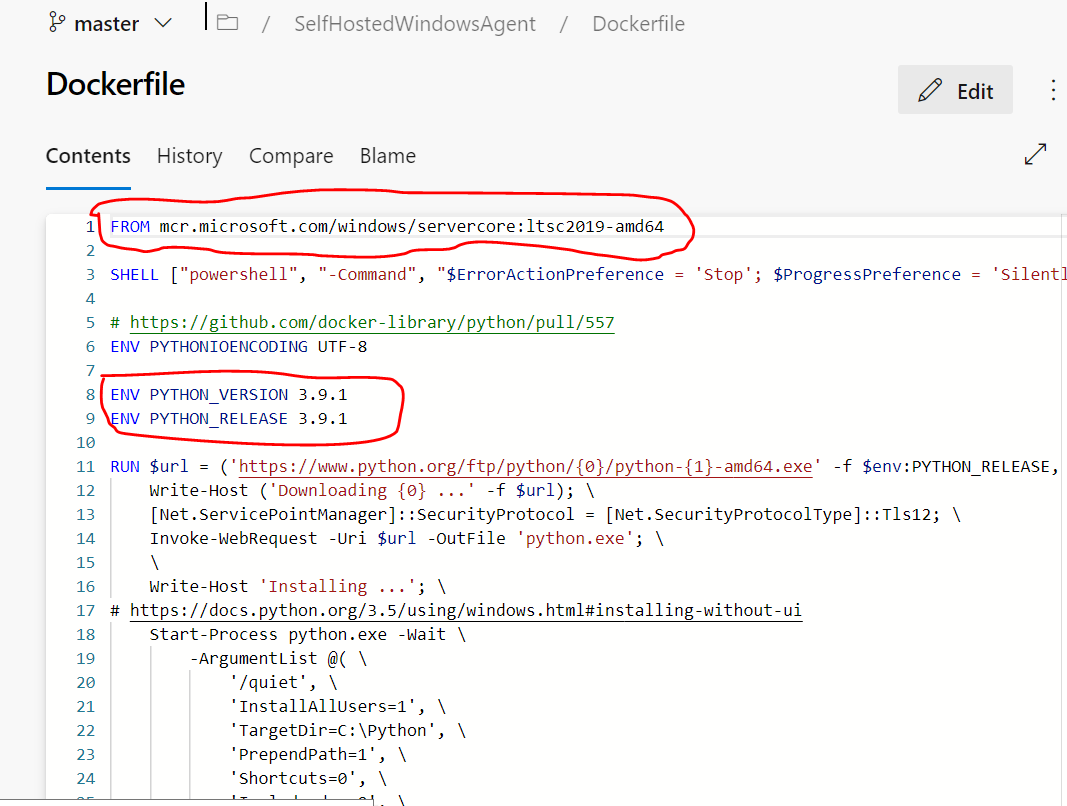
how patching will work ?

HaCT installs a [windows agent](https://dev.azure.com/uniperteamservices/_settings/agentpools?poolId=187&view=jobs) as a Docker container and deploy it through the [HaCT managed AKS Cluster](https://portal.azure.com/#@Uniper.onmicrosoft.com/resource/subscriptions/75310040-ed56-4e53-83e0-80f23ef48cd6/resourceGroups/cmcwind-paas-prd-rgp-002/providers/Microsoft.ContainerService/managedClusters/cmcwin-prd-001/overview).

And install other software’s like Python, SVN, ant, Java etc... On that machine as required by jobs.

* We need to update all vulnerabilities In the docker file as mentioned above in the path SelfHostedWindowsAgent/Dockerfile in repo.

