

Azure Enterprise-Scale / PCF V2

Low Level Design Document

# Contents

## Table of Contents

[1. Contents 2](#_Toc134474733)

[1.1. Table of Contents 2](#_Toc134474734)

[2. Document Control 3](#_Toc134474735)

[2.1. Document Information 3](#_Toc134474736)

[2.2. Distribution List 4](#_Toc134474737)

[2.3. Supporting Documents 4](#_Toc134474738)

[3. Exec Summary 4](#_Toc134474739)

[3.1. Overview 4](#_Toc134474740)

[3.2. Purpose 4](#_Toc134474741)

[3.3. Audience 4](#_Toc134474742)

[4. In Scope – Out scope of HaCT & Application Team 5](#_Toc134474743)

[4.1. RACI Matrix for Landing Zone subscriptions 5](#_Toc134474744)

[4.2. AD Group creation & Role Assignment 5](#_Toc134474745)

[4.3. Access and Approval for App Team members 5](#_Toc134474746)

[4.4. Creation/Deletion new AD Group for cluster and Namespace 6](#_Toc134474747)

[4.5. Role Assignment 6](#_Toc134474748)

[5. Default Roles and responsibilities - managed containers – modify and add a scope row 7](#_Toc134474749)

[5.1. Scope which clarifies the roles and responsibilities for cluster and namespace . 7](#_Toc134474750)

[6. Role Based Access Control in Kubernetes 8](#_Toc134474751)

[7. AS-IS Architecture 8](#_Toc134474752)

[7.1. Azure Enterprise Scale@Uniper Architecture 8](#_Toc134474753)

[7.2. Azure Enterprise Scale@Uniper – Application Team RBAC Architecture 8](#_Toc134474754)

[8. Application Team – AD Group Naming Convention 9](#_Toc134474755)

[8.1. How are Security AD Groups created? 9](#_Toc134474756)

[Naming Convention of Portal access 9](#_Toc134474757)

[8.2. Naming Convention of namespace access 9](#_Toc134474758)

[9. Default Authorization – Application Team 9](#_Toc134474759)

[9.1. Azure AD Group Ownership and Membership 9](#_Toc134474760)

[10. Application Team - Role Based Access Control 10](#_Toc134474761)

[10.1. Application Support Engineers Default Access 10](#_Toc134474762)

[10.1.1. Lower Environment – DEV/SANDBOX 10](#_Toc134474763)

[10.1.2. Upper Environment – UAT/PROD 10](#_Toc134474764)

[11. Naming Convention of cluster 11](#_Toc134474765)

[11.1. Cluster Naming Convention 11](#_Toc134474766)

[11.2. AD group naming convention 11](#_Toc134474767)

[11.3. Vnet and Subnet naming convention 11](#_Toc134474768)

[12. Cluster Creation in ES@uniper 12](#_Toc134474769)

[12.1. Sandbox and DEV environment 12](#_Toc134474770)

[12.2. UAT and PROD environment 12](#_Toc134474771)

[13. RBAC within the cluster 12](#_Toc134474772)

[13.1. Cluster level 12](#_Toc134474773)

[14. Policies 12](#_Toc134474774)

[15. Log Analytics workspace 13](#_Toc134474775)

[16. Upgrade/Patch of cluster on regular basis 13](#_Toc134474776)

[17. Monitoring of cluster 13](#_Toc134474777)

[18. Recommendations 14](#_Toc134474778)

[19. Appendix 14](#_Toc134474779)

[A. Policy Description 14](#_Toc134474780)

[A.1. Kubernetes cluster containers should only use allowed images 15](#_Toc134474781)

[A.2. Kubernetes cluster containers should run with a read only root file system 15](#_Toc134474782)

[A.3. Kubernetes clusters should be accessible only over https 15](#_Toc134474783)

[A.4. Kubernetes clusters should not use the default namespace 15](#_Toc134474784)

[A.5. Kubernetes clusters should use internal load balancers 15](#_Toc134474785)

[A.6. Kubernetes clusters should not allow container privilege escalation 15](#_Toc134474786)

[A.7. AKS clusters should have defender profile enabled 15](#_Toc134474787)

[A.8. Resource logs in Azure Kubernetes Service should be enabled 15](#_Toc134474788)

[B. Glossary 15](#_Toc134474789)

[C. Create Automated Namepace + role and role binding yaml 15](#_Toc134474790)

[D. Supporting Links 18](#_Toc134474791)

# Document Control

## Document Information

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Version | Name | Role | Comments |
| 02/05/2023 | 0.1 | Shobana |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Table 1: Document Information

