Issues regarding applications in Kubernetes

Last updated by | Sughasini Karmugilan | Apr 9, 2024 at 11:50 AM GMT+2

Common issues for applications in an AKS cluster and their troubleshooting.

> Errors are listed from generic to specific, hence subsections of an error are subsets of it. To solve the innermost one, please also check the solution for its parent.

Contents

Unable to recognize "": no matches for kind "XXX" in version "YYY/ZZZ"

Example of error message

> unable to recognize "": no matches for kind "HorizontalPodAutoscaler" in version "autoscaling/v2beta1"

Root cause
You are trying to:

- create a resource using an API version which is not currently available in the cluster, or
- update an existing resource which **current** definition is using an API version not currently available in the cluster.

Solution

1. Check what API versions are available in your cluster:

```
$ kubectl api-versions
aadpodidentity.k8s.io/v1
admissionregistration.k8s.io/v1
[…]
v1
```

2. If you do not find the one in the error, check the deprecated API migration guide as it might have been deprecated and you need to change it in your code;
3. Check the upstream reference for any of the **available** API versions you want to use;
4. Correct your manifest accordingly;
5. Apply the new, correct manifest.

UPGRADE FAILED: current release manifest contains removed kubernetes api(s) for this kubernetes version

Example of error message

> Error: UPGRADE FAILED: current release manifest contains removed kubernetes api(s) for this kubernetes version and it is therefore unable to build the kubernetes objects for performing the diff.

Root Cause

The resource you are trying to upgrade is defined with an API version that existed before, but does not exist anymore. This is mostly due to an upgrade to the Kubernetes version.

Solution

Remove the old resource, then recreate it with an API version that currently exists in the cluster.

Admission webhook "validation.gatekeeper.sh" denied the request

Error message example

> Error from server (Forbidden): error when creating ".\pod.yaml": admission webhook "validation.gatekeeper.sh" denied the request: [azurepolicy-k8sazurev2containerallowedimag-3d911ced0696013a2f87] Container image

Root cause

Gatekeeper is denying you to use a Kubernetes resource which has one or more attributes violating a policy.

This can be due to one or more Azure Policies for FSCP 3.0.
It's the same issue as cluster sesources violating a policy, but restricted to policies enforced by Gatekeeper.

Solution

1. Read the whole error message.
   Gatekeeper should give you a policy name and an explanation. If that is not enough:
   1. Check the policy given by Gatekeeper:
      ```
      $ kubectl get constrainttemplates.templates.gatekeeper.sh
      […]
      k8sazurev2containerallowedimages   51d
      ```
   2. Try to understand what is asking of you:
      ```
      $ kubectl get constrainttemplates.templates.gatekeeper.sh k8sazurev2containerallowedimages -o jso
      package k8sazurev2containerallowedimages
      […]
      ```

2. Adjust your code to satisfy the policy.
   Keep the index of policy driven values for parameters at hand as a cheatsheet.
3. Deploy again your resources with the new changes.
4. Go back to point 1 and follow the steps again until the resources are deployed correctly.

Pods keep getting terminated

Check the pods' logs (`kubectl logs`) and the cluster's events (`kubectl get events`).
They usually contain one or more reason for the termination of the pod.

Pods keep getting evicted

Root cause
Eviction has little to do about overcommitting. It has instead a lot to do about **effective resource usage**.
Check the current nodes and pod usage (`kubectl top`) to see how much of their resources are actually used.

This issue is not about pods' resource request or limits. It's about **overcrowding**.
When a node's resources are approaching full use, it starts not accepting or even evicting pods to keep the load manageable. If the pods have no priority nor quality of service set, the node will not discriminate what pod to evict.

Workaround
Try and use nodes with more resources so that the usage of the nodes' resources goes below 100% at all times.

Keep in mind the chosen node size must support encryption at host .
Check:

- Azure VM comparison for an easy reference table;
- Find valid parameter values in Azure.

Pods cannot connect to each other

Root cause
AKS clusters come with a mandated default `deny-all` global network policy.

Solution
Create network policies with higher priority .

Volumes using a Disk Encryption Set won't start mounting

Thanks to @Matthieu Simon for a viable solution to this.

Root cause
Assigning the *Contributor* role to the AKS identity for the scope of the Disk Encryption Set would solve this.
However, that built-in role is denied by policy.