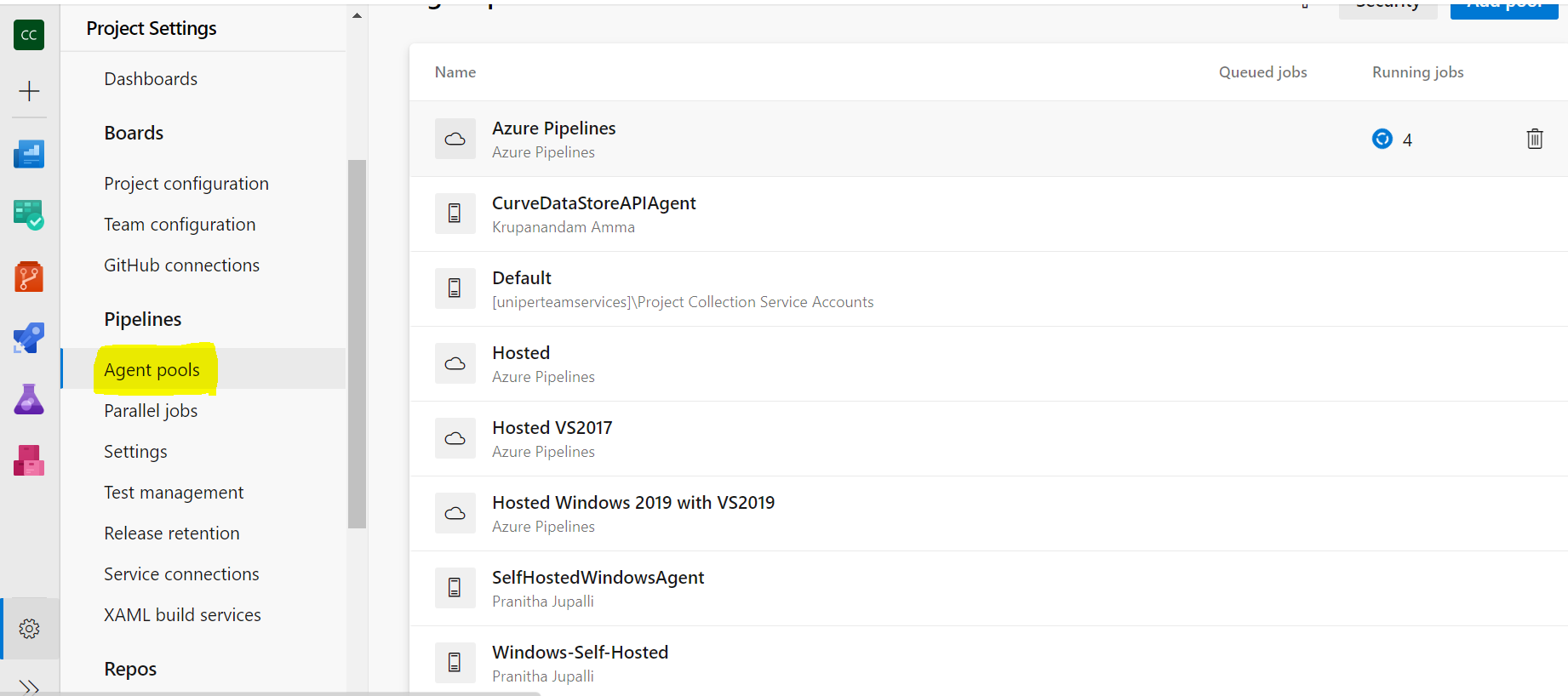


* To update the self hosted agent with new versions of capabilites, change the versions in the Dockerfile of the agent.
* To update python, please replace the old version with new version by setting the PYTHON\_VERSION, PYTHON\_RELEASE and PYTHON\_PIP\_VERSION.
* To update svn, please replace the URL of old version with the URL of newest version.
* To update git-lfs, please replace the URL of old version with the URL of newest version.

Then we need to run the build pipline named as “**selfHostedWindowsAgent**”.

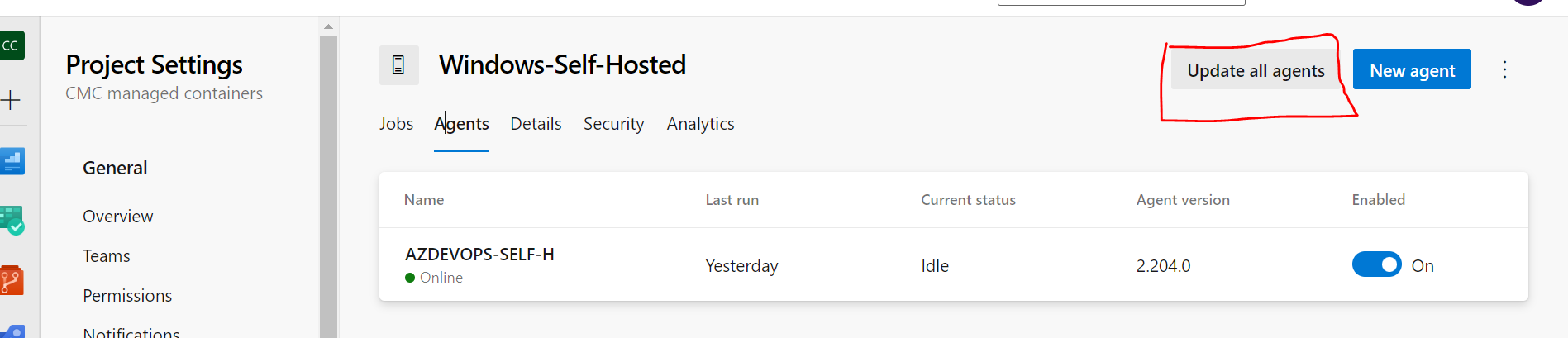
or

Click on the project Settings 🡪 scroll down to the Pipelines Section 🡪 Agent Pools as shown in the below image.



