### **Knowledge Transfer(KT)Document : SRE**

#### Document Information

* Document Title: Tenable, Okta CDN, APIGEE, Mobile-api’s in PEO, BODT & OCI Infrastructure,Oracle fusion , Apex, Environments overview, HRIS architecture.
* Version:1.1
* Date:06th May 2024 - 10th Apr 2024
* Prepared By: SRE-Coforge
* Reviewed By: Satya Akella (Coforge)

Table of Contents

[**Knowledge Transfer(KT)Document : SRE** 1](#_Toc166506944)

[1. Overview of Security Management in SRE: 3](#_Toc166506945)

[1.1.1 Utilization of Tenable: 3](#_Toc166506946)

[1.1.2 Introduction of VR Module: 3](#_Toc166506947)

[1.1.3 Vendor Management for SaaS Products: 3](#_Toc166506948)

[1.1.4 Staying Up-to-Date with Stable Versions: 4](#_Toc166506949)

[1.1.5 Ownership and Accountability: 4](#_Toc166506950)

[1.1.6 Exception Handling with SPER: 4](#_Toc166506951)

[1.1.7 Vendor Collaboration for Remediation: 4](#_Toc166506952)

[1.1.8 Documentation and Communication: 4](#_Toc166506953)

[1.2 Vulnerability Management and Remediation: 4](#_Toc166506954)

[1.2.1 Vendor Interaction for Vulnerability Remediation: 4](#_Toc166506955)

[1.2.2 Daily Vulnerability Reports: 5](#_Toc166506956)

[1.2.3 Risk Scoring and Reporting: 5](#_Toc166506957)

[1.2.4 Rescan Option and Email Notifications: 5](#_Toc166506958)

[1.2.5 Security Management Workflow: 5](#_Toc166506959)

[1.2.6 Handling Exceptions and Critical Vulnerabilities: 5](#_Toc166506960)

[1.2.7 Emergency Vulnerabilities and Remediation Process: 5](#_Toc166506961)

[1.2.8 Vendor Communication and Upgrade Analysis: 5](#_Toc166506962)

[1.3. On going projects in Current Quarter(Q2) : 6](#_Toc166506963)

[1.3.1 Enhance App Deploy Process Post-Deploy: 6](#_Toc166506964)

[1.3.2 OCI Migration: 6](#_Toc166506965)

[1.3.3 Transforming to Cloud-Native Solutions: 6](#_Toc166506966)

[1.3.4 RabbitMQ Migration to Kubernetes: 6](#_Toc166506967)

[1.3.5 Dispose HRIS Tools & Observability Project: 6](#_Toc166506968)

[1.3.6 Okta Security Issues & SMS MFA Retirement: 6](#_Toc166506969)

[1.3.7 Okta Classic Engine to Okta Identity Engine Upgrade: 6](#_Toc166506970)

[1.3.8 DCP (Next-Gen Platform) Support & Benefit Plan Library: 6](#_Toc166506971)

[1.3.9 Encrypting Context.xml Passwords: 6](#_Toc166506972)

[2. Introduction to Okta CDN (Content Delivery Network): 7](#_Toc166506973)

[2.1.1 Environment Overview: 7](#_Toc166506974)

[2.1.2 Purpose of Okta CDN: 7](#_Toc166506975)

[2.1.3 CDN Setup: 7](#_Toc166506976)

[2.1.4 Static Content Hosting: 7](#_Toc166506977)

[2.1.5 Load Balancing and Health Checks: 7](#_Toc166506978)

[2.1.6 Impact Analysis and Testing: 8](#_Toc166506979)

[2.1.7 Uniform Content Delivery: 8](#_Toc166506980)

[2.2 Monitoring and Troubleshooting: 8](#_Toc166506981)

[3 Introduction to Apigee : 8](#_Toc166506982)

[3.1.1 Use Cases of Apigee: 8](#_Toc166506983)

[3.1.2 Authentication Mechanisms: 8](#_Toc166506984)

[3.1.3 Request Flow Overview: 9](#_Toc166506985)

[3.1.4 Infrastructure Overview: 9](#_Toc166506986)

[3.1.5 Developer Experience and Documentation: 9](#_Toc166506987)

[3.1.6 Operational Tasks: 9](#_Toc166506988)

[3.1.7 Analytics and Reporting: 9](#_Toc166506989)

[3.1.8 Operational Challenges: 9](#_Toc166506990)

[3.1.9 Conclusion: 9](#_Toc166506991)

[3.2 Overview of PEO Mobile API’s: 10](#_Toc166506992)

[3.2.1 Mobile Architecture: 10](#_Toc166506993)

