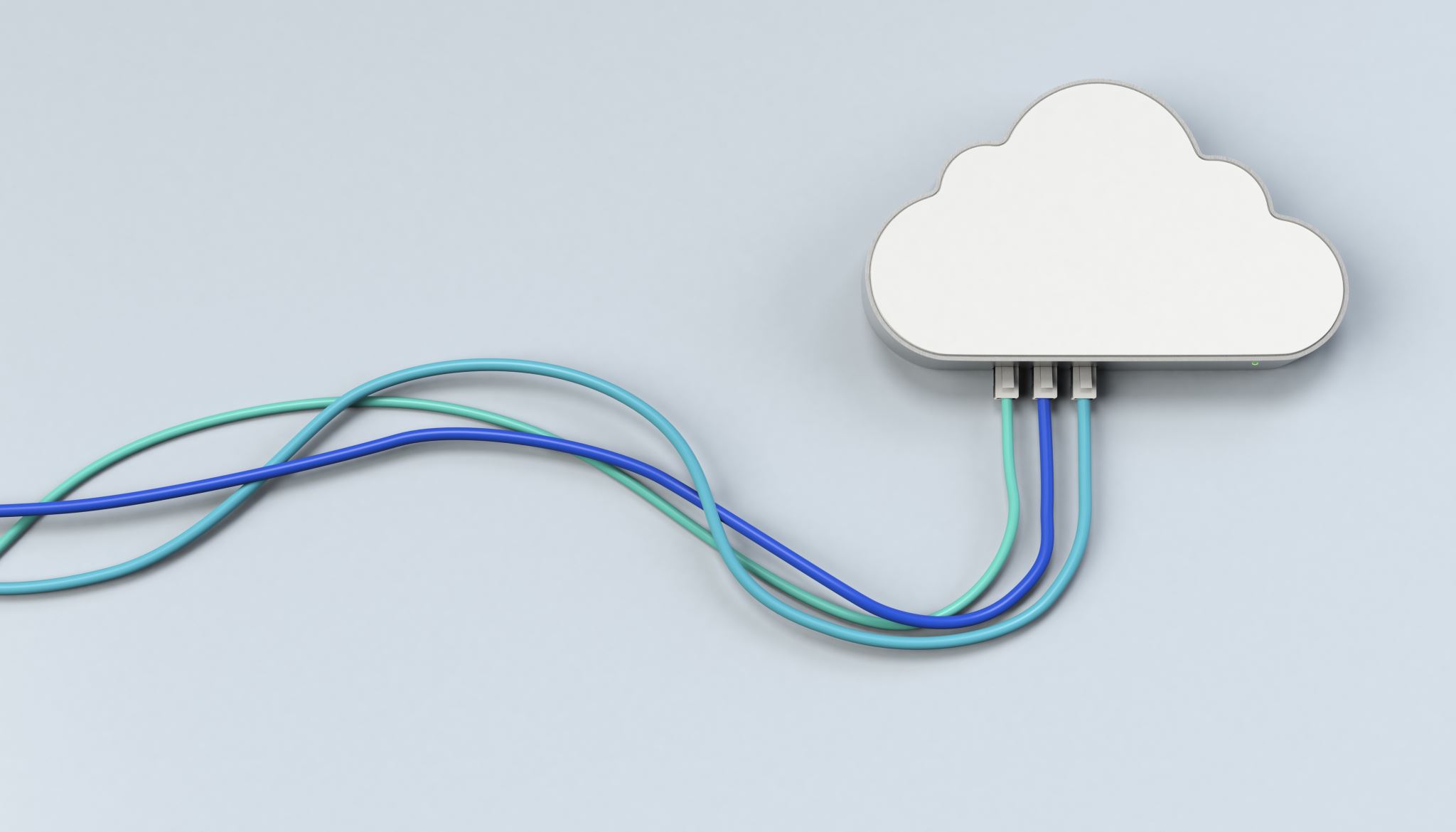
infe

Project Nexus (LLM Inference Multi-cloud Support)

An AzureML-Arc based solution

*This document will be revised based on the current Nexus architecture and components (6/16/23).*

# Motivation

Large language models are demonstrating their powers in many different domains – e.g., code generation (co-pilot), QnA (ChatGPT) - and it is just the beginning. In the meantime, the serving of these LLMs require a tremendous amount of GPU capacity, which might not be even fulfilled by a single cloud provider due to the supply-chain constraints. In the near future WebXT needs a significant batch of A100 GPUs that beyond what Azure can provide, so the team is looking into whether it is feasible to leverage the capacity from other cloud provider(s) and what is the right software stack for that to work. This document is describing a proposal based on [AzureML’s multi-cloud/on-premises solution](https://techcommunity.microsoft.com/t5/azure-arc-blog/realizing-machine-learning-anywhere-with-azure-kubernetes/ba-p/3470783#:~:text=Azure%20Arc-enabled%20ML%20enables%20teams%20of%20ML%20professionals,use%20patterns%20previously%20unthinkable%20in%20cloud%20setting%20environment.).