Click On the Agent Pools than it will list the Agent pools which are available in Projects as shown in above image.

From the list of Agent Pools, we need to select the Agent Pool and click on the same agent Pool then on the top bar we can find the “**Update all agents**” button as shown in the below image and click on it.



And we need to to update the latest images from the ACR repository Microsoft as below and we need to update the minor releases of the vulnerability software’s in the docker file.

**Schedule:** Every 3 months

We need to upgrade the agent during non- business hours.

**Responsibilities:**

HaCT is responsible for configuring and maintaining a proper configuration of docker windows agent – installing software’s and build tools, updating them, ensuring that all build servers in an agent pool have required configuration.

**Communication:** connect us on - [UIT-HACT-AKS@uniper.energy](mailto:UIT-HACT-AKS@uniper.energy)

Upgrading Kured

Check the kured version,chart and APP version

#helm ls -n kured

Search the kured repo

#helm search repo | grep -I kured

Upgrade the kured

#helm upgrade kured kured/kured -n kured

After update Check the kured version,chart and APP version

#helm ls -n kured

Pod will restart after upgrade We can check Kured pod is ready or not

#Kubectl get all -n kured

Note:-No Downtime require for this and no need to communicate to Application users for this.

Upgrading ingress-nginx

With Helm[¶](https://kubernetes.github.io/ingress-nginx/deploy/upgrade/#with-helm)

If you installed ingress-nginx using the Helm command in the deployment docs so its name is ingress-nginx, you should be able to upgrade using

helm upgrade --reuse-values ingress-nginx ingress-nginx/ingress-nginx

Without Helm[¶](https://kubernetes.github.io/ingress-nginx/deploy/upgrade/#without-helm)

To upgrade your ingress-nginx installation, it should be enough to change the version of the image in the controller Deployment.

I.e. if your deployment resource looks like (partial example):

kind: Deployment

metadata:

name: ingress-nginx-controller

namespace: ingress-nginx

spec:

replicas: 1

selector: ...

template:

metadata: ...

spec:

containers:

- name: ingress-nginx-controller

image: registry.k8s.io/ingress-nginx/controller:v1.0.4@sha256:545cff00370f28363dad31e3b59a94ba377854d3a11f18988f5f9e56841ef9ef

args: ...

simply change the v1.0.4 tag to the version you wish to upgrade to. The easiest way to do this is e.g. (do note you may need to change the name parameter according to your installation):

kubectl set image deployment/ingress-nginx-controller \

controller=registry.k8s.io/ingress-nginx/controller:v1.0.5@sha256:55a1fcda5b7657c372515fe402c3e39ad93aa59f6e4378e82acd99912fe6028d \

-n ingress-nginx

How does it relate to the cluster patching?

We need to check the compatibility of the ingress controllers with Kubernetes version. So we can upgrade this as per required version for particular Kubernetes Version.



Upgrading Secrets Store CSI Driver and Azure Key Vault Provider

helm upgrade to the latest chart release in the repo will update the Azure Key Vault Provider and Secrets Store CSI Driver to the compatible versions

**helm upgrade csi-secrets-store secrets-store-csi-driver/secrets-store-csi-driver --namespace=NAMESPACE**

Set NAMESPACE to the same namespace where the driver was originally installed, (i.e. kube-system)

**Example:-**

* This updates the driver version to v0.0.14+
* This updates the provider version to 0.0.9+
* This updates the driver manifest to include the flag --grpc-supported-providers=azure to enable communication between driver and provider using gRPC

Run the following commands to confirm the images have been updated -

1. secrets-store container in secrets-store-csi-driver pod is running v0.0.14+

➜ kubectl get ds -l app**=**secrets-store-csi-driver -o jsonpath**=**'{range .items[\*]}{.spec.template.spec.containers[1].image}{"\n"}'

mcr.microsoft.com/k8s/csi/secrets-store/driver:v0.0.14

1. secrets-store container in the secrets-store-csi-driver pod contains the arg --grpc-supported-providers=azure