[3.2.2 Authentication Workflow: 10](#_Toc166506994)

[3.2.3 Access and Documentation: 10](#_Toc166506995)

[3.3 Conclusion: 10](#_Toc166506996)

[4. Introduction to BODT and Oracle Cloud Infrastructure (OCI) 10](#_Toc166506997)

[4.1.1 Key Components and Integrations 10](#_Toc166506998)

[4.1.2 Operational Support and Troubleshooting 11](#_Toc166506999)

[4.1.3 Administrative Functions and Settings 11](#_Toc166507000)

[4.1.4 Conclusion 11](#_Toc166507001)

[4.3 Oracle Fusion: 12](#_Toc166507002)

[4.3.1 Bank key management within Oracle Fusion: 12](#_Toc166507003)

[4.3.2 Oracle Search Configuration 13](#_Toc166507004)

[4.4 Oracle Apex Overview 13](#_Toc166507005)

[4.4.1 Purpose and Functionality 13](#_Toc166507006)

[4.4.2 Administration and Access 14](#_Toc166507007)

[4.4.3Application Management 14](#_Toc166507008)

[4.4.4 Deployment Process 14](#_Toc166507009)

[4.5 Enterprise Planning and Budgeting Cloud Service (EPBCS) 14](#_Toc166507010)

[4.5.1 EPM Automate 14](#_Toc166507011)

[4.5.2 EPBCS Interface 15](#_Toc166507012)

[Conclusion : 15](#_Toc166507013)

[5. Overview of Environments : 15](#_Toc166507014)

[5.1 Performance Environment : 15](#_Toc166507015)

[5.2 Production Environment: 16](#_Toc166507016)

[5.3 Post-Production Environment (LTE1): 16](#_Toc166507017)

[5.4 Demo Environment (Demo1): 16](#_Toc166507018)

[5.4.1 Content Delivery Network (CDN) and Static Content for Demo1: 16](#_Toc166507019)

[5.4.2 Future Plans for Demo Environment: 16](#_Toc166507020)

[5.5 Overview of SL Environments: 16](#_Toc166507021)

[5.5.1 Infrastructure Configuration: 17](#_Toc166507022)

[5.5.2 Hosting and Technology Stack: 17](#_Toc166507023)

[5.5.3 Maintenance and Patching: 17](#_Toc166507024)

[5.5.4 Future Plans and Challenges: 17](#_Toc166507025)

[5.5.5 Rebuilding Process: 18](#_Toc166507026)

[5.6 Platform Environment Configuration: 18](#_Toc166507027)

[5.6.1 Maintenance Approach: 18](#_Toc166507028)

[5.6.2 Maintenance Responsibility: 18](#_Toc166507029)

[5.6.3 Future Plans and Challenges: 18](#_Toc166507030)

[6. HRIS Platform Overview: 18](#_Toc166507031)

[6.1 Authentication and Access: 19](#_Toc166507032)

[6.2 VPC Structure: 19](#_Toc166507033)

[6.3 Application Components: 19](#_Toc166507034)

[6.4 Infrastructure Deployment: 19](#_Toc166507035)

[6.5 Database Setup: 19](#_Toc166507036)

[6.6 External Communication and Security: 20](#_Toc166507037)

[6.7 Continuous Deployment (CD): 20](#_Toc166507038)

### Overview of Security Management in SRE:

* + Responsibility is to ensure the security of assets, including software and hardware.
  + Utilize tools like Tenable for security scans on all servers to identify vulnerabilities.
  + Critical vulnerabilities require collaboration with vendors or development teams for remediation.

### 1.1.1 Utilization of Tenable:

* + Tenable agents run on each VM, conducting security scans to detect vulnerabilities.
  + Detailed reports provide information on installed software, vulnerabilities, and remediation steps.
  + Collaboration with vendors or development teams for remediation, especially for critical vulnerabilities.

### 1.1.2 Introduction of VR Module:

* + VR module provides an overview of vulnerabilities categorized by severity.
  + Critical vulnerabilities must be remediated within specified timeframes to maintain security compliance.
  + Ability to mark false positives in the VR module with detailed explanations.

### 1.1.3 Vendor Management for SaaS Products:

* + For SaaS products like OKTA, vendors maintain security themselves as SREs don't have access for scanning.
  + OKTA and similar SaaS products are configured by SREs, but security is managed by the vendor.

### 1.1.4 Staying Up-to-Date with Stable Versions:

* + Aim to remain on stable versions released by vendors.
  + Typically aim for the stable version (N-1) to ensure reliability before adopting newer releases.