## Distribution List

|  |  |  |
| --- | --- | --- |
| Distributed to | Role | Company |
|  | Cloud Security Architect | UNIPER |
|  |  |  |
|  |  |  |

Table 2: Distribution List

## Supporting Documents

|  |  |
| --- | --- |
| **Document Name** | **Version** |
| Azure Enterprise-Scale / PCF V2  High Level Design Document | Version 1.0 |

Table 3: Supporting Documents

Reference:

<https://uniper.sharepoint.com.mcas.ms/sites/CloudWorksTeam/HaCT/Forms/AllItems.aspx?isAscending=false&id=%2Fsites%2FCloudWorksTeam%2FHaCT%2FProject%20Documentation%2FPCFv2%2FPCFv2%20LLD%2FReview%20%2D%20Completed&sortField=Modified&viewid=3305bd1d%2Da258%2D4b01%2Dacde%2Dc9a0cf7707bd>

# Exec Summary

## Overview

Uniper has Azure AD at present, which is synced with On-Premises active directory. RBAC is an authorisation system built on Azure Resource Manager that provides fine-grained access management of Azure resources. Using RBAC, you can restrict access based on the need to know and least privilege security principles. Access management for cloud resources is a critical function for UNIPER when using cloud services.

Role-based access control provides each worker privileges based on what role they have in the organization.

## Purpose

This LLD documents covers RBAC requirements, detailed RBAC structure, RBAC configurations and maintenance of these RBAC infrastructure with respect to AKS which will be deployed in Enterprise Scale@Uniper, its management groups and landing zone subscriptions.

## Audience

The intended audience for this document will be UNIPER architects and UNIPER Enterprise Scale@Uniper project management.

# In Scope – Out scope of HaCT & Application Team

## RACI Matrix for Landing Zone subscriptions

|  |  |  |
| --- | --- | --- |
| **R = Responsibilities A = Accountable C = Consulted I = Informed** | **Application Team** | **HaCT Cloud Engineer** |
| Requesting for new AD group creation other than default | R, A |  |
| Create of role assignment other than default | R, A | C, I |
| Deletion of role assignment other than default | R, A | C, I |
| Access for Application member | R, A |  |
| Process of Approval flow | R, A |  |
| Custom role binding | R,A,C | I |
| Excluding/Exemption of Policy | C, I | R, A |

Table 4: Responsibility assignment matrix

## AD Group creation & Role Assignment

|  |
| --- |
| **HaCT Responsibilities** |
| 1. HaCT Team will be creating AD Group and performing default role assignments to Application Team on their ordered subscription.(only suggestions by AKS team and changes as per over time )  If in case application team needs any AD group creation they can reach AD team for the group creation . |
|  |
|  |

Table 5: In/Out Scope - AD Group creation and Role Assignment

## Access and Approval for App Team members

|  |  |
| --- | --- |
| **Responsibilities** | |
| **Application Team** | **HaCT Team** |
| Application Manager is responsible to grant access to Application Team member on the required subscription  Grant/Revoke access to Application team must be taken care by Application Manager.  Application Managers must assess permitted users and give application team members access. | HaCT Team will be creating the default AD Groups with access on the subscriptions ([section 7.2.](#_Naming_Convention)), assign App Manager as Owner of AD Groups of Contributor and Reader ad group & Member of Reader AD Group ([section 8.2.](#_Azure_AD_Group)). Will share the details to App Manager. |

Table 6: In/Out Scope - Access for App Team members

## Creation/Deletion new AD Group for cluster and Namespace

|  |  |
| --- | --- |
| **Responsibilities** | |
| **Application Team** | **HaCT Team** |
| Application Team needs to contact the UNIPER Directory Service team. Catalog to place request using SNOW link - [Directory Service - Manage Active Directory groups](https://uniperprod.service-now.com.mcas.ms/unipersp?id=sc_cat_item_uni&sys_id=fc3fe7ab8742d5d0fc79c9d30cbb35f5) | HaCT is not responsible for creating the AD Group for Enterprise Scale@Uniper except for the default AD Groups. |
| AD Group should be Security type with proper description. | HaCT team will be removing the role assignment of other AD Group type role assignment except Security type ad groups |
| Addition/Removal of Members into the AD Group should be taken care by Application Team or via UNIPER Directory Service team. |  |

