**2.1 Accountable Roles and High-Level Process Description**

**Accountable Roles:**

1. **Container Platform Team:**
   * Manages Kubernetes cluster deployment, configuration, and ongoing maintenance.
   * Responsible for creating and enforcing baseline configurations through Helm charts and Terraform scripts.
   * Ensures compliance with enterprise security standards and approves container-related configurations.
2. **CPS Team:**
   * Provides network-level resources, including CIDR IP ranges and subnets, for new landing zones.
   * Facilitates the creation of secure environments for Kubernetes clusters.
3. **Application Teams (Asset Teams):**
   * Develop and deploy applications on the pre-configured clusters.
   * Operate within defined RBAC policies and pipeline controls to ensure security and operational compliance.
4. **DevOps Team:**
   * Manages the automation and CI/CD pipelines used for deploying containers and applications.
   * Oversees scanning and validation tools (e.g., Snyk, OPA) integrated within pipelines to detect vulnerabilities or misconfigurations.

**High-Level Description of Process:**

* **Input Activities:**
  + Application teams submit requests for new cluster setups or container deployments.
  + CPS provides necessary network configurations, including CIDR allocations.
* **Execution and Decision Points:**
  + Clusters are provisioned using automated Jenkins pipelines and Terraform scripts.
  + Baseline configurations are enforced using Helm charts and validated by OPA policies during deployment.
* **Systems Involved:**
  + Jenkins pipelines for cluster creation and application deployment.
  + AWS EKS for hosting Kubernetes clusters.
  + Artifactory for managing container images.
  + Security tools like Snyk and SonarQube for vulnerability scanning.
* **Outputs Produced:**
  + Secure Kubernetes clusters with standardized configurations.
  + Fully deployed applications adhering to enterprise security and performance standards.

**Potential Cyber Risk:**

* Unauthorized cluster creation or misconfigured containers could lead to security breaches or operational downtime.
* Lack of monitoring for network policies may result in lateral attacks or data exfiltration.
* Privilege escalation risks through inadequate RBAC configurations could allow unauthorized access or changes.

**2.2 Granular Risks and Key Controls Identified**

**Risk Identified: P1.R1**

**Risk:** Unauthorized cluster deployment outside of approved pipelines.  
**Relevance:** High. Unauthorized clusters can bypass standard security baselines, leading to non-compliance and increased attack surfaces.

**Control Identified: P1.C1**

**Control:** Restrict cluster creation to approved Jenkins pipelines integrated with Terraform scripts.  
**Details:** Cluster creation is limited to the Container Platform Team, with IAM roles managed through Jenkins automation. Pipeline logs are reviewed, and non-compliance triggers immediate remediation actions.

**Risk Identified: P1.R2**

**Risk:** Privilege escalation within Kubernetes clusters.  
**Relevance:** Medium. Application teams with elevated access could make unauthorized modifications, impacting cluster stability or security.

**Control Identified: P1.C2**

**Control:** Enforce Role-Based Access Control (RBAC) policies.  
**Details:** RBAC ensures application teams have only read-only access to Kubernetes resources. Permissions are periodically audited, and violations are flagged for review.

**Risk Identified: P1.R3**

**Risk:** Misconfigured container baseline settings.  
**Relevance:** High. Misconfigurations could introduce vulnerabilities, resource mismanagement, or operational inefficiencies.

**Control Identified: P1.C3**

**Control:** Enforce baseline configurations through Open Policy Agent (OPA) policies.  
**Details:** OPA policies are integrated into pipelines to validate configurations against predefined standards. Deployment failures occur if configurations deviate from baselines.

**Risk Identified: P1.R4**

**Risk:** Unregulated container network traffic.  
**Relevance:** High. Lack of enforcement for network policies may lead to data exfiltration or lateral movement within clusters.

**Control Identified: P1.C4**

**Control:** Define and enforce Kubernetes network policies and AWS security groups.  
**Details:** Network policies restrict pod-to-pod communications and limit external access. AWS security groups enforce inbound and outbound traffic rules for each cluster.

This workpaper documents the key risks and controls identified in the container deployment process, aligning with the audit scope to ensure comprehensive risk management and security compliance.

1. **2.1 Accountable Roles and High-Level Process Description**

**Accountable Roles:**

* 1. **Container Platform Team:**
     + Manages Kubernetes cluster deployment, configuration, and ongoing maintenance.
     + Responsible for creating and enforcing baseline configurations through Helm charts and Terraform scripts.
     + Ensures compliance with enterprise security standards and approves container-related configurations.
  2. **CPS Team:**
     + Provides network-level resources, including CIDR IP ranges and subnets, for new landing zones.
     + Facilitates the creation of secure environments for Kubernetes clusters.
  3. **Application Teams (Asset Teams):**
     + Develop and deploy applications on the pre-configured clusters.
     + Operate within defined RBAC policies and pipeline controls to ensure security and operational compliance.
  4. **DevOps Team:**
     + Manages the automation and CI/CD pipelines used for deploying containers and applications.
     + Oversees scanning and validation tools (e.g., Snyk, OPA) integrated within pipelines to detect vulnerabilities or misconfigurations.