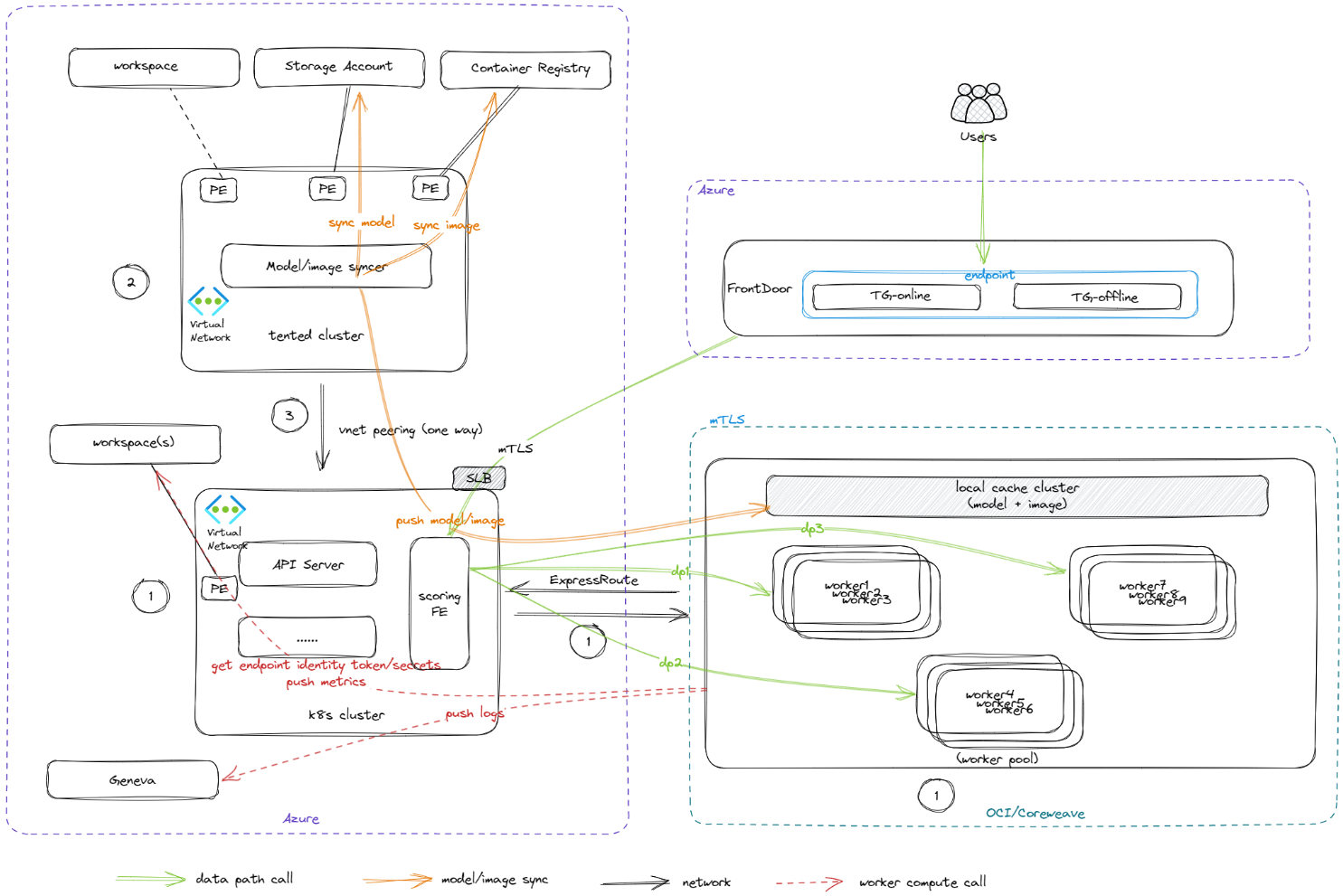
# Requirements

* Cluster setup
* Cloud provider cluster setup
* AzureML Arc compute setup
* Multi-node model hosting
* Node group level health probe
* Node group management (group allocation, re-grouping, etc)
* Group level traffic management
* DeploymentFlow integration
* Preset deployment workflow integration
* Quota Management
* Reliability and efficiency
* Customized group scheduling (UD/FD)
* Node group Gracefully shutdown
* Deployment resilience (bad node)
* Transparent Infra components update/vulnerability fix
* Bad node detection/eviction/replacement
* Security and Network
* Secure network between other cloud and Azure
* Data Exfiltration Prevention
* ~~Pull IPP model/container image in other clouds~~ (model/container will be hosted in Azure blob/ACR which are connected as private endpoints)
* System/Container events auditing
* Monitoring/observability
* Eyes-off mode DRI support
* Logs/Metrics (especially group level metrics like capacity, etc.)

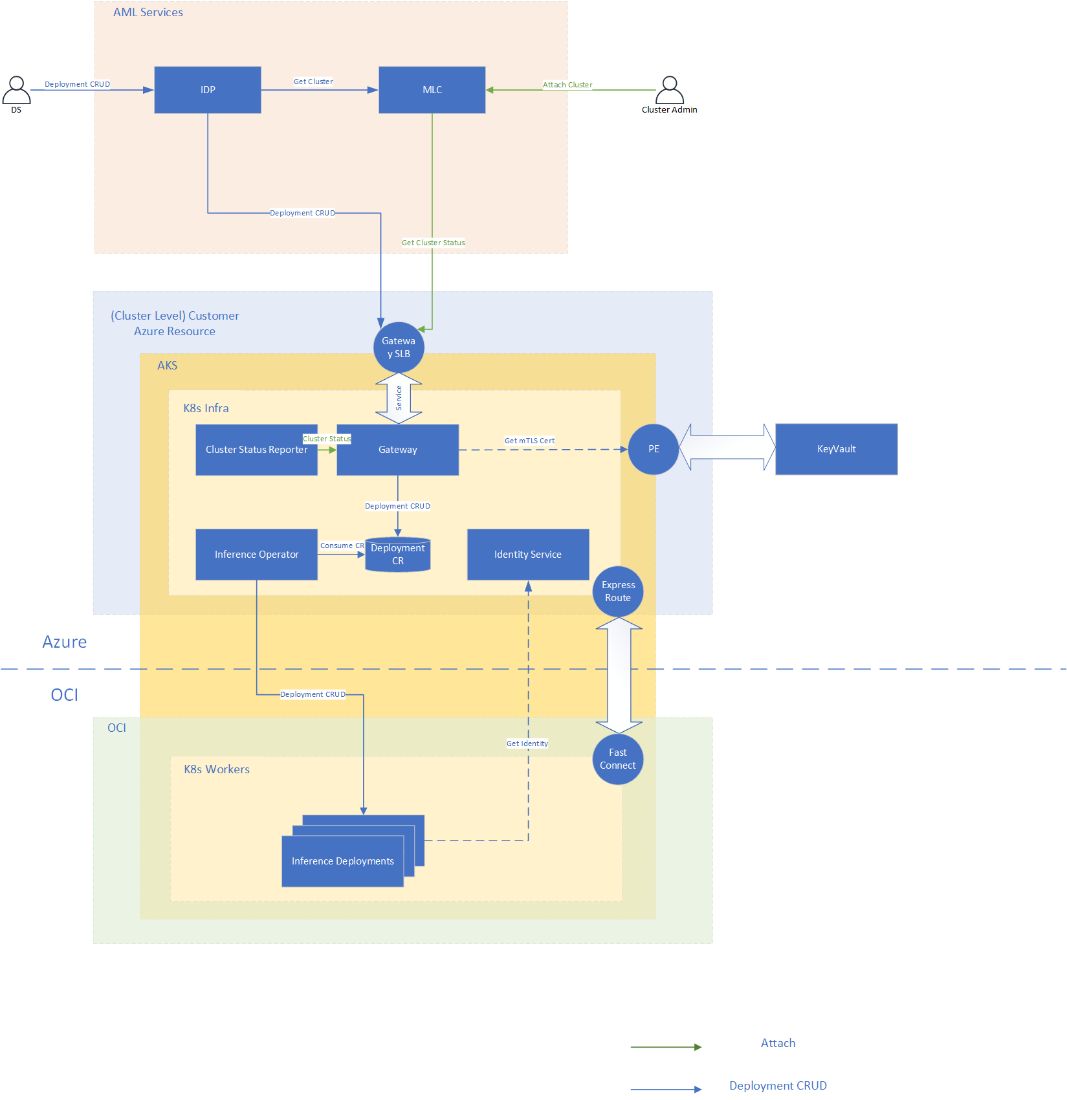
# Terminology

|  |  |
| --- | --- |
|  |  |
| Model instance | a group of nodes to serve as one model replica; the user can specify how many nodes they want to group into a single model instance; All scale operations on the deployment must be performed at the model instance level. |
|  |  |

# High-level Architecture



# Control Plane



## Network Security

A picture containing text, diagram, line, parallel

Description automatically generated

From a network perspective, the design contains the following network entities.