➜ kubectl get ds -l app**=**secrets-store-csi-driver -o jsonpath**=**'{range .items[\*]}{.spec.template.spec.containers[1].args}{"\n"}'

**[**"--debug=true","--endpoint=**$(**CSI\_ENDPOINT**)**","--nodeid=**$(**KUBE\_NODE\_NAME**)**","--provider-volume=/etc/kubernetes/secrets-store-csi-providers","--grpc-supported-providers=azure","--metrics-addr=:8080"**]**

1. csi-secrets-store-provider-azure pod is running 0.0.9+

➜ kubectl get ds -l app**=**csi-secrets-store-provider-azure -o jsonpath**=**'{range .items[\*]}{.spec.template.spec.containers[0].image}{"\n"}'

mcr.microsoft.com/oss/azure/secrets-store/provider-azure:0.0.9

**How and when are users informed?**

We can inform the Users of that namespace where there Provider and Secret driver are installed through Email 2 weeks Prior with 3 Reminders.

**Who has which responsibility in the process?**

AKS team (HACT) will take care this activity completely.

**How does the communication work?**

Will send communication via email.

**How does it relate to the cluster patching?**

We can check the Compatibility of drivers and Providers with AKS.

# Helm Upgrade

**This command upgrades a release to a new version of a chart.**

helm upgrade [RELEASE] [CHART] [flags]

**Now We can check the version of Helm**

helm version –-template

# **How and when are users informed?**

No need to inform to user as no downtime require for this.

# **Who has which responsibility in the process?**

AKS team will take care this activity.

# **How does the communication work?**

No communication requires as no downtime need to take for this activity.

# **How does it relate to the cluster patching?**

We can check the Helm Version Support Policy. When a new version of Helm is released, it is compiled against a particular minor version of Kubernetes. For example, Helm 3.0.0 interacts with Kubernetes using the Kubernetes 1.16.2 client, so it is compatible with Kubernetes 1.16.

For example, if you are using a version of Helm 3 that was compiled against the Kubernetes 1.17 client APIs, then it should be safe to use with Kubernetes 1.17, 1.16, 1.15, and 1.14. If you are using a version of Helm 2 that was compiled against the Kubernetes 1.16 client APIs, then it should be safe to use with Kubernetes 1.16 and 1.15.

| **Helm Version** | **Supported Kubernetes Versions** |
| --- | --- |
| 3.9.x | 1.24.x - 1.21.x |
| 3.8.x | 1.23.x - 1.20.x |
| 3.7.x | 1.22.x - 1.19.x |
| 3.6.x | 1.21.x - 1.18.x |
| 3.5.x | 1.20.x - 1.17.x |
| 3.4.x | 1.19.x - 1.16.x |
| 3.3.x | 1.18.x - 1.15.x |
| 3.2.x | 1.18.x - 1.15.x |

# Steps to create a new cluster in Azure

## Step 1

Create a Snow request for the resource group creation by using below URL with Subscription, App name, Location, Target Environment, Resource Group name, Resource Group owner, EAM\_ID, Users who need access

<https://uniperprod.service-now.com/unipersp?id=sc_cat_item_uni&sys_id=ab56d3dadbf82c10d1ebbaf1f3961995>

then we will receive a shopping card with all the details for review from the Architect team. Once we approve, we will get a Resource group.

## Step 2

We need to create Subnet, same as above we need to create one SNOW ticket to the network team by using below URL:

https://uniperprod.service-now.com/unipersp/?id=sc\_cat\_item\_uni&sys\_id=706657dadbf82c10d1ebbaf1f396197a

## Step 3

Next we need to crate a managed Identity from the portal and we need to get a access on V-Net for the same we need to send a mail as below details to the security team([uit-hact-security-services@uniper.energy](mailto:uit-hact-security-services@uniper.energy)):

routetable name            : rt-iaas-services-high-npd-fuse-aks-v2-snet-XXX

Subnet                             : high-npd-fuse-aks-v2-snet-XXX

Vnet                                 : high-nrd-vnet-XXX

Manage Identity name : cmc-prehisech-identity

## Step 4

We need to run a pipeline in “**CMC managed containers**” project in Azure DevOps,

* Go to the release pipelines and create new release in the “**AKS Cluster Creation using Terraform**” .
* Update the variables and click on deploy as shown below.