**High-Level Description of Process:**

* 1. **Input Activities:**
     + Application teams submit requests for new cluster setups or container deployments.
     + CPS provides necessary network configurations, including CIDR allocations.
  2. **Execution and Decision Points:**
     + Clusters are provisioned using automated Jenkins pipelines and Terraform scripts. (**Risk Identified: P1.R1**)
     + Baseline configurations are enforced using Helm charts and validated by OPA policies during deployment. (**Risk Identified: P1.R3**)
  3. **Systems Involved:**
     + Jenkins pipelines for cluster creation and application deployment.
     + AWS EKS for hosting Kubernetes clusters.
     + Artifactory for managing container images. (**Risk Identified: P1.R3**)
     + Security tools like Snyk and SonarQube for vulnerability scanning.
  4. **Outputs Produced:**
     + Secure Kubernetes clusters with standardized configurations.
     + Fully deployed applications adhering to enterprise security and performance standards.

**Potential Cyber Risk:**

* 1. Unauthorized cluster creation or misconfigured containers could lead to security breaches or operational downtime.
  2. Lack of monitoring for network policies may result in lateral attacks or data exfiltration. (**Risk Identified: P1.R4**)
  3. Privilege escalation risks through inadequate RBAC configurations could allow unauthorized access or changes. (**Risk Identified: P1.R2**)

1. **You said:**

now mention the controls in paragraph wording format 5w2h

1. **ChatGPT said:**

**Controls in Paragraph Format (5W2H)**

**Control Identified: P1.C1 – Restrict Unauthorized Cluster Creation**

To address the risk of unauthorized cluster creation (**P1.R1**), the organization restricts cluster provisioning to authorized pipelines managed by the Container Platform Team. These pipelines, integrated with Jenkins and Terraform, enforce baseline configurations and standardize deployments. Only IAM roles explicitly provisioned through CyberArk grant permissions for cluster creation. Automated scripts validate IAM policies and ensure access is restricted to pre-approved roles. Periodic pipeline and IAM audits are conducted to identify and rectify deviations. This ensures all clusters are compliant with enterprise standards and minimizes security vulnerabilities.

* 1. **What:** Restrict cluster creation to authorized Jenkins pipelines.
  2. **Who:** Managed by the Container Platform Team with IAM controls reviewed by the DevOps Team.
  3. **Where:** Across all AWS EKS environments.
  4. **When:** During cluster provisioning and quarterly audits.
  5. **Why:** To prevent configuration drift and ensure enterprise-wide compliance.
  6. **How:** Through IAM role validations, automated audits, and pipeline access reviews.
  7. **How Much:** Continuous monitoring and quarterly role audits ensure compliance.

**Control Identified: P1.C2 – Enforce Role-Based Access Control (RBAC)**

To mitigate the risk of privilege escalation in Kubernetes clusters (**P1.R2**), RBAC policies restrict application teams to read-only access. Permissions are defined at the Kubernetes namespace level, limiting actions that teams can perform on resources. IAM roles and policies are reviewed monthly to detect and address access violations. Automation scripts prevent unauthorized changes to RBAC configurations, and access logs are continuously monitored to flag anomalies.

* 1. **What:** Implement and enforce RBAC policies to restrict permissions.
  2. **Who:** Managed by the Container Platform Team with audits by the DevOps Team.
  3. **Where:** Across all Kubernetes clusters and namespaces.
  4. **When:** At cluster provisioning and during monthly access reviews.
  5. **Why:** To prevent unauthorized modifications and ensure operational stability.
  6. **How:** Use of IAM roles, Kubernetes role definitions, and automated access monitoring.
  7. **How Much:** Monthly audits and real-time alerts for violations ensure robust control.

**Control Identified: P1.C3 – Enforce Baseline Configurations with OPA**

To prevent misconfigurations in container deployments (**P1.R3**), Open Policy Agent (OPA) policies enforce predefined baseline settings. These policies restrict runtime parameters such as privileged containers, root access, and memory limits. Helm charts used in deployments are standardized, allowing only limited, approved modifications by application teams. The deployment pipelines validate all configurations against the OPA policies, automatically failing non-compliant deployments. Regular policy reviews and updates ensure that evolving security standards are met.