* OCI vcn. It is connected to Azure vnet via ExpressRoute and FastConnect.
  + All outbound and inbound traffics are disabled by default.
  + It can only access the peered Ainzure vnet via ExpressRoute/FastConnect connection.
* Azure vnet. It is connected to OCI vcn via ExpressRoute and FastConnect. NSG rules need to be applied according to the following network requirements.
  + K8s control plane is installed in this vnet.
  + Azure Arc agent is installed in this vnet.
    - Azure Arc - [Network requirements](https://learn.microsoft.com/en-us/azure/azure-arc/kubernetes/network-requirements?tabs=azure-cloud#details)
    - [Data exchange between cluster and Azure](https://learn.microsoft.com/en-us/azure/azure-arc/kubernetes/conceptual-data-exchange#data-exchange-between-cluster-and-azure)
  + Azureml extension is installed in this vnet.
    - [Network requirements](https://learn.microsoft.com/en-us/azure/machine-learning/how-to-access-azureml-behind-firewall?view=azureml-api-2&tabs=ipaddress%2Cpublic#scenario-workspace-created-with-the-hbi_workspace-flag-enabled) For system log, we have different design and will not use kusto.
  + Azureml Arc extension creates a private endpoint of Relay service for control plane.
    - Azureml IDP calls to the relay server via public address with trusted service firewall rule.
  + Azureml Arc extension creates public load balancer for Mesh on data plane. This public load balancer needs NSG rules to allow requests from FrontDoor.
  + waAzureml workspace creates a private endpoint in this vnet.
    - Identity Proxy calls Azureml identity bridge via this private endpoint.
  + How model and image is pushed to staging area in this vnet is TBD.
* NSG rules for Azure vnet needing below service tags:
  + Infra images download from MCR: **MicrosoftContainerRegistry, AzureFrontDoor.FirstParty**
  + Cluster certificate download from keyvault: **AzureKeyVault, AzureActiveDirectory**

## Open Questions for Security Review

In control plane and data plane, we would like to confirm if the following design is allowed from security perspective.

* Relay is used for communication between AzureML deployment plane (DP) and AKS cluster. We can create a private endpoint for the relay service in AKS vnet. From DP to relay service, we will disable public network access and enable trusted Microsoft services to bypass firewall.
* In data plane, a public load balancer will be created in AKS vnet for FrontDoor to call mesh in AKS. This is consistent with the current MIR design.
* Pull model/image/key from a scoped registry. There will be a separate user assigned identity which has access to the Blob/ACR/KV in scoped registry.
* Who’s responsible for encrypting model files, and setting up the UAI that can access the models/images/kvs?
* Is it allowed to pull the storage account(via PE), container registry(via PE) and KV from Nexus AML boundary?
* If the “scoped registry” proposal is approved, we will be in charge of publishing/deleting models from each scope?

# Data Plane

## MIR Frontdoor

User will send traffic to frontdoor, the endpoint/deployment user aware is MIR endpoint/deployment.

We need to pass the endpoint and deployment info from MIR frontdoor to Mesh so that Mesh has enough information to route the traffic.

* Add new deployment property “ComputeType”, if this endpoint is created for ARC, then set this property to “amlarc”For amlarc endpoint, apply the following changes:
  + Rewrite request Url to include endpoint name like this /api/v1/endpoint/{endpoint\_name}/…”, this is the request url format Mesh can handle.
  + Set request header “azureml-model-deployment” with the real mir deployment name.
  + Validate ARC cluster Cert, one ARC cluster will have only one cert, so this will be different from the endpoint level cert, we can use subscription as cert scope:.
    - Cert subject name: {SubscriptGUID}.{Region}.ml.azure.com

## Mesh

Mesh detail design spec could be found here at [Nexus\_dataplane\_in\_cluster.docx](https://microsoft.sharepoint.com/:w:/t/ProjectNexux-AzureMLinternal2/IQESwYFmZ5svTIh1Yz_IkvCaAX0-SsZkFMQ-uZIgKTqaZQI?e=dtO2qK)

* Mesh should have a public SLB so FD can forward request to it. Use the NSG and mTLS to verify the traffic between FD and Cluster.
* Disable Auth since auth is already done by FD.
* Enable mTLS
  + FD cert SAN: {region}.inference.ml.azure.com
  + ARC cluster cert SAN: {SubscriptGUID}.{Region}.ml.azure.com
* Header based routing (already supported)
* Always route to Rank-0
  + Rank-0 pod should has label “rank-0”
* Sticky routing(align with MIR)
  + Pod level sticky routing (refer to MIR)
* Response code rewrite (align with MIR)
  + 429 (503 in scoringfe)
  + 424 (502 in scoringfe)
* MIR endpoint/deployment mapping with ARC endpoint/deployment
* Requirement for distributed inference:
  + Set all deployments pods with label rank info:

For example: rank: 0

For non distributed inference, rank: 0 needs to be added as well.