### 1.1.5 Ownership and Accountability:

* + Segregation of vulnerabilities by application or team responsibility.
  + Assignment of vulnerabilities for remediation and tracking in a centralized spreadsheet.

### 1.1.6 Exception Handling with SPER:

* + Exception requests (SPER) are submitted for vulnerabilities that cannot be remediated within the specified timeframe.
  + Detailed documentation required, including reasons for the exception and affected servers.
  + Approval process involves security team review before exemption is granted.

### 1.1.7 Vendor Collaboration for Remediation:

* + Collaboration with vendors for vulnerabilities related to vendor-provided software.
  + Vendor recommendations or fixes implemented based on severity and impact.
  + We need to act according to the SLA and fix the issue.

### 1.1.8 Documentation and Communication:

* + Maintaining records of vulnerabilities and remediation efforts in a centralized system.
  + Communication with vendors regarding identified vulnerabilities and their resolution status.

Overall, the process involves thorough scanning, identification, collaboration, and documentation to ensure the security compliance of assets managed by SRE.

Top of Form

### Vulnerability Management and Remediation:

### 1.2.1 Vendor Interaction for Vulnerability Remediation:

* + SREs communicate with vendors directly to address vulnerabilities.
  + Vendors create cases and provide fixes based on product details provided by SREs.
  + Vendors provide instructions or patches to remediate vulnerabilities.
  + SREs follow vendor instructions to apply fixes.

### 1.2.2 Daily Vulnerability Reports:

* + Vulnerability reports are received daily at 11:00 AM Central Time.
  + Scans run at 8:00 AM to ensure completeness of reports.

### 1.2.3 Risk Scoring and Reporting:

* + Reports include risk scores and ratings based on vulnerability severity (e.g high, medium, low).
  + Focus on high and medium-risk vulnerabilities.
  + Discrepancies between reports and module dashboards are noted for validation.

### 1.2.4 Rescan Option and Email Notifications:

* + In the dashboard, there's an option for rescanning to check if vulnerabilities are still present after fixes in lower environments, but Trinet SRE team do not have access to rescan as of now.
  + Currently, there's no functionality to send email notifications after rescans.
  + Work is in progress to enable this feature, which will likely involve launching rescans from Tenable rather than the VR module.

### 1.2.5 Security Management Workflow:

* + Scanning for vulnerabilities is essential after software updates or patches in various environments (e.g., dev, QA, prod).
  + Instant gratification is desired for scan results, but waiting for 24 hours is standard practice.
  + Ad hoc scans can be requested for immediate validation.

### 1.2.6 Handling Exceptions and Critical Vulnerabilities:

* + Exception requests may be necessary if critical vulnerabilities remain unresolved beyond 30 days due to production deployment constraints.

### 1.2.7 Emergency Vulnerabilities and Remediation Process:

* + Emergency vulnerabilities, such as the log4j issue encountered in the past, require immediate action.
  + Remediation involves collaboration between engineering and SRE teams to upgrade libraries, rebuild code, and redeploy applications promptly.
  + Emergency CRs are created and approved at the VP level for rapid deployment.

### 1.2.8 Vendor Communication and Upgrade Analysis:

* + Vendor recommendations for upgrades are considered, including stability and bundled fixes in the latest versions.
  + Proper analysis, testing, and performance checks are conducted before upgrading to the most recent stable versions.
  + Effective communication with OS teams is crucial for resolving ownership disputes and addressing vulnerabilities effectively.

This summary captures the key points discussed regarding vulnerability management, communication challenges, emergency remediation, and upgrade procedures within the SRE team.

### 1.3. On going projects in Current Quarter(Q2) :

### 1.3.1 Enhance App Deploy Process Post-Deploy:

* + Goal: Improve deployment process to reduce production issues.
  + Status: In progress and expected to complete in Q2.

### 1.3.2 OCI Migration:

* + Completed migrating from on-premises to Oracle Cloud (OCI).
  + Currently converting VMs to OCI native VMs for cost optimization.

### 1.3.3 Transforming to Cloud-Native Solutions:

* + Completed migration to Kubernetes, resolving issues with APIs.

### 1.3.4 RabbitMQ Migration to Kubernetes:

* + Goal: Migrate from VMs to containers on Kubernetes.
  + Status: In progress, expected to complete in Q2 or spill over to Q3.

### 1.3.5 Dispose HRIS Tools & Observability Project:

* + Goal: Evaluate and possibly replace tools like DataDog to save costs.
  + Status: Ongoing with the focus on cost optimization.

### 1.3.6 Okta Security Issues & SMS MFA Retirement:

* + Goal: Transition from Okta's SMS MFA to Twilio for improved security.
  + Status: Work in progress, involving platform security and engineering teams.

