**KYVERNO BEST PRACTICES FOR KUBERNETES CLUSTER MANAGEMENT**

The following illustrations shows the best practices implementation for you k8s cluster:

**1.0 To Disallow Default Namespace**

Kubernetes Namespaces are an optional feature that provide a way to segment and isolate cluster resources across multiple applications and users. As a best practice in kubernetes note that workloads should be isolated with Namespaces. Namespaces should be a must and the default (empty) Namespace should not be prohibited.

This policy validates that Pods must specify a Namespace name other than the name of “default”.

For implementation/adjustments of the namespace policy described above, use the manifest file “[disallow\_default\_namespace.yaml](https://github.com/kyverno/policies/raw/main//best-practices/disallow_default_namespace/disallow_default_namespace.yaml)” inside the manifests folder

To deploy, use this command:

kubectl apply -f /pathtomanifestfolder/disallow\_default\_namespace.yaml

**2.0 Require “run as non root”**

Most Docker containers and the processes inside run with non-root user, because of better security. If the container process is running with root (uid 0) it will be the same root as on the host. In this case user may get access to host from the container, thus gaining the root privilege on the host. This is of course a security concern.

However there can be a case when you need to **run a container with root privilege** because of permission issues of the volumes on the host.

This cluster policy ensures the NonRoot` property is set to `true` in your cluster

Containers must be required to run as non-root users.

For implementation/adjustments of the above policy, use the manifest file “[require-run-as-nonroot.yaml](https://github.com/kyverno/policies/raw/main//pod-security/restricted/require-run-as-nonroot/require-run-as-nonroot.yaml)” inside the manifests folder

To deploy the policy or make changes to it, use the following command:

kubectl apply -f /pathtomanifestsfolder/run-as-nonroot.yaml

**3.0 Requiring Non Root Groups**

Note that Containers should be forbidden from running with a root primary or supplementary GID. This policy ensures that the following `runAsGroup`, `supplementalGroups`, and `fsGroup` fields are set to a number greater than zero (i.e., non root).

Implementations and adjustments of the above policy can be done by using the following file in the manifest folder “[require-non-root-groups.yaml](https://github.com/kyverno/policies/raw/main//pod-security/restricted/require-non-root-groups/require-non-root-groups.yaml)”

Use the following comand to carry out this effect

kubectl apply -f /pathtomanifestsfolder/require-non-groups.yaml

**4.0 Requiring Limits and Requests in your k8s cluster**

As application workloads share cluster resources, it is important to limit resources requested and consumed by each Pod. It is recommended to require resource requests and limits per Pod, especially for memory and CPU. If a Namespace level request or limit is specified, defaults will automatically be applied to each Pod based on the LimitRange configuration, Hence This policy validates that all containers have something specified for memory and CPU requests and memory limits.

To make effect this policy, use the [require\_pod\_requests\_limits.yaml](https://github.com/kyverno/policies/raw/main//best-practices/require_pod_requests_limits/require_pod_requests_limits.yaml) file inside the manifests folder by running the below command:

kubectl apply -f /pathtomaifestsfolder/require\_pod\_requests\_limits.yaml

**5.0 Denying Privilege Escalation**  
Privilege escalation, such as via set-user-ID or set-group-ID file mode, should not be allowed. This policy ensures that the `allowPrivilegeEscalation` fields are either undefined or set to `false`.

This can be implemented using the following file called [deny-privilege-escalation.yaml](https://github.com/kyverno/policies/raw/main//pod-security/restricted/deny-privilege-escalation/deny-privilege-escalation.yaml) file inside the manifests folder

The shell command to use to deploy this policy to the cluster is:

kubectl apply -f /pathtomanifests/policies/deny-privilege-escalation.yaml

**SAMPLE DEPLOYMENTS OBEYING THE ABOVE DEFINED POLICIES**

**1.0 To Disallow Default Namespace**

**Sample usage is illustrated here:**

/pathtomanifests/deploy/namespace\_sample.yaml

You can create a namespace using the command:

kubectl apply -f /pathtomanifests/deploy/disallow\_default\_namespace.yaml

You can switch to operate in the development namespace for instance.

kubectl config use-context dev

You can verify your current context by doing the following:

kubectl config current-context

dev

At this point, all requests we make to the Kubernetes cluster from the command line are scoped to the development namespace.

**2.0 Require “run as non root”**

**Check for the sample usage here for your deployments**

/pathtomanifests/deploy/run-as-nonroot.yaml

**3.0 Requiring Non Root Groups**

**Check under the folder for the sample manifest:**

/pathtomanifests/deploy/security-context.yaml

It is a configuration file for a Pod that has a securityContext and an emptyDir volume.

In the configuration file, the runAsUser field specifies that for any Containers in the Pod, all processes run with user ID 1000. The runAsGroup field specifies the primary group ID of 3000 for all processes within any containers of the Pod. If this field is omitted, the primary group ID of the containers will be root(0). Any files created will also be owned by user 1000 and group 3000 when runAsGroup is specified. Since fsGroup field is specified, all processes of the container are also part of the supplementary group ID 2000. The owner for volume /data/demo and any files created in that volume will be Group ID 2000.

**4.0 Requiring Limits and Requests in your k8s cluster**

**Check under the folder for the sample manifest:**

/pathtomanifests/deploy/require\_pod\_requests\_limits.yaml

**5.0 Denying Privilege Escalation**

**Check the file under the directory for sample usage:**

/pathtomanifests/deploy/deny-privilege-escalation.yaml