* + For distributed training, rank 0 becomes healthy when other ranks are healthy.
* Requirements for Preset:
  + When creating online deployment, register the ScoringFE service IP info to the frontdoor. And endpoint type.
* Scoringfe to model worker with http in cluster (align with MIR)
* Envoy metrics publish to geneva, currently we only have QPM and latencies, need to figure out the gap and add the missing metrics
* Scoringfe forward request to CMP port (not model port) if CMP is enabled

## Mesh certificate management



~~Scenarios when installing an extension:~~

* ~~(manual)Generate certificate with ARC cluster cert SAN manually by using Keyvault in Vienna infra regional subscription with Name using cluster name. (JIT by AME account)~~
* ~~(manual) Get the AML extension subscription ID and add the subscription ID to identity bridge config whitelist. (JIT by AME account)~~
* ~~(manual)Install the amlarc extension by sslSecret parameter with using a test certificate, keyvault name and certificate name.~~
  + ~~sslSecret : The name of the Kubernetes secret in the azureml namespace. This config is used to store cert.pem (PEM-encoded TLS/SSL cert) and key.pem (PEM-encoded TLS/SSL key), which are required for inference HTTPS endpoint support when allowInsecureConnections is set to False. Details could be found~~ [~~here~~](https://learn.microsoft.com/en-us/azure/machine-learning/how-to-deploy-kubernetes-extension?view=azureml-api-2&tabs=deploy-extension-with-cli)~~.~~
* ~~(manual)Get the amlarc extension client id and add it to the identity bridge config whitelist managed identity and assign the kevvault the keyvault reader access. (key vault and the extension needs in the same tenant(AME).)~~
* ~~(Automation) CertGetter in amlarc extension in the cluster to download the certificate from the Keyvault every 24 hours by using the extension msi by calling identity bridge. Identity bridge verifies the extension msi and sends related certificate. the keyvault. The service checks the certificate whether it is changed. If it is changed, it will update the sslSecret secret (See above chart).~~
* ~~(Automation)The Keyvault will update the certificate when it is near expiration and will be downloaded and updated to sslsecret secret in the cluster.~~

~~Work to do:~~

* ~~Update identity bridge to support the auth basing on extension token and the config whitelist~~
* ~~Add CertGetter in gateway in extension to support downloading the certificate and update the k8s secret.~~

Scenarios when installing the helm chart:

* Infra team (one time work): Config the subscription(which holds the keyvault) in the onecert with certificate domain name: nexus.<location>.cloudapp.azure.com.
* Infra team: Create Keyvault which apply a certificate with
  + Subject: <cluster-name>.nexus.<location>.cloudapp.azure.com
  + SAN:

DNS Name= <cluster-name>.nexus.<location>.cloudapp.azure.com

DNS Name=\*. <cluster-name>.nexus.<location>.cloudapp.azure.com

DNS Name=\*. {SubscriptionGUID}.{shortRegion}.ml.azure.com

* Infra team: Assign cluster identity to the keyvault reader access.
* Infra team: Install nexus by parameter keyvault name, cluster name, region
* App team: CertGetter in the cluster to download the certificate from the Keyvault every 24 hours by using the cluster msi by calling the keyvault.The service checks the certificate whether it is changed. If it is changed, it will update the k8s secret.

Notes: The Keyvault will update the certificate when it is near expiration and will be downloaded and updated to sslsecret secret in the cluster.

## CMP

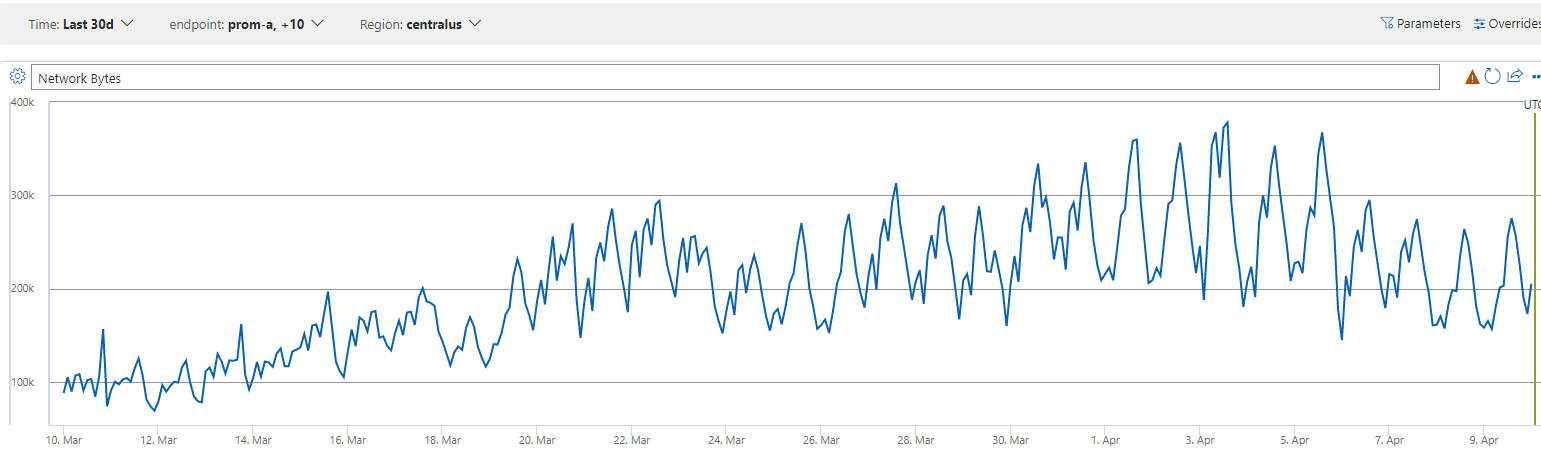
* Update user model deployment spec to include cmp as side-car
* CMP can get UAI (for RAI access) token (from identity proxy -> identity bridge?)
* CMP can access RAI endpoint (proper NSG config)
* Where to pull CMP image? From MCR? Is there security concerns for DV3?

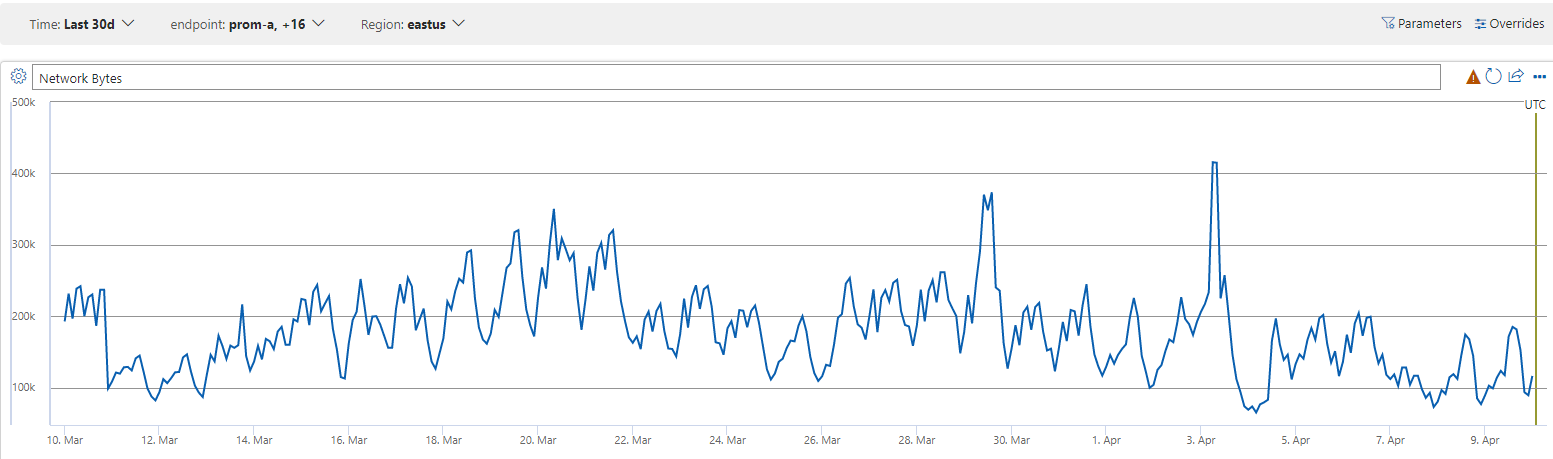
## Model Data Collection (MDC)