### 1.3.7 Okta Classic Engine to Okta Identity Engine Upgrade:

* + Goal: Upgrade Okta's classic engine to identity engine for enhanced security features.
  + Status: In the pipeline, involving heavy testing and resource identification.

### 1.3.8 DCP (Next-Gen Platform) Support & Benefit Plan Library:

* + Involvement in shadowing the DCP project for knowledge transfer and support.

### 1.3.9 Encrypting Context.xml Passwords:

* + Goal: Encrypt passwords in context.xml file to enhance security.
  + Status: Work in progress led by the SRE team.

Additionally, we discussed HR Passport application walkthrough of functionalities, including employee and admin views, and emphasized the importance of performance monitoring using AppDynamics.

Finally, we have observed a debugging scenario related to users getting logged out after login, focusing on API communication between services and backend database connections.

Overall, the projects span various aspects like cloud migration, security enhancements, tool optimization, and application support, with ongoing efforts to improve efficiency and reliability.

## 2. Introduction to Okta CDN (Content Delivery Network):

* + CDN stands for Content Delivery Network, which is a layer used to serve static content like the login page dashboard.
  + It ensures efficient delivery of static content to users, enhancing performance and reliability.

### 2.1.1 Environment Overview:

* + Different Okta environments are used for various purposes, including development, QE, stage, production.
  + Production environments are singularly served by respective Okta tenants, while non-production environments share tenants.

### Purpose of Okta CDN:

* + Okta CDN is introduced to address challenges in shared environments (Dev, Stage) where maintaining high availability and preventing cross-connections are crucial.
  + It ensures consistent access to the login page across shared environments, minimizing impact during maintenance or downtime.

### 2.1.3 CDN Setup:

* + CDN URLs are configured for different environments, such as **CDN devqe.hrpassport.com** for Dev QE and **CDN stg.hrpassport.com** for Stage.
  + Multiple CDN servers are maintained for load balancing and high availability, enhancing reliability.

### 2.1.4 Static Content Hosting:

* + Static content, including HTML, CSS, fonts, and JavaScript files, is hosted on CDN servers under designated folders like **apps/passport/HTTP/HTML**.
  + Content updates are versioned and carefully managed to ensure smooth deployment across environments.

### 2.1.5 Load Balancing and Health Checks:

* + Load balancers distribute traffic among multiple CDN servers to ensure uninterrupted service.
  + Health checks monitor server status, with servers marked offline if issues arise, ensuring continuous availability.

### 2.1.6 Impact Analysis and Testing:

* + Changes to the sign-in page code are carefully executed, considering potential impacts on critical environments like sales demos.
  + Thorough testing post-change ensures functionality and prevents disruptions, with immediate action taken in case of issues.

### 2.1.7 Uniform Content Delivery:

* + Changes made to the login page in one environment (e.g., Stage) are propagated uniformly across all environments sharing the same Okta tenant.
  + Accessing different environments results in fetching the same static content from CDN servers, ensuring consistency in user experience.

### 2.2 Monitoring and Troubleshooting:

* + Regular monitoring of CDN servers and load balancers helps identify and address issues promptly to maintain service reliability.
  + Troubleshooting involves diagnosing server health, connectivity, and configuration issues, ensuring minimal downtime.

Overall, Okta CDN plays a critical role in optimizing content delivery, ensuring high availability, and facilitating seamless user experiences across various Okta environments.

### Introduction to Apigee :

* + Apigee is an API management tool used for managing and exposing API’s securely to external clients.
  + API’s are crucial for businesses to expose their services to external developers for integration into various applications.

### 3.1.1 Use Cases of Apigee:

* + External developers integrate APIs into their applications for various purposes such as displaying holiday calendars or fetching company data.
  + Apigee facilitates secure access to internal APIs for external clients through authentication mechanisms like OAuth 2.0 and API keys.

### 3.1.2 Authentication Mechanisms:

* + Internal applications use API keys for authentication, while external clients use OAuth 2.0 with client ID and client secret.
  + OAuth 2.0 involves exchanging client credentials for access tokens, which are then used to authenticate requests to Apigee.

### 3.1.3 Request Flow Overview:

* + Requests from external clients are routed through Apigee, which forwards them to internal APIs after authentication.
  + Internal APIs process the requests and return the data, which is then sent back to the external clients through Apigee.

### 3.1.4 Infrastructure Overview:

* + Apigee manages the API proxies, which act as public endpoints for accessing internal APIs without exposing internal URLs.
  + Proxy configurations are set up to direct requests to internal API endpoints while maintaining security.

