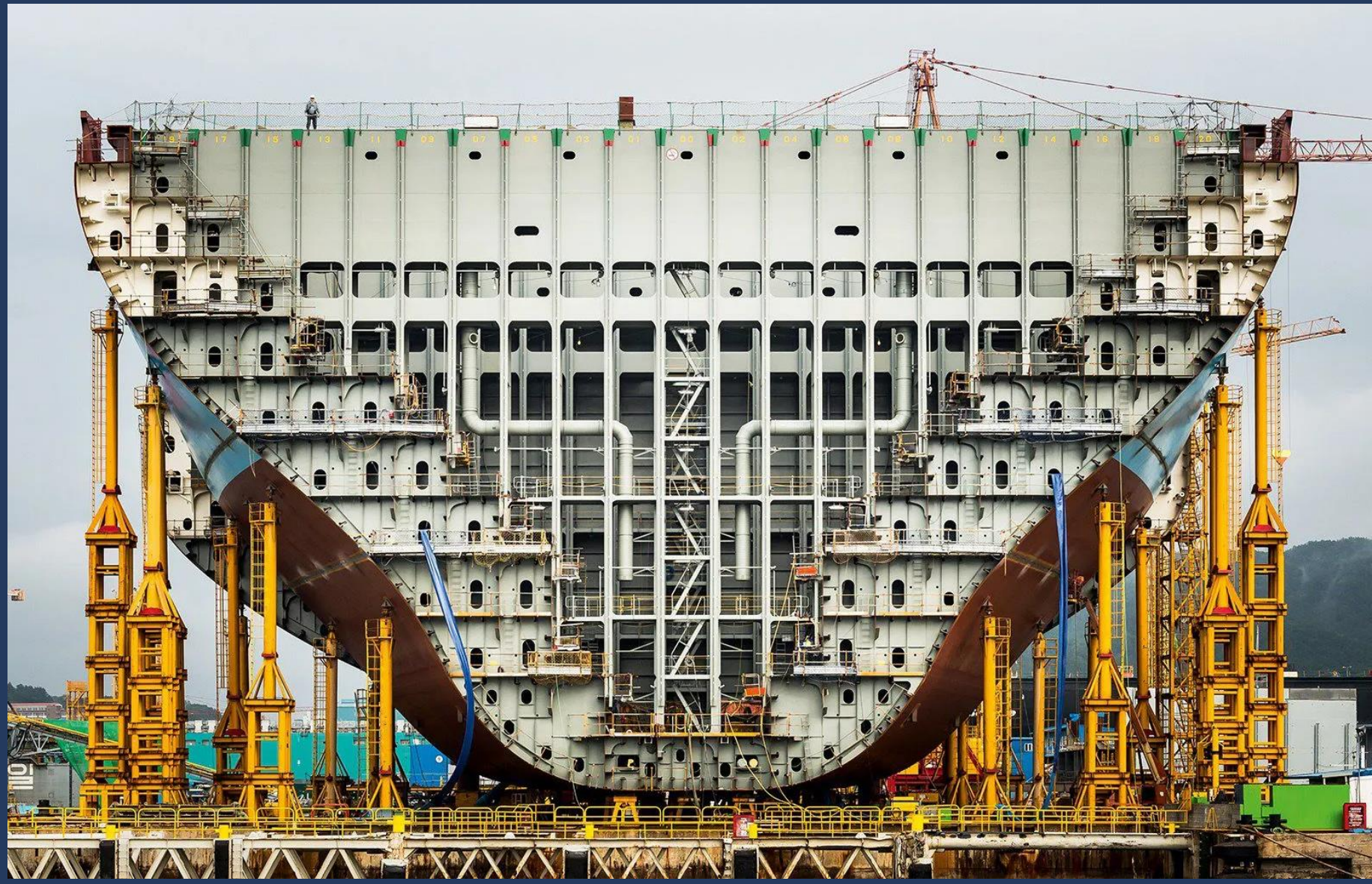
Microsoft

Kubernetes Technical Briefing

Kubernetes Internals for Developers



Agenda

- Kubernetes Objects
- Kubernetes API Reference
- API Object Versions

Creating Kubernetes Definitions

- When you're creating a YAML manifest, how do you know:
  - What properties are available?
  - What are is the type of each property? String, List, Map (Dictionary), Integer?
  - Where to place which property in the file?
  - How much to indent each property?
  - Which properties are required, and which are optional?
  - What are the default values of optional properties?
- To better understand the structure of Kubernetes YAML files, it's helpful to examine some of its class definitions.