* Align with MIR
* Details could be found at [here](https://github.com/Azure/azureml-monitoring-preview/blob/main/mdc-documentation/mdc-payload-logging-with-cli.md#2-enable-private-preview-features-in-your-environment).
* Model Data Collection (MDC) (preview) supports three data sink options:
  + Workspace Blob Storage: Use workspace private endpoint
  + Custom Blob Storage URI: To check.
  + EventHub： Eventhub service tag needed if using Eventhub.

Bandwidth Usage

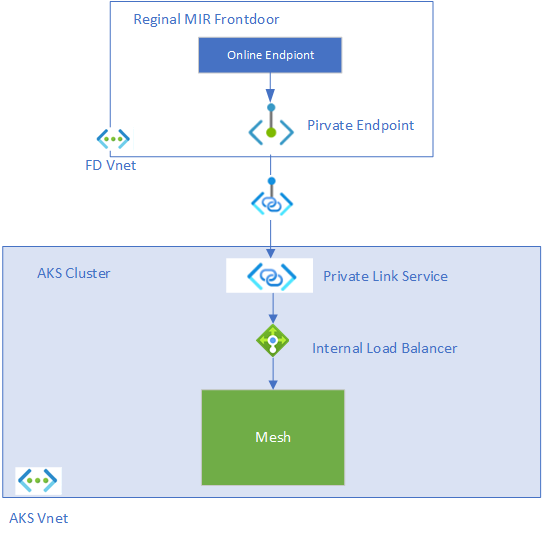
[MIR frontdoor dashboard](https://portal.microsoftgeneva.com/s/BC41BE37?overrides=%5b%7b%22query%22:%22//*%5bid='endpoint'%5d%22,%22key%22:%22value%22,%22replacement%22:%22prom-a,prom-a-nd40,prom-batch-nd40,prom-d-nd40,prom-ppo-32k,prom-ppo-a,prom-ppo-a-nd40,prom-sahara,prom-sahara-a,prom-sahara-eval,prom-vesta-eval%22%7d,%7b%22query%22:%22//*%5bid='Region'%5d%22,%22key%22:%22value%22,%22replacement%22:%22centralus%22%7d%5d%20)





Private Endpoint Option (Optional)

If public SLB on AKS cluster passes security review, then we will take public SLB option. Otherwise, this private endpoint option will be used to make the traffic through private link from MIR Frontdoor to Mesh.



The flow to manage the private endpoint is:

1. In ARC cluster creation flow
   1. Create internal load balancer for Mesh
   2. Create private link service for intenral load balancer, get the private link service resourceId
   3. Create PE for all reginal Mir frontdoors (use which identity? How to manage the FD list?)
   4. Get the PE private IP, this IP will be used later for deployment (where to store this IP?)
2. In online deployment creation flow
   1. Get the PE IP that online deployment will be created on, and set the IP as target IP in frontdoor route table

# Data exfiltration prevention

DV3 has a very high-standard security requirement to make sure the model & image’s safety, in MIR, this is done by using Managed Vnet with very limited outbound whitelist(NSG rules):

1. Tented ACR & Storage
2. Logs -> [Geneva Ingestion Gateway (GIG)](https://eng.ms/docs/products/geneva/collect/references/genevaingestiongateway)
3. Metrics -> Runhistory (via PE)
4. Certificate -> AML service (via PE)
5. Token
   1. AML service (for online endpoint identity, via PE)
   2. IMDS (tented identity, via azure proxy vm)

For AML Arc, we need to think about what outbound connections are required and need to have them reviewed with DV3 security team.

AML Arc outbound connections:

1. Azure Relay
2. Azure Event hub
3. Azure Arc ([network requirements](https://learn.microsoft.com/en-us/azure/azure-arc/kubernetes/quickstart-connect-cluster#meet-network-requirements))

Network requrements

AzureML Extension

* [Network requirements](https://learn.microsoft.com/en-us/azure/machine-learning/how-to-access-azureml-behind-firewall?view=azureml-api-2&tabs=ipaddress%2Cpublic#scenario-workspace-created-with-the-hbi_workspace-flag-enabled)

Azure-Arc

* [Network requirements](https://learn.microsoft.com/en-us/azure/azure-arc/kubernetes/network-requirements?tabs=azure-cloud#details)
* [Data exchange between cluster and Azure](https://learn.microsoft.com/en-us/azure/azure-arc/kubernetes/conceptual-data-exchange#data-exchange-between-cluster-and-azure)

# Preset Flow for AMLArc

See https://microsoftapc-my.sharepoint.com/:w:/g/personal/mingj\_microsoft\_com/Ed7PvVayyL5NrJAs75jV8ncBcbK2jBkp6EnG0rFRL05V-A?e=Ejvucy

# Model & Image Download

We are adopting a push model where models/weights are “pushed” from tented registry/blob into third-party worker pool’s private network.