Table 7: In/Out Scope - Creation/Deletion new AD Group

## Role Assignment

|  |  |
| --- | --- |
| **Responsibilities** | |
| **Application Team** | **HaCT Team** |
| App Team's responsible to perform the role assignment for themselves on-demand | HaCT Team will be working on to identify other critical roles. If HaCT identifies Critical role, it will get appended to the list. |
| Application team are requested to use the Least privilege principle and perform the role assignment. Recommendation from HaCT is to check resource specific role and assign what is required to perform the activity.  We request that Application Team in roles that have critical/high privilege access, such as "Owner," "User Access Administrator," and "Resource Policy Contributor," refrain from using them across ESLZ subscriptions, resource groups, and resources. | If in case mentioned role assignments are identified during audit process, HaCT Team will removing immediately |
|  | On noticing role assignments apart from Reader for Application team members in PROD subscription, HaCT Team will be removing the respective role assignment. |

# Default Roles and responsibilities - managed containers – modify and add a scope row

## Scope which clarifies the roles and responsibilities for cluster and namespace .

| **Role name** | **Description** | Responsibilities |
| --- | --- | --- |
| Cluster administrator | The cluster administrator role is assigned to the Cloud Management Center.  It creates and manages the AKS clusters. | * Fulfillment of container requests * Monitoring of cluster health * Upgrades and Patching (cluster, node pools, ingress controller, Kured daemon, Kubecost dashboard) * Resource management on cluster level * Namespace creation and namespace administrator assignment * Azure Security Policies and exceptions * Node pool creation and management * Management of system namespaces of cluster |
| Namespace administrator | The namespace administrator role is assigned to an application team.  It manages a namespace inside an AKS cluster. | * Monitor namespace and applications inside the namespace * Upgrades and patching of containers inside the namespace * Resource management on namespace level * Deployment of applications into cluster * Namespace Network policies * Testing cluster upgrades * Define and manage additional roles inside a namespace |
| Developer | The namespace administrator can define additional roles inside a cluster.  The developer role is an example of a role that could be defined. | * These depend completely on the role definition by the namespace administrator but cannot exceed it |

Customized RBAC witinin cluster :

Application can create their own customized RBAC

Pre-requisite – Namespace Admin should create and assign permission to the team members

Table 8: In/Out Scope - Role Assignment

# Role Based Access Control in Kubernetes

Role-based access control provides each application team member/HaCT Cloud Engineer privileges based on what role they have in the organization.

Hact Managed containers has been integrated with Azure Active Directory (AD) which helps customers to manage access to kubernetes resources based on existing identity and group membership. Your existing Azure AD users and groups can be provided access to AKS resources and with an integrated sign-on experience.

Kubernetes RBAC provides granular filtering of user actions. With this control mechanism:

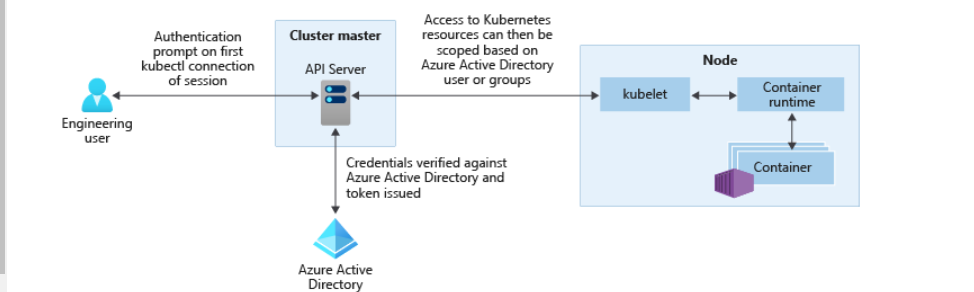
Assign users or user groups permission to create and modify resources or view logs from running application workloads.

Limit scope permissions to a single namespace or across the entire AKS cluster.

Create roles to define permissions, and then assign those roles to users with role bindings.