### 3.1.5 Developer Experience and Documentation:

* + Apigee generates developer portals from Swagger files, providing documentation and instructions for integrating with APIs.
  + Products and apps are managed within Apigee, allowing external developers to register and obtain access to APIs.

### 3.1.6 Operational Tasks:

* + Certificate management involves renewing SSL certificates for secure communication between Apigee and internal services.
  + Incident management includes troubleshooting issues such as service downtime or incorrect API key usage in testing environments.

### Analytics and Reporting:

* + Apigee provides metrics and reports on API usage, performance, and errors, aiding in monitoring and improving developer experience.
  + Custom reports and analytics help in identifying trends and optimizing API performance.

### 3.1.8 Operational Challenges:

* + Testing environments may experience frequent incidents due to deployments, troubleshooting, or service disruptions.
  + Troubleshooting involves identifying potential issues in API authentication, routing, or service availability.

### 3.1.9 Conclusion:

* + Understanding the request flow, authentication mechanisms, and operational tasks is essential for managing Apigee effectively and ensuring smooth API integration for external clients.

### Overview of PEO Mobile API’s:

### 3.2.1 Mobile Architecture:

* + The mobile architecture is similar to the regular web architecture, with mobile users accessing mobile.trinet.com through F5 load balancers.
  + Mobile web servers (MBL) host HTTPD and act as reverse proxies, pointing to internal mobile app servers running APIs such as API Mobile and API Mobile Ng.
  + Migration of mobile APIs from VMs to Kubernetes is ongoing, with API Mobile Ng taking precedence for certain functionalities.

### Authentication Workflow:

* + Authentication for mobile users involves the same components as for web users, including API Trinet and Okta.
  + API Auth plays a key role in verifying user existence and roles, ensuring a similar authentication workflow for both mobile and web users.

### 3.2.3 Access and Documentation:

* + Access to the mobile app for internal colleagues has been revoked, with only external clients retaining access.
  + Documentation for mobile app screens is unavailable, as the team no longer has access to it.

### 3.3 Conclusion:

Overall, the summary highlights the intricacies of the architecture, emphasizing the importance of understanding request flows, authentication mechanisms, and ongoing migrations.

### **4. Introduction to BODT and Oracle Cloud Infrastructure (OCI)**

BODT, or Back Office Digital Transformation, is an ongoing project aimed at migrating financial and HCM (Human Capital Management) applications from on-premises to the Oracle Cloud. The architecture primarily revolves around Oracle Cloud Infrastructure (OCI) and interfaces with on-premises systems, particularly focusing on Oracle Integration Cloud (OIC) and integrations with banks for financial transactions.

### **4.1.1 Key Components and Integrations**

* **OCI Tenants**: BODT operates within the TriNet OCI tenant, distinct from other tenants like the TriNet group. Multiple environments including dev, test, UAT, and prod are utilized.
* **Oracle Fusion**: A central component of the architecture, comprising various environments including Hyperion planning (EPBCS), Apex for custom applications, and the ATP database for automatic transaction processing.
* **Oracle Integration Cloud (OIC)**: Oracle Integration Cloud serves as the middleware for integrating various systems within the organization. It facilitates communication between Oracle Fusion and external systems, streamlining data exchange processes. Key functionalities within OIC include managing connections and integrations, ensuring smooth interoperability between disparate systems.

### **4.1.2 Operational Support and Troubleshooting**

* **Integration Management**: Monitoring and managing integrations within OIC, ensuring their status, scheduling, and execution. Troubleshooting integration failures and orchestrations when necessary.
* **Error Handling and Notifications**: Configuring error notifications and distribution lists within OIC, often linking to ServiceNow for automated ticket creation when integrations fail.
* **Environment Pairing**: Each OIC instance is paired with specific Oracle Fusion instances and ATP databases, enabling targeted communication and integration based on business needs.

### **4.1.3 Administrative Functions and Settings**

* **File Management**: Supporting file operations within OIC, such as creating directories for integrations that write files to the cloud server.
* **User Management**: Basic user viewing capabilities, although user access is primarily managed by the cloud team using IAM.
* **Configuration and Notification Settings**: Configuring IP addresses, ports, and distribution lists for error notifications, ensuring efficient communication and incident management.

### **4.1.4 Con**clusion

BODT's architecture, centered around OCI and OIC, enables seamless integration between various financial and HCM applications. With robust operational support and administrative functionalities, the system ensures smooth operation and efficient troubleshooting across different environments and integration points.