Sample Kubernetes Class Definition

A Pod object (as represented by a C# class) is an example of a typical Kubernetes class definition:

```
public class Pod
{
    public string ApiVersion { get; set; }
    public string Kind { get; set; }
    public ObjectMeta Metadata { get; set; }
    public PodSpec Spec { get; set; }
    public PodStatus Status { get; set; }
}
```

```
public class ObjectMeta
{
    public string Name { get; set; }
    public Dictionary<string, string> Labels { get; set; }
    public Dictionary<string, string> Annotations { get; set; }
    public string NamespaceProperty { get; set; }
    public DateTime? CreationTimestamp { get; set; }
```

```
public class PodSpec
{
    public List<Container> Containers { get; set; }
    public List<Volume> Volumes { get; set; }
    public string ServiceAccount { get; set; }
    public string RestartPolicy { get; set; }
    public string NodeName { get; set; }
```

Sample Kubernetes Class Definition Continued

Nested classes in Pod:

```
public class PodSpec
{
    public List<Container> Containers { get; set; }
    public List<Volume> Volumes { get; set; }
    public string ServiceAccount { get; set; }
    public string RestartPolicy { get; set; }
    public string NodeName { get; set; }
```

```
public class Container
{
    public string Name { get; set; }
    public string Image { get; set; }
    public List<VolumeMount> VolumeMounts { get; set; }
    public List<EnvVar> Env { get; set; }
    public List<string> Command { get; set; }
    public List<string> Args { get; set; }
    public Probe ReadinessProbe { get; set; }
    public List<ContainerPort> Ports { get; set; }
    public Probe LivenessProbe { get; set; }
    public Lifecycle Lifecycle { get; set; }
    public string ImagePullPolicy { get; set; }
```

```
public class Volume
{
    public string Name { get; set; }
    public EmptyDirVolumeSource EmptyDir { get; set; }
    public SecretVolumeSource Secret { get; set; }
    public AzureFileVolumeSource AzureFile { get; set; }
    public AzureDiskVolumeSource AzureDisk { get; set; }
    public ConfigMapVolumeSource ConfigMap { get; set; }
```

YAML Definitions map to Object Instances

YAML definition sent to Kubernetes will result in the creation of a Pod object, with nested objects populated

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test-ebs
spec:
  containers:
  - image: k8s.gcr.io/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /test-ebs
      name: test-volume
  volumes:
  - name: test-volume
```

```
public class Pod
{
    public string ApiVersion { get; set; }
    public string Kind { get; set; }
    public ObjectMeta Metadata { get; set; }
    public PodSpec Spec { get; set; }
```

```
public class PodSpec
{
    public List<Container> Containers { get; set; }
    public List<Volume> Volumes { get; set; }
```

```
public class Container
{
    public string Name { get; set; }
    public string Image { get; set; }
    public List<VolumeMount> VolumeMounts { get; set; }
```