# Questions:

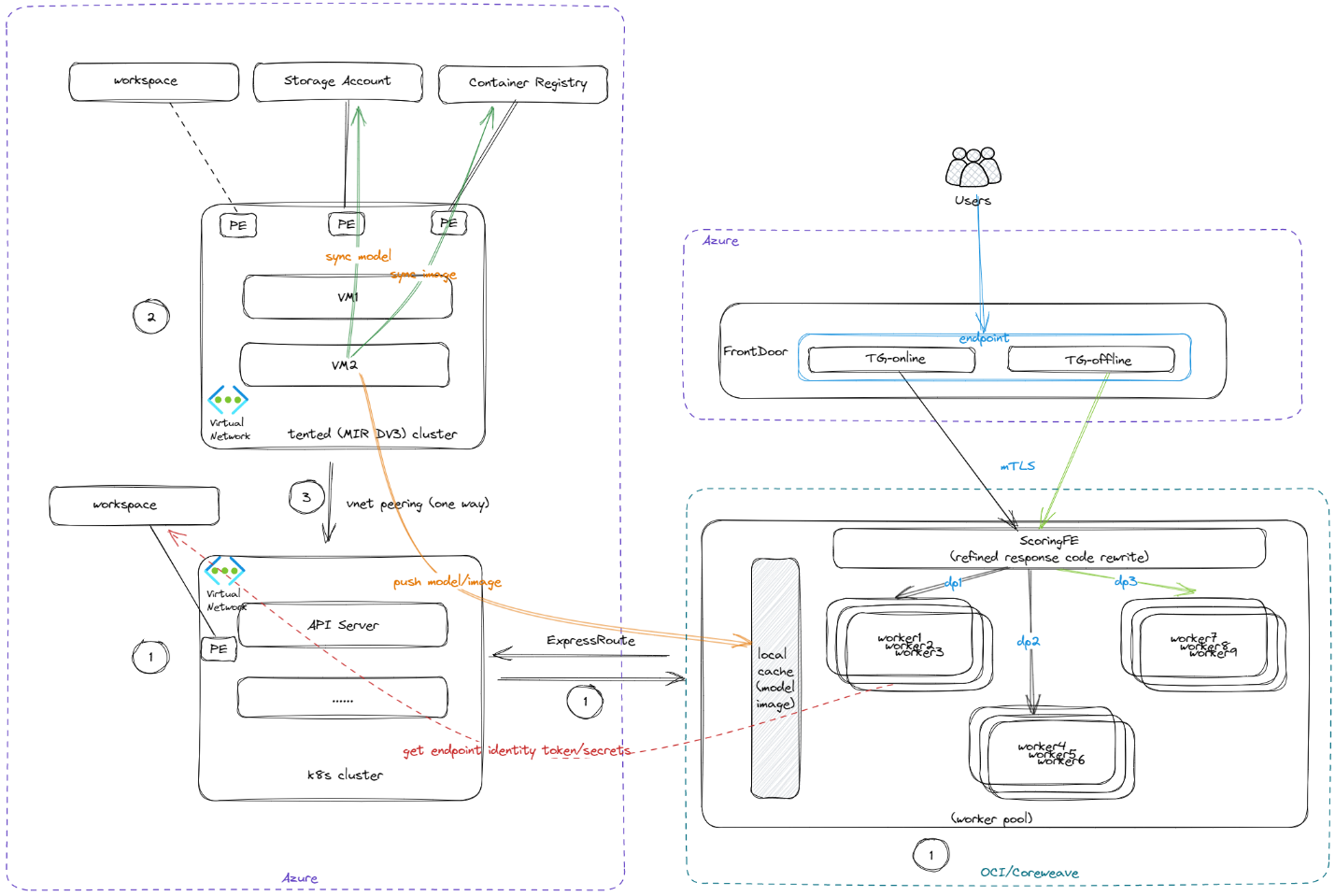
* Which service should be responsible for syncing the model/image from Azure to local cluster storage, when the sync is triggered?
* How many models/images need to be maintained in local storage, what the estimation total storage size is, especially, how to retire old versions, manually or automatically delete oldest version?
* How is CMP image downloaded? From MCR or local image registry?
* Are the local storage nodes stable? Does the membership need to be dynamically updated frequently?
* How many models are to be cached, and how to maintain, especially, how to retire old versions? - At lease 4-5 version(1.2T per version)
* Is it safe that when keep the model files / images on a node after the deployment is removed? - Not a must.
* …

There are 3(or could be more) potential paths to push down the needed model and weights:

1. Artifact push service directly pushes data to in-cluster cache within the 3rd party network boundary. Deployment pulls data from in-cluster cache.
2. Artifact push service pushes data to a Staging area in Azure vNet boundary, and a service pushes data down(or sync-ed from) to in-cluster cache within the 3rd party ntework boundary. Deployment pulls data from in-cluster cache.
3. Artifact push service pushes data to a cache in Azure vNet boundary, and deployment pulls data from in-Azure cache.

Option 1: potential security concerns

For Option 2, one potential implementation mechanism is to use a MIR cluster as a proxy



We will pre-configure a MIR cluster which has permission to download the model the cluster. This MIR cluster will be vnet-peering with AmlArc cluster. When user deploy new deployment to the AmlArc cluster, the workflow will check if the model exists on the MIR cluster and do file sync to get the model from the MIR cluster.

## Proposed storage solution

### Model & Image publish

* TBD

### Image download

* Image is downloaded from the asset registry service by kubelet directly. The asset registry follows the OCI distribution spec.

### Model download

* User pod has a PVC pointing to the required model version.
* The corresponding PV should be created when the model is published to the OCI cluster.
* The PV is attached (downloaded actually) to the node where the user Pod is scheduled. The download action is triggered by CSI controller plugin when it receives ControllerPublishVolume RPC call and is done by CSI node plugin.
* CSI node plugin is a Daemonset running on every worker node and acts as the download client, it first tries to resolve the model manifest from the cluster asset registry and get the full list of storage items to download, then it starts the download in parallel from the local storage cluster.

### Model & Image reconcile / repair

* The asset push service in Azure should periodically poll the cluster asset registry service to check whether there’s any asset broken and try to push it again.
* The asset registry service in the cluster should maintain the latest health state of each asset in case the underlying storage cluster.

A picture containing text, diagram, plan, parallel

Description automatically generated

# Metrics and Logging

[OCI-Network-Logging-Metrics-Review.docx](https://microsoft.sharepoint-df.com/:w:/t/AzureArc-enabledML/ESLDv91_hX1FmlMdx9xyD2YBR7lwTufQu3x2OSIuJxHJeA?e=LhiWHH)