### 4.3 Oracle Fusion:

1. **Oracle Fusion:** Oracle Fusion is the heart of the Business Operations and Development (BODT) team, operating within the Oracle Cloud environment.
2. **Service Account Management:** It maintains service accounts within Fusion, ensuring passwords are rotated, accounts are active, and troubleshooting any issues. However, role assignments for these accounts are handled by a security administrator.
3. **Schedule Processes:** Similar to the Process Monitor in PeopleSoft, Fusion provides a tool to manage scheduled processes. This tool allows for monitoring and troubleshooting processes, including accessing logs for individual processes.
4. **Cloud Hosted Environment:** Fusion is hosted and managed entirely in the cloud, limiting access to server and database logs due to data sensitivity. However, access to individual process logs is available for monitoring and troubleshooting purposes.
5. **Integration of Financials and HCM:** Fusion integrates financial and human capital management (HCM) functionalities, eliminating the need for separate products like PeopleSoft Financials and HCM. This integration streamlines processes such as payroll and benefits administration, providing a unified solution for various business operations.

This provides a comprehensive overview of managing service accounts, troubleshooting processes, and understanding the integrated nature of Oracle Fusion within your organization's operations.

### 4.3.1 Bank key management within Oracle Fusion:

1. **Accessing Setup and Maintenance:** Navigate through the settings box to access Setup and Maintenance, then proceed to Payments and select Manage Transmission Configurations.
2. **Protocol Selection:** Within the Manage Transmission Configurations page, choose the appropriate protocol based on whether it's for outbound or inbound transactions. The primary protocols dealt with are Secure File Transfer Protocol (SFTP) for outbound transactions and Secure File Transfer Retrieval Protocol for inbound transactions.
3. **Bank Selection and Configuration:** Select the desired bank, such as Bank of America, KB Bank, or Wells Fargo. Enter the provided FTP server details and account information. Generate or upload the required public and private keys for secure file transmission. The private key is generated internally, while the public key is shared with the bank.
4. **Key Generation:** Use the Oracle Fusion product to generate key pairs. A "Quick Create" feature is available for convenient key generation. Each key pair is associated with a specific outbound transaction process, ensuring secure validation of transactions.
5. **Key Expiration Management:** Monitor the expiration dates of keys, both those generated internally and those provided by banks. Some keys may expire without warning, causing disruptions in transactions. Work on developing a better system for key management and proactive expiration alerts is underway.
6. **Key Storage and Alert System:** Currently, there's a need to establish a centralized system for storing keys and implementing alerts for impending key expirations. Efforts led by Avishek are focused on developing a more efficient key management system to ensure smoother transaction processes.

Overall, the process involves setting up secure connections with banks by configuring transmission protocols, generating or uploading key pairs, and managing key expiration to maintain uninterrupted transaction flows.

### **4.3.2 Oracle Search Configuration**

The Oracle Search enhances user experience by providing a unified search functionality across applications. Configuring and maintaining Oracle Search involves setting up indexes and scheduling processes to ensure accurate and efficient search results. Understanding this feature is essential for optimizing user productivity and system usability.

### **4.4 Oracle Apex Overview**

Oracle Apex, serving as a critical development tool within the organization's ecosystem, enables the creation of custom applications on the Oracle cloud platform. The platform is segregated into multiple environments, including test, UAT, and prod, facilitating a structured development and deployment process.

### **4.4.1 Purpose and Functionality**

One of the primary applications built using Oracle Apex is the "pay when paid" (PWP) dashboard, utilized by the financial team for critical financial insights. This dashboard, though inaccessible due to sensitive information, highlights the tool's capability in delivering robust solutions for complex business requirements.

### **4.4.2 Administration and Access**

Access to Oracle Apex is typically managed through workspace accounts, with administration responsibilities often shared between the SRE and DBA teams. While SRE involvement may be limited, understanding the tool's intricacies.

### **4.4.3 Application Management**

Upon logging into Oracle Apex, users are presented with various applications, each serving distinct purposes within the organization's workflow. These applications, including the PWP dashboard, and settlement dashboards, demonstrate the versatility and flexibility of Oracle Apex in addressing diverse business needs.

### **4.4.4 Deployment Process**

Oracle Apex facilitates the deployment of custom applications, managed through dedicated deployment accounts and workflows. While SRE involvement in this process may vary, maintaining familiarity with deployment mechanisms ensures seamless integration and operational efficiency.

### 4.5 Enterprise Planning and Budgeting Cloud Service (EPBCS)

EPBCS, short for Enterprise Planning and Budgeting Cloud Service, is a financial reporting and forecasting product replacing the legacy Hyperion system. It plays a crucial role in generating financial reports for the CFO and financial team. One of its key components is EPM Automate, which acts as an agent facilitating data exchange between various sources and EPBCS.