Workaround
assign read permission on the DES, but deploy the volumes in your AKS RG to allow your user identity to create the encrypted volumes.

Storage class example:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: managed-csi-with-des
parameters:
  skuName: StandardSSD_LRS
  diskEncryptionSetID: DISK_ENCRYPTION_SET_ID
  resourceGroup: AKS_GENERATED_RG
provisioner: disk.csi.azure.com
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
```

Unable to fetch key vault secret using secret store CSI driver with User Managed Identity

"Error: MountVolume.SetUp failed for volume "secrets-store" : rpc error: code = Unknown desc = failed to mount secrets store objects for pod, err: rpc error: code = Unknown desc = failed to mount objects, error: failed to create auth config, error: failed to get credentials, nodePublishSecretRef secret is not set"

Root Cause: When you use user managed identity on secret store CSI driver to fetch the secrets from key vault, it is mandatory to assign the UMI to VMSS. Failing that will cause issue in connecting with key vault. Also check the access on key vault. Sometimes even after assigning the UMI to VMSS cause mount issues, this is due to the fact keys we used under parameters on secret class provider object. you must pass userAssignedIdentityID not clientID **(This clientID can be used only when you use secret store CSI driver with workload identity)**

```
  parameters:
    usePodIdentity: "false"
    useVMManagedIdentity: "true" # Set to true for using managed identity
    userAssignedIdentityID: <CLIENT-ID># Set the clientID of the user-assigned managed identity to use
```

Internal error occurred: failed calling webhook

"validate.nginx.ingress.kubernetes.io ": failed to call webhook: Post "https://ingress-nginx-controller-admission.ingress-nginx.svc:443/networking/v1/ingresses?timeout=10s ": tls: failed to verify certificate: x509: certificate signed by unknown authority

Root Cause

Validating Webhook Configuration and nginx admission should have same ca. If the ca used in secret is missing in validating webhook you encounter this error.

Solution:

kubectl get ValidatingWebhookConfiguration ingress-nginx-admission -o jsonpath='{.webhooks[0].clientConfig.caBundle}'

kubectl -n ingress-nginx get secret ingress-nginx-admission -o jsonpath='{.data.ca}'

Check both produces same certificates, if not patch the ValidatingWebhookConfiguration ingress-nginx-admission with correct cert same as secret.

Load balancer IP is not assigned to ingress object - "ingress does not contain a valid IngressClass"

If ingress controller created with Load Balancer(LB) IP then all the ingress objects will inherit the LB IP. In some cases the IP address of ingress object is empty. Check the logs of Nginx controller pod and if you could see Error "ingress does not contain a valid IngressClass"

Root Cause This happens when the --controller-class in ingress controller config is not set to default or running multiple ingress controller could cause this error.

Solution Add the below annotation in your ingress object

```
metadata:
  name: <INGRESS-NAME>
  annotations:
    kubernetes.io/ingress.class: "nginx"
```

Upstream timed out (110: Operation timed out) while connecting to upstream

Root Cause There are multiple cause for this error

1. Application service endpoint is not up and running
2. Application service port mismatch with application port listing on
3. Application response is slow
4. IP or port blocked by Network policy

Solution For case 1 and 2 application team has to fix it.

For case 3 add below annotation according to application response. nginx.ingress.kubernetes.io/proxy-connect-timeout

For case 4 add the network policy to open the required IP or port. Refer Network policy creation example and calico policy creation example

Traffic is not reaching my application

Check:

1. the loadbalancer service is setup correctly. For example: `externalTrafficPolicy` is set to `local`
2. your ingress is receiving traffic from the Load Balancer
3. your ingress is set to forward the traffic to the correct service, at the correct port and with the correct protocol
4. your service is set to forward the traffic to the correct pods, at the correct port and with the correct protocol
5. your pods are exposing the correct container and ports
6. your pods are receiving the traffic
7. eventual sidecars that are intercepting the traffic are correctly processing the traffic (including eventual certificates)
8. eventual sidecars that are intercepting the traffic are not breaking the flow of traffic to the destination container

9. the correct container in each of the pods is receiving traffic
10. the correct container in each of the pods is listening to the correct port with the correct protocol
11. the correct container in each of the pods is correctly processing the traffic (including eventual certificates)

References:

- troubleshooting ingress and services traffic flows
- troubleshooting the nginx ingress controller