# AS-IS Architecture

## Azure Enterprise Scale@Uniper Architecture

  
Figure 1: ES@Uniper Architecture

Ref: <https://learn.microsoft.com/en-us/azure/aks/concepts-identity>

Figure 2: ES@Uniper - HaCT Cloud Engineer Access

## Azure Enterprise Scale@Uniper – Application Team RBAC Architecture

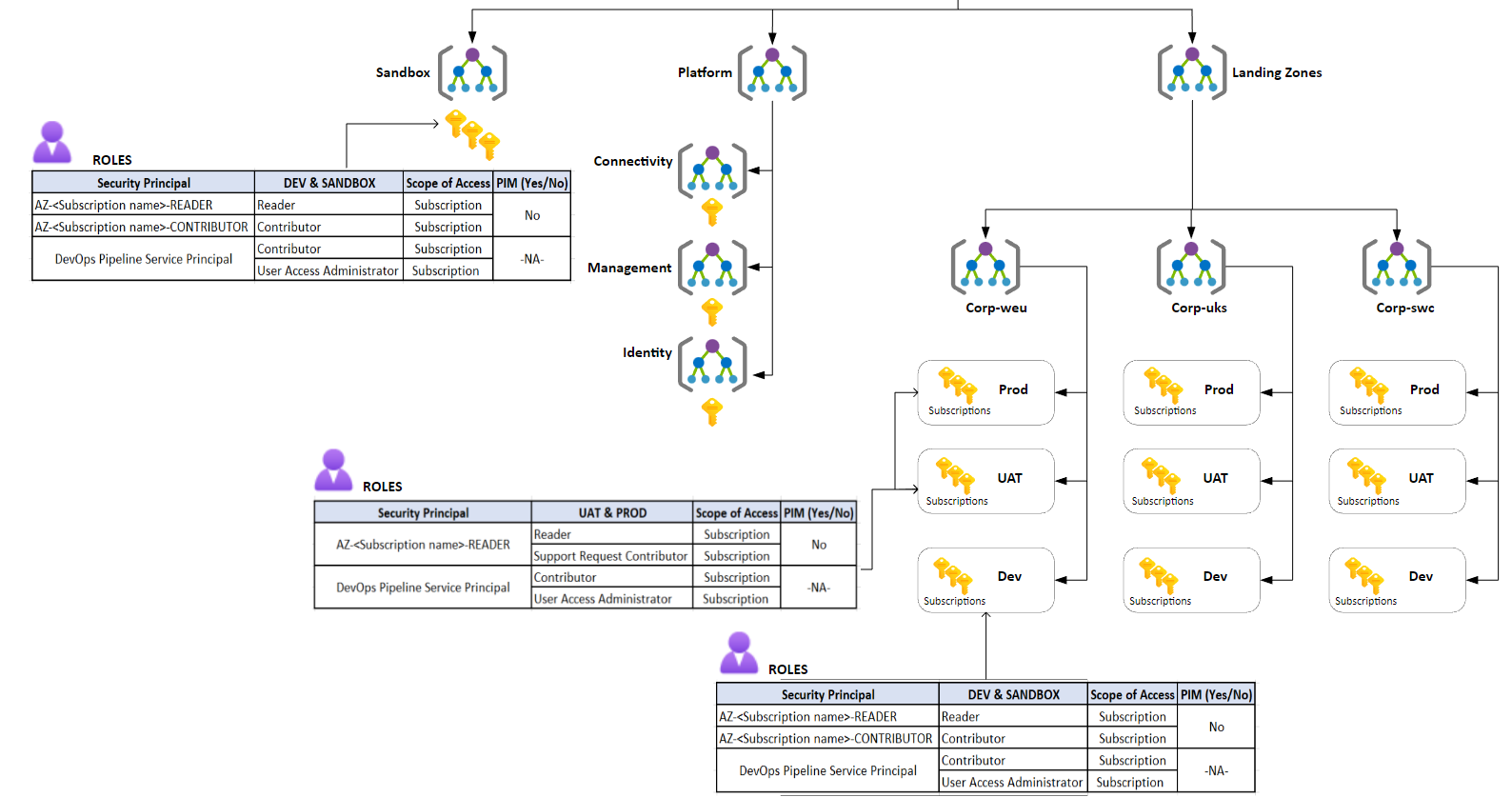


Figure 3: ES@Uniper - Application Team Access

Remark: Detailed explanation regarding the default role is provided under the [section 9](#_Application_Team_-).

# Application Team – AD Group Naming Convention

## How are Security AD Groups created?

## Naming Convention of Portal access

Application Owners and Team Members order subscriptions by submitting requests to create subscriptions using catalogue services.

During the process of deployment of Subscriptions via IaC \*, respective AD Groups are created with the standard naming convention pattern and Application Managers are assigned as Owners/Member of AD Groups.

## Naming Convention of namespace access

PCFv2 security AD group convention pattern is used.

AZ- PCFv2-CORP-DEV-C\_MA3-DTFU081-01- READER

|  |  |  |
| --- | --- | --- |
| Azure | Subscription Name | Role Name |

* AZ-<Subscription name>-READER
* AZ-<Subscription name>-CONTRIBUTOR

Example : AZ-F\_OI3-B4-HaCT-AKS-PAAS-PREPROD-Reader

# Default Authorization – Application Team

## Azure AD Group Ownership and Membership

During the deployment of subscription and its respective AD groups, Application Manager will be configured as Owner of Reader and Contributor ad groups.

|  |  |  |
| --- | --- | --- |
|  | **Reader - AD Group** | **Contributor - AD Group** |
| ***Owner*** | App Manager | App Manager |
| ***Member*** | App Manager, \* | \* |

Table 11: AAD Group Member and Owner

**\***Application Manager can add his application team members into the AD Groups depending on the requirement.

# Application Team - Role Based Access Control

AAD groups and SPNs created during subscription deployment using IaC supported by the HaCT Automation Team which will enable application team members to access the application. The table below provides information about the role and scope of each AAD group and SPN created.