This table lists the metrics/logs currenlty used by DV3 (refer to The [DV3 dashboard](https://dataexplorer.azure.com/dashboards/603ebed2-246f-4bc2-8aa1-66100cd6bd0a?p-_startTime=6hours&p-_endTime=now&p-endpoint=v-prom-sahara-a&p-kustoregion=v-viennausc&p-overrideEndpoint=all&p-UnifiedFeed=v-Yes&p-statusCode=all&p-deployment=all&p-logFilterText=all&p-amlErrorReason=all&p-bucketSize=v-10m&p-takeLines=v-1000&p-interestingLogFilter=all&p-SingularityKustoRegion=v-openaiusc.centralus&p-collapse=v-none&p-MaxGeneratedTokensCapping=all#3cca8fad-ec00-45e5-b0bc-9fa659e7fe25) in MIR):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Area | Metric Name | Source | Source Table | Comments | required in OCI |
| Availability | Availability | Frontdoor | AzureMLFrontdoorAccessLog | Measure Request success rate SLA | no |
| Requests | RequestCount | Frontdoor | AzureMLFrontdoorAccessLog | Measure traffic volume | no |
| Latency | TotalDuration | Frontdoor | AzureMLFrontdoorAccessLog | Measure request latency | no |
| Latency | ResponseDuration | Frontdoor | AzureMLFrontdoorAccessLog | Measure request latency | no |
| EngineAPI | Response Code | Inference-Server | AMLManagedComputeLogs | Measure request volume by different response code  rc=strcat(statusCode, '-', onlineEndpointName, '/', deploymentName) | yes |
| Redis Event | redisNodeMaintenanceScheduled | Inference-Server | AMLManagedComputeLogs | Measure different redis events count | yes |
| Redis Event | redisNodeMaintenanceStarting | Inference-Server | AMLManagedComputeLogs |  | yes |
| Redis Event | redisNodeMaintenanceStart | Inference-Server | AMLManagedComputeLogs |  | yes |
| Redis Event | redisNodeMaintenanceFailoverComplete | Inference-Server | AMLManagedComputeLogs |  | yes |
| Redis Event | redisNodeMaintenanceEnded | Inference-Server | AMLManagedComputeLogs |  | yes |
| Tokens per bin | n\_context\_tokens | Inference-Server | AMLManagedComputeLogs | Measure token generation | yes |
| Tokens per bin | n\_generated\_tokens | Inference-Server | AMLManagedComputeLogs |  | yes |
| Tokens per bin | total\_tokens | Inference-Server | AMLManagedComputeLogs |  | yes |
| Leader Controller | LeaderControllerNow | VMAgent  Inference-server  AzureBatch | AMLManagedComputeLogs  WATaskAccount | Table to list more info about the current leaderController | yes |
| Build Details | Build Version | Inference-Server | AMLManagedComputeLogs | List the current build version for engine and osimage | yes |
| Links | Links | Frontdoor | AzureMLFrontdoorAccessLog | List usful links for data dog and dashboard | no |
| Bytes RX/TX | RxBytes | Frontdoor | AzureMLFrontdoorAccessLog | Mesure inbound/outbound bytes | no |
| Bytes RX/TX | TxBytes | Frontdoor | AzureMLFrontdoorAccessLog |  | no |
| Forward Pass Duration By Ring | Duration | Inference-Server | AMLManagedComputeLogs |  | yes |
| Rings with Fwd Pass | Count of Rings | Inference-Server | AMLManagedComputeLogs |  | yes |
| Forward Pass Presence by Ring | Count | Inference-Server | AMLManagedComputeLogs |  | yes |
| CUDA Errors | Count | Inference-Server | AMLManagedComputeLogs | Mesure CUDA errors | yes |
| GPU Subscribers | Count | Inference-Server | AMLManagedComputeLogs | Measure Liveness ping subscribers count | yes |
| Contact Support | Count | Inference-Server | AMLManagedComputeLogs | Measure contact support count | yes |
| Trend of active node count | activeNodeCount | Inference-Server | AMLManagedComputeLogs | Measure active node count in ring | yes |
| Liveness Probe Failed | liveness\_failed | Inference-Server | AMLManagedComputeLogs | Measure liveness probe failed count | yes |
| Restarts | restarts | Inference-Server | AMLManagedComputeLogs | Measure restart count | yes |
| Batcher channel closed by controller | batcher\_closed | Inference-Server | AMLManagedComputeLogs |  | yes |
| 429 reasons | Count of different reasons | Inference-Server | AMLManagedComputeLogs | Mesure different 429 reasons | yes |
| FD Logs |  | Frontdoor | AzureMLFrontdoorAccessLog | frontdoor logs | no |
| Content Moderation Proxy |  | CMP | AMLManagedComputeLogs | cmp logs | yes |
| Engine Logs |  | Inference-Server | AMLManagedComputeLogs | engine logs | yes |
| Engine Logs Histogram | Count | Inference-Server | AMLManagedComputeLogs | Measure engine logs volume | yes |
| Pipereplica Logs |  | Inference-Server | AMLManagedComputeLogs | pipereplica logs | yes |
| ERRORS in each component | Count | Inference-Server | AMLManagedComputeLogs | engine error distribution | yes |
| Token from Requests | prompt\_total, generated\_tokens | Inference-Server | AMLManagedComputeLogs | billing details | yes |
| Request Count (EngineAPI) |  | Inference-Server | AMLManagedComputeLogs |  | yes |
| Requests Latency (EngineAPI) |  | Inference-Server | AMLManagedComputeLogs |  | yes |
| Generation Forward Pass time | gpu\_duration\_ms | Inference-Server | AMLManagedComputeLogs |  | yes |
| Prompt Processing Time | total\_blocking\_gpu, recv\_gpu | Inference-Server | AMLManagedComputeLogs |  | yes |
| Memory Limit |  | Inference-Server | AMLManagedComputeLogs |  | yes |
| Not enough KV blocks |  | Inference-Server | AMLManagedComputeLogs |  | yes |
| KV Block Usage and 429s |  | Inference-Server | AMLManagedComputeLogs |  | yes |
| Token Generation Batch size (Pipereplica) |  | Inference-Server | AMLManagedComputeLogs |  | yes |
| CPU/GPU time (Pipereplica) |  | Inference-Server | AMLManagedComputeLogs |  | yes |
| Token Loading Latency (EngineAPI) |  | Inference-Server | AMLManagedComputeLogs |  | yes |
| InstanceCount Expected vs Actual |  | Analytics | DeploymentsState |  | no |
| Request Latency by Token Generation (EngineAPI) |  | Inference-Server | AMLManagedComputeLogs |  | yes |
| Context Tokens Distribution |  | Inference-Server | AMLManagedComputeLogs |  | yes |
| Generated Tokens Histogram |  | Inference-Server | AMLManagedComputeLogs |  | yes |
| Timespan Layer Cake |  | Frontdoor  Envoy L1/2  CMP  Inference-Sever | AzureMLFrontdoorAccessLog  AMLManagedComputeLogs |  | yes |
| Billing Log Loss |  | Frontdoor  Inference-Server | AzureMLFrontdoorAccessLog  AMLManagedComputeLogs | The billing log loss tile utilizes the endpoint, deployment, and time range parameters to find the number of requests received by frontdoor that do not have an associated billing log and are therefore unaccounted for in context load and token generation rates. | yes |

# Identity

[Identity On Nexus.docx](https://microsoft.sharepoint.com/:w:/t/ProjectNexux-AzureMLinternal2/EW2lDR9LYM5DjUrZte1sqHQBNp0f3LORH9SbevjUZsnDJQ?e=0i81vx)

# Health Monitoring and Repair

We leverage [Compute Health Manager Draft Design.docx](https://microsoft.sharepoint-df.com/:w:/t/AzureArc-enabledML/EW-2qlwq5nZOsYgrpgcncQABT9inrakoYH81Kd6BXCYXPw?e=duoM8H) for node problem detection and analysis. In this session, we will focus on compute specific logic that includes workload precheck, workload repair, and node repair.