* 1. **What:** Use OPA policies to enforce configuration baselines during deployment.
  2. **Who:** Managed by the Container Platform Team with validation in Jenkins pipelines.
  3. **Where:** Across all namespaces and clusters during application deployments.
  4. **When:** During deployment and biannual policy reviews.
  5. **Why:** To maintain compliance with enterprise security and operational baselines.
  6. **How:** Integration of OPA policies into pipelines; deployment failures on non-compliance.
  7. **How Much:** Automated compliance checks with biannual updates to OPA rules.

**Control Identified: P1.C4 – Enforce Container Network Policies**

To address the risk of unregulated container network traffic (**P1.R4**), Kubernetes network policies and AWS security groups are used to restrict communications between pods, namespaces, and external entities. Network policies define allowed traffic paths, while AWS security groups enforce inbound and outbound rules. These configurations are embedded within Terraform scripts and managed through Jenkins pipelines. Regular audits ensure that network policies align with enterprise guidelines, and unauthorized changes trigger alerts for review and remediation.

* 1. **What:** Define and enforce Kubernetes network policies and AWS security groups.
  2. **Who:** Managed by the Container Platform Team with oversight by the CPS Team.
  3. **Where:** Across all EKS clusters and associated namespaces.
  4. **When:** At deployment and during annual network audits.
  5. **Why:** To prevent data exfiltration and unauthorized lateral traffic.
  6. **How:** Use Terraform for policy implementation; audit logs for deviations.
  7. **How Much:** Continuous monitoring with annual reviews of network rules and configurations.

These controls ensure that identified risks are effectively mitigated, aligning with enterprise security and compliance objectives.

Top of Form

Bottom of Form

**Audit Document for Container Deployment Process**

**2.1 Document Accountability, Roles, and Process Overview**

**Accountable Parties:**

* **NEF DevOps Team:** Manages container lifecycle, including base image creation, deployment pipelines, and cluster management for EKS and AKS platforms.
* **Application Security Team (AppSec):** Oversees vulnerability scanning tools (e.g., Sneak, SonarQube) and ensures compliance with security standards during CI/CD processes.
* **CPS Team:** Provides hardened AMIs used for nodes in EKS/AKS clusters and ensures infrastructure readiness.
* **Cluster Management Team (KO Team):** Manages cluster setup and configuration, ensuring that the infrastructure is up-to-date and secure.

**High-Level Process Description:**

1. **Input Activities:**
   * Base Docker images are created using Artifactory or CPS-provided hardened AMIs.
   * Application teams provide their libraries and configurations to be added to base images.
   * Security configurations, such as mandatory tools (e.g., Sneak), are defined and integrated into pipelines.
2. **Execution:**
   * Images are built and scanned for vulnerabilities during the CI pipeline process.
   * Deployment pipelines, managed via Harness, control the deployment of these containerized workloads.
   * Helm charts and YAML configuration files dictate cluster and container configurations.
   * Clusters, nodes, and pods are set up and managed by the NEF DevOps and KO teams.
3. **Output:**
   * Secure and containerized workloads are deployed across standardized clusters.
   * Vulnerabilities are addressed before production deployment.
   * Logs and configurations are monitored to ensure operational efficiency and security.

**2.2 Granular Risks and Key Controls Identified**

**Risk Identified: P1.R1**

* **Risk:** Base Docker images lack adequate security hardening aligned with industry standards like CIS benchmarks.
* **Relevance:** Using insecure base images can introduce vulnerabilities into the deployment pipeline and compromise workloads.

**Control Identified: P1.C1**

* **Control:** Mandate the use of base images pre-hardened to CIS or NIST standards. Integrate automated checks within the CI/CD pipeline to validate compliance during the build phase.

**Risk Identified: P1.R2 (access one)**

* **Risk:** Tenants have limited access to modify critical configurations, which could lead to non-compliance with best practices or inefficiencies.
* **Relevance:** Without a mechanism for tenant-specific adjustments, containerized applications may not fully comply with operational or security standards.

**Control Identified: P1.C2**

* **Control:** Provide restricted yet sufficient access for tenants to adjust configurations within predefined parameters. Validate all configuration changes through approval pipelines managed by the NEF DevOps team.

**Risk Identified: P1.R3**

* **Risk:** Excessive load on pipelines due to frequent image pulls from Artifactory, leading to deployment delays.
* **Relevance:** Delays in pipeline execution could affect production timelines and reliability.

**Control Identified: P1.C3**

* **Control:** Implement caching mechanisms for commonly used base images in Artifactory to reduce load and improve pipeline efficiency.

**Risk Identified: P1.R4**

* **Risk:** Lack of granular visibility into logging and monitoring at the cluster and container levels.
* **Relevance:** Insufficient logging could hinder the identification of security breaches or operational issues.

**Control Identified: P1.C4**

* **Control:** Enhance logging mechanisms to include detailed container-level events and provide clear documentation on log types. Ensure logs are monitored via Splunk and integrated with SoC dashboards.