
- Data Management and Secrets StoreOur architecture ensures the integrity and confidentiality of data management through the strategic placement of our data stores and stringent access controls:
    - Aurora Database / Postgres: Positioned within a private subnet, our Aurora Database provides high-performance data storage solutions. Its isolation from public access reinforces security and supports our robust data management practices.
    - MKS / Apache Kafka: Also nestled in a private subnet, Apache Kafka handles our data streaming requirements with an emphasis on data integrity and secure transmission, ensuring reliable and secure data flow across our services.
    - Secrets Management: AWS Secrets Manager is implemented to safeguard sensitive information. It is intricately configured to manage secrets for the TM Vault Cluster and Aurora Database, automating the rotation and management of credentials, thus bolstering our data security framework.
    - This combination of secure data storage and stringent secrets management underscores our commitment to a secure, resilient, and compliant data architecture.

TM Vault Cluster

The TM Vault Cluster, a critical component of our deployment, is hosted within Amazon Elastic Kubernetes Service (EKS) and operates exclusively in private subnets to enhance security and network isolation. The cluster is composed of several integral groups, each serving a distinct purpose within the infrastructure:

- TM Deployments:
  - Vault Core Components: The backbone of the TM Vault, these components are the primary actors in processing and managing financial transactions. They are architected for high availability and fault tolerance across multiple EKS nodes.
  - State Management: Utilizes Amazon RDS with Aurora PostgreSQL, ensuring transactional consistency and reliability. The use of Aurora offers benefits such as self-healing storage and enhanced performance.
- Observability Group:
  - Grafana: Deployed to provide a comprehensive visualization of metrics, enabling real-time and historical data analysis for system monitoring.
  - AlertManager: Configured within the cluster to handle alerts generated by monitoring services, streamlining the notification process for operational events.
  - Prometheus/Thanos: Integrated for high-fidelity monitoring and long-term metric storage, facilitating a robust monitoring solution that scales with the cluster.
- CI/CD Services:
  - Argo UI: Provides a user-friendly interface for managing deployments, enhancing the transparency and control over CI/CD processes.
  - Argo CD: Automated continuous deployment tool used for managing Kubernetes applications, ensuring that applications are deployed as defined in Git, synchronizing application states, and enabling automated or manual deployments.
- Service Mesh with Istio:
  - The cluster leverages Istio as a service mesh, which sits at the network communication layer to control how different parts of an application share data with one another. This is not just limited to load balancing, but also includes detailed monitoring, a rich set of routing rules, and resiliency across the network.
  - Istio's integration with EKS simplifies observability, network traffic management, and security enforcement without requiring changes to the application code.
  - With Istio's capabilities, the TM Vault Cluster enjoys advanced traffic management, enhanced security features including end-to-end encryption, and access control policies that are pivotal for protecting sensitive financial data.
- Network Configuration and Security:
  - The EKS clusters are configured with strict network policies, ensuring that only authorized services can communicate with each other. This minimizes the risk of internal threats and provides a robust framework against network attacks.
  - The cluster's network architecture is designed to restrict internet access, with all inbound and outbound traffic tightly regulated by AWS Network Access Control Lists (NACLs) and Security Groups, providing a layered security approach.

This TM Vault Cluster represents a modern, containerized environment that is scalable, resilient, and secure, mirroring the agile and robust nature of cloud-native application architecture. The synergy between Amazon EKS and Istio service mesh forms a powerful foundation for deploying and managing the TM Vault's microservices effectively.