## Application Support Engineers Default Access

### Lower Environment – DEV/SANDBOX

|  |  |  |  |
| --- | --- | --- | --- |
| **Security Principal** | **DEV & SANDBOX** | **Scope of Access** | **PIM (Yes/No)** |
| AZ-<Subscription name>-READER | Reader | Subscription | No |
| AZ-<Subscription name>-CONTRIBUTOR | Contributor | Subscription |
| DevOps Pipeline Service Principal | Contributor | Subscription | -NA- |
| User Access Administrator | Subscription |

Table 12: Application Team Access - Lower Environment

**Business Justification/Reason of above role assignments:**

* Contributor access granted to application users and service principals to deployment of resources.
* UAM access is assigned to Application SP to create/delete the role assignments to Application team members and SPs.
* PIM is not implemented for Application team members in Lower environment

### Upper Environment – UAT/PROD

|  |  |  |  |
| --- | --- | --- | --- |
| **Security Principal** | **UAT & PROD** | **Scope of Access** | **PIM (Yes/No)** |
| AZ-<Subscription name>-READER | Reader | Subscription | No |
| Support Request Contributor | Subscription |
| DevOps Pipeline Service Principal | Contributor | Subscription | -NA- |
| User Access Administrator | Subscription |

# Naming Convention of cluster

## Cluster Naming Convention

{type of resource}-{app name}-{env}-{region}-{instance}

Aks- dataasset- dev- weu- 01-

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| type of resource | app name | env | region | instance |

Ex: AKS-dataasset-dev-weu-01

## AD group naming convention

CLUSTER AD Group – Azure Portal Access

AZ-      AKS-coode-dev-weu-01         76987-                                           DEV-                  AKSClusterUserRole

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Azure | Cluster Name | EAMID | Environment | Role Name |

AD Group – Namespace access

AZ-              data-asset-inventory-             76987-                                           DEV-             NameSpaceAdmin

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Azure | Namespace Name | EAMID | Environment | Role Name |

## Vnet and Subnet naming convention

|  |  |  |
| --- | --- | --- |
| Asset Type | Scope | Format and examples |
| **Virtual network** | Resource group | *<resource>-<location>-<environment>-<purpose>-<int>* |
|  |
| vnet-weu-prd-core-01 |
| vnet-weu-dev-coode-001 |
| vnet-uks-int-fuse-001 |
| **Subnet** | Virtual network | *<purpose>-<env>-<app name>-<resource>-<int>* |
| high-npd-fuse-aks-v2-snet-003 |
| high-dev-pkfg-snet-aks-001 |
| high-dev-pkfg-snet-aks-001 |
| **Load balancer** | Resource group | *<resource>-<location>-<environment>-<purpose>-<int>* |
| lb-weu-prd-core-01 |
| lb-uks-int-fuse-001 |

# Cluster Creation in ES@uniper

TBU

## Sandbox and DEV environment -TBU

## UAT and PROD environment-TBU

# RBAC within the cluster

Application Team’s DevOps pipeline Service Principal have complete privileges to assign them with the access on the scope of cluster and its respective namespace.

## Cluster level

As per HaCT AKS cluster standard,

* ***In DEV and Sandbox environment,*** Application team are advised with assigning cluster-reader” custom role for application team as initial step. Later based on their least privilege model, application team can create separate Security Type AD Group for each type of role and role binding and assign with the Application team with the access based on their work.
* ***In UAT and PROD environment ,*** All the deployment, Monitoring, configuration are performed via Configured Azure Devops pipeline. Application team are not advised to use their Human KID to access the cluster. (unless there is an emergency).

Note: Automated namespace creation with role and role binding yaml file is explained in the section Appendix C.

# Policiess

To improve the security of your Azure Kubernetes Service (AKS) cluster, Policies are applied and enforce built-in security policies on your cluster using Azure Policy. Azure Policy helps to enforce organizational standards and to assess compliance at-scale.

Below is the list of policies which are applied on clusters in ES@UNIPER.

