Kubernetes Installer Requirements derived from RA-2

This page will go through the requirements for a Kubernetes installer based on RA-2.

It differs from the existing comparison of installers, as Kubeadm has been removed. Some of the remaining installers still utilize Kubeadm as a bootstrapping tool, but it will not be considered as an installer by itself.

Component Level Architecture Requirements

The chosen runtime must be conformant with the Kubernetes Container Runtime Interface (CRI) and the Open Container Initiative (OCI) runtime spec.

Reference: TBD (below content is based on the High Level Architecture)

Feature	Airship	Kind	KubeOne	Kubespray	OpenShift Origin (OKD)
Container Runtime	Docker	Docker	Docker	Docker	Docker

A list of some of the avaialble OCI runtimes and CRIs can be seen below.

OCI Runtimes:

- RunC (Native)
- Crun (Native)
- Kata-containers (Virtualized)
- gVisor (Sandboxed)
- Nable-containers (Sandboxed)

CRIs:

- Containerd (uses runC by default)
- Cri-o (uses runC by default)

Docker should be mentioned here as well, as it uses containerd and runC as part of the "container engine".

The above comparison table is not complete, but covers a minimum that fulfills the requirement for conformance with Kubernetes CRI and OCI runtime spec. Any installer using Kubeadm has the potential to select between Docker, cri-o and containerd as the container runtime, but this may not be fully supported through the installer.

Align with the Kubernetes version support policy

Reference: ra2.k8s.005

The version support policy states that the most recent 3 minor releases (n-2) are supported. The reference implementation, and therefore the Kubernetes installer, must align with this policy.

Feature	Airship	Kind	KubeOne	Kubespray	OpenShift Origin (OKD)
Follow support policy	Yes	Yes	Yes	Yes	Yes

The specific version of Kubernetes (or more specifically, the tools that are installed) depends to the version of the installer used. Here it is assumed that the most recent installer release is used.

A set of Kubelet features must be enabled

The features are:

- CPU Manager (Beta) - ra2.k8s.006
- Topology Manager (Beta) - ra2.k8s.006
- Device Plugin (Beta) - ra2.k8s.007
- IPv6 Dual Stack Feature - ra2.k8s.010

As of Kubernetes v1.18, all features are in "Beta" state and enabled by default. Prior to this, Topology Manager has been in "Alpha" state, and had to be enabled through a Kubelet feature-gates.

For Topology Manager to be useful, a policy must be specified using the below argument passed to Kubelet, as the default value of "none" is similar to not having the Topology Manager enabled.

```
--topology-manager-policy=[none (default) | best-effort | restricted | single-numa-node]
```

If the installer uses Kubespray, this can be configured through the `kubelet_custom_flags` variable.

Feature	Airship	Kind	KubeOne	Kubespray	OpenShift Origin (OKD)
Use Topology Manager Policy	Configurable	Likely configurable	Not configurable	Configurable	Unsure?

As seen in the above table, the support for changing the policy varies between installers. None of the installers change the policy by default, but there are varying levels of support available.

Where configurable, it is usually through specifying flags to be passed to kubelet similar to the argument shown above. For the other cases there are limited control over both feature-gates and flags, which in some cases prevent setting the policy.

While Topology Manager (alongside the other two features) provides a level of NUMA awareness to the cluster, it should be noted that this doesn't include Hugepages.

Also, the scheduler is not topology aware, so any POD that fails to schedule due to the topology manager will remain in terminated state. This can be solved by utilizing a ReplicaSet or by running the POD as part of a deployment.

Container Network Services

The CNI Plugin chosen by the installer should be compliant with the default Kubernetes networking assumptions.