Information Flows

- EKS Cluster - TM VaultAccess and Core Operations
  - DevOps' access - for Persona
    - arrow #1.1 - Represents the secure connection from the DevOps personnel to the VPN gateway, initiating a secure session into the TM environment.
    - arrow #1.2 - Illustrates the pathway from the VPN to the Public Hosted Zone, through which DevOps can manage and monitor public-facing services.
  - Site Reliability Engineer/SRE's access - for Persona
    - arrow #2.1 - Depicts the Site Reliability Engineer accessing the VPN to engage in reliability and maintenance tasks within the TM environment.
    - arrow #2.2 - Showcases the connection from the VPN to the Public Hosted Zone for SRE to perform operations on publicly accessible endpoints.
  - TM Setup Engineer/TMSE's access - for Persona
    - arrow #3.1 - Connects the TM Setup Engineer to the Shared Services Jumphost, providing a secure management point for shared resources.
    - arrow #3.2 - Links the TM Setup Engineer to the TM Vault's dedicated Jumphost within the EKS cluster, facilitating direct access to the core EKS infrastructure.
  - Smart Contract Developer's access - for Persona
    - arrow #4.1 - Signifies the Smart Contract Developer's entry to the VPN, establishing a secured link to the development tools.
    - arrow #4.2 - Points from the VPN towards the Public Hosted Zone, enabling access to APIs necessary for smart contract deployment.
  - Apptications' access - for Components
    - arrow #5.1 - From external entities to the TM's network infrastructure and gateway "network-tm-tgw", signifying ingress traffic into the TM environment.
    - arrow #5.2 - Directs traffic from "network-tm-tgw" to the Apps-[ENV]'s VPC and into the private subnet where the "EKS Cluster - Apps" resides.
    - arrow #5.3 - Apps" microservices consuming HTTP API towards the Public Hosted Zone, indicating the service exposure to the HTTP endpoints.

        - arrow #5.4 - From "EKS Cluster - Apps" microservices consuming Streaming API to TM's MKS/Apache Kafka, outlining the internal data streaming mechanisms.
- Deployment and Configuration Management
    - CI/CD Flow - to manage cloud infrastructure
        - arrow #6.1 - Indicates the pathway from CI/CD APIs in the Public Hosted Zone through the DMZ, secured by WAF "sandbox-tm-argocd" and routed through ALB "k8s-argocd-argocdar", secured with the certificate "argui.tm.blx-demo.com".
        - arrow #6.2 - Connects the CI/CD APIs DMZ to the "EKS Cluster - TM Vault" where the CI/CD tools like Argo UI reside.
        - arrow #6.3 - Represents the internal process flow from Argo UI to Argo CD, detailing the CI/CD pipelines' interactions within the EKS cluster.
        - arrow #6.4 - Extends from the CI/CD zone to the Infra as Code (IaC) components like Terraform and Spacelift, emphasizing the use of IaC methodologies.
    - Observability Flow
        - arrow #7.1 - Maps the traffic from Observability APIs in the Public Hosted Zone to its DMZ, secured by WAF "sandbox-tm-monitoring" and ALB "k8s-tmmonito-tmmonito", with wildcard certificate "*.observability.tm.blx-demo.com".
        - arrow #7.2 - From the DMZ to the "EKS Cluster - TM Vault" Observability zone, encompassing tools such as Grafana, AlertManager, and Prometheus (Thanos) for system insights.
    - TM Setup Flow - to manage TM Vault infrastructure
        - arrow #8.1 - Connects the external Thought Machine's private docker images registry docker.external.thoughtmachine.io to the TM Setup Engineer. This represents the flow of custom docker images that are required for the TM Vault setup.
        - arrow #8.2 - Leads from the TM Setup Engineer to the "EKS Cluster - TM Vault"'s Jumphost, indicating a secure administrative path for the TM Setup Engineer to access the EKS Cluster.
        - arrow #8.3 - From the "EKS Cluster - TM Vault"'s Jumphost to the TM Operator within the cluster. This arrow signifies the operational management activities performed by the TM Operator, such as deploying updates or patches to the TM Vault.
    - Core Banking (Core API) Flow
        - arrow #9.1 - Shows the connection from the Public Hosted Zone's Core APIs block to the corresponding DMZ, which is secured by WAF "sandbox-tm-vault-core", routed through ALB "k8s-istiosys-tmingres", and protected with a wildcard certificate "*.tm.blx-demo.com". This flow demonstrates the secure exposure of the Core Banking APIs to the internet while ensuring data confidentiality and integrity.
        - arrow #9.2 - Extends from the Core APIs block's DMZ to the "EKS Cluster - TM Vault"'s TM Vault Deployments zone, specifically to the Istio Ingress hosted within the EKS Cluster. This illustrates the ingress of API traffic into the TM Vault, where Istio manages and routes the traffic to the appropriate services within the cluster.
- TM Vault System Operations
    - Istio Service Mesh Management
        - arrow #10.1 - Control flow from Istio to the TM Vault Deployments zone, representing command and control signaling.
        - arrow #10.2 - Data management flow from Istio to the TM Vault Deployments zone, depicting the data plane communication.
        - arrow #10.3 - Ingress management flow from Istio to the Istio Ingress, showing the routing of external traffic into the service mesh.
    - Secrets Management
        - arrow #11.1 - From the "EKS Cluster - TM Vault" to the AWS Secrets Manager, outlining the secure handling of all operational secrets.
        - arrow #11.2 - From the Aurora DB within the "EKS Cluster - TM Vault" to the Secrets Manager, ensuring database credentials are managed securely.
    - Persistence Flow
        - arrow #12.1 - Data flow from the "EKS Cluster - TM Vault" to the PostgreSQL connection pool "pg pool", indicating database interaction.
        - arrow #12.2 - From "pg pool" to the TM Vault's Aurora DB/Postgres, illustrating data persistence mechanisms.
    - Streaming Flow
        - arrow #13.1 - From the "EKS Cluster - TM Vault" to the TM Vault's MKS/Apache Kafka, denoting internal streaming data paths.
    - Container Management
        - arrow #14.1 - Connects the Shared Services' Jumphost to ECR within the Shared Services zone, enabling image caching from the private docker registry.
        - arrow #14.2 - From the "EKS Cluster - TM Vault" to ECR, allowing the TM Vault to pull both proprietary and custom images.
- Audit Operations
    - Audit Flow
        - arrow #15.1 - From the Shared Services' Jumphost to the AWS Inspector within the Shared Services zone for security assessments.
        - arrow #15.2 - From the "EKS Cluster - TM Vault" to the AWS Inspector for continuous security scanning.
        - arrow #15.3 - From the "EKS Cluster - Apps" to the Audit zone's AWS Inspector, Security Hub, and GuardDuty, signifying the comprehensive audit trail.
- Data Lake Integration
    - Data Ingestion Flow
        - arrow #16.1 - Data flow from TM Vault's Aurora DB/Postgres to the Data Lake, showing the ingestion of structured data.
        - arrow #16.2 - From TM Vault's MKS/Apache Kafka to the Data Lake, illustrating the stream of event data into the data analytics platform.