```
public class VolumeMount
{
    public string Name { get; set; }
    public string MountPath { get; set; }
    public bool? ReadOnlyProperty { get; set; }
```

Applying Object Definitions

- ***Kubectl*** converts YAML to JSON
- It makes an HTTP POST request to the Kubernetes API Server.
- The Kubernetes API Server:
  - Accepts JSON definitions in the body of a Request
  - Deserializes them into object instances and populates empty properties with defaults.
  - Serializes and saves them (as JSON) in its internal database.

Kubectl client

Kubernetes API Server

Kubernetes API Reference

- Kubernetes API Reference lists all the available class definitions
- API are arranged in groups
- Class definitions contain property types, descriptions and defaults.
- Objects can have multiple versions

API Groups

The API Groups and their versions are summarized in the following table.

Group	Version
admissionregistration.k8s.io	v1, v1beta1
apiextensions.k8s.io	v1, v1beta1
apiregistration.k8s.io	v1, v1beta1
apps	v1

Explore the Kubernetes API Reference:

https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.19

Kubernetes Supports Multiple Object Versions

Kubernetes supports deprecated object versions for several upgrades

HorizontalPodAutoscalerSpec v1 autoscaling

Appears In:
- HorizontalPodAutoscaler [autoscaling/v1]

Field	Description
`maxReplicas` *integer*	upper limit for the number of pods that can be set by the autoscaler; cannot be smaller than MinReplicas.
`minReplicas` *integer*	minReplicas is the lower limit for the number of replicas to which the autoscaler can scale down. It defaults to 1 pod. minReplicas is allowed to be 0 if the alpha feature gate HPAScaleToZero is enabled and at least one Object or External metric is configured. Scaling is active as long as at least one metric value is available.
`scaleTargetRef` *CrossVersionObjectReference*	reference to scaled resource; horizontal pod autoscaler will learn the current resource consumption and will set the desired number of pods by using its Scale subresource.
`targetCPUUtilizationPercentage` *integer*	target average CPU utilization (represented as a percentage of requested CPU) over all the pods; if not specified the default autoscaling policy will be used.

HorizontalPodAutoscalerSpec v2beta2 autoscaling

Appears In:
- HorizontalPodAutoscaler [autoscaling/v2beta2]

Field	Description
`behavior` *HorizontalPodAutoscalerBehavior*	behavior configures the scaling behavior of the target in both Up and Down directions (scaleUp and scaleDown fields respectively). If not set, the default HPAScalingRules for scale up and scale down are used.
`maxReplicas` *integer*	maxReplicas is the upper limit for the number of replicas to which the autoscaler can scale up. It cannot be less that minReplicas.
`metrics` *MetricSpec array*	metrics contains the specifications for which to use to calculate the desired replica count (the maximum replica count across all metrics will be used). The desired replica count is calculated multiplying the ratio between the target value and the current value by the current number of pods. Ergo, metrics used must decrease as the pod count is increased, and vice-versa. See the individual metric source types for more information about how each type of metric must respond. If not set, the default metric will be set to 80% average CPU utilization.
`minReplicas` *integer*	minReplicas is the lower limit for the number of replicas to which the autoscaler can scale down. It defaults to 1 pod. minReplicas is allowed to be 0 if the alpha feature gate HPAScaleToZero is enabled and at least one Object or External metric is configured. Scaling is active as long as at least one metric value is available.
`scaleTargetRef` *CrossVersionObjectReference*	scaleTargetRef points to the target resource to scale, and is used to the pods for which metrics should be collected, as well as to actually change the replica count.

Kubernetes Definition Versions

Make sure the version in the YAML matches the definition you want to create

```yaml
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: complex-web-hpa-1
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: complex-web-dep
  minReplicas: 2
  maxReplicas: 12
  metrics:
  # Averages take the average of the given
  # metric across ALL Pods matching the target
  - type: Resource
    resource: ···

  behavior:
    scaleDown:
      # The stabilization window is used to restr
      # replicas when the metrics used for scalin
      stabilizationWindowSeconds: 30
```

HorizontalPodAutoscalerSpec v2beta2 autoscaling

Appears In:

- HorizontalPodAutoscaler [autoscaling/v2beta2]

Field	Description
behavior *HorizontalPodAutoscalerBehavior*	behavior configures the scaling behavior of the target in both Up and Down directions (scaleUp and scaleDown fields respectively). If not set, the default HPAScalingRules for scale up and scale down are used.
maxReplicas *integer*	maxReplicas is the upper limit for the number of replicas to which the autoscaler can scale up. It cannot be less that minReplicas.
metrics *MetricSpec array*	metrics contains the specifications for which to use to calculate the desired replica count (the maximum replica count across all metrics will be used). The desired replica count is calculated multiplying the ratio between the target value and the current value by the current number of pods. Ergo, metrics used must decrease as the pod count is increased, and vice-versa. See the individual metric source types for more information about how each type of metric must respond. If not set, the default metric will be set to 80% average CPU utilization.
minReplicas *integer*	minReplicas is the lower limit for the number of replicas to which the autoscaler can scale down. It defaults to 1 pod. minReplicas is allowed to be 0 if the alpha feature gate HPAScaleToZero is enabled and at least one Object or External metric is configured. Scaling is active as long as at least one metric value is available.
scaleTargetRef *CrossVersionObjectReference*	scaleTargetRef points to the target resource to scale, and is used to the pods for which metrics should be collected, as well as to actually change the replica count.

Microsoft

Thank you