### 4.5.1 EPM Automate

EPM Automate serves as the traffic police, orchestrating data exchange processes within EPBCS. It acts as a server agent responsible for accepting requests, pulling data from sources like databases, and delivering it to EPBCS for reporting and analysis. Essential tasks include installing, configuring, and managing the agent, as well as monitoring logs for troubleshooting purposes.

* **Installation and Configuration**: EPM Automate requires installation and configuration to ensure seamless operation. Once installed, it can be managed through Windows services, allowing simple start and stop commands.
* **Folder Structure and Logs**: The agent's folder structure contains important logs, such as data extraction processes and system activity. Monitoring these logs provides insights into the agent's performance and any potential issues.
* **INI File**: The INI file serves as the configuration file for the agent, containing connection information, encrypted passwords, and other settings required for data exchange. It ensures secure communication and compliance with security protocols.

### 4.5.2 EPBCS Interface

The EPBCS interface provides access to data exchange scripts and pipelines, which are managed by the development team. These scripts orchestrate data extraction from various sources, passing through the EPM Automate agent for processing. Users can monitor process details and track the status of data exchange operations within the EPBCS interface.

* **Data Exchange Pipelines**: EPBCS users can access data exchange pipelines to launch scripts for extracting, transforming, and loading data into EPBCS. These pipelines are managed by designated engineers and monitored for successful execution.
* **Process Monitoring**: The EPBCS interface allows users to monitor the status of data exchange processes, view process details, and track the progress of data extraction tasks. This visibility ensures transparency and facilitates troubleshooting in case of failures.
* **User Access and Permissions**: Access to EPBCS and its functionalities is controlled through user permissions, ensuring that only authorized personnel can view, manage, and execute data exchange processes. This helps maintain data integrity and security.

### Conclusion :

EPBCS, with its EPM Automate agent and intuitive interface, plays a pivotal role in financial reporting and forecasting within the organization. Understanding its components, functionalities, and administration processes is essential for ensuring smooth operations and reliable data analysis.

### 5. Overview of Environments :

### 5.1 Performance Environment :

* + For reliable performance testing, the infrastructure in the performance environment needs to mirror that of the production environment.
  + This includes having similar numbers of web servers and proxy servers.

### Production Environment:

* + Production is divided into two environments: Live and Standby (DR), currently deployed in Phoenix and Ashburn regions.
  + There's a scheduled switch to Ashburn in June or July.

### 5.3 Post-Production Environment (LTE1):

* + LTE1 maintains data synchronization with production, allowing the customer support team to troubleshoot and fix issues reported by clients before pushing fixes to production.

### 5.4 Demo Environment (Demo1):

* + Used for sales demonstrations to potential clients.
  + Despite not having identical infrastructure to production, it's treated with utmost importance due to its impact on sales revenue.
  + Any interruptions or failures in Demo1 are considered critical.

### Content Delivery Network (CDN) and Static Content for Demo1:

* + CDN servers play a crucial role in ensuring the availability of non-production environments, especially Demo1.
  + Instances of Demo1 being unavailable due to CDN server issues underscore the importance of monitoring and maintenance.
  + CDN servers host shared static content for non-production environments (STG1, STG2, Demo1).
  + Hosting static content on CDN servers ensures availability and decouples dependencies, preventing issues like downtime during code catch-up.

### 5.4.2 Future Plans for Demo Environment:

* + There are plans to move demo functionality directly to the production environment, eliminating the need for Demo1.
  + This transition is expected in the near future but is not imminent.

### 5.5 Overview of SL Environments:

* + SL environments, or slim environments, are simplified versions of the regular production environments.
  + Currently, there are three SL environments: SLO 1, SLO 2, and SLO 3.
  + SL environments serve as pure developer playgrounds for various scrum teams, such as benefits, company onboarding, employee onboarding, and terminations teams.
  + These environments are tailored for testing and experimentation, providing a sandbox environment for teams to work in.

### Infrastructure Configuration:

* + Each SL environment includes PeopleSoft setups, with one instance each in SLO 1, SLO 2, and SLO 3.
  + Additionally, there are platform environments, such as con B1 which utilize shared databases across SL environments for cost optimization.

### 5.5.2 Hosting and Technology Stack:

* + SL environments are hosted on Amazon Web Services (AWS), while regular production environments are on Oracle.
  + PeopleSoft instances are built using CloudFormation templates and EC2 instances.
  + Platform environments run on Kubernetes clusters, with each environment segregated by namespace.