AWS Well-Architectured Framework Considerations

The AWS Well-Architected Framework provides a consistent approach for customers and partners to evaluate architectures and implement designs that can scale over time. Here we will consider the five pillars of the AWS Well-Architected Framework in relation to the TM Vault Core architecture on AWS:

- Operational Excellence:

    - Automation: The use of Terraform and Spacelift for Infrastructure as Code (IaC) suggests a high degree of automation in provisioning and managing resources, which is key to operational excellence.

  - Monitoring and Logging: The integration of Grafana, AlertManager, and Prometheus/Thanos within the Observability component indicates a robust monitoring and logging setup that can aid in performance and health tracking.

- Security:

  - Identity and Access Management: The architecture involves SAML integration for SSO, suggesting a strong identity management practice.
  - Data Encryption: The use of ACM for certificate management ensures that data in transit is encrypted.
  - Protection: The deployment of WAFs in the DMZs indicates a proactive approach to threat protection.
  - Data Security: The presence of AWS Secret Manager suggests that sensitive information is being managed securely.

- Reliability:

  - High Availability: The architecture spans multiple Availability Zones, enhancing fault tolerance and service availability.
  - Backup and Restore: While not explicitly mentioned, the use of Aurora DB typically comes with snapshot and rollback capabilities, which are essential for data reliability.
  - Service Discovery: The use of Istio service mesh implies that there is a mechanism in place for service discovery and management within the Kubernetes cluster.

- Performance Efficiency:

  - Compute Resources: EKS is used to manage Kubernetes services, which can be tuned for performance efficiency. The architecture also seems to leverage auto-scaling groups which can adjust to demand.
  - Load Balancing: Application Load Balancers (ALB) are used to distribute traffic, which can improve the performance and reduce latency.

- Cost Optimization:

  - Right-Sizing: The use of auto-scaling suggests that the architecture is designed to match demand, avoiding over-provisioning and reducing costs.
  - Resource Disposal: While not explicitly detailed, using IaC with Terraform allows for the precise management of resources, including their disposal when not needed.