A picture containing text, screenshot, diagram, font

Description automatically generated

Two new services will be added:

* ClusterHealthManager: this service is responsible for workload repair, including cordoning bad nodes and draining workloads. It also acts as a node repair history provider, storing nodes' repair history in configmaps. Additionally, it monitors the health manager’s heartbeat of each node and node not-ready events to prevent situations where nodes are unable to report their unhealthy status themselves.
* NodeProblemResolver: running on each node to perform node repair. for hot repair and reboot, we prefer to do it locally; for action such as reimage and re-provisioning, the repair will be routed to a compute service like AKS or OCI.

Precheck flow:

Precheck is to ensure the node is in good status before user container is started. Node precheck will be triggered inside the init-container of the user pod. If the precheck fails, workload manager will take immediate action by cordoning the node and rescheduling the pod on other nodes.

A picture containing text, screenshot, diagram, line

Description automatically generated

Repair flow:

Azure core team might provide a new API to wrapper external compute related operations in the future, details have not yet come out, so in current design, we would call into OCI layer for node repair directly.

A picture containing text, diagram, screenshot

Description automatically generated

HealthManager currently requests four typical repair actions from NodeProblemResolver: hotrepair, reboot, reimage, and reprovision. For hotrepair, the action can be performed locally. However, for the other repair actions, we need to route the requests to OCI cloud service:

* Authentication:

OCI supports Instance Principal Authentication. After setting up required resources and policies, NodeProblemResolver running on each OCI instance can call Oracle Cloud Infrastructure public services, without the need to configure user credentials or a configuration file; This would require some additional steps during cluster setup to [authorize OCI workers to make OCI API calls](https://docs.oracle.com/en-us/iaas/Content/Identity/Tasks/callingservicesfrominstances.htm).

* Private access:

To ensure that this traffic does not go over the internet, we probably need to setup [service gateway](https://www.oracle.com/cloud/networking/service-gateway/service-gateway-supported-services/).

Additionally, if a node is in such bad health that it cannot trigger the repair itself, we will rely on ClusterHealthManager from Azure infra node to redirect the request to some other healthy node. We will let another worker make the call to OCI API service, ensuring a reliable repair process. as shown in the following diagram

A picture containing text, diagram, screenshot, font

Description automatically generated

Questions:

1. Will node management be supported by AKS service in the near future?
2. OCI does not support reimage right now, looking into alternative solutions that can achieve a similar outcome.

Appendix

Work item breakdown:

1. K8s compute setup & Networking
2. Connect customer vnet and OCI VCN via ExpressRoute&FastConnect
3. Create PE to arc, workspace, tented acr/blob(??).
4. Relay or proxy for DP->k8s cluster communication.
   1. Proxy requires code change in MLC and DP
5. Arc control plane should go through private link
6. Node Health manager
7. Node repair operator for OCI compute
8. Health manager to cordon node and drain job
9. Node repair config & node repair history
10. Node grouping
11. Expose correct envs for DV3.
12. Update identity flow to handle tented arc/blob
13. Handle scale up and down.
14. Add failure analysis.
15. Always send SIGTERM signal to rank 0 for gracefully shutdown.
16. Route scoring request to rank 0 pod.
17. Update scoringfe auto-scale logic.
18. Telemetry
19. Service log: Geneva
20. gpu/cpu/memory metrics: az monitor
21. Support for Sentinel for system/container events
22. Performance
23. Shared model
24. Redis of OCI
25. Model Download/Image pull
    1. Model/Image cache layer in the cluster
    2. Model/Image publish solutions
26. Presets workflow
    1. Registry and preset workflow support for AzureML Arc
27. Frontdoor with ScoringFE integration

# Summary of POC

**Overview**

|  |
| --- |
| A picture containing graphical user interface  Description automatically generated |

**Connect Azure network to OCI networks with ExpressRoute/FastConnect**

|  |
| --- |
| Diagram that shows Azure to O C I - single V Net. |

--

1. https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/connectivity-to-other-providers-oci

2. https://docs.oracle.com/en-us/iaas/Content/Network/Concepts/azure.htm

**Setup Network**

We created two virtual networks in Azure and Oracle respectively and connect them with an interconnection with Azure ExpressRoute and Oracle FastConnect. The maximum bandwidth is 10Gbps.

**Create workspace**

We created a private AzureML workspace with private resource (storage account, container registry and key vault) and connect them to the Azure virtual networks (VNet) via private endpoint.

We also configured the DNS records for the resources in Oracle virtual networks (VCN).

**Install AMLArc Extension and attach compute**

We created an Azure Relay connect it to the VNet via private endpoint. We installed AMLArc extension on OKE and attached it to the AzureML workspace via this relay.

**Deploy model**

1. **Submit request**: The user submits the deploy request from a VM in the Azure VNet.
2. **Start distributed deployment**: The AMLArc extension reads number of the workers (nodes) from environment variable of the deployment specification and create workers for the deployment.
3. **Download image and model**: Every worker downloads the image from the private registry and downloads model from private storage account (blob) to the OKE.
4. **Start MPI task**: AMLArc extension set up SSH environment for every worker and start distributed MPI task on all workers.

**Logs and metrics**

System logs are sent to Kusto by The AMLArc extension via public network.

Running "az online-deployment log" command could get deployment logs.

**Performance**

We run following test,

1. Network bandwidth test  
   Transmission from Azure VM to Oracle VM: **977.01Mbit/s**
2. Blob download test  
   Download from blob to Oracle VM: **532.44Mbit/s**  
   (Standard, GRS, AzCli 10 maximum-connections)

The model downloading relies on,

1. Storage Account (Blob)
   1. Account type.
   2. Number of total connections
   3. Throttling.
2. Network.
   1. Azure Virtual network gateway.
   2. ExpressRoute/FastConnect bandwidth
   3. OKE node (VM) bandwidth

For now, the test is limited by the total network bandwidth of the Oracle VM (0.7Gbps). We are also not sure if multiple interconnections could increase bandwidth.