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Policy Name** | **Effect** | **Scope** |
| 1 | Kubernetes cluster containers should only use allowed images | Deny | RG |
| 2 | Kubernetes cluster containers should run with a read only root file system | Deny |  |
| 3 | Kubernetes clusters should be accessible only over https | Deny |  |
| 4 | Kubernetes clusters should not use the default namespace | Deny |  |
| 5 | Kubernetes clusters should use internal load balancers | Deny |  |
| 6 | Kubernetes clusters should not allow container privilege escalation | Deny |  |
| 7 | AKS clusters should have defender profile enabled | Audit |  |
| 8 | Resource logs in Azure Kubernetes Service should be enabled | AuditIfNotExists |  |
| 9 | Kubernetes clusters should disable automounting API credentials | Audit |  |
| 10 | Running containers as root user should be avoided | Audit |  |
| 11 | Usage of pod HostPath volume mounts should be restricted to a known list to restrict node access from compromised containers | Audit |  |
| 12 | Usage of host networking and ports should be restricted | Audit |  |

Detailed description of each policies are explained under the section Appendix A.

# Log Analytics workspace

Application team can use decentralised log analytics workspace instead of creating a new dedicated log analytics workspace (LAW) for each cluster, application teams can leverage the decentralised dedicated LAWs that are already available in their respective subscriptions. These LAWs are provisioned during the deployment of each subscription and can be used for log collection and analysis of cluster and its respective namesapce. (dedicated LAW details )

Example :  
When Application Team UniIIoT needs to set up a Cluster for their project, they can map the dedicated de-centralizied LAW from the RG "rg-weu-dev-mlogspaasc3d912-01" instead creating a new LAW in the same RG where they are provisioning the cluster.  
  Subscription : ESLZ-CORP-DEV-F\_OI4-UniIIoT-01  
  Resource Group : rg-weu-dev-mlogspaasc3d912-01  
  Log Analytics Workspace Name : law-weu-corp-dev-f-oi4-uniiiot-01

# Upgrade/Patch of cluster on regular basis

Minor Upgrade is performed on quarterly basis followed by poc, pre and prd cluster .

Patch update is performed in an automated manner and we recommend the same in new clusters .

* As per this Wiki page [AKS Upgrade Communication Procedure](https://wiki.intranet.uniper.energy/sales/display/CF/AKS+Upgrade+Communication+Procedure) need to do upgrade AKS.
* Upgrade needs to be performed sequentially. It needs to be 1.19.x to 1.20.x to 1.21.x
* Every 3 months we need to upgrade AKS Cluster.
* Need to raise ticket with Microsoft team request for dedicated person during our upgradation process.

# Monitoring of cluster

At present we have Kubecost , Prometheus installed in our cmc managed clusters and application team can make use of the same to implement the features .

Kubecost Installation reference to WIKI as below

[A Guide to Install Kubecost - Hosting Framework - Uniper Wiki](https://wiki.intranet.uniper.energy/sales/display/CF/A+Guide+to+Install+Kubecost)

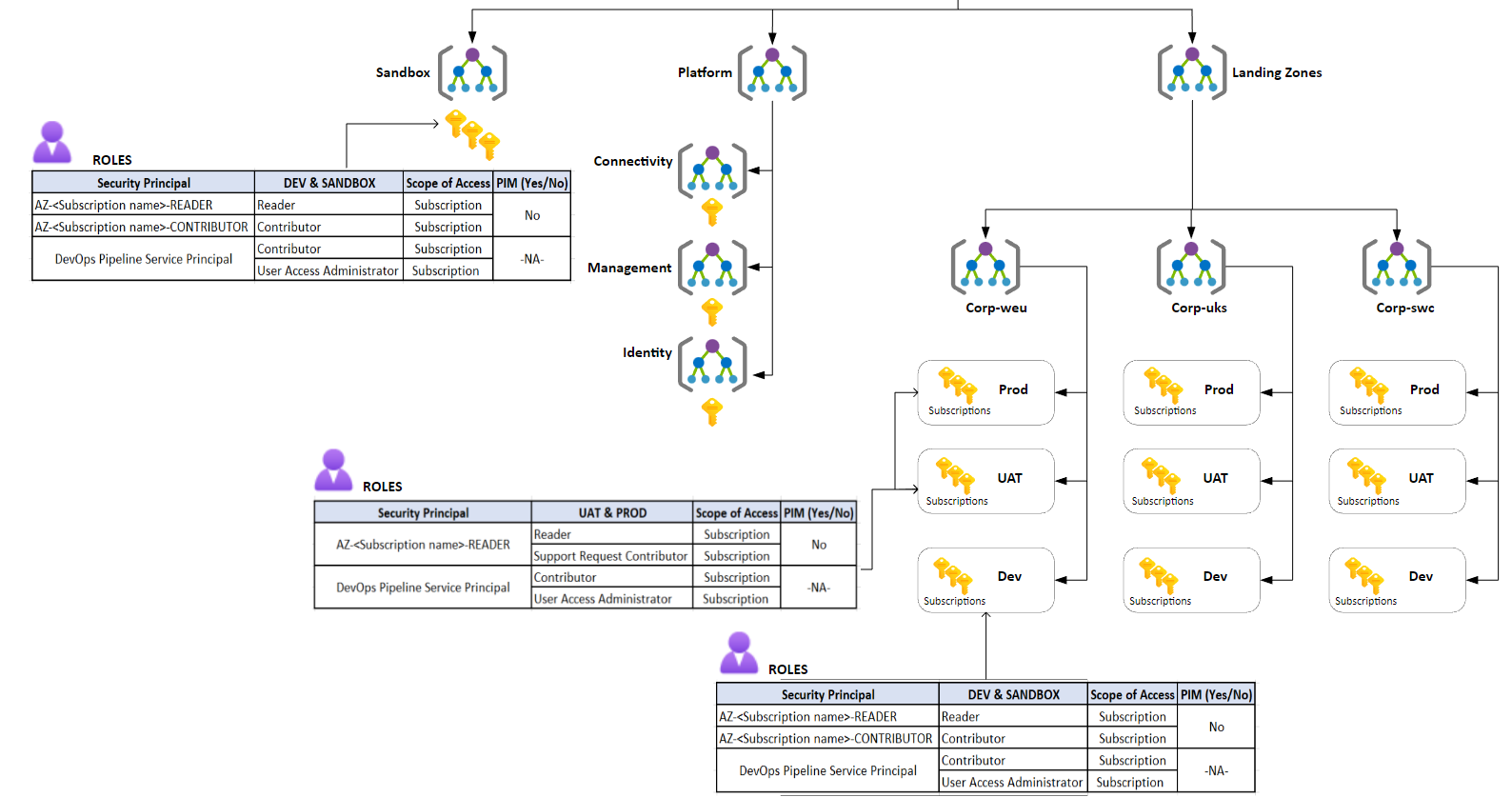
Prometheus Installation reference to WIKI as below