- Additional Considerations:

  - Sustainability: Not directly addressed in the framework, but by optimizing resource usage, the architecture can contribute to AWS's sustainability goals.
  - Disaster Recovery: While details are not provided, the multi-AZ approach and data management practices should be aligned with a robust disaster recovery plan.

The architecture's adherence to the AWS Well-Architected Framework indicates a thoughtful and structured approach to cloud resource utilization, ensuring that the TM Vault Core is scalable, secure, performant, and cost-effective. However, to fully validate the architecture against the Well-Architected Framework, a detailed review using the AWS Well-Architected Tool or a similar assessment process would be necessary.

Attachments:

banklitex-tm-aws-architecture.drawio.svg (image/svg+xml) banklitex-tm-aws-architecture.drawio.png (image/png) architecture_platform_k8s_tm_v2.drawio.png (image/png) architecture_platform_k8s_tm_v2.drawio.png (image/png) architecture_platform_k8s_tm_v2.drawio.png (image/png) architecture_platform_k8s_tm_v2.drawio.png (image/png) architecture_platform_k8s_tm_v2.drawio.png (image/png) architecture_platform_k8s_tm_v2.drawio.png (image/png) architecture_platform_k8s_tm_v2.drawio.png (image/png) architecture_platform_k8s_tm_v2.drawio.png (image/png) architecture_platform_k8s_tm_v2.drawio.png (image/png) architecture_platform_k8s_tm_v2.drawio.png (image/png) architecture_platform_k8s_tm_v2.drawio.png (image/png) architecture_platform_k8s_tm_v2.drawio.png (image/png) architecture_platform_k8s_tm_v2.drawio.drawio (application/octet-stream) architecture_platform_k8s_tm_v2.drawio.png (image/png) architecture_platform_k8s_tm_v2.drawio.drawio (application/octet-stream) architecture_platform_k8s_tm_v2.tm.png (image/png) architecture_platform_k8s_tm_v2.drawio (application/octet-stream) architecture_platform_k8s_tm_v2.tm.png (image/png) architecture_platform_k8s_tm_v2.tm.png (image/png) architecture_platform_k8s_tm_v2.tm.png (image/png) architecture_platform_k8s_tm_v2.tm.png (image/png) architecture_platform_k8s_tm_v2.tm.png (image/png) architecture_platform_k8s_tm_v2.tm.png (image/png) architecture_platform_k8s_tm_v2.drawio (application/octet-stream) architecture_platform_k8s_tm_v2.tm.png (image/png) architecture_platform_k8s_tm_v2.tm.png (image/png) architecture_platform_k8s_tm_v2.drawio (application/octet-stream) architecture_platform_k8s_tm_v2.drawio (application/octet-stream) architecture_platform_k8s_tm_v2.tm.png (image/png) architecture_platform_k8s_tm_v2.tm.png (image/png) architecture_platform_k8s_tm_v2.dl.png (image/png) architecture_platform_k8s_tm_v2.dl.png (image/png) architecture_platform_k8s_tm_v2.dl.png (image/png) architecture_platform_k8s_tm_v2.dl.png (image/png) architecture_platform_k8s_tm_v2.tm.png (image/png) architecture_platform_k8s_tm_v2.dl.png (image/png) architecture_platform_k8s_tm_v2.dl.png (image/png) architecture_platform_k8s_tm_v2.dl.png (image/png) architecture_platform_k8s_tm_v2.tm.png (image/png) architecture_platform_k8s_tm_v2.dl.png (image/png) architecture_platform_k8s_tm_v2.dl.png (image/png) architecture_platform_k8s_tm_v2.drawio (application/octet-stream) architecture_platform_k8s_tm_v2.tm.png (image/png) architecture_platform_k8s_tm_v2.dl.png (image/png) architecture_platform_k8s_tm_v2.drawio (application/octet-stream) architecture_platform_k8s_tm_v2.tm.png (image/png) architecture_platform_k8s_tm_v2.drawio (application/octet-stream) architecture_platform_k8s_tm_v2.dl.png (image/png) architecture_platform_k8s_tm_v2.drawio (application/octet-stream) architecture_platform_k8s_tm_v2.dl.png (image/png) architecture_platform_k8s_tm_v2.drawio (application/octet-stream)