### 5.5.3 Maintenance and Patching:

* + PeopleSoft instances have not been rebuilt since November 2022 due to challenges with the build process and the decision to avoid unnecessary rebuilds.
  + Instead of rebuilding instances, the Unix team applies OS patches directly using the patch manager.
  + Platform environments are managed through code snapshots, which are periodically created and deployed. However, this process has encountered difficulties and delays.

### 5.5.4 Future Plans and Challenges:

* + There are discussions about offboarding SL environments from AWS to Oracle Cloud Infrastructure (OCI) for cost savings and better database pricing.
  + Challenges include maintaining and troubleshooting SL environments, especially when developers lack deployment knowledge or encounter issues with the deployment process.
  + The preferred approach for resolving issues is to rebuild the environment from scratch using updated code snapshots and deployment pipelines.

### Rebuilding Process:

* + To rebuild an environment like con B1, developers can select the desired snapshot from the confluence page and deploy it using the appropriate pipeline.
  + Rebuilding involves tearing down existing resources, deploying new ones from the selected snapshot, and then troubleshooting any remaining issues.

### 5.6 Platform Environment Configuration:

* + Platform environments run on Kubernetes clusters within AWS, utilizing namespaces to segregate resources for each environment.

### Maintenance Approach:

* + In case of issues with platform environments, the preferred approach is to rebuild the entire environment from scratch rather than troubleshooting individual components.
  + This saves time and ensures a clean slate for troubleshooting, allowing developers to start fresh and address any underlying issues.

.

### 5.6.2 Maintenance Responsibility:

* + Developers are responsible for basic troubleshooting and maintenance of SL environments, including deploying code updates and configurations.
  + Incidents raised by teams trigger support actions from the infrastructure team to unblock issues and ensure the environments are functional.

### 5.6.3 Future Plans and Challenges:

* + Discussions are underway to potentially offload SL environments from AWS to Oracle Cloud Infrastructure (OCI) for cost savings.
  + Challenges include maintaining consistency across environments, ensuring regular usage, and troubleshooting issues promptly to minimize downtime.

In summary, the discussion elaborates the environments present in the PEO and PeopleSoft Architecture , future plans with those environments , CDN servers and maintenance approaches of those environments.

### 6. HRIS Platform Overview:

* + HRIS (Human Resources Information System) is a platform acquired by TriNet, previously known as Zenefits.
  + It operates separately from the PEO (Professional Employer Organization) platform.
  + The HRIS platform is developed in Python and hosted on Amazon Web Services (AWS).
  + Unlike PEO, HRIS uses MySQL databases running on AWS Aurora clusters.

### 6.1 Authentication and Access:

* + Authentication to the HRIS AWS environment is via Okta, which delegates authentication to TriNet's Active Directory (AD) account.
  + HRIS utilizes AWS-specific services like Aurora, RDS, and ECS (Elastic Container Service), along with general cloud services.

### 6.2 VPC Structure:

* + HRIS operates within multiple VPCs, with separate production and non-production environments.
  + Production VPC hosts load balancers in public subnets and application components in private subnets.
  + Staging VPC replicates production setups on a smaller scale for testing and development purposes.

### Application Components:

* + The HRIS application is built using the Django Python web framework.
  + It consists of various modules like payroll, customer service, and backend processes
  + HRIS uses ECS clusters to host application instances, with separate clusters for frontend and backend services.
  + Redis clusters are employed for caching, and Elasticsearch is utilized for search functionality.

### 6.4 Infrastructure Deployment:

* + The HRIS infrastructure is deployed using AWS CloudFormation, Ansible, and Terraform scripts.
  + Blue-green deployment strategy is implemented for updates, with entire stacks recreated nightly for seamless updates.
  + Infrastructure code is managed in GitHub repositories, with separate accounts for HRIS and DCP (Data Control Platform).

### 6.5 Database Setup:

* + MySQL databases for HRIS are deployed using RDS in production and EC2 instances for non-production environments.
  + Non-production databases are refreshed every two hours to maintain consistency with production data.

### 6.6 External Communication and Security:

* + Outgoing communications are routed through NAT Gateways and Internet Gateways for external connectivity.
  + Internal communications utilize VPNs and internal Application Load Balancers (ALBs) for segregation and security.

### 6.7 Continuous Deployment (CD):

* + HRIS follows a CD pipeline, with nightly stack recreations for updates and maintenance.
  + Code changes are managed and deployed using GitHub repositories and CI/CD tools like Ansible, Terraform, and AWS CloudFormation.

Overall, the HRIS architecture features a robust setup for hosting HR-related applications and services, leveraging AWS services and best practices for scalability, reliability, and security