[A Guide to use Prometheus - Hosting Framework - Uniper Wiki](https://wiki.intranet.uniper.energy/sales/display/CF/A+Guide+to+use+Prometheus)

# Recommendations

Application team should maintain separate RG for cluster creation .

Cluster Admin access should be limited to team members .



Header name - Azure Enterprise Scale@Uniper – Application Team RBAC Architecture

# Appendix

## Policy Description

### A.1. Kubernetes cluster containers should only use allowed images

Description

Use images from trusted registries to reduce the Kubernetes cluster's exposure risk to unknown vulnerabilities, security issues and malicious images.

       Mode: Audit, Deny, Disable

### A.2. Kubernetes cluster containers should run with a read only root file system

Description

  Run containers with a read only root file system to protect from changes at run-time with malicious binaries being added to PATH in a Kubernetes cluster.

           Mode: Audit, Deny, Disable

### A.3. Kubernetes clusters should be accessible only over https

Description

Use of HTTPS ensures authentication and protects data in transit from network layer eavesdropping attacks.

           Mode: Audit, Deny, Disable

### A.4. Kubernetes clusters should not use the default namespace

Description

Prevent usage of the default namespace in Kubernetes clusters to protect against unauthorized access for ConfigMap, Pod, Secret, Service, and Service Account resource types.

           Mode: Audit, Deny, Disable

### A.5. Kubernetes clusters should use internal load balancers

Description

Use internal load balancers to make a Kubernetes service accessible only to applications running in the same virtual network as the Kubernetes cluster

Mode: Audit, Deny, Disabled

### A.6. Kubernetes clusters should not allow container privilege escalation

Description

Do not allow containers to run with privilege escalation to root in a Kubernetes cluster. This recommendation is part of CIS 5.2.5 which is intended to improve the security of your Kubernetes environments. This policy is generally available for Kubernetes Service (AKS), and preview for Azure Arc enabled Kubernetes

Mode: Audit, Deny, Disable

### A.7. AKS clusters should have defender profile enabled

Description

Microsoft Defender for Containers provides cloud-native Kubernetes security capabilities including environment hardening, workload protection, and run-time protection. When you enable the SecurityProfile.AzureDefender on your Azure Kubernetes Service cluster, an agent is deployed to your cluster to collect security event data.

Mode: Audit, Deny, Disabled

### A.8. Resource logs in Azure Kubernetes Service should be enabled

Description

Azure Kubernetes Service's resource logs can help recreate activity trails when investigating security incidents. Enable it to make sure the logs will exist when needed

Mode: AuditIfNotExists, Disabled

## Glossary

## Create Automated Namepace + role and role binding yaml

|  |
| --- |
| # Commit all YAMLS required for creating namespaces, roles, rolebindings 'ADA\_WindFC' in this single file.  # Namespace Naming Convention(lowercase Only) : applicationname + iterationplanid + env + servicecomponent i.e. adawindfc-163752-dev  # Remove the namespace prefix in the roles and role bindings  ## Namespace Admin  # Use this for 'namespace admin' user access  # This below lines of code creates Namespace, Role with Namespace Admin privileges & Role Bindings in a namespace.  # Note: Deployments and Services is ADA\_WindFC developers responsibilty  ---  kind: Namespace  apiVersion: v1  metadata:  labels:  environment: #{environment}#  iteraplanid: '#{iteraplanID}#'  application\_name: #{appname}#  owner\_kid: #{ownerKID}#  contact\_kid: #{ownerKID}#    name: #{appname}#-#{iteraplanID}#-#{environment}#  ---  # Creates a ResourceQuota 'requests-resourcequota' Specific to project namespace  ---  apiVersion: v1  kind: ResourceQuota  metadata:  name: requests-resourcequota  namespace: #{appname}#-#{iteraplanID}#-#{environment}#  spec:  hard:  requests.cpu: #{request-cpu}# # metric - 'm' millicore, containers can get a single req with 500m or 50 requests with 10m  requests.memory: #{request-memory}# # metrics - 'Mib' or 'Gib' total requested memory by container < 1GB  limits.cpu: #{limits-cpu}# # Maximum CPU which can be claimed by combined or single container  limits.memory: #{limits-memory}# # Maximum Memory which can be claimed by combined or single container  ---  # Creates a Role 'namespace-admin' Specific to project namespace  kind: Role  apiVersion: rbac.authorization.k8s.io/v1  metadata:  name: namespace-admin  namespace: #{appname}#-#{iteraplanID}#-#{environment}#  rules:  - apiGroups: ["extensions", "apps", "networking.k8s.io","metrics.k8s.io", "rbac.authorization.k8s.io", "autoscaling"]  resources: ["\*"]  verbs: ["\*"]  - apiGroups: [""]  resources:  - bindings  - configmaps  - endpoints  - events  - limitranges  - persistentvolumeclaims  - persistentvolumes  - pods  - pods/log  - pods/attach  - pods/exec  - podtemplates  - replicationcontrollers  - secrets  - serviceaccounts  - services  verbs: ["\*"]  - apiGroups: [""]  resources:  - resourcequotas  verbs: ["get", "list", "watch"]  - apiGroups: ["batch"]  resources:  - jobs  - cronjobs  verbs: ["\*"]  ---  # Below code creates 'namespace-admin-rbg' RoleBinding which attaches the AAD Group or users to a Role with Namespace admin priveleges.  # Note: Email IDs ust be in lowercase only.  ---  kind: RoleBinding  apiVersion: rbac.authorization.k8s.io/v1  metadata:  name: namespace-admin  namespace: #{appname}#-#{iteraplanID}#-#{environment}#  subjects:  - kind: Group  name: #{groupid}# # group only  roleRef:  kind: Role #this must be Role or ClusterRole  name: namespace-admin # must match the name of the Role  apiGroup: rbac.authorization.k8s.io  ---  # Note: Role to access ingrss-controller namespace to reader ingress-controller logs  apiVersion: rbac.authorization.k8s.io/v1  kind: RoleBinding  metadata:  name: #{appname}#-#{iteraplanID}#-#{environment}#-reader  namespace: ingress-controller  roleRef:  apiGroup: rbac.authorization.k8s.io  kind: Role  name: ingress-reader-access  subjects:  - apiGroup: rbac.authorization.k8s.io  kind: Group  name: #{groupid}# # group only  ---  # Note: Role to list all namespaces in the cluster  apiVersion: rbac.authorization.k8s.io/v1  kind: ClusterRoleBinding  metadata:  name: #{appname}#-#{iteraplanID}#-#{environment}#-reader  roleRef:  apiGroup: rbac.authorization.k8s.io  kind: ClusterRole  name: uniper-reader  subjects:  - apiGroup: rbac.authorization.k8s.io  kind: Group  name: #{groupid}# # group only  ---  # Note: Network policy this will alllow communication within the namesapce and ingress controller  apiVersion: networking.k8s.io/v1  kind: NetworkPolicy  metadata:  name: default-network-policy  namespace: #{appname}#-#{iteraplanID}#-#{environment}#  spec:  podSelector: {}  policyTypes:  - Ingress  - Egress  ingress:  - from:  - podSelector: {}  - from:  - namespaceSelector:  matchLabels:  name: ingresscontroller |

## Supporting Links